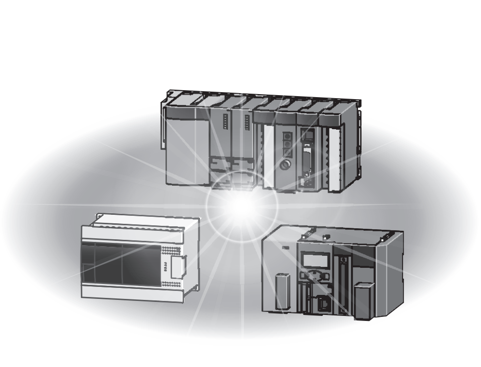
MITSUBISHI
ELECTRIC

Programmable Controller

MELSEC Q series MELSEC L series

MELSEC-Q/L Structured Programming Manual (Common Instructions)



SAFETY PRECAUTIONS

(Read these precautions before using this product.)

Before using MELSEC-Q or -L series programmable controllers, please read this manual and the related manuals introduced in this manual, and pay full attention to safety to handle the product correctly.

Please store this manual in a safe place and make it accessible when required. Always forward a copy of the manual to the end user.

CONDITIONS OF USE FOR THE PRODUCT

(1) Mitsubishi Electric 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.

INTRODUCTION

Thank you for purchasing the Mitsubishi Electric MELSEC-Q or -L series programmable controllers.

Before using this product, please read this manual and the relevant manuals carefully and develop familiarity with the programming specifications to handle the product correctly.

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

MEMO

CONTENTS

CHAPTER 6 BASIC INSTRUCTIONS 139

CHAPTER 8 DATA LINK INSTRUCTIONS 799

CHAPTER 9 MULTIPLE CPU DEDICATED INSTRUCTIONS 828

CHAPTER 10 MULTIPLE CPU HIGH SPEED TRANSMISSION DEDICATED INSTRUCTIONS 846

CHAPTER 11 REDUNDANT SYSTEM INSTRUCTIONS (FOR REDUNDANT CPU) 863

APPENDIX 867

INDEX 869

INSTRUCTION INDEX 871

MANUALS

Relevant manuals

The manuals related to this product are listed below. Order each manual as needed, referring to the following lists.

■Structured programming

Manual name <Manual number>	Description
MELSEC-Q/L/F Structured Programming Manual (Fundamentals) <SH-080782ENG>	Methods and languages for structured programming
MELSEC-Q/L Structured Programming Manual (Application Functions) <SH-080784ENG>	Specifications and functions of application functions that can be used in structured programs
MELSEC-Q/L Structured Programming Manual (Special Instructions) <SH-080785ENG>	Specifications and functions of special instructions, such as module dedicated instructions, PID control instructions, and built-in I/O function instructions, that can be used in structured programs

■Operation of GX Works2

Manual name <Manual number>	Description
GX Works2 Version 1 Operating Manual (Common) <SH-080779ENG>	System configuration, parameter settings, and online operations of GX Works2, which are common to Simple projects and Structured projects
GX Works2 Version 1 Operating Manual (Structured Project) <SH-080781ENG>	Operations, such as programming and monitoring in Structured projects, of GX Works2
GX Works2 Beginner's Manual (Structured Project) <SH-080788ENG>	Basic operations, such as programming, editing, and monitoring in Structured projects, of GX Works2. This manual is intended for first-time users of GX Works2.

Terms

This manual uses the generic terms and abbreviations listed in the following table to discuss the software packages and programmable controller CPUs. Corresponding module models are also listed if needed.

Term	Description
Application function	A generic term for the functions, such as functions and function blocks, defined in IEC 61131-3. (It is executed by combinations of multiple common instructions in a programmable controller.)
Basic model QCPU	A generic term for the Q00JCPU, Q00CPU, and Q01CPU
Built-in Ethernet port LCPU	A generic term for the L02CPU, L02CPU-P, L06CPU, L06CPU-P, L26CPU, L26CPU-P, L26CPU-BT, and L26CPU-PBT
Built-in Ethernet port QCPU	A generic term for the Q03UDVCPU, Q03UDECPU, Q04UDVCPU, Q04UDPVCPU, Q04UDEHCPU, Q06UDVCPU, Q06UDPVCPU, Q06UDEHCPU, Q10UDEHCPU, Q13UDVCPU, Q13UDPVCPU, Q13UDEHCPU, Q20UDEHCPU, Q26UDVCPU, Q26UDPVCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU
CC-Link IE	A generic term for CC-Link IE Controller Network and CC-Link IE Field Network
Common instruction	A generic term for the sequence instructions, basic instructions, application instructions, data link instructions, multiple CPU dedicated instructions, multiple CPU high-speed transmission dedicated instructions, and redundant system instructions
CPU module	A generic term for the QCPU (Q mode) and LCPU
GX Works2	Product name for the MELSEC programmable controller software package
High Performance model QCPU	A generic term for the Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU, and Q25HCPU
High-speed Universal model QCPU	A generic term for the Q03UDVCPU, Q04UDVCPU, Q06UDVCPU, Q13UDVCPU, and Q26UDVCPU
IEC61131-3	The abbreviation for the IEC 61131-3 international standard
LCPU	A generic term for the L02SCPU, L02SCPU-P, L02CPU, L02CPU-P, L06CPU, L06CPU-P, L26CPU, L26CPU-P, L26CPU-BT, and L26CPU-PBT
MELSECNET/H	The abbreviation for the MELSECNET/H network system
Personal computer	A generic term for personal computer on which Windows® operates
Process CPU	A generic term for the Q02PHCPU, Q06PHCPU, Q12PHCPU, and Q25PHCPU
QCPU (Q mode)	A generic term for the Basic model QCPU, High Performance model QCPU, Process CPU, Redundant CPU, and Universal model QCPU
QnCPU	A generic term for the Q00JCPU, Q00CPU, Q01CPU, and Q02CPU
QnHCPU	A generic term for the Q02HCPU, Q06HCPU, Q12HCPU, and Q25HCPU
QnPHCPU	A generic term for the Q02PHCPU, Q06PHCPU, Q12PHCPU, and Q25PHCPU
QnPRHCPU	A generic term for the Q12PRHCPU and Q25PRHCPU
QnU(D)(H)CPU	A generic term for the Q02UCPU, Q03UDCPU, Q04UDHCPU, Q06UDHCPU, Q10UDHCPU, Q13UDHCPU, Q20UDHCPU, and Q26UDHCPU
QnUCPU	A generic term for the Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, Q03UDCPU, Q03UDVCPU, Q03UDECPU, Q04UDHCPU, Q04UDVCPU, Q04UDPVCPU, Q04UDEHCPU, Q06UDHCPU, Q06UDVCPU, Q06UDPVCPU, Q06UDEHCPU, Q10UDHCPU, Q10UDEHCPU, Q13UDHCPU, Q13UDVCPU, Q13UDPVCPU, Q13UDEHCPU, Q20UDHCPU, Q20UDEHCPU, Q26UDHCPU, Q26UDVCPU, Q26UDPVCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU
QnUD(H)CPU	A generic term for the Q03UDCPU, Q04UDHCPU, Q06UDHCPU, Q10UDHCPU, Q13UDHCPU, Q20UDHCPU, and Q26UDHCPU
QnUDE(H)CPU	A generic term for the Q03UDECPU, Q04UDEHCPU, Q06UDEHCPU, Q10UDEHCPU, Q13UDEHCPU, Q20UDEHCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU
QnUDPVCPU	A generic term for the Q04UDPVCPU, Q06UDPVCPU, Q13UDPVCPU, and Q26UDPVCPU
QnUDVCPU	A generic term for the Q03UDVCPU, Q04UDVCPU, Q06UDVCPU, Q13UDVCPU, and Q26UDVCPU
Redundant CPU	A generic term for the Q12PRHCPU and Q25PRHCPU
Special instruction	A generic term for module dedicated instructions, PID control instructions, socket communication function instructions, built-in I/O function instructions, and data logging function instructions
Universal model Process CPU	A generic term for the Q04UDPVCPU, Q06UDPVCPU, Q13UDPVCPU, and Q26UDPVCPU
Universal model QCPU	A generic term for the Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, Q03UDCPU, Q03UDVCPU, Q03UDECPU, Q04UDHCPU, Q04UDVCPU, Q04UDPVCPU, Q04UDEHCPU, Q06UDHCPU, Q06UDVCPU, Q06UDPVCPU, Q06UDEHCPU, Q10UDHCPU, Q10UDEHCPU, Q13UDHCPU, Q13UDVCPU, Q13UDPVCPU, Q13UDEHCPU, Q20UDHCPU, Q20UDEHCPU, Q26UDHCPU, Q26UDVCPU, Q26UDPVCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU

1 OVERVIEW

1.1 Purpose of This Manual

This manual explains the common instructions used for creating structured programs. Manuals for reference are listed in the following table according to their purpose.

For information such as the contents and number of each manual, refer to the list of 'Related manuals'.

☞ Page 12 Relevant manuals

Operation of GX Works2

Purpose		Summary	Detail
Installation	Learning the operating environment and installation method	—	🕮 GX Works2 Installation Instructions
	Learning a USB driver installation method	—	🕮 GX Works2 Version 1 Operating Manual (Common)
Operation of GX Works2	Learning all functions of GX Works2	🕮 GX Works2 Version 1 Operating Manual (Common)	—
	Learning the project types and available languages in GX Works2		—
	Learning the basic operations and operating procedures when creating a simple project for the first time	—	🕮 GX Works2 Beginner's Manual (Simple Project)
	Learning the basic operations and operating procedures when creating a structured project for the first time	—	🕮 GX Works2 Beginner's Manual (Structured Project)
	Learning the operations of available functions regardless of project type.	—	🕮 GX Works2 Version 1 Operating Manual (Common)
	Learning the functions and operation methods for programming	🕮 GX Works2 Version 1 Operating Manual (Common)	🕮 GX Works2 Version 1 Operating Manual (Simple Project) 🕮 GX Works2 Version 1 Operating Manual (Structured Project)
	Learning data setting methods for intelligent function module	—	🕮 GX Works2 Version 1 Operating Manual (Intelligent Function Module)

Operations in each programming language

For details of instructions used in each programming language, refer to the following.

☞ Page 15 Details of instructions in each programming language

Purpose		Summary	Detail
Simple Project	Ladder	🕮 GX Works2 Beginner's Manual (Simple Project)	🕮 GX Works2 Version 1 Operating Manual (Simple Project)
	SFC	🕮 GX Works2 Beginner's Manual (Simple Project)[*1]	
	ST	🕮 GX Works2 Beginner's Manual (Structured Project)	🕮 GX Works2 Version 1 Operating Manual (Structured Project)
Structured Project	Ladder	🕮 GX Works2 Beginner's Manual (Simple Project)	🕮 GX Works2 Version 1 Operating Manual (Simple Project)
	SFC	🕮 GX Works2 Beginner's Manual (Simple Project)[*1]	
	Structured ladder/FBD	🕮 GX Works2 Beginner's Manual (Structured Project)	🕮 GX Works2 Version 1 Operating Manual (Structured Project)
	ST		

[*1] MELSAP3 and FX series SFC only

Details of instructions in each programming language

Purpose		Summary	Detail
All languages	Learning details of programmable controller CPU error codes, special relays, and special registers	—	📖 Use's Manual (Hardware Design, Maintenance and Inspection) for the CPU module used
Using ladder language	Learning the types and details of common instructions	—	📖 MELSEC-Q/L Programming Manual (Common Instruction)
	Learning the types and details of instructions for intelligent function modules	—	📖 Manual for the intelligent function module used
	Learning the types and details of instructions for network modules	—	📖 Manual for the network module used
	Learning the types and details of instructions for the PID control function	—	📖 MELSEC-Q/L/QnA Programming Manual (PID Control Instructions)
	Learning the types and details of the process control instructions	—	📖 MELSEC-Q Programming/Structured Programming Manual (Process Control Instructions)
Using SFC language	Learning details of specifications, functions, and instructions of SFC (MELSAP3)	—	📖 MELSEC-Q/L/QnA Programming Manual (SFC)
Using structured ladder/FBD/ST language	Learning the fundamentals for creating a structured program	—	📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)
	Learning the types and details of common instructions	—	📖 MELSEC-Q/L Structured Programming Manual (Common Instructions)
	Learning the types and details of instructions for intelligent function modules	📖 MELSEC-Q/L Structured Programming Manual (Special Instructions)	📖 Manual for the intelligent function module used
	Learning the types and details of instructions for network modules		📖 Manual for the network module used
	Learning the types and details of instructions for the PID control function		📖 MELSEC-Q/L/QnA Programming Manual (PID Control Instructions)
	Learning the types and details of application functions	—	📖 MELSEC-Q/L Structured Programming Manual (Application Functions)
	Learning the types and details of the process control instructions	—	📖 MELSEC-Q Programming/Structured Programming Manual (Process Control Instructions)

2 INSTRUCTION TABLES

2.1 Types of Instructions

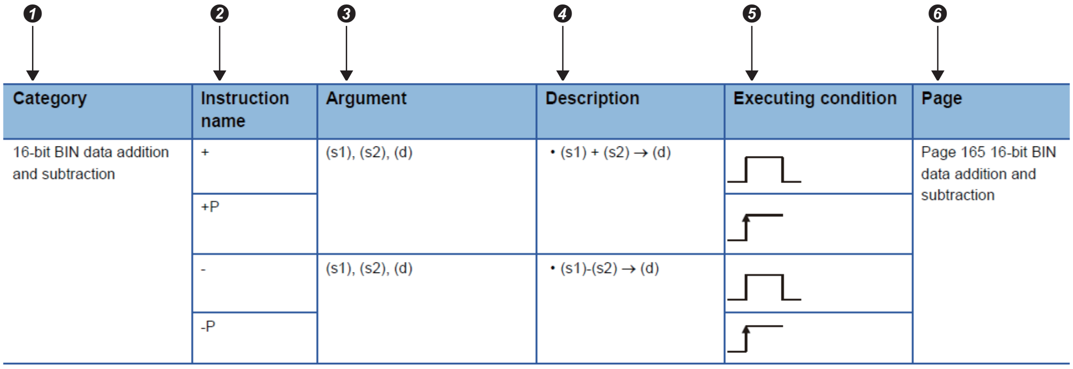
Instructions for CPU modules are classified into sequence instructions, basic instructions, application instructions, data link instruction, multiple CPU dedicated instructions, multiple CPU high speed transmission dedicated instructions, and redundant system instructions. These types of instructions are listed in the following Table.

Type of instruction		Description	Reference
Sequence instruction	Contact instruction	Operation start, series connection, parallel connection	Page 83 SEQUENCE INSTRUCTIONS
	Bond instruction	Ladder block connection, operation result pulse, operation result store/read	
	Output instruction	Bit device output, pulse output, output inversion	
	Shift instruction	Bit device shift	
	Master control instruction	Master control	
	End instruction	Program conclusion	
	Other instructions	Instructions not classified into the above types, such as program stop and no operation	
Basic instruction	Comparison operation instruction	Comparison such as =, >, and <	Page 139 BASIC INSTRUCTIONS
	Arithmetic operation instruction	Addition, subtraction, multiplication, and division of BIN, BCD, floating-point data or string	
	Data conversion instruction	Conversion from BIN to BCD data and vice versa, conversion from BIN data to floating-point data and vice versa	
	Data transfer instruction	Specified data transfer	
	Program branch instruction	Program jump	
	Program execution control instruction	Enable/disable interrupt programs	
	I/O refresh instruction	Partial refresh execution	
	Other convenient instructions	Instructions such as up/down counter, teaching timer, special function timer, and rotary table shortest direction control	
Application instruction	Logical operation instruction	Logical operations such as logical OR and logical AND	Page 318 APPLICATION INSTRUCTIONS
	Rotation instruction	Specified data rotation	
	Shift instruction	Specified data shift	
	Bit processing instruction	Bit set/reset, bit test, bit device batch reset	
	Data processing instruction	Data processes such as 16-bit data search, decode, and encode	
	Structured instruction	Repeated operation, subroutine program call, selection of refresh, fixed index setting	
	Data table operation instruction	Data read/write from/to data tables	
	Buffer memory access instruction	Data read/write from/to intelligent function module	
	Display instruction	ASCII code print, reset the annunciator	
	Debug/error diagnostics instruction	Special format error check, changing check format of the CHK instruction	
	String processing instruction	Conversion from BIN/BCD data to ASCII data and vice versa, conversion from BIN data to character string data and vice versa, conversion from floating-point data to character string data and vice versa, character string process	
	Special function instruction	Trigonometric functions, conversion from degree to radian and vice versa, exponential operation, natural logarithm, common logarithm, square root	
	Data control instruction	Upper/lower limit control, dead band control, zone control, scaling	
	File register switching instruction	File register block numbers switch, file register/comment file specification	
	Clock instruction	Clock data (year, month, day, hour, minute, second, and day of week) read/write, clock data (hour, minute, and second) addition/subtraction, time data conversion from hour/minute/second format to seconds and vice versa, date (year, month, and day) comparison, clock data (hour, minute, and second) comparison	
	Extended Clock Instruction	Clock data (year, month, day, hour, minute, second, day of week, and millisecond) read/write, clock data (hour, minute, second, and millisecond) addition/subtraction	
	Program control instruction	Switch instruction of program executing conditions	
	Other instructions	Instructions not classified into the above types, such as WDT reset and timing clock	

Type of instruction		Description	Reference
Data Link instruction	Network link refresh instruction	Specified network module refresh	Page 799 DATA LINK INSTRUCTIONS
	Routing information read/register instruction	Routing data read/register	
	Refresh device write/read instruction	Refresh device write/read	
Multiple CPU dedicated instruction	Multiple CPU dedicated instruction	Data write to host CPU shared memory, data read from other CPU shared memory	Page 828 MULTIPLE CPU DEDICATED INSTRUCTIONS
Multiple CPU high speed transmission dedicated instruction	Multiple CPU device data write/read instruction	Device data write to other CPU, device data read from other CPU	Page 846 MULTIPLE CPU HIGH SPEED TRANSMISSION DEDICATED INSTRUCTIONS
Redundant system instruction	Redundant CPU dedicated instruction	System switching	Page 863 REDUNDANT SYSTEM INSTRUCTIONS (FOR REDUNDANT CPU)

2.2 How to Read Instruction Tables

The instruction tables found from Page 19 Sequence Instructions to Page 60 Redundant System Instruction (For Redundant CPU) have been made according to the following format:



Category	Instruction name	Argument	Description	Executing condition	Page
16-bit BIN data addition and subtraction	+	(s1), (s2), (d)	• (s1) + (s2) → (d)	⎍	Page 165 16-bit BIN data addition and subtraction
	+P			↑	
	-	(s1), (s2), (d)	• (s1)-(s2) → (d)	⎍	
	-P			↑	

Description

❶ Classifies instructions by application.

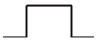
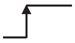
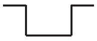
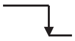
❷ Indicates the instructions used in a programs.

❸ Indicates the arguments of the instruction.

Symbol	Name	Description
(s), (s1)	Source	Stores data before operation.
(d), (d1)	Destination	Indicates the destination of data after operation.
n, n1	—	Specifies the number of devices and the number of transfers.
p	—	Specifies the pointer number.

❹ Indicates the processing details of each instruction.

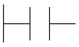
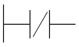
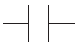
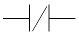
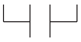
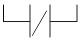
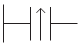
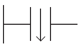
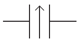
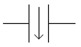
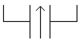
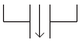
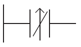
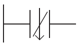
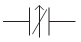
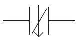
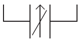
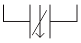
❺ Details of executing conditions of each instruction are as follows:

Symbol	Executing condition
No symbol	Indicates a non-conditional executing instruction that is always executed regardless of the ON/OFF status of the precondition of the instruction. When the precondition is OFF, the instruction performs 'OFF' processing.
	Indicates an 'executed while ON' type instruction that is executed only while the precondition is ON. When the precondition is OFF, the instruction is not executed and does not perform processing.
	Indicates an 'executed once at ON' type instruction that is executed only at the rising edge (OFF to ON) of the precondition of the instruction.
	When the precondition is OFF, the instruction is not executed and does not perform processing. When the precondition is ON, the instruction is not executed and does not perform processing.
	Indicates an 'executed once at OFF' type instruction that is executed only at the falling edge (ON to OFF) of the precondition of the instruction. The instruction is not executed afterwards even when the condition is OFF and thus does not perform processing.

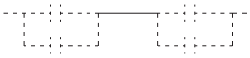
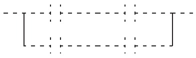
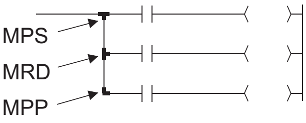
❻ Indicates the pages on which the instructions are explained.

2.3 Sequence Instructions

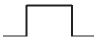
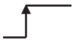
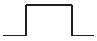
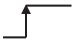
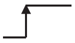
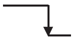
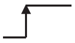
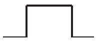
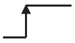
Contact instructions

Category	Instruction name	Symbol used in structured ladder/ FBD	Description	Executing condition	Page
Contact	LD		• Starts a logical operation. (Starts a logical operation with normally open contact)		Page 83 Operation start, series connection, parallel connection
	LDI		• Starts a logical NOT operation. (Starts a logical operation with normally closed contact)		
	AND		• Performs a logical AND operation (Normally open contact series connection)		
	ANDI		• Performs a negative AND operation (Normally closed contact series connection)		
	OR		• Performs a logical OR operation (Normally open contact parallel connection)		
	ORI		• Performs a negative OR operation (Normally closed contact parallel connection)		
	LDP		• Starts a rising edge operation		Page 86 Edge operation start, edge series connection, edge parallel connection
	LDF		• Starts a falling edge pulse operation		
	ANDP		• Rising edge series connection		
	ANDF		• Falling edge series connection		
	ORP		• Rising edge parallel connection		
	ORF		• Falling edge parallel connection		
	LDPI		• Starts a negated rising edge operation		Page 89 Negated edge operation start, negated edge series connection, negated edge pulse parallel connection
	LDFI		• Starts a negated falling edge operation		
	ANDPI		• Negated rising edge series connection		
	ANDFI		• Negated falling edge series connection		
	ORPI		• Negated rising edge parallel connection		
	ORFI		• Negated falling edge parallel connection		

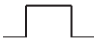
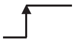
Bond instructions

Category	Instruction name	Symbol used in structured ladder/FBD	Description	Executing condition	Page
Bond	ANB		• Performs a logical AND operation between logical blocks (Series connection between logical blocks)		Page 92 Ladder block series connection and parallel connection
	ORB		• Performs a logical OR operation between logical blocks (Parallel connection between logical blocks)		
	MPS		• Stores operation results		Page 94 Operation result push/read/pop
	MRD		• Reads operation results stored in the MPS instruction		
	MPP		• Reads and resets operation results stored in the MPS instruction		
	INV	—	• Inverts operation results		Page 97 Operation result inversion
	MEP	—	• Outputs a pulse at a rising edge of operation result		Page 99 Pulse conversion of operation result
	MEF	—	• Outputs a pulse at a falling edge of operation result		
	EGP	—	• Outputs a pulse at a rising edge of operation result		Page 101 Pulse conversion of edge relay operation result
	EGF	—	• Outputs a pulse at a falling edge of operation result		

Output instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Output	OUT	(d)	• Outputs devices (excluding timers, counters and annunciators)		Page 103 Out (excluding timers, counters and annunciators)
	OUT_T	(s1), (s2), (d)	• Outputs devices (Timer)		Page 105 Timer
	OUTH_T				
	OUT_C	(s1), (s2), (d)	• Outputs devices (Counter)		Page 109 Counter
	OUT	(d)	• Outputs devices (Annunciator output)		Page 111 Annunciator output
	SET	(d)	• Sets devices		Page 113 Setting devices (excluding annunciators)
				When annunciator (F) is used 	Page 117 Setting and resetting annunciators
	RST	(d)	• Resets devices		Page 115 Resetting devices (excluding annunciators)
				When annunciator (F) is used 	Page 117 Setting and resetting annunciators
	PLS	(d)	• Generates a pulse for one program cycle at the rising of the input signal		Page 120 Rising edge and falling edge outputs
	PLF	(d)	• Generates a pulse for one program cycle at the falling of the input signal		
	FF	(s)	• Inverts device outputs		Page 123 Bit device output inversion
	DELTA	(d)	• Direct output pulse		Page 125 Pulse conversion of direct output
	DELTAP	(d)			

Shift instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Shift	SFT	(d)	• 1-bit shift of a device		Page 127 Bit device shift
	SFTP				

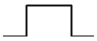
Master control instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Master control	MC	n*, (d)	• Starts master control		Page 130 Setting and resetting master control
	MCR	n*	• Resets master control		

End instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Program end	FEND	—	• Ends main program		Page 135 Ending main routine program

Other instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Stop	STOP	—	• Stops sequence program operations after the input condition is satisfied • Executes the sequence program when the RUN/STOP (key) is switched back to the RUN position		Page 137 Sequence program stop

2.4 Basic Instructions

Comparison operation instructions

Category	Instruction name	Argument	Description	Executing condition	Page
16-bit BIN data comparison	LD=	(s1), (s2)	• Conduction state when (s1) = (s2) • Non-conduction state when (s1) ≠ (s2)		Page 139 16-bit BIN data comparison
	AND=	(s1), (s2)			
	OR=	(s1), (s2)			
	LD<>	(s1), (s2)	• Conduction state when (s1) ≠ (s2) • Non-conduction state when (s1) = (s2)		
	AND<>	(s1), (s2)			
	OR<>	(s1), (s2)			
	LD<=	(s1), (s2)	• Conduction state when (s1) ≤ (s2) • Non-conduction state when (s1) > (s2)		
	AND<=	(s1), (s2)			
	OR<=	(s1), (s2)			
	LD<	(s1), (s2)	• Conduction state when (s1) < (s2) • Non-conduction state when (s1) ≥ (s2)		
	AND<	(s1), (s2)			
	OR<	(s1), (s2)			
	LD>=	(s1), (s2)	• Conduction state when (s1) ≥ (s2) • Non-conduction state when (s1) < (s2)		
	AND>=	(s1), (s2)			
	OR>=	(s1), (s2)			
	LD>	(s1), (s2)	• Conduction state when (s1) > (s2) • Non-conduction state when (s1) ≤ (s2)		
	AND>	(s1), (s2)			
	OR>	(s1), (s2)			
32-bit BIN data comparison	LDD=	(s1), (s2)	• Conduction state when ((s1)+1, (s1)) = ((s2)+1, (s2)) • Non-conduction state when ((s1)+1, (s1)) ≠ ((s2)+1, (s2))		Page 142 32-bit BIN data comparison
	ANDD=	(s1), (s2)			
	ORD=	(s1), (s2)			
	LDD<>	(s1), (s2)	• Conduction state when ((s1)+1, (s1)) ≠ ((s2)+1, (s2)) • Non-conduction state when ((s1)+1, (s1)) = ((s2)+1, (s2))		
	ANDD<>	(s1), (s2)			
	ORD<>	(s1), (s2)			
	LDD<=	(s1), (s2)	• Conduction state when ((s1)+1, (s1)) ≤ ((s2)+1, (s2)) • Non-conduction state when ((s1)+1, (s1)) > ((s2)+1, (s2))		
	ANDD<=	(s1), (s2)			
	ORD<=	(s1), (s2)			
	LDD<	(s1), (s2)	• Conduction state when ((s1)+1, (s1)) < ((s2)+1, (s2)) • Non-conduction state when ((s1)+1, (s1)) ≥ ((s2)+1, (s2))		
	ANDD<	(s1), (s2)			
	ORD<	(s1), (s2)			
	LDD>=	(s1), (s2)	• Conduction state when ((s1)+1, (s1)) ≥ ((s2)+1, (s2)) • Non-conduction state when ((s1)+1, (s1)) < ((s2)+1, (s2))		
	ANDD>=	(s1), (s2)			
	ORD>=	(s1), (s2)			
	LDD>	(s1), (s2)	• Conduction state when ((s1)+1, (s1)) > ((s2)+1, (s2)) • Non-conduction state when ((s1)+1, (s1)) ≤ ((s2)+1, (s2))		
	ANDD>	(s1), (s2)			
	ORD>	(s1), (s2)			

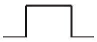
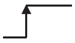
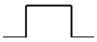
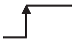
Category	Instruction name	Argument	Description	Executing condition	Page
Floating-point data comparison (single precision)	LDE=	(s1), (s2)	• Conduction state when ((s1)+1, (s1)) = ((s2)+1, (s2)) • Non-conduction state when ((s1)+1, (s1)) ≠ ((s2)+1, (s2))		Page 145 Floating-point data comparison (single precision)
	ANDE=	(s1), (s2)			
	ORE=	(s1), (s2)			
	LDE<>	(s1), (s2)	• Conduction state when ((s1)+1, (s1)) ≠ ((s2)+1, (s2)) • Non-conduction state when ((s1)+1, (s1)) = ((s2)+1, (s2))		
	ANDE<>	(s1), (s2)			
	ORE<>	(s1), (s2)			
	LDE<=	(s1), (s2)	• Conduction state when ((s1)+1, (s1)) ≤ ((s2)+1, (s2)) • Non-conduction state when ((s1)+1, (s1)) > ((s2)+1, (s2))		
	ANDE<=	(s1), (s2)			
	ORE<=	(s1), (s2)			
	LDE<	(s1), (s2)	• Conduction state when ((s1)+1, (s1)) < ((s2)+1, (s2)) • Non-conduction state when ((s1)+1, (s1)) ≥ ((s2)+1, (s2))		
	ANDE<	(s1), (s2)			
	ORE<	(s1), (s2)			
	LDE>=	(s1), (s2)	• Conduction state when ((s1)+1, (s1)) ≥ ((s2)+1, (s2)) • Non-conduction state when ((s1)+1, (s1)) < ((s2)+1, (s2))		
	ANDE>=	(s1), (s2)			
	ORE>=	(s1), (s2)			
	LDE>	(s1), (s2)	• Conduction state when ((s1)+1, (s1)) > ((s2)+1, (s2)) • Non-conduction state when ((s1)+1, (s1)) ≤ ((s2)+1, (s2))		
	ANDE>	(s1), (s2)			
	ORE>	(s1), (s2)			

Category	Instruction name	Argument	Description	Executing condition	Page
Floating-point data comparison (double precision)	LDED=	(s1), (s2)	• Conduction state when ((s1)+3, (s1)+2, (s1)+1, (s1)) = ((s2)+3, (s2)+2, (s2)+1, (s2)) • Non-conduction state when ((s1)+3, (s1)+2, (s1)+1, (s1)) ≠ ((s2)+3, (s2)+2, (s2)+1, (s2))		Page 148 Floating-point data comparison (double precision)
	ANDED=	(s1), (s2)			
	ORED=	(s1), (s2)			
	LDED<>	(s1), (s2)	• Conduction state when ((s1)+3, (s1)+2, (s1)+1, (s1)) ≠ ((s2)+3, (s2)+2, (s2)+1, (s2)) • Non-conduction state when ((s1)+3, (s1)+2, (s1)+1, (s1)) = ((s2)+3, (s2)+2, (s2)+1, (s2))		
	ANDED<>	(s1), (s2)			
	ORED<>	(s1), (s2)			
	LDED<=	(s1), (s2)	• Conduction state when ((s1)+3, (s1)+2, (s1)+1, (s1)) ≤ ((s2)+3, (s2)+2, (s2)+1, (s2)) • Non-conduction state when ((s1)+3, (s1)+2, (s1)+1, (s1)) > ((s2)+3, (s2)+2, (s2)+1, (s2))		
	ANDED<=	(s1), (s2)			
	ORED<=	(s1), (s2)			
	LDED<	(s1), (s2)	• Conduction state when ((s1)+3, (s1)+2, (s1)+1, (s1)) < ((s2)+3, (s2)+2, (s2)+1, (s2)) • Non-conduction state when ((s1)+3, (s1)+2, (s1)+1, (s1)) ≥ ((s2)+3, (s2)+2, (s2)+1, (s2))		
	ANDED<	(s1), (s2)			
	ORED<	(s1), (s2)			
	LDED>=	(s1), (s2)	• Conduction state when ((s1)+3, (s1)+2, (s1)+1, (s1)) ≥ ((s2)+3, (s2)+2, (s2)+1, (s2)) • Non-conduction state when ((s1)+3, (s1)+2, (s1)+1, (s1)) < ((s2)+3, (s2)+, (s2)+1, (s2))		
	ANDED>=	(s1), (s2)			
	ORED>=	(s1), (s2)			
	LDED>	(s1), (s2)	• Conduction state when ((s1)+3, (s1)+2, (s1)+1, (s1)) > ((s2)+3, (s2)+2, (s2)+1, (s2)) • Non-conduction state when ((s1)+3, (s1)+2, (s1)+1, (s1)) ≤ ((s2)+3, (s2)+2, (s2)+1, (s2))		
	ANDED>	(s1), (s2)			
	ORED>	(s1), (s2)			

Category	Instruction name	Argument	Description	Executing condition	Page
Character string data comparison	LD$=	(s1), (s2)	• Compares the character string of (s1) with the character string of (s2) character by character.[1] • Conduction state when (Character string of (s1)) = (Character string of (s2)) • Non-conduction state when (Character string of (s1)) ≠ (Character string of (s2))		Page 152 Character string data comparison
	AND$=	(s1), (s2)			
	OR$=	(s1), (s2)			
	LD$<>	(s1), (s2)	• Compares the character string of (s1) with the character string of (s2) character by character.[1] • Conduction state when (Character string of (s1)) ≠ (Character string of (s2)) • Non-conduction state when (Character string of (s1)) = (Character string of (s2))		
	AND$<>	(s1), (s2)			
	OR$<>	(s1), (s2)			
	LD$<=	(s1), (s2)	• Compares the character string of (s1) with the character string of (s2) character by character.[1] • Conduction state when (Character string of (s1)) ≤ (Character string of (s2)) • Non-conduction state when (Character string of (s1)) > (Character string of (s2))		
	AND$<=	(s1), (s2)			
	OR$<=	(s1), (s2)			
	LD$<	(s1), (s2)	• Compares the character string of (s1) with the character string of (s2) character by character.[1] • Conduction state when (Character string of (s1)) < (Character string of (s2)) • Non-conduction state when (Character string of (s1)) ≥ (Character string of (s2))		
	AND$<	(s1), (s2)			
	OR$<	(s1), (s2)			
	LD$>=	(s1), (s2)	• Compares the character string of (s1) with the character string of (s2) character by character.[1] • Conduction state when (Character string of (s1)) ≥ (Character string of (s2)) • Non-conduction state when (Character string of (s1)) < (Character string of (s2))		
	AND$>=	(s1), (s2)			
	OR$>=	(s1), (s2)			
	LD$>	(s1), (s2)	• Compares the character string of (s1) with the character string of (s2) character by character.[1] • Conduction state when (Character string of (s1)) > (Character string of (s2)) • Non-conduction state when (Character string of (s1)) ≤ (Character string of (s2))		
	AND$>	(s1), (s2)			
	OR$>	(s1), (s2)			

[1] The following shows the comparison conditions when character strings are compared.

- Match condition: All character strings are matched.
- Conditions for larger character string:For different character strings, character strings with larger character codes. For different character string lengths, larger character string lengths.
- Conditions for smaller character string:For different character strings, character strings with smaller character codes. For different character string lengths, smaller character string lengths.

Category	Instruction name	Argument	Description	Executing condition	Page
16-bit BIN block data comparison	BKCMP=	(s1), (s2), n, (d)	• Compares the 16-bit BIN data n points from the device specified for (s1) or constant with the 16-bit BIN data n points from the device specified for (s2), and stores the result to (d) and the following devices.		Page 156 16-bit BIN block data comparison
	BKCMP<>	(s1), (s2), n, (d)			
	BKCMP<=	(s1), (s2), n, (d)			
	BKCMP<	(s1), (s2), n, (d)			
	BKCMP>=	(s1), (s2), n, (d)			
	BKCMP>	(s1), (s2), n, (d)			
	BKCMP=P	(s1), (s2), n, (d)			
	BKCMP<>P	(s1), (s2), n, (d)			
	BKCMP<=P	(s1), (s2), n, (d)			
	BKCMP<P	(s1), (s2), n, (d)			
	BKCMP>=P	(s1), (s2), n, (d)			
	BKCMP>P	(s1), (s2), n, (d)			
32-bit BIN block data comparison	DBKCMP=	(s1), (s2), n, (d)	• Compares the 32-bit BIN data n points from the device specified for (s1) or constant with the 32-bit BIN data n points from the device specified for (s2), and stores the result to (d) and the following devices		Page 159 32-bit BIN block data comparison
	DBKCMP<>	(s1), (s2), n, (d)			
	DBKCMP<=	(s1), (s2), n, (d)			
	DBKCMP<	(s1), (s2), n, (d)			
	DBKCMP>=	(s1), (s2), n, (d)			
	DBKCMP>	(s1), (s2), n, (d)			
	DBKCMP=P	(s1), (s2), n, (d)			
	DBKCMP<>P	(s1), (s2), n, (d)			
	DBKCMP<=P	(s1), (s2), n, (d)			
	DBKCMP<P	(s1), (s2), n, (d)			
	DBKCMP>=P	(s1), (s2), n, (d)			
	DBKCMP>P	(s1), (s2), n, (d)			

Arithmetic operation instructions

Category	Instruction name	Argument	Description	Executing condition	Page
16-bit BIN data addition and subtraction	+	(s1), (s2), (d)	• (s1) + (s2) → (d)	⎍	Page 163 16-bit BIN data addition and subtraction
	+P			↑	
	-	(s1), (s2), (d)	• (s1)-(s2) → (d)	⎍	
	-P			↑	
32-bit BIN data addition and subtraction	D+	(s1), (s2), (d)	• ((s1)+1, (s1))+((s2)+1, (s2)) → ((d)+1, (d))	⎍	Page 165 32-bit BIN data addition and subtraction
	D+P			↑	
	D-	(s1), (s2), (d)	• ((s1)+1, (s1)) - ((s2)+1, (s2)) → ((d)+1, (d))	⎍	
	D-P			↑	
16-bit BIN data multiplication and division	*	(s1), (s2), (d)	• (s1) × (s2) → ((d)+1, (d))	⎍	Page 167 16-bit BIN data multiplication and division
	*P			↑	
	/	(s1), (s2), (d)	• (s1) ÷ (s2) → Quotient (d), Remainder ((d)+1)	⎍	
	/P			↑	
32-bit BIN data multiplication and division	D*	(s1), (s2), (d)	• ((s1)+1, (s1)) × ((s2)+1, (s2)) → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 170 32-bit BIN data multiplication and division
	D*P			↑	
	D/	(s1), (s2), (d)	• ((s1)+1, (s1)) ÷ ((s2)+1, (s2)) → Quotient ((d)+1, (d)), Remainder ((d)+3, (d)+2)	⎍	
	D/P			↑	
4-digit BCD data addition and subtraction	B+	(s1), (s2), (d)	• (s1) + (s2) → (d)	⎍	Page 173 4-digit BCD data addition and subtraction
	B+P			↑	
	B-	(s1), (s2), (d)	• (s1)-(s2) → (d)	⎍	
	B-P			↑	
8-digit BCD data addition and subtraction	DB+	(s1), (s2), (d)	• ((s1)+1, (s1))+((s2)+1, (s2)) → ((d)+1, (d))	⎍	Page 175 8-digit BCD data addition and subtraction
	DB+P			↑	
	DB-	(s1), (s2), (d)	• ((s1)+1, (s1)) - ((s2)+1, (s2)) → ((d)+1, (d))	⎍	
	DB-P			↑	
4-digit BCD data multiplication and division	B*	(s1), (s2), (d)	• (s1) × (s2) → ((d)+1, (d))	⎍	Page 177 4-digit BCD data multiplication and division
	B*P			↑	
	B/	(s1), (s2), (d)	• (s1) ÷ (s2) → Quotient (d), Remainder ((d)+1)	⎍	
	B/P			↑	

Category	Instruction name	Argument	Description	Executing condition	Page
8-digit BCD data multiplication and division	DB*	(s1), (s2), (d)	• ((s1)+1, (s1)) × ((s2)+1, (s2)) → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 179 8-digit BCD data multiplication and division
	DB*P			↑	
	DB/	(s1), (s2), (d)	• ((s1)+1, (s1)) ÷ ((s2)+1, (s2)) → Quotient ((d)+1, (d)), Remainder ((d)+3, (d)+2)	⎍	
	DB/P			↑	
Floating-point data addition and subtraction (single precision)	E+	(s1), (s2), (d)	• ((s1)+1, (s1))+((s2)+1, (s2)) → ((d)+1, (d))	⎍	Page 182 Floating-point data addition and subtraction (single precision)
	E+P			↑	
	E-	(s1), (s2), (d)	• ((s1)+1, (s1)) - ((s2)+1, (s2)) → ((d)+1, (d))	⎍	
	E-P			↑	
Floating-point data addition and subtraction (double precision)	ED+	(s1), (s2), (d)	• ((s1)+3, (s1)+2, (s1)+1, (s1))+((s2)+3, (s2)+2, (s2)+1, (s2)) → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 184 Floating-point data addition and subtraction (double precision)
	ED+P			↑	
	ED-	(s1), (s2), (d)	• ((s1)+3, (s1)+2, (s1)+1, (s1)) - ((s2)+3, (s2)+2, (s2)+1, (s2)) → ((d)+3, (d)+2, (d)+1, (d))	⎍	
	ED-P			↑	
Floating-point data multiplication and division (single precision)	E*	(s1), (s2), (d)	• ((s1)+1, (s1)) × ((s2)+1, (s2)) → ((d)+1, (d))	⎍	Page 186 Floating-point data multiplication and division (single precision)
	E*P			↑	
	E/	(s1), (s2), (d)	• ((s1)+1, (s1)) ÷ ((s2)+1, (s2)) → Quotient ((d)+1, (d))	⎍	
	E/P			↑	
Floating-point data multiplication and division (double precision)	ED*	(s1), (s2), (d)	• ((s1)+3, (s1)+2, (s1)+1, (s1)) × ((s2)+3, (s2)+2, (s2)+1, (s2)) → (d)+3, (d)+2, (d)+1, (d))	⎍	Page 188 Floating-point data multiplication and division (double precision)
	ED*P			↑	
	ED/	(s1), (s2), (d)	• ((s1)+3, (s1)+2, (s1)+1, (s1)) ÷ ((s2)+3, (s2)+2, (s2)+1, (s2)) → Quotient ((d)+3, (d)+2, (d)+1, (d))	⎍	
	ED/P			↑	
16-bit BIN block data addition and subtraction	BK+	(s1), (s2), n, (d)	• Batch adds 16-bit BIN data n points from (s1) and data of n points from (s2).	⎍	Page 190 16-bit BIN block data addition and subtraction
	BK+P			↑	
	BK-	(s1), (s2), n, (d)	• Batch subtracts 16-bit BIN data n points from (s2) from the data of n points from (s1).	⎍	
	BK-P			↑	

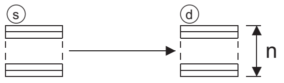
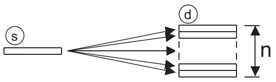
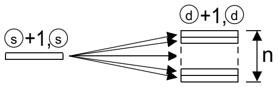
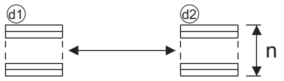
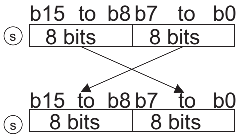
Category	Instruction name	Argument	Description	Executing condition	Page
32-bit BIN block data addition and subtraction	DBK+	(s1), (s2), n, (d)	• Adds 32-bit BIN data or contacts n points from the device specified for (s1) or constant and 32-bit BIN data n points from the device specified for (s2), and stores the result to (d) and the following devices.	⎍	Page 193 32-bit BIN block data addition and subtraction
	DBK+P			↑	
	DBK-	(s1), (s2), n, (d)	• Subtracts 32-bit BIN data or contacts n points from the device specified for (s2) or constant from 32-bit BIN data n points from the device specified for (s1), and stores the result to (d) and the following devices.	⎍	
	DBK-P			↑	
Character string data concatenation	$+	(s1), (s2), (d)	• Connects the character string specified for (s2) to the character string specified for (s1) and stores the result to (d) and the following devices.	⎍	Page 196 Character string data concatenation
	$+P			↑	
BIN data increment	INC	(d)	• (d) + 1 → (d)	⎍	Page 198 16-bit BIN data increment and decrement
	INCP			↑	
	DINC	(d)	• ((d)+1, (d))+1 → ((d)+1, (d))	⎍	Page 200 32-bit BIN data increment and decrement
	DINCP			↑	
BIN data decrement	DEC	(d)	• (d)-1 → (d)	⎍	Page 198 16-bit BIN data increment and decrement
	DECP			↑	
	DDEC	(d)	• ((d)+1, (d))-1 → ((d)+1, (d))	⎍	Page 200 32-bit BIN data increment and decrement
	DDECP			↑	

Data conversion instructions

Category	Instruction name	Argument	Description	Executing condition	Page
BCD data conversion	BCD	(s), (d)	BCD conversion • ⓢ → ⓓ BIN (0 to 9999)	⎍	Page 202 BIN data to 4-/8-digit BCD data conversion
	BCDP			↑	
	DBCD	(s), (d)	BCD conversion • (ⓢ+1,ⓢ) → (ⓓ+1,ⓓ) BIN (0 to 99999999)	⎍	
	DBCDP			↑	
BIN data conversion	BIN	(s), (d)	BIN conversion • ⓢ → ⓓ BCD (0 to 9999)	⎍	Page 205 4-/8-digit BCD data to BIN data conversion
	BINP			↑	
	DBIN	(s), (d)	BIN conversion • (ⓢ+1,ⓢ) → (ⓓ+1,ⓓ) BCD (0 to 99999999)	⎍	
	DBINP			↑	
BIN data to floating-point data conversion (single precision)	FLT	(s), (d)	Conversion to real number • ⓢ → (ⓓ+1,ⓓ) BIN(-32768 to 32767)	⎍	Page 208 16-/32-bit BIN data to floating-point data conversion (single precision)
	FLTP			↑	
	DFLT	(s), (d)	Conversion to real number • (ⓢ+1,ⓢ) → (ⓓ+1,ⓓ) BIN (-2147483648 to 2147483647)	⎍	
	DFLTP			↑	
BIN data to floating-point data conversion (double precision)	FLTD	(s), (d)	Conversion to real number • ⓢ → (ⓓ+3,ⓓ+2,ⓓ+1,ⓓ) BIN (-32768 to 32767)	⎍	Page 211 16-/32-bit BIN data to floating-point data conversion (double precision)
	FLTDP			↑	
	DFLTD	(s), (d)	Conversion to real number • (ⓢ+1,ⓢ) → (ⓓ+3,ⓓ+2,ⓓ+1,ⓓ) BIN (-2147483648 to 2147483647)	⎍	
	DFLTDP			↑	
Floating-point data to BIN data conversion (single precision)	INT	(s), (d)	Conversion to BIN data • (ⓢ+1,ⓢ) → ⓓ Real number (-32768 to 32767)	⎍	Page 213 Floating-point data to 16-/32-bit BIN data conversion (single precision)
	INTP			↑	
	DINT	(s), (d)	Conversion to BIN data • (ⓢ+1,ⓢ) → (ⓓ+1,ⓓ) Real number (-2147483648 to 2147483647)	⎍	
	DINTP			↑	
Floating-point data to BIN data conversion (double precision)	INTD	(s), (d)	Conversion to BIN data • (ⓢ+3,ⓢ+2,ⓢ+1,ⓢ) → ⓓ Real number (-32768 to 32767)	⎍	Page 216 Floating-point data to 16-bit/32-bit BIN data conversion (double precision)
	INTDP			↑	
	DINTD	(s), (d)	Conversion to BIN data • (ⓢ+3,ⓢ+2,ⓢ+1,ⓢ) → (ⓓ+1,ⓓ) Real number (-2147483648 to 2147483647)	⎍	
	DINTDP			↑	
16-bit BIN data to 32-bit BIN data conversion	DBL	(s), (d)	Conversion • ⓢ → (ⓓ+1,ⓓ) BIN (-32768 to 32767)	⎍	Page 219 16-bit BIN data to 32-bit BIN data conversion
	DBLP			↑	
32-bit BIN data to 16-bit BIN data conversion	WORD	(s), (d)	Conversion • (ⓢ+1,ⓢ) → ⓓ BIN (-32768 to 32767)	⎍	Page 221 32-bit BIN data to 16-bit BIN data conversion
	WORDP			↑	

Category	Instruction name	Argument	Description	Executing condition	Page
BIN data to Gray code data conversion	GRY	(s), (d)	Conversion to Gray code data • ⓢ → ⓓ BIN (-32768 to 32767)	⎍	Page 223 16-/32-bit BIN data to Gray code conversion
	GRYP			↑	
	DGRY	(s), (d)	Conversion to Gray code data • (ⓢ+1,ⓢ) → (ⓓ+1,ⓓ) BIN (-2147483648 to 2147483647)	⎍	
	DGRYP			↑	
Gray code data to BIN data conversion	GBIN	(s), (d)	Conversion to BIN data • ⓢ → ⓓ Gray code (-32768 to 32767)	⎍	Page 225 Gray code to 16-/32-bit BIN data conversion
	GBINP			↑	
	DGBIN	(s), (d)	Conversion to BIN data • (ⓢ+1,ⓢ) → (ⓓ+1,ⓓ) Gray code (-2147483648 to 2147483647)	⎍	
	DGBINP			↑	
Sign inversion (two's complement)	NEG	(d)	• ⓓ → ⓓ BIN data	⎍	Page 227 Two's complement of 16-/32-bit BIN data (sign inversion)
	NEGP			↑	
	DNEG	(d)	• (ⓓ+1,ⓓ) → (ⓓ+1,ⓓ) BIN data	⎍	
	DNEGP			↑	
Floating-point sign inversion (single-precision)	ENEG	(d)	• (ⓓ+1,ⓓ) → (ⓓ+1,ⓓ) Real number data	⎍	Page 229 Sign inversion of floating-point data (single precision)
	ENEGP			↑	
Floating-point sign inversion (double-precision)	EDNEG	(d)	• (ⓓ+3,ⓓ+2,ⓓ+1,ⓓ) → (ⓓ+3,ⓓ+2,ⓓ+1,ⓓ) Real number data	⎍	
	EDNEGP			↑	
Block data conversion	BKBCD	(s), n, (d)	• Batch converts BIN data n points from (s) to BCD data and stores the result to (d) and the following devices.	⎍	Page 233 16-bit BIN block data to 4-digit BCD block data conversion
	BKBCDP			↑	
	BKBIN	(s), n, (d)	• Batch converts BCD data n points from (s) to BIN data and stores the result to (d) and the following devices.	⎍	Page 236 4-digit BCD block data to 16-bit BIN block data conversion
	BKBINP			↑	
Floating-point data single precision to double precision conversion	ECON	(s), (d)	Conversion to double precision • (ⓢ+1,ⓢ) → (ⓓ+3,ⓓ+2,ⓓ+1,ⓓ) 32-bit floating-point real number	⎍	Page 239 Single-precision to double-precision conversion
	ECONP			↑	
Floating-point data double precision to single precision conversion	EDCON	(s), (d)	Conversion to single precision • (ⓢ+3,ⓢ+2,ⓢ+1,ⓢ) → (ⓓ+1,ⓓ) 64-bit floating-point real number	⎍	Page 241 Double-precision to single-precision conversion
	EDCONP			↑	

Data transfer instructions

Category	Instruction name	Argument	Description	Executing condition	Page
16-bit data transfer	MOV	(s), (d)	• ⓢ → ⓓ	⊓	Page 243 16-/32-bit data transfer
	MOVP			↑	
32-bit data transfer	DMOV	(s), (d)	• (ⓢ+1,ⓢ) → (ⓓ+1,ⓓ)	⊓	
	DMOVP			↑	
Floating-point data transfer (single precision)	EMOV	(s), (d)	• (ⓢ+1,ⓢ) → (ⓓ+1,ⓓ) Real number data	⊓	Page 245 Floating-point data transfer (single precision)
	EMOVP			↑	
Floating-point data transfer (double precision)	EDMOV	(s), (d)	• (ⓢ+3,ⓢ+2,ⓢ+1,ⓢ) → (ⓓ+3,ⓓ+2,ⓓ+1,ⓓ) Real number data	⊓	Page 247 Floating-point data transfer (double precision)
	EDMOVP			↑	
Character string data transfer	$MOV	(s), (d)	• Transfers the character string data specified for (s) to (d) and the following devices.	⊓	Page 249 Character string data transfer
	$MOVP			↑	
16-bit data negation transfer	CML	(s), (d)	• $\overline{\text{ⓢ}}$ → ⓓ	⊓	Page 252 16-/32-bit data negation transfer
	CMLP			↑	
32-bit data negation transfer	DCML	(s), (d)	• $\overline{\text{(ⓢ+1,ⓢ)}}$ → (ⓓ+1,ⓓ)	⊓	
	DCMLP			↑	
Block data transfer	BMOV	(s), n, (d)		⊓	Page 255 16-bit block data transfer
	BMOVP			↑	
Same 16-bit BIN data block transfer	FMOV	(s), n, (d)		⊓	Page 259 Identical 16-bit block data transfer
	FMOVP			↑	
Same 32-bit BIN data block transfer	DFMOV	(s), n, (d)		⊓	Page 262 Identical 32-bit block data transfer
	DFMOVP			↑	
16-bit data exchange	XCH	(d1), (d2)	• ⓓ1 ↔ ⓓ2	⊓	Page 265 16-/32-bit data exchange
	XCHP			↑	
32-bit data exchange	DXCH	(d1), (d2)	• (ⓓ1+1,ⓓ1) ↔ (ⓓ2+1,ⓓ2)	⊓	
	DXCHP			↑	
Block data exchange	BXCH	n, (d1), (d2)		⊓	Page 268 16-bit block data exchange
	BXCHP			↑	
Upper and lower bytes exchange	SWAP	(s)	 b15 to b8 b7 to b0 ⓢ 8 bits 8 bits b15 to b8 b7 to b0 ⓢ 8 bits 8 bits	⊓	Page 270 Upper and lower bytes exchange
	SWAPP			↑	

Program branch instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Jump	CJ	p	• Jumps to p when the input condition is satisfied.	⎍	Page 272 Pointer branch
	SCJ	p	• Jumps to p from the next scan after the input condition is satisfied.	⎍	
	JMP	p	• Unconditionally jumps to p.		
	GOEND	—	• Jumps to the FEND instruction when the input condition is satisfied.	⎍	Page 276 Jump to END processing

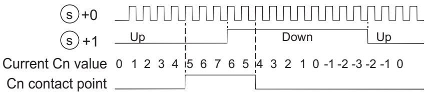
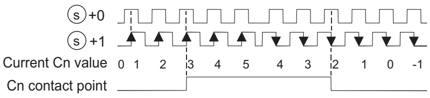
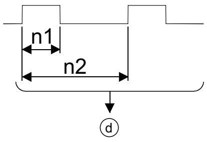
Program execution control instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Disable interrupt	DI	—	• Disables the execution of interrupt programs.		Page 278 Interrupt disable/enable, interrupt program mask
Enable interrupt	EI	—	• Cancels the execution disabled status of the interrupt program.		
Disable/enable interrupt setting	IMASK	(s)	• Disables/enables interrupts for each interrupt program.		
Recovery	IRET	—	• Returns to the sequence program from the interrupt program.		Page 287 Recovery from interrupt programs

I/O refresh instructions

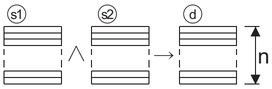
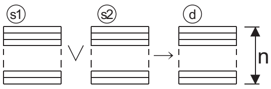
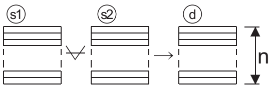
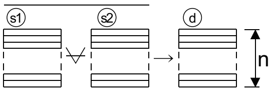
Category	Instruction name	Argument	Description	Executing condition	Page
I/O refresh	RFS	(s), n	• Performs partial refresh of the corresponding inputs/outputs during one scan.	⎍	Page 290 I/O refresh
	RFSP			↑	

Other convenient instructions

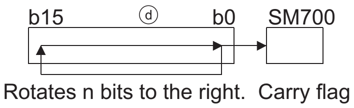
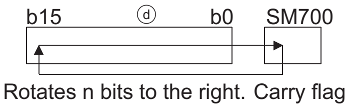
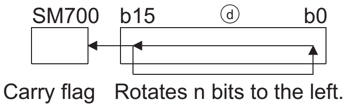
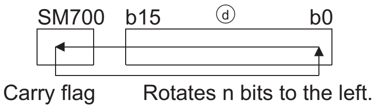
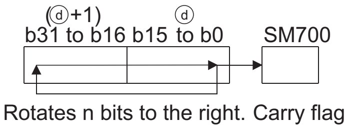
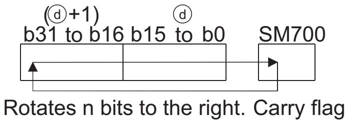
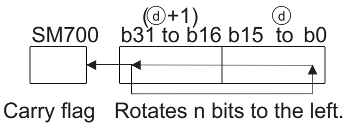
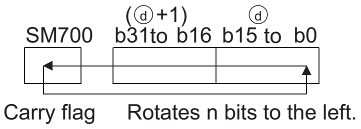
Category	Instruction name	Argument	Description	Executing condition	Page
Up/down counter	UDCNT1	(s), n, (d)	 ⓢ+0 ⓢ+1 Up Down Up Current Cn value 0 1 2 3 4 5 6 7 6 5 4 3 2 1 0 -1 -2 -3 -2 -1 0 Cn contact point	⎍	Page 292 Single-phase input up/down counter
	UDCNT2	(s), n, (d)	 ⓢ+0 ⓢ+1 Current Cn value 0 1 2 3 4 5 4 3 2 1 0 -1 Cn contact point	⎍	Page 295 Two-phase input up/down counter
Teaching timer	TTMR	n, (d)	• (Time that TTMR is ON) × n → ⓓ n=0:1, n=1:10, n=2:100	⎍	Page 298 Teaching timer
Special function timer	STMR	(s), n, (d)	• The 4 points from the bit device specified for (d) perform the following operations depending on the ON/OFF status of input condition of the STMR instruction: (d)+0: Off-delay timer output (d)+1: One-shot timer output after Off (d)+2: One-shot timer output after On (d)+3: On-delay + Off-delay timer output		Page 300 Special function timer
Shortest direction control	ROTC	(s), n1, n2, (d)	• Rotates the rotary table divided into n1, from the stop position to the position specified for ((s)+1) in the shortest path.	⎍	Page 303 Rotary table shortest direction control
Ramp signal	RAMP	n1, n2, n3, (d1), (d2)	• Changes the data of the device specified for (d1) from n1 to n2 in n3 scans.	⎍	Page 306 Ramp signal
Pulse density	SPD	(s), n, (d)	• Counts the pulse inputs of the device specified for (s) for the time specified by n and stores the result to the device specified for (d).	⎍	Page 309 Pulse density measurement
Fixed cycle pulse output	PLSY	n1, n2, (d)	• (n1)Hz → ⓓ Output n2 times	⎍	Page 311 Fixed cycle pulse output
Pulse width modulation	PWM	n1, n2, (d)		⎍	Page 313 Pulse width modulation
Matrix input	MTR	(s), n, (d1), (d2)	• Sequentially captures the data of 16 points multiplied by n columns from the device specified for (s), and stores the captured data to (d2) and the following devices.	⎍	Page 315 Matrix input

2.5 Application Instructions

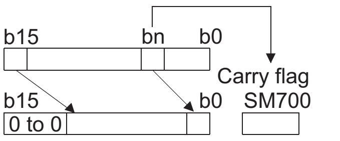
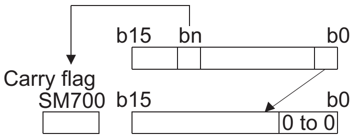
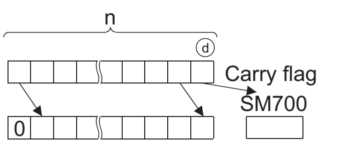
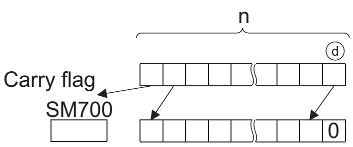
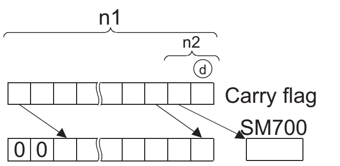
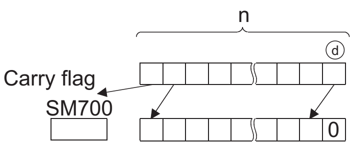
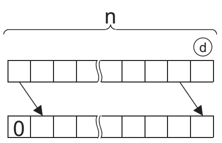
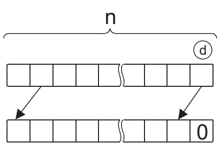
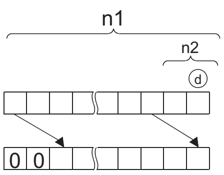
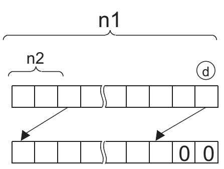
Logical operation instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Logical AND	WAND	(s1), (s2), (d)	(s1)∧(s2)→(d)	⎍	Page 319 Logical AND operation on 16-/32-bit data
	WANDP			↑	
	DAND	(s1), (s2), (d)	((s1)+1, (s1)) ∧ ((s2)+1, (s2)) → ((d)+1, (d))	⎍	
	DANDP			↑	
	BKAND	(s1), (s2), n, (d)		⎍	Page 322 Logical AND operation on block data
	BKANDP			↑	
Logical OR	WOR	(s1), (s2), (d)	(s1)∨(s2)→(d)	⎍	Page 325 Logical OR operation on 16-/32-bit data
	WORP			↑	
	DOR	(s1), (s2), (d)	((s1)+1, (s1)) ∨ ((s2)+1, (s2)) → ((d)+1, (d))	⎍	
	DORP			↑	
	BKOR	(s1), (s2), n, (d)		⎍	Page 328 Logical OR operation on block data
	BKORP			↑	
Exclusive OR	WXOR	(s1), (s2), (d)	(s1)$\veebar$(s2)→(d)	⎍	Page 331 Exclusive OR operation on 16-/32-bit data
	WXORP			↑	
	DXOR	(s1), (s2), (d)	((s1)+1, (s1)) $\veebar$ ((s2)+1, (s2)) → ((d)+1, (d))	⎍	
	DXORP			↑	
	BKXOR	(s1), (s2), n, (d)		⎍	Page 334 Exclusive OR operation on block data
	BKXORP			↑	
Exclusive NOR	WXNR	(s1), (s2), (d)	• $\overline{\textcircled{s1} \veebar \textcircled{s2}} \rightarrow \textcircled{d}$	⎍	Page 337 Exclusive NOR operation on 16-/32-bit data
	WXNRP			↑	
	DXNR	(s1), (s2), (d)	• $\overline{(\textcircled{s1}+1, \textcircled{s1}) \veebar (\textcircled{s2}+1, \textcircled{s2})} \rightarrow (\textcircled{d}+1, \textcircled{d})$	⎍	
	DXNRP			↑	
	BKXNR	(s1), (s2), n, (d)		⎍	Page 340 Exclusive NOR operation on block data
	BKXNRP			↑	

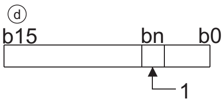
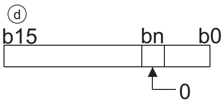
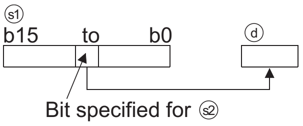
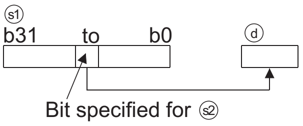
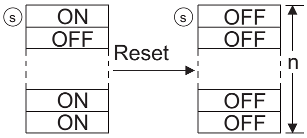
Rotation instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Right rotation	ROR	n, (d)	 b15 ⓓ b0 SM700 Rotates n bits to the right. Carry flag	⎍	Page 343 Right rotation of 16-bit data
	RORP			↑	
	RCR	n, (d)	 b15 ⓓ b0 SM700 Rotates n bits to the right. Carry flag	⎍	
	RCRP			↑	
Left rotation	ROL	n, (d)	 SM700 b15 ⓓ b0 Carry flag Rotates n bits to the left.	⎍	Page 347 Left rotation of 16-bit data
	ROLP			↑	
	RCL	n, (d)	 SM700 b15 ⓓ b0 Carry flag Rotates n bits to the left.	⎍	
	RCLP			↑	
Right rotation	DROR	n, (d)	 (ⓓ+1) ⓓ b31 to b16 b15 to b0 SM700 Rotates n bits to the right. Carry flag	⎍	Page 351 Right rotation of 32-bit data
	DRORP			↑	
	DRCR	n, (d)	 (ⓓ+1) ⓓ b31 to b16 b15 to b0 SM700 Rotates n bits to the right. Carry flag	⎍	
	DRCRP			↑	
Left rotation	DROL	n, (d)	 (ⓓ+1) ⓓ SM700 b31 to b16 b15 to b0 Carry flag Rotates n bits to the left.	⎍	Page 353 Left rotation of 32-bit data
	DROLP			↑	
	DRCL	n, (d)	 (ⓓ+1) ⓓ SM700 b31to b16 b15 to b0 Carry flag Rotates n bits to the left.	⎍	
	DRCLP			↑	

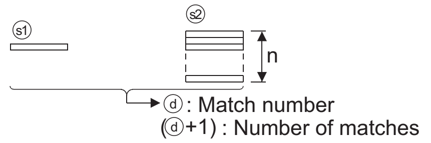
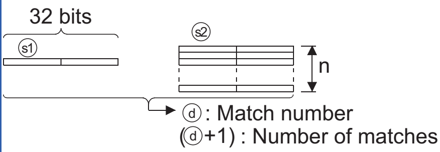
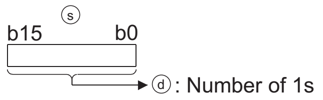
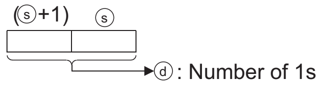
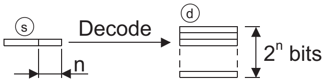
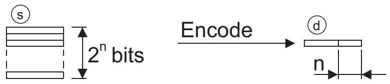
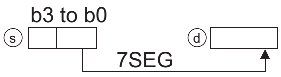
Shift instructions

Category	Instruction name	Argument	Description	Executing condition	Page
n-bit shift	SFR	n, (d)	 b15 bn b0 Carry flag b15 b0 SM700 0 to 0	⎍	Page 355 n-bit right/left shift of 16-bit data
	SFRP			↑	
	SFL	n, (d)	 b15 bn b0 Carry flag SM700 b15 b0 0 to 0	⎍	
	SFLP			↑	
1-bit shift of n-bit data	BSFR	n, (d)		⎍	Page 358 1-bit right/left shift of n-bit data
	BSFRP			↑	
	BSFL	n, (d)		⎍	
	BSFLP			↑	
n-bit shift of n-bit data	SFTBR	n1, n2, (d)		⎍	Page 361 n-bit right/left shift of n-bit data
	SFTBRP			↑	
	SFTBL	n1, n2, (d)		⎍	
	SFTBLP			↑	
1-word shift of n-word data	DSFR	n, (d)		⎍	Page 364 1-word right/left shift of n-word data
	DSFRP			↑	
	DSFL	n, (d)		⎍	
	DSFLP			↑	
n-word shift of n-word data	SFTWR	n1, n2, (d)		⎍	Page 367 n-word right/left shift of n-word data
	SFTWRP			↑	
	SFTWL	n1, n2, (d)		⎍	
	SFTWLP			↑	

Bit processing instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Bit set/reset	BSET	n, (d)		⎍	Page 370 Bit set and reset of word devices
	BSETP			↑	
	BRST	n, (d)		⎍	
	BRSTP			↑	
Bit test	TEST	(s1), (s2), (d)		⎍	Page 372 Bit test
	TESTP			↑	
	DTEST	(s1), (s2), (d)		⎍	
	DTESTP			↑	
Bit device batch reset	BKRST	(s), n	 s ON OFF ON ON Reset s OFF OFF OFF OFF n	⎍	Page 375 Batch reset of bit devices
	BKRSTP			↑	

Data processing instructions

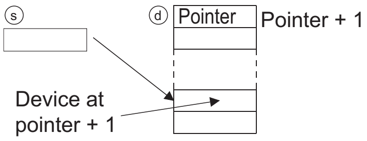
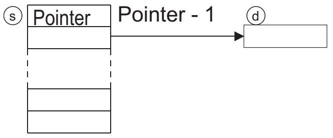
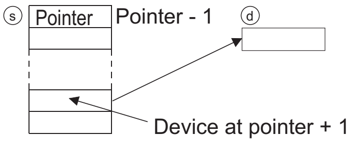
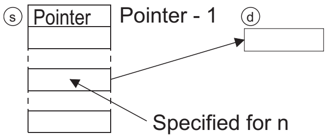
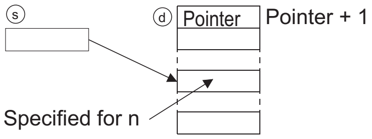
Category	Instruction name	Argument	Description	Executing condition	Page
Data search	SER	(s1), (s2), n, (d)	 s1 s2 n d : Match number (d+1) : Number of matches	⎍	Page 377 16-/ 32-bit data search
	SERP			↑	
	DSER	(s1), (s2), n, (d)	 32 bits s1 s2 n d : Match number (d+1) : Number of matches	⎍	
	DSERP			↑	
Bit check	SUM	(s), (d)		⎍	Page 381 16-/ 32-bit data bit check
	SUMP			↑	
	DSUM	(s), (d)		⎍	
	DSUMP			↑	
Decode	DECO	(s), n, (d)	Decode from 8 to 256 	⎍	Page 383 Decoding from 8 to 256 bits
	DECOP			↑	
Encode	ENCO	(s), n, (d)	Decode from 256 to 8 	⎍	Page 386 Encoding from 256 to 8 bits
	ENCOP			↑	
7-segment decode	SEG	(s), (d)		⎍	Page 389 7-segment decode
	SEGP			↑	
Separation and connection	DIS	(s), n, (d)	• Separates 16-bit data specified for (s) in units of 4 bits and stores the result to lower 4 bits n points from (d) (n≤4)	⎍	Page 392 4-bit separation of 16-bit data
	DIPS			↑	
	UNI	(s), n, (d)	• Connects lower 4-bit data n points from the device specified for (s) and stores the result to the device specified for (d) (n≤4)	⎍	Page 394 4-bit connection of 16-bit data
	UNIP			↑	
	NDIS	(s1), (s2), (d)	• Separates the data in (s1) and the following devices per bits specified by the devices from (s2), and stores the result to (d) and the following devices.	⎍	Page 396 Separation and connection of random data
	NDISP			↑	
	NUNI	(s1), (s2), (d)	• Connects the data in (s1) and the following devices per bits specified by the devices from (s2), and stores the result to (d) and the following devices.	⎍	
	NUNIP			↑	
	WTOB	(s), n, (d)	• Separates 16-bit data specified for (s) in units of 8 bits for n points of devices and stores the result in order from the device specified for (d).	⎍	Page 401 Separation and connection of data in units of bytes
	WTOBP			↑	
	BTOW	(s), n, (d)	• Connects lower 8 bits of 16-bit data for n points from the device specified for (s) to 16 bits and stores the result sequentially to the devices starting from the one specified for (d).	⎍	
	BTOWP			↑	

Category	Instruction name	Argument	Description	Executing condition	Page
Search	MAX	(s), n, (d)	• Searches the data n points from the device specified for (s) in units of 16 bits and stores the maximum value to the device specified for (d).	⎍	Page 405 Maximum value search of 16-/32-bit data
	MAXP			↑	
	DMAX	(s), n, (d)	• Searches the data 2 multiplied by n points from the device specified for (s) in units of 32 bits and stores the maximum value to the device specified for (d).	⎍	
	DMAXP			↑	
	MIN	(s), n, (d)	• Searches the data n points from the device specified for (s) in units of 16 bits and stores the minimum value to the device specified for (d).	⎍	Page 408 Minimum value search of 16-/32-bit data
	MINP			↑	
	DMIN	(s), n, (d)	• Searches the data 2 multiplied by n points from the device specified for (s) in units of 32 bits and stores the minimum value to the device specified for (d).	⎍	
	DMINP			↑	
Sort	SORT	(s1), n, (s2), (d1), (d2)	• Sorts the data n points from the device specified for (s1) in units of 16 bits. [n multiplied by (n-1) ÷ 2 scans required]	⎍	Page 411 Sorting 16-/32-bit data
	DSORT	(s1), n, (s2), (d1), (d2)	• Sorts the data 2 multiplied by n points from the device specified for (s1) in units of 32 bits. [n multiplied by (n-1) ÷ 2 scans required]	⎍	
Total calculation	WSUM	(s), n, (d)	• Adds all 16-bit BIN data n points from the device specified for (s) and stores the result to the device specified for (d).	⎍	Page 415 Total calculation of 16-bit data
	WSUMP			↑	
	DWSUM	(s), n, (d)	• Adds all 32-bit BIN data n points from the device specified for (s) and stores the result to the device specified for (d).	⎍	Page 417 Total calculation of 32-bit data
	DWSUMP			↑	
Average calculation	MEAN	(s), n, (d)	• Calculates the average of n points (unit of 16-bit) from the device specified for (s) and stores the result to the device specified for (d).	⎍	Page 419 Average calculation of 16-/32-bit data
	MEANP			↑	
	DMEAN	(s), n, (d)	• Calculates the average of n points (unit of 32-bit) from the device specified for (s) and stores the result to the device specified for (d).	⎍	
	DMEANP			↑	

Structured instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Repeat	FOR	n	• Executes the operation n times within the FOR to NEXT instruction loop.		Page 421 FOR to NEXT instruction loop
	NEXT	—			
	BREAK	p, (d)	• Forcibly terminates the execution of the [FOR] to [NEXT] instruction loop and jumps to the pointer p.	⎍	Page 424 Forced termination of FOR to NEXT instruction loop
	BREAKP			↑	
Subroutine program call	CALL	p	• Executes the subroutine program p when the input condition is satisfied.	⎍	Page 426 Subroutine program call
	CALLP			↑	
Return from subroutine program	RET	—	• Returns from the subroutine program.		Page 429 Return from subroutine program
Selection of refresh instructions	COM	—	• Executes the auto refresh of intelligent function modules, the auto refresh of link refresh, and the communication processing.		Page 430 Refresh
			• Executes the auto refresh of intelligent function modules, the link refresh, the auto refresh of CPU shared memory, and the communication processing.		Page 433 Selection of refresh(COM)
	CCOM	—	• Executes the auto refresh of intelligent function modules, the auto refresh of the CPU shared memory, and the communication processing.	⎍	Page 437 Selection of refresh(CCOM(P))
	CCOMP			↑	

Data table operation instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Data table process	FIFW	(s), (d)		⎍	Page 439 Writing data to data table
	FIFWP			↑	
	FIFR	(s), (d)		⎍	Page 442 Reading oldest data from data table
	FIFRP			↑	
	FPOP	(s), (d)		⎍	Page 445 Reading newest data from data table
	FPOPP			↑	
	FDEL	(s), n, (d)		⎍	Page 448 Deleting/ inserting data from/to data table
	FDELP			↑	
	FINS	(s), n, (d)		⎍	
	FINSP			↑	

Buffer memory access instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Data read	FROM	n1, n2, n3, (d)	• Reads data in units of 16 bits from the intelligent function module.	⊓	Page 451 Reading 1-/2-word data from intelligent function module
	FROMP			↑	
	DFRO	n1, n2, n3, (d)	• Reads data in units of 32 bits from the intelligent function module.	⊓	
	DFROP			↑	
Data write	TO	(s), n1, n2, n3	• Writes data in units of 16 bits to the intelligent function module.	⊓	Page 454 Writing 1-/2-word data to intelligent function module
	TOP			↑	
	DTO	(s), n1, n2, n3	• Writes data in units of 32 bits to the intelligent function module.	⊓	
	DTOP			↑	

Display instructions

Category	Instruction name	Argument	Description	Executing condition	Page
ASCII print	PR	(s), (d)	• When SM701 is ON, outputs 8 points of ASCII code (16 characters) from the device specified for (s) to the output module.	↑	Page 458 Printing ASCII code
	PR	(s), (d)	• When SM701 is OFF, outputs ASCII code data from the device specified for (s) up to 00H to the output module.		
	PRC	(s), (d)	• Converts comments in the device specified for (s) to ASCII code data and outputs the result to the output module.		Page 462 Printing comments
Reset	LEDR	—	• Resets the error display or the annunciator.	↑	Page 466 Resetting error display or annunciator

Debug/error diagnostics instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Check	CHKST	—	• Executes the CHK instruction when the CHKST instruction is executed. • Jumps to the step following the CHK instruction when the CHKST instruction is not executed.		Page 468 Special format error check
	CHK	—	• Normal operation → SM80: OFF, SD80: 0 • Abnormal operation → SM80: ON, SD80: Error number		
	CHKCIR	—	• Start of change in a ladder pattern checked by the CHK instruction		Page 472 Changing check format of the CHK instruction
	CHKEND	—	• End of change in a ladder pattern checked by the CHK instruction		

String processing instructions

Category	Instruction name	Argument	Description	Executing condition	Page
BIN data to decimal ASCII data conversion	BINDA	(s), (d)	• Converts 1 word of BIN data specified for (s) to 5 digits of decimal ASCII data and stores the result to the word device specified for (d).	⊓	Page 475 16-/32-bit BIN data to decimal ASCII data conversion
	BINDAP			↑	
	DBINDA	(s), (d)	• Converts 2 words of BIN data specified for (s) to 10 digits of decimal ASCII data and stores the result to (d) and the following word devices.	⊓	
	DBINDAP			↑	
BIN data to hexadecimal ASCII data conversion	BINHA	(s), (d)	• Converts 1 word of BIN data specified for (s) to 4 digits of hexadecimal ASCII data and stores the result to (d) and the following word devices.	⊓	Page 479 16-/32-bit BIN data to hexadecimal ASCII data conversion
	BINHAP			↑	
	DBINHA	(s), (d)	• Converts 2 words of BIN data specified for (s) to 8 digits of hexadecimal ASCII data and stores the result to (d) and the following word devices.	⊓	
	DBINHAP			↑	
BCD data to decimal ASCII data conversion	BCDDA	(s), (d)	• Converts 1 word of BCD data specified for (s) to 4 digits of decimal ASCII data and stores the result to (d) and the following word devices.	⊓	Page 482 4-/8-digit BCD data to decimal ASCII data conversion
	BCDDAP			↑	
	DBCDDA	(s), (d)	• Converts 2 words of BCD data specified for (s) to 8 digits of decimal ASCII data and stores the result to (d) and the following word devices.	⊓	
	DBCDDAP			↑	
Decimal ASCII data to BIN data conversion	DABIN	(s), (d)	• Converts 5 digits of decimal ASCII data specified for (s) to 1 word of BIN data and stores the result to the word device specified for (d).	⊓	Page 486 Decimal ASCII data to 16-/32-bit BIN data conversion
	DABINP			↑	
	DDABIN	(s), (d)	• Converts 10 digits of decimal ASCII data specified for (s) to 2 words of BIN data and stores the result to the word device specified for (d).	⊓	
	DDABINP			↑	
Hexadecimal ASCII data to BIN data conversion	HABIN	(s), (d)	• Converts 4 digits of hexadecimal ASCII data specified for (s) to 1 word of (16-bit) BIN data and stores the result to the word device specified for (d).	⊓	Page 489 Hexadecimal ASCII data to 16-/32-bit BIN data conversion
	HABINP			↑	
	DHABIN	(s), (d)	• Converts 8 digits of hexadecimal ASCII data specified for (s) to 2 words of BIN data and stores the result to the word device specified for (d).	⊓	
	DHABINP			↑	
Decimal ASCII data to BCD data conversion	DABCD	(s), (d)	• Converts 4 digits of decimal ASCII data specified for (s) to 1 word of BCD data and stores the result to the word device specified for (d).	⊓	Page 492 Decimal ASCII data to 4-/8-digit BCD data conversion
	DABCDP			↑	
	DDABCD	(s), (d)	• Converts 8 digits of decimal ASCII data specified for (s) to 2 words of BCD data and stores the result to the word device specified for (d).	⊓	
	DDABCDP			↑	
Device comment data read	COMRD	(s), (d)	• Stores the comment data from the device specified for (s) and stores it to the device specified for (d).	⊓	Page 495 Reading device comment data
	COMRDP			↑	

Category	Instruction name	Argument	Description	Executing condition	Page
Character string length detection	LEN	(s), (d)	• Stores the length (number of characters) of character string data stored in the device specified for (s) to the device specified for (d).	⎍	Page 498 Character string length detection
	LENP			↑	
BIN data to decimal character string data conversion	STR	(s1), (s2), (d)	• Converts 1 word of BIN value specified for (s2) to character string data according to the total number of digits and number of digits in the fractional part specified for (s1), and stores the result to the device specified for (d).	⎍	Page 500 16-/32-bit BIN data to character string data conversion
	STRP			↑	
	DSTR	(s1), (s2), (d)	• Converts 2 words of BIN value specified for (s2) to character string data according to the total number of digits and number of digits in the fractional part specified for (s1), and stores the result to the device specified for (d).	⎍	
	DSTRP			↑	
Decimal character string data to BIN data conversion	VAL	(s), (d1), (d2)	• Converts character string data contain a decimal point specified for (s) to 1 word of BIN value and number of digits in the fractional part, and stores the result to the devices specified for (d1) and (d2).	⎍	Page 505 Character string data to 16-/32-bit BIN data conversion
	VALP			↑	
	DVAL	(s), (d1), (d2)	• Converts character string data contain a decimal point specified for (s) to 2 words of BIN data and number of digits in the fractional part, and stores the result to the devices specified for (d1) and (d2).	⎍	
	DVALP			↑	
Floating-point data to character string data conversion	ESTR	(s1), (s2), (d)	• Converts floating-point data specified for (s) to character string data according to the display specification specified for (s) and stores the result to the device specified for (d).	⎍	Page 510 Floating-point data to character string data conversion
	ESTRP			↑	
Character string data to floating-point data conversion	EVAL	(s), (d)	• Converts character string data specified for (s) to floating-point data and stores the result to the device specified for (d).	⎍	Page 516 Character string data to floating-point data conversion
	EVALP			↑	
Hexadecimal BIN data to ASCII data conversion	ASC	(s), n, (d)	• Converts 1 word of BIN value in (s) and the following word devices to hexadecimal ASCII data and stores the number of characters specified for n to (d) and the following word devices.	⎍	Page 521 Hexadecimal BIN data to ASCII data conversion
	ASCP			↑	
ASCII data to hexadecimal BIN data conversion	HEX	(s), n, (d)	• Converts n numbers of hexadecimal ASCII data in (s) and the following word devices to BIN data and stores the result to (d) and the following word devices.	⎍	Page 524 ASCII data to hexadecimal BIN data conversion
	HEXP			↑	

Category	Instruction name	Argument	Description	Executing condition	Page
String processing	RIGHT	(s), n, (d)	• Stores n characters from the end of the character string data specified for (s) to the device specified for (d).	⎍	Page 527 Extraction of character string data from right/ left
	RIGHTP			↑	
	LEFT	(s), n, (d)	• Stores n characters from the start of the character string data specified for (s) to the device specified for (d).	⎍	
	LEFTP			↑	
	MIDR	(s1), (s2), (d)	• Stores the specified number of characters from the position specified for (s2) of the character string data specified for (s1) to the device specified for (d).	⎍	Page 530 Random selection and replacement in character string data
	MIDRP			↑	
	MIDW	(s1), (s2), (d)	• Stores the specified number of characters from the character string data (s1) to the position specified for (s2) of the character string data (d).	⎍	
	MIDWP			↑	
	INSTR	(s1), (s2), n, (d)	• Searches for the character string data (s1) from the nth character of the character string data (s2) and stores matched positions to (d).	⎍	Page 535 Character string data search
	INSTRP			↑	
	STRINS	(s), n, (d)	• Inserts the character string data specified for (s) to the n-th character (insert position) from the top of the character string data specified for (d).	⎍	Page 538 Character string data insert
	STRINSP			↑	
	STRDEL	n1, n2, (d)	• Deletes n2 characters from n1-th character (deletion start position) from the top of the character string data specified for (d).	⎍	Page 541 Character string data delete
	STRDELP			↑	
Floating-point data to BCD format data conversion	EMOD	(s1), (s2), (d)	• Converts floating-point data (s1) to BCD data of the fractional part digits specified for (s2) and stores the result to the device specified for (d).	⎍	Page 543 Floating-point data to BCD format conversion
	EMODP			↑	
BCD format data to floating-point data conversion	EREXP	(s1), (s2), (d)	• Converts BCD data (s1) to floating-point data with the fractional part digits specified for (s2) and stores the result to the device specified for (d).	⎍	Page 546 BCD format to floating-point data conversion
	EREXPP			↑	

Special function instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Trigonometric functions (single-precision floating-point data)	SIN	(s), (d)	• Sin((s)+1, (s)) → ((d)+1, (d))	⎍	Page 548 SIN operation on floating-point data (single precision)
	SINP			↑	
	COS	(s), (d)	• Cos((s)+1, (s)) → ((d)+1, (d))	⎍	Page 553 COS operation on floating-point data (single precision)
	COSP			↑	
	TAN	(s), (d)	• Tan((s)+1, (s)) → ((d)+1, (d))	⎍	Page 559 TAN operation on floating-point data (single precision)
	TANP			↑	
	ASIN	(s), (d)	• Sin^{-1}((s)+1, (s)) → ((d)+1, (d))	⎍	Page 565 SIN^{-1} operation on floating-point data (single precision)
	ASINP			↑	
	ACOS	(s), (d)	• Cos^{-1}((s)+1, (s)) → ((d)+1, (d))	⎍	Page 571 COS^{-1} operation on floating-point data (single precision)
	ACOSP			↑	
	ATAN	(s), (d)	• Tan^{-1}((s)+1, (s)) → ((d)+1, (d))	⎍	Page 577 TAN^{-1} operation on floating-point data (single precision)
	ATANP			↑	
Trigonometric functions (double-precision floating-point data)	SIND	(s), (d)	• Sin((s)+3, (s)+2, (s)+1, (s)) → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 551 SIN operation on floating-point data (double precision)
	SINDP			↑	
	COSD	(s), (d)	• Cos((s)+3, (s)+2, (s)+1, (s)) → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 556 COS operation on floating-point data (double precision)
	COSDP			↑	
	TAND	(s), (d)	• Tan((s)+3, (s)+2, (s)+1, (s)) → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 562 TAN operation on floating-point data (double precision)
	TANDP			↑	
	ASIND	(s), (d)	• Sin^{-1}((s)+3, (s)+2, (s)+1, (s)) → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 568 SIN^{-1} operation on floating-point data (double precision)
	ASINDP			↑	
	ACOSD	(s), (d)	• Cos^{-1}((s)+3, (s)+2, (s)+1, (s)) → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 574 COS^{-1} operation on floating-point data (double precision)
	ACOSDP			↑	
	ATAND	(s), (d)	• Tan^{-1}((s)+3, (s)+2, (s)+1, (s)) → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 580 TAN^{-1} operation on floating-point data (double precision)
	ATANDP			↑	

Category	Instruction name	Argument	Description	Executing condition	Page
Degree to radian, radian to degree conversion	RAD	(s), (d)	• ((s)+1, (s)) → ((d)+1, (d)) • Degree → Radian conversion	⎍	Page 583 Degree to radian conversion on floating-point data (single precision)
	RADP			↑	
	RADD	(s), (d)	• ((s)+3, (s)+2, (s)+1, (s)) → ((d)+3, (d)+2, (d)+1, (d)) • Degree → Radian conversion	⎍	Page 586 Degree to radian conversion on floating-point data (double precision)
	RADDP			↑	
	DEG	(s), (d)	• ((s)+1, (s)) → ((d)+1, (d)) • Radian → Degree conversion	⎍	Page 589 Radian to degree conversion on floating-point data (single precision)
	DEGP			↑	
	DEGD	(s), (d)	• ((s)+3, (s)+2, (s)+1, (s)) → ((d)+3, (d)+2, (d)+1, (d)) • Radian → Degree conversion	⎍	Page 592 Radian to degree conversion on floating-point data (double precision)
	DEGDP			↑	
Exponentiation	POW	(s1), (s2), (d)	• $((s1)+1, (s1))^{((s2)+1, (s2))}$ → ((d)+1, (d))	⎍	Page 595 Exponentiation on floating-point data (single precision)
	POWP			↑	
	POWD	(s1), (s2), (d)	• $((s1)+3, (s1)+2, (s1)+1, (s1))^{((s2)+3, (s2)+2, (s2)+1, (s2))}$ → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 598 Exponentiation on floating-point data (double precision)
	POWDP			↑	
Square root	SQR	(s), (d)	• $\sqrt{((s)+1, (s))}$ → ((d)+1, (d))	⎍	Page 601 Square root operation on floating-point data (single precision)
	SQRP			↑	
	SQRD	(s), (d)	• $\sqrt{((s)+3, (s)+2, (s)+1, (s))}$ → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 604 Square root operation on floating-point data (double precision)
	SQRDP			↑	
Exponential	EXP	(s), (d)	• $e^{((s)+1, (s))}$ → ((d)+1, (d))	⎍	Page 607 Exponential operation on floating-point data (single precision)
	EXPP			↑	
	EXPD	(s), (d)	• $e^{((s)+3, (s)+2, (s)+1, (s))}$ → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 610 Exponential operation on floating-point data (double precision)
	EXPDP			↑	

Category	Instruction name	Argument	Description	Executing condition	Page
Natural logarithm	LOG	(s), (d)	• $\log_e$((s)+1, (s)) → ((d)+1, (d))	⎍	Page 613 Natural logarithm operation on floating-point data (single precision)
	LOGP			↑	
	LOGD	(s), (d)	• $\log_e$((s)+3, (s)+2, (s)+1, (s)) → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 616 Natural logarithm operation on floating-point data (double precision)
	LOGDP			↑	
Common logarithm operation	LOG10	(s), (d)	• $\log_{10}$((s)+1, (s)) → ((d)+1, (d))	⎍	Page 619 Common logarithm operation on floating-point data (single precision)
	LOG10P			↑	
	LOG10D	(s), (d)	• $\log_{10}$((s)+3, (s)+2, (s)+1, (s)) → ((d)+3, (d)+2, (d)+1, (d))	⎍	Page 622 Common logarithm operation on floating-point data (double precision)
	LOG10DP			↑	
Random number generation	RND	(d)	Generates random numbers in the range from 0 to number less than 32767 and stores it to the device specified for (d).	⎍	Page 625 Random number generation and series update
	RNDP			↑	
Random number series update	SRND	(s)	Updates the random number series according to the content of 16-bit BIN data stored in the device specified for (s).	⎍	
	SRNDP			↑	
Square root	BSQR	(s), (d)	• $\sqrt{(s)}$ → (d)+0 Integer part +1 Fractional part	⎍	Page 627 Square root operation on 4-/8-digit BCD data
	BSQRP			↑	
	BDSQR	(s), (d)	• $\sqrt{(s)+1,(s)}$ → (d)+0 Integer part +1 Fractional part	⎍	
	BDSQRP			↑	

Category	Instruction name	Argument	Description	Executing condition	Page
Trigonometric functions (BCD format)	BSIN	(s), (d)	• Sin ⓢ → ⓓ+0 Sign +1 Integer part +2 Fractional part	⎍	Page 630 SIN operation on data in BCD format
	BSINP			↑	
	BCOS	(s), (d)	• Cos ⓢ → ⓓ+0 Sign +1 Integer part +2 Fractional part	⎍	Page 633 COS operation on data in BCD format
	BCOSP			↑	
	BTAN	(s), (d)	• Tan ⓢ → ⓓ+0 Sign +1 Integer part +2 Fractional part	⎍	Page 636 TAN operation on data in BCD format
	BTANP			↑	
	BASIN	(s), (d)	• Sin^{-1}ⓢ → ⓓ+0 Sign +1 Integer part +2 Fractional part	⎍	Page 639 SIN^{-1} operation on data in BCD format
	BASINP			↑	
	BACOS	(s), (d)	• Cos^{-1}ⓢ → ⓓ+0 Sign +1 Integer part +2 Fractional part	⎍	Page 642 COS^{-1} operation on data in BCD format
	BACOSP			↑	
	BATAN	(s), (d)	• Tan^{-1}ⓢ → ⓓ+0 Sign +1 Integer part +2 Fractional part	⎍	Page 645 TAN^{-1} operation on data in BCD format
	BATANP			↑	

Data control instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Upper/lower limit control	LIMIT	(s1), (s2), (s3), (d)	• When (s3) < (s1), stores the value of (s1) to (d). • When (s1) ≤ (s3) ≤ (s2), stores the value of (s3) to (d). • When (s2) < (s3), stores the value of (s2) to (d).	⊓	Page 647 Upper and lower limit controls of 16-/32-bit BIN data
	LIMITP			↑	
	DLIMIT	(s1), (s2), (s3), (d)	• When ((s3)+1, (s3)) < ((s1)+1, (s1)), stores the values of ((s1)+1, (s1)) to ((d)+1, (d)). • When ((s1)+1, (s1)) ≤ ((s3)+1, (s3)) < ((s2)+1, (s2)), stores the values of ((s3)+1, (s3)) to ((d)+1, (d)). • When ((s2), (s2)+1) < ((s3), (s3)+1), stores the values of ((s2)+1, (s2)) to ((d)+1, (d)).	⊓	
	DLIMITP			↑	
Dead band control	BAND	(s1), (s2), (s3), (d)	• When (s1) ≤ (s3) ≤ (s2), 0 → (d) • When (s3) < (s1), (s3) - (s1) → (d) • When (s2) < (s3), (s3) - (s2) → (d)	⊓	Page 650 Dead band control of 16-/32-bit BIN data
	BANDP			↑	
	DBAND	(s1), (s2), (s3), (d)	• When ((s1)+1, (s1)) ≤ ((s3)+1, (s3)) ≤ ((s2)+1, (s2)), 0 → ((d)+1, (d)) • When ((s3)+1, (s3)) < ((s1)+1, (s1)), ((s3)+1, (s3)) - ((s1)+1, (s1)) → ((d)+1, (d)) • When ((s2)+1, (s2)) < ((s3)+1, (s3)), ((s3)+1, (s3)) - ((s2)+1, (s2)) → ((d)+1, (d))	⊓	
	DBANDP			↑	
Zone control	ZONE	(s1), (s2), (s3), (d)	• When (s3) = 0, 0 → (d) • When (s3) > 0, (s3) + (s2) → (d) • When (s3) < 0, (s3) + (s1) → (d)	⊓	Page 653 Zone control of 16-/32-bit BIN data
	ZONEP			↑	
	DZONE	(s1), (s2), (s3), (d)	• When ((s3)+1, (s3)) = 0, 0 → ((d)+1, (d)) • When ((s3)+1, (s3)) > 0, ((s3)+1, (s3)) + ((s2)+1, (s2)) → ((d)+1, (d)) • When ((s3)+1, (s3)) < 0, ((s3)+1, (s3)) + ((s1)+1, (s1)) → ((d)+1, (d))	⊓	
	DZONEP			↑	
Coordinate by point data	SCL	(s1), (s2), (d)	• Process the scaling on the conversion data (unit of 16-bit data) specified for (s2) with the input value specified for (s1), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in (s2) and the following devices.	⊓	Page 656 Scaling (coordinate by point data)
	SCLP			↑	
	DSCL	(s1), (s2), (d)	• Process the scaling on the conversion data (unit of 32-bit data) specified for (s2) with the input value specified for (s1), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in (s2) and the following devices.	⊓	
	DSCLP			↑	

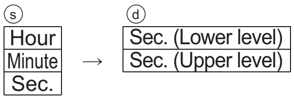
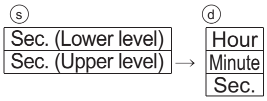
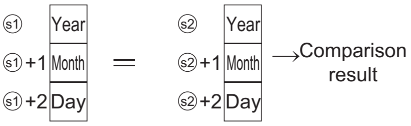
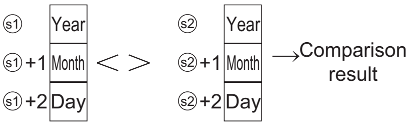
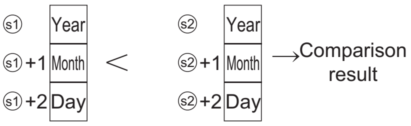
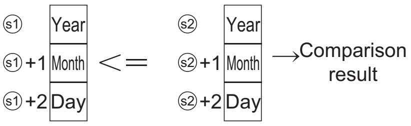
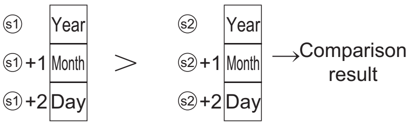
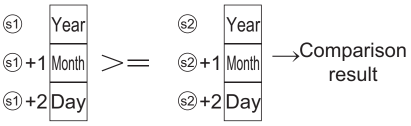
Category	Instruction name	Argument	Description	Executing condition	Page
Coordinate by X/Y data	SCL2	(s1), (s2), (d)	• Process the scaling on the conversion data (unit of 16-bit data) specified for (s2) with the input value specified for (s1), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in (s2) and the following devices.	⎍	Page 660 Scaling (coordinate by X/Y data)
	SCL2P			↑	
	DSCL2	(s1), (s2), (d)	• Process the scaling on the conversion data (unit of 32-bit data) specified for (s2) with the input value specified for (s1), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in (s2) and the following devices.	⎍	
	DSCL2P			↑	

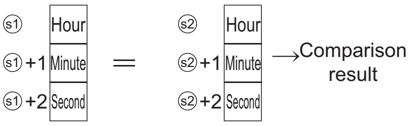
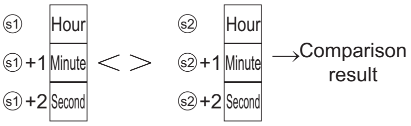
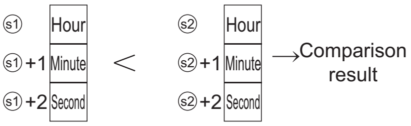
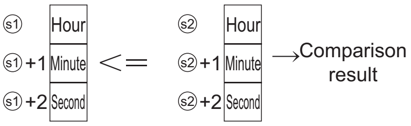
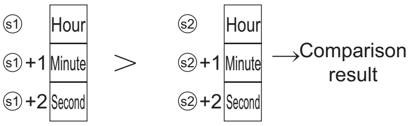
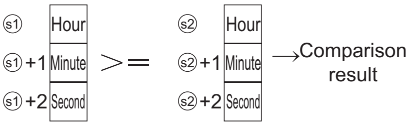
File register switching instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Block number switch	RSET	(s)	• Changes the block number of the extended file register to the number specified for (s).	⎍	Page 664 Switching file register block numbers
	RSETP			↑	
File set	QDRSET	(s)	• Sets the file name used as a file register.	⎍	Page 667 Setting file register files
	QDRSETP			↑	
	QCDSET	(s)	• Sets file names used as comment files.	⎍	Page 670 Setting files for comments
	QCDSETP			↑	

Clock instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Clock data read/write	DATERD	(d)	• (Clock elements) → ⓓ+0 Year +1 Month +2 Day +3 Hour +4 Minute +5 Sec. +6 Day of week	⎍	Page 673 Reading clock data
	DATERDP			↑	
	DATEWR	(s)	• ⓓ+0 Year →(Clock elements) +1 Month +2 Day +3 Hour +4 Minute +5 Sec. +6 Day of week	⎍	Page 676 Writing clock data
	DATEWRP			↑	
Clock data addition/subtraction	DATE+	(s1), (s2), (d)	(s1) Hour Minute Sec. + (s2) Hour Minute Sec. → ⓓ Hour Minute Sec.	⎍	Page 679 Clock data addition
	DATE+P			↑	
	DATE-	(s1), (s2), (d)	(s1) Hour Minute Sec. − (s2) Hour Minute Sec. → ⓓ Hour Minute Sec.	⎍	Page 681 Clock data subtraction
	DATE-P			↑	

Category	Instruction name	Argument	Description	Executing condition	Page
Clock data conversion	SECOND	(s), (d)	 s Hour Minute Sec. → d Sec. (Lower level) Sec. (Upper level)	⎍	Page 684 Time data conversion (hour/minute/second format to seconds)
	SECONDP			↑	
	HOUR	(s), (d)	 s Sec. (Lower level) Sec. (Upper level) → d Hour Minute Sec.	⎍	Page 686 Time data conversion (seconds to hour/minute/second format)
	HOURP			↑	
Date comparison	LDDT=	(s1), (s2), n	 s1 Year s1+1 Month s1+2 Day = s2 Year s2+1 Month s2+2 Day →Comparison result		Page 688 Date data comparison
	ANDDT=	(s1), (s2), n			
	ORDT=	(s1), (s2), n			
	LDDT<>	(s1), (s2), n	 s1 Year s1+1 Month s1+2 Day <> s2 Year s2+1 Month s2+2 Day →Comparison result		
	ANDDT<>	(s1), (s2), n			
	ORDT<>	(s1), (s2), n			
	LDDT<	(s1), (s2), n	 s1 Year s1+1 Month s1+2 Day < s2 Year s2+1 Month s2+2 Day →Comparison result		
	ANDDT<	(s1), (s2), n			
	ORDT<	(s1), (s2), n			
	LDDT<=	(s1), (s2), n	 s1 Year s1+1 Month s1+2 Day <= s2 Year s2+1 Month s2+2 Day →Comparison result		
	ANDDT<=	(s1), (s2), n			
	ORDT<=	(s1), (s2), n			
	LDDT>	(s1), (s2), n	 s1 Year s1+1 Month s1+2 Day > s2 Year s2+1 Month s2+2 Day →Comparison result		
	ANDDT>	(s1), (s2), n			
	ORDT>	(s1), (s2), n			
	LDDT>=	(s1), (s2), n	 s1 Year s1+1 Month s1+2 Day >= s2 Year s2+1 Month s2+2 Day →Comparison result		
	ANDDT>=	(s1), (s2), n			
	ORDT>=	(s1), (s2), n			

Category	Instruction name	Argument	Description	Executing condition	Page
Time data comparison	LDTM=	(s1), (s2), n	 s1 Hour s1+1 Minute s1+2 Second = s2 Hour s2+1 Minute s2+2 Second →Comparison result		Page 692 Time data comparison
	ANDTM=	(s1), (s2), n			
	ORTM=	(s1), (s2), n			
	LDTM<>	(s1), (s2), n	 s1 Hour s1+1 Minute s1+2 Second <> s2 Hour s2+1 Minute s2+2 Second →Comparison result		
	ANDTM<>	(s1), (s2), n			
	ORTM<>	(s1), (s2), n			
	LDTM<	(s1), (s2), n	 s1 Hour s1+1 Minute s1+2 Second < s2 Hour s2+1 Minute s2+2 Second →Comparison result		
	ANDTM<	(s1), (s2), n			
	ORTM<	(s1), (s2), n			
	LDTM<=	(s1), (s2), n	 s1 Hour s1+1 Minute s1+2 Second <= s2 Hour s2+1 Minute s2+2 Second →Comparison result		
	ANDTM<=	(s1), (s2), n			
	ORTM<=	(s1), (s2), n			
	LDTM>	(s1), (s2), n	 s1 Hour s1+1 Minute s1+2 Second > s2 Hour s2+1 Minute s2+2 Second →Comparison result		
	ANDTM>	(s1), (s2), n			
	ORTM>	(s1), (s2), n			
	LDTM>=	(s1), (s2), n	 s1 Hour s1+1 Minute s1+2 Second >= s2 Hour s2+1 Minute s2+2 Second →Comparison result		
	ANDTM>=	(s1), (s2), n			
	ORTM>=	(s1), (s2), n			

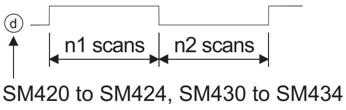
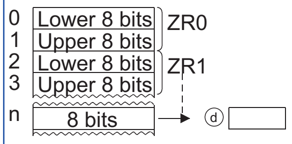
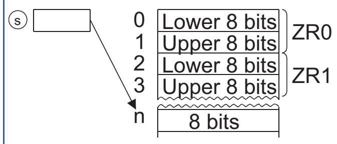
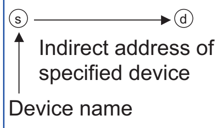
Extended clock instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Extended clock data read	S_DATERD	(d)	• (Clock elements) → ⓓ +0 Year +1 Month +2 Day +3 Hour +4 Minute +5 Sec. +6 Day of week +7 Millisecond	⎍	Page 696 Read extended clock data
	SP_DATERD			↑	
Extended clock data addition/ subtraction	S_DATE+	(s1), (s2), (d)	ⓢ1 ⓢ2 ⓓ Hour Hour Hour Minute Minute Minute Sec. + Sec. → Sec. — — — Millisecond Millisecond Millisecond	⎍	Page 700 Addition of extended clock data
	SP_DATE+			↑	
	S_DATE-	(s1), (s2), (d)	ⓢ1 ⓢ2 ⓓ Hour Hour Hour Minute Minute Minute Sec. − Sec. → Sec. — — — Millisecond Millisecond Millisecond	⎍	Page 703 Subtraction of extended clock data
	SP_DATE-			↑	

Program control instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Program control instructions	PSTOP	(s)	• Sets the specified program in the standby status.	⎍	Page 706 Program standby
	PSTOPP			↑	
	POFF	(s)	• Turns OFF the coil of the OUT instruction of the specified program and sets the program in the standby status.	⎍	Page 708 Program output OFF standby
	POFFP			↑	
	PSCAN	(s)	• Registers the specified program as a scan execution type program.	⎍	Page 710 Registering program as scan execution type
	PSCANP			↑	
	PLOW	(s)	• Registers the specified program as a low-speed execution type program.	⎍	Page 712 Registering program as low-speed execution type
	PLOWP			↑	
	LDPCHK	(s)	• Conduction state when the program of the specified file name is in execution. • Non-conduction state when the program of the specified file name is not in execution.		Page 714 Checking program execution status
	ANDPCHK	(s)			
	ORPCHK	(s)			

Other instructions

Category	Instruction name	Argument	Description	Executing condition	Page
WDT reset	WDT	—	• Resets WDT in the sequence program.	⎍	Page 716 Resetting watchdog timer
	WDTP			↑	
Timing clock	DUTY	n1, n2, (d)		↑	Page 718 Timing pulse generation
Time check	TIMCHK	(s1), (s2), (d)	• Measures the ON time of the input condition, and when the input condition remains ON over the preset time, turns ON the device specified for (d).	⎍	Page 720 Time check
Direct 1-byte read/write	ZRRDB	n, (d)	 0 1 2 3 n Lower 8 bits Upper 8 bits Lower 8 bits Upper 8 bits 8 bits ZR0 ZR1 d	⎍	Page 722 Reading 1 byte directly from file register
	ZRRDBP			↑	
	ZRWRB	n, (s)	 s 0 1 2 3 n Lower 8 bits Upper 8 bits Lower 8 bits Upper 8 bits 8 bits ZR0 ZR1	⎍	Page 725 Writing 1 byte directly to file register
	ZRWRBP			↑	
	ADRSET	(s), (d)	 s d Indirect address of specified device Device name	⎍	Page 728 Reading indirect address
	ADRSETP			↑	
Numeric input from keyboard	KEY	(s), n, (d1), (d2)	• Imports ASCII data into 8 points of the input module specified for (s), converts the data to hexadecimal data and stores the result to (d1) and the following devices.	⎍	Page 729 Numeric input from keyboard
Batch save of index register	ZPUSH	(d)	• Saves the contents of index registers to (d) and the following devices.	⎍	Page 733 Batch save and recovery of index registers
	ZPUSHP			↑	
Batch recovery of index register	ZPOP	(d)	• Reads data saved in (d) and the following devices into the index registers.	⎍	
	ZPOPP			↑	
Module information read	UNIRD	n1, n2, (d)	• Reads the module information whose amount is specified for n2 from the start I/O number specified for n1, and stores the information to (d) and the following devices.	⎍	Page 736 Reading module information
	UNIRDP			↑	
Module type read	TYPERD	n, (d)	• Reads the module type of the start I/O number specified for n, and stores the information to (d) and the following devices.	⎍	Page 742 Reading module type
	TYPERDP			↑	
Trace set	TRACE	—	• Stores trace data set in the peripheral in the number of set times when SM800, SM801, or SM802 is turned ON to the sampling trace file.	↑	Page 747 Trace set/reset
Trace reset	TRACER	—	• Resets data set by the TRACE instruction.	↑	
Data write to specified file	SP_FWRITE	(s0), (s1), (s2), (d0), (d1)	• Writes data to the specified file.	↑	Page 750 Writing data to specified file
Data read from specified file	SP_FREAD	(s0), (s1), (d0), (d1), (d2)	• Reads data from the specified file.	↑	Page 763 Reading data from specified file

Category	Instruction name	Argument	Description	Executing condition	Page
Data write to standard ROM	SP_DEVST	(s1), (s2), n, (d)	• Writes data to the device data storage file in the standard ROM.	↑	Page 777 Writing data to standard ROM
Data read from standard ROM	S_DEVLD	(s), n, (d)	• Reads data from the device data storage file in the standard ROM.	⎍	Page 780 Reading data from standard ROM
	SP_DEVLD			↑	
Program load from memory	PLOADP	(s), (d)	• Transfers a program stored in the memory card or standard ROM (other than drive 0) to drive 0 and sets the program in the standby type.	↑	Page 782 Program load from memory card
Program unload from program memory	PUNLOADP	(s), (d)	• Deletes a standby type program stored in the program memory (drive 0) from the memory.	↑	Page 785 Program unload from program memory
Load + unload	PSWAPP	(s1), (s2), (d)	• Deletes a standby type program stored in the program memory (drive 0) specified for (s1) from the memory, and transfers a program stored in the memory card or standard ROM (other than drive 0) specified for (s2) to drive 0 and places the program in the standby type.	↑	Page 788 Load and unload
File register high-speed block transfer	RBMOV	(s), n, (d)	• Batch transfers n points of 16-bit data from the device specified for (s) to n points of devices starting from the one specified for (d).	⎍	Page 791 File register high-speed block transfer
	RBMOVP			↑	
User message	UMSG	(s)	• Displays specified character strings on the display module as a user message.	⎍	Page 796 User message

2.6 Data Link Instructions

Network refresh instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Network refresh	S_ZCOM_J	Jn*	• Refreshes specified network.	⊓	Page 799 Network Refresh Instructions
	SP_ZCOM_J			↑	
	S_ZCOM_U	Un*		⊓	
	SP_ZCOM_U			↑	

Reading/Registering routing information

Category	Instruction name	Argument	Description	Executing condition	Page
Routing information read	S_RTREAD	n, (d)	• Reads data set in the routing parameter.	⊓	Page 803 Reading routing information
	SP_RTREAD			↑	
Routing information registration	S_RTWRITE	n, (s)	• Writes routing data to the location specified in the routing parameter.	⊓	Page 805 Registering routing information
	SP_RTWRITE			↑	

Refresh device write/read instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Refresh device write	S_REFDVWRB	n1, (s1), (s2), n2, (d1)	• Writes data in 1-bit units to the specified refresh device.	⊓	Page 808 Refresh device write (in 1-bit units)
	SP_REFDVWRB			↑	
	S_REFDVWRW	n1, (s1), (s2), n2, (d1)	• Writes data in 16-bit units to the specified refresh device.	⊓	Page 813 Refresh device write (in 16-bit units)
	SP_REFDVWRW			↑	
Refresh device read	S_REFDVRDB	n1, (s1), (s2), n2, (d1)	• Reads data in 1-bit units from the specified refresh device.	⊓	Page 818 Refresh device read (in 1-bit units)
	SP_REFDVRDB			↑	
	S_REFDVRDW	n1, (s1), (s2), n2, (d1)	• Reads data in 16-bit units from the specified refresh device.	⊓	Page 823 Refresh device read (in 16-bit units)
	SP_REFDVRDW			↑	

2.7 Multiple CPU Dedicated Instructions

Writing data to host CPU shared memory

Category	Instruction name	Argument	Description	Executing condition	Page
Data write to host CPU shared memory	S_TO	(s1), (s2), (s3), (s4), (d)	• Writes devices of the host station to the CPU shared memory of the host station CPU module.	⎍	Page 831 Writing data to host CPU shared memory(S(P)_TO)
	SP_TO			↑	
	TO	(s), n1, n2, n3	• Writes devices of the host station to the CPU shared memory of the host station CPU module.	⎍	
	TOP			↑	
	DTO	(s), n1, n2, n3	• Writes devices of the host station to the CPU shared memory of the host station CPU module in units of 32-bit data.	⎍	
	DTOP			↑	

Reading data from other CPU shared memory

Category	Instruction name	Argument	Description	Executing condition	Page
Data read from other CPU shared memory	FROM	n1, n2, n3, (d)	• Reads devices from the CPU shared memory of another station CPU module to the host station.	⎍	Page 839 Reading data from other CPU shared memory
	FROMP			↑	
	DFRO	n1, n2, n3, (d)	• Reads devices from the CPU shared memory of another station CPU module to the host station in units of 32-bit data.	⎍	
	DFROP			↑	

2.8 Multiple CPU High Speed Transmission Dedicated Instructions

Multiple CPU high speed transmission dedicated instructions

Category	Instruction name	Argument	Description	Executing condition	Page
Device data write to other CPU	D_DDWR	n, (s1), (s2), (d1), (d2)	• Writes data stored in the device (s2) and the following devices in the host CPU at the time of multiple CPU system configuration, to the device (d1) and the following devices specified in the other CPU (n) for the number of points specified for (s1)+1.	⎍	Page 855 Writing Device Data to Other CPUs
	DP_DDWR	n, (s1), (s2), (d1), (d2)		↑	
Device data read from other CPU	D_DDRD	n, (s1), (s2), (d1), (d2)	• Stores data read from the device (s2) and the following devices specified in the other CPU (n) for the number of points specified for (s1)+1, to the device (d1) and the following devices specified in the host CPU at the time of multiple CPU system configuration.	⎍	Page 859 Reading Device Data from Other CPUs
	DP_DDRD	n, (s1), (s2), (d1), (d2)		↑	

2.9 Redundant System Instruction (For Redundant CPU)

Redundant system instruction (for redundant CPU)

Category	Instruction name	Argument	Description	Executing condition	Page
System switching	SP_CONTSW	(s), (d)	• Switches between the control system and standby system at the END processing of the scan executed with the SP_CONTSW instruction.	↑	Page 863 System Switching

3 CONFIGURATION OF INSTRUCTIONS

3.1 Configuration of Instructions

The instructions available for CPU modules can be divided into an instruction name and arguments.

The application of an instruction name and arguments are as follows:

- Instruction name: Indicates the function of the instruction.
- Argument: Indicates the I/O data used in the instruction.

Arguments are classified into source data, destination data, number of devices, executing condition, and execution status.

Source (s)

A source is data used in an operation.

The following source types are available depending on the device specified in an instruction.

Type	Description
Constant	Specifies a numeric value used in an operation. Constants are set during programming so that they cannot be changed while the program is being executed. Perform index setting when using them as variable data.
Bit device and word device	Specifies the device in which the data used in the operation are stored. Data must be stored to the specified device before executing the operation. By changing the data to be stored to the specified device while a program is being executed, the data used in the instruction can be changed.

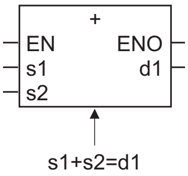
Contacts cannot be directly input to sources that use bit devices.

Destination (d)

Data after the operation are stored to a destination. Some instructions require to store data used in the operation to the destination before the operation.

Ex.

Addition instruction of 16-bit BIN data



Set a device in which data are to be stored to a destination.

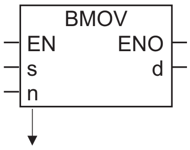
Coils cannot be directly connected to destinations which store bit devices.

Number of devices and number of transfers (n)

The number of devices and number of transfers used in an instruction that uses multiple devices are specified for n.

Ex.

Block data transfer instruction



Specifies the number of transfers used by a BMOV instruction

A value in the range of 0 to 32767 can be set for the number of devices and number of transfers. When the number of devices or number of transfers is 0, the instruction performs no processing.

Executing condition (EN)

An input variable EN inputs an executing condition of an instruction.

Execution status (ENO)

An output variable ENO outputs an execution status.

Point

For details of the instruction configurations of labels and structures etc., refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

3.2 Precautions on Programming

The following cases cause an operation error during the execution of basic instructions and application instructions in a CPU module.

- Errors described on each page of instruction explanation are occurred.
- No intelligent function module is mounted at the specified I/O number position when an intelligent function module device is used.
- The specified buffer memory address does not exist when an intelligent function module device is used.
- The corresponding network does not exist when a link device is used.
- No network module is mounted at the specified I/O number position when a link device is used.
- A CPU module is not mounted at the start I/O number position of the specified CPU module when a multiple CPU area device is used. (For Universal model QCPU (except for Q00UJCPU) only)
- The specified shared memory address does not exist when a multiple CPU area device is used. (For Universal model QCPU (except for Q00UJCPU) only)
- The setting that crosses internal user devices and extended data registers (D)/extended link registers (W) is set. (For Universal model QCPU (except for Q00UJCPU) and LCPU)

Point

The following are the results of the file register data write/read when the file register setting is not set, or when the file register setting is set, but a file register file does not exist.

- For High Performance model QCPU, Process CPU, and Redundant CPU
  An error does not occur even when writing/reading data to/from file registers. However, "0H" is stored when reading from file registers.
- For Universal model QCPU and LCPU
  The OPERATION ERROR (error code: 4101) occurs when writing/reading data to/from file registers. By setting the PLC parameter to not check the device range, an error is not detected. (Refer to Page 64 Device range check.)

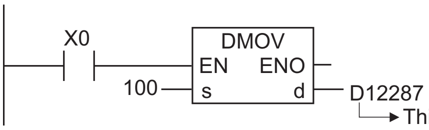
Device range check

The following explains how the range is checked for devices used in basic instructions and application instructions in a CPU module.

■Instructions that handle fixed-length devices (such as MOV or DMOV)

- For Basic model QCPU, High Performance model QCPU, Process CPU, and Redundant CPU

The device range is not checked. If the data exceed the corresponding device range, the data are written to another devices. [*1]For example, when the data registers are assigned in 12K points, an error does not occur even if the data exceed D12287.



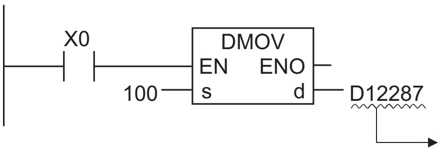
This specifies D12287 and D12288 as the target devices for executing the DMOV instruction However, since D12288 does not exist, data in another device is corrupted.

The device range is not checked when the index setting is applied.
If the result of the index setting exceeds the corresponding device range, the data are written to another devices.[*1]

*1 For the assignment order of internal user devices, refer to ☞ Page 66 Character string data.

- For Universal model QCPU and LCPU

The device range is checked. When the data exceed the corresponding device range, an operation error occurs. For example, when data registers are assigned in 12K points, an error occurs if the data exceed D12287.



This specifies D12287 and D12288 as the target devices for executing the DMOV instruction. However, since D12288 does not exist, an operation error occurs.

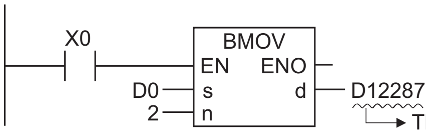
The device range is checked even when the index setting is applied.
By changing the settings of the PLC parameter, the device range is not checked.[*2]

*2 For changing the setting of device range check, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module to be used.

■Instructions that handle variable-length devices (such as BMOV or FMOV that specifies the number of transfers)

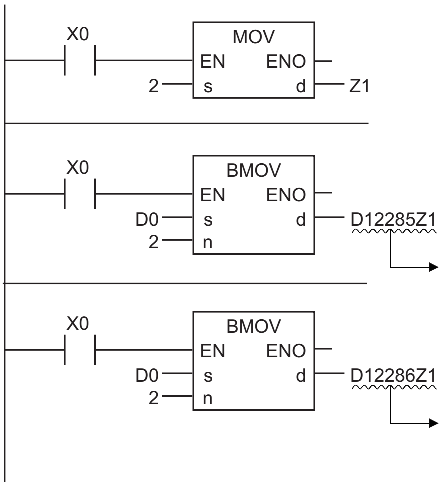
- For Basic model QCPU, High Performance model QCPU, Process CPU, and Redundant CPU

The device range is checked. When the data exceed the corresponding device range, an operation error occurs. For example, when data registers are assigned in 12K points, an error occurs if the data exceed D12287.



This specifies D12287 and D12288 as the target devices for executing the BMOV instruction. However, since D12288 does not exist, an operation error occurs.

The device range is checked even when the index setting is applied. However, when the start number of the device exceeds the corresponding device range with the index setting, an error occurs.

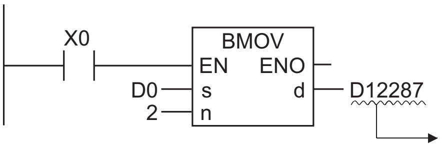


This specifies D12287 and D12288 as the target devices for executing the BMOV instruction. However, since D12288 does not exist, an operation error occurs.

This specifies D12288 as the start number of the device. However, since it is outside of the device range, an operation error occurs.

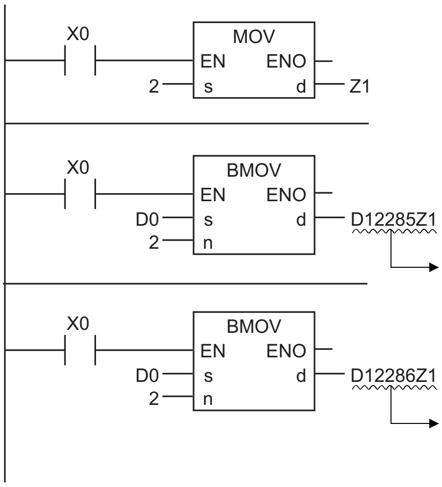
- For Universal model QCPU and LCPU

The device range is checked. When the data exceed the corresponding device range, an operation error occurs. For example, when data registers are assigned in 12K points, an error occurs if the data exceed D12287.



This specifies D12287 and D12288 as the target devices for executing the BMOV instruction. However, since D12288 does not exist, an operation error occurs.

The device range is checked even when the index setting is applied.
Also, when the start number of the device exceeds the corresponding device range with the index setting, an error occurs.



This specifies D12287 and D12288 as the target devices for executing the BMOV instruction. However, since D12288 does not exist, an operation error occurs.

This specifies D12288 as the start number of the device. However, since it is outside of the device range, an operation error occurs.

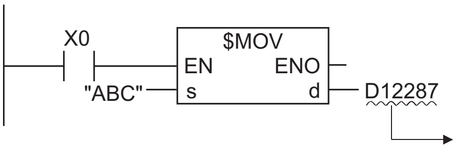
By changing the settings of the PLC parameter, the device range is not checked.[*1]

[*1] For changing the setting of device range check, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module to be used.

■Character string data

- For Basic model QCPU, High Performance model QCPU, Process CPU, and Redundant CPU

Since character string data are treated as variable-length data without exception, the device range is checked.
When the data exceed the corresponding device range, an operation error occurs.
For example, when data registers are assigned in 12K points, an error occurs if the data exceed D12287.

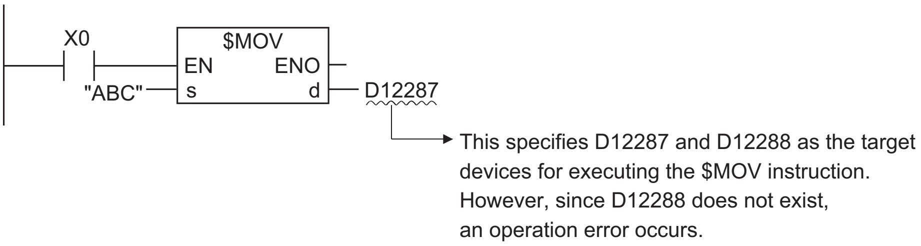


This specifies D12287 and D12288 as the target devices for executing the $MOV instruction. However, since D12288 does not exist, an operation error occurs.

When the index modification is executed, the device range is checked.
However, when the index modification is executed, an error does not occur if the start device number exceeds the corresponding device range. Other devices are accessed.

- For Universal model QCPU and LCPU

Since character string data are treated as variable-length data without exception, the device range is checked.
When the data exceed the corresponding device range, an operation error occurs.
For example, when data registers are assigned in 12K points, an error occurs if the data exceed D12287.



This specifies D12287 and D12288 as the target devices for executing the $MOV instruction. However, since D12288 does not exist, an operation error occurs.

When the index modification is executed, the device range is checked. When the index modification is executed, an error occurs if the start device number exceeds the corresponding device range.
The PLC parameter setting enables the device range check not to be performed.[*1]

[*1] For the setting method which the device range check is not performed when the index modification is executed, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module used.

■Direct access output (DY)

When the index setting is applied to direct access output (DY), the device range is checked.

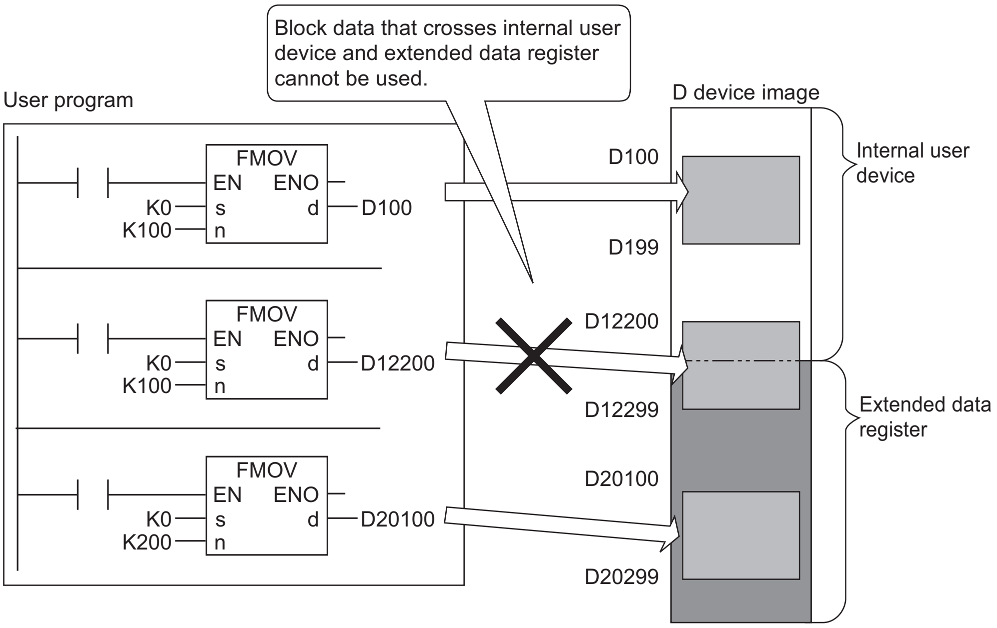
■Precautions on using extended data registers (D) and extended link registers (W)[*1]

An index setting that crosses internal user devices and extended data registers (D)/extended link registers (W) cannot be applied in the following specification methods as shown below. If it is specified, an OPERATION ERROR (Error code: 4101) occurs.

[*1] Universal model QCPU and LCPU other than Q00UJCPU

- Index setting
- Indirect specification
- Specification in the instruction that uses block data

Block data are the data described below.
- Data used for the instruction in which multiple words (such as FMOV, BMOV, BK+) are the operating target.
- Control data configured with more than two words and specified for SP_FWRITE, SP_FREAD.
- Data format data more than 32 bits. (BIN 32-bit, real number, indirect address of device)



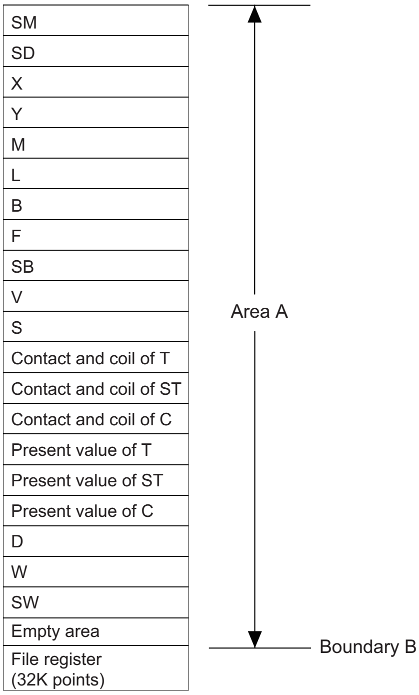
■Precautions when using Universal model QCPU/LCPU

For the Universal model QCPU and LCPU, an error occurs when the following access are executed with the instructions or data shown below. (Error code: 4101)

Instructions and data
• Instructions that handle fixed-length devices (such as MOV or DMOV) • Instructions that handle variable-length devices (such as BMOV or FMOV that specifies the number of transfers) • Character string data

Access method
(1) Data access that exceeds the device boundary by the index modification (range of area A)[*1] (2) Data access that exceeds the file register boundary by the index modification (3) Data access to file registers (R, ZR) when file register files are not set (4) Data access to file registers (R, ZR) that exceeds the range of file register files

*1 The following shows the assignment order of devices.



However, by setting the PLC parameter to not check the device range, an error is not detected if the above access are executed.

For the Universal model QCPU, the operation differs according to the serial number as shown by the following table.[*2]

Device range setting at index modification	First five digits of the serial number (Universal model QCPU)		LCPU
	'10021' or earlier	'10022' or later	
Applied	Errors are detected in the data access of 1) to 4)		
Not applied	Errors are detected in the data access of 2) to 4)	No error is detected	

*2 For the setting method which the device range check is not performed when the index modification is executed, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module used.

For the Universal model QCPU and LCPU, the index modification that crosses internal user devices (SW) and file registers (R) cannot be applied. (Error code: 4101)

Point

For the method for changing the assignment of internal user devices, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module used.

Device data check

The following explains how the data are checked for devices used in basic instructions and application instructions in a CPU module.

■BIN data

Even if the operation results in overflow or underflow, an error does not occur, and the carry flag (SM700) is not turned ON.

■BCD data

Checks whether each digit has a BCD value (0 to 9). If each digit has a value other than 0 to 9 (A to F), an operation error occurs.

Even if the operation results in overflow or underflow, an error does not occur, and the carry flag (SM700) is not turned ON.

■Floating-point data

If the operation results are as follows in an instruction that performs single-precision floating-point operation, an operation error occurs.

- 1.0×2^{-127} or less
- 1.0×2^{128} or more

If the operation results are as follows in an instruction that performs double-precision floating-point operation, an operation error occurs.

- 1.0×2^{-1023} or less
- 1.0×2^{1024} or more

■Character string data

Data check is not performed.

Buffer memory access

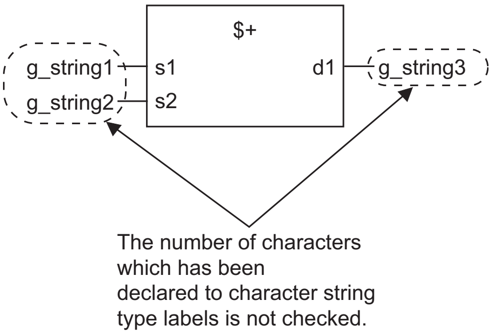
For accessing buffer memories, using instructions with intelligent function module devices (from Un\G0) is recommended.

Multiple CPU shared memory access

For accessing multiple CPU shared memories, using instructions with multiple CPU area devices (from U3En\G10000) is recommended.

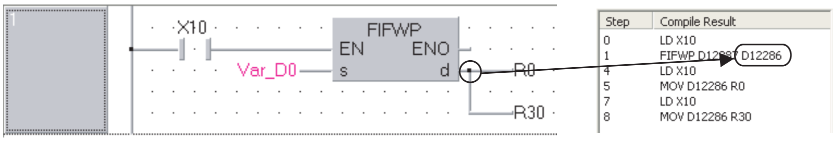
Number of characters in character string type label check

In the operation of character string type labels, the number of characters which has been declared to character string type labels is not checked. When using instructions/functions which process character strings, check not to exceed the number of characters which has been declared to character string type labels.


$+
g_string1
g_string2
s1
s2
d1
g_string3
The number of characters which has been declared to character string type labels is not checked.

Branch point of line from destination

When a line is branched, a temporary variable is created on the branch point. When a line which is from destination is branched as shown below, the operation result may not be output to the devices or labels of branch destination correctly since the operation result is stored on a temporary variable which is created on the branch point. For details of temporary variable, refer to GX Works2 Version 1 Operating Manual (Structured Project).



Step	Compile Result
0	LD X10
1	FIFWP D12287 D12286
4	LD X10
5	MOV D12286 R0
7	LD X10
8	MOV D12286 R30

The operation result is stored on a temporary variable.

3.3 Executing Conditions for Instructions

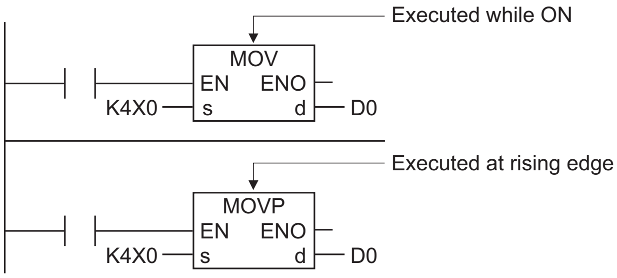
This section describes the types of executing condition for sequence instructions, basic instructions, and application instructions executed in a CPU module.

Type	Description	Example
Non-conditional execution	Instructions executed regardless of ON/OFF status of the device.	LD=
Executed while ON	Instructions executed while the input condition is ON.	MOV instruction, FROM instruction
Executed at rising edge	Instructions executed only at the rising edge (OFF to ON) of the input condition.	PLS instruction, MOVP instruction
Executed while OFF	Instructions executed while the input condition is OFF.	—
Executed at falling edge	Instructions executed only at the falling edge (ON to OFF) of the input condition.	PLF instruction

For basic instructions and application instructions that are equivalent to coils, when both 'Executed while ON' and 'Executed at rising edge' are applicable, 'P' is appended to an instruction name to distinguish executing conditions.

- An instruction executed while ON: Instructions name
- An instruction executed at rising edge: Instructions name P

The following shows how the executing conditions of 'Executed while ON' and 'Executed at rising edge' are specified for the MOV(P) instruction.



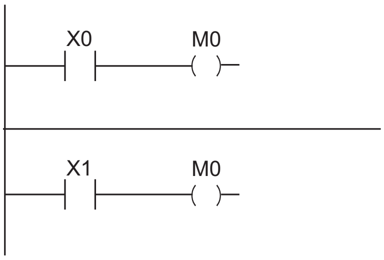
3.4 Operation of OUT Instructions, SET/RST Instructions, or PLS/PLF Instructions Using Same Device

This section describes the operation when the OUT instructions, SET/RST instructions, or PLS/PLF instructions that use a same device are executed multiple times in one scan.

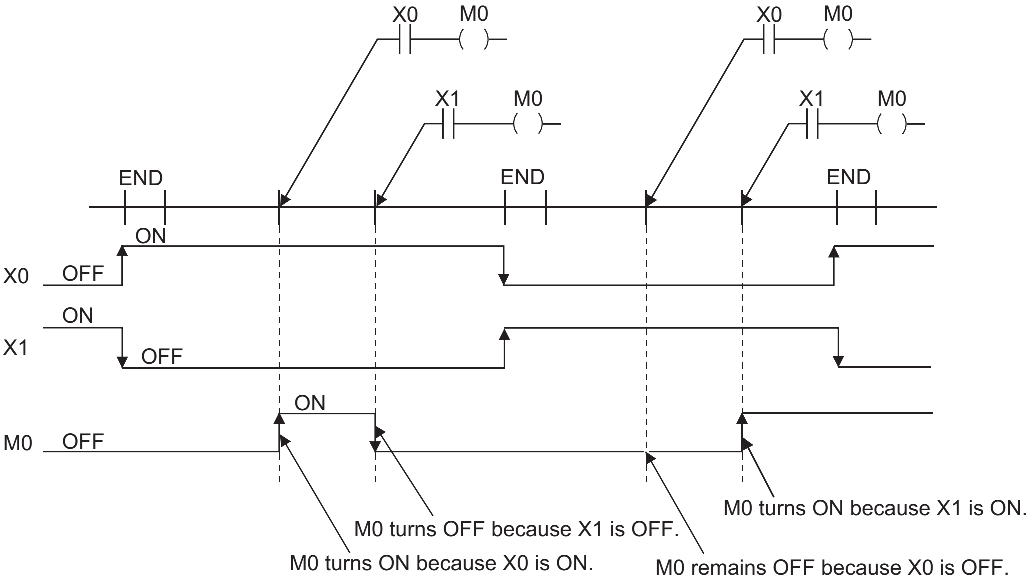
OUT instructions that use a same device

Avoid executing the OUT instructions that use a same device multiple times during one scan. If the OUT instructions that use a same device are executed multiple times during one scan, the specified device is turned ON/OFF according to the operation result obtained by each execution of the OUT instruction. Since the ON/OFF status of the specified device is determined by each execution of the OUT instruction, turning ON/OFF may be repeated during one scan. The following figures show the operation example of the ladder that turns ON/OFF the same internal relay (M0) by the inputs X0 and X1.

[Ladder]



[Timing chart]

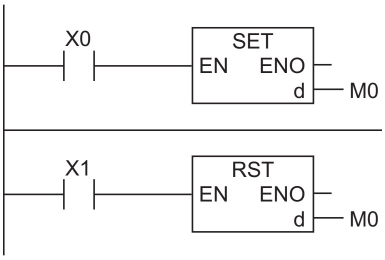

X0
M0
X1
M0
END
END
END
X0
OFF
ON
X1
ON
OFF
M0
OFF
ON
M0 turns ON because X1 is ON.
M0 turns OFF because X1 is OFF.
M0 turns ON because X0 is ON.
M0 remains OFF because X0 is OFF.

For refresh type CPU modules, the ON/OFF status of the OUT instruction executed last during one scan is output when output (Y) is specified for the OUT instruction.

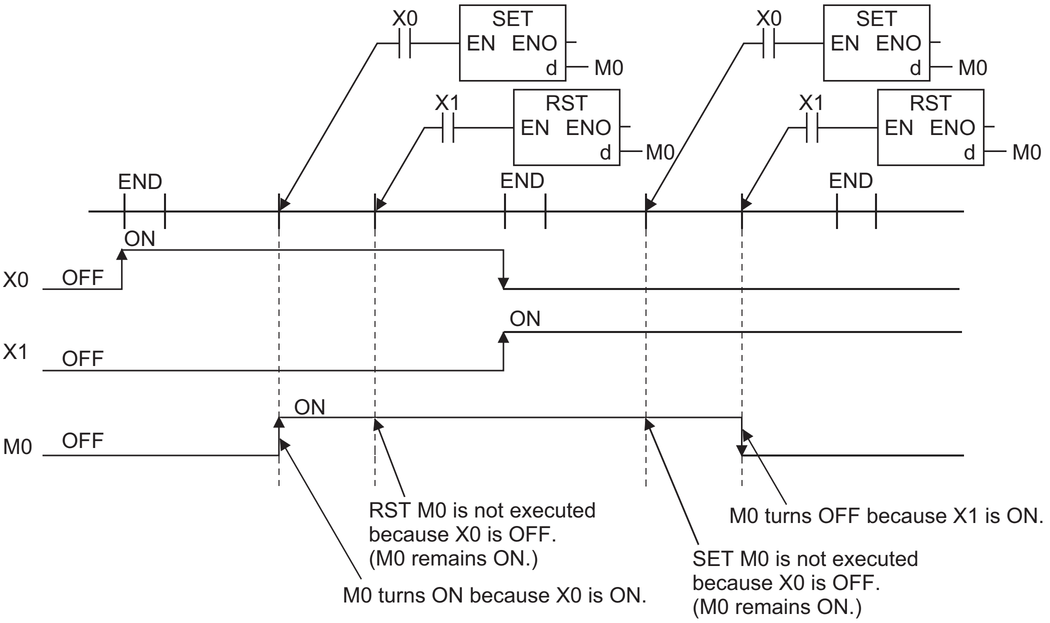
SET/RST instructions that use a same device

- The SET instruction turns ON the specified device when the execution command is ON, while it performs no operation when the execution command is OFF. Thus, if the SET instructions that use a same device are executed multiple times during one scan, the specified device is turned ON when any of the execution commands is turned ON.
- The RST instruction turns OFF the specified device when the execution command is ON, while it performs no operation when the execution command is OFF. Thus, if the RST instructions that use a same device is executed multiple times during one scan, the specified device is turned OFF when any of the execution commands is turned ON.
- If the SET and RST instructions use the same device in one scan, the SET instruction turns ON the specified device when the execution command is ON, while the RST instruction turns OFF the specified device when the execution command is ON. When the execution command of the SET and RST instructions is OFF, the ON/OFF status of the specified device does not change.

[Ladder]



[Timing chart]

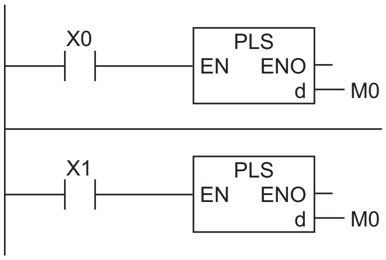

X0
SET
EN ENO
d
M0
X1
RST
EN ENO
d
M0
END
END
END
ON
X0
OFF
ON
X1
OFF
ON
M0
OFF
RST M0 is not executed because X0 is OFF. (M0 remains ON.)
M0 turns ON because X0 is ON.
M0 turns OFF because X1 is ON.
SET M0 is not executed because X0 is OFF. (M0 remains ON.)

For refresh type CPU modules, the ON/OFF status of the SET/RST instruction executed last during one scan is output when output (Y) is specified for the SET/RST instruction.

PLS instructions that use a same device

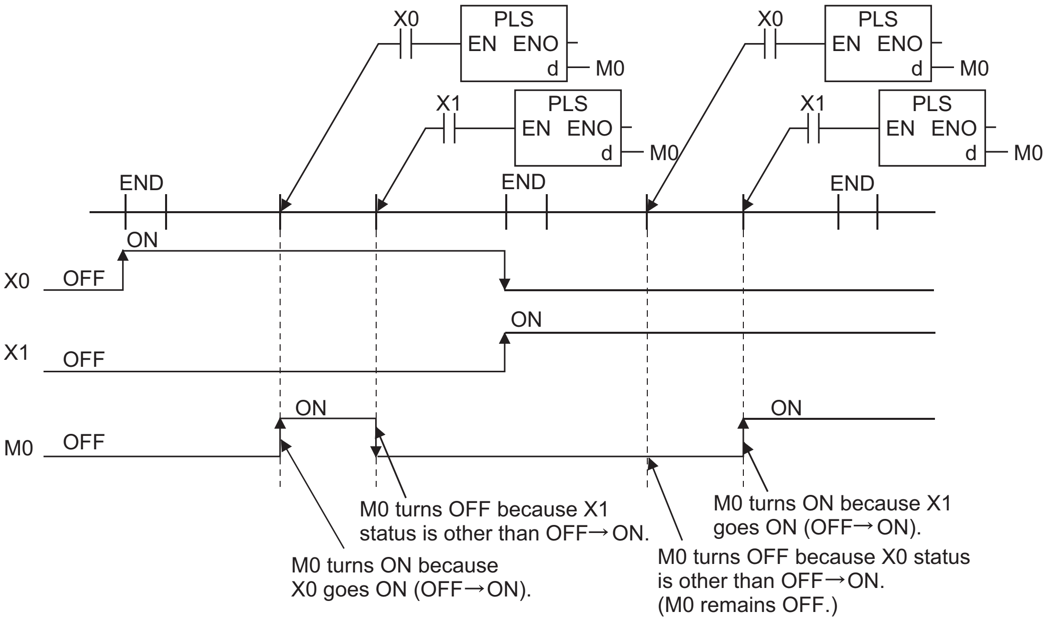
The PLS instruction turns ON the specified device when the execution command turns from OFF to ON. When the status is other than turning from OFF to ON, (OFF to OFF, ON to ON or ON to OFF), the PLS instruction turns OFF the specified device. If the PLS instructions that use a same device are executed multiple times during one scan, the PLS instructions turn ON the specified device when the execution command turns from OFF to ON. In the status other than turning from OFF to ON, they turn OFF the specified device. Thus, when the PLS instructions that use a same device are executed multiple times during one scan, the devices turned ON in the PLS instructions may not be turned ON for one scan.

[Ladder]



[Timing chart]

- The ON/OFF timing differs between X0 and X1 (The specified device is not turned ON for one scan)



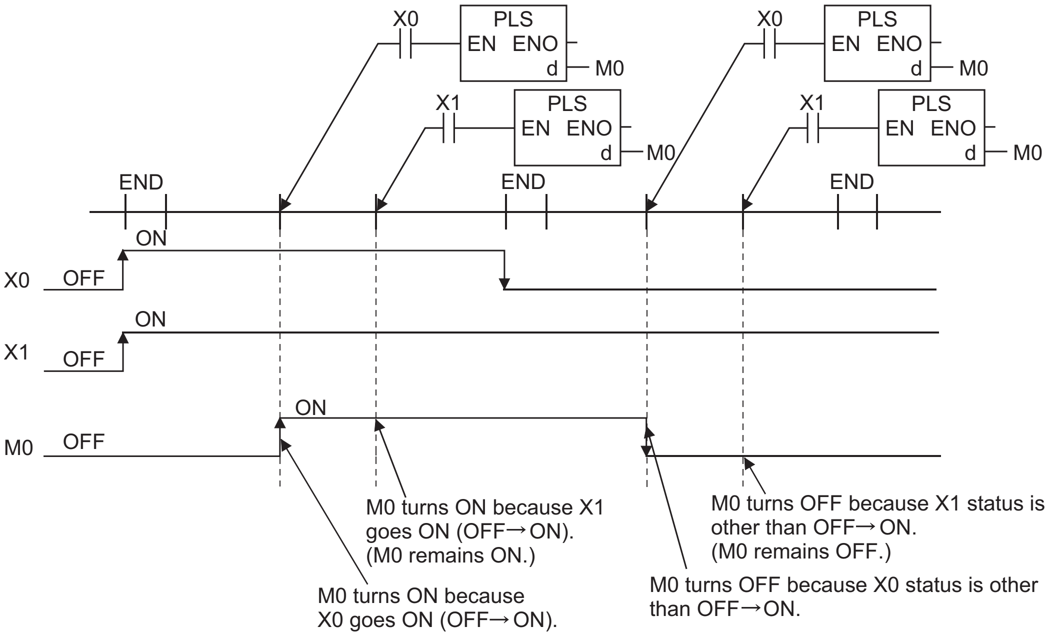
M0 turns OFF because X1 status is other than OFF→ON.

M0 turns ON because X0 goes ON (OFF→ON).

M0 turns ON because X1 goes ON (OFF→ON).

M0 turns OFF because X0 status is other than OFF→ON. (M0 remains OFF.)

- The timings of turning from OFF to ON between X0 and X1 are same



M0 turns ON because X1 goes ON (OFF→ON). (M0 remains ON.)

M0 turns ON because X0 goes ON (OFF→ON).

M0 turns OFF because X1 status is other than OFF→ON. (M0 remains OFF.)

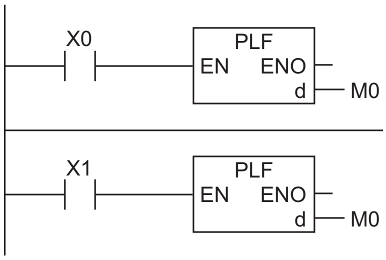
M0 turns OFF because X0 status is other than OFF→ON.

For refresh type CPU modules, the ON/OFF status of the PLS instruction executed last during one scan is output when output (Y) is specified for the PLS instruction.

PLF instructions that use a same device

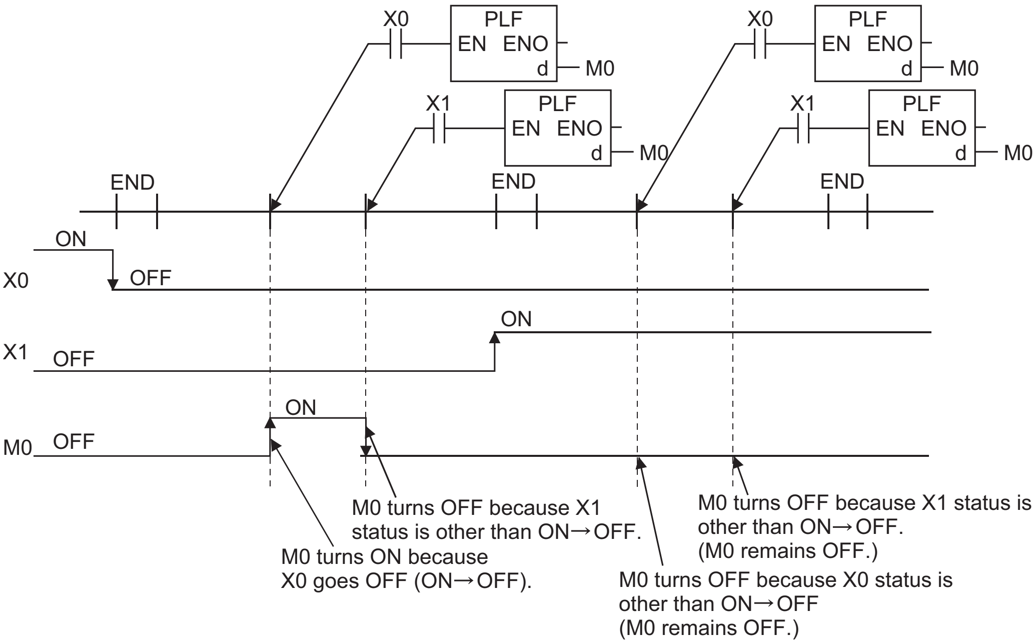
The PLF instruction turns ON the specified device when the execution command turns from ON to OFF. If the status is other than turning from ON to OFF (OFF to OFF, OFF to ON or ON to OFF), the PLF instruction turns OFF the specified device. If the PLF instructions that use a same device are executed multiple times during one scan, the PLF instructions turn ON the specified device when the execution command turns from ON to OFF. In the status other than turning from ON to OFF, they turn OFF the specified device. Thus, when the PLF instructions that use a same device are executed multiple times during one scan, the devices turned ON in the PLF instruction may not be turned ON for one scan.

[Ladder]



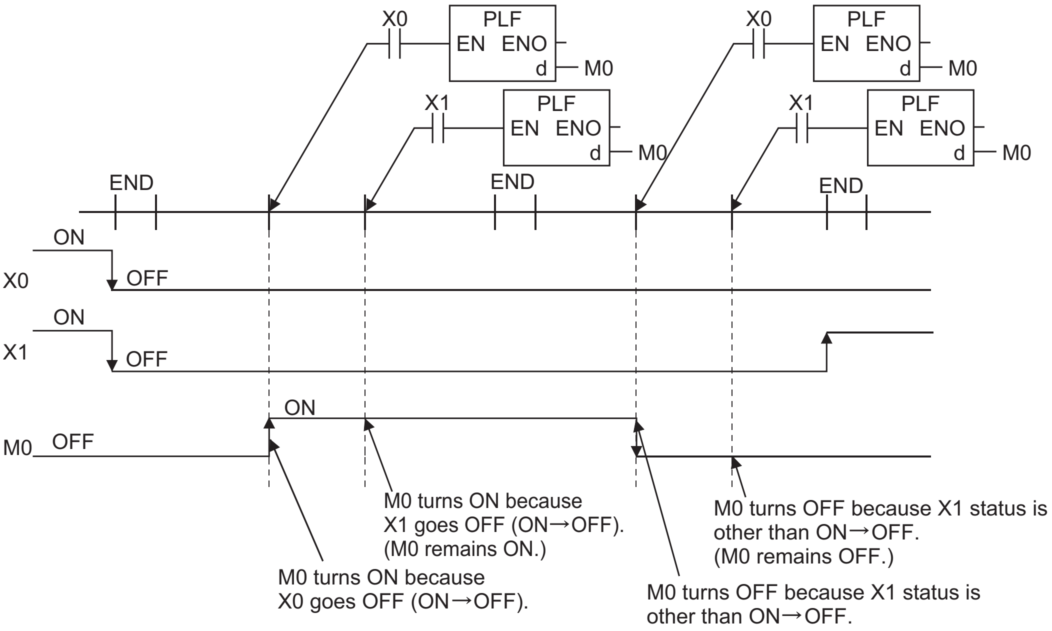
[Timing chart]

- The ON/OFF timing differs between X0 and X1 (The specified device is not turned ON for one scan)



X0
PLF
EN ENO
d
M0
X1
PLF
EN ENO
d
M0
X0
PLF
EN ENO
d
M0
X1
PLF
EN ENO
d
M0
END
END
END
X0
ON
OFF
X1
OFF
ON
M0
OFF
ON
M0 turns OFF because X1 status is other than ON→OFF.
M0 turns ON because X0 goes OFF (ON→OFF).
M0 turns OFF because X1 status is other than ON→OFF. (M0 remains OFF.)
M0 turns OFF because X0 status is other than ON→OFF (M0 remains OFF.)

- The timings of turning from ON to OFF between X0 and X1 are same



X0
PLF
EN ENO
d
M0
X1
PLF
EN ENO
d
M0
X0
PLF
EN ENO
d
M0
X1
PLF
EN ENO
d
M0
END
END
END
X0
ON
OFF
X1
ON
OFF
M0
OFF
ON
M0 turns ON because X1 goes OFF (ON→OFF). (M0 remains ON.)
M0 turns ON because X0 goes OFF (ON→OFF).
M0 turns OFF because X1 status is other than ON→OFF. (M0 remains OFF.)
M0 turns OFF because X1 status is other than ON→OFF.

For refresh type CPU modules, the ON/OFF status of the PLF instruction executed last during one scan is output when output (Y) is specified for the PLF instruction.

3.5 Precautions on Using File Registers

This section describes the precautions on using file registers in QCPU (Q mode) and LCPU.

CPU module that does not support file registers

Q00JCPU and Q00UJCPU do not support file registers.
To use file registers, use CPU modules other than Q00JCPU and Q00UJCPU.

Setting file registers to be used

To use file registers, they need to be set in the PLC parameter or the QDRSET instruction. (With Q00CPU, Q01CPU or LCPU, they do not need to be set because the PLC parameter is set to 'Use file register'. The QDRSET instruction cannot be used for LCPU.) To use file registers, they need to be set in the PLC parameter or the QDRSET instruction. If the file registers are not set, normal operations cannot be performed in the instructions that use the file registers.

Point

Even when file registers are not set in the PLC parameter, a program that uses file registers can be created.
For CPU modules other than Universal model QCPU and LCPU, an error does not occur when that program is written to the CPU module.
Note that the data cannot be written/read correctly to/from file registers.
For Universal model QCPU and LCPU, an error occurs if the program that uses file registers is executed.

Reserving the file register area

■For Basic model QCPU

The file register area is reserved in the standard RAM in advance. Users do not need to reserve the file register area.

■For High Performance model QCPU, Process CPU, Redundant CPU, and Universal model QCPU (except for High-speed Universal model QCPU and Universal model Process CPU)

To use file registers, register them to the standard RAM/memory card and reserve the file register area.

■For High-speed Universal model QCPU, Universal model Process CPU, and LCPU

To use file registers, register them to the standard RAM and reserve the file register area.

Point

For the method for setting file registers and the memory which can be used the file registers of each CPU module, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module used.

Specifying the file register number exceeding the registered number

■For Basic model QCPU, High Performance model QCPU, Process CPU, and Redundant CPU

An error does not occur when data are written/read to/from the file register numbers that are greater than the registered number of points. Note that the data cannot be written/read correctly to/from file registers.

■For Universal model QCPU and LCPU

When data are written/read to/from the file register numbers that are greater than the registered number of points, an error occurs. (Error code: 4101)

Method for specifying file registers

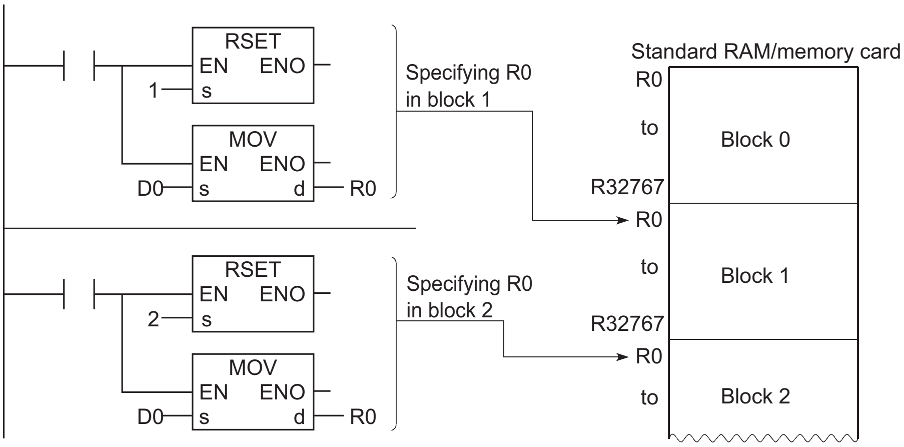
The methods for specifying file registers are the block switching method and the serial number access method.

■Block switching method

The block switching method specifies file register points being used in units of 32K points (one block).

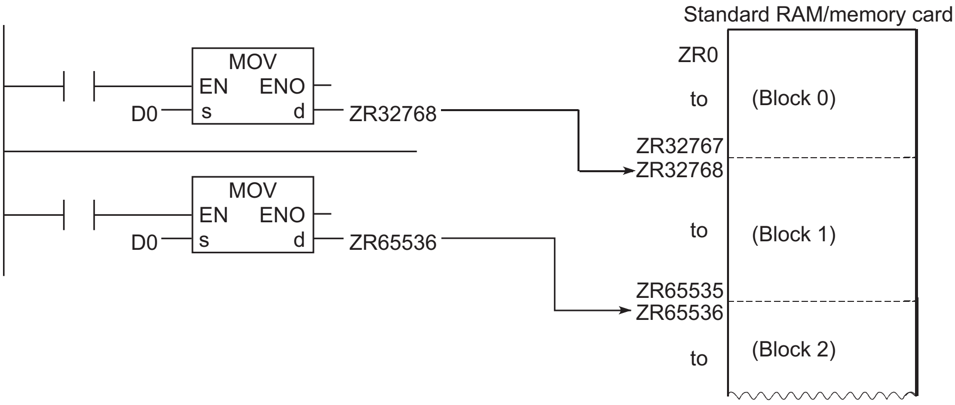
File registers exceeding 32K points is specified by switching block numbers of the file registers used in the RSET instruction.

File register points are specified in the range from R0 to R32767 for each block.



■Serial number access method

The serial number access method specifies file registers exceeding 32K points with serial device numbers. File registers of multiple blocks can be used as continuous file registers. Use 'ZR' as a device name.



Settings and restrictions when specifying file registers to refresh devices

■Setting refresh devices

Refresh devices can be set by the following settings.

- Refresh settings of CC-Link IE module (Not available for LCPU)
- Refresh settings of CC-Link IE Field Network (Not available for Basic model QCPU, High Performance model QCPU, Process CPU, Redundant CPU, Universal model QCPU with a serial number whose first five digits are "12011" or earlier, and LCPU with a serial number whose first five digits are "13011" or earlier.)
- Refresh settings of MELSECNET/H (Not available for LCPU)
- Refresh settings of CC-Link
- Auto refresh settings of the intelligent function module
- Auto refresh settings of the multiple CPU system (Not available for LCPU)

■Restrictions

The following are the restrictions when specifying file registers to refresh devices.

- For QCPU (Q mode), the refresh cannot be performed correctly if the file register which has the same name as the program is specified by the PLC parameter. When the file register whose file name is same as the program name is used, the refresh is performed to the file register whose file name is same as the program name that is set at the last number in the program settings. To read or write the refresh data, use the QDRSET instruction to switch to the corresponding file register.
- The refresh cannot be performed correctly if the file name of file register or drive number is changed by using the QDRSET instruction. (The QDRSET instruction cannot be used for LCPU.) When the file name or drive number is changed on QDRSET instruction, the link refresh is performed to the set file when the END processing is carried out. To read or write the refresh data, specify the set file at the END processing. Note that If the drive number is changed by using the QDRSET instruction when "ZR" is specified for the device in a CPU module other than Universal model QCPU, an error (LINK PARA ERROR (3101)) occurs. (If "R" is specified for the device, an error does not occur.)
- When a block number is selected by using the RSET instruction, the refresh is performed to the file register (R) in the selected block number. When a block number is switched on RSET instruction, the refresh is performed to the file register (R) in the block number when the END processing is carried out. To read or write the refresh data, specify the file register of the block number at the END processing.

Precautions on using file registers in the flash memory

This section describes the precautions on the flash memory that can use file registers.

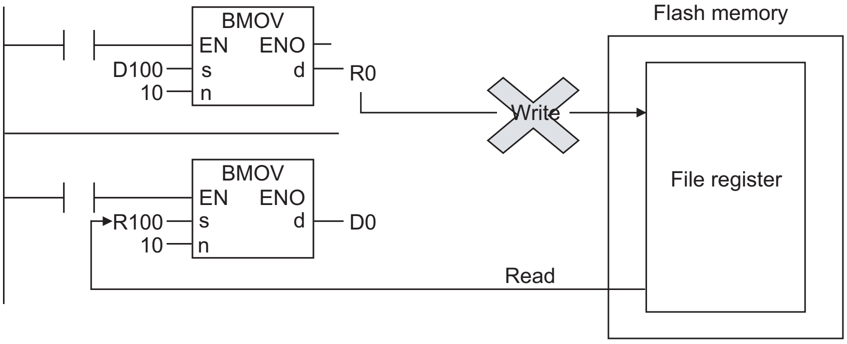
■Applicable flash memory

- Flash card

■Precautions

File registers in a flash memory can be only read in sequence programs.

(File registers cannot be written to a flash memory in sequence programs.)

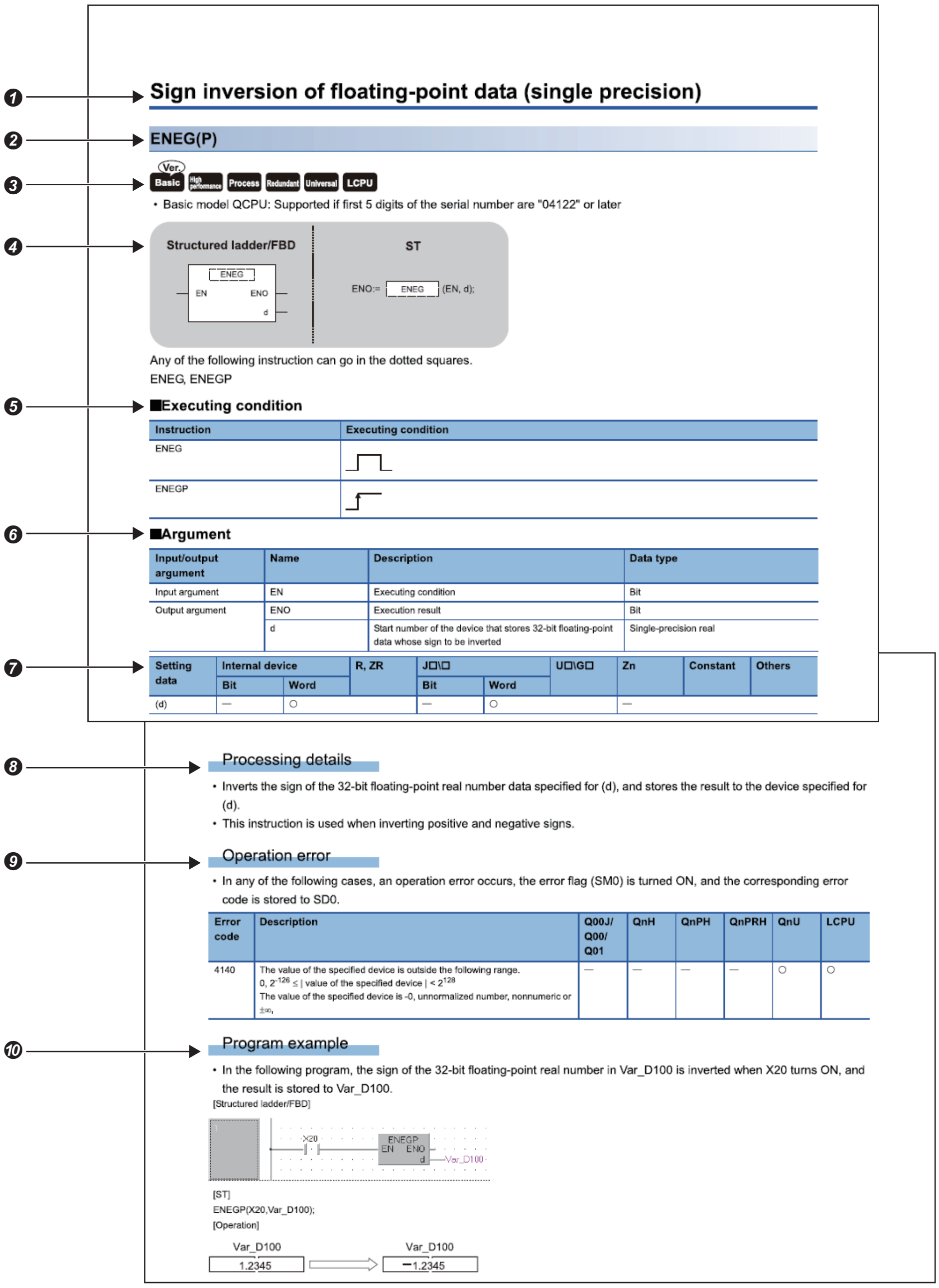


When using a flash memory for file registers, write data to a flash memory in advance.

Use the programming tool to write data to a flash card.

4 HOW TO READ INSTRUCTIONS

Chapter 5 and after provide detailed explanation on each instruction in the layout as shown below.



Sign inversion of floating-point data (single precision)

ENEG(P)

Ver. Basic High performance Process Redundant Universal LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

Structured ladder/FBD

ENEG
EN ENO
d

ST

ENO:= ENEG (EN, d);

Any of the following instruction can go in the dotted squares.

ENEG, ENEGP

■Executing condition

Instruction	Executing condition
ENEG	⎍
ENEGP	↑

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores 32-bit floating-point data whose sign to be inverted	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(d)	—	○		—	○		—		

Processing details

- Inverts the sign of the 32-bit floating-point real number data specified for (d), and stores the result to the device specified for (d).
- This instruction is used when inverting positive and negative signs.

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-126} \leq$ | value of the specified device | $< 2^{128}$ The value of the specified device is -0, unnormalized number, nonnumeric or ±∞,	—	—	—	—	○	○

Program example

- In the following program, the sign of the 32-bit floating-point real number in Var_D100 is inverted when X20 turns ON, and the result is stored to Var_D100.

[Structured ladder/FBD]

X20
ENEGP
EN ENO
d
Var_D100

[ST]

ENEGP(X20,Var_D100);

[Operation]

Var_D100 1.2345 ⇒ Var_D100 −1.2345

❶Indicates an outline of an instruction.

❷Indicates an instruction to be explained.

❸Indicates the CPU modules that can use the instruction.

Icon						Description
Basic model QCPU	High Performance model QCPU	Process CPU	Redundant CPU	Universal model QCPU	LCPU	
Basic	High performance	Process	Redundant	Universal	LCPU	Icons without any marks indicate that the CPU module can use the corresponding instructions.
Ver. Basic	Ver. High performance	Ver. Process	Ver. Redundant	Ver. Universal	Ver. LCPU	Icons with a "Ver." mark indicate that the CPU module can use the corresponding instructions under certain restrictions. related to function version and software version.
✕ Basic	✕ High performance	✕ Process	✕ Redundant	✕ Universal	✕ LCPU	Icons with an "×" mark indicate that the CPU module cannot use the corresponding instructions.

❹Indicates the description format of the instruction in the structured ladder/FBD/ST language.

❺Indicates the instruction name and executing condition of the instruction.

Executing condition	Non-conditional execution	Executed at ON	Executed on the rising edge	Executed at OFF	Executed on the falling edge
Description on corresponding pages	Non-conditional execution	⎍	↑	⊔	↓

❻Indicates the names of input and output arguments, and the data type of each argument. For details of each data type, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

❼Devices that can be used in the instruction are marked with ○.

The following table shows applicable classification for usable devices.

Device classification	Internal device (system, user)		File register R, ZR	Link direct devices J□\□*4*6		Intelligent function module U□\G□	Index register Zn	Constant*5	Others *5
	Bit	Word		Bit	Word				
Usable device *1	X, Y, M, L, SM, F, B, SB, FX*2, FY*2	T*3, ST*3, C*3, D, W, SD, SW, FD*2	R, ZR	J□\X J□\Y J□\B J□\SB	J□\W J□\SW	U□\G□	Z	K, H, E, $	P, I, J, U, DX, DY, N, B, L, TR, BL\S, V

*1 For description of each device, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module used.

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

*3 T, ST or C can be used in word data only (they cannot be used in bit data) when they are used in the instructions other than the following instructions.
[Usable instructions in bit data]
LD, LDI, AND, ANI, OR, ORI, LDP, LDF, ANDP, ANDF, ORP, ORF, LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI, OUT, RST, and BKRST

*4 These devices can be used in CC-Link IE, MELSECNET/H, and MELSECNET/10.

*5 The Constant and Others columns describe settable devices.

*6 Link direct devices (J□\□) cannot be used for LCPU.

❽Indicates the processing performed by the instruction.

❾Indicates whether to exist the related error. When an error exists, conditions that cause an error are described.

❿Indicates program examples in the structured ladder/FBD/ST language.

5 SEQUENCE INSTRUCTIONS

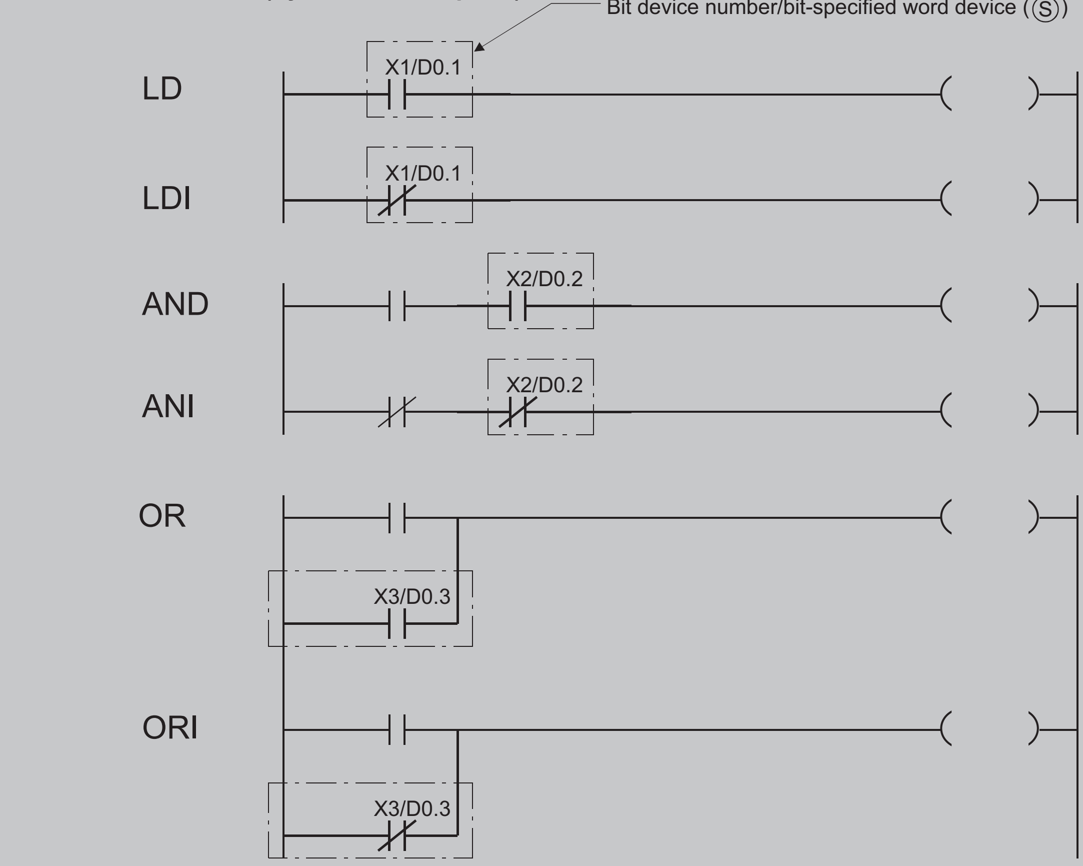
5.1 Contact Instructions

Operation start, series connection, parallel connection

LD, LDI, AND, ANI, OR, ORI

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD (Symbolic description)



■Executing condition

Instruction	Executing condition
LD, LDI, AND, ANI, OR, ORI	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	s	Devices used as contacts	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				DX
(s)	○						—		○

Point

When BL, S, TR, BL\S, or BL\TR is used, refer to SFC control instructions of the MELSEC-Q/L/QnA Programming Manual (SFC).

Processing details

■LD, LDI

- LD is the normally open operation start instruction, and LDI is the normally closed operation start instruction. They read ON/OFF information from the specified device*1, and use that as the operation result.

*1 For a bit-specified word device, the device turns ON or OFF depending on the 1/0 status of the specified bit.

■AND, ANI

- AND is the normally open series connection instruction, and ANI is the normally closed series connection instruction. They read the ON/OFF data of the specified bit device*1, perform an AND operation on that data and the operation result to that point, and take this value as the operation result.

*1 For a bit-specified word device, the device turns ON or OFF depending on the 1/0 status of the specified bit.

■OR, ORI

- OR is the normally open single parallel connection instruction, and ORI is the normally closed single parallel connection instruction. They read ON/OFF information from the specified device*1, and perform an OR operation with the operation results to that point, and use the resulting value as the operation result.

*1 For a bit-specified word device, the device turns ON or OFF depending on the 1/0 status of the specified bit.

Point

A bit of word device is specified in hexadecimal. Bit b11 of D0 would be D0.B.
For more information on bit specification of word device, refer to the following.
🕮 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

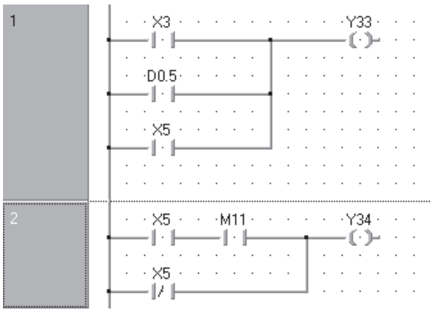
Operation error

- There is no operation error.

Program example

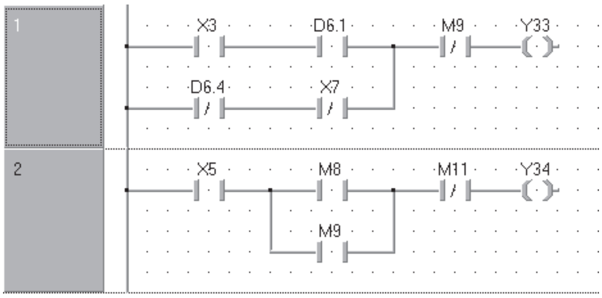
- The program using the LD, AND, OR, and ORI instructions.

[Structured ladder/FBD]



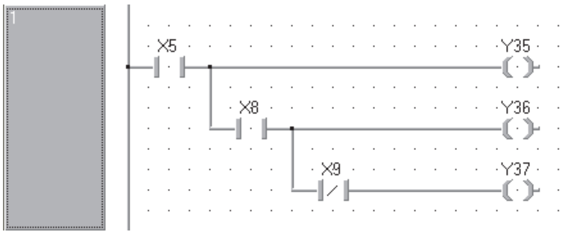
- The program connecting contacts using the ANB and ORB instructions.

[Structured ladder/FBD]



- The parallel program of the OUT instructions.

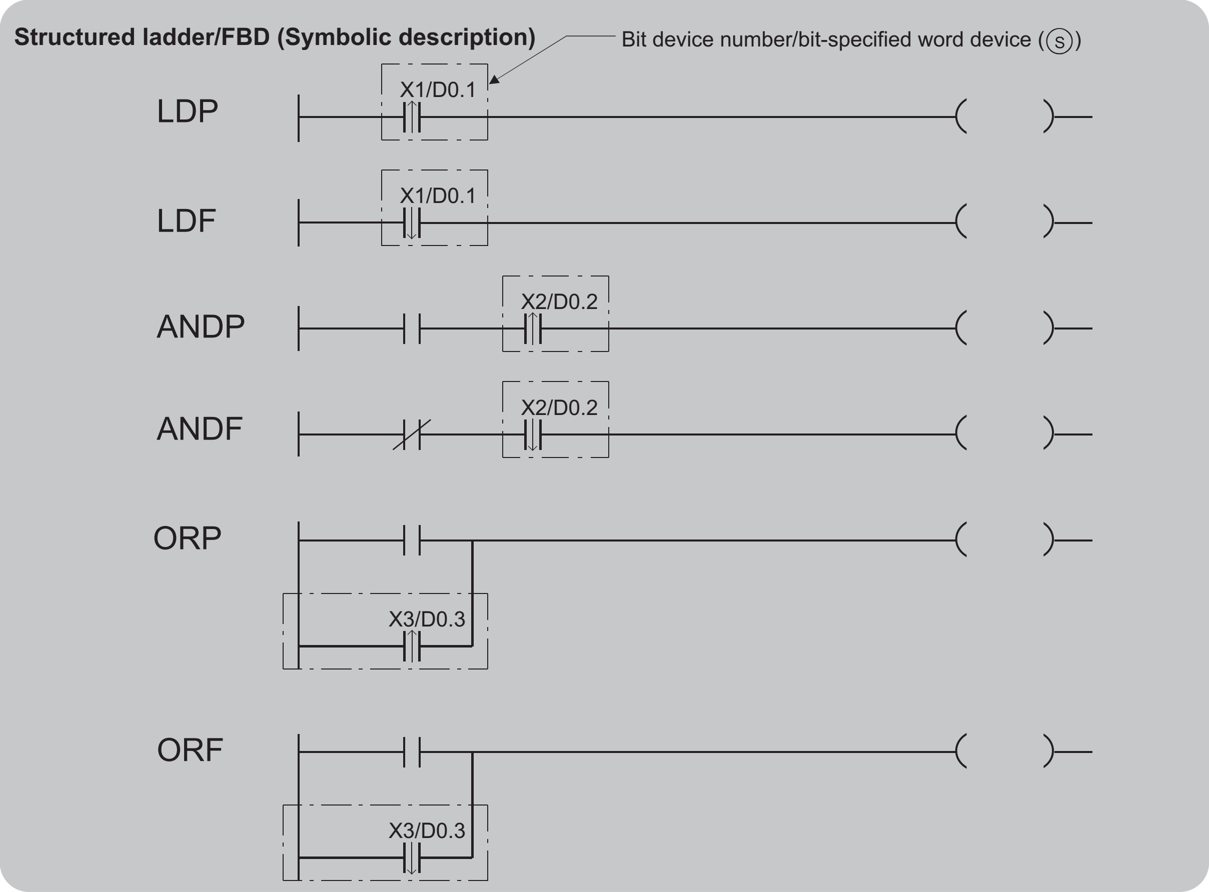
[Structured ladder/FBD]

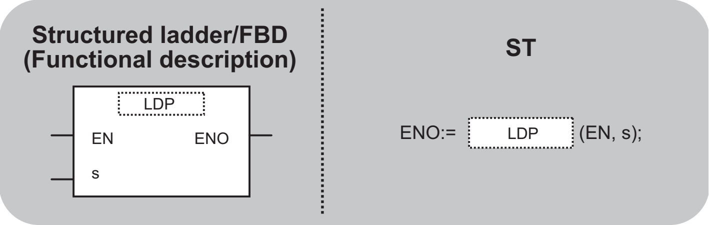


Edge operation start, edge series connection, edge parallel connection

LDP, LDF, ANDP, ANDF, ORP, ORF

Basic High performance Process Redundant Universal LCPU


Structured ladder/FBD (Symbolic description)
Bit device number/bit-specified word device (Ⓢ)
LDP
X1/D0.1
LDF
X1/D0.1
ANDP
X2/D0.2
ANDF
X2/D0.2
ORP
X3/D0.3
ORF
X3/D0.3



ENO:= LDP (EN, s);

Any of the following instruction can go in the dotted squares.
LDP, LDF, ANDP, ANDF, ORP, ORF

■Executing condition

Instruction	Executing condition
LDP, LDF, ANDP, ANDF, ORP, ORF	Non-conditional execution

■Argument

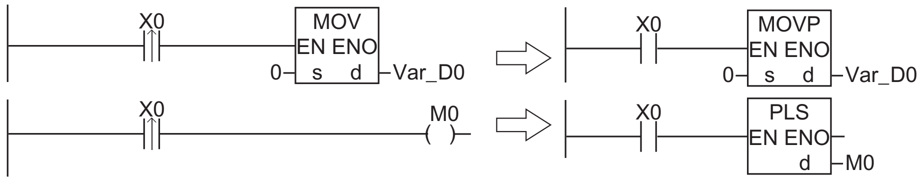
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Devices used as contacts	Bit
Output argument	ENO	Executing condition	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others DX
	Bit	Word		Bit	Word				
(s)	○						—		○

Processing details

■LDP, LDF

- LDP is the rising edge operation start instruction, and it is ON only at the rising edge of the specified bit device (when it turns from OFF to ON). For a bit-specified word device, it is ON only when the specified bit changes from 0 to 1. In cases where there is only an LDP instruction, it acts identically to instructions for the creation of a pulse that are executed while ON(□P).



- LDF is the falling edge operation start instruction, and it is ON only at the falling edge of the specified bit device (when it turns from ON to OFF). For a bit-specified word device, it is ON only when the specified bit changes from 1 to 0.

■ANDP, ANDF

- ANDP is a rising edge series connection instruction, and ANDF is a falling edge series connection instruction. They perform AND operations with the operation result to that point, and take the resulting value as the operation result. The ON/OFF data used by ANDP and ANDF are indicated in the table below.

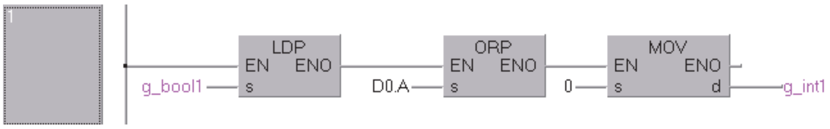
Device specified in ANDP or ANDF		ANDP status	ANDF status
Bit device	Bit-specified word device		
OFF→ON	0→1	ON	OFF
OFF	0	OFF	
ON	1		
ON→OFF	1→0		ON

■ORP, ORF

- ORP is a rising edge parallel connection instruction, and ORF is a falling edge serial connection instruction. They perform OR operations with the operation result to that point, and take the resulting value as the operation result. The ON/OFF data used by ORP and ORF are indicated in the table below.

Device specified in ORP or ORF		ORP status	ORF status
Bit device	Bit-specified word device		
OFF→ON	0→1	ON	OFF
OFF	0	OFF	
ON	1		
ON→OFF	1→0		ON

- The ORP or ORF instruction performs OR operation of operation result between (s) and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400. When using the ORP or ORF instruction, connect EN and ENO in series as shown below.



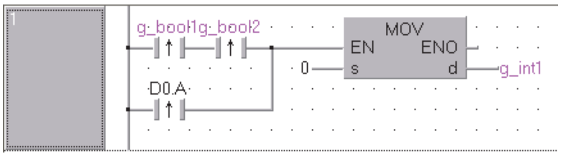
Operation error

- There is no operation error.

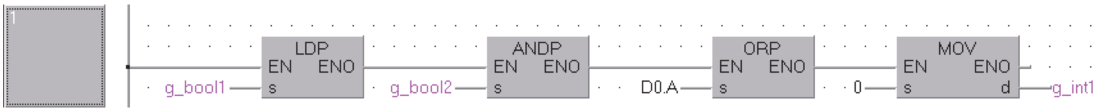
Program example

- In the following program, the MOV instruction is executed at the rising edge of g_bool1 and g_bool2, or b10 (bit 10) of data register D0.

[Structured ladder/FBD (Symbolic description)]



[Structured ladder/FBD (Symbolic description)]



[ST (When using IF syntax)]

```
IF ((LDP(TRUE,g_bool1) AND LDP(TRUE,g_bool2)) OR LDP(TRUE,D0.A)) THEN
    MOV(TRUE,0,g_int1);
END_IF;
```

[ST (When not using IF syntax)]

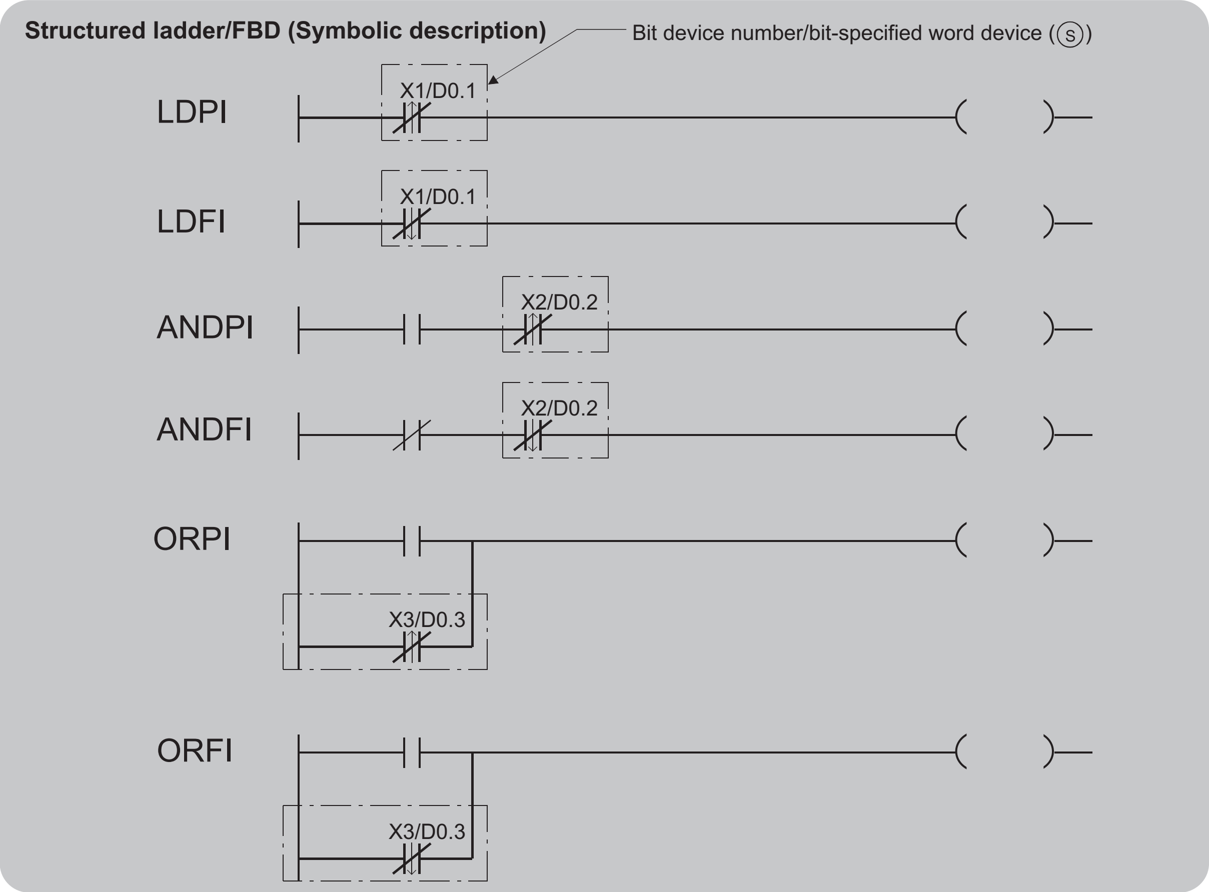
```
MOV(ORP(ANDP(LDP(TRUE,g_bool1),g_bool2),D0.A),0,g_int1);
```

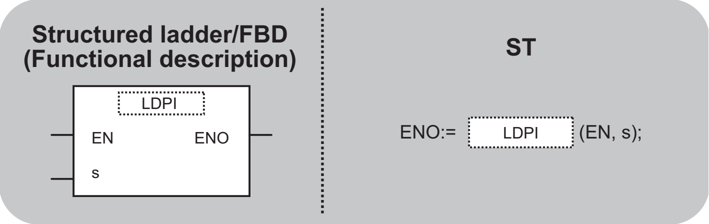
Negated edge operation start, negated edge series connection, negated edge pulse parallel connection

LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI

Basic ✕ High performance ✕ Process ✕ Redundant ✕ Ver. Universal LCPU

- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported


Structured ladder/FBD (Symbolic description)
Bit device number/bit-specified word device (Ⓢ)
LDPI
X1/D0.1
LDFI
X1/D0.1
ANDPI
X2/D0.2
ANDFI
X2/D0.2
ORPI
X3/D0.3
ORFI
X3/D0.3



ENO:= LDPI (EN, s);

Any of the following instruction can go in the dotted squares.
LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI

■Executing condition

Instruction	Executing condition
LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Devices used as contacts	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others DX
	Bit	Word		Bit	Word				
(s)	○	—	○	○	—	○	—		○

Processing details

■LDPI, LDFI

- LDPI is a negated rising edge operation start instruction, and it is ON when the specified bit device is OFF or ON, or at the falling edge of the specified bit device (when it turns from ON to OFF). For a bit-specified word device, it is ON only when the specified bit is 0 or 1, or it changes from 1 to 0.
- LDFI is a negated falling edge operation start instruction, and it is ON when the specified bit device is OFF or ON, or at the rising edge of the specified bit device (when it turns from OFF to ON). For a bit-specified word device, it is ON only when the specified bit is 0 or 1, or it changes from 0 to 1

Device specified in 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

■ANDPI, ANDFI

- ANDPI is a negated rising edge series connection instruction, and ANDFI is a negated falling edge series connection instruction. They perform AND operations with the operation result to that point, and take the resulting value as the operation result. The ON/OFF data used by ANDPI and ANDFI are indicated in the table below.

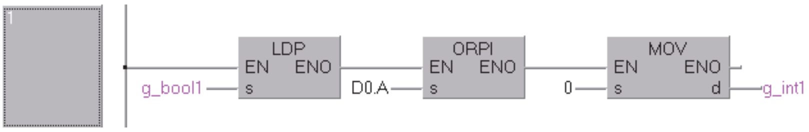
Device specified in ANDPI or ANDFI		ANDPI status	ANDFI status
Bit device	Bit-specified word device		
OFF→ON	0→1	OFF	ON
OFF	0	ON	ON
ON	1	ON	ON
ON→OFF	1→0	ON	OFF

■ORPI, ORFI

- ORPI is a negated rising edge parallel connection instruction, and ORFI is a negated falling edge serial connection instruction. They perform OR operations with the operation result to that point, and take the resulting value as the operation result. The ON/OFF data used by ORPI and ORFI are indicated in the table below.

Device specified in 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

- The ORPI or ORFI instruction performs OR operation of operation result between (s) and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400. When using the ORPI or ORFI instruction, connect EN and ENO in series as shown below.



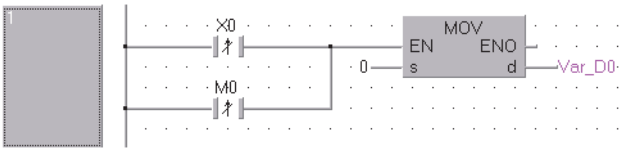
Operation error

- There is no operation error.

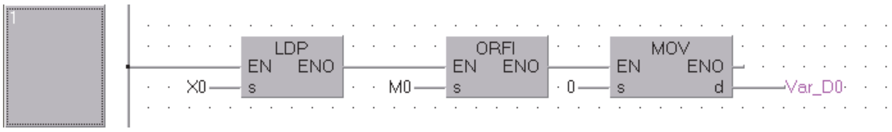
Program example

- In the following program, the value 0 is stored to Var_D0 in any of the following status: X0 is ON/OFF, X0 turns from ON to OFF, M0 is ON/OFF, or M0 turns from OFF to ON.

[Structured ladder/FBD (Symbolic description)]



[Structured ladder/FBD (Functional description)]



[ST (When using IF syntax)]

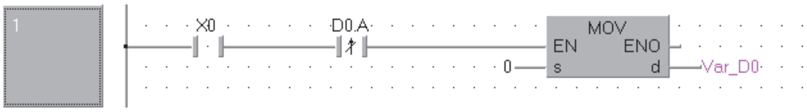
```
IF (LDPI(TRUE,X0) OR LDFI(TRUE,M0)) THEN
    MOV(TRUE,0,Var_D0);
END_IF;
```

[ST (When not using IF syntax)]

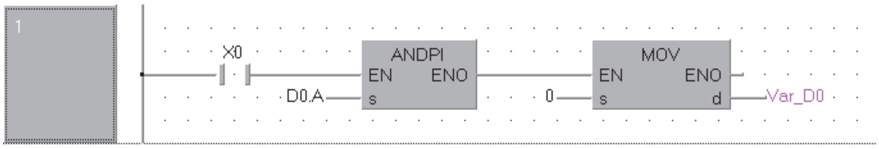
```
MOV(ORFI(LDPI(TRUE,X0),M0),0,Var_D0);
```

- In the following program, the value 0 is stored to Var_D0 when X0 is ON and b10 (bit 10) of D10 is ON/OFF or turns from ON to OFF.

[Structured ladder/FBD (Symbolic description)]



[Structured ladder/FBD (Functional description)]



[ST (When using IF syntax)]

```
IF (X0 & LDPI(TRUE,D0.A)) THEN
    MOV(TRUE,0,Var_D0);
END_IF;
```

[ST (When not using IF syntax)]

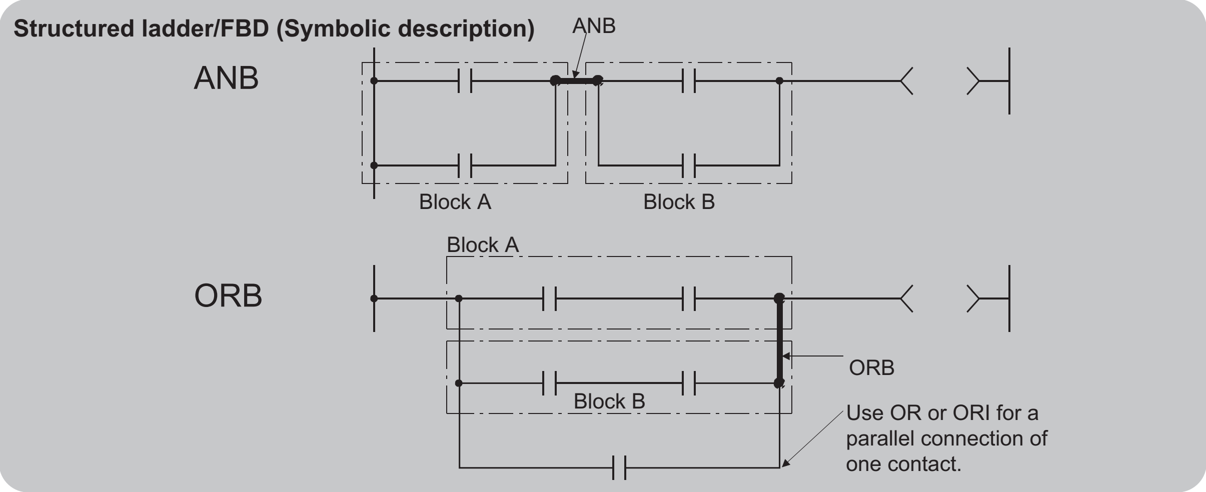
```
MOV((ANDPI(TRUE,X0),D0.A),0,D5);
```

5.2 Bond Instructions

Ladder block series connection and parallel connection

ANB, ORB

Basic | High performance | Process | Redundant | Universal | LCPU


Structured ladder/FBD (Symbolic description)
ANB
ANB
Block A
Block B
ORB
Block A
Block B
ORB
Use OR or ORI for a parallel connection of one contact.

■Executing condition

Instruction	Executing condition
ANB, ORB	Non-conditional execution

Processing details

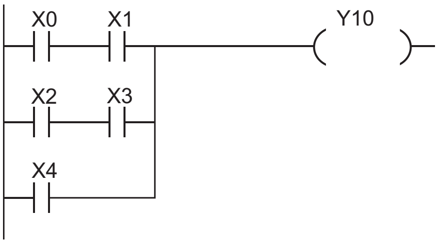
■ANB

- Performs an AND operation on block A and block B, and takes the resulting value as the operation result.
- The symbol for ANB is not the contact symbol, but rather is the connection symbol.

■ORB

- Conducts an OR operation on Block A and Block B, and takes the resulting value as the operation result.
- ORB is used to perform parallel connections for ladder blocks with two or more contacts. For ladder blocks with only one contact, use OR or ORI.

[Structured ladder/FBD]



- The ORB symbol is not the contact symbol, but rather is the connection symbol.

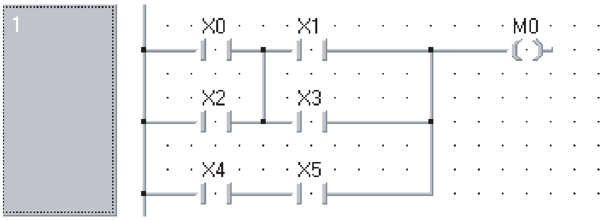
Operation error

- There is no operation error.

Program example

- The program using the ANB and ORB instructions.

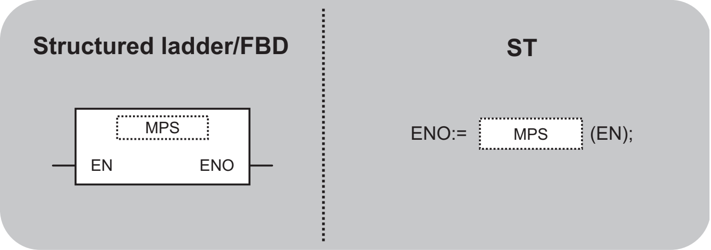
[Structured ladder/FBD]



Operation result push/read/pop

MPS, MRD, MPP

Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.
MPS, MRD, MPP

■Executing condition

Instruction	Executing condition
MPS, MRD, MPP	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

■MPS

- Stores the memory of the operation result (ON or OFF) immediately prior to the MPS instruction.
- Up to 16 MPS instructions can be used successively. If the MPP instruction is used during this process, the number of uses calculated for the MPS instruction will be decremented by one.

■MRD

- Reads the operation result stored in the MPS instruction, and uses that result to perform the operation in the next step.

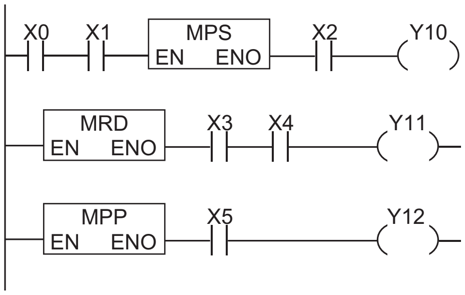
■MPP

- Reads the operation result stored in the MPS instruction, and uses that result to perform the operation in the next step.
- Clears the operation results stored in the MPS instruction.
- Subtracts 1 from the number of MPS instruction times of use.

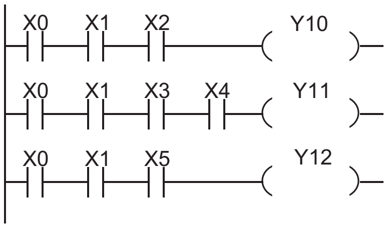
Point

The following shows the ladders using and without using the MPS, MRD, and MPP instructions.

- Ladder using the MPS, MRD, and MPP instructions



- Ladder without using the MPS, MRD, and MPP instructions



The MPS and MPP instructions must be used the same number of times.

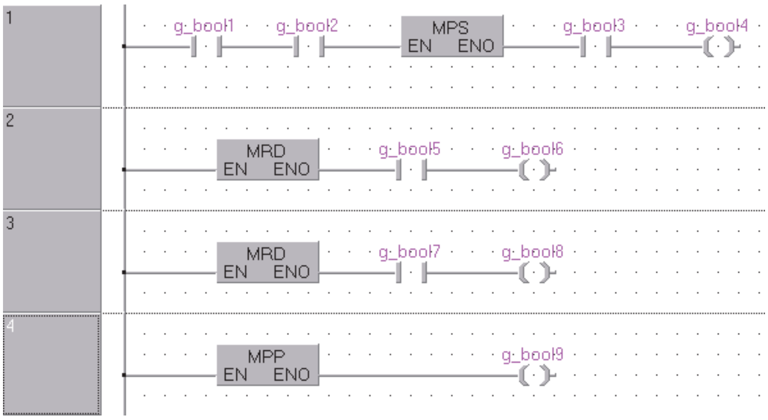
Operation error

- There is no operation error.

Program example

- The program using the MPS, MRD, and MPP instructions.

[Structured ladder/FBD]

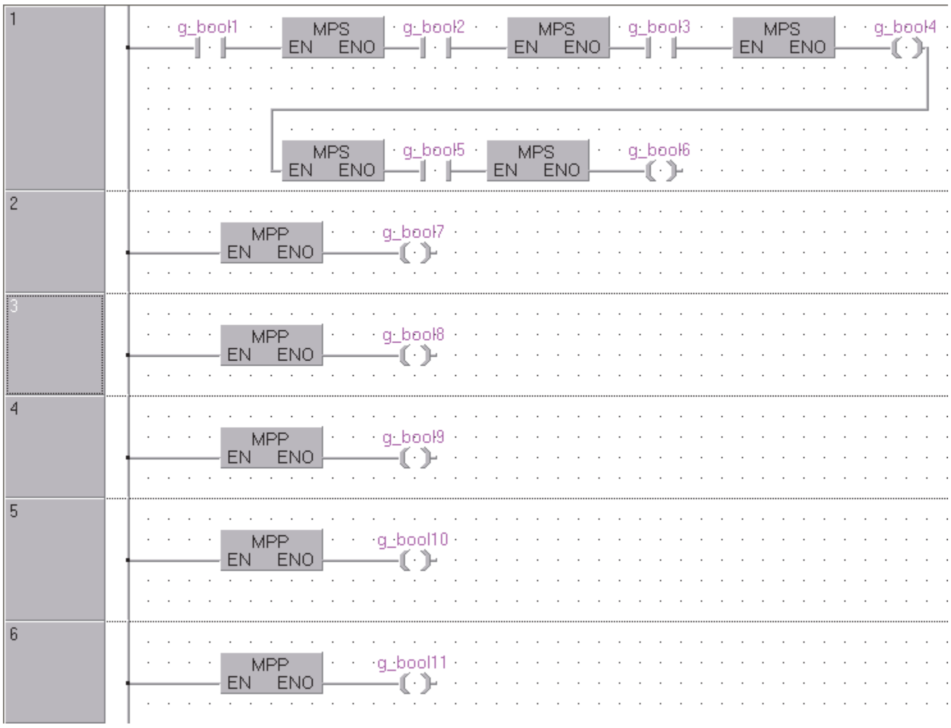


[ST]

```
OUT(MPS(g_bool1 AND g_bool2) AND g_bool3,g_bool4);
OUT(MRD(TRUE) AND g_bool5,g_bool6);
OUT(MRD(TRUE) AND g_bool7,g_bool8);
OUT(MPP(TRUE),g_bool9);
```

- The program using the MPS and MPP instructions successively.

[Structured ladder/FBD]



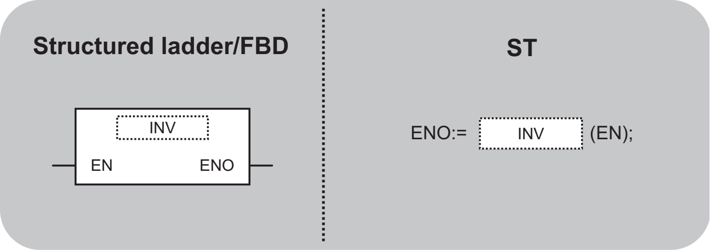
[ST]

```
OUT(MPS(MPS(MPS(MPS(g_bool1) AND g_bool2) AND g_bool3) AND g_bool4) AND g_bool5,g_bool6);
OUT(MPP(TRUE),g_bool7);
OUT(MPP(TRUE),g_bool8);
OUT(MPP(TRUE),g_bool9);
OUT(MPP(TRUE),g_bool10);
OUT(MPP(TRUE),g_bool11);
```

Operation result inversion

INV

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

INV

■Executing condition

Instruction	Executing condition
INV	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

- Inverts the operation result immediately prior to the INV instruction.

Operation result immediately prior to the INV instruction	Operation result following the execution of the INV instruction
OFF	ON
ON	OFF

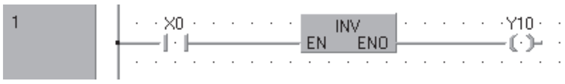
Operation error

- There is no operation error.

Program example

- The program which inverts the X0 ON/OFF data, and outputs from Y10.

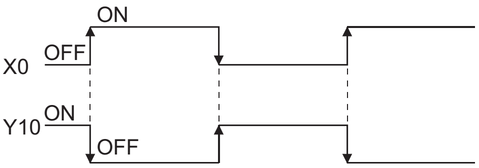
[Structured ladder/FBD]



[ST]

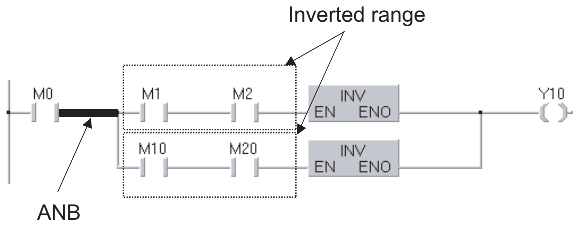
OUT(INV(X0),Y10);

[Timing chart]



Point

- The INV instruction operates based on the results of calculation made until the INV instruction is given. Accordingly, use it in the same position as that of AND. The INV instruction cannot be used at the LD and OR positions.
- When using a ladder block, the operation result is inverted within the range of the ladder block. To operate a ladder using the INV instruction in combination with the ANB instruction, pay attention to the range which is inverted.

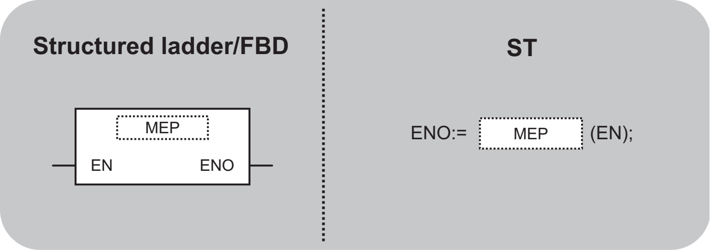


For details of the ANB instruction, refer to ☞ Page 92 Ladder block series connection and parallel connection.

Pulse conversion of operation result

MEP, MEF

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

MEP, MEF

■Executing condition

Instruction	Executing condition
MEP, MEF	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

■MEP

- If operation results up to the MEP instruction are rising edge (from OFF to ON), turns ON (conduction state). If operation results up to the MEP instruction are anything other than rising edge, turns OFF (non-conduction state).
- Use of the MEP instruction simplifies pulse conversion processing when multiple contacts are connected in series.

■MEF

- If operation results up to the MEF instruction are falling edge (from ON to OFF), turns ON (conduction state). If operation results up to the MEF instruction are anything other than falling edge, turns OFF (non-conduction state).
- Use of the MEF instruction simplifies pulse conversion processing when multiple contacts are connected in series.

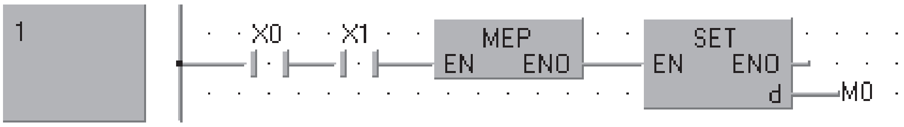
Operation error

- There is no operation error.

Program example

- The program which performs pulse conversion to the operation results of X0 and X1

[Structured ladder/FBD]



[ST]

```
IF X0 AND X1 THEN
    SET(MEP(TRUE),M0);
END_IF;
```

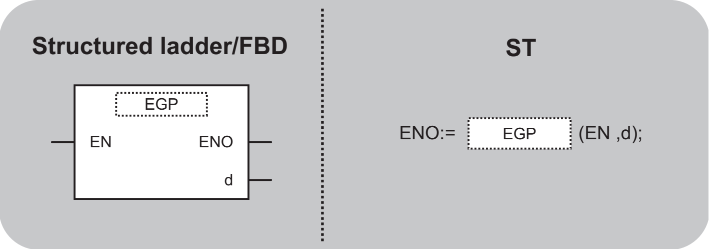
Point

- The MEP and MEF instructions will occasionally not function properly when pulse conversion is conducted for a contact that has been indexed by a subroutine program or FOR to NEXT instruction loop. If pulse conversion is to be conducted for a contact that has been indexed by a subroutine program or FOR to NEXT instruction loop, use the EGP/EGF instructions.
- Because the MEP and MEF instructions operate with the operation results immediately prior to the MEP and MEF instructions, AND should be used at the same position. The MEP and MEF instructions cannot be used at the LD or OR position.

Pulse conversion of edge relay operation result

EGP, EGF

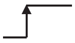
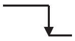
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

EGP, EGF

■Executing condition

Instruction	Executing condition
EGP	
EGF	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Edge relay number that stores the operation result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				V
(d)	—								○

Processing details

■EGP

- Operation results up to the EGP instruction are stored in memory by the edge relay (V).
- Turns ON (conduction state) at the rising edge (OFF to ON) of the operation result up to the EGP instruction. If the operation result up to the EGP instruction is other than a rising edge (ON to ON, ON to OFF, or OFF to OFF), it turns OFF (non-conduction state).
- The EGP instruction is used when conducting pulse operations for indexed programs in subroutine programs and FOR to NEXT instruction loop.
- The EGP instruction can be used like AND.

■EGF

- Operation results up to the EGF instruction are stored in memory by the edge relay (V).
- Turns ON at the falling edge (from ON to OFF) of the operation result up to the EGF instruction. If the operation result up to the EGF instruction is other than a falling edge (OFF to ON, ON to ON, or OFF to OFF), it turns OFF (non-conduction state).
- The EGF instruction is used when conducting pulse operations for indexed programs in subroutine programs and FOR to NEXT instruction loop.
- The EGF instruction can be used like AND.

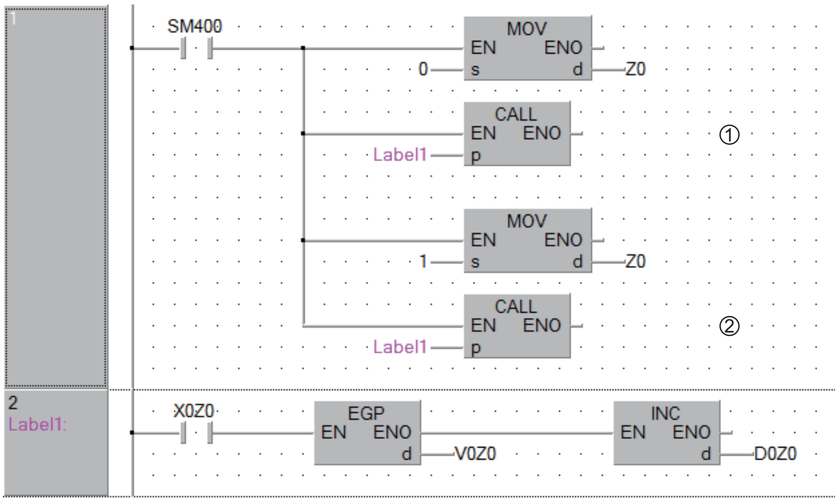
Operation error

- There is no operation error.

Program example

- The program using the EGP instruction.

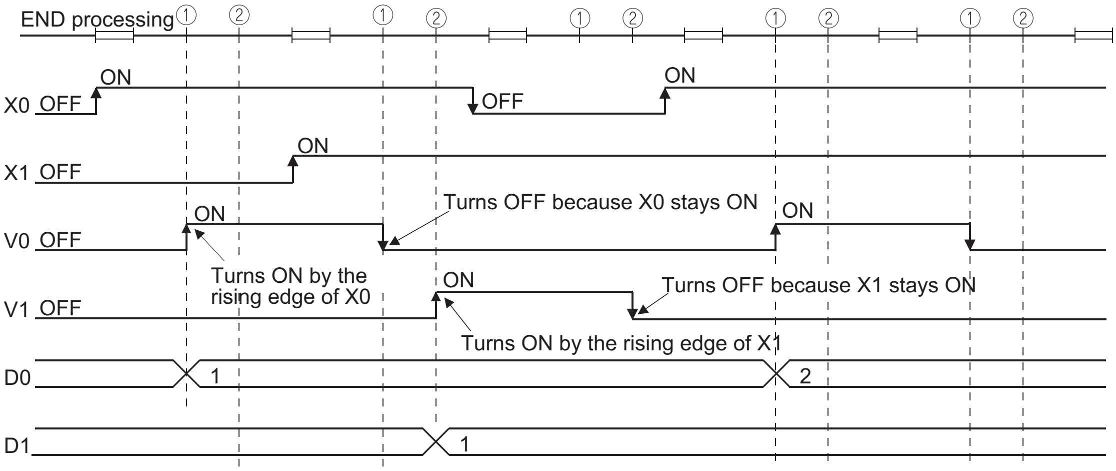
[Structured ladder/FBD]



[ST]

```
INC(EGP(X0Z0,V0Z0),D0Z0);
```

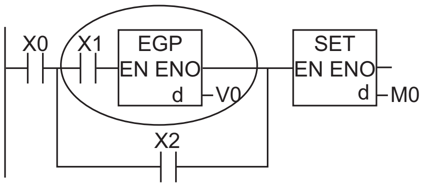
[Operation]



END processing
① ②
X0 OFF
ON
OFF
ON
X1 OFF
ON
V0 OFF
ON
Turns ON by the rising edge of X0
Turns OFF because X0 stays ON
ON
V1 OFF
ON
Turns ON by the rising edge of X1
Turns OFF because X1 stays ON
D0
1
2
D1
1

Point

- Since the EGP and EGF instructions are executed according to the operation results performed immediately before the EGP and EGF instructions, these instructions must be used at the same position as AND. (Refer to ☞ Page 83 Operation start, series connection, parallel connection.) The EGP and EGF instructions cannot be used at the LD or OR position.
- EGP and EGF instructions cannot be used at the ladder block positions shown below.

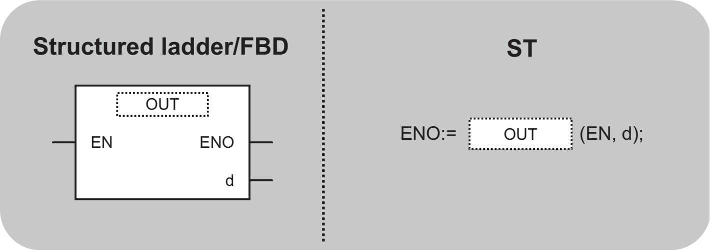


5.3 Output Instructions

Out (excluding timers, counters and annunciators)

OUT

Basic High performance Process Redundant Universal LCPU



The following instruction can go in the dotted squares.

OUT

■Executing condition

Instruction	Executing condition
OUT	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Device number to be turned ON or OFF	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others DY
	Bit	Word		Bit	Word				
(d)	○(Other than T, CF)	○					—		○

Processing details

- Operation results up to the OUT instruction are output to the specified device.

Item	Operation result	Coil
For bit devices	OFF	OFF
	ON	ON

Item	Operation result	Bit specified
For bit-specified word devices	OFF	0
	ON	1

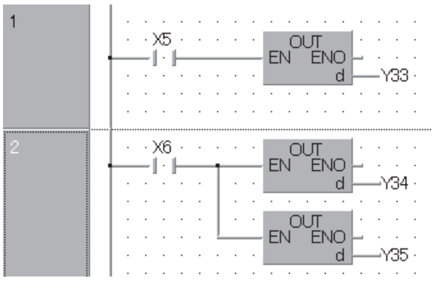
Operation error

- There is no operation error.

Program example

- For bit devices

[Structured ladder/FBD]

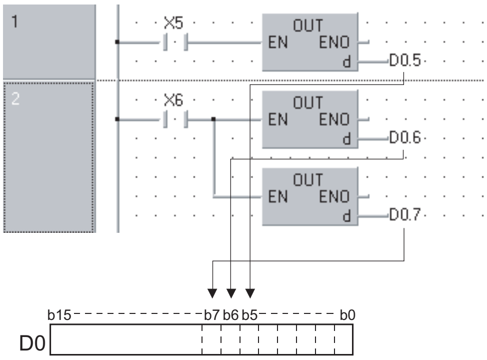


[ST]

```
OUT(X5,Y33);
OUT(X6,Y34);
OUT(X6,Y35);
```

- For bit-specified word devices

[Structured ladder/FBD]



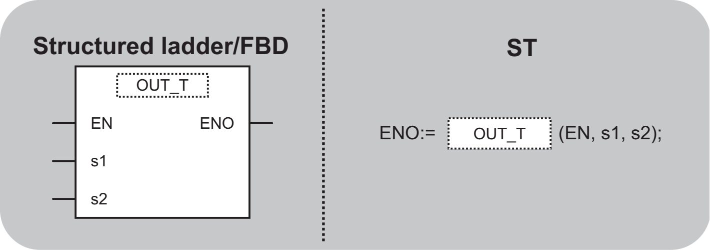
[ST]

```
OUT(X5,D0.5);
OUT(X6,D0.6);
OUT(X6,D0.7);
```

Timer

OUT_T, OUTH_T

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

OUT_T, OUTH_T

■Executing condition

Instruction	Executing condition
OUT_T, OUTH_T	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1(TCoil)	Timer number	Bit
	s2(TValue)	Timer set value (Setting range: 1 to 32767)	ANY16
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K	Others
	Bit	Word		Bit	Word				
(s1)	○ (TC only)	—	—	—	—		—	—	—
(s2)	—	○ (Other than T or C)	○	—	○		—	○	—

Processing details

- When the operation results up to the OUT(H)_T instruction are ON, the timer coil turns ON and the timer counts up to the value that has been set, when the time up status (total numeric value is equal to or greater than the setting value), the contact responds as follows.
  - Normally open contact: Conduction
  - Normally closed contact: Non-conduction
- The contact responds as follows when the operation result up to the OUT(H)_T instruction is a change from ON to OFF.

Type of timer	Timer coil	Current value of timer	Prior to time up		After time up	
			Normally open contact	Normally closed contact	Normally open contact	Normally closed contact
Low-speed timer	OFF	0	Non-conduction	Conduction	Non-conduction	Conduction
High-speed timer						
Low-speed retentive timer	OFF	Retains the Current value	Non-conduction	Conduction	Conduction	Non-conduction
High-speed retentive timer						

- To clear the current value of a retentive timer and turn the contact OFF after time up, use the RST instruction.
- A negative number (-32768 to -1) cannot be set as the setting value for the timer. If the set value is 0, the timer will time out when the time the OUT(H)_T instruction is executed.
- The following processing is conducted when the OUT(H)_T instruction is executed. In cases where the JMP instruction or the like is used to jump to the OUT(H)_T instruction while the OUT(H)_T instruction is ON, no current value update or contact ON/OFF operation is conducted. Also, if the same OUT(H)_T instruction is conducted two or more times during the same scan, the current value of the number of repetitions executed will be updated.
  - OUT(H)_T T□ coil turned ON or OFF
  - OUT(H)_T T□ contact turned ON or OFF
  - OUT(H)_T T□ current value updated
- Index setting for timer coils or contacts can be conducted only by Z0 or Z1 (except for the QnUDVCPU and QnUDPVCPU).
- Timer setting value has no limitation for index setting.

Point

- Time limit of the timer is set in the PLC system of the PLC parameter dialog box.

[For the Basic model QCPU, high Performance model QCPU, Process CPU, and Redundant CPU]

Low speed timer/Low speed retentive timer: 1ms to 1000ms (default: 100ms, setting unit: 1ms)

High speed timer/High speed retentive timer: 0.1ms to 100.0ms (default: 10.0ms, setting unit: 0.1ms)

[For the Universal model QCPU and LCPU]

Low speed timer/Low speed retentive timer: 1ms to 1000ms (default: 100ms, setting unit: 1ms)

High speed timer/High speed retentive timer: 0.01ms to 100.0ms (default: 10.0ms, setting unit: 0.01ms)

- For information on timer counting methods, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

Operation error

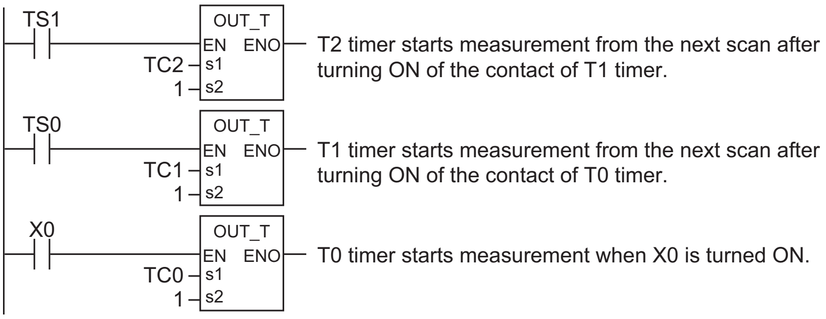
- There is no operation error.

Precautions

- When creating a program in which the operation of the timer contact triggers the operation of other timer, create the program for the timer that operates later first. In any of the following cases, all timers turn ON at the same scan if the program is created in the order the timers operate.
  - If the set value is smaller than a scan time
  - If 1 is set

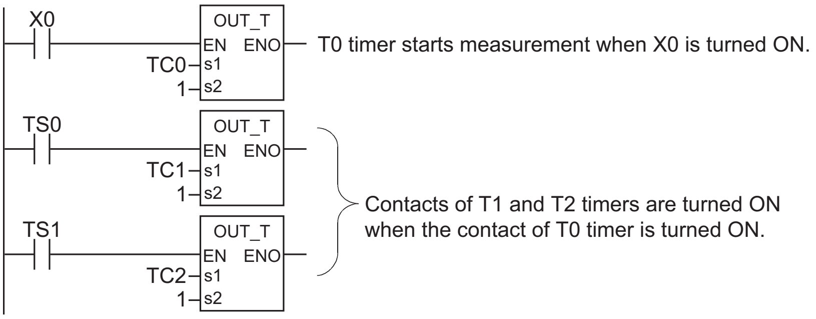
Ex.

For timers T0 to T2, the program is created in the order the timer operates later.


TS1
OUT_T
EN ENO
TC2 s1
1 s2
T2 timer starts measurement from the next scan after turning ON of the contact of T1 timer.
TS0
OUT_T
EN ENO
TC1 s1
1 s2
T1 timer starts measurement from the next scan after turning ON of the contact of T0 timer.
X0
OUT_T
EN ENO
TC0 s1
1 s2
T0 timer starts measurement when X0 is turned ON.

Ex.

For timers T0 to T2, the program is created in the order of timer operation.

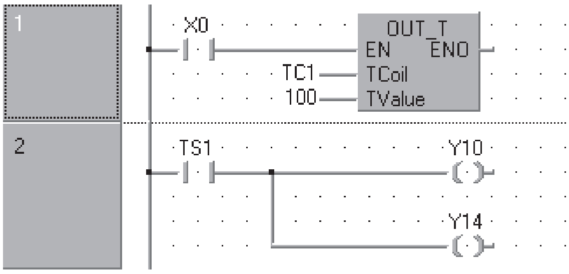

X0
OUT_T
EN ENO
TC0 s1
1 s2
T0 timer starts measurement when X0 is turned ON.
TS0
OUT_T
EN ENO
TC1 s1
1 s2
TS1
OUT_T
EN ENO
TC2 s1
1 s2
Contacts of T1 and T2 timers are turned ON when the contact of T0 timer is turned ON.

- When using timer devices in a program, it is necessary to separate the usage of devices into following classification depending on the position of use. When T or ST is entered, they are treated as TN or STN respectively.
  - When using it for a contact: TS, STS
  - When using it for a coil: TC, STC
  - When using it for a current value: TN, STN

Program example

- In the following program, Y10 and Y14 are turned ON, 10 seconds after X0 has turned ON. The setting value of the low-speed timer indicates its default time limit (100ms).

[Structured ladder/FBD]

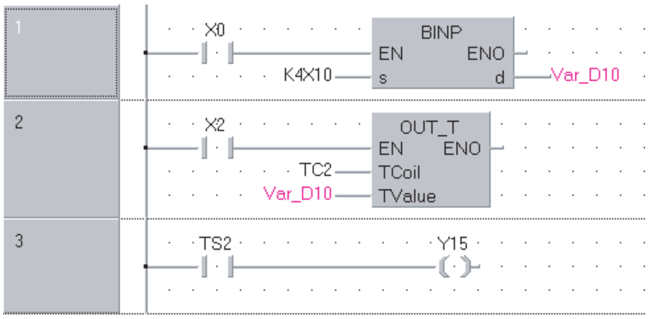


[ST]

```
OUT_T(X0,TC1,100);
OUT(TS1,Y10);
OUT(TS1,Y14);
```

- In the following program, the BCD data of X10 to X1F are set as the timer's setting value.

[Structured ladder/FBD]



Converts BCD data from X10 to X1F to BIN data, and stores the result to Var_D10.

The data stored in Var_D10 are used for measurement as a set value when X2 turns ON.

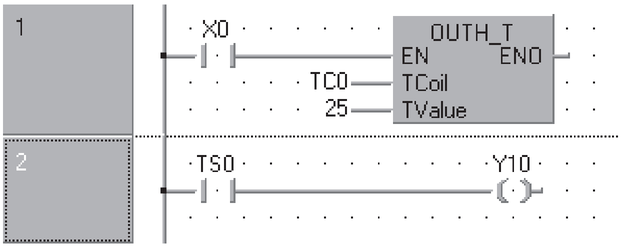
Y15 turns ON when T2 counts out.

[ST]

```
BINP(X0,K4X10,Var_D10);
OUT_T(X2,TC2,Var_D10);
OUT(TS2,Y15);
```

- In the following program, Y10 is turned ON, 250 milliseconds after X0 has turned ON. The setting value of the high-speed timer indicates its default time limit (10ms).

[Structured ladder/FBD]



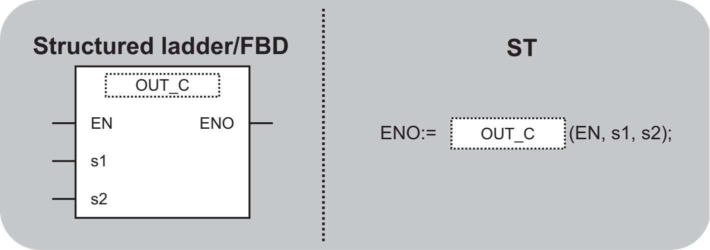
[ST]

```
OUTH_T(X0,TC0,25);
OUT(TS0,Y10);
```

Counter

OUT_C

Basic High performance Process Redundant Universal LCPU



The following instruction can go in the dotted squares.

OUT_C

■Executing condition

Instruction	Executing condition
OUT_C	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1(CCoil)	Counter number	Bit
	s2(CValue)	Counter set value (Setting range: 1 to 32767)	ANY16
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K	Others
	Bit	Word		Bit	Word				
(s1)	○ (CC only)	—	—	—	—		—	—	—
(s2)	—	○ (Other than T or C)	○	—	○		—	○	—

Processing details

- When the operation results up to the OUT_C instruction change from OFF to ON, 1 is added to the current value (count value) and the count up status (current value set value), and the contacts respond as follows
  - Normally open contact: Conduction
  - Normally closed contact: Non-conduction
- No count is performed with the operation results at ON. (There is no need to perform pulse conversion on count input.)
- After the count up status is reached, there is no change in the count value or the contacts until the RST instruction is executed.
- A negative number (-32768 to -1) cannot be set as the setting value for the timer. If the set value is 0, the processing is identical to that which takes place for 1.
- Indexing for the counter coil and contact can use only Z0 and Z1. (except for the QnUDVCPU and QnUDPVCPU).
- Counter setting value has no limitation for index setting.
- When using counter devices in a program, it is necessary to separate the usage of devices into following classification depending on the position of use. When C is entered, it is treated as CN.
  - When using it for a contact: CS
  - When using it for a coil: CC
  - When using it for a current value: CN

Point

For counter counting methods, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

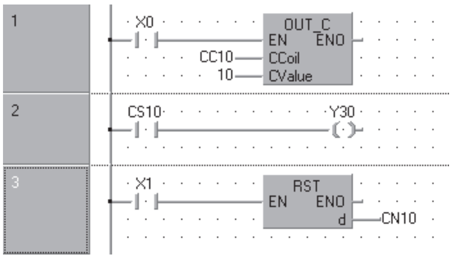
Operation error

- There is no operation error.

Program example

- In the following program, Y30 is turned ON after X0 has turned ON 10 times, and the counter is reset when X1 turns ON.

[Structured ladder/FBD]

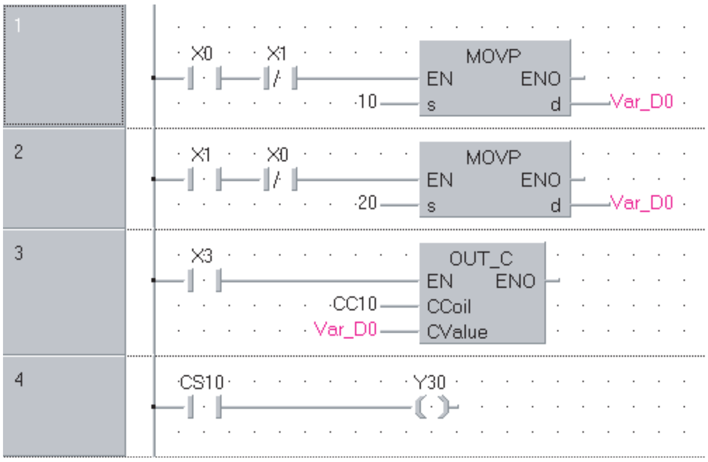


[ST]

```
OUT_C(X0,CC10,10);
OUT(CS10,Y30);
RST(X1,CN10);
```

- In the following program, the setting value for C10 is set at 10 when X0 turns ON, and at 20 when X1 turns ON.

[Structured ladder/FBD]



Stores 10 to D0 when X0 turns ON.

Stores 20 to D0 when X1 turns ON.

C10 counts data stored in D0 as a setting value.

Y30 turns ON when C10 counts out.

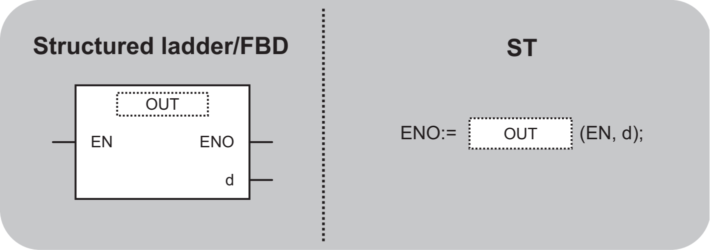
[ST]

```
IF X0 AND NOT(X1) THEN
    MOVP(TRUE,10,Var_D0);
END_IF;
IF NOT(X0) AND X1 THEN
    MOVP(TRUE,20,Var_D0);
END_IF;
OUT_C(X3,CC10,Var_D0);
OUT(CS10,Y30);
```

Annunciator output

OUT

Basic | High performance | Process | Redundant | Universal | LCPU



ENO:= OUT (EN, d);

The following instruction can go in the dotted squares.

OUT

■Executing condition

Instruction	Executing condition
OUT	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Annunciator number to be turned ON	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(d)	○ F only	—							

Processing details

- Operation results up to the OUT instruction are output to the specified annunciator.
- The following responses occur when an annunciator (F) is turned ON.
  - The "USER" LED[*1] turns ON.
  - The annunciator numbers which are ON (F numbers) are stored to special registers (SD64 to SD79).
  - The value of SD63 is incremented by 1.

*1 For Basic model QCPU, the "ERR" LED displays on the front of the CPU module.

- If the value of SD63 is 16 (which happens when 16 annunciators are already ON), even if a new annunciator is turned ON, its number will not be stored to SD64 to SD79.
- The following responses occur when the annunciator is turned OFF by the OUT instruction.
  - The coil turns OFF, but there are no changes in the status of the "USER" LED[*2] and the content of the values stored to SD63 to SD79.
  - Use the RST instruction to make the "USER" LED[*2] turn OFF as well as to delete the annunciator which was turned OFF by the OUT instruction from SD63 to SD79.

*2 For Basic model QCPU, the "ERR" LED displays on the front of the CPU module.

Point

For details of annunciators, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module to be used.

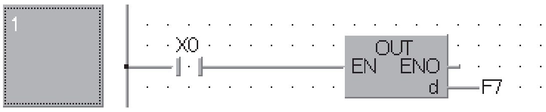
Operation error

- There is no operation error.

Program example

- In the following program, F7 is turned ON when X0 turns ON, and the value 7 is stored to the devices from SD64 to SD79.

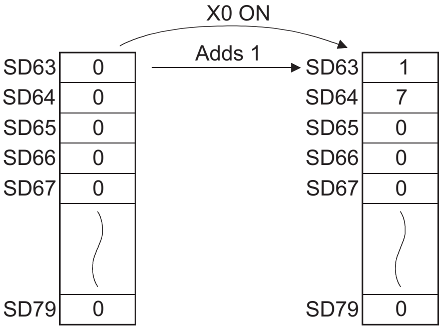
[Structured ladder/FBD]



[ST]

```
OUT(X0,F7);
```

[Operation]



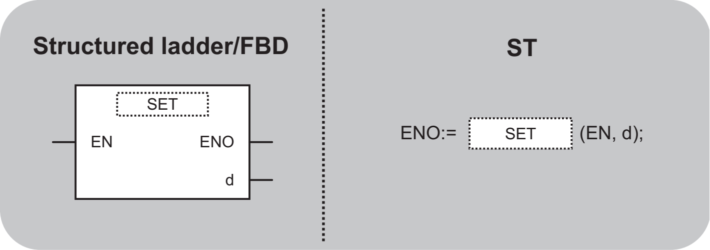
SD63	0
SD64	0
SD65	0
SD66	0
SD67	0
SD79	0

SD63	1
SD64	7
SD65	0
SD66	0
SD67	0
SD79	0

Setting devices (excluding annunciators)

SET

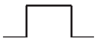
Basic | High performance | Process | Redundant | Universal | LCPU



The following instruction can go in the dotted squares.

SET

■Executing condition

Instruction	Executing condition
SET	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Bit device number or bit-specified word device to be turned ON	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				DY
(d)	○	○ (Other than T or C)		○			—		○

Point

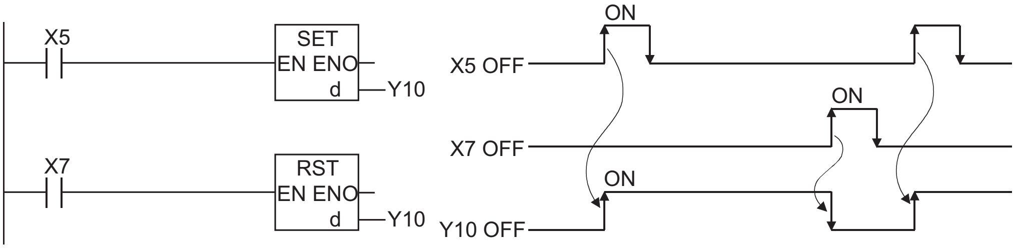
When BL, S, TR, BL\S, or BL\TR is used, refer to SFC control instructions of the MELSEC-Q/L/QnA Programming Manual (SFC).

Processing details

- When the execution command is turned ON, the status of the specified devices becomes as shown below:

Device	Device status
Bit device	Turns coils and contacts ON
Bit-specified word device	Sets the value of specified bit to 1

- Devices turned ON remain ON even when the execution command is turned OFF. Devices turned ON by the SET instruction can be turned OFF by the RST instruction.



- When the execution command is OFF, the status of devices does not change.

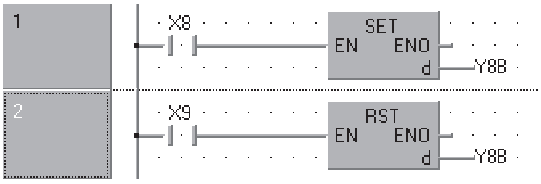
Operation error

- There is no operation error.

Program example

- In the following program, Y8B is set (ON) when X8 turns ON, and Y8B is reset (OFF) when X9 turns ON.

[Structured ladder/FBD]

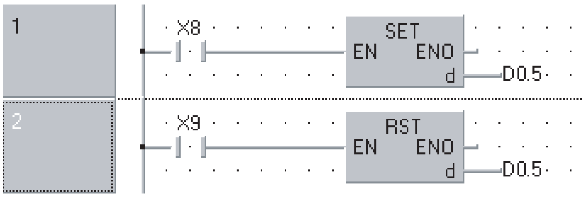


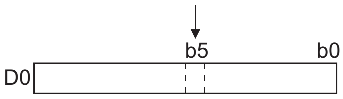
[ST]

```
SET(X8,Y8B);
RST(X9,Y8B);
```

- In the following program, the value of D0 b5 (bit 5) is set to 1 when X8 turns ON, and the bit value is set to 0 when X9 turns ON.

[Structured ladder/FBD]





[ST]

```
SET(X8,D0.5);
RST(X9,D0.5);
```

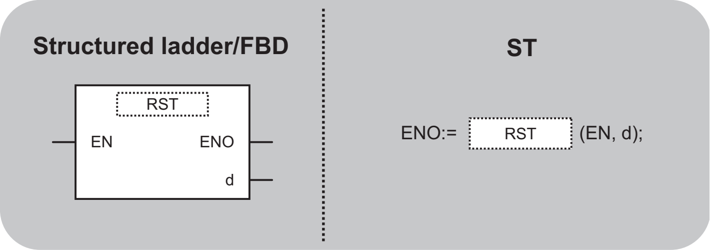
Point

When using X as a device, use the device numbers that are not used for the actual input. If the same number is used for the actual input device and input X, the data of the actual input will be written over with the input X specified in the SET instruction.

Resetting devices (excluding annunciators)

RST

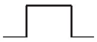
Basic | High performance | Process | Redundant | Universal | LCPU



The following instruction can go in the dotted squares.

RST

■Executing condition

Instruction	Executing condition
RST	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Bit device number or bit-specified word device to be reset Word device number to be reset	ANY_SIMPLE

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others DY
	Bit	Word		Bit	Word				
(d)	○							—	○

Point

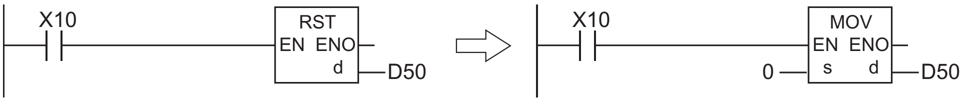
When BL, S, TR, BL\S, or BL\TR is used, refer to SFC control instructions of the MELSEC-Q/L/QnA Programming Manual (SFC).

Processing details

- When the execution command is turned ON, the status of the specified devices becomes as shown below:

Device	Device status
Bit device	Turns coils and contacts OFF
Timer and counter	Sets the current value to 0, and turns coils and contacts OFF
For bit-specified word devices	Sets the value of specified bit to 0
Word device other than timer and counter	Sets the value to 0

- When the execution command is OFF, the status of devices does not change.
- The functions of the word devices specified by the RST instruction are identical to the following ladder.



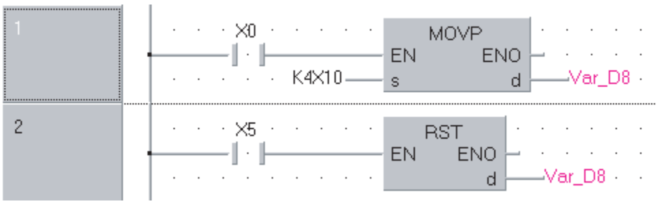
Operation error

- There is no operation error.

Program example

- In the following program, the value of the data register is set to 0.

[Structured ladder/FBD]



Stores the data from X10 to X1F to Var_D8 when X0 turns ON.

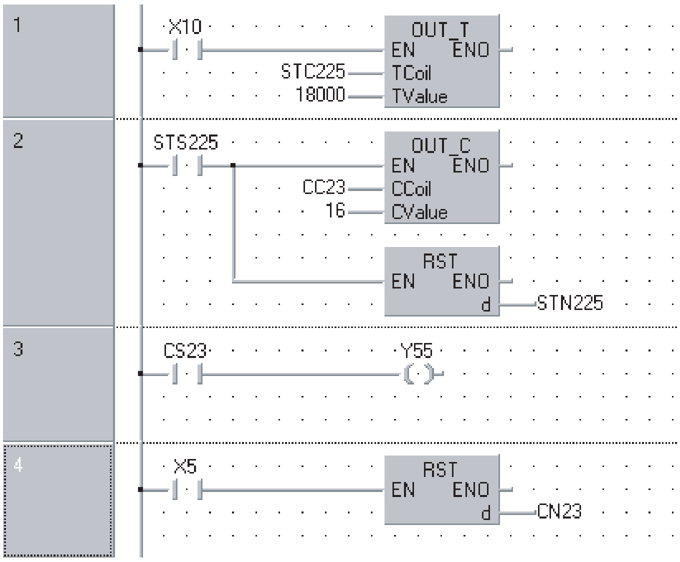
Resets the data in Var_D8 to 0 when X5 turns ON.

[ST]

```
MOVP(X0,K4X10,Var_D8);
RST(X5,Var_D8);
```

- In the following program, the 100ms retentive timer and counter are reset.

[Structured ladder/FBD]



When ST225 is set as retentive timer, it turns ON when X4's ON time reaches 30 minutes.

Counts the number of times ST225 turned ON.

Resets the coil, contact, and current value of ST255 when the contact of ST255 turns ON.

Y55 turns ON when C23 counts out.

Resets the data in C23 when X5 turns ON.

[ST]

```
OUT_T(X4,STC225,18000);
IF STS225 THEN
    OUT_C(TRUE,CC23,16);
    RST(TRUE,STN225);
END_IF;
OUT(CS23,Y55);
RST(X5,CN23);
```

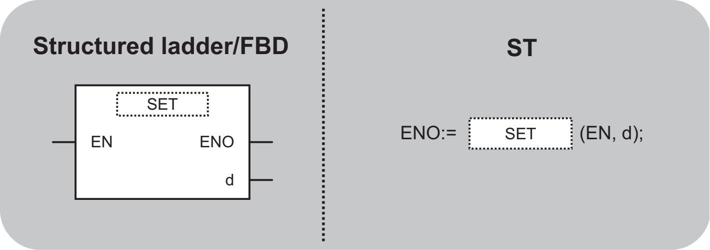
Point

When resetting timers and counter devices, only coils (TC, STC, CC) and current values (TN, STN, CN) can be specified for output argument (d). Contacts (TS, STS, CS) cannot be specified.

Setting and resetting annunciators

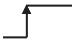
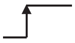
SET, RST

Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.
SET, RST

■Executing condition

Instruction	Executing condition
SET	
RST	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Annunciator number to be set	Bit
		Annunciator number to be reset	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(d)	○ F only	—							

Processing details

■SET

- The annunciator specified for (d) is turned ON when the execution command is turned ON.
- The following responses occur when an annunciator (F) is turned ON.
  - The "USER" LED[*1] turns ON.
  - The annunciator numbers which are ON (F numbers) are stored to special registers (SD64 to SD79).
  - The value of SD63 is incremented by 1.

[*1] For Basic model QCPU, the "ERR" LED displays on the front of the CPU module.

- If the value of SD63 is 16 (which happens when 16 annunciators are already ON), even if a new annunciator is turned ON, its number will not be stored to SD64 to SD79.

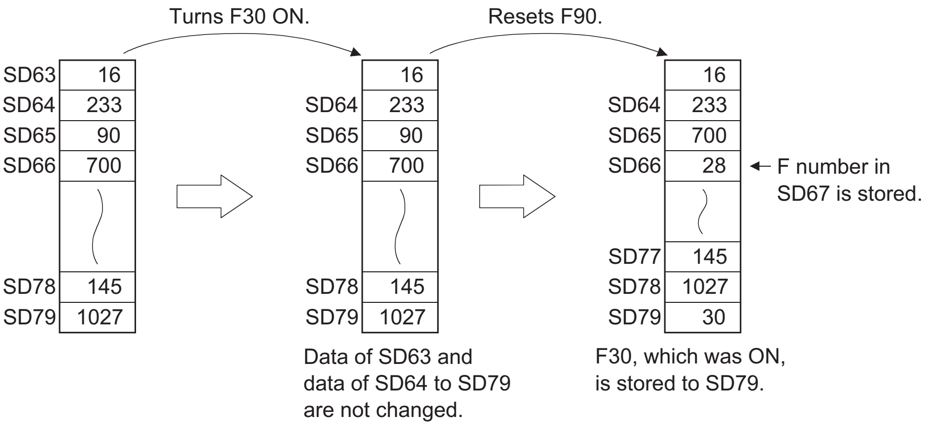
■RST

- The annunciator specified for (d) is turned OFF when the execution command is turned ON.
- The annunciator numbers (F numbers) of annunciators that have turned OFF are deleted from the special registers (SD64 to SD79), and the value of SD63 is decremented by 1.
- When the value of SD63 is "16", the annunciator numbers are deleted from SD64 to SD79 by the use of the RST instruction. If the annunciators whose numbers are not registered to SD64 to SD79 are ON, these numbers are registered. If all annunciator numbers from SD64 to SD79 are turned OFF, the "USER" LED[*1] is turned OFF.

[*1] For Basic model QCPU, the "ERR" LED displays on the front of the CPU module.

Ex.

[Operations which take place when SD63 is 16]



Point

For details of annunciators, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module to be used.

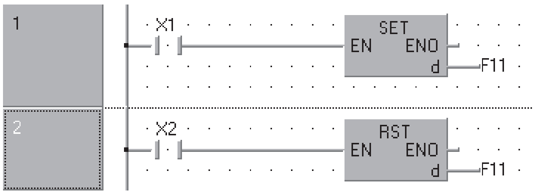
Operation error

- There is no operation error.

Program example

- In the following program, the annunciator F11 is turned ON when X1 turns ON, and the value 11 is stored to the special register (SD64 to SD79). Further, the annunciator F11 is reset when X2 turns ON, and the value 11 is deleted from the special registers (SD64 to SD79).

[Structured ladder/FBD]

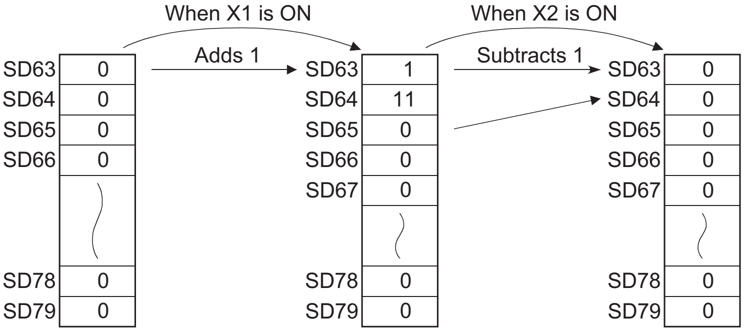


[ST]

SET(X1,F11);

RST(X2,F11);

[Operation]

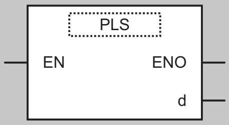


Rising edge and falling edge outputs

PLS, PLF

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



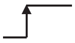
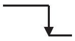
ST

ENO:= PLS (EN, d);

Any of the following instruction can go in the dotted squares.

PLS, PLF

■Executing condition

Instruction	Executing condition
PLS	
PLF	

■Argument

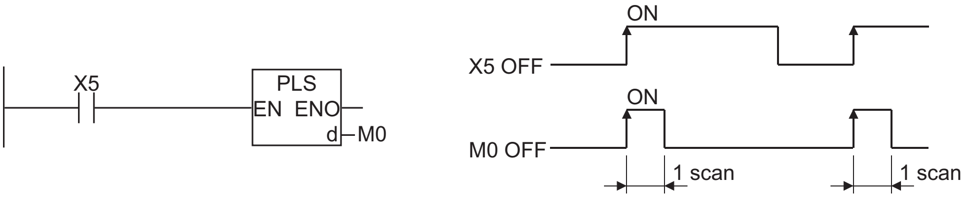
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Device to be set as pulse output	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others DY
	Bit	Word		Bit	Word				
(d)	○						—		○

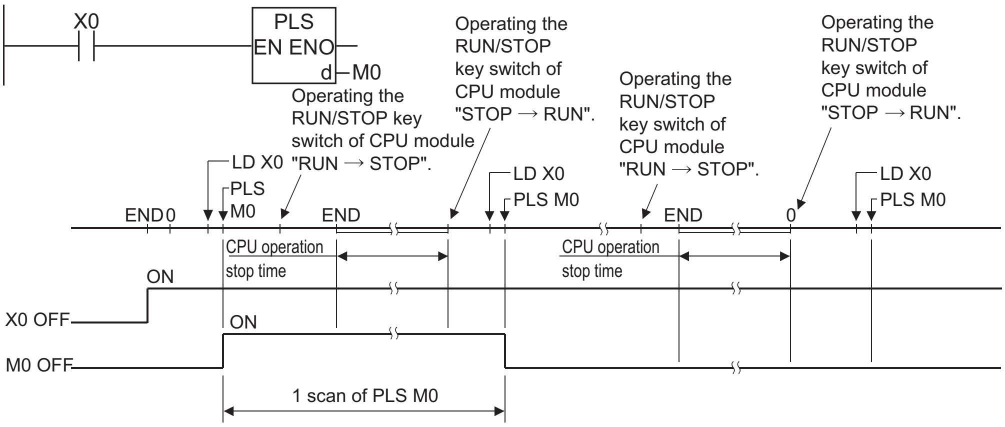
Processing details

■PLS

- Turns ON the specified device when the execution command is turned from OFF to ON, and turns OFF the device in any other case the execution command is turned from OFF to ON (ON to ON, ON to OFF or OFF to OFF).
- When there is one PLS instruction for the device specified for (d) during one scan, the specified device turns ON one scan.
- Refer to ☞ Page 72 Operation of OUT Instructions, SET/RST Instructions, or PLS/PLF Instructions Using Same Device for the operation to be performed when the PLS instruction for the same device is executed more than once during one scan.



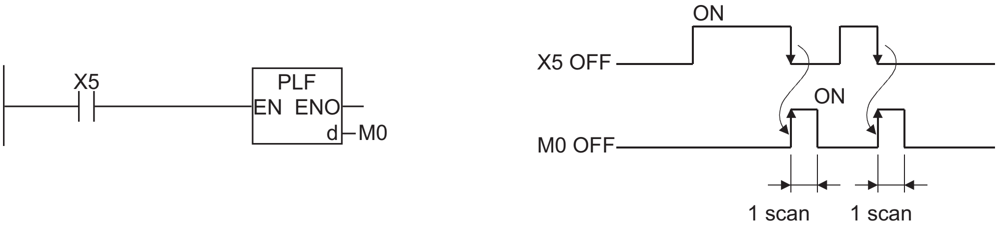
- If the switch is changed from RUN to STOP after the execution of the PLS instruction, the PLS instruction will not be executed again even if the switch is set back to RUN.


X0
PLS
EN ENO
d
M0
Operating the RUN/STOP key switch of CPU module "RUN → STOP".
Operating the RUN/STOP key switch of CPU module "STOP → RUN".
Operating the RUN/STOP key switch of CPU module "RUN → STOP".
Operating the RUN/STOP key switch of CPU module "STOP → RUN".
LD X0
PLS M0
LD X0
PLS M0
LD X0
PLS M0
END
0
END
END
0
CPU operation stop time
CPU operation stop time
ON
X0 OFF
ON
M0 OFF
1 scan of PLS M0

- When specifying a latch relay (L) for the execution command and turning the power supply from OFF to ON with the latch relay ON, the execution command turns from OFF to ON at the first scan, executing the PLS instruction and turning ON the specified device. The device turned ON at the first scan after power-ON turns OFF at the next PLS instruction.

■PLF

- Turns ON the specified device when the execution command is turned from ON to OFF, and turns OFF the device in any other case the execution command is turned from ON to OFF (OFF to OFF, OFF to ON or ON to ON).
- When there is one PLF instruction for the device specified for (d) during one scan, the specified device turns ON one scan.
- Refer to ☞ Page 72 Operation of OUT Instructions, SET/RST Instructions, or PLS/PLF Instructions Using Same Device for the operation to be performed when the PLS instruction for the same device is executed more than once during one scan.



- If the switch is changed from RUN to STOP after the execution of the PLF instruction, the PLF instruction will not be executed again even if the switch is set back to RUN.

Point

Note that the device specified for (d) may remain ON for more than one scan if the PLS or PLF instruction is jumped by the CJ instruction or if the executed subroutine program was not called by the CALL instruction.

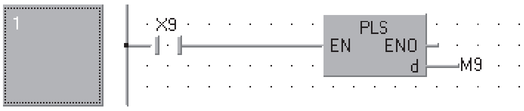
Operation error

- There is no operation error.

Program example

- In the following program, the PLS instruction is executed when X9 turns ON.

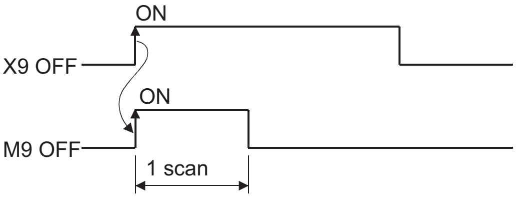
[Structured ladder/FBD]



[ST]

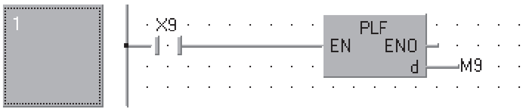
PLS(X9,M9);

[Timing chart]



- In the following program, the PLF instruction is executed when X9 turns OFF.

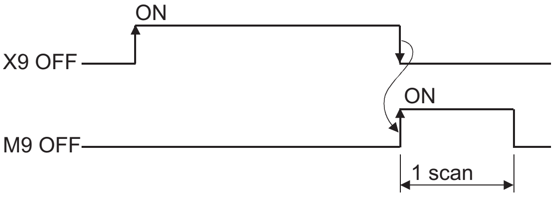
[Structured ladder/FBD]



[ST]

PLF(X9,M9);

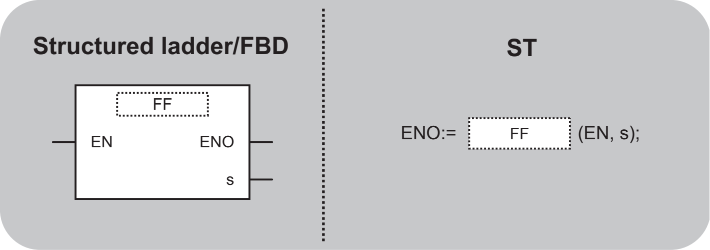
[Timing chart]



Bit device output inversion

FF

Basic | High performance | Process | Redundant | Universal | LCPU

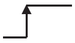


ENO:= FF (EN, s);

The following instruction can go in the dotted squares.

FF

■Executing condition

Instruction	Executing condition
FF	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Device number whose status to be inverted	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others DY
	Bit	Word		Bit	Word				
(s)	○						—		○

Processing details

- Inverts the output status of the device specified for (s) when the execution command is turned from OFF to ON.

Device	Device status	
	Prior to 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

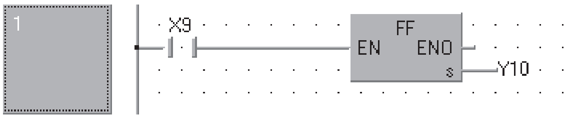
Operation error

- There is no operation error.

Program example

- In the following program, the output of Y10 is inverted when X9 turns ON.

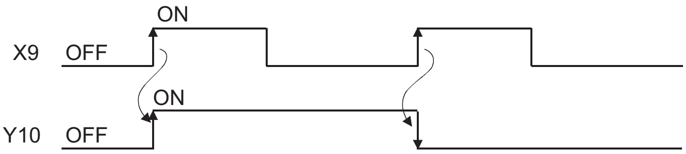
[Structured ladder/FBD]



[ST]

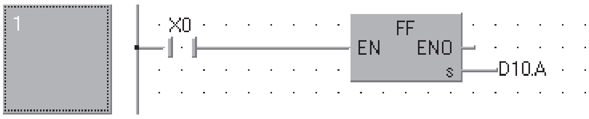
```
FF(X9,Y10);
```

[Timing chart]



- In the following program, b10 (bit 10) of D10 is inverted when X0 turns ON.

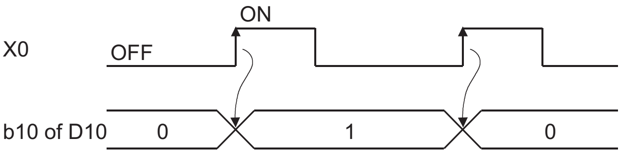
[Structured ladder/FBD]



[ST]

```
FF(X0,D10.A);
```

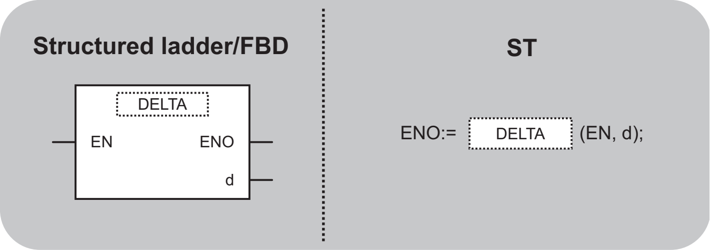
[Timing chart]



Pulse conversion of direct output

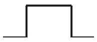
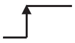
DELTA(P)

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.
DELTA, DELTAP

■Executing condition

Instruction	Executing condition
DELTA	
DELTAP	

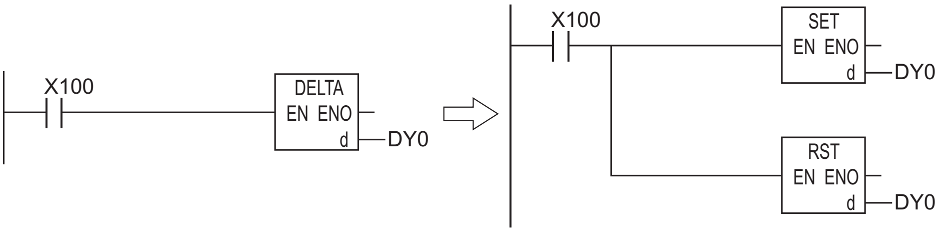
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Bit to be set as pulse output	Bit

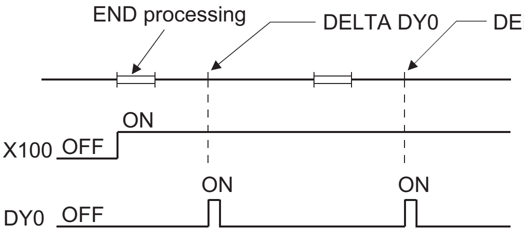
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others DY
	Bit	Word		Bit	Word				
(d)	—								○

Processing details

- Performs pulse output of direct access output (DY) specified for (d). If DY0 of the DELTA instruction has been specified, the resulting operation will be identical to the ladder shown below, which uses the SET and RST instructions.



[Operation]



- The DELTA(P) instruction is used by commands for rising edge execution for an intelligent function module.

Operation error

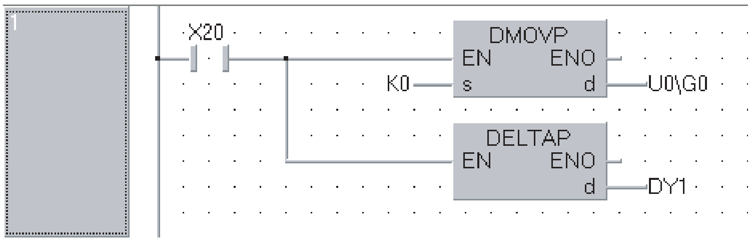
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	A specified direct access output number has exceeded the CPU module output range.	○	○	○	○	○	○

Program example

- In the following program, CH1 of the QD62 module mounted at slot 0 of the main base unit is preset, when X20 turns ON.

[Structured ladder/FBD]



Stores the preset value (0) in the buffer memory addresses (0 and 1) of the QD62.

Outputs the preset command.

[ST]

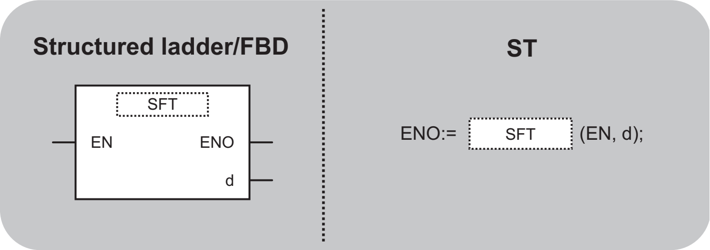
```
DMOVP(X20,K0,U0\G0);
DELTAP(X20,DY1);
```

5.4 Shift Instructions

Bit device shift

SFT(P)

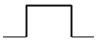
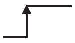
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

SFT, SFTP

■Executing condition

Instruction	Executing condition
SFT	
SFTP	

■Argument

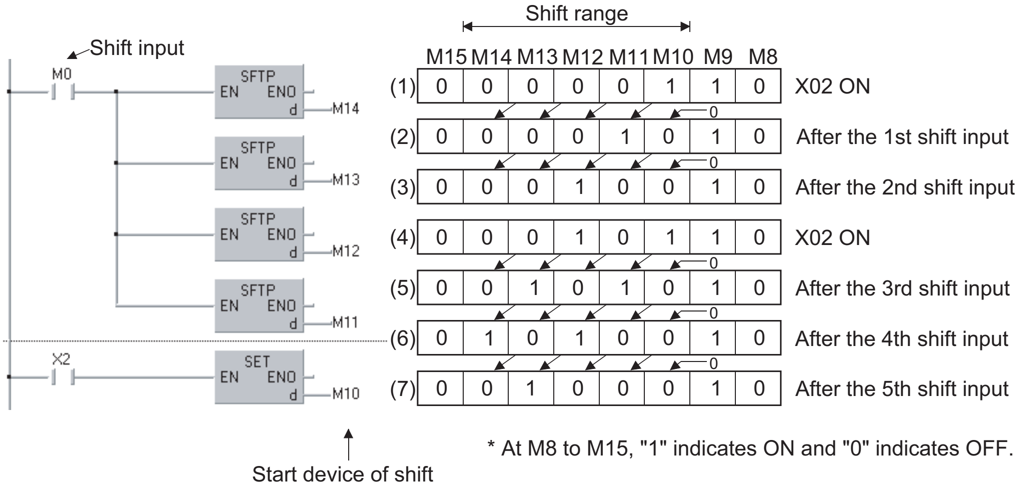
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Device number to be shifted	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others DY
	Bit	Word		Bit	Word				
(d)	○ (Other than T or C)						—		○

Processing details

■When a bit device is used

- Shifts to a device specified for (d) the ON/OFF status of the device immediately prior to the one specified for (d), and turns the prior device OFF. For example, if M11 has been specified in the SFT instruction, when the SFT instruction is executed, it will shift the ON/OFF status of M10 to M11, and turns M10 OFF.
- Turn the first device to be shifted ON with the SET instruction.
- When the SFT and SFTP instructions are to be used consecutively, the program starts from the device with the larger number.

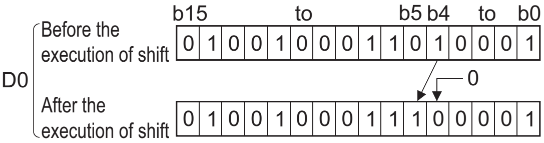


	M15	M14	M13	M12	M11	M10	M9	M8	
(1)	0	0	0	0	0	1	1	0	X02 ON
(2)	0	0	0	0	1	0	1	0	After the 1st shift input
(3)	0	0	0	1	0	0	1	0	After the 2nd shift input
(4)	0	0	0	1	0	1	1	0	X02 ON
(5)	0	0	1	0	1	0	1	0	After the 3rd shift input
(6)	0	1	0	1	0	0	1	0	After the 4th shift input
(7)	0	0	1	0	0	0	1	0	After the 5th shift input

* At M8 to M15, "1" indicates ON and "0" indicates OFF.

■When a bit-specified word device is used

- Shifts to a bit in the device specified for (d) the 1/0 status of the bit immediately prior to the one specified for (d), and turns the prior bit to 0. For example, if D0.5 (bit 5 [b5] of D0) has been specified in the SFT instruction, when the SFT instruction is executed, it will shift the 1/0 status of b4 of D0 to b5, and turns b4 to 0.



Before the execution of shift	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0	1
After the execution of shift	0	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1

Operation error

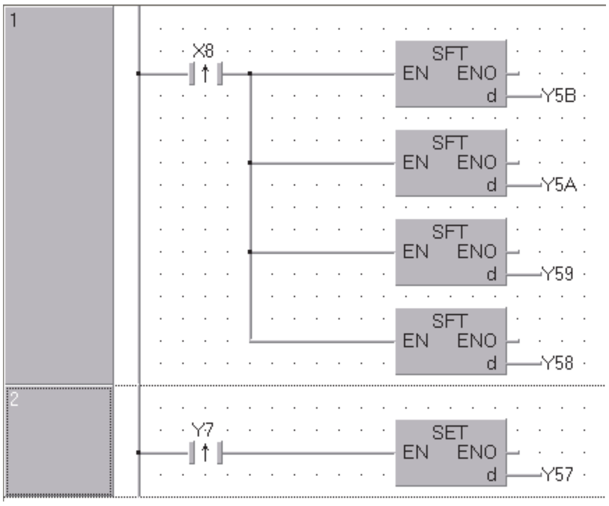
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The specified device exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the devices from Y57 to Y5B are shifted when X8 turns ON.

[Structured ladder/FBD]



Executes shift for Y57 to Y5B when X8 turns ON.

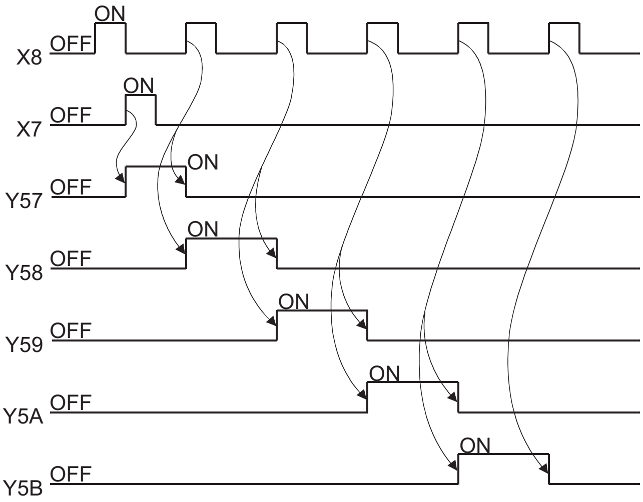
(Start programming from the device with the larger number.)

Turns Y57 ON when X7 turns ON.

[ST]

```
IF LDP(TRUE, X8) THEN
    SFT(TRUE, Y5B);
    SFT(TRUE, Y5A);
    SFT(TRUE, Y59);
    SFT(TRUE, Y58);
END_IF;
IF LDP(TRUE, Y7) THEN
    SET(TRUE, Y57);
END_IF;
```

[Timing chart]



Precautions

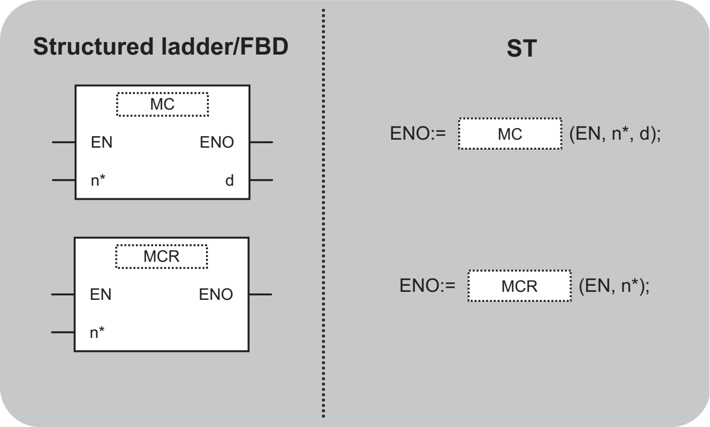
When using the SFT instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

5.5 Master Control Instructions

Setting and resetting master control

MC, MCR

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

MC, MCR

■Executing condition

Instruction	Executing condition
MC, MCR	Non-conditional execution

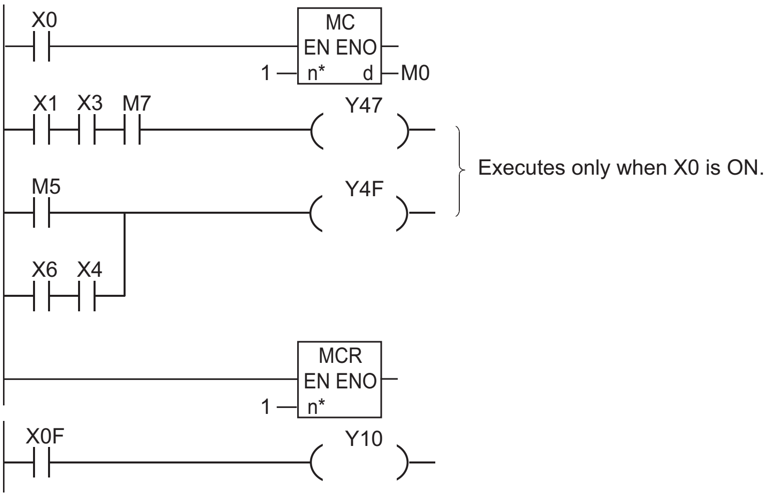
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n*	Nesting (N0 to N14)	ANY16
Output argument	ENO	Execution result	Bit
	d	Device number to be turned ON (When using the MC instruction)	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others	
	Bit	Word		Bit	Word				N	DY
n*	—						—		○	—
(d)	○						—		—	○

Processing details

- The master control instruction is used to enable the creation of highly efficient ladder switching sequence programs, through the opening and closing of a common bus for ladders.
- A ladder using the master control is as follows.



■MC

- If the execution command of the MC instruction is ON when master control is started, the result of the operation from the MC instruction to the MCR instruction will be exactly as the instruction (ladder) shows.
  If the execution command of the MC instruction is OFF, the result of the operation from the MC instruction to the MCR instruction will be as shown below.

Device	Device status
High-speed timer Low-speed timer	Count value turns to 0, coils and contacts all turn OFF.
High-speed retentive timer Low-speed retentive timer Counter	Coils turn OFF, but counter values and contacts all retain current status.
Devices in the OUT instruction	All turned OFF
Devices in the SET, RST instructions Devices in the SFT instruction Devices in the basic, application instructions	Retain current status

- Even when the MC instruction is OFF, instructions from the MC instruction to the MCR instruction will be executed, so scan time will not be shortened.

Point

When a ladder with master control contains instructions that do not require any contact instruction (such as the FOR to NEXT instruction loop, EI and DI instructions), the CPU module executes these instructions regardless of the ON/OFF status of the MC instruction execution command.

- By changing the device specified for (d), the MC instruction can use the same nesting (N) number as often as desired.
- Coils from devices specified for (d) are turned ON when the MC instruction is ON.
  Further, using these same devices with the OUT instruction or other instructions will cause them to become duplicated coils, so devices specified for (d) should not be used within other instructions.

■MCR

- This is the instruction for recovery from the master control, and indicates the end of the master control range of operation.
- Do not place contacts before the MCR instruction.
- Use the MC instruction and MCR instruction of the same nesting number as a set.

However, when the MCR instructions are nested in one place, all master controls can be terminated with the lowest nesting number. (Refer to the "Precautions for nesting" in the program example.)

Operation error

- There is no operation error.

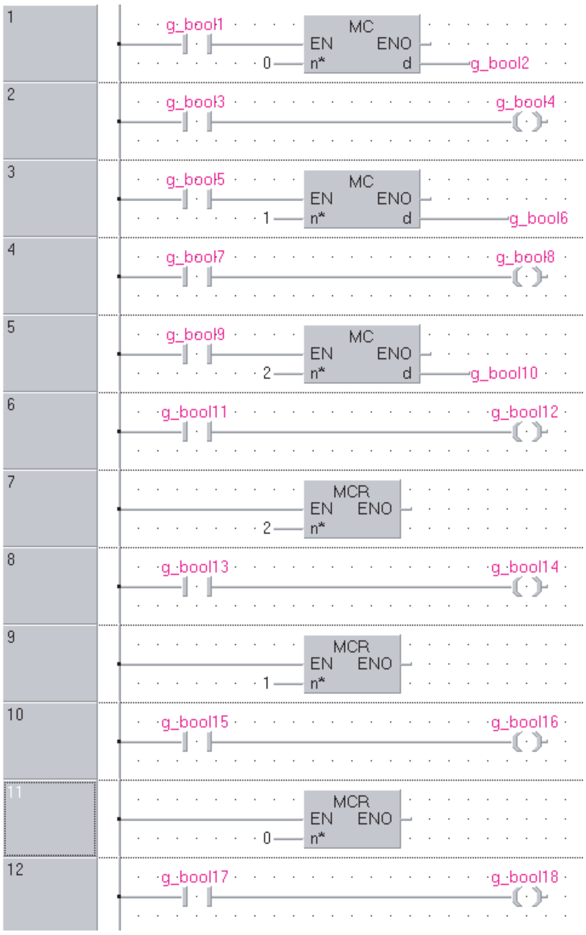
Program example

The master control instruction can be used in nesting. The different master control regions are distinguished by nesting number. Nesting can be performed from N0 to N14.

The use of nesting enables the creation of ladders which successively limit the executing condition of the program.

A ladder using nesting would appear as shown below.

[Structured ladder/FBD]



Executes when
g_bool1 is ON.

Executes when
g_bool1 and g_bool5 are ON.

Executes when g_bool1, g_bool5,
and g_bool9 are ON.

Executes when
g_bool1 and g_bool5 are ON.

Executes when
g_bool1 is ON.

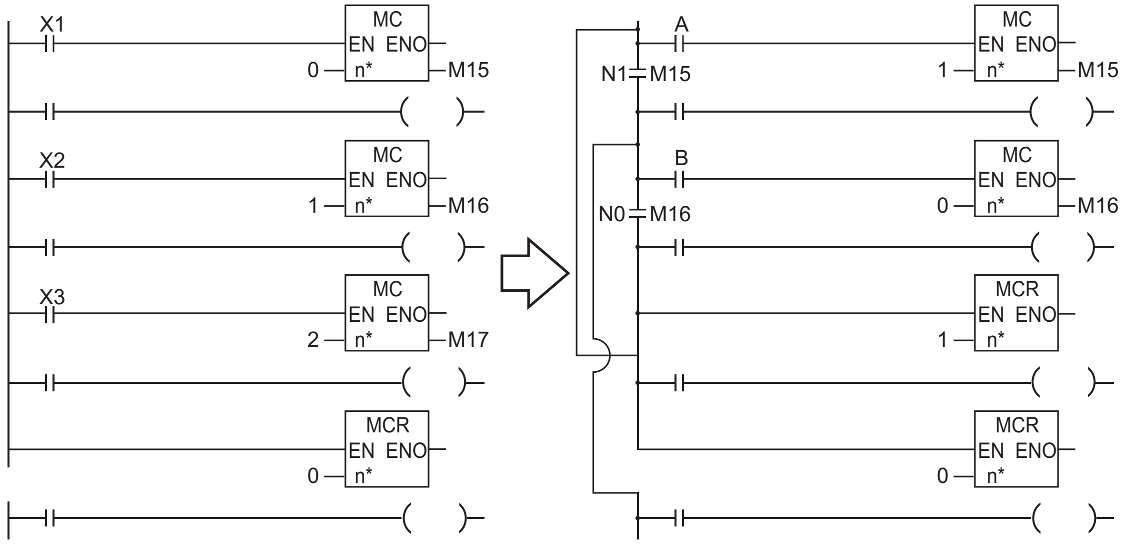
Not related to the status of
g_bool1, g_bool5, and g_bool9.

Precautions

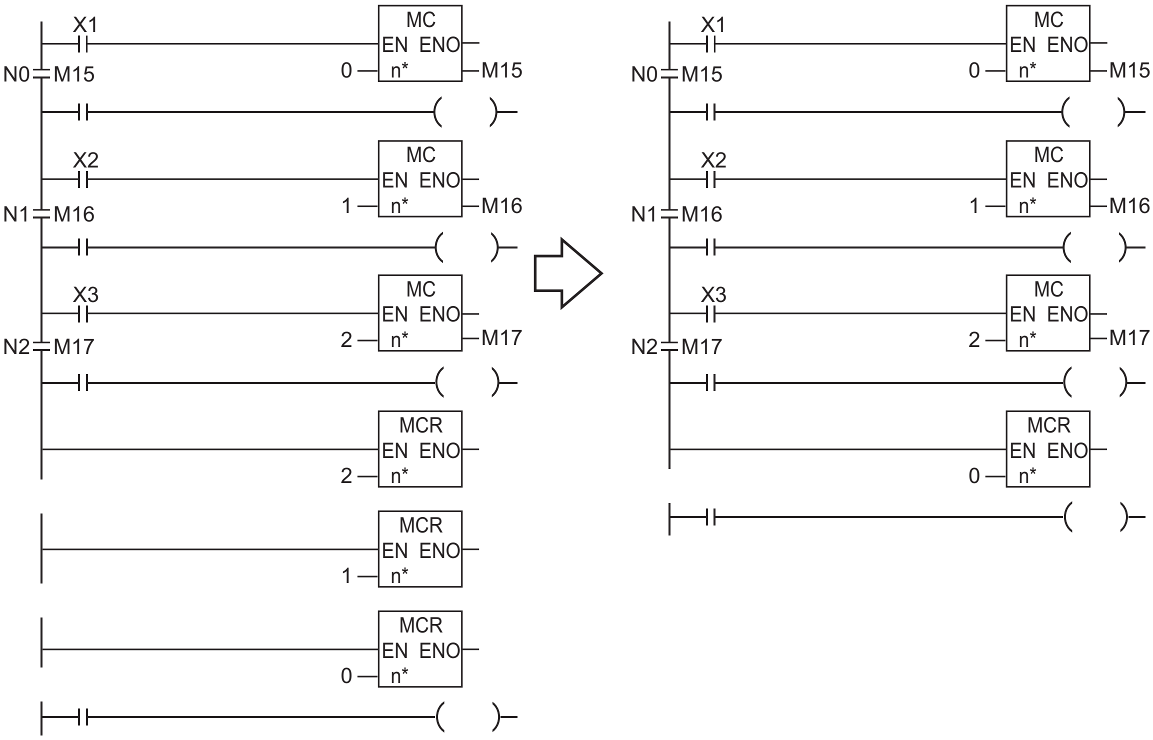
■Precautions when using nesting architecture

- Nesting can be used up to 15 times (N0 to N14)

When using nesting, nests should be inserted from the lower to higher nesting number with the MC instruction, and from the higher to the lower order with the MCR instruction. If this order is reversed, there will be no nesting architecture, and the CPU module will not be capable of performing correct operations. For example, if nesting is specified in the order N1 to N0 by the MC instruction, and also specified in the N1 to N0 order by the MCR instruction, the vertical bus will intersect and a correct master control ladder will not be produced.



- If the nesting architecture results in MCR instructions concentrated in one location, all master controls can be terminated by use of just the lowest nesting number.



■Precautions when using MC/MCR instruction in structured ladder/FBD/ST

- MC/MCR instruction is not required when using structured ladder/FBD/ST. Although, if using MC/MCR instruction, nesting number should be used in the ascending order from 0 regardless of whether MC/MCR instruction is used or not in nesting.
- The released nesting number and the following number using a MCR instruction can be used in a next MC instruction.

■Precautions for using the MC instruction

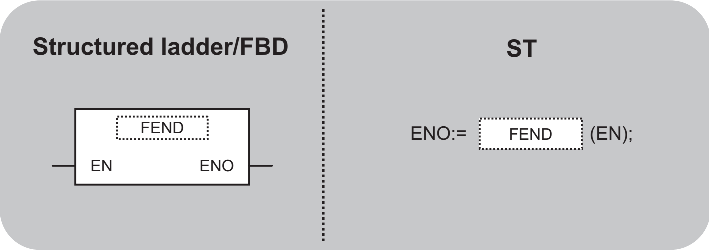
- When using the MC instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

5.6 End Instructions

Ending main routine program

FEND

Basic High performance Process Redundant Universal LCPU



The following instruction can go in the dotted squares.

FEND

■Executing condition

Instruction	Executing condition
FEND	Non-conditional execution

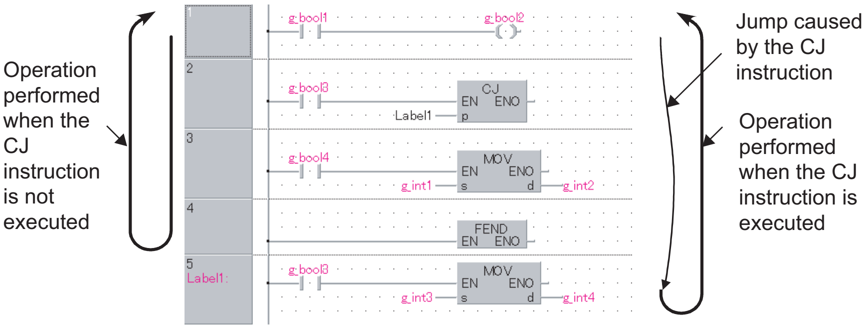
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

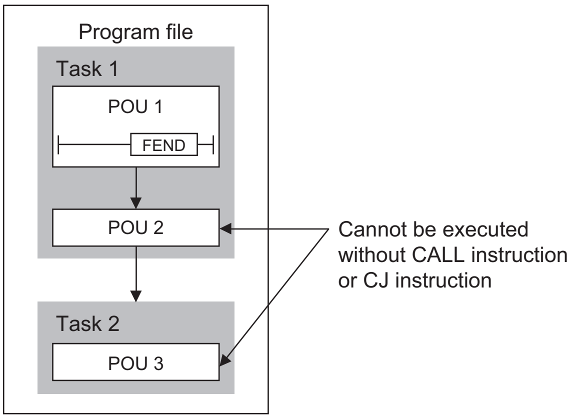
Processing details

- The FEND instruction is used in cases where the CJ instruction or other instructions are used to cause a branch in the sequence program operations.
- Execution of the FEND instruction will cause the CPU module to terminate the program it was executing.
- Even sequence programs following the FEND instruction can be displayed in ladder display using a programming tool.



Operation performed when the CJ instruction is not executed
1
g_bool1
g_bool2
2
g_bool3
CJ
EN ENO
Label1
p
3
g_bool4
MOV
EN ENO
g_int1
s d
g_int2
4
FEND
EN ENO
5
Label1:
g_bool3
MOV
EN ENO
g_int3
s d
g_int4
Jump caused by the CJ instruction
Operation performed when the CJ instruction is executed

- When the multiple tasks or multiple POUs are registered in program files, tasks or POU which are set after POUs with FEND instruction cannot be executed without the CALL instruction or CJ instruction.



Program file
Task 1
POU 1
FEND
POU 2
Task 2
POU 3
Cannot be executed without CALL instruction or CJ instruction

Operation error

- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

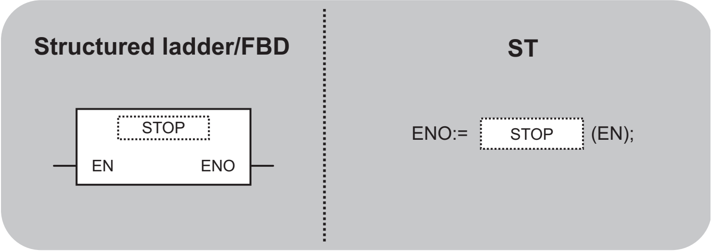
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4200	The FEND instruction is executed after the execution of the FOR instruction, and before the execution of the NEXT instruction.	○	○	○	○	○	○
4211	The FEND instruction is executed after the execution of the CALL instruction, and before the execution of the RET instruction.	○	○	○	○	○	○
4221	The FEND instruction is executed while an interrupt program, and before the execution of the IRET instruction.	○	○	○	○	○	○
4230	The FEND instruction is executed within the CHKCIR and CHKEND instruction loop.	○	○	○	○	○	○

5.7 Other Instructions

Sequence program stop

STOP

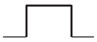
Basic | High performance | Process | Redundant | Universal | LCPU



The following instruction can go in the dotted squares.

STOP

■Executing condition

Instruction	Executing condition
STOP	

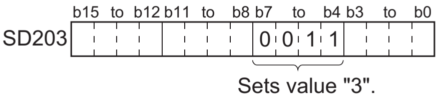
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

- Resets the output (Y) and stops the CPU module operation when the execution command is turned ON. (The same result will take place if the switch is turned to the STOP setting.)
- Execution of the STOP instruction will cause the value of b4 to b7 of the special register SD203 to become 3.



- In order to restart CPU module operations after the execution of the STOP instruction, return the switch from RUN to STOP, and back to the RUN position.

Operation error

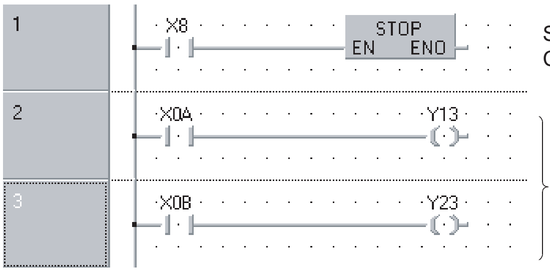
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4200	The STOP instruction was executed before the execution of the NEXT instruction and after the execution of the FOR instruction.	○	○	○	○	○	○
4211	The STOP instruction was executed before the execution of the RET instruction and after the execution of the CALL instruction.	○	○	○	○	○	○
4221	The STOP instruction was executed during an interrupt program prior to the execution of the IRET instruction.	○	○	○	○	○	○
4223	The STOP instruction was executed in the fixed scan execution program.	—	—	—	—	○	○
4230	The STOP instruction was executed within the CHKCIR to CHKEND instruction loop.	○	○	○	○	○	○

Program example

- In the following program, the CPU module is stopped when X8 turns ON.

[Structured ladder/FBD]



Stops the programmable controller CPU when X8 turns ON.

Sequence program

[ST]

```
STOP(X8);
OUT(X0A,Y13);
OUT(X0B,Y23);
```

6 BASIC INSTRUCTIONS

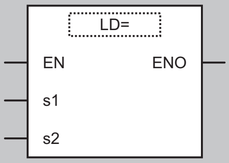
6.1 Comparison Operation Instructions

16-bit BIN data comparison

LD□, AND□, OR□

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



ST

Not supported

Any of the following instruction can go in the dotted squares.

LD=, LD<>, LD<=, LD<, LD>=, LD>, AND=, AND<>, AND<=, AND<, AND>=, AND>, OR=, OR<>, OR<=, OR<, OR>=, OR>

■Executing condition

Instruction	Executing condition
LD□, AND□, OR□	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1, s2	Comparison data, or start number of the device that stores comparison data	ANY16
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○								—
(s2)	○								—

Processing details

- Treats 16-bit BIN data from device specified for (s1) and 16-bit BIN data from device specified for (s2) as a normally open contact, and performs comparison operation.
- The results of the comparison operations for each instruction are as follows.

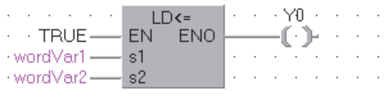
Instruction symbol	Condition	Comparison result	Condition	Comparison result
LD=, AND=, OR=	(s1)=(s2)	Conduction state	(s1)≠(s2)	Non-conduction state
LD<>, AND<>, OR<>	(s1)≠(s2)		(s1)=(s2)	
LD<=, AND<=, OR<=	(s1)≤(s2)		(s1)>(s2)	
LD<, AND<, OR<	(s1) < (s2)		(s1)≥(s2)	
LD>=, AND>=, OR>=	(s1)≥(s2)		(s1)<(s2)	
LD>, AND>, OR>	(s1)>(s2)		(s1)≤(s2)	

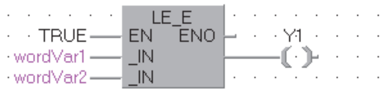
- When assigning hexadecimal constants to (s1) and (s2), and if the numeric value (8 to F) whose most significant bit (b15) is 1 is specified as a constant, the value is considered as a negative BIN value in comparison operation.
- When assigning "word [unsigned]/ bit string [16 bits]" type label to (s1) and (s2), the type is considered as word [signed] type in comparison operation. For comparison operation as "word [unsigned]/ bit string [16 bits]" type, use application functions EQ_E, NE_E, LE_E, LT_E, GE_E, GT_E. For details on the application functions, refer to the following.

📖 MELSEC-Q/L Structured Programming Manual (Application Functions)

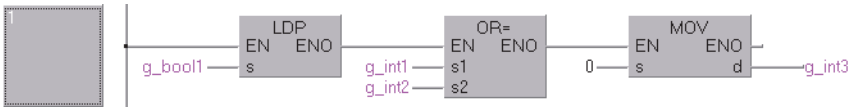
Instruction symbol	Condition	Corresponding application function
LD=, AND=, OR=	(s1)=(s2)	EQ_E
LD<>, AND<>, OR<>	(s1)≠(s2)	NE_E
LD<=, AND<=, OR<=	(s1)≤(s2)	LE_E
LD<, AND<, OR<	(s1)<(s2)	LT_E
LD>=, AND>=, OR>=	(s1)≥(s2)	GE_E
LD>, AND>, OR>	(s1)>(s2)	GT_E

Program example	Input value	Returned value

 LD<=
TRUE
EN
ENO
Y0
wordVar1
s1
wordVar2
s2
	wordVar1:= 1
	wordVar2:= 32768
		OFF
		Since 32768 of wordVar2 is considered as word [signed] type, executes comparison operation with -32768, and the output becomes OFF.

 LE_E
TRUE
EN
ENO
Y1
wordVar1
_IN
wordVar2
_IN
	wordVar1:= 1
	wordVar2:= 32768
		ON
		Using an application function executes comparison operation as "word [unsigned]/ bit string [16 bits]" type, and the output becomes ON.

- The OR=, OR<>, OR<=, OR<, OR>=, or OR> instruction performs OR operation of operation result between "(s1), (s2)" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400. When using the OR=, OR<>, OR<=, OR<, OR>=, or OR> instruction, connect EN and ENO in series as shown below.



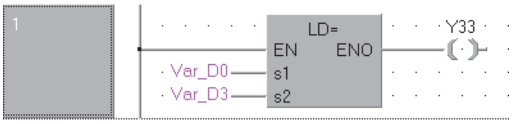
Operation error

- There is no operation error.

Program example

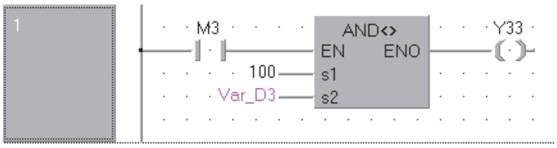
- In the following program, the value in Var_D0 is compared with the value in Var_D3, and Y33 turns ON when the values are identical.

[Structured ladder/FBD]



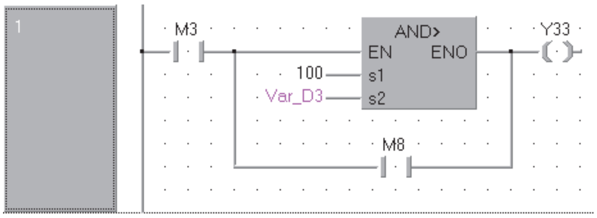
- In the following program, the BIN value of 100 is compared with the value in Var_D3 when M3 is ON, and Y33 turns ON when the value in Var_D3 is other than 100.

[Structured ladder/FBD]



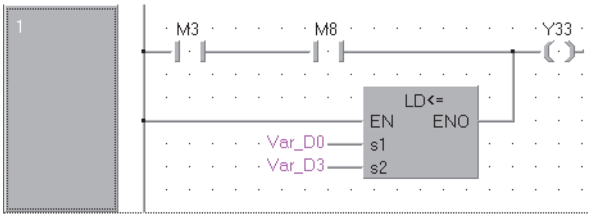
- In the following program, the BIN value of 100 is compared with the value in Var_D3 when M3 turns ON, and Y33 turns ON when the value in Var_D3 is less than 100 or when M8 is ON.

[Structured ladder/FBD]



- In the following program, the value in Var_D0 is compared with the value in Var_D3, and Y33 turns ON when the value in Var_D3 is equal or greater than the value in Var_D0 or when both M3 and M8 are ON.

[Structured ladder/FBD]

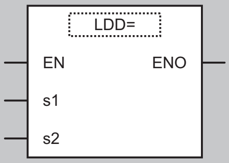


32-bit BIN data comparison

LDD□, ANDD□, ORD□

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



ST

Not supported

The following instruction can go in the dotted squares.

LDD=, LDD<>, LDD<=, LDD<, LDD>=, LDD>, ANDD=, ANDD<>, ANDD<=, ANDD<, ANDD>=, ANDD>, ORD=, ORD<>, ORD<=, ORD<, ORD>=, ORD>

■Executing condition

Instruction	Executing condition
LDD□, ANDD□, ORD□	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1, s2	Comparison data, or start number of the device that stores comparison data	ANY32
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○								—
(s2)	○								—

Processing details

- Treats 32-bit BIN data from device specified for (s1) and 32-bit BIN data from device specified for (s2) as a normally open contact, and performs comparison operation.
- The results of the comparison operations for each instruction are as follows.

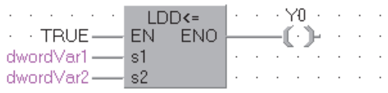
Instruction symbol	Condition	Comparison result	Condition	Comparison result
LDD=, ANDD=, ORD=	(s1)=(s2)	Conduction state	(s1)≠(s2)	Non-conduction state
LDD<>, ANDD<>, ORD<>	(s1)≠(s2)		(s1)=(s2)	
LDD<=, ANDD<=, ORD<=	(s1)≤(s2)		(s1)>(s2)	
LDD<, ANDD<, ORD<	(s1)<(s2)		(s1)≥(s2)	
LDD>=, ANDD>=, ORD>=	(s1)≥(s2)		(s1)<(s2)	
LDD>, ANDD>, ORD>	(s1)>(s2)		(s1)≤(s2)	

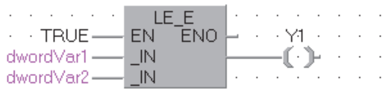
- When assigning hexadecimal constants to (s1) and (s2), and if the numeric value (8 to F) whose most significant bit (b31) is 1 is specified as a constant, the value is considered as a negative BIN value in comparison operation.
- Data used for comparison should be specified by a 32-bit instruction (such as DMOV instruction). If specification is made with a 16-bit instruction (such as MOV instruction), comparisons of large and small values cannot be performed correctly.
- When assigning "double word [unsigned]/ bit string [32 bits]" type label to (s1) and (s2), the type is considered as double word [signed] type in comparison operation. For comparison operation as "double word [unsigned]/ bit string [32 bits]" type, use application functions EQ_E, NE_E, LE_E, LT_E, GE_E, GT_E. For details on the application functions, refer to the following.

📖 MELSEC-Q/L Structured Programming Manual (Application Functions)

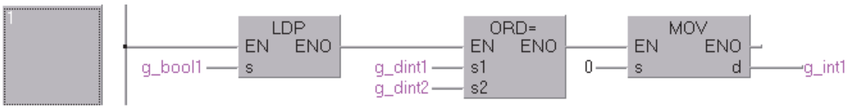
Instruction symbol	Condition	Corresponding application function
LDD=, ANDD=, ORD=	(s1)=(s2)	EQ_E
LDD<>, ANDD<>, ORD<>	(s1)≠(s2)	NE_E
LDD<=, ANDD<=, ORD<=	(s1)≤(s2)	LE_E
LDD<, ANDD<, ORD<	(s1)<(s2)	LT_E
LDD>=, ANDD>=, ORD>=	(s1)≥(s2)	GE_E
LDD>, ANDD>, ORD>	(s1)>(s2)	GT_E

Program example	Input value	Returned value

	dwordVar1 := 1
	dwordVar2 := 2147483648
		OFF
		Since 2147483648 of dwordVar2 is considered as double word [signed] type, executes comparison operation with -2147483648, and the output becomes OFF.

	dwordVar1 := 1
	dwordVar2 := 2147483648
		ON
		Using an application function executes comparison operation as "double word [unsigned]/ bit string [32 bits]" type, and the output becomes ON.

- The ORD=, ORD<>, ORD<=, ORD<, ORD>=, or ORD> instruction performs OR operation of operation result between "(s1), (s2)" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400. When using the ORD=, ORD<>, ORD<=, ORD<, ORD>=, or ORD> instruction, connect EN and ENO in series as shown below.



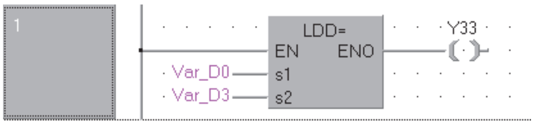
Operation error

- There is no operation error.

Program example

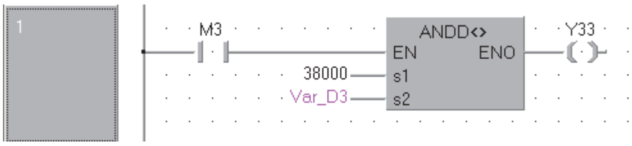
- In the following program, the value in Var_D0 is compared with the value in Var_D3, and Y33 turns ON when the values are identical.

[Structured ladder/FBD]



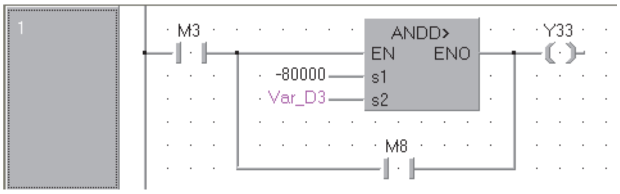
- In the following program, the BIN value of 38000 is compared with the value in Var_D3 when M3 is ON, and Y33 turns ON when the value in Var_D3 is other than 38000.

[Structured ladder/FBD]



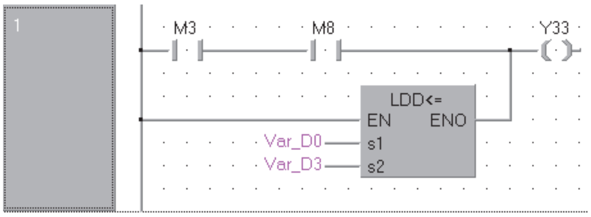
- In the following program, the BIN value of -80000 is compared with the value in Var_D3 when M3 is ON, and Y33 turns ON when the value in Var_D3 is less than -80000, or when M8 is ON.

[Structured ladder/FBD]



- In the following program, the value in Var_D0 is compared with the value in Var_D3, and Y33 turns ON when the value in Var_D3 is equal or greater than the value in Var_D0 or when both M3 and M8 are ON.

[Structured ladder/FBD]



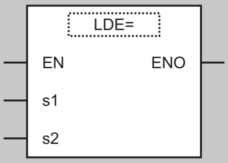
Floating-point data comparison (single precision)

LDE□, ANDE□, ORE□

Ver. Basic | High performance | Process | Redundant | Universal | LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

Structured ladder/FBD



ST

Not supported

Any of the following instruction can go in the dotted squares.

LDE=, LDE<>, LDE<=, LDE<, LDE>=, LDE>, ANDE=, ANDE<>, ANDE<=, ANDE<, ANDE>=, ANDE>, ORE=, ORE<>, ORE<=, ORE<, ORE>=, ORE>

■Executing condition

Instruction	Executing condition
LDE□, ANDE□, ORE□	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1, s2	Comparison data, or start number of the device that stores comparison data	Single-precision real
Output argument	ENO	Execution result	Bit

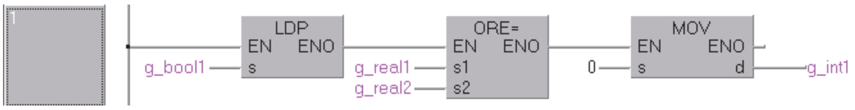
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—	○		—	○	—
(s2)	—	○		—	○		—	○	—

Processing details

- Treats the floating-point data from device specified for (s1) and the floating-point data from device specified for (s2) as a normally open contact, and performs comparison operation.
- The results of the comparison operations for each instruction are as follows.

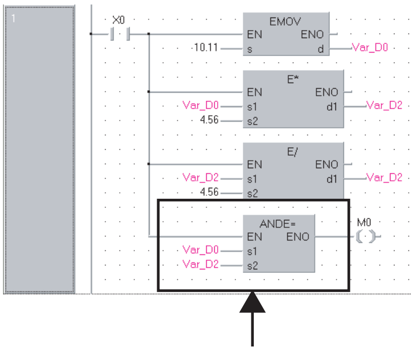
Instruction symbol	Condition	Comparison result	Condition	Comparison result
LDE=, ANDE=, ORE=	(s1)=(s2)	Conduction state	(s1)≠(s2)	Non-conduction state
LDE<>, ANDE<>, ORE<>	(s1)≠(s2)		(s1)=(s2)	
LDE<=, ANDE<=, ORE<=	(s1)≤(s2)		(s1)>(s2)	
LDE<, ANDE<, ORE<	(s1)<(s2)		(s1)≥(s2)	
LDE>=, ANDE>=, ORE>=	(s1)≥(s2)		(s1)<(s2)	
LDE>, ANDE>, ORE>	(s1)>(s2)		(s1)≤(s2)	

- The ORE=, ORE<>, ORE<=, ORE<, ORE>=, or ORE> instruction performs OR operation of operation result between "(s1), (s2)" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400. When using the ORE=, ORE<>, ORE<=, ORE<, ORE>=, or ORE> instruction, connect EN and ENO in series as shown below.



Point

Note that when using the □E= instruction, the values of operation result may not be equal depending on the error.



Var_D0 and Var_D2 may not be equal.

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

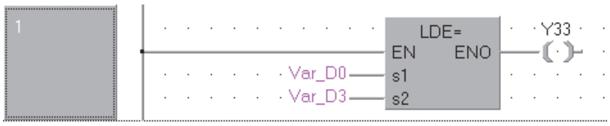
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of the specified device is -0.[*1]	○	○	○	○	—	—
4140	The value of the specified device is outside the following range. 0, $2^{-126} \leq$ | value of the specified device | $< 2^{128}$ The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○

[*1] There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Program example

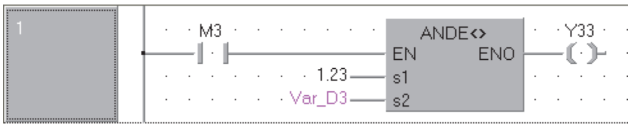
- In the following program, the value in Var_D0 is compared with the value in Var_D3, and Y33 turns ON when the values are identical.

[Structured ladder/FBD]



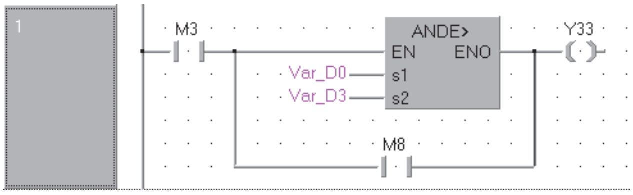
- In the following program, the floating-point real number 1.23 is compared with the value in Var_D3 when M3 is ON, and Y33 turns ON when the value in Var_D3 is other than 1.23.

[Structured ladder/FBD]



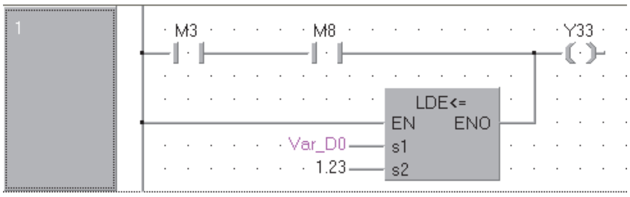
- In the following program, the value in Var_D0 is compared with the value in Var_D3 when M3 is ON, and Y33 turns ON when the value in Var_D3 is less than the value in Var_D0 or when M8 is ON.

[Structured ladder/FBD]



- In the following program, the value in Var_D0 is compared with the floating-point real number 1.23, and Y33 turns ON when 1.23 is equal to or greater than the value in Var_D0 or when both M3 and M8 are ON.

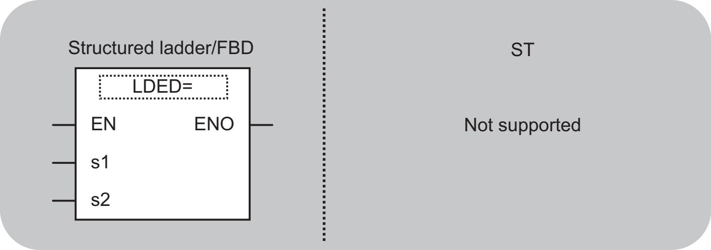
[Structured ladder/FBD]



Floating-point data comparison (double precision)

LDED□, ANDED□, ORED□

✗ Basic ✗ High performance ✗ Process ✗ Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

LDED=, LDED<>, LDED<=, LDED<, LDED>=, LDED>, ANDED=, ANDED<>, ANDED<=, ANDED<, ANDED>=, ANDED>, ORED=, ORED<>, ORED<=, ORED<, ORED>=, ORED>

■Executing condition

Instruction	Executing condition
LDED□, ANDED□, ORED□	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1, s2	Comparison data, or start number of the device that stores comparison data	Double-precision real
Output argument	ENO	Execution result	Bit

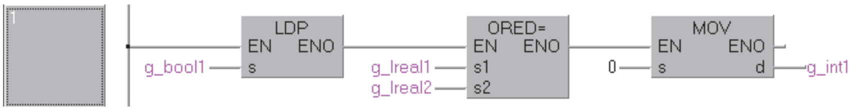
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				○	—
(s2)	—	○		—				○	—

Processing details

- Treats the 64-bit floating-point real number specified for (s1) and the 64-bit floating-point real number specified for (s2) as a normally open contact, and performs comparison operation.
- The results of the comparison operations for each instruction are as follows.

Instruction symbol	Condition	Comparison result	Condition	Comparison result
LDED=, ANDED=, ORED=	(s1)=(s2)	Conduction state	(s1)≠(s2)	Non-conduction state
LDED<>, ANDED<>, ORED<>	(s1)≠(s2)		(s1)=(s2)	
LDED<=, ANDED<=, ORED<=	(s1)≤(s2)		(s1)>(s2)	
LDED<, ANDED<, ORED<	(s1)<(s2)		(s1)≥(s2)	
LDED>=, ANDED>=, ORED>=	(s1)≥(s2)		(s1)<(s2)	
LDED>, ANDED>, ORED>	(s1)>(s2)		(s1)≤(s2)	

- The ORED=, ORED<>, ORED<=, ORED<, ORED>=, or ORED> instruction performs OR operation of operation result between "(s1), (s2)" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400. When using the ORED=, ORED<>, ORED<=, ORED<, ORDE>=, or ORED> instruction, connect EN and ENO in series as shown below.



- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

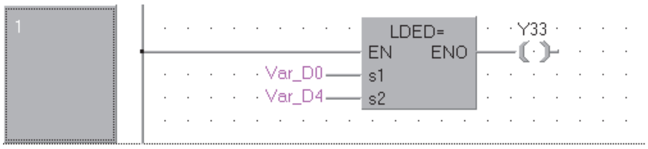
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | $< 2^{1024}$ The value of the specified device is -0.	—	—	—	—	○	○

Program example

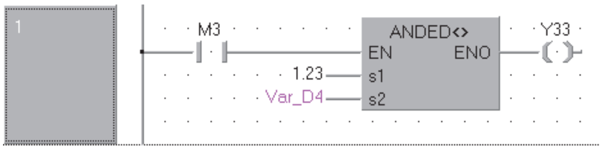
- In the following program, the value in Var_D0 is compared with the value in Var_D4, and Y33 turns ON when the value in Var_D0 and the value in Var_D4 are matched.

[Structured ladder/FBD]



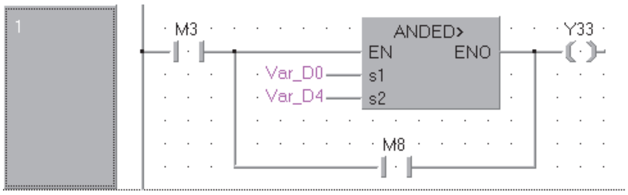
- In the following program, the floating-point real number 1.23 is compared with the value in Var_D4 when M3 is ON, and Y33 turns ON when the value in Var_D4 is other than 1.23.

[Structured ladder/FBD]



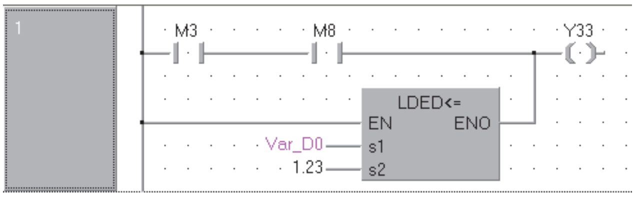
- In the following program, the value in Var_D0 is compared with the value in Var_D4, and Y33 turns ON when the value in Var_D4 is smaller than the value in Var_D0 or when M8 is ON.

[Structured ladder/FBD]



- In the following program, the value in Var_D0 is compared with the floating-point real number 1.23, and Y33 turns ON when 1.23 is equal to or greater than the value in Var_D0 or when both M3 and M8 are ON.

[Structured ladder/FBD]

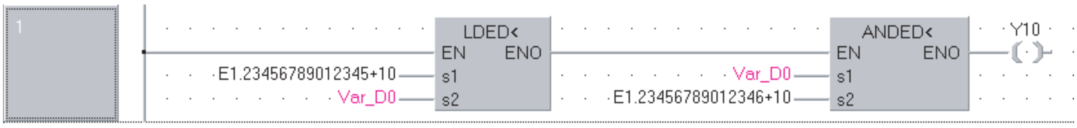


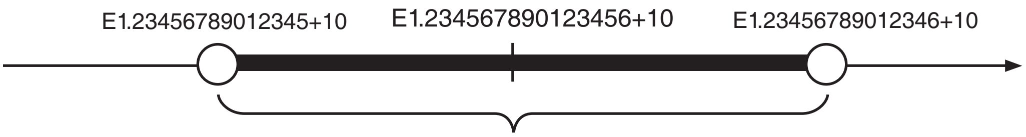
Precautions

- Since the maximum number of digits of real number that can be input by the programming tool is 15 digits, the comparison with the real number whose number of significant figures is 16 or more cannot be made by the instruction shown in this section. When judging match/mismatch with the real number whose significant figures is 16 or more by the instruction in this section, compare it with the approximate values of the real number to be compared and judge by the sizes.

Ex.

When judging the match of E1.234567890123456+10 (Number of significant figures is 16) and the double-precision floating-point data.

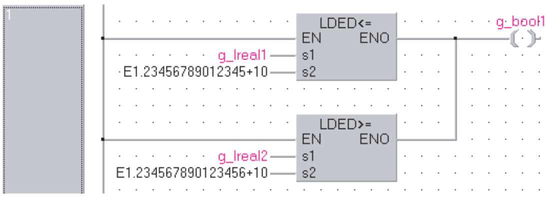


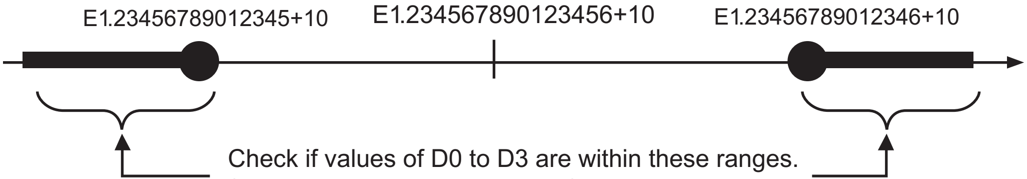


Check if values of D0 to D3 are within this range. (Boundary values are not included.)

Ex.

Judging the mismatch of E1.234567890123456+10 (Number of significant figures is 16) and the double-precision floating-point data.





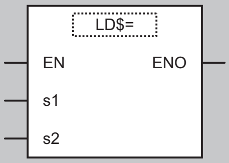
Check if values of D0 to D3 are within these ranges.
(Boundary values are included.)

Character string data comparison

LD$□, AND$□, OR$□

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



ST

Not supported

Any of the following instruction can go in the dotted squares.

LD$=, LD$<>, LD$<=, LD$<, LD$>=, LD$>, AND$=, AND$<>, AND$<=, AND$<, AND$>=, AND$>, OR$=, OR$<>, OR$<=, OR$<, OR$>=, OR$>

■Executing condition

Instruction	Executing condition
LD$□, AND$□, OR$□	Non-conditional execution

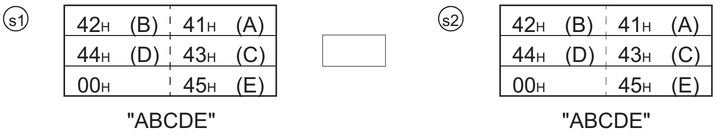
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1, s2	Comparison data, or start number of the device that stores comparison data	String
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				○	—
(s2)	—	○		—				○	—

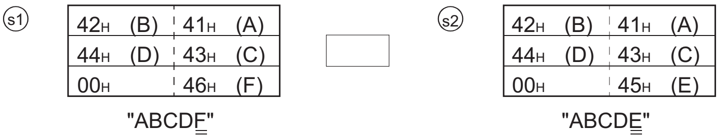
Processing details

- Treats the character string data specified for (s1) and the character string data specified for (s2) as a normally open contact, and performs comparison operation.
- A comparison operation involves the character-by-character comparison of the ASCII code from the first character in the character string.
- The character string data of (s1) and (s2) for the comparison refers to the data up to 00H.
- If all character string data are matched, the comparison result will be matched.



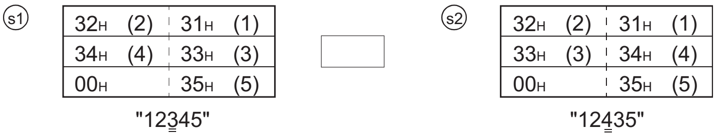
Instruction symbol	Comparison result	Instruction symbol	Comparison result
LD$=, AND$=, OR$=	Conduction state	LD$<, AND$<, OR$<	Non-conduction state
LD$<>, AND$<>, OR$<>	Non-conduction state	LD$>=, AND$>=, OR$>=	Conduction state
LD$<=, AND$<=, OR$<=	Conduction state	LD$>, AND$>, OR$>	Non-conduction state

- If the character string data are different, the character string with the larger character code will be larger.



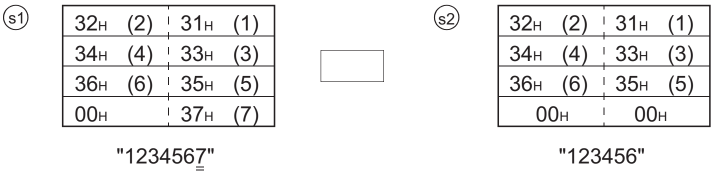
Instruction symbol	Comparison result	Instruction symbol	Comparison result
LD$=, AND$=, OR$=	Non-conduction state	LD$<, AND$<, OR$<	Non-conduction state
LD$<>, AND$<>, OR$<>	Conduction state	LD$>=, AND$>=, OR$>=	Conduction state
LD$<=, AND$<=, OR$<=	Non-conduction state	LD$>, AND$>, OR$>	Conduction state

- If the character string data are different, the first different sized character code will determine whether the character string is larger or smaller.



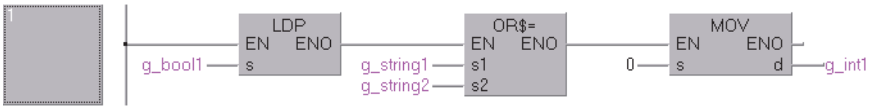
Instruction symbol	Comparison result	Instruction symbol	Comparison result
LD$=, AND$=, OR$=	Non-conduction state	LD$<, AND$<, OR$<	Conduction state
LD$<>, AND$<>, OR$<>	Conduction state	LD$>=, AND$>=, OR$>=	Non-conduction state
LD$<=, AND$<=, OR$<=	Conduction state	LD$>, AND$>, OR$>	Non-conduction state

- If the character string data specified for (s1) and (s2) are of different lengths, the data with the longer character string will be larger.



Instruction symbol	Comparison result	Instruction symbol	Comparison result
LD$=, AND$=, OR$=	Non-conduction state	LD$<, AND$<, OR$<	Non-conduction state
LD$<>, AND$<>, OR$<>	Conduction state	LD$>=, AND$>=, OR$>=	Conduction state
LD$<=, AND$<=, OR$<=	Non-conduction state	LD$>, AND$>, OR$>	Conduction state

- The OR$=, OR$<>, OR$<=, OR$<, OR$>=, or OR$> instruction performs OR operation of operation result between "(s1), (s2)" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400. When using the OR$=, OR$<>, OR$<=, OR$<, OR$>=, or OR$> instruction, connect EN and ENO in series as shown below.



Operation error

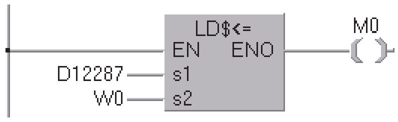
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The code 00H does not exist within the corresponding device range, starting from the device number specified for (s1) and (s2). The character string of (s1) and (s2) exceeds 16383 characters.	—	○	○	○	○	○

Point

The character string data comparison operation instruction checks the device range while comparing the character string data. For this reason, if the 00H code does not exist in the corresponding device range, the instruction outputs the comparison operation result instead of returning an operation error when mismatch of characters is detected.

Example



Data of (s1)

D12287	"B"	"A"
W0	00H	"C"

Data of (s2)

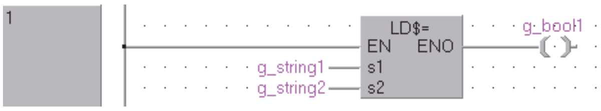
D10	"Z"	"A"
D11	00H	"C"

If (s1) and (s2) data are as shown above, the second character of (s1) does not match with that of (s2), and the comparison result is expressed as (s1) ≠ (s2) (the operation result is "non-conduction"). Though the 00H code is not included within the (s1) device range, no operation error is returned, because the mismatch is detected at D12287, which is within the device range.

Program example

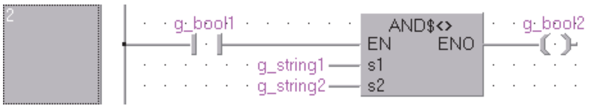
- In the following program, the character string data stored in g_string1 is compared with the character string data stored in g_string2, and g_bool1 turns ON when they are matched.

[Structured ladder/FBD]



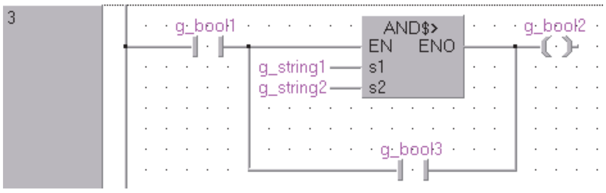
- In the following program, the character string g_string1 is compared with the character string data stored in g_string2 and when g_bool1 is ON, and g_bool2 turns ON when the character string data stored in g_string2 is other than g_string1.

[Structured ladder/FBD]



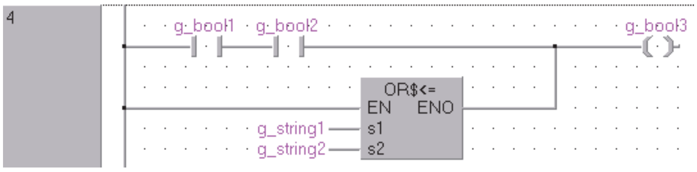
- In the following program, the character string data stored in g_string1 is compared with the character string data stored in g_string2 when g_bool1 is ON, and g_bool2 turns ON when the character string data stored in g_string2 is equal to or shorter than the character string data stored in g_string1, or when g_bool3 is ON.

[Structured ladder/FBD]



- In the following program, the character string data stored in g_string1 is compared with the character string g_string2, and g_bool3 turns ON when g_string2 is equal to or longer than the character string data stored in g_string1 or when both g_bool1 and g_bool2 are ON.

[Structured ladder/FBD]

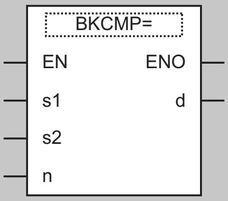


16-bit BIN block data comparison

BKCMP□(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



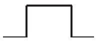
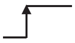
ST

Not supported

Any of the following instruction can go in the dotted squares.

BKCMP=, BKCMP<>, BKCMP<=, BKCMP<, BKCMP>=, BKCMP>, BKCMP=P, BKCMP<>P, BKCMP<=P, BKCMP<P, BKCMP>=P, BKCMP>P

■Executing condition

Instruction	Executing condition
□BKCMP	
BKCMP□P	

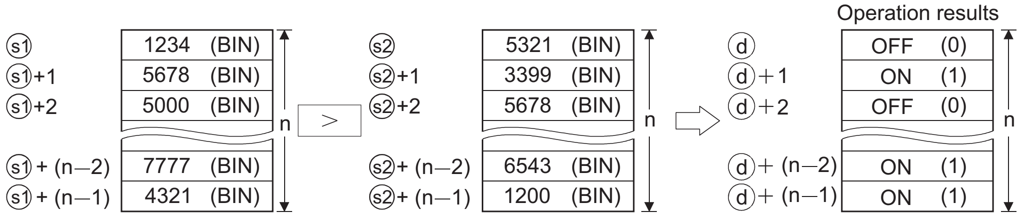
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Data to be compared, or start number of the device that stores data to be compared	ANY16
	s2	Start number of the device that stores comparison data	ANY16
	n	Number of data to be compared	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Bit

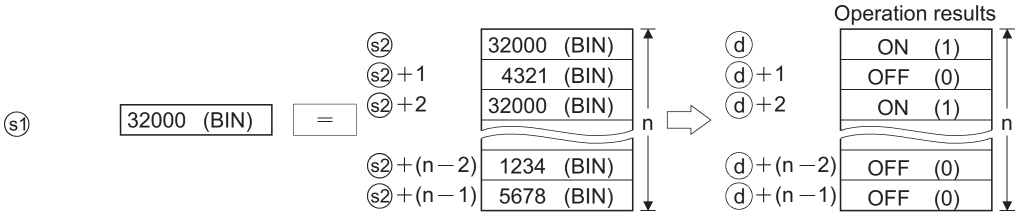
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				○	—
(s2)	—	○		—				—	—
n	○	○		○				○	—
(d)	○	○		—				—	—

Processing details

- Compares the 16-bit BIN data n points from the device number specified for (s1) with the 16-bit BIN data n points from the device number specified for (s2), and stores the operation result to the device specified for (d) and the following devices.
  - If the comparison condition has been met, the corresponding device of (d) turns ON.
  - If the comparison condition has not been met, the corresponding device of (d) turns OFF.



- The comparison operation is performed in units of 16 bits.
- The constant can be specified between -32768 and 32767 (BIN 16 bits) for (s1).



- The results of the comparison operations for each instruction are as follows.

Instruction symbol	Condition	Comparison result	Condition	Comparison result
BKCMP=(P)	(s1)=(s2)	ON (1)	(s1)≠(s2)	OFF (0)
BKCMP<>	(s1)≠(s2)		(s1)=(s2)	
BKCMP<=(P)	(s1)≤(s2)		(s1)>(s2)	
BKCMP<(P)	(s1)<(s2)		(s1)≥(s2)	
BKCMP>=(P)	(s1)≥(s2)		(s1)<(s2)	
BKCMP>(P)	(s1)>(s2)		(s1)≤(s2)	

- If all comparison results stored to devices n points from (d) are ON (1), SM704 (block comparison signal) turns ON.

Operation error

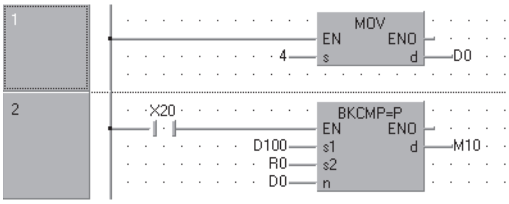
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of the n points from the device specified for (s1), (s2) or (d) exceeds the corresponding device. The range of n points from the device specified for (s1) overlaps with the range of n points from the device specified for (d). The range of n points from the device specified for (s2) overlaps with the range of n points from the device specified for (d).	○	○	○	○	○	○

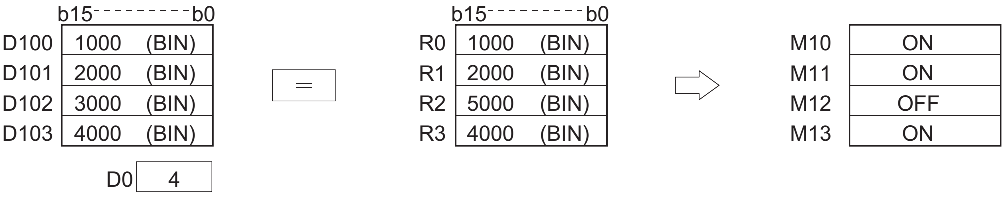
Program example

- In the following program, the data stored in D100 to D103 is compared with the data stored in R0 to R3 when X20 turns ON, and the result is stored to M10 and the following devices.

[Structured ladder/FBD]

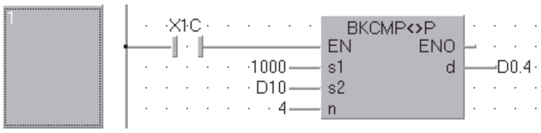


[Operation]

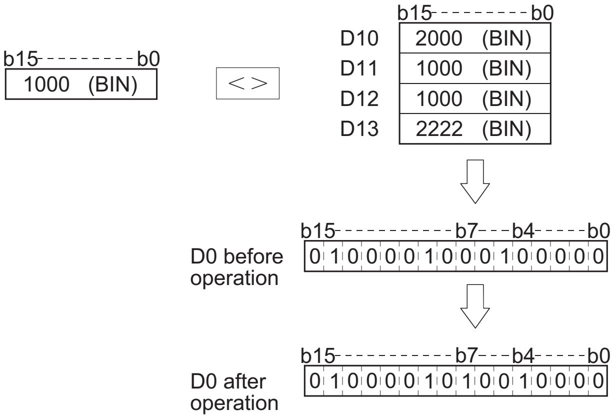


- In the following program, the constant K1000 is compared with the data stored in D10 to D13 when X1C turns ON, and the result is stored to b4 to b7 of D0.

[Structured ladder/FBD]

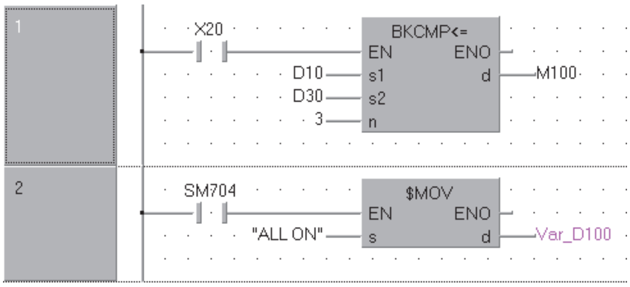


[Operation]

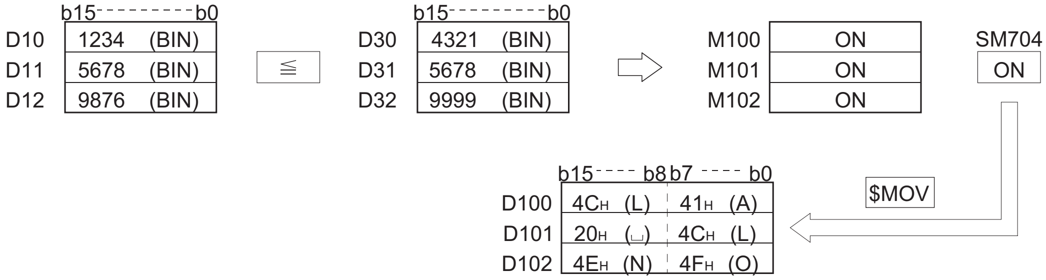


- In the following program, the data in D10 to D12 are compared with the data in D30 to D32 when X20 turns ON, and the result is stored to M100 and the following devices. The character string "ALL ON" is transferred to Var_D100 and the following variables when all devices of M100 and the following turn to 1 (ON status).

[Structured ladder/FBD]



[Operation]

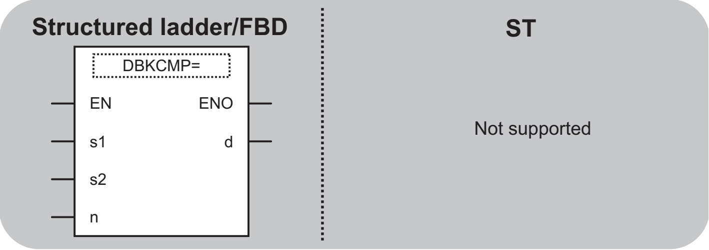


32-bit BIN block data comparison

DBKCMP□(P)

Basic High performance Process Redundant Ver. Universal LCPU

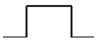
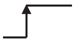
- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported



Any of the following instruction can go in the dotted squares.

DBKCMP=, DBKCMP<>, DBKCMP<=, DBKCMP<, DBKCMP>=, DBKCMP>, DBKCMP=P, DBKCMP<>P, DBKCMP<=P, DBKCMP<P, DBKCMP>=P, DBKCMP>P

■Executing condition

Instruction	Executing condition
DBKCMP□	
DBKCMP□P	

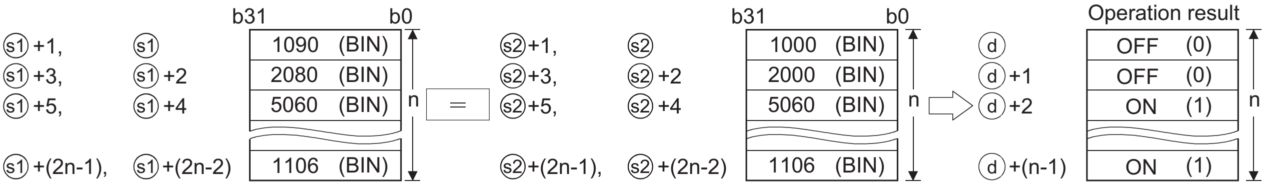
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Data to be compared, or start number of the device that stores data to be compared	ANY32
	s2	Start number of the device that stores comparison data	ANY32
	n	Number of data to be compared	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Bit

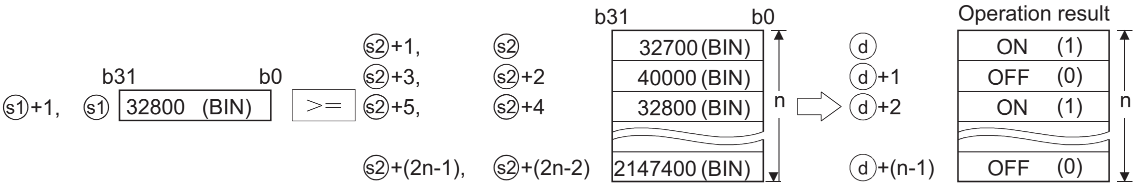
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	—	○	○	—				○	—
(s2)	—	○	○	—				—	—
n	—	○	○	○				○	—
(d)	○	—	○	—				—	—

Processing details

- Compares the 32-bit BIN data n points from the device number specified for (s1) with the 32-bit BIN data n points from the device number specified for (s2), and stores the operation result to the device specified for (d) and the following devices.
  - If the comparison condition has been met, the corresponding device of (d) turns ON.
  - If the comparison condition has not been met, the corresponding device of (d) turns OFF.



- The comparison operation is performed in units of 32 bits.
- The constant can be specified between -2147483648 and 2147483647 (BIN 32 bits) for (s1).



- (d) is specified in the device range other than n points from (s1) and n points from (s2).
- The results of the comparison operations for each instruction are as follows.

Instruction symbol	Condition	Comparison result	Condition	Comparison result
DBKCMP=(P)	(s1)=(s2)	ON (1)	(s1)≠(s2)	OFF (0)
DBKCMP<>(P)	(s1)≠(s2)		(s1)=(s2)	
DBKCMP<=(P)	(s1)≤(s2)		(s1)>(s2)	
DBKCMP<(P)	(s1)<(s2)		(s1)≥(s2)	
DBKCMP>=(P)	(s1)≥(s2)		(s1)<(s2)	
DBKCMP>(P)	(s1)>(s2)		(s1)≤(s2)	

- If all comparison results stored to devices n points from (d) are ON (1), or any of the result is OFF (0), the following special relays turn ON/OFF according to the condition. For the standby type programs, special relays based on the call source program are turned ON/OFF.

No.	Special relay	All comparison results are ON (1)			Any of the comparison result is OFF (0)		
		Initial execution/ scan	Interrupt (other than I45)/fixed scan execution	Interrupt (I45)	Initial execution/ scan	Interrupt (other than I45)/fixed scan execution	Interrupt (I45)
1	SM704	ON	ON	ON	OFF	OFF	OFF
2	SM716	ON	—	—	OFF	—	—
3	SM717	—	ON	—	—	OFF	—
4	SM718	—	—	ON	—	—	OFF

- No processing is performed if the value specified for n is 0.

Operation error

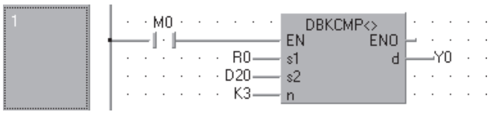
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	A negative value is specified for n.	—	—	—	—	○	○
4101	The range of n points from the device specified for (s1), (s2) or (d) exceeds the corresponding device. The range of n points from the device specified for (s1) overlaps with the range of n points from the device specified for (d). The range of n points from the device specified for (s2) overlaps with the range of n points from the device specified for (d).	—	—	—	—	○	○

Program example

- In the following program, the data stored in R0 to R5 are compared with the data stored in D20 to D25 when M0 turns ON, and the results are stored to Y0 to Y2.

[Structured ladder/FBD]

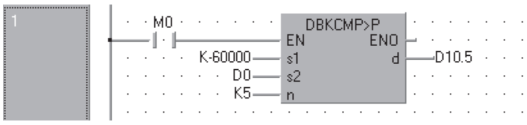


[Operation]

	b31 b0			b31 b0			
R1,R0	-2147483000	< >	D21,D20	-2147483000	⇨	Y0	OFF (0)
R3,R2	0		D23,D22	1		Y1	ON (1)
R5,R4	2147483000		D25,D24	2147482999		Y2	ON (1)

- In the following program, the constant is compared with the data stored in D0 to D9 when M0 turns ON, and the results are stored to D10.5 to D10.9.

[Structured ladder/FBD]



[Operation]

b31 b0			b31 b0			
-60000	>	D1,D0	-70000	⇨	D10.5	ON (1)
		D3,D2	50000		D10.6	OFF (0)
		D5,D4	-32768		D10.7	OFF (0)
		D7,D6	32767		D10.8	OFF (0)
		D9,D8	0		D10.9	OFF (0)

Point

When bits of word device are specified, only the specified bit devices that store the operation results are changed.

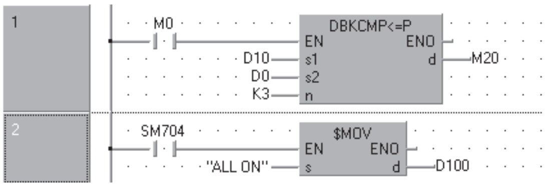
	D10.F															D10.0
Before execution	0	0	1	0	1	1	1	1	1	0	0	1	1	0	0	0
After execution	0	0	1	0	1	1	0	0	0	0	1	1	1	0	0	0

No changes

No changes

- In the following scan program, the data in D0 to D5 are compared with the data in D10 to D15 when M0 turns ON, and the results are stored to M20 to M22. The character string data "ALL ON" are transferred to D100 and the following devices when all devices of M20 to M22 turn ON.

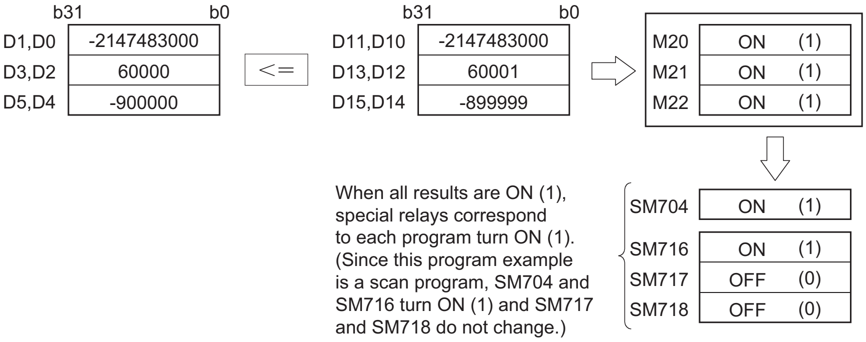
[Structured ladder/FBD]



```
1   M0                DBKCMP<=P
    -| |-------------EN       ENO-
              D10----s1         d----M20
               D0----s2
               K3----n

2   SM704             $MOV
    -| |-------------EN     ENO-
         "ALL ON"----s        d----D100
```

[Operation]



	b31 b0			b31 b0			
D1,D0	-2147483000	<=	D11,D10	-2147483000	⇨	M20	ON (1)
D3,D2	60000		D13,D12	60001		M21	ON (1)
D5,D4	-900000		D15,D14	-899999		M22	ON (1)

⇩

When all results are ON (1), special relays correspond to each program turn ON (1). (Since this program example is a scan program, SM704 and SM716 turn ON (1) and SM717 and SM718 do not change.)

SM704	ON (1)
SM716	ON (1)
SM717	OFF (0)
SM718	OFF (0)

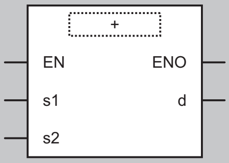
6.2 Arithmetic Operation Instructions

16-bit BIN data addition and subtraction

+(P), -(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



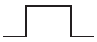
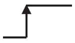
ST

Not supported

Any of the following instruction can go in the dotted squares.

+, +P, -, -P

■Executing condition

Instruction	Executing condition
+ -	
+P -P	

■Argument

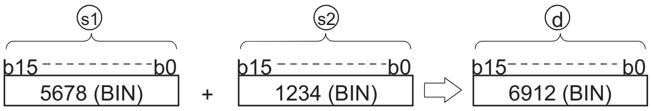
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Data to be added or subtracted, or start number of the device that stores data to be added or subtracted	ANY16
	s2	Start number of the device that stores the operation result	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(d)	○							—	—

Processing details

■+(P)

- Adds 16-bit BIN data specified for (s1) to 16-bit BIN data specified for (s2) and stores the result of the addition to the variable specified for (d).



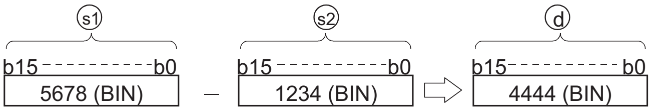
- Values can be specified between -32768 and 32767 (BIN 16 bits) for (s1) and (s2).
- The judgment of whether the value is positive or negative is made by the most significant bit (b15).
  - 0: Positive
  - 1: Negative
- The following is the result when an underflow or overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.

· 32767 (7FFFH) +2 (0002H) → −32767 (8001H) ······Since bit 15 value is "1", result of operation takes a negative value.

· −32768 (8000H) +−2 (FFFEH) → 32766 (7FFEH) ········Since bit 15 value is "0", result of operation takes a positive value.

■-(P)

- Subtracts 16-bit BIN data specified for (s1) from 16-bit BIN data for (s2) and stores the result of the subtraction to the variable specified for (d).



- Values can be specified between -32768 and 32767 (BIN 16 bits) for (s1) and (s2).
- The judgment of whether the value is positive or negative is made by the most significant bit (b15).
  - 0: Positive
  - 1: Negative
- The following is the result when an underflow or overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.

· −32768 (8000H) −2 (0002H) → 32766 (7FFEH) ········Since bit 15 value is "0", result of operation takes a positive value.

· 32767 (7FFFH) −−2 (FFFEH) → −32767 (8001H) ······Since bit 15 value is "1", result of operation takes a negative value.

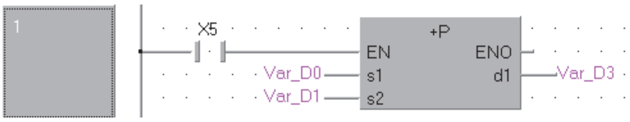
Operation error

- There is no operation error.

Program example

- In the following program, the value in Var_D3 is added to the value in Var_D0 when X5 turns ON, and outputs the result to Var_D3.

[Structured ladder/FBD]

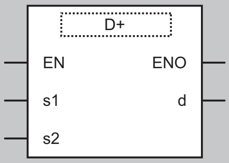


32-bit BIN data addition and subtraction

D+(P), D-(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



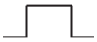
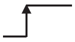
ST

Not supported

Any of the following instruction can go in the dotted squares.

D+, D+P, D-, D-P

■Executing condition

Instruction	Executing condition
D+ D-	
D+P D-P	

■Argument

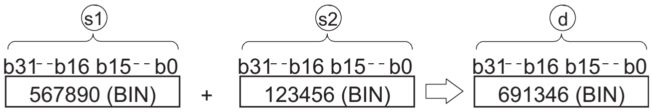
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Data to be added or subtracted, or start number of the device that stores data to be added or subtracted	ANY32
	s2	Start number of the device that stores the operation result	ANY32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(d)	○							—	—

Processing details

■D+(P)

- Adds 32-bit BIN data specified for (s1) to 32-bit BIN data specified for (s2), and stores the result of the addition to the variable specified for (d).



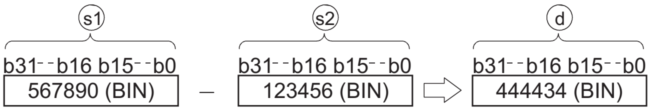
- Values can be specified between -2147483648 and 2147483647 (BIN 32 bits) for (s1) and (s2).
- Judgment of whether the value is positive or negative is made on the basis of the most significant bit (b31).
  - 0: Positive
  - 1: Negative
- The following is the result when an underflow or overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.

```
· 2147483647    +2        ──────▶  −2147483647···Since bit 31 value is "1",
  (7FFFFFFFH)   (00000002H)   (80000001H)      result of operation takes a negative value.

· −2147483648 + −2        ──────▶  2147483646······Since bit 31 value is "0",
  (80000000H)   (FFFFFFFEH)   (7FFFFFFEH)      result of operation takes a positive value.
```

■D-(P)

- Subtracts 32-bit BIN data specified for (s1) from 32-bit BIN data specified for (s2) and stores the result of the subtraction to the variable specified for (d).



- Values can be specified between -2147483648 and 2147483647 (BIN 32 bits) for (s1) and (s2).
- Judgment of whether the value is positive or negative is made on the basis of the most significant bit (b31).
  - 0: Positive
  - 1: Negative
- The following is the result when an underflow or overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.

```
· −2147483648 − 2        ──────▶  2147483646······Since bit 31 value is "0",
  (80000000H)   (00000002H)   (7FFFFFFEH)      result of operation takes a positive value.

· 2147483647    − −2      ──────▶  −2147483647···Since bit 31 value is "1",
  (7FFFFFFFH)   (FFFFFFFEH)   (80000001H)      result of operation takes a negative value.
```

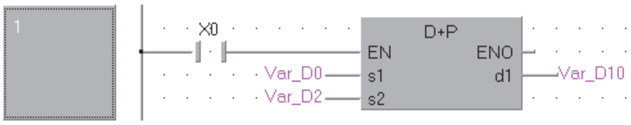
Operation error

- There is no operation error.

Program example

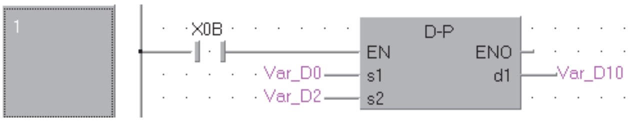
- In the following program, the value in Var_D0 is added to the value in Var_D2 when X0 turns ON, and the result is stored to Var_D10.

[Structured ladder/FBD]



- In the following program, the value in Var_D2 is subtracted from the value in Var_D0 when X0B turns ON, and the result is stored to Var_D10.

[Structured ladder/FBD]

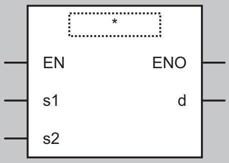


16-bit BIN data multiplication and division

*(P), /(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



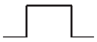
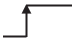
ST

Not supported

Any of the following instruction can go in the dotted squares.

*, *P, /, /P

■Executing condition

Instruction	Executing condition
* /	
*P /P	

■Argument

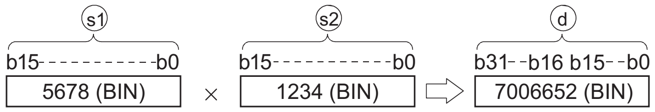
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Multiplicand or dividend data, or start number of the device that stores multiplicand or dividend data	ANY16
	s2	Multiplier or divisor data, or start number of the device that stores multiplier or divisor data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores the operation result	Array of ANY16/32 (1..2)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(d)	○							—	—

Processing details

■*(P)

- Multiplies 16-bit BIN data specified for (s1) and 16-bit BIN data specified for (s2), and stores the result to the variable specified for (d).



- When (d) is a bit device, specification is made from the lower bits.

Ex.

K1 ··· Lower 4 bits (b0 to b3)

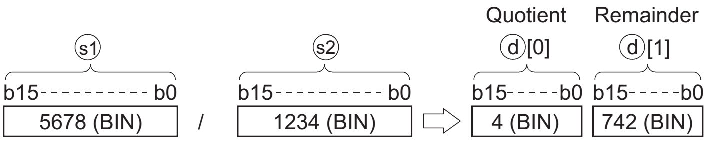
K4 ··· Lower 16 bits (b0 to b15)

K8 ··· 32 bits (b0 to b31)

- Values can be specified between -32768 and 32767 (BIN 16 bits) for (s1) and (s2).
- Judgments whether (s1), (s2), and (d) are positive or negative are made on the basis of the most significant bit (b15 for (s1) and (s2), and b31 for (d)).
  - 0: Positive
  - 1: Negative

■/(P)

- Divides 16-bit BIN data specified for (s1) and 16-bit BIN data specified for (s2), and stores the result to the variable specified for (d).



- Quotient and remainder of the division result are stored in 32-bit data.
  - Quotient: Stored to (d)[0] (16 bits).
  - Remainder: Stored to (d)[1] (16 bits).
- Values can be specified between -32768 and 32767 (BIN 16 bits) for (s1) and (s2).
- Judgment whether values for (s1), (s2), (d)[0], and (d)[1] are positive or negative is made on the basis of the most significant bit (b15). (Sign is attached to both the quotient and remainder.)
  - 0: Positive
  - 1: Negative

Operation error

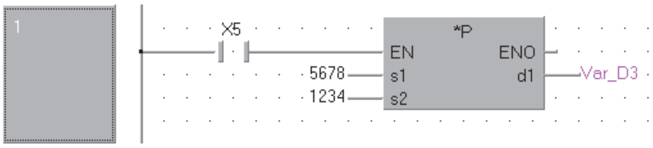
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	Divisor (s2) is 0.	○	○	○	○	○	○

Program example

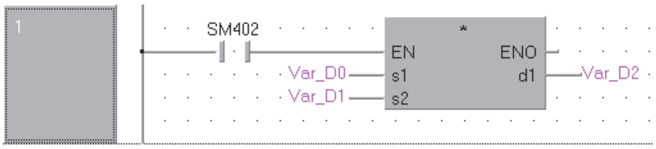
- In the following program, the BIN value 5678 is multiplied by the BIN value 1234 when X5 turns ON, and the result is stored to Var_D3.

[Structured ladder/FBD]



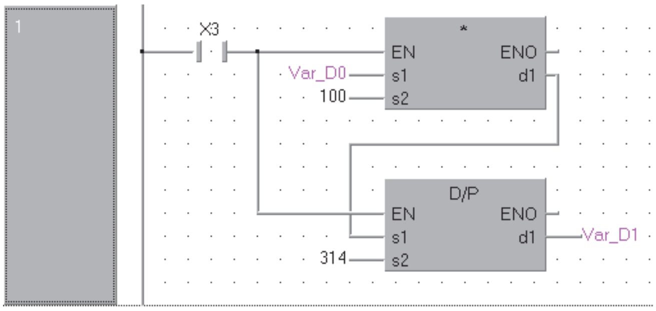
- In the following program, the BIN value in Var_D0 is multiplied by the BIN value in Var_D1, and the result is stored to Var_D2.

[Structured ladder/FBD]



- In the following program, the value in Var_D0 is divided by 3.14 when X3 turns ON, and the result is stored to Var_D1.

[Structured ladder/FBD]

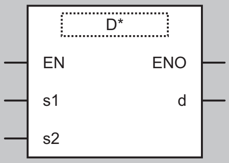


32-bit BIN data multiplication and division

D*(P), D/(P)

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



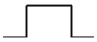
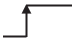
ST

Not supported

Any of the following instruction can go in the dotted squares.

D*, D*P, D/, D/P

■Executing condition

Instruction	Executing condition
D* D/	
D*P D/P	

■Argument

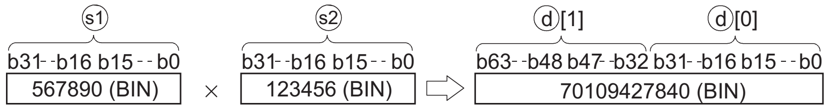
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Multiplicand or dividend data, or start number of the device that stores multiplicand or dividend data	ANY32
	s2	Multiplier or divisor data, or start number of the device that stores multiplier or divisor data	ANY32
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores the operation result	Array of ANY32 (1..2)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○			○					—
(s2)	○			○					—
(d)	○			—					—

Processing details

■D*(P)

- Multiplies the 32-bit BIN data specified for (s1) by the 32-bit BIN data specified for (s2), and stores the result to the variable specified for (d).



- If (d) is a bit device, only the lower 32 bits of the multiplication result is considered, and the upper 32 bits cannot be specified.

Ex.

K1 ··· Lower 4 bits (b0 to b3)

K4 ··· Lower 16 bits (b0 to b15)

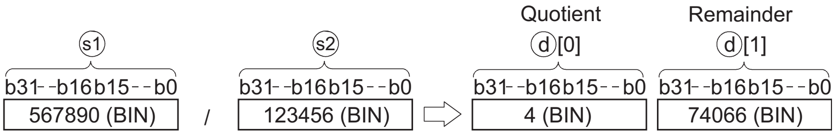
K8 ··· Lower 32 bits (b0 to b31)

If the upper 32 bits of the bit device are required for the result of the multiplication operation, first temporarily store the data in a word device, then transfer the word device data to the bit device by designating ((d)2) and ((d)3) data.

- Values can be specified between -2147483648 and 2147483647 (BIN 32 bits) for (s1) and (s2).
- Judgments whether (s1), (s2), and (d) are positive or negative are made on the basis of the most significant bit (b31 for (s1) and (s2), and b63 for (d)).

■D/(P)

- Divides the 32-bit BIN data specified for (s1) by the 32-bit BIN data specified for (s2), and stores the result to the variable specified for (d).



- Quotient and remainder of the division result are stored in 64-bit data.
  - Quotient: Stored to (d)[0] (32 bits).
  - Remainder: Stored to (d)[1] (32 bits).
- Values can be specified between -2147483648 and 2147483647 (BIN 32 bits) for (s1) and (s2).
- Judgment whether values for (s1), (s2), (d)[0], and (d)[1] are positive or negative is made on the basis of the most significant bit (b31). (Sign is attached to both the quotient and remainder.)
  - 0: Positive
  - 1: Negative

Operation error

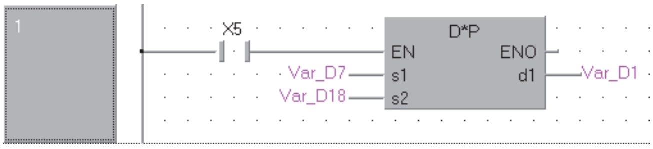
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	Divisor (s2) is 0.	○	○	○	○	○	○

Program example

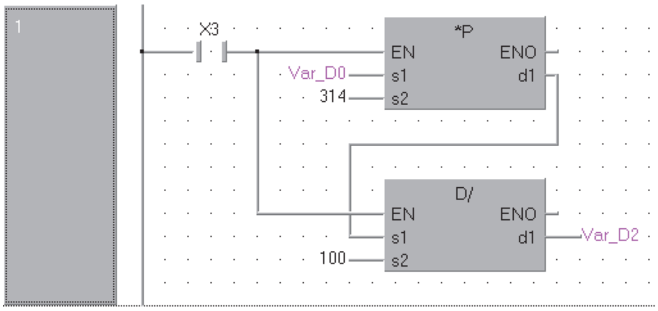
- In the following program, the BIN value in Var_D7 is multiplied by the BIN value in Var_D18 when X5 turns ON, and the result is stored to Var_D1.

[Structured ladder/FBD]



- In the following program, the value in Var_D0 is multiplied by 3.14 when X3 turns ON, and the result is stored to Var_D2.

[Structured ladder/FBD]

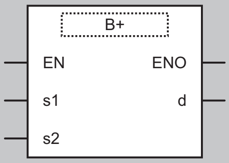


4-digit BCD data addition and subtraction

B+(P), B-(P)

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



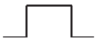
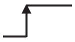
ST

Not supported

Any of the following instruction can go in the dotted squares.

B+, B+P, B-, B-P

■Executing condition

Instruction	Executing condition
B+ B-	
B+P B-P	

■Argument

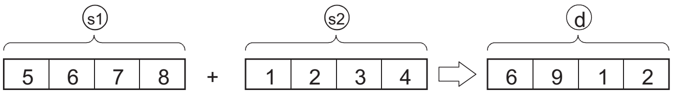
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Data to be added or subtracted, or start number of the device that stores data to be added or subtracted	ANY16
	s2	Start number of the device that stores the operation result	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(d)	○							—	—

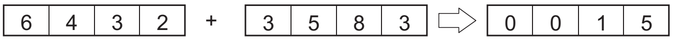
Processing details

■B+(P)

- Adds the 4-digit BCD data specified for (s1) and the 4-digit BCD data specified for (s2), and stores the result of the addition to the variable specified for (d).

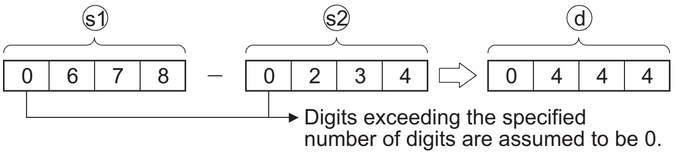


- Values can be specified between 0 and 9999 (4-digit BCD) for (s1), (s2) and (d).
- If the result of the addition operation exceeds 9999, the higher bits are ignored. The carry flag (SM700) in this case does not turn ON.

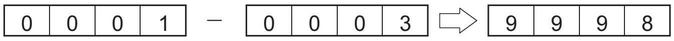


■B-(P)

- Subtracts the 4-digit BCD data specified for (s1) and the 4-digit BCD data specified for (s2), and stores the result of the subtraction to the variable specified for (d).


s1
s2
d
0 6 7 8
−
0 2 3 4
0 4 4 4
Digits exceeding the specified number of digits are assumed to be 0.

- Values can be specified between 0 and 9999 (4-digit BCD) for (s1), (s2) and (d).
- The following is the result when an underflow is generated by the subtraction operation. The carry flag (SM700) in this case does not turn ON.



Operation error

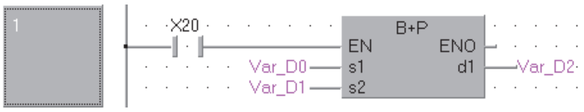
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The (s1) or (s2) BCD value is outside the range of 0 to 9999.	○	○	○	○	○	○

Program example

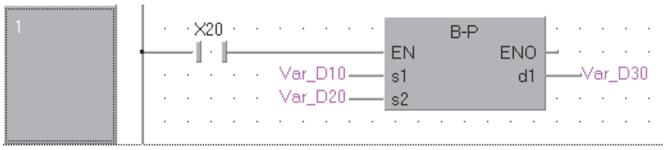
- In the following program, the BCD value in Var_D0 is added to the BCD value in Var_D1 when X20 turns ON, and the result is stored to Var_D2.

[Structured ladder/FBD]



- In the following program, the BCD value in Var_D20 is subtracted from the BCD value in Var_D10 when X20 turns ON, and the result is stored to Var_D30.

[Structured ladder/FBD]

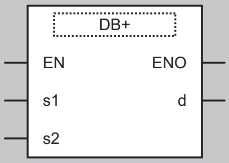


8-digit BCD data addition and subtraction

DB+(P), DB-(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



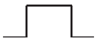
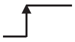
ST

Not supported

Any of the following instruction can go in the dotted squares.

DB+, DB+P, DB-, DB-P

■Executing condition

Instruction	Executing condition
DB+ DB-	
DB+P DB-P	

■Argument

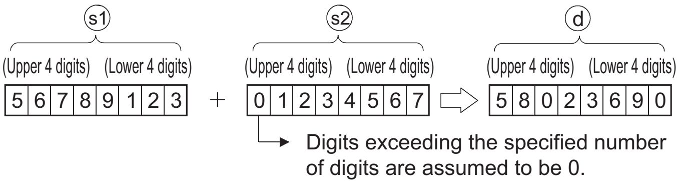
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Data to be added or subtracted, or start number of the device that stores data to be added or subtracted	ANY32
	s2	Addition or subtraction data, or start number of the device that stores addition or subtraction data	ANY32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(d)	○							—	—

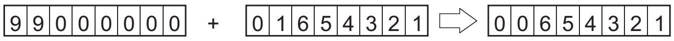
Processing details

■DB+(P)

- Adds the 8-digit BCD data specified for (s1) and the 8-digit BCD data specified for (s2), and stores the result of the addition to the variable specified for (d).

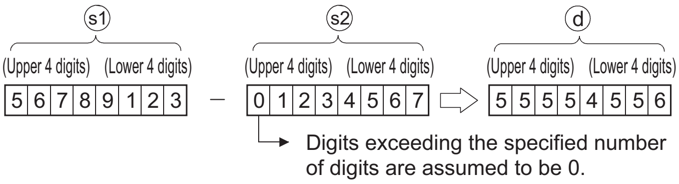

s1
s2
d
(Upper 4 digits) (Lower 4 digits)
(Upper 4 digits) (Lower 4 digits)
(Upper 4 digits) (Lower 4 digits)
5 6 7 8 9 1 2 3
+
0 1 2 3 4 5 6 7
5 8 0 2 3 6 9 0
Digits exceeding the specified number of digits are assumed to be 0.

- Values can be specified between 0 and 99999999 (8-digit BCD) for (s1) and (s2).
- If the result of the addition operation exceeds 99999999, the upper bits will be ignored. The carry flag (SM700) in this case does not turn ON.

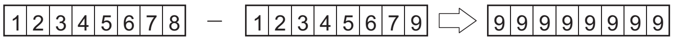


■DB-(P)

- Subtracts the 8-digit BCD data specified for (s1) and the 8-digit BCD data specified for (s2), and stores the result of the subtraction to the variable specified for (d).


s1
s2
d
(Upper 4 digits) (Lower 4 digits)
(Upper 4 digits) (Lower 4 digits)
(Upper 4 digits) (Lower 4 digits)
5 6 7 8 9 1 2 3
−
0 1 2 3 4 5 6 7
5 5 5 5 4 5 5 6
Digits exceeding the specified number of digits are assumed to be 0.

- Values can be specified between 0 and 99999999 (8-digit BCD) for (s1) and (s2).
- The following is the result when an underflow is generated by the subtraction operation. The carry flag (SM700) in this case does not turn ON.



Operation error

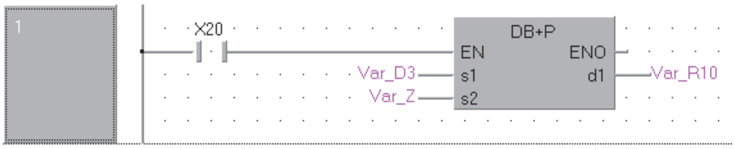
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The (s1) or (s2) BCD value is outside the range of 0 to 99999999.	○	○	○	○	○	○

Program example

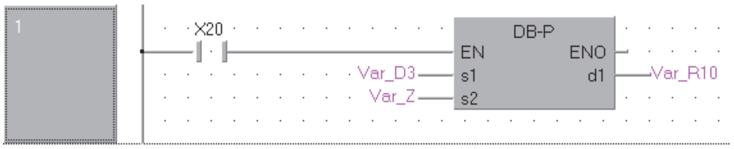
- In the following program, the BCD value in Var_D3 is added to the BCD value in Var_ Z when X20 turns ON, and the result is stored to Var_R10.

[Structured ladder/FBD]



- In the following program, the BCD data in Var_Z is subtracted from the BCD data in Var_D3 when X20 turns ON, and the result is stored to Var_R10.

[Structured ladder/FBD]

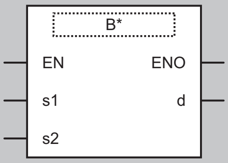


4-digit BCD data multiplication and division

B*(P), B/(P)

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



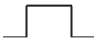
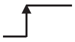
ST

Not supported

Any of the following instruction can go in the dotted squares.

B*, B*P, B/, B/P

■Executing condition

Instruction	Executing condition
B* B/	
B*P B/P	

■Argument

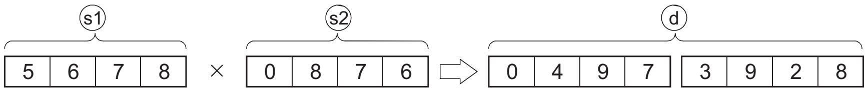
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Multiplicand or dividend data, or start number of the device that stores multiplicand or dividend data	ANY16
	s2	Multiplier or divisor data, or start number of the device that stores multiplier or divisor data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores the operation result	Array of ANY16/32 (1..2)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(d)	○							—	—

Processing details

■B*(P)

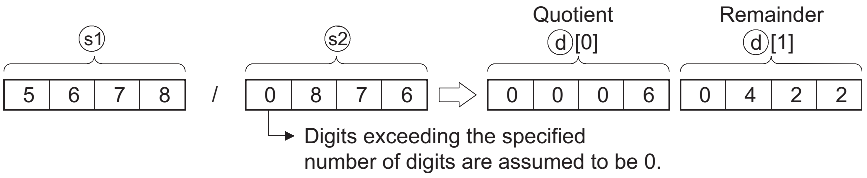
- Multiplies BCD data specified for (s1) and BCD data specified for (s2), and stores the result to the variable specified for (d).



- Values can be specified between 0 and 9999 (4-digit BCD) for (s1) and (s2).

■B/(P)

- Divides BCD data specified for (s1) by BCD data specified for (s2), and stores the result to the variable specified for (d).


Quotient
Remainder
s1
s2
d[0]
d[1]
5 6 7 8
/
0 8 7 6
0 0 0 6
0 4 2 2
Digits exceeding the specified
number of digits are assumed to be 0.

- Quotient and remainder of the division result are stored in 32-bit data.
  - Quotient (4-digit BCD): Stored to (d)[0] (16 bits).
  - Remainder (4-digit BCD): Stored to (d)[1] (16 bits).

Operation error

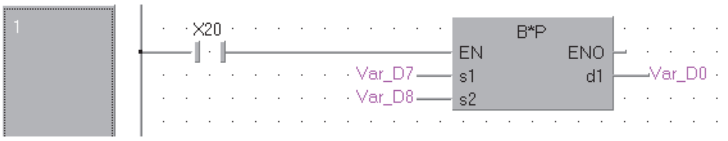
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The (s1) or (s2) BCD value is outside the range of 0 to 9999. Divisor (s2) is 0.	○	○	○	○	○	○

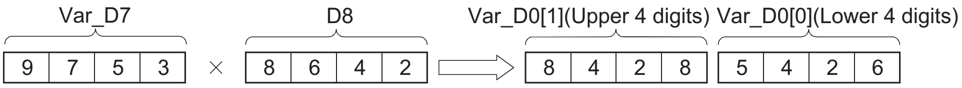
Program example

- In the following program, the BCD value in Var_D7 is multiplied by the BCD value in Var_D8 when X20 turns ON, and the result is stored to Var_D0.

[Structured ladder/FBD]

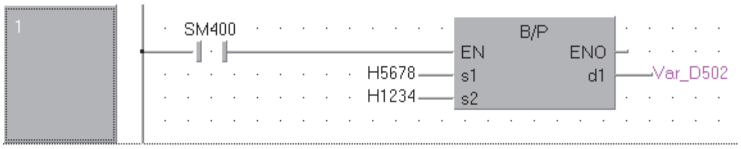


[Operation]

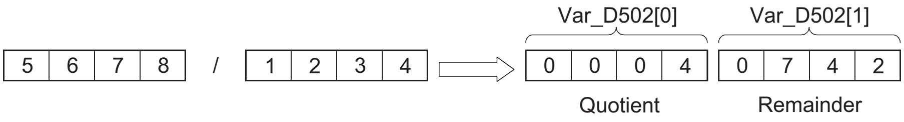


- In the following program, the BCD value 5678 is divided by the BCD value 1234, and the result is stored to Var_D502.

[Structured ladder/FBD]



[Operation]

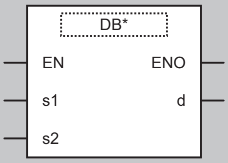


8-digit BCD data multiplication and division

DB*(P), DB/(P)

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



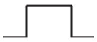
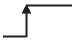
ST

Not supported

Any of the following instruction can go in the dotted squares.

DB*, DB*P, DB/, DB/P

■Executing condition

Instruction	Executing condition
DB* DB/	
DB*P DB/P	

■Argument

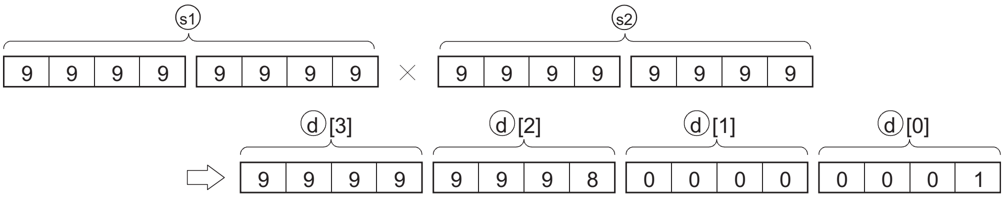
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Multiplicand or dividend data, or start number of the device that stores multiplicand or dividend data	ANY32
	s2	Multiplier or divisor data, or start number of the device that stores multiplier or divisor data	ANY32
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores the operation result	Array of ANY16 (1..4), array of ANY32 (1..2)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○			○					—
(s2)	○			○					—
(d)	○			—					—

Processing details

■DB*(P)

- Multiplies the 8-digit BCD data specified for (s1) and the 8-digit BCD data specified for (s2), and stores the product to the variable specified for (d).



- If (d) is a bit device, the lower 8 digits (lower 32 bits) of the multiplication result is considered, and the upper 8 digits (upper 32 bits) cannot be specified.

Ex.

K1 ··· Lower 1 bits (b0 to b3)

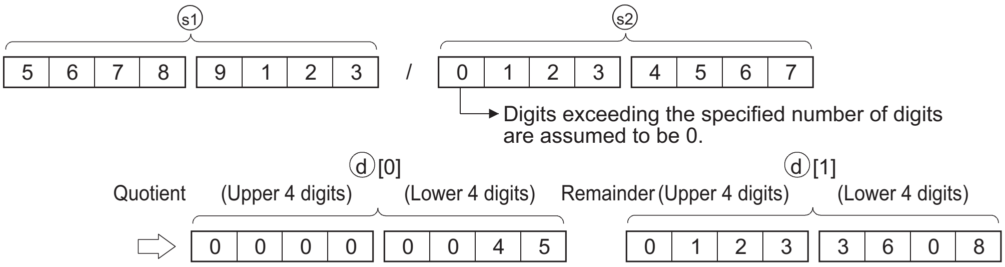
K4 ··· Lower 4 bits (b0 to b15)

K8 ··· Lower 8 bits (b0 to b31)

- Values can be specified between 0 and 99999999 (8-digit BCD) for (s1) and (s2).

■DB/(P)

- Divides 8-digit BCD data specified for (s1) by 8-digit BCD data specified for (s2), and stores the result to the variable specified for (d).



- Quotient and remainder of the division result are stored in 64-bit data.
  - Quotient (8-digit BCD): Stored to (d)[0] (32 bits).
  - Remainder (8-digit BCD): Stored to (d)[1] (32 bits).

Operation error

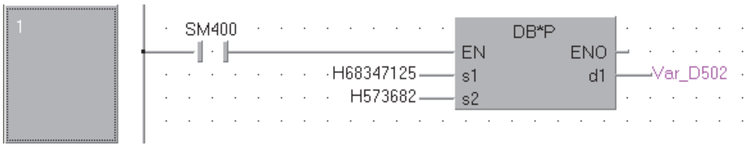
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The (s1) or (s2) BCD value is outside the range of 0 to 99999999. Divisor (s2) is 0.	○	○	○	○	○	○

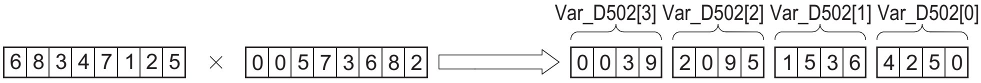
Program example

- In the following program, the BCD value 68347125 is multiplied by the BCD value 573682, and the result is stored to Var_D502.

[Structured ladder/FBD]

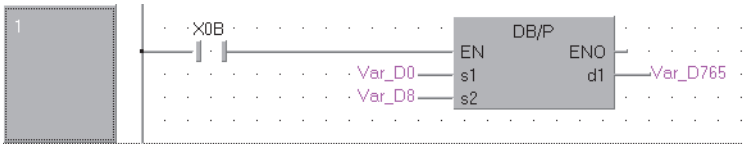


[Operation]

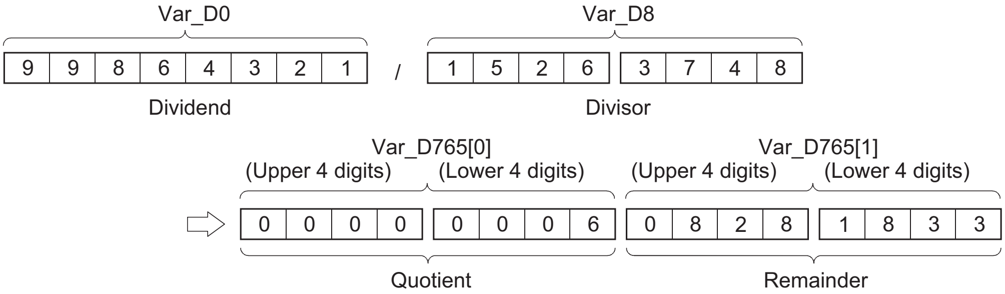


- In the following program, the BCD value in Var_D0 is divided by the BCD value in Var_D8 when X0B turns ON, and the result is stored to Var_D765.

[Structured ladder/FBD]



[Operation]



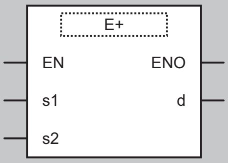
Floating-point data addition and subtraction (single precision)

E+(P), E-(P)

Ver. Basic | High performance | Process | Redundant | Universal | LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

Structured ladder/FBD



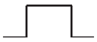
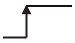
ST

Not supported

Any of the following instruction can go in the dotted squares.

E+, E+P, E-, E-P

■Executing condition

Instruction	Executing condition
E+ E-	
E+P E-P	

■Argument

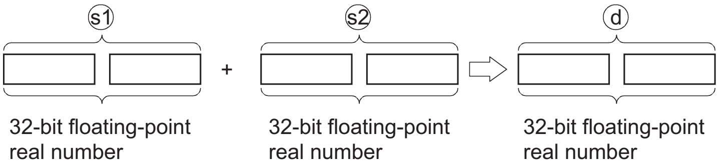
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Data to be added or subtracted, or start number of the device that stores data to be added or subtracted	Single-precision real
	s2	Addition or subtraction data, or start number of the device that stores addition or subtraction data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—	○		—	○	—
(s2)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	○	—

Processing details

■E+(P)

- Adds the 32-bit floating-point real number specified for (s1) and the 32-bit floating-point real number specified for (s2), and stores the sum to the variable specified for (d).


s1
s2
d
+
32-bit floating-point real number
32-bit floating-point real number
32-bit floating-point real number

- Values which can be specified for (s1), (s2) and (d), and which can be stored, are as follows:

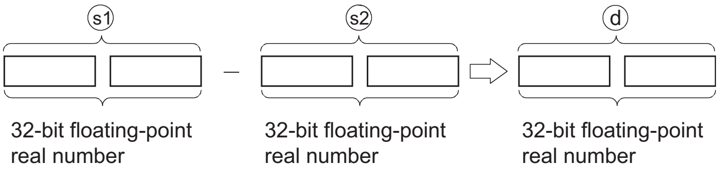
0, $2^{-126} \leq$ | specified value (storing value) | $< 2^{128}$

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

■E-(P)

- Subtracts the 32-bit floating-point real number specified for (s1) and the 32-bit floating-point real number specified for (s2), and stores the result to the variable specified for (d).


s1
s2
d
32-bit floating-point real number
32-bit floating-point real number
32-bit floating-point real number

- Values which can be specified for (s1), (s2) and (d), and which can be stored, are as follows:

0, $2^{-126} \leq$ | specified value (storing value) | < 2^{128}

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

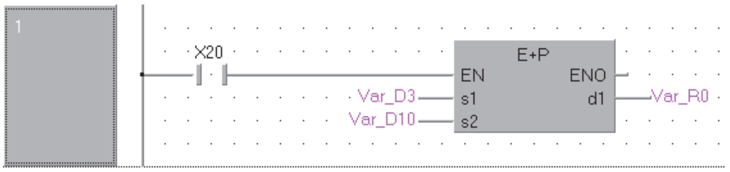
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of the specified device is outside the following range. 0, $2^{-126} \leq$	value of the specified device	< 2^{128} The value of the specified device is -0.[*1]	○	○	○	○	—	—
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{128} \leq$	operation result		—	—	—	—	○	○
4140	The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○

[*1] There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

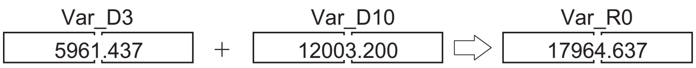
Program example

- In the following program, the floating-point real number Var_D3 is added to the floating-point real number Var_D10 when X20 turns ON, and the result is stored to Var_R0.

[Structured ladder/FBD]

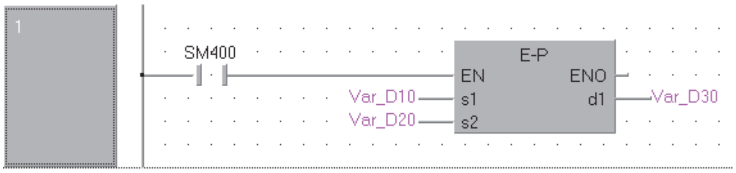


[Operation]

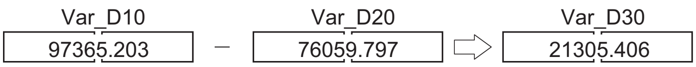


- In the following program, the floating-point real number Var_D20 is subtracted from the floating-point real number Var_D10, and the result is stored to Var_D30.

[Structured ladder/FBD]



[Operation]

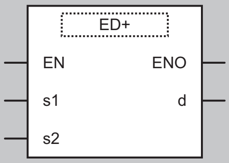


Floating-point data addition and subtraction (double precision)

ED+(P), ED-(P)

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



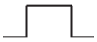
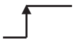
ST

Not supported

Any of the following instruction can go in the dotted squares.

ED+, ED+P, ED-, ED-P

■Executing condition

Instruction	Executing condition
ED+ ED-	
ED+P ED-P	

■Argument

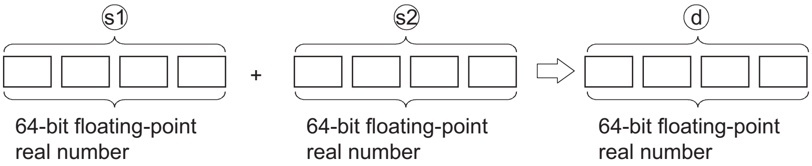
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Data to be added or subtracted, or start number of the device that stores data to be added or subtracted	Double-precision real
	s2	Addition or subtraction data, or start number of the device that stores addition or subtraction data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				○	—
(s2)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

■ED+(P)

- Adds the 64-bit floating-point real number specified for (s1) and the 64-bit floating-point real number specified for (s2), and stores the sum to the variable specified for (d).


(s1)
+
(s2)
(d)
64-bit floating-point real number
64-bit floating-point real number
64-bit floating-point real number

- Values which can be specified for (s1), (s2) and (d), and which can be stored, are as follows:

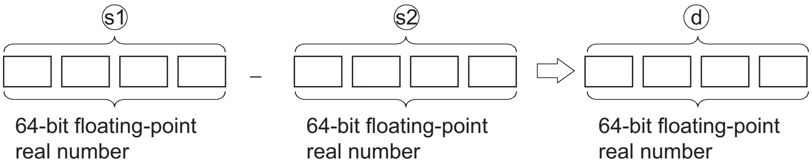
0, $2^{-1022} \leq$ | specified value (storing value) | < 2^{1024}

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

■ED-(P)

- Subtracts the 64-bit floating-point real number specified for (s1) from the 64-bit floating-point real number specified for (s2), and stores the result to (d).


s1
s2
d
64-bit floating-point real number
64-bit floating-point real number
64-bit floating-point real number

- Values which can be specified for (s1), (s2) and (d), and which can be stored, are as follows:

0, $2^{-1022} \leq$ | specified value (storing value) | < 2^{1024}

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

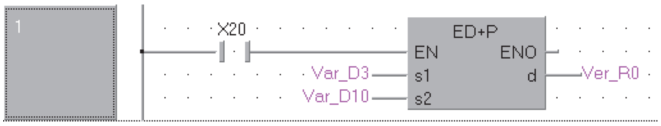
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | < 2^{1024} The value of the specified device is -0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

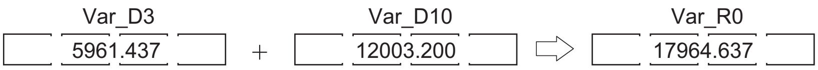
Program example

- In the following program, the 64-bit floating-point real number Var_D3 is added to the 64-bit floating-point real number Var_D10 when X20 turns ON, and the result is stored to Var_R0.

[Structured ladder/FBD]

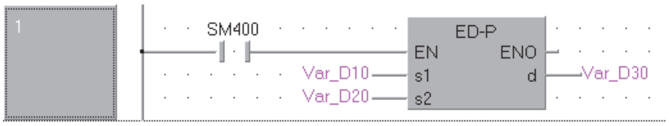


[Operation]

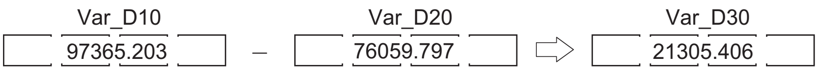


- In the following program, the 64-bit floating-point real number Var_D20 is subtracted from the 64-bit floating-point real number Var_D10, and the result is stored to Var_D30.

[Structured ladder/FBD]



[Operation]



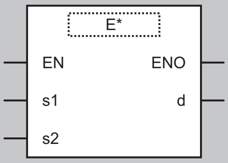
Floating-point data multiplication and division (single precision)

E*(P), E/(P)

Ver. Basic | High performance | Process | Redundant | Universal | LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

Structured ladder/FBD



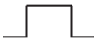
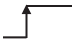
ST

Not supported

Any of the following instruction can go in the dotted squares.

E*, E*P, E/, E/P

■Executing condition

Instruction	Executing condition
E* E/	
E*P E/P	

■Argument

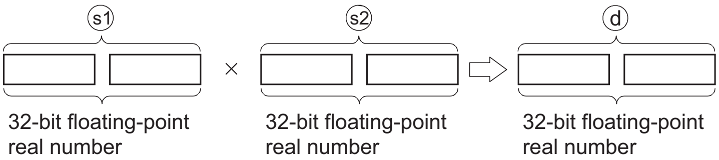
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Multiplicand or dividend data, or start number of the device that stores multiplicand or dividend data	Single-precision real
	s2	Multiplier or divisor data, or start number of the device that stores multiplier or divisor data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—	○		—	○	—
(s2)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

■E*(P)

- Multiplies the 32-bit floating-point real number specified for (s1) by the 32-bit floating-point real number specified for (s2) and stores the operation result to the variable specified for (d).


s1
s2
d
×
32-bit floating-point real number
32-bit floating-point real number
32-bit floating-point real number

- Values which can be specified for (s1), (s2) and (d), and which can be stored, are as follows:

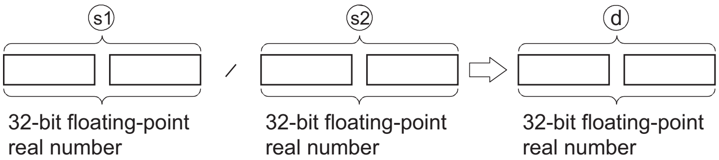
0, $2^{-126} \leq$ | specified value (storing value) | $< 2^{128}$

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

■E/(P)

- Divides the 32-bit floating-point real number specified for (s1) by the 32-bit floating-point real number specified for (s2) and stores the operation result to the variable specified for (d).


s1
/
s2
d
32-bit floating-point real number
32-bit floating-point real number
32-bit floating-point real number

- Values which can be specified for (s1), (s2) and (d), and which can be stored, are as follows:

0, 2^{-126} ≤ | specified value (storing value) | < 2^{128}

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

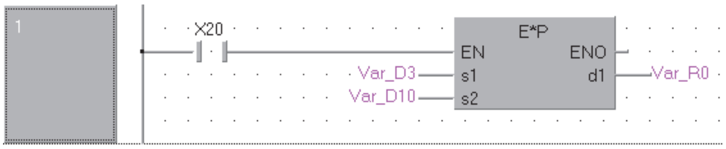
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of the specified device is outside the following range. 0, 2^{-126} ≤ | value of the specified device | < 2^{128} The value of the specified device is -0.	○	○	○	○	—	—
	Divisor (s2) is 0.	○	○	○	○	○	—
4141	The operation result exceeds the following range. (An overflow occurs.) 2^{128} ≤ | operation result |	—	—	—	—	○	○
4140	The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○

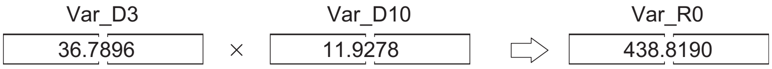
Program example

- In the following program, the 32-bit floating-point real number in Var_D3 is multiplied by the 32-bit floating-point real number in Var_D10, and the result is stored to Var_R0.

[Structured ladder/FBD]

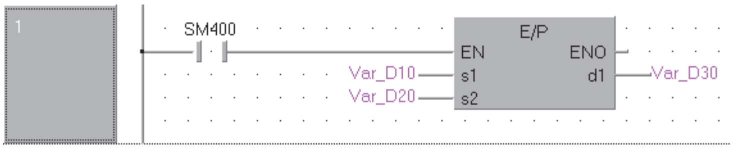


[Operation]

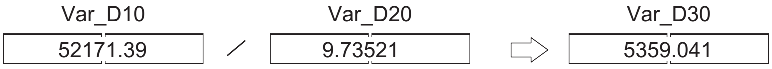


- In the following program, the 32-bit floating-point real number in Var_D10 is divided by the 32-bit floating-point real number in Var_D20, and the result is stored to Var_D30.

[Structured ladder/FBD]



[Operation]

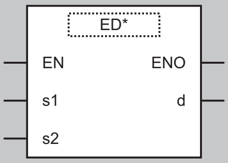


Floating-point data multiplication and division (double precision)

ED*(P), ED/(P)

☒ Basic ☒ High performance ☒ Process ☒ Redundant Universal LCPU

Structured ladder/FBD



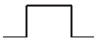
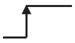
ST

Not supported

Any of the following instruction can go in the dotted squares.

ED*, ED*P, ED/, ED/P

■Executing condition

Instruction	Executing condition
ED* ED/	
ED*P ED/P	

■Argument

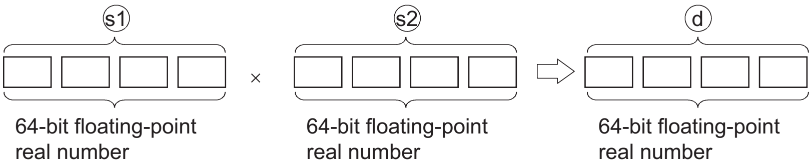
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Multiplicand or dividend data, or start number of the device that stores multiplicand or dividend data	Double-precision real
	s2	Multiplier or divisor data, or start number of the device that stores multiplier or divisor data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				○	—
(s2)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

■ED*(P)

- Multiplies the 64-bit floating-point real number specified for (s1) and the 64-bit floating-point real number specified for (s2), and stores the result to the variable specified for (d).


s1
s2
d
×
64-bit floating-point real number
64-bit floating-point real number
64-bit floating-point real number

- Values which can be specified for (s1), (s2) and (d), and which can be stored, are as follows:

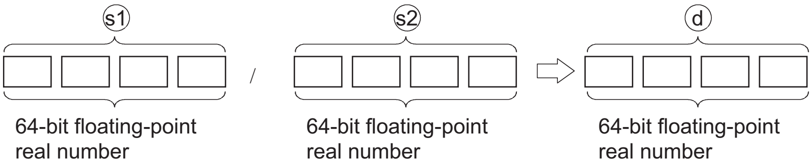
$0, 2^{-1022} \leq |\text{ specified value (storing value) }| < 2^{1024}$

- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

■ED/(P)

- Divides the 64-bit floating-point real number specified for (s1) by the 64-bit floating-point real number specified for (s2), and stores the result to (d).


s1
/
s2
d
64-bit floating-point real number
64-bit floating-point real number
64-bit floating-point real number

- Values which can be specified for (s1), (s2) and (d), and which can be stored, are as follows:

0, $2^{-1022} \leq$ | specified value (storing value) | < 2^{1024}

- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

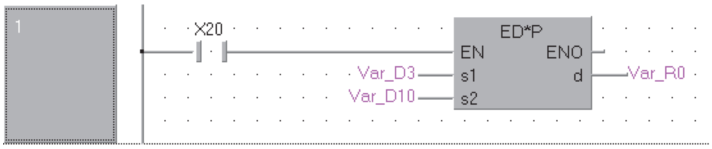
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | < 2^{1024} The value of the specified device is -0.	—	—	—	—	○	○
4100	Divisor (s2) is 0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

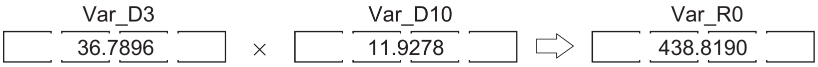
Program example

- In the following program, the 64-bit floating-point real number in Var_D3 is multiplied by the 64-bit floating-point real number in Var_D10, and the result is stored to Var_R0.

[Structured ladder/FBD]

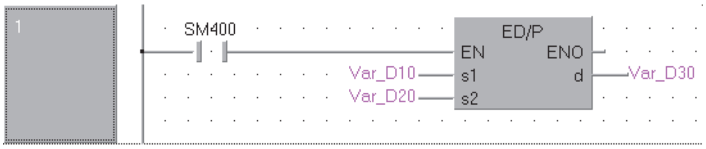


[Operation]

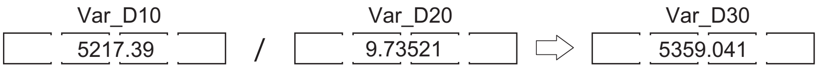


- In the following program, the 64-bit floating-point real number in Var_D10 is divided by the 64-bit floating-point real number in Var_D20, and the result is stored to Var_D30.

[Structured ladder/FBD]



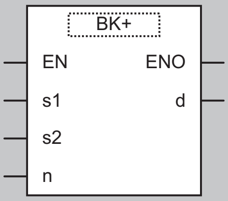
[Operation]



16-bit BIN block data addition and subtraction

BK+(P), BK-(P)

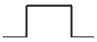
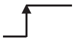
Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD	ST
	Not supported

Any of the following instruction can go in the dotted squares.

BK+, BK+P, BK-, BK-P

■Executing condition

Instruction	Executing condition
BK+ BK-	
BK+P BK-P	

■Argument

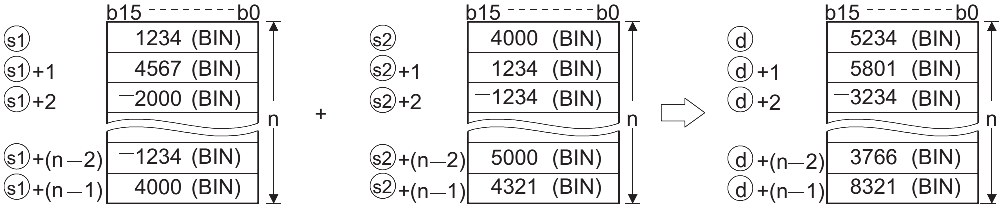
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores data to be added or subtracted	ANY16
	s2	Addition or subtraction data, or start number of the device that stores addition or subtraction data	ANY16
	n	Number of addition or subtraction data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				—	—
(s2)	—	○		—				○	—
n	—	○		○				○	—
(d)	—	○		—				—	—

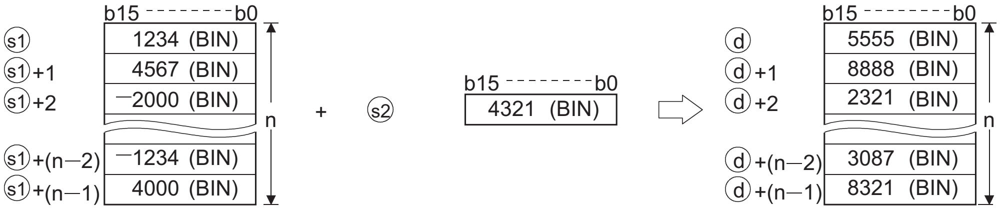
Processing details

■BK+(P)

- Adds BIN data n points from the device specified for (s1) and BIN data n points from the device specified for (s2), and stores the results to (d) and the following devices.



- Block addition is performed in units of 16 bits.
- The constant can be specified between -32768 and 32767 (BIN 16 bits) for (s2).



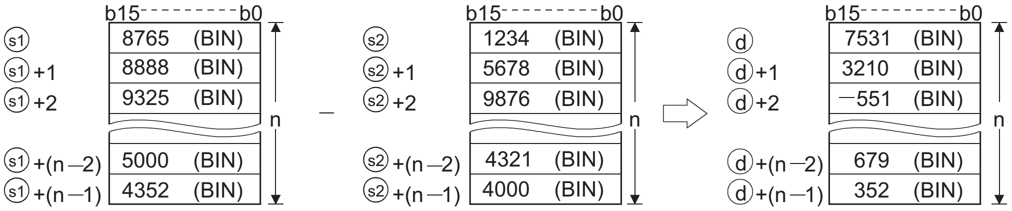
- The following is the result when an underflow or overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.

· 32767 (7FFFH) +2 (0002H) → −32767 (8001H)

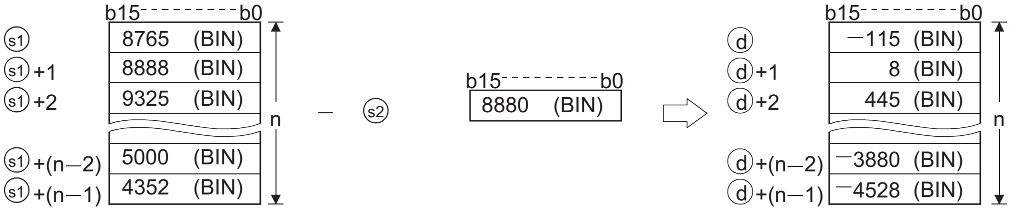
· −32767 (8001H) +−2 (FFFEH) → 32767 (7FFFH)

■BK-(P)

- Subtracts BIN data n points from the device specified for (s1) from BIN data n points from the device specified for (s2), and stores the results to (d) and the following devices.



- Block subtraction is performed in units of 16 bits.
- The constant can be specified between -32768 and 32767 (BIN 16 bits) for (s2).



- The following is the result when an underflow or overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.

· −32768 (8000H) −2 (0002H) → 32766 (7FFEH)

· 32767 (7FFFH) − −2 (FFFEH) → −32767 (8001H)

Operation error

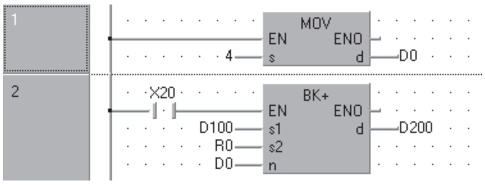
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (s1), (s2) or (d) exceeds the corresponding device. The range of n points from the device specified for (s1) overlaps with the range of n points from the device specified for (d). (Except when the same device is assigned to (s1) and (d)) The range of n points from the device specified for (s2) overlaps with the range of n points from the device specified for (d). (Except when the same device is specified for (s2) and (d))	○	○	○	○	○	○

Program example

- In the following program, the data stored in D100 to D103 are added to the data stored in R0 to R3 when X20 turns ON, and the results are stored to D200 and the following devices.

[Structured ladder/FBD]



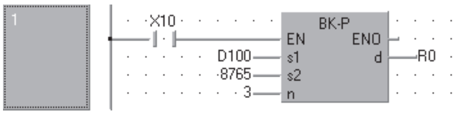
[Operation]

	b15 - - - - - - - - - b0			b15 - - - - - - - - - b0			b15 - - - - - - - - - b0
D100	6789 (BIN)		R0	1234 (BIN)		D200	8023 (BIN)
D101	7821 (BIN)	+	R1	2032 (BIN)	⇨	D201	9853 (BIN)
D102	5432 (BIN)		R2	−3252 (BIN)		D202	2180 (BIN)
D103	3520 (BIN)		R3	−1000 (BIN)		D203	2520 (BIN)

D0	4

- In the following program, the constant 8765 is subtracted from the data in D100 to D102 when X1C is turned ON, and the results are stored to R0 and the following devices.

[Structured ladder/FBD]



[Operation]

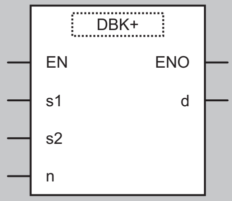
	b15 - - - - - - - - - b0		b15 - - - - - - - - - b0			b15 - - - - - - - - - b0
D100	12345 (BIN)				R0	3580 (BIN)
D101	8701 (BIN)	−	8765 (BIN)	⇨	R1	−64 (BIN)
D102	3502 (BIN)				R2	−5263 (BIN)

32-bit BIN block data addition and subtraction

DBK+(P), DBK-(P)

✗ Basic ✗ High performance ✗ Process ✗ Redundant Ver. Universal LCPU

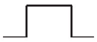
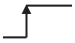
- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported

Structured ladder/FBD	ST
	Not supported

Any of the following instruction can go in the dotted squares.

DBK+, DBK+P, DBK-, DBK-P

■Executing condition

Instruction	Executing condition
DBK+ DBK-	
DBK+P DBK-P	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores data to be added or subtracted	ANY32
	s2	Addition or subtraction data, or start number of the device that stores addition or subtraction data	ANY32
	n	Number of addition or subtraction data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY32

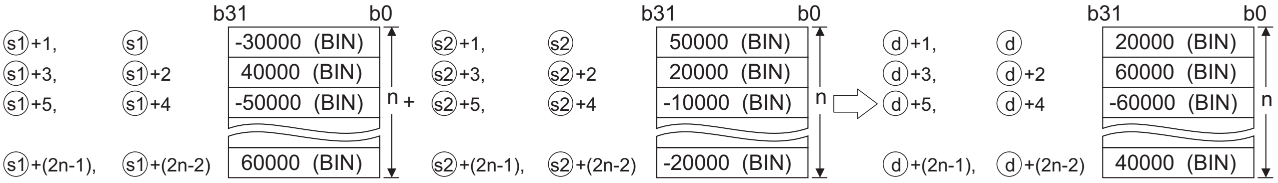
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				—	—
(s2)	—	○		—				○	—
n	○	○		○				○	—
(d)	—	○		—				—	—

Processing details

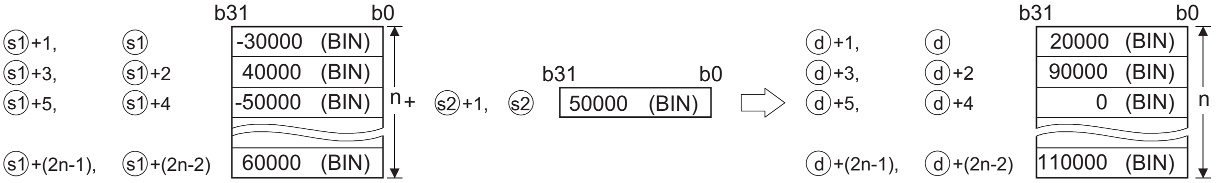
■DBK+(P)

- Adds 32-bit BIN data n points from the device specified for (s1) and 32-bit BIN data n points from the device specified for (s2) or a constant, and stores the results to (d) and the following devices.

• When a device is specified for (s2)



• When a constant is specified for (s2)



- Block addition is performed in units of 32 bits.
- The constant can be specified between -2147483648 and 2147483647 (BIN 32 bits) for (s2).
- No processing is performed if the value specified for n is 0.
- The following is the result when an overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.

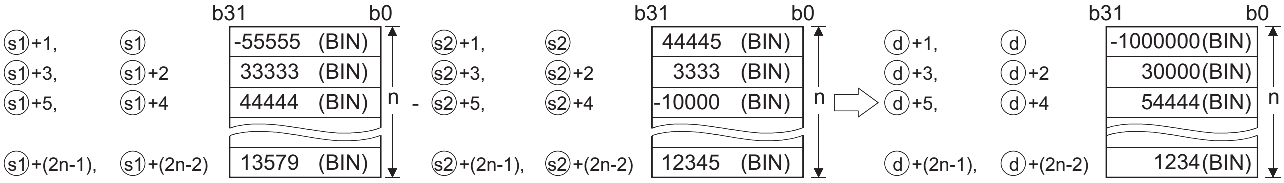
· K2147483647+K2 → K-2147483647
(7FFFFFFFH) (00000002H) (80000001H)

· K-2147483647 +K-2 → K2147483647
(80000001H) (FFFFFFFEH) (7FFFFFFFH)

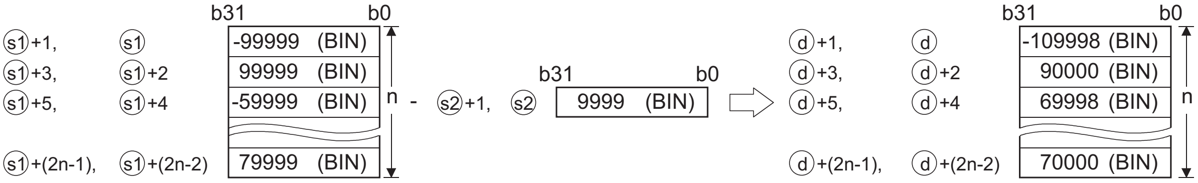
■DBK-(P)

- Subtracts 32-bit BIN data n points from the device specified for (s1), or a constant from 32-bit BIN data n points from the device specified for (s2), and stores the results to (d) and the following devices.

• When a device is specified for (s2)



• When a constant is specified for (s2)



- Block subtraction is performed in units of 32 bits.
- The constant can be specified between -2147483648 and 2147483647 (BIN 32 bits) for (s2).
- No processing is performed if the value specified for n is 0.
- (d) is specified in the device range other than n points from (s1) and n points from (s2). Note that a same device can be specified for (s1) and (s2).

- The following is the result when an overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.

· K2147483647 -K-2 → K-2147483647
(7FFFFFFFH) (00000002H) (80000001H)

· K-2147483647 -K2 → K2147483647
(80000001H) (FFFFFFFEH) (7FFFFFFFH)

Operation error

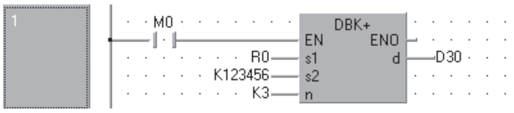
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	A negative value is specified for n.	—	—	—	—	○	○
4101	The range of n points from the device specified for (s1), (s2) or (d) exceeds the corresponding device. The range of n points from the device specified for (s1) overlaps with the range of n points from the device specified for (d). (Except when the same device is assigned to (s1) and (d)) The range of n points from the device specified for (s2) overlaps with the range of n points from the device specified for (d). (Except when the same device is specified for (s2) and (d))	—	—	—	—	○	○

Program example

- In the following program, the data stored in R0 to R5 are added to the constant when M0 turns ON, and the results are stored to D30 to D35.

[Structured ladder/FBD]

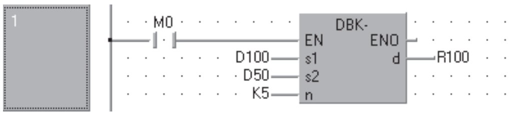


[Operation]

	b31 b0					b31 b0
R1,R0	600000				D31,D30	723456
R3,R2	-800000	+	123456	⇨	D33,D32	-676544
R5,R4	-123456				D35,D34	0

- In the following program, the data stored in D50 to D59 are subtracted from the data in D100 to D109 when M0 turns ON, and the results are stored to R100 to R109.

[Structured ladder/FBD]



[Operation]

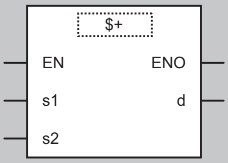
	b31 b0			b31 b0			b31 b0
D101,D100	12345		D51,D50	11111		R101,R100	1234
D103,D102	54321		D53,D52	-11111	⇨	R103,R102	65432
D105,D104	-12345	—	D55,D54	22222		R105,R104	-34567
D107,D106	-54321		D57,D56	-22222		R107,R106	-32099
D109,D108	99999		D59,D58	33333		R109,R108	66666

Character string data concatenation

$+(P)

~~Basic~~ High performance Process Redundant Universal LCPU

Structured ladder/FBD



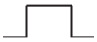
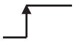
ST

Not supported

Any of the following instruction can go in the dotted squares.

$+, $+P

■Executing condition

Instruction	Executing condition
$+	
$+P	

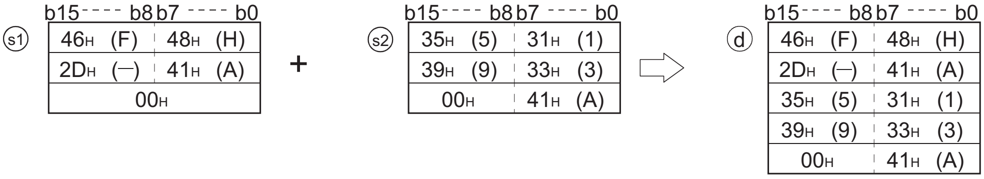
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Connection data, or start number of the device that stores connection data	String
	s2	Data to be connected, or start number of the device that stores data to be connected	String
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	String

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				○	—
(s2)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Connects the character string data specified for (s2) after the character string data specified for (s1) and stores the result to the variable specified for (d).



- This instruction ignores the 00H which indicates the end of character string data specified for (s1), and connects the character string specified for (s2) is appended to the last character of the (s1) string.

Operation error

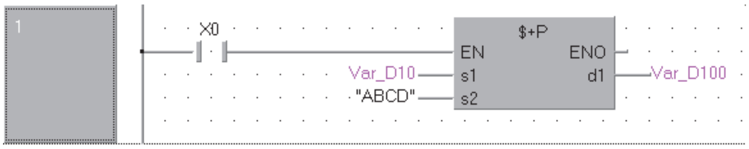
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The number of device points starting from the device specified in (d) is insufficient to store all character strings. The device numbers specified by (s2) and (d) overlap. The character string of (s1), (s2) and (d) exceeds 16383 characters.	—	○	○	○	○	○

Program example

- In the following program, the character string "ABCD" is connected to the character string stored in Var_D10 when X0 turns ON, and the results are stored to Var_D100 and the following devices.

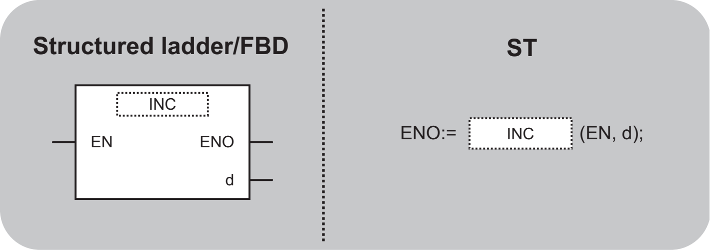
[Structured ladder/FBD]



16-bit BIN data increment and decrement

INC(P), DEC(P)

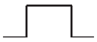
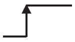
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

INC, INCP, DEC, DECP

■Executing condition

Instruction	Executing condition
INC DEC	
INCP DECP	

■Argument

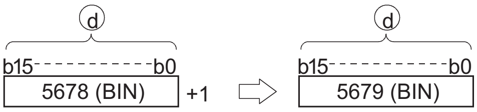
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Start number of the device (16-bit data) to be incremented or decremented by 1	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(d)	○							—	

Processing details

■INC(P)

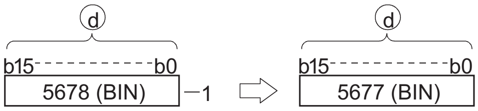
- Adds 1 to the variable (16-bit data) specified for (d).



- When the instruction is executed with the variable specified for (d), which value is 32767, the value -32768 is stored to the variable specified for (d).

■DEC(P)

- Adds -1 to the variable (16-bit data) specified for (d).



- When the instruction is executed with the variable specified for (d), which value is -32768, the value 32767 is stored to the device specified for (d).

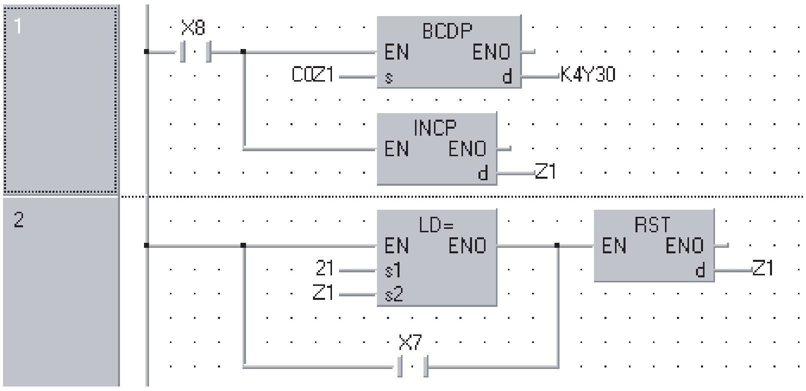
Operation error

- There is no operation error.

Program example

- In the following program, the current values from the counter C0 to C20 are output to Y30 to Y3F in BCD data each time X8 turns ON. (When current value is less than 9999)

[Structured ladder/FBD]



Outputs the current value of C (0+Z1) to Y30 to Y3F in BCD data.

Increments the value in Z1.

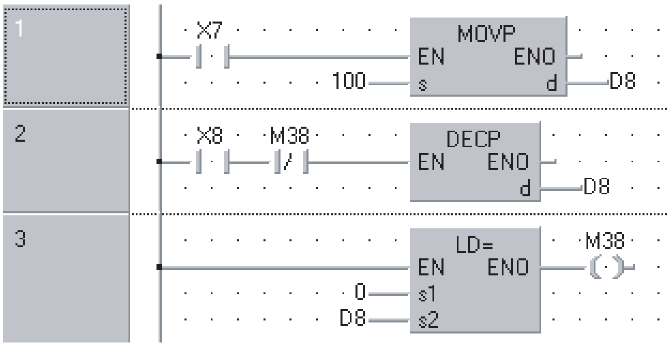
When the value in Z1 reaches 21, or by resetting X7, 0 is set to Z1.

[ST]

```
BCDP(X8,C0Z1,K4Y30);
INCP(X8,Z1);
IF Z1=21 OR X7 THEN
    RST(TRUE,Z1);
END_IF;
```

- The following is a down-counter program.

[Structured ladder/FBD]



Transfers 100 to D8 when X7 turns ON.

When M38 is OFF, decrements the value in D8 each time X8 turns from OFF to ON.

M38 turns ON when the value in D8 reaches 0.

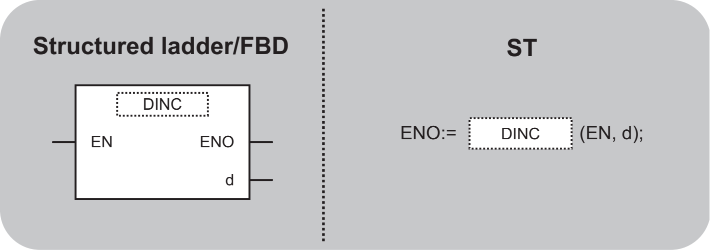
[ST]

```
MOVP(X7,100,D8);
IF X8 AND NOT(M38) THEN
    DECP(TRUE,D8);
END_IF;
OUT(D8=0,M38);
```

32-bit BIN data increment and decrement

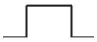
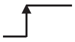
DINC(P), DDEC(P)

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.
DINC, DINCP, DDEC, DDECP

■Executing condition

Instruction	Executing condition
DINC DDEC	
DINCP DDECP	

■Argument

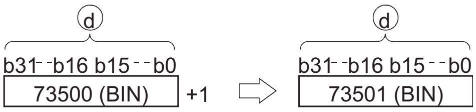
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Start number of the device (32-bit data) to be incremented or decremented by 1	ANY32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(d)	○							—	

Processing details

■DINC(P)

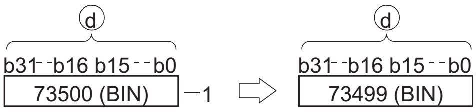
- Adds 1 to the variable (32-bit data) specified for (d).



- When the instruction is executed with the variable specified for (d), which value is 2147483647, the value -2147483648 is stored to the variable specified for (d).

■DDEC(P)

- Adds -1 to the variable (32-bit data) specified for (d).



- When the instruction is executed with the variable specified for (d), which value is 0, the value -1 is stored to the variable specified for (d).

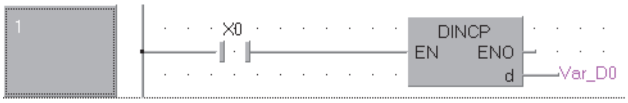
Operation error

- There is no operation error.

Program example

- In the following program, 1 is added to the data in Var_D0 when X0 turns ON.

[Structured ladder/FBD]

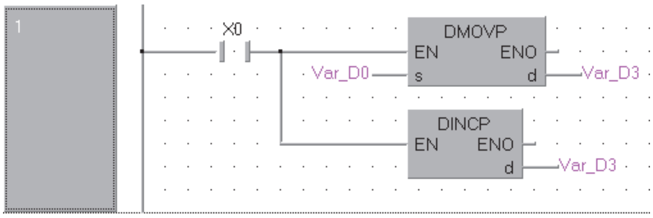


[ST]

```
DINCP(X0,Var_D0);
```

- In the following program, 1 is added to the data set in Var_D0 when X0 turns ON, and the result is stored to Var_D3.

[Structured ladder/FBD]

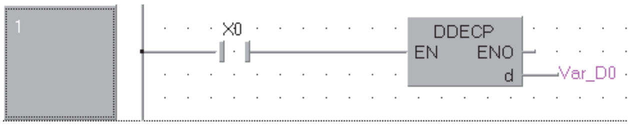


[ST]

```
DMOVP(X0,Var_D0,Var_D3);
DINCP(X0,Var_D3);
```

- In the following program, -1 is added to the data in Var_D0 when X0 turns ON.

[Structured ladder/FBD]

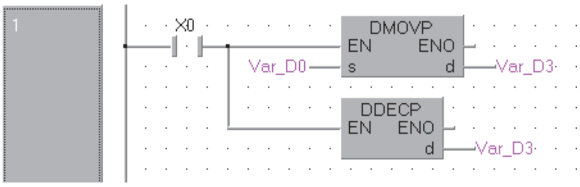


[ST]

```
DDECP(X0,Var_D0);
```

- In the following program, -1 is added to the data set in Var_D0 when X0 turns ON, and the result is stored to Var_D3.

[Structured ladder/FBD]



[ST]

```
DMOVP(X0,Var_D0,Var_D3);
DDECP(X0,Var_D3);
```

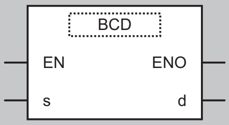
6.3 Data Conversion Instructions

BIN data to 4-/8-digit BCD data conversion

BCD(P), DBCD(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



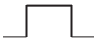
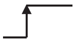
ST

ENO:= BCD (EN, s, d);

Any of the following instruction can go in the dotted squares.

BCD, BCDP, DBCD, DBCDP

■Executing condition

Instruction	Executing condition
BCD DBCD	
BCDP DBCDP	

■Argument

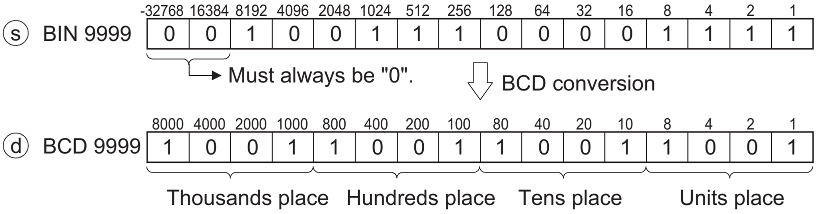
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores BIN data	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores BCD data after conversion	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○							○	—
(d)	○							—	—

Processing details

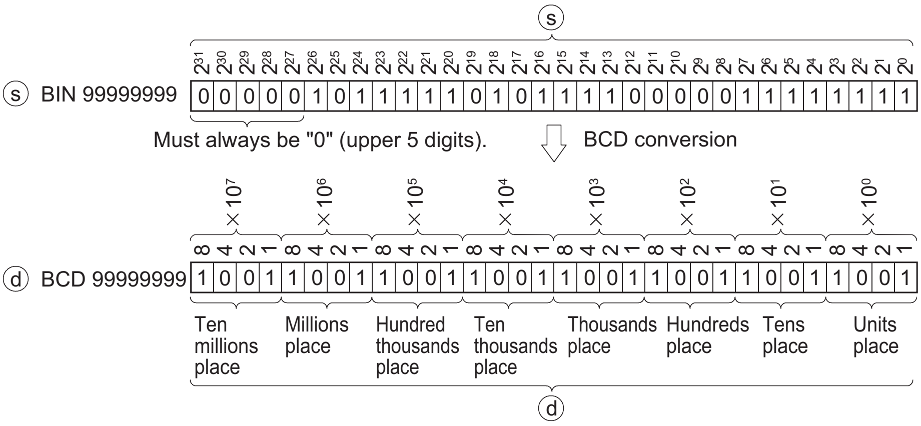
■BCD(P)

- Converts BIN data (0 to 9999) in the device specified for (s) to BCD data, and stores the result to the device specified for (d).



■DBCD(P)

- Converts BIN data (0 to 99999999) in the device specified for (s) to BCD data, and stores the result to the device specified for (d).



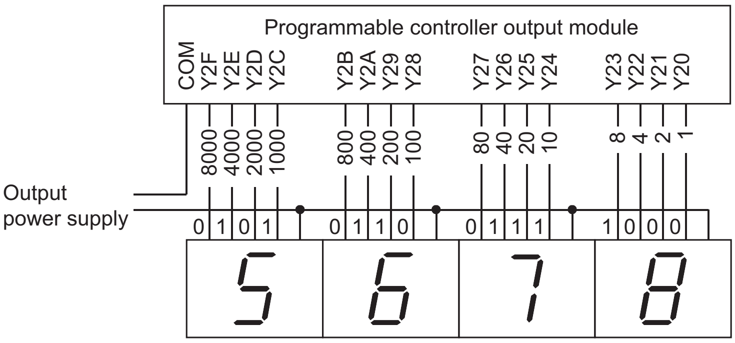
Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The data of (s) is other than 0 to 9999 at the execution of the BCD(P) instruction.	○	○	○	○	○	○
	The data of (s)+1 and (s) is other than 0 to 99999999 at the execution of the DBCD(P) instruction.	○	○	○	○	○	○

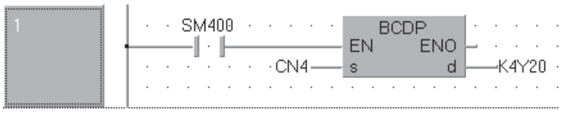
Program example

- In the following program, the current value of C4 from Y20 to Y2F is output to the BCD display device.



7-segment display unit

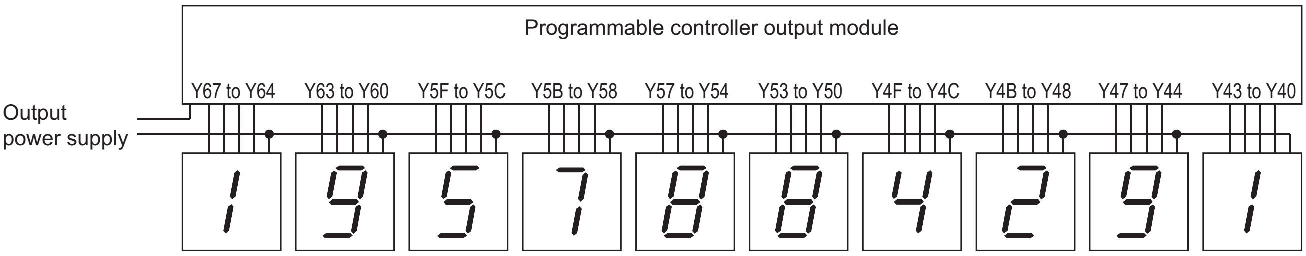
[Structured ladder/FBD]



[ST]

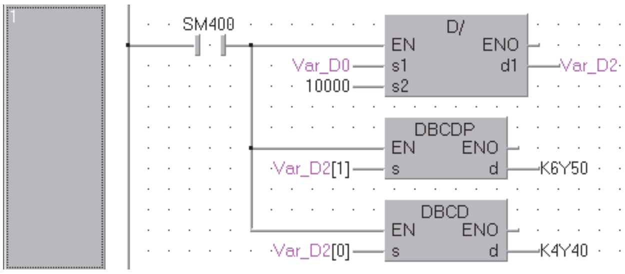
```
BCDP(SM400,CN4,K4Y20);
```

- In the following program, 32-bit data from D0 to D1 are output to devices from Y40 to Y67.



7-segment display unit

[Structured ladder/FBD]



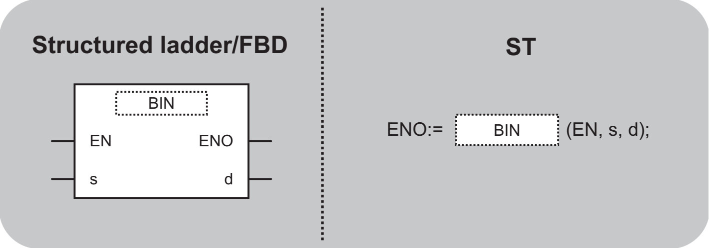
[ST]

```
IF SM400 THEN
    Var_D2[0]:=Var_D0/10000;
    DBCDP(TRUE,Var_D2[1],K6Y50);
    DBCD(TRUE,Var_D2[0],K4Y40);
END_IF;
```

4-/8-digit BCD data to BIN data conversion

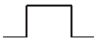
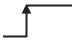
BIN(P), DBIN(P)

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.
BIN, BINP, DBIN, DBINP

■Executing condition

Instruction	Executing condition
BIN DBIN	
BINP DBINP	

■Argument

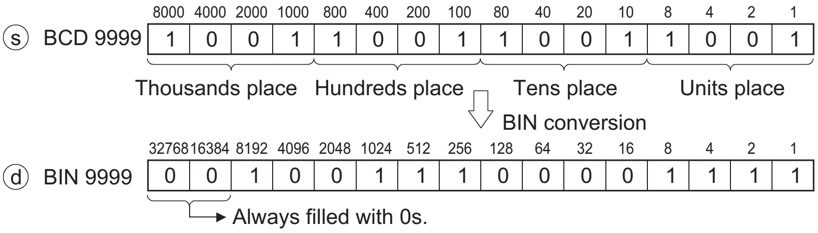
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores BIN data	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores BCD data after conversion	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○							○	—
(d)	○							—	—

Processing details

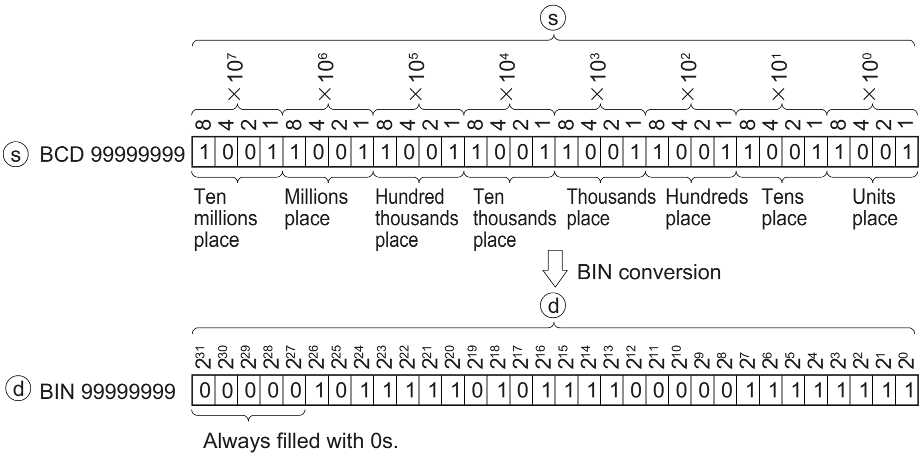
■BIN(P)

- Converts the BCD data (0 to 9999) in the device specified for (s) to the BIN data, and stores the result to the device specified for (d).



■DBIN(P)

- Converts the BCD data (0 to 99999999) in the device specified for (s) to the BIN data, and stores the result to the device specified for (d).



Operation error

- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	Values other than 0 to 9 are specified for each digit of (S).	○	○	○	○	○	○

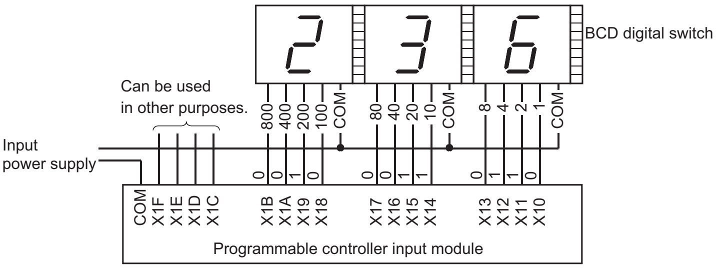
The error above can be suppressed by turning ON SM722.

However, the instruction is not executed regardless of whether SM722 is turned ON or OFF if the specified value is out of the available range.

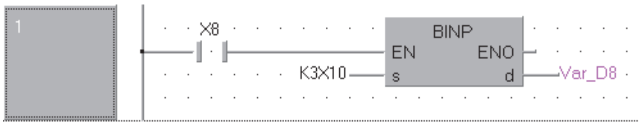
For the BINP and DBINP instruction, the next operation will not be performed until the command (executing condition) is turned from OFF to ON regardless of the presence or absence of an error.

Program example

- In the following program, the BCD data in X10 to X1B are converted to the BIN data when X8 turns ON, and the result is stored to Var_D8.



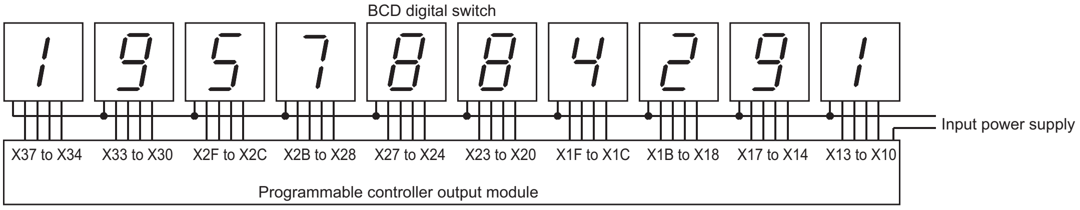
[Structured ladder/FBD]



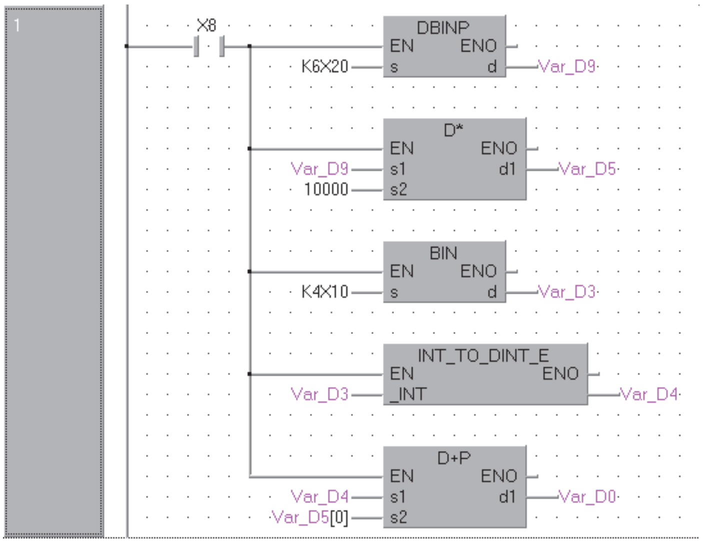
[ST]

```
BINP(X8,K3X10,Var_D8);
```

- In the following program, the BCD data in X10 to X37 are converted to BIN data when X8 turns ON, and the result is stored to Var_D0. (Addition of the BCD data in X20 to X37 converted to BIN data and the BCD data in X10 to X1F converted to BIN data.) However, when BCD value exceeding 2147483647 is specified in X10 to X37, Var_D0 becomes a negative value because the value is beyond the range a 32-bit device can handle.



[Structured ladder/FBD]



[ST]

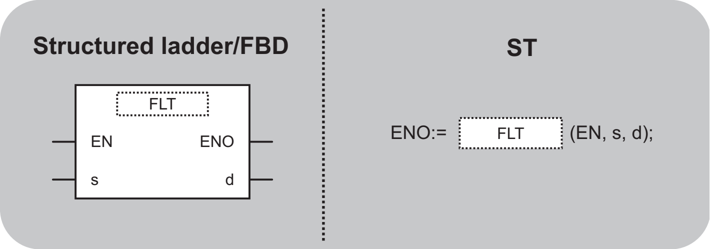
```
IF X8 THEN
    DBINP(TRUE,K6X20,Var_D9);
    Var_D5[0]:=Var_D9*10000;
    BIN(TRUE,K4X10,Var_D3);
    INT_TO_DINT_E(TRUE,Var_D3,Var_D4);
    Var_D0:=Var_D4+Var_D5[0];
END_IF;
```

16-/32-bit BIN data to floating-point data conversion (single precision)

FLT(P), DFLT(P)

Ver. Basic High performance Process Redundant Universal LCPU

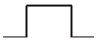
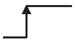
- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later



Any of the following instruction can go in the dotted squares.

FLT, FLTP, DFLT, DFLTP

■Executing condition

Instruction	Executing condition
FLT DFLT	
FLTP DFLTP	

■Argument

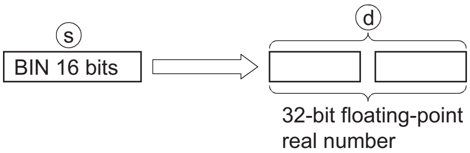
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Integer data to be converted to 32-bit floating-point data, or start number of the device that stores integer data	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted 32-bit floating-point data	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○	○		○	○		○		—
(d)	—	○		—	○		—		—

Processing details

■FLT(P)

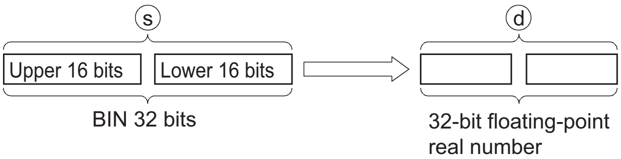
- Converts 16-bit BIN data specified for (s) to the 32-bit floating-point real number, and stores the result to the device specified for (d).



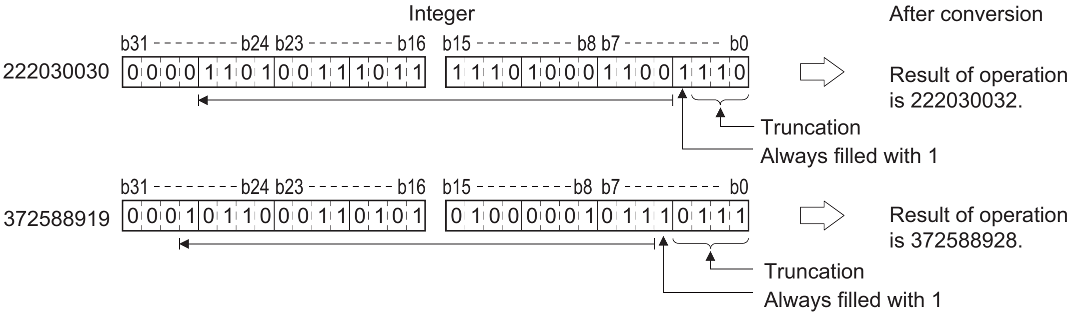
- BIN values between -32768 and 32767 can be specified for (s).

■DFLT(P)

- Converts 32-bit BIN data specified for (s) to the 32-bit floating-point real number, and stores the result to the device specified for (d).



- BIN values between -2147483648 and 2147483647 can be specified for (s).
- Due to the fact that 32-bit floating-point real numbers are processed by 32-bit single precision, the number of significant figures is 24 bits if the display is binary and approximately 7 digits if the display is decimal. For this reason, if the integer value exceeds the range of -16777216 to 16777215 (24-bit BIN value), errors can be generated in the conversion value. As for the conversion result, the 25th bit from the highest bit of the integer value is rounded off and the 26th bit and later are truncated.



Integer
After conversion
b31 - - - - - - - - b24 b23 - - - - - - - - b16 b15 - - - - - - - - - b8 b7 - - - - - - - - - b0
222030030
0 0 0 0 1 1 0 1 0 0 1 1 1 0 1 1 1 1 1 0 1 0 0 0 1 1 0 0 1 1 1 0
Result of operation is 222030032.
Truncation
Always filled with 1
b31 - - - - - - - - b24 b23 - - - - - - - - b16 b15 - - - - - - - - - b8 b7 - - - - - - - - - b0
372588919
0 0 0 1 0 1 1 0 0 0 1 1 0 1 0 1 0 1 0 0 0 0 0 1 0 1 1 1 0 1 1 1
Result of operation is 372588928.
Truncation
Always filled with 1

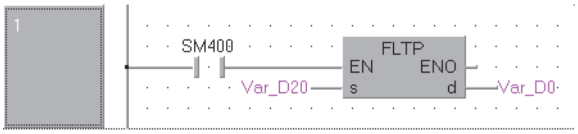
Operation error

- There is no operation error.

Program example

- In the following program, the 16-bit BIN value in Var_D20 is converted to the 32-bit floating-point real number, and the result is stored to Var_D0.

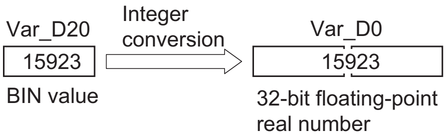
[Structured ladder/FBD]



[ST]

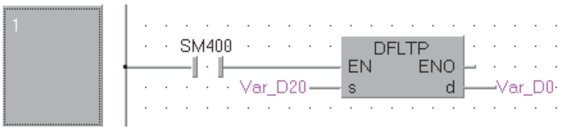
FLTP(SM400,Var_D20,Var_D0);

[Operation]



- In the following program, the 32-bit BIN value in Var_D20 is converted to the 32-bit floating point real number, and the result is stored to Var_D0.

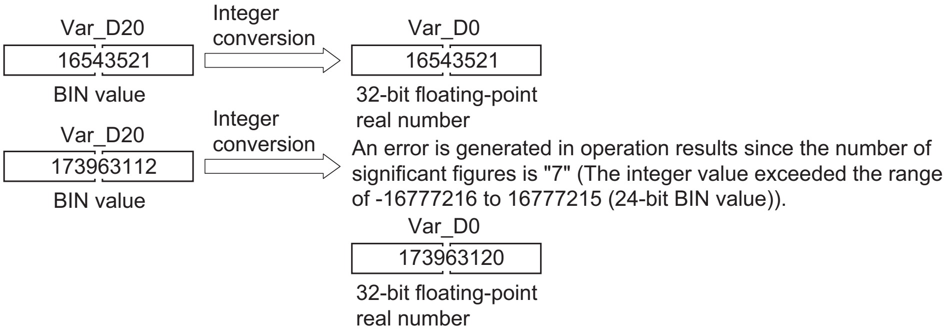
[Structured ladder/FBD]



[ST]

DFLTP(SM400,Var_D20,Var_D0);

[Operation]

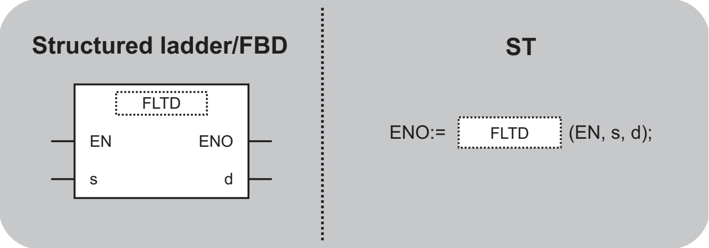


An error is generated in operation results since the number of significant figures is "7" (The integer value exceeded the range of -16777216 to 16777215 (24-bit BIN value)).

16-/32-bit BIN data to floating-point data conversion (double precision)

FLTD(P), DFLTD(P)

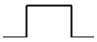
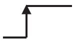
Basic High performance Process Redundant Universal LCPU



ENO:= FLTD (EN, s, d);

Any of the following instruction can go in the dotted squares.
FLTD, FLTDP, DFLTD, DFLTDP

■Executing condition

Instruction	Executing condition
FLTD DFLTD	
FLTDP DFLTDP	

■Argument

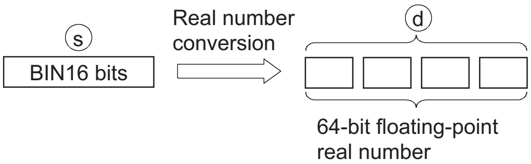
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Integer data to be converted to 64-bit floating-point data, or start number of the device that stores integer data	Word [signed]/double word [signed]
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted 64-bit floating-point data	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—			○		—
(d)	—	○		—			—		—

Processing details

■FLTD(P)

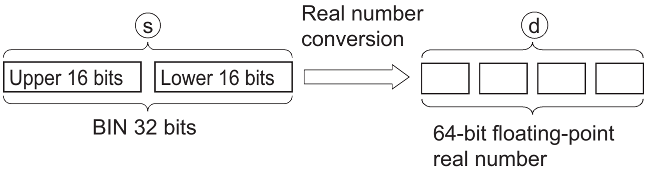
- Converts 16-bit BIN data specified for (s) to a 64-bit floating-point real number, and stores the result to the device specified for (d).



- BIN values between -32768 and 32767 can be specified for (s).

■DFLTD(P)

- Converts 32-bit BIN data specified for (s) to a 64-bit floating-point real number, and stores the result to (d).



- BIN values between -2147483648 and 2147483647 can be specified for (s), (s) + 1.

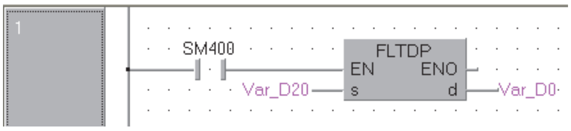
Operation error

- There is no operation error.

Program example

- In the following program, the 16-bit BIN value in Var_D20 is converted to the 64-bit floating-point real number, and the result is stored to Var_D0.

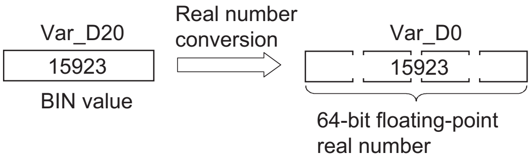
[Structured ladder/FBD]



[ST]

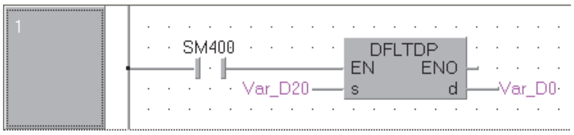
FLTDP(SM400,Var_D20,Var_D0);

[Operation]



- In the following program, the 64-bit BIN value in Var_D20 is converted to the 32-bit floating point real number, and the result is stored to Var_D0.

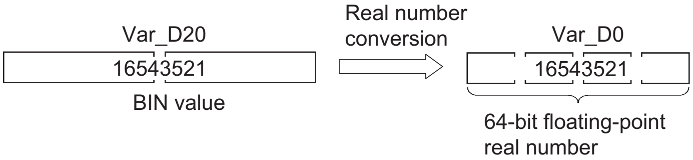
[Structured ladder/FBD]



[ST]

DFLTDP(SM400,Var_D20,Var_D0);

[Operation]

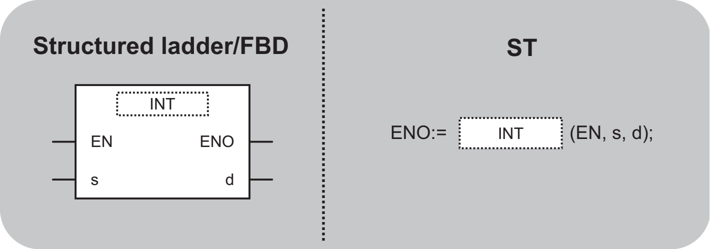


Floating-point data to 16-/32-bit BIN data conversion (single precision)

INT(P), DINT(P)

Ver. Basic High performance Process Redundant Universal LCPU

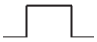
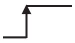
- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later



Any of the following instruction can go in the dotted squares.

INT, INTP, DINT, DINTP

■Executing condition

Instruction	Executing condition
INT DINT	
INTP DINTP	

■Argument

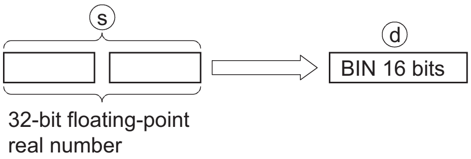
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Floating-point data to be converted to BIN value, or start number of the device that stores floating-point data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted BIN value	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	○	○		○	○		○	—	—

Processing details

■INT(P)

- Converts the 32-bit floating-point real number specified for (s) to 16-bit BIN data and stores the result to the device specified for (d).

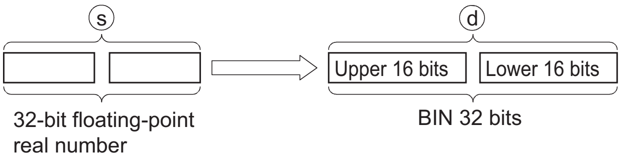


- 32-bit floating-point real numbers between -32768 and 32767 can be specified for (s).
- The integer value stored in (d) is stored as 16-bit BIN values.
- After conversion, the first digit from the decimal point of real number is rounded off.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

■DINT(P)

- Converts the 32-bit floating-point real number specified for (s) to 32-bit BIN data and stores the result to the device specified for (d).



- 32-bit floating-point real numbers between -2147483648 and 2147483647 can be specified for (s).
- The integer value stored in (d) is stored as BIN 32 bits.
- After conversion, the first digit from the decimal point of real number is rounded off.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

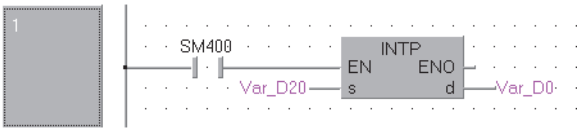
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-126} \leq$ | value of the specified device | $< 2^{128}$ The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○
4100	The 32-bit floating-point data specified for (s) is outside the range of -32768 to 32767 for the INT instruction.	○	○	○	○	○	○
	The 32-bit floating-point data specified for (s) is outside the range of -2147483648 to 2147483647 for the DINT instruction.	○	○	○	○	○	○

Program example

- In the following program, the 32-bit floating-point real number in Var_D20 is converted to the 16-bit BIN value, and the result is stored to Var_D0.

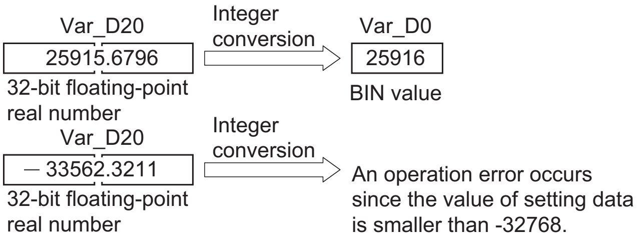
[Structured ladder/FBD]



[ST]

INTP(SM400,Var_D20,Var_D0);

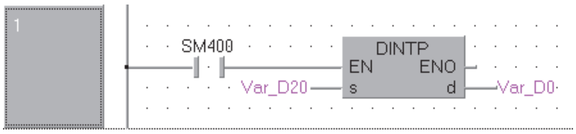
[Operation]



An operation error occurs since the value of setting data is smaller than -32768.

- In the following program, the 32-bit floating-point real number in Var_D20 is converted to the 32-bit BIN value, and the result is stored to Var_D0.

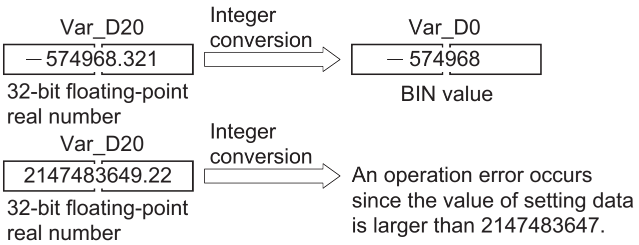
[Structured ladder/FBD]



[ST]

DINTP(SM400,Var_D20,Var_D0);

[Operation]



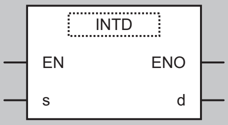
An operation error occurs since the value of setting data is larger than 2147483647.

Floating-point data to 16-bit/32-bit BIN data conversion (double precision)

INTD(P), DINTD(P)

~~Basic~~ ~~High performance~~ ~~Process~~ ~~Redundant~~ Universal LCPU

Structured ladder/FBD



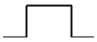
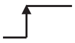
ST

ENO:= INTD (EN, s, d);

Any of the following instruction can go in the dotted squares.

INTD, INTDP, DINTD, DINTDP

■Executing condition

Instruction	Executing condition
INTD DINTD	
INTDP DINTDP	

■Argument

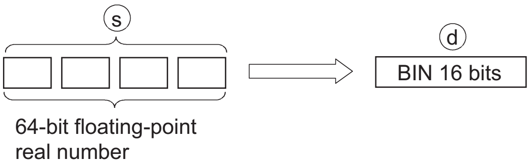
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	64-bit floating-point data to be converted to BIN value, or start number of the device that stores 64-bit floating-point data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted BIN value	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—			—	○	—
(d)	—	○		—			○	—	—

Processing details

■INTD(P)

- Converts the 64-bit floating-point real number specified for (s) to 16-bit BIN data and stores the result to the device specified for (d).

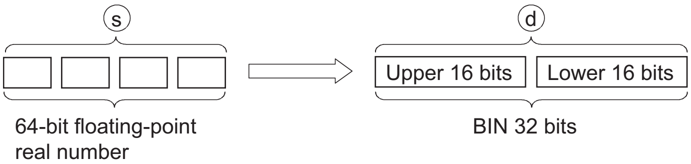


- 64-bit floating-point real numbers between -32768.0 and 32767.0 can be specified for (s) + 3, (s) + 2, (s) + 1 or (s).
- The integer value stored in (d) is stored as 16-bit BIN values.
- After conversion, the first digit from the decimal point of the 64-bit floating-point real number is rounded off.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

■DINTD(P)

- Converts the 64-bit floating-point real number specified for (s) to 32-bit BIN data and stores the result to the device specified for (d).



- 64-bit floating-point real numbers between -2147483648.0 and 2147483647.0 can be specified for (s) + 3, (s) + 2, (s) + 1 or (s).
- The integer value stored in (d) + 1 and (d) is stored as BIN 32 bits.
- After conversion, the first digit from the decimal point of the 64-bit floating-point real number is rounded off.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

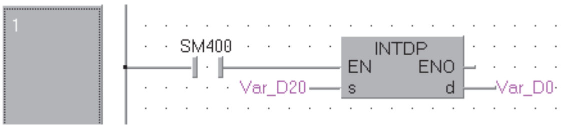
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | $< 2^{1024}$ The value of the specified device is -0, unnormalized number, nonnumeric, or ±∞.	—	—	—	—	○	○
4100	The 64-bit floating-point data set for (s) is outside the range of -32768 to 32767 for the INTD instruction.	—	—	—	—	○	○
	The 64-bit floating-point data set for (s) is outside the range of -2147483648 to 2147483647 for the DINTD instruction.	—	—	—	—	○	○

Program example

- In the following program, the 64-bit floating-point real number in Var_D20 is converted to the 16-bit BIN value, and the result is stored to Var_D0.

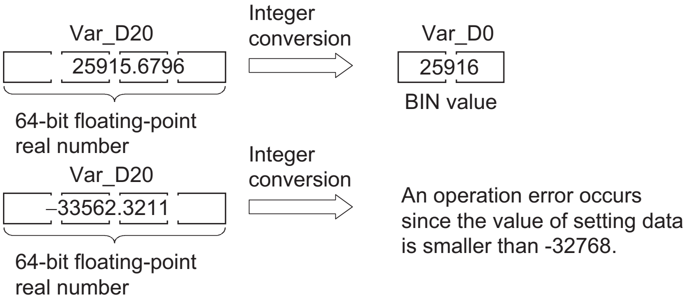
[Structured ladder/FBD]



[ST]

INTDP(SM400,Var_D20,Var_D0);

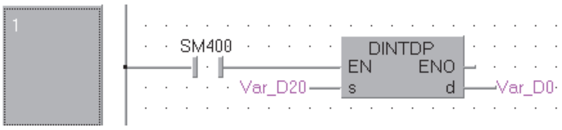
[Operation]



An operation error occurs since the value of setting data is smaller than -32768.

- In the following program, the 64-bit floating-point real number in Var_D20 is converted to the 32-bit BIN value, and the result is stored to Var_D0.

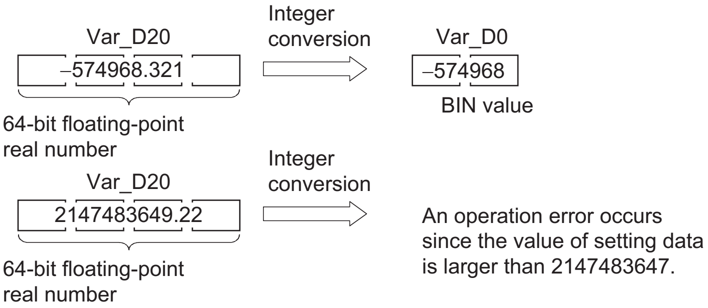
[Structured ladder/FBD]



[ST]

DINTDP(SM400,Var_D20,Var_D0);

[Operation]

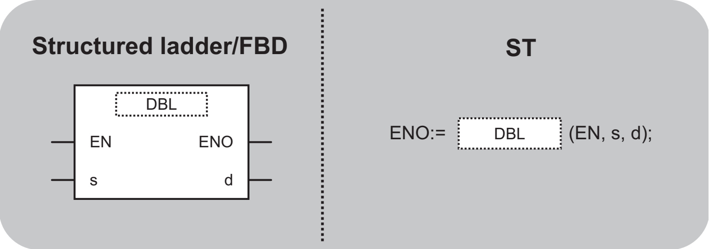


An operation error occurs since the value of setting data is larger than 2147483647.

16-bit BIN data to 32-bit BIN data conversion

DBL(P)

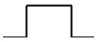
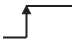
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

DBL, DBLP

■Executing condition

Instruction	Executing condition
DBL	
DBLP	

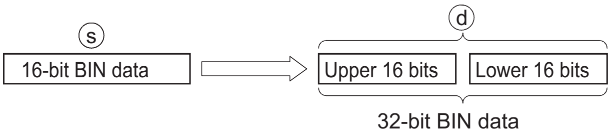
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	16-bit BIN data, or start number of the device that stores 16-bit BIN data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted 32-bit BIN data	ANY32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○							○	—
(d)	○							—	—

Processing details

- Converts 16-bit BIN data in the device specified for (s) to 32-bit BIN data with a sign, and stores the result to the device specified for (d).



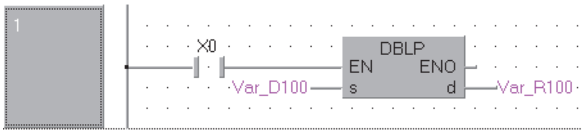
Operation error

- There is no operation error.

Program example

- In the following program, the 16-bit BIN value in Var_D100 is converted to the 32-bit BIN value when X20 turns ON, and the result is stored to Var_R100.

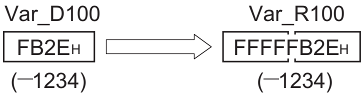
[Structured ladder/FBD]



[ST]

DBLP(X20,Var_D100,Var_R100);

[Operation]

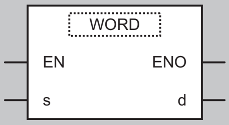


32-bit BIN data to 16-bit BIN data conversion

WORD(P)

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



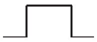
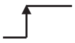
ST

ENO:= WORD (EN, s, d);

Any of the following instruction can go in the dotted squares.

WORD, WORDP

■Executing condition

Instruction	Executing condition
WORD	
WORDP	

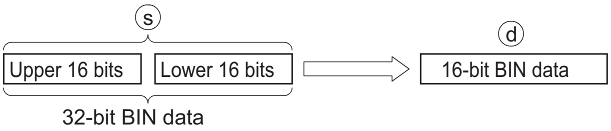
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	32-bit BIN data, or start number of the device that stores 32-bit BIN data	ANY32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted 16-bit BIN data	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○							○	—
(d)	○							—	—

Processing details

- Converts 32-bit BIN data in the device specified for (s) to 16-bit BIN data with a sign, and stores the result to the device specified for (d).
- Devices can be specified in the range from -32768 to 32767.



Operation error

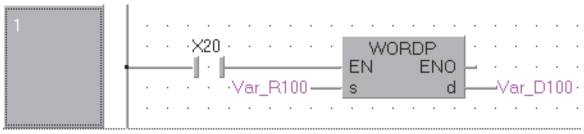
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of the device specified for (s) is outside the range of -32768 to 32767.	○	○	○	○	○	○

Program example

- In the following program, the 32-bit BIN value in Var_R100 is converted to the 16-bit BIN value when X20 turns ON, and the result is stored to Var_D100.

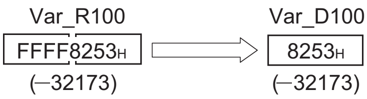
[Structured ladder/FBD]



[ST]

WORDP(X20,Var_R100,Var_D100);

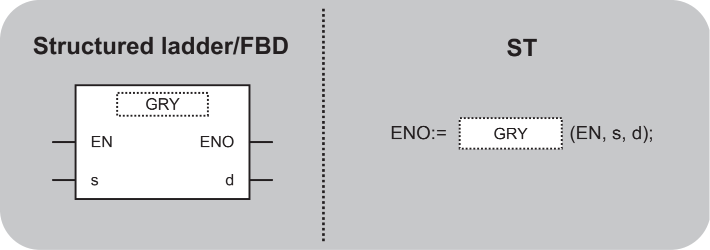
[Operation]



16-/32-bit BIN data to Gray code conversion

GRY(P), DGRY(P)

Basic **High performance** **Process** **Redundant** **Universal** **LCPU**

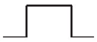
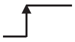


```
ENO:= GRY (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

GRY, GRYP, DGRY, DGRYP

■Executing condition

Instruction	Executing condition
GRY DGRY	
GRYP DGRYP	

■Argument

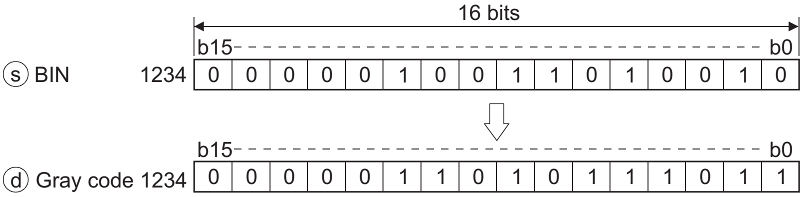
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	BIN data, or start number of the device that stores BIN data	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted Gray code	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○							○	—
(d)	○							—	—

Processing details

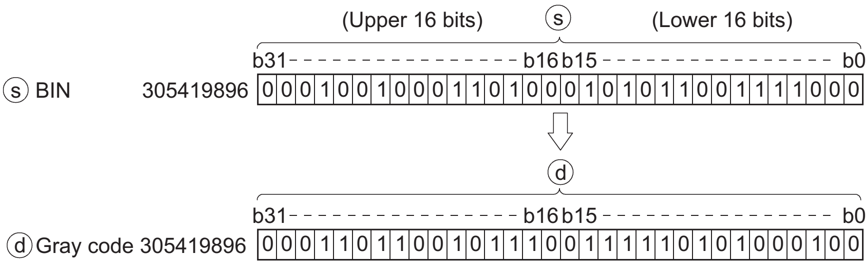
■GRY(P)

- Converts 16-bit BIN data in the device specified for (s) to Gray code, and stores the result to the device specified for (d). Negative values cannot be specified for (S).



■DGRY(P)

- Converts 32-bit BIN data in the device specified for (s) to Gray code, and stores the result to the device specified for (d). Negative values cannot be specified for (S).



Operation error

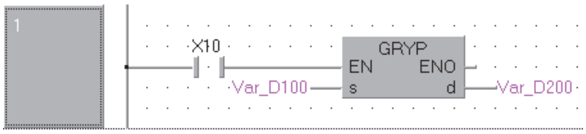
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of (s) is a negative number.	○	○	○	○	○	○

Program example

- In the following program, the BIN value in Var_D100 is converted to Gray code when X10 turns ON, and the result is stored to Var_D200.

[Structured ladder/FBD]

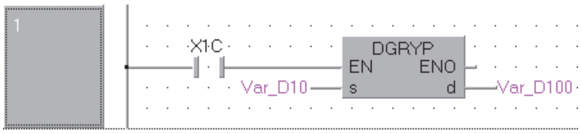


[ST]

```
GRYP(X10,Var_D100,Var_D200);
```

- In the following program, the BIN value in Var_D10 is converted to Gray code when X1C turns ON, and the result is stored to Var_D100.

[Structured ladder/FBD]



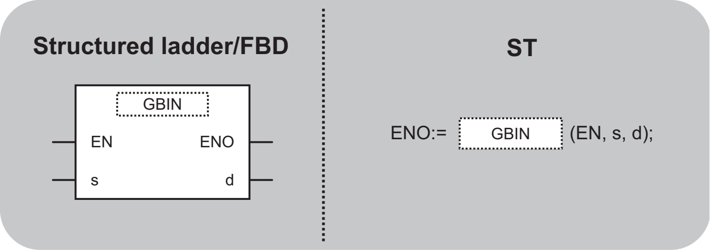
[ST]

```
DGRYP(X1C,Var_D10,Var_D100);
```

Gray code to 16-/32-bit BIN data conversion

GBIN(P), DGBIN(P)

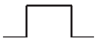
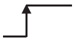
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

GBIN, GBINP, DGBIN, DGBINP

■Executing condition

Instruction	Executing condition
GBIN DGBIN	
GBINP DGBINP	

■Argument

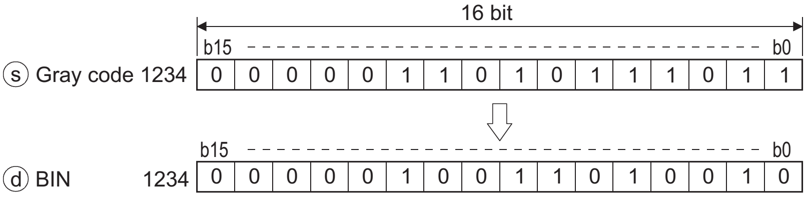
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Gray code data, or start number of the device that stores Gray code data	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted BIN data	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○							○	—
(d)	○							—	—

Processing details

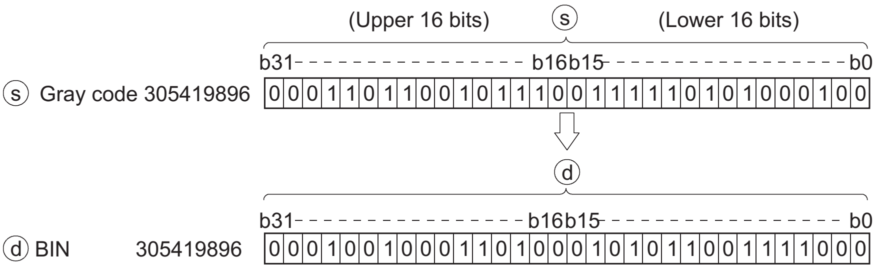
■GBIN(P)

- Converts Gray code data in the device specified for (S) to 16-bit BIN data and stores the result to the device specified for (d).



■DGBIN(P)

- Converts Gray code data in the device specified for (S) to 32-bit BIN data and stores the result to the device specified for (d).



Operation error

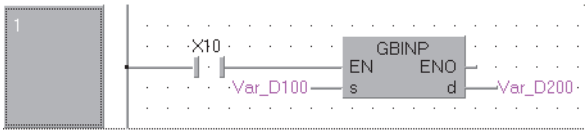
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of (s) is outside the range of 0 to 32767 for the GBIN(P) instruction.	○	○	○	○	○	○
	The value of (s) is outside the range of 0 to 2147483647 for the DGBIN(P) instruction.	○	○	○	○	○	○

Program example

- In the following program, the Gray code in Var_D100 is converted to the BIN value when X10 turns ON, and the result is stored to Var_D200.

[Structured ladder/FBD]

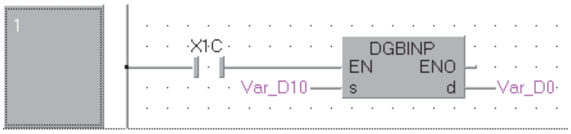


[ST]

GBINP(X10,Var_D100,Var_D200);

- In the following program, the Gray code in Var_D10 is converted to the BIN value when X1C turns ON, and the result is stored to Var_D0.

[Structured ladder/FBD]



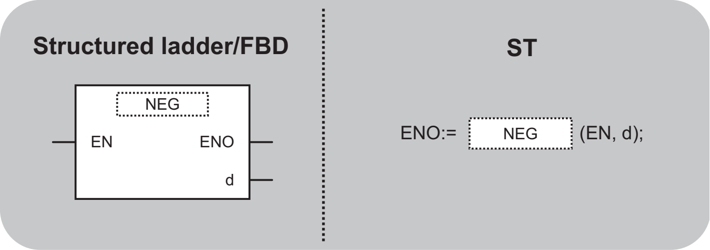
[ST]

DGBINP(X1C,Var_D10,Var_D0);

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

NEG(P), DNEG(P)

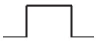
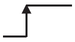
Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

NEG, NEGP, DNEG, DNEGP

■Executing condition

Instruction	Executing condition
NEG DNEG	
NEGP DNEGP	

■Argument

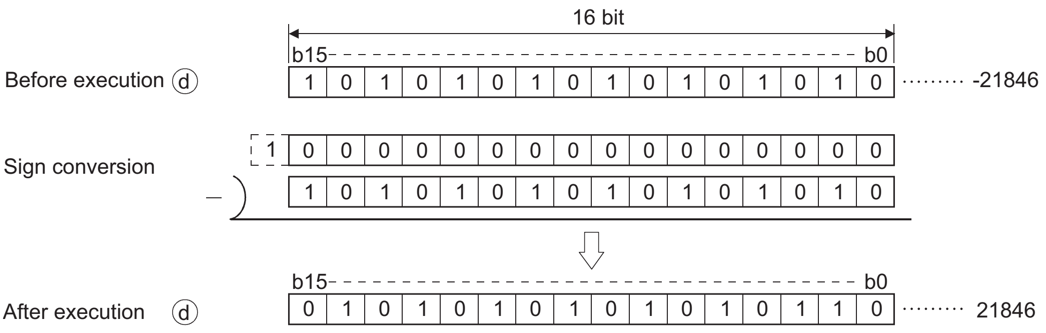
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores data to be converted by the two's complement	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(d)	○							—	

Processing details

■NEG(P)

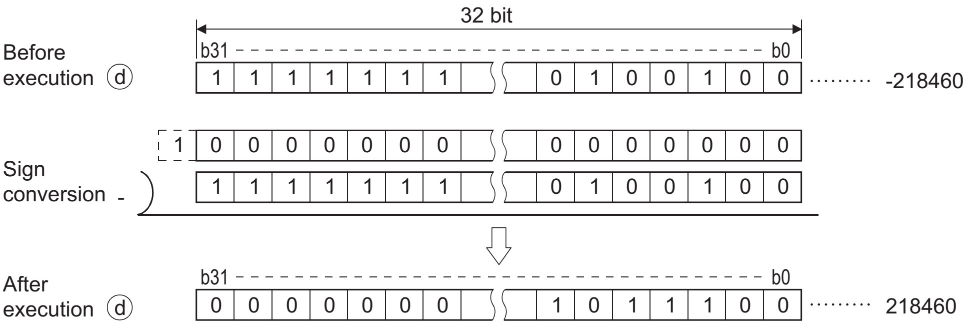
- Inverts the sign of the 16-bit device specified for (d) and stores the result to the device specified for (d).



- This instruction is used when inverting positive and negative signs.

■DNEG(P)

- Inverts the sign of the 32-bit device specified for (d) and stores the result to the device specified for (d).



- This instruction is used when inverting positive and negative signs.

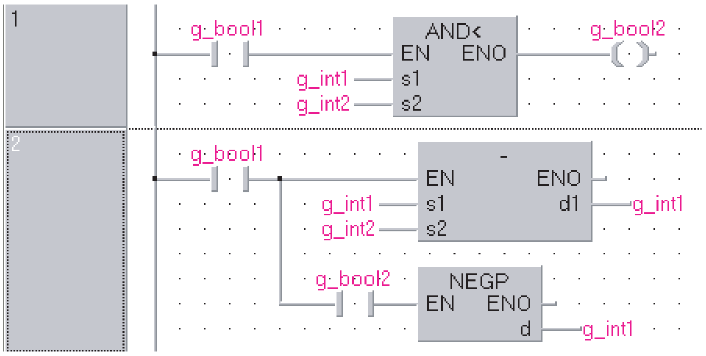
Operation error

- There is no operation error.

Program example

- In the following program, the value in g_int2 is subtracted from the value in g_int1 when g_bool1 turns ON, and if the result is negative, the absolute value is defined.

[Structured ladder/FBD]



When the value in g_int2 is larger than the value in g_int1, g_bool2 turns ON.

The value in g_int2 is subtracted from the value in g_int1.

When g_bool2 is ON, the absolute value (two's complement) is obtained.

[ST]

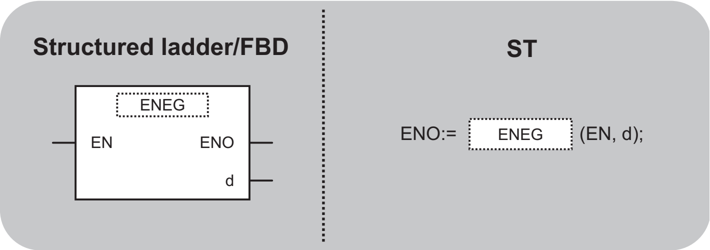
```
IF g_bool1 THEN
    OUT(g_int1<g_int2,g_bool2);
    g_int1:=g_int1-g_int2;
    NEGP(g_bool2,g_int1);
END_IF;
```

Sign inversion of floating-point data (single precision)

ENEG(P)

Ver. Basic High performance Process Redundant Universal LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

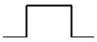
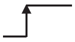


ENO:= ENEG (EN, d);

Any of the following instruction can go in the dotted squares.

ENEG, ENEGP

■Executing condition

Instruction	Executing condition
ENEG	
ENEGP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores 32-bit floating-point data whose sign to be inverted	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(d)	—	○		—	○		—		

Processing details

- Inverts the sign of the 32-bit floating-point real number data specified for (d), and stores the result to the device specified for (d).
- This instruction is used when inverting positive and negative signs.

Operation error

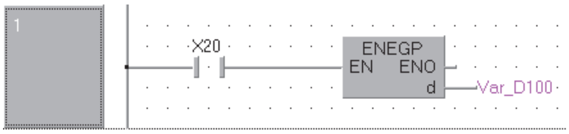
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-126} \leq$ | value of the specified device | < 2^{128} The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○

Program example

- In the following program, the sign of the 32-bit floating-point real number in Var_D100 is inverted when X20 turns ON, and the result is stored to Var_D100.

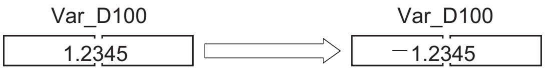
[Structured ladder/FBD]



[ST]

ENEGP(X20,Var_D100);

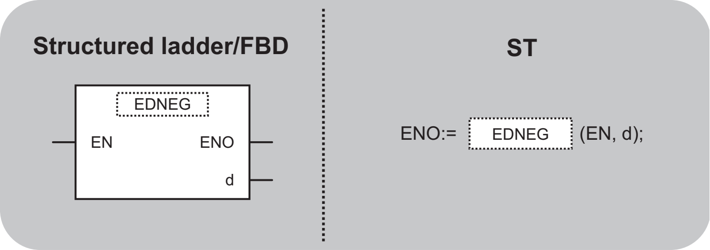
[Operation]



Sign inversion of floating-point data (double precision)

EDNEG(P)

✗Basic ✗High performance ✗Process ✗Redundant Universal LCPU

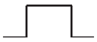
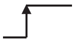


```
ENO:= EDNEG (EN, d);
```

Any of the following instruction can go in the dotted squares.

EDNEG, EDNEGP

■Executing condition

Instruction	Executing condition
EDNEG	
EDNEGP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores 64-bit floating-point data whose sign to be inverted	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(d)	—	○		—					

Processing details

- Inverts the sign of the 64-bit floating-point real number data specified for (d), and stores the result to the device specified for (d).
- This instruction is used when inverting positive and negative signs.

Operation error

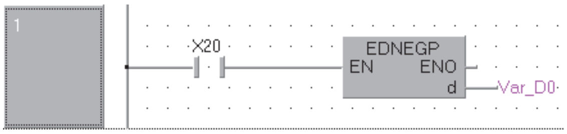
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq \lvert \text{value of the specified device} \rvert < 2^{1024}$ The value of the specified device is -0.	—	—	—	—	○	○

Program example

- In the following program, the sign of the 64-bit floating-point real number in Var_D0 is inverted when X20 turns ON, and the result is stored to Var_D0.

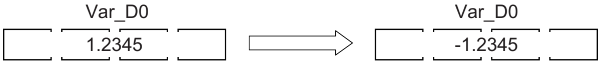
[Structured ladder/FBD]



[ST]

EDNEGP(X20,Var_D0);

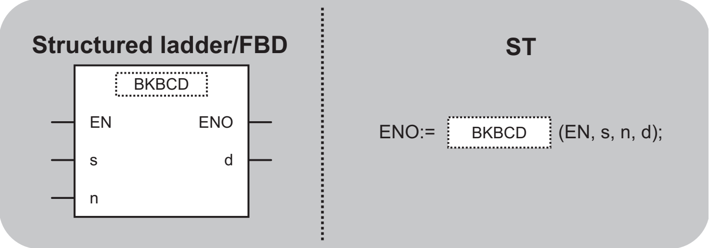
[Operation]



16-bit BIN block data to 4-digit BCD block data conversion

BKBCD(P)

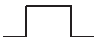
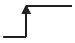
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

BKBCD, BKBCDP

■Executing condition

Instruction	Executing condition
BKBCD	
BKBCDP	

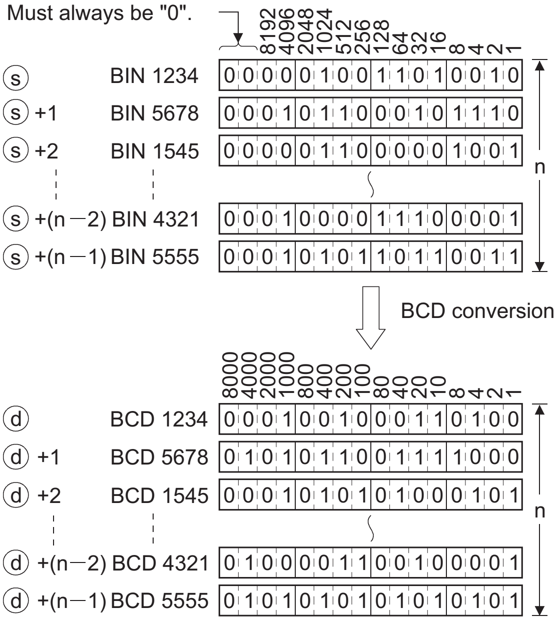
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores BIN data	ANY16
	n	Number of converted data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores BCD data after conversion	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—					—
n	○	○		○					—
(d)	—	○		—					—

Processing details

- Converts the BIN data (0 to 9999) n points from the device specified for (s) to the BCD data, and stores the results to the device specified for (d) and the following devices.


Must always be "0".
8192 4096 2048 1024 512 256 128 64 32 16 8 4 2 1
(s) BIN 1234 0000 0100 1101 0010
(s) +1 BIN 5678 0001 0110 0010 1110
(s) +2 BIN 1545 0000 0110 0000 1001
(s) +(n−2) BIN 4321 0001 0000 1110 0001
(s) +(n−1) BIN 5555 0001 0101 1011 0011
n
BCD conversion
8000 4000 2000 1000 800 400 200 100 80 40 20 10 8 4 2 1
(d) BCD 1234 0001 0010 0011 0100
(d) +1 BCD 5678 0101 0110 0111 1000
(d) +2 BCD 1545 0001 0101 0100 0101
(d) +(n−2) BCD 4321 0100 0011 0010 0001
(d) +(n−1) BCD 5555 0101 0101 0101 0101
n

Operation error

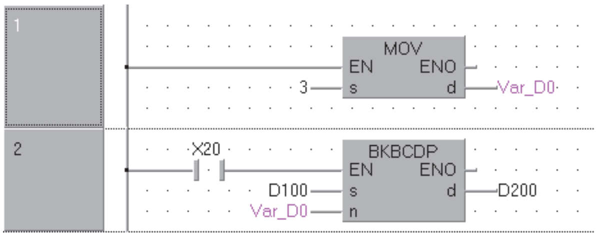
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The data n points from the device specified for (s) is outside the range of 0 to 9999.	○	○	○	○	○	○
4101	The range of n points from the device specified for (s), (d) exceeds the corresponding device. The device ranges of (s) and (d) overlap.	○	○	○	○	○	○

Program example

- In the following program, the BIN values in D100 to D102 are converted to the BCD values when X20 turns ON, and the results are stored to D200 and the following devices.

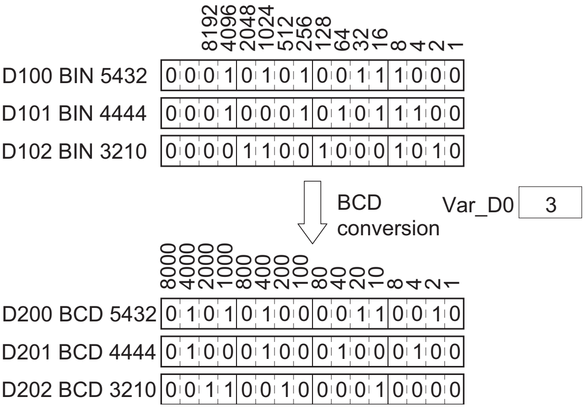
[Structured ladder/FBD]



[ST]

```
Var_D0:=3;
BKBCDP(X20,D100,Var_D0,D200);
```

[Operation]



	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1		
D100 BIN 5432	0	0	0	1	0	1	0	1	0	0	1	1	1	0	0	0
D101 BIN 4444	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	0
D102 BIN 3210	0	0	0	0	1	1	0	0	1	0	0	0	1	0	1	0

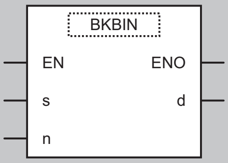
	8000	4000	2000	1000	800	400	200	100	80	40	20	10	8	4	2	1
D200 BCD 5432	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0
D201 BCD 4444	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
D202 BCD 3210	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

4-digit BCD block data to 16-bit BIN block data conversion

BKBIN(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



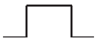
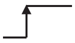
ST

```
ENO:= BKBIN (EN, s, n, d);
```

Any of the following instruction can go in the dotted squares.

BKBIN, BKBINP

■Executing condition

Instruction	Executing condition
BKBIN	
BKBINP	

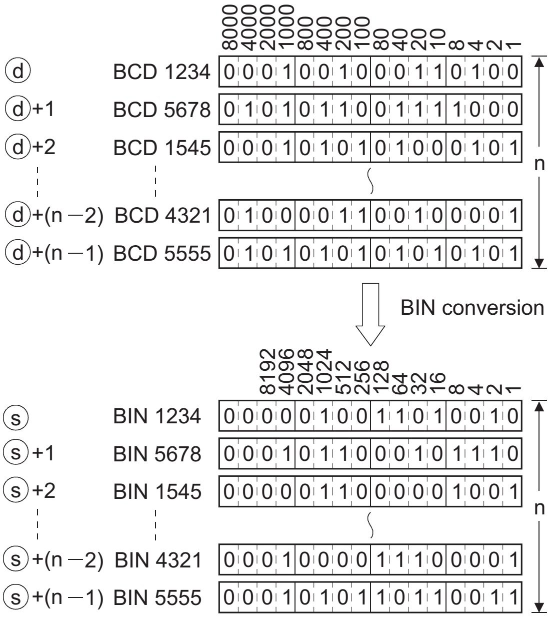
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores BCD data	ANY16
	n	Number of converted data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted BIN data	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—					—
n	○	○		○					—
(d)	—	○		—					—

Processing details

- Converts the BCD data (0 to 9999) n points from the device specified for (s) to the BIN data, and stores the results to the device specified for (d) and the following devices.


8000 4000 2000 1000 800 400 200 100 80 40 20 10 8 4 2 1
ⓓ BCD 1234 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0
ⓓ+1 BCD 5678 0 1 0 1 0 1 1 0 0 1 1 1 1 0 0 0
ⓓ+2 BCD 1545 0 0 0 1 0 1 0 1 0 1 0 0 0 1 0 1
ⓓ+(n−2) BCD 4321 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 1
ⓓ+(n−1) BCD 5555 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
n
BIN conversion
8192 4096 2048 1024 512 256 128 64 32 16 8 4 2 1
ⓢ BIN 1234 0 0 0 0 0 1 0 0 1 1 0 1 0 0 1 0
ⓢ+1 BIN 5678 0 0 0 1 0 1 1 0 0 0 1 0 1 1 1 0
ⓢ+2 BIN 1545 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1
ⓢ+(n−2) BIN 4321 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 1
ⓢ+(n−1) BIN 5555 0 0 0 1 0 1 0 1 1 0 1 1 0 0 1 1
n

Operation error

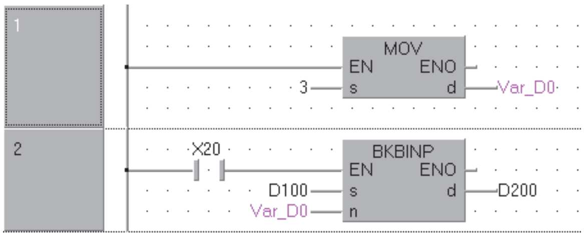
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The data n points from the device specified for (s) is outside the range of 0 to 9999.	○	○	○	○	○	○
4101	The range of n points from the device specified for (s), (d) exceeds the corresponding device. The device ranges of (s) and (d) overlap.	○	○	○	○	○	○

Program example

- In the following program, the BCD values in D100 to D102 are converted to the BIN values when X20 turns ON, and the results are stored to D200 and the following devices.

[Structured ladder/FBD]



[ST]

```
Var_D0:=3;
BKBINP(X20,D100,Var_D0,D200);
```

[Operation]

	8000	4000	2000	1000	800	400	200	100	80	40	20	10	8	4	2	1
D100 BCD 8080	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
D101 BCD 7654	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
D102 BCD 9999	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1

BIN conversion
(when Var_D0=3)

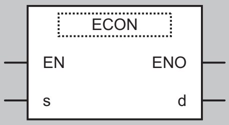
			8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
D200 BIN 8080	0	0	0	1	1	1	1	1	1	0	0	1	0	0	0	0
D201 BIN 7654	0	0	0	1	1	1	0	1	1	1	1	0	0	1	1	0
D202 BIN 9999	0	0	1	0	0	1	1	1	0	0	0	0	1	1	1	1

Single-precision to double-precision conversion

ECON(P)

☒Basic ☒High performance ☒Process ☒Redundant Universal LCPU

Structured ladder/FBD



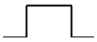
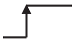
ST

ENO:= ECON (EN, s, d);

Any of the following instruction can go in the dotted squares.

ECON, ECONP

■Executing condition

Instruction	Executing condition
ECON	
ECONP	

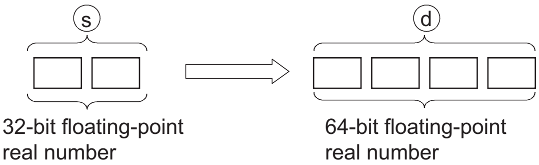
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Conversion source data, or start number of the device that stores conversion source data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted data	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Converts 32-bit floating-point real number specified for (s) to 64-bit floating-point real number, and stores the conversion result to the device specified for (d).


s
d
32-bit floating-point real number
64-bit floating-point real number

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

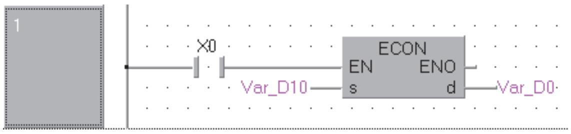
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-126} \leq$ | value of the specified device | $< 2^{128}$ The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○

Program example

- In the following program, the 32-bit floating-point real number in Var_D10 is converted to 64-bit floating-point real number when X0 turns ON, and the result is output to Var_D0.

[Structured ladder/FBD]



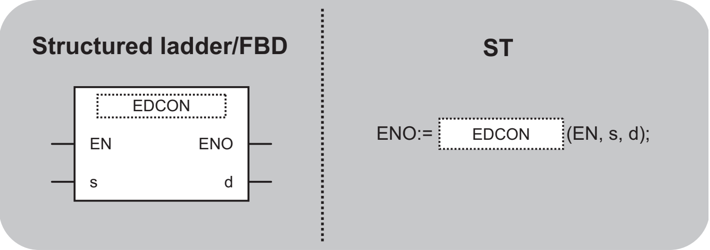
[ST]

```
ECON(X0,Var_D10,Var_D0);
```

Double-precision to single-precision conversion

EDCON(P)

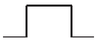
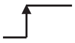
✗Basic ✗High performance ✗Process ✗Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

EDCON, EDCONP

■Executing condition

Instruction	Executing condition
EDCON	
EDCONP	

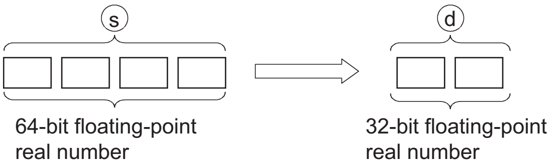
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Conversion source data, or start number of the device that stores conversion source data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted data	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—			—	○	—
(d)	—	○		—			○	—	—

Processing details

- Converts 64-bit floating-point real number specified for (s) to 32-bit floating-point real number, and stores the conversion result to the device specified for (d).


s
d
64-bit floating-point real number
32-bit floating-point real number

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

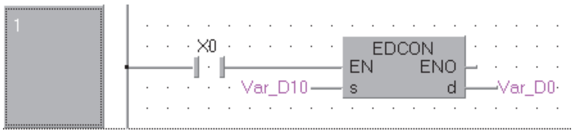
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | $< 2^{1024}$ The value of the specified device is -0.	—	—	—	—	○	○
4141	The conversion result exceeds the following range (An overflow occurs.) $2^{128} \leq$ | conversion result |	—	—	—	—	○	○

Program example

- In the following program, the 64-bit floating-point real number in Var_D10 is converted to 32-bit floating-point real number when X0 turns ON, and the result is output to Var_D0.

[Structured ladder/FBD]



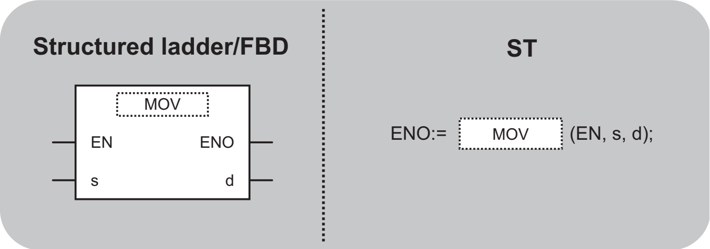
[ST]
```
EDCON(X0,Var_D10,Var_D0);
```

6.4 Data Transfer Instructions

16-/32-bit data transfer

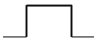
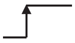
MOV(P), DMOV(P)

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.
MOV, MOVP, DMOV, DMOVP

■Executing condition

Instruction	Executing condition
MOV DMOV	
MOVP DMOVP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data to be transferred, or start number of the device that stores data to be transferred	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Transfer destination of the device number data	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○							○	—
(d)	○							—	—

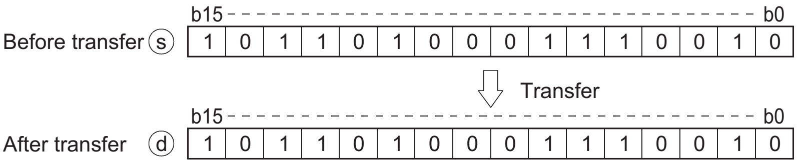
Point

When BL, S, TR, BL\S, or BL\TR is used, refer to SFC control instructions of the MELSEC-Q/L/QnA Programming Manual (SFC).

Processing details

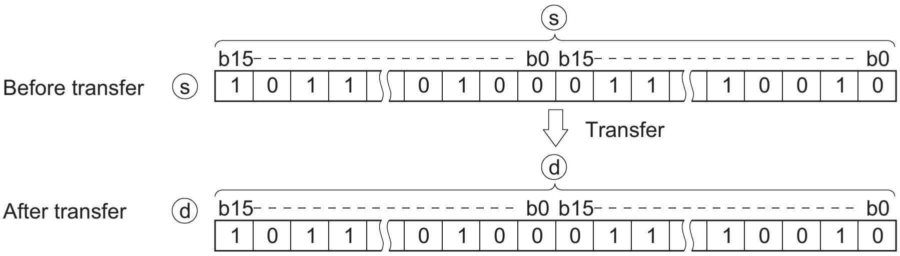
■MOV(P)

- Transfers the 16-bit data in the device specified for (s) to the device specified for (d).



■DMOV(P)

- Transfers the 32-bit data in the device specified for (s) to the device specified for (d).



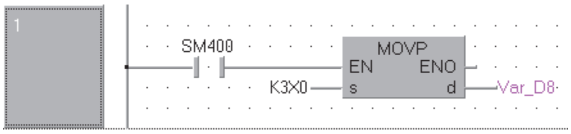
Operation error

- There is no operation error.

Program example

- In the following program, the input data in the devices from X0 to XB are transferred to Var_D8.

[Structured ladder/FBD]

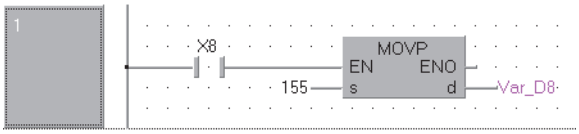


[ST]

MOVP(SM400,K3X0,Var_D8);

- In the following program, 155 is transferred to Var_D8 when X8 turns ON.

[Structured ladder/FBD]

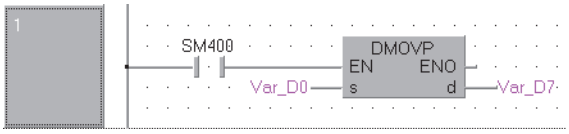


[ST]

MOVP(X8,155,Var_D8);

- In the following program, the data in Var_D0 are transferred to Var_D7.

[Structured ladder/FBD]

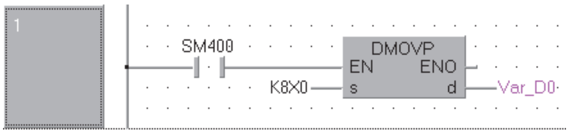


[ST]

DMOVP(SM400,Var_D0,Var_D7);

- In the following program, the data in the devices from X0 to X1F are transferred to Var_D0.

[Structured ladder/FBD]



[ST]

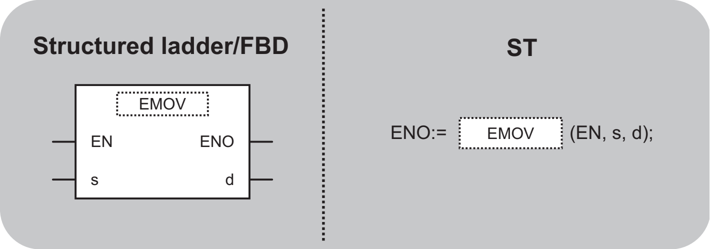
DMOVP(SM400,K8X0,Var_D0);

Floating-point data transfer (single precision)

EMOV(P)

Ver. Basic High performance Process Redundant Universal LCPU

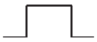
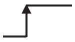
- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later



Any of the following instruction can go in the dotted squares.

EMOV, EMOVP

■Executing condition

Instruction	Executing condition
EMOV	
EMOVP	

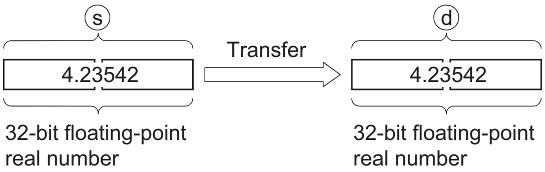
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data to be transferred, or start number of the device that stores data to be transferred	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores transfer destination of data	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Transfers the 32-bit floating-point real number data in the device specified for (s) to the device specified for (d).


s
d
Transfer
4.23542
4.23542
32-bit floating-point real number
32-bit floating-point real number

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

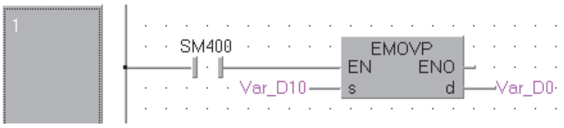
Operation error

- There is no operation error.

Program example

- In the following program, the real number in Var_D10 is transferred to Var_D0.

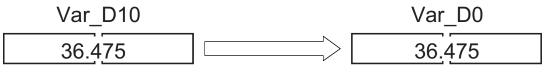
[Structured ladder/FBD]



[ST]

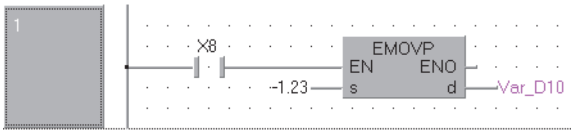
EMOVP(SM400,Var_D10,Var_D0);

[Operation]



- In the following program, the real number -1.23 is transferred to Var_D10 when X8 turns ON.

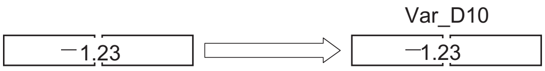
[Structured ladder/FBD]



[ST]

EMOVP(X8,-1.23,Var_D10);

[Operation]

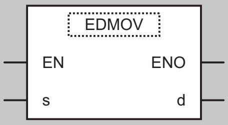


Floating-point data transfer (double precision)

EDMOV(P)

✕Basic ✕High performance ✕Process ✕Redundant Universal LCPU

Structured ladder/FBD



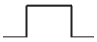
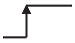
ST

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

Any of the following instruction can go in the dotted squares.

EDMOV, EDMOVP

■Executing condition

Instruction	Executing condition
EDMOV	
EDMOVP	

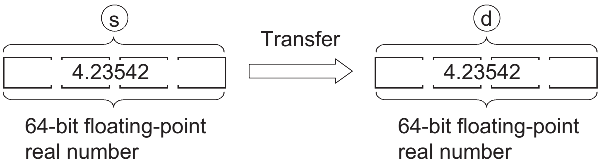
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data to be transferred, or start number of the device that stores data to be transferred	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores transfer destination of data	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Transfers the 64-bit floating-point real number specified for (s) to the device specified for (d).


s
d
Transfer
4.23542
4.23542
64-bit floating-point real number
64-bit floating-point real number

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

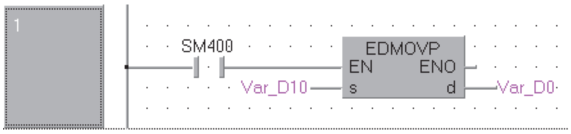
Operation error

- There is no operation error.

Program example

- In the following program, the 64-bit floating-point real number in Var_D10 is transferred to Var_D0.

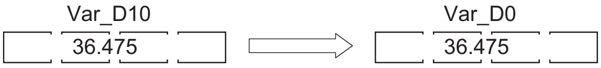
[Structured ladder/FBD]



[ST]

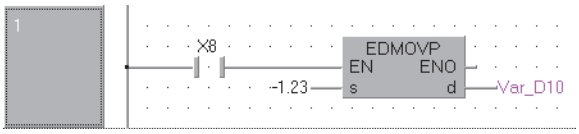
EDMOVP(SM400,Var_D10,Var_D0);

[Operation]



- In the following program, the real number -1.23 is transferred to Var_D10 when X8 turns ON.

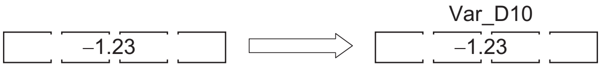
[Structured ladder/FBD]



[ST]

EDMOVP(X8,-1.23,Var_D10);

[Operation]

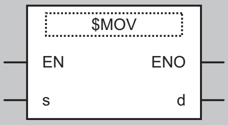


Character string data transfer

$MOV(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



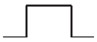
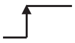
ST

Not supported

Any of the following instruction can go in the dotted squares.

$MOV, $MOVP

■Executing condition

Instruction	Executing condition
$MOV	
$MOVP	

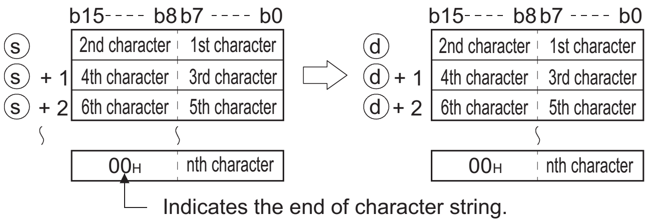
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Character string data to be transferred (Maximum string length: 32 characters), or start number of the device that stores character string data to be transferred	String
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores character string data to be transferred	String

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

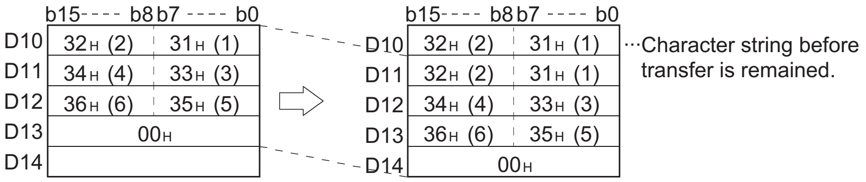
Processing details

- Transfers the character string data specified for (s) to the device specified for (d) and the following devices. Transfers the character string data with double quotation (") specified for or the character string data from the specified device number to the device number that stores 00H at once.

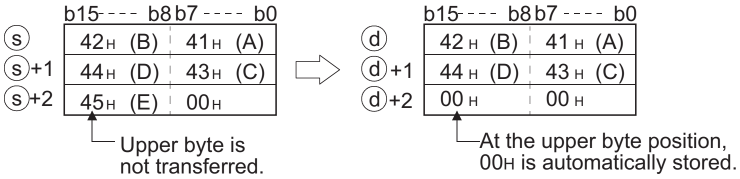


b15 ---- b8 b7 ---- b0
s | 2nd character | 1st character
s + 1 | 4th character | 3rd character
s + 2 | 6th character | 5th character
00H | nth character
d | 2nd character | 1st character
d + 1 | 4th character | 3rd character
d + 2 | 6th character | 5th character
00H | nth character
Indicates the end of character string.

- Processing will be performed without error even in cases where the range for the devices storing the character string data to be transferred ((s) to (s)+n) overlaps with the range of the devices which will store the character string data after it has been transferred ((d) to (d)+n). The following occurs when the character string data that had been stored in D10 to D13 is transferred to D11 to D14:



- If the 00H code is being stored to lower bytes of (s)+n, 00H will be stored to both the upper bytes and the lower bytes of (d)+n.



Operation error

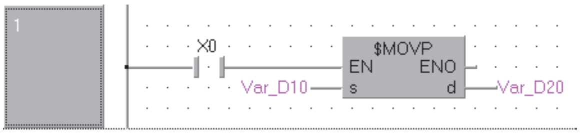
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	There is no 00H code stored between the device number specified for (s) and the corresponding device. The entire character string linked from the device number specified for (d) to the final device number of the corresponding device cannot be stored. The character string of (s) exceeds 16383 characters.	○	○	○	○	○	○

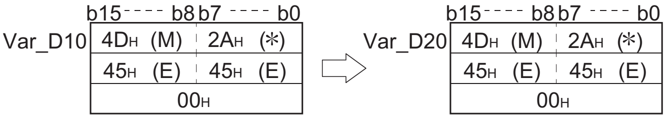
Program example

- In the following program, the character string data in Var_D10 are transferred to Var_D20 when X0 turns ON.

[Structured ladder/FBD]



[Operation]

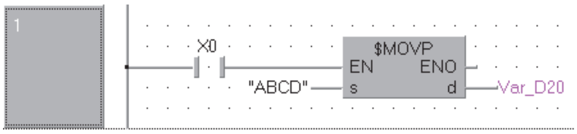


Var_D10	b15 ---- b8	b7 ---- b0
	$4D_H$ (M)	$2A_H$ (*)
	45_H (E)	45_H (E)
	00_H	

Var_D20	b15 ---- b8	b7 ---- b0
	$4D_H$ (M)	$2A_H$ (*)
	45_H (E)	45_H (E)
	00_H	

- In the following program, the character string "ABCD" is transferred to Var_D20 when X0 turns ON.

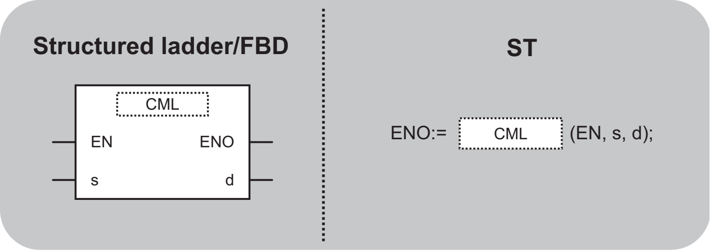
[Structured ladder/FBD]



16-/32-bit data negation transfer

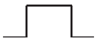
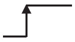
CML(P), DCML(P)

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.
CML, CMLP, DCML, DCMLP

■Executing condition

Instruction	Executing condition
CML DCML	
CMLP DCMLP	

■Argument

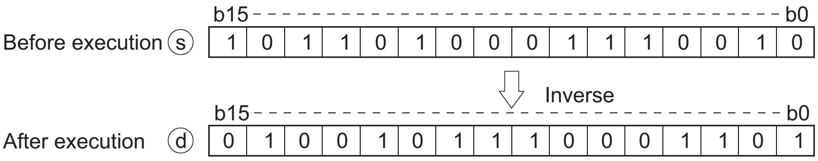
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data to be inverted, or start number of the device that stores data to be inverted	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores inversion result	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○							○	—
(d)	○							—	—

Processing details

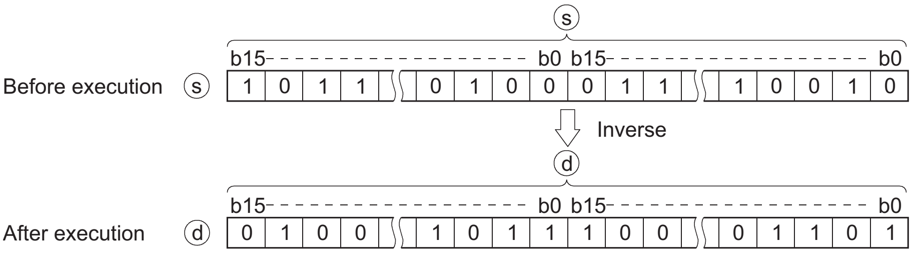
■CML(P)

- Inverts 16-bit data specified for (s) bit by bit, and transfers the result to the device specified for (d).



■DCML(P)

- Inverts 32-bit data specified for (s) bit by bit, and transfers the result to the device specified for (d).



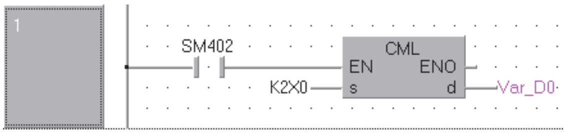
Operation error

- There is no operation error.

Program example

- In the following program, the data in the devices from X0 to X7 are inverted, and the results are transferred to Var_D0.

[Structured ladder/FBD]

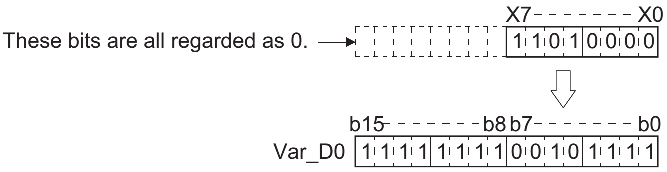


[ST]

CML(SM402,K2X0,Var_D0);

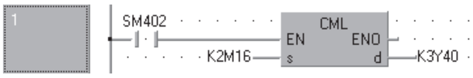
[Operation]

If the number of bits of ⓢ is smaller than the number of bits of ⓓ


These bits are all regarded as 0.
X7 - - - - - - - X0
1 1 0 1 0 0 0 0
b15 - - - - - - b8 b7 - - - - - - - b0
Var_D0
1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1

- In the following program, the data in the devices from M16 to M23 are inverted, and the results are transferred to the devices from Y40 to Y47.

[Structured ladder/FBD]

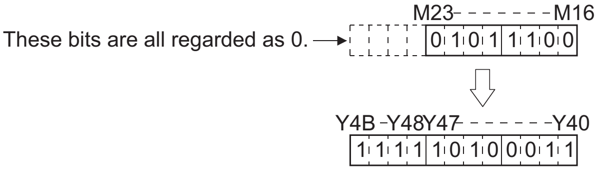


[ST]

CML(SM402,K2M16,K3Y40);

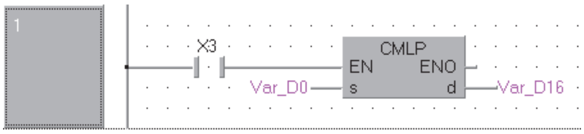
[Operation]

If the number of bits of ⓢ is smaller than the number of bits of ⓓ


These bits are all regarded as 0.
M23 - - - - - - M16
0 1 0 1 1 1 0 0
Y4B - Y48 Y47 - - - - - - Y40
1 1 1 1 1 0 1 0 0 0 1 1

- In the following program, the data in Var_D0 are inverted when X3 turns ON, and the result is transferred to Var_D16.

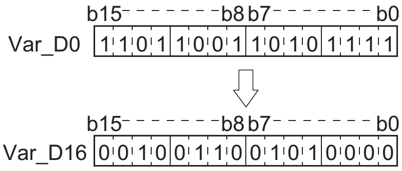
[Structured ladder/FBD]



[ST]

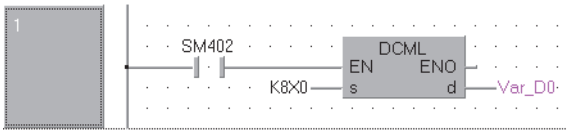
CMLP(X3,Var_D0,Var_D16);

[Operation]


b15 - - - - - - b8 b7 - - - - - - - b0
Var_D0
1 1 0 1 1 0 0 1 1 0 1 0 1 1 1 1
b15 - - - - - - b8 b7 - - - - - - - b0
Var_D16
0 0 1 0 0 1 1 0 0 1 0 1 0 0 0 0

- In the following program, the data in the devices from X0 to X1F are inverted, and the results are transferred to Var_D0.

[Structured ladder/FBD]

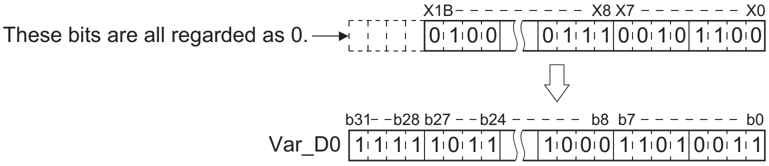


[ST]

DCML(SM402,K8X0,Var_D0);

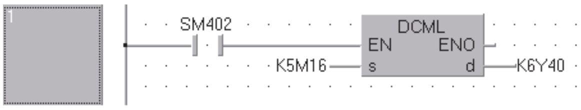
[Operation]

If the number of bits of ⓢ is smaller than the number of bits of ⓓ



- In the following program, the data in the devices from M16 to M35 are inverted, and the results are transferred to the devices from Y40 to Y63.

[Structured ladder/FBD]

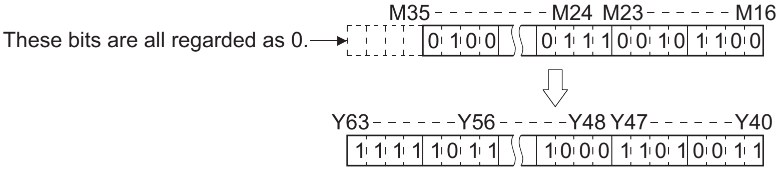


[ST]

DCML(SM402,K5M16,K6Y40);

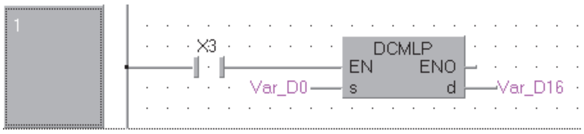
[Operation]

If the number of bits of ⓢ is smaller than the number of bits of ⓓ



- In the following program, the data in Var_D0 are inverted when X3 turns ON, and the result is transferred to Var_D16.

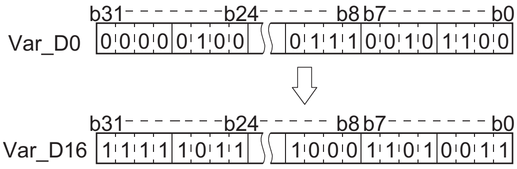
[Structured ladder/FBD]



[ST]

DCMLP(X3,Var_D0,Var_D16);

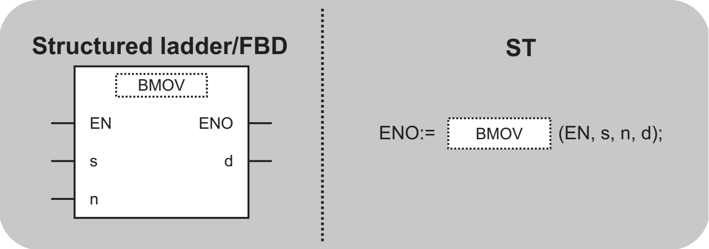
[Operation]



16-bit block data transfer

BMOV(P)

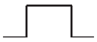
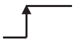
Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

BMOV, BMOVP

■Executing condition

Instruction	Executing condition
BMOV	
BMOVP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores data to be transferred	ANY_SIMPLE
	n	Number of transfers	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device at the transfer destination	ANY_SIMPLE

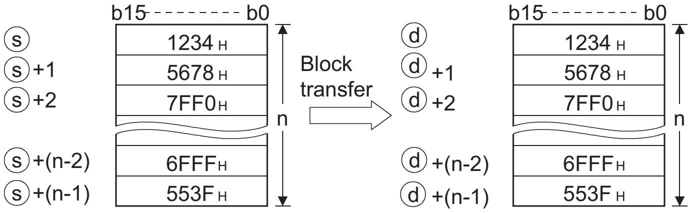
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○						—		—
n	○						○		—
(d)	○						—		—

Point

When BL, S, TR, BL\S, or BL\TR is used, refer to SFC control instructions of the MELSEC-Q/L/QnA Programming Manual (SFC).

Processing details

- Batch transfers the 16-bit data n points from the device specified for (s) to the n points of devices from the one specified for (d).



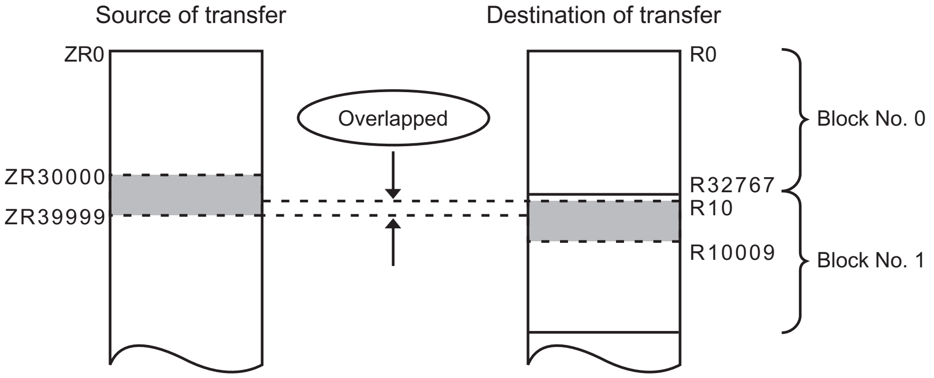
- Transfers can be accomplished even in cases where there is an overlap between the source and destination device. In the case of transmission to the smaller device number, transmission is from (s); for transmission to the larger device number, transmission is from (s) + (n-1). However, as shown in the example below, when transferring data from R to ZR, or from ZR to R, the range to be transferred and the range of destination must not overlap. Transfer from R to R, or from ZR to ZR can be performed without any problem.
  - ZR transfer range ((specified start number of ZR) to (specified start number of ZR + the number of transfers -1))
  - R transfer range ((specified start number of R + file register block number × 32768) to (specified start number of R + file register block number × 32768 + the number of transfers -1))

Ex.

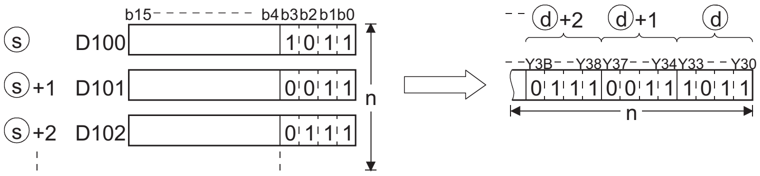
Transfer ranges of ZR and R overlap when transferring 10000 blocks of data from ZR30000 (source) to R10 (block No.1 of the destination).

- ZR transfer range → (30000) to (30000 + 10000 - 1) → (30000)~(39999)
- R transfer range → (10 + (1 × 32768)) to (10 + (1 × 32768) + 10000 - 1)
  → (32778) to (42777)

Therefore, the range 32778 to 39999 overlaps and the data are not correctly transferred.



- When (s) is a word device and (d) is a bit device, the target for the word device will be the number of bits specified for digit-specified bit device. If K1Y30 has been specified for (d), the lower four bits of the word device specified for (s) will become the object.



- If bit device has been specified for (s) and (d), then (s) and (d) should always have the same number of digits.
- When using a link direct device and an intelligent function module device for (s) and (d), only either of (s) or (d) can be used.
- Whether to disable or enable the device range check at the execution of the BMOV instruction can be selected by the device range check disable flag (SM237). (Only when the subset condition is satisfied) The device range check for the devices (s) to (s) + (n-1) and (d) to (d) + (n-1) is not performed when SM237 is ON.

Precautions

Do not perform the following accesses when SM 237 is ON.

- An access in which the index setting exceeds the device range.
- An access in which devices from (d) to (d) + (n-1) cross over the boundary of the device range.[*1]
- An access to file registers without setting file registers.
- An access to the area which does not contain multiple CPU high speed transmission area devices. (For QCPU (Q mode) only)

[*1] Refer to the DFMOV instruction.

Point

SM237 can be used for the following CPU modules only.

- Universal model QCPU with a serial number whose first five digits are '10012' or higher
- LCPU

Operation error

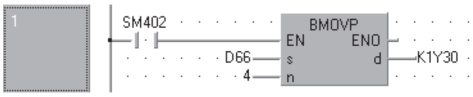
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device range of n points from (s) or (d) exceeds the corresponding device range. The link direct device and the unit access device are specified in both (s) and (d).	○	○	○	○	○	○

Program example

- In the following program, the lower 4-bit data in the devices from D66 to D69 are transferred to the devices from Y30 to Y3F in units of 4 points.

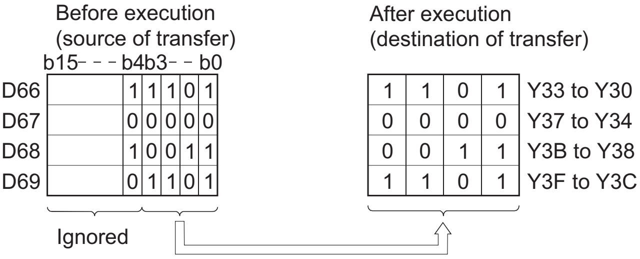
[Structured ladder/FBD]



[ST]

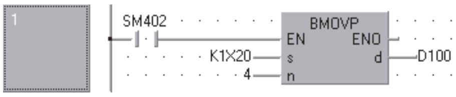
BMOVP(SM402,D66,4,K1Y30);

[Operation]



- In the following program, the data in the devices from X20 to X2F are transferred to the devices from D100 to D103 in units of 4 points.

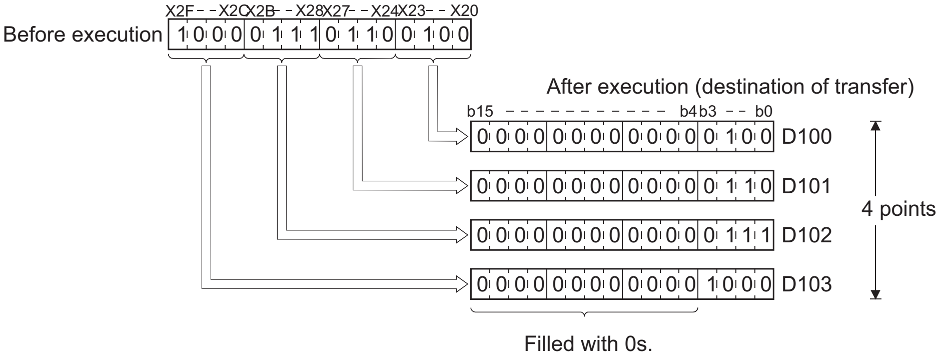
[Structured ladder/FBD]



[ST]

BMOVP(SM402,K1X20,4,D100);

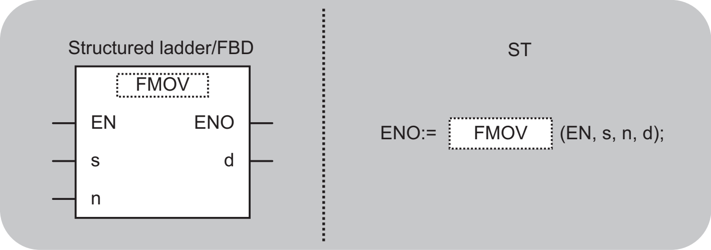
[Operation]


X2F - - X2C X2B - - X28 X27 - - X24 X23 - - X20
Before execution
1 0 0 0 0 1 1 1 0 1 1 0 0 1 0 0
After execution (destination of transfer)
b15 - - - - - - - - - - - b4 b3 - - b0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 D100
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 D101
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 D102
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 D103
4 points
Filled with 0s.

Identical 16-bit block data transfer

FMOV(P)

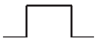
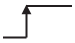
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

FMOV, FMOVP

■Executing condition

Instruction	Executing condition
FMOV	
FMOVP	

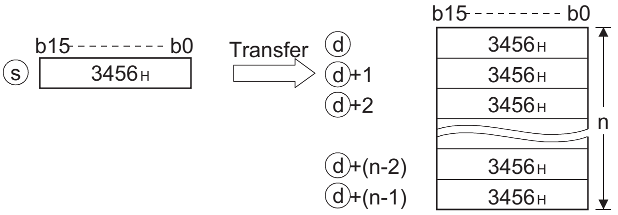
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data to be transferred, or start number of the device that stores data to be transferred	ANY16
	n	Number of transfers	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device at the transfer destination	ANY

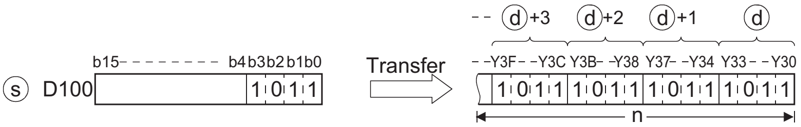
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○						○		—
n	○						○		—
(d)	○						—		—

Processing details

- Transfers the 16-bit data from the device specified for (s) to the n points of devices from the one specified for (d).



- When (s) is a word device and (d) is a bit device, the target for the word device specified for (s) will be the number of bits specified for digit-specified bit device. If K1Y30 is specified for (d), the lower 4 bits of the word device specified for (s) becomes a target.


b15 - - - - - - - - b4 b3 b2 b1 b0
s D100
1 0 1 1
Transfer
- - d+3 d+2 d+1 d
- -Y3F- -Y3C Y3B- -Y38 Y37- -Y34 Y33- -Y30
1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
n

- If bit device has been specified for (s) and (d), then (s) and (d) should always have the same number of digits.
- Whether to disable or enable the device range check at the execution of the FMOV instruction can be selected by the device range check disable flag (SM237). (Only when the subset condition is satisfied) The device range check for the devices (d) to (d)+(n-1) is not performed when SM237 is ON.

Precautions

Do not perform the following accesses when SM 237 is ON.

- An access in which the index setting exceeds the device range.
- An access in which devices from (d) to (d) + (n-1) cross over the boundary of the device range.[*1]
- An access to file registers without setting file registers.
- An access to the area which does not contain multiple CPU high speed transmission area devices. (For QCPU (Q mode) only)

[*1] Refer to the DFMOV instruction.

Point

SM237 can be used for the following CPU modules only.

- Universal model QCPU with a serial number whose first five digits are '10012' or higher
- LCPU

Operation error

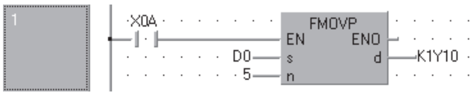
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device range of n points from (d) exceeds the corresponding device range.	○	○	○	○	○	○

Program example

- In the following program, the lower 4-bit data in D0 are transferred to the devices from Y10 to Y23 in units of 4 bits when X0A turns ON.

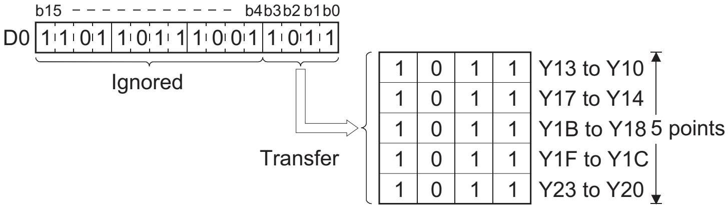
[Structured ladder/FBD]



[ST]

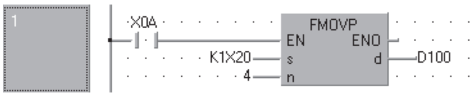
FMOVP(X0A,D0,5,K1Y10);

[Operation]



- In the following program, the data in the devices from X20 to X23 are transferred to the devices from D100 to D103 when X0A turns ON.

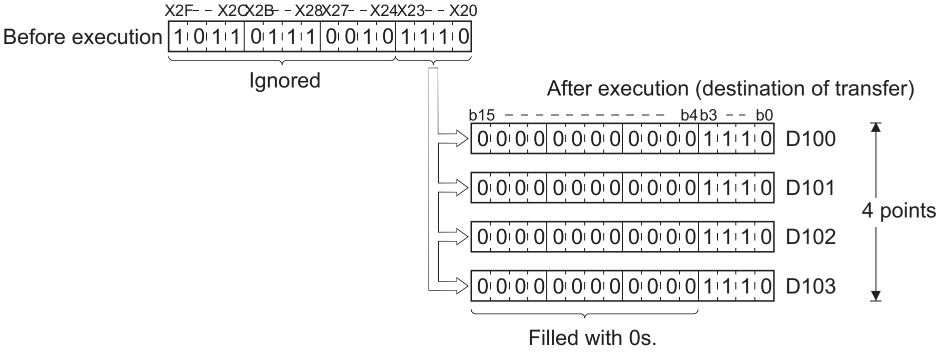
[Structured ladder/FBD]



[ST]

FMOVP(X0A,K1X20,4,D100);

[Operation]

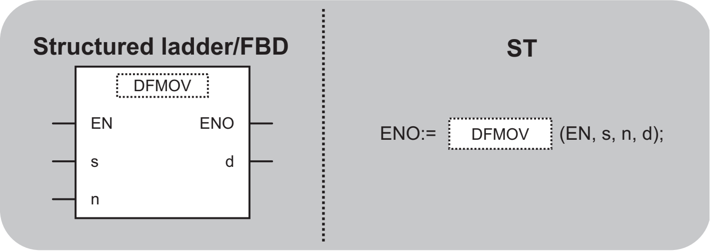

X2F
X2C
X2B
X28
X27
X24
X23
X20
Before execution
1 0 1 1 0 1 1 1 0 0 1 0 1 1 1 0
Ignored
After execution (destination of transfer)
b15
b4
b3
b0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 D100
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 D101
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 D102
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 D103
4 points
Filled with 0s.

Identical 32-bit block data transfer

DFMOV(P)

✗ Basic ✗ High performance ✗ Process ✗ Redundant Ver. Universal LCPU

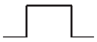
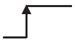
- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported



Any of the following instruction can go in the dotted squares.

DFMOV, DFMOVP

■Executing condition

Instruction	Executing condition
DFMOV	
DFMOVP	

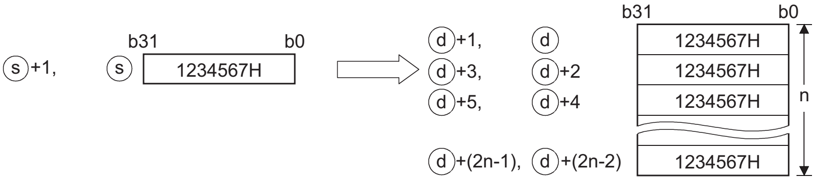
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data to be transferred, or start number of the device that stores data to be transferred	ANY32
	n	Number of transfers	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device at the transfer destination	ANY32

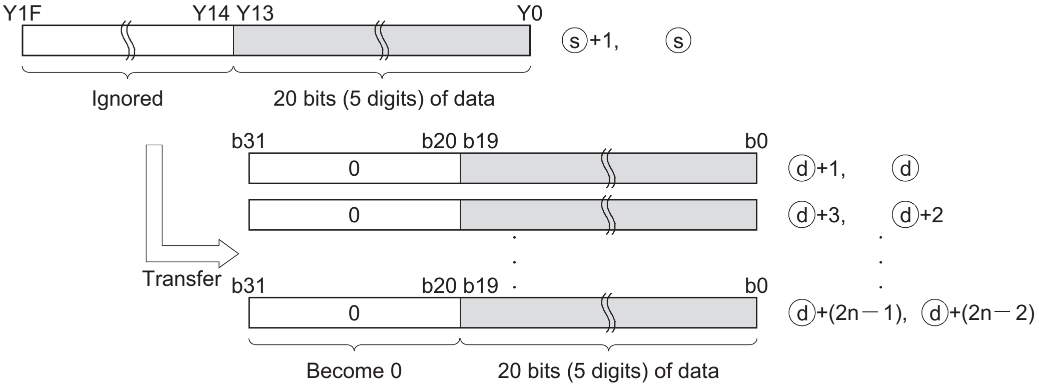
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○						○		—
n	○						○		—
(d)	○						—		—

Processing details

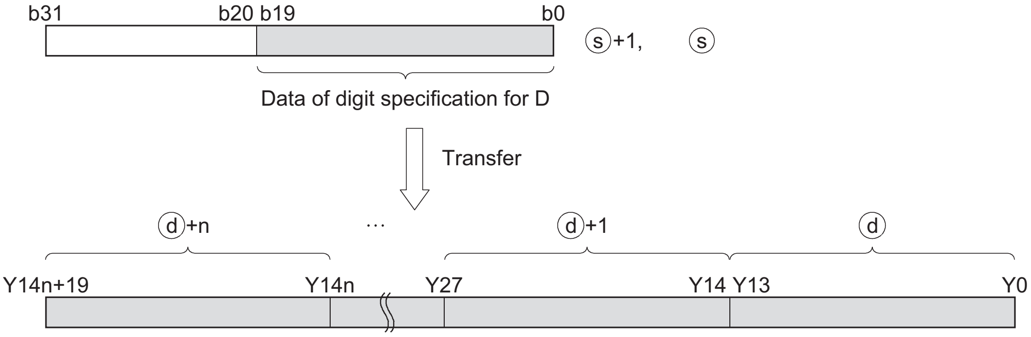
- Transfers the 32-bit data n points from the device specified for (s) to the n points of devices from the one specified for (d).



- When digits are specified for (s), only the data of digit specification are transferred. If K5Y0 is specified for (s), the lower 20 bits (5 digits) of the word device specified for (s) become the targets.



- When digits are specified for (d), the data of digit specification for (d) are transferred. If K5Y0 is specified for (d), the lower 20 bits of the word device specified for (s) become the targets. If digits are specified for both (s) and (d), the data of digit specification for (d) are transferred regardless of the number of specified digits.



- No processing is performed if the value specified for n is 0.
- Whether to disable or enable the device range check at the execution of the DFMOV instruction can be selected by the device range check disable flag (SM237). (Only when the subset condition is satisfied)

Operation error

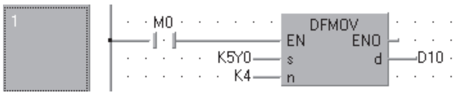
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	A negative value is specified for n.	—	—	—	—	○	○
4101	n points of data to be transferred exceed the device range of (d).	—	—	—	—	○	○

Program example

- In the following program, the data in the devices from Y0 to Y13 (20 bits) are stored to the devices from D10 to D17 when M0 turns ON.

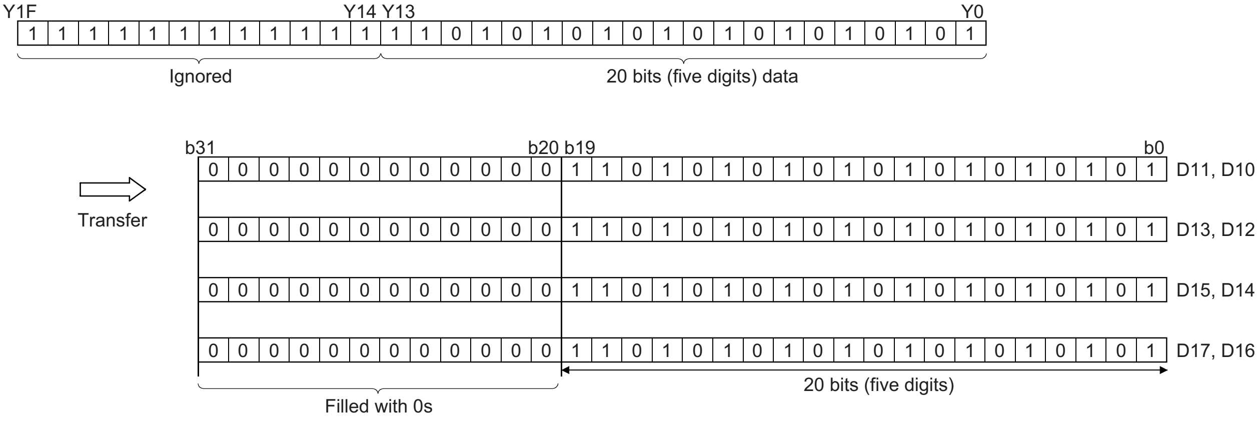
[Structured ladder/FBD]



[ST]

DFMOV(M0,K5Y0,K4,D10);

[Operation]



```
Y1F                                                 Y14 Y13                                                                                   Y0
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
```

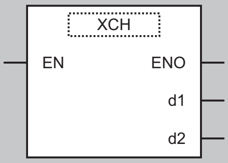
```
b31                     b20 b19                                   b0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1  D11, D10
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1  D13, D12
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1  D15, D14
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1  D17, D16
```

16-/32-bit data exchange

XCH(P), DXCH(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



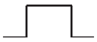
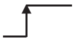
ST

ENO:= XCH (EN, d1, d2);

Any of the following instruction can go in the dotted squares.

XCH, XCHP, DXCH, DXCHP

■Executing condition

Instruction	Executing condition
XCH DXCH	
XCHP DXCHP	

■Argument

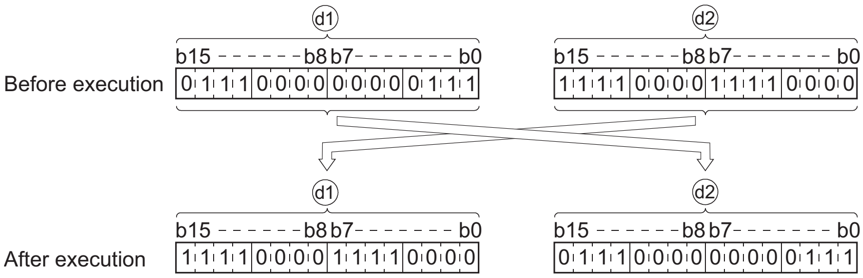
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d1, d2	Start number of the device that stores data to be exchanged	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(d1)	○							—	
(d2)	○							—	

Processing details

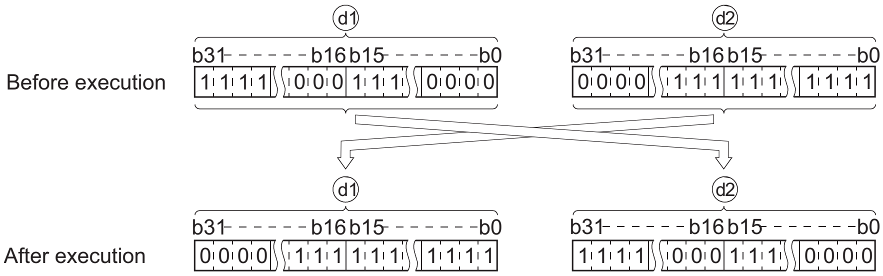
■XCH(P)

- Exchanges the 16-bit data between (d1) and (d2).



■DXCH(P)

- Exchanges the 32-bit data between (d1) and (d2).


d1
d2
Before execution
b31 - - - - - - b16 b15 - - - - - - b0
1 1 1 1 0 0 0 1 1 1 0 0 0 0
b31 - - - - - - b16 b15 - - - - - - b0
0 0 0 0 1 1 1 1 1 1 1 1 1 1
d1
d2
After execution
b31 - - - - - - b16 b15 - - - - - - b0
0 0 0 0 1 1 1 1 1 1 1 1 1 1
b31 - - - - - - b16 b15 - - - - - - b0
1 1 1 1 0 0 0 1 1 1 0 0 0 0

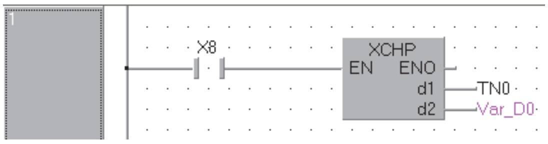
Operation error

- There is no operation error.

Program example

- In the following program, the current value of T0 and the value of Var_D0 are exchanged when X8 turns ON.

[Structured ladder/FBD]

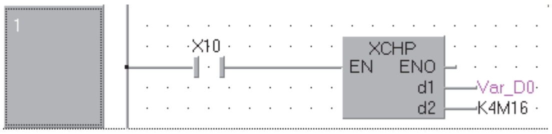


[ST]

XCHP(X8,TN0,Var_D0);

- In the following program, the data in Var_D0 and the data in the devices from M16 to M31 are exchanged when X10 turns ON.

[Structured ladder/FBD]

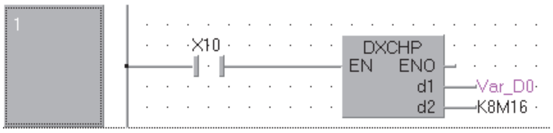


[ST]

XCHP(X10,Var_D0,K4M16);

- In the following program, the data in Var_D0 and the data in the devices from M16 to M47 are exchanged when X10 turns ON.

[Structured ladder/FBD]

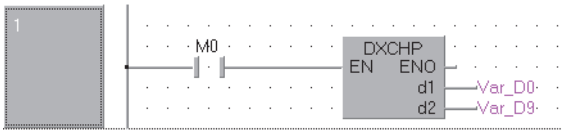


[ST]

DXCHP(X10,Var_D0,K8M16);

- In the following program, the data in Var_D0 and the data in Var_D9 are exchanged when M0 turns ON.

[Structured ladder/FBD]



[ST]

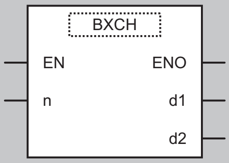
DXCHP(M0,Var_D0,Var_D9);

16-bit block data exchange

BXCH(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



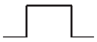
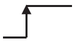
ST

ENO:= BXCH (EN, n, d1, d2);

Any of the following instruction can go in the dotted squares.

BXCH, BXCHP

■Executing condition

Instruction	Executing condition
BXCH	
BXCHP	

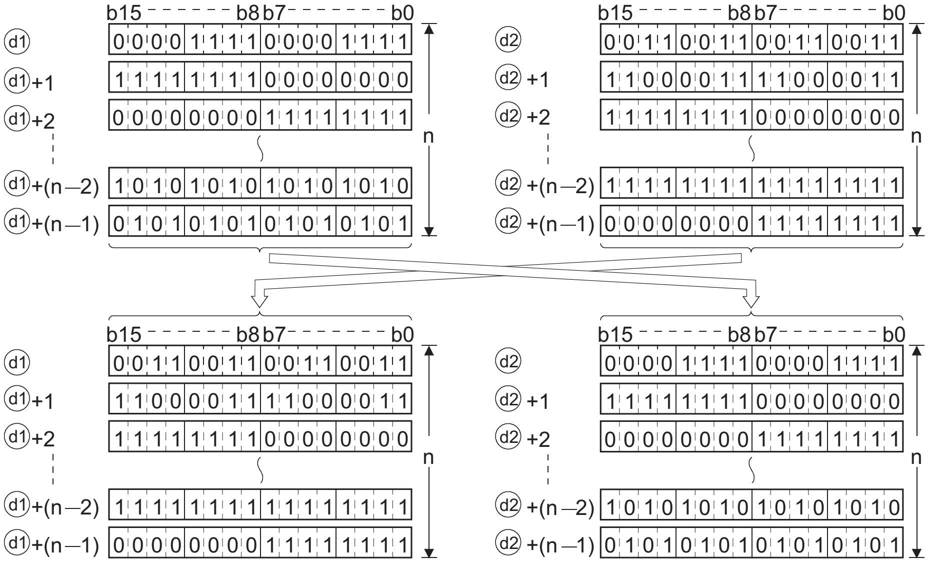
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Number of exchanges	ANY16
Output argument	ENO	Execution result	Bit
	d1, d2	Start number of the device that stores data to be exchanged	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○	○		○					—
(d1)	—	○		—					—
(d2)	—	○		—					—

Processing details

- Exchanges the 16-bit data n points from the device specified for (d1) and the 16-bit data n points from the device specified for (d2).



Operation error

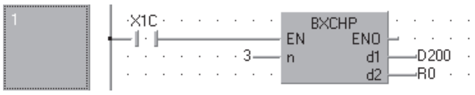
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device range of n points from the device specified for (d1), (d2) exceeds the corresponding device. The (d1) and (d2) devices overlap.	○	○	○	○	○	○

Program example

- In the following program, the data 3 points from D200 and the data 3 points from R0 are exchanged when X1C turns ON.

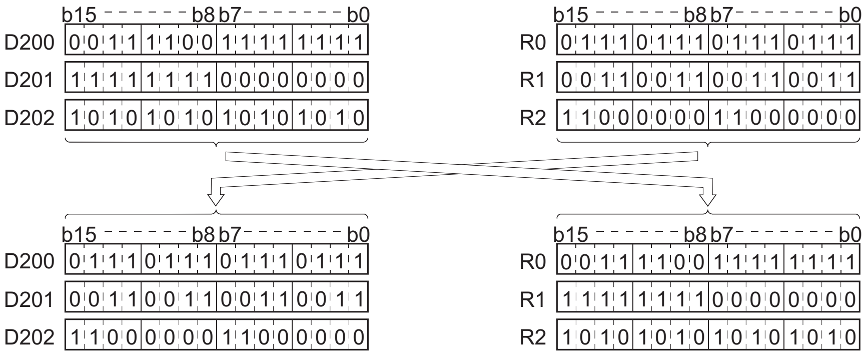
[Structured ladder/FBD]



[ST]

BXCHP(X1C,3,D200,R0);

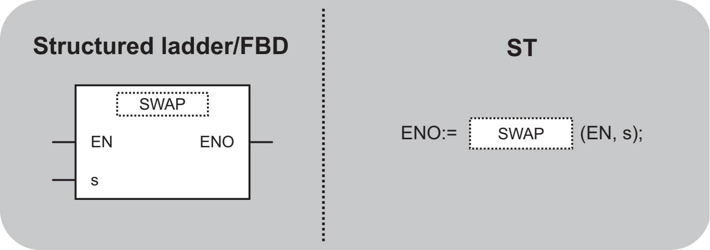
[Operation]



Upper and lower bytes exchange

SWAP(P)

Basic High performance Process Redundant Universal LCPU

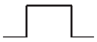
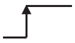


```
ENO:= SWAP (EN, s);
```

Any of the following instruction can go in the dotted squares.

SWAP, SWAPP

■Executing condition

Instruction	Executing condition
SWAP	
SWAPP	

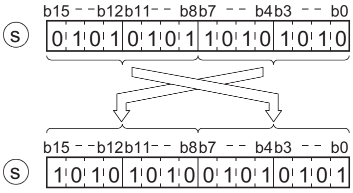
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores data to be exchanged	ANY16
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s)	○							—	

Processing details

- Exchanges the higher 8 bits and lower 8 bits of the device specified for (s).


b15 - - b12 b11 - - b8 b7 - - b4 b3 - - b0
s
0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0
b15 - - b12 b11 - - b8 b7 - - b4 b3 - - b0
s
1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1

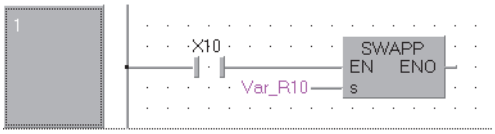
Operation error

- There is no operation error.

Program example

- In the following program, the higher 8 bits and the lower 8 bits in Var_R10 are exchanged when X10 turns ON.

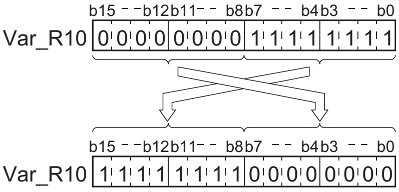
[Structured ladder/FBD]



[ST]

SWAPP(X10,Var_R10);

[Operation]


b15 - - b12 b11 - - b8 b7 - - b4 b3 - - b0
Var_R10
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
b15 - - b12 b11 - - b8 b7 - - b4 b3 - - b0
Var_R10
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

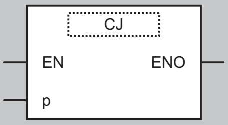
6.5 Program Branch Instructions

Pointer branch

CJ, SCJ, JMP

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



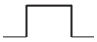
ST

Not supported

Any of the following instruction can go in the dotted squares.

CJ, SCJ, JMP

■Executing condition

Instruction	Executing condition
CJ, SCJ	
JMP	Non-conditional execution

■Argument

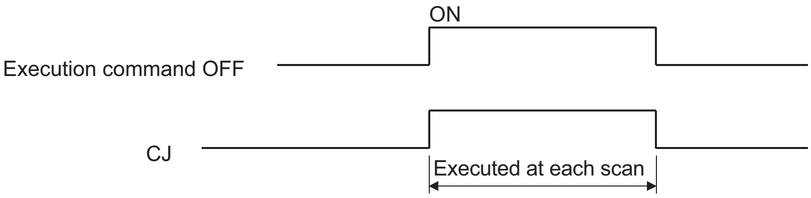
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	p	Pointer number or ladder block label of the jump destination	ANY16
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				P
p	—								○

Processing details

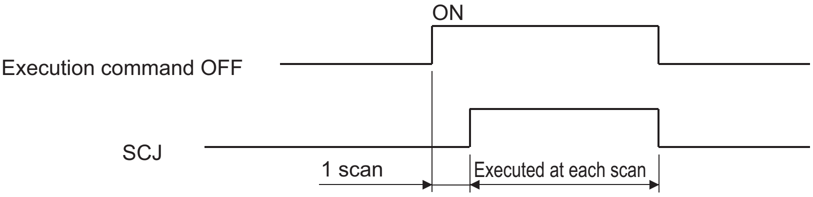
■CJ

- Executes the program of the specified pointer number or ladder block label when the execution command is ON.
- When the execution command is OFF, the program at the next step is executed.



■SCJ

- Executes the program of the specified pointer number or ladder block label from the next scan after the execution command turns from OFF to ON.
- Executes the program in the next step when the execution command is OFF or turns from ON to OFF.



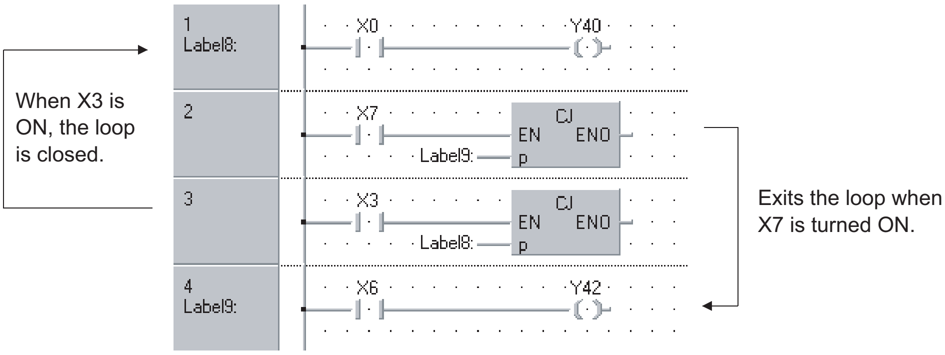
■JMP

- Unconditionally executes the program of the specified pointer number or ladder block label.

Point

Note the following points when using the CJ, SCJ, and JMP instructions.

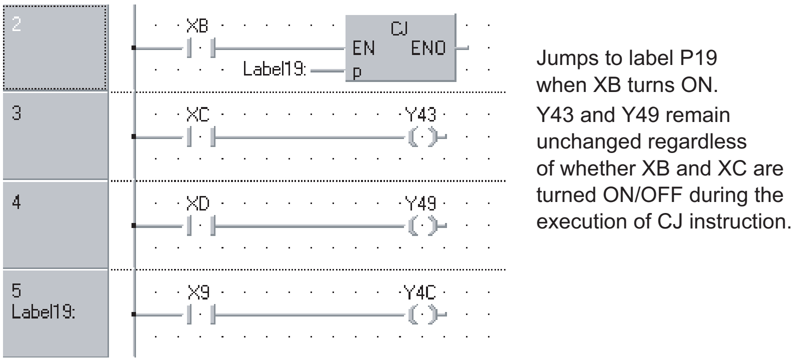
- After the timer coil has turned ON, accurate measurements cannot be made if there is an attempt to jump the timer of a coil that has been turned ON using the CJ, SCJ or JMP instruction.
- Scan time is shortened if the CJ, SCJ or JMP instruction is used to force a jump to the OUT instruction.
- Scan time is shortened if the CJ, SCJ or JMP instruction is used to force a jump to the rear.
- The CJ, SCJ, and JMP instructions can be used to jump to a step prior to the step currently being executed. However, it is necessary to consider methods to get out of the loop so that the watchdog timer does not time out in the process.



When X3 is ON, the loop is closed.

Exits the loop when X7 is turned ON.

- The device to which a jump has been made with the CJ, SCJ or JMP instruction does not change.



Jumps to label P19 when XB turns ON.

Y43 and Y49 remain unchanged regardless of whether XB and XC are turned ON/OFF during the execution of CJ instruction.

- The jumping ranges of CJ, SCJ, and JMP instructions are different according to the pointer type specified for p. The pointer number can jump to pointer numbers within the same program file. The ladder block label can jump to the ladder block labels within the same POU.

Operation error

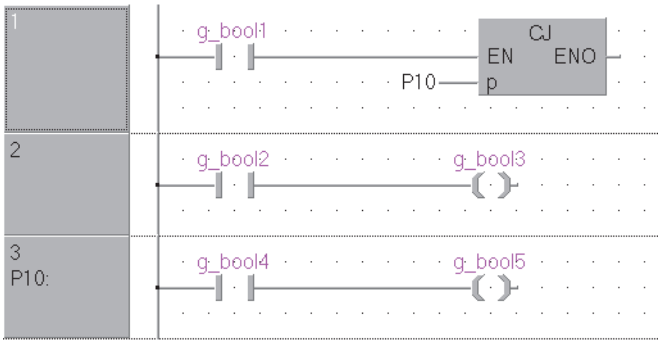
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4210	The pointer number which is not used as a label in the same program is specified. The common pointer number in another program is specified	○	○	○	○	○	○

Program example

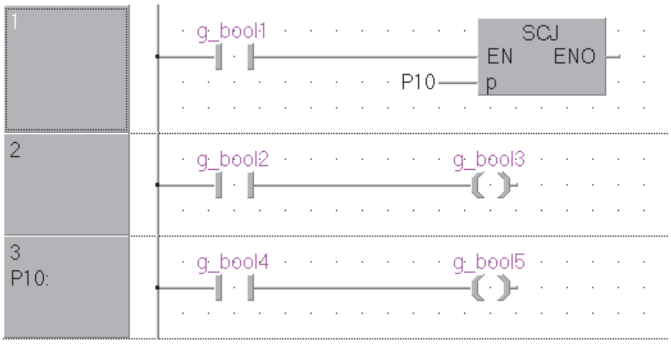
- The following program jumps to P10 when g_bool1 turns ON.

[Structured ladder/FBD]



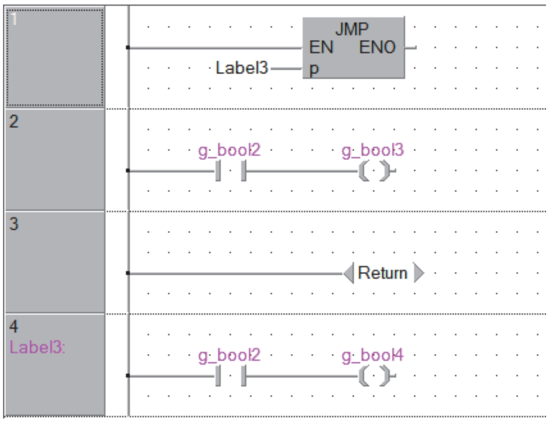
- The following program jumps to P10 from the next scan after g_bool1 is turned ON.

[Structured ladder/FBD]



- The following program jumps to Label3.

[Structured ladder/FBD]



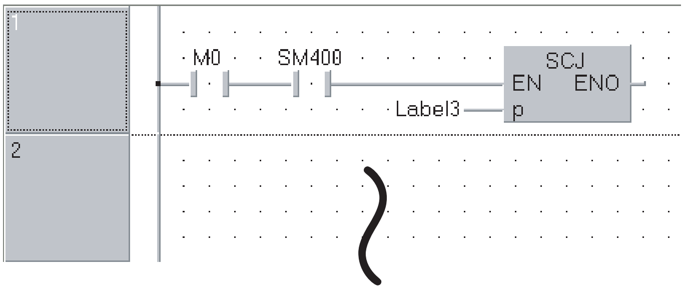
Precautions

- For Universal model QCPU and LCPU, AND SM400 needs to be inserted directly prior to the SCJ instruction.

Ex.

[Program example 1]

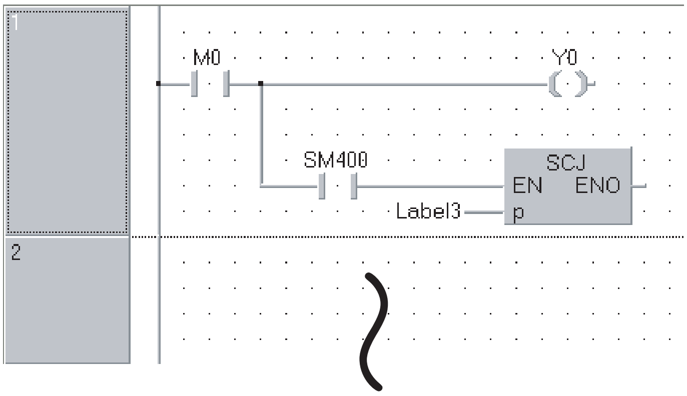
[Structured ladder/FBD]



Ex.

[Program example 2]

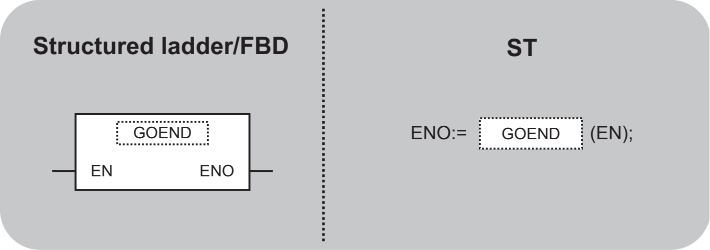
[Structured ladder/FBD]



Jump to END processing

GOEND

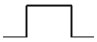
Basic | High performance | Process | Redundant | Universal | LCPU



The following instruction can go in the dotted squares.

GOEND

■Executing condition

Instruction	Executing condition
GOEND	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

- Jumps to the FEND instruction or END processing in the same program file.

Operation error

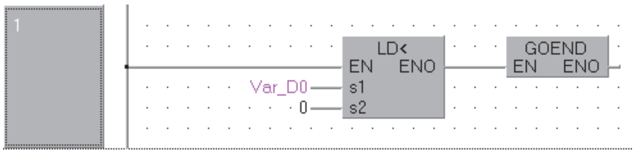
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4200	The GOEND instruction has been executed after the execution of the FOR instruction, and prior to the execution of the NEXT instruction.	○	○	○	○	○	○
4211	The GOEND instruction has been executed after the execution of the CALL instruction, and prior to the execution of the RET instruction.	○	○	○	○	○	○
4221	The GOEND instruction has been executed during an interrupt program, and prior to the execution of the IRET instruction.	○	○	○	○	○	○
4230	The GOEND instruction was executed within the CHKCIR to CHKEND instruction loop.	○	○	○	○	○	○

Program example

- In the following program, execution of the program file is terminated if Var_D0 holds a negative number.

[Structured ladder/FBD]



[ST]

```
GOEND(Var_D0<0);
```

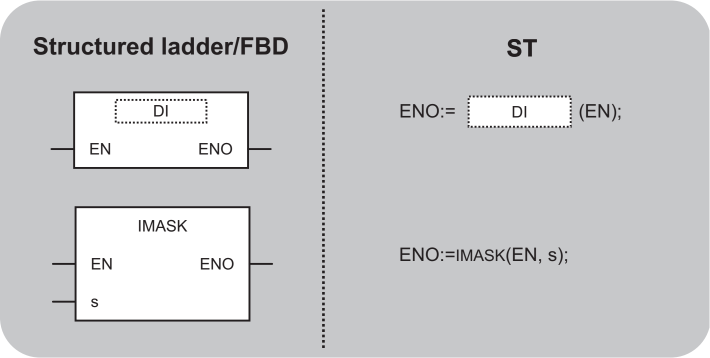
6.6 Program Execution Control Instructions

Interrupt disable/enable, interrupt program mask

DI, EI, IMASK

Basic High performance Process Redundant Universal LCPU

■For Basic model QCPU



Any of the following instruction can go in the dotted squares.

DI, EI

■Executing condition

Instruction	Executing condition
DI, EI, IMASK	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Interrupt mask data, or start number or the array of the device that stores interrupt mask data (when using the IMASK instruction)	Array of ANY16 (0..15)
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s)	—	○		—					

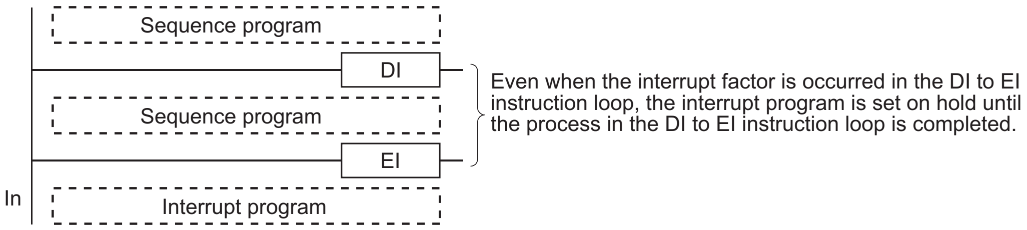
Processing details

■DI

- Disables the execution of an interrupt program until the EI instruction has been executed, even if a start cause for the interrupt program occurs.
- A DI status is entered when power is turned ON or when the CPU module is reset.

■EI

- Releases the disable interrupt status of the DI instruction, and enables the execution of the interrupt program with the interrupt pointer number which is allowed by the IMASK instruction. When the IMASK instruction is not executed, I32 to I47 are disabled.



Even when the interrupt factor is occurred in the DI to EI instruction loop, the interrupt program is set on hold until the process in the DI to EI instruction loop is completed.

■IMASK

- Enables/disables the execution of the interrupt program marked by the specified interrupt pointer by using the bit pattern of 8 points from the device specified for (s).
  - 1(ON) ··· Interrupt program execution enabled
  - 0(OFF) ··· Interrupt program execution disabled
- The interrupt pointer numbers corresponding to each bit are as shown below.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
ⓢ[0]	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
ⓢ[1]	I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
ⓢ[2]	I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
ⓢ[3]	I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
ⓢ[4]	I79	I78	I77	I76	I75	I74	I73	I72	I71	I70	I69	I68	I67	I66	I65	I64
ⓢ[5]	I95	I94	I93	I92	I91	I90	I89	I88	I87	I86	I85	I84	I83	I82	I81	I80
ⓢ[6]	I111	I110	I109	I108	I107	I106	I105	I104	I103	I102	I101	I100	I99	I98	I97	I96
ⓢ[7]	I127	I126	I125	I124	I123	I122	I121	I120	I119	I118	I117	I116	I115	I114	I113	I112

- When the power is turned ON or when the CPU module has been reset, the execution of interrupt programs I0 to I31, I48 to I127 is enabled, and the execution of interrupt programs I32 to I47 is disabled.
- The status of devices (s)[0] to (s)[7] are stored to SD715 to SD717 and SD781 to SD785 (storage area for the IMASK instruction mask pattern).
- Although the special registers are separated as SD715 to SD717 and SD781 to SD785, device numbers should be specified as (s)[0] to (s)[7] successively.

Point

- For information on interrupt conditions, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.
- The DI status (interrupt disabled) is active during the execution of an interrupt program. Do not insert the EI instructions in interrupt programs to attempt the execution of multiple interrupts, with interrupt programs running inside interrupt programs.
- If there are the EI and DI instructions within a master control, these instructions will be executed regardless of the execution or non-execution status of the MC instruction.

Operation error

- There is no operation error.

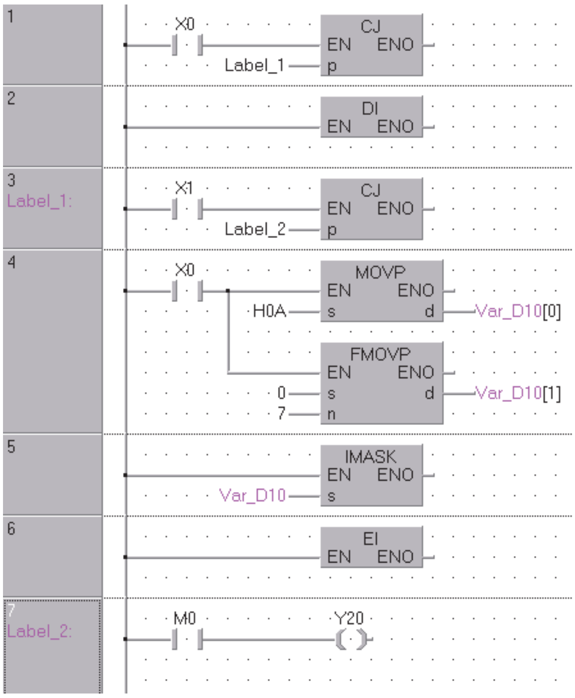
Program example

- In the following program, the interrupt programs with the interrupt pointer number I1 and I3 are set in the enable execution status while X0 is ON.

[Structured ladder/FBD]

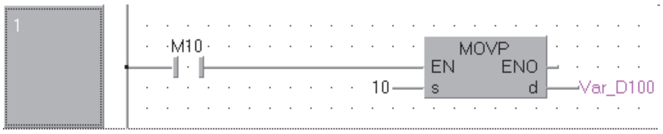
- Task_01 [Always] ··· interrupt enable program

The FEND instruction is not required.



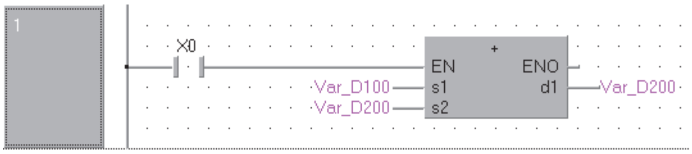
- Task_Interrupt No1 [Event] ··· I1 interrupt program

The IRET instruction is not required.



- Task_Interrupt No3 [Event] ··· I3 interrupt program

The IRET instruction is not required.



[ST]

- Task_01 [Always] ··· interrupt enable program

```
IF (X0=FALSE) THEN
    DI(TRUE);
END_IF;
IF (X1=FALSE) THEN
    IF X0 THEN
        MOVP(TRUE,H0A,Var_D10[0]);
        FMOVP(TRUE,0,7,Var_D10[1]));
    END_IF;
    IMASK(TRUE,Var_D10);
    EI(TRUE);
END_IF;
OUT(M0,Y20);
```

- Task_Interrupt No1 [Event] ··· I1 interrupt program

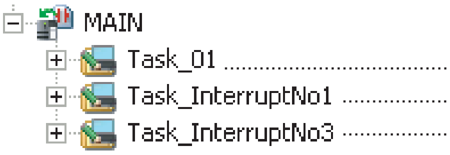
```
MOVP(M10,10,Var_D100);
```

- Task_Interrupt No3 [Event] ··· I3 interrupt program

```
IF M11 THEN
    Var_D200 := Var_D100 + Var_D200;
END_IF;
```

Point

For interrupt programs, create multiple tasks to register interrupt enable programs and interrupt programs.

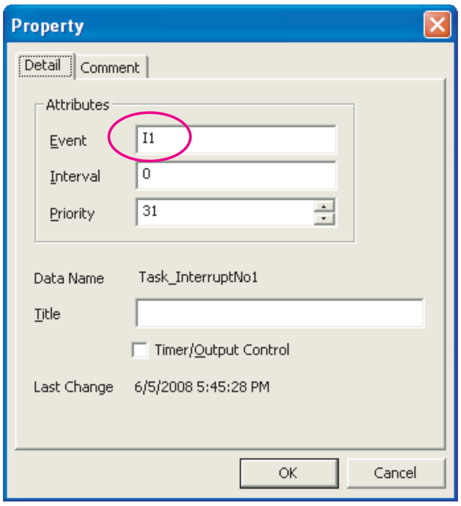


MAIN
- Task_01 .... Interrupt enable program
- Task_InterruptNo1 .... Interrupt I1 program
- Task_InterruptNo3 .... Interrupt I3 program

Task_Interrupt No1 [Always] to [Event] ··· Change to a interrupt program.

Task_Interrupt No3 [Always] to [Event] ··· Change to a interrupt program.

[Property setting]



Event: I1

Interval: 0

Priority: 31

Data Name: Task_InterruptNo1

Title:

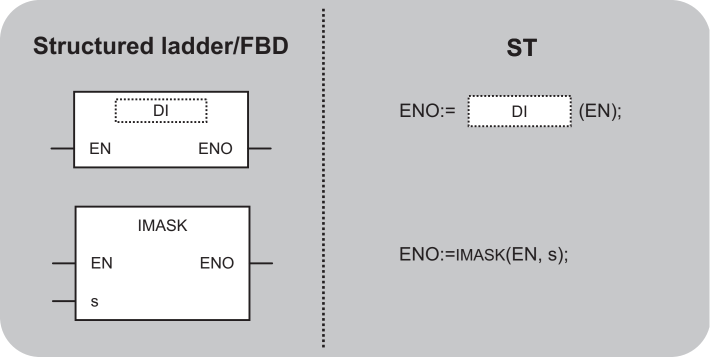
□ Timer/Output Control

Last Change: 6/5/2008 5:45:28 PM

Enter the device I1 or I3.

DI, EI, IMASK

■For High Performance model QCPU, Universal model QCPU, and LCPU



Any of the following instruction can go in the dotted squares.

DI, EI

■Executing condition

Instruction	Executing condition
DI, EI, IMASK	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Interrupt mask data, or start number or the array of the device that stores interrupt mask data (when using the IMASK instruction)	Array of ANY16 (0..15)
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s)	—	○		—					

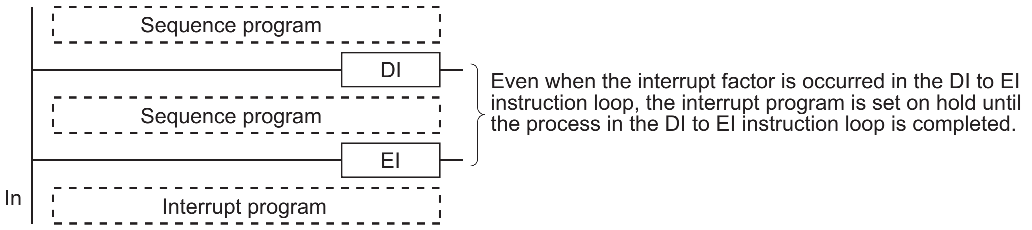
Processing details

■DI

- Disables the execution of an interrupt program until the EI instruction has been executed, even if a start cause for the interrupt program occurs.
- A DI status is entered when power is turned ON or when the CPU module is reset.

■EI

- Releases the disable interrupt status of the DI instruction, and enables the execution of the interrupt program with the interrupt pointer number which is allowed by the IMASK instruction and the fixed scan execution type program. When the IMASK instruction is not executed, I32 to I47 are disabled.



Even when the interrupt factor is occurred in the DI to EI instruction loop, the interrupt program is set on hold until the process in the DI to EI instruction loop is completed.

■IMASK

- Enables/disables the execution of the interrupt program marked by the specified interrupt pointer by using the bit pattern of 16 points from the device specified for (s).
  - 1(ON) ··· Interrupt program execution enabled
  - 0(OFF) ··· Interrupt program execution disabled
- The interrupt pointer numbers corresponding to each bit are as shown below.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
ⓢ[0]	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
ⓢ[1]	I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
ⓢ[2]	I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
ⓢ[3]	I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
ⓢ[4]	I79	I78	I77	I76	I75	I74	I73	I72	I71	I70	I69	I68	I67	I66	I65	I64
ⓢ[5]	I95	I94	I93	I92	I91	I90	I89	I88	I87	I86	I85	I84	I83	I82	I81	I80
ⓢ[6]	I111	I110	I109	I108	I107	I106	I105	I104	I103	I102	I101	I100	I99	I98	I97	I96
ⓢ[7]	I127	I126	I125	I124	I123	I122	I121	I120	I119	I118	I117	I116	I115	I114	I113	I112
ⓢ[8]	I143	I142	I141	I140	I139	I138	I137	I136	I135	I134	I133	I132	I131	I130	I129	I128
ⓢ[9]	I159	I158	I157	I156	I155	I154	I153	I152	I151	I150	I149	I148	I147	I146	I145	I144
ⓢ[10]	I175	I174	I173	I172	I171	I170	I169	I168	I167	I166	I165	I164	I163	I162	I161	I160
ⓢ[11]	I191	I190	I189	I188	I187	I186	I185	I184	I183	I182	I181	I180	I179	I178	I177	I176
ⓢ[12]	I207	I206	I205	I204	I203	I202	I201	I200	I199	I198	I197	I196	I195	I194	I193	I192
ⓢ[13]	I223	I222	I221	I220	I219	I218	I217	I216	I215	I214	I213	I212	I211	I210	I209	I208
ⓢ[14]	I239	I238	I237	I236	I235	I234	I233	I232	I231	I230	I229	I228	I227	I226	I225	I224
ⓢ[15]	I255	I254	I253	I252	I251	I250	I249	I248	I247	I246	I245	I244	I243	I242	I241	I240

- The following are the results when the power is turned ON or when the CPU module has been reset.

CPU module	Description
For High Performance model QCPU	The execution of interrupt programs I0 to I31, I48 to I255 is enabled, and the execution of interrupt programs I32 to I47 is disabled.
For Universal model QCPU and LCPU	The execution of interrupt programs I0 to I31, I45 to I255 is enabled, and the execution of interrupt programs I32 to I44 is disabled.

- The status of devices (s)[0] to (s)[15] are stored to SD715 to SD717 and SD781 to SD793 (storage area for the IMASK instruction mask pattern).
- Although the special registers are separated as SD715 to SD717 and SD781 to SD793, device numbers should be specified as (s)[0] to (s)[15] successively.

Point

- For information on interrupt conditions, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.
- The DI status (interrupt disabled) is active during the execution of an interrupt program. Do not insert the EI instructions in interrupt programs to attempt the execution of multiple interrupts, with interrupt programs running inside interrupt programs.
- If there are the EI and DI instructions within a master control, these instructions will be executed regardless of the execution or non-execution status of the MC instruction.

Operation error

- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (s) exceeds the corresponding device range.	—	—	—	—	○	○

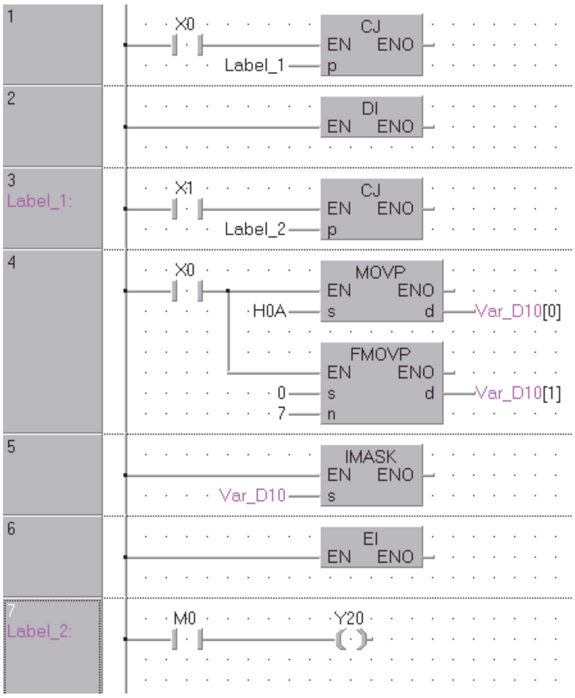
Program example

- In the following program, the interrupt programs with the interrupt pointer number I1 and I3 are set in the enable execution status while X0 is ON.

[Structured ladder/FBD]

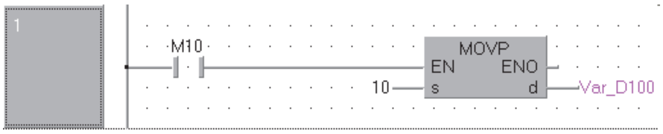
• Task_01 [Always] ··· interrupt enable program

The FEND instruction is not required.



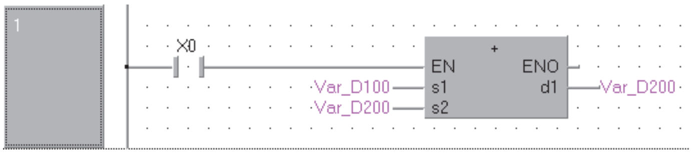
• Task_Interrupt No1 [Event] ··· I1 interrupt program

The IRET instruction is not required.



• Task_Interrupt No3 [Event] ··· I3 interrupt program

The IRET instruction is not required.



[ST]

- Task_01 [Always] ··· interrupt enable program

```
IF (X0=FALSE) THEN
    DI(TRUE);
END_IF;
IF (X1=FALSE) THEN
    IF X0 THEN
        MOVP(TRUE,H0A,Var_D10[0]);
        FMOVP(TRUE,0,15,Var_D10[1]);
    END_IF;
    IMASK(TRUE,Var_D10);
    EI(TRUE);
END_IF;
OUT(M0,Y20);
```

- Task_Interrupt No1 [Event] ··· I1 interrupt program

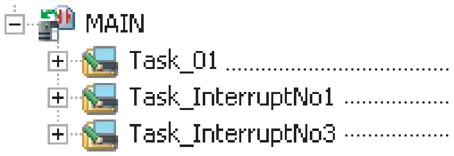
```
MOVP(M10,10,Var_D100);
```

- Task_Interrupt No3 [Event] ··· I3 interrupt program

```
IF M11 THEN
    Var_D200 := Var_D100 + Var_D200;
END_IF;
```

Point

For interrupt programs, create multiple tasks to register interrupt enable programs and interrupt programs.



MAIN

Task_01 .... Interrupt enable program

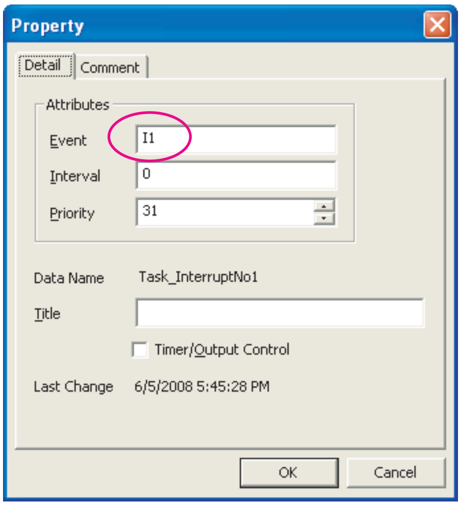
Task_InterruptNo1 .... Interrupt I1 program

Task_InterruptNo3 .... Interrupt I3 program

Task_Interrupt No1 [Always] to [Event] ··· Change to a interrupt program.

Task_Interrupt No3 [Always] to [Event] ··· Change to a interrupt program.

[Property setting]



Property

Detail | Comment

Attributes

Event: I1

Interval: 0

Priority: 31

Data Name: Task_InterruptNo1

Title:

□ Timer/Output Control

Last Change: 6/5/2008 5:45:28 PM

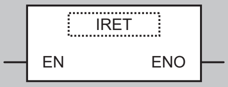
Enter the device I1 or I3.

Recovery from interrupt programs

IRET

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



ST

Not supported

The following instruction can go in the dotted squares.

IRET

■Executing condition

Instruction	Executing condition
IRET	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

- Used to end the interrupt program processing forcedly.
- Do not use when execute the interrupt program processing completely.

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4220	A pointer corresponding to the interrupt number does not exist.	○	○	○	○	○	○
4221	The FEND, GOEND or STOP instruction was executed after the interruption and before the IRET instruction execution.	○	○	○	○	○	○
4223	The IRET instruction was executed before executing the interrupt program.	○	○	○	○	○	○
	The IRET instruction was executed in the fixed scan execution type program.	—	—	—	—	○	○

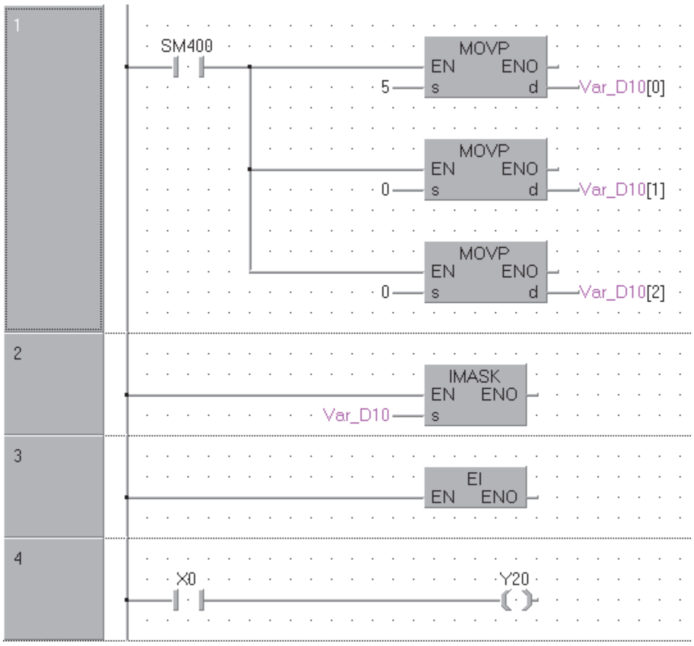
Program example

- In the following program, 1 is subtracted from Var_D0 if g_bool1 is ON, and 1 is added to Var_D0 if g_bool1 is OFF when the interrupt of No.3 occurs.

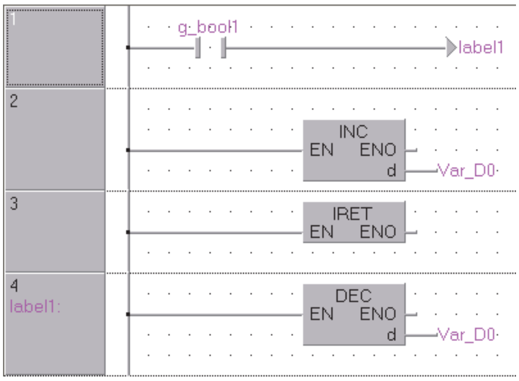
[Structured ladder/FBD]

- Task_01 [Always] ··· interrupt program

The FEND instruction is not required.

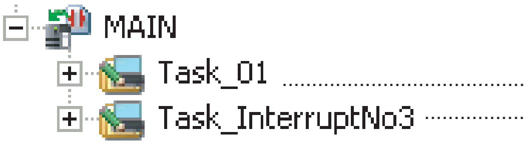


- Task_Interrupt No3 [Event] ··· interrupt program



Point

For interrupt programs, create multiple tasks to register interrupt enable programs and interrupt programs.



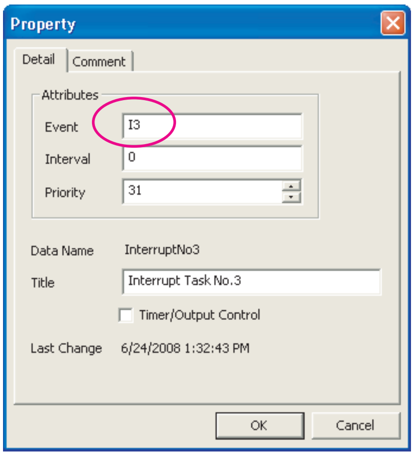
MAIN

Task_01 .... Interrupt enable program

Task_InterruptNo3 .... Interrupt program

Task_Interrupt No3 [Always] to [Event] ··· Change to a interrupt program.

[Property setting]



Event	I3
Interval	0
Priority	31
Data Name	InterruptNo3
Title	Interrupt Task No.3
□ Timer/Output Control	
Last Change	6/24/2008 1:32:43 PM

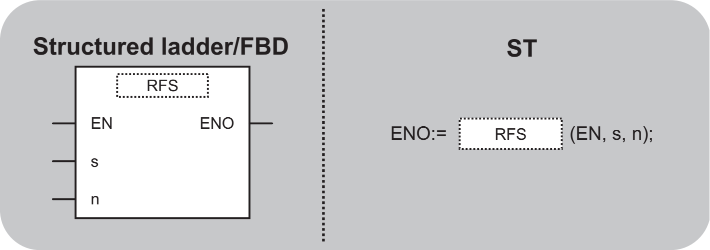
Enter the device I3.

6.7 I/O Refresh Instructions

I/O refresh

RFS(P)

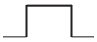
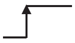
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

RFS, RFSP

■Executing condition

Instruction	Executing condition
RFS	
RFSP	

■Argument

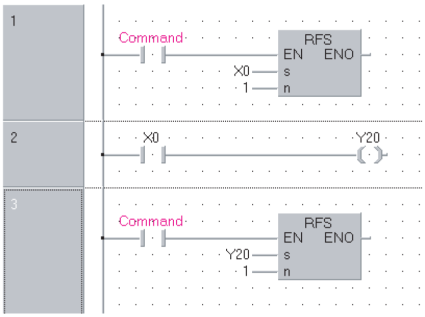
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that executes the refresh function	Bit
	n	Number of refreshes	ANY16
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○ (X, Y only)	—							—
n	○	○							—

Processing details

- Refreshes only the device being scanned during a scan, and functions to fetch external input or to output data to an output module.
- Fetching of input from or sending output to an external source is conducted in batch only after the END processing in the program is executed, so it is not possible to output a pulse signal to an outside source during the execution of a scan. When the I/O refresh instruction is executed, the inputs (X) or outputs (Y) of the corresponding device numbers are refreshed forcibly midway through program execution. Therefore, a pulse signal can be output to an external source during a scan.
- Use direct access inputs (DX) or direct access outputs (DY) to refresh inputs (X) or outputs (Y) in units of 1 point.

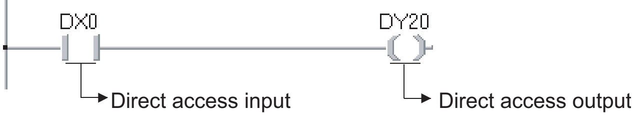
[Program based on the RFS instruction]



Refreshes X0.

Refreshes Y20.

[Program based on direct access input and direct access output]



Operation error

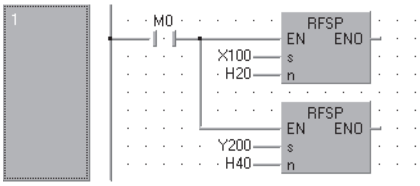
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (s) exceeds the proximate I/O range.	○	○	○	○	○	—

Program example

- In the following program, the devices from X100 to X11F, and from Y200 to Y23F are refreshed when M0 turns ON.

[Structured ladder/FBD]



[ST]

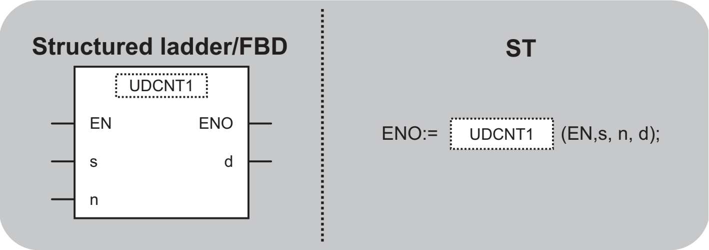
```
RFSP(M0,X100,H20);
RFSP(M0,Y200,H40);
```

6.8 Other Convenient Instructions

Single-phase input up/down counter

UDCNT1

~~Basic~~ High performance Process ~~Redundant~~ Universal LCPU

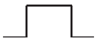


```
ENO:= UDCNT1 (EN,s, n, d);
```

The following instruction can go in the dotted squares.

UDCNT1

■Executing condition

Instruction	Executing condition
UDCNT1	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	s[0]:Input number for the counter input s[1]:For setting the up/down counter • OFF: Up-counter • ON: Down-counter	Array of bit (1..2)
	n	Set value	ANY16
Output argument	ENO	Execution result	Bit
	d	Counter number that starts counting by the UDCNT1 instruction	ANY16

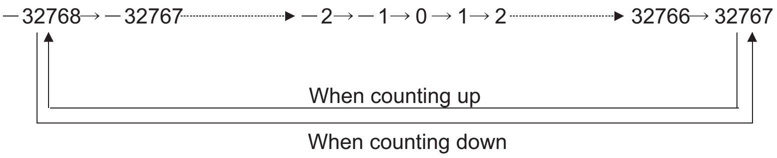
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○(X only)*1	—	—	—					—
n	△*2	△*2	△*2	○					—
(d)	—	△*2(C only)	—	—					—

*1 Specify the array in which X is set as a device for global label. X devices can be used only in the range of I/O points (accessible points to the actual I/O module).

*2 Local devices and file registers per program cannot be used as setting data.

Processing details

- When the input specified for (s)[0] turns from OFF to ON, the current value of the counter specified for (d) will be updated.
- The direction of the count is determined by the ON/OFF status of the input specified for (s)[1].
  - OFF: Up counter (counts by adding to the current value)
  - ON: Down counter (counts by subtracting from the current value)
- When the count is going up, the counter contact specified for (d) turns ON when the current value becomes identical with the setting value specified for n. However, the current value count will continue even when the contact of the counter specified for (d) turns ON. (Refer to Program Example)
- When the count is going down, the counter for the contact specified for (d) turns OFF when the current value reaches the setting value - 1. (Refer to Program Example)
- The counter specified for (d) is a ring counter. If it is counting up when the current value is 32767, the current value will become -32768. Further, if it is counting down when the current value is -32768, the current value will become 32767. The count processing performed on the current value is as shown below.



- The UDCNT1 instruction triggers counting when the execution command is turned from OFF to ON and suspends counting when the execution command is turned from ON to OFF. When the execution command is turned from OFF to ON again, the counting resumes from the suspended value.
- The RST instruction clears the current value of the counter specified for (d) and turns the contact OFF.

Point

- With the UDCNT1 instruction, the argument device data are registered in the work area of the CPU module and counting operation is processed as a system interrupt. (The device data registered in the work area is cleared by turning the execution command OFF, or turning the STOP/RUN switch from STOP to RUN.) For this reason, the pulses that can be counted must have longer ON and OFF times than the interrupt interval of the CPU module. The interrupt interval of the CPU module is 1ms.
- The setting value cannot be changed during counting directed by the UDCNT1 instruction (while the execution command is ON). To change the setting value, turn OFF the execution command.
- Counters which have been specified by the UDCNT1 instruction cannot be used by other instructions. If they are used by other instructions, they will not be capable of returning an accurate count.
- The UDCNT1 instruction can be used as many as 6 times within all the programs being executed. The seventh and the subsequent UDCNT1 instructions are not processed.

Operation error

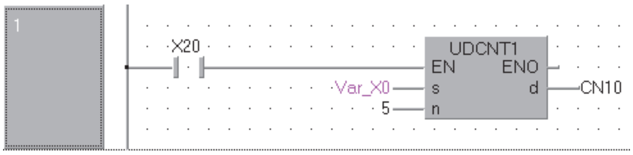
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (s) exceeds the corresponding device range.	—	○	○	—	○	○

Program example

- In the following program, the number of turns from OFF to ON in Var_X0 are counted using C0 (up/down counter) after X20 has turned ON.

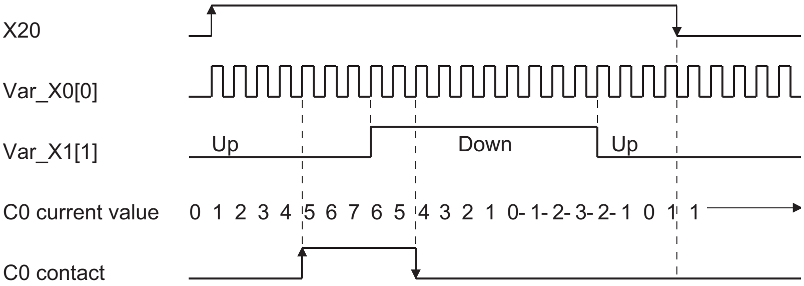
[Structured ladder/FBD]



[ST]

```
UDCNT1(X20,Var_X0,5,C0);
```

[Operation]

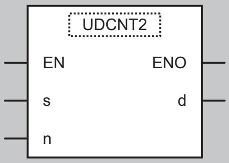


Two-phase input up/down counter

UDCNT2

✕Basic | High performance | Process | ✕Redundant | Universal | LCPU

Structured ladder/FBD



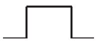
ST

ENO:= UDCNT2 (EN,s, n, d);

The following instruction can go in the dotted squares.

UDCNT2

■Executing condition

Instruction	Executing condition
UDCNT2	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	s[0]:Input number for the counter input (A-phase pulse) s[1]:Input number for the counter input (B-phase pulse)	Array of bit (1..2)
	n	Set value	ANY16
Output argument	ENO	Execution result	Bit
	d	Counter number that starts counting by the UDCNT2 instruction	ANY16

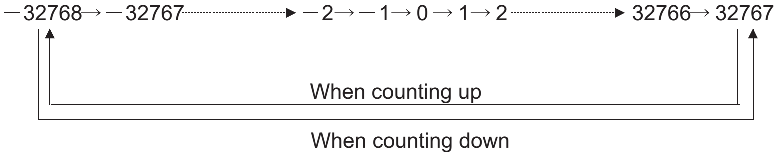
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○(X only)[*1]	—	—	—					—
n	△[*2]	△[*2]	△[*2]	○					—
(d)	—	△[*2](C only)	—	—					—

*1 Specify the array in which X is set as a device for global label. X devices can be used only in the range of I/O points (accessible points to the actual I/O module).

*2 Local devices and file registers per program cannot be used as setting data.

Processing details

- The current value of the counter specified for (d) is updated depending on the status of the input specified for (s)[0] (A-phase pulse) and the status of the input specified for (s)[1] (B-phase pulse).
- Direction of the count is determined in the following manner.
  - When (s)[0] is ON, if (s)[1] turns from OFF to ON, count-up operation is performed. (Values are added to the current value of the counter.)
  - When (s)[0] is ON, if (s)[1] turns from ON to OFF, count-down operation is performed. (Values are subtracted from the current value of the counter.)
  - No count operation is performed if (s)[0] is OFF.
- When the count is going up, the counter contact specified for (d) turns ON when the current value becomes identical with the setting value specified for n. However, the current value count will continue even when the contact of the counter specified for (d) turns ON. (Refer to Program Example)
- When the count is going down, the counter for the contact specified for (d) turns OFF when the current value reaches the setting value - 1. (Refer to Program Example)
- The counter specified for (d) is a ring counter. If it is counting up when the current value is 32767, the current value will become -32768. Further, if it is counting down when the current value is -32768, the current value will become 32767. The count processing performed on the current value is as shown below.



- Count processing conducted according to the UDCNT2 instruction begins when the execution command is turned from OFF to ON, and is suspended when it is turned from ON to OFF. When the execution command is turned from OFF to ON again, the counting resumes from the suspended value.
- The RST instruction clears the current value of the counter specified for (d) and turns the contact OFF.

Point

- With the UDCNT2 instruction, the argument device data are registered in the work area of the CPU module and counting operation is processed as a system interrupt. (The device data registered in the work area is cleared by turning the execution command OFF, or turning the STOP/RUN switch from STOP to RUN.) For this reason, the pulses that can be counted must have longer ON and OFF times than the interrupt interval of the CPU module. The interrupt interval of the CPU module is 1ms.
- The setting value cannot be changed during counting directed by the UDCNT2 instruction (while the execution command is ON). To change the setting value, turn OFF the execution command.
- Counters which have been specified by the UDCNT2 instruction cannot be used by other instructions. If they are used by other instructions, they will not be capable of returning an accurate count.
- The UDCNT2 instruction can be used as many as 5 times within all the programs being executed. The sixth and the subsequent UDCNT2 instructions are not processed.

Operation error

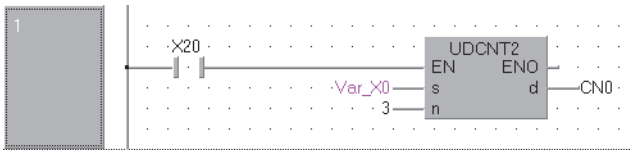
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (s) exceeds the corresponding device range.	—	○	○	—	○	○

Program example

- In the following program, the status of Var_X0[0] and Var_X0[1] are counted by C0 (up/down counter) after X20 has turned ON.

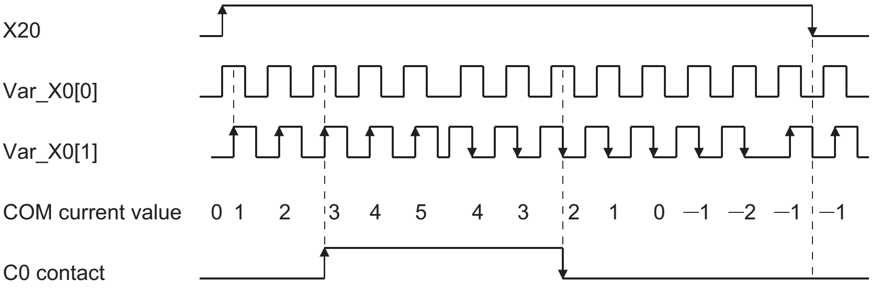
[Structured ladder/FBD]



[ST]

```
UDCNT2(X20,Var_X0,3,CN0);
```

[Operation]

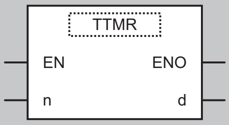


Teaching timer

TTMR

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



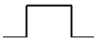
ST

ENO:= TTMR (EN, n, d);

The following instruction can go in the dotted squares.

TTMR

■Executing condition

Instruction	Executing condition
TTMR	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Multiplier of the measurement value	ANY16
Output argument	ENO	Execution result	Bit
	d	d[0]:Measurement value storage device d[1]:For CPU module system use	Array of ANY16 (1..2)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	—	○		○					—
(d)	—	○		—					—

Processing details

- Measures the time while the execution command is ON in units of seconds, multiplies by the multiplier specified for n, and stores the result to the device specified for (d).
- Clears the device specified for (d)[0] or (d)[1] when the execution command is turned from OFF to ON.
- The multipliers that can be specified for n are as shown below.

n	Multiplier
0	1
1	10
2	100

Point

- Time measurements are conducted when the TTMR instruction is executed. Using the JMP or similar instruction to jump the TTMR instruction will make it impossible to get an accurate measurement.
- Do not change the multiplier specified for n while the TTMR instruction is being executed. Changing this multiplier will result in an inaccurate value being returned.
- The TTMR instruction can also be used in low-speed execution type programs.
- The device specified for (d)[1] is used by the system of the CPU module, so users should not change its value. If users change this value, the value stored to the device specified for (d)[0] will no longer be accurate.

- No processing is performed when the value specified for n is other than 0 to 2.

Operation error

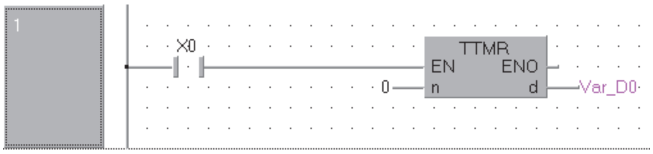
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (d) exceeds the corresponding device range.	—	○	○	—	○	○

Program example

- In the following program, amount of time when X0 is ON is stored to Var_D0.

[Structured ladder/FBD]



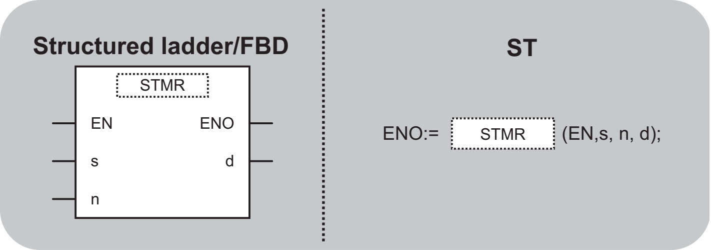
[ST]

TTMR(X0,0,Var_D0);

Special function timer

STMR

Basic ✕ | High performance | Process | Redundant ✕ | Universal | LCPU



ENO:= STMR (EN,s, n, d);

The following instruction can go in the dotted squares.

STMR

■Executing condition

Instruction	Executing condition
STMR	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Timer number	ANY16
	n	Set value	ANY16
Output argument	ENO	Execution result	Bit
	d	d[0]:Off-delay timer output d[1]:One-shot timer output after OFF d[2]:One-shot timer output after ON d[3]:On-delay and off-delay timer output	Array of bit (1..4)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	△[1]	—	—					—
n	○	○	○	○					—
(d)	○	—	—	—					—

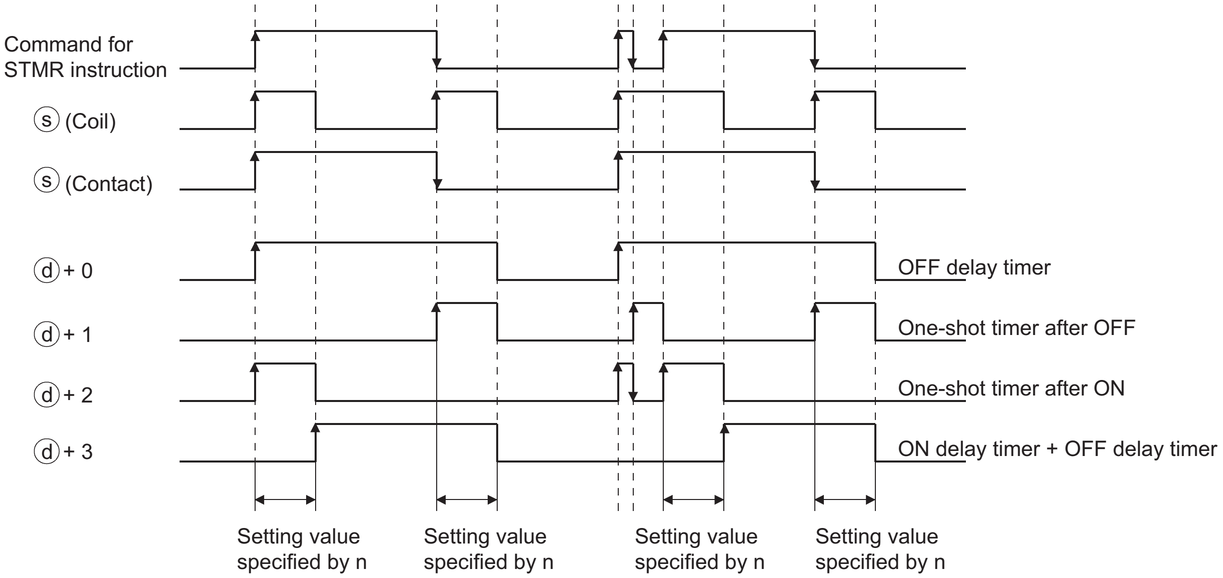
[1] Only for timers (T)

Processing details

- The STMR instruction uses the 4 points from the device specified for (d) to perform four types of timer output.

Item	Description
OFF-delay timer output ((d)[0])	Turns ON at the rising edge command of the STMR instruction, and after the falling edge command, turns OFF when the amount of time specified for n has passed.
One-shot timer output after OFF ((d)[1])	Turns ON at the falling edge command of the STMR instruction, and turns OFF when the amount of time specified for n has passed.
One-shot timer output after ON ((d)[2])	Turns ON at the rising edge the command for the STMR instruction, and turns OFF either when the amount of time specified for n has passed, or when the command of the STMR instruction turns OFF.
ON-delay and OFF-delay timer output ((d)[3])	Turns ON at the falling edge of the timer coil, and after the falling edge command of the STMR instruction, turns OFF when the amount of time specified for n has passed.

- The timer coil specified for (s) turns ON at the rising edge and falling edge command of the STMR instruction, and starts measurement of the current value.
  - The timer coil measures to the point where the value reaches the setting value specified for n, then enters a time up status and turns OFF.
  - If the command of the STMR instruction turns OFF before the timer coil reaches the time up status, it will remain ON. Timer measurement is continued at this time. When the STMR instruction command turns ON once again, the current value will be cleared to 0 and measurement will begin once again.
- The timer contact turns ON at the rising edge command of the STMR instruction, and after the falling edge is reached, the timer coil turns OFF at the falling edge command of the STMR instruction. The timer contact is used by the CPU module system, and cannot be used by the user.


Command for STMR instruction
ⓢ (Coil)
ⓢ (Contact)
ⓓ + 0
ⓓ + 1
ⓓ + 2
ⓓ + 3
OFF delay timer
One-shot timer after OFF
One-shot timer after ON
ON delay timer + OFF delay timer
Setting value specified by n
Setting value specified by n
Setting value specified by n
Setting value specified by n

- Measurement of the current value of the timer specified in the STMR instruction is conducted regardless of ON/OFF status of the STMR instruction. If the STMR instruction is jumped with the JMP or similar instruction, it will not be possible to get accurate measurement.
- Measurement unit for the timer specified for (d) is identical to the low-speed timer.
- Values between 0 and 32767 can be set for n. No processing is performed when the value specified for n is other than 0 to 32767.
- The timer specified for (s) cannot be used in the OUT instruction. If the STMR instruction and the OUT instruction use the same timer number, accurate operation will not be conducted.

Operation error

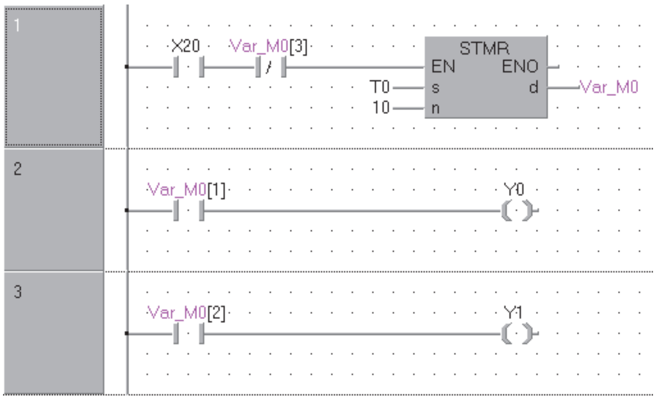
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (d) exceeds the corresponding device range.	—	○	○	—	○	○

Program example

- In the following program, Y0 and Y1 are turned ON and OFF every second (flicker) when X20 turns ON. (Uses 100ms timer)

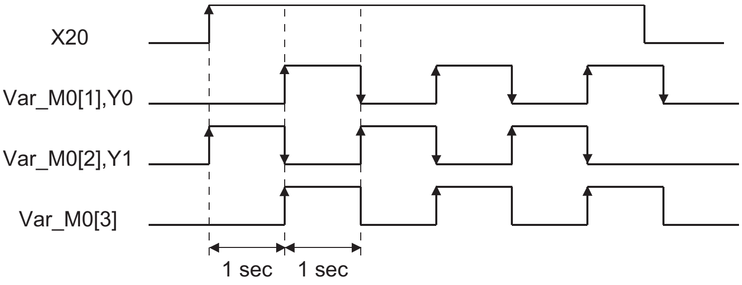
[Structured ladder/FBD]



[ST]

```
STMR(X20 AND NOT(Var_M0[3]),T0,10,Var_M0);
OUT(Var_M0[1],Y0);
OUT(Var_M0[2],Y1);
```

[Timing chart]



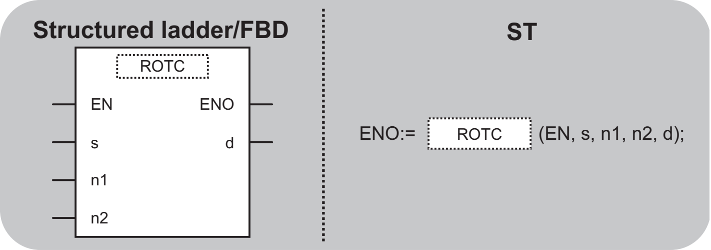
Precautions

Note that the STMR instruction operates when the instruction is used within the range written data by the online program change. For details, refer to the User's Manual (Function Explanation, Program Fundamentals) for the CPU module used.

Rotary table shortest direction control

ROTC

✗Basic | High performance | Process | ✗Redundant | Universal | LCPU

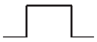


ENO:= ROTC (EN, s, n1, n2, d);

The following instruction can go in the dotted squares.

ROTC

■Executing condition

Instruction	Executing condition
ROTC	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	s[0]: For measuring the number of table rotations (for system) s[1]: Call station number s[2]: Call item number	Array of ANY16 (1..3)
	n1	Number of table divisions (2 to 32767)	ANY16
	n2	Number of low-speed sections (value from 0 to n1)	ANY16
Output argument	ENO	Execution result	Bit
	d	d[0]:A-phase input signal d[1]:B-phase input signal d[2]:0 point detection input signal d[3]:High-speed forward rotation output signal (for system) d[4]:Low-speed forward rotation output signal (for system) d[5]:Stop output signal (for system) d[6]:Low-speed reverse rotation output signal (for system) d[7]:High-speed reverse rotation output signal (for system)	Array of bit (1..8)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—					—
n1	○	○		○					—
n2	○	○		○					—
(d)	○	—		—					—

Processing details

- This control functions to enable shortest direction control of the rotary table to the position of the station number specified for (s)[1] in order to remove or deposit an item whose number has been specified for (s)[2] on a rotary table with equal divisions of the value specified for n1.
- The item number and station number are controlled as items allocated by counterclockwise rotation.
- The system uses (s)[0] as a counter to instruct it as to what item is at which number counting from station number 0. Do not rewrite the sequence program data. Accurate controls will not be possible in cases where users have rewritten the data.
- The value of n2 should be less than the number of table divisions that were specified for n1.
- (d)[0] and (d)[1] are A and B phase input signals that are used to detect whether the direction of the rotary table rotation is forward or reverse. The direction of rotation is judged by whether the B-phase pulse is at its rising or falling pulse when the A-phase pulse is ON.
  - When the B-phase is at the rising edge: Forward rotation (clockwise rotation)
  - When the B-phase is at the falling edge: Reverse rotation (counterclockwise rotation)
- (d)[2] is the 0 point detection signal that turns ON when item number 0 has arrived at the station No. 0. When the device specified for (d)[2] turns ON while the ROTC instruction is being executed, (s)[0] is cleared. It is best to perform this clear operation first, then to begin shortest direction control with the ROTC instruction.
- The data from (d)[3] to (d)[7] consists of output signals needed to control the table's operation. The output signal of one of the devices from (d)[3] to (d)[7] will turn ON in response to the execution results of the ROTC instruction.
- If the command for the ROTC instruction is OFF, clears all (d)[3] to (d)[7] without performing shortest direction control.
- The ROTC instruction can be used only one time in all programs where it is executed. Attempts to use it more than one time will result in inaccurate operations.
- No processing is performed when the value of (s)[0] to (s)[2], or the value of n2 is greater than n1.

Operation error

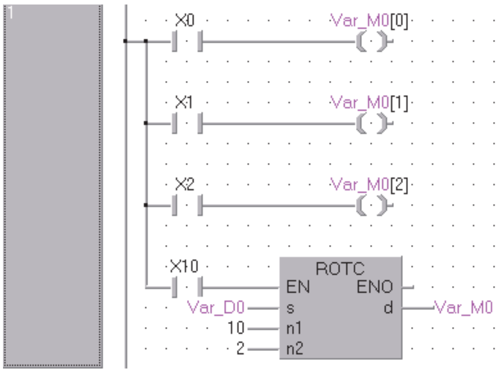
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (s) or (d) exceeds the corresponding device range.	—	○	○	—	○	○

Program example

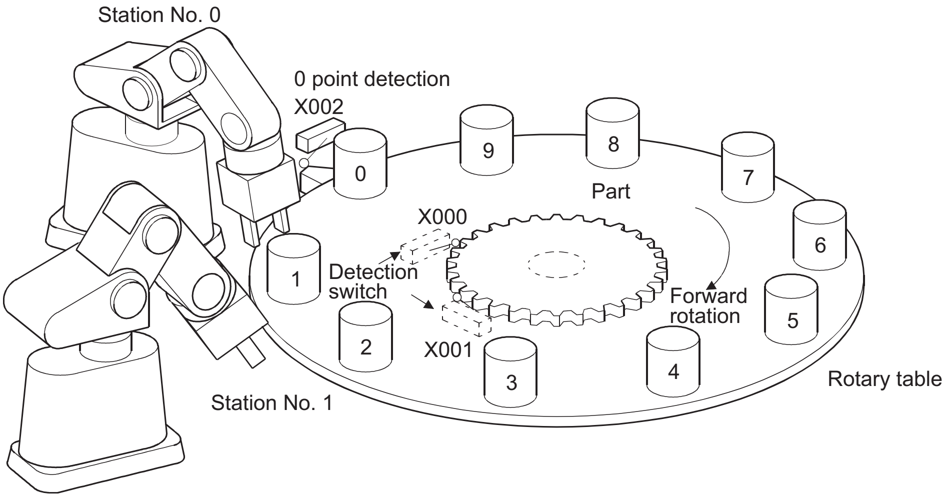
- In the following program, item at section Var_D0[2] on a 10-division rotary table is taken in and out at section Var_D0[1], and the rotation direction and the control speed of the motor are determined by two sections ahead and behind of the item when the table is rotated at low speed.

[Structured ladder/FBD]



[ST]

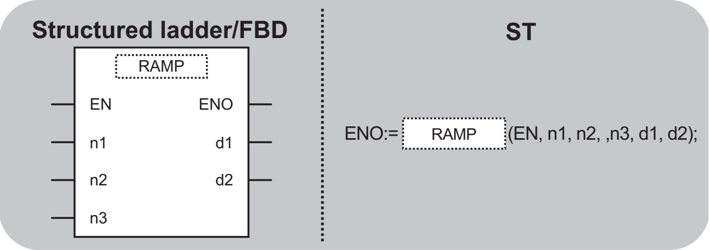
```
OUT(X0,Var_M0[0]);
OUT(X1,Var_M0[1]);
OUT(X2,Var_M0[2]);
ROTC(X10,Var_D0,10,2,Var_M0);
```



Ramp signal

RAMP

Basic High performance Process Redundant Universal LCPU

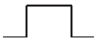


ENO:= RAMP (EN, n1, n2, ,n3, d1, d2);

The following instruction can go in the dotted squares.

RAMP

■Executing condition

Instruction	Executing condition
RAMP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Initial value	ANY16
	n2	Final value	ANY16
	n3	Number of transitions	ANY16
Output argument	ENO	Execution result	Bit
	d1	d1[0]:Current value d1[1]:Number of executions	Array of ANY16 (1..2)
	d2	d2[0]:Completion device d2[1]:Bit for selecting data retainment at completion	Array of bit (1..2)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n1	○	○						○	—
n2	○	○						○	—
n3	○	○						○	—
(d1)	○	○						—	—
(d2)	○	—						—	—

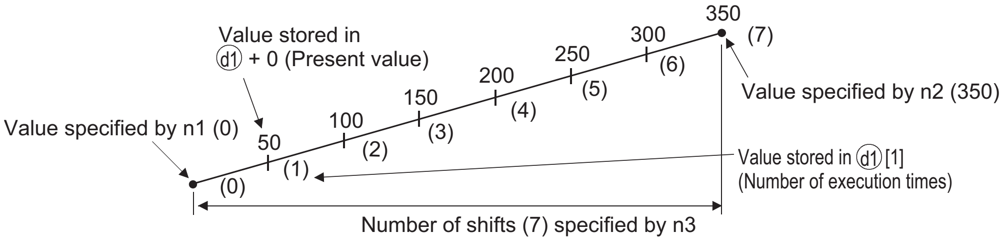
Processing details

- When the execution command is ON, the following processing is executed.
  - Shifts from the value specified for n1 to the value specified for n2 in the number of times specified for n3.
  - For n3, specify the number of scans (number of shifts) required for shift from n1 to n2. No processing is performed when the value specified for n3 is other than 0 to 32768.
  - The system uses (d1)[1] to store the number of times the instruction has been executed.
  - The value of one variation (one scan) is obtained by the expression below.

$$\text{Value of one variation (one scan)} = \frac{\text{(Value specified by n2)} - \text{(Value specified by n1)}}{\text{(Value specified by n3)}}$$

Ex.

0 is varied to 350 in seven scans as shown below.



350
(7)
Value stored in
d1 + 0 (Present value)
300
(6)
250
(5)
Value specified by n2 (350)
200
(4)
150
(3)
100
(2)
Value specified by n1 (0)
50
(1)
(0)
Value stored in d1 [1]
(Number of execution times)
Number of shifts (7) specified by n3

When the calculated one variation is indivisible, compensation is made to achieve the value specified for n2 by the number of shifts specified for n3.

Hence, a linear ramp may not be made.

- If the scan is performed for the number of moves specified for n3, the completion device specified for (d2)[0] will turn ON. The ON/OFF status of the completion device and the value of (d1)[0] are determined by the ON/OFF status of the device specified for (d2)[1].
  - When (d2)[1] is OFF, (d2)[0] will turn OFF at the next scan, and the RAMP instruction will begin a new transition operation from the initial value.
  - When (d2)[1] is ON, (d2)[0] will remain ON, and the value of (d1)[0] will not change.
- When the command is turned OFF during the execution of this instruction, the value of (d1)[0] will not change following this. When the command turns ON again, the RAMP instruction will begin a new transition from the initial value.
- Do not change the values of n1 and n2 before the completion device specified for (d2)[0] turns ON. Since the same expression is used every scan to calculate the value stored in (d1)[1], changing n1/n2 may cause a sudden variation.

Operation error

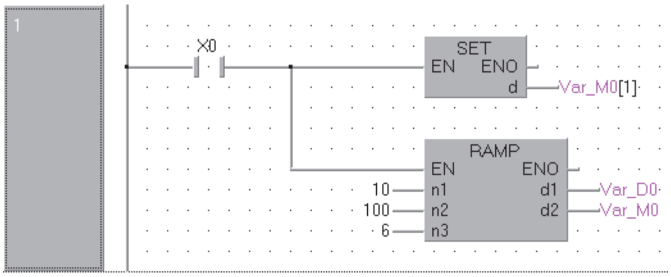
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (d1) or (d2) exceeds the corresponding device range.	—	○	○	—	○	○

Program example

- In the following program, the value of Var_D0 is changed from 10 to 100 in a total of 6 scans, and the value of Var_D0 is saved when the transition has been completed.

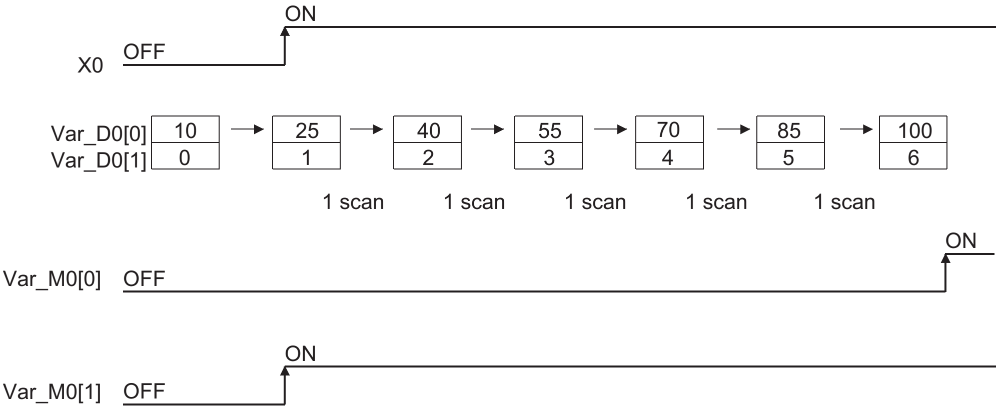
[Structured ladder/FBD]



[ST]

```
SET(X0,Var_M0[1]);
RAMP(X0,10,100,6,Var_D0,Var_M0);
```

[Timing chart]

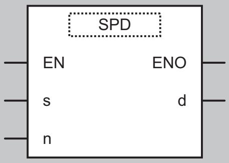

ON
X0 OFF
Var_D0[0] 10 → 25 → 40 → 55 → 70 → 85 → 100
Var_D0[1] 0 1 2 3 4 5 6
1 scan 1 scan 1 scan 1 scan 1 scan
ON
Var_M0[0] OFF
ON
Var_M0[1] OFF

Pulse density measurement

SPD

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



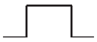
ST

```
ENO:= SPD (EN, s, n, d);
```

The following instruction can go in the dotted squares.

SPD

■Executing condition

Instruction	Executing condition
SPD	

■Argument

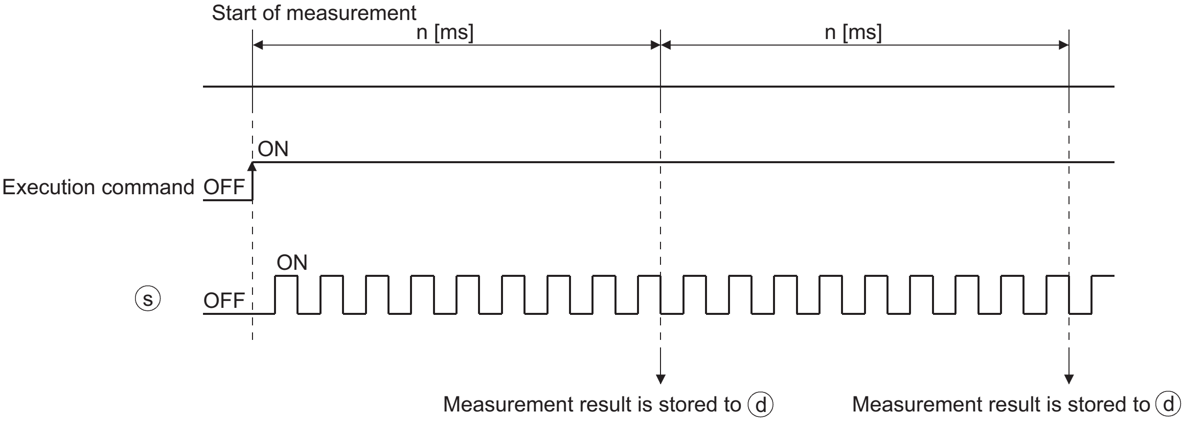
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Pulse input	Bit
	n	Measurement time (unit: ms)	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the measurement result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○(X only)	—		—					—
n	△[*1]	△[*1]		○					—
(d)	—	△[*1]		—					—

*1 Local devices and file registers per program cannot be used as setting data.

Processing details

- The number of turning from OFF to ON input from the device specified for (s) is counted for just the amount of time specified for n, and results of the count are stored to the device specified for (d).


Start of measurement
n [ms]
n [ms]
ON
Execution command OFF
ON
ⓢ OFF
Measurement result is stored to ⓓ
Measurement result is stored to ⓓ

- When the measurement directed by the SPD instruction has been completed, the measurement is executed again from 0. Turn OFF the execution command to stop the measurement directed by the SPD instruction.

Point

- With the SPD instruction, the argument device data are registered in the work area of the CPU module and counting operation is processed as a system interrupt. (The device data registered in the work area is cleared by turning the execution command OFF, or turning the STOP/RUN switch from STOP to RUN.) For this reason, the pulses that can be counted must have longer ON and OFF times than the interrupt interval of the CPU module. The interrupt interval of the CPU module is 1ms.
- When the High Performance model QCPU is used and n = 0, the instruction is not processed.
- The SPD instruction can be used as many as 6 times within all the programs being executed. The seventh and the subsequent SPD instructions are not processed.
- While the measurement is in execution (while the command input is ON) by the SPD instruction, the setting value cannot be changed. Turn OFF the command input before changing the setting value.

Operation error

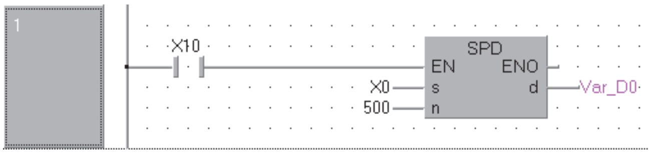
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (s) exceeds the corresponding device range.	—	○	○	—	○	○

Program example

- In the following program, the pulses input to X0 are measured for a period of 500ms when X10 turns ON, and the result is stored to Var_D0.

[Structured ladder/FBD]



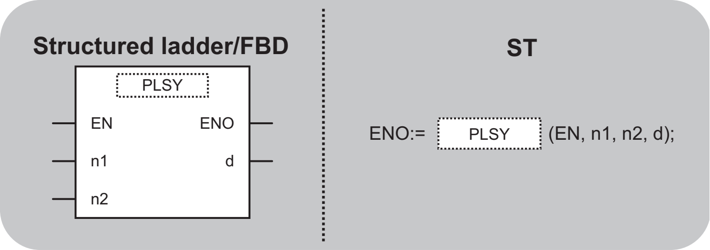
[ST]

```
SPD(X10,X0,500,Var_D0);
```

Fixed cycle pulse output

PLSY

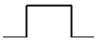
~~Basic~~ High performance Process ~~Redundant~~ Universal LCPU



The following instruction can go in the dotted squares.

PLSY

■Executing condition

Instruction	Executing condition
PLSY	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Frequency, or device number that stores frequency	ANY16
	n2	Output counts, or device number that stores output counts	ANY16
Output argument	ENO	Execution result	Bit
	d	Device number that generates pulse outputs	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n1	○	○							—
n2	○	○							—
(d)	△(Y only)	—							—

Processing details

- Outputs a pulse at a frequency specified for n1 in the number of times specified for n2, to the output module with the output signal (Y) specified for (d).
- Frequency of n1 can be set between 1 and 100Hz. If n1 is other than 1 to 100Hz, the PLSY instruction will not be executed.
- The number of outputs that can be specified for n2 is between 0 and 65535 (0000H to 0FFFFH). If n2 is set to 0, pulses are continuously output.
- Only an output number corresponding to the output module can be specified for pulse output (d).
- Pulse output commences with the rising edge command of the PLSY instruction. Pulse output is suspended when the PLSY instruction command turns OFF.

Point

- With the PLSY instruction, the argument device data are registered in the work area of the CPU module and outputting operation is processed as a system interrupt. (The device data registered in the work area is cleared by turning the execution command OFF, or turning the STOP/RUN switch from STOP to RUN.) For this reason, the pulses that can be output must have longer ON and OFF times than the interrupt interval of the CPU module. The interrupt interval of the CPU module is 1ms.
- Do not change the argument for the PLSY instruction during pulse output directed by the PLSY instruction (while the execution command is ON). To change the argument, turn OFF the execution command.
- The PLSY instruction can be used only once in all programs executed by the CPU module. The second and the subsequent PLSY instructions are not processed.

Operation error

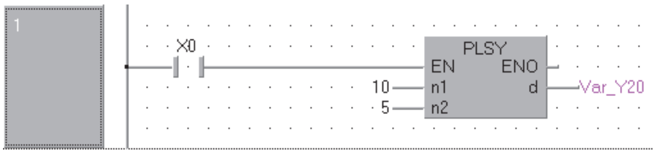
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (d) exceeds the corresponding device range.	—	○	○	—	○	○

Program example

- In the following program, 10Hz pulse is output for 5 times to Var_Y20 when X0 turns ON.

[Structured ladder/FBD]



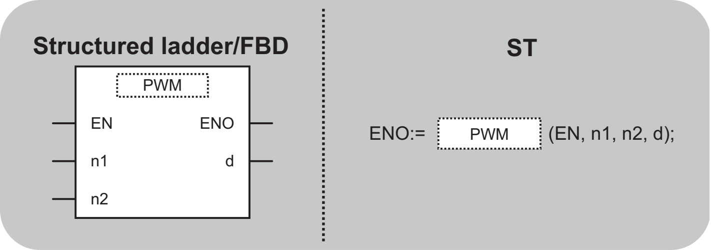
[ST]

```
PLSY(X0,10,5,Var_Y20);
```

Pulse width modulation

PWM

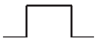
~~Basic~~ High performance Process ~~Redundant~~ Universal LCPU



The following instruction can go in the dotted squares.

PWM

■Executing condition

Instruction	Executing condition
PWM	

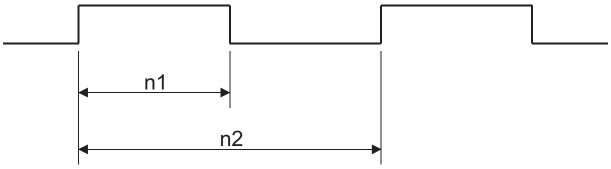
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	ON-time, or device number that stores ON-times	ANY16
	n2	Cycle, or device number that stores cycles	ANY16
Output argument	ENO	Execution result	Bit
	d	Device number that generates pulse outputs	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n1	○	○							—
n2	○	○							—
(d)	△(Y only)	—							—

Processing details

- Outputs the ON-time specified for n1 and the pulse of the cycle specified for n2 to an output module specified for (d).



- The setting ranges for n1 and n2 are shown below.

CPU module model	Setting range for n1 and n2 [ms][*1]
High Performance model QCPU, Process CPU, Universal model QCPU, LCPU	1 to 65535 (0001 to 0FFFF)

*1 The value specified for n1 should be less than the value specified for n2.

Point

- With the PWM instruction, the argument device data are registered in the work area of the CPU module and outputting operation is processed as a system interrupt. (The device data registered in the work area is cleared by turning the execution command OFF, or turning the STOP/RUN switch from STOP to RUN.) The interrupt interval of the CPU module (interrupt interval of n1, n2) is 1ms. For this reason, the PWM instruction can be used only once within all the programs being executed by the CPU module.
- When both n1 and n2 are 0, n1≥n2, or the PWM instruction is executed twice or more, the instruction is not processed.
- Do not change the argument for the PWM instruction during pulse output directed by the PWM instruction (while the execution command is ON). To change the argument, turn OFF the execution command.

Operation error

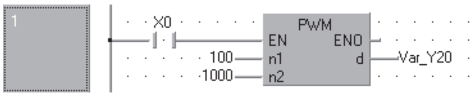
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (d) exceeds the corresponding device range.	—	○	○	—	○	○

Program example

- In the following program, 100ms pulse is output for every second to Var_Y20 when X0 turns ON.

[Structured ladder/FBD]



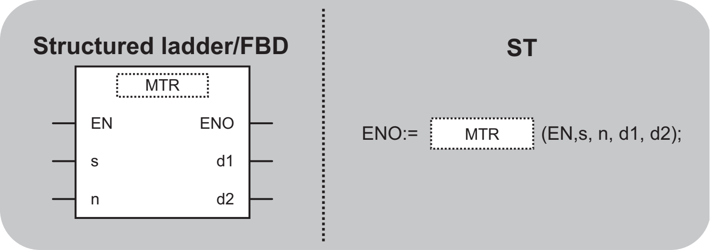
[ST]

```
PWM(X0,100,1000,Var_Y20);
```

Matrix input

MTR

~~Basic~~ High performance Process ~~Redundant~~ Universal LCPU



ENO:= MTR (EN,s, n, d1, d2);

The following instruction can go in the dotted squares.

MTR

■Executing condition

Instruction	Executing condition
MTR	_⎍_

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start device for data input	Bit
	n	Number of input rows	ANY16
Output argument	ENO	Execution result	Bit
	d1	Start device for data output	Bit
	d2	Start number of the device that stores matrix input data	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○(X only)	—							—
n	○	○							—
(d1)	○(Y only)	—							—
(d2)	○	—							—

Processing details

- It reads the input from 16 points × n rows from the input number specified for (s), then stores fetched input data to the device specified for (d2) and the following devices.
- One row (16 points) can be fetched in 1 scan.
- Fetching from the first to the nth row is repeated.
- The first through the 16th points store the first row of data and the next 16 points store the second row of data at the devices following the device specified for (d2). For this reason, the space of 16 × n points from the device specified for (d2) are occupied by the MTR instruction.
- (d1) is the output needed to select the row which will be fetched, and the system automatically turns it ON and OFF. It uses the n points from the device specified for (d1).
- Only device numbers divisible by 16 can be specified for (s), (d1) and (d2).
- For n, a value in the range from 2 to 8 can be assigned.
- No processing is performed in any of the following cases.
  - The device number specified for (s), (d1), or (d2) is not a multiple of 16.
  - The device specified for (s) is outside of the actual input range.
  - The device specified for (d1) is outside of the actual output range.
  - The space 16 × n points following the device specified for (d2) exceeds the corresponding device range.
  - The value for n is not between 2 and 8.

Operation error

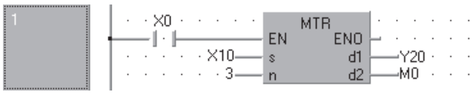
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device other than the input (X) was specified for (s). The device other than the output (Y) was specified for (d1).	—	○	○	—	○	○

Program example

- In the following program, the 16 points × 3 matrix from X10 and the following devices are fetched when X0 turns ON, and the matrix is stored to M0 and the following devices.

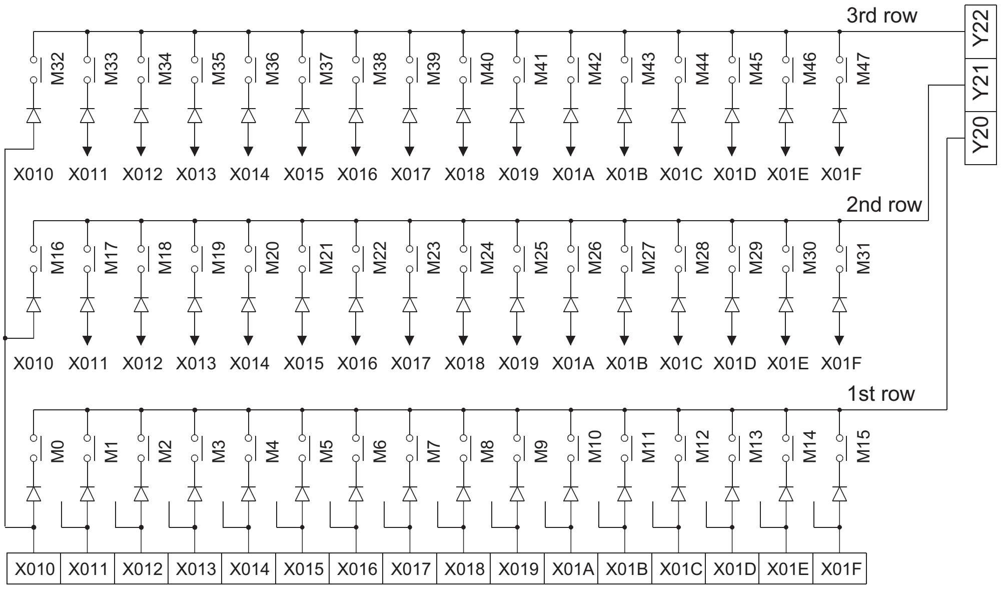
[Structured ladder/FBD]



[ST]

```
MTR(X0,X10,3,Y20,M0);
```

[Operation]


3rd row
Y22
Y21
Y20
M32
M33
M34
M35
M36
M37
M38
M39
M40
M41
M42
M43
M44
M45
M46
M47
X010 X011 X012 X013 X014 X015 X016 X017 X018 X019 X01A X01B X01C X01D X01E X01F
2nd row
M16
M17
M18
M19
M20
M21
M22
M23
M24
M25
M26
M27
M28
M29
M30
M31
X010 X011 X012 X013 X014 X015 X016 X017 X018 X019 X01A X01B X01C X01D X01E X01F
1st row
M0
M1
M2
M3
M4
M5
M6
M7
M8
M9
M10
M11
M12
M13
M14
M15
X010 X011 X012 X013 X014 X015 X016 X017 X018 X019 X01A X01B X01C X01D X01E X01F

Precautions

- Note that the MTR instruction directly operates on actual input and output. The output (d1) that had been turned ON by the MTR instruction does not turn OFF when the MTR command turns OFF. Turn OFF the specified output (d1) in the sequence program.
- The MTR instruction execution interval must be longer than the total of response time of input and output modules. If the set interval is shorter than the value indicated above, an input cannot be read correctly. If the scan time in a sequence program is short, select the constant scan and set the scan time longer than the total of response time.
- When using the MTR instruction, do not branch a line from (d2). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

7 APPLICATION INSTRUCTIONS

7.1 Logical Operation Instructions

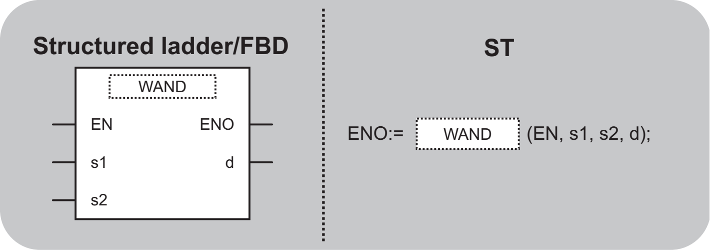
The logical operation instructions perform logical OR, logical AND or other logical operations in unit of 1 bit.

Operation	Description	Formula for Operation	Example		
			A	B	Y
Logical AND (AND)	A bit is set to 1 only when both input A and input B are 1, otherwise set to 0	Y=A/B	0	0	0
			0	1	0
			1	0	0
			1	1	1
Logical OR (OR)	A bit is set to 0 only when both input A and input B are 0, otherwise set to 1	Y=A+B	0	0	0
			0	1	1
			1	0	1
			1	1	1
Exclusive OR (XOR)	A bit is set to 0 when input A and input B are equal, otherwise set to 1	$Y=\overline{A}B+A\overline{B}$	0	0	0
			0	1	1
			1	0	1
			1	1	0
Exclusive NOR (XNR)	A bit is set to 1 when input A and input B are equal, otherwise set to 0	$Y=(\overline{A}+B)(A+\overline{B})$	0	0	1
			0	1	0
			1	0	0
			1	1	1

Logical AND operation on 16-/32-bit data

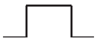
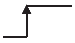
WAND(P), DAND(P)

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.
WAND, WANDP, DAND, DANDP

■Executing condition

Instruction	Executing condition
WAND, DAND	
WANDP, DANDP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1, s2	Data on which the logical AND operation is performed, or start number of the device that stores data	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores operation result	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(d)	○							—	—

Processing details

■WAND(P)

- Performs a logical AND operation on each bit of the 16-bit data of the device specified for (s1) and the 16-bit data of the device specified for (s2), and the results are stored to the device specified for (d).

	b15							b8	b7							b0
s1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1

AND

	b15							b8	b7							b0
s2	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0

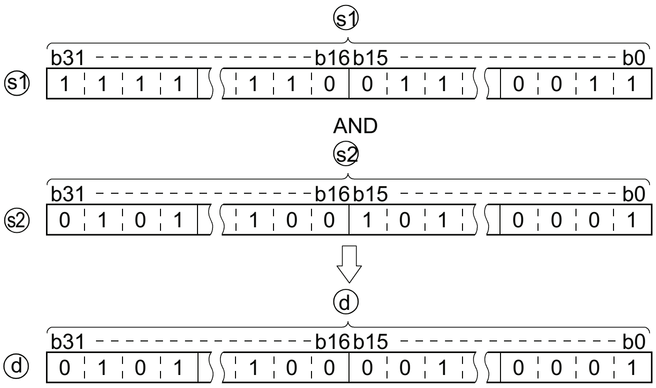
⇩

	b15							b8	b7							b0
d	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0

- For bit devices, the operation is performed as 0s are stored to the bit devices which follow the digit-specified bit devices. (Refer to Program Example)

■DAND(P)

- Performs a logical AND operation on each bit of the 32-bit data for the device specified for (s1) and the 32-bit data for the device specified for (s2), and stores the results to the device specified for (d).



s1
b31 b16 b15 b0
s1 1 1 1 1 … 1 1 0 0 1 1 … 0 0 1 1
AND
s2
b31 b16 b15 b0
s2 0 1 0 1 … 1 0 0 1 0 1 … 0 0 0 1
d
b31 b16 b15 b0
d 0 1 0 1 … 1 0 0 0 0 1 … 0 0 0 1

- For bit devices, the operation is performed as 0s are stored to the bit devices which follow the digit-specified bit devices. (Refer to Program Example)

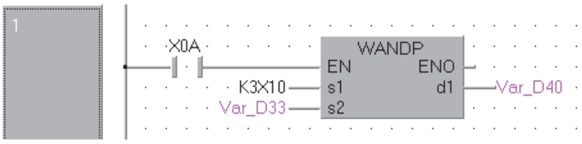
Operation error

- There is no operation error.

Program example

- In the following program, the logical AND operation is performed on the data from X10 to X1B and the data in Var_D33 when X0A turns ON, and the results are stored to Var_D40.

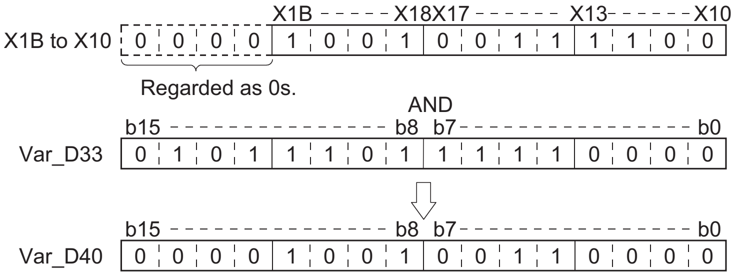
[Structured ladder/FBD]



[ST]

WANDP(X0A,K3X10,Var_D33,Var_D40);

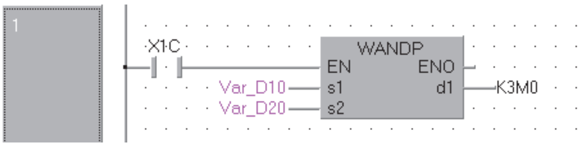
[Operation]



X1B X18 X17 X13 X10
X1B to X10 0 0 0 0 1 0 0 1 0 0 1 1 1 1 0 0
Regarded as 0s.
AND
b15 b8 b7 b0
Var_D33 0 1 0 1 1 1 0 1 1 1 1 1 0 0 0 0
b15 b8 b7 b0
Var_D40 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0

- In the following program, the logical AND operation is performed on the data in Var_D10 and Var_D20 when X1C turns ON, and the results are stored to the devices from M0 to M11.

[Structured ladder/FBD]



[ST]

```
WANDP(X1C,Var_D10,Var_D20,K3M0);
```

[Operation]

	b15 - - -	- - - b8	b7 - - -	- - - b0
Var_D10	0 0 0 0	1 0 1 0	1 0 1 0	1 0 1 0

AND

	b15 - - -	- - - b8	b7 - - -	- - - b0
Var_D20	0 0 0 1	1 1 1 1	1 1 1 1	1 1 1 1

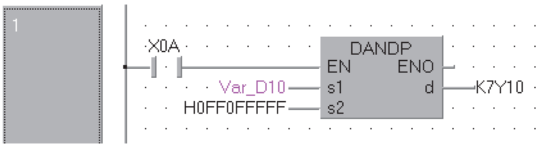
⇩

M15 - - - M12	M11 - - - M8	M7 - - - M4	M3 - - - M0
1 1 1 1	1 0 1 0	1 0 1 0	1 0 1 0

Not changed

- In the following program, the digit in the hundred-thousands place of the 8-digit BCD value in Var_D10 (sixth digit from the end) is masked to 0 when X0A turns ON, and the results are stored to the devices from Y10 to Y2B.

[Structured ladder/FBD]



[ST]

```
DANDP(X0A,Var_D10,H0FF0FFFFF,K7Y10);
```

[Operation]

	b31 - - -			- - - b16	b15 - - -			- - - b0
Var_D10(BCD12345678)	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0

AND

	b31 - - -			- - - b16	b15 - - -			- - - b0
H FF0FFFFF	1 1 1 1	1 1 1 1	0 0 0 0	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1

⇩

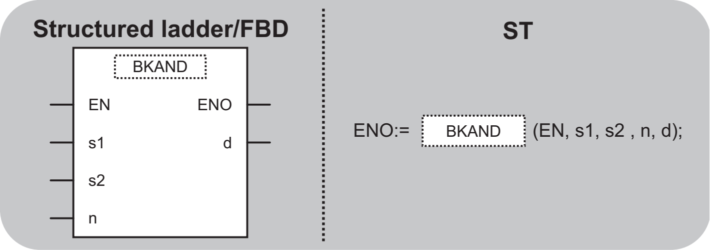
	Y2F - Y2C	Y2B - Y28	Y27 - Y24	Y23 - Y20	Y1F - Y1C	Y1B - Y18	Y17 - Y14	Y13 - Y10
Y2B to Y10	1 1 1 0	0 0 1 0	0 0 0 0	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0

Not changed

Logical AND operation on block data

BKAND(P)

Basic | High performance | Process | Redundant | Universal | LCPU

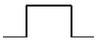
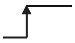


```
ENO:= BKAND (EN, s1, s2 , n, d);
```

Any of the following instruction can go in the dotted squares.

BKAND, BKANDP

■Executing condition

Instruction	Executing condition
BKAND	
BKANDP	

■Argument

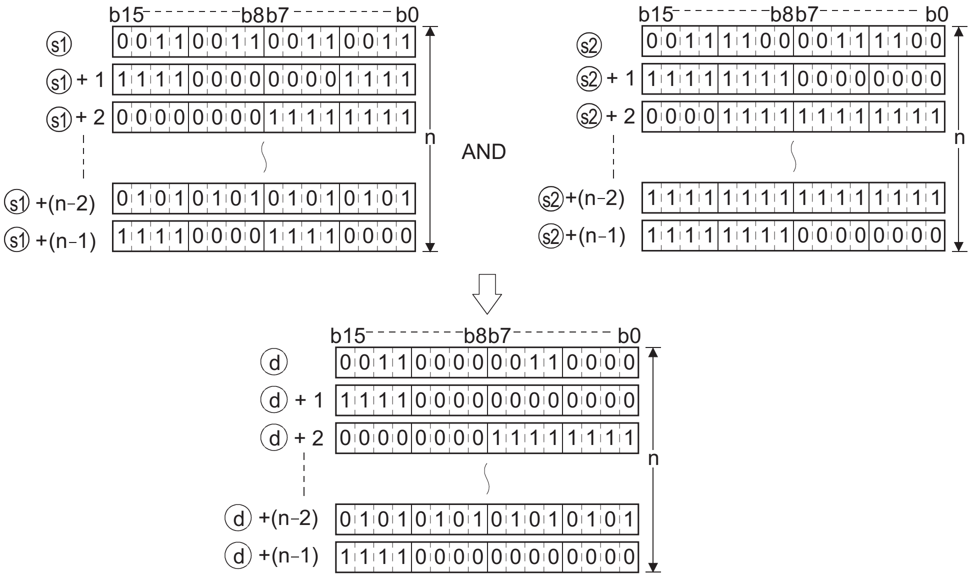
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores data on which the logical AND operation is performed	ANY16
	s2	Logical AND operation data, or start number of the device that stores data	ANY16
	n	Number of operation data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)*1	—	○		—				—	—
(s2)*1	—	○		—				○	—
n	○	○		○				○	—
(d)*1	—	○		—				—	—

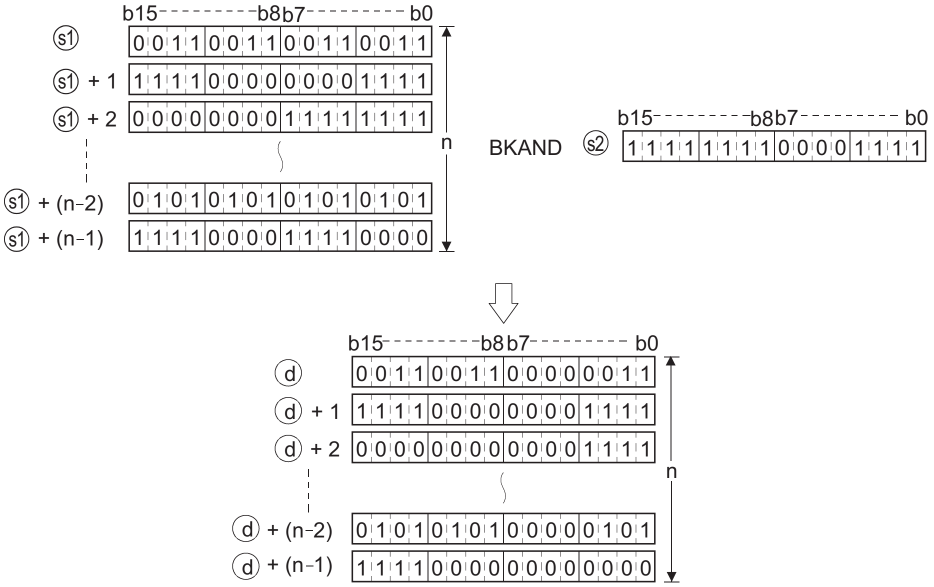
*1 Same device numbers can be specified for (s1) and (d), or (s2) and (d).

Processing details

- Performs a logical AND operation on the data n points from the device specified for (s1), and the data n points from the device specified for (s2), and stores the results to the device specified for (d) and the following devices.


b15 b8 b7 b0
s1 0011 0011 0011 0011
s1 + 1 1111 0000 0000 1111
s1 + 2 0000 0000 1111 1111
s1 +(n-2) 0101 0101 0101 0101
s1 +(n-1) 1111 0000 1111 0000
n
AND
s2 0011 1100 0011 1100
s2 + 1 1111 1111 0000 0000
s2 + 2 0000 1111 1111 1111
s2 +(n-2) 1111 1111 1111 1111
s2 +(n-1) 1111 1111 0000 0000
d 0011 0000 0011 0000
d + 1 1111 0000 0000 0000
d + 2 0000 0000 1111 1111
d +(n-2) 0101 0101 0101 0101
d +(n-1) 1111 0000 0000 0000

- The constant can be specified for (s2) between -32768 and 32767 (16-bit BIN data).


b15 b8 b7 b0
s1 0011 0011 0011 0011
s1 + 1 1111 0000 0000 1111
s1 + 2 0000 0000 1111 1111
s1 + (n-2) 0101 0101 0101 0101
s1 + (n-1) 1111 0000 1111 0000
n
BKAND
s2 1111 1111 0000 1111
d 0011 0011 0000 0011
d + 1 1111 0000 0000 1111
d + 2 0000 0000 0000 1111
d + (n-2) 0101 0101 0000 0101
d + (n-1) 1111 0000 0000 0000

Operation error

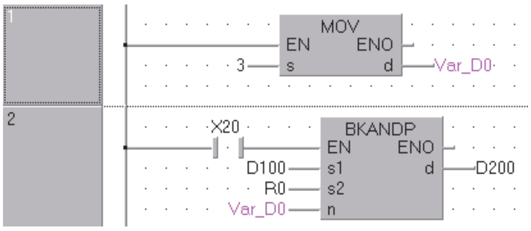
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (s1), (s2) or (d) exceeds the corresponding device. The device range of n points from the device specified for (s1) overlaps with the device range of n points from the device specified for (d). (Except when the same device is specified for (s1) and (d)) The device range of n points from the device specified for (s2) overlaps with the device range of n points from the device specified for (d). (Except when the same device is specified for (s2) and (d))	○	○	○	○	○	○

Program example

- In the following program, the logical AND operation is performed on the data in the devices from D100 to D102 and the data in the devices from R0 to R2 when X20 turns ON, and the results are stored to D200 and the following devices.

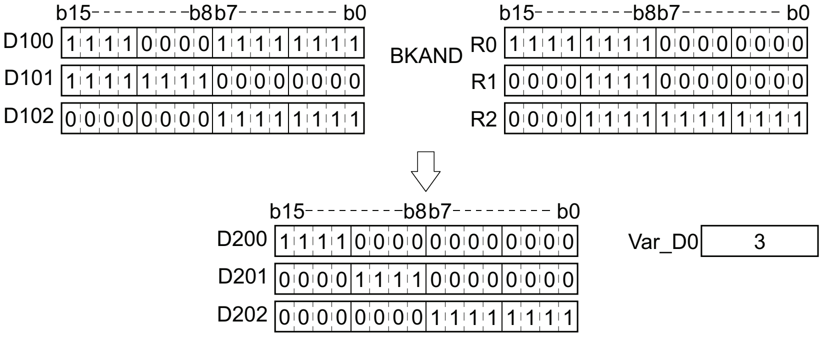
[Structured ladder/FBD]



[ST]

Var_D0:=K3
BKANDP(X20,D100,R0,Var_D0,D200);

[Operation]



	b15 - - - - - - - - b8	b7 - - - - - - - - b0			b15 - - - - - - - - b8	b7 - - - - - - - - b0
D100	1 1 1 1 0 0 0 0	1 1 1 1 1 1 1 1	BKAND	R0	1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0
D101	1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0		R1	0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 0
D102	0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1		R2	0 0 0 0 1 1 1 1	1 1 1 1 1 1 1 1

⇩

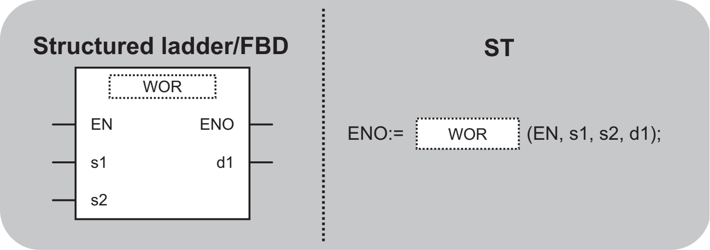
	b15 - - - - - - - - b8	b7 - - - - - - - - b0
D200	1 1 1 1 0 0 0 0	0 0 0 0 0 0 0 0
D201	0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 0
D202	0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1

Var_D0 3

Logical OR operation on 16-/32-bit data

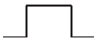
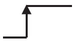
WOR(P), DOR(P)

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.
WOR, WORP, DOR, DORP

■Executing condition

Instruction	Executing condition
WOR, DOR	
WORP, DORP	

■Argument

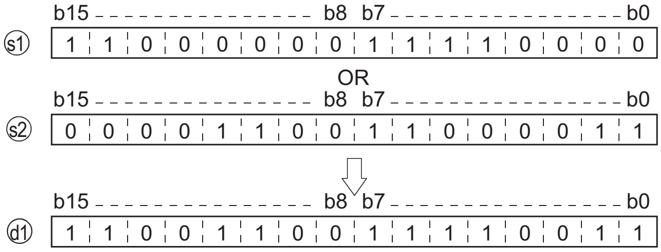
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1, s2:	Data on which the logical OR operation is performed, or start number of the device that stores data	ANY16/32
Output argument	ENO	Execution result	Bit
	d1	Start number of the device that stores operation result	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\□G	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(d1)	○							—	—

Processing details

■WOR (P)

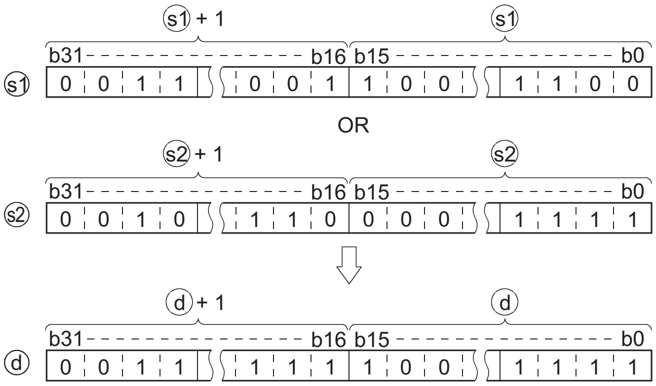
- Performs a logical OR operation on each bit of the 16-bit data of the device specified for (s1) and the 16-bit data of the device specified for (s2), and stores the results to the device specified for (d1).


b15 b8 b7 b0
(s1) 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0
OR
b15 b8 b7 b0
(s2) 0 0 0 0 1 1 0 0 1 1 0 0 0 0 1 1
b15 b8 b7 b0
(d1) 1 1 0 0 1 1 0 0 1 1 1 1 0 0 1 1

- For bit devices, the operation is performed as 0s are stored to the bit devices which follow the digit-specified bit devices. (Refer to Program Example)

■DOR(P)

- Performs a logical OR operation on each bit of the 32-bit data of the device specified for (s1) and the 32-bit data of the device specified for (s2), and stores the results to the device specified for (d).


(s1) + 1
(s1)
b31 b16 b15 b0
(s1) 0 0 1 1 0 0 1 1 0 0 1 1 0 0
OR
(s2) + 1
(s2)
b31 b16 b15 b0
(s2) 0 0 1 0 1 1 0 0 0 0 1 1 1 1
(d) + 1
(d)
b31 b16 b15 b0
(d) 0 0 1 1 1 1 1 1 0 0 1 1 1 1

- For bit devices, the operation is performed as 0s are stored to the bit devices which follow the digit-specified bit devices. (Refer to Program Example)

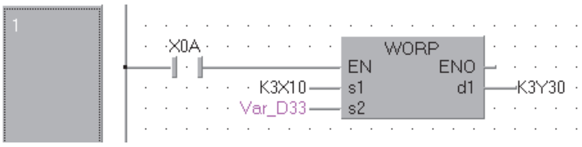
Operation error

- There is no operation error.

Program example

- In the following program, the logical OR operation is performed on the data from X10 to X1B, and the data in Var_D33 when X0A turns ON, and the results are output to Y30 to Y3B.

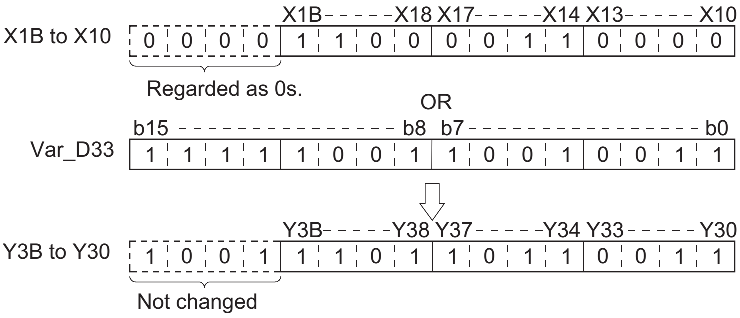
[Structured ladder/FBD]



[ST]

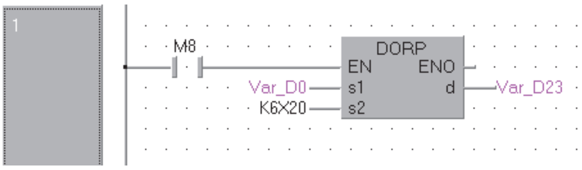
WORP(X0A,K3X10,Var_D33,K3Y30);

[Operation]


X1B to X10
X1B - - - - -X18 X17- - - - -X14 X13- - - - - X10
0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0
Regarded as 0s.
OR
Var_D33
b15 - - - - - - - - - - - - - - - b8 b7 - - - - - - - - - - - - - - - - b0
1 1 1 1 1 0 0 1 1 0 0 1 0 0 1 1
Y3B to Y30
Y3B- - - - -Y38 Y37- - - - -Y34 Y33- - - - -Y30
1 0 0 1 1 1 0 1 1 0 1 1 0 0 1 1
Not changed

- In the following program, the logical OR operation is performed on the 32-bit data in Var_D0, and the 24-bit data from X20 to X37 when M8 turns ON, and the results are stored to Var_D23.

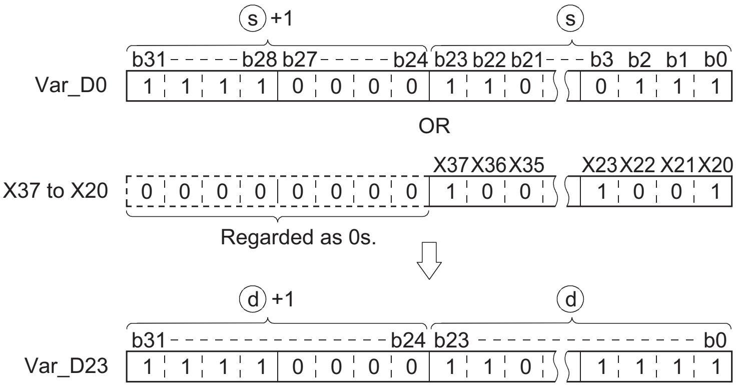
[Structured ladder/FBD]



[ST]

DORP(M8,Var_D0,K6X20,Var_D23);

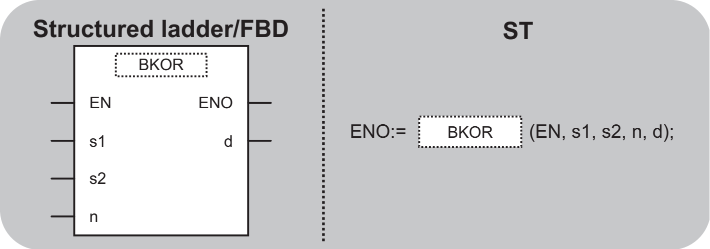
[Operation]


s +1
s
Var_D0
b31- - - - - b28 b27- - - - - b24 b23 b22 b21- - - b3 b2 b1 b0
1 1 1 1 0 0 0 0 1 1 0 0 1 1 1
OR
X37 to X20
X37 X36 X35 X23 X22 X21 X20
0 0 0 0 0 0 0 0 1 0 0 1 0 0 1
Regarded as 0s.
d +1
d
Var_D23
b31- - - - - - - - - - - - - - - - b24 b23 - - - - - - - - - - - - - - - b0
1 1 1 1 0 0 0 0 1 1 0 1 1 1 1

Logical OR operation on block data

BKOR(P)

Basic | High performance | Process | Redundant | Universal | LCPU

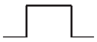
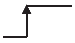


```
ENO:= BKOR (EN, s1, s2, n, d);
```

Any of the following instruction can go in the dotted squares.

BKOR, BKORP

■Executing condition

Instruction	Executing condition
BKOR	
BKORP	

■Argument

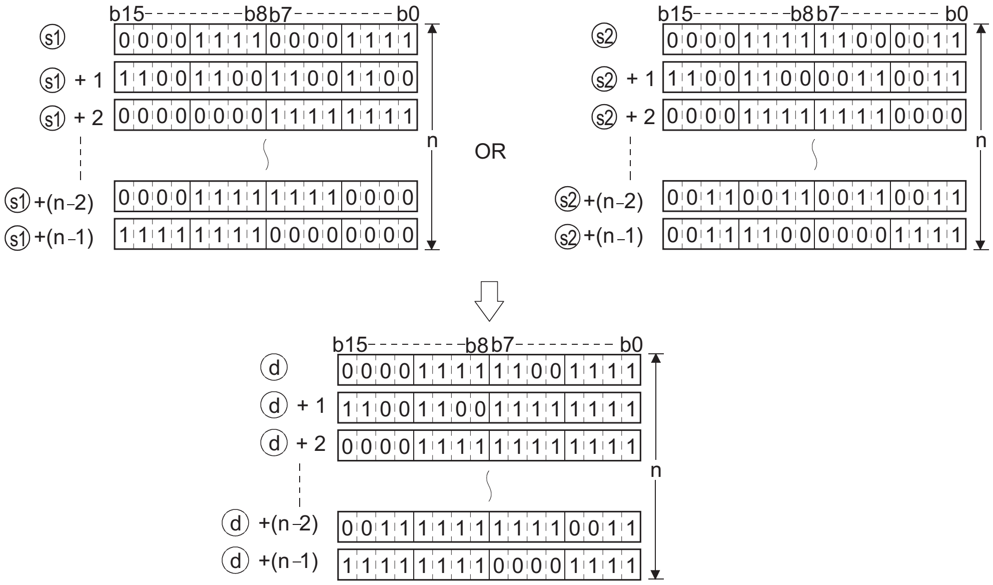
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores data on which the logical OR operation is performed	ANY16
	s2	Logical OR operation data, or start number of the device that stores data	ANY16
	n	Number of operation data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the Logical OR operation result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1) [*1]	—	○		—				—	—
(s2) [*1]	—	○		—				○	—
n	○	○		○				○	—
(d) [*1]	—	○		—				—	—

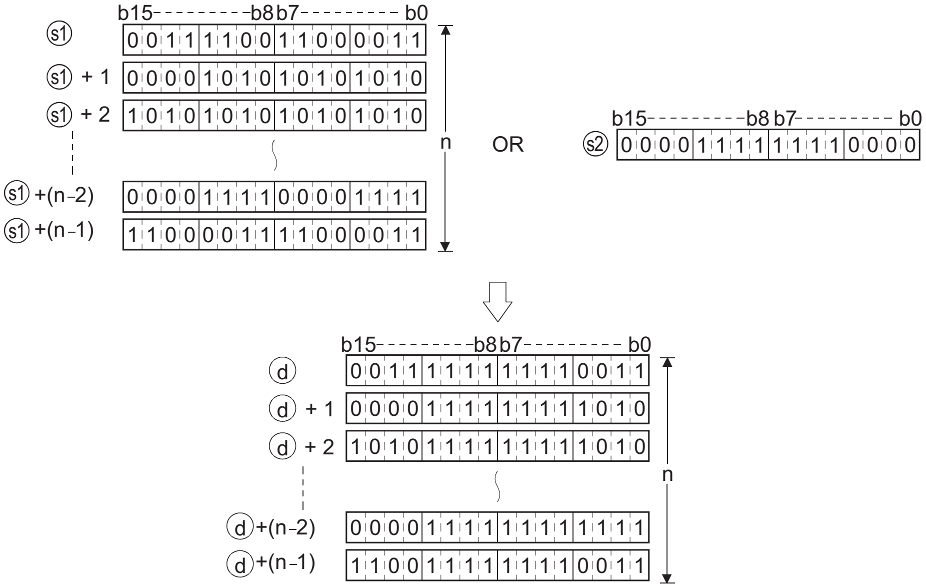
[*1] Same device numbers can be specified for (s1) and (d), or (s2) and (d).

Processing details

- Performs a logical OR operation on the data n points from the device specified for (s1), and the data n points from the device specified for (s2), and stores the results to the device specified for (d) and the following devices.


b15 b8 b7 b0
s1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
s1 + 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
s1 + 2 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
s1 +(n-2) 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
s1 +(n-1) 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
n
OR
s2 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1
s2 + 1 1 1 0 0 1 1 0 0 0 0 1 1 0 0 1 1
s2 + 2 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
s2 +(n-2) 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
s2 +(n-1) 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1
d 0 0 0 0 1 1 1 1 1 1 0 0 1 1 1 1
d + 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1
d + 2 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
d +(n-2) 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1
d +(n-1) 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1

- The constant can be specified for (s2) between -32768 and 32767 (16-bit BIN data).


b15 b8 b7 b0
s1 0 0 1 1 1 1 0 0 1 1 0 0 0 0 1 1
s1 + 1 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 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 1 1 1 1 0 0 0 0 1 1 1 1
s1 +(n-1) 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1
n
OR
s2 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
d 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1
d + 1 0 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0
d + 2 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0
d +(n-2) 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
d +(n-1) 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1

Operation error

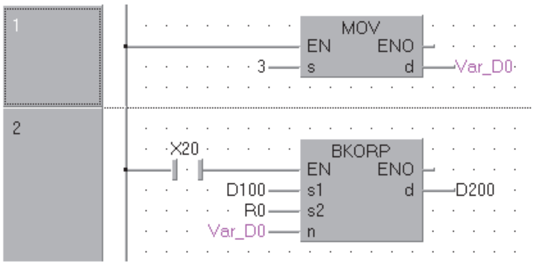
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (s1), (s2) or (d) exceeds the corresponding device. The device range of n points from the device specified for (s1) overlaps with the device range of n points from the device specified for (d). (Except when the same device is specified for (s1) and (d)) The device range of n points from the device specified for (s2) overlaps with the device range of n points from the device specified for (d). (Except when the same device is specified for (s2) and (d))	○	○	○	○	○	○

Program example

- In the following program, the logical OR operation is performed on the data in the devices from D100 to D102 and the data in the devices from R0 to R2 when X20 turns ON, and the results are stored to D200 and the following devices.

[Structured ladder/FBD]



[ST]

```
Var_D0:=K3;
BKORP(X20,D100,R0,Var_D0,D200);
```

[Operation]

	b15							b8	b7							b0
D100	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0
D101	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
D102	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0

OR

	b15							b8	b7							b0
R0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
R1	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
R2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1

⇩

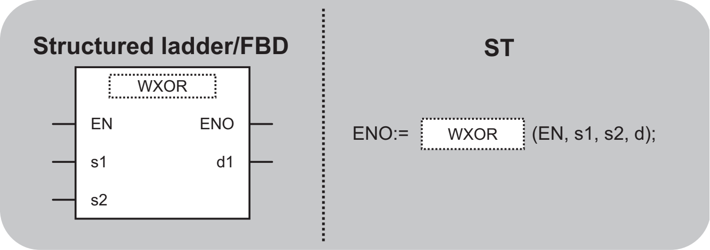
	b15							b8	b7							b0
D200	0	0	1	1	1	0	1	0	1	0	1	0	1	0	1	0
D201	0	0	1	1	1	1	1	1	1	1	1	1	0	0	1	1
D202	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1	1

Var_D0	3

Exclusive OR operation on 16-/32-bit data

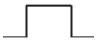
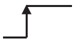
WXOR(P), DXOR(P)

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.
WXOR, WXORP, DXOR, DXORP

■Executing condition

Instruction	Executing condition
WXOR, DXOR	
WXORP, DXORP	

■Argument

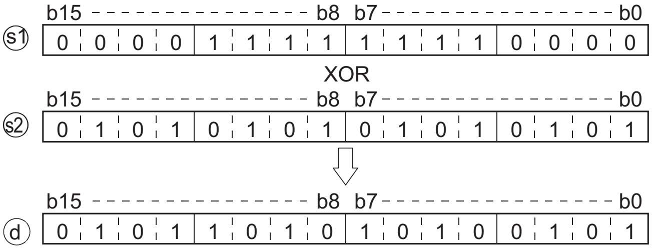
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1, s2:	Data on which the exclusive OR operation is performed, or start number of the device that stores data	ANY16/32
Output argument	ENO	Execution result	Bit
	d1	Start number of the device that stores exclusive OR operation result	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(d1)	○							—	—

Processing details

■WXOR(P)

- Performs an exclusive OR operation on each bit of the 32-bit data of the device specified for (s1) and the 32-bit data of the device specified for (s2), and stores the results to the device specified for (d).



b15 - - - b8 b7 - - - b0
(s1) 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0

XOR

b15 - - - b8 b7 - - - b0
(s2) 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

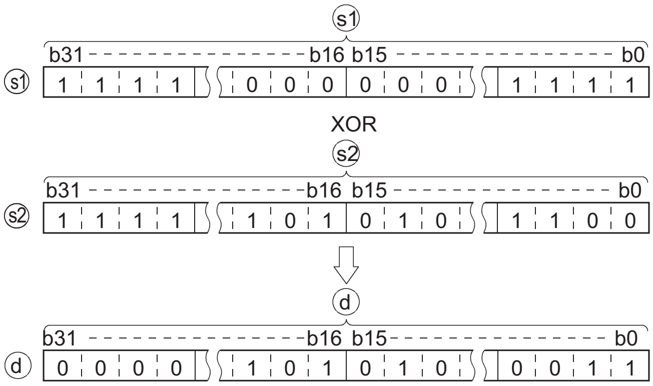
⇩

b15 - - - b8 b7 - - - b0
(d) 0 1 0 1 1 0 1 0 1 0 1 0 0 1 0 1

- For bit devices, the operation is performed as 0s are stored to the bit devices which follow the digit-specified bit devices. (Refer to Program Example)

■DXOR(P)

- Performs an exclusive OR operation on each bit of the 32-bit data of the device specified for (s1) and the 32-bit data of the device specified for (s2), and stores the results to the device specified for (d1).



(s1)
b31 b16 b15 b0
(s1) 1 1 1 1 0 0 0 0 0 0 1 1 1 1
XOR
(s2)
b31 b16 b15 b0
(s2) 1 1 1 1 1 0 1 0 1 0 1 1 0 0
(d)
b31 b16 b15 b0
(d) 0 0 0 0 1 0 1 0 1 0 0 0 1 1

- For bit devices, the operation is performed as 0s are stored to the bit devices which follow the digit-specified bit devices.

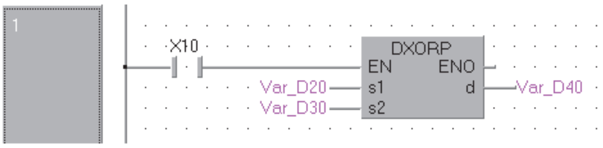
Operation error

- There is no operation error.

Program example

- In the following program, the exclusive OR operation is performed on the data from X10 to X1B and the data in Var_D33 when X10 turns ON, and the results are output to Y30 to Y3B.

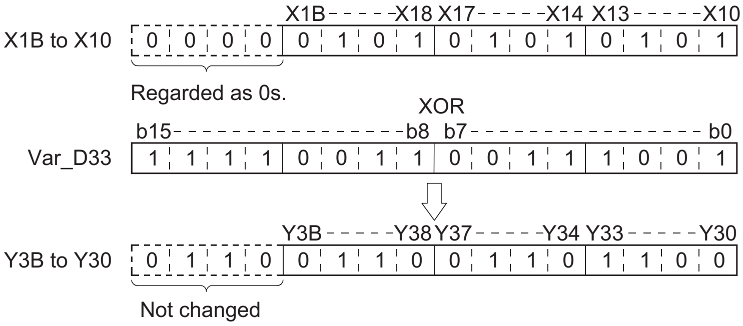
[Structured ladder/FBD]



[ST]

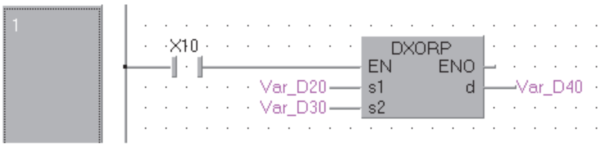
WXORP(X10,K3X10,Var_D33,K3Y30);

[Operation]



- In the following program, the exclusive OR operation is performed on the data in Var_D20 and the data in Var_D30 when X10 turns ON, and the results are stored to Var_D40.

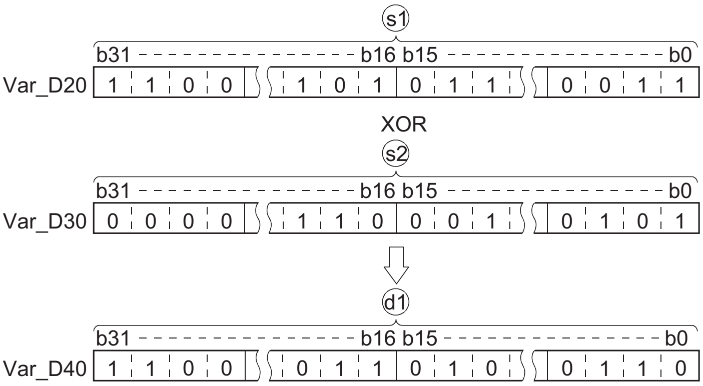
[Structured ladder/FBD]



[ST]

```
DXORP(X10,Var_D20,Var_D30,Var_D40);
```

[Operation]

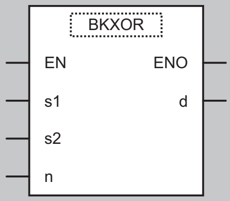

s1
b31 b16 b15 b0
Var_D20 1 1 0 0 1 0 1 0 1 1 0 0 1 1
XOR
s2
b31 b16 b15 b0
Var_D30 0 0 0 0 1 1 0 0 0 1 0 1 0 1
d1
b31 b16 b15 b0
Var_D40 1 1 0 0 0 1 1 0 1 0 0 1 1 0

Exclusive OR operation on block data

BKXOR(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



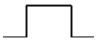
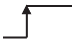
ST

ENO:= BKXOR (EN, s1, s2, n, d);

Any of the following instruction can go in the dotted squares.

BKXOR, BKXORP

■Executing condition

Instruction	Executing condition
BKXOR	
BKXORP	

■Argument

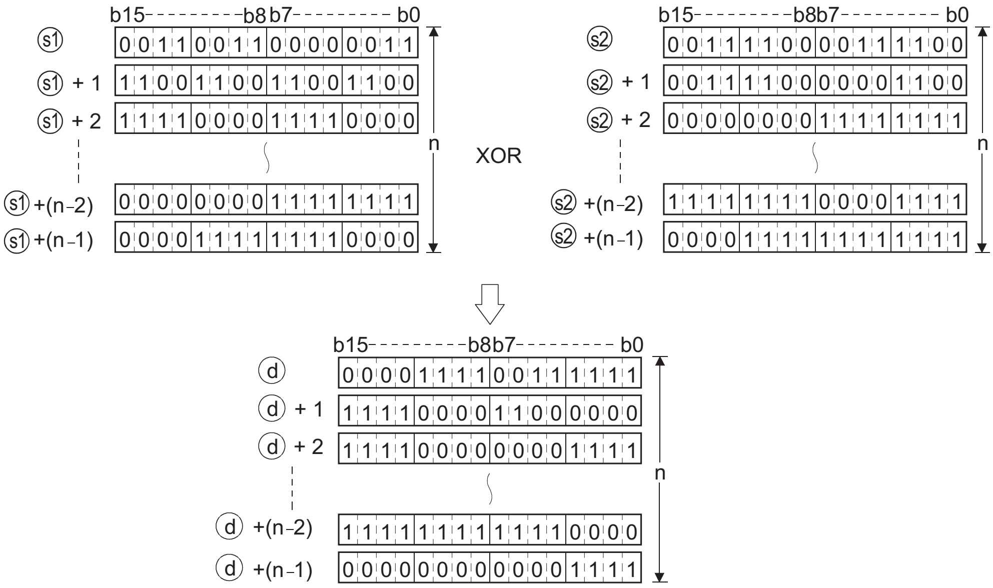
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores data on which the exclusive OR operation is performed	ANY16
	s2	Exclusive OR operation data, or start number of the device that stores data	ANY16
	n	Number of operation data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1) [*1]	—	○		—				—	—
(s2) [*1]	—	○		—				○	—
n	○	○		○				○	—
(d) [*1]	—	○		—				—	—

[*1] Same device numbers can be specified for (s1) and (d), or (s2) and (d).

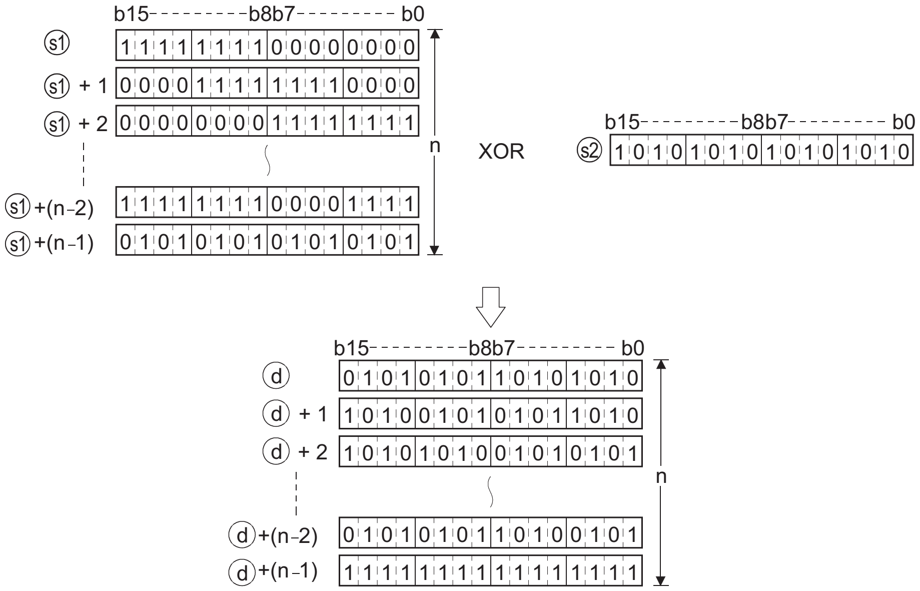
Processing details

- Performs an exclusive OR operation on the data n points from the device specified for (s1), and the data n points from the device specified for (s2), and stores the results to the device specified for (d) and the following devices.



b15 - - - - - - - - - b8 b7 - - - - - - - - - b0
s1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 1 1
s1 + 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
s1 + 2 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
s1 +(n-2) 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
s1 +(n-1) 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
n
XOR
b15 - - - - - - - - - b8b7 - - - - - - - - - b0
s2 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0
s2 + 1 0 0 1 1 1 1 0 0 0 0 0 0 1 1 0 0
s2 + 2 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
s2 +(n-2) 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1
s2 +(n-1) 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
n
b15 - - - - - - - - - b8b7 - - - - - - - - - b0
d 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 1
d + 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0
d + 2 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1
d +(n-2) 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
d +(n-1) 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
n

- The constant can be specified for (s2) between -32768 and 32767 (16-bit BIN data).



b15 - - - - - - - - - b8b7 - - - - - - - - - b0
s1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
s1 + 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
s1 + 2 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
s1 +(n-2) 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1
s1 +(n-1) 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
n
XOR
b15 - - - - - - - - - b8b7 - - - - - - - - - b0
s2 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
b15 - - - - - - - - - b8b7 - - - - - - - - - b0
d 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0
d + 1 1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0
d + 2 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1
d +(n-2) 0 1 0 1 0 1 0 1 1 0 1 0 0 1 0 1
d +(n-1) 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
n

Operation error

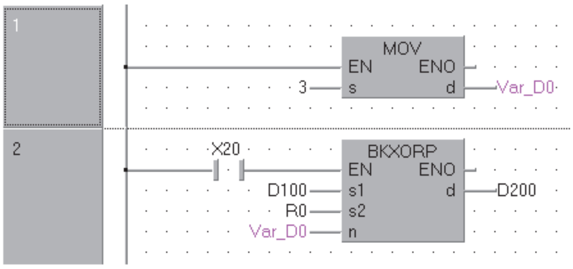
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (s1), (s2) or (d) exceeds the corresponding device. The device range of n points from the device specified for (s1) overlaps with the device range of n points from the device specified for (d). (Except when the same device is specified for (s1) and (d)) The device range of n points from the device specified for (s2) overlaps with the device range of n points from the device specified for (d). (Except when the same device is specified for (s2) and (d))	○	○	○	○	○	○

Program example

- In the following program, the exclusive OR operation is performed on the data in the devices from D100 to D102 and the data in the devices from R0 to R2 when X20 turns ON, and the results are stored to D200 and the following devices.

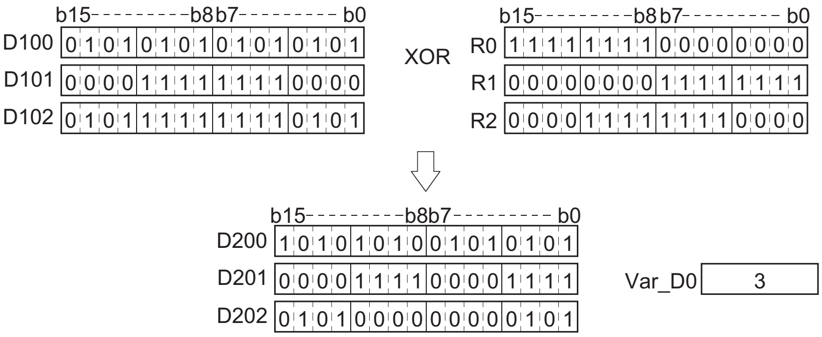
[Structured ladder/FBD]



[ST]

```
Var_D0:=3;
BKXORP(X20,D100,R0,Var_D0,D200);
```

[Operation]



	b15							b8	b7							b0
D100	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
D101	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
D102	0	1	0	1	1	1	1	1	1	1	1	1	0	1	0	1

XOR

	b15							b8	b7							b0
R0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R2	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0

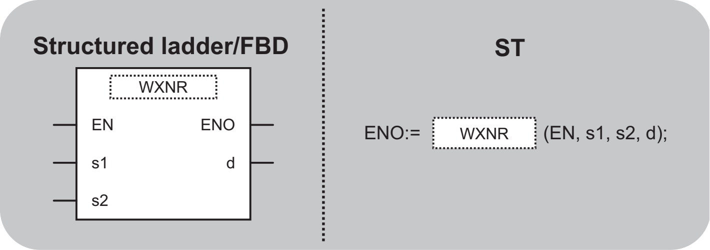
	b15							b8	b7							b0
D200	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1
D201	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
D202	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1

Var_D0	3

Exclusive NOR operation on 16-/32-bit data

WXNR(P), DXNR(P)

Basic | High performance | Process | Redundant | Universal | LCPU

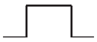
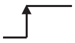


```
ENO:= WXNR (EN, s1, s2, d);
```

Any of the following instruction can go in the dotted squares.

WXNR, WXNRP, DXNR, DXNRP

■Executing condition

Instruction	Executing condition
WXNR, DXNR	
WXNRP, DXNRP	

■Argument

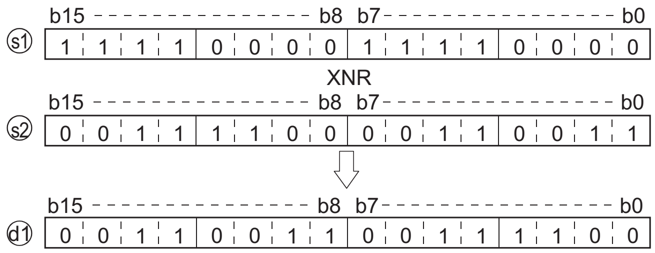
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1, s2	Data on which the exclusive NOR operation is performed, or start number of the device that stores data	ANY16/32
Output argument	ENO	Execution result	Bit
	d1	Start number of the device that stores exclusive NOR operation result	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(d1)	○							—	—

Processing details

■WXNR(P)

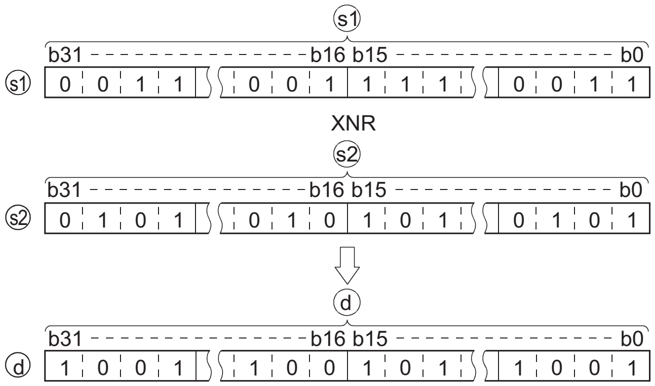
- Performs an exclusive NOR operation on the 16-bit data of the device specified for (s1)and the 16-bit data of the device specified for (s2), and stores the results to the device specified for (d1).


b15 b8 b7 b0
(s1) 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
XNR
b15 b8 b7 b0
(s2) 0 0 1 1 1 1 0 0 0 0 1 1 0 0 1 1
b15 b8 b7 b0
(d1) 0 0 1 1 0 0 1 1 0 0 1 1 1 1 0 0

- For bit devices, the operation is performed as 0s are stored to the bit devices which follow the digit-specified bit devices.

■DXNR(P)

- Performs an exclusive NOR operation on the 32-bit data of the device specified for (s1) and the 32-bit data of the device specified for (s2), and stores the results to the device specified for (d).


(s1)
b31 b16 b15 b0
(s1) 0 0 1 1 0 0 1 1 1 1 0 0 1 1
XNR
(s2)
b31 b16 b15 b0
(s2) 0 1 0 1 0 1 0 1 0 1 0 1 0 1
(d)
b31 b16 b15 b0
(d) 1 0 0 1 1 0 0 1 0 1 1 0 0 1

- For bit devices, the operation is performed as 0s are stored to the bit devices which follow the digit-specified bit devices.

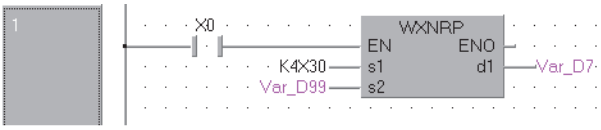
Operation error

- There is no operation error.

Program example

- In the following program, the exclusive NOR operation is performed on the 16-bit data from X30 to X3F and the data in Var_D99 when X0 turns ON, and the results are stored to Var_D7.

[Structured ladder/FBD]



[ST]

WXNRP(X0,K4X30,Var_D99,Var_D7);

[Operation]

	X3F			X3C	X3B			X38	X37			X34	X33			X30
X3F ~X30	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

XNR

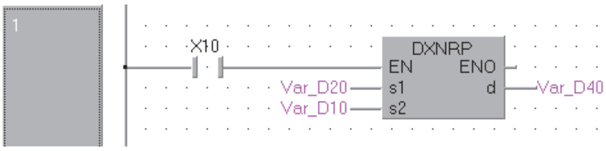
	b15							b8	b7							b0
Var_D99	1	1	0	0	1	0	0	1	1	1	1	1	1	1	0	0

⇩

	b15							b8	b7							b0
Var_D7	0	1	1	0	0	0	1	1	0	1	0	1	0	1	1	0

- In the following program, the exclusive NOR operation is performed on the 32-bit data in Var_D20 and the data in Var_D10 when X10 turns ON, and the results are stored to Var_D40.

[Structured ladder/FBD]



[ST]

DXNRP(X10,Var_D20,Var_D10,Var_D40);

[Operation]

s1

	b31						b16	b15						b0
Var_D20	0	1	0	1	1	0	1	0	1	0	0	1	0	1

XNR

s2

	b31						b16	b15						b0
Var_D10	0	1	1	0	0	1	0	1	1	0	1	1	0	0

⇩

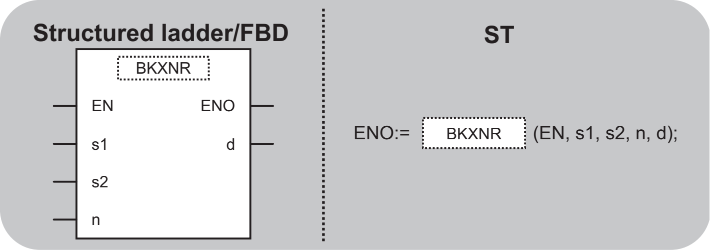
d

	b31						b16	b15						b0
Var_D40	1	1	0	0	0	0	0	0	1	1	0	1	1	0

Exclusive NOR operation on block data

BKXNR(P)

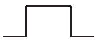
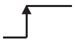
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

BKXNR, BKXNRP

■Executing condition

Instruction	Executing condition
BKXNR	
BKXNRP	

■Argument

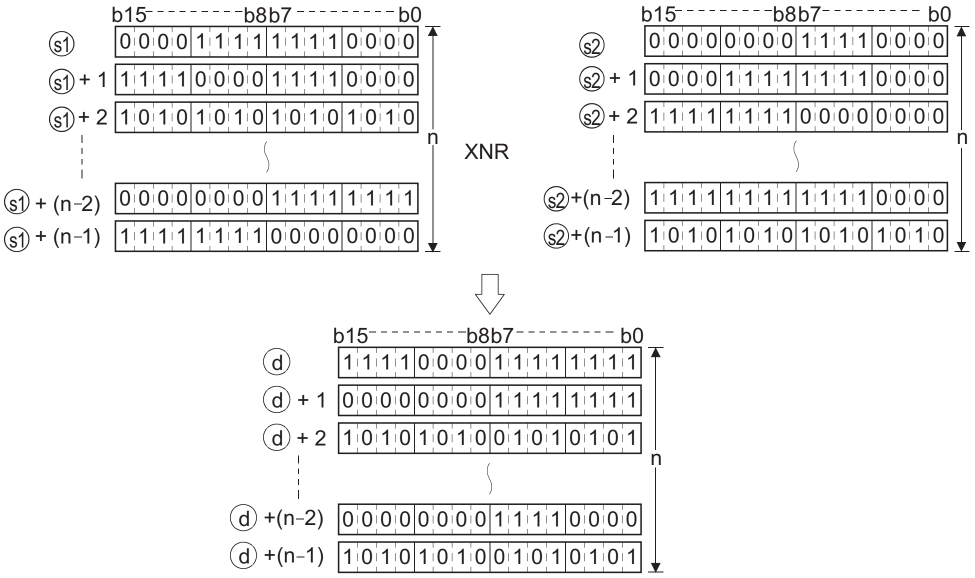
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores data on which the exclusive NOR operation is performed	ANY16
	s2	Exclusive NOR operation data, or start number of the device that stores data	ANY16
	n	Number of operation data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)*1	—	○		—				—	—
(s2)*1	—	○		—				○	—
n	○	○		○				○	—
(d)*1	—	○		—				—	—

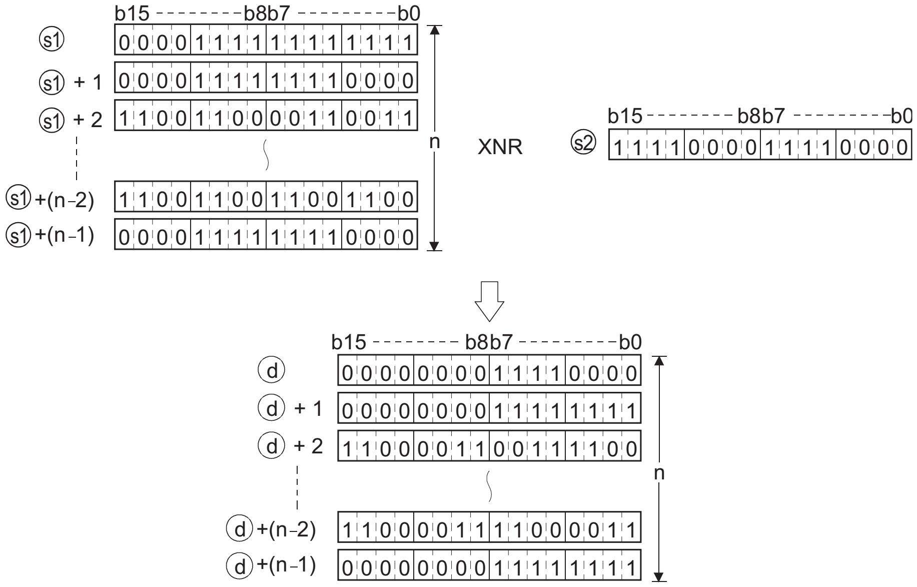
*1 Same device numbers can be specified for (s1) and (d), or (s2) and (d).

Processing details

- Performs an exclusive NOR operation on the data n points from the device specified for (s1), and the data n points from the device specified for (s2), and stores the results to the device specified for (d) and the following devices.


b15 b8b7 b0
s1 0000 1111 1111 0000
s1 + 1 1111 0000 1111 0000
s1 + 2 1010 1010 1010 1010
s1 + (n-2) 0000 0000 1111 1111
s1 + (n-1) 1111 1111 0000 0000
n
XNR
b15 b8b7 b0
s2 0000 0000 1111 0000
s2 + 1 0000 1111 1111 0000
s2 + 2 1111 1111 0000 0000
s2+(n-2) 1111 1111 1111 0000
s2+(n-1) 1010 1010 1010 1010
n
b15 b8b7 b0
d 1111 0000 1111 1111
d + 1 0000 0000 1111 1111
d + 2 1010 1010 0101 0101
d +(n-2) 0000 0000 1111 0000
d +(n-1) 1010 1010 0101 0101
n

- The constant can be specified for (s2) between -32768 and 32767 (16-bit BIN data).


b15 b8b7 b0
s1 0000 1111 1111 1111
s1 + 1 0000 1111 1111 0000
s1 + 2 1100 1100 0011 0011
s1+(n-2) 1100 1100 1100 1100
s1+(n-1) 0000 1111 1111 0000
n
XNR
b15 b8b7 b0
s2 1111 0000 1111 0000
b15 b8b7 b0
d 0000 0000 1111 0000
d + 1 0000 0000 1111 1111
d + 2 1100 0011 0011 1100
d+(n-2) 1100 0011 1100 0011
d+(n-1) 0000 0000 1111 1111
n

Operation error

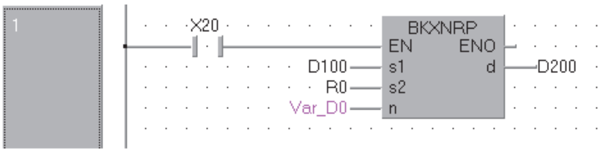
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (s1), (s2) or (d) exceeds the corresponding device. The device range of n points from the device specified for (s1) overlaps with the device range of n points from the device specified for (d). (Except when the same device is specified for (s1) and (d)) The device range of n points from the device specified for (s2) overlaps with the device range of n points from the device specified for (d). (Except when the same device is specified for (s2) and (d))	○	○	○	○	○	○

Program example

- In the following program, the exclusive NOR operation is executed on the data in the devices from D100 to D102 and the data in the devices from R0 to R2 when X20 turns ON, and the results are stored to D200 and the following devices.

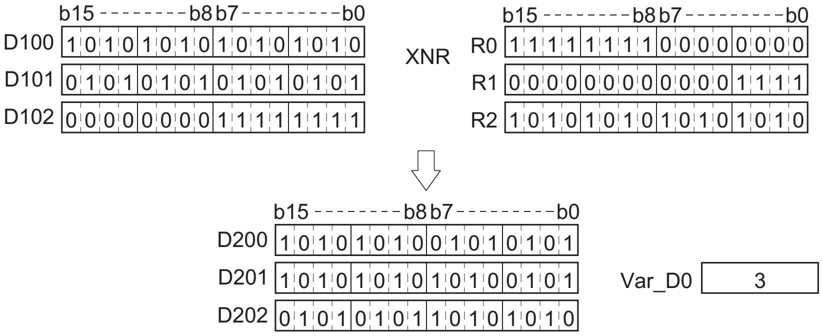
[Structured ladder/FBD]



[ST]

```
BKXNRP(X20,D100,R0,Var_D0,D200);
```

[Operation]



	b15							b8	b7							b0
D100	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
D101	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
D102	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

	b15							b8	b7							b0
R0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
R2	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

	b15							b8	b7							b0
D200	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1
D201	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	1
D202	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0

Var_D0	3

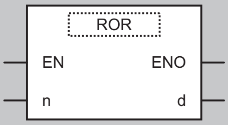
7.2 Rotation Instructions

Right rotation of 16-bit data

ROR(P), RCR(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



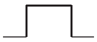
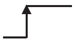
ST

ENO:= ROR (EN, n, d);

Any of the following instruction can go in the dotted squares.

ROR, RORP, RCR, RCRP

■Executing condition

Instruction	Executing condition
ROR, RCR	
RORP, RCRP	

■Argument

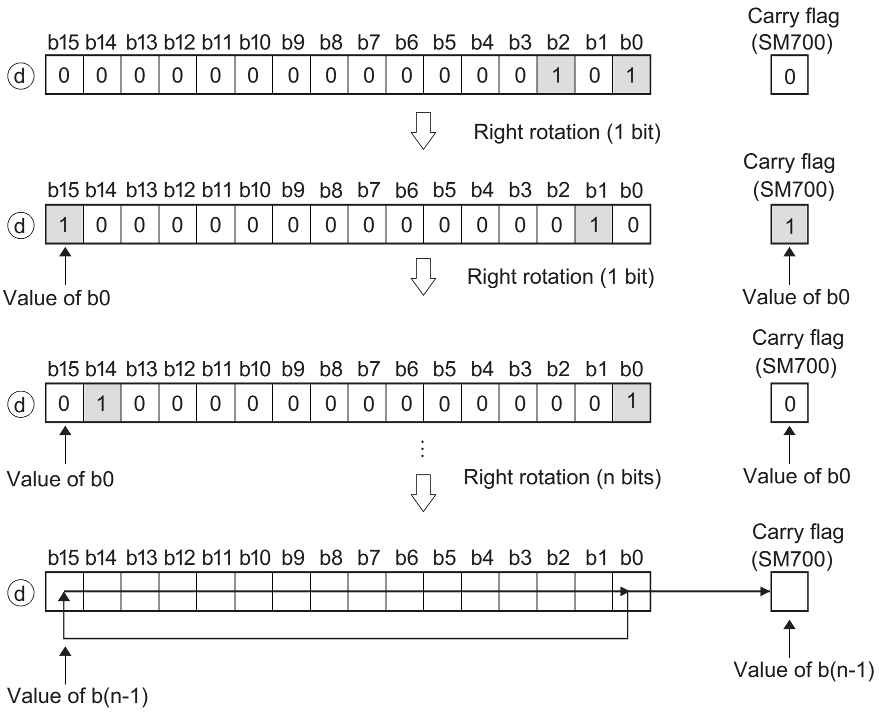
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Number of rotations (0 to 15)	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device to be rotated	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○							○	—
(d)	○							—	—

Processing details

■ROR(P)

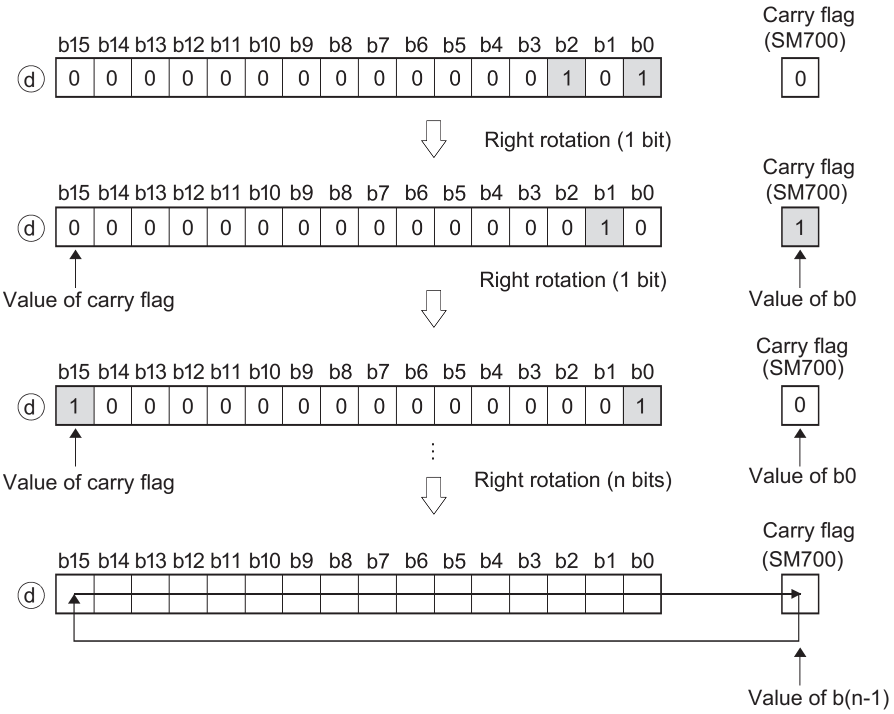
- Rotates 16-bit data of the device specified for (d), not including the carry flag, n bits to the right. The carry flag is ON or OFF depending on the status prior to the execution of the ROR instruction.



- When a bit device is specified for (d), a rotation is performed within the device range set by digit specification. The number of bits that rotate is the remainder of n divided by 'number of bits specified by digit specification'. For example, when n = 15 and 'number of bits specified by digit specification' = 12 bits, the remainder of 15 ÷ 12 = 1 is 3, thus the data are rotated 3 bits.
- Specify any of 0 to 15 for n. If the value specified for n is 16 or higher, the remainder of n divided by 16 is used for rotation. For example, when n = 18, the remainder of 18 ÷ 16 = 1 is 2, thus the data are rotated 2 bits to the right.

■RCR(P)

- Rotates 16-bit data of the device specified for (d), including the carry flag, n bits to the right. The carry flag is ON or OFF depending on the status prior to the execution of the RCR instruction.



- When a bit device is specified for (d), a rotation is performed within the device range set by digit specification. The number of bits that rotate is the remainder of n divided by 'number of bits specified by digit specification'. For example, when n = 15 and 'number of bits specified by digit specification' = 12 bits, the remainder of 15 ÷ 12 = 1 is 3, thus the data are rotated 3 bits.
- Specify any of 0 to 15 for n. If the value specified for n is 16 or higher, the remainder of n divided by 16 is used for rotation. For example, when n = 18, the remainder of 18 ÷ 16 = 1 is 2, thus the data are rotated 2 bits to the right.

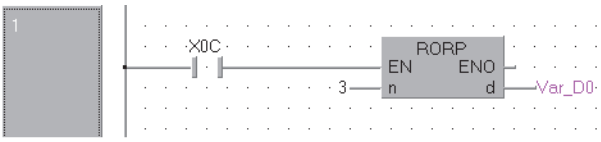
Operation error

- There is no operation error.

Program example

- In the following program, the data in Var_D0 are rotated, not including the carry flag, 3 bits to the right, when X0C turns ON.

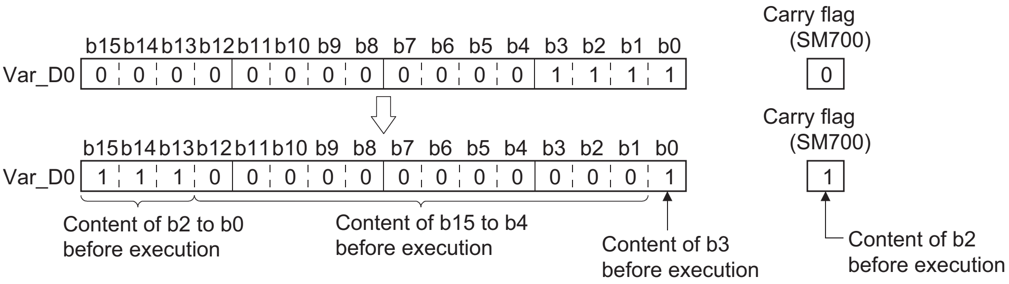
[Structured ladder/FBD]



[ST]

RORP(X0C,3,Var_D0);

[Operation]

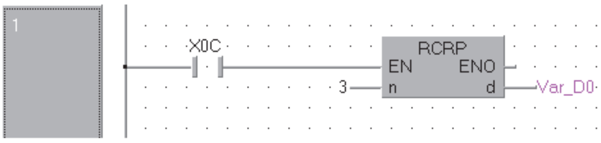


	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	Carry flag (SM700)
Var_D0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
Var_D0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Content of b2 to b0 before execution
Content of b15 to b4 before execution
Content of b3 before execution
Content of b2 before execution

- In the following program, the data in Var_D0 are rotated, including the carry flag, 3 bits to the right, when X0C turns ON.

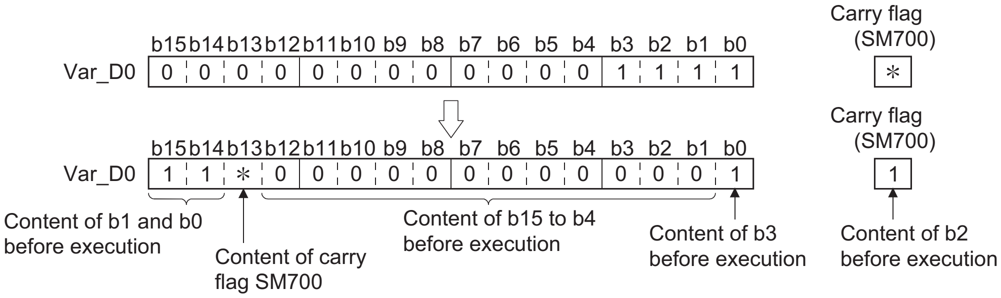
[Structured ladder/FBD]



[ST]

RCRP(X0C,3,Var_D0);

[Operation]



	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	Carry flag (SM700)
Var_D0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	∗
Var_D0	1	1	∗	0	0	0	0	0	0	0	0	0	0	0	0	1	1

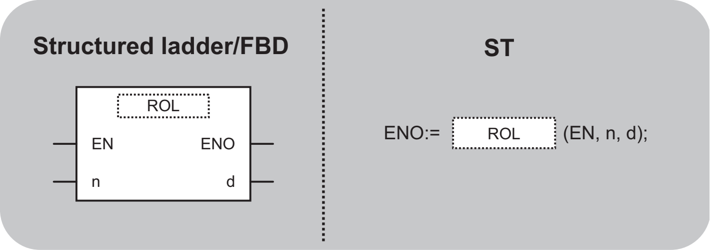
Content of b1 and b0 before execution
Content of carry flag SM700
Content of b15 to b4 before execution
Content of b3 before execution
Content of b2 before execution

∗ON/OFF status of the carry flag depends on its status before the execution of RCR instruction.

Left rotation of 16-bit data

ROL(P), RCL(P)

Basic | High performance | Process | Redundant | Universal | LCPU

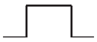
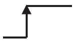


```
ENO:= ROL (EN, n, d);
```

Any of the following instruction can go in the dotted squares.

ROL, ROLP, RCL, RCLP

■Executing condition

Instruction	Executing condition
ROL, RCL	
ROLP, RCLP	

■Argument

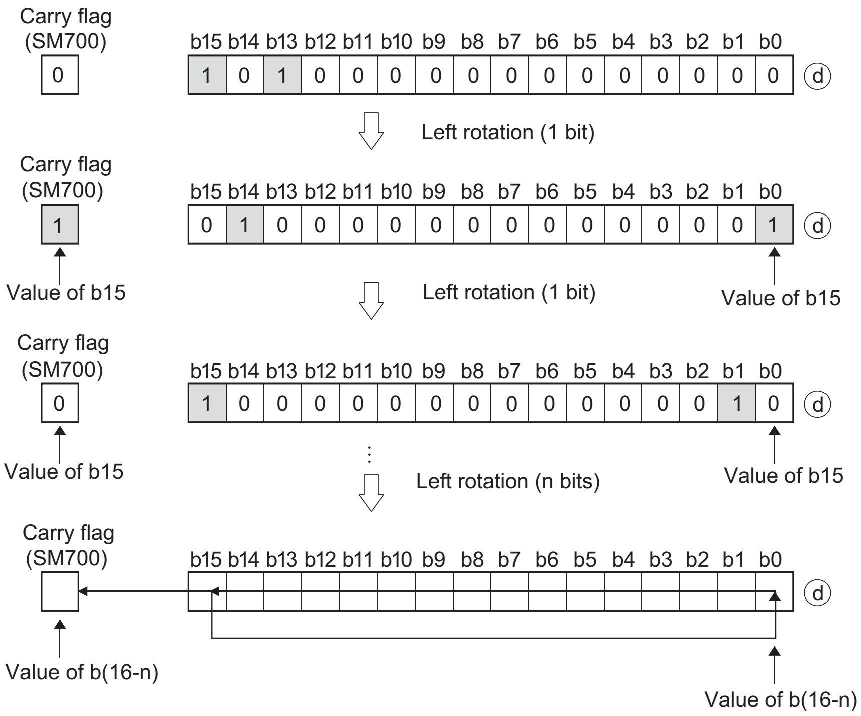
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Number of rotations (0 to 15)	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device to be rotated	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○							○	—
(d)	○							—	—

Processing details

■ROL(P)

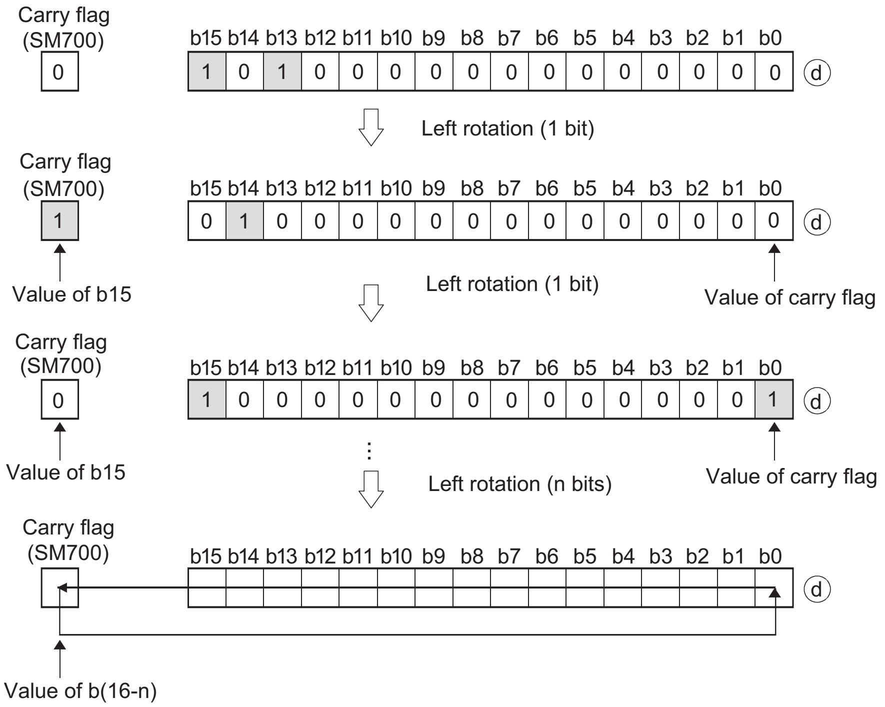
- Rotates 16-bit data of the device specified for (d), not including the carry flag, n bits to the left. The carry flag turns ON or OFF depending on its status prior to the execution of the ROL instruction.



- When a bit device is specified for (d), a rotation is performed within the device range set by digit specification. The number of bits that rotate is the remainder of n divided by 'number of bits specified by digit specification'. For example, when n = 15 and 'number of bits specified by digit specification' = 12 bits, the remainder of 15 ÷ 12 = 1 is 3, thus the data are rotated 3 bits.
- Specify any of 0 to 15 for n. If the value specified for n is 16 or higher, the remainder of n divided by 16 is used for rotation. For example, when n = 18, the remainder of 18 ÷ 16 = 1 is 2, thus the data are rotated 2 bits to the left.

■RCL(P)

- Rotates 16-bit data of the device specified for (d), including the carry flag, n bits to the left. The carry flag turns ON or OFF depending on its status prior to the execution of the RCL instruction.



- When a bit device is specified for (d), a rotation is performed within the device range set by digit specification. The number of bits that rotate is the remainder of n divided by 'number of bits specified by digit specification'. For example, when n = 15 and 'number of bits specified by digit specification' = 12 bits, the remainder of 15 ÷ 12 = 1 is 3, thus the data are rotated 3 bits.
- Specify any of 0 to 15 for n. If the value specified for n is 16 or higher, the remainder of n divided by 16 is used for rotation. For example, when n = 18, the remainder of 18 ÷ 16 = 1 is 2, thus the data are rotated 2 bits to the left.

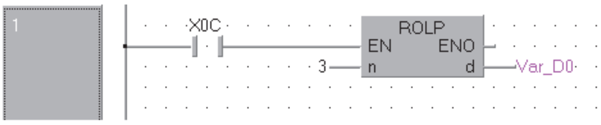
Operation error

- There is no operation error.

Program example

- In the following program, the data in Var_D0 are rotated, not including the carry flag, 3 bits to the left, when X0C turns ON.

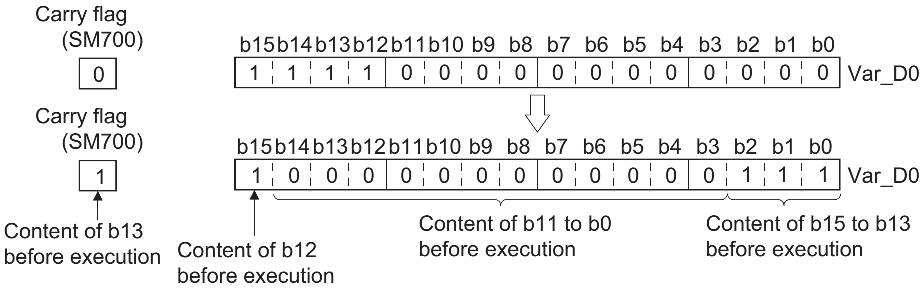
[Structured ladder/FBD]



[ST]

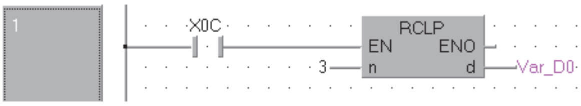
ROLP(X0C,3,Var_D0);

[Operation]



- In the following program, the data in Var_D0 are rotated, including the carry flag, 3 bits to the left, when X0C turns ON.

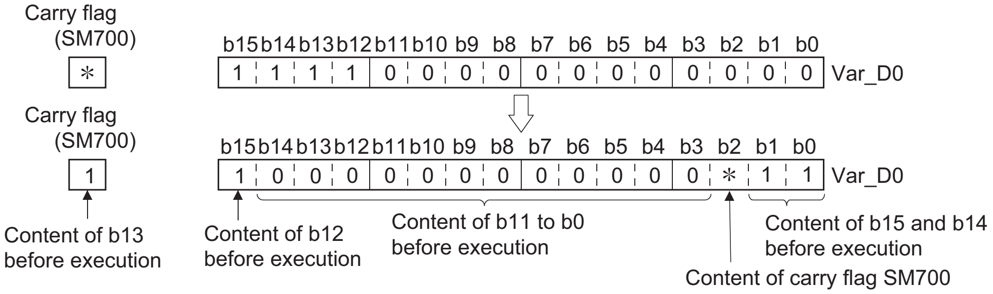
[Structured ladder/FBD]



[ST]

RCLP(X0C,3,Var_D0);

[Operation]



b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

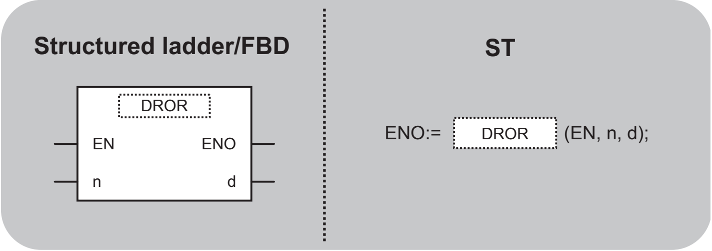
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
1	0	0	0	0	0	0	0	0	0	0	0	0	*	1	1

*ON/OFF status of the carry flag depends on its status before the execution of RCL instruction.

Right rotation of 32-bit data

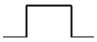
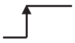
DROR(P), DRCR(P)

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.
DROR, DRORP, DRCR, DRCRP

■Executing condition

Instruction	Executing condition
DROR, DRCR	
DRORP, DRCRP	

■Argument

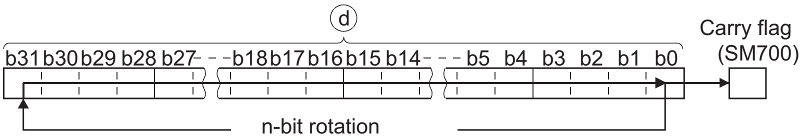
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Number of rotations (0 to 31)	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device to be rotated	ANY32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○							○	—
(d)	○							—	—

Processing details

■DROR(P)

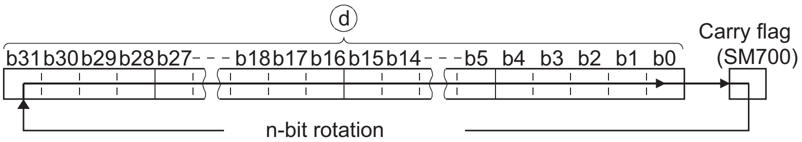
- Rotates 32-bit data of the device specified for (d), not including the carry flag, n bits to the right. The carry flag turns ON or OFF depending on its status prior to the execution of the DROR instruction.



- When a bit device is specified for (d), a rotation is performed within the device range set by digit specification. The number of bits that rotate is the remainder of n divided by 'number of bits specified by digit specification'. For example, when n = 31 and 'number of bits specified by digit specification' = 24 bits, the remainder of 31 ÷ 24 = 1 is 7, thus the data are rotated 7 bits.
- Specify any of 0 to 31 for n. If the value specified for n is 32 or higher, the remainder of n divided by 32 is used for rotation. For example, when n = 34, the remainder of 34 ÷ 32 = 1 is 2, thus the data are rotated 2 bits to the right.

■DRCR(P)

- Rotates 32-bit data of the device specified for (d), including the carry flag, n bits to the right. The carry flag turns ON or OFF depending on its status prior to the execution of the DRCR instruction.


ⓓ
b31 b30 b29 b28 b27- - - b18 b17 b16 b15 b14- - - b5 b4 b3 b2 b1 b0
Carry flag
(SM700)
n-bit rotation

- When a bit device is specified for (d), a rotation is performed within the device range set by digit specification. The number of bits that rotate is the remainder of n divided by 'number of bits specified by digit specification'. For example, when n = 31 and 'number of bits specified by digit specification' = 24 bits, the remainder of 31 ÷ 24 = 1 is 7, thus the data are rotated 7 bits.
- Specify any of 0 to 31 for n. If the value specified for n is 32 or higher, the remainder of n divided by 32 is used for rotation. For example, when n = 34, the remainder of 34 ÷ 32 = 1 is 2, thus the data are rotated 2 bits to the right.

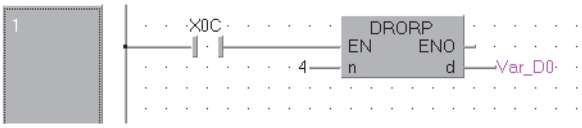
Operation error

- There is no operation error.

Program example

- In the following program, the data in Var_D0 are rotated, not including the carry flag, 4 bits to the right, when X0C turns ON.

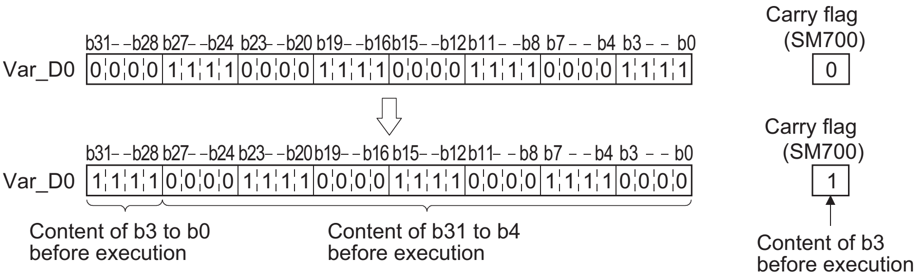
[Structured ladder/FBD]



[ST]

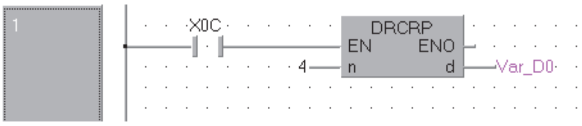
DRORP(X0C,4,Var_D0);

[Operation]


b31- -b28 b27- -b24 b23- -b20 b19- -b16 b15- -b12 b11- -b8 b7- - b4 b3- - b0
Var_D0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
Carry flag
(SM700)
0
b31- -b28 b27- -b24 b23- -b20 b19- -b16 b15- -b12 b11- -b8 b7- - b4 b3- - b0
Var_D0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
Carry flag
(SM700)
1
Content of b3 to b0
before execution
Content of b31 to b4
before execution
Content of b3
before execution

- In the following program, the data in Var_D0 are rotated, including the carry flag, 4 bits to the right, when X0C turns ON.

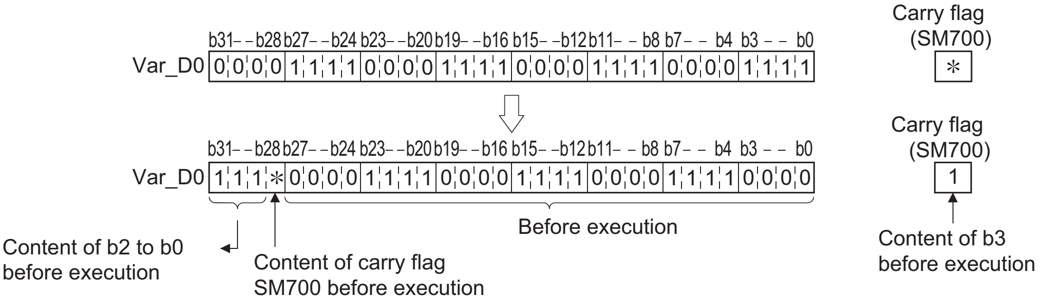
[Structured ladder/FBD]



[ST]

DRCRP(X0C,4,Var_D0);

[Operation]

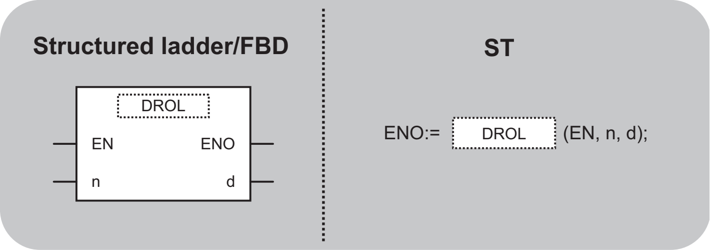

b31- -b28 b27- -b24 b23- -b20 b19- -b16 b15- -b12 b11- -b8 b7- - b4 b3- - b0
Var_D0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
Carry flag
(SM700)
*
b31- -b28 b27- -b24 b23- -b20 b19- -b16 b15- -b12 b11- -b8 b7- - b4 b3- - b0
Var_D0 1 1 1 * 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
Carry flag
(SM700)
1
Content of b2 to b0
before execution
Content of carry flag
SM700 before execution
Before execution
Content of b3
before execution

*: ON/OFF status of the carry flag depends on its status before the execution of DRCR instruction.

Left rotation of 32-bit data

DROL(P), DRCL(P)

Basic | High performance | Process | Redundant | Universal | LCPU

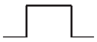
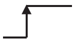


```
ENO:= DROL (EN, n, d);
```

Any of the following instruction can go in the dotted squares.

DROL, DROLP, DRCL, DRCLP

■Executing condition

Instruction	Executing condition
DROL, DRCL	
DROLP, DRCLP	

■Argument

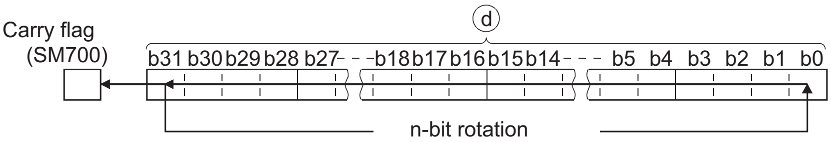
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Number of rotations (0 to 31)	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device to be rotated	ANY32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○							○	—
(d)	○							—	—

Processing details

■DROL(P)

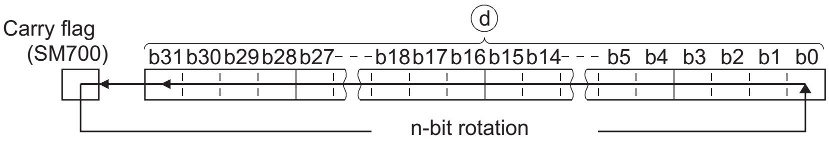
- Rotates 32-bit data of the device specified for (d), not including the carry flag, n bits to the left. The carry flag turns ON or OFF depending on its status prior to the execution of the DROL instruction.



- When a bit device is specified for (d), a rotation is performed within the device range set by digit specification. The number of bits that rotate is the remainder of n divided by 'number of bits specified by digit specification'. For example, when n = 31 and 'number of bits specified by digit specification' = 24 bits, the remainder of 31 ÷ 24 = 1 is 7, thus the data are rotated 7 bits.
- Specify any of 0 to 31 for n. If the value specified for n is 32 or higher, the remainder of n divided by 32 is used for rotation. For example, when n = 34, the remainder of 34 ÷ 32 = 1 is 2, thus the data are rotated 2 bits to the left.

■DRCL(P)

- Rotates 32-bit data of the device specified for (d), including the carry flag, n bits to the left. The carry flag turns ON or OFF depending on its status prior to the execution of the DRCL instruction.


Carry flag
(SM700)
ⓓ
b31 b30 b29 b28 b27 - - b18 b17 b16 b15 b14 - - - b5 b4 b3 b2 b1 b0
n-bit rotation

- When a bit device is specified for (d), a rotation is performed within the device range set by digit specification. The number of bits that rotate is the remainder of n divided by 'number of bits specified by digit specification'. For example, when n = 31 and 'number of bits specified by digit specification' = 24 bits, the remainder of 31 ÷ 24 = 1 is 7, thus the data are rotated 7 bits.
- Specify any of 0 to 31 for n. If the value specified for n is 32 or higher, the remainder of n divided by 32 is used for rotation. For example, when n = 34, the remainder of 34 ÷ 32 1 is 2, the data are rotated 2 bits to the left.

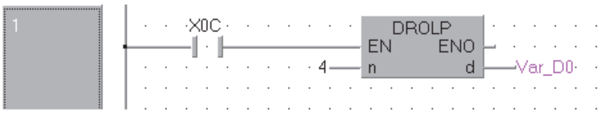
Operation error

- There is no operation error.

Program example

- In the following program, the data in Var_D0 are rotated, not including the carry flag, 4 bits to the left, when X0C turns ON.

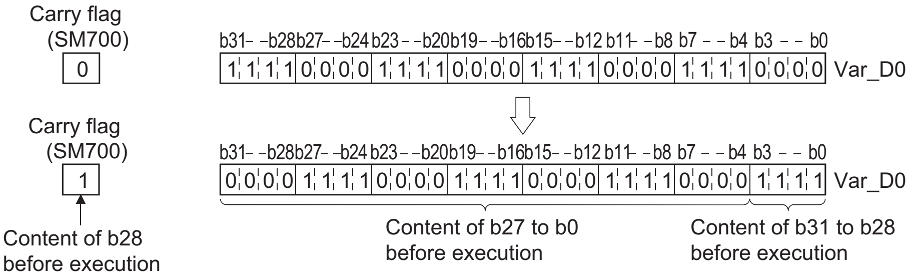
[Structured ladder/FBD]



[ST]

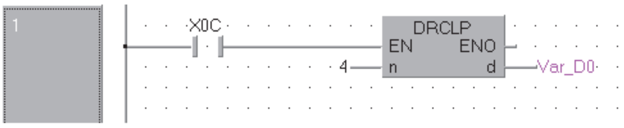
DROLP(X0C,4,Var_D0);

[Operation]



- In the following program, the data in Var_D0 are rotated, including the carry flag, 4 bits to the left, when X0C turns ON.

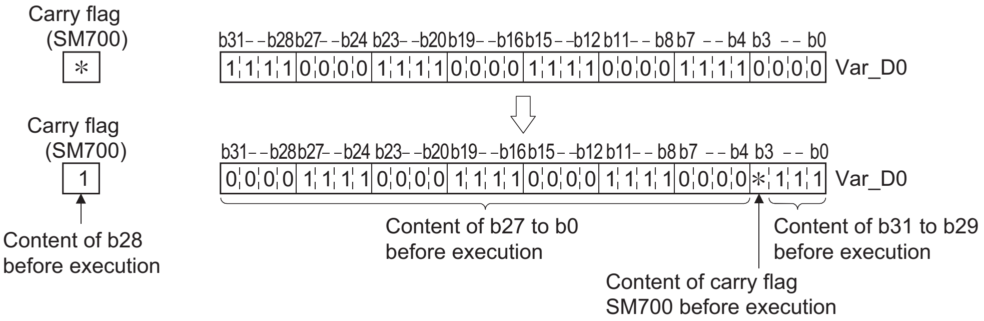
[Structured ladder/FBD]



[ST]

DRCLP(X0C,4,Var_D0);

[Operation]



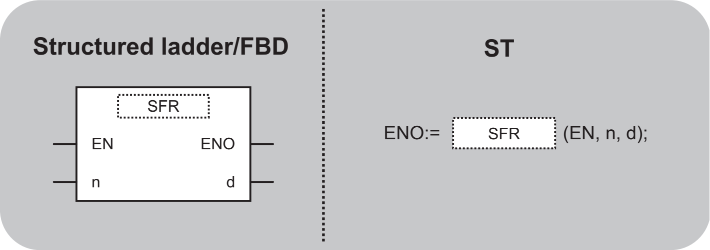
* : ON/OFF status of the carry flag depends on its status before the execution of DRCL instruction.

7.3 Shift Instructions

n-bit right/left shift of 16-bit data

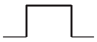
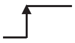
SFR(P), SFL(P)

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.
SFR, SFRP, SFL, SFLP

■Executing condition

Instruction	Executing condition
SFR, SFL	
SFRP, SFLP	

■Argument

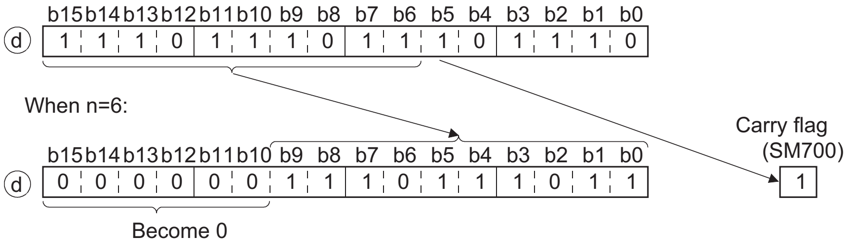
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Number of shifts (0 to 15)	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores data to be shifted	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○							○	—
(d)	○							—	—

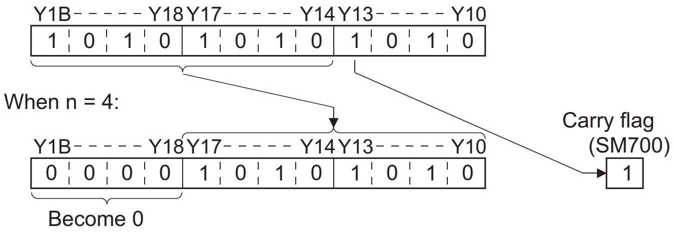
Processing details

■SFR(P)

- Shifts to the right by n bits of the 16-bit data in the device specified for (d). The n bits from the most significant bit become 0.


b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0
d
1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
When n=6:
Carry flag
(SM700)
b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0
d
0 0 0 0 0 0 1 1 1 0 1 1 1 0 1 1
1
Become 0

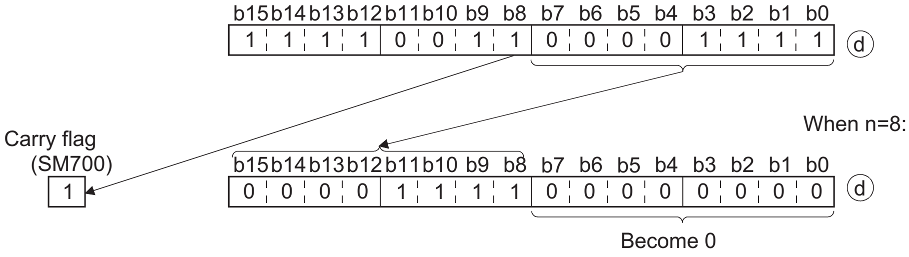
- When a bit device is specified for (d), a right shift is performed within the device range of points by digit specification.


Y1B - - - - - Y18 Y17 - - - - - Y14 Y13 - - - - - Y10
1 0 1 0 1 0 1 0 1 0 1 0
When n = 4:
Carry flag
(SM700)
Y1B - - - - - Y18 Y17 - - - - - Y14 Y13 - - - - - Y10
0 0 0 0 1 0 1 0 1 0 1 0
1
Become 0

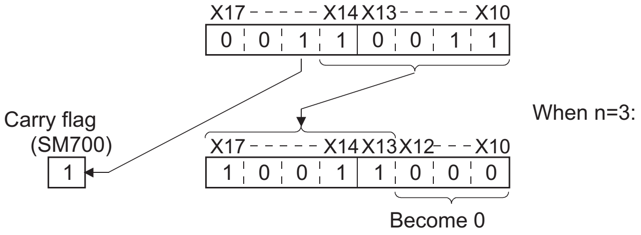
- Specify any of 0 to 15 for n. The number of bits that shift is the remainder of n divided by 'number of bits specified by digit specification'. For example, when n = 15 and 'number of bits specified by digit specification' = 8 bits, the remainder of 18 ÷ 16 = 1 is 2, thus the data are shifted 7 bits.

■SFL(P)

- Shifts 16-bit data in the device specified for (d) n bits to the left. The n bits from the lowest bit become 0.


b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0
1 1 1 1 0 0 1 1 0 0 0 0 1 1 1 1
d
When n=8:
Carry flag
(SM700)
1
b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0
0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0
d
Become 0

- When a bit device is specified for (d), a left shift is performed within the device range set by digit specification.


X17 - - - - - X14 X13 - - - - - X10
0 0 1 1 0 0 1 1
When n=3:
Carry flag
(SM700)
1
X17 - - - - - X14 X13 X12 - - - X10
1 0 0 1 1 0 0 0
Become 0

- Specify any of 0 to 15 for n. If the value specified for n is 16 or higher, the remainder of n divided by 16 is used for left shift. For example, when n = 18, the remainder of 18 ÷ 16 = 1 is 2, thus the data are shifted 2 bits to the left.

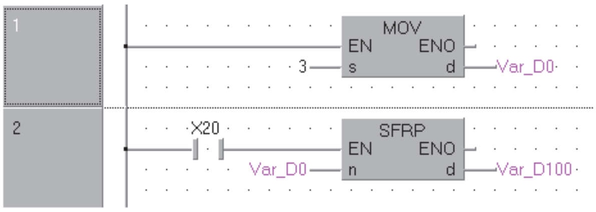
Operation error

- There is no operation error.

Program example

- In the following program, the data in Var_D0 are shifted to the right by the number of bits specified for Var_D100 when X20 turns ON.

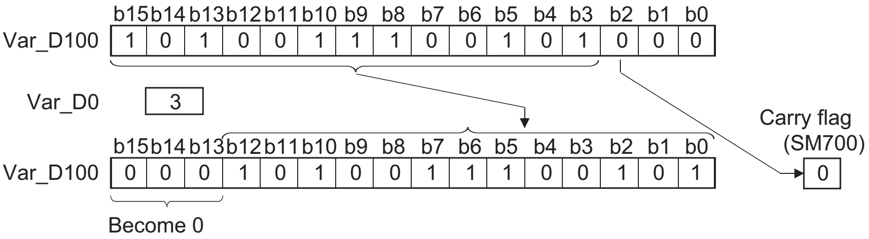
[Structured ladder/FBD]



[ST]

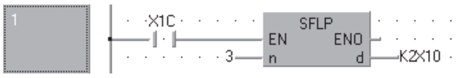
```
Var_D0:=3;
SFRP(X20,Var_D0,Var_D100);
```

[Operation]



- In the following program, the data in the devices from X10 to X17 are shifted 3 bits to the left when X1C turns ON.

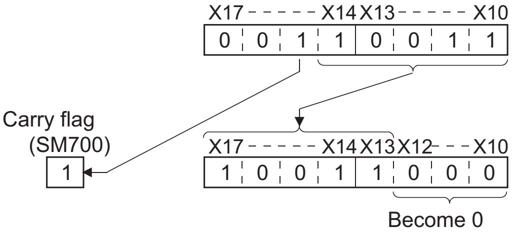
[Structured ladder/FBD]



[ST]

```
SFLP(X1C,3,K2Y10);
```

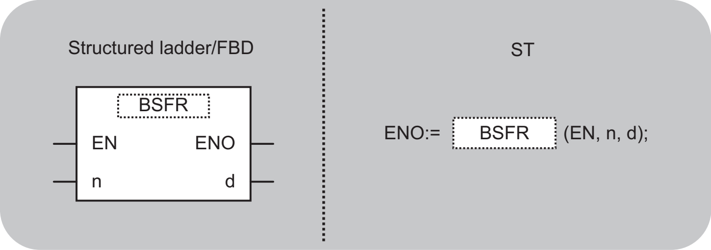
[Operation]


X17 - - - - - X14 X13 - - - - - X10
0 0 1 1 0 0 1 1
Carry flag
(SM700)
1
X17 - - - - - X14 X13 X12 - - - X10
1 0 0 1 1 0 0 0
Become 0

1-bit right/left shift of n-bit data

BSFR(P), BSFL(P)

Basic | High performance | Process | Redundant | Universal | LCPU

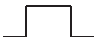
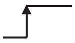


```
ENO:= BSFR (EN, n, d);
```

Any of the following instruction can go in the dotted squares.

BSFR, BSFRP, BSFL, BSFLP

■Executing condition

Instruction	Executing condition
BSFR, BSFL	
BSFRP, BSFLP	

■Argument

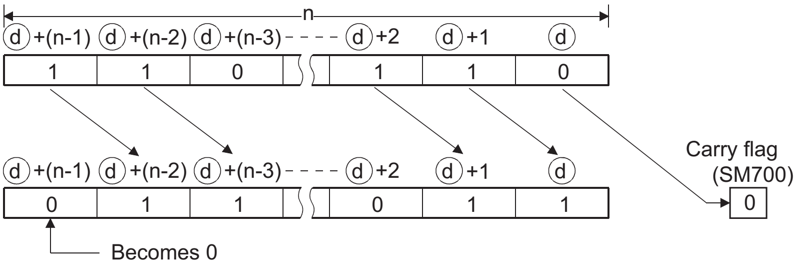
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Number of devices to be shifted	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device to be shifted	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○	○							—
(d)	○	—							—

Processing details

■BSFR(P)

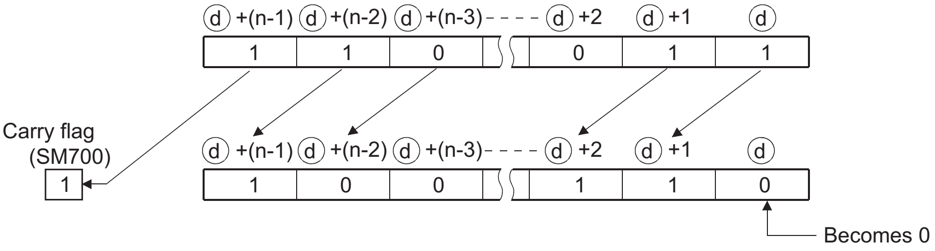
- Shifts the data n points from the device specified for (d) to the right by 1 bit.


n
ⓓ+(n-1) ⓓ+(n-2) ⓓ+(n-3) - - - - ⓓ+2 ⓓ+1 ⓓ
1 1 0 1 1 0
ⓓ+(n-1) ⓓ+(n-2) ⓓ+(n-3) - - - - ⓓ+2 ⓓ+1 ⓓ
0 1 1 0 1 1
Carry flag
(SM700)
0
Becomes 0

- The device specified for (d) + (n - 1) becomes 0.

■BSFL(P)

- Shifts the data n points from the device specified for (d) to the left by 1 bit.



(d)+(n-1) (d)+(n-2) (d)+(n-3) (d)+2 (d)+1 (d)
1 1 0 0 1 1
Carry flag
(SM700)
1
(d)+(n-1) (d)+(n-2) (d)+(n-3) (d)+2 (d)+1 (d)
1 0 0 1 1 0
Becomes 0

- The device specified for (d) becomes 0.

Operation error

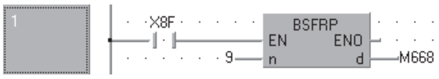
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (d) exceeds the corresponding device.	○	○	○	○	○	○

Program example

- In the following program, the data in the devices from M668 to M676 are shifted to the right when X8F turns ON.

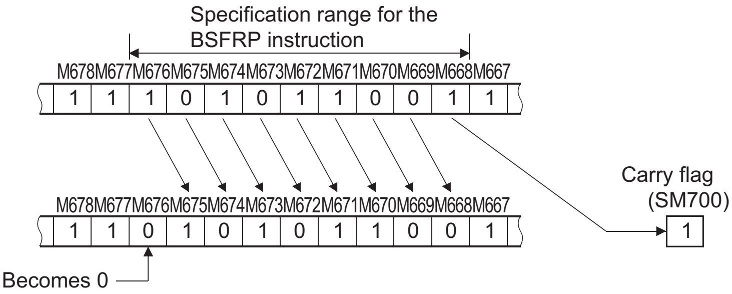
[Structured ladder/FBD]



[ST]

BSFRP(X8F,9,M668);

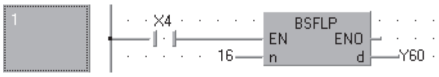
[Operation]



Specification range for the BSFRP instruction
M678 M677 M676 M675 M674 M673 M672 M671 M670 M669 M668 M667
1 1 1 0 1 0 1 1 0 0 1 1
M678 M677 M676 M675 M674 M673 M672 M671 M670 M669 M668 M667
1 1 0 1 0 1 0 1 1 0 0 1
Carry flag
(SM700)
1
Becomes 0

- In the following program, the data in the devices from Y60 to Y6F are shifted to the left when X4 turns ON.

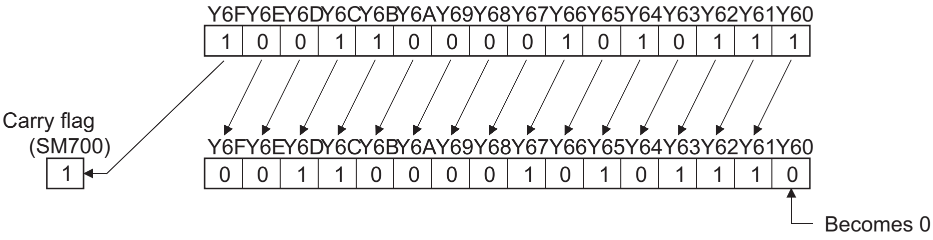
[Structured ladder/FBD]



[ST]

BSFLP(X4,16,Y60);

[Operation]



Y6F Y6E Y6D Y6C Y6B Y6A Y69 Y68 Y67 Y66 Y65 Y64 Y63 Y62 Y61 Y60
1 0 0 1 1 0 0 0 0 1 0 1 0 1 1 1
Carry flag
(SM700)
1
Y6F Y6E Y6D Y6C Y6B Y6A Y69 Y68 Y67 Y66 Y65 Y64 Y63 Y62 Y61 Y60
0 0 1 1 0 0 0 0 1 0 1 0 1 1 1 0
Becomes 0

Precautions

When using the BSFR/BSFL instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

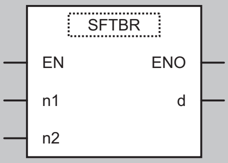
n-bit right/left shift of n-bit data

SFTBR(P), SFTBL(P)

✗Basic ✗High performance ✗Process ✗Redundant Ver. Universal LCPU

- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported

Structured ladder/FBD

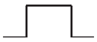
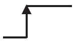


ST

ENO:= SFTBR (EN, n1, n2, d);

Any of the following instruction can go in the dotted squares.
SFTBR, SFTBRP, SFTBL, SFTBLP

■Executing condition

Instruction	Executing condition
SFTBR, SFTBL	
SFTBRP, SFTBLP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Number of devices to be shifted	ANY16
	n2	Number of shifts	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device to be shifted	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n1	—	○	○	○					—
n2	—	○	○	○					—
(d)	○*1	—	○	—					—

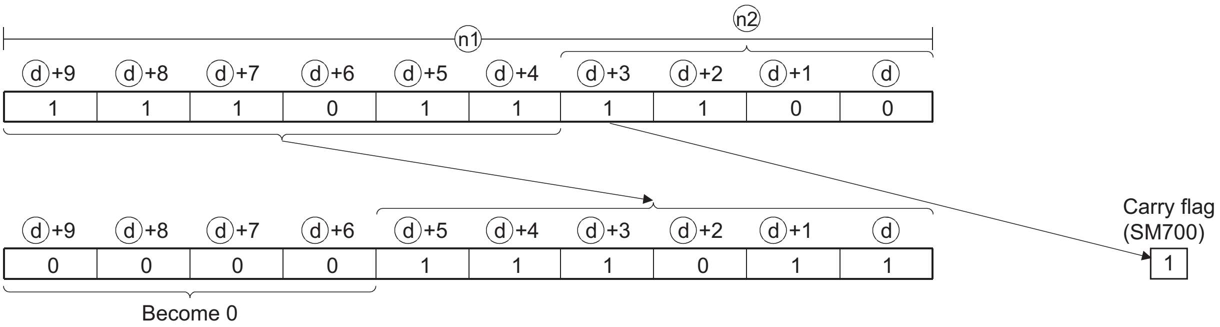
*1 T, C, ST, or S device cannot be used.

Processing details

■SFTBR(P)

- Shifts the n1-bit data from the device specified for (d) to the right by n2 bit.

When n1 = 10, n2 = 4



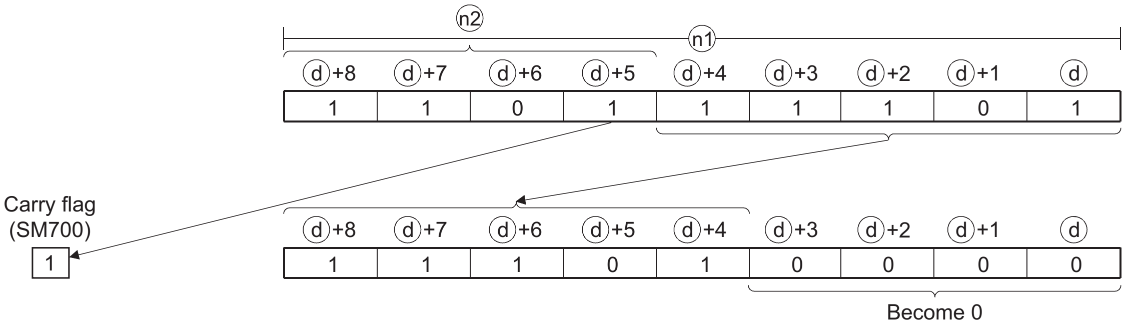
n1
n2
d+9 d+8 d+7 d+6 d+5 d+4 d+3 d+2 d+1 d
1 1 1 0 1 1 1 1 0 0
d+9 d+8 d+7 d+6 d+5 d+4 d+3 d+2 d+1 d
0 0 0 0 1 1 1 0 1 1
Become 0
Carry flag
(SM700)
1

- n1 and n2 are specified as n1 > n2. When n1 ≤ n2, bits are shifted by the amount of the remainder obtained from dividing n2 by n1. However, no processing is performed if the value of the remainder obtained from dividing n2 by n1 is 0.
- n1 is specified within the range from 1 to 64.
- The n2 bits from the most significant bit become 0. When n1 < n2, the bits equal to the amount of the remainder obtained from dividing n2 by n1 become 0.
- No processing is performed if 0 is specified for n1 or n2.

■SFTBL(P)

- Shifts the n1-bit data from the device specified for (d) to the left by n2 bit.

When n1 = 10, n2 = 4



n2
n1
d+8 d+7 d+6 d+5 d+4 d+3 d+2 d+1 d
1 1 0 1 1 1 1 0 1
Carry flag
(SM700)
1
d+8 d+7 d+6 d+5 d+4 d+3 d+2 d+1 d
1 1 1 0 1 0 0 0 0
Become 0

- n1 and n2 are specified as n1 > n2. When n1 ≤ n2, bits are shifted by the amount of the remainder obtained from dividing n2 by n1. However, no processing is performed if the value of the remainder obtained from dividing n2 by n1 is 0.
- n1 is specified within the range from 1 to 64.
- The n2 bits from the lowest bit become 0. When n1 < n2, the bits equal to the amount of the remainder obtained from dividing n2 by n1 become 0.
- No processing is performed if 0 is specified for n1 or n2.

Operation error

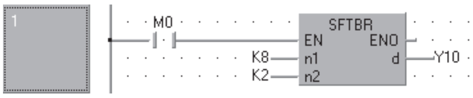
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for n1 is not within 0 to 64. A negative value is specified for n2.	—	—	—	—	○	○
4101	The device points specified for n1 exceed the device range specified for (d).	—	—	—	—	○	○

Program example

- In the following program, the data in the devices from Y10 to Y17 (8 bits) specified for (d) are shifted to the right by 2 bits (n2) when M0 turns ON.

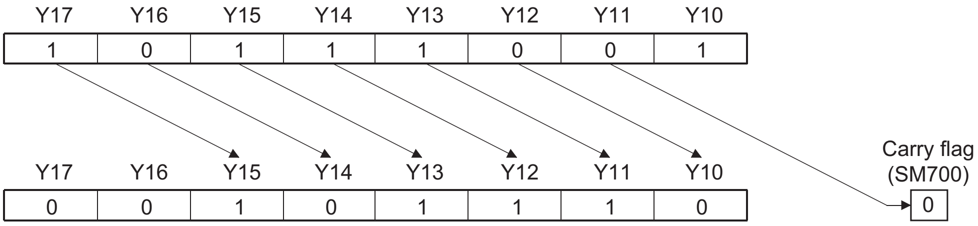
[Structured ladder/FBD]



[ST]

SFTBR(M0,K8,K2,Y10);

[Operation]



Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
1	0	1	1	1	0	0	1

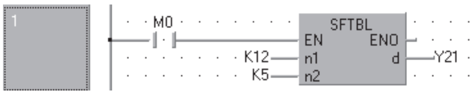
Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
0	0	1	0	1	1	1	0

Carry flag (SM700)

0

- In the following program, the data in the devices from Y21 to Y2C (12 bits) specified for (d) are shifted to the left by 5 bits (n2) when M0 turns ON.

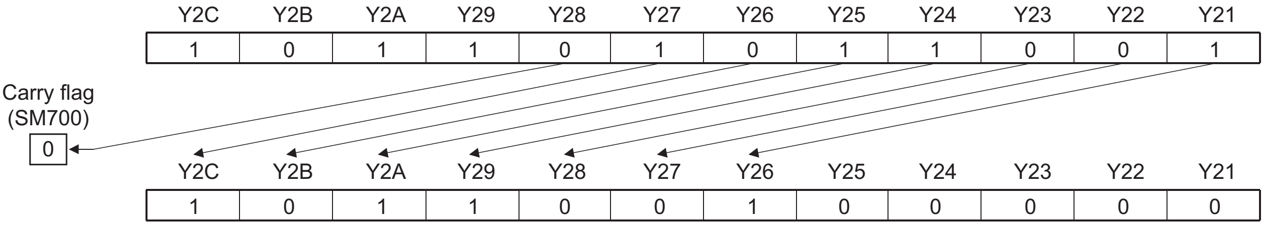
[Structured ladder/FBD]



[ST]

SFTBL(M0,K12,K5,Y21);

[Operation]



Y2C	Y2B	Y2A	Y29	Y28	Y27	Y26	Y25	Y24	Y23	Y22	Y21
1	0	1	1	0	1	0	1	1	0	0	1

Carry flag (SM700)

0

Y2C	Y2B	Y2A	Y29	Y28	Y27	Y26	Y25	Y24	Y23	Y22	Y21
1	0	1	1	0	0	1	0	0	0	0	0

Precautions

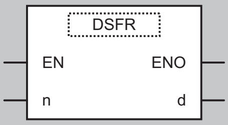
When using the SFTBR/SFTBL instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

1-word right/left shift of n-word data

DSFR(P), DSFL(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



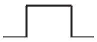
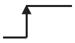
ST

ENO:= DSFR (EN, n, d);

Any of the following instruction can go in the dotted squares.

DSFR, DSFRP, DSFL, DSFLP

■Executing condition

Instruction	Executing condition
DSFR, DSFL	
DSFRP, DSFLP	

■Argument

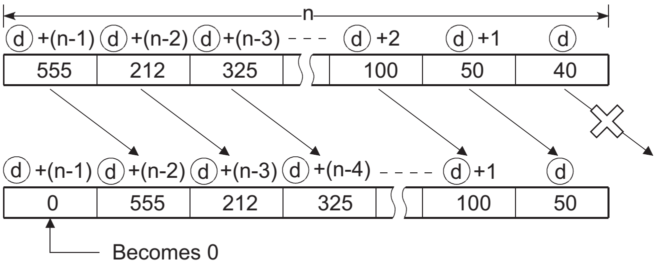
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Number of devices to be shifted	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device to be shifted	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○	○		○					—
(d)	—	○		—					—

Processing details

■DSFR(P)

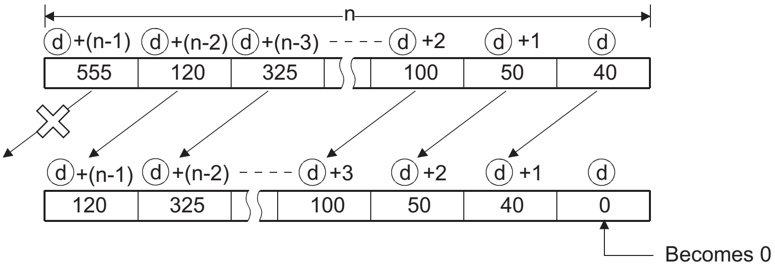
- Shifts the data n points from the device specified for (d) to the right by 1 word.



- The device specified for (d) + (n - 1) becomes 0.

■DSFL(P)

- Shifts the data n points from the device specified for (d) to the left by 1 word.



(d)+(n-1)	(d)+(n-2)	(d)+(n-3)	- - - -	(d)+2	(d)+1	(d)
555	120	325		100	50	40

(d)+(n-1)	(d)+(n-2)	- - - -	(d)+3	(d)+2	(d)+1	(d)
120	325		100	50	40	0

- The device specified for (d) becomes 0.

Operation error

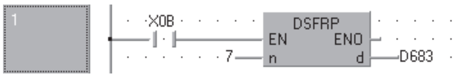
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (d) exceeds the corresponding device.	○	○	○	○	○	○

Program example

- In the following program, the data from D683 to D689 are shifted to the right when X0B turns ON.

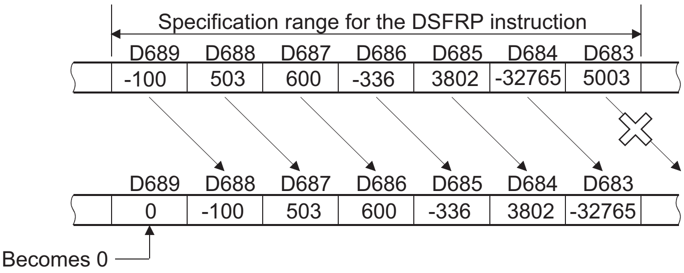
[Structured ladder/FBD]



[ST]

DSFRP(X0B,7,D683);

[Operation]

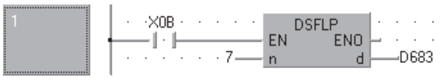


D689	D688	D687	D686	D685	D684	D683
-100	503	600	-336	3802	-32765	5003

D689	D688	D687	D686	D685	D684	D683
0	-100	503	600	-336	3802	-32765

- In the following program, the data from D683 to D689 are shifted to the left when X0B turns ON.

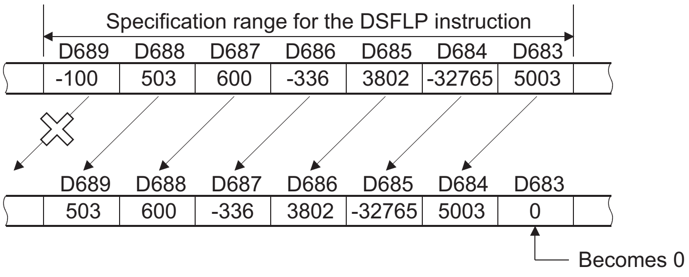
[Structured ladder/FBD]



[ST]

DSFLP(X0B,7,D683);

[Operation]



D689	D688	D687	D686	D685	D684	D683
-100	503	600	-336	3802	-32765	5003

D689	D688	D687	D686	D685	D684	D683
503	600	-336	3802	-32765	5003	0

Precautions

When using the DSFR/DSFL instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

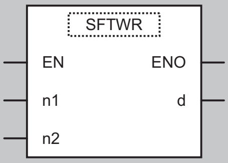
n-word right/left shift of n-word data

SFTWR(P), SFTWL(P)

Basic High performance Process Redundant Ver. Universal LCPU

- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported

Structured ladder/FBD

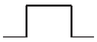
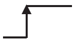


ST

ENO:= SFTWR (EN, n1, n2, d);

Any of the following instruction can go in the dotted squares.
SFTWR, SFTWRP, SFTWL, SFTWLP

■Executing condition

Instruction	Executing condition
SFTWR, SFTWL	
SFTWRP, SFTWLP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Number of words to be shifted	ANY16
	n2	Number of words	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device to be shifted	ANY16

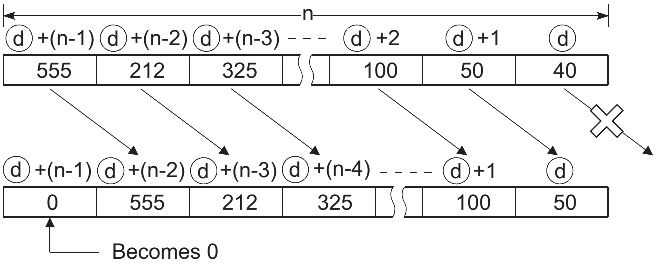
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n1	—	○	○	○					—
n2	—	○	○	○					—
(d)	—	○	○	—					—

Processing details

■SFTWR(P)

- Shifts the n1-word data from the device specified for (d) to the right by n2 word.

When n1 = 9, n2 = 4

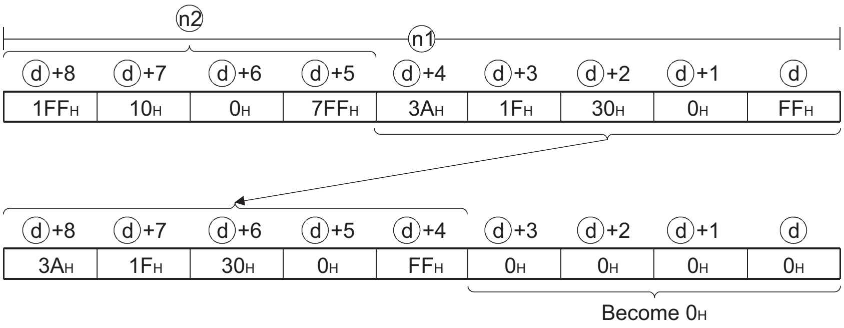


- The n2 words from the most significant word become 0.
- No processing is performed if 0 is specified for n1 or n2.
- When n1 ≤ n2, all n1-word data from the device specified for (d) become 0.

■SFTWL(P)

- Shifts the n1-word data from the device specified for (d) to the right by n2 word.

When n1 = 9, n2 = 4



- The n2 words from the lowest word becomes 0.
- No processing is performed if 0 is specified for n1 or n2.
- When n1 ≤ n2, all n1-word data from the device specified for (d) become 0.

Operation error

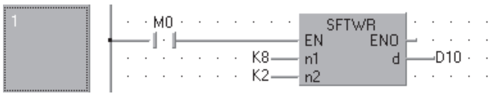
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	A negative value is specified for n1 or n2.	—	—	—	—	○	○
4101	The device points specified for n1 exceed the device range specified for (d).	—	—	—	—	○	○

Program example

- In the following program, the 8-word data (n1) from D10 specified for (d) are shifted to the right by 2 words (n2) when M0 turns ON.

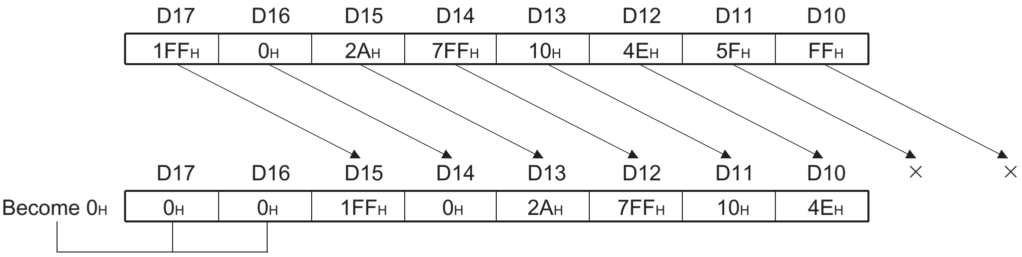
[Structured ladder/FBD]



[ST]

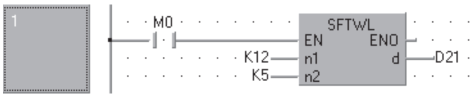
SFTWR(M0,K8,K2,D10);

[Operation]



- In the following program, the 12-word data (n1) from D21 specified for (d) are shifted to the left by 5 words (n2) when M0 turns ON.

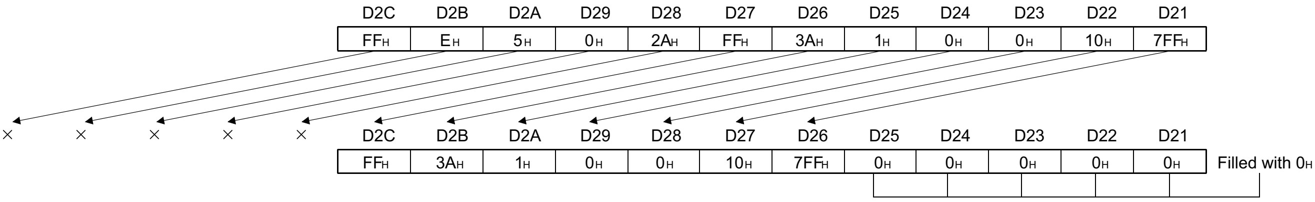
[Structured ladder/FBD]



[ST]

SFTWL(M0,K12,K5,D21);

[Operation]



Precautions

When using the SFTWR/SFTWL instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

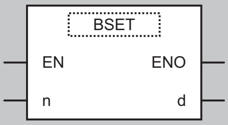
7.4 Bit Processing Instructions

Bit set and reset of word devices

BSET(P), BRST(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD

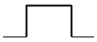
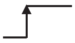


ST

ENO:= BSET (EN, n, d);

Any of the following instruction can go in the dotted squares.
BSET, BSETP, BRST, BRSTP

■Executing condition

Instruction	Executing condition
BSET, BRST	
BSETP, BRSTP	

■Argument

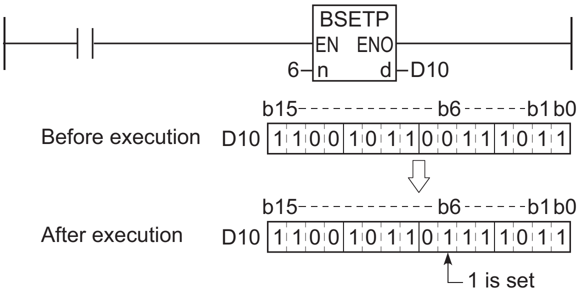
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Bit number to be set or reset (0 to 15)	ANY16
Output argument	ENO	Execution result	Bit
	d	Device number whose bit is set or reset	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○							○	—
(d)	○							—	—

Processing details

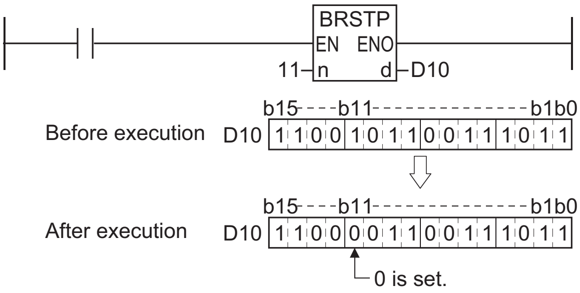
■BSET(P)

- Sets (to 1) the nth bit of the word device specified for (d).
- If n exceeds 15, bit set/reset is performed with the lower 4 bits of the data.



■BRST(P)

- Resets (to 0) the nth bit of the word device specified for (d).
- If n exceeds 15, bit set/reset is performed with the lower 4 bits of the data.



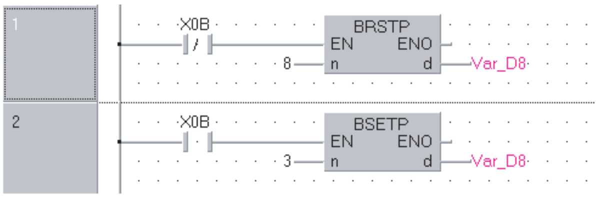
Operation error

- There is no operation error.

Program example

- In the following program, the 8th bit (b8) of Var_D8 is reset to 0 when X0B turns OFF, and the 3rd bit (b3) of Var_D8 is set to 1 when X0B turns ON.

[Structured ladder/FBD]



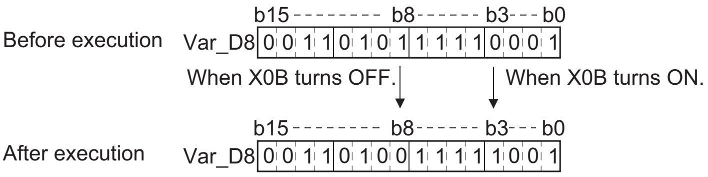
Resets b8 of Var_D8.

Sets b3 of Var_D8.

[ST]

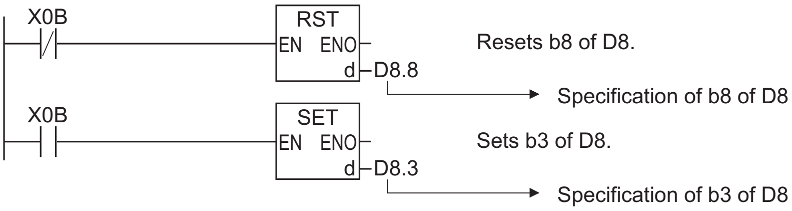
```
BRSTP(NOT(X0B),8,Var_D8);
BSETP(X0B,3,Var_D8);
```

[Operation]



Point

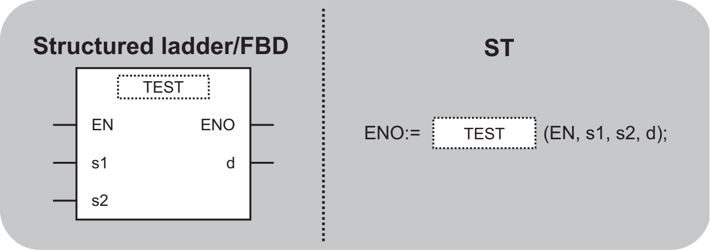
Bit set or reset of word devices can also be performed by bit specification of word device.
For details on bit specification of word device, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.
The processing of the program example would be performed as shown below with the bit specification of word device.



Bit test

TEST(P), DTEST(P)

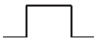
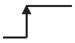
Basic High performance Process Redundant Universal LCPU



ENO:= TEST (EN, s1, s2, d);

Any of the following instruction can go in the dotted squares.
TEST, TESTP, DTEST, DTESTP

■Executing condition

Instruction	Executing condition
TEST, DTEST	
TESTP, DTESTP	

■Argument

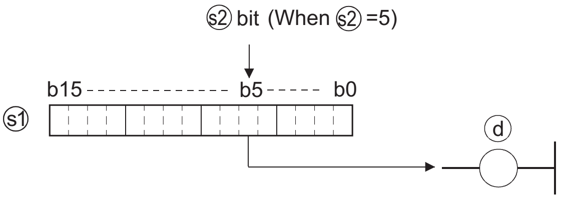
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start device number of the device that stores bit data to be extracted	ANY16/32
	s2	Position of bit data to be extracted (0 to 15 (TEST)/0 to 31 (DTEST))	ANY16
Output argument	ENO	Execution result	Bit
	d	Bit device number that stores extracted bit data	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○						○	—	—
(s2)	○						○	○	—
(d)	○						—	—	—

Processing details

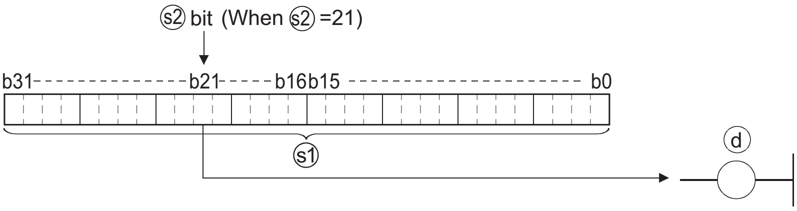
■TEST(P)

- Fetches bit data from the location specified for (s2) within the word device specified for (s1), and writes it to the bit device specified for (d).
- The bit device specified for (d) is turned OFF when the corresponding bit is 0 and ON when it is 1.
- The position specified for (s2) specifies the position of each bit in a 1-word data block (0 to 15). When 16 or higher value is specified for (s2), the target is the bit data at the position indicated by the remainder of n divided by 16. For example, when n = 18, the remainder of 18 ÷ 16 1 is 2, thus the target is the data of b2.



■DTEST(P)

- Fetches bit data from the location specified for (s2) within the 2-word device specified for (s1), and writes it to the bit device specified for (d).
- The bit device specified for (d) is turned OFF when the corresponding bit is 0 and ON when it is 1. If (n) exceeds 15, the instruction sets lower 4 bits of data.
- The position specified for (s2) specifies the position of each bit in a 2-word data block (0 to 31). When 32 or higher value is specified for (s2), the target is the bit data at the position indicated by the remainder of n divided by 32. For example, when n = 34, the remainder of 34 ÷ 32 1 is 2, thus the target is the data of b2.



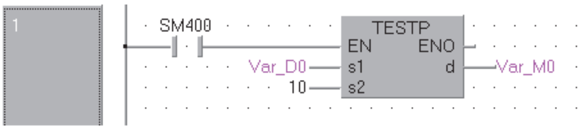
Operation error

- There is no operation error.

Program example

- In the following program, Var_M0 is turned ON or OFF depending on the status of the 10th bit of the 1-word data block (Var_D0).

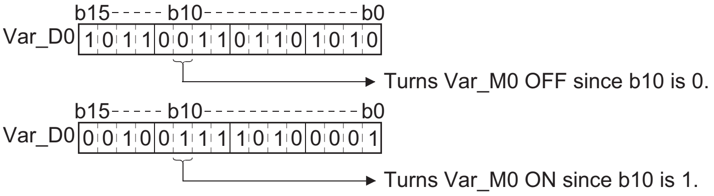
[Structured ladder/FBD]



[ST]

```
TESTP(SM400,Var_D0,10,Var_M0);
```

[Operation]



	b15					b10										b0
Var_D0	1	0	1	1	0	0	1	1	0	1	1	0	1	0	1	0

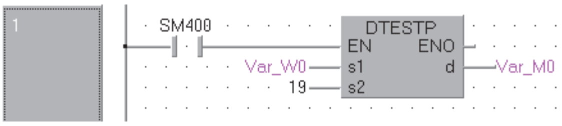
Turns Var_M0 OFF since b10 is 0.

	b15					b10										b0
Var_D0	0	0	1	0	0	1	1	1	1	0	1	0	0	0	0	1

Turns Var_M0 ON since b10 is 1.

- In the following program, Var_M0 is turned ON or OFF, depending on the status of the 19th bit of the 2-word data (Var_W0).

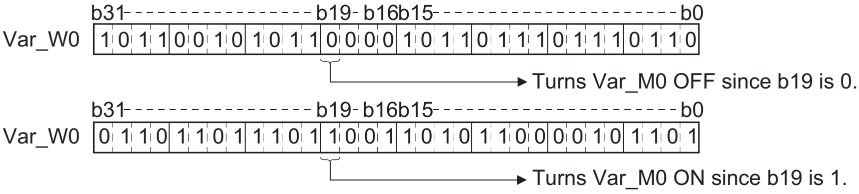
[Structured ladder/FBD]



[ST]

```
DTESTP(SM400,Var_W0,19,Var_M0);
```

[Operation]



	b31												b19			b16	b15															b0
Var_W0	1	0	1	1	0	0	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0	1	1	1	0	1	1	0

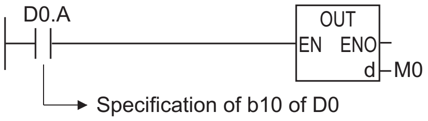
Turns Var_M0 OFF since b19 is 0.

	b31												b19			b16	b15															b0
Var_W0	0	1	1	0	1	1	0	1	1	1	0	1	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1

Turns Var_M0 ON since b19 is 1.

Point

Programs using the bit test instruction can be rewritten as programs using bit specification of word device. If the program were changed to use bit specification of word device, it would appear as follows.

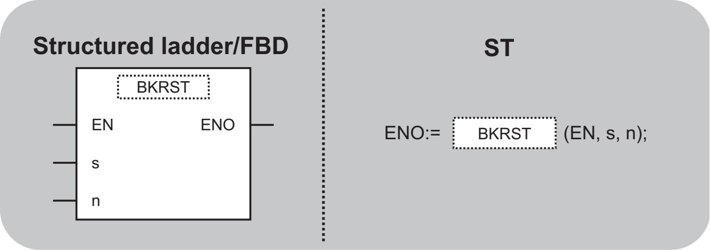


M0 turns ON/OFF depending on the ON/OFF status of b10 of D0 (D0.A).

Batch reset of bit devices

BKRST(P)

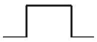
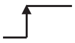
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

BKRST, BKRSTP

■Executing condition

Instruction	Executing condition
BKRST	
BKRSTP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device to be reset	Bit
	n	Number of devices to be reset	ANY16
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○			—					—
n	○			○					—

Processing details

- Resets bit devices n points from the bit device specified for (s).

Device	Status
Annunciator (F)	• Turns OFF bit devices n points from annunciator (F) number specified for (s). • Deletes annunciator number turned OFF from SD64 to SD79 and compresses remaining data forward. • Stores annunciators stored in SD64 to SD79, to SD63.
Timer (T) Counter (C)	Sets 0 for the current value n points from the timer (T) or counter (C) number specified for (s), and turns the coil or contact OFF.
Bit devices other than the above	Turns OFF coil or contact n points from the device specified for (s).

- If the specified device is OFF, the device status will not change.

Operation error

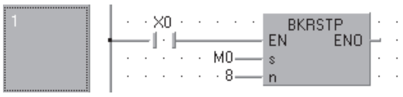
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (d) exceeds the corresponding device.	○	○	○	○	○	○

Program example

- In the following program, the devices from M0 to M7 are turned OFF when X0 turns ON.

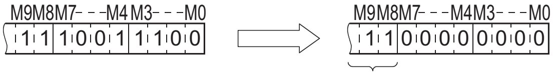
[Structured ladder/FBD]



[ST]

BKRSTP(X0,M0,8);

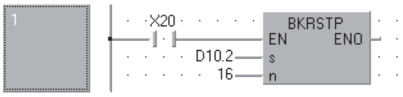
[Operation]



Not changed

- In the following program, the data from 2nd bit (b2) of D10 to 1st bit (b1) of D11 are set to 0 when X20 turns ON.

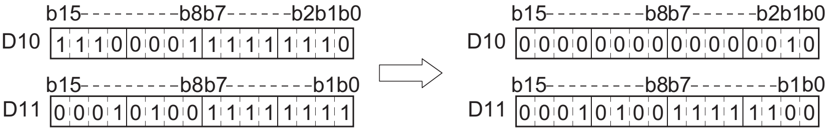
[Structured ladder/FBD]



[ST]

BKRSTP(X20,D10.2,16);

[Operation]


b15- - - - - - - - -b8b7- - - - - -b2b1b0
D10
1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 0
b15- - - - - - - - -b8b7- - - - - - - -b1b0
D11
0 0 0 1 0 1 0 0 1 1 1 1 1 1 1 1
b15- - - - - - - - -b8b7- - - - - -b2b1b0
D10
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
b15- - - - - - - - -b8b7- - - - - - - -b1b0
D11
0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 0

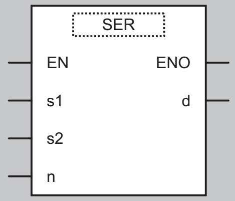
7.5 Data Processing Instructions

16-/32-bit data search

SER(P), DSER(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



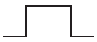
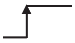
ST

ENO:= SER (EN, s1, s2, n, d);

Any of the following instruction can go in the dotted squares.

SER, SERP, DSER, DSERP

■Executing condition

Instruction	Executing condition
SER, DSER	
SERP, DSERP	

■Argument

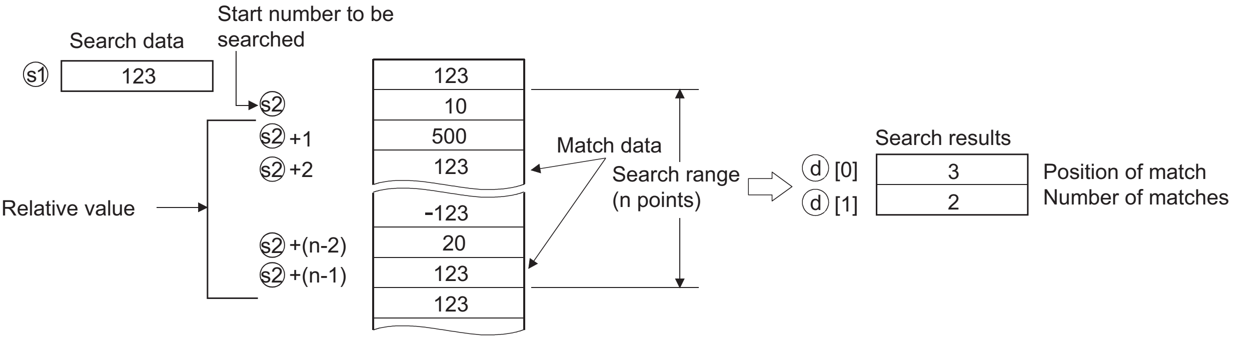
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Search data	ANY16/32
	s2	Data to be searched	ANY16/32
	n	Number of searches	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the search result	Array of ANY16 (0..1)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○	○		○	○			○	—
(s2)	—	○		—	—			—	—
n	○	○		○	○			○	—
(d)	—	○		—	○			—	—

Processing details

■SER(P)

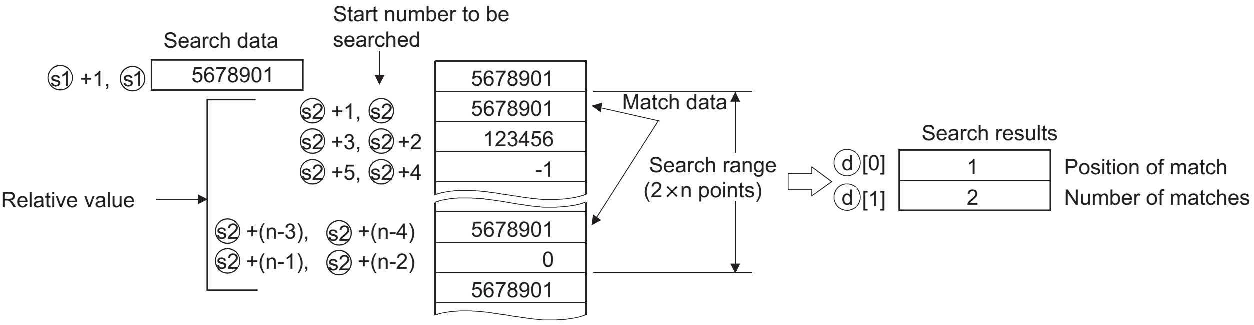
- Searches n points from the 16-bit data of the device specified for (s2), regarding 16-bit data of the device specified for (s1) as a keyword. Then, the number of matches with the keyword is stored to the device specified for (d)[1], and the relative value from (s2) to the first matched device number is stored to the device specified for (d)[0].



- No processing is performed if n is 0 or a negative value.
- If no matches are found in the search, the device specified for (d) becomes 0.

■DSER(P)

- Searches n points from the device specified for (s2) in units of 32 bits (2 × n points in units of 16 bits) regarding 32-bit data of the device specified for (s1) as a keyword. Then, the number of matches with the keyword is stored to the device specified for (d)[1], and the relative value from (s2)to the first matched device number is stored to the device specified for (d)[0].



- No processing is performed if n is 0 or a negative value.
- If no matches are found in the search, the device specified for (d) becomes 0.

Point

If the data to be searched using the SER(P) or DSER(P) instruction is sorted in the ascending order, searches can be accelerated by the use of the binary search method, which is activated by turning SM702 [*1] ON. However, correct search results are not obtained if SM702 is turned ON when the data to be searched are not sorted in the ascending order.

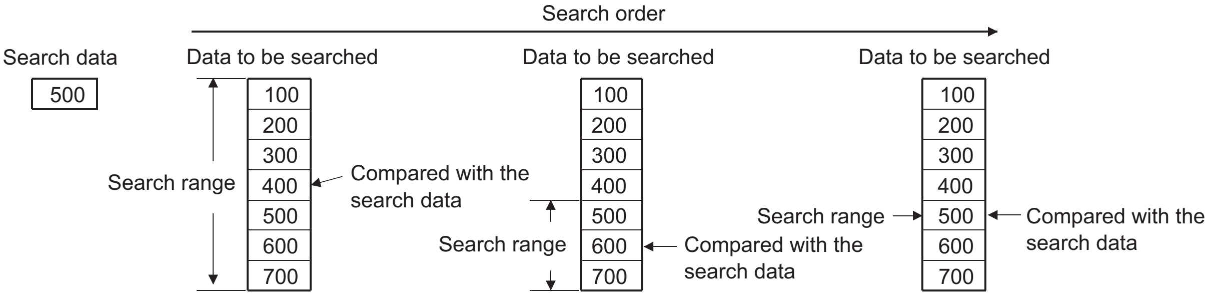
[*1] SM702 is the special relay for setting the search method.

- SM702 OFF: Linear search method (linear search method)

(Comparison with the search data starts from the beginning of the data to be searched.)

- SM702 ON: Binary search method

(Obtains the center value of the sorted array and decides if the obtained value is larger or smaller than the search value, then, chooses the area for search between the larger and smaller value divisions. By repeating this process, the area for search is narrowed down.)



Operation error

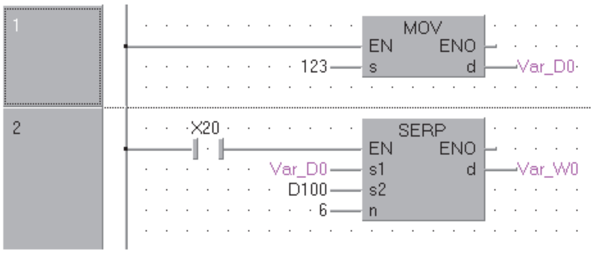
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The n points from the device (s2) exceeds the specified device range.	○	○	○	○	○	○
	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the devices from D100 to D105 are searched by the data in Var_D0 when X20 turns ON, and the results are stored to Var_W0.

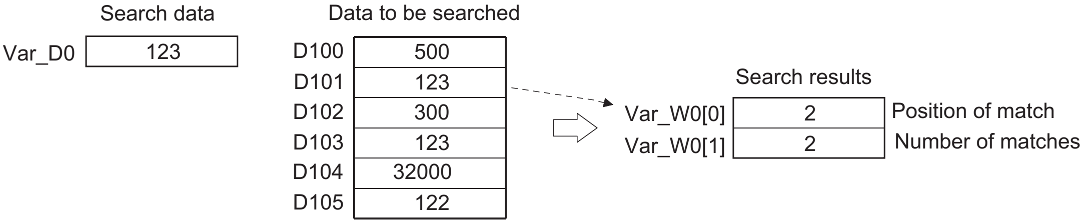
[Structured ladder/FBD]



[ST]

```
Var_D0:=123;
SERP(X20,Var_D0,D100,6,Var_W0);
```

[Operation]



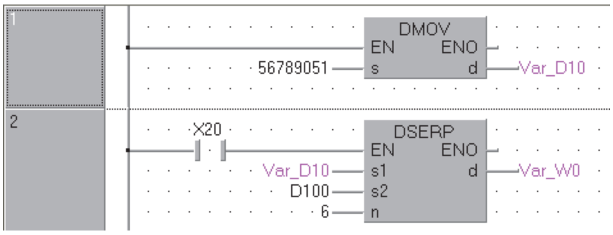
Search data	
Var_D0	123

Data to be searched	
D100	500
D101	123
D102	300
D103	123
D104	32000
D105	122

Search results		
Var_W0[0]	2	Position of match
Var_W0[1]	2	Number of matches

- In the following program, the devices from D100 to D111 are searched by the data in Var_D10 when X20 turns ON, and the results are stored to Var_W0.

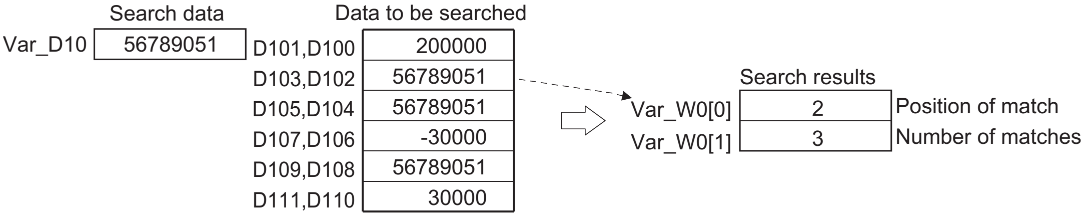
[Structured ladder/FBD]



[ST]

```
Var_D10:=56789051;
DSERP(X20,Var_D10,D100,6,Var_W0);
```

[Operation]



Search data	
Var_D10	56789051

Data to be searched	
D101,D100	200000
D103,D102	56789051
D105,D104	56789051
D107,D106	-30000
D109,D108	56789051
D111,D110	30000

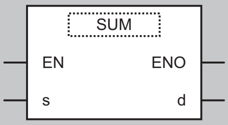
Search results		
Var_W0[0]	2	Position of match
Var_W0[1]	3	Number of matches

16-/32-bit data bit check

SUM(P), DSUM(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



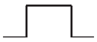
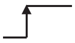
ST

ENO:= SUM (EN, s, d);

Any of the following instruction can go in the dotted squares.

SUM, SUMP, DSUM, DSUMP

■Executing condition

Instruction	Executing condition
SUM, DSUM	
SUMP, DSUMP	

■Argument

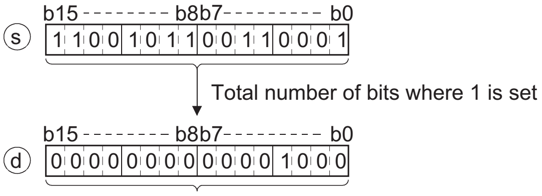
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data whose total number of bits set to 1 is counted	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the total number of bits	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○							○	—
(d)	○							—	—

Processing details

■SUM(P)

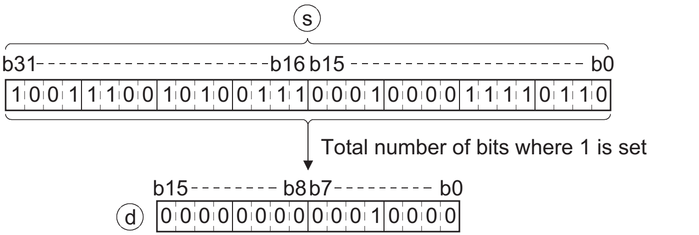
- From the 16-bit data in the device specified for (s), stores the total number of bits set to 1, to the device specified for (d).


b15 -------- b8b7--------- b0
s
1 1 0 0 1 0 1 1 0 0 1 1 0 0 0 1
Total number of bits where 1 is set
b15 -------- b8b7--------- b0
d
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

Stores the total number of bits where 1 is set in BIN data.
(There are 8 bits where 1 is set in the example.)

■DSUM(P)

- From the 32-bit data in the device specified for (s), stores the total number of bits where 1 is set, to the device specified for (d).



Stores the total number of bits where 1 is set in BIN data.
(There are 16 bits where 1 is set in the example.)

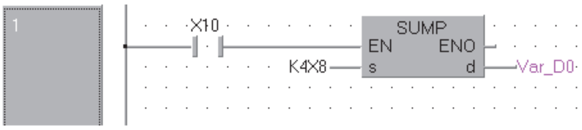
Operation error

- There is no operation error.

Program example

- In the following program, the number of bits which are turned ON in the devices from X8 to X17 are stored to Var_D0 when X10 turns ON.

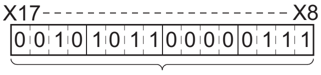
[Structured ladder/FBD]



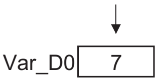
[ST]

SUMP(X10,K4X8,D0);

[Operation]

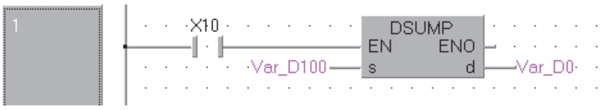


Stores the total number of bits where 1 is set to Var_D0.



- In the following program, the number of bits which are turned ON in Var_D100 are stored to Var_D0 when X10 turns ON.

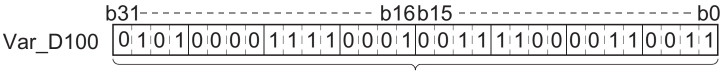
[Structured ladder/FBD]



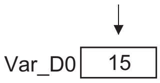
[ST]

DSUMP(X10,Var_D100,Var_D0);

[Operation]



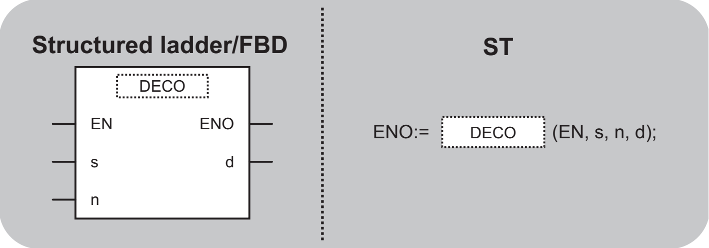
Stores the total number of bits where 1 is set to D0.



Decoding from 8 to 256 bits

DECO(P)

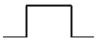
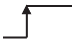
Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

DECO, DECOP

■Executing condition

Instruction	Executing condition
DECO	
DECOP	

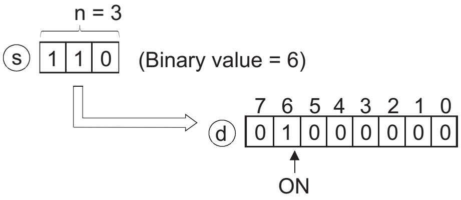
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Decoding data, or start number of the device that stores decoding data	ANY_SIMPLE
	n	Valid bit length (1 to 8), no processing on 0	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores decoding result	ANY_SIMPLE

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○			○				○	—
n	○			○				○	—
(d)	○			—				—	—

Processing details

- Turns ON the bit position of (d), which corresponds to the binary value specified by the lower n bits of (s).

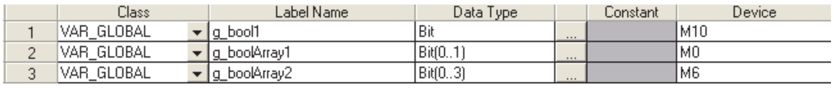


- The value of n can be specified between 1 and 8.
- If n = 0, there will be no operation, and the content of (d) will not change.
- Bit devices are treated as 1 bit, and word devices as 16 bits.
- More than n bits are required for (s). When using a label, specify the data type of n bits, or array type label of n bits.
- More than 2^n bits are required for (d). When using a label, specify the data type of 2^n bits, or array type label of 2^n bits.

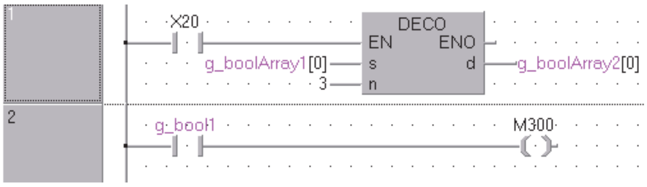
Precautions

The devices set as array type labels specified for (s) and (d) are used for the operation regardless of whether the required device points are reserved or not. Therefore, when the devices which are out of the range of device points set as array type labels are used for another operation, an operation error occurs and it may cause malfunction of the CPU module.

Ex.

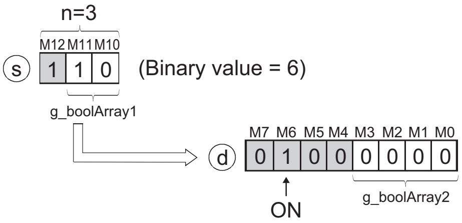


	Class	Label Name	Data Type		Constant	Device
1	VAR_GLOBAL	g_bool1	Bit	...		M10
2	VAR_GLOBAL	g_boolArray1	Bit(0..1)	...		M0
3	VAR_GLOBAL	g_boolArray2	Bit(0..3)	...		M6



As shown above, the binary value including the devices of the 3rd bit (M12) is used because 3 bits are required for (s). The next device and the following devices (M4 to M7) are used for the 4th bit to the 7th because 8 bits are required for (d).

When 6 is specified for the binary value used for (s) as shown below, g_bool1 turns ON because M6 turns ON.



Operation error

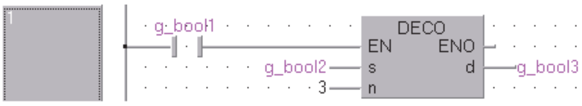
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value for n is not between 0 and 8.	○	○	○	○	○	○
4101	The range of 2^n bits from (d) exceeds the corresponding device. The range of n bits from (s) exceeds the each setting area of the corresponding device.	○	○	○	○	○	○

Program example

- In the following program, the 3 bits from g_bool2 (X0) are decoded when g_bool1 turns ON, and the results are stored to g_bool3 (M10) and the following devices.

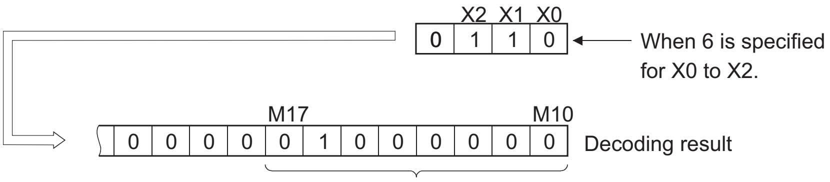
[Structured ladder/FBD]



[ST]

DECOP(g_bool1,g_bool2,3,g_bool3);

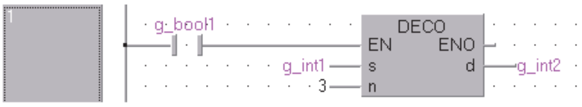
[Operation]



If 3 bits are specified as significant bits, 8 points are occupied.

- In the following program, the lower 3 bits of g_int1 (D0) are decoded when g_bool1 turns ON, and the results are stored to g_int2 (D10) and the following devices.

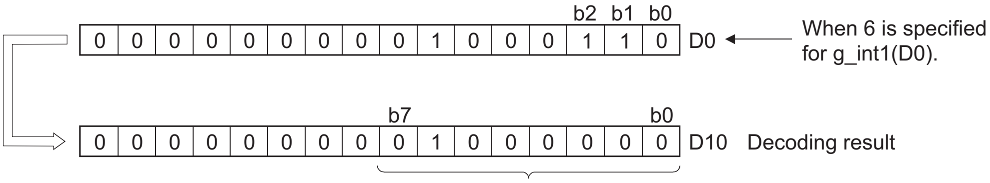
[Structured ladder/FBD]



[ST]

DECOP(g_bool1,g_int1,3,g_int2);

[Operation]

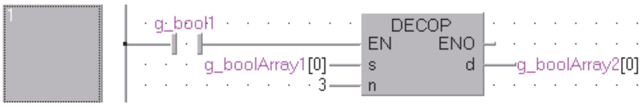


If 3 bits are specified as significant bits, 8 points are occupied.

- In the following program, the 3 bits from g_boolArray1[0] are decoded when g_bool1 turns ON, and the results are stored to g_boolArray2[0] and the following devices. The array label settings are as follows.

Class	Label	Data type	Device
VAR_GLOBAL	g_boolArray1	Bit(0..2)	X0
VAR_GLOBAL	g_boolArray2	Bit(0..7)	M10

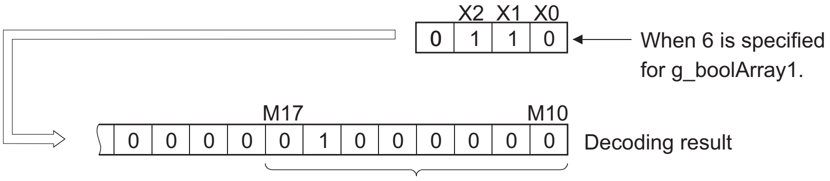
[Structured ladder/FBD]



[ST]

DECOP(g_bool1,g_boolArray1[0],3,g_boolArray2[0]);

[Operation]



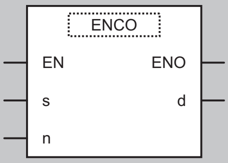
If 3 bits are specified as significant bits, 8 points are occupied.

Encoding from 256 to 8 bits

ENCO(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



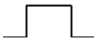
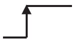
ST

ENO:= ENCO (EN, s, n, d);

Any of the following instruction can go in the dotted squares.

ENCO, ENCOP

■Executing condition

Instruction	Executing condition
ENCO	
ENCOP	

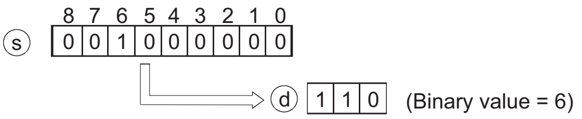
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores encoding data	ANY_SIMPLE
	n	Valid bit length (1 to 8), no processing on 0	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores encoding result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○			—				—	—
n	○			○				○	—
(d)	○			○				—	—

Processing details

- Stores the binary value corresponding to the bits set to 1 in the 2^n-bit data of (s) to (d).



- The value of n can be specified between 1 and 8.
- If n = 0, there will be no operation, and the content of (d) will not change.
- Bit devices are treated as 1 bit, and word devices as 16 bits.
- If more than 1 bit is set to 1, processing will be performed at the upper bit position.

- More than 2^n bits are required for (s). When using a label, use the data type that can reserve 2^n bits, or reserve 2^n bits or more for (d) or later devices by using array. For data type that can be used for (s) and the number of array elements by the value of n, refer to the following table.

Value specified n	Data types that can be specified for (s) and number of array element								
	Bit	Word (signed)	Double word (signed)	Word (unsigned)/ 16-bit string	Double word (unsigned)/ 32-bit string	Single-precision real	Double-precision real	String (32 characters)	Time
1	1 point	1 point	1 point	1 point	1 point	1 point	1 point	1 point	1 point
2	4 points	1 point	1 point	1 point	1 point	1 point	1 point	1 point	1 point
3	8 points	1 point	1 point	1 point	1 point	1 point	1 point	1 point	1 point
4	16 points	1 point	1 point	1 point	1 point	1 point	1 point	1 point	1 point
5	32 points	2 points	1 point	2 points	1 point	1 point	1 point	1 point	1 point
6	64 points	4 points	2 points	4 points	2 points	2 points	1 point	1 point	2 points
7	128 points	8 points	4 points	8 points	4 points	4 points	2 points	1 point	4 points
8	256 points	16 points	8 points	16 points	8 points	8 points	4 points	2 points	8 points

Operation error

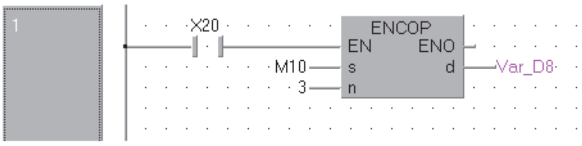
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value for n is not between 0 and 8. All data of 2^n bits from (s) are 0.	○	○	○	○	○	○
4101	The range of 2^n bits from (s) exceeds the corresponding device.	○	○	○	○	○	○

Program example

- In the following program, the 3 bits from M10 are encoded when X20 turns ON, and the results are stored to Var_D8.

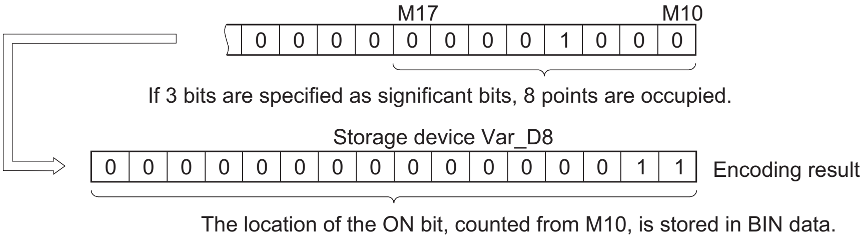
[Structured ladder/FBD]



[ST]

ENCOP(X20,M10,3,Var_D8);

[Operation]



0	0	0	0	0	0	0	0	1	0	0	0

If 3 bits are specified as significant bits, 8 points are occupied.

Storage device Var_D8

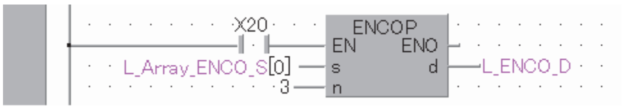
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Encoding result

The location of the ON bit, counted from M10, is stored in BIN data.

- The 3 bits from L_ArrayENCO_S[0] are encoded when X20 turns ON, and the results are stored to L_ENCO_D. Example of the program using an array label

[Structured ladder/FBD]

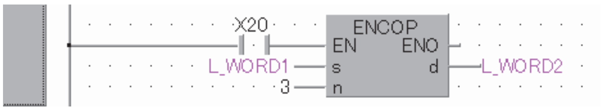


[ST]

ENCOP(X20,L_ArrayENCO_S[0],3,L_ENCO_D);

- The lower 3 bits of L_WORD1 are encoded when X20 turns ON, and the results are stored to L_WORD2. Example of the program using a label

[Structured ladder/FBD]



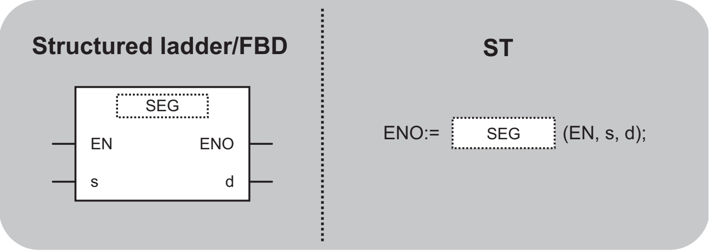
[ST]

ENCOP(X20,L_WORD1,3,L_WORD2);

7-segment decode

SEG(P)

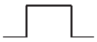
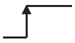
Basic **High performance** **Process** **Redundant** **Universal** **LCPU**



Any of the following instruction can go in the dotted squares.

SEG, SEGP

■Executing condition

Instruction	Executing condition
SEG	
SEGP	

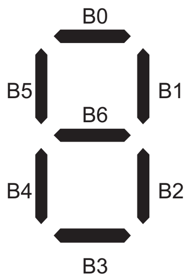
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Decoding data, or start number of the device that stores decoding data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the decoding result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○							○	—
(d)	○							—	—

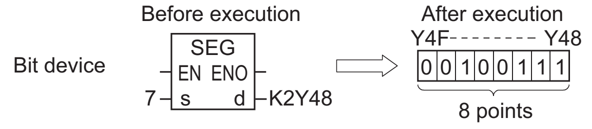
Processing details

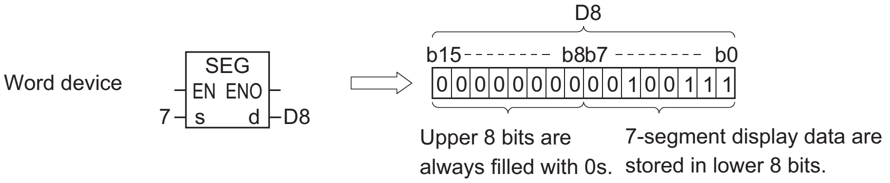
- Decodes the data from 0 to F specified for the lower 4 bits of (s) to 7-segment display data, and stores to (d).

(s)		Configuration of 7 segments	(d)								Display data
Hexadecimal	Bit pattern		B7	B6	B5	B4	B3	B2	B1	B0[*1]	
0	0000		0	0	1	1	1	1	1	1	0
1	0001		0	0	0	0	0	1	1	0	1
2	0010		0	1	0	1	1	0	1	1	2
3	0011		0	1	0	0	1	1	1	1	3
4	0100		0	1	1	0	0	1	1	0	4
5	0101		0	1	1	0	1	1	0	1	5
6	0110		0	1	1	1	1	1	0	1	6
7	0111		0	0	1	0	0	1	1	1	7
8	1000		0	1	1	1	1	1	1	1	8
9	1001		0	1	1	0	1	1	1	1	9
A	1010		0	1	1	1	0	1	1	1	A
B	1011		0	1	1	1	1	1	0	0	b
C	1100		0	0	1	1	1	0	0	1	C
D	1101		0	1	0	1	1	1	1	0	d
E	1110		0	1	1	1	1	0	0	1	E
F	1111		0	1	1	1	0	0	0	1	F

[*1] Start number of bit device, or the lowest bit of word device

- If (d) is a bit device, indicates the start number of the devices storing the 7-segment display data, and if it is a word device, indicates the number of the device storing the data.




Word device
SEG
EN ENO
7
s
d
D8
D8
b15
b8b7
b0
0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1
Upper 8 bits are always filled with 0s.
7-segment display data are stored in lower 8 bits.

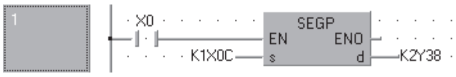
Operation error

- There is no operation error.

Program example

- In the following program, the data from XC to XF are converted to 7-segment display data and the results are output to the devices from Y38 to Y3F when X0 turns ON.

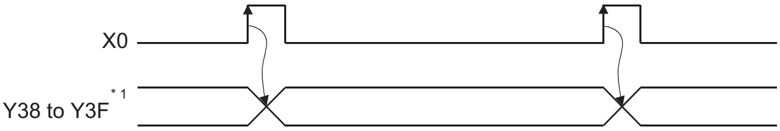
[Structured ladder/FBD]



[ST]

```
SEGP(X0,K1X0C,K2Y38);
```

[Timing chart]

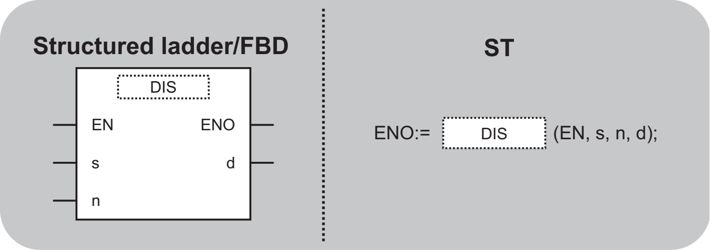


*1: The data Y38 to Y3F will not change until the next data are output.

4-bit separation of 16-bit data

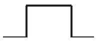
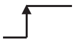
DIS(P)

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.
DIS, DISP

■Executing condition

Instruction	Executing condition
DIS	
DISP	

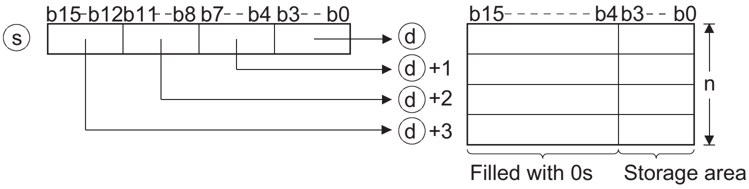
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores data to be separated	ANY16
	n	Number of separations (1 to 4), no processing on 0	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores separated data	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○	○		○					—
n	○	○		○					—
(d)	—	○		—					—

Processing details

- Stores the lower n digits (1 digit is 4 bits) of the 16-bit data specified for (s) to the lower 4 bits n points from the device specified for (d).



- The upper 12 bits n points from the device specified for (d) become 0.
- The value of n can be specified between 1 and 4. When 0 is specified, no processing will be performed and the content of (d) will not change.
- If n = 0, there will be no processing, and the content of n points from (d) will not change.

Operation error

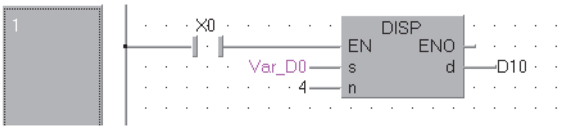
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value for n is not between 0 and 4.	○	○	○	○	○	○
4101	The range of n points from the device specified for (d) exceeds the corresponding device.	○	○	○	○	○	○

Program example

- In the following program, the 16-bit data in Var_D0 are separated into 4-bit groups, and the results are stored to the devices from D10 to D13 when X0 turns ON.

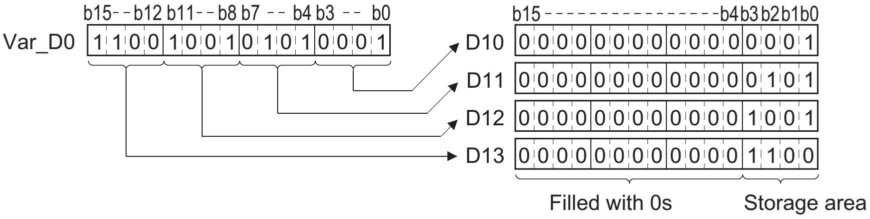
[Structured ladder/FBD]



[ST]

DISP(X0,Var_D0,4,D10);

[Operation]


b15- -b12 b11- -b8 b7 - - b4 b3 - - b0
Var_D0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 1
b15 - - - - - - - - - - - - - - - -b4 b3 b2 b1 b0
D10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
D11 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
D12 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1
D13 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
Filled with 0s
Storage area

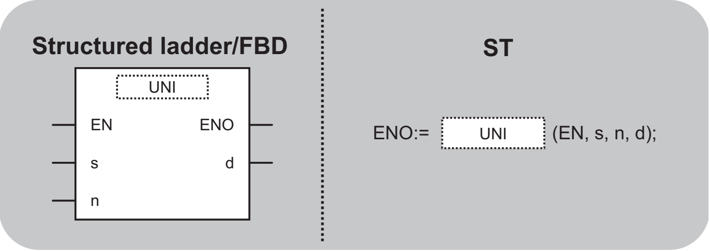
Precautions

When using the DIS instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

4-bit connection of 16-bit data

UNI(P)

Basic | High performance | Process | Redundant | Universal | LCPU

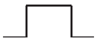
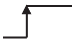


ENO:= UNI (EN, s, n, d);

Any of the following instruction can go in the dotted squares.

UNI, UNIP

■Executing condition

Instruction	Executing condition
UNI	
UNIP	

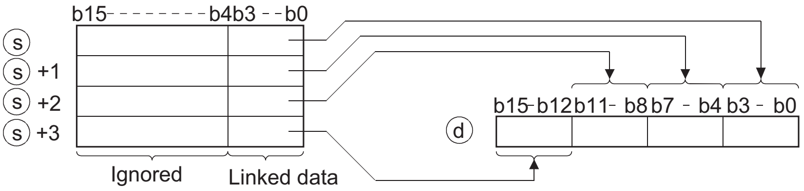
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores data to be connected	ANY16
	n	Number of connections (1 to 4), no processing on 0	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores connected data	ANY16

Setting data	Internal device		R, ZR	J□\□		□U\□G	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				—	—
n	○	○		○				○	—
(d)	○	○		○				—	—

Processing details

- Connects lower 4 bits of 16-bit data n points from device specified for (s) to 16-bit device specified for (d).



- The bits of the upper (4 - n.) digits of the device specified for (d) become 0.
- The value of n can be specified between 1 and 4.
- If n = 0, there will be no processing, and the content of device (d) will not change.

Operation error

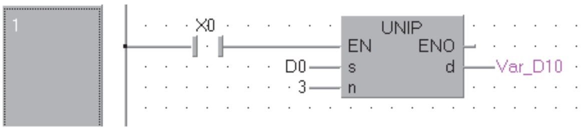
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value for n is not between 0 and 4.	○	○	○	○	○	○
4101	The range of n points from the device specified for (s) exceeds the corresponding device.	○	○	○	○	○	○

Program example

- In the following program, the lower 4 bits of the devices from D0 to D2 are connected when X0 turns ON, and they are stored to Var_D10.

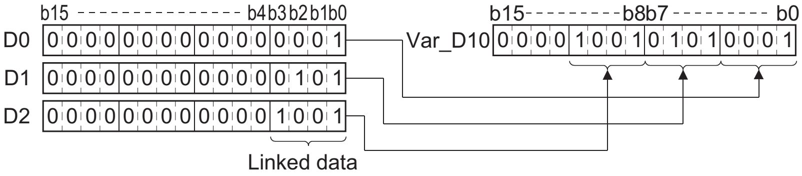
[Structured ladder/FBD]



[ST]

UNIP(X0,D0,3,Var_D10);

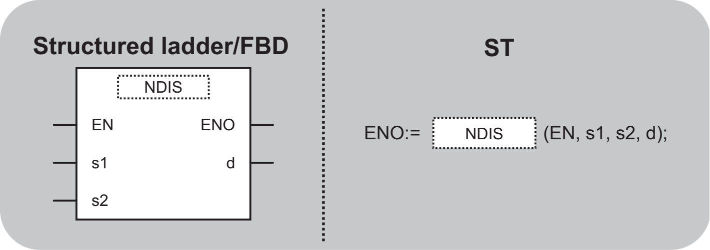
[Operation]



Separation and connection of random data

NDIS(P), NUNI(P)

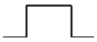
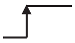
Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

NDIS, NDISP, NUNI, NUNIP

■Executing condition

Instruction	Executing condition
NDIS, NUNI	
NDISP, NUNIP	

■Argument

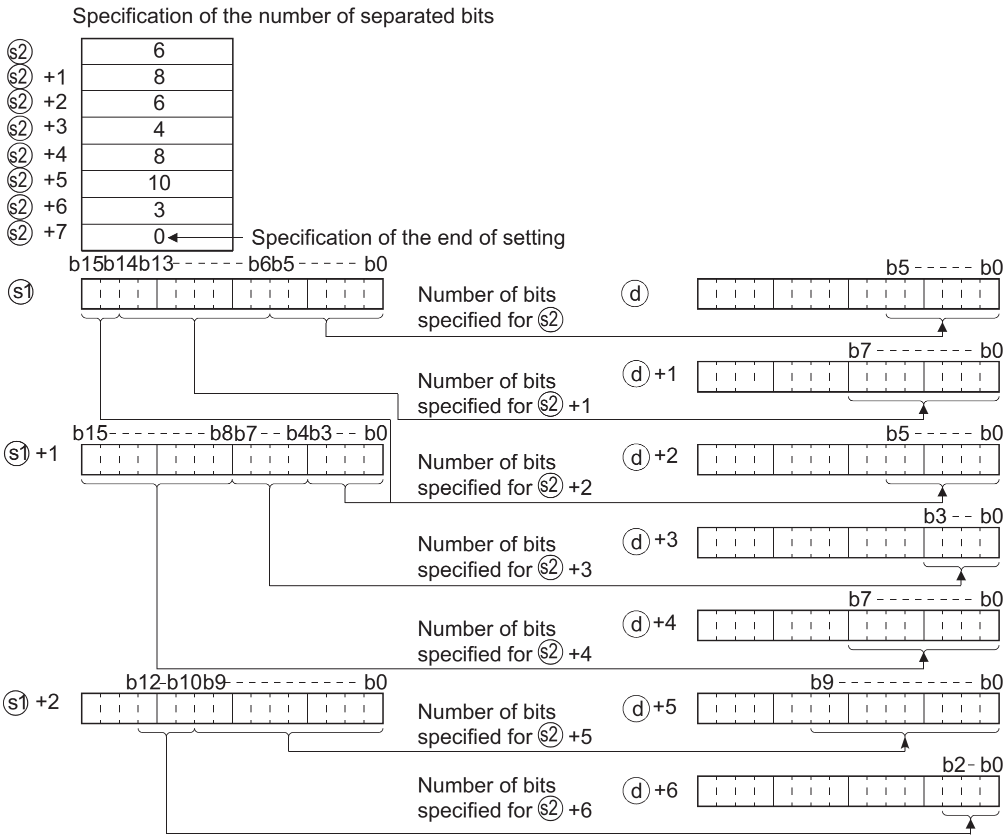
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores data to be separated or connected	ANY16
	s2	Start number of the device that stores units of separations and connections	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores separated or connected data	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—					
(s2)	—	○		—					
(d)	—	○		—					

Processing details

■NDIS(P)

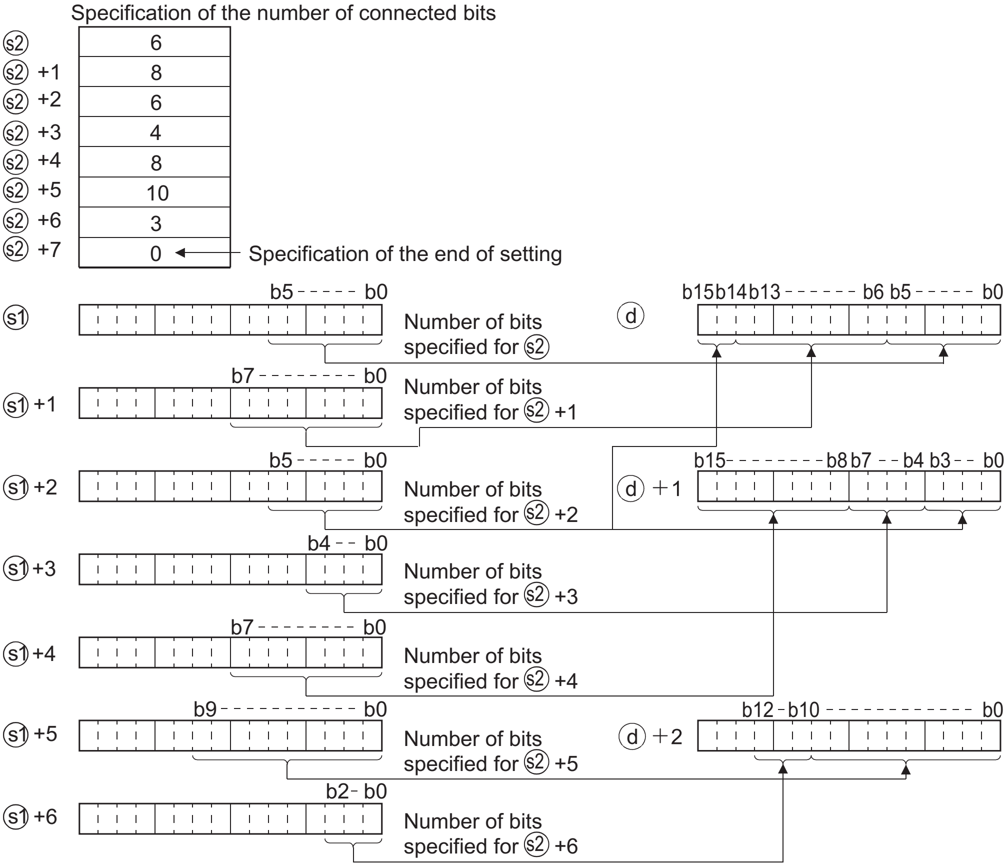
- Separates each bit of data stored in the device number specified for (s1) and the following devices in blocks whose numbers of bits are specified for (s2), and stores the data to the device number specified for (d) and the following devices.


Specification of the number of separated bits
s2 6
s2 +1 8
s2 +2 6
s2 +3 4
s2 +4 8
s2 +5 10
s2 +6 3
s2 +7 0
Specification of the end of setting
b15b14b13 b6b5 b0
s1
Number of bits specified for s2
d
b5 b0
Number of bits specified for s2 +1
d +1
b7 b0
b15 b8b7 b4b3 b0
s1 +1
Number of bits specified for s2 +2
d +2
b5 b0
Number of bits specified for s2 +3
d +3
b3 b0
Number of bits specified for s2 +4
d +4
b7 b0
b12 b10b9 b0
s1 +2
Number of bits specified for s2 +5
d +5
b9 b0
Number of bits specified for s2 +6
d +6
b2 b0

- The number of separated bits specified for (s2) can be specified within a range of 1 to 16 bits.
- Bits from the device number specified for (s2) to the device number where 0 is stored are processed as separated bits.
- Do not overlap the device range for data to be separated ((s1) to end range of (s1)) with the device range which stores the separated data ((d) to end range of (d)). If overlapped, the correct operation result may not be obtained.
- Do not specify the same device number for (s1), (s2), and (d). If the same device is specified for (s1), (s2), and (d), the operation does not perform correctly.

■NUNI(P)

- Connects each bit of data stored in the device number specified for (s1) and the following devices in blocks whose numbers of bits are specified for (s2), and stores them to the device number specified for (d) and the following devices.



Specification of the number of connected bits
s2 6
s2 +1 8
s2 +2 6
s2 +3 4
s2 +4 8
s2 +5 10
s2 +6 3
s2 +7 0
Specification of the end of setting
s1
b5 - - - - - b0
Number of bits specified for s2
s1 +1
b7 - - - - - - - - b0
Number of bits specified for s2 +1
s1 +2
b5 - - - - - b0
Number of bits specified for s2 +2
s1 +3
b4 - - b0
Number of bits specified for s2 +3
s1 +4
b7 - - - - - - - - b0
Number of bits specified for s2 +4
s1 +5
b9 - - - - - - - - - - - - b0
Number of bits specified for s2 +5
s1 +6
b2 - b0
Number of bits specified for s2 +6
d
b15b14b13 - - - - - - b6 b5 - - - - - b0
d +1
b15 - - - - - - - - b8 b7 - - b4 b3 - - b0
d +2
b12 - b10 - - - - - - - - - - - - b0

- The number of connected bits specified for (s2) can be within a range of from 1 to 16 bits.
- Processing will be performed on the number of bits to be connected from the device number specified for (s2) to the device number storing 0.
- Do not overlap the device range for data to be connected ((s1) to end range of (s1)) with the device range which stores the connected data ((d) to end range of (d)). If overlapped, the correct operation result may not be obtained.
- Do not specify the same device number for (s1), (s2), and (d). If the same device is specified for (s1), (s2), and (d), the operation does not perform correctly.

Operation error

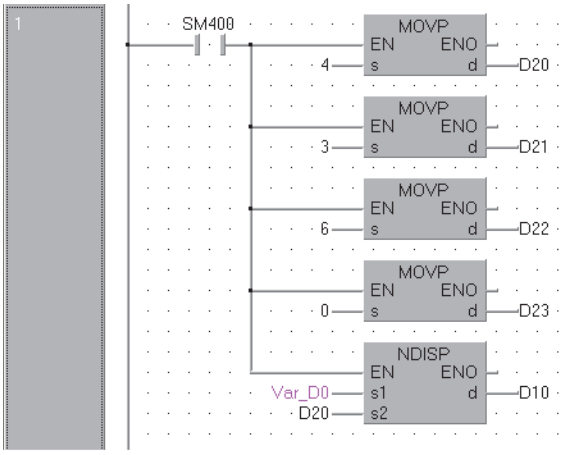
- n any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The number of separated or connected bits specified for (s2) is not set within the range from 1 to 16 bits.	○	○	○	○	○	○
4101	The device use range specified for (s1) or (d) exceeds the final device number of their respective devices because of the number of separated or connected bits specified for (s2).	○	○	○	○	○	○

Program example

- In the following program, the data of 4, 3, and 6 bits respectively from the lower bits of Var_D0 are separated, and they are stored to the devices from D10 to D12.

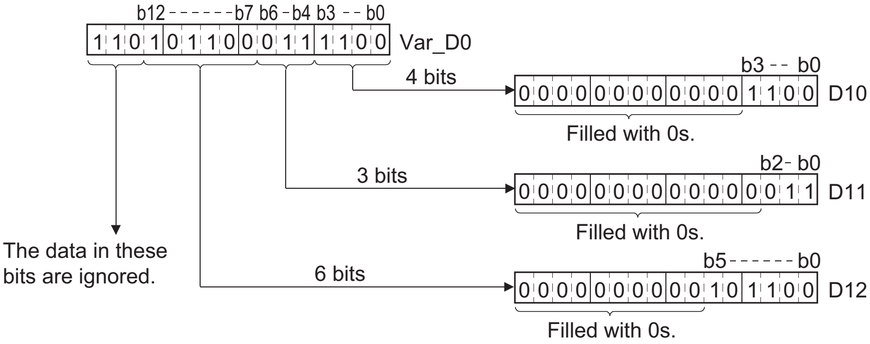
[Structured ladder/FBD]



[ST]

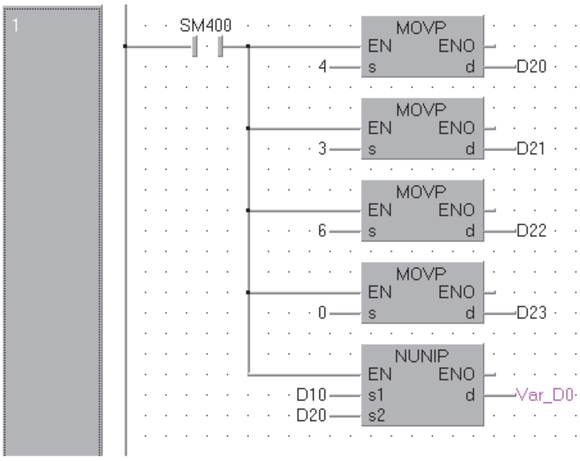
```
MOVP(SM400,4,D20);
MOVP(SM400,3,D21);
MOVP(SM400,6,D22);
MOVP(SM400,0,D23);
NDISP(SM400,Var_D0,D20,D10);
```

[Operation]



- In the following program, the lower 4 bits of data in D10, the lower 3 bits of data in D11, and the lower 6 bits of data in D12 are connected, and they are stored to Var_D0.

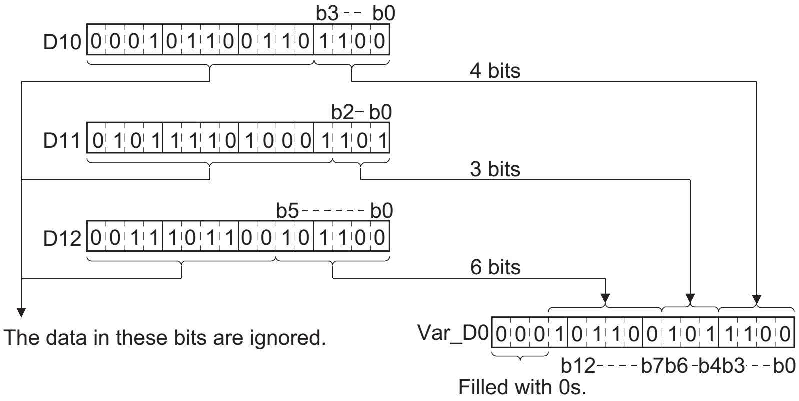
[Structured ladder/FBD]



[ST]

```
MOVP(SM400,4,D20);
MOVP(SM400,3,D21);
MOVP(SM400,6,D22);
MOVP(SM400,0,D23);
NUNIP(SM400,D10,D20,Var_D0);
```

[Operation]

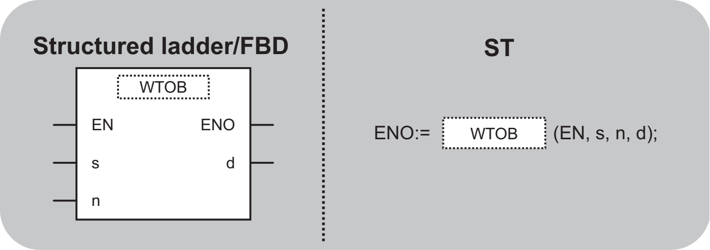


b3 - - b0
D10 0 0 0 1 0 1 1 0 0 1 1 0 1 1 0 0
4 bits
b2 - b0
D11 0 1 0 1 1 1 1 0 1 0 0 0 1 1 0 1
3 bits
b5 - - - - - - b0
D12 0 0 1 1 1 0 1 1 0 0 1 0 1 1 0 0
6 bits
The data in these bits are ignored.
Var_D0 0 0 0 1 0 1 1 0 0 1 0 1 1 1 0 0
b12 - - - - b7 b6 - b4 b3 - - b0
Filled with 0s.

Separation and connection of data in units of bytes

WTOB(P), BTOW(P)

Basic | High performance | Process | Redundant | Universal | LCPU

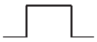
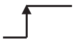


```
ENO:= WTOB (EN, s, n, d);
```

Any of the following instruction can go in the dotted squares.

WTOB, WTOBP, BTOW, BTOWP

■Executing condition

Instruction	Executing condition
WTOB, BTOW	
WTOBP, BTOWP	

■Argument

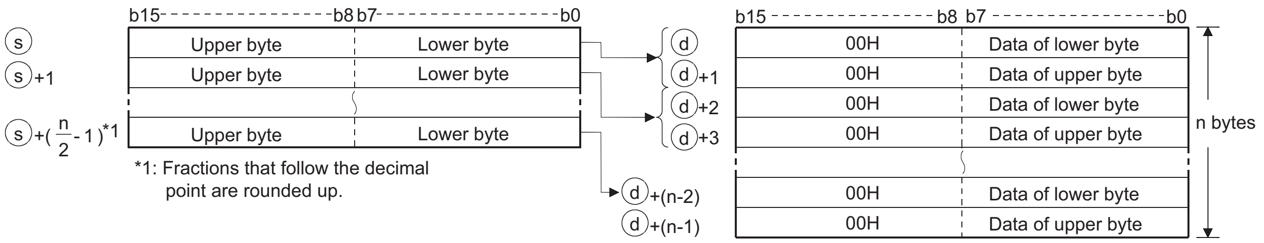
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores data to be separated or connected in unit of bytes	ANY16
	n	Number of byte data to be separated or connected	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the separation or connection result in unit of bytes	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—					—
n	○	○		○					—
(d)	—	○		—					—

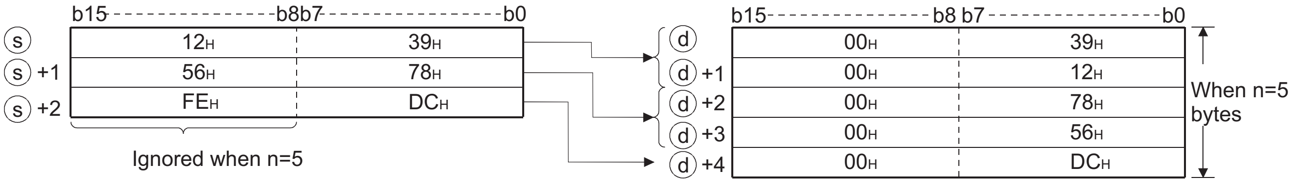
Processing details

■WTOB(P)

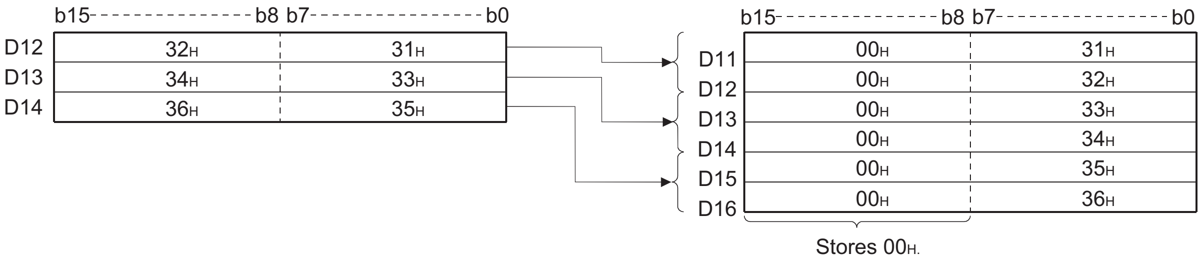
- Separates n bytes of the 16-bit data stored in the device specified for (s) and the following devices, and stores them to the device specified for (d) and the following devices.



For example, if n = 5, data through the lower 8 bits of (s) to ((s)+2) would be stored to (d) to ((d)+4).



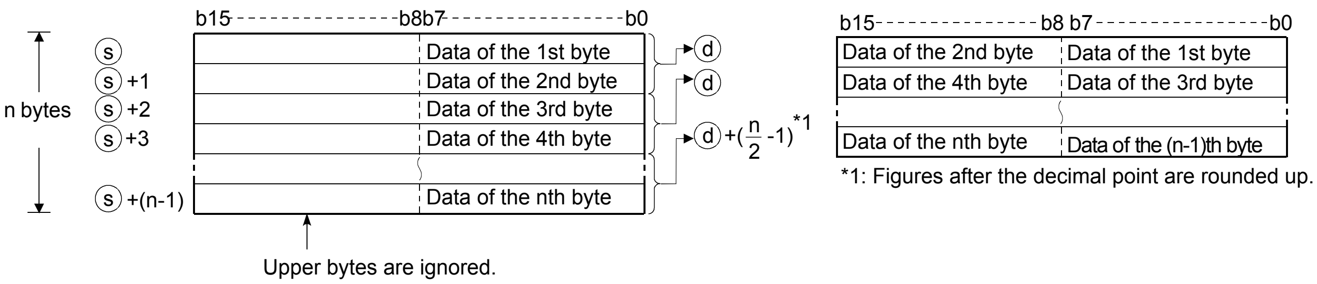
- Setting the number of bytes with n automatically determines the range of the 16-bit data specified for (s) and the range of the devices to store the byte data specified for (d).
- No processing will be performed when the number of bytes specified for n is 0.
- The 00H code will automatically be stored to the upper 8 bits of the byte storage device specified for (d).



- Separation is correctly processed even when the device range ((s) to (s) + (n-1)) where the data to be separated is stored overlaps with the device range ((d) to (d) + (n/2)-1)) where the separated data will be stored.

■BTOW(P)

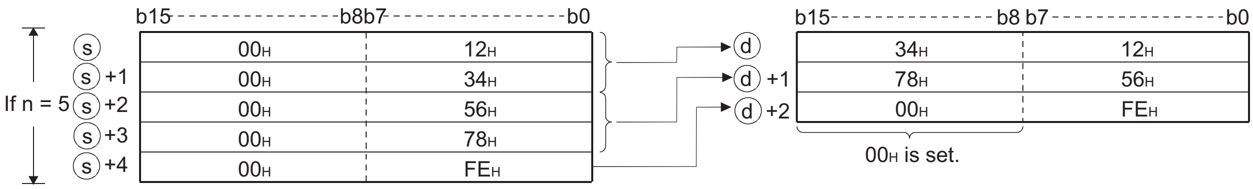
- Connects the lower 8 bits of the 16-bit data in n words stored in the device specified for (s) and the following devices in units of 1-word and stores it to the device specified for (d) and the following devices. The upper 8 bits of n-word data stored in the device specified for (s) and the following devices will be ignored. Further, if n is an odd number, 0 is stored to the upper 8 bits of the device where the nth byte data are stored.



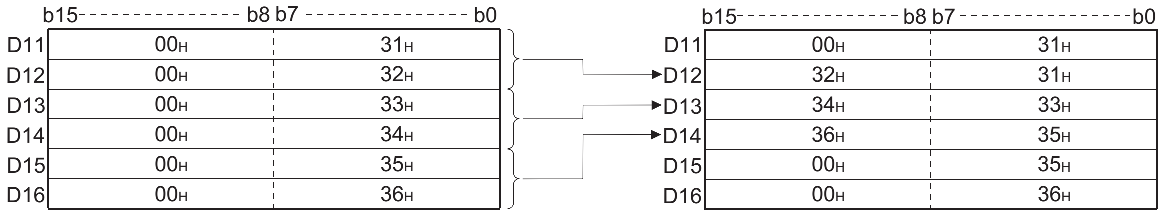
*1: Figures after the decimal point are rounded up.

Upper bytes are ignored.

For example, if n = 5, the lower 8-bit data from (s) to ((s)+4) are connected and stored to (d) to ((d)+2).



- Setting the number of bytes with n automatically determines the range of the byte data specified for (s) and the range of the devices to store the connected data specified for (d).
- No processing will be performed when the number of bytes specified for n is 0.
- The upper 8 bits of the byte data storage device specified for (s) are ignored, and the lower 8 bits are used.
- Connection is correctly processed even when the device range ((s) to (s)+(n-1)) where the data to be connected is stored overlaps with the device range ((d) to (d)+((n/2)-1)) where the connected data will be stored. For example, the following will take place in a case where the lower 8 bits of D11 to D16 are to be stored to D12 to D14.



Operation error

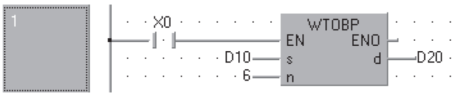
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of the number of bytes specified for n following the device number specified for (s) exceeds the corresponding device range. The range of the number of bytes specified for n following the device number specified for (d) exceeds the corresponding device range.	○	○	○	○	○	○

Program example

- In the following program, the data from D10 to D12 are separated in units of bytes when X0 turns ON, and they are stored to the devices from D20 to D25.

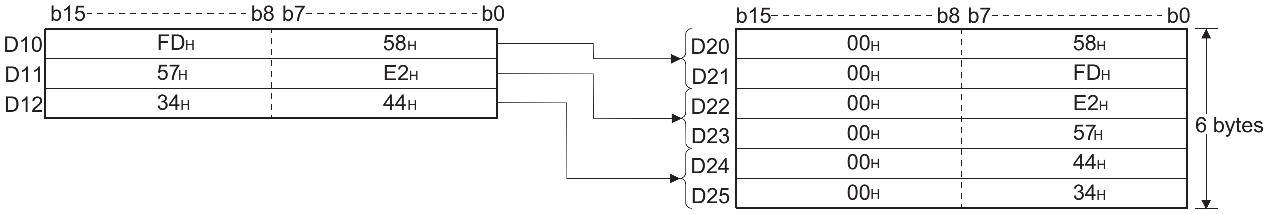
[Structured ladder/FBD]



[ST]

```
WTOBP(X0,D10,6,D20);
```

[Operation]

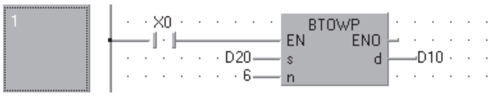


	b15–b8	b7–b0
D10	FDH	58H
D11	57H	E2H
D12	34H	44H

	b15–b8	b7–b0
D20	00H	58H
D21	00H	FDH
D22	00H	E2H
D23	00H	57H
D24	00H	44H
D25	00H	34H

- In the following program, the lower 8-bit data from D20 to D25 are connected when X0 turns ON, and they are stored to the devices from D10 to D12.

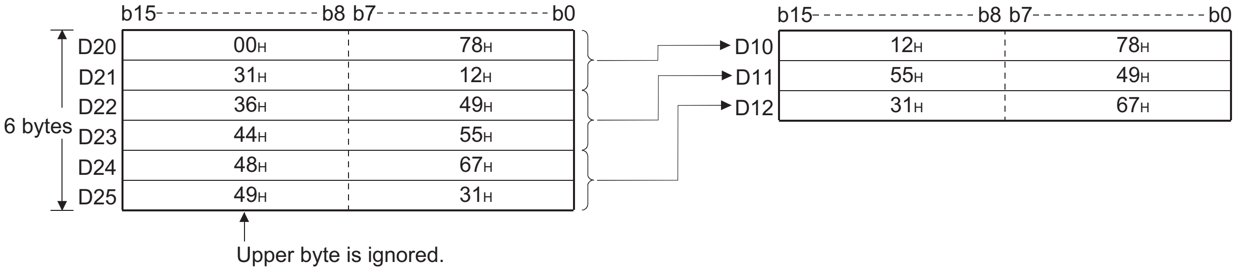
[Structured ladder/FBD]



[ST]

```
BTOWP(X0,D20,6,D10);
```

[Operation]



	b15–b8	b7–b0
D20	00H	78H
D21	31H	12H
D22	36H	49H
D23	44H	55H
D24	48H	67H
D25	49H	31H

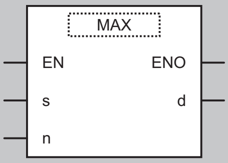
	b15–b8	b7–b0
D10	12H	78H
D11	55H	49H
D12	31H	67H

Maximum value search of 16-/32-bit data

MAX(P), DMAX(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



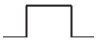
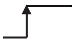
ST

ENO:= MAX (EN, s, n, d);

Any of the following instruction can go in the dotted squares.

MAX, MAXP, DMAX, DMAXP

■Executing condition

Instruction	Executing condition
MAX, DMAX	
MAXP, DMAXP	

■Argument

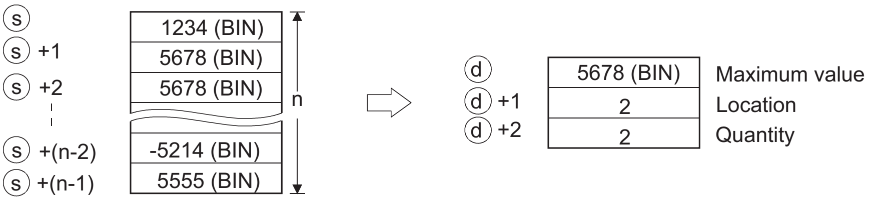
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device in which the maximum value is searched	ANY16/32
	n	Number of search data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the search result of the maximum value	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—					—
n	○	○		○					—
(d)	—	○		—					—

Processing details

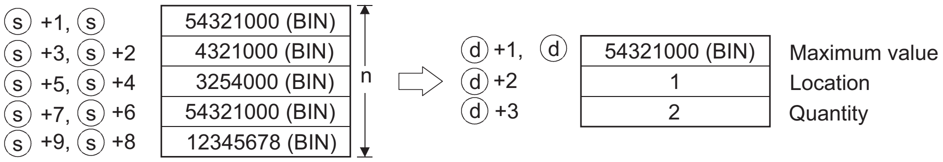
■MAX(P)

- Searches for the maximum value from the n points of 16-bit BIN data in the device specified for (s), and stores the searched maximum value to the device specified for (d). Searches from the device specified for (s) and counted from (s), stores the location of device number which is stored in the maximum value of the first search to (d)+1 and stores the number of maximum values to (d)+2.



■DMAX(P)

- Searches for the maximum value in the n points of 32-bit BIN data, from the device specified for (s), and stores the searched maximum value to the device specified for (d) and (d)+1. Searches from the device specified for (s) and counted from (s), stores the location of device number which is stored in the maximum value of the first search to (d)+2 and stores the number of maximum values to (d)+3.



Operation error

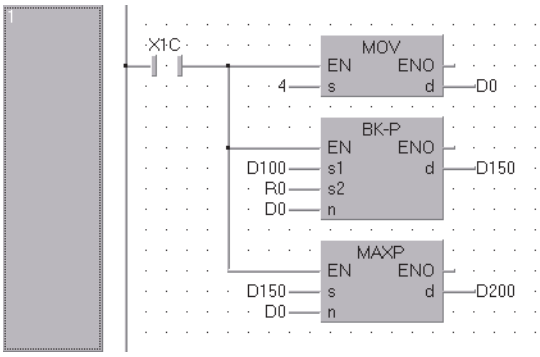
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (s) exceeds the corresponding device.	○	○	○	○	○	○
4101	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- n the following program, the data from D100 to D103 are subtracted from the data from R0 to R3 when X1C turns ON, and the maximum value is searched in the subtraction results, then the search results are stored to the devices from D200 to D202.

[Structured ladder/FBD]



[ST]

```
IF X1C THEN
        D0:=4;
        D150:=D100-R0;
        D151:=D101-R1;
        D152:=D102-R2;
        D153:=D103-R3;
        MAXP(TRUE,D150,D0,D200);
END_IF;
```

[Operation]

	b15-------- b0
D100	4321 (BIN)
D101	5432 (BIN)
D102	4444 (BIN)
D103	5000 (BIN)

−

	b15-------- b0
R0	5000 (BIN)
R1	4000 (BIN)
R2	4000 (BIN)
R3	6000 (BIN)

⇨

	b15-------- b0
D150	-679 (BIN)
D151	1432 (BIN)
D152	444 (BIN)
D153	-1000 (BIN)

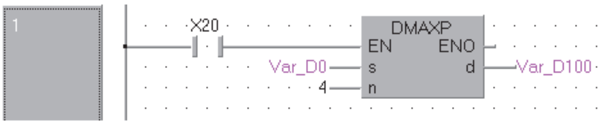
⇨

D200	1432	Maximum value
D201	2	Location
D202	1	Quantity

D0	4

- In the following program, the maximum value is searched from 32-bit data in the data 4 points from Var_D0 and stores the result to Var_D100 when X20 turns ON.

[Structured ladder/FBD]



[ST]

```
DMAXP(X20,Var_D0,4,Var_D100);
```

[Operation]

Var_D0	3786213 (BIN)
	− 3235 (BIN)
	8744740 (BIN)
	7141821 (BIN)

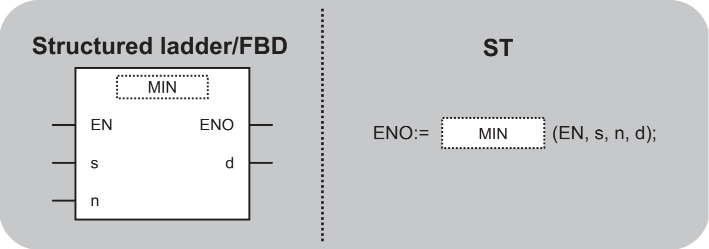
⇨

Var_D100	8744740
	3
	1

Minimum value search of 16-/32-bit data

MIN(P), DMIN(P)

Basic | High performance | Process | Redundant | Universal | LCPU

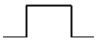
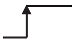


```
ENO:= MIN (EN, s, n, d);
```

Any of the following instruction can go in the dotted squares.

MIN, MINP, DMIN, DMINP

■Executing condition

Instruction	Executing condition
MIN, DMIN	
MINP, DMINP	

■Argument

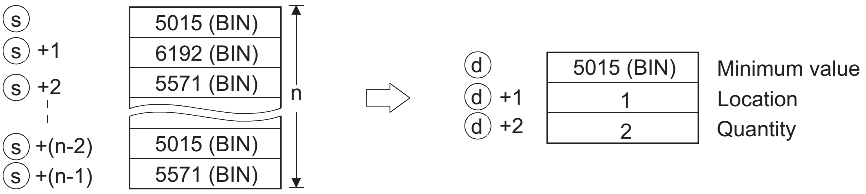
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device in which the minimum value is searched	ANY16/32
	n	Number of search data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the search result of the minimum value	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—					—
n	○	○		○					—
(d)	—	○		—					—

Processing details

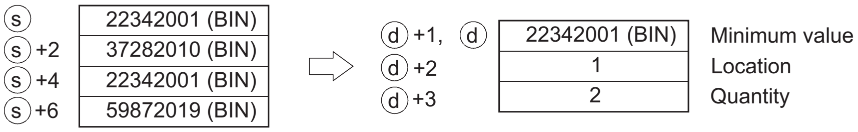
■MIN(P)

- Searches for the minimum value in the n points of 16-bit BIN data, from the device specified for (s), and stores searched minimum value to the device specified for (d). Searches from the device specified for (s) and counted from (s), stores the location of device number which is stored in the minimum value of the first search to (d)+1 and stores the number of minimum values to (d)+2.



■DMIN(P)

- Searches for the minimum value in the n points of 32-bit BIN data, from the device specified for (s), and stores searched minimum value to the devices specified for (d) and (d)+1. Searches from the device specified for (s) and counted from (s), stores the location of the device number which is stored in the minimum value of the first search to (d)+2 and stores the number of minimum values to (d)+3.



Operation error

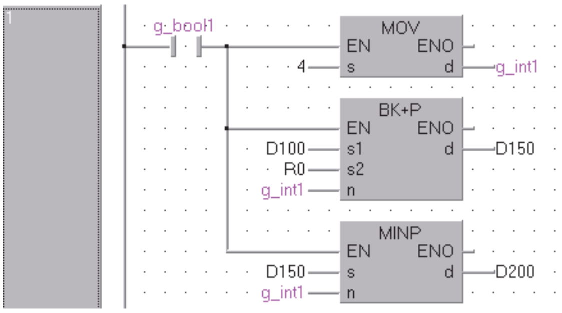
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (s) exceeds the corresponding device.	—	—	—	—	○	○
	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the data from D100 to D103 and the data from R0 to R3 are added when g_bool1 turns ON, and the minimum value is searched in the addition result, then the search results are stored to the devices from D200 to D202.

[Structured ladder/FBD]



[ST]

```
IF g_bool1 THEN
        g_int1:=4;
        D150:=D100+R0;
        D151:=D101+R1;
        D152:=D102+R2;
        D153:=D103+R3;
        MINP(TRUE,D150,g_int1,D200);
END_IF;
```

[Operation]

	b15-------- b0			b15-------- b0			b15-------- b0
D100	5542 (BIN)		R0	5500 (BIN)		D150	11042 (BIN)
D101	5857 (BIN)	+	R1	4000 (BIN)	⇨	D151	9857 (BIN)
D102	4590 (BIN)		R2	4500 (BIN)		D152	9090 (BIN)
D103	4450 (BIN)		R3	6000 (BIN)		D153	10450 (BIN)

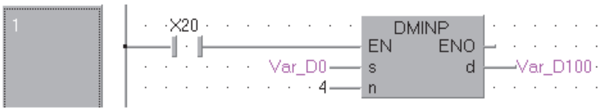
D0	4

⇨

D200	9090	Minimum value
D201	3	Location
D202	1	Quantity

- In the following program, the minimum value is searched from 32-bit data in the data 4 points from Var_D0 (D0) and stores the result to Var_D100 (D100) when X20 turns ON.

[Structured ladder/FBD]



[ST]

```
DMINP(X20,Var_D0,4,Var_D100);
```

[Operation]

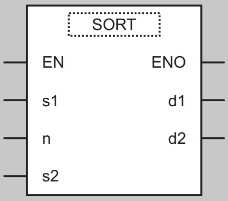
Var_D0			Var_D100	
	57020175 (BIN)			−69386
	2070166 (BIN)	⇨		4
	3596045 (BIN)			1
	−69386 (BIN)			

Sorting 16-/32-bit data

SORT, DSORT

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



ST

ENO:= SORT (EN, s1, n, s2, d1, d2);

Any of the following instruction can go in the dotted squares.

SORT, DSORT

■Executing condition

Instruction	Executing condition
SORT, DSORT	⎍

■Argument

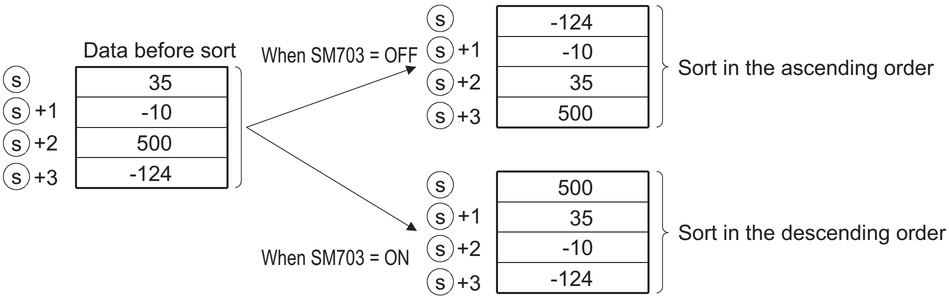
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of table to be sorted	ANY16/32
	n	Number of sort data	ANY16
	s2	Number of data to be compared in one sort execution	ANY16
Output argument	ENO	Execution result	Bit
	d1	Bit device number to be turned on at sort completion	Bit
	d2	Device for system use	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—					—
n	○	○		○					—
(s2)	○	○		○					—
(d1)	○	—		—					—
(d2)	—	○		—					—

Processing details

■SORT

- Sorts 16-bit BIN data n points from (s1) in ascending or descending order. Sort order is specified by the ON or OFF status of SM703.
  - When SM703 is OFF: Sorts in ascending order
  - When SM703 is ON: Sorts in descending order



- Several scans are required for sorts performed by the SORT instruction. The number of scans executed until completion is the value obtained by dividing the maximum number of times executed until the completion of the sort by the number of data blocks compared at one execution specified for (s2). (Fractional part is rounded up.) When the value of (s2) is increased, the number of scans until completion of the sort is reduced, but the amount of time per scan is lengthened.
- The maximum number of executions until completion of the sort should be calculated according to the following equation.

Ex.

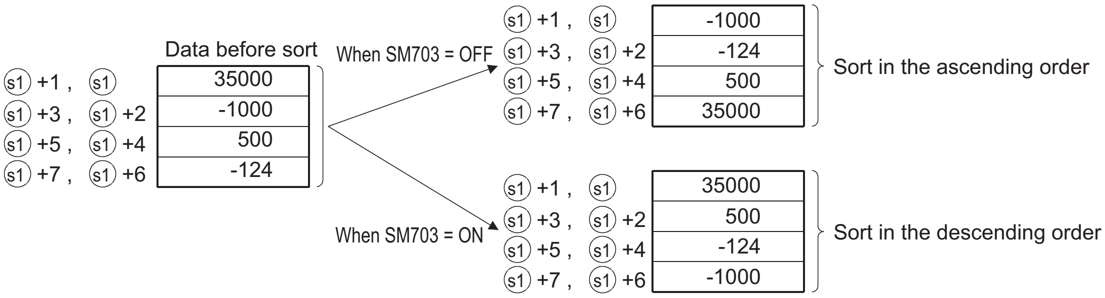
Maximum number of executions until completion = (n) × (n - 1) ÷ 2 [times executed]

When n = 10, the number of executions is obtained as 10 × (10 - 1) ÷ 2 = 45 [times executed]. If (s2) = 2, then the number of scans until the completion of sort is calculated as 45 ÷ 2 = 22.5 → 23 [scans].

- The device specified for (d1) (the completion device) is turned OFF by the execution of the SORT instruction, and turned ON when the sort is completed. Because the device specified for (d1) is retained in the ON state after the completion of the sort, the user must turn it OFF if required.
- The 2 points from the device specified for (d2) are used by the system during the execution of the SORT instruction. These 2 points from the device specified for (d2) should therefore not be used by the user. Changing these points may cause an error code to be returned. (Error code: 4100)
- If the value of n is changed during the execution of the SORT instruction, the sort will be performed in accordance with the number of sort data blocks after the change.
- If the execution command is turned OFF during the execution of the SORT instruction, the sort is suspended. The sort resumes from the beginning when the execution command is turned ON again.
- To execute another sort operation immediately after the completion of the previous sort, turn OFF the execution command once, then turn it ON.

■DSORT

- Sorts 32-bit BIN data n points from (s1) in ascending or descending order. Sort order is specified by the ON or OFF status of SM703.
  - When SM703 is OFF: Sorts in ascending order
  - When SM703 is ON: Sorts in descending order



- Several scans are required for sorts performed by the DSORT instruction. The number of scans executed until completion is the value obtained by dividing the maximum number of times executed until the completion of the sort by the number of data blocks compared at one execution specified for (s2). (Fractional part is rounded up.) When the value of (s2) is increased, the number of scans until completion of the sort is reduced, but the amount of time per scan is lengthened.
- The maximum number of executions until completion of the sort should be calculated according to the following equation.

Ex.

Maximum number of executions until completion = (n) × (n - 1) ÷ 2 [times executed]

When n = 10, the number of executions is obtained as 10 × (10 - 1) ÷ 2 = 45 [times executed]. If (s2) = 2, then the number of scans until the completion of sort is calculated as 45 ÷ 2 = 22.5 → 23 [scans].

- The device specified for (d1) (the completion device) is turned OFF by the execution of the DSORT instruction, and turned ON when the sort is completed. Because the device specified for (d1) is retained in the ON state after the completion of the sort, the user must turn it OFF if required.
- The 2 points from the device specified for (d2) are used by the system during the execution of a DSORT instruction. These 2 points from the device specified for (d2) should therefore not be used by the user. Changing these points may cause an error code to be returned. (Error code: 4100)
- If the value of n is changed during the execution of the DSORT instruction, the sort will be performed in accordance with the number of sort data blocks after the change.
- If the execution command is turned OFF during the execution of the DSORT instruction, the sort is suspended. The sort resumes from the beginning when the execution command is turned ON again.
- To execute another sort operation immediately after the completion of the previous sort, turn OFF the execution command once, then turn it ON.

Operation error

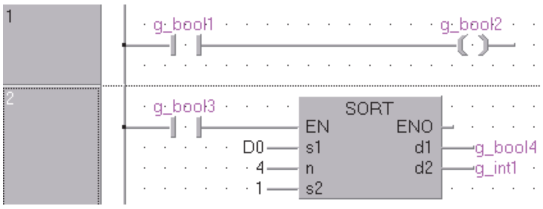
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	(s2) is 0 or a negative value. In the second and subsequent scan, the value of (d2) to be used in the system is n or more. In the second and subsequent scan, the value of (d2) to be used in the system is (d2) < (d2) + 1.	○	○	○	○	○	○
4101	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.	○	○	○	○	○	○
	For the SORT(P) instruction, the range of n points from (s1) exceeds the corresponding device range.	○	○	○	○	○	○
	For the DSORT(P) instruction, the range of 2 × n points from (s1) exceeds the corresponding device range.	○	○	○	○	○	○

Program example

- In the following program, the 16-bit BIN data from D0 to D3 are sorted in the ascending or descending order when g_bool1 turns ON.

[Structured ladder/FBD]

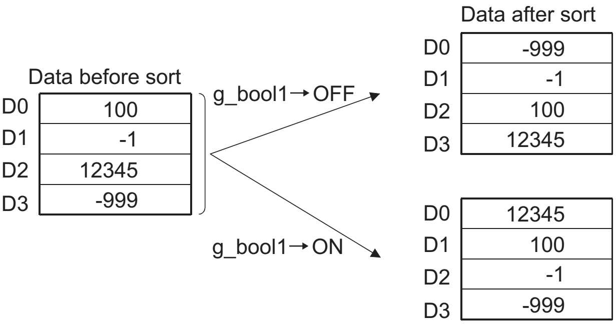


[ST]

OUT(g_bool1,g_bool2);

SORT(g_boo3,D0.4,1,g_bool4,g_int1);

[Operation]



Data before sort

D0	100
D1	-1
D2	12345
D3	-999

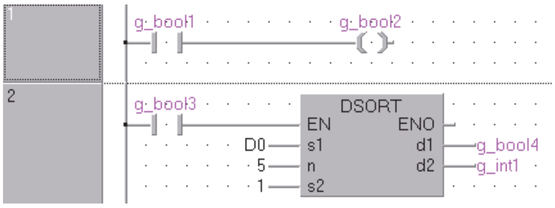
Data after sort

D0	-999
D1	-1
D2	100
D3	12345

D0	12345
D1	100
D2	-1
D3	-999

- In the following program, the 32-bit BIN data from D0 to D9 are sorted in the ascending or descending order when g_bool1 turns ON.

[Structured ladder/FBD]

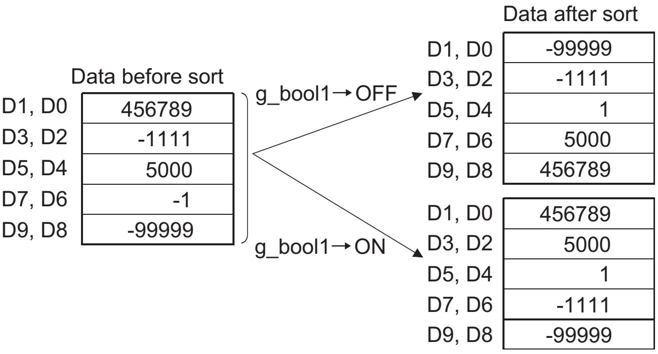


[ST]

OUT(g_bool1,g_bool2);

DSORT(g_boo3,D0.5,1,g_bool4,g_int1);

[Operation]



Data before sort

D1, D0	456789
D3, D2	-1111
D5, D4	5000
D7, D6	-1
D9, D8	-99999

Data after sort

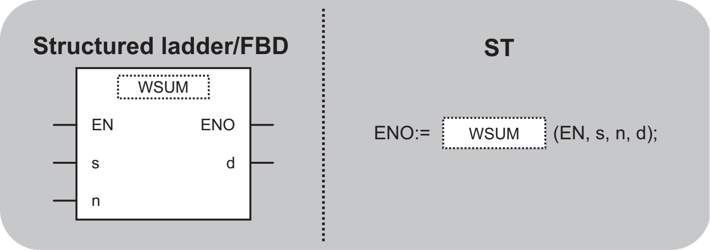
D1, D0	-99999
D3, D2	-1111
D5, D4	1
D7, D6	5000
D9, D8	456789

D1, D0	456789
D3, D2	5000
D5, D4	1
D7, D6	-1111
D9, D8	-99999

Total calculation of 16-bit data

WSUM(P)

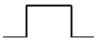
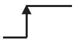
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

WSUM, WSUMP

■Executing condition

Instruction	Executing condition
WSUM	
WSUMP	

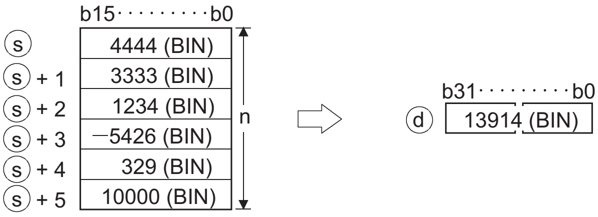
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores data whose total value is evaluated	ANY16
	n	Number of data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the total value	ANY32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				—	—
n	○	○		○				○	—
(d)	○	○		○				—	—

Processing details

- Adds all16-bit BIN data of n points from the device specified for (s), and stores the result to the device specified for (d).



Operation error

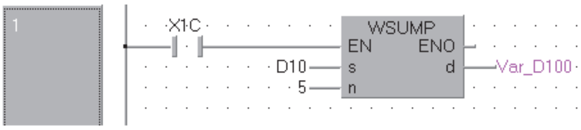
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (s) exceeds the corresponding device.	○	○	○	○	○	○

Program example

- In the following program, the 16-bit BIN data from D10 to D14 are added when X1C turns ON, and the result is stored to Var_D100.

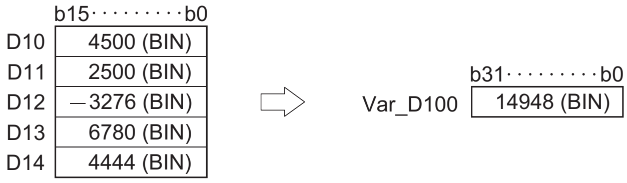
[Structured ladder/FBD]



[ST]

WSUMP(X1C,D10,5,Var_D100);

[Operation]



	b15·········b0
D10	4500 (BIN)
D11	2500 (BIN)
D12	−3276 (BIN)
D13	6780 (BIN)
D14	4444 (BIN)

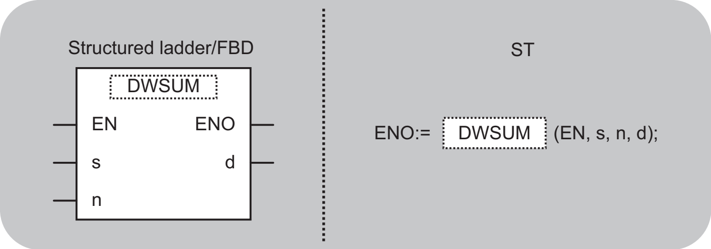
⇨

	b31·········b0
Var_D100	14948 (BIN)

Total calculation of 32-bit data

DWSUM(P)

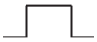
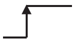
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

DWSUM, DWSUMP

■Executing condition

Instruction	Executing condition
DWSUM	
DWSUMP	

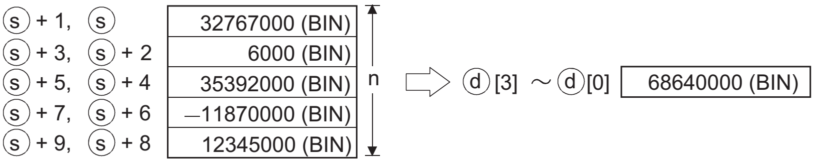
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores data whose total value is evaluated	ANY32
	n	Number of data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores the total value	Array of ANY16 (0..3)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				—	—
n	○	○		○				○	—
(d)	○	○		—				—	—

Processing details

- Adds all 32-bit BIN data n points from the device specified for (s), and stores the result to the 4 points of devices (4 words) from the one specified for (d).



Operation error

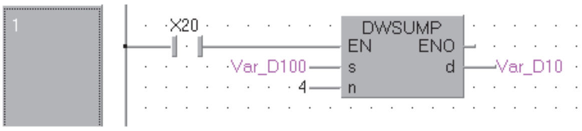
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The range of n points from the device specified for (s) exceeds the corresponding device.	○	○	○	○	○	○
	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the 32-bit BIN data in Var_D100 are added when X20 turns ON, and stores the result to Var_D10.

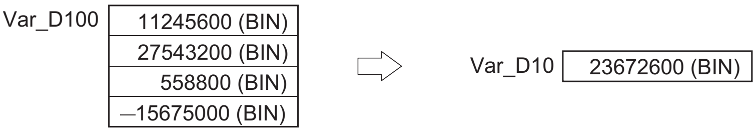
[Structured ladder/FBD]



[ST]

```
DWSUMP(X20,Var_D100,4,Var_D10);
```

[Operation]


Var_D100

11245600 (BIN)
27543200 (BIN)
558800 (BIN)
–15675000 (BIN)

Var_D10 23672600 (BIN)

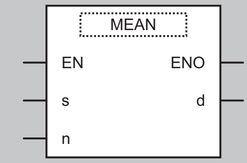
Average calculation of 16-/32-bit data

MEAN(P), DMEAN(P)

Basic High performance Process Redundant Ver. Universal LCPU

- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported

Structured ladder/FBD



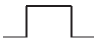
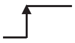
ST

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

Any of the following instruction can go in the dotted squares.

MEAN, MEANP, DMEAN, DMEANP

■Executing condition

Instruction	Executing condition
MEAN, DMEAN	
MEANP, DMEANP	

■Argument

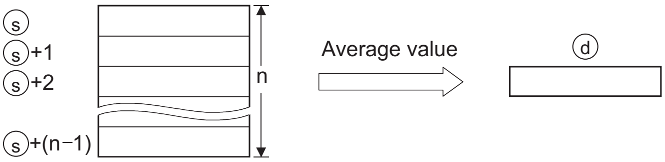
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores data whose average value is evaluated	ANY16/32
	n	Number of data, or device number that stores the number of data. (Setting range: 1 to 32767)	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the average value	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○	○	—				—	—
n	—	○	○	○				○	—
(d)	—	○	○	—				—	—

Processing details

■MEAN(P)

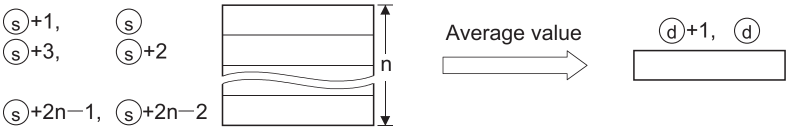
- Evaluates the average value of n points (16-bit BIN data) from the device specified for (s), and stores the result to the device specified for (d).



- If the result is not an integer value, the fractional part is rounded up.
- No processing is performed if the value specified for n is 0.

■DMEAN(P)

- Evaluates the average value of n points (32-bit BIN data) from the device specified for (s), and stores the result to the device specified for (d).



- If the result is not an integer value, the fractional part is rounded up.
- No processing is performed if the value specified for n is 0.

Operation error

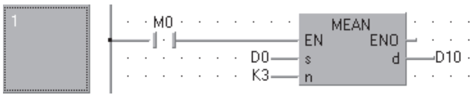
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for n is not within 0 to 32767.	—	—	—	—	○	○
4101	The n points of device specified for (s) exceed the specified device range.	—	—	—	—	○	○

Program example

- In the following program, the average value of 16-bit BIN data in D0 to D2 is evaluated when M0 turns ON, and the result is stored to D10.

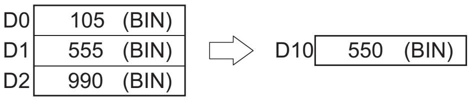
[Structured ladder/FBD]



[ST]

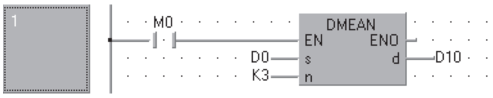
MEAN(M0,D0,K3,D10);

[Operation]



- In the following program, the average value of 32-bit BIN data in D0 to D5 is evaluated when M0 turns ON, and the result is stored to D10, D11.

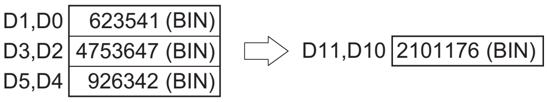
[Structured ladder/FBD]



[ST]

DMEAN(M0,D0,K3,D10);

[Operation]

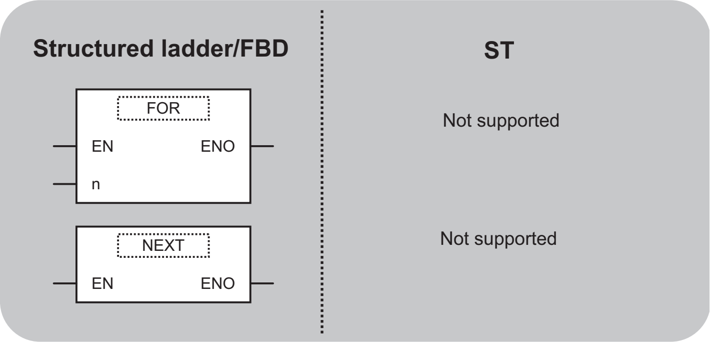


7.6 Structured Instructions

FOR to NEXT instruction loop

FOR, NEXT

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

FOR, NEXT

■Executing condition

Instruction	Executing condition
FOR, NEXT	Non-conditional execution

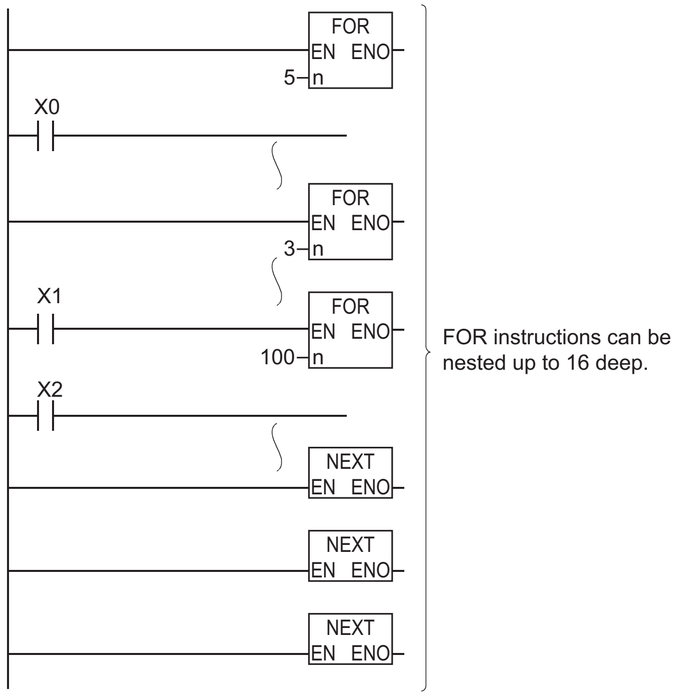
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Number of repeats of the FOR to NEXT instruction loop (1 to 32767)	ANY16
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○								—

Processing details

- Executes the processing of the next step of the NEXT instruction, when the process of the FOR to NEXT instruction loop is executed n times without conditions.
- The value of n can be specified with the value between 1 and 32767. If it is specified in the range from -32768 to 0, executes the same processing as n = 1.
- If you do not desire to execute the process of the FOR to NEXT instruction loop, use the CJ or SCJ instruction to jump.
- Up to 16 nesting levels are applicable for the FOR instruction.



Operation error

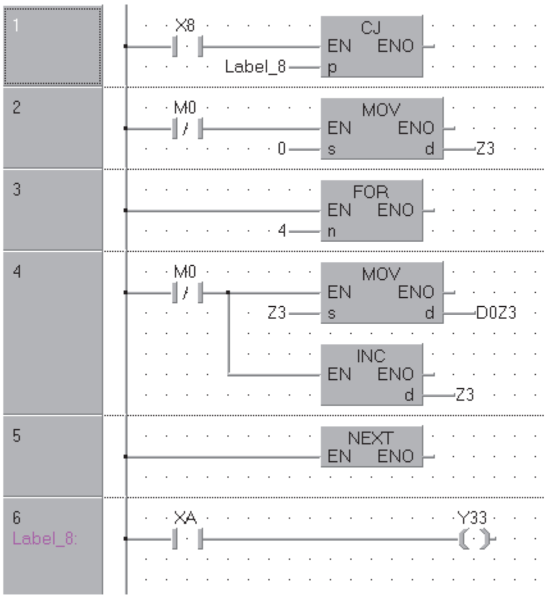
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4200	The FEND or GOEND instruction was executed before the execution of the NEXT instruction and after the execution of the FOR instruction. The STOP instruction has been inserted between the FOR and NEXT instructions.	○	○	○	○	○	○
4201	The NEXT instruction is executed prior to the execution of the FOR instruction.	○	○	○	○	○	○
4202	The 17th FOR instruction is executed when FOR instructions have been nested.	○	○	○	○	○	○

Program example

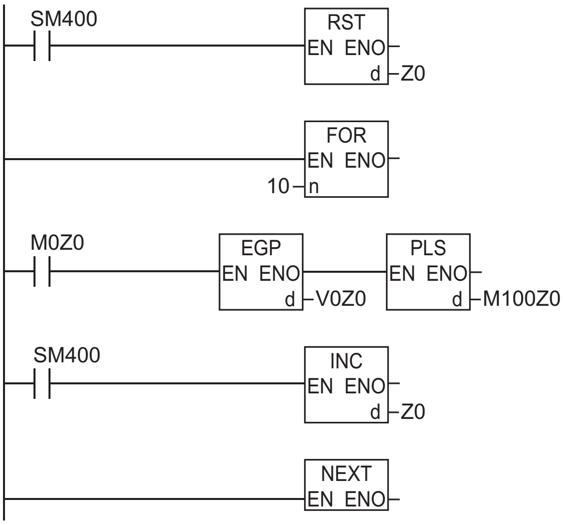
- In the following program, the FOR to NEXT instruction loop is executed when X8 is OFF, and not executed when X8 is ON.

[Structured ladder/FBD]



Point

- To force an end to the repetitious execution of the FOR to NEXT instruction loop, insert the BREAK instruction. For details on the BREAK instruction, refer to ☞ Page 424 Forced termination of FOR to NEXT instruction loop.
- Use the EGP or EGF instruction to perform the pulse operation of an indexed program in the FOR to NEXT instruction loop. Note, however, that rise and fall instructions are not available on the operation output side. For details on the EGP and EGF instructions, refer to ☞ Page 101 Pulse conversion of edge relay operation result. The program samples are shown below.



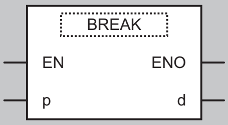
- Branching into the FOR to NEXT instruction loop using the JMP instruction or other branch instructions are not applicable from the outside of the FOR to NEXT instruction loop.

Forced termination of FOR to NEXT instruction loop

BREAK(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD

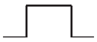
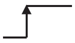


ST

Not supported

Any of the following instruction can go in the dotted squares.
BREAK, BREAKP

■Executing condition

Instruction	Executing condition
BREAK	
BREAKP	

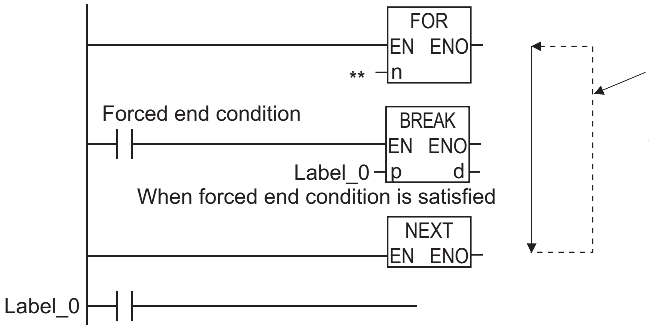
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	p	Pointer number of branch destination when the repeat process is forcibly terminated	ANY16
Output argument	ENO	Execution result	Bit
	d:	Device number that stores the remaining numbers of repeats	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others P
	Bit	Word		Bit	Word				
p	—							—	○
(d)	○							—	—

Processing details

- Forces an end to the FOR to NEXT instruction loop and shifts the operation to the pointer specified for Pn. Only a pointer within the same program file can be assigned to Pn. If a pointer of the other program file is used, an operation error will be returned.



If the BREAK instruction is not executed, program returns to the FOR instruction as many times as the number specified with the FOR instruction.

- The remaining number of the FOR to NEXT instruction loops is stored to (d) at the time of forced termination. Note that the remaining number includes the operation when the BREAK(P) instruction is executed.
- The BREAK(P) instruction can be used only in the FOR to NEXT instruction loop.
- The BREAK(P) instruction can be used only when there is only one level of nesting. When termination is forced to the multiple nesting levels, execute the same number of the BREAK(P) instructions for the nesting levels.

Operation error

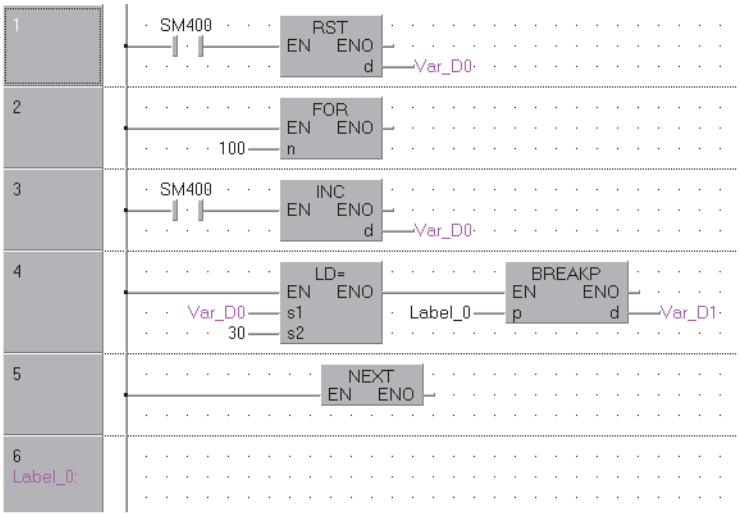
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4203	The BREAK(P) instruction is used in a case other than with the FOR to NEXT instruction loop.	○	○	○	○	○	○
4210	The jump destination for the pointer specified for p does not exist. The pointer of another program file is specified for p.	○	○	○	○	○	○

Program example

- In the following program, the forced termination is executed on the processes in the FOR to NEXT instruction loop when the value of Var_D0 reaches 30 (when the FOR to NEXT instruction loop has been executed 30 times).

[Structured ladder/FBD]

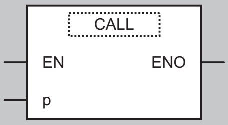


Subroutine program call

CALL(P)

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



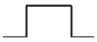
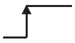
ST

Not supported

Any of the following instruction can go in the dotted squares.

CALL, CALLP

■Executing condition

Instruction	Executing condition
CALL	
CALLP	

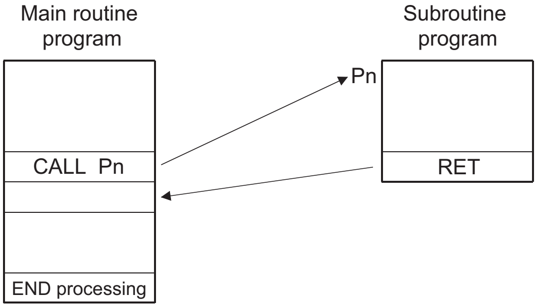
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	p	Start pointer number of the subroutine program	ANY16
Output argument	ENO	Execution result	Bit

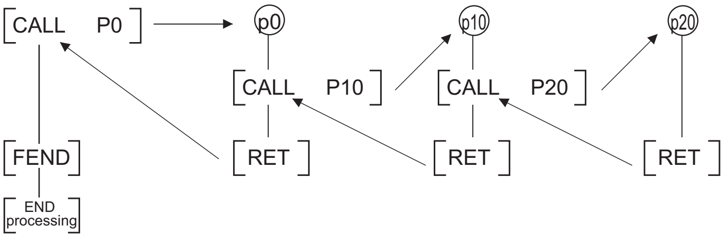
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others P
	Bit	Word		Bit	Word				
p	—	—		—					○

Processing details

- When the CALL(P) instruction is executed, executes the subroutine program of the program specified for Pn.
- The CALL(P) instruction can execute subroutine programs specified for a pointer within the same program file and subroutine programs specified for a common pointer.



- Up to 16 nesting levels are applicable for the CALL(P) instruction.
- The remaining number of the FOR to NEXT instruction loops is stored to (d) at the time of forced termination. Note that the remaining number includes the operation when the BREAK(P) instruction is executed.



- Devices which are turned ON within subroutine programs will be latched even if the subroutine program is not executed.

Operation error

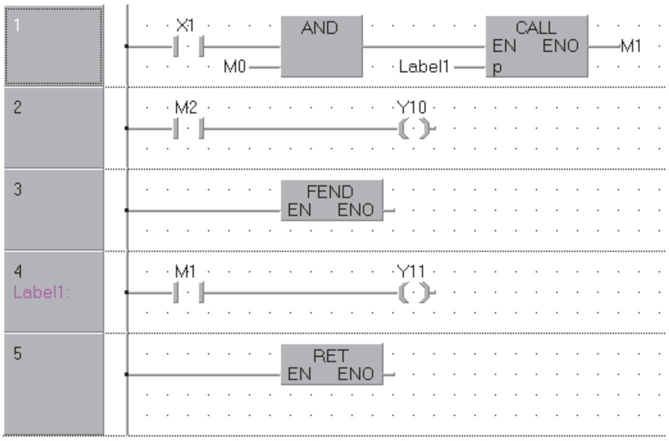
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4210	There is no subroutine program for the pointer specified in the CALL(P) instruction.	○	○	○	○	○	○
4211	The FEND, GOEND, or STOP instruction is executed after the execution of the CALL(P) instruction, and prior to the execution of the RET instruction.	○	○	○	○	○	○
4212	The RET instruction is executed prior to the execution of the CALL(P) instruction.	○	○	○	○	○	○
4213	The 17th nesting level is executed.	○	○	○	○	○	○

Program example

- In the following program, the subroutine program is executed when X20 turns ON.

[Structured ladder/FBD]

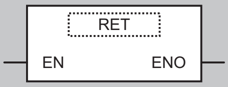


Return from subroutine program

RET

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



ST

Not supported

Any of the following instruction can go in the dotted squares.

RET

■Executing condition

Instruction	Executing condition
RET	Non-conditional execution

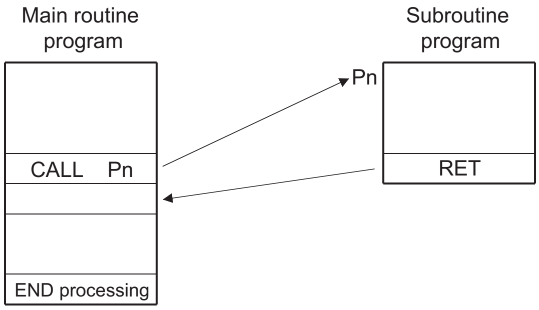
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

- Indicates the end of the subroutine program.
- When the RET instruction is executed, returns to the next step of the CALL(P) instruction which called the subroutine program.



Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4211	The FEND, GOEND or STOP instruction is executed after the execution of the CALL(P) instruction, and prior to the execution of the RET instruction.	○	○	○	○	○	○
4212	The RET instruction is executed prior to the execution of the CALL(P) instruction.	○	○	○	○	○	○

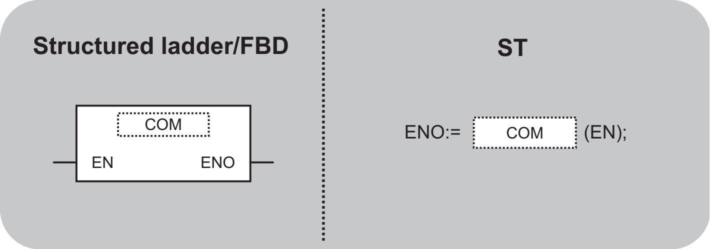
Refresh

COM

Basic | High performance | Process | Redundant ✕ | Universal ✕ | LCPU ✕

For the COM instruction of the following CPU modules, refer to ☞ Page 433 Selection of refresh(COM).

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later
- High Performance model QCPU: Supported if first 5 digits of the serial number are "04012" or later
- Process CPU: Supported if first 5 digits of the serial number are "07032" or later
- Redundant CPU
- Universal model QCPU
- LCPU



Any of the following instruction can go in the dotted squares.

COM

■Executing condition

Instruction	Executing condition
COM	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

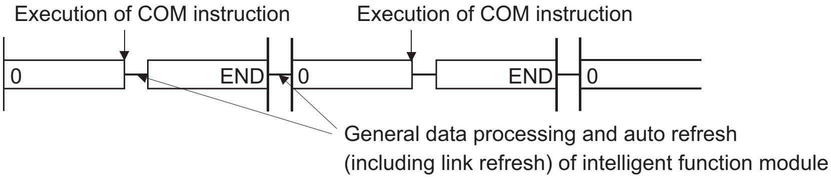
Processing details

- Use the COM instruction in any of the following cases.
  - For increasing the speed of transmission/reception processing to/from the remote I/O stations.
  - For ensuring reliable data transmission/reception with other stations that use different scan times during the execution of the data link.
- The processing of the COM instruction differs depending on whether the special relay SM775 is ON or OFF.
  - When SM775 is OFF: Performs auto refresh and service processing[*1][*2]
  - When SM775 is ON: Performs service processing only[*1]

[*1] The following processes are performed in service processing.
- Monitor processing of other station
- Read processing by the serial communications module of the buffer memory of another intelligent function module

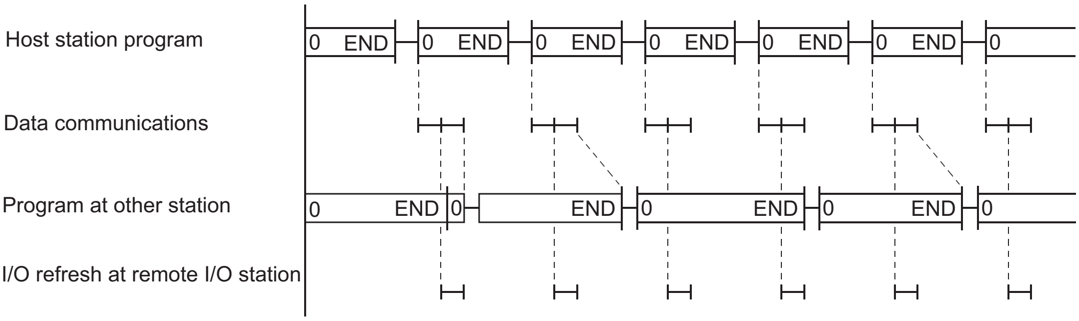
[*2] The auto refresh includes the following processes.
- Refresh of MELSECNET/10 and MELSECNET/H
- CC-Link refresh
- Auto refresh of intelligent function modules.

- At the point of the execution of the COM instruction, the CPU module temporarily stops the processing of the sequence program, and performs the same operation as ordinary data processing as well as auto refresh of intelligent function modules (including link refreshes) at the END processing. However, the low speed cyclic refresh of MELSECNET/10 or MELSECNET/H is not performed.

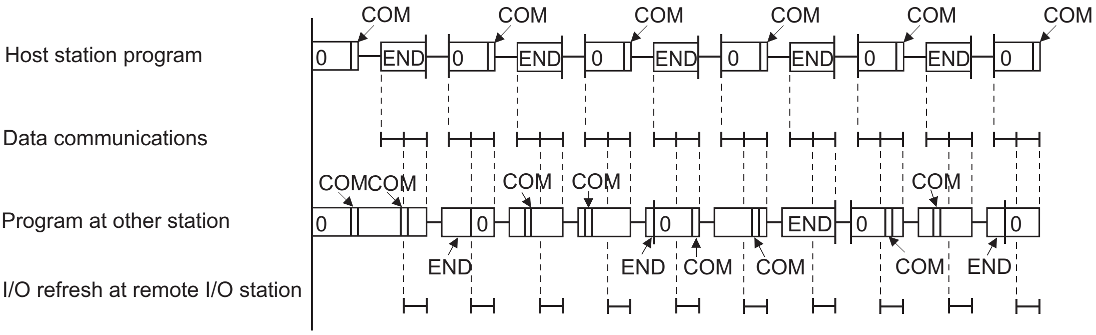

Execution of COM instruction
Execution of COM instruction
0
END
0
END
0
General data processing and auto refresh
(including link refresh) of intelligent function module

- The COM instruction can be used in a sequence program any number of times. However, note that the scan time of the sequence program will be lengthened by the time taken for service processing and the auto refresh (including the link refresh) of the intelligent function modules.
- Data communications using the COM instruction

Example of data communications when the COM instruction is not used



Example of data communications when the COM instruction has been used



- When the COM instruction is used at the host station, it is possible to increase the number of data communication repetitions with the remote I/O station unconditionally, as shown in (b) above, and thus to speed up data communications.
- In cases where the remote station scan time is longer than the scan time of the host station, the COM instruction used at the remote station side can avoid the occurrence of timing failure in which the data cannot be fetched, as shown in (a).
- When the COM instruction has been used at the other station, a link refresh will be performed each time that station receives a command from the host station.

· Step 0 ～COM instruction
· COM instruction ～COM instruction
· COM instruction ～END

Link refresh can be performed once in each of these intervals.

- If the scan time from the linked station is longer than the sequence program scan time at the host station, specifying the COM instruction at the host station will not increase the speed of data communications.

· Step 0 ～COM instruction
· COM instruction ～COM instruction
· COM instruction ～END

Link refresh can be performed once in each of these intervals.

Point

The programs in which the COM instruction cannot be used are shown below.

- Low-speed execution type programs
- Interrupt programs
- Fixed scan execution type programs

Operation error

- There is no operation error.

Selection of refresh(COM)

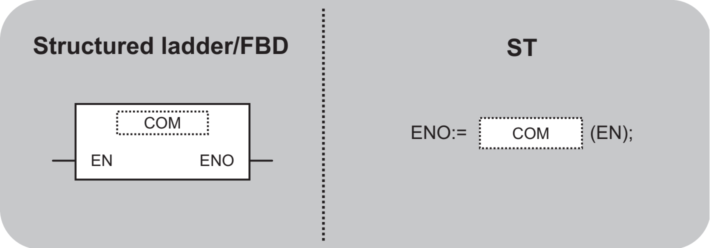
COM

Ver. Basic Ver. High performance Ver. Process Redundant Universal LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later
- High Performance model QCPU: Supported if first 5 digits of the serial number are "04012" or later
- Process CPU: Supported if first 5 digits of the serial number are "07032" or later

For the COM instruction of the following CPU modules, refer to ☞ Page 430 Refresh.

- Basic model QCPU: Supported if first 5 digits of the serial number are "04121" or earlier
- High Performance model QCPU: Supported if first 5 digits of the serial number are "04011" or earlier
- Process CPU: Supported if first 5 digits of the serial number are "07031" or earlier



Any of the following instruction can go in the dotted squares.

COM

■Executing condition

Instruction	Executing condition
COM	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

- The COM instruction is used to perform processing such as I/O refresh at any time during execution of the sequence program.
- When the COM instruction is executed, the following processing can be performed

Description	QCPU (Q mode)	LCPU
I/O refresh	○	○
CC-Link refresh	○	○
CC-Link IE Controller Network refresh	○	×
CC-Link IE Field Network refresh	○[*1]	○[*2]
MELSECNET/H refresh	○	×
Auto refresh of intelligent function modules	○	○
Auto refresh using the QCPU standard area of the multiple CPU system	○	×
Importing input/output data from CPUs that are not within the multiple CPU system	○	×
Auto refresh using the multiple CPU high speed transmission area of the multiple CPU system	○	×
Communication with display module	×	○
Service processing (communication with programming tool, GOT or other peripherals)	○	○

*1 CPU modules with a serial number whose first 5 digits are '12012' or higher are supported.
*2 CPU modules with a serial number whose first 5 digits are '13012' or higher are supported.

Point

The following processes are performed in service processing.

- Monitor processing of other station
- Read process of another intelligent function module buffer memory by the serial communication module

- Turning OFF SM775 executes all processing except for I/O refresh.

- When selecting refresh items

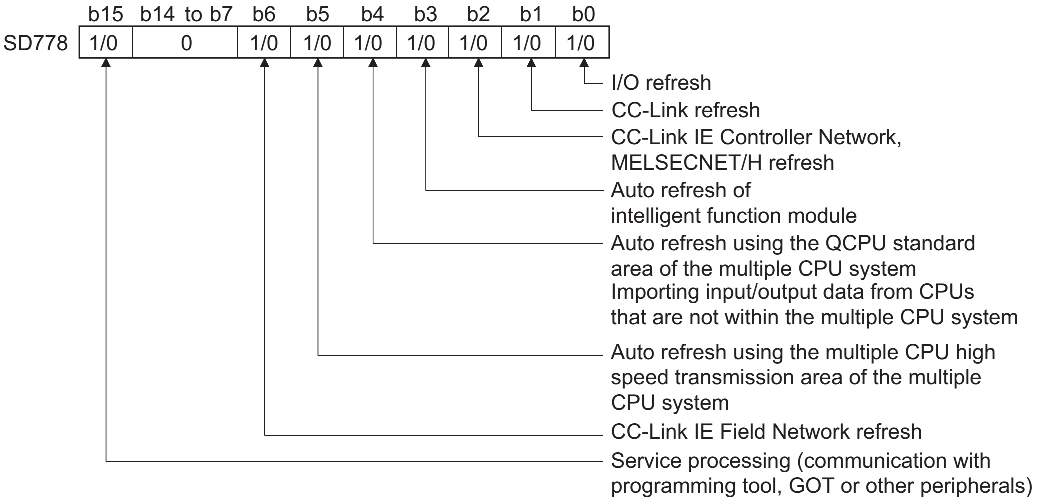
- Select the processing by SD778, and turn SM775 ON. The following table shows the refresh items when SM775 is ON or OFF, and the processing that can be specified for SD778.

Description	QCPU (Q mode)		LCPU	
	When SM775 is OFF	When SM775 is ON	When SM775 is OFF	When SM775 is ON
I/O refresh	Not executed	Whether to be executed or not can be selected	Not executed	Whether to be executed or not can be selected.
CC-Link refresh	Executed		Executed	
CC-Link IE Controller Network refresh			-	-
CC-Link IE Field Network refresh			Executed	Whether to be executed or not can be selected
MELSECNET/H refresh			-	-
Auto refresh of intelligent function modules			Executed	Whether to be executed or not can be selected
Auto refresh using the QCPU standard area of the multiple CPU system			-	-
Importing input/output data from CPUs that are not within the multiple CPU system			-	-
Auto refresh using the multiple CPU high speed transmission area of the multiple CPU system			-	-
Communication with display module	-	-	Executed	Whether to be executed or not can be selected
Service processing (communication with programming tool, GOT or other peripherals)	Executed	Whether to be executed or not can be selected		

- Select whether to execute the processing or not using the bits of SD778. Whether to execute each bit of SD778 or not can be specified as shown below.

QCPU (Q mode)

Bit of SD778	Executed	Not executed
b0 to b6	1	0
b15	0	1



SD778
b15
b14 to b7
b6
b5
b4
b3
b2
b1
b0
1/0
0
1/0
1/0
1/0
1/0
1/0
1/0
1/0
I/O refresh
CC-Link refresh
CC-Link IE Controller Network, MELSECNET/H refresh
Auto refresh of intelligent function module
Auto refresh using the QCPU standard area of the multiple CPU system
Importing input/output data from CPUs that are not within the multiple CPU system
Auto refresh using the multiple CPU high speed transmission area of the multiple CPU system
CC-Link IE Field Network refresh
Service processing (communication with programming tool, GOT or other peripherals)

Ex.

To make only the send/receive processing with the remote I/O station faster, specify the MELSECNET/H refresh only. (Set only b2 and b15 of SD778 to 1 (SD778: 8004H).)

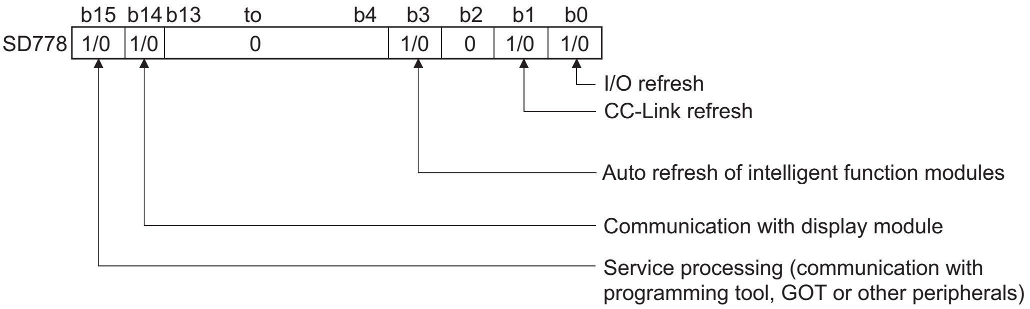
Point

Refresh between the multiple CPUs by the COM instruction is performed under the following conditions.

- Receive operation from other CPUs: When b4 of SD778 (auto refresh of CPU shared memory) is 1.
- end operation from host CPU: When b15 of SD778 (execution/non-execution of service processing) is 0

LCPU

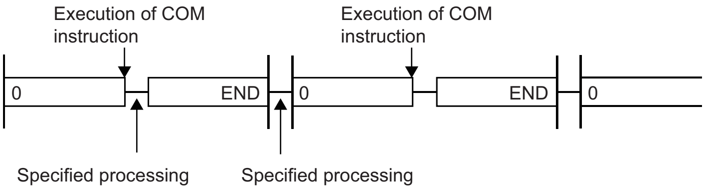
Bit of SD778	Executed	Not executed
0 to b3	1	0
b14	1	0


b15
b14
b13
to
b4
b3
b2
b1
b0
SD778
1/0
1/0
0
1/0
0
1/0
1/0
I/O refresh
CC-Link refresh
Auto refresh of intelligent function modules
Communication with display module
Service processing (communication with programming tool, GOT or other peripherals)

Ex.

To make only the display module processing faster, specify the Communication with display module only. (Set b14 and b15 of SD778 to 1 (SD778: C000H).)

- The COM instruction can be used in a sequence program any number of times.



- The COM instruction can be used in a sequence program any number of times. However, note that the sequence program scan time will be lengthened by the time taken for the processing selected for SD778.
- For Universal model QCPU and LCPU, an interrupt is permitted during the execution of the COM instruction. However, note that the data may be separated if the refresh data are used in an interrupt program.
- For Built-in Ethernet port QCPU, Built-in Ethernet port LCPU, and LCPU, when the service processing is performed using the COM instruction with the Ethernet connected, the processing time may be extended.

Point

The programs in which the COM instruction cannot be used are shown below:

- Low-speed execution type programs
- Interrupt programs
- Fixed scan execution type programs

For the redundant CPU, there are restrictions on use of the COM instruction. For details on the restrictions, refer to the following manual.

- QnPRHCPU User's Manual (Redundant System)

Operation error

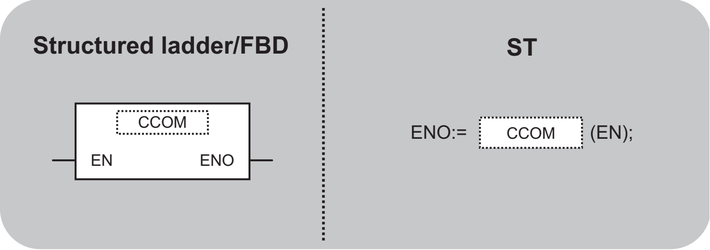
- There is no operation error.

Selection of refresh(CCOM(P))

CCOM(P)

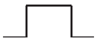
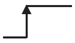
✗Basic ✗High performance ✗Process ✗Redundant Ver. Universal LCPU

- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported



Any of the following instruction can go in the dotted squares.
CCOM, CCOMP

■Executing condition

Instruction	Executing condition
CCOM	
CCOMP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

For details of the functions, refer to ☞ Page 433 Selection of refresh(COM).

Operation error

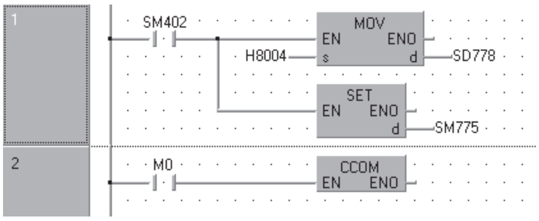
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The CCOM(P) instruction was executed in the QnU(D)(H)CPU whose serial number (first five digits) is "10101" or earlier.	—	—	—	—	○	—

Program example

- The program in which the execution of refresh instruction can be switched by turning M0 ON/OFF.

[Structured ladder/FBD]

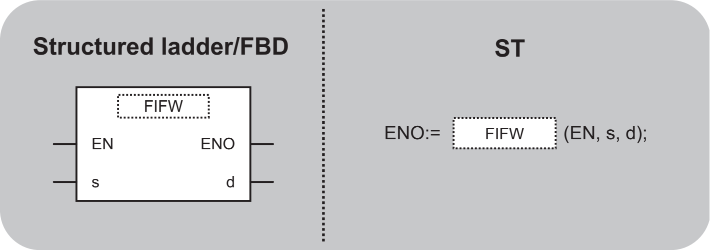


7.7 Data Table Operation Instructions

Writing data to data table

FIFW(P)

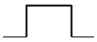
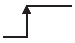
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

FIFW, FIFWP

■Executing condition

Instruction	Executing condition
FIFW	
FIFWP	

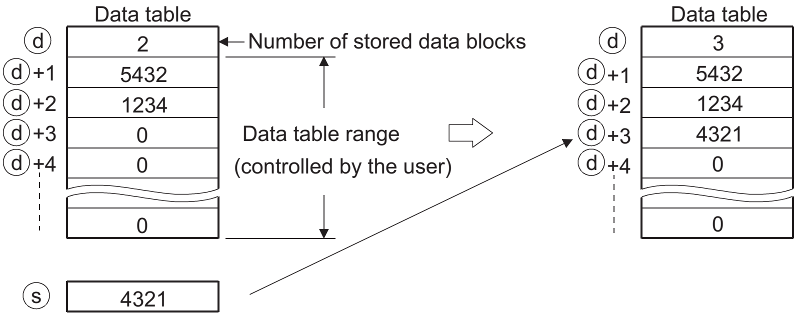
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Write data to the table, or start device number that stores data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the table	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○	○		○				○	—
(d)	—	○		—				—	—

Processing details

- Stores the 16-bit data specified for (s) to the data table specified for (d). The number of data blocks stored in the table is stored to (d), and the data specified for (s) are stored to (d)+1 and the following devices.



- When the FIFW(P) instruction is executed for the first time, any values specified for (d) should be cleared.
- The number of data blocks to be written in the data table and the data table range should be managed by a user. (Refer to Program Example)

Operation error

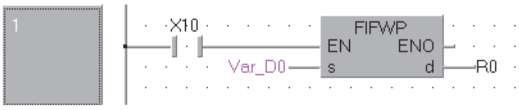
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The FIFW(P) instruction is executed when the value of (d) is FFFFH.	○	○	○	○	○	○
4101	The data table range exceeds the corresponding device range when the FIFW(P) instruction is executed.	○	○	○	○	○	○

Program example

- In the following program, the data in Var_D0 are stored to the data table from R0 when X10 turns ON.

[Structured ladder/FBD]



[ST]

FIFWP(X10,Var_D0,R0);

[Operation]

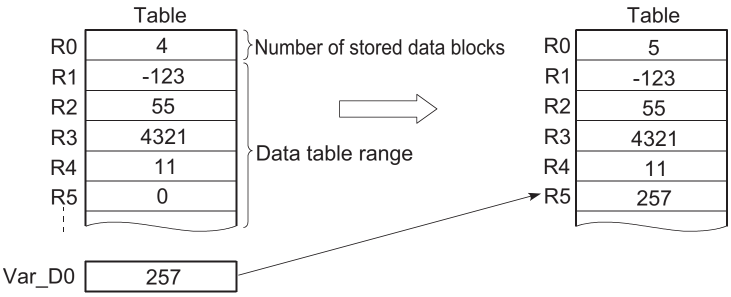
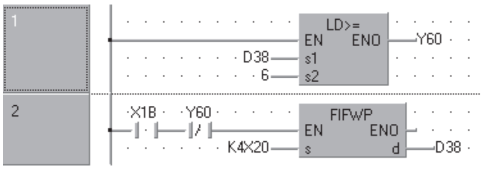


	Table			Table
R0	4	Number of stored data blocks	R0	5
R1	-123	Data table range	R1	-123
R2	55		R2	55
R3	4321		R3	4321
R4	11		R4	11
R5	0		R5	257
Var_D0	257			

- In the following program, the data from X20 to X2F are stored to the data table from D38 to D44 when X1B turns ON, and if there are more than 6 data blocks to be stored, Y60 is turned ON and the FIFW(P) instruction is disabled.

[Structured ladder/FBD]



[ST]

OUT(D38>=6,Y60);

FIFWP(X1B AND NOT(Y60),K4X20,D38);

[Operation]

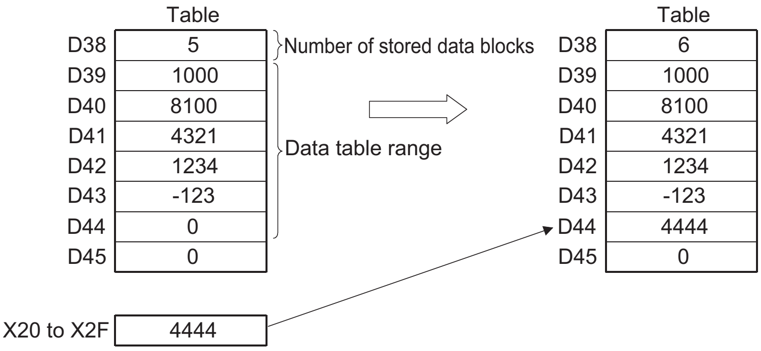


	Table			Table
D38	5	Number of stored data blocks	D38	6
D39	1000	Data table range	D39	1000
D40	8100		D40	8100
D41	4321		D41	4321
D42	1234		D42	1234
D43	-123		D43	-123
D44	0		D44	4444
D45	0		D45	0
X20 to X2F	4444			

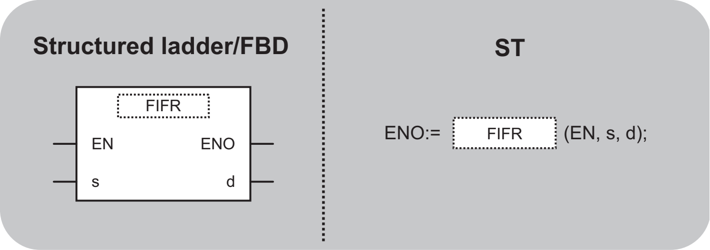
Precautions

When using the FIFW instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

Reading oldest data from data table

FIFR(P)

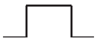
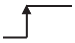
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

FIFR, FIFRP

■Executing condition

Instruction	Executing condition
FIFR	
FIFRP	

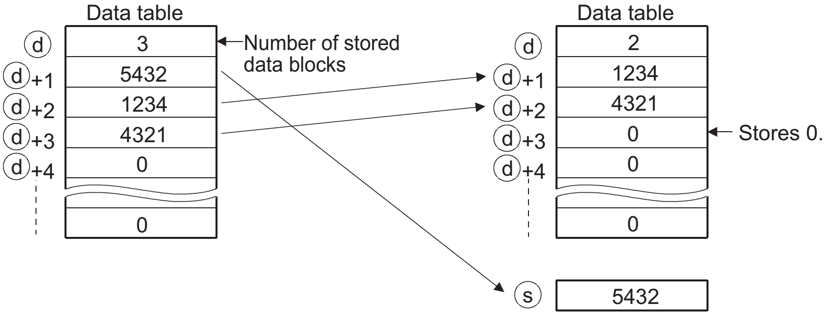
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores read data from the table	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the table	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s)	○	○		○				—	
(d)	—	○		—				—	

Processing details

- Stores the oldest data ((d)+1) input to the table specified for (d) to the device specified for (s). After the execution of the FIFR(P) instruction, the data in the table are all compressed up by one block.



- Set the interlock to avoid executing the FIFR(P) instruction if the value stored in (d) is 0. (Refer to Program Example)

Operation error

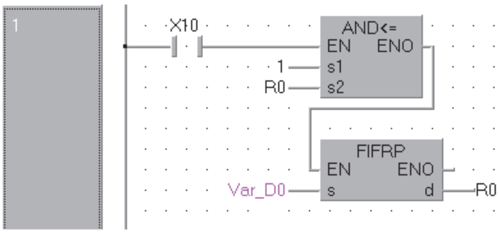
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The FIFR(P) instruction is executed when the value of (d) is 0.	○	○	○	○	○	○
4101	The data table range exceeds the corresponding device range when the FIFR(P) instruction is executed.	○	○	○	○	○	○

Program example

- In the following program, the data in R1 from the data table R0 to R7 are stored to Var_D0 when X10 turns ON.

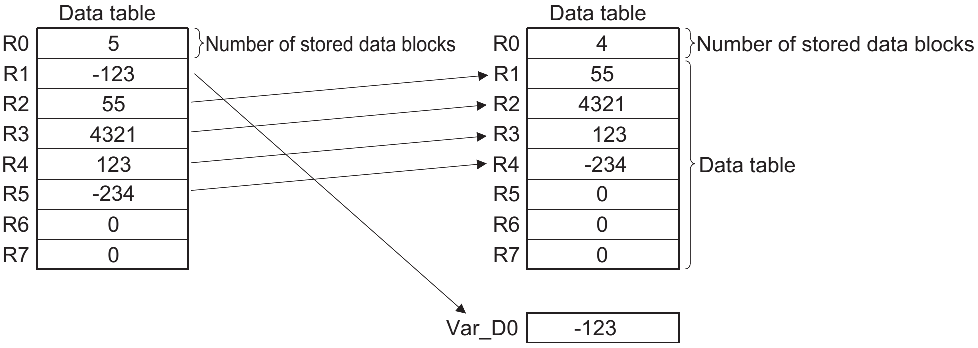
[Structured ladder/FBD]



[ST]

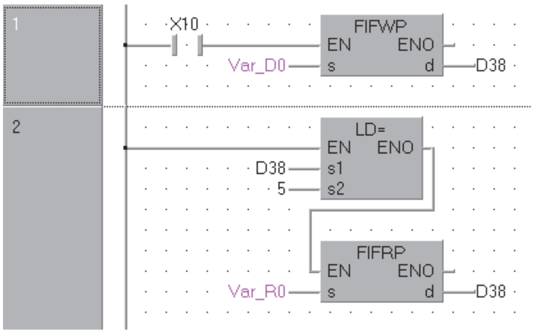
FIFRP(X10 AND R0>=1,Var_D0,R0);

[Operation]



- In the following program, the data in Var_D0 are stored to the data table D38 to D43 when X10 turns ON, and when the number of stored data reaches 5, the last-stored data in D39 of the data table are stored to Var_R0.

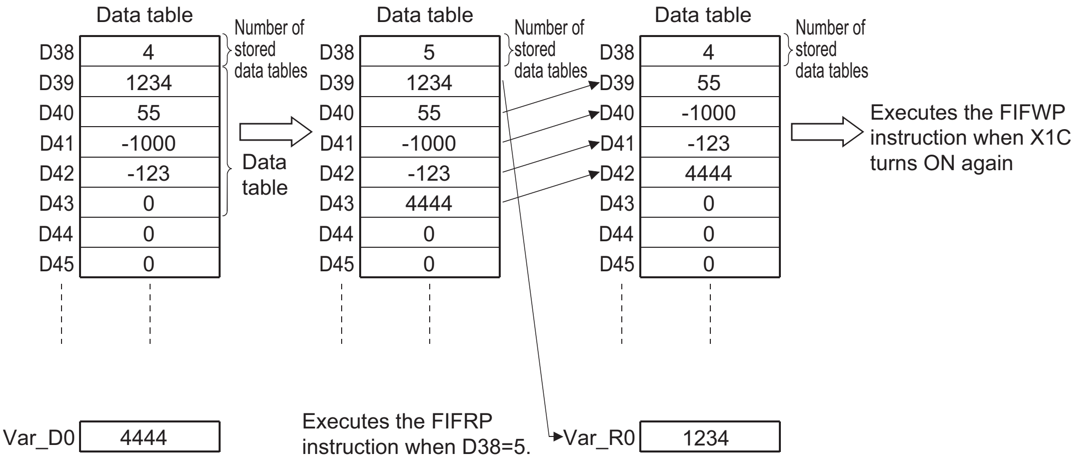
[Structured ladder/FBD]



[ST]

```
FIFWP(X10,Var_D0,D38);
FIFRP(D38=5,Var_R0,D38);
```

[Operation]



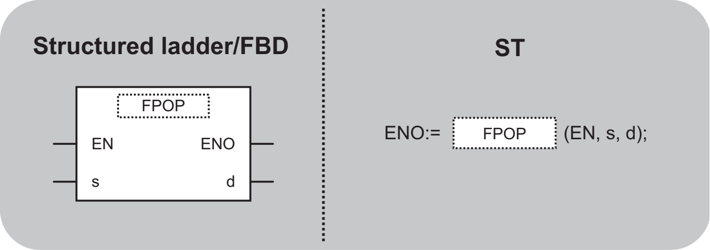
Precautions

When using the FIFR instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

Reading newest data from data table

FPOP(P)

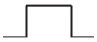
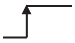
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

FPOP, FPOPP

■Executing condition

Instruction	Executing condition
FPOP	
FPOPP	

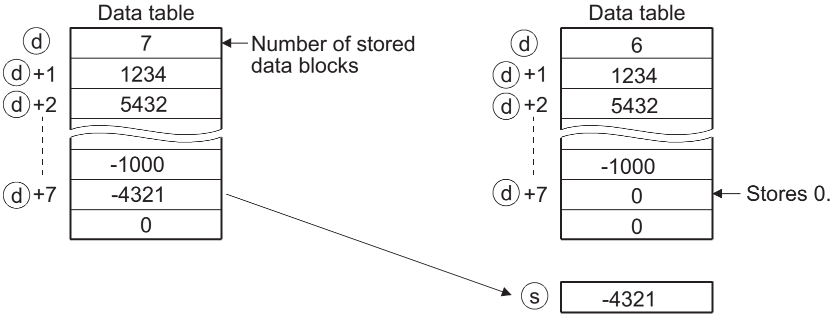
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores read data from the table	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the table	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s)	○	○		○				—	
(d)	—	○		—				—	

Processing details

- Stores the newest data input to the table specified for (d) to the device specified for (s). After the execution of the FPOP(P) instruction, the device storing the data read by the FPOP(P) instruction becomes 0.



- Set the interlock to avoid executing the FPOP(P) instruction when the value stored in (d) is 0. (Refer to Program Example)

Operation error

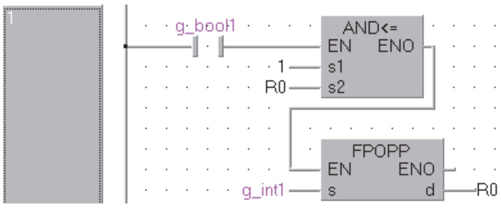
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The FPOP(P) instruction is executed when the value of (d) is 0.	○	○	○	○	○	○
4101	The data table range exceeds the corresponding device range when the FPOP(P) instruction is executed.	○	○	○	○	○	○

Program example

- In the following program, the last-stored data in the data table R0 to R7 are stored to g_int1 when g_bool1 turns ON.

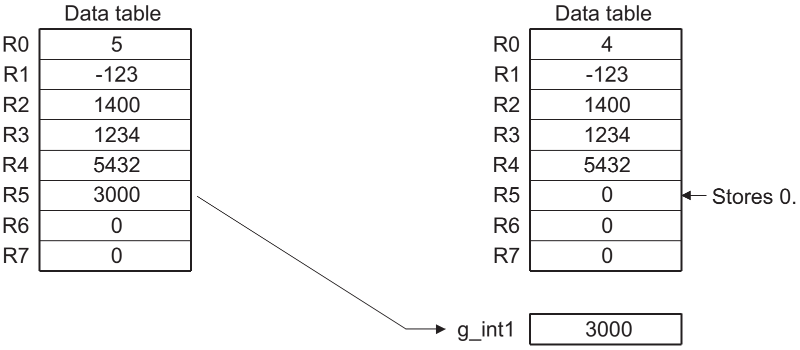
[Structured ladder/FBD]



[ST]

```
FPOPP(g_bool1 AND R0>=1,g_int1,R0);
```

[Operation]



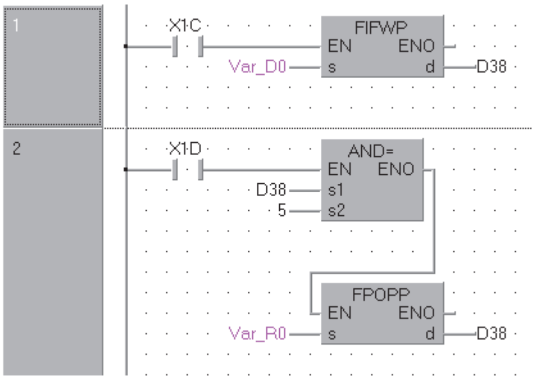
	Data table
R0	5
R1	-123
R2	1400
R3	1234
R4	5432
R5	3000
R6	0
R7	0

	Data table	
R0	4	
R1	-123	
R2	1400	
R3	1234	
R4	5432	
R5	0	Stores 0.
R6	0	
R7	0	

g_int1	3000

- In the following program, the data in Var_D0 are stored to the data table D38 to D43 when X1C turns ON, and when the number of stored data reaches 5, and X1D turns ON, the last-stored data in the data table are stored to Var_R0.

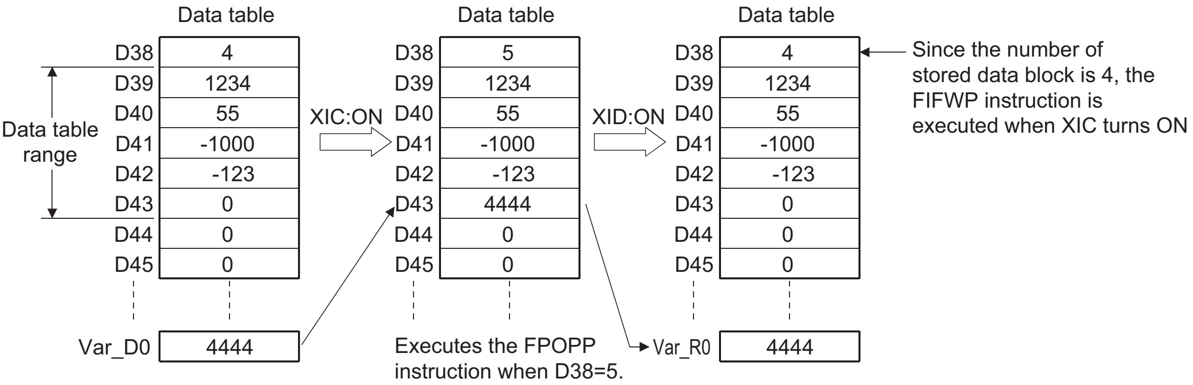
[Structured ladder/FBD]



[ST]

```
FIFWP(X1C,Var_D0,D38);
FPOPP(X1D AND D38=5,Var_R0,D38);
```

[Operation]



	Data table		Data table		Data table
D38	4	D38	5	D38	4
D39	1234	D39	1234	D39	1234
D40	55	D40	55	D40	55
D41	-1000	D41	-1000	D41	-1000
D42	-123	D42	-123	D42	-123
D43	0	D43	4444	D43	0
D44	0	D44	0	D44	0
D45	0	D45	0	D45	0
Var_D0	4444			Var_R0	4444

Since the number of stored data block is 4, the FIFWP instruction is executed when XIC turns ON

Executes the FPOPP instruction when D38=5.

Precautions

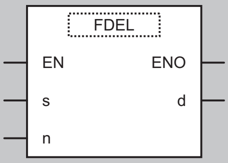
When using the FPOP instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

Deleting/inserting data from/to data table

FDEL(P), FINS(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



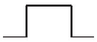
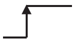
ST

ENO:= FDEL (EN, s, n, d);

Any of the following instruction can go in the dotted squares.

FDEL, FDELP, FINS, FINSP

■Executing condition

Instruction	Executing condition
FDEL, FINS	
FDELP, FINSP	

■Argument

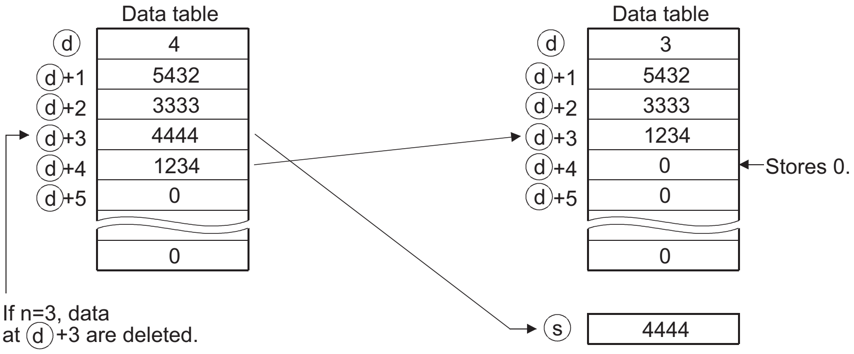
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device which stores insert data or the device which stores delete data.	ANY16
	n	Position of the table where data are inserted or deleted	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the table	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○	○		○				—	—
n	○	○		○				○	—
(d)	—	○		—				—	—

Processing details

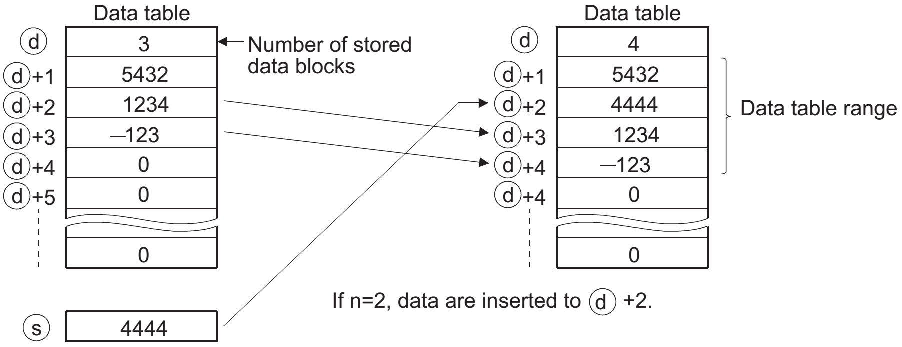
■FDEL(P)

- Deletes the nth block of data from the data table specified for (d), and stores the deleted data to the device specified for (s). After the execution of the FDEL(P) instruction, (n+1)th data and the following data in the table are compressed forward by one block.



■FINS(P)

- Inserts the 16-bit data specified for (s) in the nth block of the data table specified for (d). After the execution of the FINS(P) instruction, the data in the table following the inserted block are all dropped one position.



Operation error

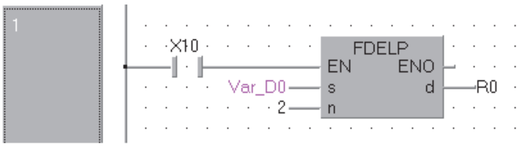
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The FDEL(P) or FINS(P) instruction is executed when n = 0. The FDEL(P) instruction is executed when the value of (d) is 0. The FINS(P) instruction is executed when the value of (d) is FFFFH.	○	○	○	○	○	○
4101	The nth position from (d) is larger than the data storage number at the execution of the FDEL(P) instruction. The nth position from (d) is larger than the "data storage number + 1" at the execution of the FINS(P) instruction. The value of n exceeds the device range of the table (D) at the execution of the FDEL(P) or FINS(P) instruction. The data table range exceeds the corresponding device range when the FDEL(P) or FINS(P) instruction is executed.	○	○	○	○	○	○

Program example

- In the following program, the second data from the table R0 to R7 are deleted when X10 turns ON, and the deleted data are stored to Var_D0.

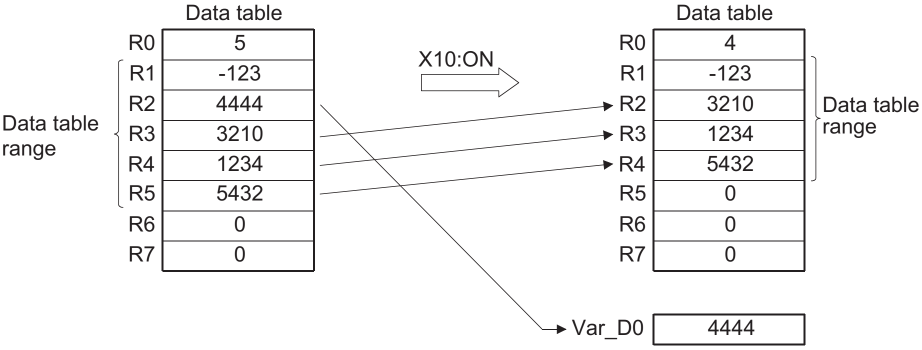
[Structured ladder/FBD]



[ST]

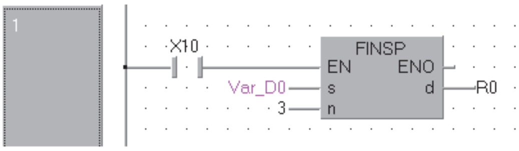
FDELP(X10,Var_D0,2,R0);

[Operation]



- In the following program, the data in Var_D0 are inserted to the third position of the data table R0 to R7 when X10 turns ON.

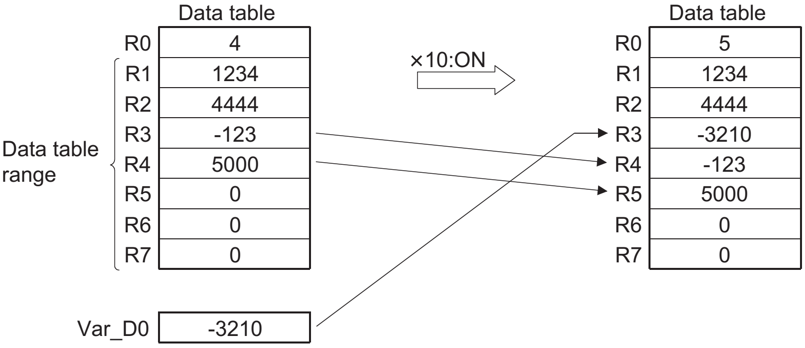
[Structured ladder/FBD]



[ST]

FINSP(X10,Var_D0,3,R0);

[Operation]



Precautions

When using the FDEL/FINS instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

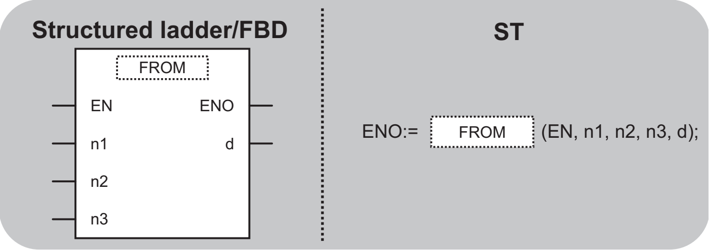
7.8 Buffer Memory Access Instructions

Reading 1-/2-word data from intelligent function module

FROM(P), DFRO(P)

Basic | High performance | Process | Redundant | Ver. Universal | LCPU

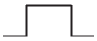
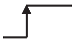
- Universal model QCPU: Not supported for Q00UJCPU



Any of the following instruction can go in the dotted squares.

FROM, FROMP, DFRO, DFROP

■Executing condition

Instruction	Executing condition
FROM, DFRO	
FROMP, FROP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Start I/O number of the intelligent function module[*1]	ANY16
	n2	Start address of data to be read	ANY16
	n3	Number of data to be read	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores read data	ANY16/32

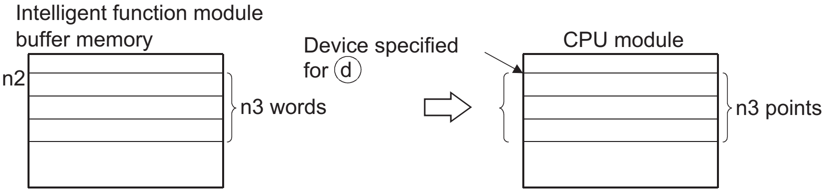
*1 Specified by the upper 3 digits of start I/O number expressed in 4-digit hexadecimal.

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others U
	Bit	Word		Bit	Word				
n1	○			○				○	○
n2	○			○				○	—
n3	○			○				○	—
(d)	○			—				—	—

Processing details

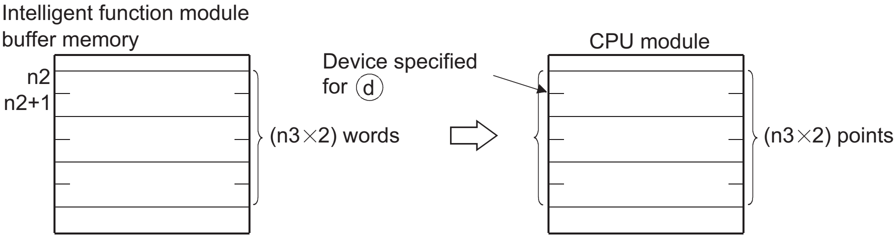
■FROM(P)

- Reads the data in n3 words from the buffer memory address specified for n2 of the intelligent function module specified for n1, and stores the data to the device specified for (d) and the following devices.



■DFRO(P)

- Reads the data in (n3 × 2) words from the buffer memory address specified for n2 of the intelligent function module specified for n1, and stores the data to the device specified for (d) and the following devices.



Point

Data read from intelligent function modules is also possible with the use of intelligent function module devices. For the intelligent function module device, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

Operation error

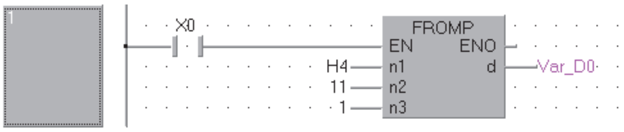
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
1402	An error has been detected in the intelligent function module at the execution of the instruction.	○	○	○	○	○	○
1412	There has been no exchange of signals with the intelligent function module at the execution of the instruction.	○	○	○	○	○	○
2110	The I/O number specified for n1 is not for the intelligent function module.	○	○	○	○	○	○
4101	The range of n3 points (2 × n3 points for the DFRO(P) instruction) from the device specified for (d) exceeds the specified device range. The address specified for n2 is outside the buffer memory range.	○	○	○	○	○	○

Program example

- In the following program, the CH1 digital output value of the Q68ADV mounted with I/O numbers 040 to 05F is read to Var_D0 when X0 turned ON. (Reads 1 word of data from address 11 of the buffer memory.)

[Structured ladder/FBD]

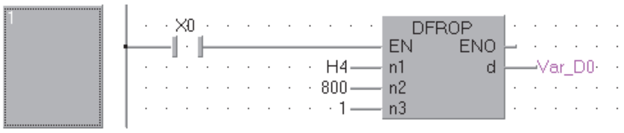


[ST]

```
FROMP(X0,H4,11,1,Var_D0);
```

- In the following program, the 1-axis current feed value of the QD75P4 mounted at the I/O numbers 040 to 05F is read to Var_D0, when X0 turns ON. (Reads data in 2 words from the address 800 and 801 of the buffer memory.)

[Structured ladder/FBD]



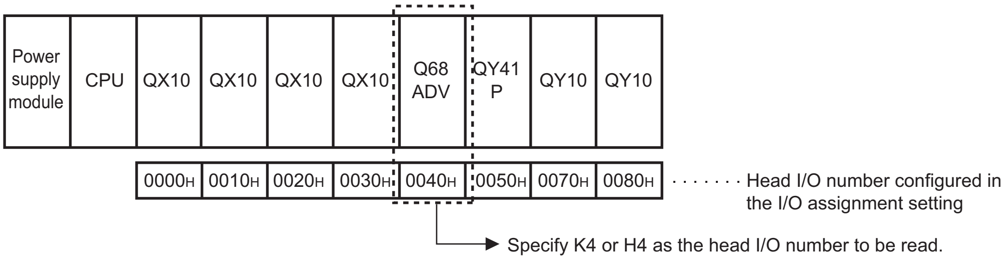
[ST]

```
DFROP(X0,H4,800,1,Var_D0);
```

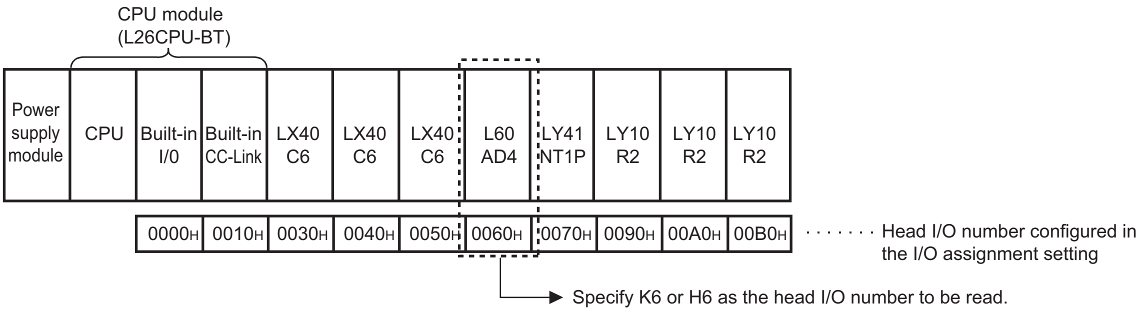
Point

The value of n1 is specified by the upper 3 digits of hexadecimal 4-digit representation of the start I/O number of the slot in which the intelligent function module is mounted.

< QCPU (Q mode) >


Power supply module
CPU
QX10
QX10
QX10
QX10
Q68 ADV
QY41 P
QY10
QY10
0000H
0010H
0020H
0030H
0040H
0050H
0070H
0080H
Head I/O number configured in the I/O assignment setting
Specify K4 or H4 as the head I/O number to be read.

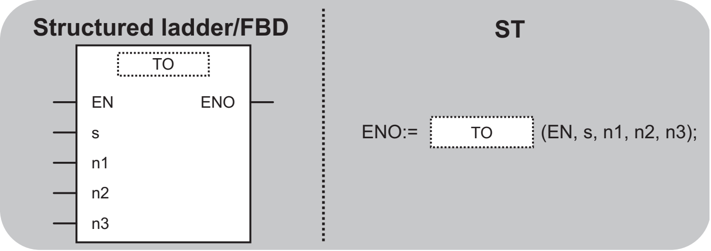
< LCPU >



Writing 1-/2-word data to intelligent function module

TO(P), DTO(P)

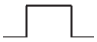
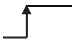
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

TO, TOP, DTO, DTOP

■Executing condition

Instruction	Executing condition
TO, DTO	
TOP, DTOP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data to be written or device start number that stores data to be written	ANY16/32
	n1	Start I/O number of the intelligent function module[*1]	ANY16
	n2	Start address of the buffer memory where the data to be written	ANY16
	n3	Number of data to be written	ANY16
Output argument	ENO	Execution result	Bit

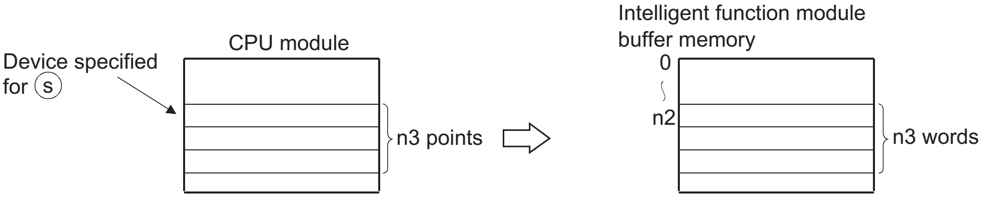
[*1] Specified by the upper 3 digits of start I/O number expressed in 4-digit hexadecimal.

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others U
	Bit	Word		Bit	Word				
(s)	○			—				○	—
n1	○			○				○	○
n2	○			○				○	—
n3	○			○				○	—

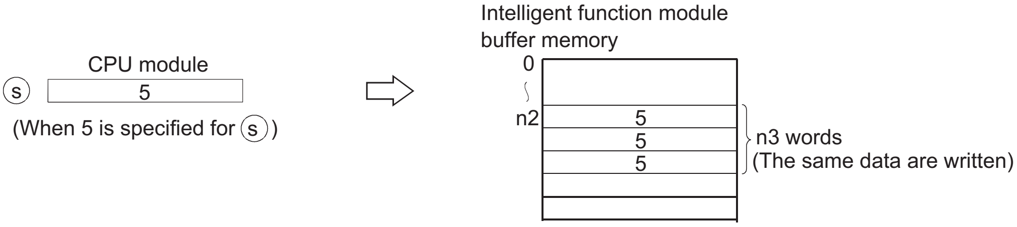
Processing details

■TO(P)

- Writes the data stored in n3 words following the device specified for (s) to the buffer memory address specified for n2 and the following addresses of the intelligent function module specified for n1.

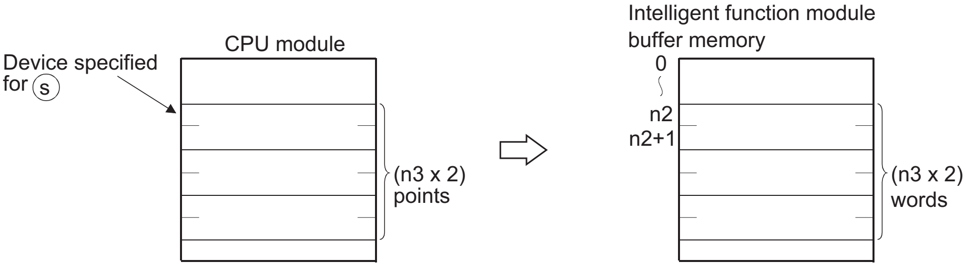


- When a constant is specified for (s), writes the same data (value specified for (s)) to the area of n3 words following the specified buffer memory. ((s) can be specified in the following range: -32768 to 32767 or 0H to FFFFH.)

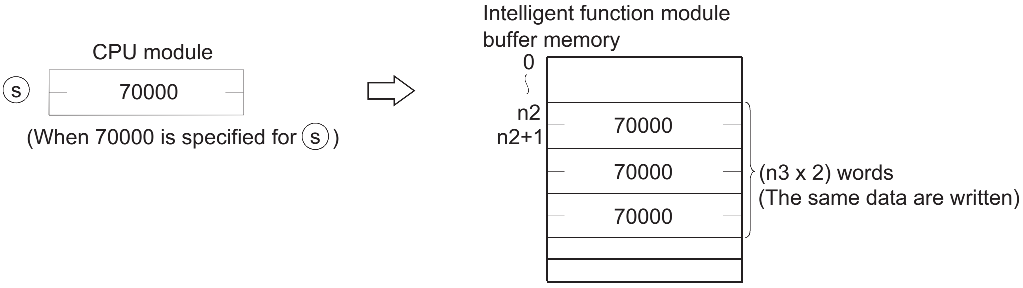

CPU module
(s)
5
(When 5 is specified for (s))
Intelligent function module buffer memory
0
n2
5
5
5
n3 words
(The same data are written)

■DTO(P)

- Writes the data stored in (n3 × 2) words following the device specified for (s) to the buffer memory address specified by n2 and the following addresses of the intelligent function module specified for n1.



- When a constant is specified for (s), writes the same data (value specified for (s)) to the area of (n3 × 2) words following the specified buffer memory. ((s) can be specified in the following range: -2147483648 to 2147483647 or 0H to FFFFFFFFH.)


CPU module
(s)
70000
(When 70000 is specified for (s))
Intelligent function module buffer memory
0
n2
n2+1
70000
70000
70000
(n3 x 2) words
(The same data are written)

Point

Data write to intelligent function modules is also possible with the use of intelligent function module devices. For the intelligent function module device, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

Operation error

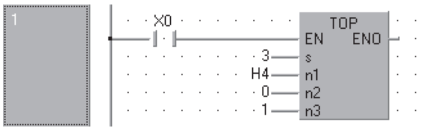
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
1402	An error has been detected in the intelligent function module at the execution of the instruction.	○	○	○	○	○	○
1412	There has been no exchange of signals with the intelligent function module at the execution of the instruction.	○	○	○	○	○	○
2110	The I/O number specified for n1 is not for the intelligent function module.	○	○	○	○	○	○
4101	The n3 points (2 × n3 points for the DTO(P) instruction) of the device specified for (s) exceed the specified device range. The address specified for n2 is outside the buffer memory range.	○	○	○	○	○	○

Program example

- In the following program, the CH1 and CH2 of the Q68ADV with the I/O numbers X/Y040 to X/Y04F are set to the "Disable A/D conversion" mode, when X0 turns ON. (Writes 3 to the buffer memory address 0.)

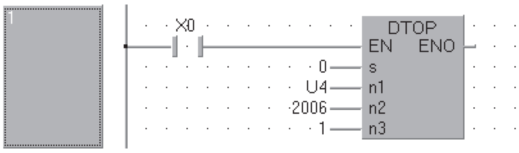
[Structured ladder/FBD]



[ST]
TOP(X0,3,H4,0,1);

- In the following program, the 1-axis positioning address/movement amount of the QD75P4 with the I/O numbers X/Y040 to X/Y05F is set to 0 when X0 turns ON. (Writes 0 to buffer memory addresses 2006 and 2007.)

[Structured ladder/FBD]

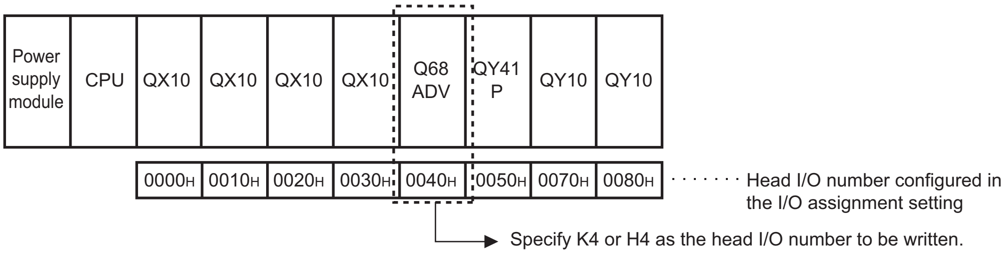


[ST]
DTOP(X0,0,U4,2006,1);

Point

The value of n1 is specified by the upper 3 digits of hexadecimal 4-digit representation of the start I/O number of the slot in which the intelligent function module is mounted.

< QCPU (Q mode) >

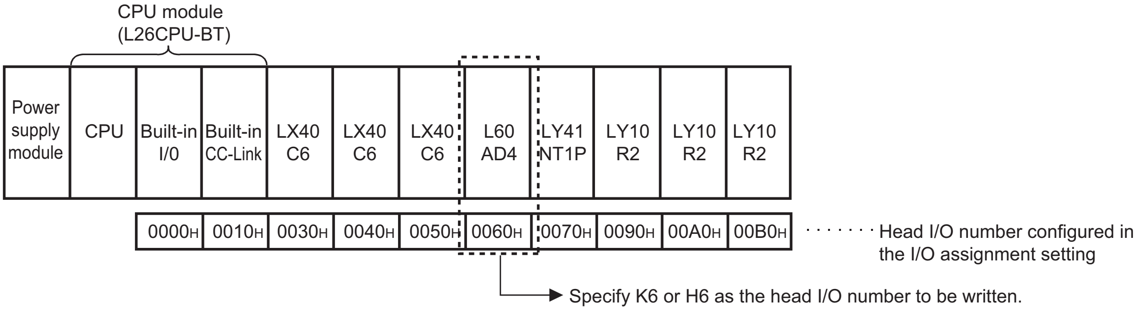


Power supply module	CPU	QX10	QX10	QX10	QX10	Q68 ADV	QY41 P	QY10	QY10
		0000H	0010H	0020H	0030H	0040H	0050H	0070H	0080H

······· Head I/O number configured in the I/O assignment setting

Specify K4 or H4 as the head I/O number to be written.

< LCPU >



CPU module (L26CPU-BT)

Power supply module	CPU	Built-in I/0	Built-in CC-Link	LX40 C6	LX40 C6	LX40 C6	L60 AD4	LY41 NT1P	LY10 R2	LY10 R2	LY10 R2
		0000H	0010H	0030H	0040H	0050H	0060H	0070H	0090H	00A0H	00B0H

······· Head I/O number configured in the I/O assignment setting

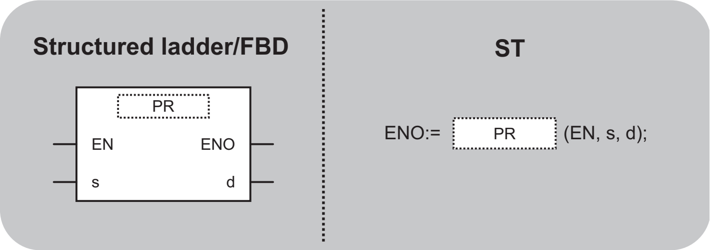
Specify K6 or H6 as the head I/O number to be written.

7.9 Display Instructions

Printing ASCII code

PR

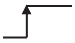
~~Basic~~ High performance Process ~~Redundant~~ ~~Universal~~ ~~LCPU~~



Any of the following instruction can go in the dotted squares.

PR

■Executing condition

Instruction	Executing condition
PR	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	ASCII code, or start number of the device that stores ASCII code	ANY16/string
Output argument	ENO	Execution result	Bit
	d	tart number of the output module that outputs ASCII code data	Bit

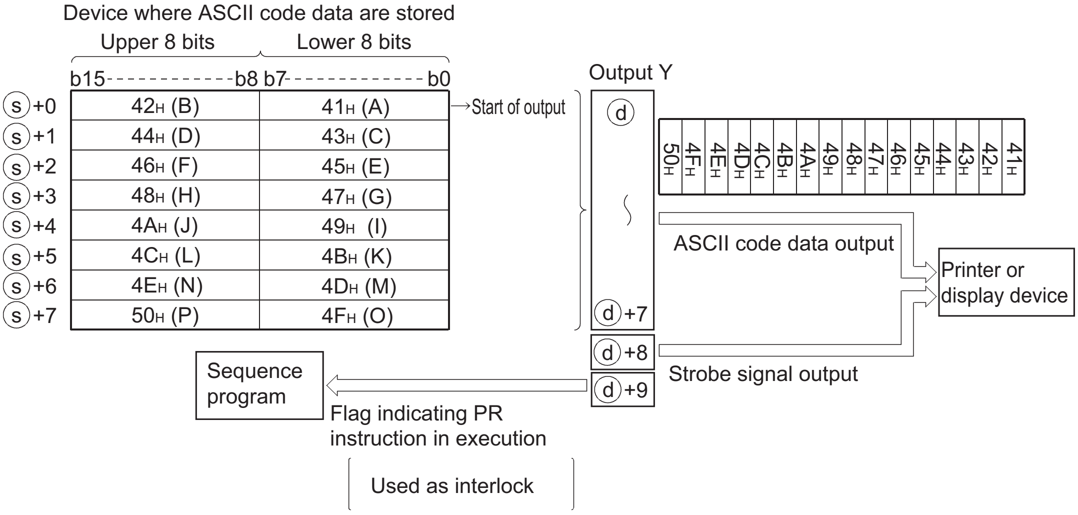
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	△*1		—			○	○	—
(d)	○(Y only)	—		—			○	—	—

*1 Local devices and file registers per program cannot be used as setting data.

Processing details

- Outputs ASCII code data stored in the device specified for (s) or ASCII code data stored in the specified device number and the following devices to the output module specified for (d). The number of characters output differs according to the ON or OFF status of SM701 (For switching the number of output characters).

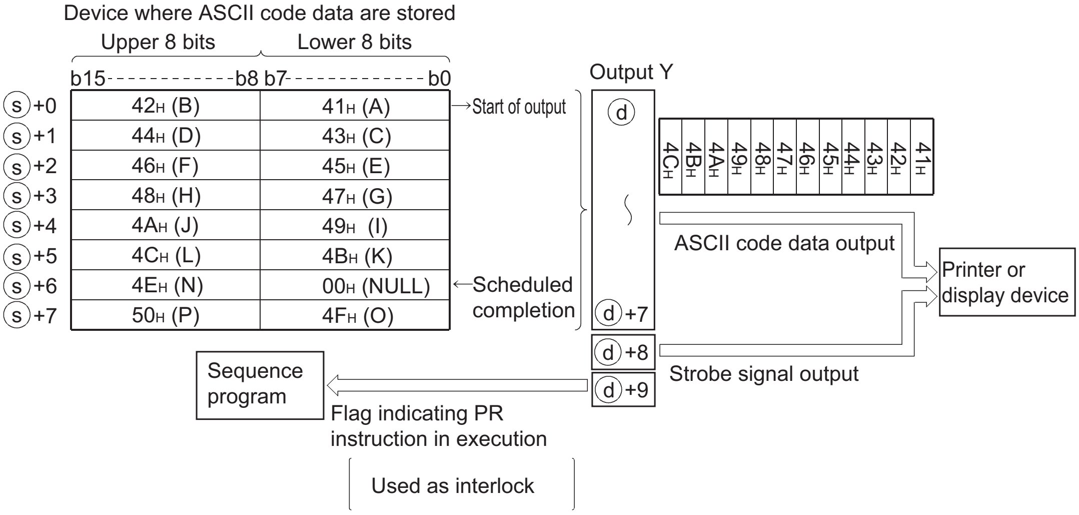
- If SM701 is ON, characters 8 points (16 characters) from the device specified for (s) will be the target of the operation.



Device where ASCII code data are stored
Upper 8 bits
Lower 8 bits
b15------------b8 b7-------------b0
Start of output
Output Y
Ⓓ
Ⓓ+7
50H 4FH 4EH 4DH 4CH 4BH 4AH 49H 48H 47H 46H 45H 44H 43H 42H 41H
ASCII code data output
Printer or display device
Ⓓ+8
Ⓓ+9
Strobe signal output
Sequence program
Flag indicating PR instruction in execution
Used as interlock

	Upper 8 bits (b15 to b8)	Lower 8 bits (b7 to b0)
Ⓢ+0	42H (B)	41H (A)
Ⓢ+1	44H (D)	43H (C)
Ⓢ+2	46H (F)	45H (E)
Ⓢ+3	48H (H)	47H (G)
Ⓢ+4	4AH (J)	49H (I)
Ⓢ+5	4CH (L)	4BH (K)
Ⓢ+6	4EH (N)	4DH (M)
Ⓢ+7	50H (P)	4FH (O)

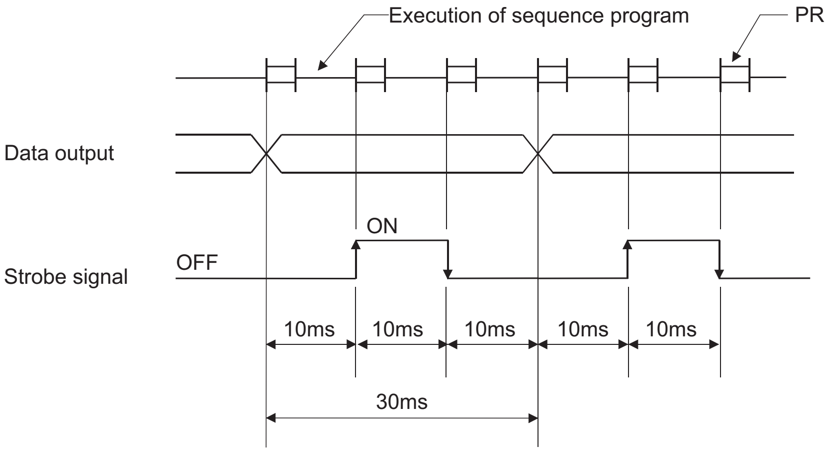
- If SM701 is OFF, everything from the device specified for (s) to the 00H code will be the target of the operation.



Device where ASCII code data are stored
Upper 8 bits
Lower 8 bits
b15------------b8 b7-------------b0
Start of output
Scheduled completion
Output Y
Ⓓ
Ⓓ+7
4CH 4BH 4AH 49H 48H 47H 46H 45H 44H 43H 42H 41H
ASCII code data output
Printer or display device
Ⓓ+8
Ⓓ+9
Strobe signal output
Sequence program
Flag indicating PR instruction in execution
Used as interlock

	Upper 8 bits (b15 to b8)	Lower 8 bits (b7 to b0)
Ⓢ+0	42H (B)	41H (A)
Ⓢ+1	44H (D)	43H (C)
Ⓢ+2	46H (F)	45H (E)
Ⓢ+3	48H (H)	47H (G)
Ⓢ+4	4AH (J)	49H (I)
Ⓢ+5	4CH (L)	4BH (K)
Ⓢ+6	4EH (N)	00H (NULL)
Ⓢ+7	50H (P)	4FH (O)

- The number of points used by the output module is 10 points from the Y address specified for (d).
- Output signals from the output module are transmitted at the rate of 30ms per character. For this reason, the time required to complete the transmission of the specified number of characters (n) will be 30ms × n (ms). At 10ms interrupt intervals, the PR instruction executes data output, strobe signal ON, and strobe signal OFF. The other instructions are executed continuously during a period between the above processes.



- In addition to the ASCII code data, the output module also outputs a strobe signal (10ms ON, 20ms OFF) from the (d) + 8 device.
- Following the execution of the PR instruction, the PR instruction execution flag ((d) + 9 device) remains ON until the completion of the transmission of the number of specified characters.
- The PR and PRC instructions can be used multiple times, but it is preferable to set the interlock with the PR instruction execution flag ((d) + 9 device) so that they will not be ON simultaneously.
- If the content of the device in which ASCII code data are stored are changed during the ASCII code data output, the modified data after change will be output.

Operation error

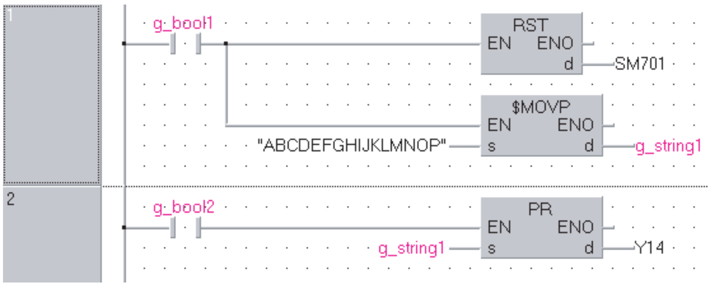
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	There is no 00H code within the range of the device specified for (s) when SM701 is OFF.	—	○	○	—	—	—

Program example

- In the following program, the string data "ABCDEFGHIJKLMNOP" is converted to ASCII code data when g_bool1 turns ON and the result is stored to g_string1 (D0), and then the stored ASCII code data in g_string1 (D0) are output to Y14 to Y1B when g_bool2 turns ON. (When SM701 is OFF)

[Structured ladder/FBD]



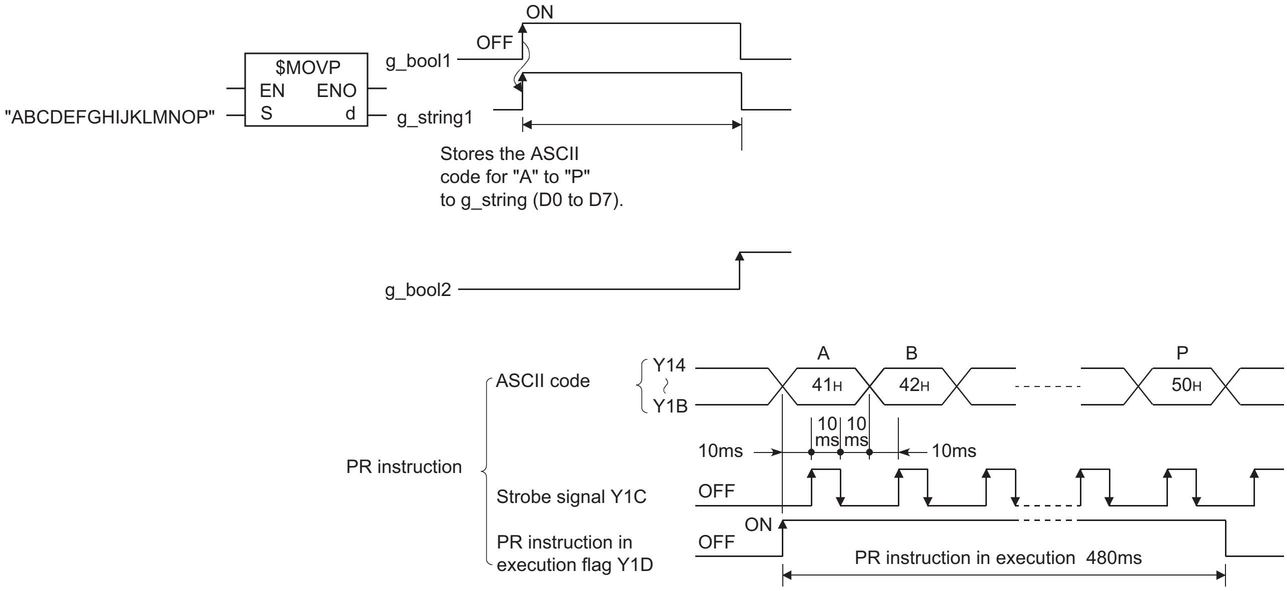
When g_bool1 turns ON, converts "ABCDEFGHIJKLMOP" to the ASCII code and outputs the result to g_string1.

When g_bool2 turns ON, outputs the ASCII code in g_string1 to Y14 to Y1B.

[ST]

```
IF g_bool1 THEN
      RST(TRUE,SM701);
      g_string1="ABCDEFGHIJKLMNOP";
END_IF;
PR(g_bool2,g_string1,Y14);
```

[Timing chart]

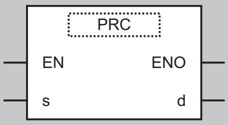


Printing comments

PRC

✗ Basic High performance Process ✗ Redundant ✗ Universal ✗ LCPU

Structured ladder/FBD



ST

```
ENO:= PRC (EN, s, d);
```

The following instruction can go in the dotted squares.

PRC

■Executing condition

Instruction	Executing condition
PRC	↑

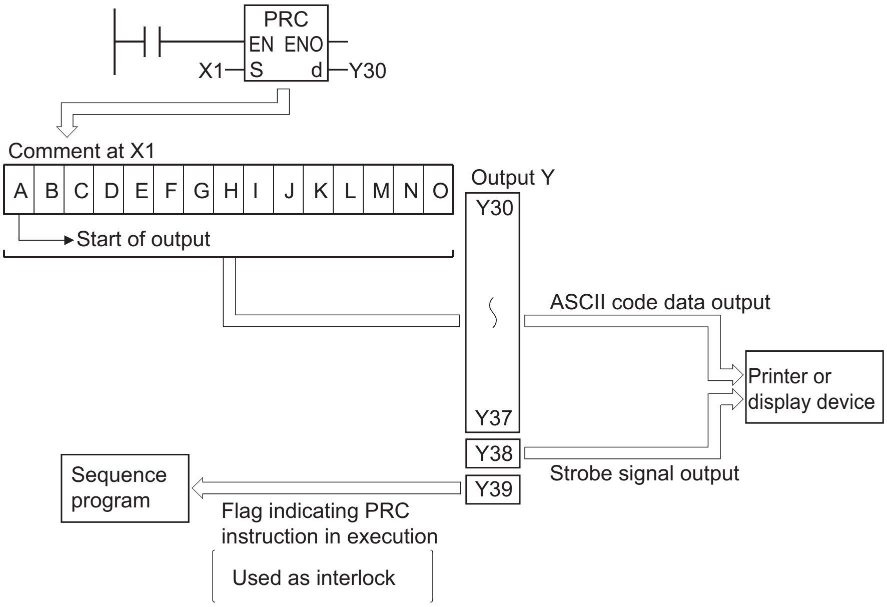
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device whose comments are printed	ANY_SIMPLE
Output argument	ENO	Execution result	Bit
	d	Start number of the output module that outputs comments	Bit

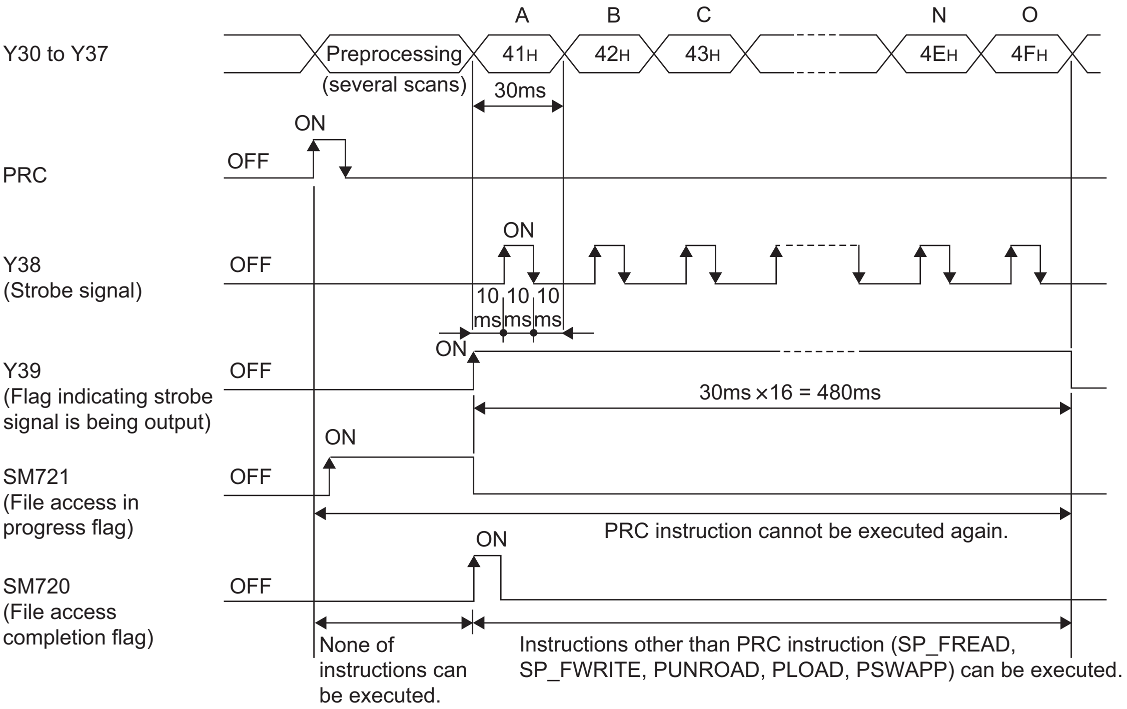
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others P,I,J,U
	Bit	Word		Bit	Word				
(s)	○	○		○			—	—	○
(d)	○(Y only)	—		—			—	—	—

Processing details

- Outputs comment data (ASCII code data) in the device specified for (s) to output module specified for (d). The number of characters output differs according to the ON or OFF status of SM701.
  - When SM701 is OFF:Comments in 32 characters
  - When SM701 is ON:Comments of upper 16 characters
  - The number of points used by the output module is 10 points from the Y address specified for (d).

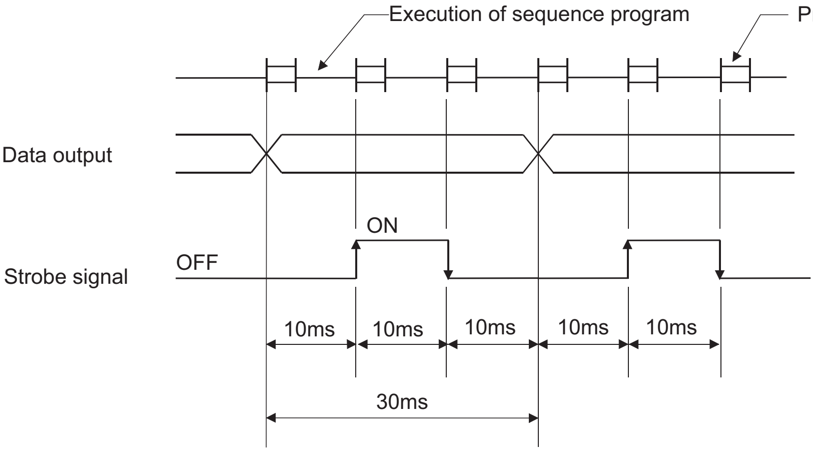


[Timing chart]



A
B
C
N
O
Y30 to Y37
Preprocessing
(several scans)
41H
42H
43H
4EH
4FH
30ms
PRC
ON
OFF
Y38
(Strobe signal)
ON
OFF
10 ms
10 ms
10 ms
Y39
(Flag indicating strobe signal is being output)
ON
OFF
30ms×16 = 480ms
SM721
(File access in progress flag)
ON
OFF
PRC instruction cannot be executed again.
SM720
(File access completion flag)
ON
OFF
None of instructions can be executed.
Instructions other than PRC instruction (SP_FREAD, SP_FWRITE, PUNROAD, PLOAD, PSWAPP) can be executed.

- Output signals from the output module are transmitted at the rate of 30ms per character. For this reason, the time required to complete the transmission of the number of specified characters will be 30ms × n (ms). At 10ms interrupt intervals, the PRC instruction executes data output, strobe signal ON, and strobe signal OFF. The other instructions are executed continuously during a period between the above processes.



- In addition to the ASCII code data, the output module also outputs a strobe signal (10ms ON, 20ms OFF) from the (d) + 8 device.
- Following the execution of the PRC instruction, the PRC instruction execution flag ((d) + 9 device) remains ON until the completion of the transmission of the number of specified characters.
- The PRC instruction can be used multiple times, but it is preferable to set the interlock with the PRC instruction execution flag ((d) + 9 device) so that they will not be ON simultaneously.
- If no comments have been registered to the device specified for (s), processing will not be performed.
- When a comment is read, SM720 turns ON for one scan after the instruction is completed. SM721 turns ON during the execution of the instruction. The PRC instruction cannot be executed while SM721 is ON. If an attempt is made, no processing is performed.

Point

- For device comments used in the PRC instruction, use comment files stored in the memory card or standard ROM. Comment files stored in the internal memory cannot be used.
- The comment file used in the PRC instruction can be set at the <<PLC File>> tab in the PLC parameter. If no comment file has been set for use by the PLC file setting, it will not be possible to output device comments with the PRC instruction.
- Do not execute the PRC instruction during an interrupt program. Otherwise, malfunction may occur.
- An error occurs when a device with no device comment is specified. Note that no error occurs even if a device with no device comment is specified when the range of device comments is set in advance with a programming tool ("Device Comment Detail Setting" on "Online Data Operation" window for writing data to the CPU module), and the device is within the range.

Operation error

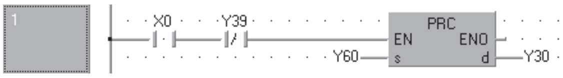
- In the following case, an operation error occurs and the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The PRC instruction is executed while device comments are written during online program change.	—	○	○	—	—	—
	A device with no device comment is specified.	—	○	○	—	—	—

Program example

- In the following program, the comment data in Y60 are output to Y30 to Y39 when X0 turns ON.

[Structured ladder/FBD]



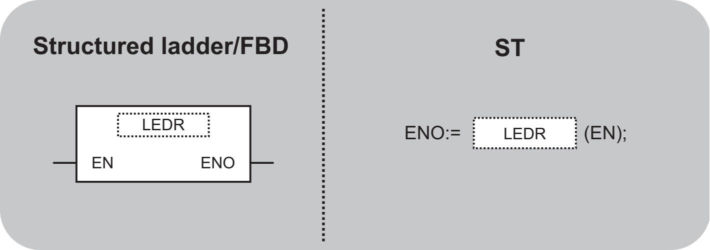
[ST]

PRC(X0 AND NOT(Y39),Y60,Y30);

Resetting error display or annunciator

LEDR

~~Basic~~ High performance Process Redundant Universal LCPU



The following instruction can go in the dotted squares.

LEDR

■Executing condition

Instruction	Executing condition
LEDR	↑

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

Resets the self-diagnostics error display so that annunciator display or operation can be continued.

With one execution of this instruction, either error display or annunciator is reset.

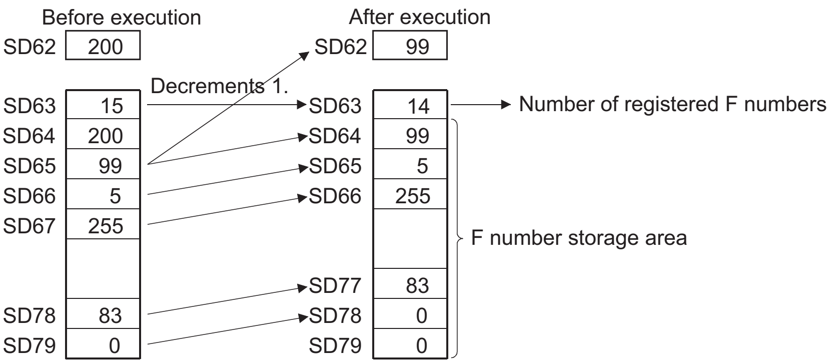
■Operation when self-diagnostics errors occur.

- When the LEDR instruction is executed, the "ERR." LED is reset. It is necessary to reset SM0, SM1, and SD0 by the user program, because they are not reset automatically. Since the cause of the self-diagnostics error being displayed has a higher priority over annunciator, no action for resetting the annunciator is taken.
- If the LEDR instruction is executed after the battery is replaced, the "BAT." LED is reset. SM51 is also turned OFF at this time.

■Operations when annunciators (F) are ON.

The following operations are performed when the LEDR instruction is executed.

- "USER" LED flickers, and is turned OFF.
- The annunciators (F) stored in SD62 and SD64 are reset, and the F numbers of SD65 to SD79 are moved up.
- The data newly stored in SD64 are transmitted to SD62.
- The value in SD63 is decremented by - 1. However, if SD63 is 0, it remains 0.



- The defaults for the error item numbers set in special registers SD207 to SD209 and order of priority are given in the table below.

Priority	Factor number (Hexadecimal)	Description	Remarks
1	1	AC DOWN SINGLE PS. DOWN SINGLE PS. ERROR	Power off Redundant base unit power supply voltage drop (QCPU (Q mode) only) Redundant power supply module fault.(QCPU (Q mode) only)
2	2	UNIT VERIFY ERR. FUSE BREAK OFF SP. UNIT ERROR SP. UNIT DOWN	I/O module verify error (QCPU (Q mode) only) Blown fuse (QCPU (Q mode) only) Special function module verify error Intelligent function module verification error (QCPU (Q mode) only) Intelligent function module error (LCPU only)
3	3	OPERATION ERROR LINK PARA. ERROR SFCP OPE. ERROR SFCP EXE. ERROR REMOTE PASS. FAIL SNTP OPE. ERROR	Operation error Link parameter error (QCPU (Q mode) only) SFC instruction operation error (PU (Q mode) only) SFC program execution error (QCPU (Q mode) only) Remote password error (QCPU (Q mode) only) SNTP error (LCPU only)
4	4	ICM. OPE. ERROR FILE OPE. ERROR EXTEND INST. ERROR OPE. MODE DIFF. CAN'T EXE. MODE TRK.TRANS. ERR. TRK.SIZE ERROR TRK.DISCONNECT FLASH ROM ERROR	Memory card operation error File access error Extended instruction error (QCPU (Q mode) only) Operation status, switch mismatch (QCPU (Q mode) only) Current mode-time function execution disabled (QCPU (Q mode) only) Tracking data transmission error (QCPU (Q mode) only) Tracking capacity excess error (QCPU (Q mode) only) Tracking cable not connected, failure (QCPU (Q mode) only) Flash ROM access count exceeded error (LCPU only)
5	5	PRG.TIME OVER	Constant scan setting time over error Low speed execution monitoring time over error (QCPU (Q mode) only)
6	6	CHK instruction	—
7	7	Annunciators	—
8	8	LED instruction	—
9	9	BATTERY ERR.	—
10	A	Clock data	—
11	B	CAN'T SWITCH STANDBY SYS. DOWN MEM. COPY EXE.	System switching error (QCPU (Q mode) only) Standby system not started/stop error (QCPU (Q mode) only) Memory copy function executed (QCPU (Q mode) only)
12	C	DISPLAY ERROR	Display unit error (LCPU only)

- If the highest priority is given to the annunciator, it can be reset with priority by the LEDR instruction. (For Basic model QCPU, High Performance model QCPU, Process CPU, and Redundant CPU)

Operation error

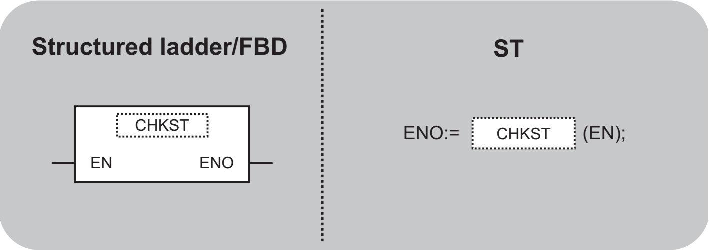
- There is no operation error.

7.10 Debug/Error Diagnostics Instructions

Special format error check

CHKST, CHK

Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

CHKST, CHK

■Executing condition

Instruction	Executing condition
CHKST, CHK	Non-conditional execution

■Argument

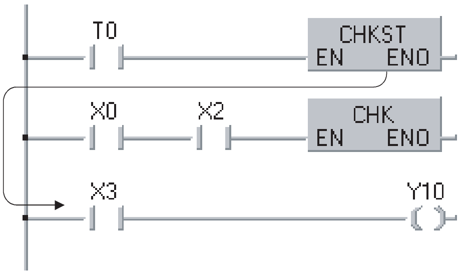
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

■CHKST

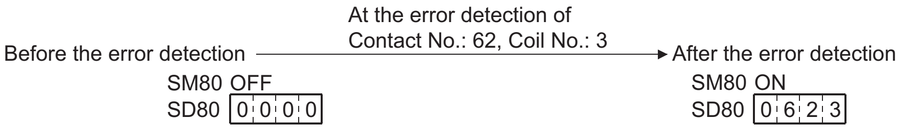
- The CHKST instruction is the instruction that starts the CHK instruction.
- If the command for the CHKST instruction is OFF, execution jumps from the CHK instruction to the next instruction.
- If the command for the CHKST instruction is ON, the CHK instruction is executed.



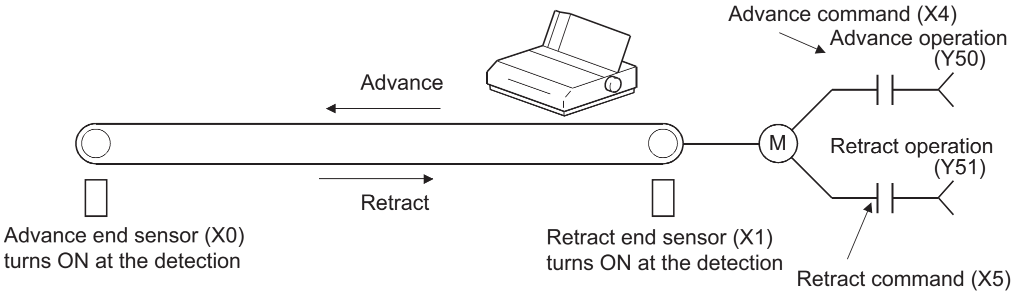
When T0 is OFF, program jumps to the instruction next to the CHK instruction.

■CHK

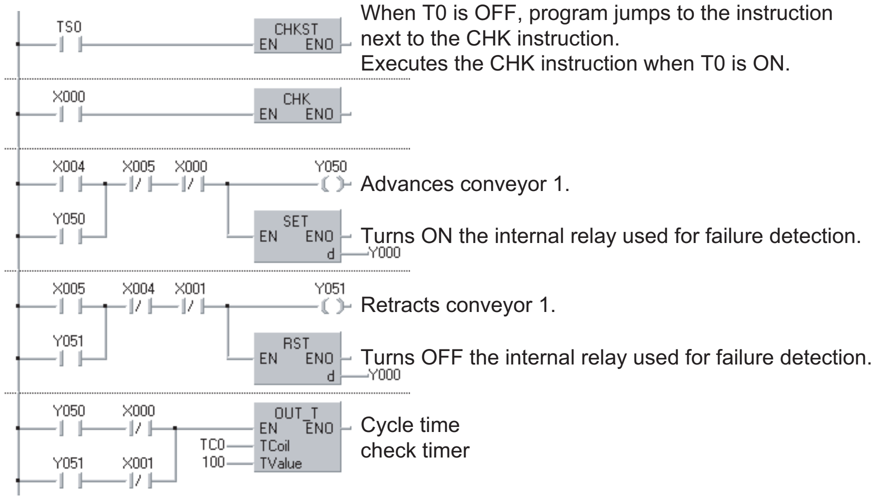
- The CHK instruction is the instruction used for the bidirectional operation as shown below to confirm the nature of the system error. When the CHK instruction is executed, an error diagnostic check is performed with the specified check conditions, and if a failure is detected, SM80 is turned ON, and the failure number is stored to SD80 as a BCD value. The error code "9010" will be returned if an error is detected. The contact number where the error was discovered is stored to the upper 3 digits of SD80 (refer to (3)), and the coil number where the error was detected (refer to (2)) is stored to the lower 1 digit of SD80.


At the error detection of
Contact No.: 62, Coil No.: 3
Before the error detection
After the error detection
SM80 OFF
SD80 0 0 0 0
SM80 ON
SD80 0 6 2 3

- The contact instruction prior to the CHK instruction does not control the execution of the CHK instruction, but rather sets the check conditions.



- A ladder such as the one shown below can be created to perform a cycle time-out check for the system shown above.



When T0 is OFF, program jumps to the instruction next to the CHK instruction.
Executes the CHK instruction when T0 is ON.

Advances conveyor 1.

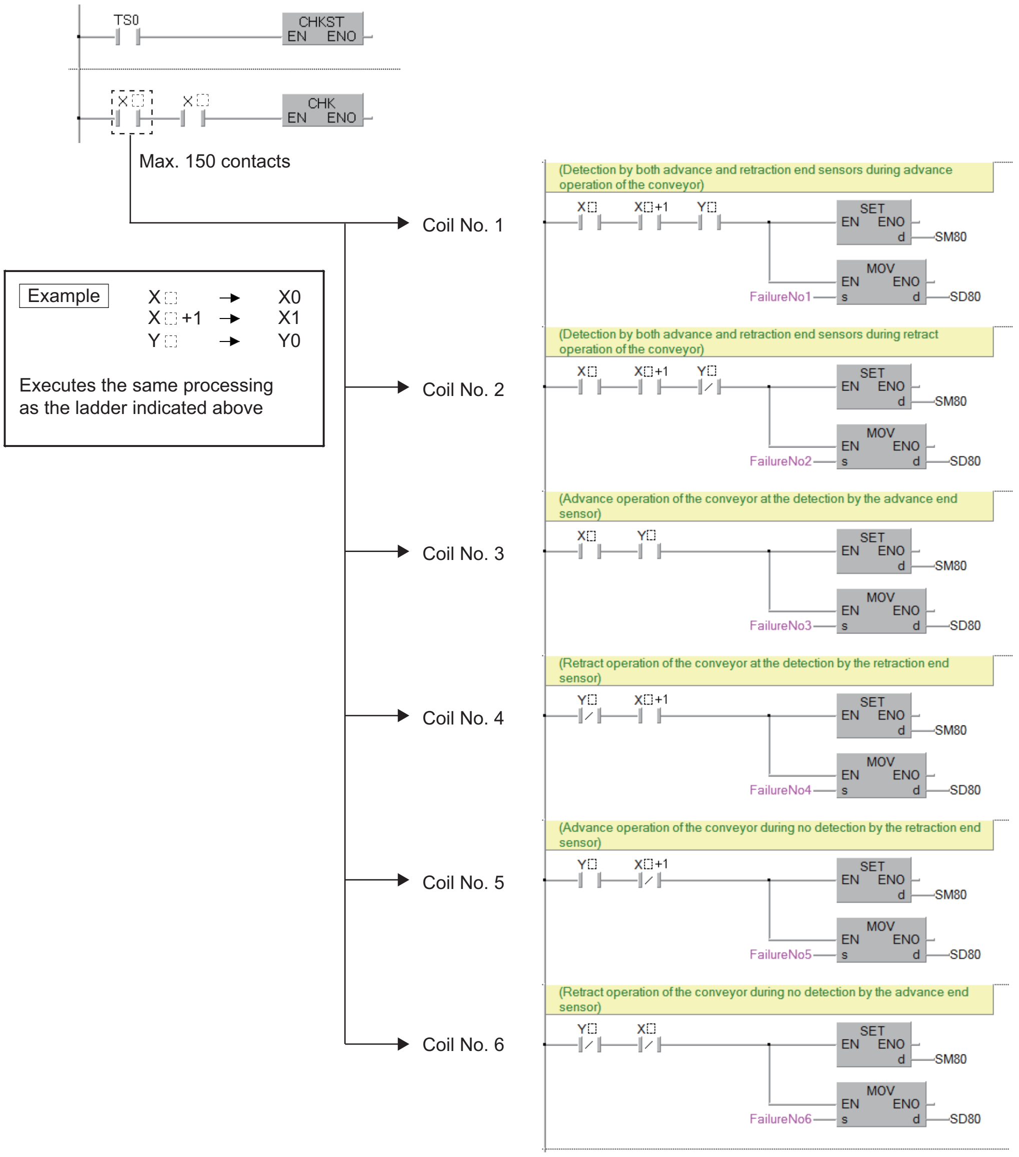
Turns ON the internal relay used for failure detection.

Retracts conveyor 1.

Turns OFF the internal relay used for failure detection.

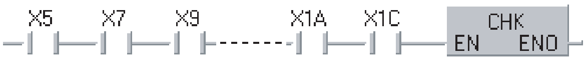
Cycle time check timer

- The following points should be taken into consideration when creating a ladder using the CHK instruction.
  - The contact numbers for the advance edge sensor and the retract edge sensor (X□) must always be in serial. Further, the contact number (X□) for the advance edge sensor should be lower than that for the retract edge.
  - Controls for the advance edge sensor contact number (X□) and output with the identical number (Y□) so that the output turns on when the advance operation is in progress and turns off when the retract operation is in progress. Output (Y□) is treated as an internal relay, and cannot be output to an external device.
- Depending on the specified contact, the CHK instruction undergoes processing identical to that shown for the ladder below. A coil number (1 to 6) that is detected by error detection is stored in lower one digit of SD80.



TS0
CHKST
EN ENO
X□ X□
CHK
EN ENO
Max. 150 contacts
Example
X□ → X0
X□+1 → X1
Y□ → Y0
Executes the same processing as the ladder indicated above
Coil No. 1
Coil No. 2
Coil No. 3
Coil No. 4
Coil No. 5
Coil No. 6
(Detection by both advance and retraction end sensors during advance operation of the conveyor)
(Detection by both advance and retraction end sensors during retract operation of the conveyor)
(Advance operation of the conveyor at the detection by the advance end sensor)
(Retract operation of the conveyor at the detection by the retraction end sensor)
(Advance operation of the conveyor during no detection by the retraction end sensor)
(Retract operation of the conveyor during no detection by the advance end sensor)
X□ X□+1 Y□
Y□ X□+1
Y□ X□
SET
EN ENO
d
SM80
MOV
EN ENO
s d
SD80
FailureNo1
FailureNo2
FailureNo3
FailureNo4
FailureNo5
FailureNo6

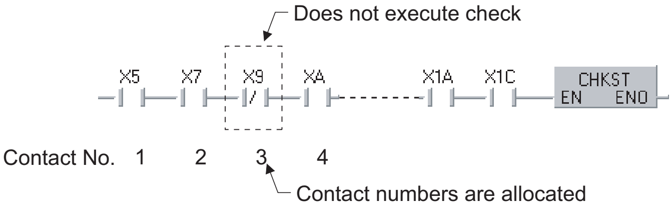
- Numbers 1 to 150 from the left base line have been assigned as contact numbers during error detection.



Contact No. → 1 2 3 ---------- 149 150

- Reset SM80 and SD80 prior to forcing the execution of the CHK instruction. After the execution of the CHK instruction, it cannot be performed once again until SM80 and SD80 have been reset. (The content of SM80 and SD80 will be preserved until they are reset by user.)
- The CHKST instruction must be placed before the CHK instruction. An error is returned if an instruction other than the LD, LDI, AND or ANI instruction is used between the CHK instruction and the CHKST instruction. (Error code: 4235)
- The CHK instruction can be written at any step of the program. However, there is a limit in the number of uses of the CHK instruction. An error is returned if the CHK instruction is used exceeding the number of uses specified as follows. (Error code: 4235)
  - Can be used up to two places in all program files being executed.
  - Can be used only one place in a single program file.

- Place LD and AND instructions prior to the CHK instruction to establish a check condition. Check conditions cannot be set using other contact instructions. If a check condition has been set with the LDI or ANI instruction, the processing for the check condition they specify will not be performed. However, contact numbers during error detection can also be assigned to the LDI and ANI instructions.



- The error detection method differs according to whether SM710 is ON or OFF.
  - If SM710 is OFF, checks are performed on coil numbers 1 through 6 in order of contact number. When the CHK instruction is executed, checks are performed on coils of contact No.1 in order from coil No.1 through coil No.6, then move on to contact No. 2 and check the coils in order from coil No.1 through coil No.6. The CHK instruction is completed when coil No. 6 of contact No. n is checked.
  - If SM710 is ON, checks are performed on contact numbers 1 through n in order of coil number. When the CHK instruction is executed, checks are performed on the ladder with coil No. 1 in order from contact No. 1 through contact No. n, then move on to the ladder with coil No. 2 and checks the ladder in order from contact No. 1 through contact No. n. The CHK instruction is completed when contact No. n of coil No. 6 is checked.
- If more than one error is detected, the number of the first error detected is stored. Error numbers detected after this are ignored.
- The CHK instruction cannot be used in a low speed execution type program. If a low speed execution type program is set in a program file containing the CHK instruction, an operation error is returned, and the CPU module operation is suspended.

Operation error

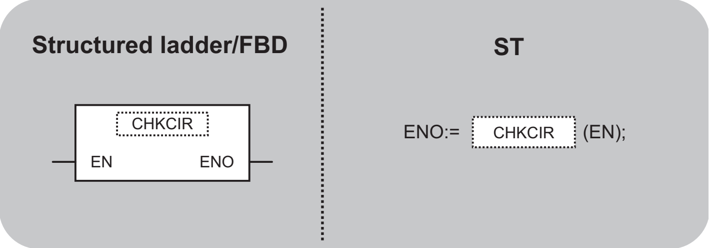
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4235	There is a parallel ladder. There are more than 150 contact instructions. The CHK instruction is not executed following the CHKST instruction. The CHK instruction is executed when no CHKST instruction has been executed. The CHKST and CHK instructions are used in a low speed execution type program. There is a instruction other than the LD, LDI, AND or ANI instruction between the CHK instruction and the CHKST instruction. The CHK instruction is used at three places or more in all program files being executed. The CHK instruction is used at two places or more in a single program file.	—	○	○	○	—	—

Changing check format of the CHK instruction

CHKCIR, CHKEND

Basic ✗ | High performance | Process | Redundant | Universal ✗ | LCPU ✗



```
ENO:= CHKCIR (EN);
```

Any of the following instruction can go in the dotted squares.

CHKCIR, CHKEND

■Executing condition

Instruction	Executing condition
CHKCIR, CHKEND	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

- The check ladder pattern that is used in the CHK instruction can be updated to any format desired. The actual error checks are performed with the CHKST and CHK instructions.
- Error checks are performed according to the check conditions specified by the CHK instruction and the ladder pattern described between the CHKCIR and CHKEND instructions.

Point

For details on the CHKST and CHK instructions, refer to ☞ Page 468 Special format error check.

To change the check format of the CHK instruction using the CHKCIR to CHKEND instruction loop, create a ladder with index setting (Z0).

- The device numbers indicated at check conditions (X2 and X8 in the figure below) are index setting values for each device number (with the exception of annunciators (F)) described in the ladder patterns. For example, the device number of X10Z0 on the ladder pattern is handled as X12 when the check condition is specified to X2, and X18 when the check condition is specified to X8.
- However, the order in which error detection is executed differs depending on whether SM710 is ON or OFF.

If SM710 is OFF, checks are performed on coil numbers 1 through the end in order of contact number.

The contact number 1 is checked with the ladder pattern from the coil number 1 to n. After that, the contact number 2 is checked with the same ladder pattern from the coil number 1 to n.

If SM710 is ON, checks are performed on contact numbers 1 through the end in order of coil number.

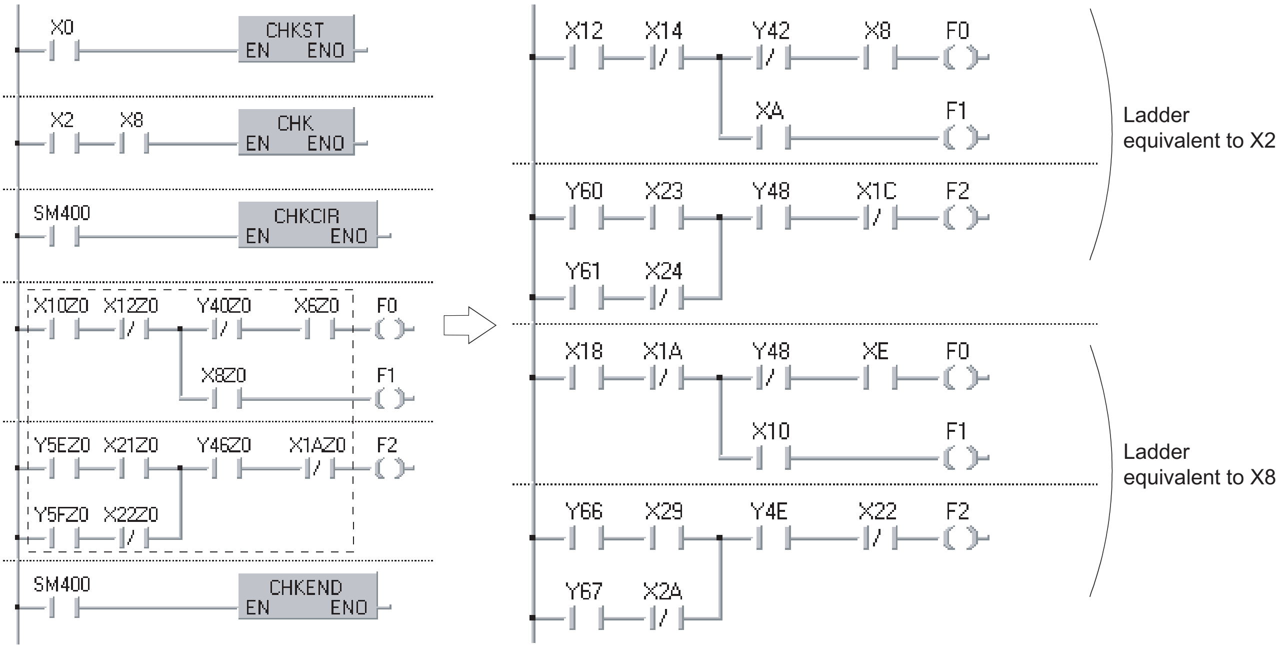
With the ladder pattern of the coil number 1, the contact numbers from 1 to n are checked. After that, with the ladder pattern of the coil number 2, the contact numbers from 1 to n are checked.

Ex.

Following is the example when SM710 is OFF.

[Left: Specified ladder between CHKCIR and CHKEND instructions]

[Right: Order of check in CPU module]

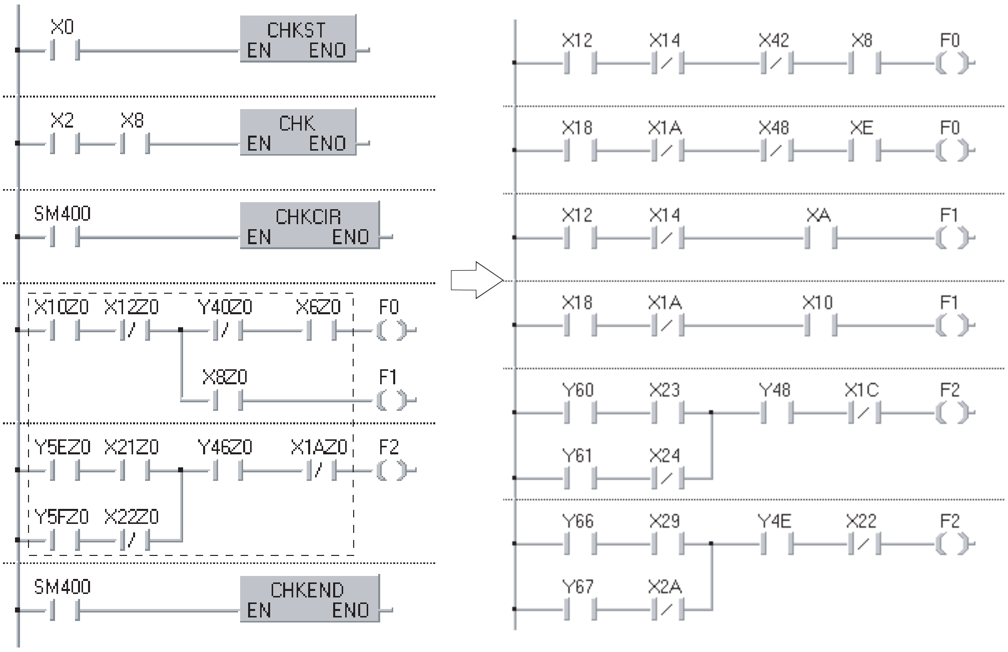


Ex.

Following is the example when SM710 is ON.

[Left: Specified ladder between CHKCIR and CHKEND instructions]

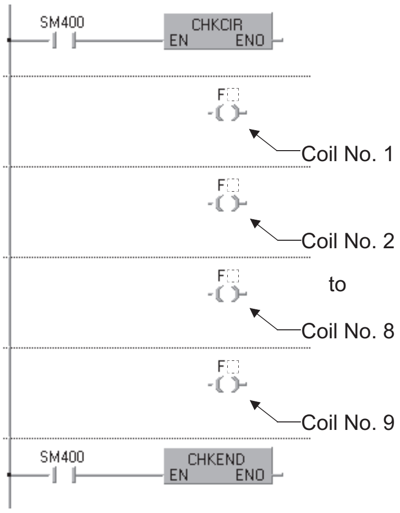
[Right: Order of check in CPU module]



- Error check checks the ON/OFF status of OUT F□ by using the ladder pattern in the various check conditions. In all check conditions, SM80 is turned ON if even one of the OUT F□ is ON in a ladder pattern. Further, the error numbers (contact numbers and coil numbers) corresponding to the OUT F□ which are turned ON are stored to SD80 in BCD order.
- The instructions that can be used in ladder patterns are as follows.

Type	Instruction
Contacts	LD, LDI, AND, ANI, OR, ORI, ANB, ORB, MPS, MPP, MRD, and comparison operation instructions
Coil	OUT F□

- The devices can be used for ladder pattern contacts are input (X) and output (Y).
- Only annunciators (F) can be used in ladder pattern coils. However, since annunciators (F) are used as a dummy, any value can be set for an annunciator (F). Further, they can overlap with no difficulties.
- ON/OFF controls can be performed without error if an annunciator (F) used during the execution of the CHK instruction has the same number as an annunciator (F) used in some other context than the CHK instruction. They are treated differently during the CHK instruction than they are in the different context.
- Since the annunciators (F) used in the CHK instruction do not turn ON/OFF actually, they are turned ON/OFF if monitored by a peripheral.
- A ladder pattern can be created up to 256 steps. Further, OUT F□ can use up to 9 coils.
- Coil numbers for ladders specified within the CHKCIR to CHKEND instruction loop are assigned with coil numbers from 1 to 9, from top to bottom.



- The CHKCIR and CHKEND instructions can be written at any step in the program desired. It can be used in up to two locations in all program files being executed. However, the CHKCIR and CHKEND instructions cannot be used in more than 1 location in a single program file.
- The CHKCIR and CHKEND instructions cannot be used in low speed execution type programs. If a program file in which the CHKCIR or CHKEND instruction is described is set as a low speed execution type program, an operation error occurs, and the High Performance model QCPU operation is suspended.

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4230	The CHKEND instruction is not executed following the execution of the CHKCIR instruction. The CHKEND instruction is executed although no CHKCIR instruction has been executed.	—	○	○	○	—	—
4235	The CHKCIR to CHKEND instruction loop is set three or more times in all program files. The CHKCIR or CHKEND instruction loop is set two or more times in a single program file. The CHKST and CHK instructions are used in a low speed execution type program. There are 10 or more F instances in a ladder pattern. A ladder pattern has 257 or more steps. A device has been encountered which cannot be used in a ladder pattern. The index setting has been performed on a ladder pattern device.	—	○	○	○	—	—

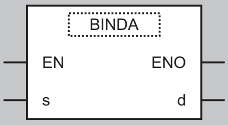
7.11 String Processing Instructions

16-/32-bit BIN data to decimal ASCII data conversion

BINDA(P), DBINDA(P)

~~Basic~~ High performance Process Redundant Universal LCPU

Structured ladder/FBD

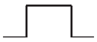
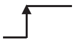


ST

ENO:= BINDA (EN, s, d);

Any of the following instruction can go in the dotted squares.
BINDA, BINDAP, DBINDA, DBINDAP

■Executing condition

Instruction	Executing condition
BINDA, DBINDA	
BINDAP, DBINDAP	

■Argument

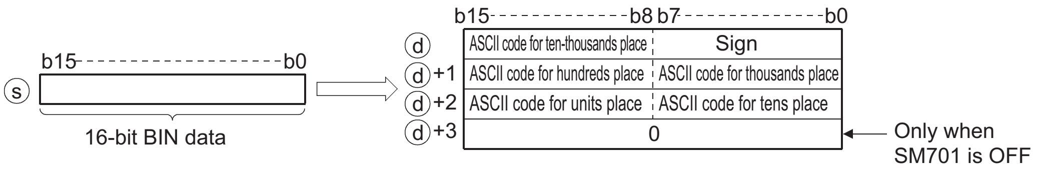
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	BIN data to be converted to ASCII code data	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores conversion result	String (8)/(12)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○	○		○					—
(d)	—	○		—					—

Processing details

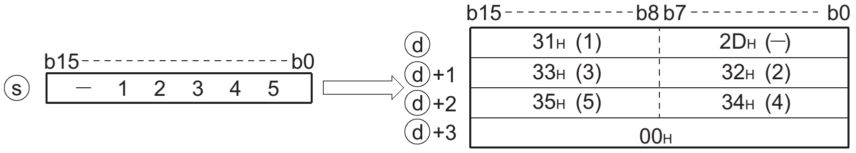
■BINDA(P)

- Converts each digit number of decimal notation of the 16-bit BIN data specified for (s) to ASCII code data, and stores the results to the device specified for (d) and the following devices.

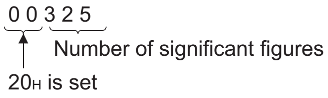


b15
b0
s
16-bit BIN data
b15
b8 b7
b0
d
ASCII code for ten-thousands place
Sign
d +1
ASCII code for hundreds place
ASCII code for thousands place
d +2
ASCII code for units place
ASCII code for tens place
d +3
0
Only when
SM701 is OFF

For example, if -12345 is specified for (s), the results are stored to the devices from (d) as shown below.



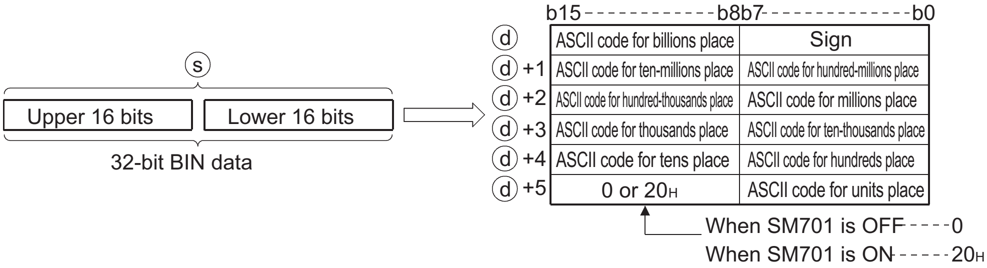
- BIN value between -32768 and 32767 can be specified for (s).
- The operation results stored to (d) are as follows.
  - The sign 20H is stored if the BIN value is positive, and the sign 2DH is stored if it is negative.
  - The sign 20H is stored for the leading zeros of significant figures. (Zero suppression is performed.)



- The storage of data in devices specified for (d)+3 differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).
- When SM701 is OFFStores 0
- When SM701 is ONDoes not change

■DBINDA(P)

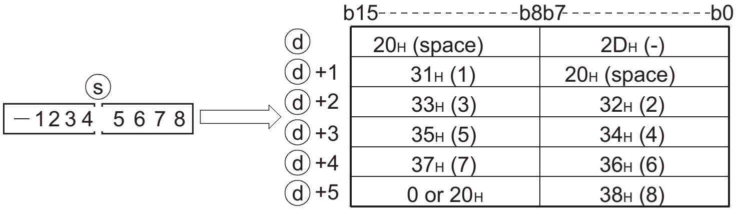
- Converts each digit number of decimal notation of the 32-bit BIN data specified for (s) to ASCII code data, and stores the results to the device specified for (d) and the following devices.



	b15 - - b8	b7 - - b0
(d)	ASCII code for billions place	Sign
(d) +1	ASCII code for ten-millions place	ASCII code for hundred-millions place
(d) +2	ASCII code for hundred-thousands place	ASCII code for millions place
(d) +3	ASCII code for thousands place	ASCII code for ten-thousands place
(d) +4	ASCII code for tens place	ASCII code for hundreds place
(d) +5	0 or 20_H	ASCII code for units place

When SM701 is OFF - - - - 0
When SM701 is ON - - - - - 20_H

For example, if the value -12345678 has been specified for (s), the results will be stored to the devices from (d) as shown below.



	b15 - - b8	b7 - - b0
(d)	20_H (space)	$2D_H$ (-)
(d) +1	31_H (1)	20_H (space)
(d) +2	33_H (3)	32_H (2)
(d) +3	35_H (5)	34_H (4)
(d) +4	37_H (7)	36_H (6)
(d) +5	0 or 20_H	38_H (8)

- BIN value between -2147483648 and 2147483647 can be specified for (s).
- The operation results stored to (d) are stored in the following way.
  - The sign 20H is stored if the BIN value is positive, and the sign 2DH is stored if it is negative.
  - The sign 20H is stored for the leading zeros of significant figures. (Zero suppression is performed.)

$$\underbrace{0\ 0}_{20_H \text{ is set}}\ \underbrace{1\ 2\ 0\ 3\ 4\ 5\ 6\ 0}_{\text{Number of significant figures}}$$

- The data stored in the upper 8 bits of the device specified for (d)+5 differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).
- When SM701 is OFFStores 0
- When SM701 is ONStores 20H

Operation error

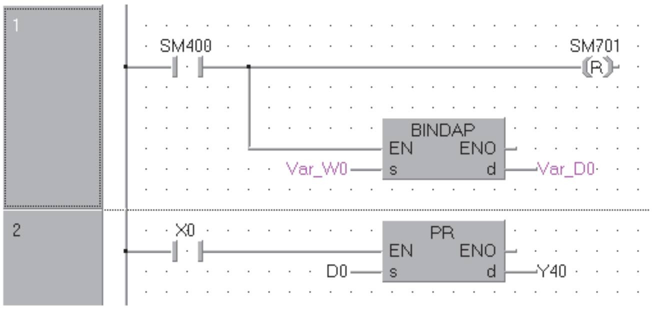
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

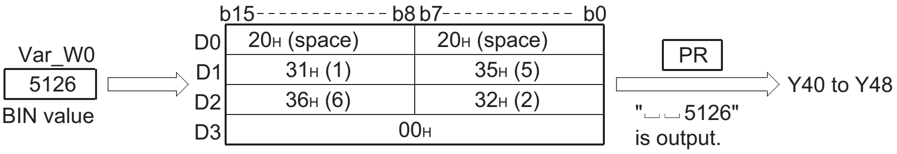
- In the following program, the 16-bit BIN data in Var_W0 are converted to the decimal ASCII code data, and the results are output to Y40 to Y48 by the PR instruction. (Global label Var_D0 is assigned to device D0.)

[Structured ladder/FBD]



[ST]
RST(SM400,SM701);
BINDAP(SM400,Var_W0,Var_D0);
PR(X0,D0,Y40);

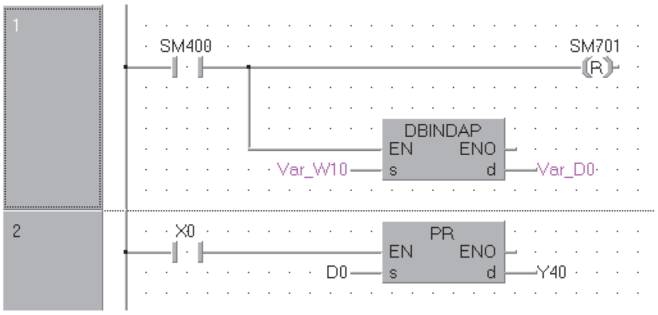
[Operation]
The ASCII code data are output to Y40 to Y48 by the execution of the PR instruction when X0 turns ON.
Since SM701 is OFF, the PR instruction outputs ASCII code data up to 00H.



	b15------------b8	b7------------b0
D0	20H (space)	20H (space)
D1	31H (1)	35H (5)
D2	36H (6)	32H (2)
D3	00H	

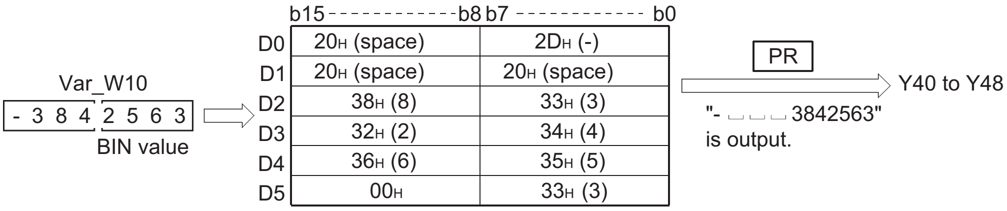
- In the following program, the 32-bit BIN data in Var_W10 are converted to the decimal ASCII code data, and the results are output to Y40 to Y48 by the PR instruction. (Global label Var_D0 is assigned to device D0.)

[Structured ladder/FBD]



[ST]
RST(SM400,SM701);
DBINDAP(SM400,Var_W10,Var_D0);
PR(X0,D0,Y40);

[Operation]
The ASCII code data are output to Y40 to Y48 by the execution of the PR instruction when X0 turns ON.
Since SM701 is OFF, the PR instruction outputs ASCII code data up to 00H.



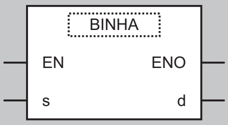
	b15------------b8	b7------------b0
D0	20H (space)	2DH (-)
D1	20H (space)	20H (space)
D2	38H (8)	33H (3)
D3	32H (2)	34H (4)
D4	36H (6)	35H (5)
D5	00H	33H (3)

16-/32-bit BIN data to hexadecimal ASCII data conversion

BINHA(P), DBINHA(P)

~~Basic~~ High performance Process Redundant Universal LCPU

Structured ladder/FBD



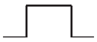
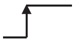
ST

ENO:= BINHA (EN, s, d);

Any of the following instruction can go in the dotted squares.

BINHA, BINHAP, DBINHA, DBINHAP

■Executing condition

Instruction	Executing condition
BINHA, DBINHA	
BINHAP, DBINHAP	

■Argument

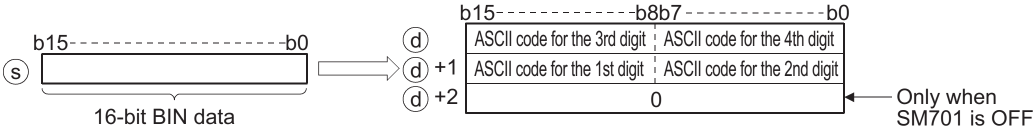
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	BIN data to be converted to ASCII code data	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores conversion result	String (6)/(10)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○	○		○					—
(d)	—	○		—					—

Processing details

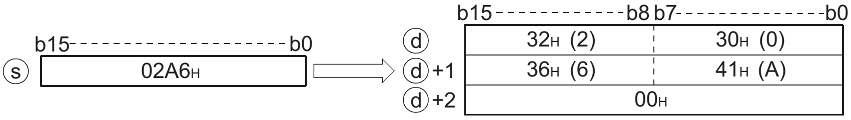
■BINHA(P)

- Converts each digit number of hexadecimal notation of the 16-bit BIN data specified for (s) to ASCII code data, and stores the results to the device specified for (d) and the following devices.



	b15–b8	b7–b0
d	ASCII code for the 3rd digit	ASCII code for the 4th digit
d +1	ASCII code for the 1st digit	ASCII code for the 2nd digit
d +2	0	

For example, if 02A6H is specified for (s), the results are stored to the devices from (d) as shown below.



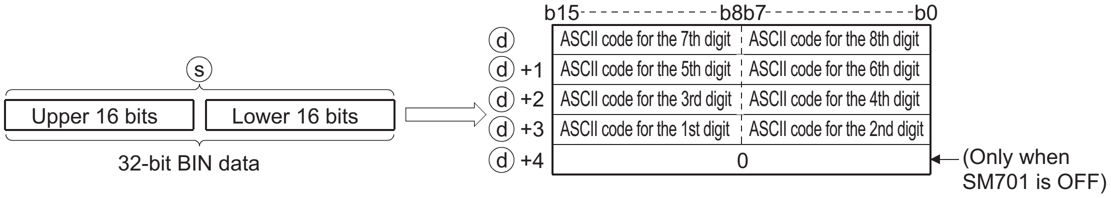
	b15–b8	b7–b0
d	32H (2)	30H (0)
d +1	36H (6)	41H (A)
d +2	00H	

- The BIN data specified for (s) can be in the range from 0H to FFFFH.
- The operation results stored to (d) are processed as 4-digit hexadecimal values. For this reason, zeros which are significant figures on the left side of the value are processed as 0. (No zero suppression is performed.)
- The data to be stored to the device specified for (d)+2 differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).

When SM701 is OFFStores 0
When SM701 is ONDoes not change

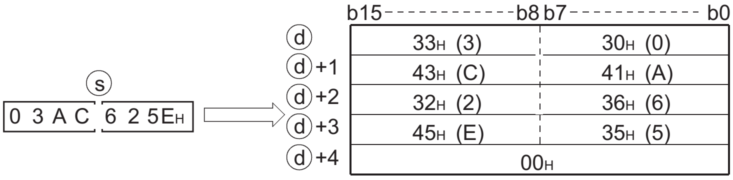
■DBINHA(P)

- Converts each digit number of hexadecimal notation of the 32-bit BIN data specified for (s) to ASCII code data, and stores the results to the device specified for (d) and the following devices.



	b15–b8	b7–b0
d	ASCII code for the 7th digit	ASCII code for the 8th digit
d +1	ASCII code for the 5th digit	ASCII code for the 6th digit
d +2	ASCII code for the 3rd digit	ASCII code for the 4th digit
d +3	ASCII code for the 1st digit	ASCII code for the 2nd digit
d +4	0	

For example, if the value 03AC625EH is specified for (s), the results are stored to the devices from (d) as shown below.



	b15–b8	b7–b0
d	33H (3)	30H (0)
d +1	43H (C)	41H (A)
d +2	32H (2)	36H (6)
d +3	45H (E)	35H (5)
d +4	00H	

- The BIN data specified for (s) can be in the range from 0H to FFFFFFFFH.
- The operation results stored to (d) are processed as 8-digit hexadecimal values. For this reason, zeros which are significant figures on the left side of the value are processed as 0. (No zero suppression is performed.)
- The data to be stored to the device specified for (d)+4 differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).
  - When SM701 is OFFStores 0
  - When SM701 is ONDoes not change

Operation error

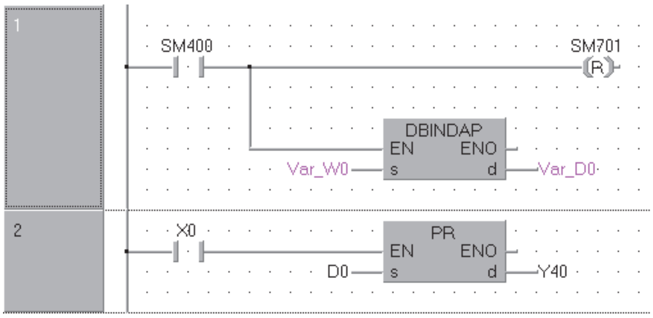
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the 16-bit BIN data in Var_W0 are converted to the hexadecimal ASCII code data, and the results are output to Y40 to Y48 by the PR instruction. (Global label Var_D0 is assigned to device D0.)

[Structured ladder/FBD]



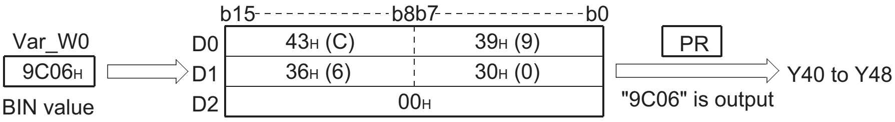
[ST]

```
RST(SM400,SM701);
BINHAP(SM400,Var_W0,Var_D0);
PR(X0,D0,Y40);
```

[Operation]

The ASCII code data are output to Y40 to Y48 by the execution of the PR instruction when X0 turns ON.

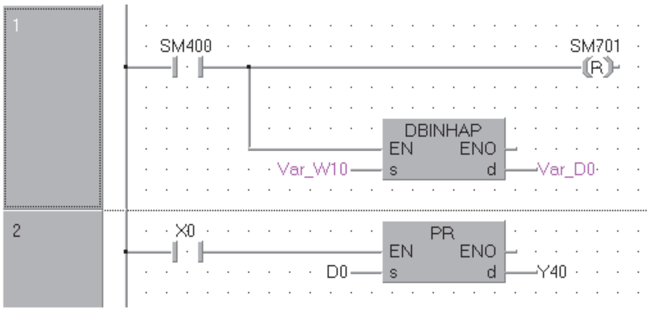
Since SM701 is OFF, the PR instruction outputs ASCII code data up to 00H.



	b15 - b8	b7 - b0
D0	43H (C)	39H (9)
D1	36H (6)	30H (0)
D2	00H	

- In the following program, the 32-bit BIN data in Var_W10 are converted to the hexadecimal ASCII code data, and the results are output to Y40 to Y48 by the PR instruction. (Global label Var_D0 is assigned to device D0.)

[Structured ladder/FBD]



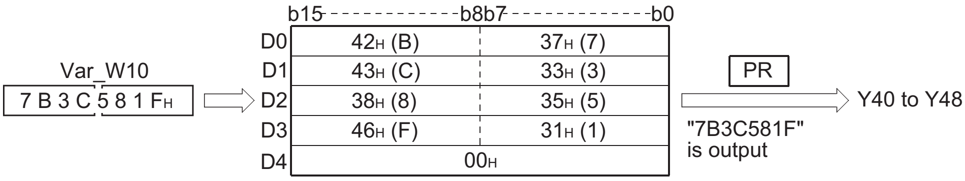
[ST]

```
RST(SM400,SM701);
DBINHAP(SM400,Var_W10,Var_D0);
PR(X0,D0,Y40);
```

[Operation]

The ASCII code data are output to Y40 to Y48 by the execution of the PR instruction when X0 turns ON.

Since SM701 is OFF, the PR instruction outputs ASCII code data up to 00H.

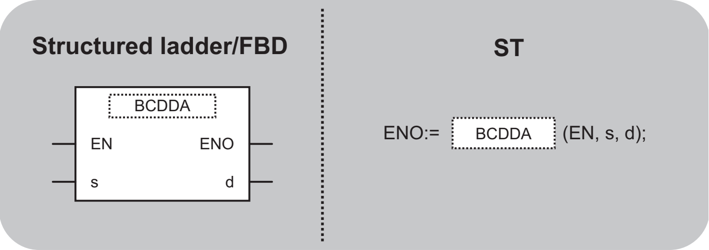


	b15 - b8	b7 - b0
D0	42H (B)	37H (7)
D1	43H (C)	33H (3)
D2	38H (8)	35H (5)
D3	46H (F)	31H (1)
D4	00H	

4-/8-digit BCD data to decimal ASCII data conversion

BCDDA(P), DBCDDA(P)

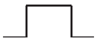
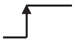
Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

BCDDA, BCDDAP, DBCDDA, DBCDDAP

■Executing condition

Instruction	Executing condition
BCDDA, DBCDDA	
BCDDAP, DBCDDAP	

■Argument

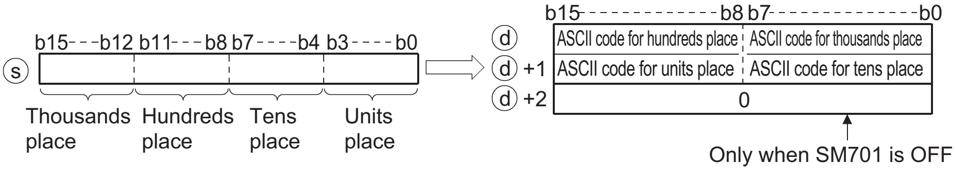
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	BCD data to be converted to ASCII code data	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores conversion result	String (6)/(10)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○	○		○					—
(d)	—	○		—					—

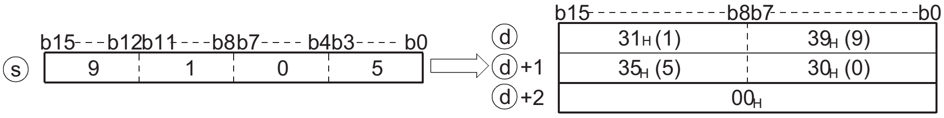
Processing details

■BCDDA(P)

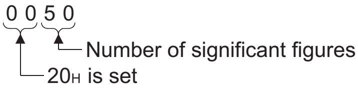
- Converts each digit number of hexadecimal notation of the 4-digit BCD data specified for (s) to ASCII code data, and stores the results to the device specified for (d) and the following devices.


b15 - - b12 b11 - - b8 b7 - - - b4 b3 - - - b0
s
Thousands place
Hundreds place
Tens place
Units place
b15 - - b8 b7 - - b0
d
d +1
d +2
ASCII code for hundreds place
ASCII code for thousands place
ASCII code for units place
ASCII code for tens place
0
Only when SM701 is OFF

For example, if the value 9105 is specified for (s), the results are stored to the devices from (d) as shown below.



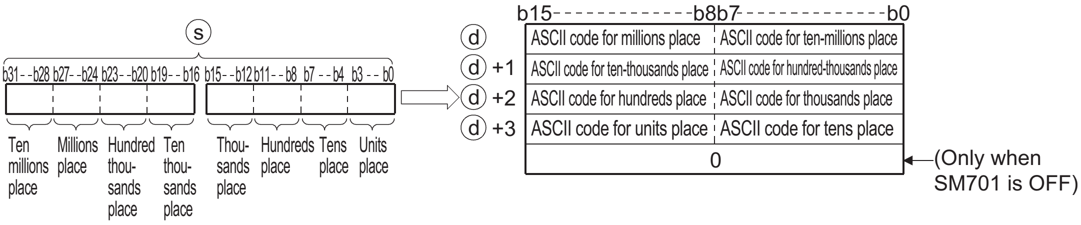
- The BCD data specified for (s) can be in the range of from 0 to 9999.
- For the results of calculation stored to the device (d), all zeros on the left side of the significant figures are zero-suppressed.



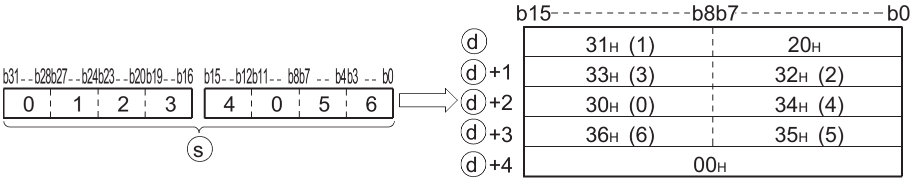
- The data to be stored to the device specified for (d)+2 differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).
  - When SM701 is OFFStores 0
  - When SM701 is ONDoes not change

■DBCDDA(P)

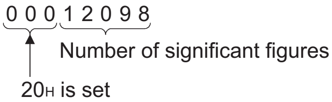
- Converts each digit number of hexadecimal notation of the 8-digit BCD data specified for (s) to ASCII code data, and stores the results to the device specified for (d) and the following devices.


s
b31 - - b28 b27 - - b24 b23 - - b20 b19 - - b16 b15 - - b12 b11 - - b8 b7 - - b4 b3 - - b0
Ten millions place
Millions place
Hundred thousands place
Ten thousands place
Thousands place
Hundreds place
Tens place
Units place
b15 - - b8b7 - - b0
d
d +1
d +2
d +3
ASCII code for millions place
ASCII code for ten-millions place
ASCII code for ten-thousands place
ASCII code for hundred-thousands place
ASCII code for hundreds place
ASCII code for thousands place
ASCII code for units place
ASCII code for tens place
0
(Only when SM701 is OFF)

For example, if the value 01234056 is specified for (s), the result are stored to the devices from (d) as shown below.



- The BCD data specified for (s) can be in the range of 0 to 99999999.
- For the results of calculation stored to the device (d), all zeros on the left side of the significant figures are zero-suppressed.



- The data to be stored to the device specified for (d)+4 differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).
  - When SM701 is OFFStores 0
  - When SM701 is ONDoes not change

Operation error

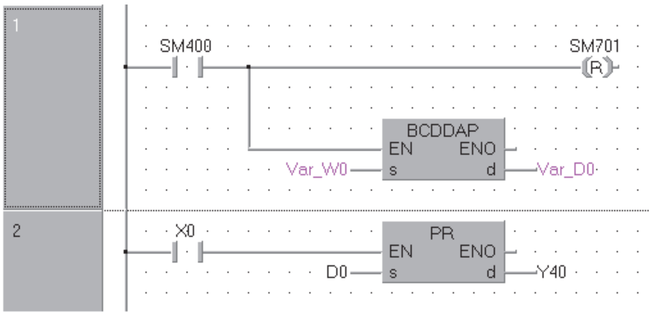
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The data of (s) during the operation of the BCDDA(P) instruction is outside the range of from 0 to 9999. The data of (s) during the operation of the DBCDDA(P) instruction is outside the range of 0 to 99999999.	—	○	○	○	○	○
4101	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

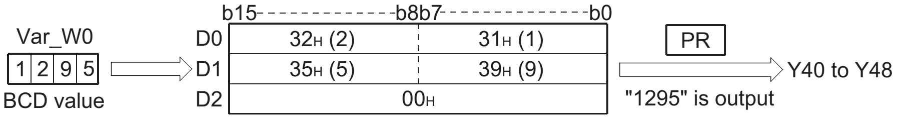
- In the following program, the 4-digit BCD data in Var_W0 are converted to the decimal ASCII code data, and the results are output to Y40 to Y48 by the PR instruction. (Global label Var_D0 is assigned to device D0.)

[Structured ladder/FBD]



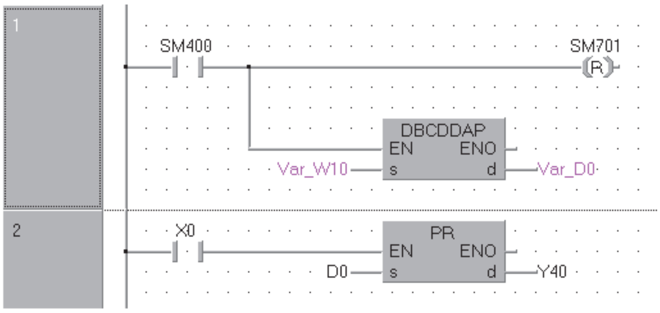
[ST]
RST(SM400,SM701);
BCDDAP(SM400,Var_W0,Var_D0);
PR(X0,D0,Y40);

[Operation]
The ASCII code data are output to Y40 to Y48 by the execution of the PR instruction when X0 turns ON.
Since SM701 is OFF, The PR instruction outputs ASCII code data up to 00H.



- In the following program, the 8-digit BCD data in Var_W10 are converted to the decimal ASCII code data, and the results are output to Y40 to Y48 by the PR instruction. (Global label Var_D0 is assigned to device D0.)

[Structured ladder/FBD]

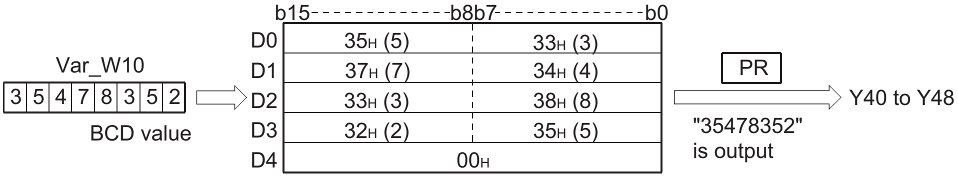


[ST]
RST(SM400,SM701);
DBCDDAP(SM400,Var_W10,Var_D0);
PR(X0,D0,Y40);

[Operation]

The ASCII code data are output to Y40 to Y48 by the execution of the PR instruction when X0 turns ON.

Since SM701 is OFF, The PR instruction outputs ASCII code data up to 00H.



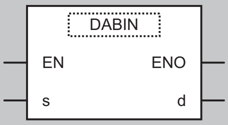
	b15 to b8	b7 to b0
D0	35H (5)	33H (3)
D1	37H (7)	34H (4)
D2	33H (3)	38H (8)
D3	32H (2)	35H (5)
D4	00H	

Decimal ASCII data to 16-/32-bit BIN data conversion

DABIN(P), DDABIN(P)

~~Basic~~ High performance Process Redundant Universal LCPU

Structured ladder/FBD



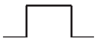
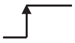
ST

ENO:= DABIN (EN, s, d);

Any of the following instruction can go in the dotted squares.

DABIN, DABINP, DDABIN, DDABINP

■Executing condition

Instruction	Executing condition
DABIN, DDABIN	
DABINP, DDABINP	

■Argument

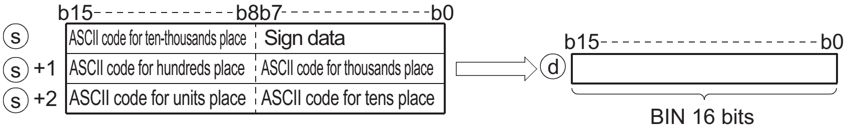
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	ASCII data to be converted to BIN value, or start number of the device that stores ASCII data	String (6)/(11)
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores conversion result	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	○	○		○				—	—

Processing details

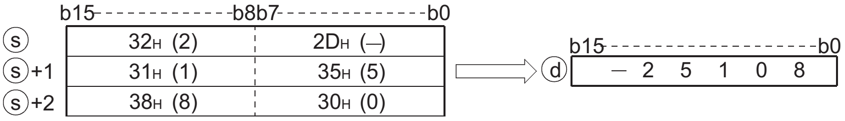
■DABIN(P)

- Converts decimal ASCII data stored in the device specified for (s) and the following devices to 16-bit BIN data, and stores the results to the device specified for (d).



	b15 - b8	b7 - b0
(s)	ASCII code for ten-thousands place	Sign data
(s) +1	ASCII code for hundreds place	ASCII code for thousands place
(s) +2	ASCII code for units place	ASCII code for tens place

For example, if the ASCII code "-25108" is specified for the devices following (s), the results will be stored to (d) as shown below.

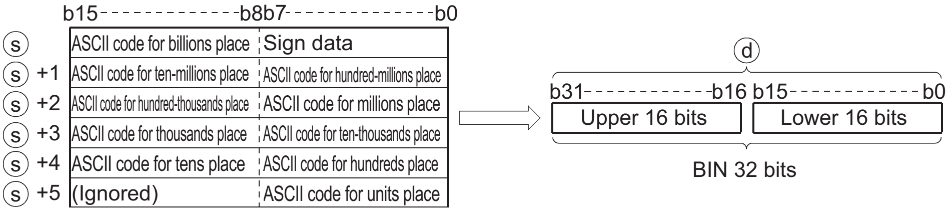


	b15 - b8	b7 - b0
(s)	32H (2)	2DH (–)
(s)+1	31H (1)	35H (5)
(s)+2	38H (8)	30H (0)

- The ASCII data specified for (s) to (s)+2 can be in the range of -32768 to 32767.
- The sign 20H will be stored if the BIN value is positive, and the sign 2DH will be stored if it is negative. (If other than 20H or 2DH is set, it will be processed as positive data.)
- ASCII code can be set for each position within the range from 30H to 39H.
- If the ASCII code set for each position is 20H or 00H, it will be processed as 30H.

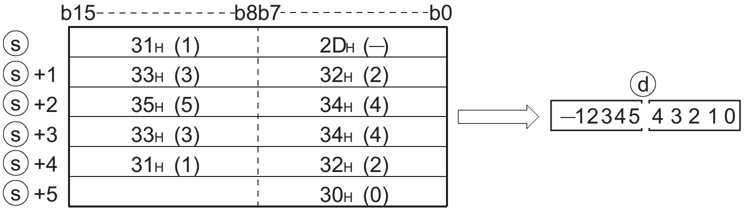
■DDABIN(P)

- Converts decimal ASCII data stored in the device specified for (s) and the following devices to 32-bit BIN data, and stores the results to the device specified for (d).



	b15 - b8	b7 - b0
(s)	ASCII code for billions place	Sign data
(s) +1	ASCII code for ten-millions place	ASCII code for hundred-millions place
(s) +2	ASCII code for hundred-thousands place	ASCII code for millions place
(s) +3	ASCII code for thousands place	ASCII code for ten-thousands place
(s) +4	ASCII code for tens place	ASCII code for hundreds place
(s) +5	(Ignored)	ASCII code for units place

For example, if the ASCII code -1234543210 is specified for the devices following (s), the result will be stored to (d) as shown below.



	b15 - b8	b7 - b0
(s)	31H (1)	2DH (–)
(s) +1	33H (3)	32H (2)
(s) +2	35H (5)	34H (4)
(s) +3	33H (3)	34H (4)
(s) +4	31H (1)	32H (2)
(s) +5		30H (0)

- The ASCII data specified for (s) to (s)+5 can be in the range of -2147483648 to 2147483647. Further, data stored in the upper bytes of (s)+5 will be ignored.
- The sign 20H will be stored if the BIN value is positive, and the sign 2DH will be stored if it is negative. (If other than 20H or 2DH is set, it will be processed as positive data.)
- ASCII code can be set for each position within the range from 30H to 39H.
- If the ASCII code set for each position is 20H or 00H, it will be processed as 30H.

Operation error

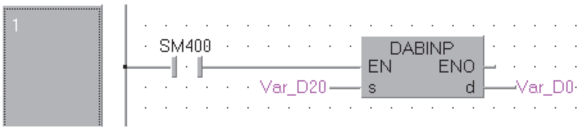
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The ASCII code specified for (s) to (s)+5 for each number is other than 30H to 39H, 20H, or 00H. The ASCII data specified for (s) to (s)+5 is outside the ranges shown below. • When DABIN(P) instruction is used: -32768 to 32767 • When DDABIN(P) instruction is used: -2147483648 to 2147483647	—	○	○	○	○	○
4101	The device specified for (s) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

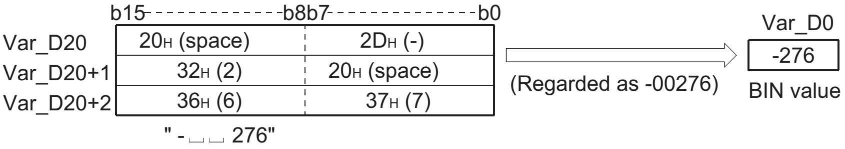
- In the following program, the 5-digit decimal ASCII data and the signs set in Var_D20 are converted to BIN value, and the result is stored to Var_D0.

[Structured ladder/FBD]



[ST]
DABINP(SM400,Var_D20,Var_D0);

[Operation]



	b15 - - - b8	b7 - - - b0
Var_D20	20H (space)	2DH (-)
Var_D20+1	32H (2)	20H (space)
Var_D20+2	36H (6)	37H (7)

"-␣␣276"

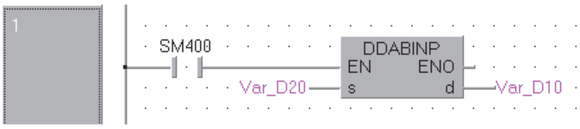
(Regarded as -00276)

Var_D0: -276

BIN value

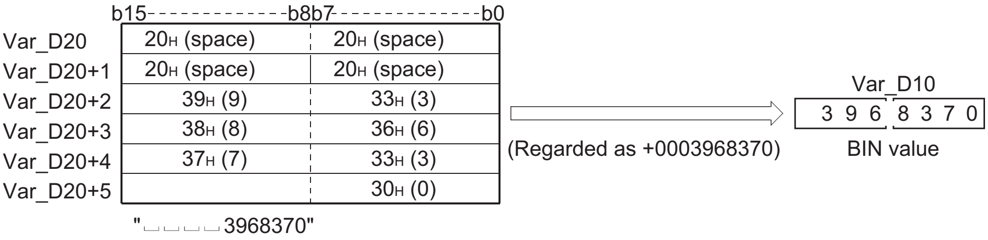
- In the following program, the 10-digit decimal ASCII data and the signs set in Var_D20 are converted to BIN value, and the result is stored to Var_D10.

[Structured ladder/FBD]



[ST]
DDABINP(SM400,Var_D20,Var_D10);

[Operation]



	b15 - - - b8	b7 - - - b0
Var_D20	20H (space)	20H (space)
Var_D20+1	20H (space)	20H (space)
Var_D20+2	39H (9)	33H (3)
Var_D20+3	38H (8)	36H (6)
Var_D20+4	37H (7)	33H (3)
Var_D20+5		30H (0)

"␣␣␣␣3968370"

(Regarded as +0003968370)

Var_D10: 3 9 6 8 3 7 0

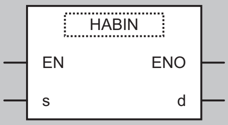
BIN value

Hexadecimal ASCII data to 16-/32-bit BIN data conversion

HABIN(P), DHABIN(P)

~~Basic~~ High performance Process Redundant Universal LCPU

Structured ladder/FBD



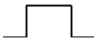
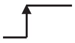
ST

```
ENO:= HABIN (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

HABIN, HABINP, DHABIN, DHABINP

■Executing condition

Instruction	Executing condition
HABIN, DHABIN	
HABINP, DHABINP	

■Argument

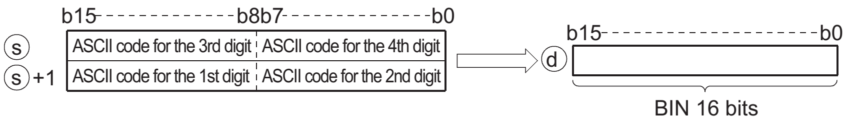
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	ASCII data to be converted to BIN value, or start number of the device that stores ASCII data	String (4)/(8)
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores conversion result	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	○	○		○				—	—

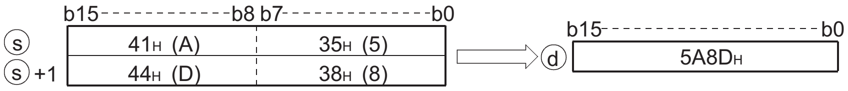
Processing details

■HABIN(P)

- Converts hexadecimal ASCII data stored in the device specified for (s) and the following devices to 16-bit BIN data, and stores the results to the device specified for (d).


b15
b8
b7
b0
s
s +1
ASCII code for the 3rd digit
ASCII code for the 4th digit
ASCII code for the 1st digit
ASCII code for the 2nd digit
d
b15
b0
BIN 16 bits

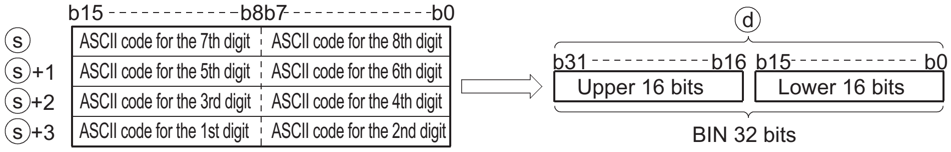
For example, if the ASCII code 5A8DH is specified for the devices following (s), the result will be stored to (d) as shown below.



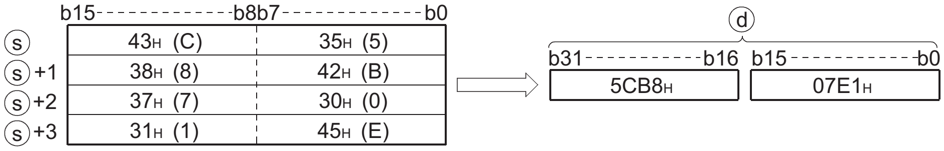
- The ASCII data specified for (s) to (s)+1 can be in the range of 0000H to FFFFH.
- The ASCII code can be in the range of 30H to 39H and 41H to 46H.

■DHABIN(P)

- Converts hexadecimal ASCII data stored in the device specified for (s) and the following devices to 32-bit BIN data, and stores the results to the device specified for (d).


b15
b8
b7
b0
s
s +1
s +2
s +3
ASCII code for the 7th digit
ASCII code for the 8th digit
ASCII code for the 5th digit
ASCII code for the 6th digit
ASCII code for the 3rd digit
ASCII code for the 4th digit
ASCII code for the 1st digit
ASCII code for the 2nd digit
d
b31
b16
b15
b0
Upper 16 bits
Lower 16 bits
BIN 32 bits

For example, if the ASCII code 5CB807E1H is specified for the devices following (s), the result will be stored to (d)+1 and (d) as shown below.



- The ASCII data specified for (s) to (s)+3 can be in the range of 00000000H to FFFFFFFFH.
- The ASCII code can be in the range of 30H to 39H and 41H to 46H.

Operation error

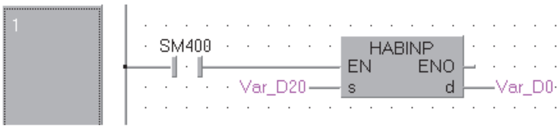
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The ASCII code for each digit specified for (s) to (s)+3 are outside the range of 30H to 39H and 41H to 46H.	—	○	○	○	○	○
4101	The device specified for (s) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the 4-digit hexadecimal ASCII data set in Var_D20 are converted to BIN value, and the results are stored to Var_D0.

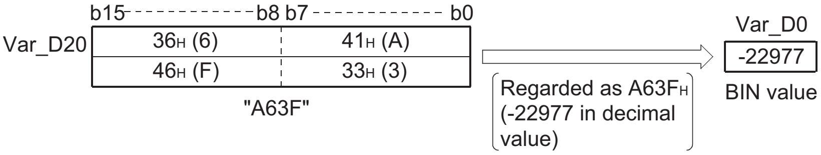
[Structured ladder/FBD]



[ST]

HABINP(SM400,Var_D20,Var_D0);

[Operation]



	b15 - b8	b7 - b0
Var_D20	36H (6)	41H (A)
	46H (F)	33H (3)

"A63F"

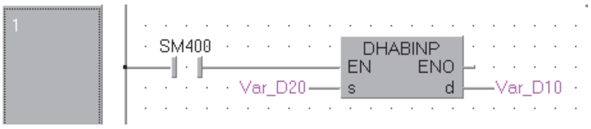
Regarded as A63FH (-22977 in decimal value)

Var_D0
-22977

BIN value

- In the following program, the 8-digit hexadecimal ASCII data set in Var_D20 are converted to BIN value, and the results are stored to Var_D10.

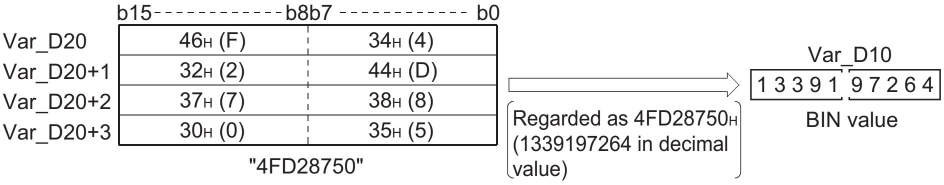
[Structured ladder/FBD]



[ST]

DHABINP(SM400,Var_D20,Var_D10);

[Operation]



	b15 - b8	b7 - b0
Var_D20	46H (F)	34H (4)
Var_D20+1	32H (2)	44H (D)
Var_D20+2	37H (7)	38H (8)
Var_D20+3	30H (0)	35H (5)

"4FD28750"

Regarded as 4FD28750H (1339197264 in decimal value)

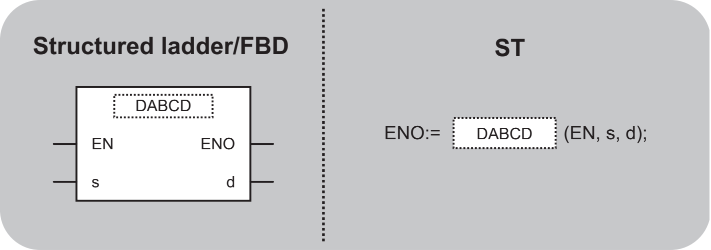
Var_D10
1 3 3 9 1 9 7 2 6 4

BIN value

Decimal ASCII data to 4-/8-digit BCD data conversion

DABCD(P), DDABCD(P)

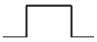
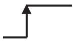
~~Basic~~ High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

DABCD, DABCDP, DDABCD, DDABCDP

■Executing condition

Instruction	Executing condition
DABCD, DDABCD	
DABCDP, DDABCDP	

■Argument

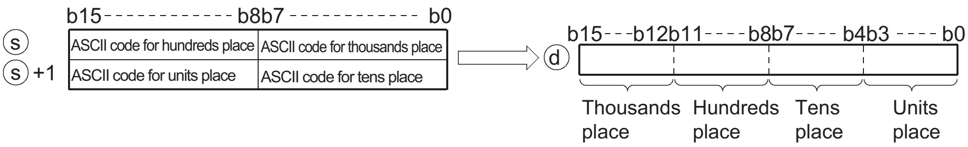
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	ASCII data to be converted to BCD value, or start number of the device that stores ASCII data	String (4)/(8)
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores conversion result	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	○	○		○				—	—

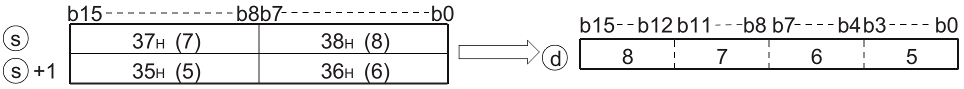
Processing details

■DABCD(P)

- Converts decimal ASCII data stored in the device specified for (s) and the following devices to 4-digit BCD data, and stores the results to the device specified for (d).


b15 b8b7 b0
(s)
(s)+1
ASCII code for hundreds place
ASCII code for thousands place
ASCII code for units place
ASCII code for tens place
(d)
b15 b12b11 b8b7 b4b3 b0
Thousands place
Hundreds place
Tens place
Units place

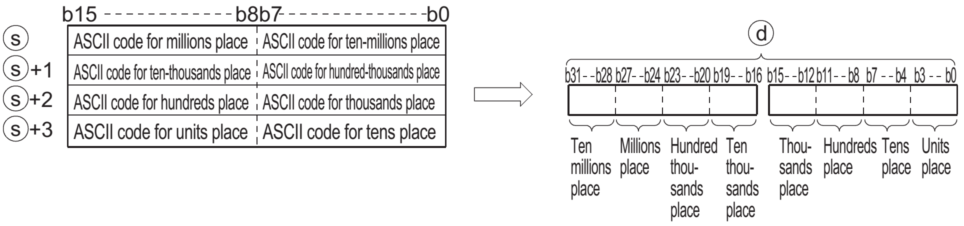
For example, if the ASCII code 8765 is specified for the devices following (s), the results will be stored to (d) as shown below.



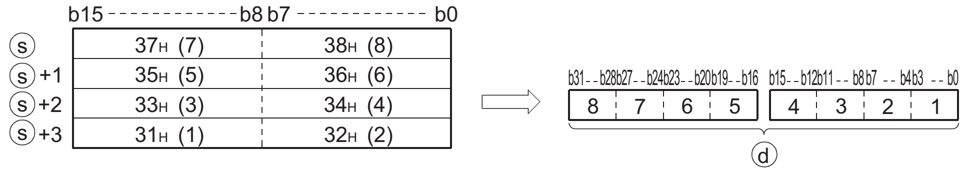
- The ASCII data specified for (s) to (s)+1 can be in the range of 0 to 9999.
- The ASCII code set at each digit can be in the range of 30H to 39H.
- If ASCII code for each digit is 20H or 00H, it is processed as 30H.

■DDABCD(P)

- Converts decimal ASCII data stored in the device specified for (s) and the following devices to 8-digit BCD data, and stores the results to the device specified for (d) and the following devices.


b15 b8b7 b0
(s)
(s)+1
(s)+2
(s)+3
ASCII code for millions place
ASCII code for ten-millions place
ASCII code for ten-thousands place
ASCII code for hundred-thousands place
ASCII code for hundreds place
ASCII code for thousands place
ASCII code for units place
ASCII code for tens place
(d)
b31 b28 b27 b24 b23 b20 b19 b16 b15 b12 b11 b8 b7 b4 b3 b0
Ten millions place
Millions place
Hundred thousands place
Ten thousands place
Thousands place
Hundreds place
Tens place
Units place

For example, if the ASCII code 87654321 is specified for the devices following (s), the results will be stored to (d) as shown below.



- The ASCII data specified for (s) to (s)+3 can be in the range of 0 to 99999999.
- The ASCII code set at each digit can be in the range of 30H to 39H.
- If ASCII code for each digit is from 20H to 00H, it is processed as 30H.

Operation error

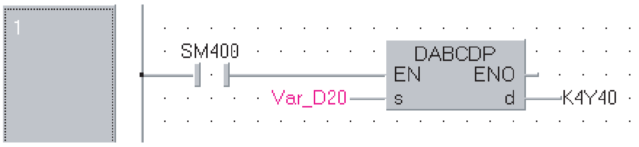
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	There are characters within the data of (s) that are outside the range of 0 to 9.	—	○	○	○	○	○
4101	The device specified for (s) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the decimal ASCII data set in Var_D20 are converted to the 4-digit BCD data, and the result is output to Y40 to Y4F.

[Structured ladder/FBD]

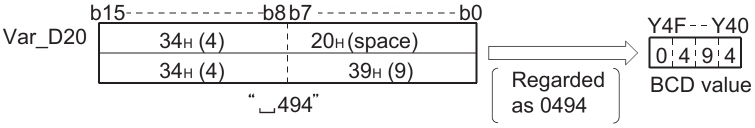


Output of the converted BCD value to the display.

[ST]

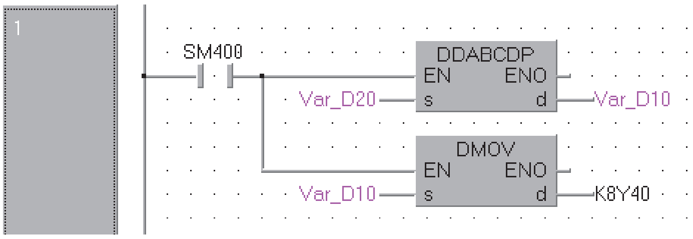
DABCDP(SM400,Var_D20,K4Y40);

[Operation]



- In the following program, the hexadecimal ASCII data set in Var_D20 are converted to 8-digit BCD data, and the result is stored to Var_D10, and also they are output to Y40 to Y5F.

[Structured ladder/FBD]



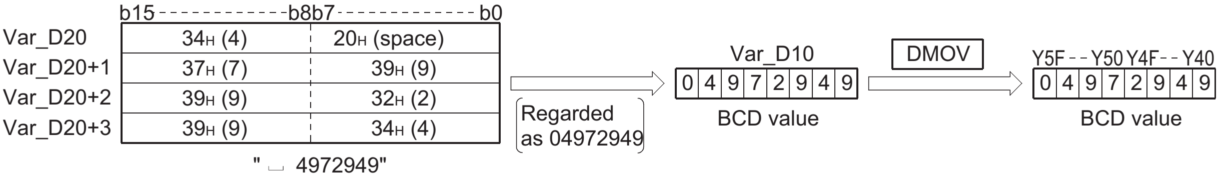
Output of the converted BCD value to the display.

[ST]

DDABCDP(SM400,Var_D20,Var_D10);

DMOV(SM400,Var_D10,K8Y40);

[Operation]

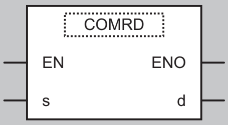


Reading device comment data

COMRD(P)

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



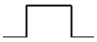
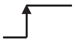
ST

ENO:= COMRD (EN, s, d);

Any of the following instruction can go in the dotted squares.

COMRD, COMRDP

■Executing condition

Instruction	Executing condition
COMRD	
COMRDP	

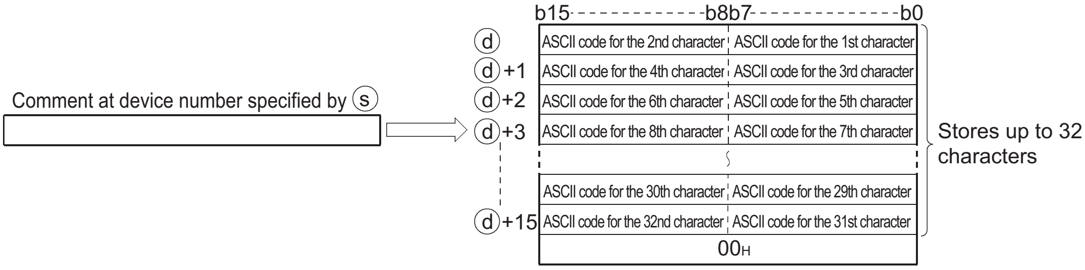
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores comment to be read	ANY_SIMPLE
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores read comments	String (32)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others (BL\S, BL\TR, BL, P, I, J, U)
	Bit	Word		Bit	Word				
(s)	○	○		○			—		○
(d)	—	○		—			—		—

Processing details

- Reads the comment data in the device specified for (s), and stores it as ASCII code data to the device specified for (d) and the following devices.

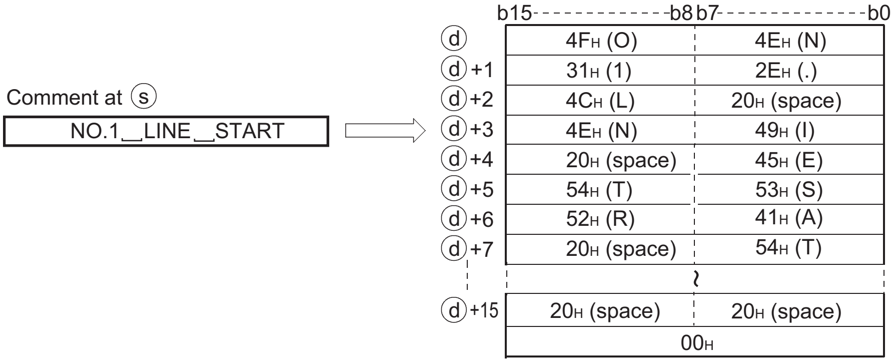


Comment at device number specified by ⓢ

	b15 - - - b8	b7 - - - b0
ⓓ	ASCII code for the 2nd character	ASCII code for the 1st character
ⓓ+1	ASCII code for the 4th character	ASCII code for the 3rd character
ⓓ+2	ASCII code for the 6th character	ASCII code for the 5th character
ⓓ+3	ASCII code for the 8th character	ASCII code for the 7th character
	⁓	
	ASCII code for the 30th character	ASCII code for the 29th character
ⓓ+15	ASCII code for the 32nd character	ASCII code for the 31st character
	00H	

Stores up to 32 characters

For example, if the comment in the device specified for (s) is "NO.1␣LINE␣START", its data will be stored to the devices following (d) as shown below.



Comment at ⓢ

NO.1␣LINE␣START

	b15 - - - b8	b7 - - - b0
ⓓ	4FH (O)	4EH (N)
ⓓ+1	31H (1)	2EH (.)
ⓓ+2	4CH (L)	20H (space)
ⓓ+3	4EH (N)	49H (I)
ⓓ+4	20H (space)	45H (E)
ⓓ+5	54H (T)	53H (S)
ⓓ+6	52H (R)	41H (A)
ⓓ+7	20H (space)	54H (T)
	⁓	
ⓓ+15	20H (space)	20H (space)
	00H	

- If the comment range is set for the device number specified for (s), and no comment is registered, all of the characters for the comment would be processed as 20H (space).
- The operation in the area following the device number where the final character of (d) is stored differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).
  - When SM701 is OFF: Does not change
  - When SM701 is ON: Stores 0
- When a comment is read, SM720 turns ON for one scan after the instruction is completed. SM721 turns ON during the execution of the instruction. While SM721 is ON, the COMRD(P) instruction cannot be executed. If the attempt is made, no processing is performed.

Point

- For device comments used with the COMRD(P) instruction, use comment files stored in the standard RAM, memory card or SD memory card. Comment files stored in the program memory cannot be used.
- Set the comment file used for the COMRD(P) instruction can be set in <<PLC file>> tab in the PLC parameter. If the used comment file has not been set in the PLC file setting, device comments cannot be output with the COMRD(P) instruction.
- Do not execute the COMRD(P) instruction in an interrupt program. Doing so will cause a malfunction.

Operation error

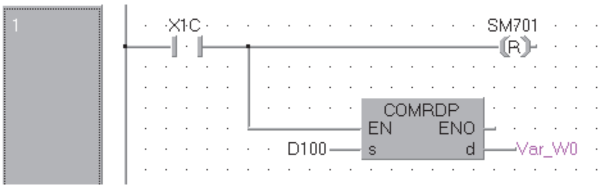
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The comment is not registered to the device number specified for (s).	—	○	○	○	○	○
4101	The device number specified for (d) is not a word device.	—	○	○	○	○	○
	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the comment set in D100 is stored as ASCII code data in Var_W0 and the following devices when X1C turns ON.

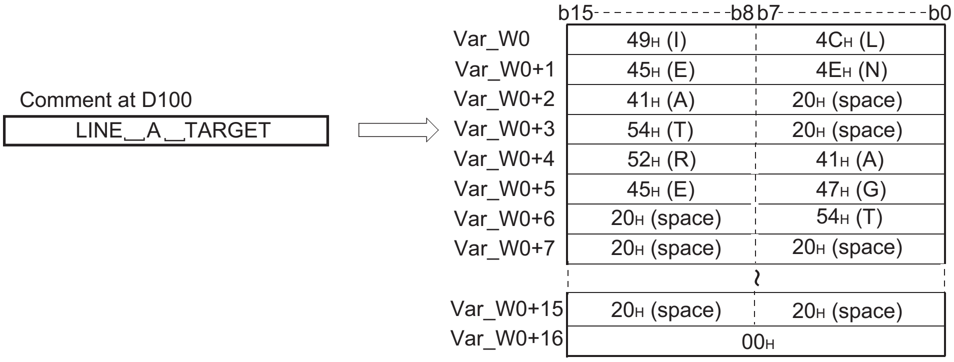
[Structured ladder/FBD]



[ST]

```
RST(X1C,SM701);
COMRDP(X1C,D100,Var_W0);
```

[Operation]



	b15–b8	b7–b0
Var_W0	49H (I)	4CH (L)
Var_W0+1	45H (E)	4EH (N)
Var_W0+2	41H (A)	20H (space)
Var_W0+3	54H (T)	20H (space)
Var_W0+4	52H (R)	41H (A)
Var_W0+5	45H (E)	47H (G)
Var_W0+6	20H (space)	54H (T)
Var_W0+7	20H (space)	20H (space)
	≀	
Var_W0+15	20H (space)	20H (space)
Var_W0+16	00H	

Precautions

- The processing completes after several scans.
- The COMRD(P) or PRC instruction is not executed if the start signal (execution command) of the COMRD(P) or PRC instruction is turned ON before completion of the instruction (while SM721 is ON). Execute the COMRD(P) or PRC instruction when SM721 is OFF.
- Two or more file comments cannot be accessed simultaneously.
- The following instructions cannot be executed simultaneously because they use SM721 in common.

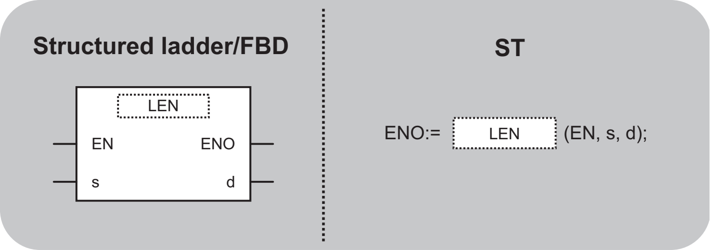
Instruction name	ON during instruction execution	ON for one scan after the completion	ON at completion with an error
SP_FREAD SP_FWRITE	SM721	Specified by instruction	(Device specified by instruction) + 1
PRC COMRD(P)		SM720	None

- For High-speed Universal model QCPU, Universal model Process CPU, and LCPU, when comment files stored in the SD memory card are used, this instruction will not be executed while SM606 (SD memory card forced disable instruction) is ON. If executed, no processing is performed.

Character string length detection

LEN(P)

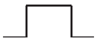
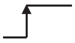
~~Basic~~ High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

LEN, LENP

■Executing condition

Instruction	Executing condition
LEN	
LENP	

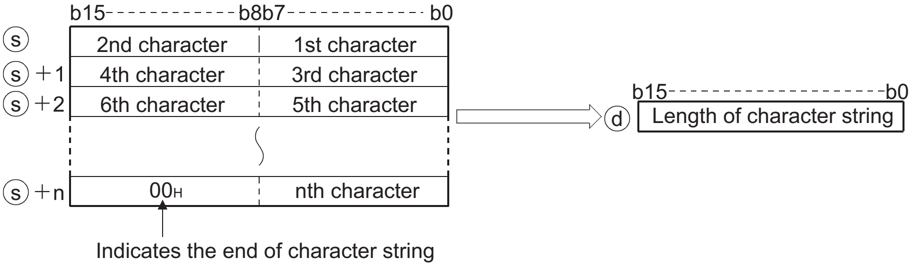
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Character string, or start number of the device that stores character string	String
Output argument	ENO	Execution result	Bit
	d	Device number that stores the detected character string length	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	○	○		○				—	—

Processing details

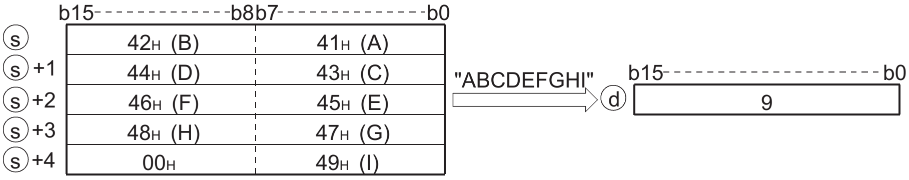
- Detects length of character string specified for (s) and stores to the device specified for (d) and the following devices. Processes the data from the device specified for (s) to the device storing 00H as a character string.



	b15 - - - b8	b7 - - - b0
(s)	2nd character	1st character
(s)+1	4th character	3rd character
(s)+2	6th character	5th character
(s)+n	00H	nth character

Indicates the end of character string
b15 - - - b0
(d)
Length of character string

For example, when the value "ABCDEFGHI" is stored to the devices following (s), the value 9 is stored to (d).



	b15 - - - b8	b7 - - - b0
(s)	42H (B)	41H (A)
(s)+1	44H (D)	43H (C)
(s)+2	46H (F)	45H (E)
(s)+3	48H (H)	47H (G)
(s)+4	00H	49H (I)

Operation error

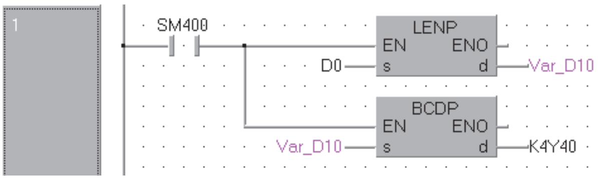
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	There is no 00H set within the corresponding device range following the device specified for (s).	—	○	○	○	○	○

Program example

- In the following program, the length of the character string from D0 is output to Y40 to Y4F as 4-digit BCD values.

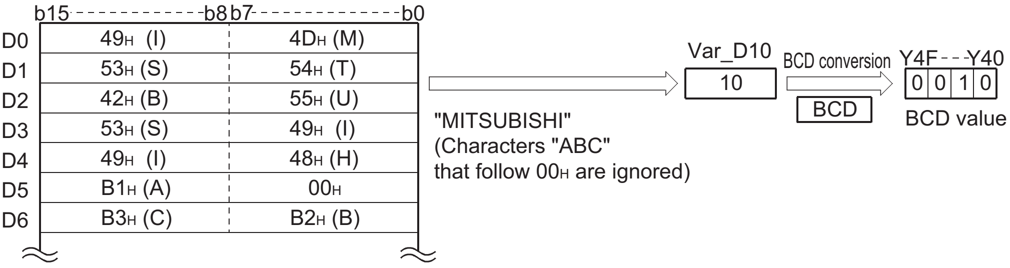
[Structured ladder/FBD]



Outputs the length of character string to a display device.

[ST]
LENP(SM400,D0,Var_D10);
BCDP(SM400,Var_D10,K4Y40);

[Operation]



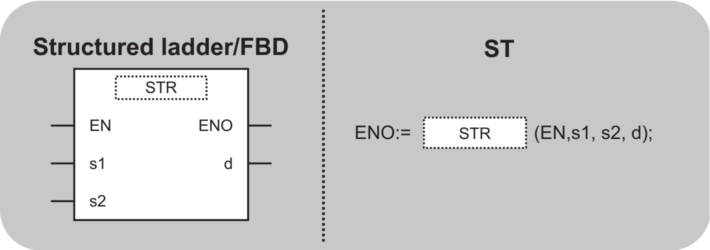
	b15 - - - b8	b7 - - - b0
D0	49H (I)	4DH (M)
D1	53H (S)	54H (T)
D2	42H (B)	55H (U)
D3	53H (S)	49H (I)
D4	49H (I)	48H (H)
D5	B1H (A)	00H
D6	B3H (C)	B2H (B)

16-/32-bit BIN data to character string data conversion

STR(P), DSTR(P)

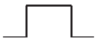
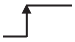
Ver. Basic High performance Process Redundant Universal LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later



Any of the following instruction can go in the dotted squares.
STR, STRP, DSTR, DSTRP

■Executing condition

Instruction	Executing condition
STR, DSTR	
STRP, DSTRP	

■Argument

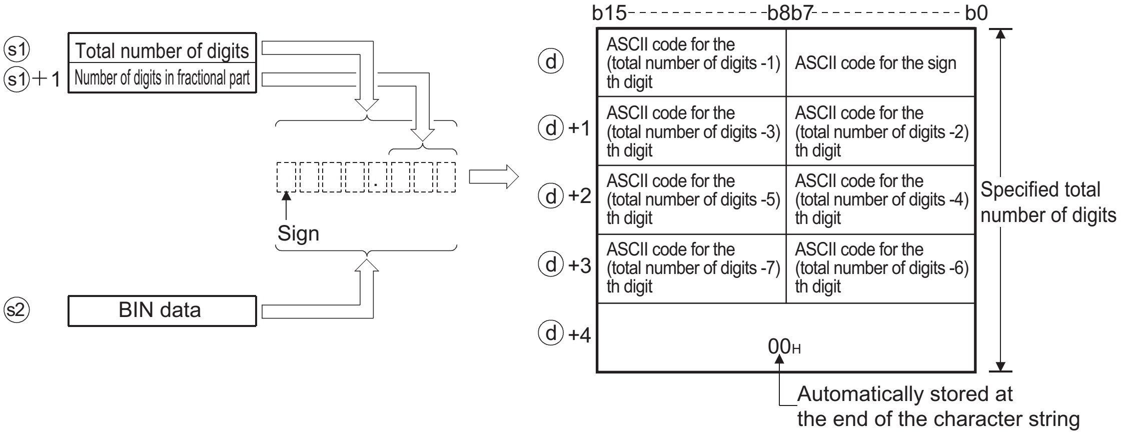
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores number of digits for placing a decimal to the value to be converted	ANY32
	s2	BIN data to be converted	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted character string data	String (9)/(14)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○	○		○				—	—
(s2)	○	○		○				○	—
(d)	—	○		—				—	—

Processing details

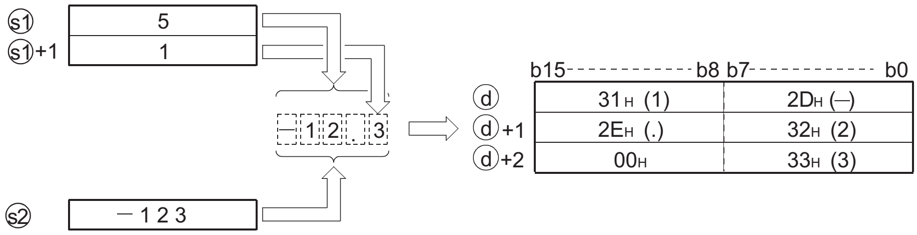
■STR(P)

- Adds a decimal point at the position specified for (s1), to the 16-bit BIN data specified for (s2), converts the data to character string data, and stores the results to the device specified for (d) and the following devices.



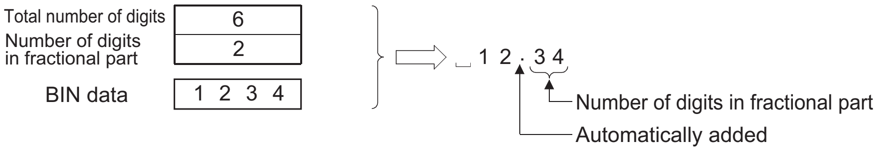
s1 Total number of digits
s1+1 Number of digits in fractional part
Sign
s2 BIN data
b15 b8 b7 b0
Specified total number of digits
Automatically stored at the end of the character string

	b15 - b8	b7 - b0
d	ASCII code for the (total number of digits -1) th digit	ASCII code for the sign
d+1	ASCII code for the (total number of digits -3) th digit	ASCII code for the (total number of digits -2) th digit
d+2	ASCII code for the (total number of digits -5) th digit	ASCII code for the (total number of digits -4) th digit
d+3	ASCII code for the (total number of digits -7) th digit	ASCII code for the (total number of digits -6) th digit
d+4	00H	



	b15 - b8	b7 - b0
d	31H (1)	2DH (−)
d+1	2EH (.)	32H (2)
d+2	00H	33H (3)

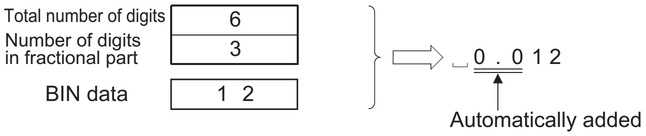
- The total number of digits that can be specified for (s1) is from 2 to 8 digits.
- The number of digits in the fractional part that can be specified for (s1)+1 is from 0 to 5 digits. However, the number of digits in the fractional part must be smaller than or equal to the total number of digits minus 3.
- BIN values between -32768 and 32767 can be specified for (s2).
- After conversion, character string data are stored to the devices following (d) as indicated below.
  - The sign 20H (space) will be stored if the BIN value is positive, and the sign 2DH (minus sign) will be stored if it is negative.
  - If the setting for the number of digits in the fractional part is anything other than 0, 2EH (.) will automatically be stored at the position before the first of the specified number of digits.



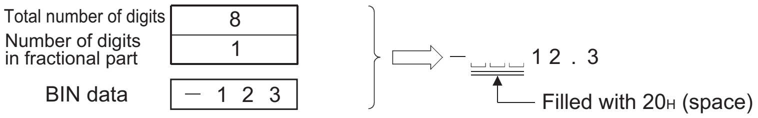
Total number of digits 6
Number of digits in fractional part 2
BIN data 1 2 3 4
␣ 1 2 . 3 4
Number of digits in fractional part
Automatically added

If the number of digits in the fractional part is 0, the ASCII code 2EH (.) will not be stored.

- If the total number of digits in the fractional part is greater than the number of BIN data digits, a zero will be added automatically and the number converted by shifting to the right, so that it would become 0.□□□□□.



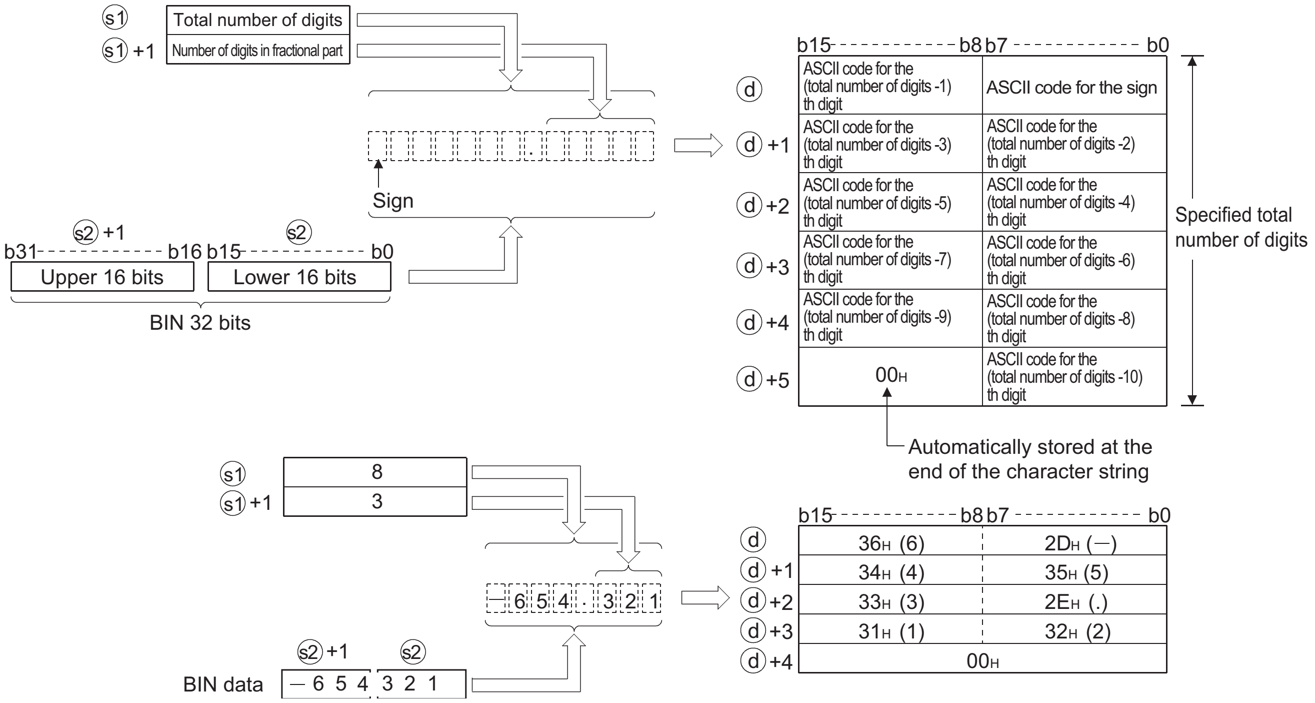
- If the total number of digits excluding the sign and the decimal point is greater than the number of BIN data digits, 20H (space) will be stored between the sign and the numeric value. If the number of BIN digits is greater, an error will be returned.



- The value 00H is automatically stored at the end of the converted character string.

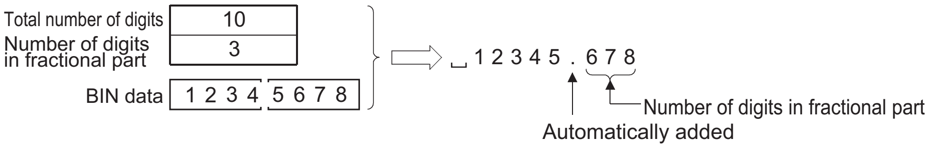
■DSTR(P)

- Adds a decimal point at the position specified for (s1)to the 32-bit BIN data specified for (s2), converts the data to character string data, and stores the results to the device specified for (d) and the following devices.

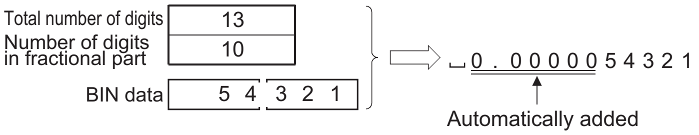


s1
Total number of digits
s1 +1
Number of digits in fractional part
Sign
s2 +1
s2
b31
b16
b15
b0
Upper 16 bits
Lower 16 bits
BIN 32 bits
b15
b8
b7
b0
d
ASCII code for the (total number of digits -1) th digit
ASCII code for the sign
d +1
ASCII code for the (total number of digits -3) th digit
ASCII code for the (total number of digits -2) th digit
d +2
ASCII code for the (total number of digits -5) th digit
ASCII code for the (total number of digits -4) th digit
d +3
ASCII code for the (total number of digits -7) th digit
ASCII code for the (total number of digits -6) th digit
d +4
ASCII code for the (total number of digits -9) th digit
ASCII code for the (total number of digits -8) th digit
d +5
00H
ASCII code for the (total number of digits -10) th digit
Specified total number of digits
Automatically stored at the end of the character string
s1
8
s1 +1
3
– 6 5 4 . 3 2 1
s2 +1
s2
BIN data
– 6 5 4 3 2 1
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

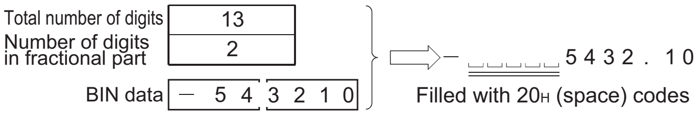
- The total number of digits that can be specified for (s1) is from 2 to 13 digits.
- The number of digits in the fractional part that can be specified for (s1)+1 is from 0 to 10 digits. However, the number of digits following the decimal point must be smaller than or equal to the total number of digits minus 3.
- BIN values between -2147483648 and 2147483647 can be specified for (s2).
- After conversion, character string data are stored to the device numbers following (d) as indicated below.
  - The sign 20H (space) will be stored if the BIN value is positive, and the sign 2DH (minus sign) will be stored if it is negative.
  - If the setting for the number of digits in the fractional part is anything other than 0, 2EH (.) will automatically be stored at the position before the first of the specified number of digits. If the number of digits in the fractional part is 0, the ASCII code 2EH (.) will not be stored.



  - If the total number of digits in the fractional part is greater than the number of BIN data digits, a zero will be added automatically and the number converted by shifting to the right, so that it would become 0□□□□.



  - If the total number of digits excluding the sign and the decimal point is greater than the number of BIN data digits, 20H (space) will be stored between the sign and the numeric value.



If the number of BIN digits is greater, an error will be returned.

  - The value 00H is automatically stored at the end of the converted character string.

Operation error

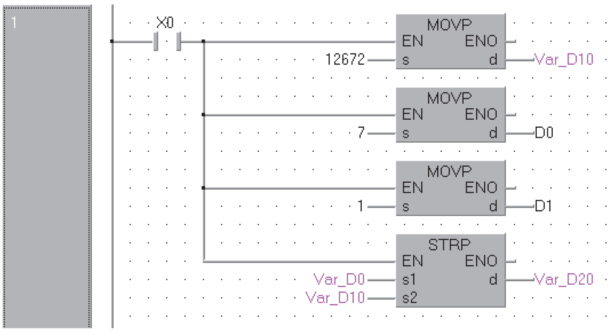
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The total number of digits specified for (s1) is outside of the ranges shown below. • When the STR(P) instruction is in use: 2 to 8 digits • When the DSTR(P) instruction is in use: 2 to 13 digits The number of digits in the fractional part specified for (s1)+1 is outside of the range shown below. • When the STR(P) instruction is in use: 0 to 5 digits • When the DSTR(P) instruction is in use: 0 to 10 digits The relationship between the total number of digits specified for (s1) and the number of digits in the fractional part specified for (s1)+1 is not as shown below. • Total number of digits minus 3 is equal to or larger than the number of digits in the fractional part. The number of digits specified for (s1) is smaller than 2 + number of digits of the BIN data specified for (s2). (Number of digits in (s1) < Number of digits of the BIN data in (s2) without a sign + Number of digits for a sign (+ or -) + Number of digits for decimal point (.))	○	○	○	○	○	○
4101	The device range where the character string specified for (d) will be stored exceeds the corresponding device range.	○	○	○	○	○	○

Program example

- In the following program, the 16-bit BIN data stored in Var_D10 are converted to character string data in accordance with the digit specification of D0 and D1 when X0 turns ON, and stores the result to Var_D20. (Var_D0 is declared as a global variable assigned with D0 and D1.)

[Structured ladder/FBD]



Sets the data.

Sets the total number of digits.

Sets the number of digits in decimal fraction.

[ST]

```
MOVP(X0,12672,Var_D10);
MOVP(X0,7,D0);
MOVP(X0,1,D1);
STRP(X0,Var_D0,Var_D10,Var_D20);
```

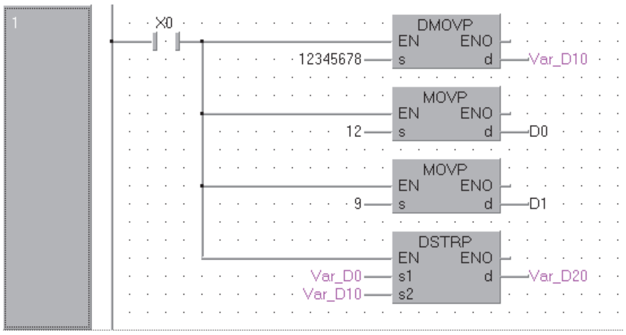
[Operation]

				b15------------b8	b7-------------b0
Var_D10	12672	⇨	Var_D20	31H (1)	20H (space)
			Var_D20+1	36H (6)	32H (2)
D0	7		Var_D20+2	2EH (.)	37H (7)
D1	1		Var_D20+3	00H	32H (2)

"␣1267.2"

- In the following program, the 32-bit BIN data stored in Var_D10 are converted to character string data in accordance with the digit specification of D0 and D1 when X0 turns ON, and stores the result to Var_D20. (Var_D0 is declared as a global variable assigned with D0 and D1.)

[Structured ladder/FBD]



Sets the data.

Sets the total number of digits.

Sets the number of digits in decimal fraction.

[ST]

```
DMOVP(X0,12345678,Var_D10);
MOVP(X0,12,D0);
MOVP(X0,9,D1);
DSTRP(X0,Var_D0,Var_D10,Var_D20);
```

[Operation]

	D11	D10
Var_D10	12345678	

D0	12
D1	9

	b15 - - - - b8	b7 - - - - b0
Var_D20	30H (0)	20H (space)
Var_D20+1	30H (0)	2EH (.)
Var_D20+2	32H (2)	31H (1)
Var_D20+3	34H (4)	33H (3)
Var_D20+4	36H (6)	35H (5)
Var_D20+5	38H (8)	37H (7)
Var_D20+6	00H	

" ␣ 0.012345678"

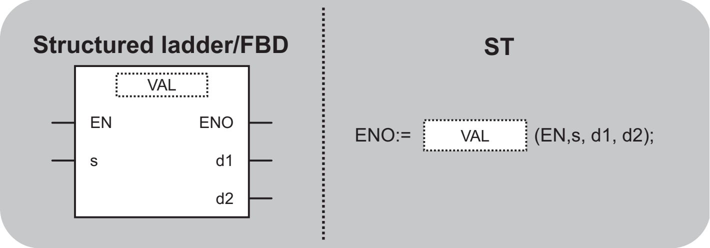
Character string data to 16-/32-bit BIN data conversion

VAL(P), DVAL(P)

Ver.

Basic | High performance | Process | Redundant | Universal | LCPU

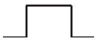
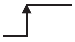
- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later



Any of the following instruction can go in the dotted squares.

VAL, VALP, DVAL, DVALP

■Executing condition

Instruction	Executing condition
VAL, DVAL	
VALP DVALP	

■Argument

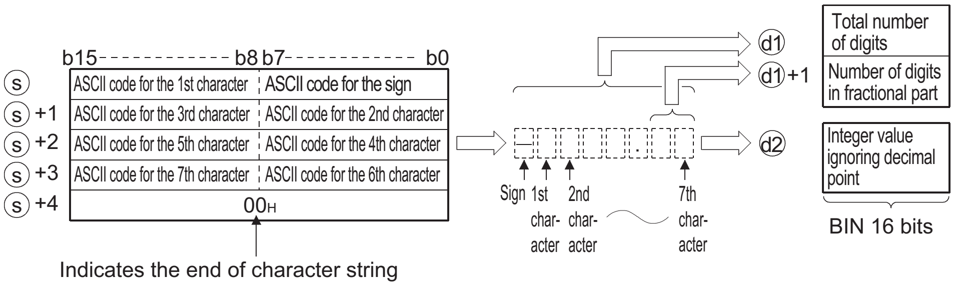
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Character string data to be converted to BIN data, or start number of the device that stores character string data	String (8)/(13)
Output argument	ENO	Execution result	Bit
	d1	Start number of the device that stores number of digits of the converted BIN data	ANY32
	d2	Start number of the device that stores converted BIN data	ANY16/32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d1)	○	○		—				—	—
(d2)	○	○		○				—	—

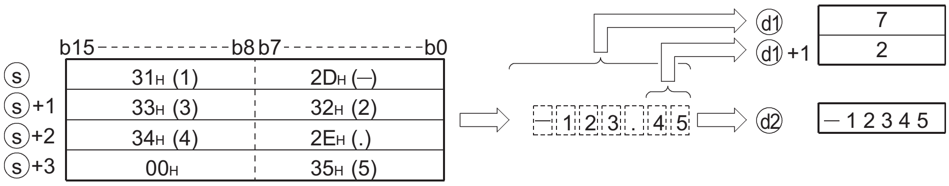
Processing details

■VAL(P)

- Converts character string data stored in the device specified for (s) and the following devices to 16-bit BIN data, and stores the number of digits and BIN data to (d1) and (d2). For converting character string data to BIN data, all data from the device specified for (s) to the device where 00H is stored will be processed as character string data.


b15 b8 b7 b0
s | ASCII code for the 1st character | ASCII code for the sign
s +1 | ASCII code for the 3rd character | ASCII code for the 2nd character
s +2 | ASCII code for the 5th character | ASCII code for the 4th character
s +3 | ASCII code for the 7th character | ASCII code for the 6th character
s +4 | 00H
Indicates the end of character string
Sign 1st character 2nd character 7th character
d1
d1 +1
d2
Total number of digits
Number of digits in fractional part
Integer value ignoring decimal point
BIN 16 bits

For example, if the character string -123.45 is specified for the devices following (s), the result would be stored to (d1) and (d2) as shown below.



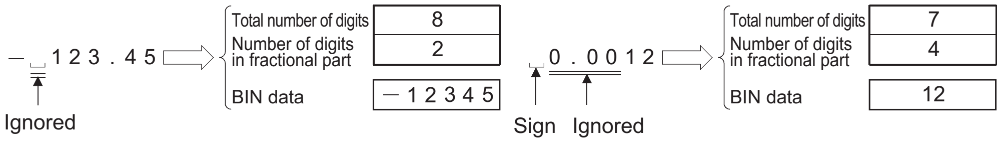
- The total number of characters that can be specified as a character string for (s) is from 2 to 8 characters.
- From 0 to 5 characters from the character string specified for (s) can become the fractional part. However, this number must not exceed the total number of digits minus 3.
- The range of the numeric character string that can be converted to BIN value is from -32768 to 32767, ignoring a decimal point. Numeric value character strings, excluding the sign and the decimal point, can be specified only within the range from 30H to 39H.

The value ignoring a decimal point means:

Ex.

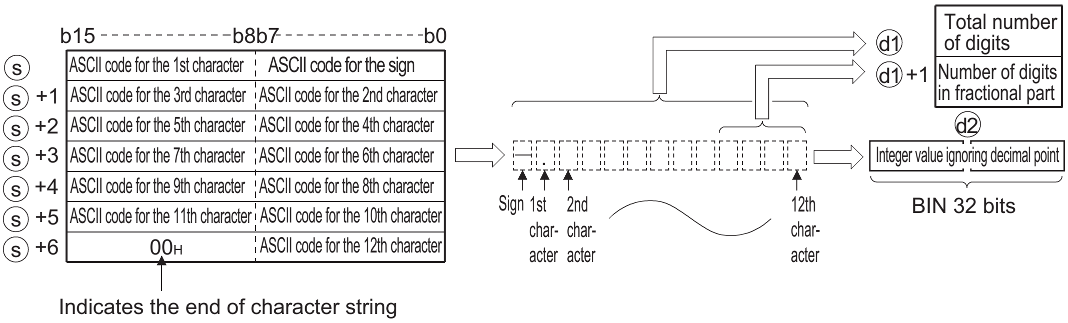
-12345.6 → -123456

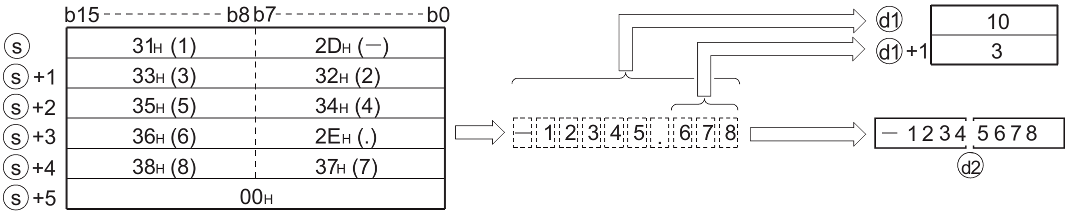
- The sign 20H will be stored if the numeric value is positive, and the sign 2DH will be stored if it is negative.
- 2EH is set for the decimal point.
- The total number of digits stored to (d1) is the total number of characters that represent numeric values including a sign and decimal point. The number of digits in the fractional part stored to (d1)+1 is the number of characters that represent fractional part separated by 2EH (.). The BIN data stored to (d2) is the character string ignoring the decimal point that has been converted to BIN value.
- In cases where the character string specified for (s) contains 20H (space) or 30H (0) between the sign and the first numeric value other than 0, these 20H and 30H are ignored in the conversion to BIN value.


− ␣ 1 2 3 . 4 5
Ignored
Total number of digits | 8
Number of digits in fractional part | 2
BIN data | − 1 2 3 4 5
0 . 0 0 1 2
Sign
Ignored
Total number of digits | 7
Number of digits in fractional part | 4
BIN data | 12

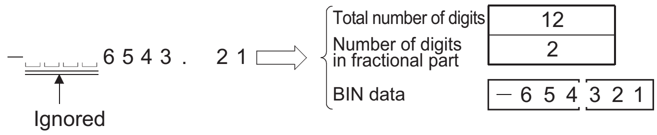
■DVAL(P)

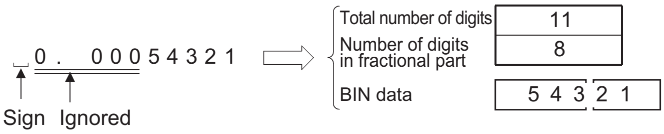
- Converts the character string data stored in the device specified for (s) and the following devices to 32-bit BIN data, and stores the digits numbers and BIN data to (d1) and (d2). For converting character string data to BIN data, all data from the device specified for (s) to the device where 00H is stored will be processed as character strings.


b15 b8 b7 b0
(s) ASCII code for the 1st character | ASCII code for the sign
(s)+1 ASCII code for the 3rd character | ASCII code for the 2nd character
(s)+2 ASCII code for the 5th character | ASCII code for the 4th character
(s)+3 ASCII code for the 7th character | ASCII code for the 6th character
(s)+4 ASCII code for the 9th character | ASCII code for the 8th character
(s)+5 ASCII code for the 11th character | ASCII code for the 10th character
(s)+6 00H | ASCII code for the 12th character
Indicates the end of character string
Sign
1st character
2nd character
12th character
(d1) Total number of digits
(d1)+1 Number of digits in fractional part
(d2) Integer value ignoring decimal point
BIN 32 bits



- The total number of characters in the character string specified for (s) is from 2 to 13 characters.
- From 0 to 10 characters in the character string specified for (s) can be the fractional part. However, this number must not exceed the total number of digits minus 3.
- The range of the numeric character string that can be converted to BIN value is from -2147483648 to 2147483647, excluding the decimal point. Numeric value character strings, excluding the sign and the decimal point, can be specified only within the range from 30H to 39H.
- The sign 20H will be stored if the numeric value is positive, and the sign 2DH will be stored if it is negative.
- 2EH is set for the decimal point.
- The total number of digits stored to (d1) is the total number of characters that represent numeric values including a sign and decimal point. The number of digits in the fractional part stored to (d1)+1 is the number of characters that represent fractional part separated by 2EH (.). The BIN data stored to (d2) is the character string ignoring the decimal point that has been converted to BIN value.
- In cases where the character string specified for (s) contains 20H (space) or 30H (0) between the sign and the first numeric value other than 0, these 20H and 30H are ignored in the conversion to BIN value.


− ␣␣␣␣ 6 5 4 3 . 2 1
Ignored
Total number of digits 12
Number of digits in fractional part 2
BIN data − 6 5 4 3 2 1


␣ 0 . 0 0 0 5 4 3 2 1
Sign
Ignored
Total number of digits 11
Number of digits in fractional part 8
BIN data 5 4 3 2 1

Operation error

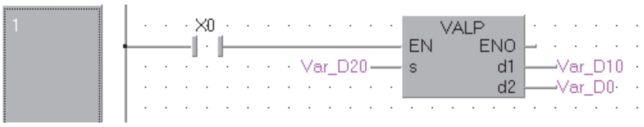
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The number of characters in the character string specified for (s) is outside of the ranges shown below. • When VAL(P) instruction is in use: 2 to 8 characters • When DVAL(P) instruction is in use: 2 to 13 characters The number of characters in the fractional part of the character string specified for (s) is outside the ranges shown below. • When VAL(P) instruction is in use: 0 to 5 characters • When DVAL(P) instruction is in use: 0 to 10 characters The total number of characters of the character string specified for (s) and the number of characters in the fractional part stand in a relationship that is outside of the range indicated below. Total number of characters minus 3 is equal to or greater than the number of characters in the fractional part. ASCII code other than 20H or 2DH has been set for the sign. ASCII code other than from 30H to 39H or 2EH (decimal point) has been set for digits of each number. There has been more than one decimal point set in the value. The value of the converted BIN value exceeds the following ranges. • When VAL(P) instruction is in use: -32768 to 32767 • When DVAL(P) instruction is in use: -2147483648 to 2147483647	○	○	○	○	○	○
4101	No 00H is set within the range from the device number specified for (s) to the last device number of the corresponding device.	○	○	○	○	○	○

Program example

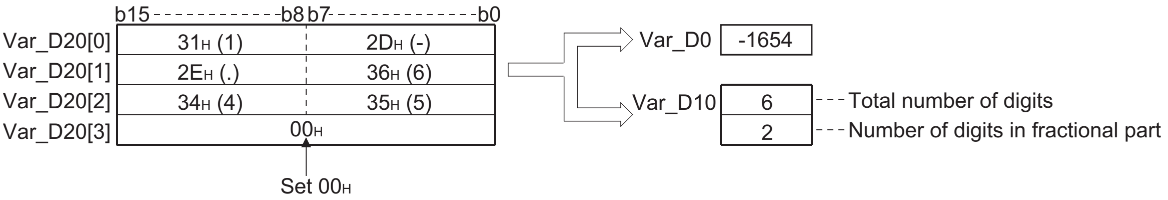
- In the following program, the character string data in Var_D20 are read as an integer value and converted to the BIN value when X0 turns ON, and the result is stored to Var_D0.

[Structured ladder/FBD]



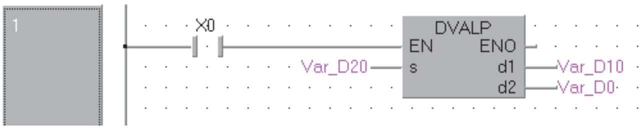
[ST]
VALP(X0,Var_D20,Var_D10,Var_D0);

[Operation]



- In the following program, the character string data in Var_D20 are read as an integer value and converted to the BIN value when X0 turns ON, and the result is stored to Var_D0.

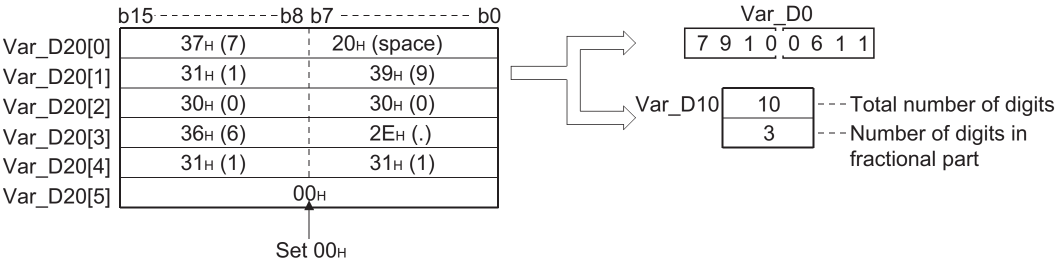
[Structured ladder/FBD]



[ST]

```
DVALP(X0,Var_D20,Var_D10,Var_D0);
```

[Operation]



	b15 ----------- b8	b7 ------------ b0
Var_D20[0]	37H (7)	20H (space)
Var_D20[1]	31H (1)	39H (9)
Var_D20[2]	30H (0)	30H (0)
Var_D20[3]	36H (6)	2EH (.)
Var_D20[4]	31H (1)	31H (1)
Var_D20[5]	00H	

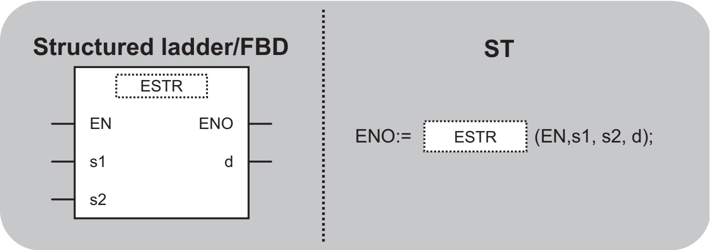
Set 00H
Var_D0
7 9 1 0 0 6 1 1
Var_D10
10 --- Total number of digits
3 --- Number of digits in fractional part

Floating-point data to character string data conversion

ESTR(P)

Ver. Basic | High performance | Process | Redundant | Universal | LCPU

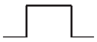
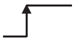
- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later



Any of the following instruction can go in the dotted squares.

ESTR, ESTRP

■Executing condition

Instruction	Executing condition
ESTR	
ESTRP	

■Argument

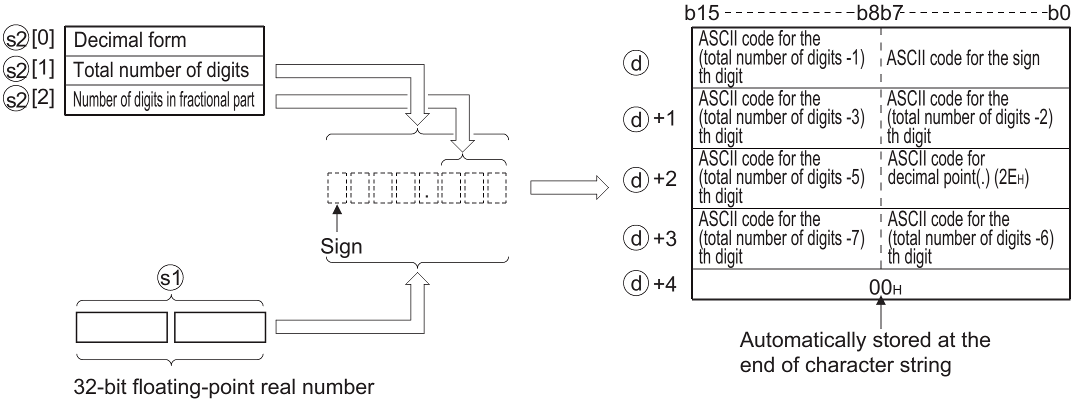
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	32-bit floating-point data to be converted, or start number of the device that stores data	Single-precision real
	s2	Start number or the array of the device that stores display specification of numeric values to be converted	Array of ANY16 (0..2)
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted character string data	String

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—	○		○	○	—
(s2)	—	○		—	—		—	—	—
(d)	—	○		—	—		—	—	—

Processing details

- Converts the 32-bit floating-point real number data specified for (s1) to a character string data according to the display specification specified for (s2), and stores the result to the device specified for (d) and the following devices.
- The post-conversion data differs depending on the display specification specified for (s2).

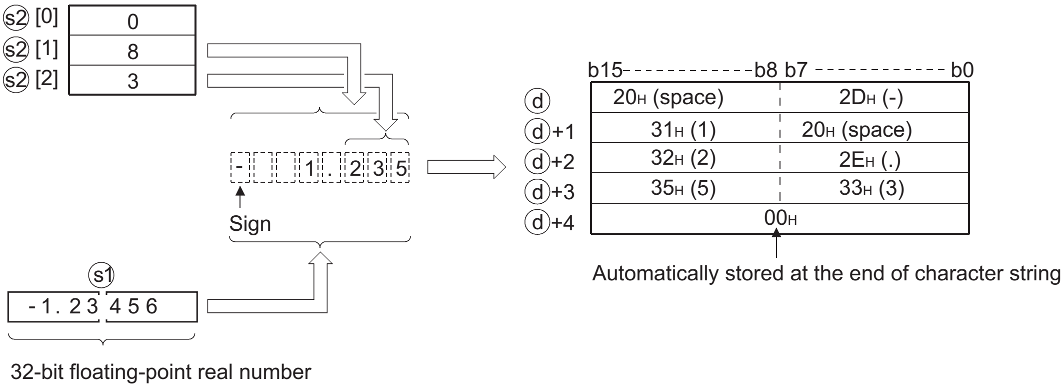
■When using decimal form



(s2)[0] Decimal form
(s2)[1] Total number of digits
(s2)[2] Number of digits in fractional part
Sign
(s1)
32-bit floating-point real number
b15
b8b7
b0
Automatically stored at the end of character string

	b15 - b8	b7 - b0
(d)	ASCII code for the (total number of digits -1) th digit	ASCII code for the sign
(d)+1	ASCII code for the (total number of digits -3) th digit	ASCII code for the (total number of digits -2) th digit
(d)+2	ASCII code for the (total number of digits -5) th digit	ASCII code for decimal point(.) (2EH)
(d)+3	ASCII code for the (total number of digits -7) th digit	ASCII code for the (total number of digits -6) th digit
(d)+4	00H	

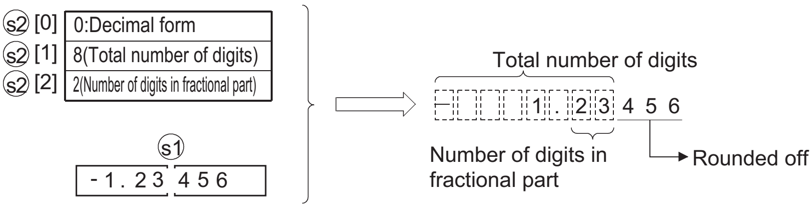
For example, when the total number of digits is 8, the number of digits in the fractional part is 3, and -1.23456 is specified, the operation result would be stored to the devices following (d) as shown below.



(s2) [0] 0
(s2) [1] 8
(s2) [2] 3
- 1 . 2 3 5
Sign
(s1)
- 1. 2 3 4 5 6
32-bit floating-point real number
b15
b8 b7
b0
Automatically stored at the end of character string

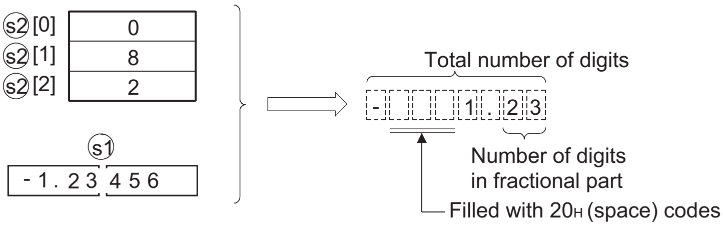
	b15 - b8	b7 - b0
(d)	20H (space)	2DH (-)
(d)+1	31H (1)	20H (space)
(d)+2	32H (2)	2EH (.)
(d)+3	35H (5)	33H (3)
(d)+4	00H	

- The total number of digits that can be specified for (s2)+1 is as shown below.
  - When the number of digits in the fractional part is 0 ··· Number of digits (max.: 24) ≥ 2
  - When the number of digits in the fractional part is other than 0 ··· Number of digits (max.: 24) ≥ (Number of digits in the fractional part + 3)
- The number of digits in the fractional part that can be specified for (s2)+2 is from 0 to 7. However, the number of digits in the fractional part must be smaller than or equal to the total number of digits minus 3.
- The converted character string data are stored to the devices following (d) as indicated below.
  - For the sign of integer part, the sign 20H (space) will be stored if the 32-bit floating-point real number is positive, and the sign 2DH (minus sign) will be stored if it is negative.
  - If the fractional part of a 32-bit floating-point real number is out of the range of the digits in the fractional part, the lower decimal values will be rounded off.



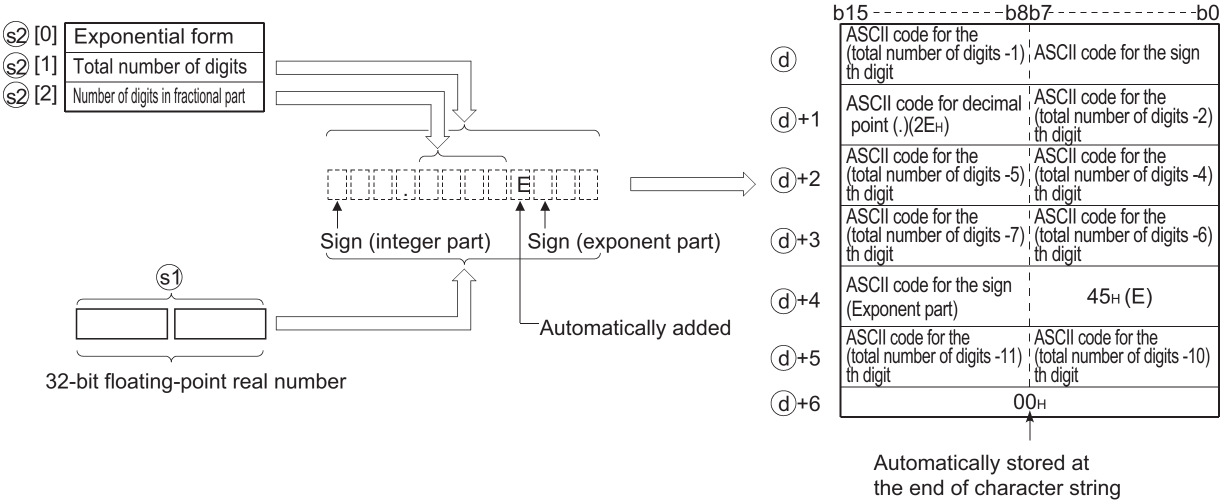
(s2) [0] 0:Decimal form
(s2) [1] 8(Total number of digits)
(s2) [2] 2(Number of digits in fractional part)
(s1)
- 1 . 2 3 4 5 6
Total number of digits
- 1 . 2 3 4 5 6
Number of digits in fractional part
Rounded off

- If the total number of digits, excluding the sign, the decimal point and the fractional part, is greater than the integer part of the 32-bit floating-point real number data, 20H (space) will be stored between the sign and the integer part. If the number of digits in the fractional part is 0, the ASCII code 2EH (.) will not be stored.
- If the total number of digits, excluding the sign, the decimal point and the fractional part, is greater than the integer part of the 32-bit floating-point real number data, 20H (space) will be stored between the sign and the integer part.



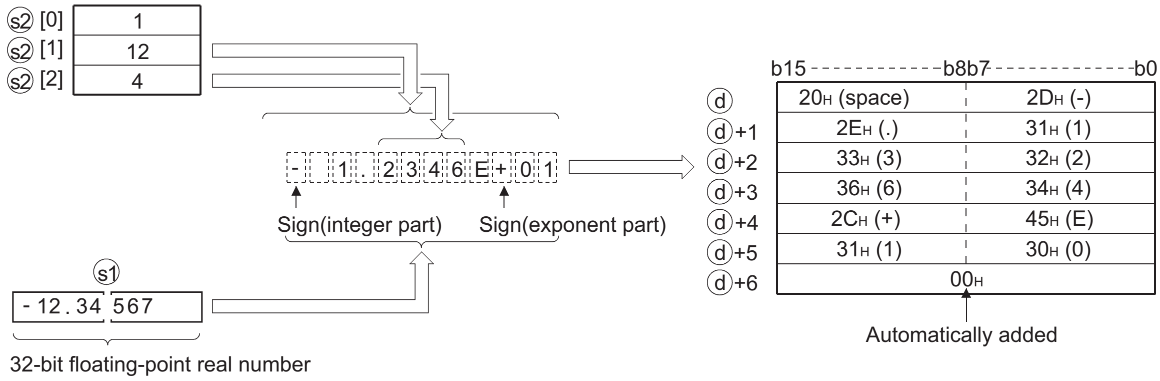
- The value 00H is automatically stored at the end of the converted character string.

■When using exponential form


s2 [0] Exponential form
s2 [1] Total number of digits
s2 [2] Number of digits in fractional part
. E
Sign (integer part)
Sign (exponent part)
s1
Automatically added
32-bit floating-point real number
Automatically stored at the end of character string

	b15 - b8	b7 - b0
d	ASCII code for the (total number of digits -1) th digit	ASCII code for the sign
d+1	ASCII code for decimal point (.)(2EH)	ASCII code for the (total number of digits -2) th digit
d+2	ASCII code for the (total number of digits -5) th digit	ASCII code for the (total number of digits -4) th digit
d+3	ASCII code for the (total number of digits -7) th digit	ASCII code for the (total number of digits -6) th digit
d+4	ASCII code for the sign (Exponent part)	45H (E)
d+5	ASCII code for the (total number of digits -11) th digit	ASCII code for the (total number of digits -10) th digit
d+6	00H	

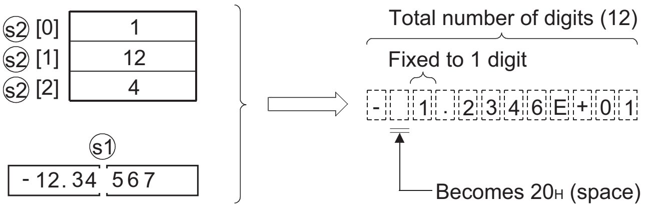
For example, when the total number of digits is 12, the number of digits in the fractional part is 4, and -12.34567 is specified, the operation results would be stored to the devices following (d) as shown below.



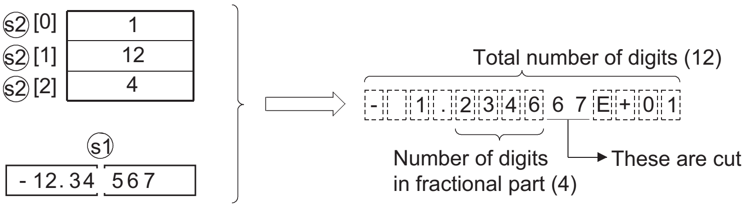
	b15 - b8	b7 - b0
d	20H (space)	2DH (-)
d+1	2EH (.)	31H (1)
d+2	33H (3)	32H (2)
d+3	36H (6)	34H (4)
d+4	2CH (+)	45H (E)
d+5	31H (1)	30H (0)
d+6	00H	

- The total number of digits that can be specified for (s2)+1 is as shown below.
  - When the number of digits in the fractional part is 0Number of digits (max.: 24) ≥ 2
  - When the number of digits in the fractional part is other than 0Number of digits (max.: 24) ≥ (Number of digits in the fractional part + 7)
- The number of digits in the fractional part that can be specified for (s2)+2 is from 0 to 7. However, the number of digits in the fractional part should be equal to or less than the total number of digits minus 7.

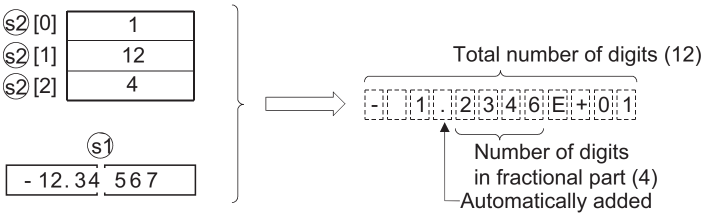
- The converted character string data are stored to the devices following (d) as indicated below.
  - If the 32-bit floating-point real number is positive in value, the sign before the integer part will be stored as ASCII code 20H (space), and if it is a negative value, the sign will be stored as 2DH (-).
  - The integer part is fixed to one digit. 20H (space) will be stored between the integer part and the sign.



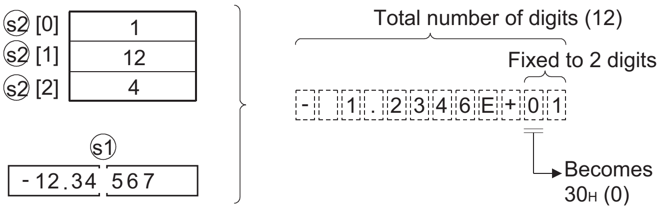
- If the fractional part of the 32-bit floating-point real number is out of the range of the digits in the fractional part, the lower decimal values will be rounded off.



- If the number of digits in the fractional part has been set at any value other than 0, 2EH (.) will automatically be stored at the position before the first of the specified number of digits. If the number of digits in the fractional part of the number is 0, the ASCII code 2EH (.) will not be stored.



- The ASCII code 2CH (+) will be stored as the sign for the exponent part of the value if the exponent is positive in value, and the code 2DH (-) will be stored if the exponent is a negative value.
- The exponent part is fixed at 2 digits. If the exponent part is only 1 digit, the ASCII code 30H (0) will be stored between the sign and the exponent part of the number.



- The value 00H is automatically stored at the end of the converted character string.

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

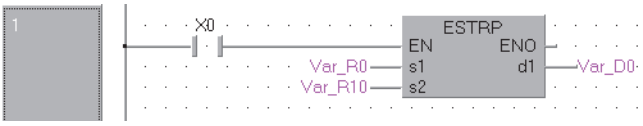
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The (s1) value is not within the range indicated below. 0, $2^{-126} \leq | (s1) | < 2^{128}$ The format specified for (s2) was neither 0 nor 1. The total number of digits specified for (s2) + 1 is outside the ranges shown below. When using decimal form • When the number of digits in the fractional part is 0: total number of digits ≥ 2 • When the number of digits in the fractional part is other than 0: total number of digits ≥ (number of digits in the fractional part +3) When using exponential form • When the number of digits in the fractional part is 0: total number of digits ≥ 6 • When the number of digits in the fractional part is other than 0: total number of digits ≥ (number of digits in the fractional part +7) The number of digits specified in the fractional part of the value for (s2)+2 was outside the ranges shown below. • When using the decimal form: number of digits in the fractional part ≤ (total number of digits - 3) • When using the exponential form: number of digits in the fractional part ≤ (total number of digits - 7) When the total number of digits exceeds the value "24".	○	○	○	○	—	—
4101	The device range to store the character string data specified for (d) exceeds the corresponding device range.	○	○	○	○	○	○
	The device specified for (s2) exceeds the corresponding device range.	—	—	—	—	○	○
4140	(s1) is out of the following range. 0, $2^{-126} \leq | (s1) | < 2^{128}$ The value of the specified device is - 0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○

Program example

- In the following program, the 32-bit floating-point real number data in Var_R0 is converted in accordance with the conversion specification stored in Var_R10 when X0 turns ON, and the result is stored to Var_D0 and the following devices.

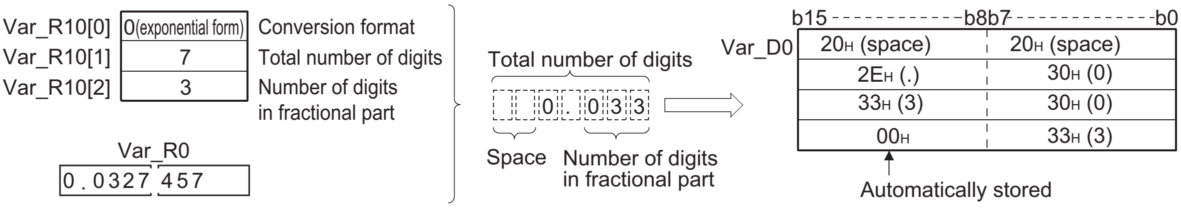
[Structured ladder/FBD]



[ST]

ESTRP(X0,Var_R0,Var_R10,Var_D0);

[Operation]



Var_R10[0]	0(exponential form)	Conversion format
Var_R10[1]	7	Total number of digits
Var_R10[2]	3	Number of digits in fractional part

Var_R0

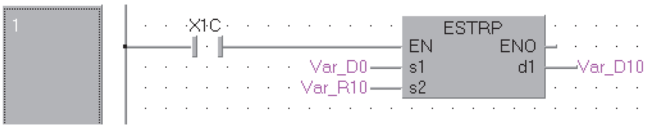
0.0327457

Var_D0

b15-b8	b7-b0
20H (space)	20H (space)
2EH (.)	30H (0)
33H (3)	30H (0)
00H	33H (3)

- In the following program, the 32-bit floating-point real number data in Var_D0 is converted in accordance with the conversion specification stored in Var_R10 when X1C turns ON, and the result is stored to Var_D10 and the following devices.

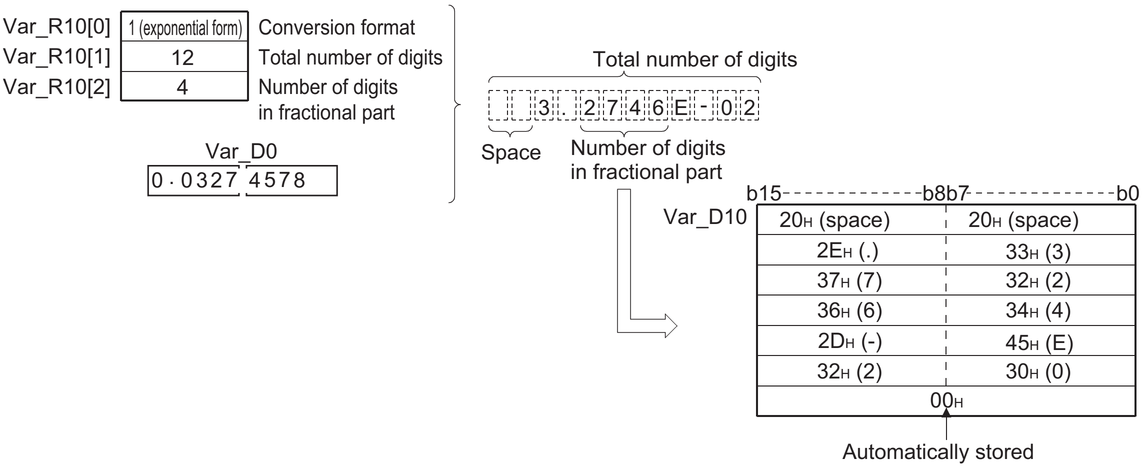
[Structured ladder/FBD]



[ST]

ESTRP(X1C,Var_D0,Var_R10,Var_D10);

[Operation]



Var_R10[0]	1 (exponential form)	Conversion format
Var_R10[1]	12	Total number of digits
Var_R10[2]	4	Number of digits in fractional part

Var_D0

0.03274578

Var_D10

b15-b8	b7-b0
20H (space)	20H (space)
2EH (.)	33H (3)
37H (7)	32H (2)
36H (6)	34H (4)
2DH (-)	45H (E)
32H (2)	30H (0)
00H	

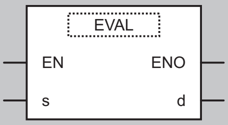
Character string data to floating-point data conversion

EVAL(P)

Ver.
Basic High performance Process Redundant Universal LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

Structured ladder/FBD

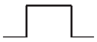
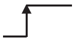


ST

ENO:= EVAL (EN, s, d);

Any of the following instruction can go in the dotted squares.
EVAL, EVALP

■Executing condition

Instruction	Executing condition
EVAL	
EVALP	

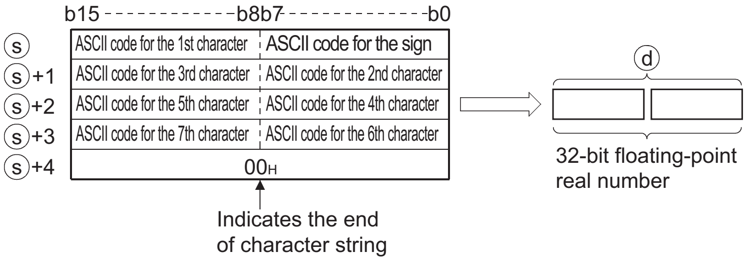
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Character string data to be converted to 32-bit floating-point real number data, or start number of the device that stores character string data	String (24)
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted 32-bit floating-point real number data	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	—		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

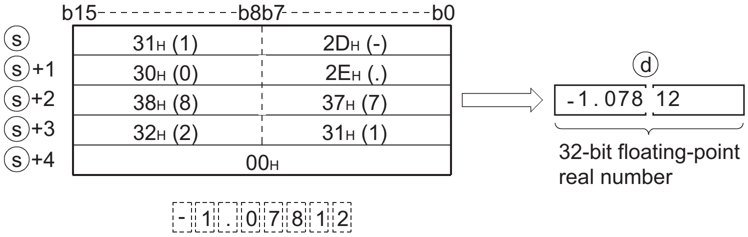
- Converts character string data stored in the device specified for (s) and the following devices to 32-bit floating-point real number data, and stores the result to the device specified for (d).
- The specified character string data can be converted to 32-bit floating-point real number data either in the decimal form or the exponential form.



	b15 - - - b8	b7 - - - b0
(s)	ASCII code for the 1st character	ASCII code for the sign
(s)+1	ASCII code for the 3rd character	ASCII code for the 2nd character
(s)+2	ASCII code for the 5th character	ASCII code for the 4th character
(s)+3	ASCII code for the 7th character	ASCII code for the 6th character
(s)+4	00H	

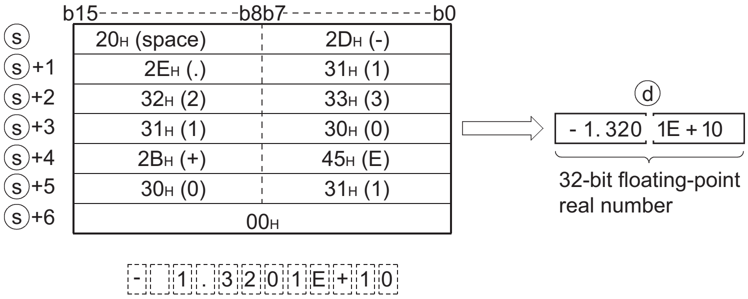
Indicates the end of character string
(d)
32-bit floating-point real number

- When using decimal form



	b15 - - - b8	b7 - - - b0
(s)	31H (1)	2DH (-)
(s)+1	30H (0)	2EH (.)
(s)+2	38H (8)	37H (7)
(s)+3	32H (2)	31H (1)
(s)+4	00H	

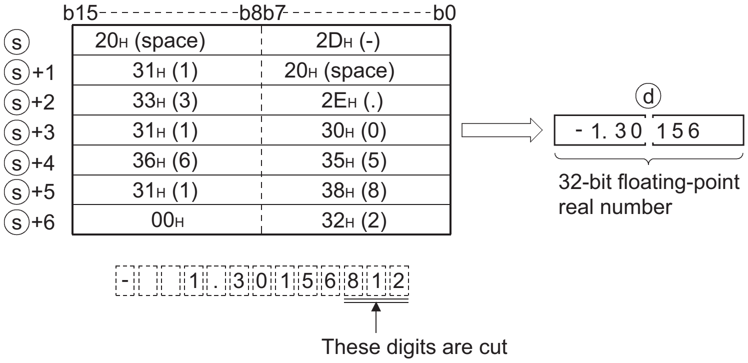
- When using exponential form



	b15 - - - b8	b7 - - - b0
(s)	20H (space)	2DH (-)
(s)+1	2EH (.)	31H (1)
(s)+2	32H (2)	33H (3)
(s)+3	31H (1)	30H (0)
(s)+4	2BH (+)	45H (E)
(s)+5	30H (0)	31H (1)
(s)+6	00H	

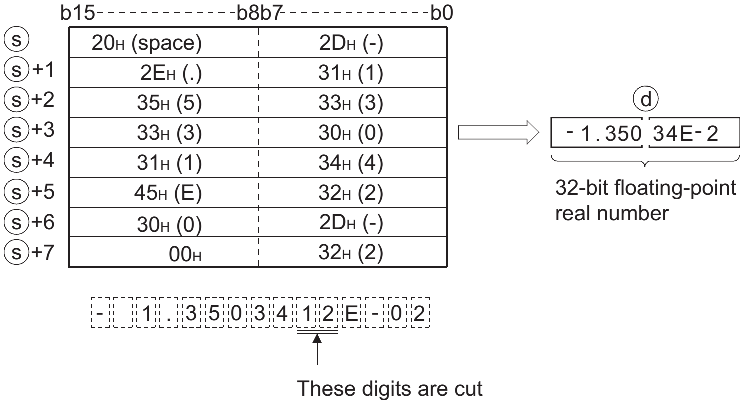
- Excluding the sign, decimal point, and exponent part of the result, 6 digits of the character string data specified for (s) to be converted to a 32-bit floating-point real number data will be effective; the 7th digit and later digit will be cut from the result.

• When using decimal form



	b15–b8	b7–b0
ⓢ	20H (space)	2DH (-)
ⓢ+1	31H (1)	20H (space)
ⓢ+2	33H (3)	2EH (.)
ⓢ+3	31H (1)	30H (0)
ⓢ+4	36H (6)	35H (5)
ⓢ+5	31H (1)	38H (8)
ⓢ+6	00H	32H (2)

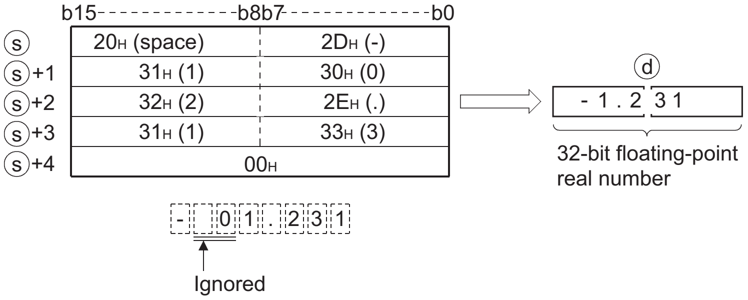
• When using exponential form



	b15–b8	b7–b0
ⓢ	20H (space)	2DH (-)
ⓢ+1	2EH (.)	31H (1)
ⓢ+2	35H (5)	33H (3)
ⓢ+3	33H (3)	30H (0)
ⓢ+4	31H (1)	34H (4)
ⓢ+5	45H (E)	32H (2)
ⓢ+6	30H (0)	2DH (-)
ⓢ+7	00H	32H (2)

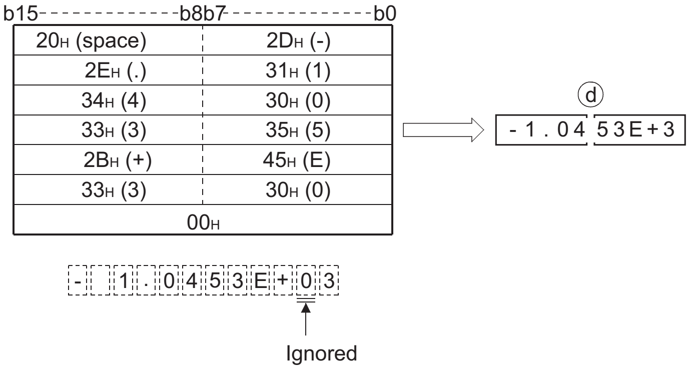
ⓓ
- 1.350 34E- 2
32-bit floating-point real number
- 1 . 3 5 0 3 4 1 2 E - 0 2
These digits are cut

- In the decimal form, if 2BH (+) is specified for the sign or if the specification of sign is omitted, conversion is made assuming a positive value. If 2DH (-) is specified for the sign, the character string data are converted as a negative value.
- In the exponential form, if 2BH (+) is specified for the sign in the exponent part or if the specification of sign is omitted, conversion is made assuming a positive value. If 2DH (-) is specified for the sign in the exponent part, the character string data are converted as a negative value.
- When the ASCII code 20H (space) or 30H (0) exists between numbers not including the initial zero in a character string data specified for (s), it will be ignored when the conversion is performed.



	b15–b8	b7–b0
ⓢ	20H (space)	2DH (-)
ⓢ+1	31H (1)	30H (0)
ⓢ+2	32H (2)	2EH (.)
ⓢ+3	31H (1)	33H (3)
ⓢ+4	00H	

- When the ASCII code 30H (0) exists between the character "E" and a number in the exponential form character string, the 30H would be ignored when the conversion is performed.



- If the 20H (space) code is contained in the character string, the code is ignored in the conversion.
- Up to 24 characters can be set for a character string. The codes 20H (space) and 30H (0) contained in the character string are also counted as a character.

Operation error

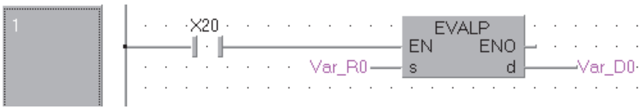
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The improper data as shown below, which cannot be converted into (s), have been set. • The characters other than "30H"(0) to "39H"(9) exist in the integral part or decimal part. • Two or more "2EH"(.) exist in the character string. • The characters other than "45H"(E), "65H" (e), "2BH" (+), "2DH" (-) exist in the character string. • The multiple exponents of "45H" (E), "65H" (e) exist in the character string. • Three digits or more numeric values are described on the exponents in the character string. • Multiple codes exist on the exponent of "2BH"(+), "2DH(-)" in the character string. • Multiple codes of "2BH"(+), "2DH(-)" exist in the positive number part of the character string in a decimal point format; in the mantissa part of the character string in an exponential format. The number of characters in the character string following (s) is either 0 or more than 24.	○	○	○	○	○	○
4101	The code 00H does not appear within the range from (s) to the corresponding device.	○	○	○	○	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) |Data after conversion| < 2^{128}	—	—	—	—	○	○

Program example

- In the following program, the character string data in Var_R0 and the following devices are converted to the 32-bit floating-point real number data when X20 turns ON, and the result is stored to Var_D0.

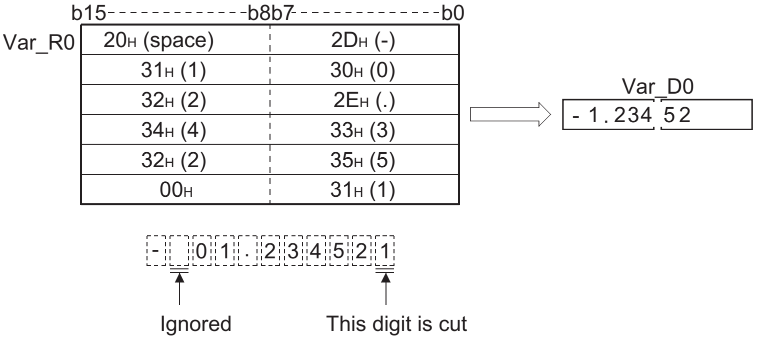
[Structured ladder/FBD]



[ST]

EVALP(X20,Var_R0,Var_D0);

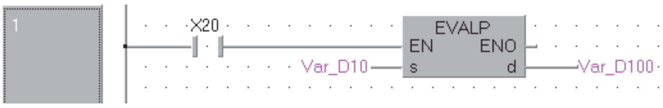
[Operation]



	b15 - - - b8	b7 - - - b0
Var_R0	20H (space)	2DH (-)
	31H (1)	30H (0)
	32H (2)	2EH (.)
	34H (4)	33H (3)
	32H (2)	35H (5)
	00H	31H (1)

- In the following program, the character string data in Var_D10 and the following devices are converted to the 32-bit floating-point real number data when X20 turns ON, and the result is stored to Var_D100.

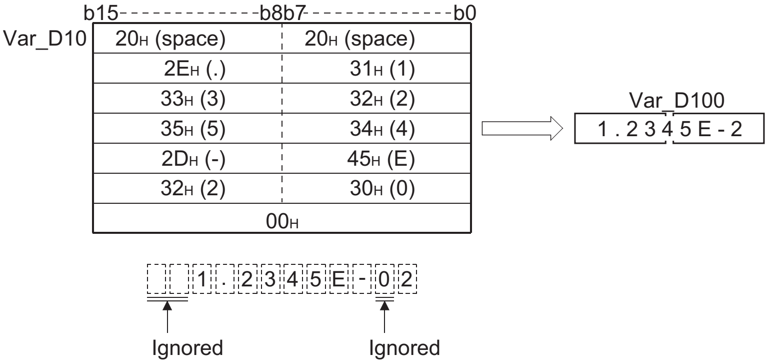
[Structured ladder/FBD]



[ST]

EVALP(X20,Var_D10,Var_D100);

[Operation]

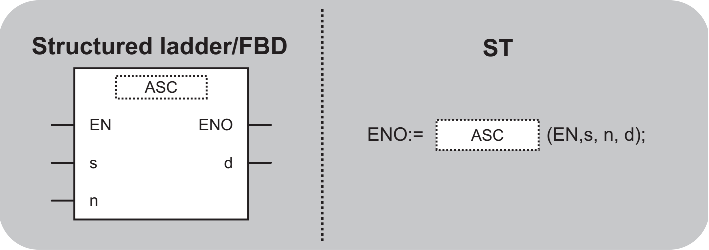


	b15 - - - b8	b7 - - - b0
Var_D10	20H (space)	20H (space)
	2EH (.)	31H (1)
	33H (3)	32H (2)
	35H (5)	34H (4)
	2DH (-)	45H (E)
	32H (2)	30H (0)
	00H	

Hexadecimal BIN data to ASCII data conversion

ASC(P)

~~Basic~~ High performance Process Redundant Universal LCPU

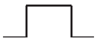
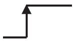


```
ENO:= ASC (EN,s, n, d);
```

Any of the following instruction can go in the dotted squares.

ASC, ASCP

■Executing condition

Instruction	Executing condition
ASC	
ASCP	

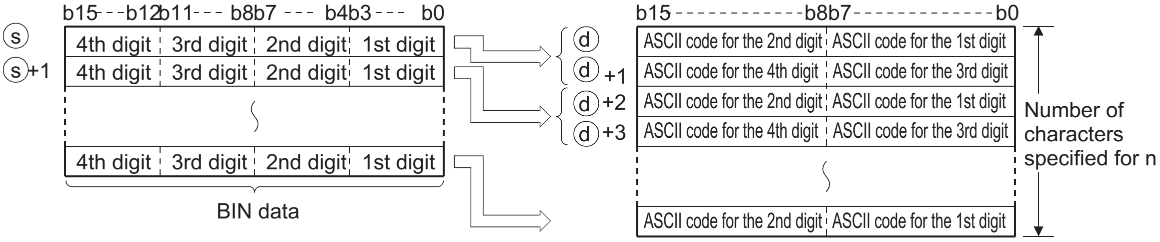
■Argument

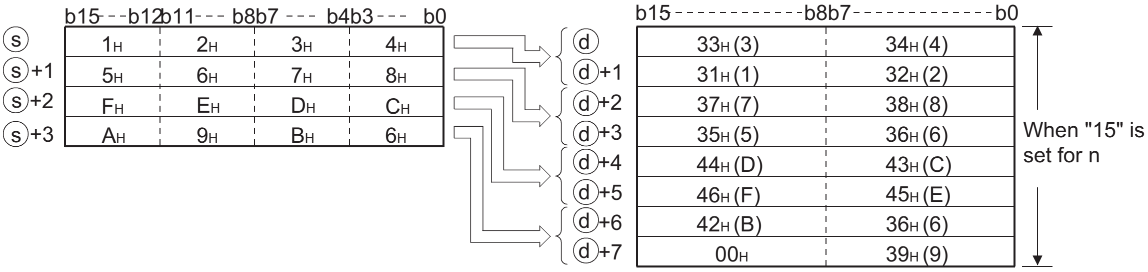
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores BIN data to be converted to character string data	ANY16
	n	Execution result	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted character string data	String

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—					—
n	○	○		○					—
(d)	—	○		—					—

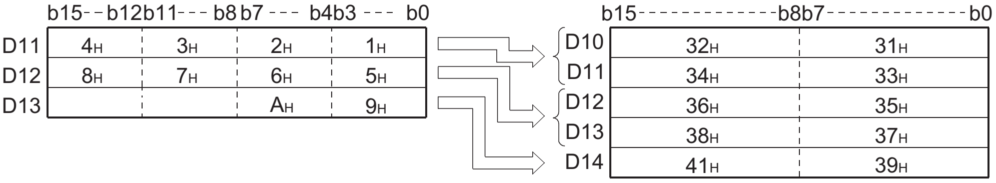
Processing details

- Converts the 16-bit BIN data stored in the device specified for (s) and the following devices to ASCII data by treating the BIN data in hexadecimal representation. Then, stores the converted data to the device specified for (d) and the following devices, for the number of characters specified for n.


b15 - - b12 b11 - - - b8 b7 - - - b4 b3 - - - b0
s
s +1
4th digit 3rd digit 2nd digit 1st digit
4th digit 3rd digit 2nd digit 1st digit
4th digit 3rd digit 2nd digit 1st digit
BIN data
d
d +1
d +2
d +3
b15 - - - b8 b7 - - - b0
ASCII code for the 2nd digit ASCII code for the 1st digit
ASCII code for the 4th digit ASCII code for the 3rd digit
ASCII code for the 2nd digit ASCII code for the 1st digit
ASCII code for the 4th digit ASCII code for the 3rd digit
ASCII code for the 2nd digit ASCII code for the 1st digit
Number of characters specified for n

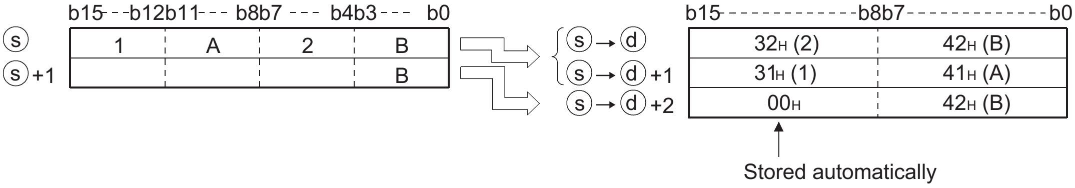


- The use of n to set the number of characters causes the BIN data range specified for (s) and the character string storage device range specified for (d) to be set automatically.
- Processing will be performed accurately even if the device range where BIN data to be converted is being stored overlaps with the device range where the converted ASCII data will be stored.



- If an odd number of characters has been specified for n, the ASCII code 00H will be automatically stored in the upper 8 bits of the final device in the range where the character string is to be stored.

When 5 characters have been specified for n.



- If the number of characters specified for n is 0, conversion processing will not be performed.

Operation error

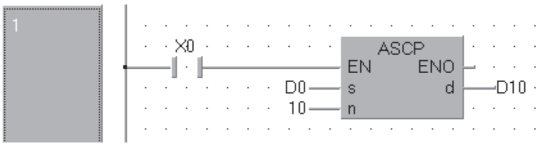
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The character numbers designated by n are those other than 0 to 16383.	—	○	○	○	○	○
4101	The range of the number of characters specified for n following the device number specified for (s) exceeds the corresponding device range. The range of the number of characters specified for n following the device number specified for (d) exceeds the corresponding device range.	—	○	○	○	○	○

Program example

- In the following program, the BIN data in D0 are read as a hexadecimal value, and converted to the character string data when X0 turns ON, and the results are stored to the devices from D10 to D14.

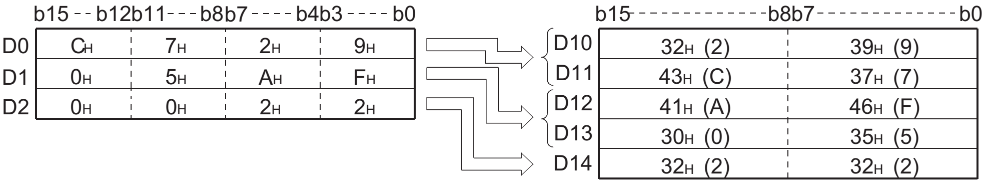
[Structured ladder/FBD]



[ST]

ASCP(X0,D0,10,D10);

[Operation]



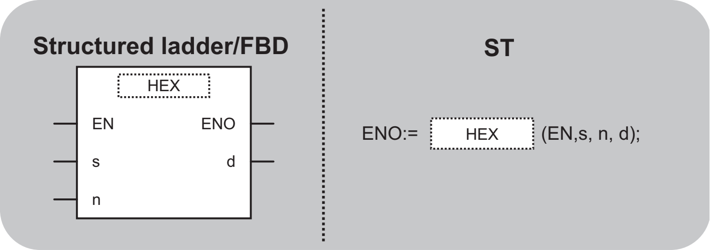
	b15 - - b12	b11 - - - b8	b7 - - - - b4	b3 - - - - b0
D0	CH	7H	2H	9H
D1	0H	5H	AH	FH
D2	0H	0H	2H	2H

	b15 - - - - - - - - - - - - b8	b7 - - - - - - - - - - - - - b0
D10	32H (2)	39H (9)
D11	43H (C)	37H (7)
D12	41H (A)	46H (F)
D13	30H (0)	35H (5)
D14	32H (2)	32H (2)

ASCII data to hexadecimal BIN data conversion

HEX(P)

Basic High performance Process Redundant Universal LCPU

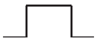
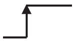


```
ENO:= HEX (EN,s, n, d);
```

Any of the following instruction can go in the dotted squares.

HEX, HEXP

■Executing condition

Instruction	Executing condition
HEX	
HEXP	

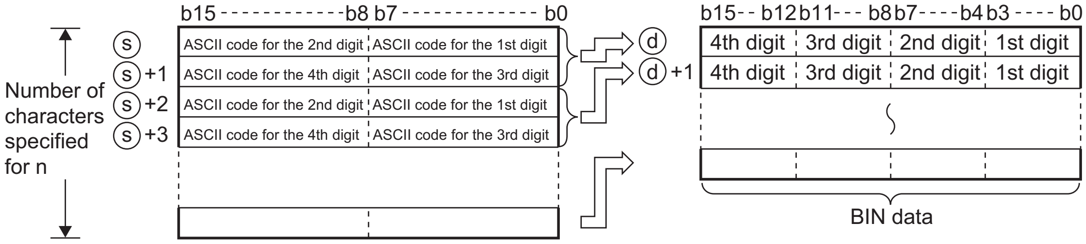
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores character string data to be converted to BIN data	String
	n	Number of characters to be stored	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted BIN data	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—					—
n	○	○		○					—
(d)	—	○		—					—

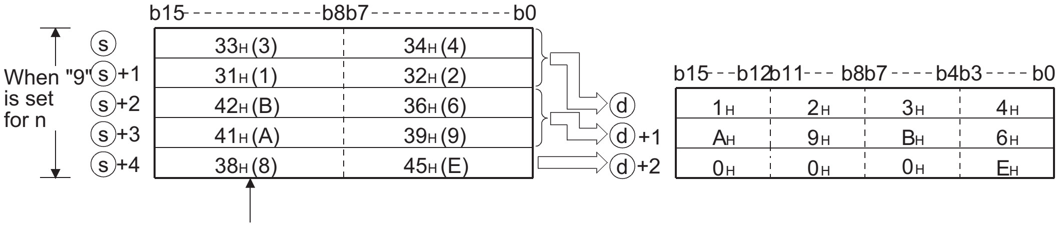
Processing details

- Converts the n characters of hexadecimal ASCII data in the device specified for (s) and the following devices to the BIN value and stores the results to the device specified for (d) and the following devices.



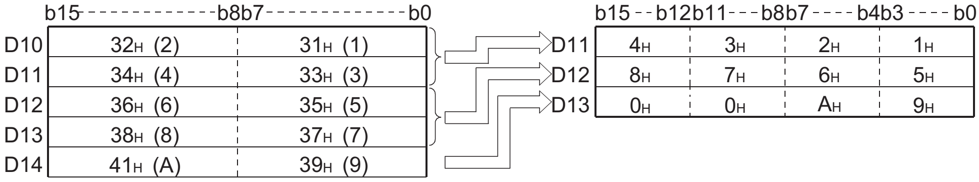
b15 - - - b8 b7 - - - b0
Number of characters specified for n
s
s +1
s +2
s +3
ASCII code for the 2nd digit
ASCII code for the 1st digit
ASCII code for the 4th digit
ASCII code for the 3rd digit
ASCII code for the 2nd digit
ASCII code for the 1st digit
ASCII code for the 4th digit
ASCII code for the 3rd digit
b15 - - b12 b11 - - - b8 b7 - - - b4 b3 - - - b0
d
d +1
4th digit
3rd digit
2nd digit
1st digit
4th digit
3rd digit
2nd digit
1st digit
BIN data

For example, if the number 9 has been specified for n, the operation will be as shown below.

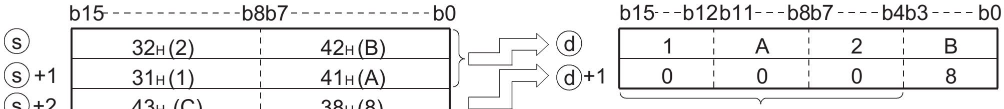


Code "38H" remains unchanged since the specified number of characters is "9".

- When the number of characters is specified for n, the range of characters specified for (s) as well as the device range specified for (d) in which the BIN data will be stored are automatically decided.
- Accurate processing will be performed even in cases where the range of devices where the ASCII data to be converted is being stored overlaps with the range of devices that will store the converted BIN data.



- If the number of characters specified for n is not divisible by 4, 0 will be automatically stored after the specified number of characters to the final device number of the devices which are storing the converted BIN value.



Value "0" is automatically stored to the area outside the range of the specified number of characters.

- If the number of characters specified for n is 0, conversion processing will not be performed.
- ASCII code that can be specified for (s) is within the range of 30H to 39H and 41H to 46H.

Operation error

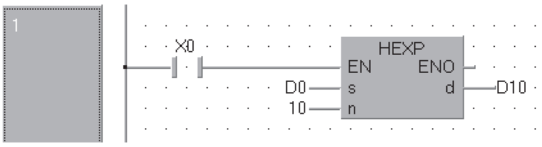
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	Characters outside the hexadecimal character string (characters that are not within the range of 30H to 39H, or 41H to 46H) have been set for (s). The character numbers designated by n are those other than 0 to 16383.	—	○	○	○	○	○
4101	The range of the number of characters specified for n following the device number specified for (s) exceeds the corresponding device range. The range of the number of characters specified for n following the device number specified for (d) exceeds the corresponding device range. The number of characters specified for n is a negative value.	—	○	○	○	○	○

Program example

- In the following program, the character string data in D0 to D4 are converted to the BIN data when X0 turns ON, and the results are stored to the devices from D10 to D14.

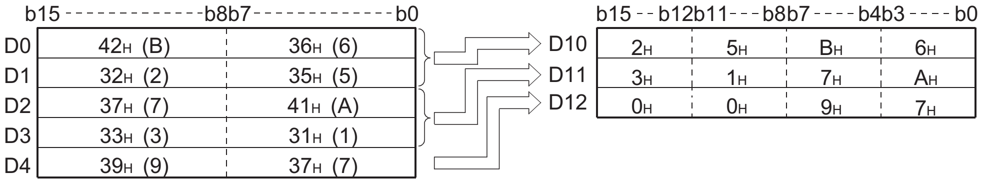
[Structured ladder/FBD]



[ST]

HEXP(X0,D0,10,D10);

[Operation]



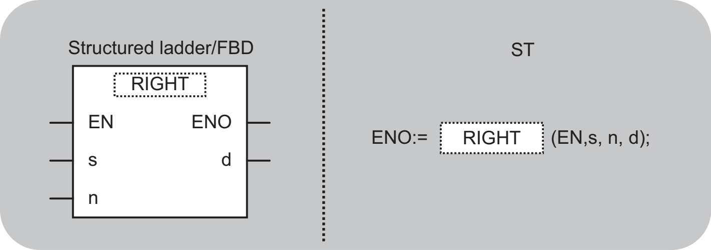
	b15------------b8	b7------------b0
D0	42_H (B)	36_H (6)
D1	32_H (2)	35_H (5)
D2	37_H (7)	41_H (A)
D3	33_H (3)	31_H (1)
D4	39_H (9)	37_H (7)

	b15--b12	b11---b8	b7----b4	b3----b0
D10	2_H	5_H	B_H	6_H
D11	3_H	1_H	7_H	A_H
D12	0_H	0_H	9_H	7_H

Extraction of character string data from right/left

RIGHT(P), LEFT(P)

Basic High performance Process Redundant Universal LCPU

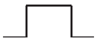
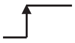


```
ENO:= RIGHT (EN,s, n, d);
```

Any of the following instruction can go in the dotted squares.

RIGHT, RIGHTP, LEFT, LEFTP

■Executing condition

Instruction	Executing condition
RIGHT LEFT	
RIGHTP, LEFTP	

■Argument

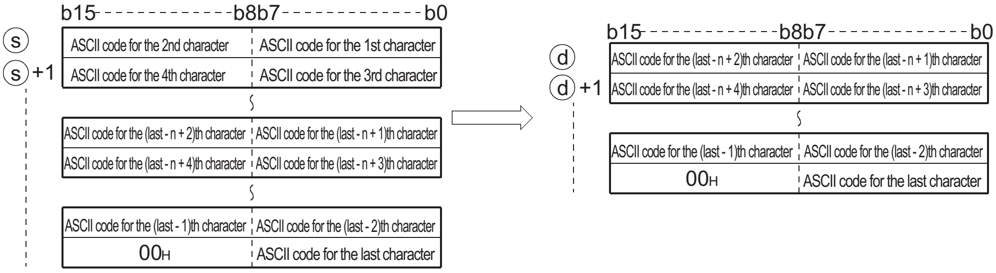
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Character string data, or start number of the device that stores character string data	String
	n	Number of characters to be extracted	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores character string data consist of n characters from right or left of s	String

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant		Others
	Bit	Word		Bit	Word			K, H	$	
(s)	—	○		—				—	○	—
n	○	○		○				○	—	—
(d)	—	○		—				—	—	—

Processing details

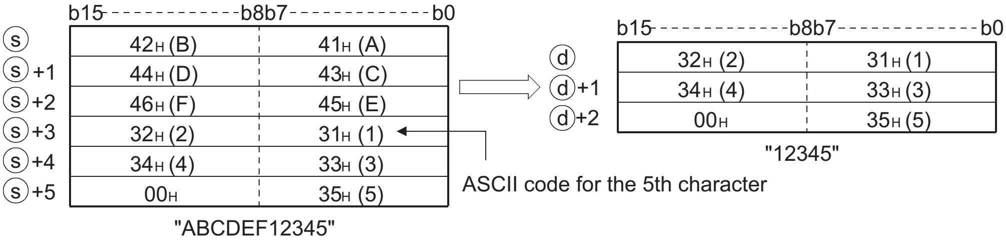
■RIGHT(P)

- Stores n characters from the right side of the character string data (the end of the character string data) which is stored in the device specified for (s) and the following devices, to the device specified for (d) and the following devices.



b15 b8b7 b0
s
s +1
ASCII code for the 2nd character
ASCII code for the 1st character
ASCII code for the 4th character
ASCII code for the 3rd character
ASCII code for the (last - n + 2)th character
ASCII code for the (last - n + 1)th character
ASCII code for the (last - n + 4)th character
ASCII code for the (last - n + 3)th character
ASCII code for the (last - 1)th character
ASCII code for the (last - 2)th character
00H
ASCII code for the last character
d
d +1
ASCII code for the (last - n + 2)th character
ASCII code for the (last - n + 1)th character
ASCII code for the (last - n + 4)th character
ASCII code for the (last - n + 3)th character
ASCII code for the (last - 1)th character
ASCII code for the (last - 2)th character
00H
ASCII code for the last character

When n = 5



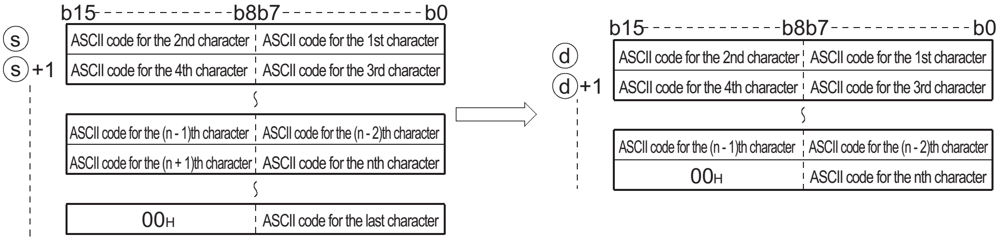
- The NULL code (00H) indicating the end of the character string is automatically appended at the end of the character string data. For the format of the character string data, refer to the following.

MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

- If the number of characters specified for n is 0, the NULL code (00H) will be stored to (d).

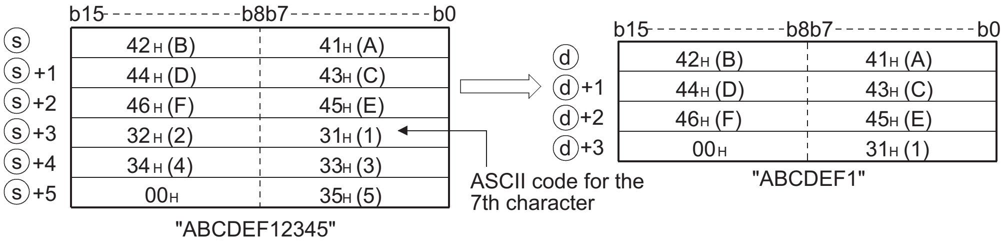
■LEFT(P)

- Stores n characters from the left side of the character string data (the beginning of the character string data) which is stored in the device specified for (s) and the following devices, to the device specified for (d) and the following devices.



b15 b8b7 b0
s
s +1
ASCII code for the 2nd character
ASCII code for the 1st character
ASCII code for the 4th character
ASCII code for the 3rd character
ASCII code for the (n - 1)th character
ASCII code for the (n - 2)th character
ASCII code for the (n + 1)th character
ASCII code for the nth character
00H
ASCII code for the last character
d
d +1
ASCII code for the 2nd character
ASCII code for the 1st character
ASCII code for the 4th character
ASCII code for the 3rd character
ASCII code for the (n - 1)th character
ASCII code for the (n - 2)th character
00H
ASCII code for the nth character

When n = 7



- The NULL code (00H) indicating the end of the character string is automatically appended to the end of the character string data. For the format of the character string data, refer to the following.

MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

- If the number of characters specified for n is 0, the NULL code (00H) will be stored to (d).

Operation error

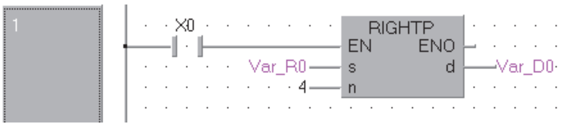
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The character number of character string designated by (s) is "0".	—	○	○	○	○	○
4101	The value of n exceeds the number of characters specified for (s). The range of n characters from (d) exceeds the corresponding device.	—	○	○	○	○	○

Program example

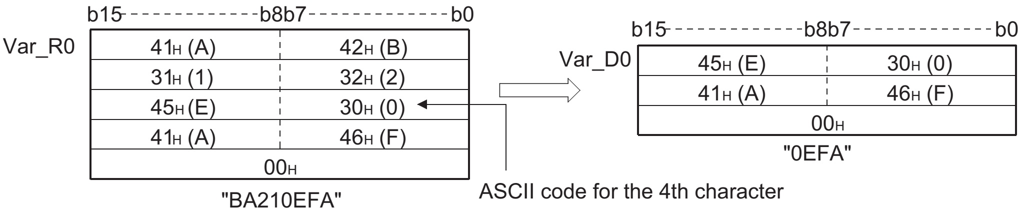
- In the following program, the 4 characters of data from the rightmost of the character string data stored in Var_R0 and the following devices are extracted and stored to Var_D0 and the following devices when X0 turns ON.

[Structured ladder/FBD]



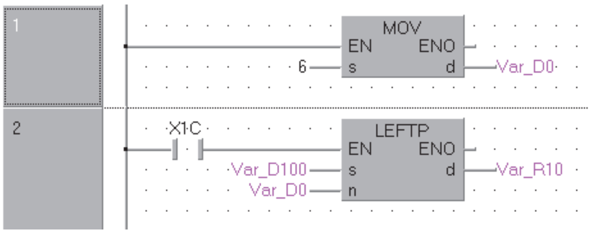
[ST]
RIGHTP(X0,Var_R0,4,Var_D0);

[Operation]



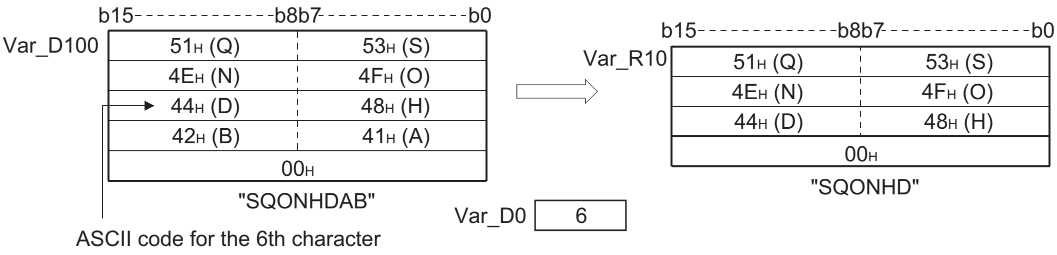
- In the following program, the number of characters corresponding to the value being stored in Var_D0 from the leftmost of the character string data in Var_D100 are extracted and stored to Var_R10 and the following devices when X1C turns ON.

[Structured ladder/FBD]



[ST]
Var_D0:=6;
LEFTP(X1C,Var_D100,Var_D0,Var_R10);

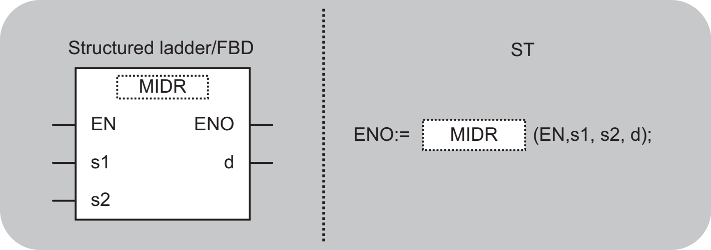
[Operation]



Random selection and replacement in character string data

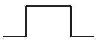
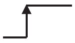
MIDR(P), MIDW(P)

~~Basic~~ High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.
MIDR, MIDRP, MIDW, MIDWP

■Executing condition

Instruction	Executing condition
MIDR, MIDW	
MIDRP, MIDWP	

■Argument

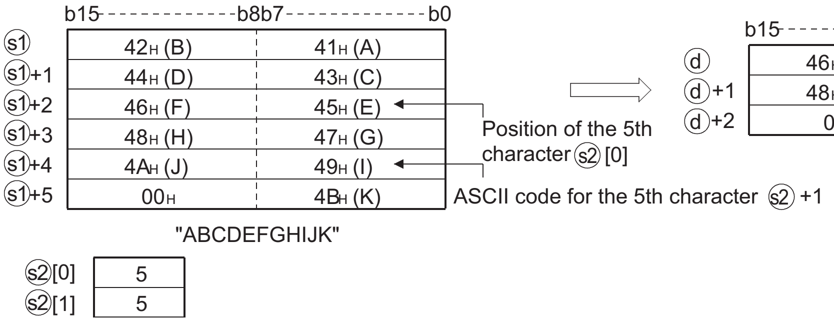
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Character string data, or start number of the device that stores character string data	String
	s2	Start number or the array of the device that stores positions of first character and number of characters • s2[0]: Position of first character • s2[1]: Number of characters	Array of ANY16 (1..2)
Output argument	ENO	Execution result	Bit
	d	Character string data from operation result	String

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				○	—
(s2)	○	○		○				—	—
(d)	—	○		—				—	—

Processing details

■MIDR(P)

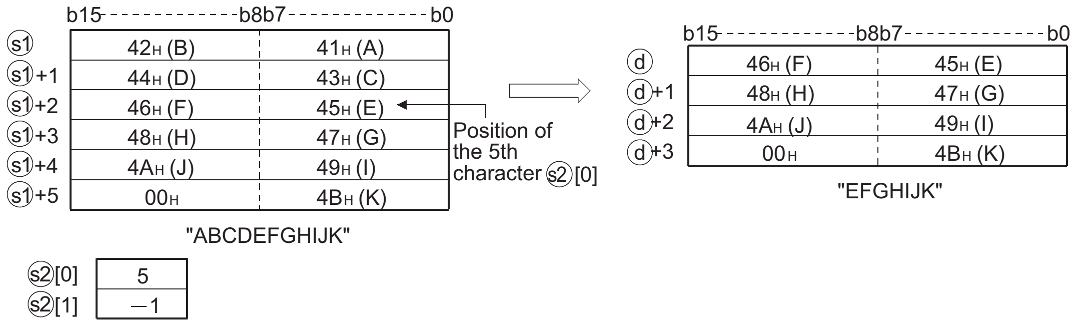
- Extracts the character string data of (s)[1] characters, from the position specified for (s2)[0], counted from the left end of the character string data specified for (s1), and stores the extracted data to the device specified for (d) and the following devices.



- The NULL code (00H) indicating the end of the character string is automatically appended to the end of the character string data. For the format of the character string data, refer to the following.

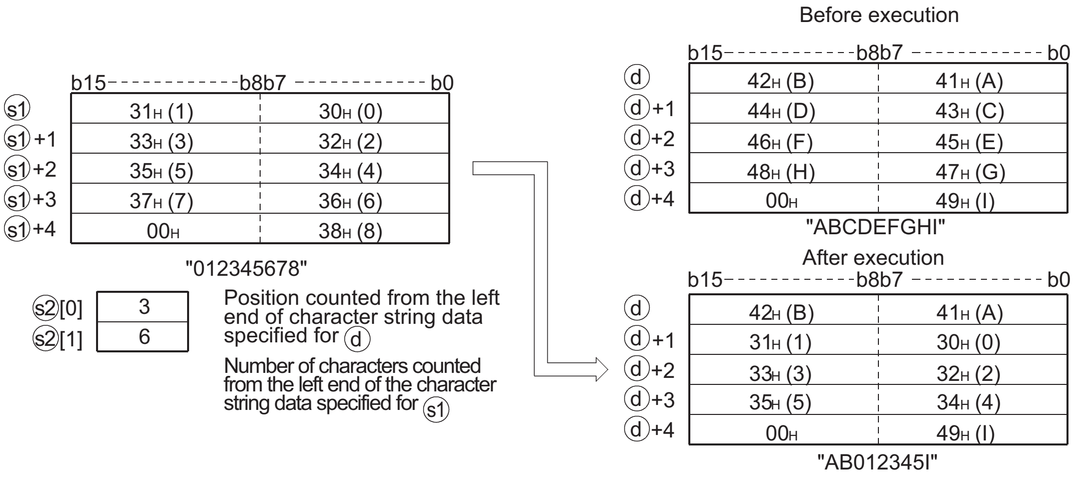
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

- If the number of characters specified for (s2)[1] is 0, the NULL code (00H) is stored in the top of (d).
- If the number of characters specified for (s2)[1] is -1, stores the data up to the final character specified for (s1) to the device specified for (d) and the following devices.



■MIDW(P)

- Extracts the character string data of (s2)[1] characters, from the left end of the character string data specified for (s1), and stores the extracted data to the character string data specified for (d) to the devices starting from the position specified for (s2)[0] from the left end.

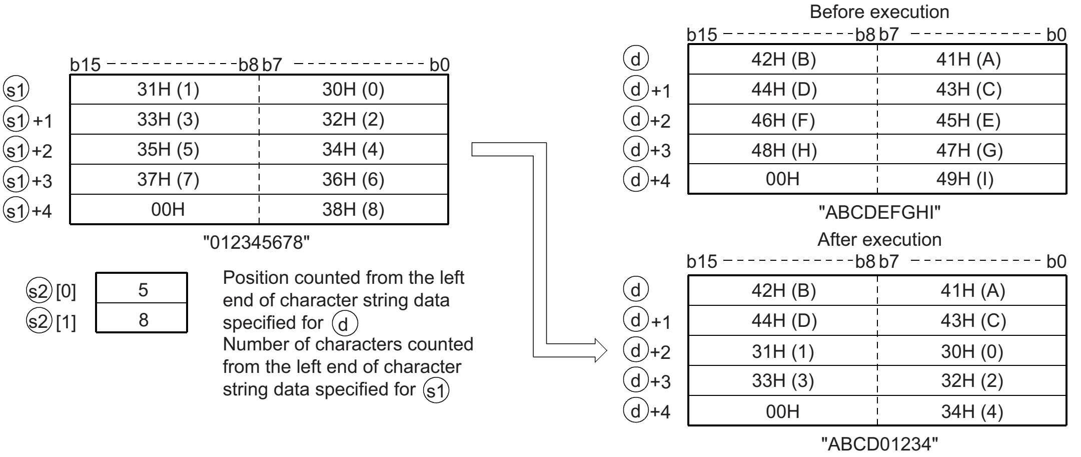


- The NULL code (00H) indicating the end of the character string is automatically appended to the end of the character string data. For the format of the character string data, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

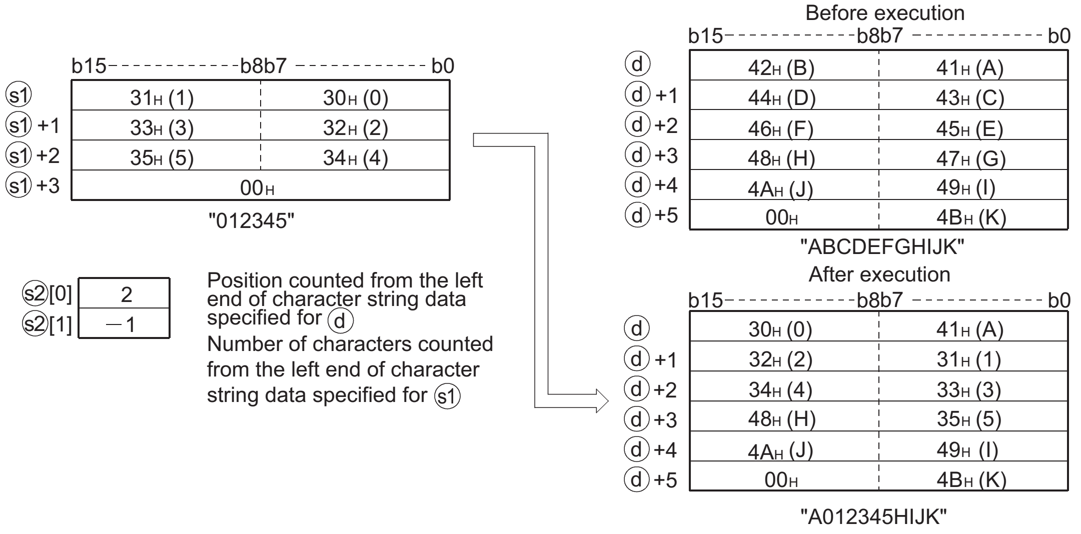
- If the number of characters specified for (s2)[1] is 0, the NULL code (00H) is stored in the top of (d).

- If the number of characters specified for (s2)[1] exceeds the final character from the character string data specified for (d), data will be stored up to the final character.



Characters "35H" (5) to "37H" (7) are not stored.

- If the number of characters specified for (s2)[1] is -1, stores the data up to the final character specified for (s1) to the device specified for (d) and the following devices.



Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

For MIDR(P) instruction

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The value of (s2)[0] exceeds the number of characters specified for (s1). The value that is added (s2)[0] and (s2)[1] exceeds the number of characters specified for (s1). The (s2)[1] number of characters from position (d) exceeds the device range of (d). When the value (s2)[0] is 0. The values of (s2)[1] are those other than the effective values (-1, 0, 1 or over). 00H does not exist within the specified device range following the devices specified for (s1).	—	○	○	○	○	○

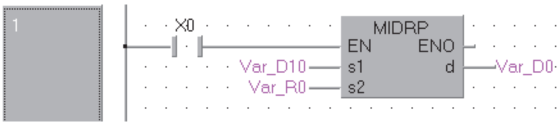
For MIDW(P) instruction

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The value of (s2)[0] exceeds the number of characters specified for (d). The value that is added (s2)[0] and (s2)[1] exceeds the number of characters specified for (s1). The (s2)[1] value exceeds the number of characters in (s1). When the value (s2)[0] is 0. The values of (s2)[1] are those other than the effective values (-1, 0, 1 or over). 00H does not exist within the specified device range following the devices specified for (s1).	—	○	○	○	○	○

Program example

- In the following program, the 3rd character through the 6th character from the left of the character string data stored in Var_D10 and the following devices are extracted and stored to Var_D0 and the following devices when X0 turns ON.

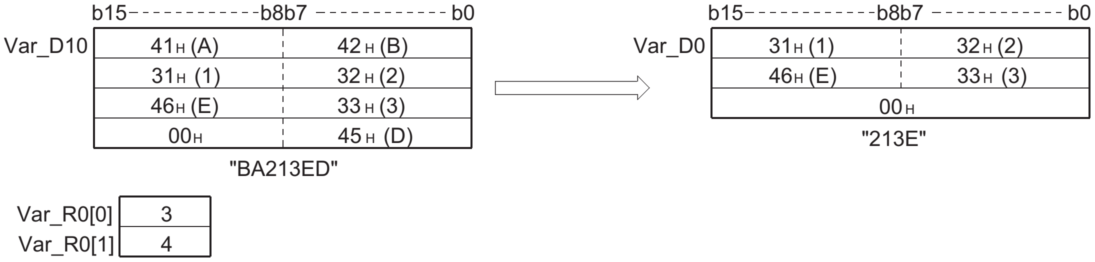
[Structured ladder/FBD]



[ST]

MIDRP(X0,Var_D10,Var_R0,Var_D0);

[Operation]



Var_D10	b15–b8	b7–b0
	41H (A)	42 H (B)
	31H (1)	32 H (2)
	46H (E)	33 H (3)
	00H	45 H (D)

"BA213ED"

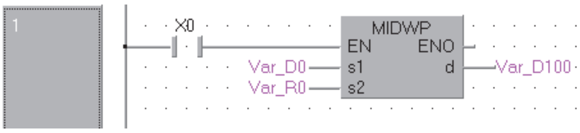
Var_D0	b15–b8	b7–b0
	31H (1)	32H (2)
	46H (E)	33H (3)
	00H	

"213E"

Var_R0[0]	3
Var_R0[1]	4

- In the following program, the 4 characters of the character string data stored in Var_D0 and the following devices are extracted and stored to the devices from the 3rd character from the left of the character string data in Var_D100 and the following devices when X0 turns ON.

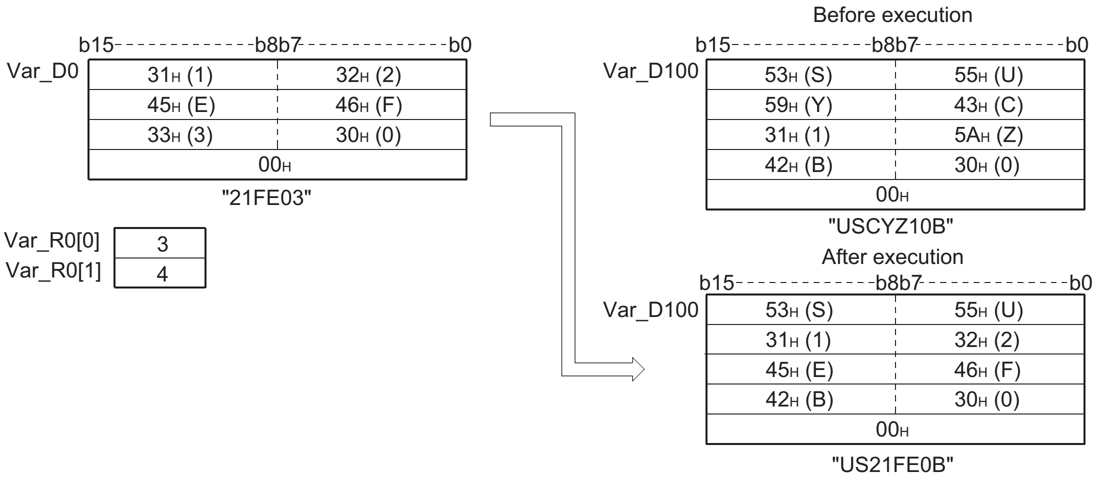
[Structured ladder/FBD]



[ST]

MIDWP(X0,Var_D0,Var_R0,Var_D100);

[Operation]



Var_D0	b15–b8	b7–b0
	31H (1)	32H (2)
	45H (E)	46H (F)
	33H (3)	30H (0)
	00H	

"21FE03"

Var_R0[0]	3
Var_R0[1]	4

Before execution

Var_D100	b15–b8	b7–b0
	53H (S)	55H (U)
	59H (Y)	43H (C)
	31H (1)	5AH (Z)
	42H (B)	30H (0)
	00H	

"USCYZ10B"

After execution

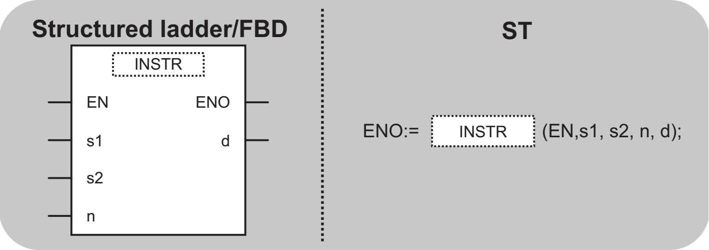
Var_D100	b15–b8	b7–b0
	53H (S)	55H (U)
	31H (1)	32H (2)
	45H (E)	46H (F)
	42H (B)	30H (0)
	00H	

"US21FE0B"

Character string data search

INSTR(P)

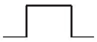
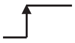
~~Basic~~ High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

INSTR, NSTRP

■Executing condition

Instruction	Executing condition
INSTR	
INSTRP	

■Argument

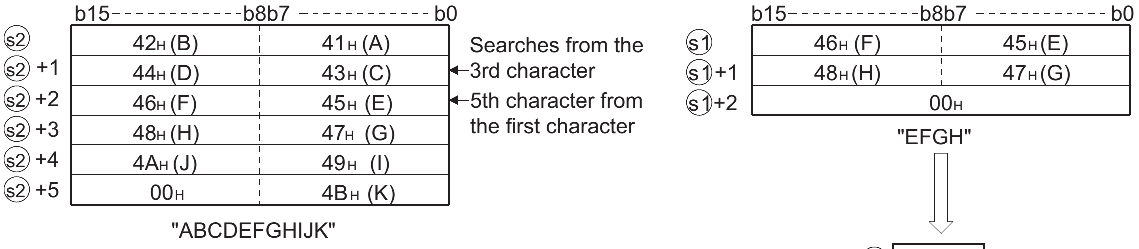
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Character string data to be searched, or start number of the device that stores character string data to be searched	String
	s2	Character string data in which character string data are searched, or start number of the device that stores character string data in which character string data are searched	String
	n	Search start position	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the search result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant		Others
	Bit	Word		Bit	Word			K, H	$	
(s1)	—	○		—				—	○	—
(s2)	—	○		—				—	○	—
n	○	○		○				○	—	—
(d)	○	○		○				—	—	—

Processing details

- Searches for the character string data specified for (s1) in the devices from the nth character from the left of the character string data specified for (s2) and stores the result to the device specified for (d). As the result of search, the location of match, counted in the number of characters from the first character of the character string data specified for (s2), is stored.

When n = 3



Searches from the 3rd character

5th character from the first character

Stores the position of the found character, counted by the number of characters from the 1st character in the character string data specified for s2.

- If there is no matching character string data, stores 0 to (d).

Operation error

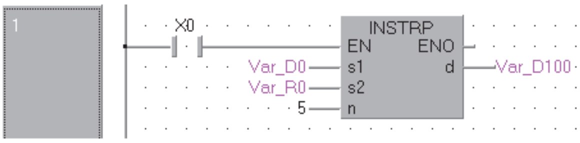
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of n exceeds the number of characters in (s2). The number of character specified by (s1) is 0. 00H (NULL) does not exist within the corresponding device range following the devices specified for (s1) and (s2). When n is a negative value or 0.	—	○	○	○	○	○

Program example

- In the following program, the character string data stored in Var_D0 and the following devices are searched from the 5th character from the left of the character string data stored in Var_R0 and the following devices when X0 turns ON, and the result is stored to Var_D100.

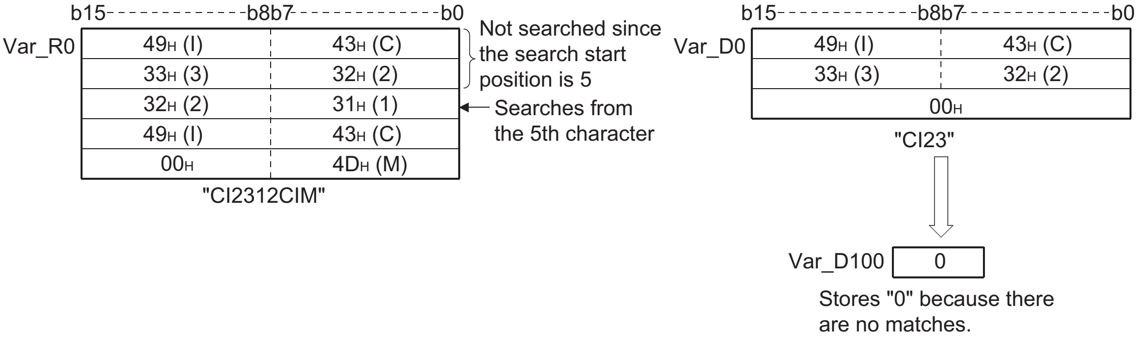
[Structured ladder/FBD]



[ST]

```
INSTRP(X0,Var_D0,Var_R0,5,Var_D100);
```

[Operation]


b15
b8b7
b0
Var_R0
Not searched since the search start position is 5
Searches from the 5th character
b15
b8b7
b0
Var_D0
Var_D100
0
Stores "0" because there are no matches.

Var_R0

b15-b8	b7-b0
49H (I)	43H (C)
33H (3)	32H (2)
32H (2)	31H (1)
49H (I)	43H (C)
00H	4DH (M)

"CI2312CIM"

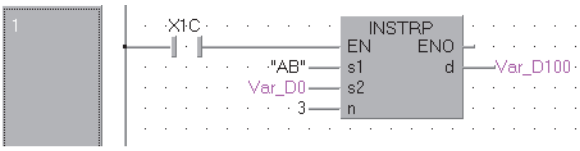
Var_D0

b15-b8	b7-b0
49H (I)	43H (C)
33H (3)	32H (2)
00H	

"CI23"

- In the following program, the character string data "AB" is searched from the 3rd character from the left of the character string data stored in Var_D0 and the following devices when X1C turns ON, and the result is stored to Var_D100.

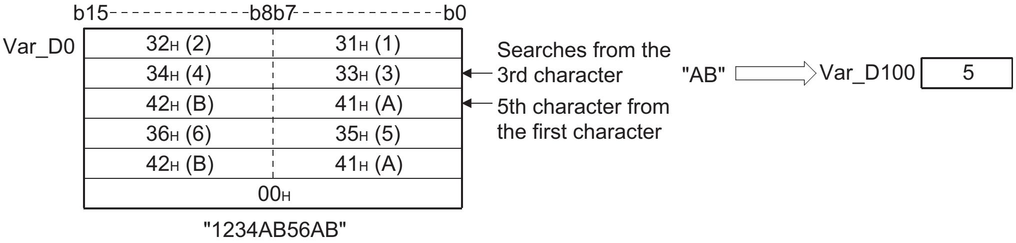
[Structured ladder/FBD]



[ST]

```
INSTRP(X0,"AB",Var_D0,3,Var_D100);
```

[Operation]


b15
b8b7
b0
Var_D0
Searches from the 3rd character
5th character from the first character
"AB"
Var_D100
5

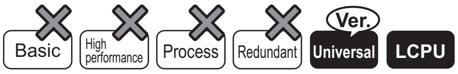
Var_D0

b15-b8	b7-b0
32H (2)	31H (1)
34H (4)	33H (3)
42H (B)	41H (A)
36H (6)	35H (5)
42H (B)	41H (A)
00H	

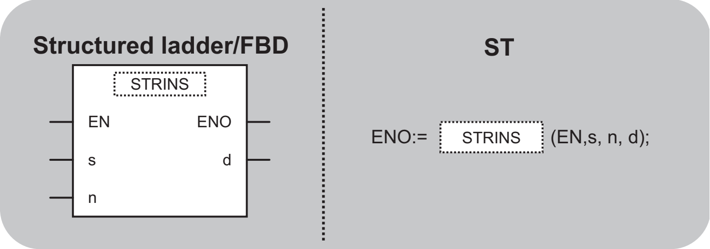
"1234AB56AB"

Character string data insert

STRINS(P)

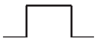
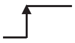


- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported



Any of the following instruction can go in the dotted squares.
STRINS, STRINSP

■Executing condition

Instruction	Executing condition
STRINS	
STRINSP	

■Argument

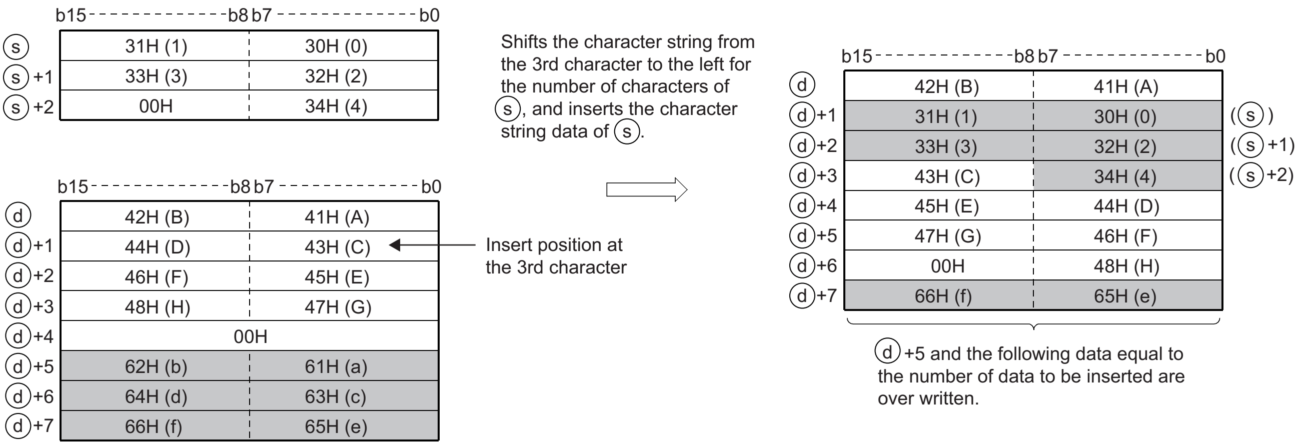
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Character string data to be inserted, or start number of the device in which character string data are stored	String
	n	Insert position (Setting range 1 ≤ n ≤ 16383)	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores character string data	String

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant		Others
	Bit	Word		Bit	Word			K, H	$	
(s)	—	○		—				—	○	—
n	—	○		○				○	—	—
(d)	—	○		—				—	—	—

Processing details

- Inserts the character string data specified for (s) to the nth character (insert position) from the start of the character string data specified for (d).

When the insert position n = 3



	b15 - - - - - - - - - - - - b8	b7 - - - - - - - - - - - - b0
ⓢ	31H (1)	30H (0)
ⓢ+1	33H (3)	32H (2)
ⓢ+2	00H	34H (4)

	b15 - - - - - - - - - - - - b8	b7 - - - - - - - - - - - - b0
ⓓ	42H (B)	41H (A)
ⓓ+1	44H (D)	43H (C)
ⓓ+2	46H (F)	45H (E)
ⓓ+3	48H (H)	47H (G)
ⓓ+4	00H	
ⓓ+5	62H (b)	61H (a)
ⓓ+6	64H (d)	63H (c)
ⓓ+7	66H (f)	65H (e)

Insert position at the 3rd character

Shifts the character string from the 3rd character to the left for the number of characters of ⓢ, and inserts the character string data of ⓢ.

	b15 - - - - - - - - - - - - b8	b7 - - - - - - - - - - - - b0	
ⓓ	42H (B)	41H (A)	
ⓓ+1	31H (1)	30H (0)	(ⓢ)
ⓓ+2	33H (3)	32H (2)	(ⓢ+1)
ⓓ+3	43H (C)	34H (4)	(ⓢ+2)
ⓓ+4	45H (E)	44H (D)	
ⓓ+5	47H (G)	46H (F)	
ⓓ+6	00H	48H (H)	
ⓓ+7	66H (f)	65H (e)	

ⓓ+5 and the following data equal to the number of data to be inserted are over written.

- If the number of characters inserted ((s) + (d)) is an even number, the NULL code (00H) is stored to the device (1 word) after the last device of the character string data.
- If the number of characters inserted ((s) + (d)) is an odd number, the NULL code (00H) is stored to the last device (upper 8 bits) of the character string data.
- If the number of characters of (d) + 1 is specified for n, the character string (s) is connected to the last of the character string (d).

Operation error

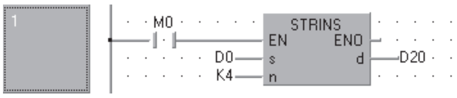
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The number of characters of the character string (s), the character string (d) or the inserted character string ((s) + (d)) exceeds 16383. n is not within the range. (1≤n≤16383) The value specified for n exceeds the number of characters of the character string (d) + 1.	—	—	—	—	○	○
4101	Any device of character string (s) and the character string (d) overlaps. The inserted character string data ((s) + (d)) exceed the range of the specified device. The NULL code (00H) does not exist within the specified device range following the devices specified for (s) and (d). The inserted character string data ((s) + (d)) overlap with the device that stores the character string (s).	—	—	—	—	○	○

Program example

- In the following program, the character string data stored in D0 and the following devices are inserted to the 4th character from the start of the character string data of device D20, when M0 turns ON.

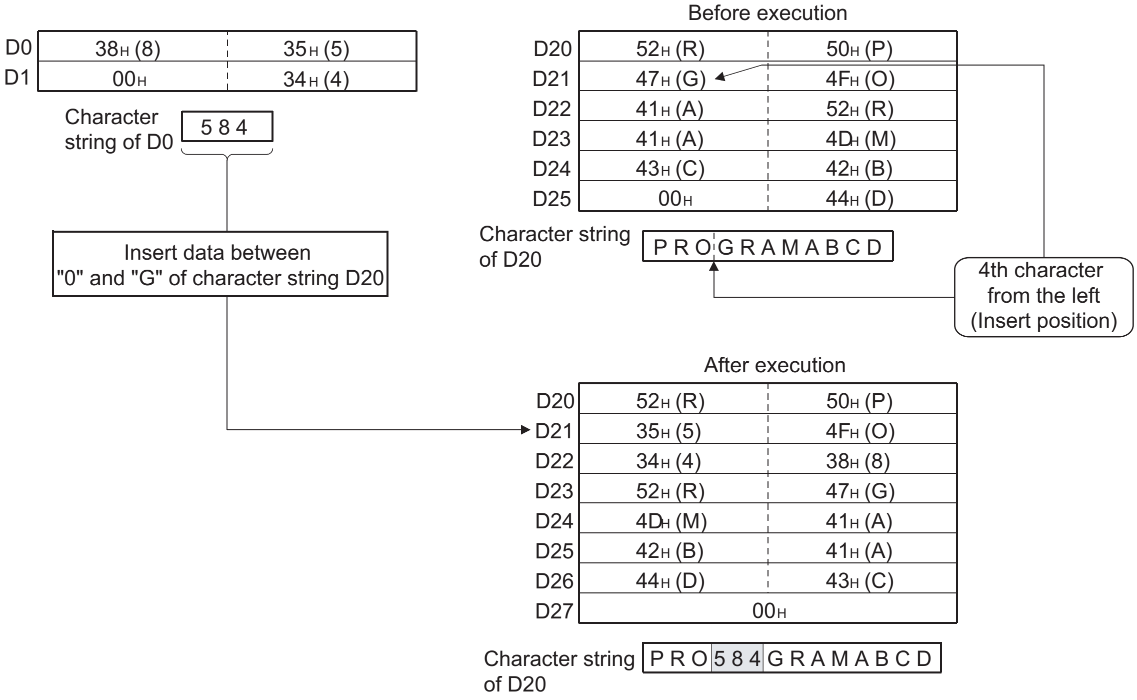
[Structured ladder/FBD]



[ST]

STRINS(M0,D0,K4,D20);

[Operation]



D0	38H (8)	35H (5)
D1	00H	34H (4)

Character string of D0: 5 8 4

Insert data between "0" and "G" of character string D20

Before execution

D20	52H (R)	50H (P)
D21	47H (G)	4FH (O)
D22	41H (A)	52H (R)
D23	41H (A)	4DH (M)
D24	43H (C)	42H (B)
D25	00H	44H (D)

Character string of D20: P R O G R A M A B C D

4th character from the left (Insert position)

After execution

D20	52H (R)	50H (P)
D21	35H (5)	4FH (O)
D22	34H (4)	38H (8)
D23	52H (R)	47H (G)
D24	4DH (M)	41H (A)
D25	42H (B)	41H (A)
D26	44H (D)	43H (C)
D27	00H	

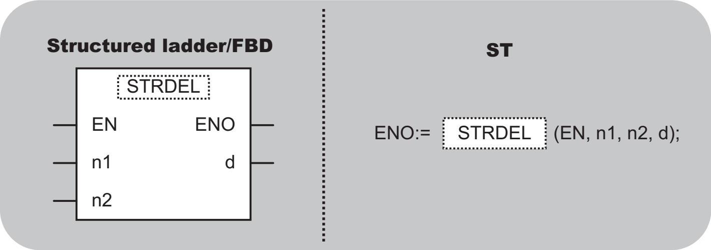
Character string of D20: P R O 5 8 4 G R A M A B C D

Character string data delete

STRDEL(P)

Basic High performance Process Redundant Ver. Universal LCPU

- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported

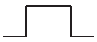
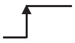


```
ENO:= STRDEL (EN, n1, n2, d);
```

Any of the following instruction can go in the dotted squares.

STRDEL, STRDELP

■Executing condition

Instruction	Executing condition
STRDEL	
STRDELP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Deletion start position (Setting range 1 ≤ n ≤ 16383)	ANY16
	n2	Number of characters to be deleted (Setting range 0 ≤ n2 ≤ 16384 - n1)	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores character string data to be deleted	String

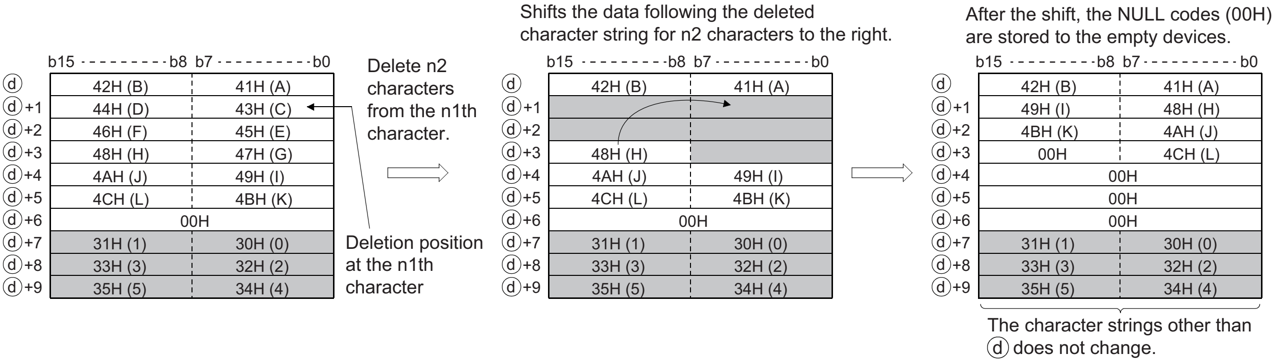
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n1	—	○		○				○	—
n2	—	○		○				○	—
(d)	—	○		—				—	—

Processing details

- Deletes n2 characters from n1th character (deletion start position) from the start of the character string data specified for (d).

Device position where character string data to be deleted: n1=3

Number of characters to be deleted: n2=5



- After the deletion, if the character string (d) is an even number, the NULL code (00H) is stored to the device (1 word) after the last device of the character string data.
- After the deletion, if the character string (d) is an odd number, the NULL code (00H) is stored to the last device (upper 8 bits) of the character string data.
- After shifting the data following the deleted character string for n2 characters to the right, the NULL codes (00H) are stored to the empty devices.

Operation error

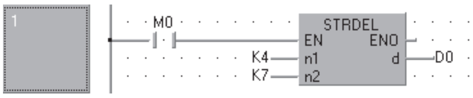
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The number of characters of the character string (d) exceeds 16383. n1 is not within the range. (1≤n1≤16383) The value specified for n1 exceeds the number of characters of the character string (d). The value specified for n2 exceeds the number of characters from n1 to the last character of the character string (d). A negative value is specified for n2.	—	—	—	—	○	○

Program example

- In the following program, the character string data stored in D0 and the following devices are deleted from the 4th character for 7 characters, when M0 turns ON.

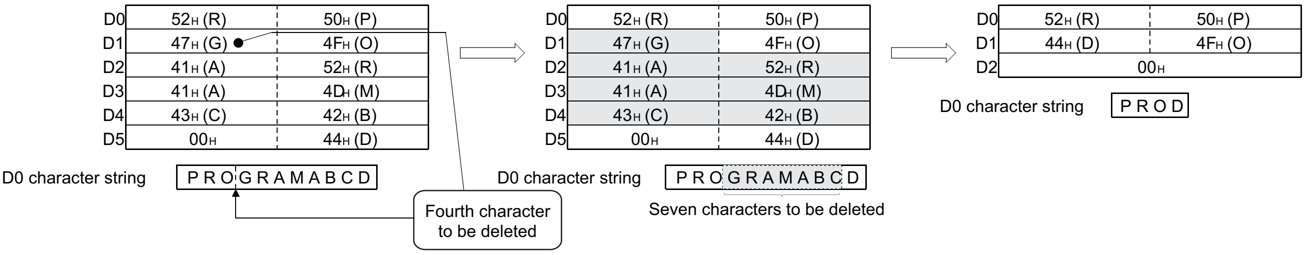
[Structured ladder/FBD]



[ST]

STRDEL(M0,K4,K7,D0);

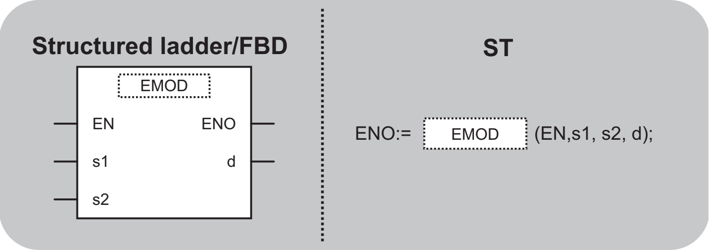
[Operation]



Floating-point data to BCD format conversion

EMOD(P)

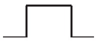
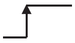
~~Basic~~ High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

EMOD, EMODP

■Executing condition

Instruction	Executing condition
EMOD	
EMODP	

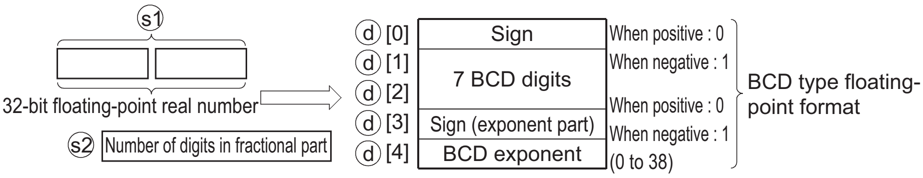
■Argument

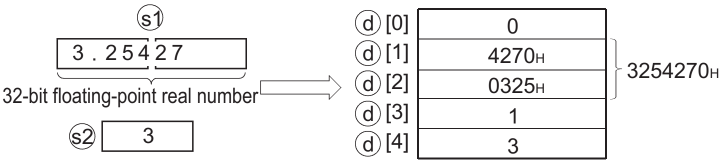
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	32-bit floating-point real number data, or start number of the device that stores 32-bit floating-point real number data	Single-precision real
	s2	Fractional part data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores data converted to BCD format	Array of ANY16 (1..5)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant		Others
	Bit	Word		Bit	Word			K, H	E	
(s1)	—	○		—	○		—		○	—
(s2)	○	○		○	○		○		—	—
(d)	—	○		—	—		—		—	—

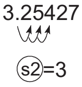
Processing details

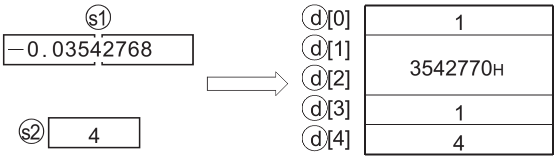
- Converts the 32-bit floating-point real number data specified for (s1) to 32-bit floating-point data in BCD format based on the fractional part digits specified for (s2), and stores the result to the device specified for (d) and the following devices.

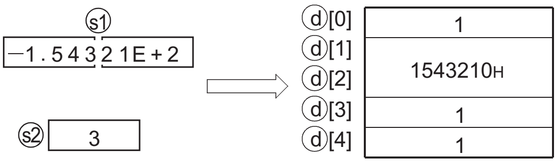




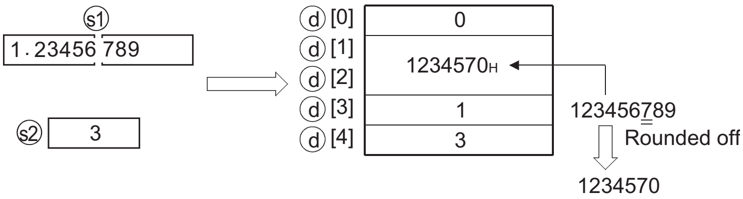
- (s2) specifies the fractional part digits of the 32-bit floating-point real number data (s1) as shown below. Values between 0 and 7 can be specified for (s2). In the example above, a decimal fraction digit is designated as shown below:







- The 7th digit of the significant figure of BCD stored in (d)[1] and (d)[2] is rounded off to make a 6-digit number.



- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

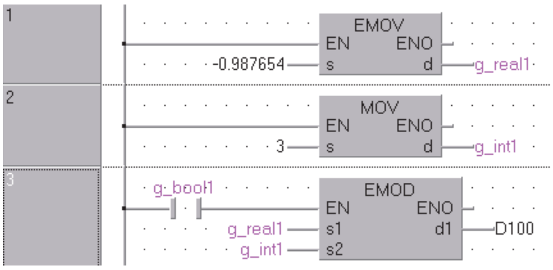
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The fractional part digits specified for (s2) is outside the range of 0 to 7. The 32-bit floating-point real number specified for (s1) is outside the following range. 0, $2^{-126} \leq$ | value of the specified device | $< 2^{128}$	—	○	○	○	○	○
4101	The device range specified for (d) exceeds the range of the corresponding device.	—	○	○	○	○	○
	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○
4140	The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○

Program example

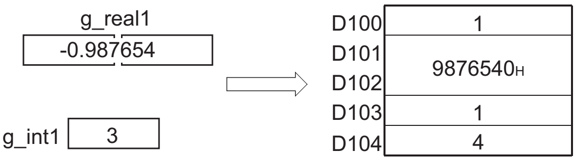
- In the following program, the 32-bit floating-point real number data in g_real1 is converted to BCD format according to the fractional part digits as specified for g_int1 when g_bool1 turns ON, and stores the results to D100 and the following devices.

[Structured ladder/FBD]



[ST]
g_real1:=-0.987654;
g_int1:=3;
EMOD(g_bool1,g_real1,g_int1,D100);

[Operation]

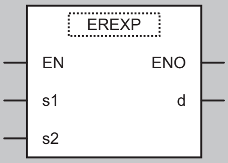


BCD format to floating-point data conversion

EREXP(P)

~~Basic~~ High performance Process Redundant Universal LCPU

Structured ladder/FBD



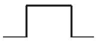
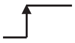
ST

ENO:= EREXP (EN,s1, s2, d);

Any of the following instruction can go in the dotted squares.

EREXP, EREXPP

■Executing condition

Instruction	Executing condition
EREXP	
EREXPP	

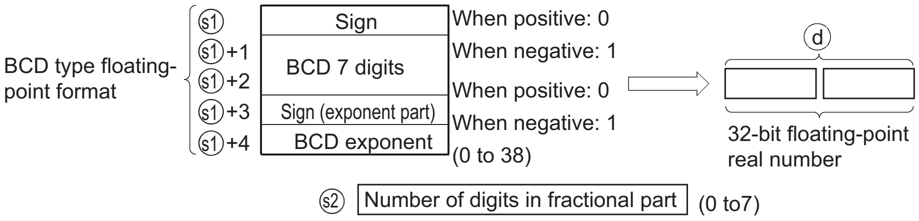
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores floating-point data in BCD format	ANY16
	s2	Fractional part data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores 32-bit floating-point real number data	Single-precision real

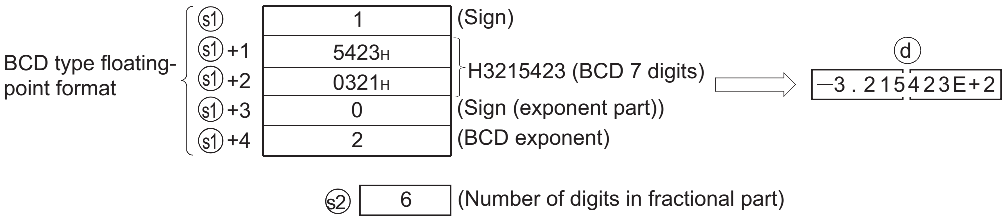
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—	—		—		—
(s2)	○	○		○	○		○		—
(d)	—	○		—	○		—		—

Processing details

- Converts the 32-bit floating-point data in BCD format specified for (s1) to the floating-point real number data according to the fractional part digits specified for (s2), and stores the result to the device specified for (d) and the following devices.



- The sign in (s1) and the sign for the exponent part in (s1)+3 is set to 0 for a positive value and to 1 for a negative value.
- 0 to 38 can be set for the BCD exponent of (s1)+4.
- 0 to 7 can be set for the fractional part digits of (s2).



Operation error

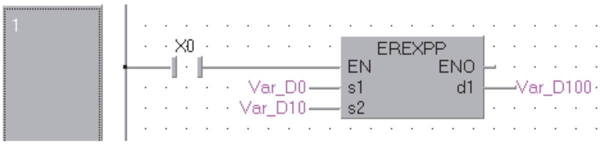
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The format specified for (s1) is neither 0 nor 1. Values other than 0 to 9 are specified for each digit of (s1)+1 and (s1)+2. The format specification specified for (s1) +3 is other than 0 or 1. The exponent data specified for (s1)+4 is outside the range of 0 to 38. The fractional part digits specified for (s2) is outside the range of 0 to 7.	—	○	○	○	○	○
4101	The device specified for (s1) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the 32-bit floating-point data in BCD format stored in Var_D0 and the following devices are converted to the floating-point real number data based on the fractional part digits stored in Var_D10 when X0 turns ON, and the result is stored to Var_D100.

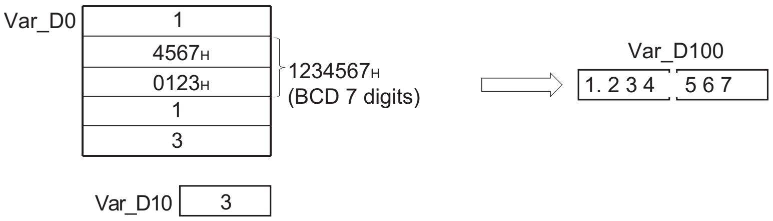
[Structured ladder/FBD]



[ST]

```
EREXPP(X0,Var_D0,Var_D10,Var_D100);
```

[Operation]



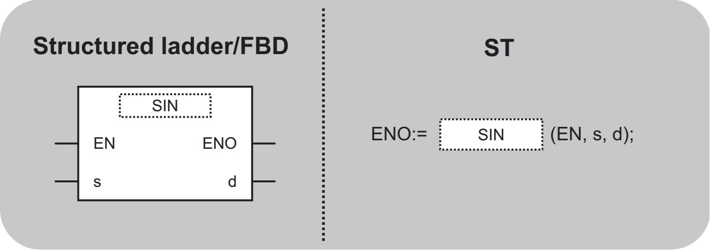
7.12 Special Function Instructions

SIN operation on floating-point data (single precision)

SIN(P)

Ver. Basic | High performance | Process | Redundant | Universal | LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

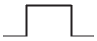
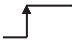


ENO:= SIN (EN, s, d);

Any of the following instruction can go in the dotted squares.

SIN, SINP

■Executing condition

Instruction	Executing condition
SIN	
SINP	

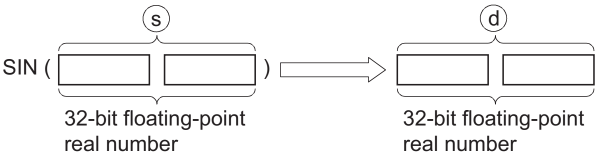
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Angle data on which the sine function is evaluated, or start number of the device that stores angle data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Evaluates the sine function on the angle specified for (s), and stores the result to the device specified for (d).


s
d
SIN (
)
32-bit floating-point real number
32-bit floating-point real number

- The angle specified for (s) is set in radians (degrees × π ÷ 180). For conversion between degrees and radians, refer to the RAD(P) and DEG(P) instructions.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

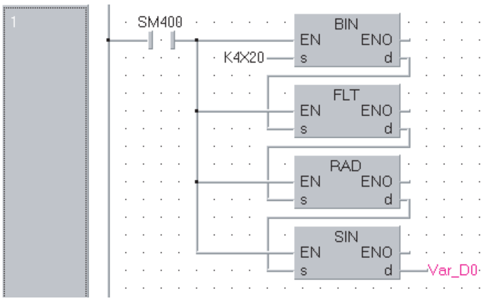
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of the specified device is -0.[*1]	○	○	○	○	—	—
4140	The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{128} \leq$ | operation result |	—	—	—	—	○	○

[*1] There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Program example

- In the following program, the sine function is evaluated on the angle set in BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 32-bit floating-point real number.

[Structured ladder/FBD]



Inputs the angle data on which the sine function is evaluated. (①)

Converts the input angle data to 32-bit floating-point real number data. (②)

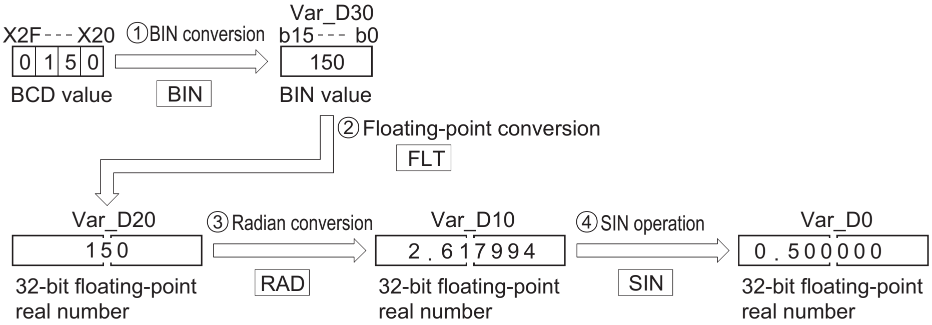
Converts degrees to radians. (③)

Sine operation on the converted angle data in radians. (④)

[ST]

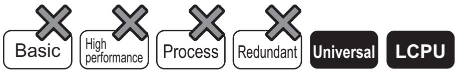
```
BIN(SM400,K4X20,Var_D30);
FLT(SM400,Var_D30,Var_D20);
RAD(SM400,Var_D20,Var_D10);
SIN(SM400,Var_D10,Var_D0);
```

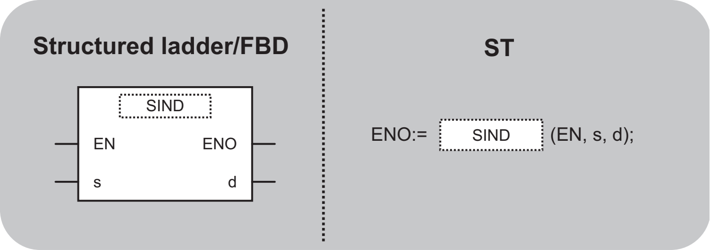
[Operation example when the value of 150 is specified for X20 to X2F]



SIN operation on floating-point data (double precision)

SIND(P)

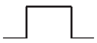
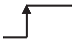




Any of the following instruction can go in the dotted squares.

SIND, SINDP

■Executing condition

Instruction	Executing condition
SIND	
SINDP	

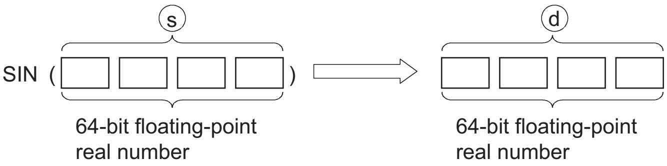
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Angle data on which the sine function is evaluated, or start number of the device that stores angle data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Evaluates the sine function on the angle specified for (s), and stores the result to the device specified for (d).


s
d
SIN (
)
64-bit floating-point real number
64-bit floating-point real number

- The angle specified for (s) is set in radians (degrees × π ÷ 180). For conversion between degrees and radians, refer to the RADD(P) and DEGD(P) instructions.
- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

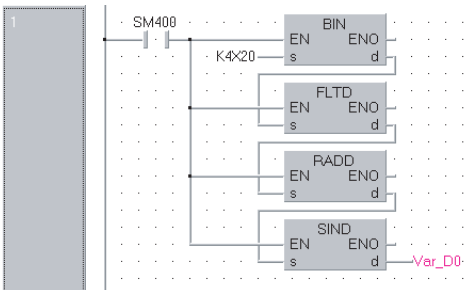
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | $< 2^{1024}$ The value of the specified device is -0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the sine function is evaluated on the angle set in BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 64-bit floating-point real number.

[Structured ladder/FBD]



Inputs the angle data on which the sine function is evaluated. (①)

Converts the input angle data to 64-bit floating-point real number data. (②)

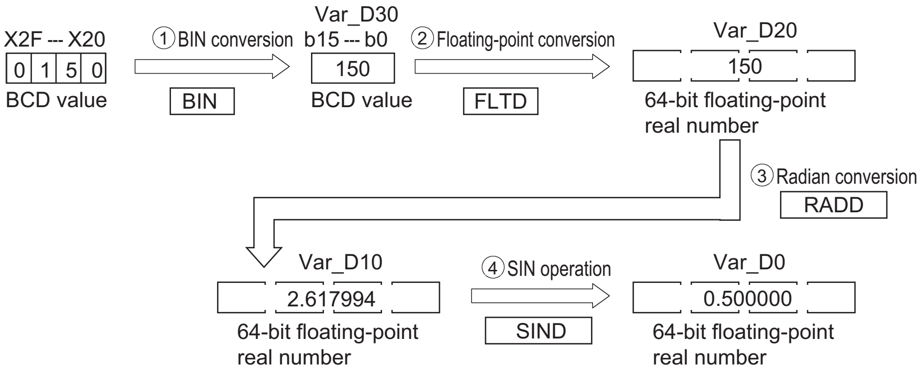
Converts degrees to radians. (③)

Sine operation on the converted angle data in radians. (④)

[ST]

```
    BIN(SM400,K4X20,Var_D30);
    FLTD(SM400,Var_D30,Var_D20);
    RADD(SM400,Var_D20,Var_D10);
    SIND(SM400,Var_D10,Var_D0);
```

[Operation example when the value of 150 is specified for X20 to X2F]

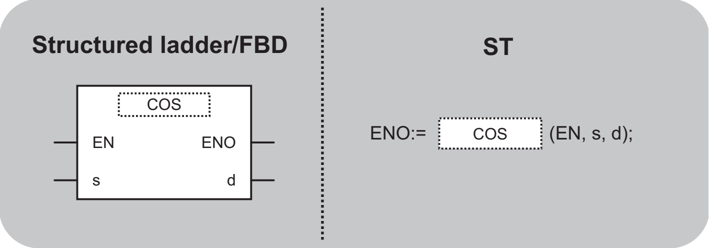


COS operation on floating-point data (single precision)

COS(P)

Ver. Basic High performance Process Redundant Universal LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

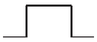
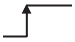


```
ENO:= COS (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

COS, COSP

■Executing condition

Instruction	Executing condition
COS	
COSP	

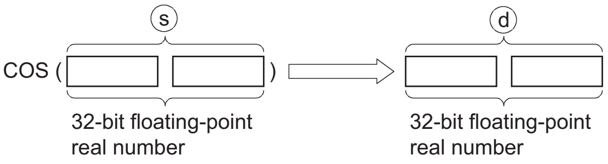
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Angle data on which the cosine function is evaluated, or start number of the device that stores angle data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Evaluates the cosine function on the angle specified for (s), and stores the result to the device specified for (d).


s
d
COS (
)
32-bit floating-point real number
32-bit floating-point real number

- The angle specified for (s) is set in radians (degrees × π ÷ 180). For conversion between degrees and radians, refer to the RAD(P) and DEG(P) instructions.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

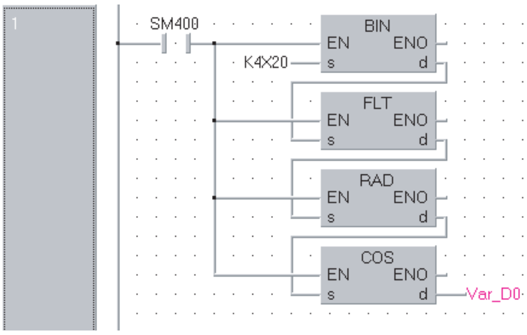
Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of the specified device is -0.[*1]	○	○	○	○	—	—
4140	The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{128} \leq$ | operation result |	—	—	—	—	○	○

[*1] There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Program example

- In the following program, the cosine function is evaluated on the angle set in BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 32-bit floating-point real number.

[Structured ladder/FBD]



Inputs the angle data on which the cosine function is evaluated. (①)

Converts the input angle data to 32-bit floating-point real number data. (②)

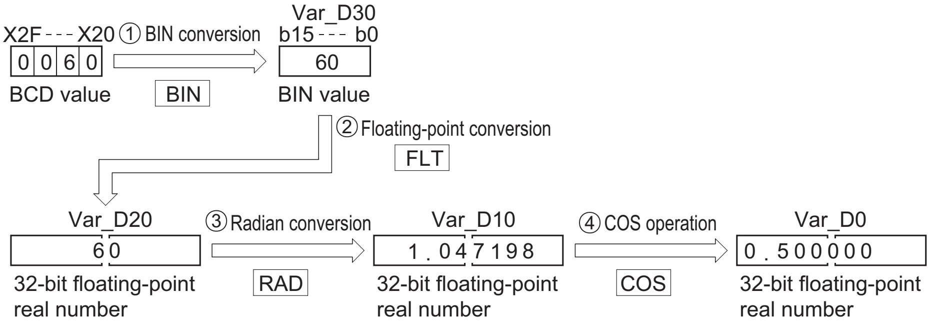
Converts degrees to radians. (③)

Cosine operation on the converted angle data in radians. (④)

[ST]

```
BIN(SM400,K4X20,Var_D30);
FLT(SM400,Var_D30,Var_D20);
RAD(SM400,Var_D20,Var_D10);
COS(SM400,Var_D10,Var_D0);
```

[Operation example when the value of 60 is specified for X20 to X2F]

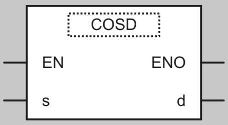


COS operation on floating-point data (double precision)

COSD(P)

✗ Basic ✗ High performance ✗ Process ✗ Redundant Universal LCPU

Structured ladder/FBD



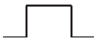
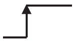
ST

ENO:= COSD (EN, s, d);

Any of the following instruction can go in the dotted squares.

COSD, COSDP

■Executing condition

Instruction	Executing condition
COSD	
COSDP	

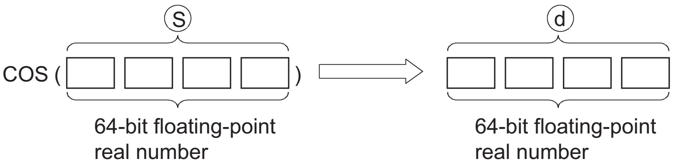
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Angle data on which the cosine function is evaluated, or start number of the device that stores angle data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Evaluates the cosine function on the angle specified for (s), and stores the result to the device specified for (d).


s
d
COS (
)
64-bit floating-point real number
64-bit floating-point real number

- The angle specified for (s) is set in radians (degrees × π ÷ 180). For conversion between degrees and radians, refer to the RADD(P) and DEGD(P) instructions.
- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

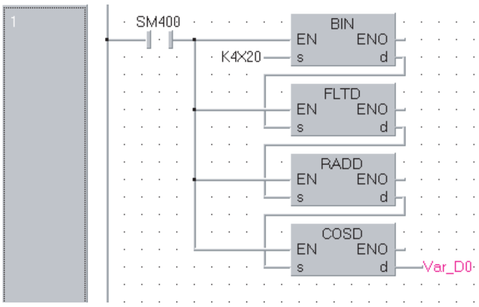
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | $< 2^{1024}$ The value of the specified device is -0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the cosine function is evaluated on the angle set in BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 64-bit floating-point real number.

[Structured ladder/FBD]



Inputs the angle data on which the cosine function is evaluated. (①)

Converts the input angle data to 64-bit floating-point real number data. (②)

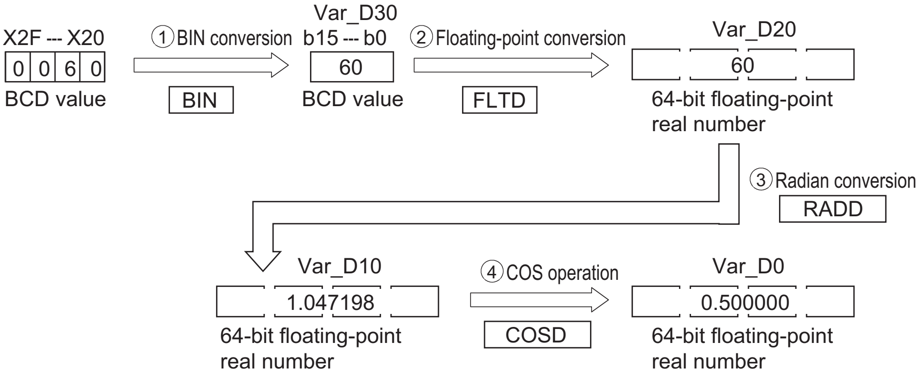
Converts degrees to radians. (③)

Cosine operation on the converted angle data in radians. (④)

[ST]

```
BIN(SM400,K4X20,Var_D30);
FLTD(SM400,Var_D30,Var_D20);
RADD(SM400,Var_D20,Var_D10);
COSD(SM400,Var_D10,Var_D0);
```

[Operation example when the value of 60 is specified for X20 to X2F]

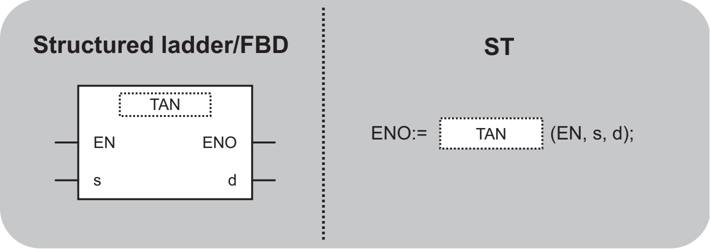


TAN operation on floating-point data (single precision)

TAN(P)

Ver. Basic | High performance | Process | Redundant | Universal | LCPU

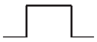
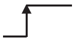
- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later



Any of the following instruction can go in the dotted squares.

TAN, TANP

■Executing condition

Instruction	Executing condition
TAN	
TANP	

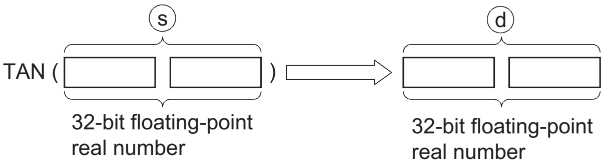
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Angle data on which the tangent function is evaluated, or start number of the device that stores angle data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Evaluates the tangent function on the angle specified for (s), and stores the result to the device specified for (d).


TAN (
)
s
d
32-bit floating-point real number
32-bit floating-point real number

- The angle specified for (s) is set in radians (degrees × π ÷ 180). For conversion between degrees and radians, refer to the RAD(P) and DEG(P) instructions.
- When the angle specified for (s) is π/2 radians, or (3/2)π radians, evaluation differences are generated for the radian value, but it does not cause an operation error.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

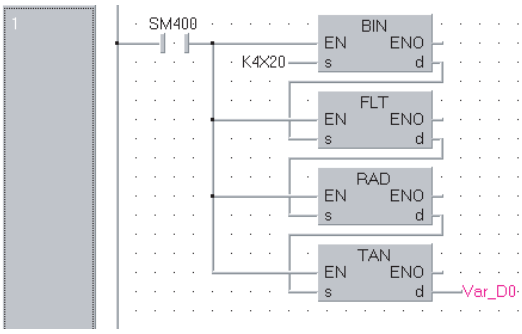
Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of the specified device is outside the following range. 0, $2^{-126} \leq$ |operation result| $<2^{128}$ The value of the specified device is -0.[*1]	○	○	○	○	—	—
4140	The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{128} \leq$| operation result |	—	—	—	—	○	○

[*1] There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Program example

- In the following program, the tangent function is evaluated on the angle set in BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 32-bit floating-point real number.

[Structured ladder/FBD]



Inputs the angle data on which the tangent function is evaluated. (①)

Converts the input angle data to 32-bit floating-point real number data. (②)

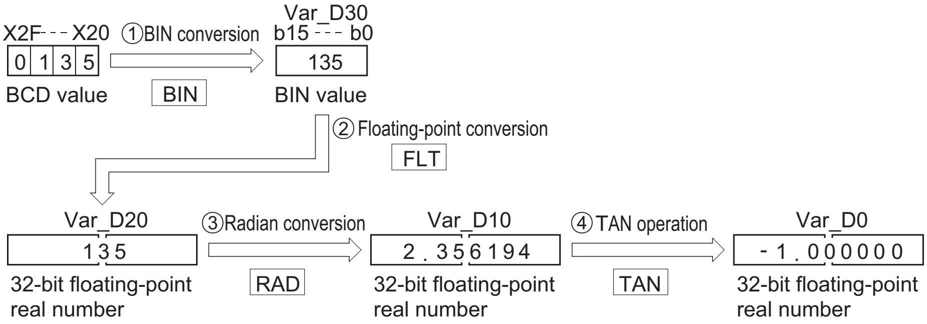
Converts degrees to radians. (③)

Tangent operation on the converted angle data in radians. (④)

[ST]

```
BIN(SM400,K4X20,Var_D30);
FLT(SM400,Var_D30,Var_D20);
RAD(SM400,Var_D20,Var_D10);
TAN(SM400,Var_D10,Var_D0);
```

[Operation example when the value of 135 is specified for X20 to X2F]

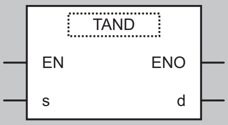


TAN operation on floating-point data (double precision)

TAND(P)

✕ Basic ✕ High performance ✕ Process ✕ Redundant Universal LCPU

Structured ladder/FBD



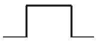
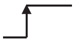
ST

ENO:= TAND (EN, s, d);

Any of the following instruction can go in the dotted squares.

TAND, TANDP

■Executing condition

Instruction	Executing condition
TAND	
TANDP	

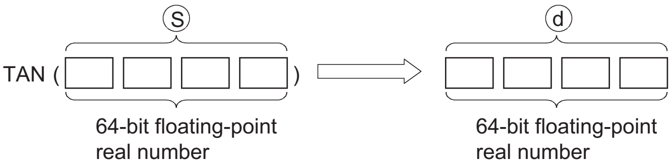
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Angle data on which the tangent function is evaluated, or start number of the device that stores angle data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Evaluates the tangent function on the angle specified for (s), and stores the result to the device specified for (d).


Ⓢ
Ⓓ
TAN (
)
64-bit floating-point real number
64-bit floating-point real number

- The angle specified for (s) is set in radians (degrees × π ÷ 180). For conversion between degrees and radians, refer to the RADD(P) and DEGD(P) instructions.
- When the angle specified for (s) is π/2 radians, or (3/2)π radians, evaluation differences are generated for the radian value, but it does not cause an operation error.
- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

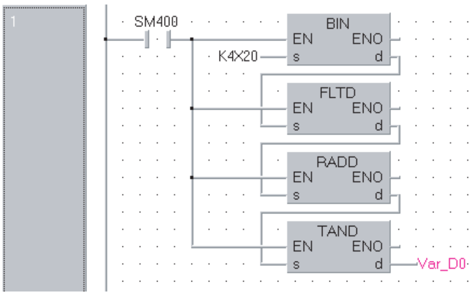
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified data | < 2^{1024} The value of the specified data is -0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the tangent function is evaluated on the angle set in BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 64-bit floating-point real number.

[Structured ladder/FBD]



Inputs the angle data on which the tangent function is evaluated. (①)

Converts the input angle data to 64-bit floating-point real number data. (②)

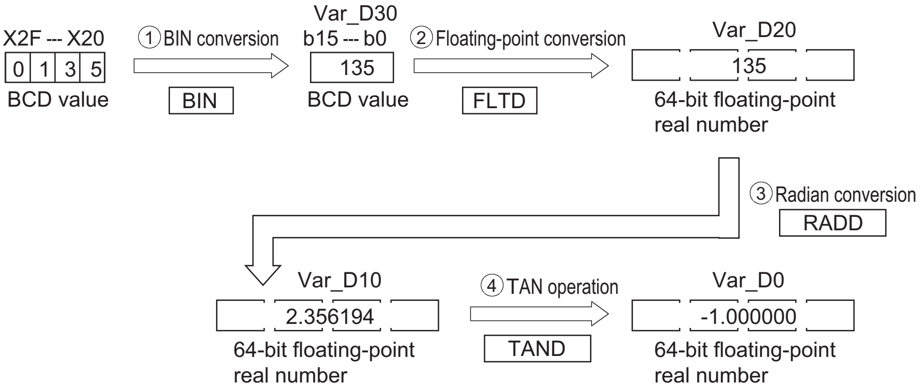
Converts degrees to radians. (③)

Tangent operation on the converted angle data in radians. (④)

[ST]

```
BIN(SM400,K4X20,Var_D30);
FLTD(SM400,Var_D30,Var_D20);
RADD(SM400,Var_D20,Var_D10);
TAND(SM400,Var_D10,Var_D0);
```

[Operation example when the value of 135 is specified for X20 to X2F]

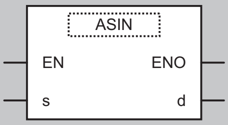


SIN^{-1} operation on floating-point data (single precision)

ASIN(P)

~~Basic~~ High performance Process Redundant Universal LCPU

Structured ladder/FBD



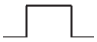
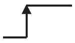
ST

ENO:= ASIN (EN, s, d);

Any of the following instruction can go in the dotted squares.

ASIN, ASINP

■Executing condition

Instruction	Executing condition
ASIN	
ASINP	

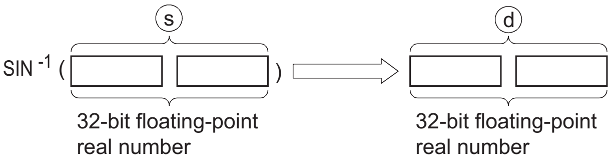
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Sine values on which the arcsine function is evaluated, or start number of the device that stores sine values	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Evaluates the arcsine function on the value specified for (s) to define an angle, and stores the result to the device specified for (d).


s
d
SIN -1 (
)
32-bit floating-point real number
32-bit floating-point real number

- The SIN value specified for (s) can be in the range from -1.0 to 1.0.
- The angle (operation result) stored to (d) is stored in radians. For more information on the conversion between radians and degrees, refer to the DEG(P) and RAD(P) instructions.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

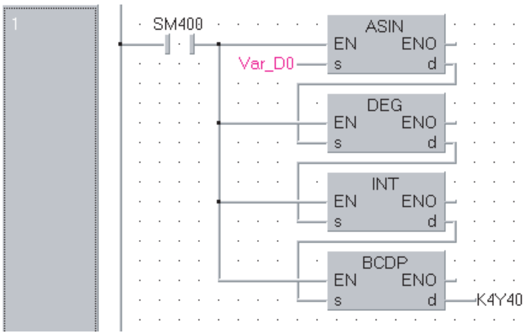
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is outside the range of -1.0 to 1.0.	—	○	○	○	○	○
	The value of the specified device is -0.[*1]	—	○	○	○	—	—
4140	The value of the specified device is outside the following range. 0, $2^{-126} \leq$ | value of the specified device | < 2^{128} The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{128} \leq$ | operation result |	—	—	—	—	○	○

*1 There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Program example

- In the following program, the arcsine function is evaluated on the 32-bit floating-point real number in Var_D0, and its angle is output to Y40 to Y4F in BCD 4 digits.

[Structured ladder/FBD]



Evaluates the angle data (in radians) by the arcsine operation. (①)

Converts radians to degrees. (②)

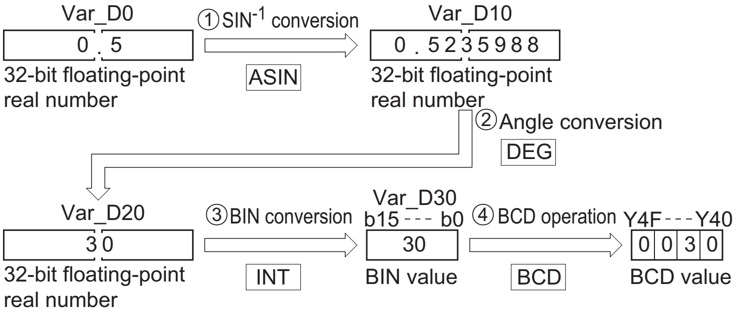
Converts the angle data in 32-bit floating-point real number to integer value. (③)

Outputs the angle data converted to integer value to the display. (④)

[ST]

```
ASIN(SM400,Var_D0,Var_D10);
DEG(SM400,Var_D10,Var_D20);
INT(SM400,Var_D20,Var_D30);
BCDP(SM400,Var_D30,K4Y40);
```

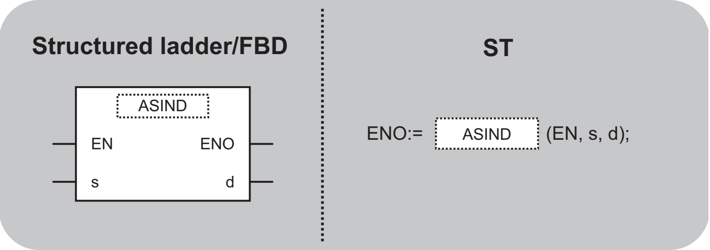
[Operation example when the value of 0.5 is set for Var_D0]



SIN^{-1} operation on floating-point data (double precision)

ASIND(P)

✕Basic ✕High performance ✕Process ✕Redundant Universal LCPU

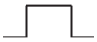
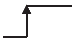


```
ENO:= ASIND (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

ASIND, ASINDP

■Executing condition

Instruction	Executing condition
ASIND	
ASINDP	

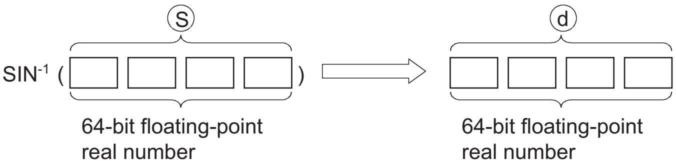
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Sine values on which the arcsine function is evaluated, or start number of the device that stores sine values	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Evaluates the arcsine function on the value specified for (s) to define an angle, and stores the result to the device specified for (d).


Ⓢ
ⓓ
SIN-1 (
)
64-bit floating-point real number
64-bit floating-point real number

- The SIN value specified for (s) can be in the range from -1.0 to 1.0.
- The angle (operation result) stored to (d) is stored in radians. For more information on the conversion between radians and degrees, refer to the DEGD(P) and RADD(P) instructions.
- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

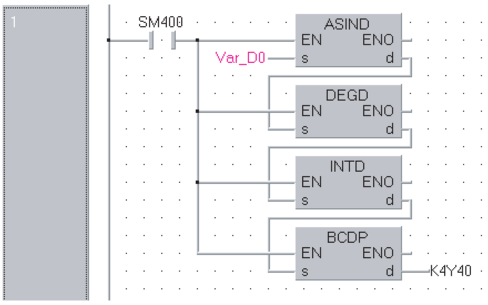
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is within the double-precision floating-point range and outside the range of -1.0 to 1.0.	—	—	—	—	○	○
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | < 2^{1024} The value of the specified device is -0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the arcsine function is evaluated on the 64-bit floating-point real number in Var_D0, and its angle is output to Y40 to Y4F in BCD 4 digits.

[Structured ladder/FBD]



Evaluates the angle data (in radians) by the arcsine operation. (①)

Converts radians to degrees. (②)

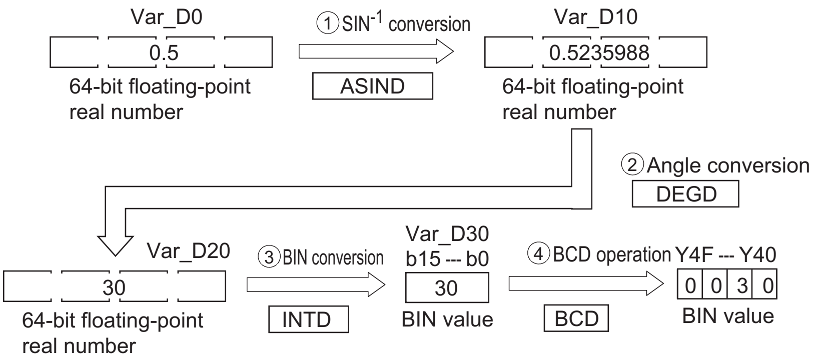
Converts the angle data in 64-bit floating-point real number to integer value. (③)

Outputs the angle data converted to integer value to the display. (④)

[ST]

```
ASIND(SM400,Var_D0,Var_D10);
DEGD(SM400,Var_D10,Var_D20);
INTD(SM400,Var_D20,Var_D30);
BCDP(SM400,Var_D30,K4Y40);
```

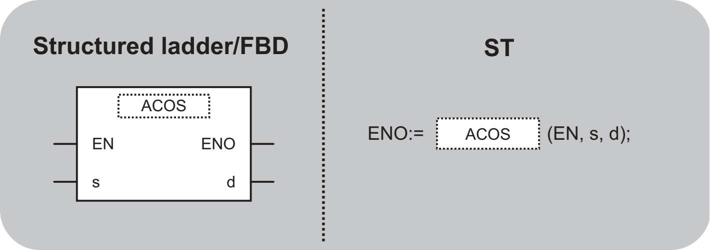
[Operation example when the value of 0.5 is set for Var_D0]



COS^{-1} operation on floating-point data (single precision)

ACOS(P)

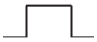
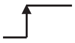
Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

ACOS, ACOSP

■Executing condition

Instruction	Executing condition
ACOS	
ACOSP	

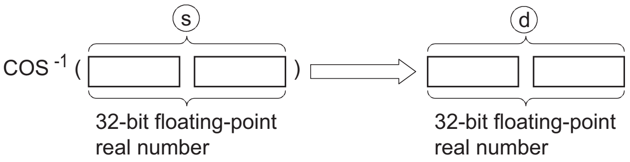
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Cosine values on which the arccosine function is evaluated, or start number of the device that stores cosine values	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Evaluates the arccosine function on the value specified for (s) to define an angle, and stores the result to the device specified for (d).


s
d
COS -1 (
)
32-bit floating-point real number
32-bit floating-point real number

- The COS value specified for (s) can be in the range from -1.0 to 1.0.
- The angle (operation result) stored to (d) is stored in radians. For more information on the conversion between radians and degrees, refer to the DEG(P) and RAD(P) instructions.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

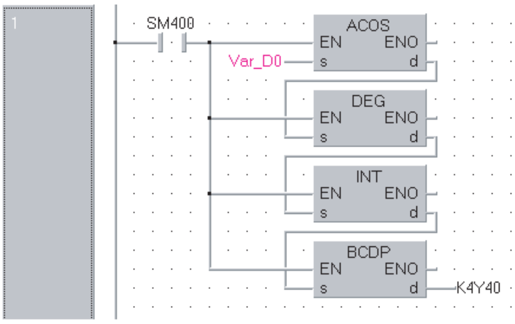
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is outside the range of -1.0 to 1.0.	—	○	○	○	○	○
	The value of the specified device is -0.[*1]	—	○	○	○	—	—
4140	The value of the specified device is outside the following range. 0, $2^{-126} \leq$ | value of the specified device | $< 2^{128}$ The specified device value is -0, unnormalized number, nonnumeric, and ±∞.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{128} \leq$ | operation result |	—	—	—	—	○	○

[*1] There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Program example

- In the following program, the arccosine function is evaluated on the 32-bit floating-point real number in Var_D0, and its angle is output to Y40 to Y4F in BCD 4 digits.

[Structured ladder/FBD]



Evaluates the angle data (in radians) by the arccosine operation. (①)

Converts radians to degrees. (②)

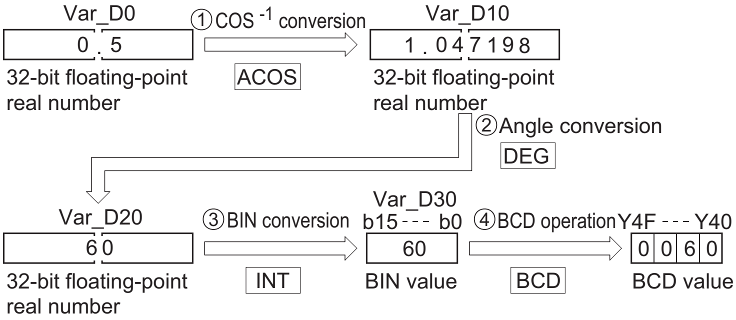
Converts the angle data in 32-bit floating-point real number to integer value. (③)

Outputs the angle data converted to integer value to the display. (④)

[ST]

```
ACOS(SM400,Var_D0,Var_D10);
DEG(SM400,Var_D10,Var_D20);
INT(SM400,Var_D20,Var_D30);
BCDP(SM400,Var_D30,K4Y40);
```

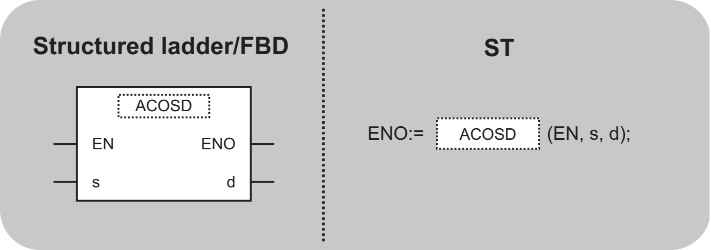
[Operation example when the value of 0.5 is set for Var_D0]



COS^{-1} operation on floating-point data (double precision)

ACOSD(P)

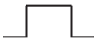
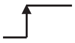
~~Basic~~ ~~High performance~~ ~~Process~~ ~~Redundant~~ Universal LCPU



Any of the following instruction can go in the dotted squares.

ACOSD, ACOSDP

■Executing condition

Instruction	Executing condition
ACOSD	
ACOSDP	

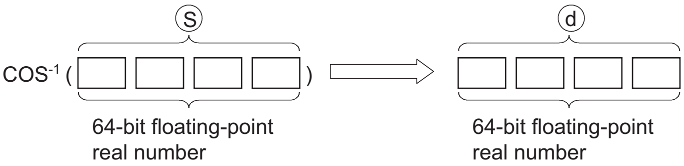
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Cosine values on which the arccosine function is evaluated, or start number of the device that stores cosine values	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Evaluates the arccosine function on the value specified for (s) to define an angle, and stores the result to the device specified for (d).


Ⓢ
ⓓ
COS-1 ( )
64-bit floating-point real number
64-bit floating-point real number

- The COS value specified for (s) can be in the range from -1.0 to 1.0.
- The angle (operation result) stored in (d) is stored in radians. For more information on the conversion between radians and degrees, refer to the DEGD(P) and RADD(P) instructions.
- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

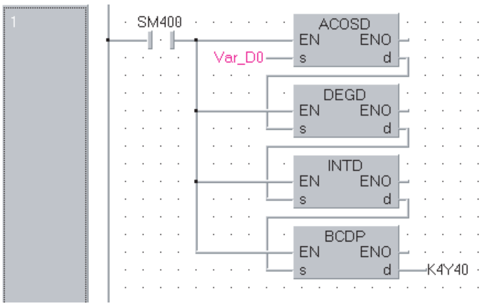
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is within the double-precision floating-point range and outside the range of -1.0 to 1.0.	—	—	—	—	○	○
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | < 2^{1024} The value of the specified device is -0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the arccosine function is evaluated on the 64-bit floating-point real number in Var_D0, and its angle is output to Y40 to Y4F in BCD 4 digits.

[Structured ladder/FBD]



Evaluates the angle data (in radians) by the arccosine operation. (①)

Converts radians to degrees. (②)

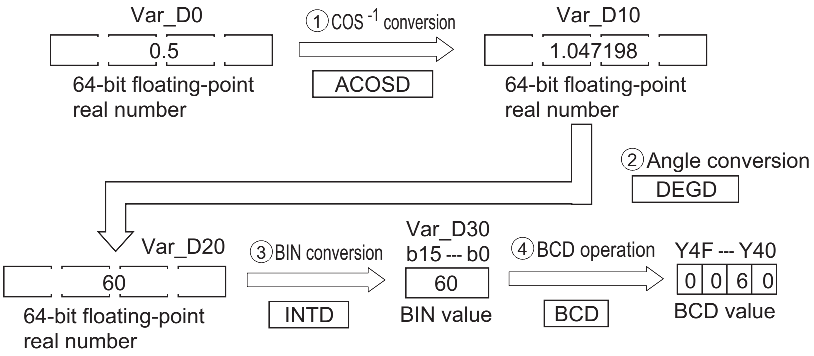
Converts the angle data in 64-bit floating-point real number to integer value. (③)

Outputs the angle data converted to integer value to the display. (④)

[ST]

```
ACOSD(SM400,Var_D0,Var_D10);
DEGD(SM400,Var_D10,Var_D20);
INTD(SM400,Var_D20,Var_D30);
BCDP(SM400,Var_D30,K4Y40);
```

[Operation example when the value of 0.5 is set for Var_D0]

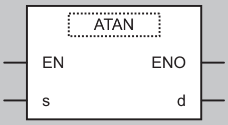


TAN^{-1} operation on floating-point data (single precision)

ATAN(P)

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



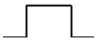
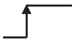
ST

```
ENO:= ATAN (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

ATAN, ATANP

■Executing condition

Instruction	Executing condition
ATAN	
ATANP	

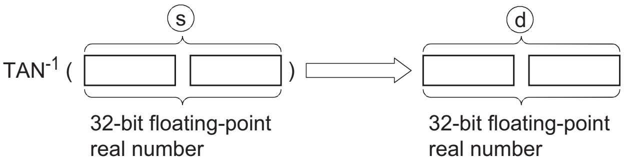
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Tangent values on which the arctangent function is evaluated, or start number of the device that stores tangent values	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Evaluates the arctangent function on the value specified for (s) to define an angle, and stores the result to the device specified for (d).


s
d
TAN-1 (
)
32-bit floating-point real number
32-bit floating-point real number

- The angle (operation result) stored to (d) is stored in radians. For more information on the conversion between radians and degrees, refer to the DEG(P) and RAD(P) instructions.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

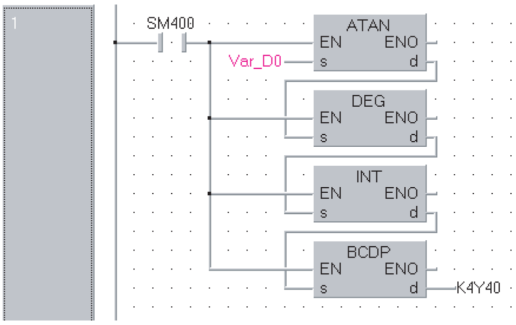
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of the specified device is -0.[*1]	—	○	○	○	—	—
4140	The value of the specified device is outside the following range. 0, $2^{-126} \leq$ | value of the specified device | $< 2^{128}$ The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{128} \leq$ | operation result |	—	—	—	—	○	○

[*1] There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Program example

- In the following program, the arctangent function is evaluated on the 32-bit floating-point real number in Var_D0, and its angle is output to Y40 to Y4F in BCD 4 digits.

[Structured ladder/FBD]



Evaluates the angle data (in radians) by the arctangent operation. (①)

Converts radians to degrees. (②)

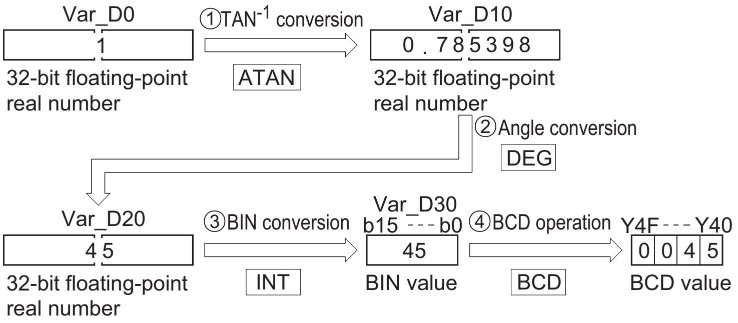
Converts the angle data in 32-bit floating-point real number to integer value. (③)

Outputs the angle data converted to integer value to the display. (④)

[ST]

```
ATAN(SM400,Var_D0,Var_D10);
DEG(SM400,Var_D10,Var_D20);
INT(SM400,Var_D20,Var_D30);
BCDP(SM400,Var_D30,K4Y40);
```

[Operation example when the value of 1 is set for Var_D0]

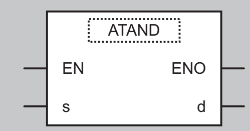


TAN^{-1} operation on floating-point data (double precision)

ATAND(P)

☒ Basic ☒ High performance ☒ Process ☒ Redundant **Universal** **LCPU**

Structured ladder/FBD



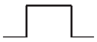
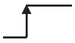
ST

ENO:= ATAND (EN, s, d);

Any of the following instruction can go in the dotted squares.

ATAND, ATANDP

■Executing condition

Instruction	Executing condition
ATAND	
ATANDP	

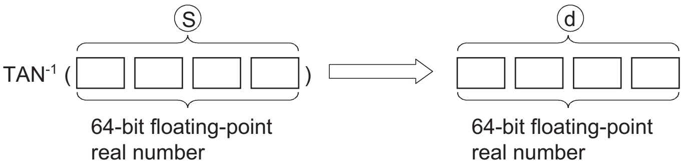
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Tangent values on which the arctangent function is evaluated, or start number of the device that stores tangent values	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Evaluates the arctangent function on the value specified for (s) to define an angle, and stores the result to the device specified for (d).


S
d
TAN-1 ( )
64-bit floating-point real number
64-bit floating-point real number

- The angle (operation result) stored to (d) is stored in radians. For more information on the conversion between radians and degree, refer to the DEGD(P) and RADD(P) instructions.
- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

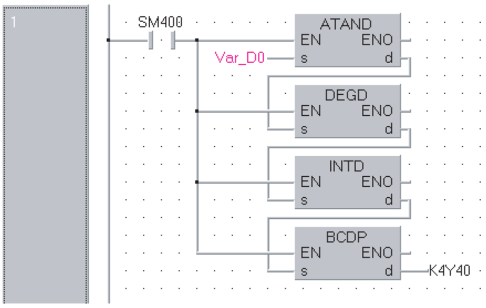
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | $< 2^{1024}$ The value of the specified device is -0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the arctangent function is evaluated on the 64-bit floating-point real number in Var_D0, and its angle is output to Y40 to Y4F in BCD 4 digits.

[Structured ladder/FBD]



Evaluates the angle data (in radians) by the arctangent operation. (①)

Converts radians to degrees. (②)

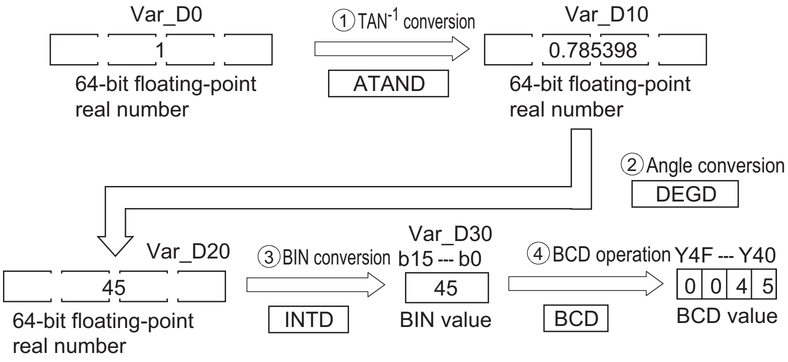
Converts the angle data in 64-bit floating-point real number to integer value. (③)

Outputs the angle data converted to integer value to the display. (④)

[ST]

```
ATAND(SM400,Var_D0,Var_D10);
DEGD(SM400,Var_D10,Var_D20);
INTD(SM400,Var_D20,Var_D30);
BCDP(SM400,Var_D30,K4Y40);
```

[Operation example when the value of 1 is set for Var_D0]



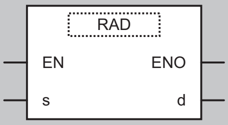
Degree to radian conversion on floating-point data (single precision)

RAD(P)

Ver.
Basic High performance Process Redundant Universal LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

Structured ladder/FBD



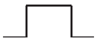
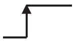
ST

ENO:= RAD (EN, s, d);

Any of the following instruction can go in the dotted squares.

RAD, RADP

■Executing condition

Instruction	Executing condition
RAD	
RADP	

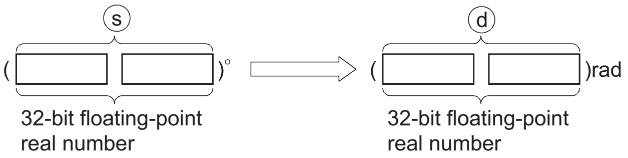
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Angle data whose unit is converted from degrees to radians, or start number of the device that stores angle data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted value in radians	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Converts the unit of angle from degrees specified for (s) to radians, and stores the result to the device specified for (d).


s
( )°
32-bit floating-point real number
d
( )rad
32-bit floating-point real number

- Conversion from degrees to radians is performed according to the following equation.

$$\text{Radians} = \text{Degrees} \times \frac{\pi}{180}$$

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

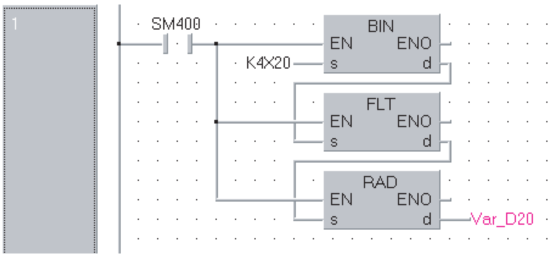
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of the specified device is -0.[*1]	○	○	○	○	—	—
4140	The value of the specified device is outside the following range. $0, 2^{-126} \leq \lvert \text{value of the specified device} \rvert < 2^{128}$ The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{128} \leq \lvert \text{operation result} \rvert$	—	—	—	—	○	○

[*1] There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Program example

- In the following program, the angle (in degrees) set in BCD 4 digits in the devices from X20 to X2F is converted to radians, and the result is stored to Var_D20 as a floating-point real number.

[Structured ladder/FBD]



Inputs the angle data to be converted to radians. (①)

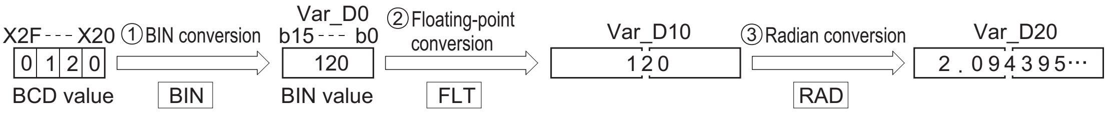
Converts the input angle data to 32-bit floating-point real number data. (②)

Converts degrees to radians. (③)

[ST]

```
BIN(SM400,K4X20,Var_D0);
FLT(SM400,Var_D0,Var_D10);
RAD(SM400,Var_D10,Var_D20);
```

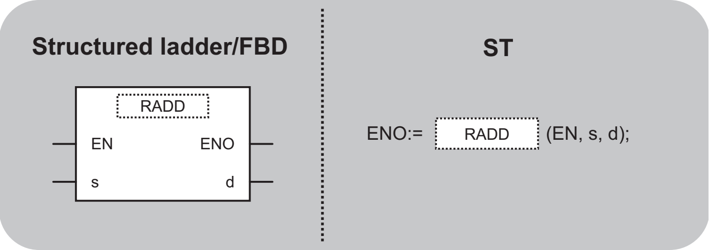
[Operation example when the value of 120 is specified for X20 to X2F]



Degree to radian conversion on floating-point data (double precision)

RADD(P)

☒ Basic ☒ High performance ☒ Process ☒ Redundant Universal LCPU

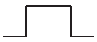
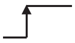


```
ENO:= RADD (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

RADD, RADDP

■Executing condition

Instruction	Executing condition
RADD	
RADDP	

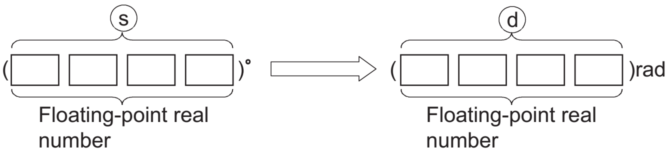
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Angle data whose unit is converted from degrees to radians, or start number of the device that stores angle data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted value in radians	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Converts the unit of angle from degrees specified for (s) to radians, and stores the result to the device specified for (d).



- Conversion from degrees to radians is performed according to the following equation.

$$\text{Radians} = \text{Degrees} \times \frac{\pi}{180}$$

- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

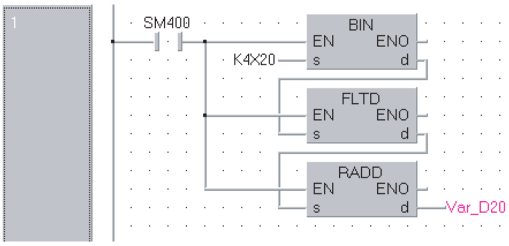
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | $< 2^{1024}$ The value of the specified device is -0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the angle (in degrees) set in BCD 4 digits in the devices from X20 to X2F is converted to radians, and the result is stored to Var_D20 as a 64-bit floating-point real number.

[Structured ladder/FBD]



Inputs the angle data to be converted to radians. (①)

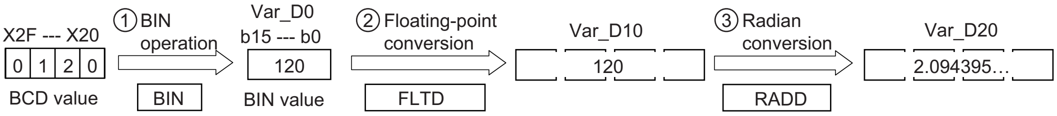
Converts the input angle data to 64-bit floating-point real number data. (②)

Converts degrees to radians. (③)

[ST]

```
BIN(SM400,K4X20,Var_D0);
FLTD(SM400,Var_D0,Var_D10);
RADD(SM400,Var_D10,Var_D20);
```

[Operation example when the value of 120 is specified for X20 to X2F]

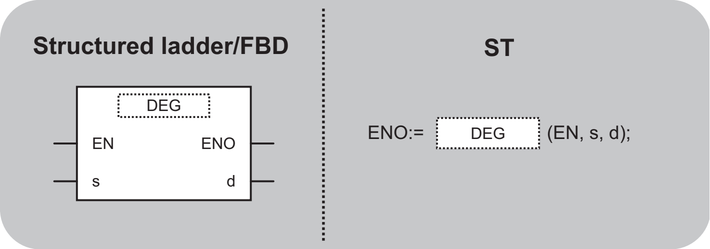


Radian to degree conversion on floating-point data (single precision)

DEG(P)

Ver. Basic | High performance | Process | Redundant | Universal | LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

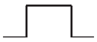
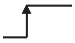


```
ENO:= DEG (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

DEG, DEGP

■Executing condition

Instruction	Executing condition
DEG	
DEGP	

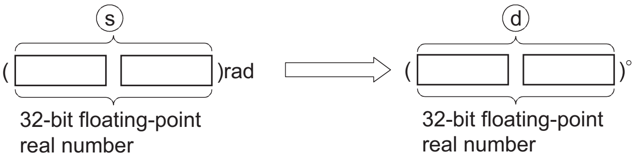
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Angle data whose unit is converted from radians to degrees, or start number of the device that stores radians angle	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted value in degrees	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Converts the unit of angle from radians specified for (s) to degrees, and stores the result to the device specified for (d).


s
d
( )rad
( )°
32-bit floating-point real number
32-bit floating-point real number

- The conversion from radians to degrees is performed according to the following equation.

$$\text{Degrees} = \text{Radians x } \frac{180}{\pi}$$

- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

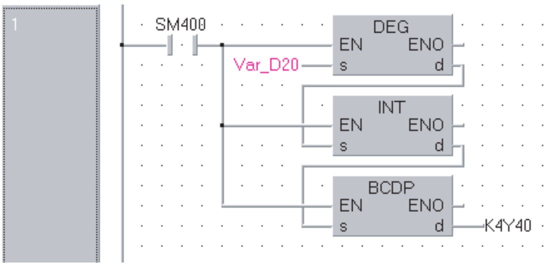
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value of the specified device is -0.[*1]	○	○	○	○	—	—
4140	The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{128} \leq$ | operation result |	—	—	—	—	○	○

[*1] There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Program example

- In the following program, the radian value set as a 32-bit floating-point real number in Var_D20 is converted to degrees, and the result is output to Y40 to Y4F as a BCD value.

[Structured ladder/FBD]



Converts radians to degrees. (①)

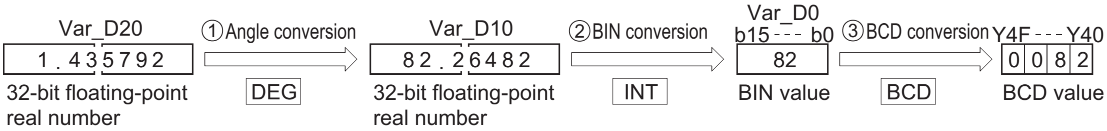
Converts the angle data in 32-bit floating-point real number to integer value. (②)

Outputs the angle data converted to integer value to the display. (③)

[ST]

```
DEG(SM400,Var_D20,Var_D10);
INT(SM400,Var_D10,Var_D0);
BCDP(SM400,Var_D0,K4Y40);
```

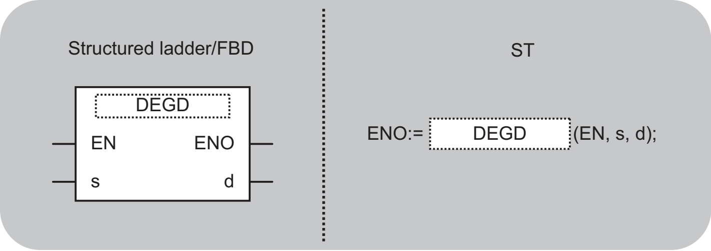
[Operation example when the value of 1.435792 is set for Var_D20]



Radian to degree conversion on floating-point data (double precision)

DEGD(P)

✕Basic ✕High performance ✕Process ✕Redundant Universal LCPU

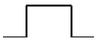
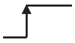


```
ENO:= DEGD (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

DEGD, DEGDP

■Executing condition

Instruction	Executing condition
DEGD	
DEGDP	

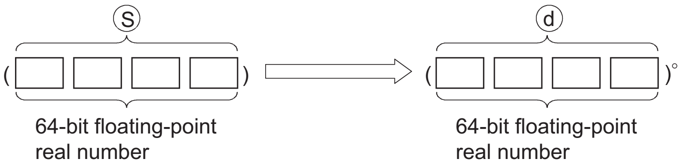
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Angle data whose unit is converted from radians to degrees, or start number of the device that stores radians angle	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores converted value in degrees	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Converts the unit of angle from radians specified for (s) to degrees, and stores the result to the device specified for (d).


s
d
64-bit floating-point real number
64-bit floating-point real number
°

- The conversion from radians to degrees is performed according to the following equation.

$$\text{Degrees} = \text{Radians} \times \frac{180}{\pi}$$

- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

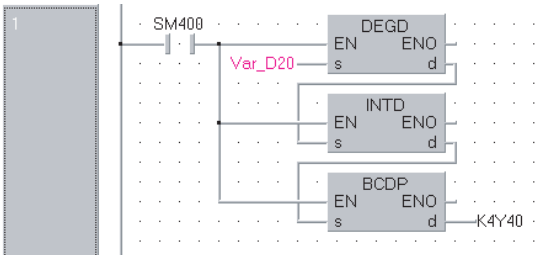
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | $< 2^{1024}$ The value of the specified device is -0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the radian value set as a 64-bit floating-point real number in Var_D20 is converted to degrees, and the result is output to Y40 to Y4F as a BCD value.

[Structured ladder/FBD]



Converts radians to degrees. (①)

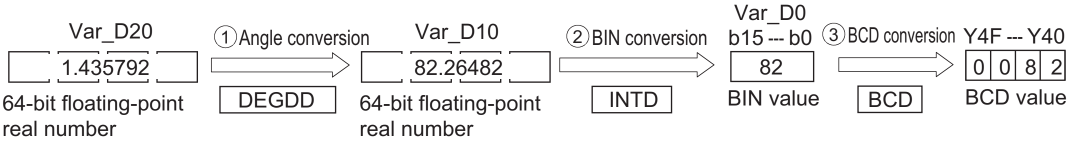
Converts the angle data in 64-bit floating-point real number to integer value. (②)

Outputs the angle data converted to integer value to the display. (③)

[ST]

```
DEGD(SM400,Var_D20,Var_D10);
INTD(SM400,Var_D10,Var_D0);
BCDP(SM400,Var_D0,K4Y40);
```

[Operation example when the value of 1.435792 is set for Var_D20]

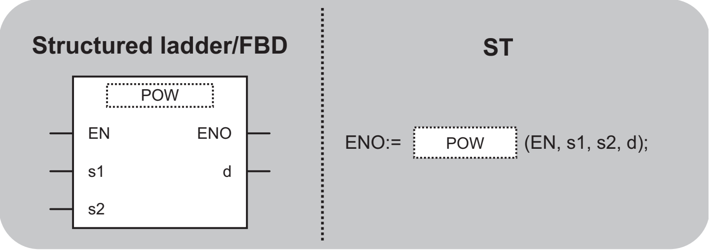


Exponentiation on floating-point data (single precision)

POW(P)

✕ Basic ✕ High performance ✕ Process ✕ Redundant Ver. Universal LCPU

- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported

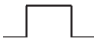
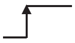


```
ENO:= POW (EN, s1, s2, d);
```

Any of the following instruction can go in the dotted squares.

POW, POWP

■Executing condition

Instruction	Executing condition
POW	
POWP	

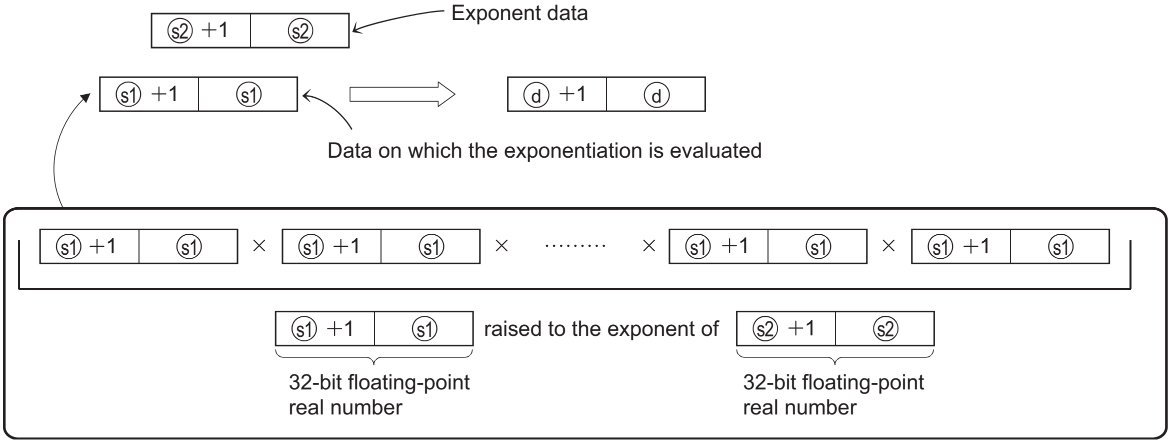
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Data on which the exponentiation is evaluated, or start number of the device that stores data on which the exponentiation is evaluated	Single-precision real
	s2	Exponent data, or start number of the device that stores the data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—	○			○	—
(s2)	—	○		—	○			○	—
(d)	—	○		—	○			—	—

Processing details

- Evaluates the exponentiation on the 32-bit floating-point real number specified for (s1) and 32-bit floating-point real number specified for (s2), and stores the result to the device specified for (d).


(s2) +1
(s2)
Exponent data
(s1) +1
(s1)
(d) +1
(d)
Data on which the exponentiation is evaluated
(s1) +1 (s1) × (s1) +1 (s1) × ········· × (s1) +1 (s1) × (s1) +1 (s1)
(s1) +1 (s1) raised to the exponent of (s2) +1 (s2)
32-bit floating-point real number
32-bit floating-point real number

- Values which can be specified for (s1) and (s2), and can be stored, are as follows:

0, $2^{-126} \leq$ | specified value (storing value) | < 2^{128}

- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

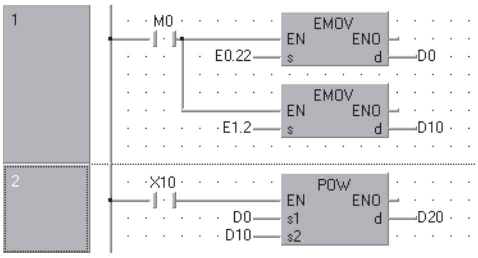
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value specified for (s1) or (s2) is not within the following range. 0, $2^{-126} \leq$ | specified value (storing value) | < 2^{128} The value specified for (s1) or (s2) is -0.	—	—	—	—	○	○
4141	The operation result is within the following range. (An overflow occurs.) $2^{128} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the exponentiation is evaluated on the 32-bit floating-point real number in D0, D1 and the 32-bit floating-point real number in D10, D11 when X10 turns ON, and the result is stored to D20, D21.

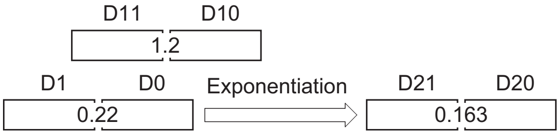
[Structured ladder/FBD]



[ST]

```
EMOV(M0,E0.22,D0);
EMOV(M0,E1.2,D10);
POW(X10,D0,D10,D20);
```

[Operation]

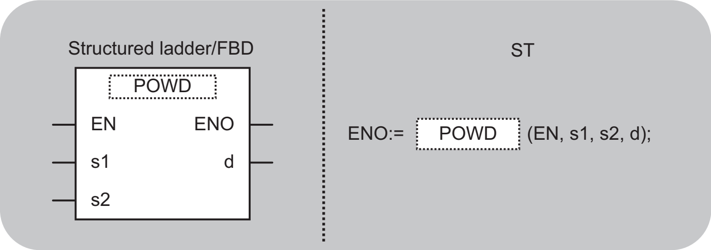


Exponentiation on floating-point data (double precision)

POWD(P)

~~Basic~~ ~~High performance~~ ~~Process~~ ~~Redundant~~ Ver. Universal LCPU

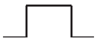
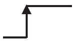
- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported



Any of the following instruction can go in the dotted squares.

POWD, POWDP

■Executing condition

Instruction	Executing condition
POWD	
POWDP	

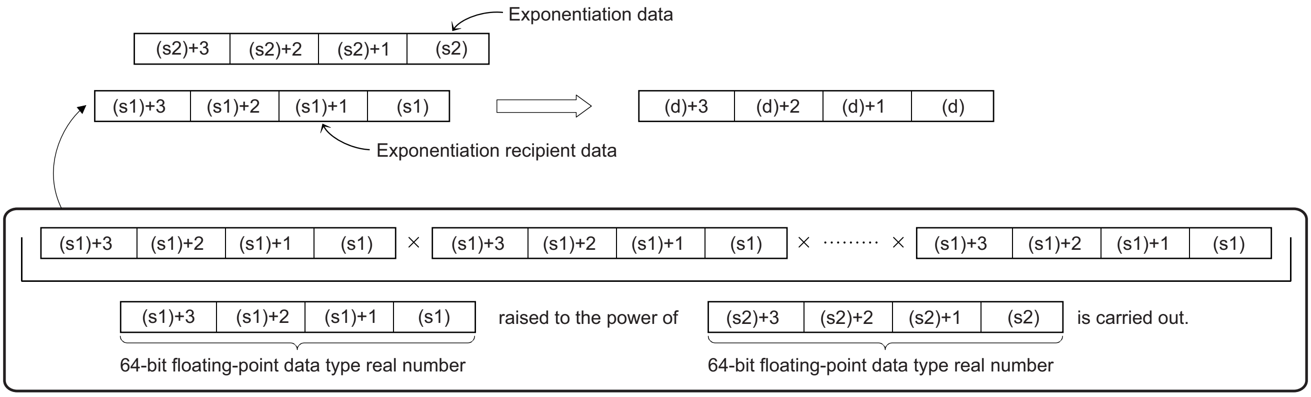
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Data on which the exponentiation is evaluated, or start number of the device that stores data on which the exponentiation is evaluated	Double-precision real
	s2	Exponent data, or start number of the device that stores the data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				○	—
(s2)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Evaluates the exponentiation on the 64-bit floating-point real number specified for (s1) and 64-bit floating-point real number specified for (s2), and stores the result to the device specified for (d).


Exponentiation data
(s2)+3
(s2)+2
(s2)+1
(s2)
(s1)+3
(s1)+2
(s1)+1
(s1)
(d)+3
(d)+2
(d)+1
(d)
Exponentiation recipient data
(s1)+3 (s1)+2 (s1)+1 (s1) × (s1)+3 (s1)+2 (s1)+1 (s1) × ········ × (s1)+3 (s1)+2 (s1)+1 (s1)
(s1)+3 (s1)+2 (s1)+1 (s1) raised to the power of (s2)+3 (s2)+2 (s2)+1 (s2) is carried out.
64-bit floating-point data type real number
64-bit floating-point data type real number

- Values which can be specified for (s1) and (s2), and can be stored, are as follows:

0, $2^{-1022} \leq$ | specified value (storing value) | < 2^{1024}

- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

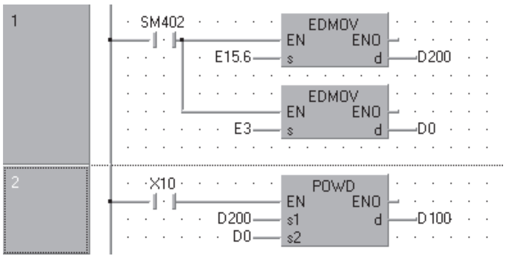
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value specified for (s1) or (s2) is not within the following range. 0, $2^{-1022} \leq$ | specified value (storing value) | < 2^{1024} The value specified for (s1) or (s2) is -0.	—	—	—	—	○	○
4141	The operation result is within the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the exponentiation is evaluated on the 64-bit floating-point real number in D200 to D203 and the 64-bit floating-point real number in D0 to D3 when X10 turns ON, and the result is stored to D100 to D103.

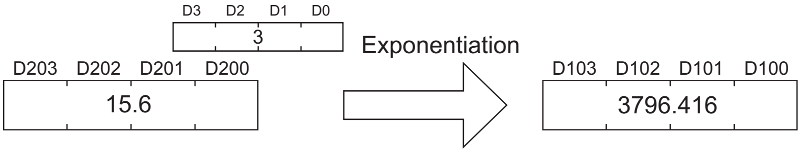
[Structured ladder/FBD]



[ST]

```
EDMOV(SM402,E15.6,D200);
EDMOV(SM402,E3,D0);
POWD(X10,D200,D0,D100);
```

[Operation]

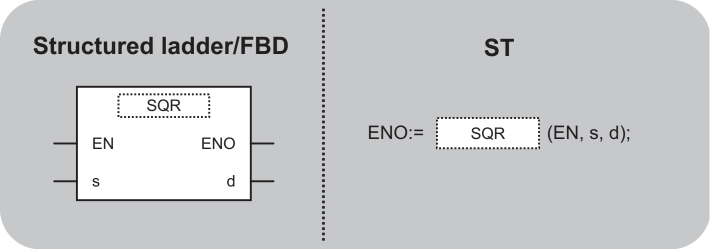


Square root operation on floating-point data (single precision)

SQR(P)

Ver. Basic | High performance | Process | Redundant | Universal | LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

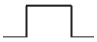
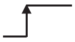


```
ENO:= SQR (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

SQR, SQRP

■Executing condition

Instruction	Executing condition
SQR	
SQRP	

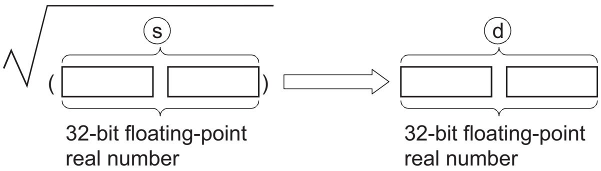
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the square root function is evaluated, or start number of the device that stores data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Evaluates the square root function on the value specified for (s), and stores the result to the device specified for (d).


s
d
32-bit floating-point real number
32-bit floating-point real number

- Only positive values can be specified for (s). (Operation cannot be performed on negative numbers.)
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

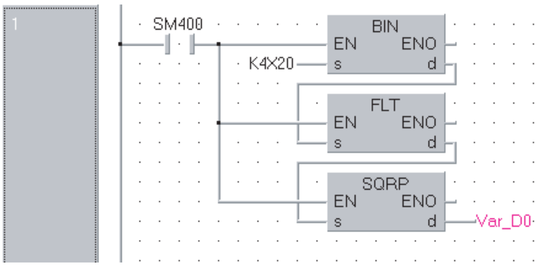
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is a negative number.	○	○	○	○	○	○
	The value of the specified device is -0.[*1]	○	○	○	○	—	—
4140	The value of the specified device is outside the following range. 0, $2^{-126} \leq$ | value of the specified device | $< 2^{128}$ The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{128} \leq$ | operation result |	—	—	—	—	○	○

*1 There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Program example

- In the following program, the square root function is evaluated on the value set in the BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 32-bit floating-point real number.

[Structured ladder/FBD]



Inputs the data on which the square root function is evaluated. (①)

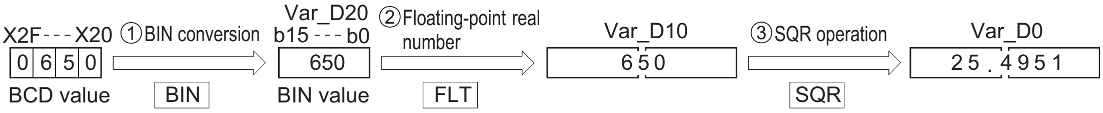
Converts the input data to 32-bit floating-point real number data. (②)

Square root operation (③)

[ST]

```
BIN(SM400,K4X20,Var_D20);
FLT(SM400,Var_D20,Var_D10);
SQRP(SM400,Var_D10,Var_D0);
```

[Operation example when the value of 650 is specified for X20 to X2F]

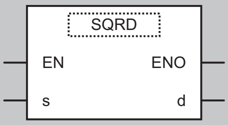


Square root operation on floating-point data (double precision)

SQRD(P)

✗Basic ✗High performance ✗Process ✗Redundant Universal LCPU

Structured ladder/FBD



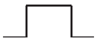
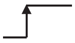
ST

```
ENO:= SQRD (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

SQRD, SQRDP

■Executing condition

Instruction	Executing condition
SQRD	
SQRDP	

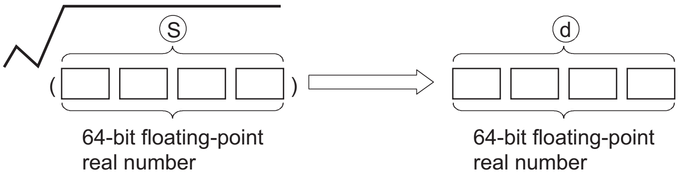
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the square root function is evaluated, or start number of the device that stores data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Evaluates the square root function on the value specified for (s), and stores the result to the device specified for (d).


Ⓢ
Ⓓ
64-bit floating-point real number
64-bit floating-point real number

- Only positive values can be specified for (s). (Operation cannot be performed on negative numbers.)
- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

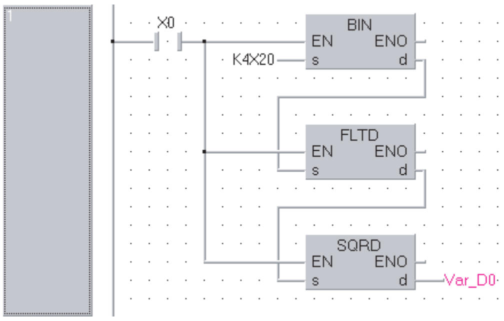
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is a negative number.	—	—	—	—	○	○
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | $< 2^{1024}$ The value of the specified device is -0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the square root function is evaluated on the value set in the BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 64-bit floating-point real number.

[Structured ladder/FBD]



Inputs the data on which the square root function is evaluated. (①)

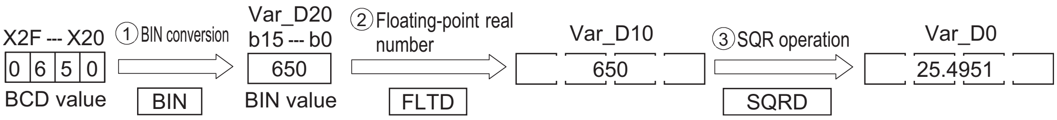
Converts the input data to 64-bit floating-point real number data. (②)

Square root operation (③)

[ST]

```
BIN(X0,K4X20,Var_D20);
FLTD(X0,Var_D20,Var_D10);
SQRD(X0,Var_D10,Var_D0);
```

[Operation example when the value of 650 is specified for X20 to X2F]

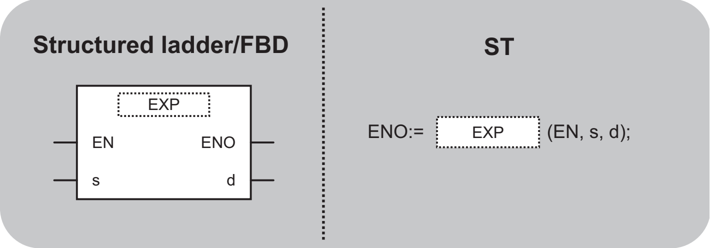


Exponential operation on floating-point data (single precision)

EXP(P)

Ver. Basic | High performance | Process | Redundant | Universal | LCPU

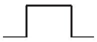
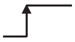
- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later



Any of the following instruction can go in the dotted squares.

EXP, EXPP

■Executing condition

Instruction	Executing condition
EXP	
EXPP	

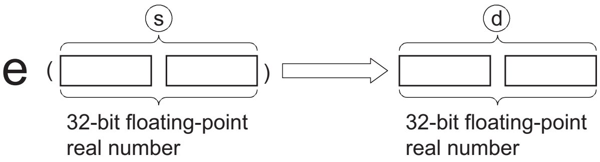
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the exponential function is evaluated, or start number of the device that stores data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Evaluates the exponential function on the value specified for (s), and stores the result to the device specified for (d).


s
d
e
32-bit floating-point real number
32-bit floating-point real number

- Exponential operation is evaluated taking the base (e) as 2.71828.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

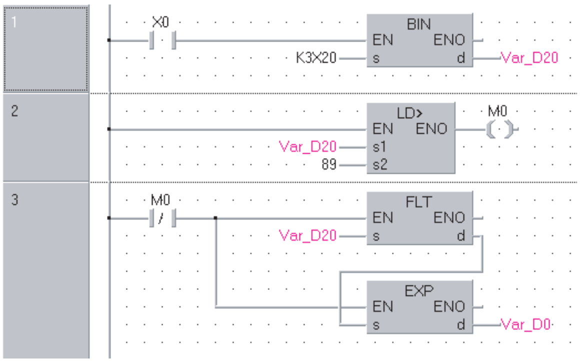
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The operation result is not within the following range. $2^{-126} \leq$ | operation result | $\leq 2^{128}$	—	○	—	—	—	—
	The operation result is not within the following range: $2^{-126} \leq$ | operation result | $< 2^{128}$	○	—	○	○	—	—
	The value of the specified device is -0.[*1]	○	○	○	○	—	—
4140	The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{128} \leq$ | operation result |	—	—	—	—	○	○

[*1] There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Program example

- In the following program, the exponential function is evaluated on the value set in BCD 2 digits in the devices from X20 to X27, and the result is stored to Var_D0 as a 32-bit floating-point real number.

[Structured ladder/FBD]



Inputs the data on which the exponential function is evaluated. (①)

Checks the value range to be evaluated.*2

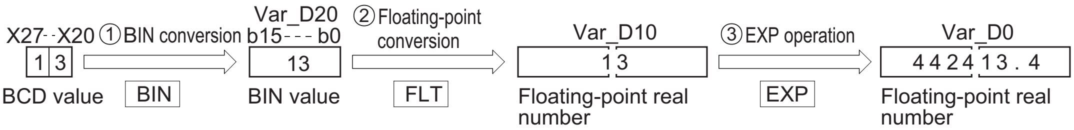
Converts the input data to 32-bit floating-point real number data. (②)

Exponential operation (③)

[ST]

```
BIN(X0,K3X20,Var_D20);
OUT(Var_D20>89,M0);
FLT(NOT (M0),Var_D20,Var_D10);
EXP(NOT (M0),Var_D10,Var_D0);
```

[Operation example when the value of 13 is specified for X20 to X27]



*2 The operation result will be under 2129 if the BCD value of X20 to X27 is less than 89, from the calculation loge 2129 = 89.4. Because setting a value of over 90 will return an operation error, turn M0 ON if a value of over 90 has been set to avoid the error.

Point

Conversion from natural logarithm to common logarithm

In the CPU module, a natural logarithm operation is performed.

To obtain a common logarithm value, enter a common logarithm value divided by 0.43429 in (s).

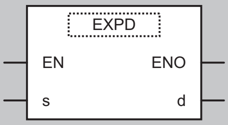
$$10^x = e^{\frac{x}{0.43429}}$$

Exponential operation on floating-point data (double precision)

EXPD(P)

~~Basic~~ ~~High performance~~ ~~Process~~ ~~Redundant~~ Universal LCPU

Structured ladder/FBD



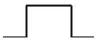
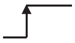
ST

ENO:= EXPD (EN, s, d);

Any of the following instruction can go in the dotted squares.

EXPD, EXPDP

■Executing condition

Instruction	Executing condition
EXPD	
EXPDP	

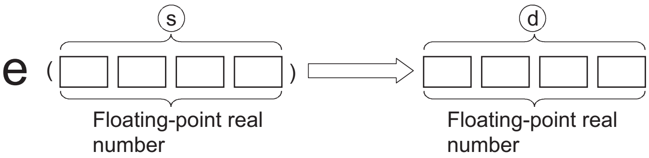
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the exponential function is evaluated, or start number of the device that stores data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Evaluates the exponential function on the value specified for (s), and stores the result to the device specified for (d).



- Exponential operation is evaluated taking the base (e) as 2.71828.
- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

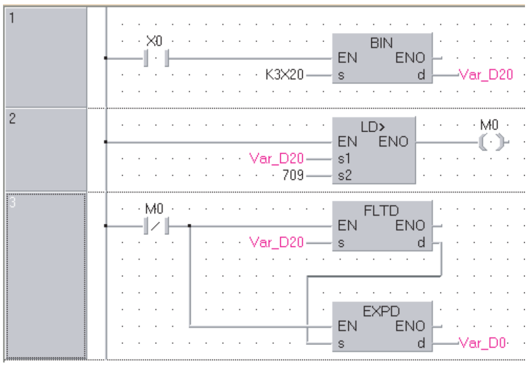
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | $< 2^{1024}$ The value of the specified device is -0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the exponential function is evaluated on the value set in BCD 2 digits in the devices from X20 to X31, and the result is stored to Var_D0 as a 64-bit floating-point real number.

[Structured ladder/FBD]



Inputs the data on which the exponential function is evaluated. (①)

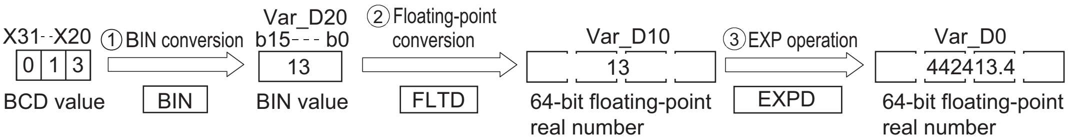
Checks the value range to be evaluated.*1

Converts the input data to 64-bit floating-point real number data. (②)

Exponential operation (③)

```
[ST]
BIN(X0,K2X20,Var_D20);
OUT(Var_D20>709,M0);
IF NOT(M0) THEN
    FLTD(TRUE,Var_D20,Var_D10);
    EXPD(TRUE,Var_D10,Var_D0);
END_IF;
```

[Operation example when the value of 13 is specified for X20 to X31]



*1 The operation result will be under 2129 if the BCD value of X20 to X31 is less than 89, from the calculation $\log_e 2129 = 89.4$. Because setting a value of over 90 will return an operation error, turn M0 ON if a value of over 90 has been set to avoid the error.

Point

Conversion from natural logarithm to common logarithm

In the CPU module, a natural logarithm operation is performed.

To obtain a common logarithm value, enter a common logarithm value divided by 0.43429 in (s) .

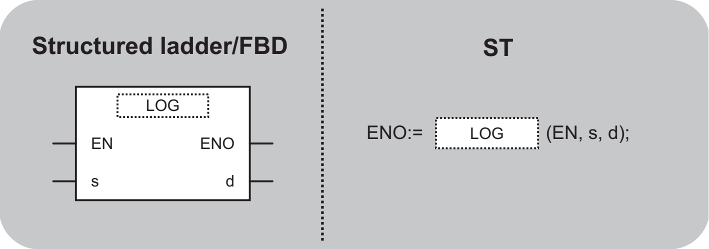
$$10^{x} = e^{\frac{x}{0.43429}}$$

Natural logarithm operation on floating-point data (single precision)

LOG(P)

Ver. Basic | High performance | Process | Redundant | Universal | LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

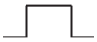
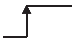


```
ENO:= LOG (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

LOG, LOGP

■Executing condition

Instruction	Executing condition
LOG	
LOGP	

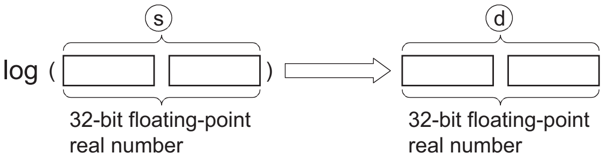
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the natural logarithm function is evaluated, or start number of the device that stores data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Evaluates the natural logarithm function on the value specified for (s) taking (e) as a base, and stores the result to the device specified for (d).


s
d
log (
)
32-bit floating-point real number
32-bit floating-point real number

- Only positive values can be specified for (s). (Operation cannot be performed on negative numbers.)
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

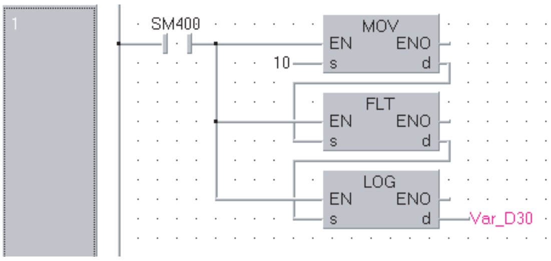
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is negative. The value specified for (s) is 0.	○	○	○	○	○	○
	The value of the specified device is -0.[1]	○	○	○	○	—	—
4140	The value of the specified device is outside the following range. 0, 2^{-126} ≤ | value of the specified device | < 2^{128} The value of the specified device is -0, unnormalized number, nonnumeric or ±∞.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) 2^{128} ≤ | operation result |	—	—	—	—	○	○

[1] There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Program example

- In the following program, the natural logarithm function is evaluated on the value 10 set in D50, and the result is stored to Var_D30.

[Structured ladder/FBD]



Inputs the data on which the natural logarithm function is evaluated. (①)

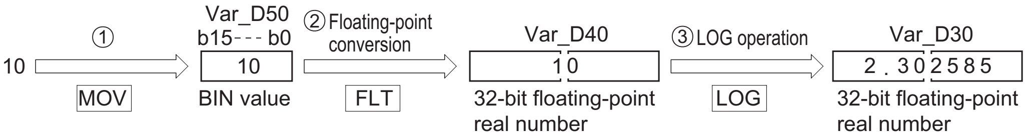
Converts the input data to 32-bit floating-point real number data. (②)

Natural logarithm operation (③)

[ST]

```
Var_D50: =10;
FLT(SM400,Var_D50,Var_D40);
LOG(SM400,Var_D40,Var_D30);
```

[Operation]



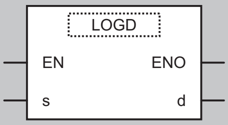
10
①
MOV
Var_D50
b15- - - b0
10
BIN value
② Floating-point conversion
FLT
Var_D40
1 0
32-bit floating-point real number
③ LOG operation
LOG
Var_D30
2 . 3 0 2 5 8 5
32-bit floating-point real number

Natural logarithm operation on floating-point data (double precision)

LOGD(P)

~~Basic~~ ~~High performance~~ ~~Process~~ ~~Redundant~~ Universal LCPU

Structured ladder/FBD



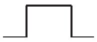
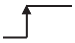
ST

ENO:= LOGD (EN, s, d);

Any of the following instruction can go in the dotted squares.

LOGD, LOGDP

■Executing condition

Instruction	Executing condition
LOGD	
LOGDP	

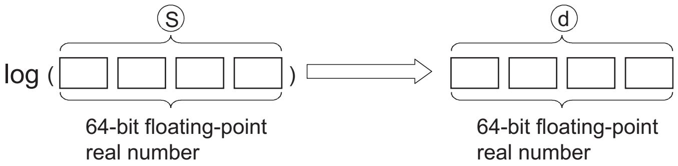
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the natural logarithm function is evaluated, or start number of the device that stores data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Evaluates the natural logarithm function on the value specified for (s) taking (e) as a base, and stores the result to the device specified for (d).


S
d
log (
)
64-bit floating-point real number
64-bit floating-point real number

- Only positive values can be specified for (s). (Operation cannot be performed on negative numbers.)
- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

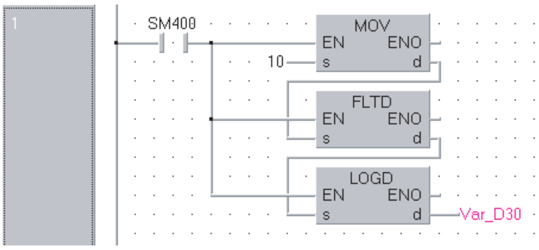
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is negative. The value specified for (s) is 0.	—	—	—	—	○	○
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | < 2^{1024} The value of the specified device is -0.	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the natural logarithm function is evaluated on the value 10 set in D50, and the result is stored to Var_D30.

[Structured ladder/FBD]



Inputs the data on which the natural logarithm function is evaluated. (①)

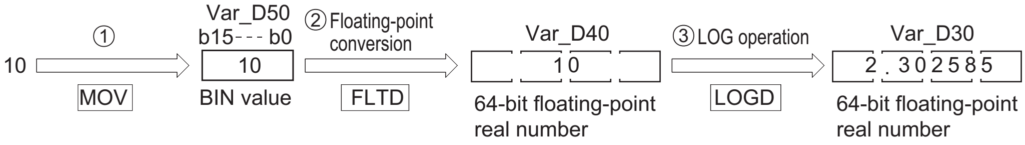
Converts the input data to 64-bit floating-point real number data. (②)

Natural logarithm operation (③)

[ST]

```
Var_D50: =10;
FLTD(SM400,Var_D50,Var_D40);
LOGD(SM400,Var_D40,Var_D30);
```

[Operation]


10
①
MOV
Var_D50
b15 - - - b0
10
BIN value
② Floating-point conversion
FLTD
Var_D40
1 0
64-bit floating-point real number
③ LOG operation
LOGD
Var_D30
2 . 3 0 2 5 8 5
64-bit floating-point real number

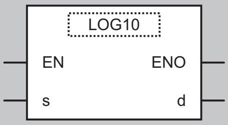
Common logarithm operation on floating-point data (single precision)

LOG10(P)

✗ Basic ✗ High performance ✗ Process ✗ Redundant Ver. Universal LCPU

- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported

Structured ladder/FBD



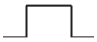
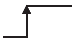
ST

ENO:= LOG10 (EN, s, d);

Any of the following instruction can go in the dotted squares.

LOG10, LOG10P

■Executing condition

Instruction	Executing condition
LOG10	
LOG10P	

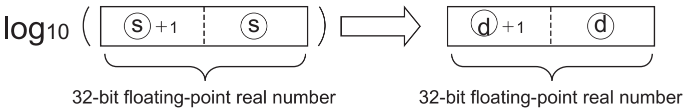
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the common logarithm function is evaluated, or start number of the device that stores data	Single-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Single-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	○		—	○	—
(d)	—	○		—	○		—	—	—

Processing details

- Evaluates the common logarithm function on the value specified for (s) taking 10 as a base, and stores the result to the device specified for (d).


log10
(s)+1
(s)
(d)+1
(d)
32-bit floating-point real number
32-bit floating-point real number

- Only positive values can be specified for (s). (Operation cannot be performed on negative numbers.)
- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

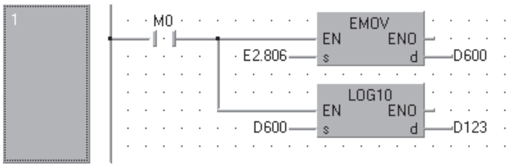
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is negative. The value specified for (s) is 0.	—	—	—	—	○	○
4140	The value of the specified device is outside the following range. 0, $2^{-126} \leq$ | value of the specified device | $< 2^{128}$ The value specified for (s) is -0.	—	—	—	—	○	○
4141	The operation result is within the following range. (An overflow occurs.) $2^{128} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the common logarithm function is evaluated on the 32-bit floating-point real number stored in D600 and D601 when M0 turns ON, and the result is stored to D123 and D124.

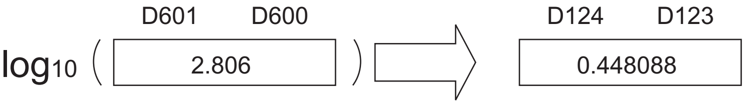
[Structured ladder/FBD]



[ST]

```
EMOV(M0,E2.806,D600);
LOG10(M0,D600,D123);
```

[Operation]

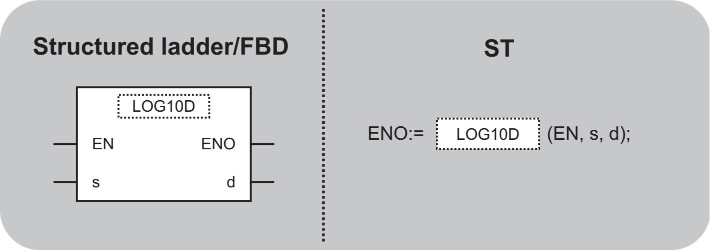


Common logarithm operation on floating-point data (double precision)

LOG10D(P)

✗ Basic ✗ High performance ✗ Process ✗ Redundant Ver. Universal LCPU

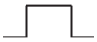
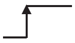
- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported



Any of the following instruction can go in the dotted squares.

LOG10D, LOG10DP

■Executing condition

Instruction	Executing condition
LOG10D	
LOG10DP	

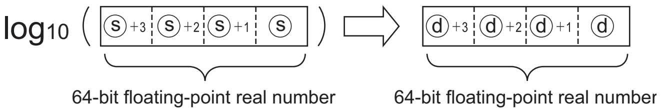
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the common logarithm function is evaluated, or start number of the device that stores data	Double-precision real
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	Double-precision real

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant E	Others
	Bit	Word		Bit	Word				
(s)	—	○		—	—			○	—
(d)	—	○		—	—			—	—

Processing details

- Evaluates the common logarithm function on the value specified for (s) taking 10 as a base, and stores the result to the device specified for (d).


log10
ⓢ+3 ⓢ+2 ⓢ+1 ⓢ
ⓓ+3 ⓓ+2 ⓓ+1 ⓓ
64-bit floating-point real number
64-bit floating-point real number

- Only positive values can be specified for (s). (Operation cannot be performed on negative numbers.)
- When the operation results in -0 or an underflow, the result is processed as 0.
- If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the following.

📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

Operation error

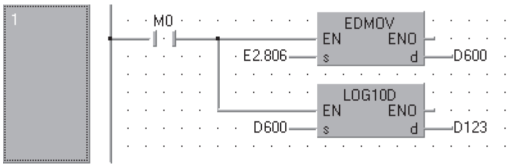
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is negative. The value specified for (s) is 0.	—	—	—	—	○	○
4140	The value of the specified device is outside the following range. 0, $2^{-1022} \leq$ | value of the specified device | $< 2^{1024}$ The value specified for (s) is -0.	—	—	—	—	○	○
4141	The operation result is within the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

Program example

- In the following program, the common logarithm function is evaluated on the 32-bit floating-point real number stored in D600 to D603 when M0 turns ON, and the result is stored to D123 to D126.

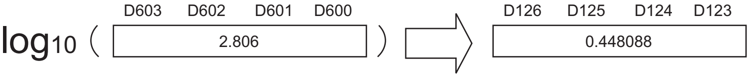
[Structured ladder/FBD]



[ST]

```
EDMOV(M0,E2.806,D600);
LOG10D(M0,D600,D123);
```

[Operation]

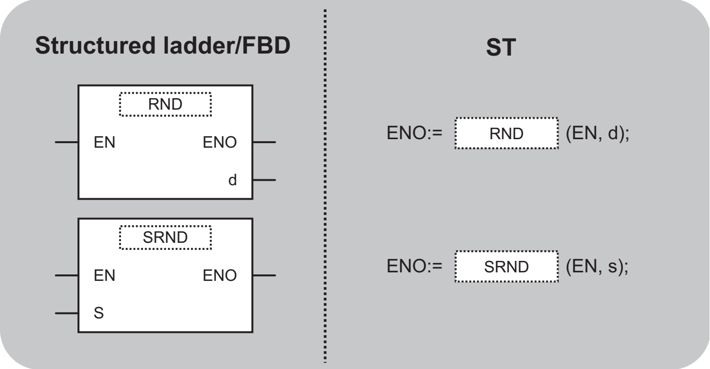


Random number generation and series update

RND(P), SRND(P)

Ver. Basic High performance Process Redundant Universal LCPU

- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later



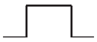
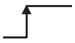
```
ENO:= RND (EN, d);

ENO:= SRND (EN, s);
```

Any of the following instruction can go in the dotted squares.

RND, RNDP , SRND, SRNDP

■Executing condition

Instruction	Executing condition
RND, SRND	
RNDP, SRNDP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Random number series data, or start number of the device that stores random number series data (When using the SRND instruction)	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores random numbers (When using the RND instruction)	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○							○	—
(d)	○							—	—

Processing details

- The random number generation instruction generates random numbers conforming to a certain calculation formula. In the calculation using the formula, the result of previous calculation is used as a coefficient. The random series update instruction can change the random number generation pattern.

■RND(P)

- Generates random numbers from 0 to 32767, and stores them to the device specified for (d).

■SRND(P)

- Updates random number series according to the 16-bit BIN data stored in the device specified for (s).

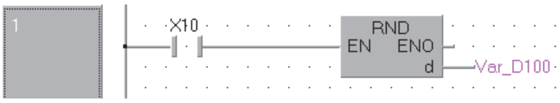
Operation error

- There is no operation error.

Program example

- In the following program, the random number is stored to Var_D100 when X10 turns ON.

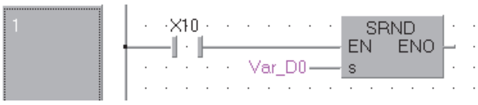
[Structured ladder/FBD]



[ST]
RND(X10,Var_D100);

- In the following program, the random number series is updated according to the value of Var_D0 when X10 turns ON.

[Structured ladder/FBD]

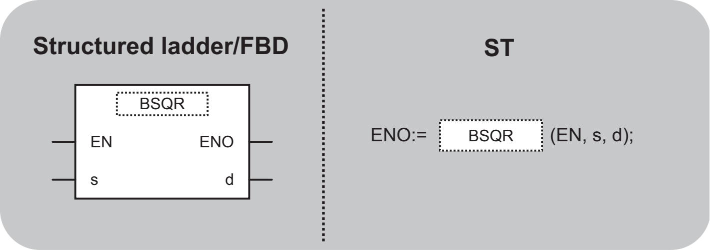


[ST]
SRND(X10,Var_D0);

Square root operation on 4-/8-digit BCD data

BSQR(P), BDSQR(P)

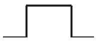
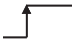
Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

BSQR, BSQRP, BDSQR, BDSQRP

■Executing condition

Instruction	Executing condition
BSQR, BDSQR	
BSQRP, BDSQRP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the square root function is evaluated, or start number of the device that stores data	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○							○	—
(d)	○							—	—

Processing details

■BSQR(P)

- Evaluates the square root function on the value specified for (s), and stores the result to the device specified for (d).

$$\sqrt{\textcircled{s}} = \overset{\textcircled{d}}{\boxed{\text{Integer part}}}.\overset{\textcircled{d}+1}{\boxed{\text{Fractional part}}}$$

- Values that can be specified for (s) are BCD values with a maximum of 4 digits (from 0 to 9999).
- The operation results of (d) and (d)+1 are stored as their respective BCD values between 0 and 9999.
- The value of the operation result is rounded off at the fifth digit of fractional part. For this reason, the fourth digit of fractional part has an error of ±1.

■BDSQR(P)

- Evaluates the square root function on the values specified for (s) and (s)+1 and stores the results to the device specified for (d).

$$\sqrt{(\underbrace{\boxed{\textcircled{s}+1}\ \boxed{\textcircled{s}}}_{\text{2-word data}})} = \overset{\textcircled{d}}{\boxed{\text{Integer part}}}.\overset{\textcircled{d}+1}{\boxed{\text{Fractional part}}}$$

- BCD value of a maximum of 8 digits (0 to 99999999) can be specified for (s) and (s)+1.
- The operation results of (d) and (d)+1 are stored as their respective BCD values between 0 and 9999.
- The value of the operation result is rounded off at the fifth digit of fractional part. For this reason, the fourth digit of fractional part has an error of ±1.

Operation error

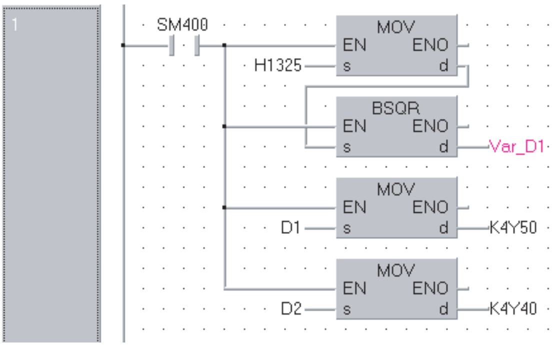
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is not a BCD value.	—	○	○	○	○	○

Program example

- In the following program, the square root function is evaluated on 1325 in BCD value, and the integer part is output in BCD 4 digits to the devices from Y50 to Y5F, and the fractional part is output in BCD 4 digits to the devices from Y40 to Y4F.

[Structured ladder/FBD]



Sets the data on which the square root function is evaluated. (①)

Square root operation (②)

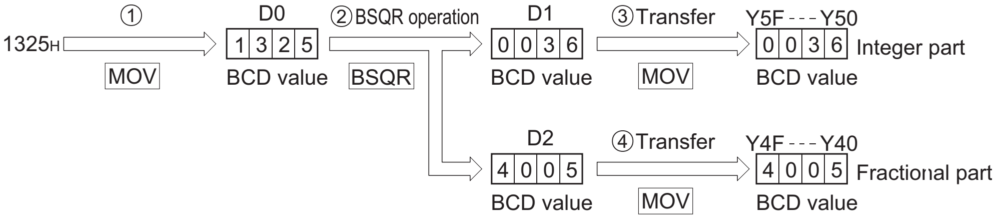
Outputs the integer part of the operation result to the display. (③)

Outputs the fractional part of the operation result to the display. (④)

[ST]

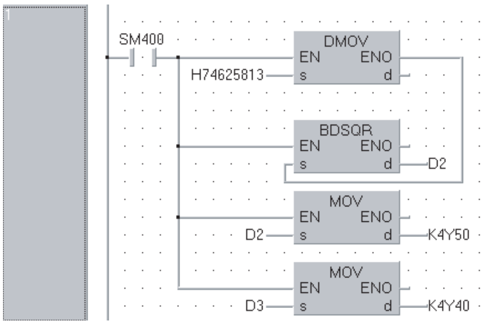
```
MOV(SM400,H1325,D0);
BSQR(SM400,D0,D1);
K4Y50: =D1;
K4Y40: =D2;
```

[Operation]



- In the following program, the square root function is evaluated on 74625813 in BCD value, and the integer part of the result is output in BCD 4 digits to the devices from Y50 to Y5F, and the fractional part is output in BCD 4 digits to the devices from Y40 to Y4F.

[Structured ladder/FBD]



Sets the data (BCD value) on which the square root function is evaluated. (①)

Square root operation (②)

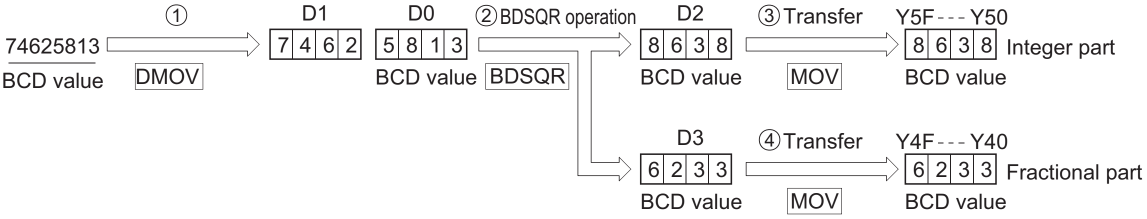
Outputs the integer part of the operation result to the display. (③)

Outputs the fractional part of the operation result to the display. (④)

[ST]

```
DMOV(SM400,H74625813,D0);
BDSQR(SM400,D0,D2);
K4Y50: =D2;
K4Y40: =D3;
```

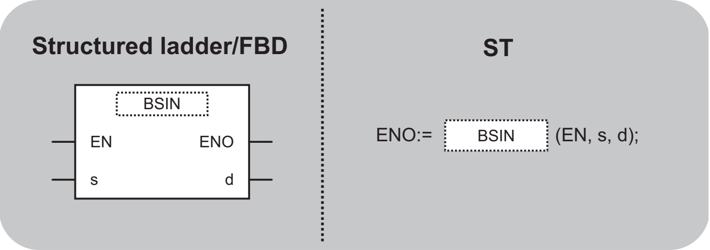
[Operation]



SIN operation on data in BCD format

BSIN(P)

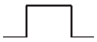
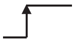
Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

BSIN, BSINP

■Executing condition

Instruction	Executing condition
BSIN	
BSINP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the sine function is evaluated, or start number of the device that stores data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores the operation result	Array of ANY16 (1..3)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○	○						○	—
(d)	—	○		—				—	—

Processing details

- Evaluates the sine function on the value (angle) specified for (s), and stores the sign of the operation result to the device specified for (d)[0], and the operation result to the devices specified for (d)[1] and (d)[2].

	ⓓ[0]	ⓓ[1]		ⓓ[2]
SIN ⓢ =	Sign	Integer part	.	Fractional part

- The value specified for (s) is a BCD value which can be between 0 and 360 degrees (in degrees).
- The sign for the operation result stored to (d)[0] will be 0 if the result is a positive value, and 1 if the result is a negative value.
- The operation results stored to (d)[0], (d)[1], and (d)[2] are BCD values between -1.0000 and 1.0000.
- The value of the operation result is rounded off at the fifth digit of fractional part.

Operation error

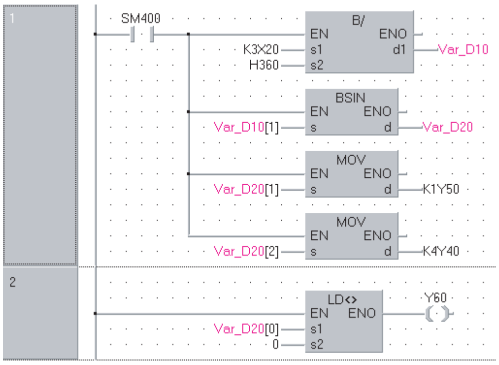
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is not a BCD value. The value specified for (s) is not in the range of 0 to 360.	—	○	○	○	○	○
4101	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the sine function is evaluated on the data specified as BCD 3 digits in X20 to X2B, and the integer part of the result is output in BCD 1 digit to the devices from Y50 to Y53, and the fractional part is output in BCD 4 digits to the devices from Y40 to Y4F. Y60 is turned ON if the operation result is negative. (If a value has been set for X20 to X2F that is greater than 360, it will be adjusted to be in the range from 0 to 360 degrees.)

[Structured ladder/FBD]



Adjusts the input angle data to be in the range of 0 to 360 degrees. (①)

Evaluates the sine function. (②)

Outputs the integer part of the operation result to the display. (③)

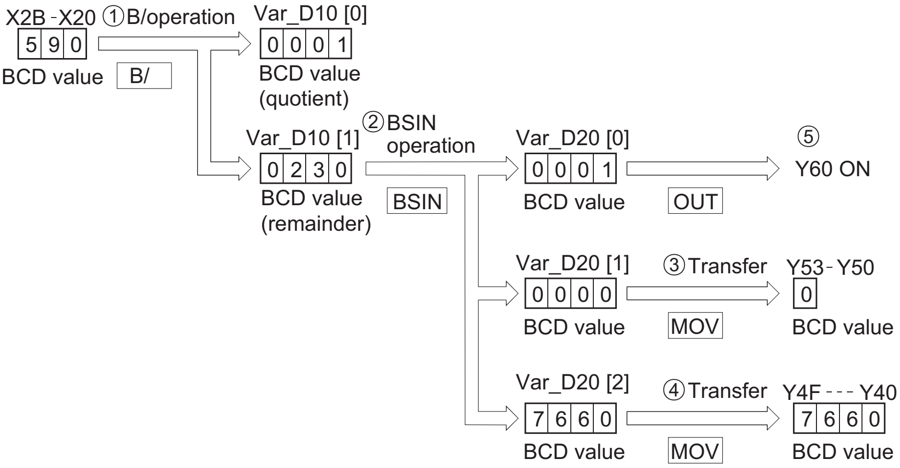
Outputs the fractional part of the operation result to the display. (④)

Outputs the sign of the operation result by turning the coil ON or OFF. (⑤)

[ST]

```
BINP(SM400,K3X20,Var_D50);
BINP(SM400,H360,Var_D51);
BCDP(SM400,Var_D50 MOD Var_D51,Var_D10);
BSIN(SM400,Var_D10,Var_D20);
K1Y50: =Var_D20[1];
K4Y40: =Var_D20[2];
OUT(Var_D20[0]<>0,Y60);
```

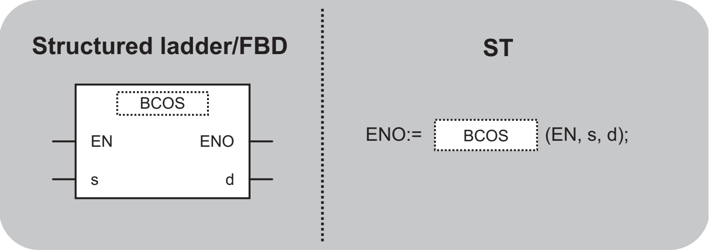
[Operation example when the value 590 is specified for X20 to X2B]



COS operation on data in BCD format

BCOS(P)

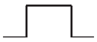
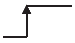
~~Basic~~ High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

BCOS, BCOSP

■Executing condition

Instruction	Executing condition
BCOS	
BCOSP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the cosine function is evaluated, or start number of the device that stores data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores the operation result	Array of ANY16 (1..3)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○	○		○					—
(d)	—	○		—					—

Processing details

- Evaluates the cosine function on the value (angle) specified for (s), then stores the sign of the operation result to the word device specified for (d)[0], and the operation result to the word device specified for (d)[1] and (d)[2].

	ⓓ[0]	ⓓ[1]		ⓓ[2]
COSⓢ =	Sign	Integer part	.	Fractional part

- The value specified for (s) is a BCD value which can be between 0 and 360 degrees (in degrees).
- The sign for the operation result stored to (d)[0] will be 0 if the result is a positive value, and 1 if the result is a negative value.
- The operation results stored to (d)[0], (d)[1], and (d)[2] are BCD values between -1.0000 and 1.0000.
- The value of the operation result is rounded off at the fifth digit of fractional part.

Operation error

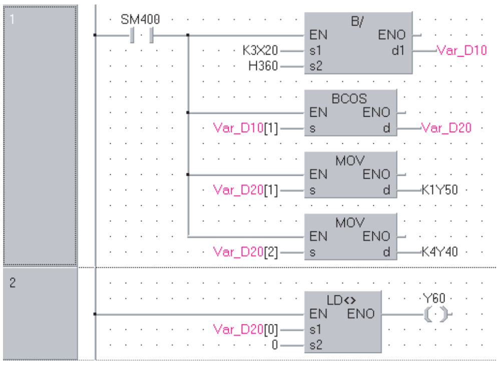
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is not a BCD value. The value specified for (s) is not in the range of 0 to 360.	—	○	○	○	○	○
4101	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the cosine function is evaluated on the data specified as BCD 3 digits in X20 to X2B, and the integer part of the result is output in BCD 1 digit to the devices from Y50 to Y53, and the fractional part is output in BCD 4 digits to the devices from Y40 to Y4F. Y60 is turned ON if the operation result is negative.

[Structured ladder/FBD]



Adjusts the input angle data to be in the range of 0 to 360 degrees. (①)

Evaluates the cosine function. (②)

Outputs the integer part of the operation result to the display. (③)

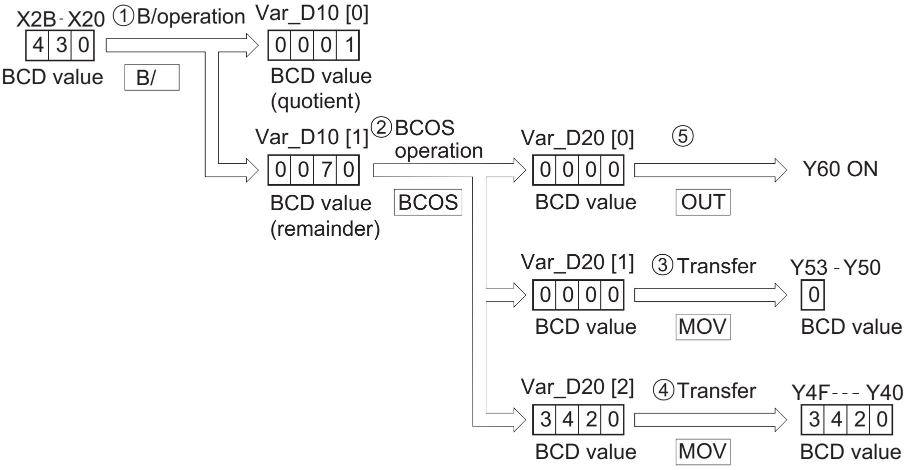
Outputs the fractional part of the operation result to the display. (④)

Outputs the sign of the operation result by turning the coil ON or OFF. (⑤)

[ST]

```
BINP(SM400,K3X20,Var_D50);
BINP(SM400,H360,Var_D51);
BCDP(SM400,Var_D50 MOD Var_D51,Var_D11);
BCOS(SM400,Var_D11,Var_D20);
K1Y50: =Var_D20[1];
K4Y40: =Var_D20[2];
OUT(Var_D20[0]<>0,Y60);
```

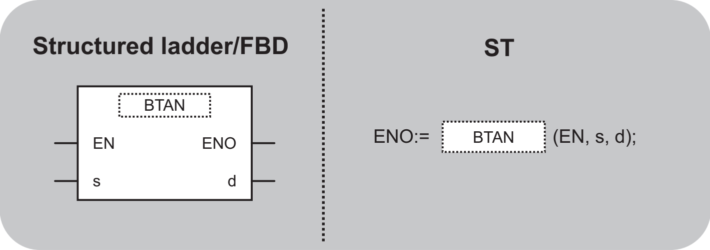
[Operation example when the value 430 is specified for X20 to X2B]



TAN operation on data in BCD format

BTAN(P)

Basic High performance Process Redundant Universal LCPU

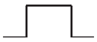
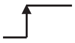


```
ENO:= BTAN (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

BTAN, BTANP

■Executing condition

Instruction	Executing condition
BTAN	
BTANP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the tangent function is evaluated, or start number of the device that stores data	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores the operation result	Array of ANY16 (1..3)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○	○		○					—
(d)	—	○		—					—

Processing details

- Evaluates the tangent function on the value (angle) specified for (s), and stores the sign of the operation result to the word device specified for (d)[0], and the operation result to the word device specified for (d)[1] and (d)[2].

	ⓓ[0]	ⓓ[1]		ⓓ[2]
TANⓢ =	Sign	Integer part	.	Fractional part

- The value specified for (s) is a BCD value which can be between 0 and 360 degrees (in degrees).
- The sign for the operation result stored to (d)[0] will be 0 if the result is a positive value, and 1 if the result is a negative value.
- The operation results stored to (d)[0], (d)[1], and (d)[2] are BCD values between -57.2901 and 57.2902.
- The value of the operation result is rounded off at the fifth digit of fractional part.

Operation error

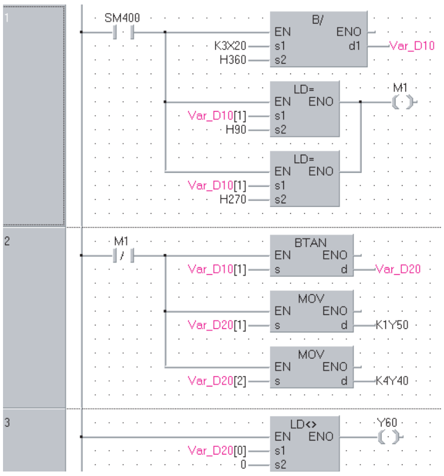
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is not a BCD value. The value specified for (s) is not in the range of 0 to 360. The value specified for (s) is 90 or 270.	—	○	○	○	○	○
4101	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the tangent function is evaluated on the data specified as BCD 3 digits in X20 to X2B, and the integer part of the result is output in the BCD 4 digits to the devices from Y50 to Y53, and the fractional part is output in BCD 4 digits to the devices from Y40 to Y4F. Y60 is turned ON if the operation result is negative.

[Structured ladder/FBD]



Adjusts the input angle data to be in the range of 0 to 360 degrees. (①)

Use M1 for the interlock to disable the operation when the input angle data is 90 or 270 degrees.

Evaluates the tangent function. (②)

Outputs the integer part of the operation result to the display. (③)

Outputs the fractional part of the operation result to the display. (④)

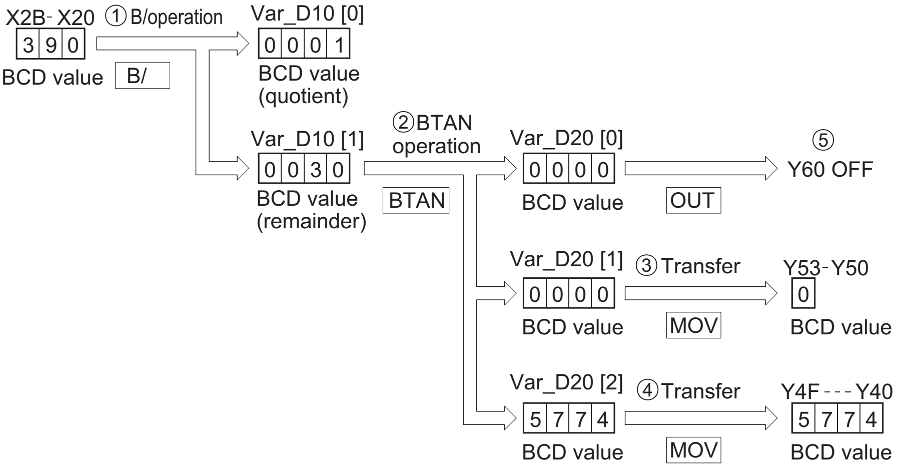
Outputs the sign of the operation result by turning the coil ON or OFF. (⑤)

[ST]

```
BINP(SM400,K3X20,Var_D50);
BINP(SM400,H360,Var_D51);
BCDP(SM400,Var_D50 MOD Var_D51,Var_D10);
OUT(Var_D10=H90 OR Var_D10=H270,M1);

IF NOT(M1) THEN
    BTAN(TRUE,Var_D10,Var_D20);
    K1Y50: =Var_D20[1];
    K4Y40: =Var_D20[2];
END_IF;
OUT(Var_D20[0]<>0,Y60);
```

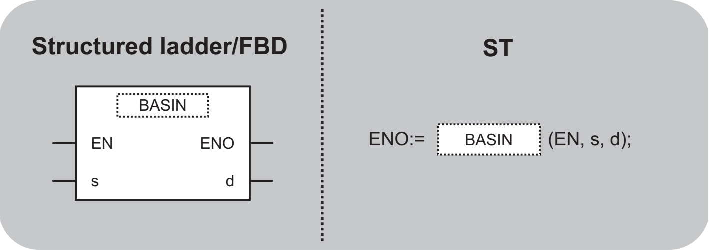
[Operation example when the value 390 is specified for X20 to X2B]



SIN^{-1} operation on data in BCD format

BASIN(P)

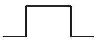
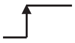
Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

BASIN, BASINP

■Executing condition

Instruction	Executing condition
BASIN	
BASINP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the arcsine function is evaluated, or start number of the device that stores data	Array of ANY16 (1..3)
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				—	
(d)	○	○		○				—	

Processing details

- Evaluates the arcsine function on the value specified for (s) and stores the operation result (angle) to the device specified for (d).

$$\mathrm{SIN}^{-1}=(\overset{\textcircled{s}[0]}{\boxed{\text{Sign}}}\ \overset{\textcircled{s}[1]}{\boxed{\text{Integer part}}}.\overset{\textcircled{s}[2]}{\boxed{\text{Fractional part}}})=\textcircled{d}$$

- A sign of the data to be evaluated is set for (s)[0]. If the evaluation value is a positive value, it stores 0, and if it is a negative value, it stores 1.
- The integer part and fractional part are stored in (s)[1] and (s)[2] respectively in BCD values. (Values between 0 and 1.0000 can be set.)
- Operation results stored to (d) are BCD values between 0 and 90 degrees, and 270 and 360 degrees (in degrees).
- The fractional part is rounded off for the operation result.

Operation error

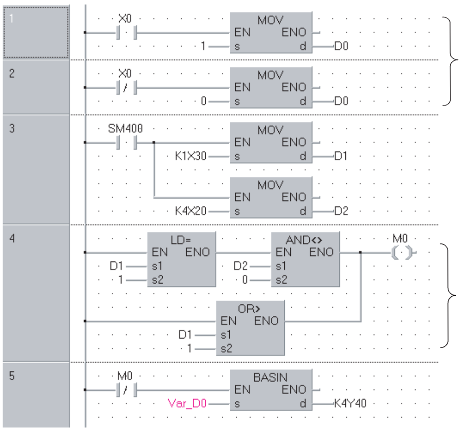
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is not a BCD value. The value specified for (s) is not in the range of -1.0000 and 1.0000.	—	○	○	○	○	○
4101	The device specified for (s) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the arcsine function is evaluated on the data in which X0 with positive and negative signs (positive when X0 is OFF, and negative when X0 is ON), the integer part in BCD 1 digit from X30 to X33 and the fractional part in BCD 4 digits from X20 to X2F, and the evaluated angle is output in BCD 4 digits to the devices from Y40 to Y4F.

[Structured ladder/FBD]



Sets the sign of the sine value. (①)

Sets the integer part of the sine value. (②)

Sets the fractional part of the sine value. (③)

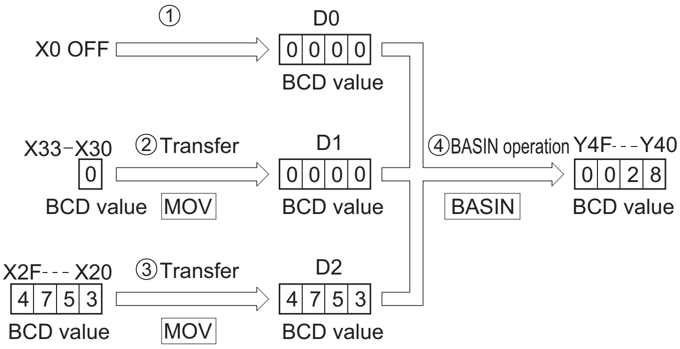
Checks the range of the set sine value.

Evaluates the arcsine function and outputs the operation result to the devices from Y40 to Y4F. (④)

[ST]

```
MOV(X0,1,D0);
MOV(NOT(X0),0,D0);
D1: =K1X30;
D2: =K4X20;
OUT((D1=1 AND D2<>0) OR D1>1,M0);
BASIN(NOT(M0),Var_D0,K4Y40);
```

[Operation example when the value 0.4753 is specified for X20 to X33]

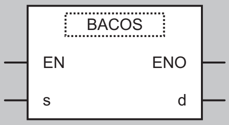


COS^{-1} operation on data in BCD format

BACOS(P)

~~Basic~~ High performance Process Redundant Universal LCPU

Structured ladder/FBD



ST

```
ENO:= BACOS (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

BACOS, BACOSP

■Executing condition

Instruction	Executing condition
BACOS	⎍
BACOSP	↑

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the arccosine function is evaluated, or start number of the device that stores data	Array of ANY16 (1..3)
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				—	—
(d)	○	○		○				—	—

Processing details

- Evaluates the arccosine function on the value specified for (s), and stores operation result (angle) to the device specified for (d).

$$\text{COS}^{-1}\ (\ \overset{\text{ⓢ[0]}}{\boxed{\text{Sign}}}\ \overset{\text{ⓢ[1]}}{\boxed{\text{Integer part}}}\ .\ \overset{\text{ⓢ[2]}}{\boxed{\text{Fractional part}}}\ ) = \text{ⓓ}$$

- A sign of the data to be evaluated is set for (s)[0]. If the evaluation value is a positive value, it stores 0, and if it is a negative value, it stores 1.
- The integer part and fractional part are stored in (s)[1] and (s)[2] respectively in BCD values. (Values between 0 and 1.0000 can be set.)
- The operation results stored to (d) are BCD values between 0 and 180 degrees (in degrees).
- The fractional part is rounded off for the operation result.

Operation error

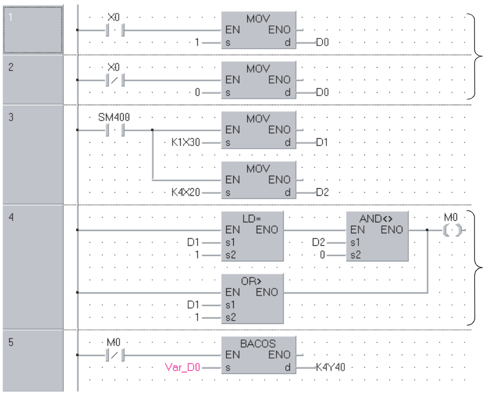
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is not a BCD value. The value specified for (s) is not in the range of -1.0000 and 1.0000.	—	○	○	○	○	○
4101	The device specified for (s) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the arccosine function is evaluated on the data in which X0 with positive and negative signs (positive when X0 is OFF, and negative when X0 is ON), the integer part in BCD 1 digit from X30 to X33 and the fractional part in BCD 4 digit from X20 to X2F, and the evaluated angle is output in BCD 4 digits to the devices from Y40 to Y4F.

[Structured ladder/FBD]



Sets the sign of the cosine value. (①)

Sets the integer part of the cosine value. (②)

Sets the fractional part of the cosine value. (③)

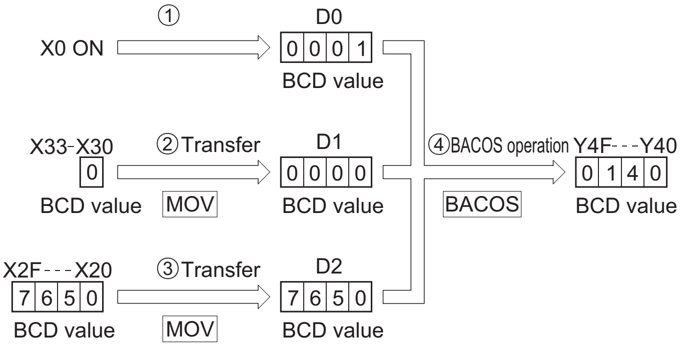
Turns M0 ON when D1>1, or D1=1 and D2≠0.

Evaluates the arccosine function and outputs the operation result to the devices from Y40 to Y4F. (④)

[ST]

```
MOV(X0,1,D0);
MOV(NOT(X0),0,D0);
D1: =K1X30;
D2: =K4X20;
OUT((D1=1 AND D2<>0) OR D1>1,M0);
BACOS(NOT(M0),Var_D0,K4Y40);
```

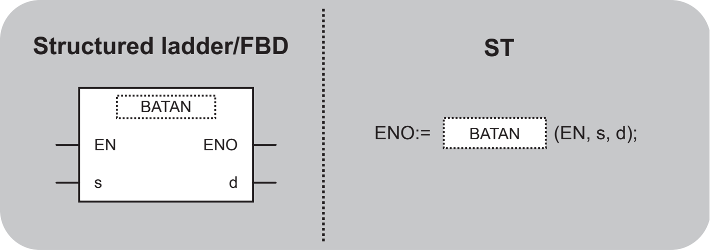
[Operation example when the value 0.7650 is specified for X0 and X20 to X33]



TAN^{-1} operation on data in BCD format

BATAN(P)

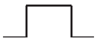
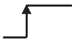
Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

BATAN, BATANP

■Executing condition

Instruction	Executing condition
BATAN	
BATANP	

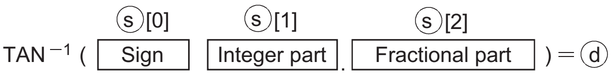
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Data on which the arctangent function is evaluated, or start number of the device that stores data	Array of ANY16 (1..3)
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the operation result	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				—	—
(d)	○	○		○				—	—

Processing details

- Evaluates the arctangent function on the value specified for (s) and stores the operation result (angle) to the device specified for (d).



- A sign of the data to be evaluated is set for (s)[0]. If the evaluation value is a positive value, it stores 0, and if it is a negative value, it stores 1.
- The integer part and fractional part are stored in (s)[1] and (s)[2] respectively in BCD values. (Values between 0 and 9999.9999 can be set.)
- Operation results stored to (d) are BCD values between 0 and 90 degrees, and 270 to 360 degrees (in degrees).
- The fractional part is rounded off for the operation result.

Operation error

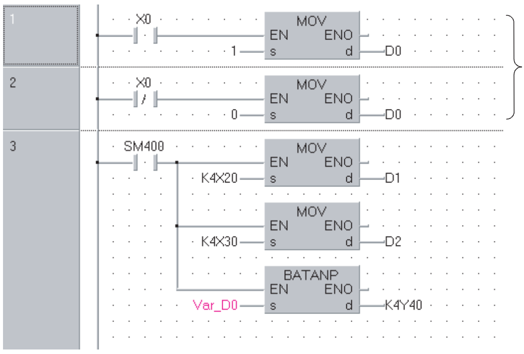
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value specified for (s) is not a BCD value.	—	○	○	○	○	○
4101	The device specified for (s) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the arctangent function is evaluated on the data in which X0 with positive and negative signs (positive when X0 is OFF, and negative when X0 is ON), the integer part in BCD 4 digits from X20 to X2F and the fractional part in BCD 4 digits from X30 to X3F, and the evaluated angle is output in BCD 4 digits to the devices from Y40 to Y4F.

[Structured ladder/FBD]



Sets the sign of the tangent value. (①)

Sets the integer part of the tangent value. (②)

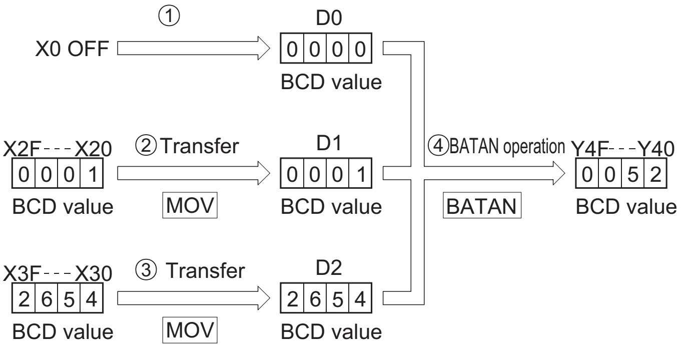
Sets the fractional part of the tangent value. (③)

Evaluates the arctangent function and outputs the operation result to the devices from Y40 to Y4F. (④)

[ST]

```
MOV(X0,1,D0);
MOV(NOT(X0),0,D0);
D1: =K4X20;
D2: =K4X30;
BATANP(TRUE,Var_D0,K4Y40);
```

[Operation example when the value 1.2654X0 is specified for X0 and X20 to X2F]



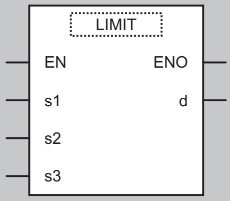
7.13 Data Control Instructions

Upper and lower limit controls of 16-/32-bit BIN data

LIMIT(_P), DLIMIT(_P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD

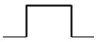
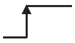


ST

ENO:= LIMIT (EN,s1, s2, s3, d);

Any of the following instruction can go in the dotted squares.
LIMIT, LIMITP, DLIMIT, DLIMITP

■Executing condition

Instruction	Executing condition
LIMIT, DLIMIT	
LIMITP, DLIMITP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Lower limit value (minimum output threshold value)	ANY16/32
	s2	Upper limit value (maximum output threshold value)	ANY16/32
	s3	Input value that controls the output value by upper and lower limit values	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Output value controlled by the upper and lower limit values	ANY16/32

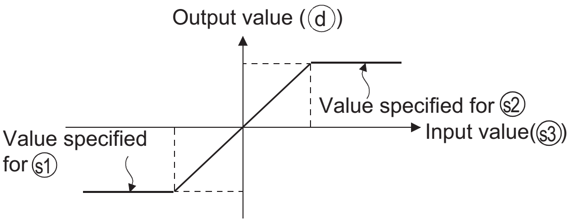
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(s3)	○							○	—
(d)	○							—	—

Processing details

■LIMIT(P)

- Controls the output value to be stored to the device specified for (d) by checking the input value (BIN 16 bits) specified for (s3) is within the range of upper and lower limit values specified for (s1) and (s2). Output value is controlled in the way shown below.

Condition	Result
Lower limit value (s1) > Input value (s3)	Lower limit value (s1) → Output value (d)
Upper limit value (s2) < Input value (s3)	Upper limit value (s2) → Output value (d)
Lower limit value (s1) ≤ Input value (s3) ≤ Upper limit value (s2)	Input value (s3) → Output value (d)

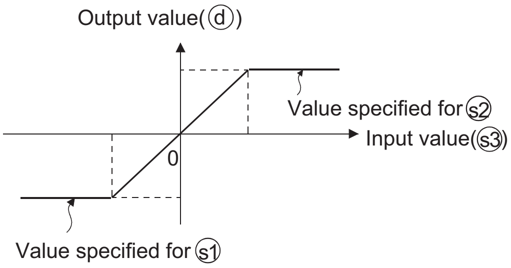


- Values between -32768 and 32767 can be specified for (s1), (s2), and (s3).
- To perform controls based only on the upper limit value, set the lower limit value specified for (s1) to -32678.
- To perform controls based only on the lower limit value, set the upper limit value specified for (s2) to 32767.

■DLIMIT(P)

- Controls the output value to be stored to the device specified for (d)by checking the input value (BIN 32 bits) specified for (s1)and (s2) is within the range of upper and lower limit values specified for (s3).

Condition	Result
Lower limit value ((s1)+1, (s1)) > Input value ((s3)+1, (s3))	Lower limit value ((s1)+1, (s1)) → Output value ((d)+1, (d))
Upper limit value ((s2)+1, (s2)) < Input value ((s3)+1, (s3))	Upper limit value ((s2)+1, (s2)) → Output value ((d)+1, (d))
Lower limit value ((s1)+1, (s1)) ≤ Input value ((s3)+1, (s3)) ≤ Upper limit value ((s2)+1, (s2))	Input value ((s3)+1, (s3)) → Output value ((d)+1, (d))



- Values between -2147483648 and 2147483647 can be specified for (s1), (s2), and (s3).
- To perform controls based only on the upper limit value, set the lower limit value specified for (s1) to -2147483648.
- To perform controls based only on the lower limit value, set the upper limit value specified for (s2) to 2147483647.

Operation error

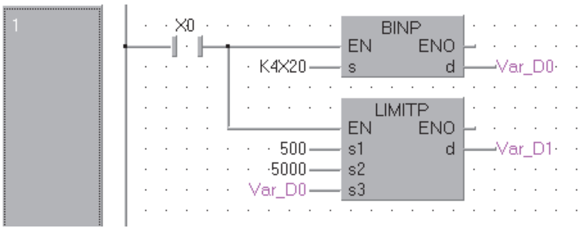
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The lower limit value specified for (s1) is larger than the upper limit value specified for (s2).	○	○	○	○	○	○

Program example

- In the following program, the limit control from 500 to 5000 is performed on the data set in BCD values from X20 to X2F when X0 turns ON, and the result is stored to Var_D1.

[Structured ladder/FBD]



[ST]

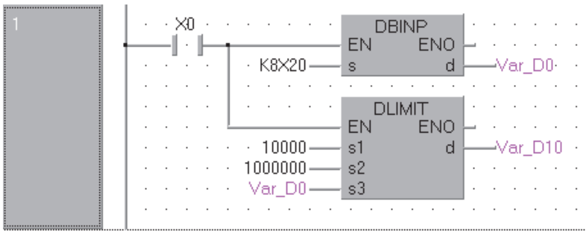
```
BINP(X0,K4X20,Var_D0);
LIMITP(X0,500,5000,Var_D0,Var_D1);
```

[Operation]

- Var_D1 becomes 500 if Var_D0 < 500. Example Var_D0 = 400 → Var_D1 = 500
- Var_D1 becomes the value of Var_D0 when 500 ≤ Var_D0 ≤ 5000. Example Var_D0 = 1300 → Var_D1 = 1300
- Var_D1 becomes 5000 when 5000 < Var_D0. Example Var_D0 = 9600 → Var_D1 = 5000

- In the following program, the limit control from 10000 to 1000000 is performed on the data set in BCD values from X20 to X3F when X0 turns ON, and the result is stored to Var_D10.

[Structured ladder/FBD]



[ST]

```
DBINP(X0,K8X20,Var_D0);
DLIMIT(X0,10000,1000000,Var_D0,Var_D10);
```

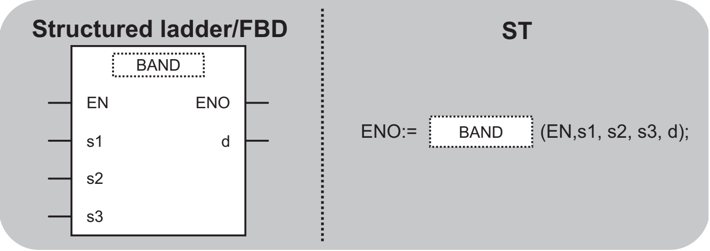
[Operation]

- Var_D10 becomes 10000 if Var_D0 are less than 10000. Example Var_D0 = 400 → Var_D10 = 10000
- Var_D10 becomes the value of Var_D0 if 10000 ≤ Var_D0 ≤ 1000000. Example Var_D0 = 345678 → Var_D10 = 345678
- Var_D10 becomes 1000000 if 1000000 < Var_D0. Example Var_D0 = 9876543 → Var_D10 = 1000000

Dead band control of 16-/32-bit BIN data

BAND(P), DBAND(P)

Basic High performance Process Redundant Universal LCPU

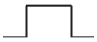
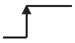


```
ENO:= BAND (EN,s1, s2, s3, d);
```

Any of the following instruction can go in the dotted squares.

BAND, BANDP, DBAND, DBANDP

■Executing condition

Instruction	Executing condition
BAND, DBAND	
BANDP, DBANDP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Lower limit value of the dead band (No output band)	ANY16/32
	s2	Upper limit value of the dead band (No output band)	ANY16/32
	s3	Input value that controls the output value by the dead band control values	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Output value controlled by the dead band control values	ANY16/32

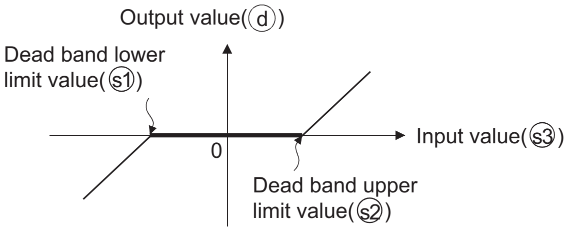
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(s3)	○							○	—
(d)	○							—	—

Processing details

■BAND(P)

- Controls the output value to be stored to the device specified for (d) by checking the input value (BIN 16 bits) specified for (s3) is within the range of dead band upper and lower limit values specified for (s1) and (s2). Output value is controlled in the way shown below.

Condition	Result
Lower limit value (s1) > Input value (s3)	Input value (s3) - Lower limit value (s1) → Output value (d)
Upper limit value (s2) < Input value (s3)	Input value (s3) - Upper limit value (s2) → Output value (d)
When Lower limit value (s1) ≤ Input value (s3) ≤ Upper limit value (s2)	0 → Output value (d)



- Values between -32768 and 32767 can be specified for (s1), (s2), and (s3).
- The output value stored to (d) is a signed 16-bit BIN value. Therefore, if the operation results exceed the range of -32768 to 32767, the following will take place.

Ex.

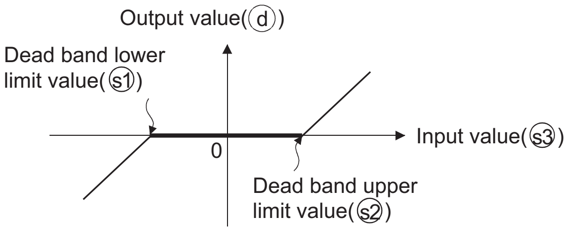
When the dead band lower limit value (s1) is 10 and the input value (s3) is -32768:

Output value = -32768 - 10 = 8000H - AH - 7FF6H = 32758

■DBAND(P)

- Controls the output value to be stored to the device specified for (d) by checking the input value (BIN 32 bits) specified for (s3) is within the range of dead band upper and lower limit values specified for (s1) and (s2). Output value is controlled in the way shown below.

Condition	Result
Lower limit value ((s1)+1, (s1)) > Input value ((s3)+1, (s3))	Input value ((s3)+1, (s3)) - Lower limit value ((s1)+1, (s1)) → Output value ((d)+1, (d))
Upper limit value ((s2)+1, (s2)) < Input value ((s3)+1, (s3))	Input value ((s3)+1, (s3)) - Upper limit value ((s2)+1, (s2)) → Output value ((d)+1, (d))
Lower limit value ((s1)+1, (s1)) ≤ Input value ((s3)+1, (s3)) ≤ Upper limit value ((s2)+1, (s2))	0 → Output value ((d)+1, (d))



- Values between -2147483648 and 2147483647 can be specified for (s1), (s2), and (s3).
- The output value stored to (d) is a signed 32-bit BIN value. Therefore, if the operation results exceed the range of -2147483648 to 2147483647, the following takes place.

Ex.

When the dead band lower limit value ((s1), (s1)+1) is 1000 and the input value ((s3), (s3)+1) is -2147483648

Output value = -2147483648 - 1000 = 80000000H-000003E8H = 7FFFFC18H = 2147482648

Operation error

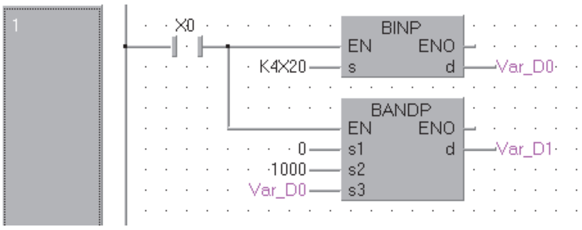
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The lower limit value specified for (s1) is greater than the upper limit value specified for (s2).	○	○	○	○	○	○

Program example

- In the following program, the dead band control applying the lower and upper limits from 0 to 1000 is performed on the data set in BCD value at X20 to X2F when X0 turns ON, and the result is stored to Var_D1.

[Structured ladder/FBD]

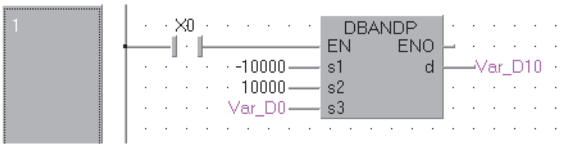


[ST]
BINP(X0,K4X20,Var_D0);
BANDP(X0,0,1000,Var_D0,Var_D1);

[Operation]
- 0 is stored to Var_D1 if 0 ≤ Var_D0 ≤ 1000. Example Var_D0 = 500 → Var_D1 = 0
- The value of (Var_D0) - 1000 is stored to Var_D1 if 1000 < Var_D0. Example Var_D0 = 7000 → Var_D1 = 6000

- In the following program, the dead band control applying the lower and upper limits from -10000 to 10000 is performed on the data set in BCD value at Var_D0 when X0 turns ON, and the result is stored to Var_D10.

[Structured ladder/FBD]



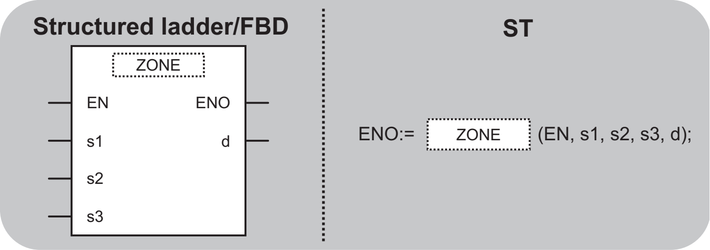
[ST]
DBANDP(X0,-10000,10000,Var_D0,Var_D10);

[Operation]
- The value Var_D0 - (-10000) is stored to Var_D10 if Var_D0 < (10000). Example Var_D0 = -12345 → Var_D10 = -2345
- The value 0 is stored to Var_D10 if -10000 ≤ Var_D0 ≤ 10000. Example Var_D0 = 6789 → Var_D10 = 0
- The value Var_D0 - 10000 is stored to Var_D10 if 10000 < Var_D0. Example Var_D0 = 50000 → Var_D10 = 40000

Zone control of 16-/32-bit BIN data

ZONE(P), DZONE(P)

Basic | High performance | Process | Redundant | Universal | LCPU

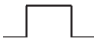
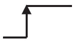


```
ENO:= ZONE (EN, s1, s2, s3, d);
```

Any of the following instruction can go in the dotted squares.

ZONE, ZONEP, DZONE, DZONEP

■Executing condition

Instruction	Executing condition
ZONE, DZONE	
ZONEP, DZONEP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Negative bias value that is added to the input value	ANY16/32
	s2	Positive bias value that is added to the input value	ANY16/32
	s3	Input value for executing the zone control	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Output value controlled by the zone control	ANY16/32

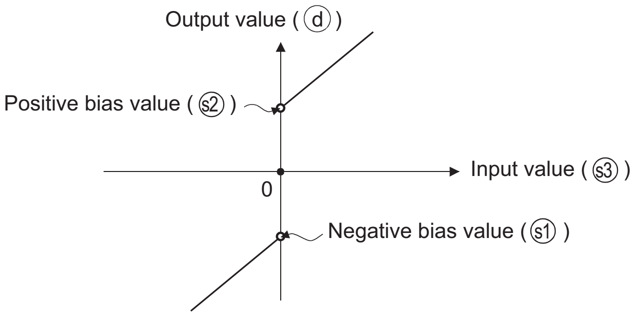
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	○							○	—
(s2)	○							○	—
(s3)	○							○	—
(d)	○							—	—

Processing details

■ZONE(P)

- Adds a bias value specified for (s1) or (s2) to an input value specified for (s3), and stores the result to the device specified for (d). Bias values are calculated in the way shown below.

Condition	Result
Input value (s3) < 0	Input value (s3) + Negative bias value (s1) → Output value (d)
Input value (s3) = 0	0 → Output value (d)
Input value (s3) > 0	Input value (s3) + Positive bias value (s2) → Output value (d)



- Values between - 32768 and 32767 can be specified for (s1), (s2), and (s3).
- The output value stored to (d) is a signed 16-bit BIN value. Therefore, if the operation results exceed the range of - 32768 to 32767, the following will take place.

Ex.

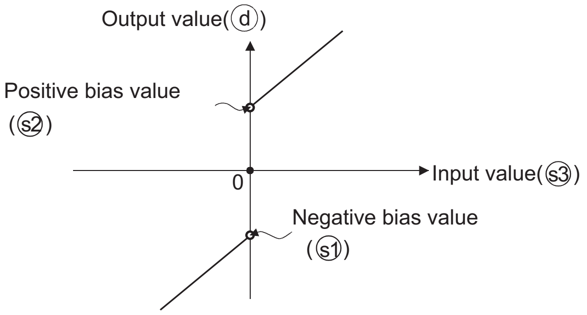
When negative bias value (s1) is -100 and input value (s3) is -32768

Output value = -32768 + (-100) = 8000H + FF9C = 7F9CH = 32668

■DZONE(P)

- Adds a bias value specified for (s1) or (s2) to an input value specified for (s3), and stores the result to the device specified for (d). Addition of the bias value is performed in the way shown below.

Condition	Result
Input value ((s3)+1, (s3)) < 0	Input value ((s3)+1, (s3)) + Negative bias value ((s1)+1, (s1)) → Output value ((d)+1, (d))
Input value ((s3)+1, (s3)) = 0	0 → Output value ((d)+1, (d))
Input value ((s3)+1, (s3)) > 0	Input value ((s3)+1, (s3)) + Positive bias value ((s2)+1, (s2)) → Output value ((d)+1, (d))



- Values between - 2147483648 and 2147483647 can be specified for (s1), (s2), and (s3).
- The value stored to (d) is a signed 32-bit BIN value. Therefore, if the operation results exceed the range of -2147483648 to 2147483647, the following takes place.

Ex.

When negative bias value ((s1), (s1)+1) is -1000 and input value ((s3), (s3)+1) is -2147483648

Output value -2147483648 + (-1000) = 80000000H + FFFFFC18H = 7FFFFC18 = 2147482648

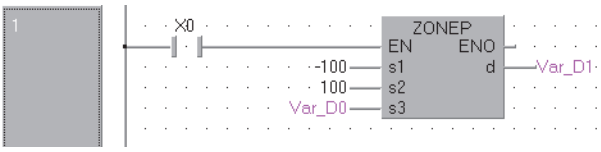
Operation error

- There is no operation error.

Program example

- In the following program, the zone control applying negative and positive bias values of -100 to 100 is performed on the data set for Var_D0 when X0 turns ON, and the result is stored to Var_D1.

[Structured ladder/FBD]



[ST]

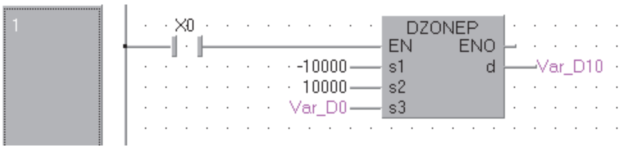
ZONEP(X0,-100,100,Var_D0,Var_D1);

[Operation]

- The value Var_D0 + (-100) is stored to Var_D1 if Var_D0 < 0. Example Var_D0 = -200 → Var_D1 = -300
- The value 0 is stored to Var_D1 if Var_D0 = 0.
- The value of Var_D0 + 100 is stored to Var_D1 if 0 < Var_D0. Example Var_D0 = 700 → Var_D1 = 800

- In the following program, the zone control applying negative and positive bias values of -10000 to 10000 is performed on the data set for Var_D0 when X1 turns ON, and the result is stored to Var_D10.

[Structured ladder/FBD]



[ST]

DZONEP(X1,-10000,10000,Var_D0,Var_D10);

[Operation]

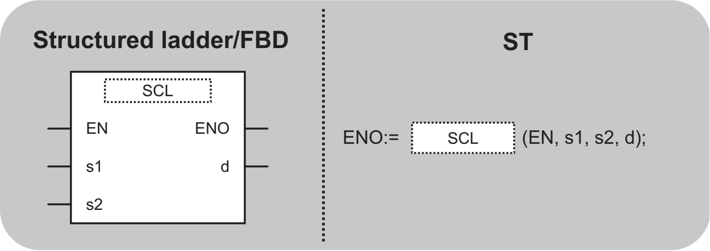
- The value Var_D1 + (-10000) is stored to Var_D10 if Var_D0 < 0. Example Var_D0 = -12345 → Var_D10 = -22345
- The value 0 is stored to Var_D10 if Var_D0 = 0.
- The value Var_D0 + 10000 is stored to Var_D10 if 0 < Var_D0. Example Var_D0 = 50000 → Var_D10 = 60000

Scaling (coordinate by point data)

SCL(P), DSCL(P)

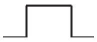
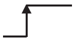
Basic ✗ High performance ✗ Process ✗ Redundant ✗ Ver. Universal LCPU

- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported



Any of the following instruction can go in the dotted squares.
SCL, SCLP, DSCL, DSCLP

■Executing condition

Instruction	Executing condition
SCL, DSCL	
SCLP, DSCLP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Input value used for scaling, or start number of the device that stores the input value	ANY16/32
	s2	Start number of the device that stores the conversion data for scaling	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the output value controlled by the scaling	ANY16/32

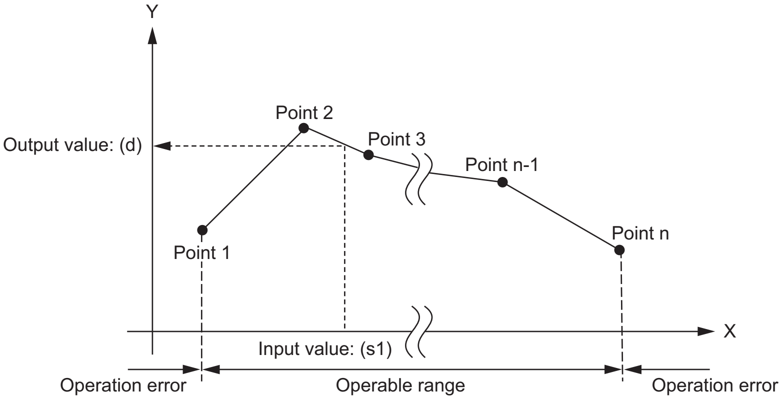
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	—	○	○	○				○	—
(s2)	—	○	○	—				—	—
(d)	—	○	○	○				—	—

Processing details

■SCL(P)

- Processes the scaling on the conversion data (unit of 16-bit data) specified for (s2) with the input value specified for (s1), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in (s2) and the following devices.

Setting item (n indicates the coordinate point specified for (s2).)		Device assignment
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

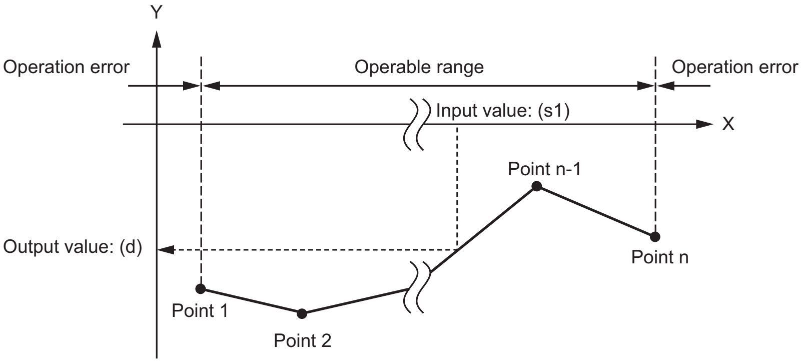


- If the result is not an integer value, the fractional part is rounded up.
- Set the X coordinate data of the conversion data for scaling in the ascending order.
- Set (s1) within the range of the conversion data for scaling (device value of (s2)).
- If the same X coordinate is specified by the multiple points, the value of Y coordinate with the highest point number is output. Set the number of coordinate points of the conversion data for scaling within 1 to 32767.

■DSCL(P)

- Processes the scaling on the conversion data (unit of 32-bit data) specified for (s2) with the input value specified for (s1), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in (s2) and the following devices.

Setting item (n indicates the coordinate point specified for (s2).)		Device assignment
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 result is not an integer value, the fractional part is rounded up.
- Set the X coordinate data of the conversion data for scaling in the ascending order.
- Set (s1) within the range of the conversion data for scaling (device value of (s2), (s2) +1).
- If the same X coordinate is specified by the multiple points, the value of Y coordinate with the highest point number is output.
- Set the number of coordinate points of the conversion data for scaling within 1 to 32767.

Precautions

- The search method differs by the ON/OFF status of SM750.

SM750	Search method	Range for number of searches
OFF	Linear search	1 ≤ number of comparisons ≤ 32767
ON	Binary search	1 ≤ number of comparisons ≤ 15

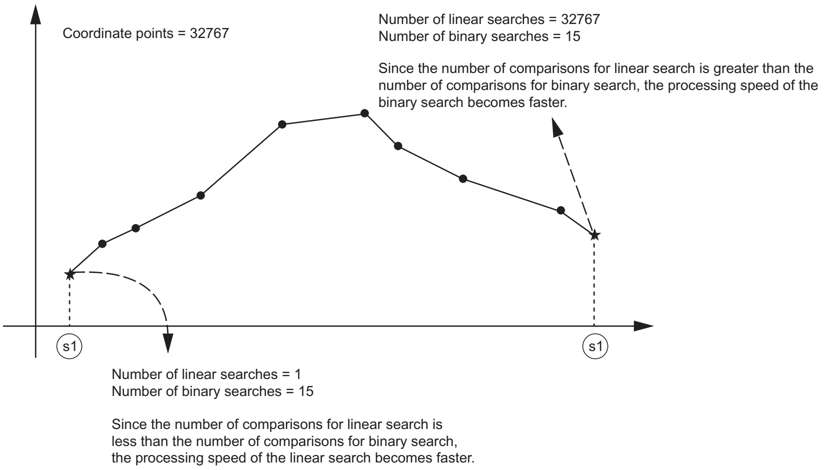
- When the conversion data for scaling are sorted in the ascending order, the search method differs by the SM750 status, and thus the processing speed also differs. The processing speed is fixed according to the number of comparisons, and the less comparison results faster processing speed.

- Faster processing speed for the linear search

With the maximum number of coordinate points, when (s1) is set between the coordinate points 1 and 15, the number of comparisons for linear search becomes less than/equal to 15, and thus the processing speed of the linear search becomes faster.

- Faster processing speed for the binary search

Since the maximum number of comparisons is 15, when (s1) is set to the coordinate point higher than 16, the number of comparisons for binary search becomes less than/equal to the number of comparisons for linear search, and thus the processing speed of the binary search becomes faster.


Coordinate points = 32767
Number of linear searches = 32767
Number of binary searches = 15
Since the number of comparisons for linear search is greater than the number of comparisons for binary search, the processing speed of the binary search becomes faster.
s1
s1
Number of linear searches = 1
Number of binary searches = 15
Since the number of comparisons for linear search is less than the number of comparisons for binary search, the processing speed of the linear search becomes faster.

Operation error

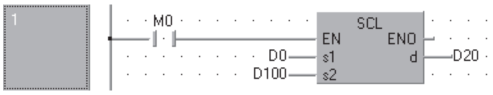
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The X coordinate data at the point in front of the conversion data for scaling (s1) are not set in the ascending order. (However, when SM750 is ON, this error is not detected.) The input value specified for (s1) is outside the range of set conversion data for scaling. The coordinate points from the device (s2) is outside the range of 1 to 32767.	—	—	—	—	○	○
4101	The coordinate points from the device (s2) is outside of the specified device range.	—	—	—	—	○	○

Program example

- In the following program, the scaling is processed on the conversion data set to D100 and the following devices with the input value specified for D0 when M0 turns ON, and the result is output to D20.

[Structured ladder/FBD]



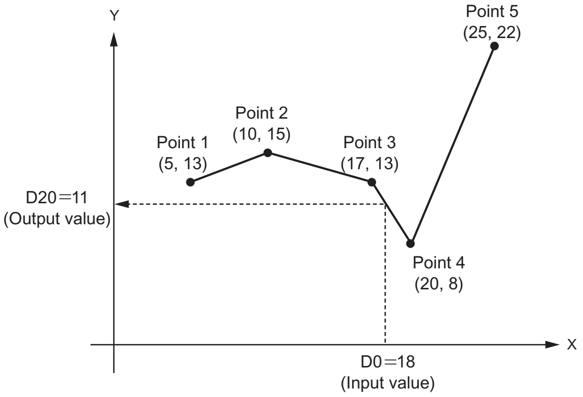
[ST]

SCL(M0,D0,D100,D20);

[Operation]

Configuration of conversion data for scaling

Setting item		Device	Setting content
Coordinate points		D100	K5
Point 1	X coordinate	D101	K5
	Y coordinate	D102	K13
Point 2	X coordinate	D103	K10
	Y coordinate	D104	K15
Point 3	X coordinate	D105	K17
	Y coordinate	D106	K13
Point 4	X coordinate	D107	K20
	Y coordinate	D108	K8
Point 5	X coordinate	D109	K25
	Y coordinate	D110	K22

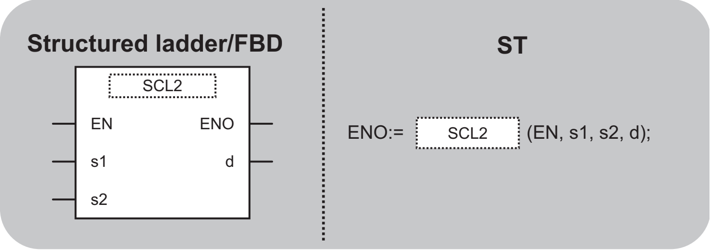


Scaling (coordinate by X/Y data)

SCL2(P), DSCL2(P)

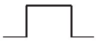
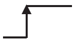
Basic High performance Process Redundant Ver. Universal LCPU

- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported



Any of the following instruction can go in the dotted squares.
SCL2, SCL2P, DSCL2, DSCL2P

■Executing condition

Instruction	Executing condition
SCL2, DSCL2	
SCL2P, DSCL2P	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Input value used for scaling, or start number of the device that stores the input value	ANY16/32
	s2	Start number of the device that stores the conversion data for scaling	ANY16/32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the output value controlled by the scaling	ANY16/32

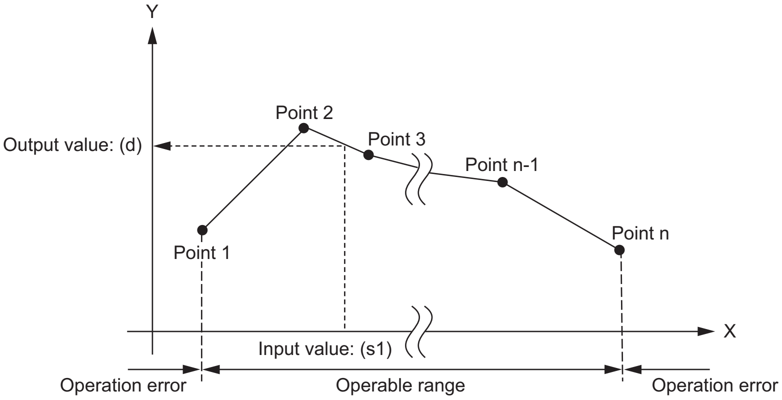
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	—	○	○	○				○	—
(s2)	—	○	○	—				—	—
(d)	—	○	○	○				—	—

Processing details

■SCL2(P)

- Processes the scaling on the conversion data (unit of 16-bit data) specified for (s2) with the input value specified for (s1), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in (s2) and the following devices.

Setting item (n indicates the coordinate point specified for (s2).)		Device assignment
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

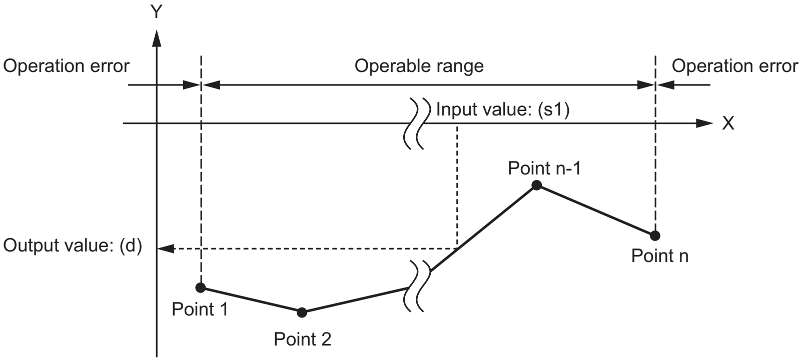


- If the result is not an integer value, the fractional part is rounded up.
- Set the X coordinate data of the conversion data for scaling in the ascending order.
- Set (s1) within the range of the conversion data for scaling (device value of (s2)).
- If the same X coordinate is specified by the multiple points, the value of Y coordinate with the highest point number is output.

■DSCL2(P)

- Processes the scaling on the conversion data (unit of 32-bit data) specified for (s2) with the input value specified for (s1), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in (s2) and the following devices.

Setting item (n indicates the coordinate point specified for (s2).)		Device assignment
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 result is not an integer value, the fractional part is rounded up.
- Set the X coordinate data of the conversion data for scaling in the ascending order.
- Set (s1) within the range of the conversion data for scaling (device value of (s2), (s2) +1).
- If the same X coordinate is specified by the multiple points, the value of Y coordinate with the highest point number is output.
- Set the number of coordinate points of the conversion data for scaling within 1 to 32767.

Point

When the conversion data for scaling are sorted in the ascending order, the search method differs by the SM750 status, and thus the processing speed also differs.
For details, refer to ☞ Page 656 Scaling (coordinate by point data).

Operation error

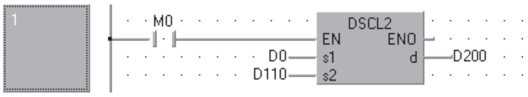
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The X coordinate data are not set in the ascending order. The input value specified for (s1) is outside the range of set conversion data for scaling. The coordinate points specified for (s2) is outside the range of 1 to 32767.	—	—	—	—	○	○
4101	The coordinate points from the device (s2) exceeds the specified device range.	—	—	—	—	○	○

Program example

- In the following program, the scaling is processed on the conversion data set to D110 and the following devices with the input value specified for D0 when M0 turns ON, and the result is output to D200.

[Structured ladder/FBD]



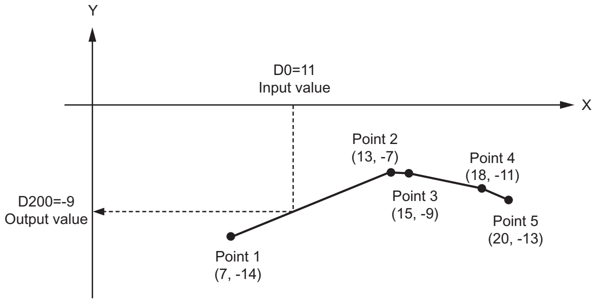
[ST]

DSCL2(M0,D0,D110,D200);

[Operation]

Configuration of conversion data for scaling

Setting item		Device	Setting content
Coordinate points		D110	K5
X coordinate	Point 1	D111	K7
	Point 2	D112	K13
	Point 3	D113	K15
	Point 4	D114	K18
	Point 5	D115	K20
Y coordinate	Point 1	D116	K-14
	Point 2	D117	K-7
	Point 3	D118	K-15
	Point 4	D119	K-11
	Point 5	D120	K-18



7.14 File Register Switching Instructions

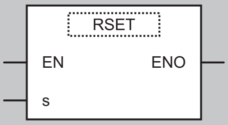
Switching file register block numbers

RSET(P)

Ver. Basic | High performance | Process | Redundant | Ver. Universal | LCPU

- Q00JCPU: Not supported
- Universal model QCPU: Not supported for Q00UJCPU

Structured ladder/FBD

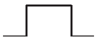
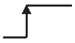


ST

ENO:= RSET (EN, s);

Any of the following instruction can go in the dotted squares.
RSET, RSETP

■Executing condition

Instruction	Executing condition
RSET	
RSETP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Block number data to be switched, or start device number that stores block number data	ANY16
Output argument	ENO	Execution result	Bit

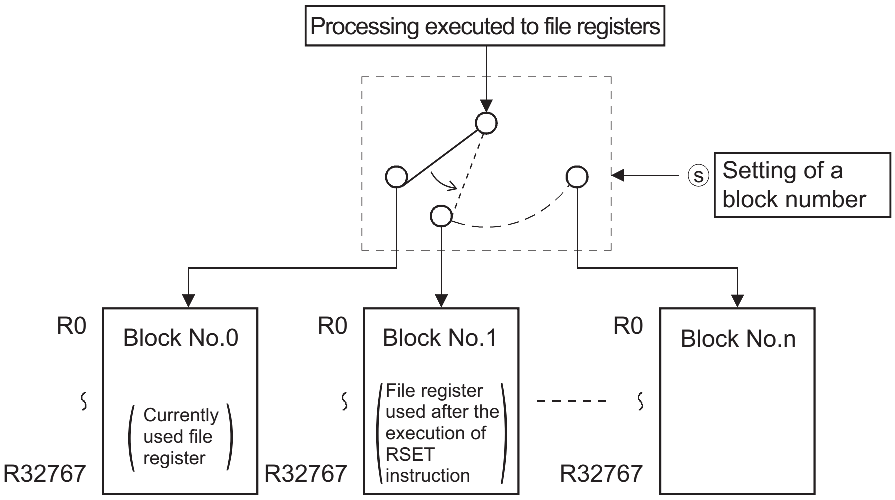
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○								—

Processing details

- Changes the file register block number used in the program to the block number stored in the device specified for (s). Following the block number change, all file registers used in the sequence program are processed to the file register of the block number after the change.

Ex.

When switching block number from block No. 0 to block No. 1


Processing executed to file registers
(s) Setting of a block number
R0
R32767
Block No.0
Currently used file register
Block No.1
File register used after the execution of RSET instruction
Block No.n

Point

Be cautious when a file register (R) is refreshed and the block No. of the file register is switched with the RSET instruction.

For the restrictions on file registers, refer to ☞ Page 78 Precautions on Using File Registers.

Operation error

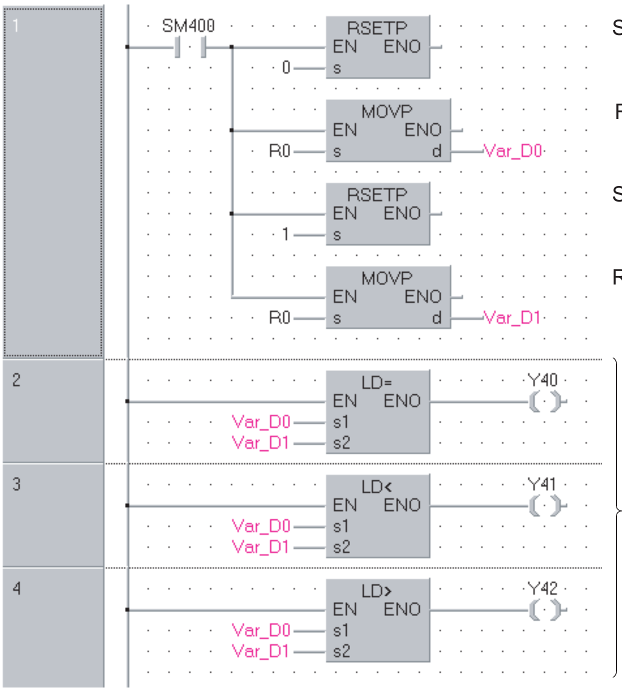
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The block number specified for (s) does not exist.	○	○	○	○	○	○
4101	There is no file register.	○	○	○	○	○	○

Program example

- In the following program, R0 of the block number 0 and the block number 1 are compared.

[Structured ladder/FBD]



Specifies the block No.0.

Reads out R0 of the block No.0.

Specifies the block No.1.

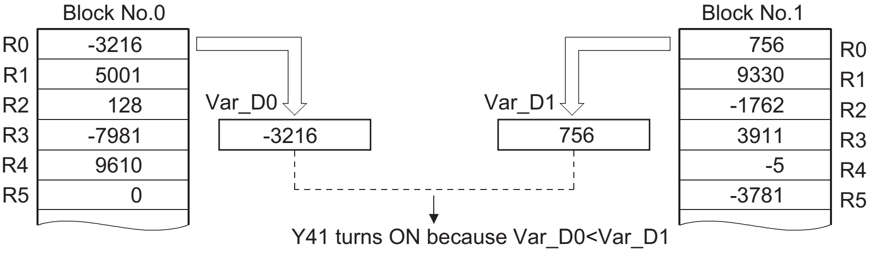
Reads out R0 of the block No.1.

Compares the read values.

[ST]

```
RSETP(SM400,0);
MOVP(SM400,R0,Var_D0);
RSETP(SM400,1);
MOVP(SM400,R0,Var_D1);
OUT(Var_D0=Var_D1,Y40);
OUT(Var_D0<Var_D1,Y41);
OUT(Var_D0>Var_D1,Y42);
```

[Operation]



	Block No.0
R0	-3216
R1	5001
R2	128
R3	-7981
R4	9610
R5	0

Block No.1	
756	R0
9330	R1
-1762	R2
3911	R3
-5	R4
-3781	R5

Y41 turns ON because Var_D0<Var_D1

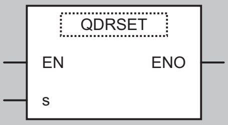
Setting file register files

QDRSET(P)

Basic High performance Process Redundant Ver. Universal LCPU

- Universal model QCPU: Not supported for Q00UJCPU

Structured ladder/FBD



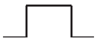
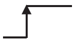
ST

```
ENO:= QDRSET (EN, s);
```

Any of the following instruction can go in the dotted squares.

QDRSET, QDRSETP

■Executing condition

Instruction	Executing condition
QDRSET	
QDRSETP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Character string data to be set with the driver number file name of file register, or start number of the device that stores character string data	String
Output argument	ENO	Execution result	Bit

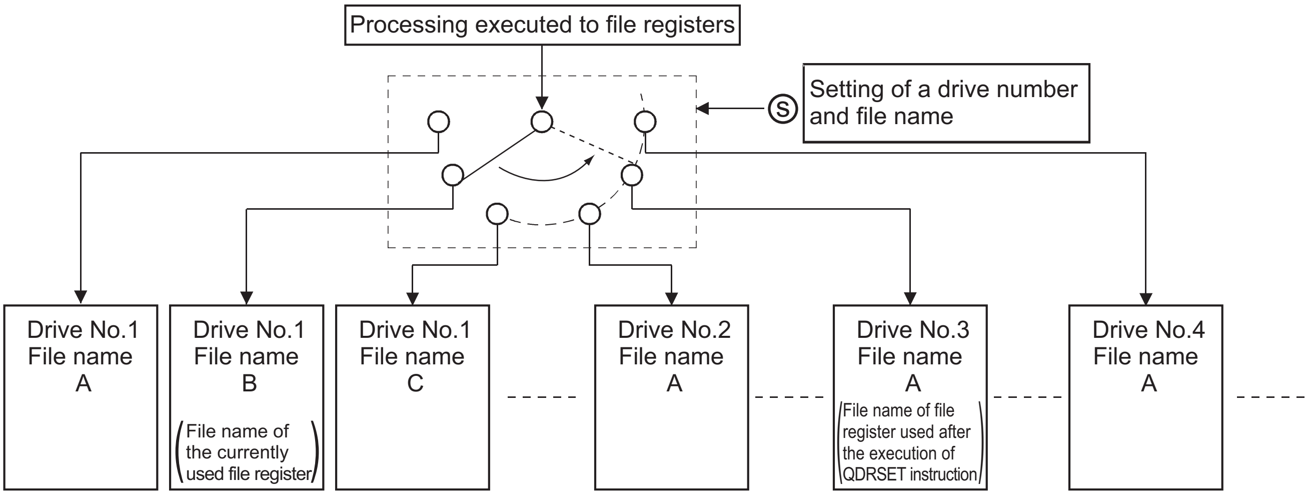
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—

Processing details

- Changes the file name of the file register used in the program to the file name being stored in the device specified for (s). After the file names have been changed, all the file registers being used by the sequence program process the file register of the block number 0 of the renamed file. Block number switch is performed by the RSET(P) instruction.

Ex.

When switching from Drive No. 1/File name B to Drive No. 3/File name A



Processing executed to file registers
Setting of a drive number and file name
(S)
Drive No.1 File name A
Drive No.1 File name B
File name of the currently used file register
Drive No.1 File name C
Drive No.2 File name A
Drive No.3 File name A
File name of file register used after the execution of QDRSET instruction
Drive No.4 File name A

- Drive number can be specified from 1 to 4. The drive number cannot be specified as drive 0 (program memory/internal memory).
- It is not necessary to specify the extension (.QDR) with the file name.
- A file name setting can be deleted by specifying the NULL character (00H) for the file name.
- File names specified with this instruction will be given priority even if a drive number and file name have been specified in the parameters.

Point

- If the file name is changed by the QDRSET(P) instruction, the file name returns to the name specified by the parameter when the CPU module is switched from STOP to RUN. To maintain the file name even after the CPU mode is changed from STOP to RUN, execute the QDRSET(P) instruction with the SM402 special relay, which turns ON during one scan when the CPU enters from STOP to RUN mode.
- Do not change the file name of the file register with the QDRSET(P) instruction when the file register is specified as the refresh device. For the restrictions on file registers, refer to ☞ Page 78 Precautions on Using File Registers.

Operation error

- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

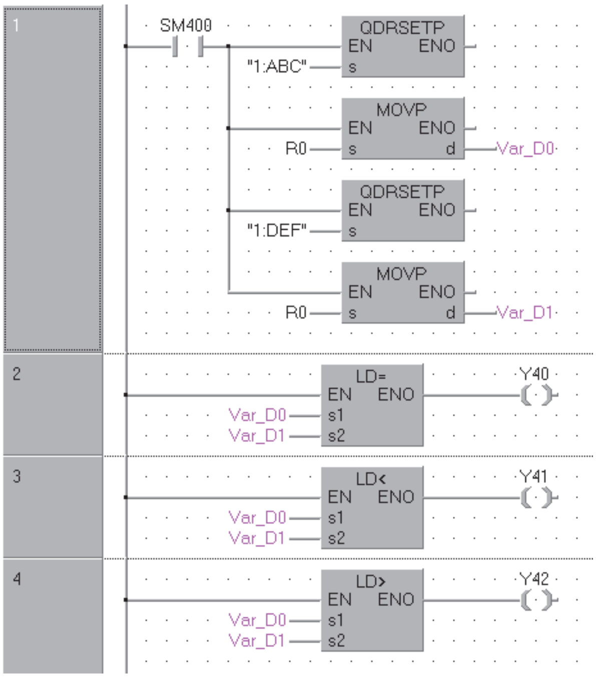
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2410	File name does not exist in the drive number specified for (s).	—	○	○	○	○	—
4100	The number other than 1 or 3 is specified into the drive No.	—	—	—	—	○[*1]	—
	The character string designated by (s) is only the drive No.	—	—	—	—	○[*1]	—
4101	The character string designated by (s) exceeds 16383 characters. "00H" does not exist at the device number or later designated by (s) in the range of the corresponding device.	—	—	—	—	○[*1]	—

*1 QnUDVCPU and QnUDPVCPU only

Program example

- In the following program, R0 of the drive number 1 "ABC" and the file name of the drive number 1 "DEF.QDR" are compared.

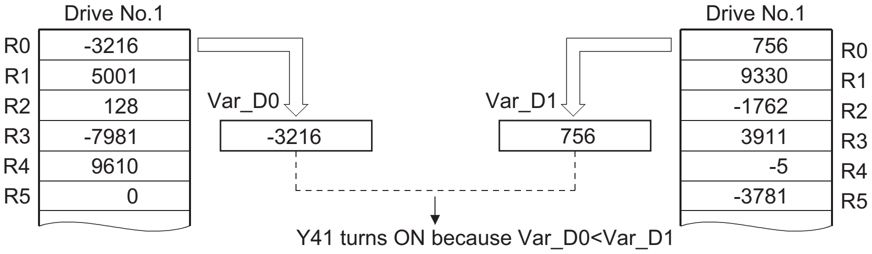
[Structured ladder/FBD]



[ST]

```
QDRSETP(X0,"1:ABC");
MOVP(X0,R0,Var_D0);
QDRSETP(X0,"1:DEF");
MOVP(X0,R0,Var_D1);
OUT(Var_D0=Var_D1,Y40);
OUT(Var_D0<Var_D1,Y41);
OUT(Var_D0>Var_D1,Y42);
```

[Operation]



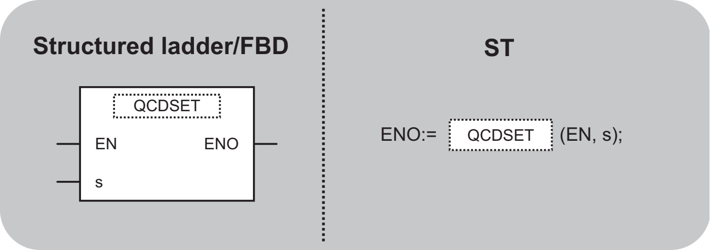
	Drive No.1
R0	-3216
R1	5001
R2	128
R3	-7981
R4	9610
R5	0

Drive No.1	
756	R0
9330	R1
-1762	R2
3911	R3
-5	R4
-3781	R5

Setting files for comments

QCDSET(P)

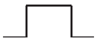
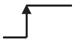
Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

QCDSET, QCDSETP

■Executing condition

Instruction	Executing condition
QCDSET	
QCDSETP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Character string data to be changed with the driver number file name of comment file, or start number of the device that stores character string data	String
Output argument	ENO	Execution result	Bit

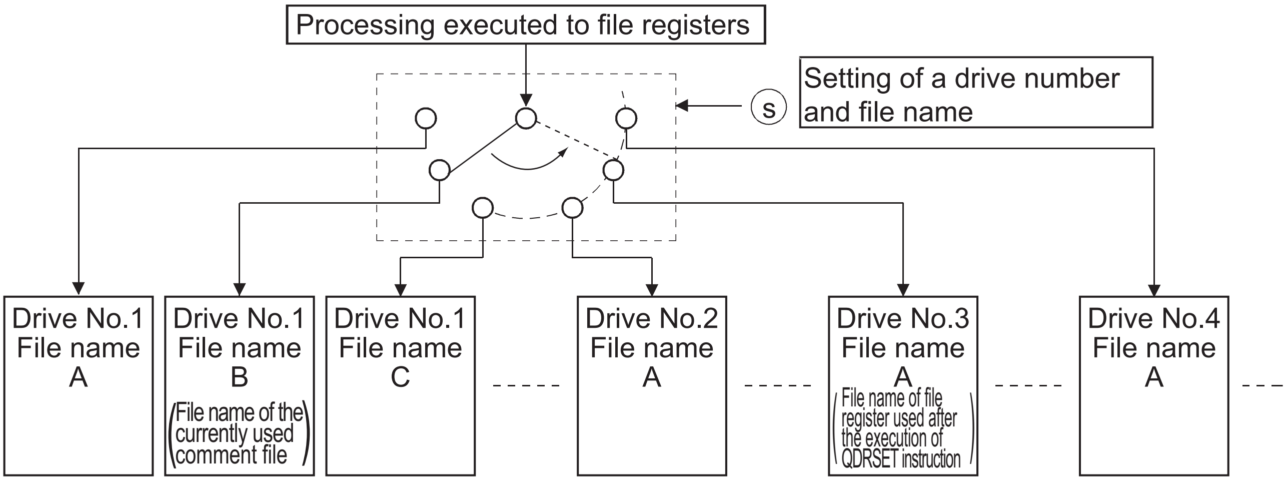
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—

Processing details

- Changes the file name of the file register used in the program to the file name being stored in the device specified for (s). After the file name change, comment data being used by the sequence program perform processing in relation to the comment data of the file name after the change.

Ex.

When switching from Drive No. 1/File name B to Drive No. 3/File name A


Processing executed to file registers
Setting of a drive number and file name
s
Drive No.1 File name A
Drive No.1 File name B
(File name of the currently used comment file)
Drive No.1 File name C
Drive No.2 File name A
Drive No.3 File name A
(File name of file register used after the execution of QDRSET instruction)
Drive No.4 File name A

- Drive number can be specified from 1 to 4. The drive number cannot be specified as drive 0 (program memory/internal memory). Drives that can be specified are different according to the CPU module. Check the drives that can be specified in the manual of the CPU module to be used.
- It is not necessary to specify the extension (.QCD) with the file name.
- A file name setting can be deleted by specifying the NULL character (00H) for the file name.
- File names specified with this instruction will be given priority even if a drive number and file name have been specified in the parameters.

Point

If the file name is changed by the QCDSET(P) instruction, the file name returns to the name specified by the parameter when the CPU module is switched from STOP to RUN.

To maintain the file name even after the CPU mode is changed from STOP to RUN, execute the QCDSET(P) instruction with the SM402 special relay, which turns ON during one scan when the CPU enters from STOP to RUN mode.

Operation error

- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2410	File name does not exist in the drive number specified for (s).	—	○	○	○	○	—

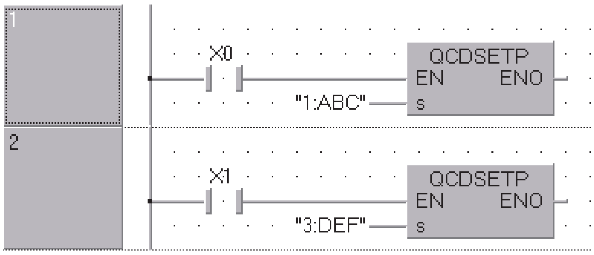
Precautions

- For Universal model QCPU and LCPU, this instruction is not executed while SM721 (File access in progress) is ON even when the execution command of this instruction is turned ON. Execute this instruction when SM721 is OFF.
- For High-speed Universal model QCPU, Universal model Process CPU, and LCPU, when specifying drive 2 (SD memory card) as drive No., this instruction will not be executed while SM606 (SD memory card forced disable instruction) is on. If executed, no processing is performed.

Program example

- In the following program, the object file of the comments are switched to the file names "ABC.QCD" in the drive number 1 when X0 turns ON, and to the file names "DEF.QCD" in the drive number 3 when X1 turns ON.

[Structured ladder/FBD]



Switches to ABC at drive No. 1.

Switches to DEF at drive No. 3.

[ST]

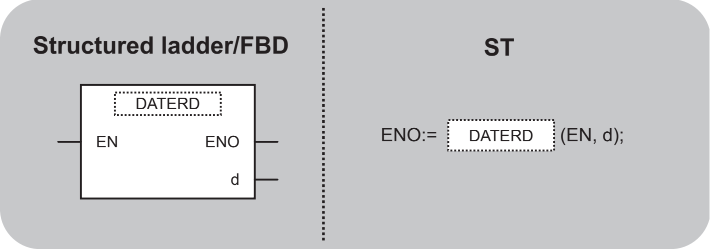
```
QCDSETP(X0,"1:ABC");
QCDSETP(X1,"3:DEF");
```

7.15 Clock Instructions

Reading clock data

DATERD(P)

Basic | High performance | Process | Redundant | Universal | LCPU

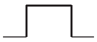
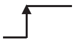


```
ENO:= DATERD (EN, d);
```

Any of the following instruction can go in the dotted squares.

DATERD, DATERDP

■Executing condition

Instruction	Executing condition
DATERD	
DATERDP	

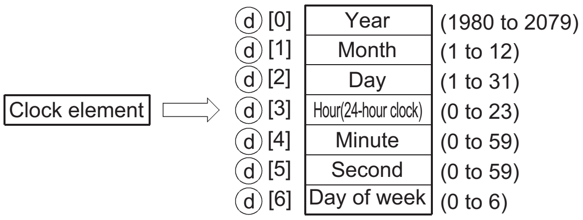
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Clock data that are read, or start number or the array of the device that stores clock data	Array of ANY16 (1..7)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(d)	—	○		—					

Processing details

- Reads year, month, day, minute, second, and day of week from the clock element of the CPU module and stores them in (d) and the following devices in BIN value.



Clock element

ⓓ [0]	Year	(1980 to 2079)
ⓓ [1]	Month	(1 to 12)
ⓓ [2]	Day	(1 to 31)
ⓓ [3]	Hour(24-hour clock)	(0 to 23)
ⓓ [4]	Minute	(0 to 59)
ⓓ [5]	Second	(0 to 59)
ⓓ [6]	Day of week	(0 to 6)

- The year in (d)[0] is stored as 4-digit year indication.
- The day of week in (d)[6] is stored as 0 to 6 to represent the days from Sunday to Saturday.

Day of week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Stored data	0	1	2	3	4	5	6

- Compensation is made automatically for leap years.

Operation error

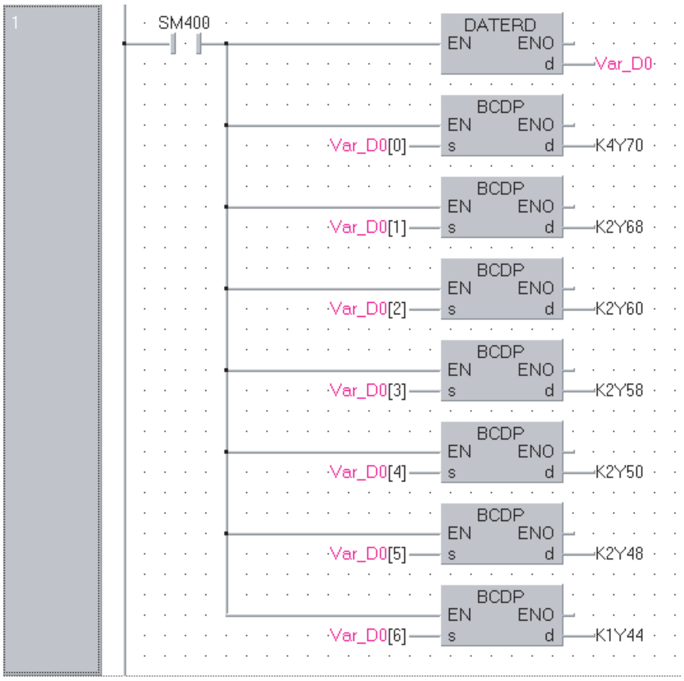
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the following clock data are output in BCD values.
  - YearY70 to Y7F
  - MonthY68 to Y6F
  - DayY60 to Y67
  - HourY58 to Y5F
  - MinuteY50 to Y57
  - SecondY48 to Y4F
  - Day of weekY44 to Y47

[Structured ladder/FBD]



Outputs "Year".

Outputs "Month".

Outputs "Day".

Outputs "Hour".

Outputs "Minute".

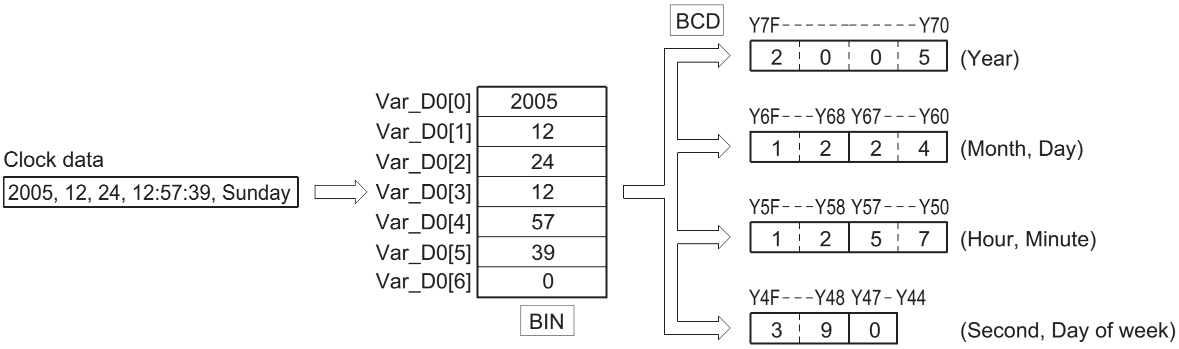
Outputs "Second".

Outputs "Day of week".

[ST]

```
DATERD(SM400,Var_D0);
BCDP(SM400,Var_D0[0],K4Y70);
BCDP(SM400,Var_D0[1],K2Y68);
BCDP(SM400,Var_D0[2],K2Y60);
BCDP(SM400,Var_D0[3],K2Y58);
BCDP(SM400,Var_D0[4],K2Y50);
BCDP(SM400,Var_D0[5],K2Y48);
BCDP(SM400,Var_D0[6],K1Y44);
```

[Operation]

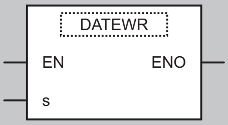


Writing clock data

DATEWR(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



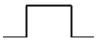
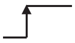
ST

ENO:= DATEWR (EN, s);

Any of the following instruction can go in the dotted squares.

DATEWR, DATEWRP

■Executing condition

Instruction	Executing condition
DATEWR	
DATEWRP	

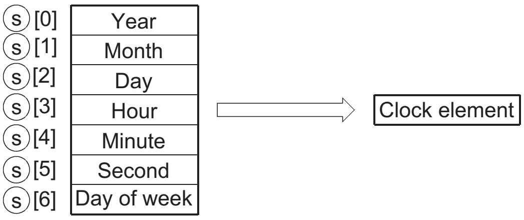
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number or the array of the device where the clock data to be written to the clock element is stored	Array of ANY16 (1..7)
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s)	—	○		—					

Processing details

- Writes clock data stored in (s) and the following devices to the clock element of the CPU module.



- Each item is set in BIN value.
- The year for (s)[0] is specified in 4-digit values between 1980 and 2079. (If a particular era name is used for year settings, leap years cannot be automatically adjusted.)
- (s)[1] specifies the month in values from 1 to 12 (January to December).
- (s)[2] specifies the day in values from 1 to 31.
- (s)[3] specifies the hour in values from 0 to 23 (using 24-hour clock, from 0 hours to 23 hundred hours). (Uses the 24-hour clock.)
- (s)[4] specifies the minute in values from 0 to 59.
- (s)[5] specifies the second in values from 0 to 59.
- (s)[6] specifies the day of week in values from 0 to 6 (Sunday to Saturday).

Day of week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Stored data	0	1	2	3	4	5	6

Operation error

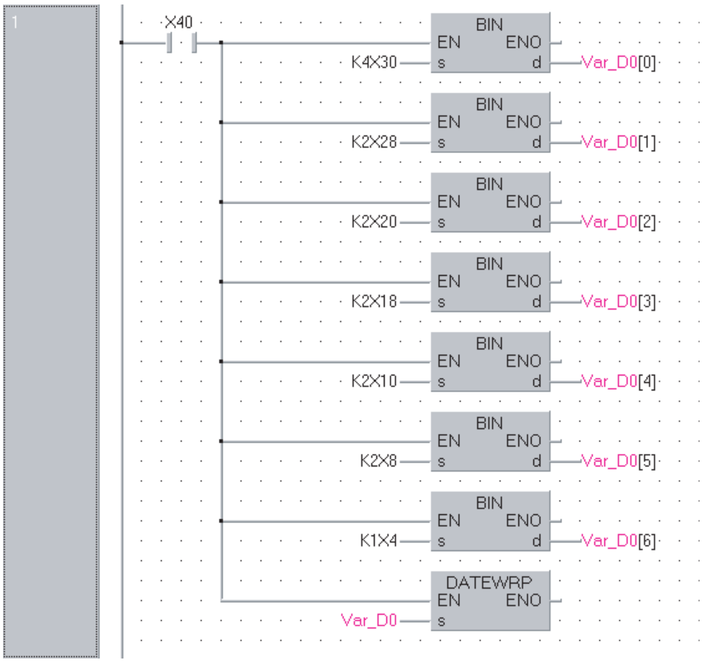
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	Each item of data have been set outside the setting range.	○	○	○	○	○	○
4101	The device specified for (s) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the following clock data entered in BCD value are written to the clock element when X40 turns ON.
  - YearX30 to X3F
  - HourX18 to X1F
  - MonthX28 to X2F
  - MinuteX10 to X17
  - DayX20 to X27
  - SecondX8 to XF
  - Day of weekX4 to X7

[Structured ladder/FBD]



Sets "Year".

Sets "Month".

Sets "Day".

Sets "Hour".

Sets "Minute".

Sets "Second".

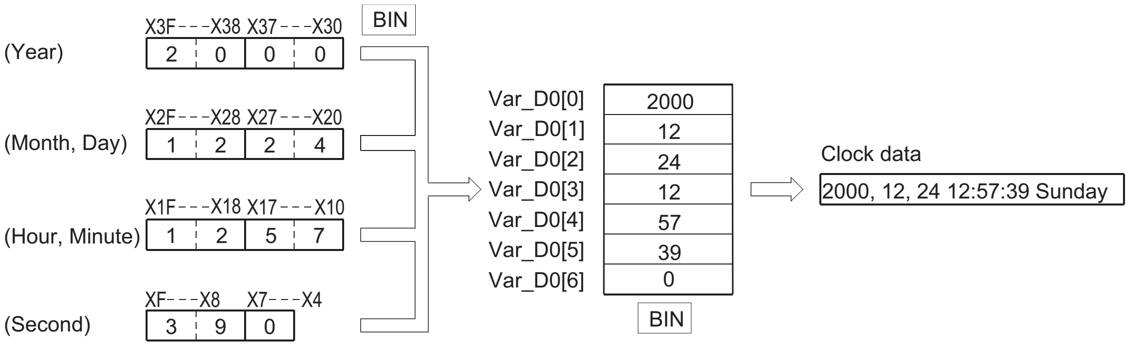
Sets "Day of week".

Writes the clock data
(Var_D0 to Var_D6)
to the clock element.

[ST]

```
BIN(X40,K4X30,Var_D0[0]);
BIN(X40,K2X28,Var_D0[1]);
BIN(X40,K2X20,Var_D0[2]);
BIN(X40,K2X18,Var_D0[3]);
BIN(X40,K2X10,Var_D0[4]);
BIN(X40,K2X8,Var_D0[5]);
BIN(X40,K1X4,Var_D0[6]);
DATEWRP(X40,Var_D0);
```

[Operation]

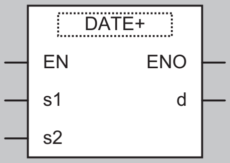


Clock data addition

DATE+(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



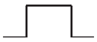
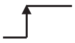
ST

Not supported

Any of the following instruction can go in the dotted squares.

DATE+, DATE+P

■Executing condition

Instruction	Executing condition
DATE+	
DATE+P	

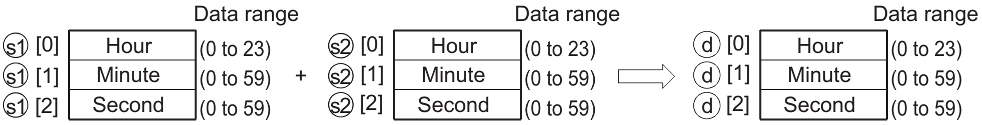
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number or the array of the device that stores clock (time) data to be added	Array of ANY16 (1..3)
	s2	Start number or the array of the device that stores added time data	Array of ANY16 (1..3)
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores clock (time) data of the addition result	Array of ANY16 (1..3)

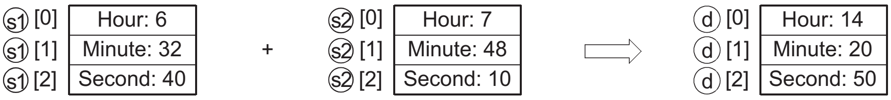
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—					
(s2)	—	○		—					
(d)	—	○		—					

Processing details

- Adds the time data specified for (s2) to the clock data specified for (s1), and stores the result to (d) and the following devices.



For example, adding the time 7:48:10 to 6:32:40 would result in the following operation.



- If the addition result of time exceeds 24 hours, 24 hours will be subtracted from the sum to make the final operation result. For example, if the time 20:20:20 were added to 14:20:30, the result would not be 34:40:50, but would instead be 10:40:50.

s1 [0]	Hour: 14		s2 [0]	Hour: 20		d [0]	Hour: 10
s1 [1]	Minute: 20	+	s2 [1]	Minute: 20	⇒	d [1]	Minute: 40
s1 [2]	Second: 30		s2 [2]	Second: 20		d [2]	Second: 50

Point

For further information regarding the data that can be set for hours, minutes, and seconds, refer to ☞ Page 676 Writing clock data.

Operation error

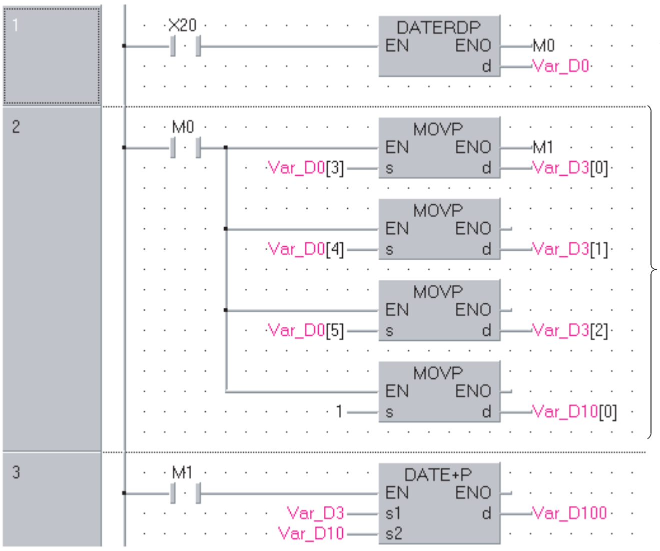
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value set for (s1) and (s2) is outside of the setting range.	○	○	○	○	○	○
4101	The device specified for (s1), (s2) or (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, 1 hour is added to the clock data read from the clock element when X20 turns ON, and the result is stored to Var_D100 and the following devices.

[Structured ladder/FBD]



Reads data of the clock element to Var_D0 and the following devices.

Sets the time to Var_D0 and the following devices.

[Operation]

- Clock data read by the DATERDP instruction.

Clock element ⇒

Var_D0[0]	1995	Year	
Var_D0[1]	5	Month	
Var_D0[2]	15	Day	
Var_D0[3]	10	Hour	
Var_D0[4]	23	Minute	Time data
Var_D0[5]	41	Second	
Var_D0[6]	2	Day of week	

- Addition by the DATE+P instruction.

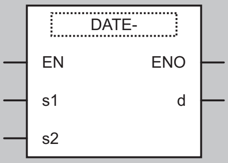
Var_D3[3]	Hour: 10		Var_D10[0]	Hour: 1		Var_D100[0]	Hour: 11
Var_D3[4]	Minute: 23	+	Var_D10[1]	Minute: 0	⇒	Var_D100[1]	Minute: 23
Var_D3[5]	Second: 41		Var_D10[2]	Second: 0		Var_D100[2]	Second: 41

Clock data subtraction

DATE-(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD



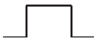
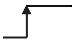
ST

Not supported

Any of the following instruction can go in the dotted squares.

DATE-, DATE-P

■Executing condition

Instruction	Executing condition
DATE-	
DATE-P	

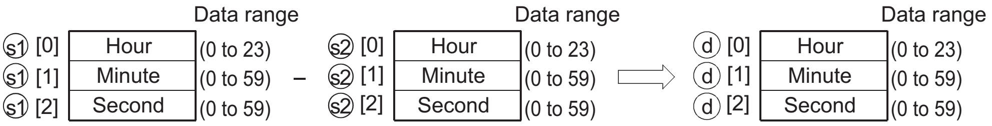
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number or the array of the device that stores clock (time) data to be subtracted	Array of ANY16 (1..3)
	s2	Start number or the array of the device that stores subtracted time data	Array of ANY16 (1..3)
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores clock (time) data of the subtraction result	Array of ANY16 (1..3)

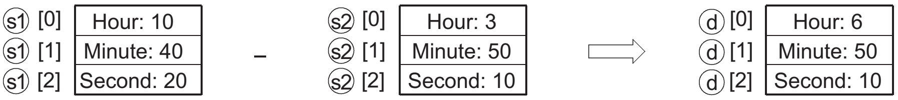
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—					
(s2)	—	○		—					
(d)	—	○		—					

Processing details

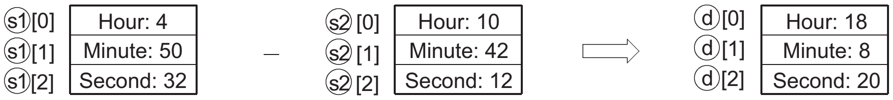
- Subtracts the time data specified for (s2) from the clock data specified for (s1), and stores the result to (d) and the following devices.



For example, if the clock time 3:50:10 were subtracted from the clock time 10:40:20, the operation would be performed as follows.



- If the operation results in a negative number, 24 will be added to the result to make a final operation result. For example, if the clock time 10:42:12 were subtracted from 4:50:32, the result would not be -6:8:20, but rather would be 18:8:20.



Point

For further information regarding the data that can be set for hours, minutes, and seconds, refer to ☞ Page 676 Writing clock data.

Operation error

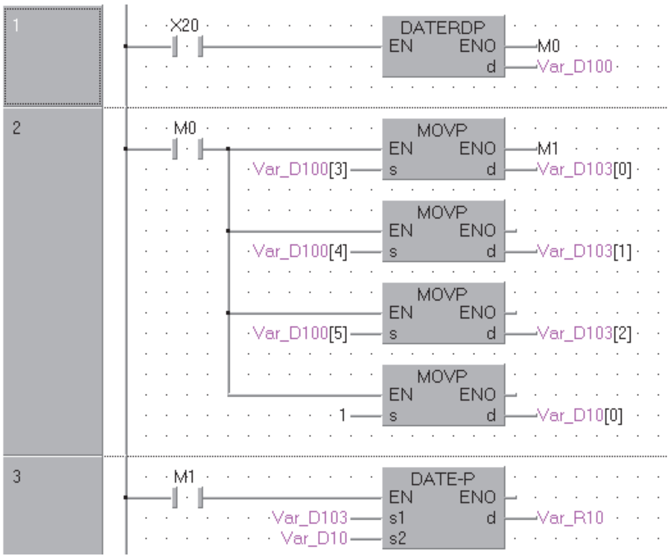
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value set for (s1) and (s2) is outside of the setting range.	○	○	○	○	○	○
4101	The device specified for (s1), (s2) or (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the time data stored in Var_D10 and the following devices are subtracted from the clock data read from the clock element when X1C turns ON, and the result is stored to Var_R10 and the following devices.

[Structured ladder/FBD]



[Operation]

- Clock data read by the DATERDP instruction.

Clock element ⇨	Var_D100[0]	1995	Year	
	Var_D100[1]	4	Month	
	Var_D100[2]	20	Day	
	Var_D100[3]	3	Hour	Time data
	Var_D100[4]	21	Minute	
	Var_D100[5]	20	Second	
	Var_D100[6]	1	Day of week	

- Subtraction by the DATE- P instruction (when 10 hours, 40 minutes, and 10 seconds have been specified for Var_D10).

Var_D103[0]	Hour: 3		Var_D10[0]	Hour: 10		Var_R10[0]	Hour: 16
Var_D103[1]	Minute: 21	—	Var_D10[1]	Minute: 40	⇨	Var_R10[1]	Minute: 41
Var_D103[2]	Second: 20		Var_D10[2]	Second: 10		Var_R10[2]	Second: 10

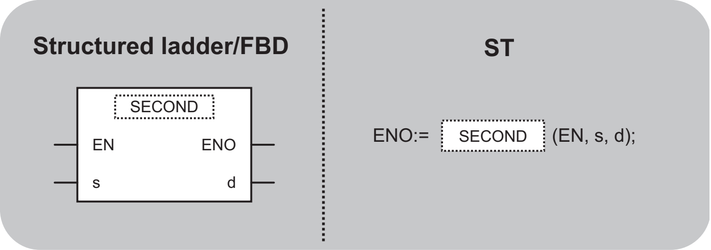
3:21:20 - 10:40:10 ⇨ -8:41:10 ⇨ 16:41:10

24 is added to this value

Time data conversion (hour/minute/second format to seconds)

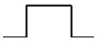
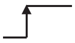
SECOND(P)

Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.
SECOND, SECONDP

■Executing condition

Instruction	Executing condition
SECOND	
SECONDP	

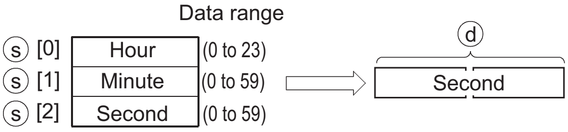
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number or the array of the device that stores clock data before the conversion	Array of ANY16 (1..3)
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores clock data after the conversion	ANY32

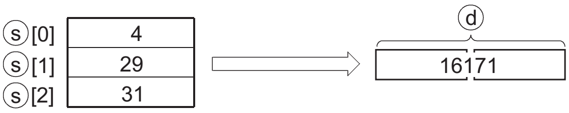
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				—	
(d)	○	○		○				—	

Processing details

- Converts the time data stored in (s) and the following devices to the data in seconds and stores the conversion result to the device specified for (d).



For example, if the value were 4 hours, 29 minutes and 31 seconds, the conversion operation would be performed as follows.



Operation error

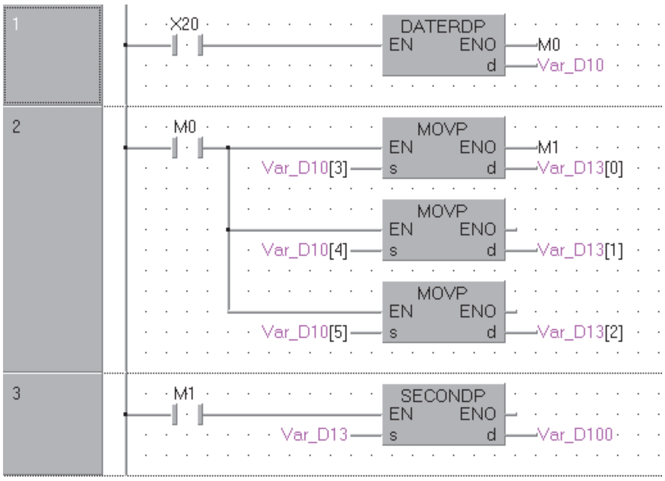
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value set for (s) is outside of the setting range.	○	○	○	○	○	○
4101	The device specified for (s) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the clock data read from the clock element are converted to the data in seconds when X20 turns ON, and the result is stored to Var_D100.

[Structured ladder/FBD]



[ST]

```
DATERDP(X20,Var_D10);
MOVP(X20,Var_D10[3],Var_D13[0]);
MOVP(X20,Var_D10[4],Var_D13[1]);
MOVP(X20,Var_D10[5],Var_D13[2]);
SECONDP(X20,Var_D13,Var_D100);
```

[Operation]

- Clock data read by the DATERDP instruction.

Clock element ⇨

Var_D10[0]	1995	Year	
Var_D10[1]	4	Month	
Var_D10[2]	20	Day	
Var_D10[3]	20	Hour	Time data
Var_D10[4]	21	Minute	
Var_D10[5]	23	Second	
Var_D10[6]	5	Day of week	

- Conversion to seconds by the SECONDP instruction.

Var_D13[0]	20
Var_D13[1]	21
Var_D13[2]	23

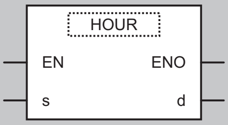
⇨ Var_D100 | 78238

Time data conversion (seconds to hour/minute/second format)

HOUR(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD

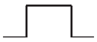
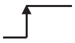


ST

ENO:= HOUR (EN, s, d);

Any of the following instruction can go in the dotted squares.
HOUR, HOURP

■Executing condition

Instruction	Executing condition
HOUR	
HOURP	

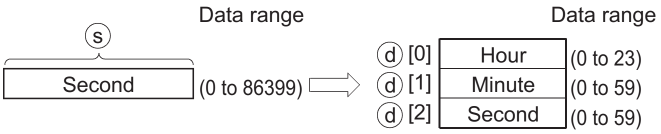
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores clock data before the conversion	ANY32
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores clock data after the conversion	Array of ANY16 (1..3)

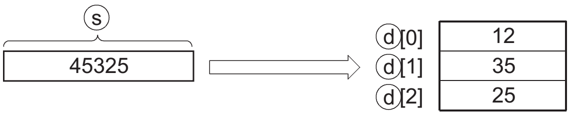
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○	○		○				○	—
(d)	—	○		—				—	—

Processing details

- Converts the data in seconds stored in the device specified for (s) to an hour/minute/second format, and stores the conversion result to (d) and the following devices.



For example, if 45325 seconds were the value specified, the conversion operation would be performed as follows.



Operation error

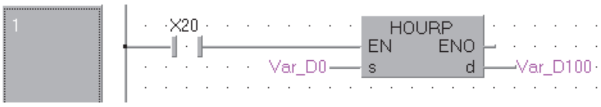
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value set for (s) is outside of the setting range.	○	○	○	○	○	○
4101	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the value in seconds stored in Var_D0 is converted to Hour/Minute/Second format when X20 turns ON, and the result is stored to Var_D100 and the following devices.

[Structured ladder/FBD]

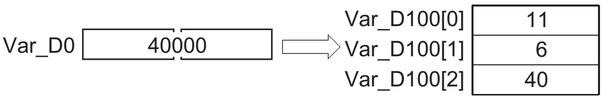


[ST]

HOURP(X20,Var_D0,Var_D100);

[Operation]

- Conversion to Hour/Minute/Second format by the HOURP instruction (when the value 40000 seconds has been specified for Var_D0).

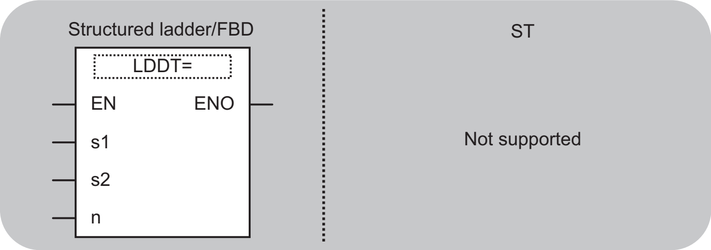


Date data comparison

LDDT□, ANDDT□ ORDT□

Basic ✗ High performance ✗ Process ✗ Redundant ✗ Ver. Universal LCPU

- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported



Any of the following instruction can go in the dotted squares.

LDDT=, ANDDT=, ORDT=, LDDT<>, ANDDT<>, ORDT<>, LDDT<=, ANDDT<=, ORDT<=, LDDT<, ANDDT<, ORDT<, LDDT>=, ANDDT>=, ORDT>=, LDDT>, ANDDT>, ORDT>

■Executing condition

Instruction	Executing condition
LDDT□, ANDDT□, ORDT□	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores the data to be compared	ANY16
	s2	Start number of the device that stores the data to be compared	ANY16
Output argument	ENO	Execution result	Bit
	n	Value to indicate the comparison target, or number of data to which the comparison target is stored	ANY16

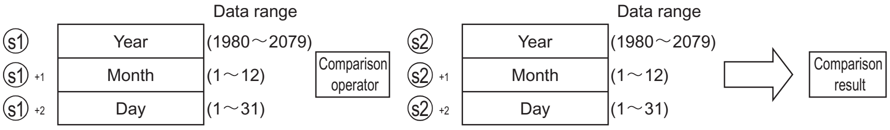
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				—	—
(s2)	—	○		—				—	—
n	—	○		○				○	—

Processing details

- Compares the date data specified for (s1) and (s2), or compares the date data specified for (s1) and the current date data. The comparison target can be selected by specifying n.

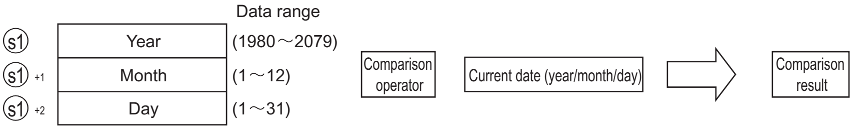
- Comparing with the specified date data

Compares the date data specified for (s1) and the date data specified for (s2) as a normally open contact according to the condition of n.



- Comparing with the current date data

Compares the date data specified for (s1) and the current date data as a normally open contact according to the condition of n.

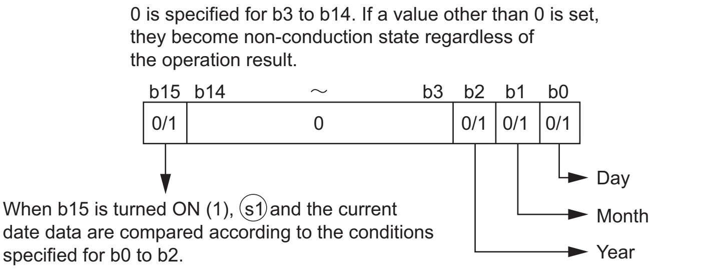


Point

When comparing with the specified date data or the current date data, if any of the following conditions apply to the setting of (s1) and (s2), an operation error (error code: 4101) or malfunction may occur.

- The index setting is specified exceeding the device range.
- A file register is specified without setting file registers.

- Each item is set in BIN value.
- The year for (s1) and (s2) is specified in 4-digit values between 1980 and 2079.
- The month for (s1) +1 and (s2) +1 is specified in values between 1 and 12 (January to December).
- The day for (s1) +2 and (s2) +2 is specified in values between 1 and 31.
- The comparison target can be specified in detail by specifying following values as shown below. The following shows the bit configuration of n.


0 is specified for b3 to b14. If a value other than 0 is set, they become non-conduction state regardless of the operation result.
b15 b14 ～ b3 b2 b1 b0
0/1 0 0/1 0/1 0/1
Day
Month
Year
When b15 is turned ON (1), (s1) and the current date data are compared according to the conditions specified for b0 to b2.

- Comparison target date (b0 to b2)

0: No comparison is performed with the comparison target date data (year/month/day).

1: Compares with the comparison target date data (year/month/day).

- Comparison target (b15)

0: Compares the date data specified for (s1) and the date data specified for (s2).

1: Compares the date data specified for (s1) and the current date data. The date data specified for (s2) are ignored.

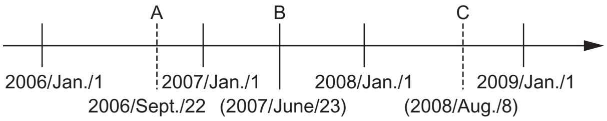
- The following table shows the bit processing of the comparison target.

The value of n for the comparison with the specified date data	The value of n for the comparison with the current date data	Comparison target date	Description
0001H	8001H	Day	Compares the day ((s1)+2) only.
0002H	8002H	Month	Compares the month ((s1)+1) only.
0003H	8003H	Month, Day	Compares the month ((s1)+1) and the day ((s1)+2).
0004H	8004H	Year	Compares the year ((s1)) only.
0005H	8005H	Year, Day	Compares the year ((s1)) and the day ((s1)+2).
0006H	8006H	Year, Month	Compares the year ((s1)) and the month ((s1)+1).
0007H	8007H	Year, Month, Day	Compares the year ((s1)), the month ((s1)+1), and the day ((s1)+2).
Other than 0001H to 0007H, 8001H to 8007H		None	No comparison operation on the year ((s1)), the month ((s1)+1), and the day ((s1)+2). (non-conduction state)

- If the data stored in the comparison target device are not recognized as date data, SM709 turns ON after the instruction execution and the device becomes a non-conduction state. Even when the date are not recognized as date data, if they are within the setting rage, SM709 does not turn ON. In addition, when the value of (s1) to (s1) +2 or (s2) to (s2) +2 exceeds the specified device range, SM709 turns ON and it becomes a non-conduction state. Note that, once SM709 turns ON, it stays ON until the reset or power OFF of CPU. Turn it OFF as required.
- The following table shows the comparison result of each instruction.

Instruction symbol	Condition	Comparison result	Condition	Comparison result
LDDT=, ANDDT=, ORDT=	(s1) = (s2)	Conduction state	(s1) ≠ (s2)	Non-conduction state
LDDT<>, ANDDT<>, ORDT<>	(s1) ≠ (s2)		(s1) = (s2)	
LDDT>, ANDDT>, ORDT>	(s1) > (s2)		(s1)≤(s2)	
LDDT<=, ANDDT<=, ORDT<=	(s1)≤(s2)		(s1) > (s2)	
LDDT<, ANDDT<, ORDT<	(s1) < (s2)		(s1)≥(s2)	
LDDT>=, ANDDT>=, ORDT>=	(s1)≥(s2)		(s1) < (s2)	

- The following shows the comparison example of date.



The following table shows the comparison result of date A, B, and C indicated above. Even when the data are compared under the same condition, the comparison result changes according to the specified comparison target.

Comparison target	Comparison condition		
	A < B	B < C	A < C
Day	○	×	×
Month	×	○	×
Year, Day	×	○	×
Year	○	○	○
Year, Day	○	○	○
Year, Month	○	○	○
Year, Month, Day	○	○	○
None	×	×	×

○: Conduction state ×: Non-conduction state

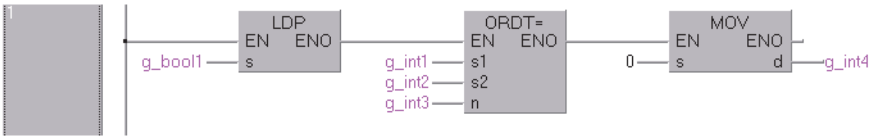
- With the date that do not actually exist, if they are settable date, they are compared according to the following conditions.

Date A: 2006/02/30 (The date does not actually exist, but it can be set.)
Date B: 2007/03/29
Date C: 2008/02/31 (The date does not actually exist, but it can be set.)

Comparison target	Comparison condition		
	A < B	B < C	A < C
Day	×	○	○
Month	○	×	×
Year, Day	○	×	○
Year	○	○	○
Year, Day	○	○	○
Year, Month	○	○	○
Year, Month, Day	○	○	○
None	×	×	×

○: Conduction state ×: Non-conduction state

- The ORDT=, ORDT<>, ORDT<=, ORDT<, ORDT>=, or ORDT> instruction performs the OR operation between the operation result of (s1), (s2) and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400. When using the ORDT=, ORDT<>, ORDT<=, ORDT<, ORDT>=, or ORDT> instruction, connect EN and ENO in series as shown below.



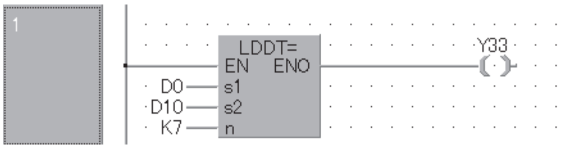
Operation error

- There is no operation error.

Program example

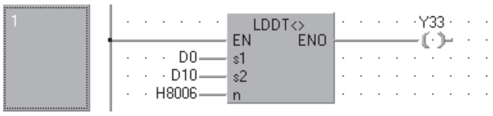
- In the following program, the data in D0 and the data in D10 (year/month/day) are compared, and Y33 turns ON if they are matched.

[Structured ladder/FBD]



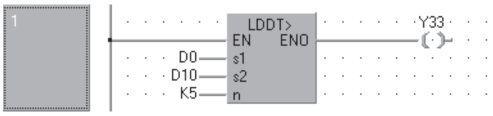
- In the following program, the data in D0 and the current date data (year/month) are compared when M0 turns ON, and Y33 turns ON if they are not matched.

[Structured ladder/FBD]



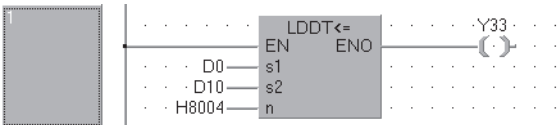
- In the following program, the data in D0 and the data in D10 (year/day) are compared when M0 turns ON, and Y33 turns ON if the value of data in D10 is less than the value of data in D0.

[Structured ladder/FBD]



- In the following program, the data in D0 and the current date data (year) are compared, and Y33 turns ON if the value of current date data is equal to/higher than the value of data in D0.

[Structured ladder/FBD]



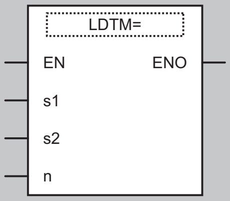
Time data comparison

LDTM□, ANDTM□, ORTM□

✕ Basic ✕ High performance ✕ Process ✕ Redundant Ver. Universal LCPU

- QnU(D)(H)CPU and QnUDE(H)CPU with a serial number (first five digits) of "10102" or later, Q00UJCPU, Q00UCPU, Q01UCPU, QnUDVCPU, and QnUDPVCPU: Supported

Structured ladder/FBD



ST

Not supported

Any of the following instruction can go in the dotted squares.

LDTM=, ANDTM=, ORTM=, LDTM<>, ANDTM<>, ORTM<>, LDTM<=, ANDTM<=, ORTM<=, LDTM<, ANDTM<, ORTM<, LDTM>=, ANDTM>=, ORTM>=, LDTM>, ANDTM>, ORTM>

■Executing condition

Instruction	Executing condition
LDTM□, ANDTM□, ORTM□	Non-conditional execution

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores the data to be compared	ANY16
	s2	Start number of the device that stores the data to be compared	ANY16
Output argument	ENO	Execution result	Bit
	n	Value to indicate the comparison target, or number of data to which the comparison target is stored	ANY16

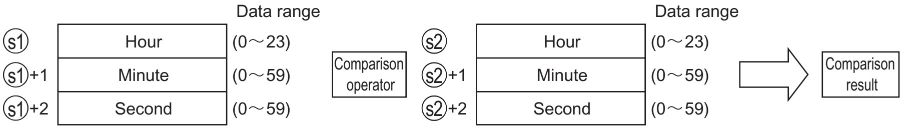
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				—	—
(s2)	—	○		—				—	—
n	—	○		○				○	—

Processing details

- Compares the time data specified for (s1) and (s2), or compares the time data specified for (s1) and the current time data. The comparison target can be selected by specifying n.

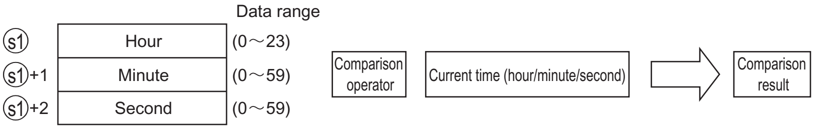
- Comparing with the specified time data

Compares the time data specified for (s1) and the time data specified for (s2) as a normally open contact according to the condition of n.



- Comparing with the current time data

Compares the time data specified for (s1) and the current time data as a normally open contact according to the condition of n.
The time data specified for (s2) are treated as dummy data and ignored.



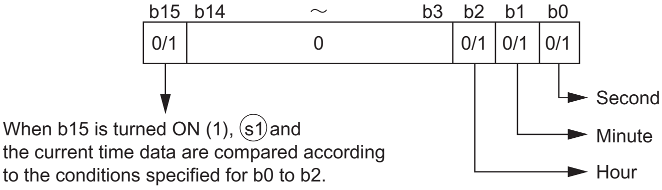
Point

When comparing with the specified time data or the current time data, if any of the following conditions apply to the setting of (s1) and (s2), an operation error (error code: 4101) or malfunction may occur.

- The index setting is specified exceeding the device range.
- A file register is specified without setting file registers.

- Each item is set in BIN value.
- The hour for (s1) and (s2) is specified in values between 0 and 23. (In the 24-hour clock)
- The minute for (s1) + 1 and (s2) + 1 is specified in values between 0 and 59.
- The second for (s1) + 2 and (s2) + 2 is specified in values between 0 and 59.
- The comparison target can be specified in detail by specifying following values as shown below. The following shows the bit configuration of n.

0 is specified for b3 to b14. If a value other than 0 is set, they become non-conduction state regardless of the operation result.



When b15 is turned ON (1), s1 and the current time data are compared according to the conditions specified for b0 to b2.

- Comparison target time (b0 to b2)

0: No comparison is performed with the comparison target time data (hour/minute/second).
1: Compares with the comparison target time data (hour/minute/second).

- Comparison target (b15)

0: Compares the time data specified for (s1) and the time data specified for (s2).
1: Compares the time data specified for (s1) and the current time data. The time data specified for (s2) are ignored.

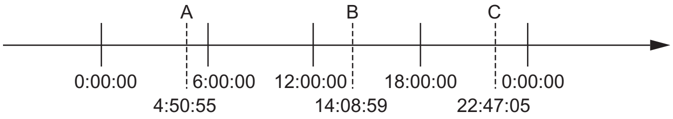
- The following table shows the bit processing of the comparison target.

The value of n for the comparison with the specified time data	The value of n for the comparison with the current time data	Comparison target time	Description
0001H	8001H	Second	Compares the second ((s1)+2) only.
0002H	8002H	Minute	Compares the minute ((s1)+1) only.
0003H	8003H	Minute, Second	Compares the minute ((s1)+1) and the second ((s1)+2).
0004H	8004H	Hour	Compares the hour ((s1)) only.
0005H	8005H	Hour, Second	Compares the hour ((s1)) and the second ((s1)+2).
0006H	8006H	Hour, Minute	Compares the hour ((s1)) and the minute ((s1)+1).
0007H	8007H	Hour, Minute, Second	Compares the hour ((s1)), the minute ((s1)+1), and the second ((s1)+2).
Other than 0001H to 0007H, 8001H to 8007H		None	No comparison operation on the hour ((s1)), the minute ((s1)+1), and the second ((s1)+2). (non-conduction state)

- If the data stored in the comparison target device are not recognized as time data, SM709 turns ON after the instruction execution and the device becomes a non-conduction state. In addition, when the value of (s1) to (s1)+2 or (s2) to (s2)+2 exceeds the specified device range, SM709 turns ON and it becomes a non-conduction state. Note that, once SM709 turns ON, it stays ON until the reset or power OFF of CPU. Turn it OFF as required.
- The following table shows the comparison result of each instruction.

Instruction symbol	Condition	Comparison result	Condition	Comparison result
LDTM=, ANDTM=, ORTM=	(s1) = (s2)	Conduction state	(s1) ≠ (s2)	Non-conduction state
LDTM<>, ANDTM<>, ORTM<>	(s1) ≠ (s2)		(s1) = (s2)	
LDTM>, ANDTM>, ORTM>	(s1) > (s2)		(s1)≤(s2)	
LDTM<=, ANDTM<=, ORTM<=	(s1)≤(s2)		(s1) > (s2)	
LDTM<, ANDTM<, ORTM<	(s1) < (s2)		(s1)≥(s2)	
LDTM>=, ANDTM>=, ORTM>=	(s1)≥(s2)		(s1) < (s2)	

- The following shows the comparison example of time.

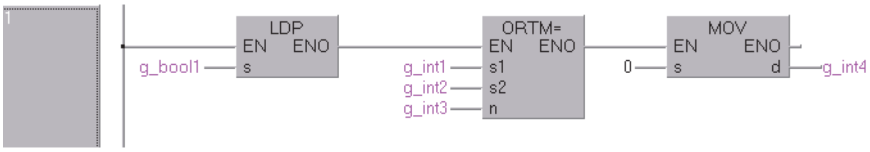


The following table shows the comparison result of time A, B, and C indicated above. Even when the data are compared under the same condition, the comparison result changes according to the specified comparison target.

Comparison target	Comparison condition		
	A < B	B < C	A < C
Second	○	×	×
Minute	×	○	×
Minute, Second	×	○	×
Hour	○	○	○
Hour, Second	○	○	○
Hour, Minute	○	○	○
Hour, Minute, Second	○	○	○
None	×	×	×

○: Conduction state ×: Non-conduction state

- The ORTM=, ORTM<>, ORTM<=, ORTM<, ORTM>=, or ORTM> instruction performs OR operation between the operation result of "(s1), (s2)" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400. When using the ORTM=, ORTM<>, ORTM<=, ORTM<, ORTM>=, or ORTM> instruction, connect EN and ENO in series as shown below.



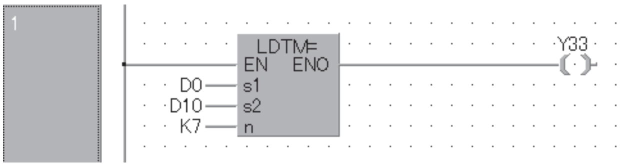
Operation error

- There is no operation error.

Program example

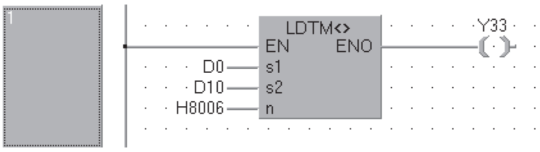
- In the following program, the data in D0 and the data in D10 (hour/minute/second) are compared, and Y33 turns ON if they are matched.

[Structured ladder/FBD]



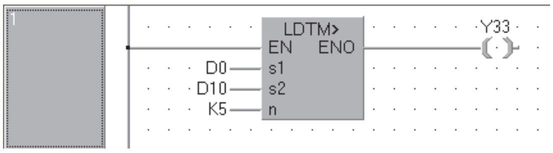
- In the following program, the data in D0 and the current time data (hour/minute) are compared, and Y33 turns ON if they are not matched.

[Structured ladder/FBD]



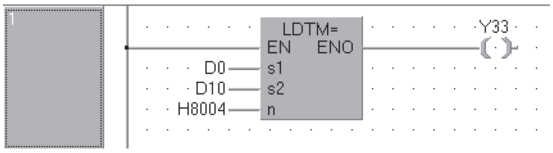
- In the following program, the data in D0 and the data in D10 (hour/second) are compared, and Y33 turns ON if the value of data in D10 is less than the value of data in D0.

[Structured ladder/FBD]



- In the following program, the data in D0 and the current date data (hour) are compared, and Y33 turns ON if the value of current time data is equal to/higher than the value of data in D0.

[Structured ladder/FBD]



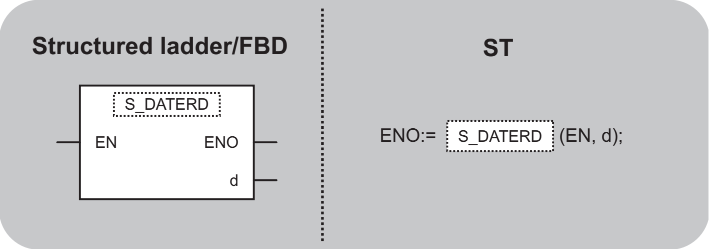
7.16 Extended Clock Instructions

Read extended clock data

S(P)_DATERD

Basic Ver. High performance Ver. Process Ver. Redundant Universal LCPU

- High Performance model QCPU: Supported if first 5 digits of the serial number are "07032" or later
- Process CPU: Supported if first 5 digits of the serial number are "07032" or later
- Redundant CPU: Supported if first 5 digits of the serial number are "07032" or later



ENO:= S_DATERD (EN, d);

Any of the following instruction can go in the dotted squares.

S_DATERD, SP_DATERD

■Executing condition

Instruction	Executing condition
S_DATERD	⎍
SP_DATERD	↑

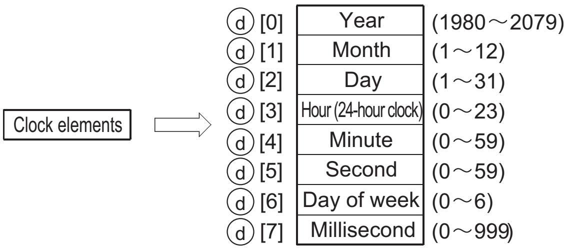
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Start number of device that stores the read clock data	ANY16(0..7)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(d)	—	○		—					

Processing details

- Reads year, month, day, minute, second, day of week, and millisecond from the clock element of the CPU module and stores them in (d) and the following devices in BIN value.



Clock elements

ⓓ [0]	Year	(1980～2079)
ⓓ [1]	Month	(1～12)
ⓓ [2]	Day	(1～31)
ⓓ [3]	Hour (24-hour clock)	(0～23)
ⓓ [4]	Minute	(0～59)
ⓓ [5]	Second	(0～59)
ⓓ [6]	Day of week	(0～6)
ⓓ [7]	Millisecond	(0～999)

- The year in (d)[0] is stored as 4-digit year indication.
- The day of week in (d)[6] is stored as 0 to 6 to represent the days from Sunday to Saturday.

Day of week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Stored data	0	1	2	3	4	5	6

- Compensation is made automatically for leap years.

Operation error

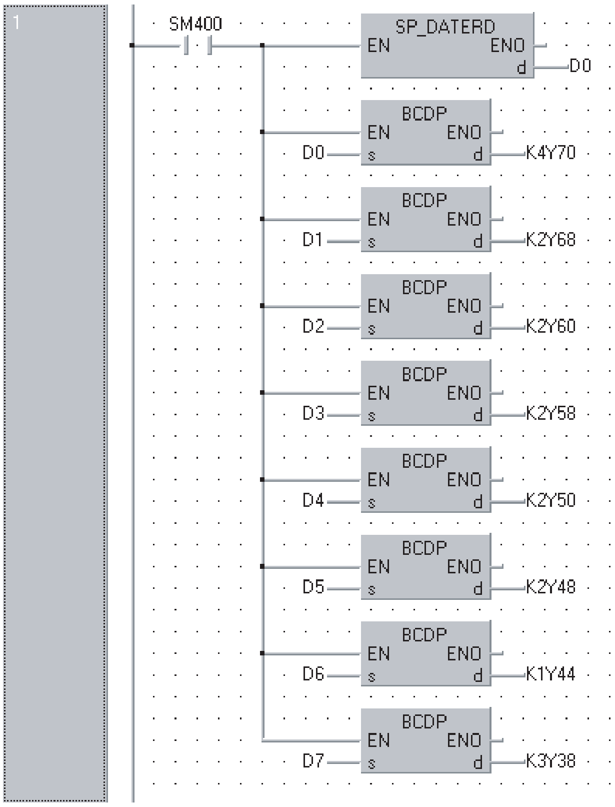
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, the following clock data are output in BCD values.
  - Year:Y70 to Y7F
  - Month:Y68 to Y6F
  - Day:Y60 to Y67
  - Hour:Y58 to Y5F
  - Minute:Y50 to Y57
  - Second:Y48 to Y4F
  - Day of week:Y44 to Y47
  - Millisecond:Y38 to Y43

[Structured ladder/FBD]



Output of year

Output of month

Output of day

Output of hour

Output of minute

Output of second

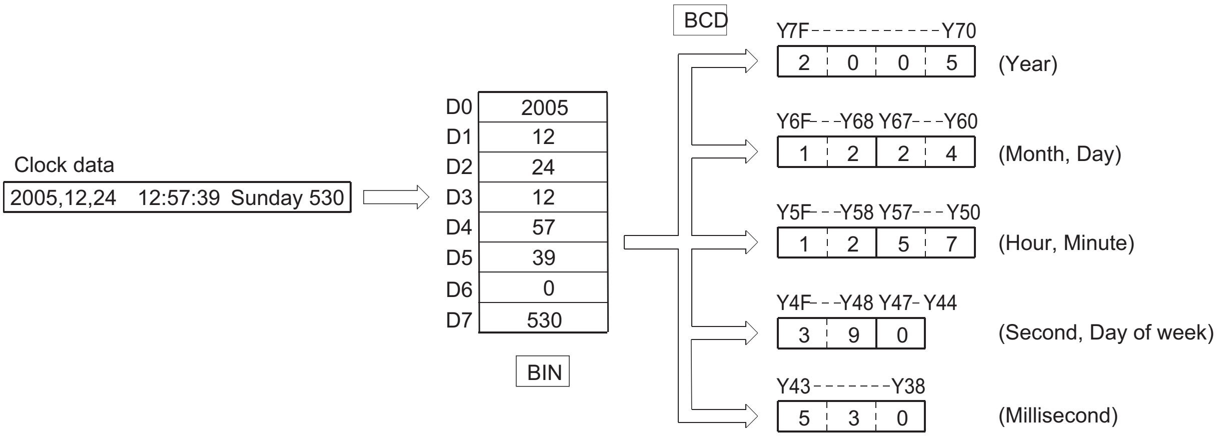
Output of day of week

Output of millisecond

[ST]

```
SP_DATERD(SM400,D0);
BCDP(SM400,D0,K4Y70);
BCDP(SM400,D1,K2Y68);
BCDP(SM400,D2,K2Y60);
BCDP(SM400,D3,K2Y58);
BCDP(SM400,D4,K2Y50);
BCDP(SM400,D5,K2Y48);
BCDP(SM400,D6,K1Y44);
BCDP(SM400,D7,K3Y38);
```

[Operation]



Precautions

- Even when the incorrect clock data are set in the CPU module, this instruction reads the data and stores them to the devices. (Example: February 30) When setting clock data using the DATEWR instruction or GX Works2, set the correct clock data.
- The difference of data when reading the millisecond clock data is 2ms maximum. (The difference between the data stored at the clock element in the CPU and data read by this instruction.)
- Digit specification of bit devices can be used when the following conditions (a) and (b) are satisfied. If the following conditions are not satisfied, an INSTRCT CODE ERR. (error code: 4004) occurs.
  - Digit specification: K4
  - Start device: multiple of 16

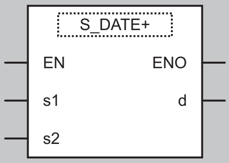
Addition of extended clock data

S(P)_DATE+

Basic Ver. High performance Ver. Process Ver. Redundant Universal LCPU

- High Performance model QCPU: Supported if first 5 digits of the serial number are "07032" or later
- Process CPU: Supported if first 5 digits of the serial number are "07032" or later
- Redundant CPU: Supported if first 5 digits of the serial number are "07032" or later

Structured ladder/FBD



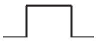
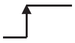
ST

Not supported

Any of the following instruction can go in the dotted squares.

S_DATE+, SP_DATE+

■Executing condition

Instruction	Executing condition
S_DATE+	
SP_DATE+	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores the clock data	ANY16(0..4)
	s2	Start number of the device that stores the clock data to be added	ANY16(0..4)
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the clock data of the addition result	ANY16(0..4)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—					
(s2)	—	○		—					
(d)	—	○		—					

Processing details

- Adds the clock data specified for (s2) to the clock data specified for (s1), and stores the result to (d) and the following devices.

		Setting data				Setting data				Setting data
s1[0]	Hour	(0～23)		s2[0]	Hour	(0～23)		d[0]	Hour	(0～23)
s1[1]	Minute	(0～59)		s2[1]	Minute	(0～59)		d[1]	Minute	(0～59)
s1[2]	Second	(0～59)	+	s2[2]	Second	(0～59)	⇨	d[2]	Second	(0～59)
s1[3]	―			s2[3]	―			d[3]	―	
s1[4]	Millisecond	(0～999)		s2[4]	Millisecond	(0～999)		d[4]	Millisecond	(0～999)

For example, adding the time 7:48:10:500 to 6:32:40:875 results in the following operation.

s1[0]	Hour: 6		s2[0]	Hour: 7		d[0]	Hour: 14
s1[1]	Minute: 32		s2[1]	Minute: 48		d[1]	Minute: 20
s1[2]	Second: 40	+	s2[2]	Second: 10	⇨	d[2]	Second: 51
s1[3]	―		s2[3]	―		d[3]	―
s1[4]	875		s2[4]	500		d[4]	375

- If the addition result of time exceeds 24 hours, 24 hours is subtracted from the sum to make the final operation result. For example, if the time 20:20:20:500 is added to 14:20:30:875, the result becomes 10:40:51:375 instead of 34:40:51:375.

s1[0]	Hour: 14		s2[0]	Hour: 20		d[0]	Hour: 10
s1[1]	Minute: 20		s2[1]	Minute: 20		d[1]	Minute: 40
s1[2]	Second: 30	+	s2[2]	Second: 20	⇨	d[2]	Second: 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 the operation.

The clock data read by the S(P)_DATERD can be added as they are.

d[0]	Hour
d[1]	Minute
d[2]	Second
d[3]	Day of week ⬅
d[4]	Millisecond

When the data are read by the S(P)_DATERD instruction, the data of day of week are stored between the devices that store the data of second and millisecond. Since the data of day of week are not operated in the S(P)DATE+ instruction, the data can be added as they are.

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The values set for (s1) and (s2) are outside of the setting range. (☞ Processing details)	―	○	○	○	○	○
4101	The device specified for (s1), (s2) or (d) exceeds the corresponding device range.	―	―	―	―	○	○

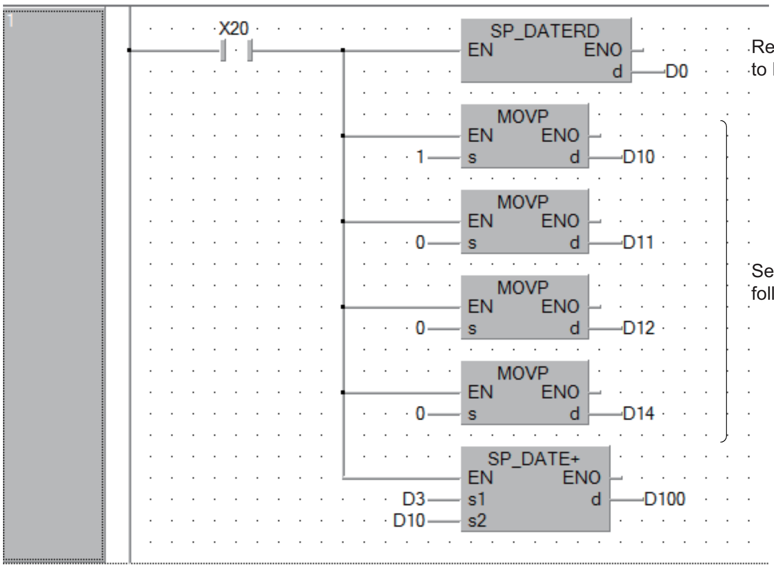
Precautions

- Digit specification of bit devices can be used when the following conditions are satisfied. If the following conditions are not satisfied, an INSTRCT CODE ERR. (error code: 4004) occurs.
  - Digit specification: K4
  - Start device: multiple of 16

Program example

- In the following program, 1 hour is added to the clock data read from the clock element when X20 turns ON, and the result is stored to D100 and the following devices.

[Structured ladder/FBD]



Reads the clock element data to D0 and the following devices.

Sets the time to D10 and the following devices.

[Operation]

- Clock data read by the SP_DATERD instruction.

Clock elements ⇨

D0	2005	Year	
D1	5	Month	
D2	17	Day	
D3	10	Hour	Clock data
D4	23	Minute	
D5	41	Second	
D6	2	Day of week	
D7	100	Millisecond	Clock data

- Addition by the SP_DATE+ instruction.

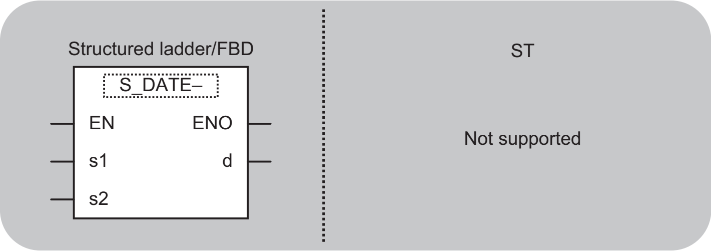
D3	Hour: 10		D10	Hour: 1		D100	Hour: 11
D4	Minute: 23		D11	Minute: 0		D101	Minute: 23
D5	Second: 41	+	D12	Second: 0	⇨	D102	Second: 41
D6	2(Tuesday)		D13	–		D103	–
D7	100		D14	0		D104	100

Subtraction of extended clock data

S(P)_DATE-

Basic Ver. High performance Ver. Process Ver. Redundant Universal LCPU

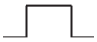
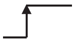
- High Performance model QCPU: Supported if first 5 digits of the serial number are "07032" or later
- Process CPU: Supported if first 5 digits of the serial number are "07032" or later
- Redundant CPU: Supported if first 5 digits of the serial number are "07032" or later



Any of the following instruction can go in the dotted squares.

S_DATE-, SP_DATE-

■Executing condition

Instruction	Executing condition
S_DATE-	
SP_DATE-	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start number of the device that stores the clock data	ANY16(0..4)
	s2	Start number of the device that stores the clock data to be subtracted	ANY16(0..4)
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores the clock data of the subtraction result	ANY16(0..4)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—					
(s2)	—	○		—					
(d)	—	○		—					

Processing details

- Subtracts the clock data specified for (s2) from the clock data specified for (s1), and stores the result to (d) and the following devices.

		Setting data				Setting data				Setting data
s1 [0]	Hour	(0～23)		s2 [0]	Hour	(0～23)		d [0]	Hour	(0～23)
s1 [1]	Minute	(0～59)		s2 [1]	Minute	(0～59)		d [1]	Minute	(0～59)
s1 [2]	Second	(0～59)	−	s2 [2]	Second	(0～59)	⇨	d [2]	Second	(0～59)
s1 [3]	−			s2 [3]	−			d [3]	−	
s1 [4]	Millisecond	(0～999)		s2 [4]	Millisecond	(0～999)		d [4]	Millisecond	(0～999)

For example, subtracting the time 3:50:10:500 from 10:40:20:875 results in the following operation.

s1 [0]	Hour: 10		s2 [0]	Hour: 3		d [0]	Hour: 6
s1 [1]	Minute: 40		s2 [1]	Minute: 50		d [1]	Minute: 50
s1 [2]	Second: 20	−	s2 [2]	Second: 10	⇨	d [2]	Second: 10
s1 [3]	−		s2 [3]	−		d [3]	−
s1 [4]	875		s2 [4]	500		d [4]	375

- If the operation results in a negative number, 24 is added to the result to make a final operation result. For example, if the time 10:42:12:500 is subtracted from 4:50:32:875, the result becomes 18:8:20:375 instead of -6:8:20:375.

s1 [0]	Hour: 4		s2 [0]	Hour: 10		d [0]	Hour: 18
s1 [1]	Minute: 50		s2 [1]	Minute: 42		d [1]	Minute: 8
s1 [2]	Second: 32	−	s2 [2]	Second: 12	⇨	d [2]	Second: 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 the operation.
The clock data read by the S(P)_DATERD can be added as they are.

d [0]	Hour	
d [1]	Minute	
d [2]	Second	
d [3]	Day of week	◀▮▮▮
d [4]	Millisecond	

When the data are read by the S(P)_DATERD instruction, the data of day of week are stored between the devices that store the data of second and millisecond.
Since the data of day of week are not operated in the S(P)_DATE- instruction, the data can be added as they are.

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The values set for (s1) and (s2) are outside of the setting range. (☞ Processing details)	—	○	○	○	○	○
4101	The device specified for (s1), (s2) or (d) exceeds the corresponding device range.	—	—	—	—	○	○

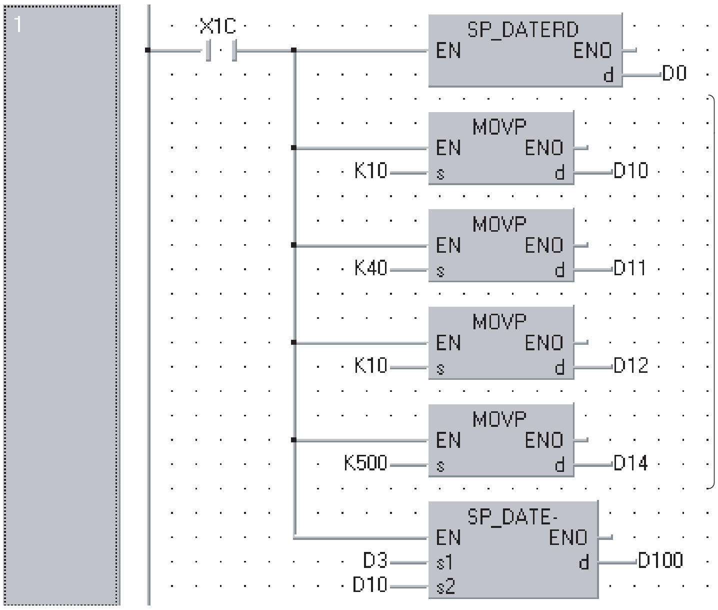
Precautions

- Digit specification of bit devices can be used when the following conditions are satisfied. If the following conditions are not satisfied, an INSTRCT CODE ERR. (error code: 4004) occurs.
  - Digit specification: K4
  - Start device: multiple of 16

Program example

- In the following program, the clock data stored in D10 and the following devices are subtracted from the clock data read from the clock element when X1C turns ON, and the result is stored to D100 and the following devices.

[Structured ladder/FBD]



Reads the clock element data to D0 and the following devices.

Sets the time to D10 and the following devices.

[Operation]

- Clock data read by the SP_DATERD instruction.

Clock elements ⇨

D0	2005	Year	
D1	2	Month	
D2	23	Day	
D3	8	Hour	Clock data
D4	42	Minute	
D5	1	Second	
D6	3	Day of week	
D7	997	Millisecond	Clock data

- Subtraction by the SP_DATE- instruction.

D3	Hour: 8
D4	Minute: 42
D5	Second: 1
D6	3(Wednesday)
D7	997

−

D10	Hour: 10
D11	Minute: 40
D12	Second: 10
D13	−
D14	500

⇨

D100	Hour: 22
D101	Minute: 1
D102	Second: 51
D103	−
D104	497

8:42:1:997 - 10:40:10:500 ⇨ -2:1:51:497

↓

Adds 24

⇩

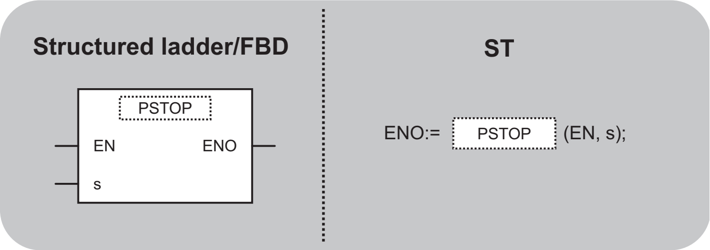
22:1:51:497

7.17 Program Control Instructions

Program standby

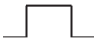
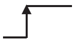
PSTOP(P)

Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.
PSTOP, PSTOPP

■Executing condition

Instruction	Executing condition
PSTOP	
PSTOPP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Character string data of the file name whose program is set to the standby type, or start number of the device that stores character string data	String
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—

Processing details

- Places the program whose file name is stored in the device specified for (s) in a standby type.
- Only the programs stored in the drive number 0 (program memory/built-in RAM) can be set as the standby type.
- The specified program is placed in a standby type when END processing is performed.
- This instruction will be given priority even in cases when a program execution type has been specified in the parameters.
- It is not necessary to specify the extension (.QPG) with the file name. (Only .QPG files will be acted on.)

Operation error

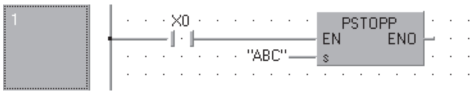
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
2410	The program with the file name specified for (s) does not exist.	—	○	○	○	○	○
2412	The program type of the file name specified for (s) is the SFC program.	—	○	○	○	○	○
4101	The file name target device (s) exceeds the corresponding device range.	—	○	○	○	○	○

Program example

- In the following program, the program with the file name ABC is set in a standby type when X0 turns ON.

[Structured ladder/FBD]



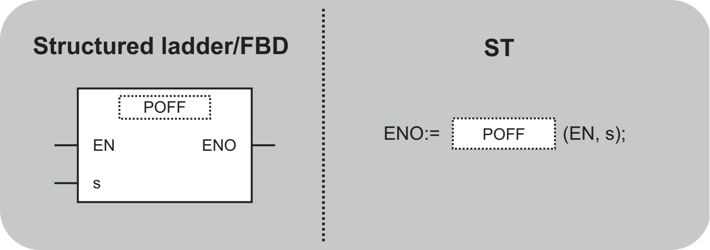
[ST]

PSTOPP(X0,"ABC");

Program output OFF standby

POFF(P)

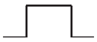
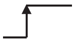
~~Basic~~ High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

POFF, POFFP

■Executing condition

Instruction	Executing condition
POFF	
POFFP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	File name whose program is set to the standby type by turning OFF the output, or start number of the device that stores file name	String
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—

Processing details

- Changes the execution type of the program whose file name stored in the device specified for (s).
  - Scan execution type:Turns OFF outputs at the next scan (Non-execution processing). Programs are set as the standby type after the subsequent scan.
  - Low speed execution type:Stops the execution of the low speed execution type program and turns OFF outputs at the next scan. Programs are set as the standby type after the subsequent scan.
- Only the programs stored in the drive number 0 (program memory/built-in RAM) can be set as the standby type.
- This instruction will be given priority even in cases when a program execution type has been specified in the parameters.
- It is not necessary to specify the extension (.QPG) with the file name. (Only .QPG files will be acted on.)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2410	The program with the file name specified for (s) does not exist.	—	○	○	○	○	○
2411	The program with the file name specified for (s) is not registered in the parameters.	—	○	○	○	○	○
4101	The file name target device (s) exceeds the corresponding device range.	—	○	○	○	○	○

Point

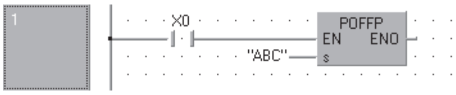
Non-execution processing is identical to the processing that is performed when the condition settings for each coil instruction are in the OFF state.
The operation results for each coil instruction following non-execution processing will be as follows, regardless of the ON/OFF status of the condition settings.

- OUT instruction: Forced OFF
- SET instruction, RST instruction, SFT instruction, Basic instruction, Application instruction: Maintains status
- PLS instruction, Pulse generation instruction (□P): Processing identical to when condition contacts are OFF
- Current value of low speed/high speed timer: 0
- Current value of retentive timer, Current value of counter: Preserves

Program example

- In the following program, the program with the file name ABC is set in the nonexecutable and the standby type when X0 turns ON.

[Structured ladder/FBD]

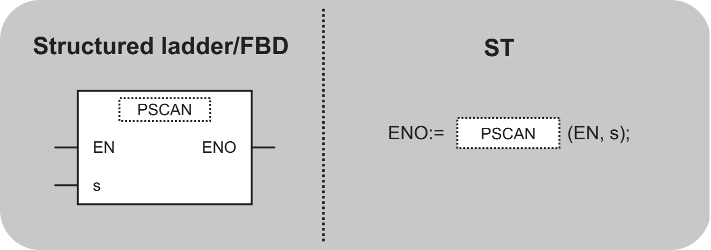


[ST]
POFFP(X0,"ABC");

Registering program as scan execution type

PSCAN(P)

Basic High performance Process Redundant Universal LCPU



ENO:= PSCAN (EN, s);

Any of the following instruction can go in the dotted squares.

PSCAN, PSCANP

■Executing condition

Instruction	Executing condition
PSCAN	⊓
PSCANP	↑

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	File name whose program is set to the scan execution type, or start number of the device that stores file name	String
Output argument	ENO	Execution result	Bit

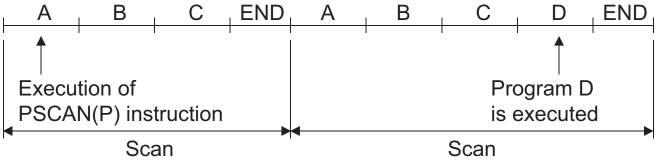
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—

Processing details

- Sets the program whose file name is stored in the device specified for (s) in the scan execution type.
- Only the programs stored in the drive number 0 (program memory/built-in RAM) can be set in the scan execution type.
- Specified programs assume the scan execution type with END processing.

Ex.

When programs A, B, and C exist and program A performs the PSCAN(P) instruction of program D.



- This instruction will be given priority even in cases when a program execution type has been specified in the parameters.
- It is not necessary to specify the extension (.QPG) with the file name. (Only .QPG files will be acted on.)

Operation error

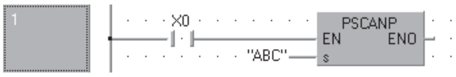
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2410	The program with the file name specified for (s) does not exist.	—	○	○	○	○	○
2411	The program with the file name specified for (s) is not registered in the parameters.	—	○	○	○	○	○
2504	The specified file name is the SFC program, and the SFC program for the other file name has been already started. (Dual activation error of the SFC program)	—	○	○	○	—	—
4101	The file name target device (s) exceeds the corresponding device range.	—	○	○	○	○	○
4131	The SFC program file name is specified while the SFC program with another file name is already operating. (Double SFC program activation error)	—	—	—	—	○	○

Program example

- In the following program, the program with the file name ABC is set in the scan execution type when X0 turns ON.

[Structured ladder/FBD]



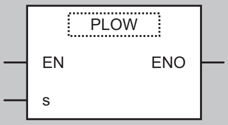
[ST]
PSCANP(X0,"ABC");

Registering program as low-speed execution type

PLOW(P)

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



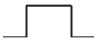
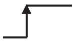
ST

ENO:= PLOW (EN, s);

Any of the following instruction can go in the dotted squares.

PLOW, PLOWP

■Executing condition

Instruction	Executing condition
PLOW	
PLOWP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	File name whose program is set to the low-speed execution type, or start number of the device that stores file name	String
Output argument	ENO	Execution result	Bit

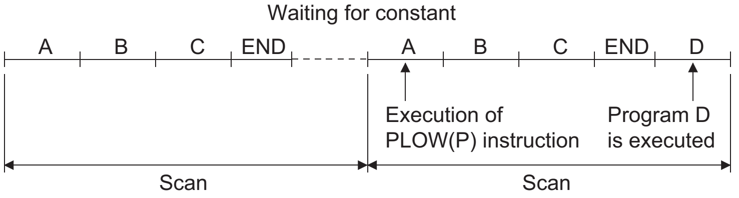
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—

Processing details

- Sets the program whose file name is stored in the device specified for (s) in the low-speed execution type.
- Only the programs stored in the drive number 0 (program memory/built-in RAM) can be set in the low-speed execution type.
- Specified programs assume the low-speed execution type with END processing.

Ex.

When programs A, B, and C exist and program A performs the PLOW(P) instruction of program D. (Assume that the constant scan has been set.)



- This instruction will be given priority even in cases when a program execution type has been specified in the parameters.
- It is not necessary to specify the extension (.QPG) with the file name. (Only .QPG files will be acted on.)

Operation error

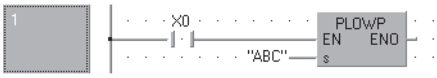
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2410	The program with the specified file name does not exist.	—	○	○	—	—	—
4235	The CHK instruction exists within the program whose file name has been specified.	—	○	○	—	—	—

Program example

- In the following program, the program with the file name ABC is set in the low-speed execution type when X0 turns ON.

[Structured ladder/FBD]



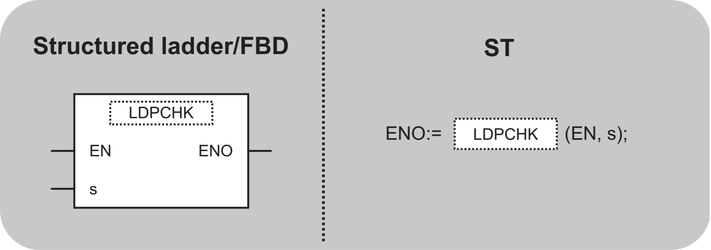
[ST]

PLOWP(X0,"ABC");

Checking program execution status

LDPCHK, ANDPCHK, ORPCHK

Basic High performance Process Redundant Universal LCPU



```
ENO:= LDPCHK (EN, s);
```

Any of the following instruction can go in the dotted squares.
LDPCHK, ANDPCHK, ORPCHK

■Executing condition

Instruction	Executing condition
LDPCHK, ANDPCHK, ORPCHK	Non-conditional execution

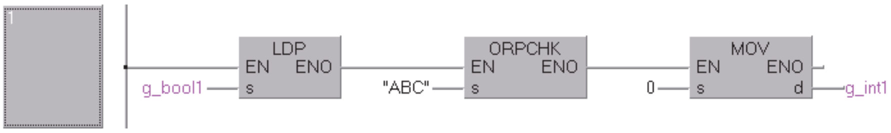
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start device number that stores file name of a program whose execution status is checked	String
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—							○	—

Processing details

- Checks whether the program of the specified file name is in execution or not (non-execution).
- The instruction is in conduction state when the program of the specified file name is in execution, and the instruction is in non-conduction state when the program is in non-execution status.
- Specify the file name without an extension (.QPG). For example, specify "ABC" when the file name is ABC.QPG.
- The ORPCHK instruction performs OR operation between the operation result of "(s1), (s2)" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400. When using the ORPCHK instruction, connect EN and ENO in series as shown below.



- Non-execution indicates that the program execution type is a stand-by type. Execution indicates that the program execution type is a scan execution type (including during output OFF (during non-execution processing)), low speed execution type or fixed scan execution type.

Operation error

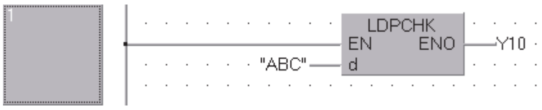
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2410	The program with the specified file name does not exist.	—	○	○	○	—	—

Program example

- In the following program, Y10 is turned ON when the program file "ABC.QPG" is in execution.

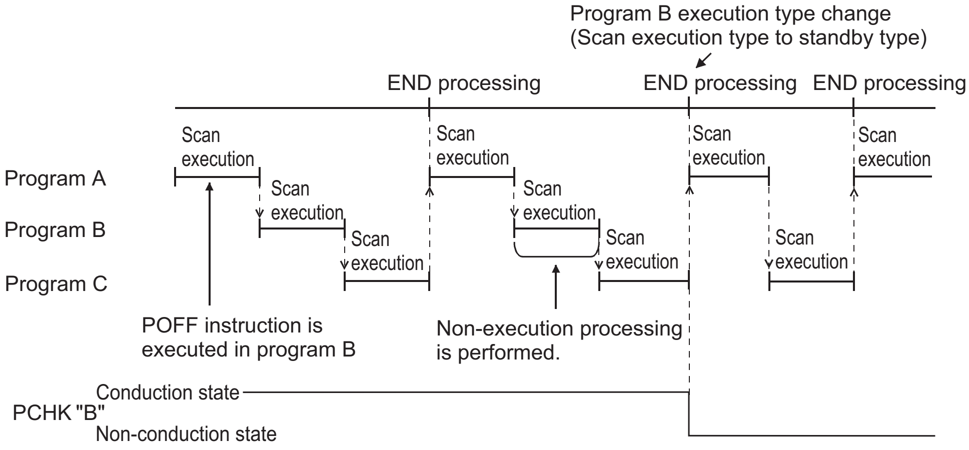
[Structured ladder/FBD]



[ST]
Y10: = LDPCHK(TRUE,"ABC");

Point

- The checking program execution status instruction (PCHK) is in conduction state when the program of the specified file name (target program) is in execution, and the instruction is in non-conduction state when the program is in non-execution.
- When the target program is set to non-execution (standby type) with the POFF(P) instruction, the PCHK instruction is in conduction state while the non-execution processing of the target program is being performed.
- At the END processing of the scan where the non-execution processing is completed, the target program is put into non-execution (standby type), and the PCHK instruction is brought into non-conduction state. Therefore, note that if the PCHK instruction is executed for the program where the non-execution processing has been completed by the POFF(P) instruction, the PCHK instruction may be brought into conduction state.
- The following chart shows the operation performed when program A executes the POFF(P) instruction of program B and program C executes the PCHK instruction of program B with the programs being executed in order of program A, program B and program C.

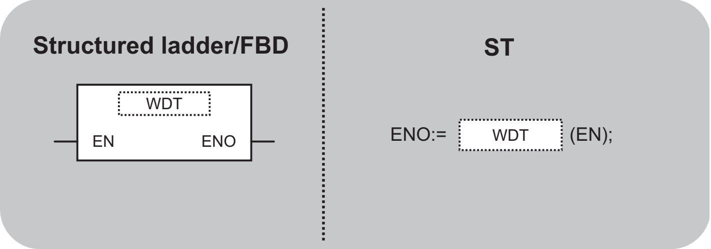


7.18 Other Instructions

Resetting watchdog timer

WDT(P)

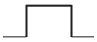
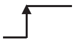
Basic | High performance | Process | Redundant | Universal | LCPU



```
ENO:= WDT (EN);
```

Any of the following instruction can go in the dotted squares.
WDT, WDTP

■Executing condition

Instruction	Executing condition
WDT	
WDTP	

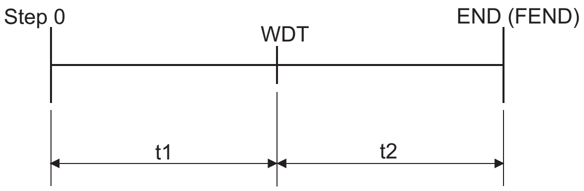
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

- Resets a watchdog timer during the execution of a sequence program.
- Used in cases where the scan time exceeds the value set for the watchdog timer due to prevailing conditions. If the scan time exceeds the watchdog timer setting value on every scan, change the watchdog timer settings on parameter settings of the programming tool.
- Make sure that the setting for t1 from step 0 to the WDT(P) instruction and the setting for t2 from the WDT(P) instruction to the END processing and FEND instruction not to exceed the setting value of the watchdog timer.



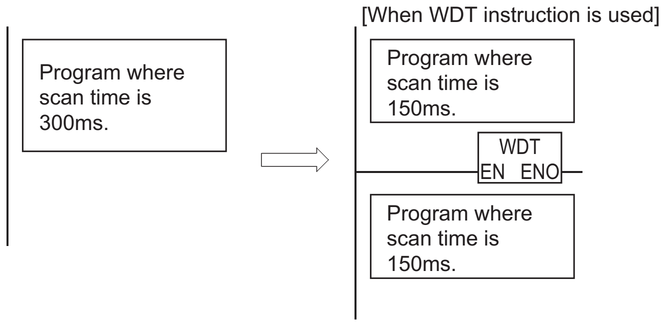
- The WDT(P) instruction can be used two or more times during a single scan, but a care should be taken in such cases, because longer time is required for turning OFF the output at the error occurrence.
- Scan time values stored in the special register will not be cleared even if the WDT(P) instruction is executed. Accordingly, there are times when the value for the scan time for the special register is greater than the value of the watchdog timer set in the parameters.

Operation error

- There is no operation error.

Program example

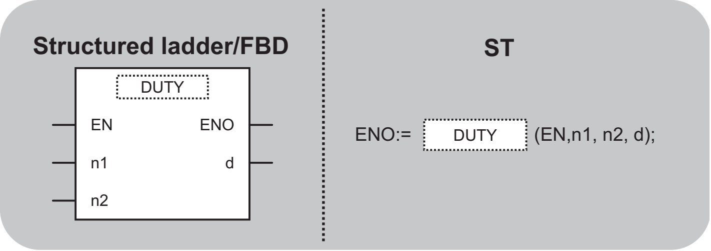
- The program which prevents occurrence of the watchdog timer error when the watchdog timer is set to 200ms, and the scan time from step 0 to the END processing is 300ms.


[When WDT instruction is used]
Program where scan time is 300ms.
Program where scan time is 150ms.
WDT
EN
ENO
Program where scan time is 150ms.

Timing pulse generation

DUTY

Basic | High performance | Process | Redundant | Universal | LCPU



The following instruction can go in the dotted squares.
DUTY

■Executing condition

Instruction	Executing condition
DUTY	↑

■Argument

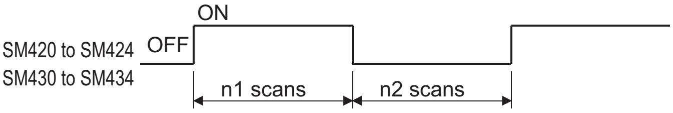
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Number of scans to be turned ON	ANY16
	n2	Number of scans to be turned OFF	ANY16
Output argument	ENO	Execution result	Bit
	d	User timing clock (SM420 to SM424, SM430 to M434)	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n1	○	○							—
n2	○	○							—
(d)	○*1	—							—

*1 Only for devices SM420 to SM424 and SM430 to SM434.

Processing details

- Turns ON the user timing clock (SM420 to SM424, SM430 to M434) specified for (d), for the duration equivalent to the number of scans specified for n1, and turns OFF for the duration equivalent to the number of scans specified for n2.



- Scan execution type programs use from SM420 to SM424, and low-speed execution type programs use from SM430 to SM434.
- The following will take place if both n1 and n2 have been set to 0.
  - n10, n2 ≥ 0 from SM420 to SM424 and from SM430 to SM434 stays OFF.
  - n1 > 0, n2 = 0 from SM420 to SM424 and from SM430 to SM434 stays ON.
- The data specified for n1, n2, and (d) is registered to the system when the DUTY instruction is executed, and the timing pulse is turned ON and OFF by the END processing.

Operation error

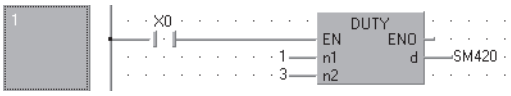
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The values of n1 and n2 are less than 0.	○	○	○	○	○	○
4101	The device specified for (d) is not from SM420 to SM424 or from SM430 to SM434.	○	○	○	○	○	○

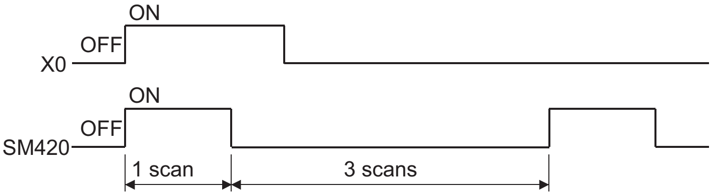
Program example

- In the following program, SM420 is turned ON for 1 scan, and OFF for 3 scans when X0 turns ON.

[Structured ladder/FBD]



[ST]
DUTY(X0,1,3,SM420);
[Operation]

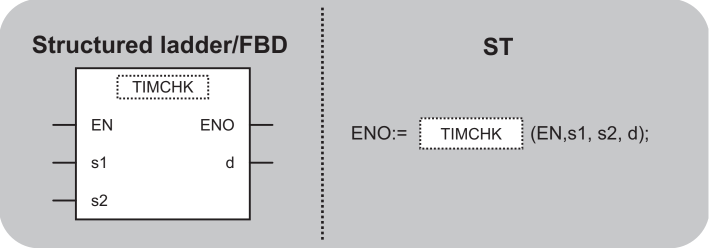


Time check

TIMCHK

Ver. Basic | High performance | Process | Redundant | Universal | LCPU

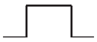
- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later



The following instruction can go in the dotted squares.

TIMCHK

■Executing condition

Instruction	Executing condition
TIMCHK	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Device that stores the measured current value	ANY16
	s2	Device that stores the set value for measurement	ANY16
Output argument	ENO	Execution result	Bit
	d	Device to be turned ON at the end of the process	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
s1	—	○		—					—
s2	○	○		○					—
(d)	○	—		—					—

Processing details

- Measures the ON time of the device used as a condition, and turns ON the device specified for (s2) if the condition device remains ON for longer than the time set to the device specified for (d).
- The current value of the device specified for (s1) is cleared to 0 and the device specified for (d) is turned OFF at the rising edge of the execution command. The current value of the device specified for (s1) and the ON status of the device specified for (d) are retained after the execution command turns OFF.
- Set the setting value of measurement in unit of 100ms.
- When the value other than 0 to 32767 is specified for (s2), (d) turns ON at the scan that is after the execution command has turned ON.

Operation error

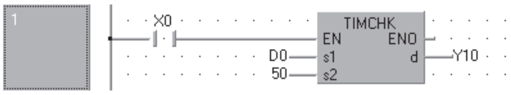
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	When the device which cannot be specified is specified	—	—	—	—	○	○

Program example

- In the following program, the ON time of X0 is set to 5 seconds, the current value storage device is set to D0 and the device that will turn ON at time out is set to Y10.

[Structured ladder/FBD]

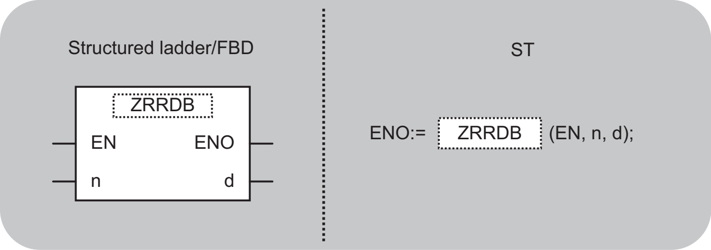


[ST]
TIMCHK(X0,D0,50,Y10);

Reading 1 byte directly from file register

ZRRDB(P)

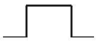
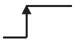
Basic High performance Process Redundant Universal LCPU



Any of the following instruction can go in the dotted squares.

ZRRDB, ZRRDBP

■Executing condition

Instruction	Executing condition
ZRRDB	
ZRRDBP	

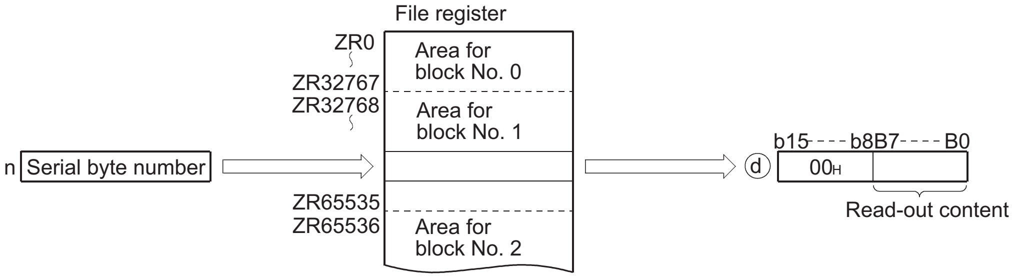
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Serial byte number whose file register is read	ANY32
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores read data	ANY16

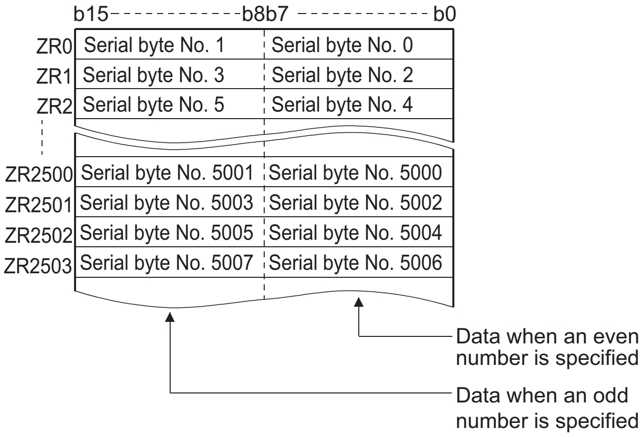
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○							○	—
(d)	○							—	—

Processing details

- Reads the serial byte number specified for n that does not signify a block number, and stores to the lower 8 bits of the device specified for (d). The upper 8 bits specified for (d) will become 00H.



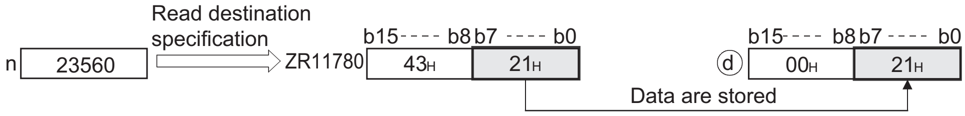
- File register numbers correspond to serial byte numbers are as indicated below.


b15------------b8b7 ------------ b0
Data when an even number is specified
Data when an odd number is specified

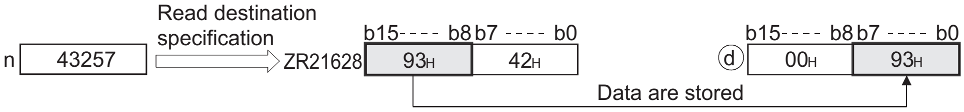
	b15–b8	b7–b0
ZR0	Serial byte No. 1	Serial byte No. 0
ZR1	Serial byte No. 3	Serial byte No. 2
ZR2	Serial byte No. 5	Serial byte No. 4
ZR2500	Serial byte No. 5001	Serial byte No. 5000
ZR2501	Serial byte No. 5003	Serial byte No. 5002
ZR2502	Serial byte No. 5005	Serial byte No. 5004
ZR2503	Serial byte No. 5007	Serial byte No. 5006

Ex.

If n = 23560 is specified, the data at the lower 8 bits of ZR11780 will be read.



If n = 43257 is specified, the data at the upper 8 bits of ZR21628 will be read.



Operation error

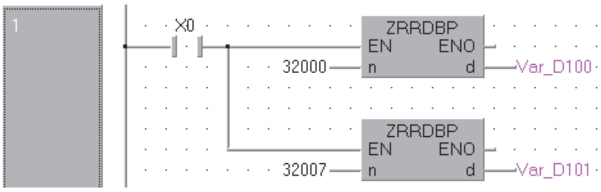
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	A device number (serial byte number) that exceeds the allowable specification range has been specified.	—	—	—	—	○	○

Program example

- In the following program, the lower bits of ZR16000 and the upper bits of R16003 are read when X0 turns ON, and the results are stored to Var_D100 and Var_D101.

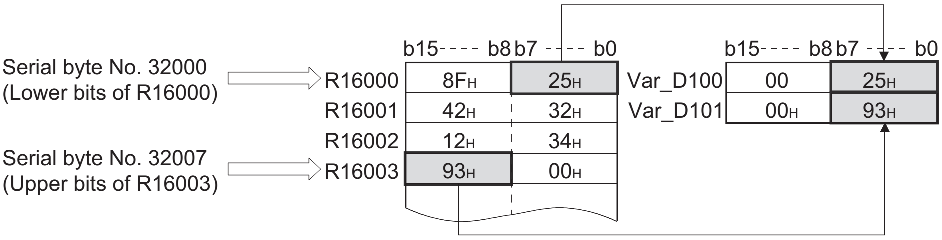
[Structured ladder/FBD]



[ST]

```
ZRRDBP(X0,32000,Var_D100);
ZRRDBP(X0,32007,Var_D101);
```

[Operation]

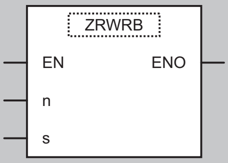


Writing 1 byte directly to file register

ZRWRB(P)

Basic High performance Process Redundant Universal LCPU

Structured ladder/FBD



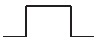
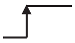
ST

ENO:= ZRWRB (EN, n, s);

Any of the following instruction can go in the dotted squares.

ZRWRB, ZRWRBP

■Executing condition

Instruction	Executing condition
ZRWRB	
ZRWRBP	

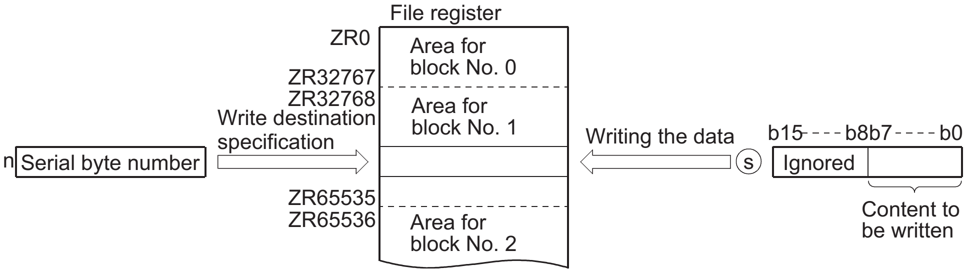
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Serial byte number whose file register is written	ANY32
	s	Start number of the device that stores data to be written	ANY16
Output argument	ENO	Execution result	Bit

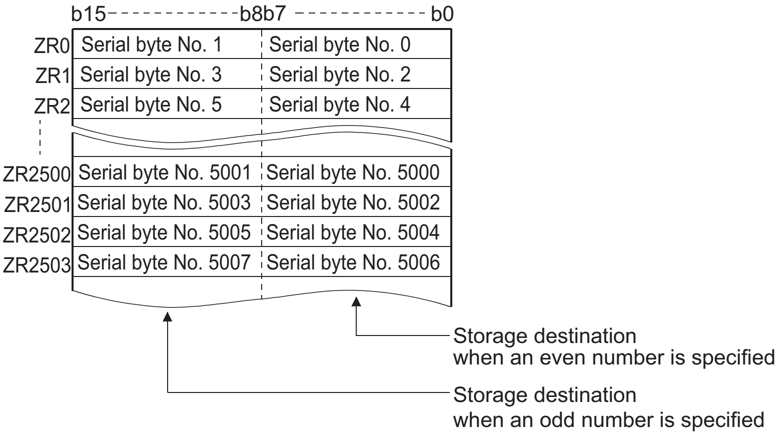
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○								—
(s)	○								—

Processing details

- Writes the lower bits of data stored in the device specified for (s) that does not signify a block number to the file register of the serial byte number specified for n. The upper 8 bits of data in the device specified for (s) are ignored.



- File register numbers correspond to serial byte numbers are as indicated below.

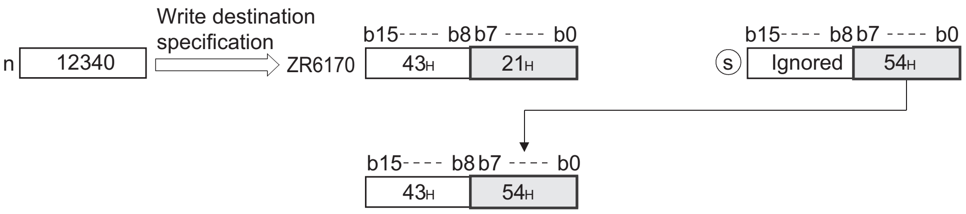


	b15 ------------ b8	b7 ------------ b0
ZR0	Serial byte No. 1	Serial byte No. 0
ZR1	Serial byte No. 3	Serial byte No. 2
ZR2	Serial byte No. 5	Serial byte No. 4
ZR2500	Serial byte No. 5001	Serial byte No. 5000
ZR2501	Serial byte No. 5003	Serial byte No. 5002
ZR2502	Serial byte No. 5005	Serial byte No. 5004
ZR2503	Serial byte No. 5007	Serial byte No. 5006

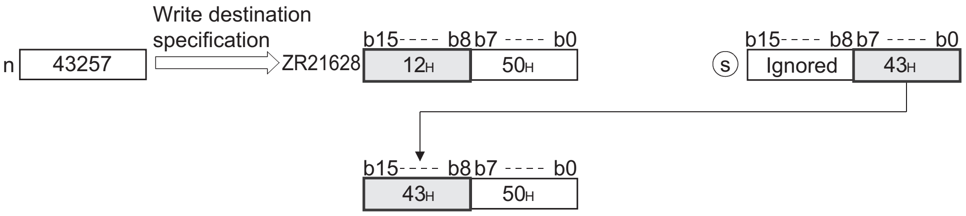
Storage destination when an even number is specified
Storage destination when an odd number is specified

Ex.

If n = 12340 is specified, the data will be written to the lower 8 bits of ZR6170.


Write destination specification
n
12340
ZR6170
b15- - - - b8 b7 - - - - b0
43H
21H
s
Ignored
54H
b15- - - - b8 b7 - - - - b0
43H
54H

If n = 43257 is specified, the data will be written to the upper 8 bits of ZR21628.


Write destination specification
n
43257
ZR21628
b15- - - - b8 b7 - - - - b0
12H
50H
s
Ignored
43H
b15- - - - b8 b7 - - - - b0
43H
50H

Operation error

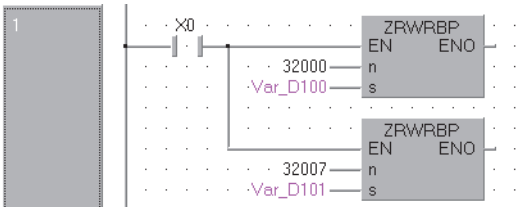
- In the following case, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	A device number (serial byte number) that exceeds the allowable specification range has been specified.	—	—	—	—	○	○

Program example

- In the following program, the data at the lower bits of Var_D101 are written to the lower bits of ZR16000 and the upper bits of R16003 when X0 turns ON.

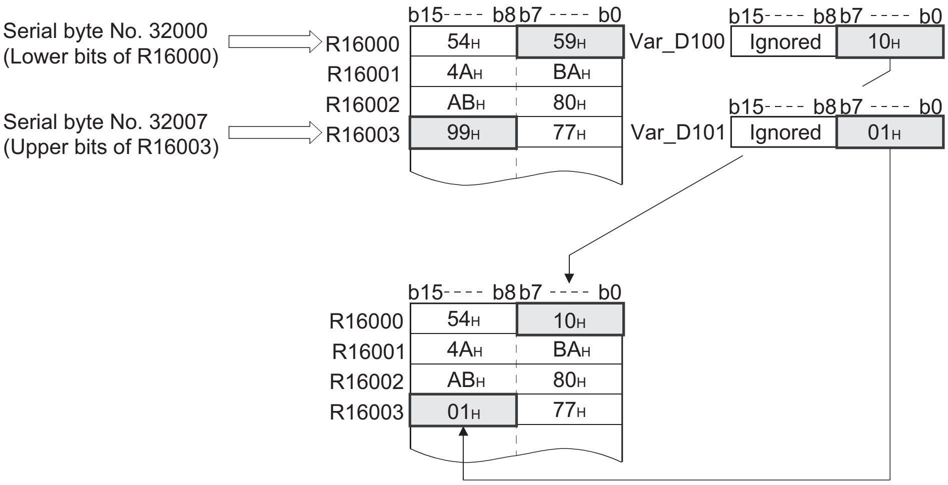
[Structured ladder/FBD]



[ST]

```
ZRWRBP(X0,32000,Var_D100);
ZRWRBP(X0,32007,Var_D101);
```

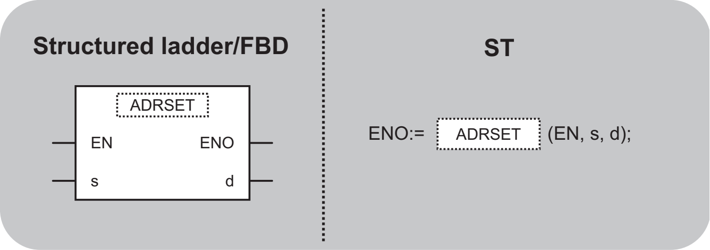
[Operation]



Reading indirect address

ADRSET(P)

Basic | High performance | Process | Redundant | Universal | LCPU

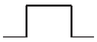
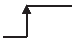


```
ENO:= ADRSET (EN, s, d);
```

Any of the following instruction can go in the dotted squares.

ADRSET, ADRSETP

■Executing condition

Instruction	Executing condition
ADRSET	
ADRSETP	

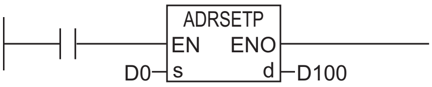
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Device that reads out indirect address	ANY_SIMPLE
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores indirect address of the device specified for s	ANY32

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
(s)	○			—					
(d)	○			—					

Processing details

- Stores the indirect address of the device specified for (s) in (d). The address stored in the device specified for (d) is used when an indirect device address is performed by the sequence program.



Stores the indirect address of D0 in D101 and D100

- Digit specification of bit devices cannot be set for (s).
- Indirect address specification (@D0) cannot be set. To use indirect address specification, use a ladder program.

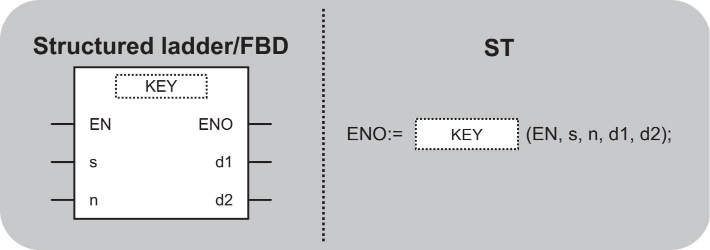
Operation error

- There is no operation error.

Numeric input from keyboard

KEY

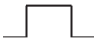
Basic High performance Process Redundant Universal LCPU



The following instruction can go in the dotted squares.

KEY

■Executing condition

Instruction	Executing condition
KEY	

■Argument

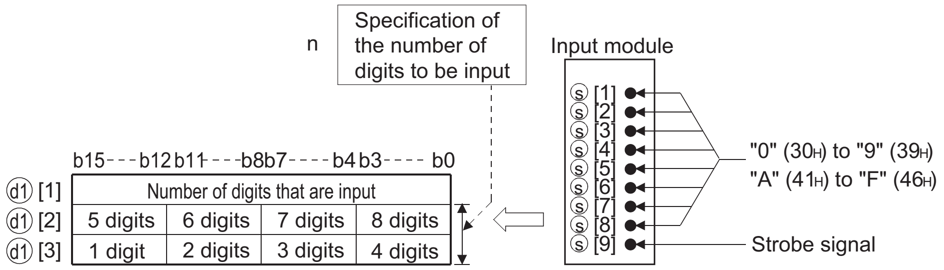
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number or the array of the device of (X) to which a numeral is input	Array of bit (1..9)
	n	Number of numeric digits to be input	ANY16
Output argument	ENO	Execution result	Bit
	d1	Start number or the array of the device where the input numeral to be stored	Array of ANY16 (1..3)
	d2	Bit device number to be turned ON at the input completion	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○(X only)[*1]	—		—			—		—
n	○	○		○			○		—
(d1)	—	○		—			—		—
(d2)	○	○		○			—		—

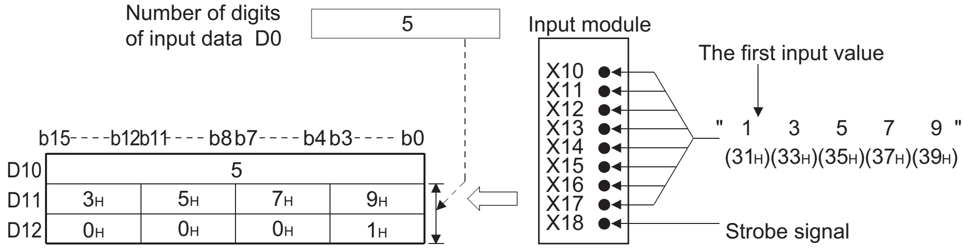
*1 Specify the array in which X is set as a device for global label.

Processing details

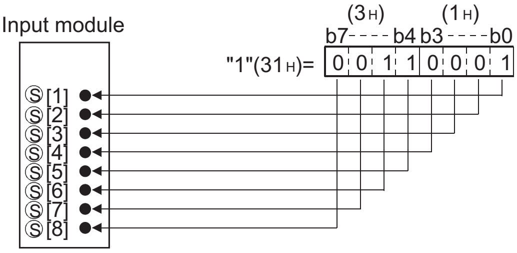
- Fetches ASCII data from the 8 points of input (X) specified for (s), converts it to hexadecimal values and stores the result to (d1) and the following devices.



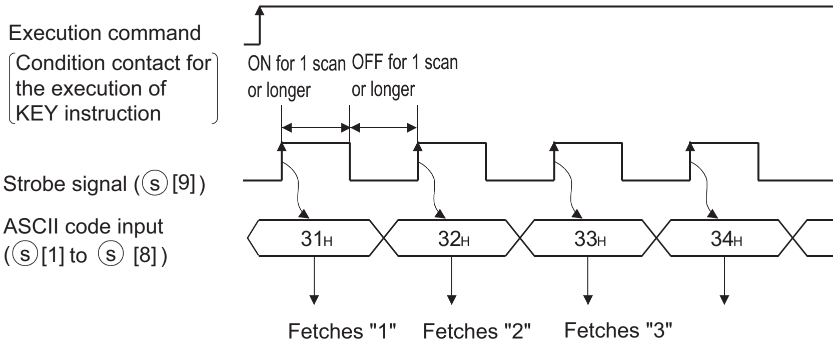
For example, in a case where the number of digits (n) has been set to 5, and the values "31", "33", "35", "37" and "39" have been input through X10 to X18 of the input module, the following will take place.



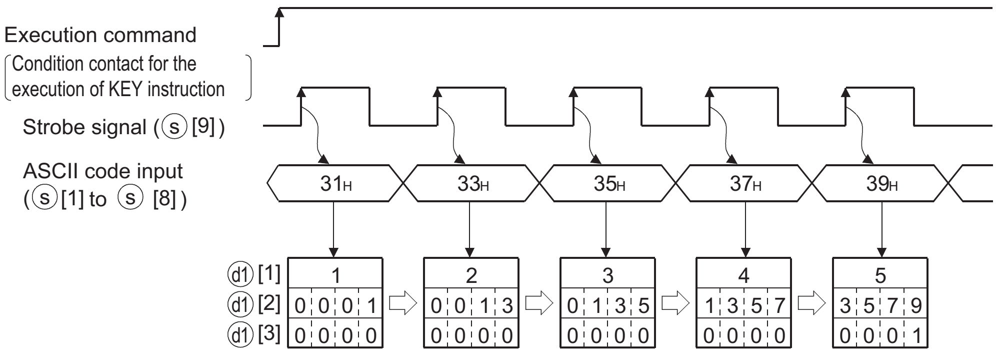
- Numeric input to input (X) specified for undergoes bit development and inputs to (s)[1] through (s)[8] as the ASCII code corresponding to the numbers. ASCII code which can be input is from 30H (0) to 39H (9), and from 41H (A) to 46H (F).



- After ASCII code is input to (s)[1] through (s)[8], the strobe signal at (s)[9] turns ON to incorporate the specified numbers internally. The strobe signal should be held at its ON or OFF status for more than one scan of the sequence program. If this time is less than 1 scan, there will be cases when the data are correctly incorporated.

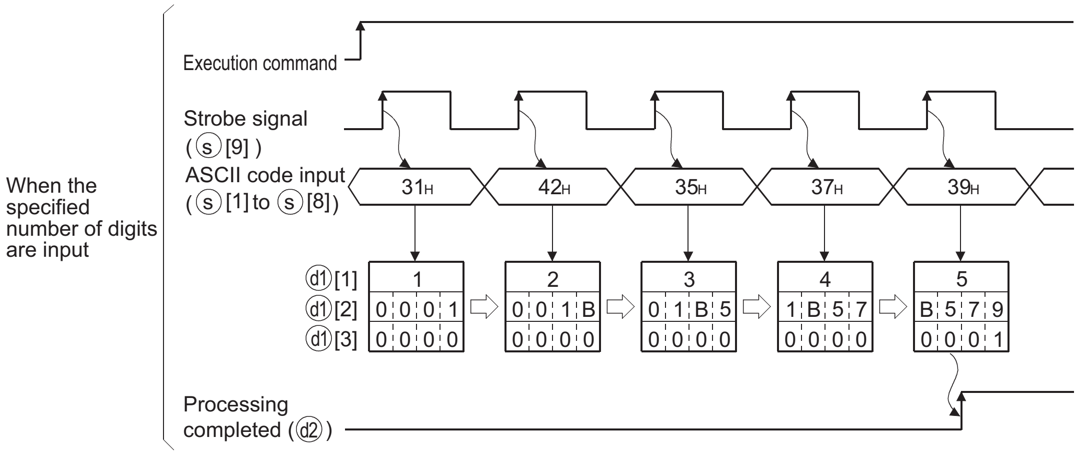


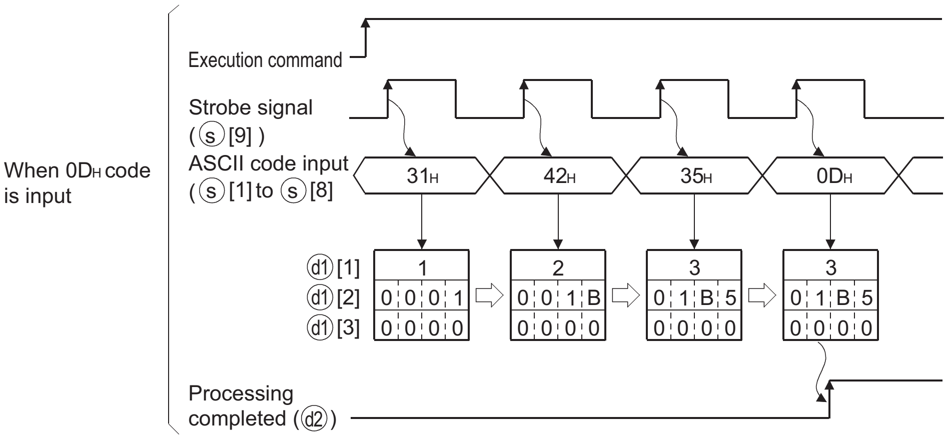
- Be sure to keep the execution command (condition contact for the KEY instruction execution) ON until the specified number of digits has been input. The KEY instruction cannot be executed if the execution command turns OFF.
- The digits for the numbers actually fetched to (d)[1] will be stored to the device specified for (d1), and the ASCII code data input to (d1)[2] and (d1)[3] are converted to hexadecimal BIN values, and stored.



- The number of digits that can be specified for n is from 1 to 8.
- Fetching of the input data is completed when any of the inputs shown below has been made. At the completion, the bit device specified for (d2) is turned ON.
  - When the number of digits specified for n has been input
  - When the "0DH" code has been input

For example, the operations at the location specified if n = 5 will be as indicated below.





- To perform input processing for a second time, clear the number of input digits and input data stored in (d1), and turn off the specified device for (d2) at the user program. If (d1) is not cleared and (d2) not turned OFF, the next input processing cannot be performed.

Operation error

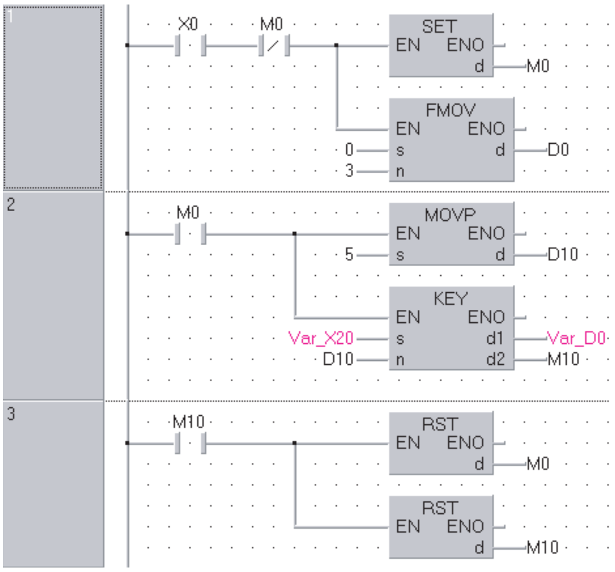
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The device specified for (s) is not an input (X) device. The number of digits specified for n is outside the range of 1 to 8.	—	○	○	—	—	—

Program example

- In the following program, the data of the 5 or fewer digits are fetched from the numeric keypad connected to the devices from Var_X20 (X20) and the following devices when X0 turns ON, and they are stored to Var_D0 (D0) and the following devices.

[Structured ladder/FBD]



Clears the previous input data.

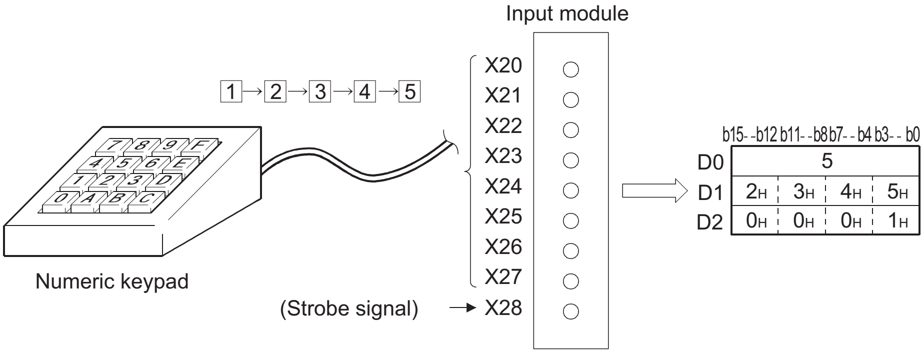
Sets the number of digits to be input.

Resets the data input completion flag.

[ST]

```
IF X0 AND NOT(M0) THEN
    SET(TRUE,M0);
    FMOVP(TRUE,0,3,Var_D0[1]);
END_IF;
MOVP(M0,5,D10);
KEY(M0,Var_X20,D10,Var_D0,M10);
RST(M10,M0);
RST(M10,M10);
```

[Operation]



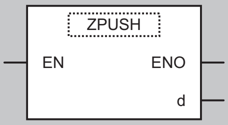
	b15- -b12	b11- -b8	b7- - b4	b3- - b0
D0	5			
D1	2H	3H	4H	5H
D2	0H	0H	0H	1H

Batch save and recovery of index registers

ZPUSH(P), ZPOP(P)

Basic | High performance | Process | Redundant | Universal | LCPU

Structured ladder/FBD

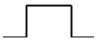
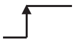


ST

ENO:= ZPUSH (EN, d);

Any of the following instruction can go in the dotted squares.
ZPUSH, ZPUSHP, ZPOP, ZPOPP

■Executing condition

Instruction	Executing condition
ZPUSH, ZPOP	
ZPUSHP, ZPOPP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit
	d	Start number of the device whose index registers are saved or recovered	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(d)	—	○		—					

Processing details

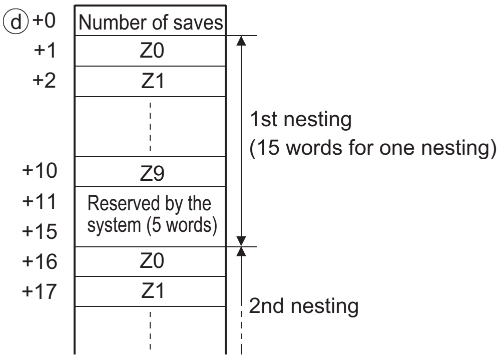
■ZPUSH(P)

- Saves the content of the following index registers to (d) and the following devices. (When content of an index register are saved, (d) + 0 (the number of saves made) is increased by 1.)

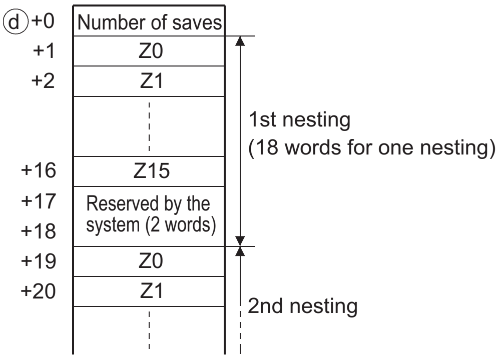
• Basic model QCPU	Z0 to Z9
• High Performance model QCPU, Process CPU, Redundant CPU	Z0 to Z15
• Universal model QCPU, LCPU	Z0 to Z19

- The ZPOP(P) instruction is used for data recovery. The ZPUSH(P) and ZPOP(P) instructions are used as a pair and can be nested.
- If nesting has been done, each time the ZPUSH(P) instruction is executed, the field used following (d) will be added, so a field large enough to accommodate the number of times the instruction will be used should be reserved from the beginning.
- The composition of the field used following (d) is as shown below.

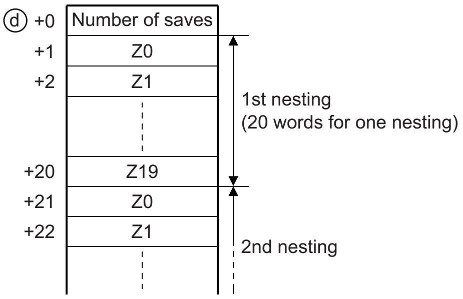
• When using Basic model QCPU



• When using High Performance model QCPU, Process CPU, and Redundant CPU



When using Universal model QCPU and LCPU



■ZPOP(P)

- Recovers the contents saved in (d) and the following devices to the index register. (When the saved content is read out to the index register, (d) + 0 (the number of saves made) is decreased by 1.)

Operation error

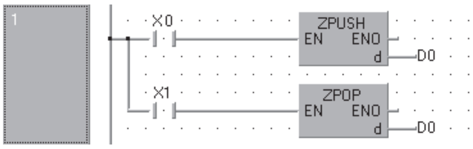
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The content of (d)+ 0 (number of saves) is 0 in the ZPOP(P) instruction.	○	○	○	○	○	○
4101	The range for the number of points to be used for (d) and later by the ZPUSH(P) instruction exceeds the corresponding device range.	○	○	○	○	○	○

Program example

- In the following program, the data of the index register which is prior to call the subroutine program are saved to D0 and the following devices, when using the index registers within the subroutine program following P0.

[Structured ladder/FBD]



[ST]
ZPUSH(X0,D0);
ZPOP(X1,D0);

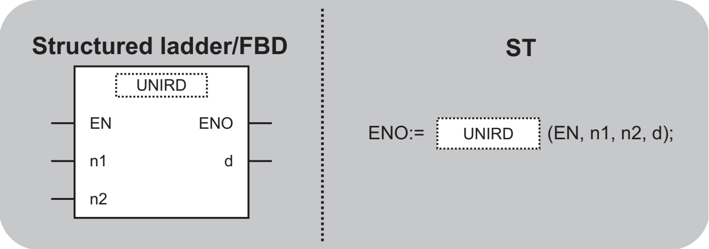
Precautions

When using the ZPUSH/ZPOP instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

Reading module information

UNIRD(P)

Basic | High performance | Process | Redundant | Universal | LCPU

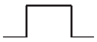
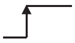


ENO:= UNIRD (EN, n1, n2, d);

Any of the following instruction can go in the dotted squares.

UNIRD, UNIRDP

■Executing condition

Instruction	Executing condition
UNIRD	
UNIRDP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	The value obtained by dividing "the start I/O number of the module information read source module" by 16 (0 to FFn)	ANY16
	n2	Number of read data (0 to 256)	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores module information	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n1	○	○		—				○	—
n2	○	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Reads the module information whose amount is specified for n2 from the module specified for n1, and stores the information to (d) and the following devices. (Reads the status of the actually mounted modules instead of the module type specified by I/O assignment.)
- The details of the module information are described as follows.

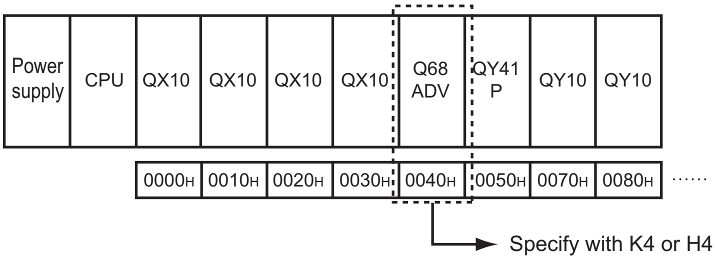
Bit	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Each module information																

Bit	Item	Description	
		QCPU (Q mode)	LCPU
b0	Number of I/O points	000: 16 points 001: 32 points 010: 48 points 011: 64 points 100: 128 points 101: 256 points 110: 512 points 111: 1024 points	
b1			
b2			
b3	Module type	000: Input module 001: Output module 010: I/O mixed module 011: Intelligent function module	000: Input module 001: Output module 011: Intelligent function module 111: CPU built-in I/O
b4			
b5			
b6	External supply power status (For future expansion)	1: External supply power is connected. 0: External supply power is not connected.	0: Fixed
b7	Presence/absence of fuse blown	1: Module's fuse is blown. 0: Normal	0: Fixed
b8	Online module change status/ execution from the standby system	1: An attempt to read module information on the extension base unit during online module change or from the CPU module of standby system in the redundant system.[*1] 0: Other than the above	0: Fixed
b9	Minor/medium error status	1: Minor/medium error occurred 0: Normal	
b10	Module error status	00: No module error 10: Medium error 01: Minor error 11: Serious error	
b11			
b12	Module standby status	1: Normal 0: Module error occurred	
b13	Empty	0: Fixed	
b14	Series type	0: Q series module	0: Fixed
b15	Module mounting status	1: Module is mounted. 0: No module is mounted.	

[*1] The Universal model QCPU used in the multiple CPU system is turned ON during the online module change of the module controlled by the other CPU.

- The value of n1 is specified by the higher 3 digits of the start I/O number of the slot from which the module information is read, when it is expressed in 4 digits in hexadecimal.

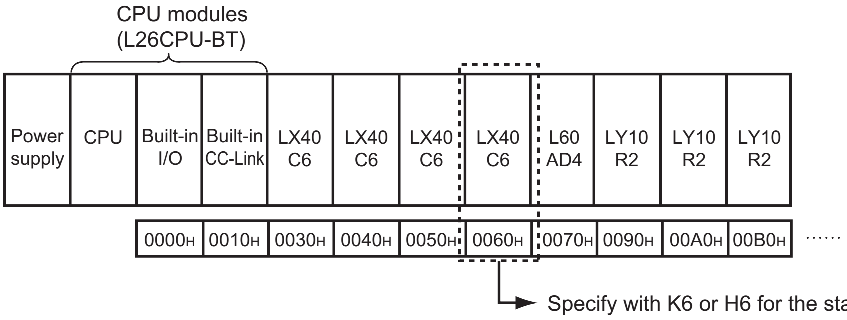
< QCPU (Q mode) >



...... Start I/O numbers of I/O assignment setting

Specify with K4 or H4 for the start I/O number of the read target.

< LCPU (L26CPU-BT) >



...... Start I/O numbers of I/O assignment setting

Specify with K6 or H6 for the start I/O number of the read target.

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

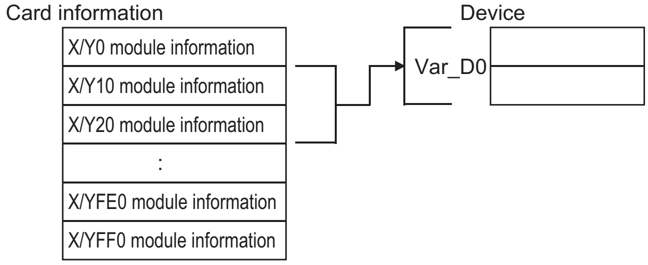
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	When n1 is other than 0 to FFH. When n2 is other than 0 to 256. When a total of n1 and n2 is equal to or greater than 257.	—	○	○	○	○	○[*2]
	When n1 is other than 0 to 3FH. When n2 is other than 0 to 64. When a total of n1 and n2 is equal to or greater than 65.	Q00/ Q01	—	—	—	—	○[*3]
	When n1 is other than 0 to FH. When n2 is other than 0 to 16. When a total of n1 and n2 is equal to or greater than 17.	Q00J	—	—	—	—	—
4101	For the device number specified for (d) and later, the range whose amount is specified for n2 exceeds the corresponding device range.	○	○	○	○	○	○

[*2] For only L06CPU, L06CPU-P, L26CPU, L26CPU-P, L26CPU-BT, L26CPU-PBT

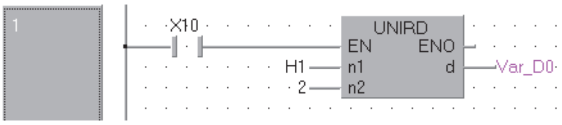
[*3] For only L02SCPU, L02SCPU-P, L02CPU, and L02CPU-P

Program example

- In the following program, the module information of I/O numbers 10H to 20H is stored to Var_D0 and the following devices when X10 turns ON.



[Structured ladder/FBD]

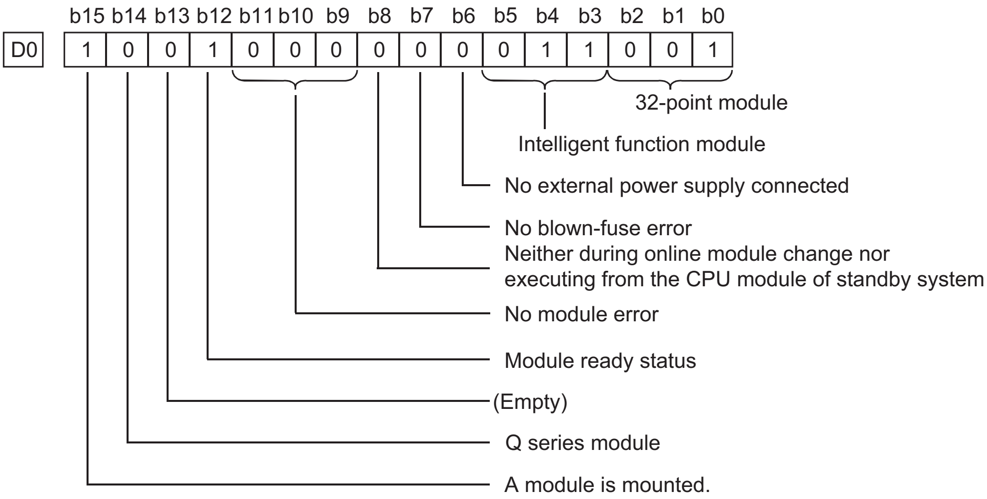


[ST]

```
UNIRD(X10,H1,2,Var_D0);
```

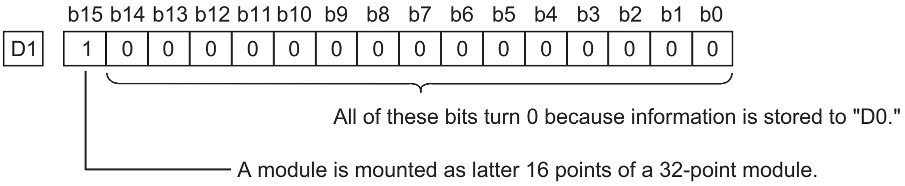
Readout result (When read to D0)

- 32-point intelligent function module for Q series



	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1

32-point module
Intelligent function module
No external power supply connected
No blown-fuse error
Neither during online module change nor executing from the CPU module of standby system
No module error
Module ready status
(Empty)
Q series module
A module is mounted.



	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	0	0	0	0	0	0	0	0

All of these bits turn 0 because information is stored to "D0."
A module is mounted as latter 16 points of a 32-point module.

- With a 48- or 64-point module, the same content as those of D1 are stored to D2 or D2 and D3 respectively.

- 32-point module for A series

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

For an A series module, all of these bits turn 0 because information is not stored.

A series module

A module is mounted.

	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	0	0	0	0	0	0	0	0

All of these bits turn 0 because information is stored to "D0."

A module is mounted as latter 16 points of a 32-point module.

- With a 48- or 64-point module, the same contents as those of D1 are stored to D2 or D2 and D3 respectively.

- Empty slot

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

For an empty slot, all of these bits turn 0.

- Online module change

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

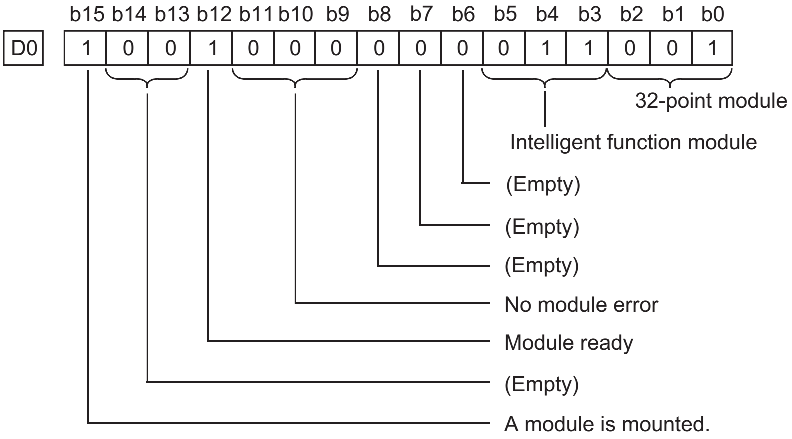
Online module change in process

- Module information on the extension base unit is tried to be read from the standby system of the redundant system in separate mode

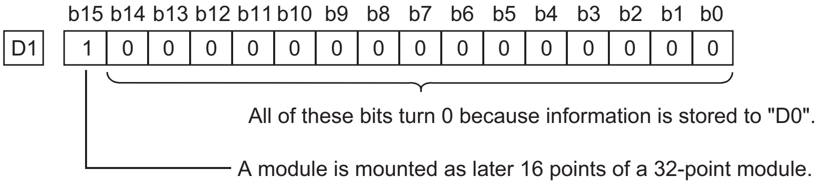
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Execution from the standby system
(Module information on the extension base unit is tried to be read from the standby system of the redundant system in separate mode.)

- 32-point intelligent function module for L series



	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1



	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	0	0	0	0	0	0	0	0

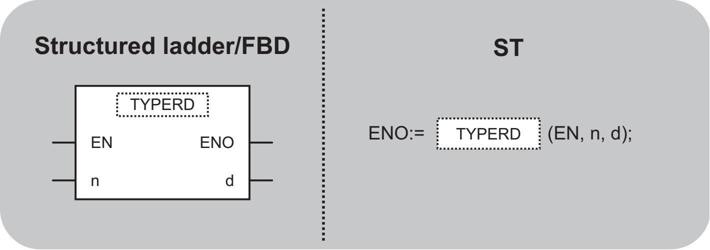
All of these bits turn 0 because information is stored to "D0".
A module is mounted as later 16 points of a 32-point module.

Reading module type

TYPERD(P)

~~Basic~~ ~~High performance~~ ~~Process~~ ~~Redundant~~ Ver. Universal LCPU

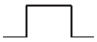
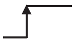
- Universal model QCPU: Supported if first 5 digits of the serial number are "11043" or later



Any of the following instruction can go in the dotted squares.

TYPER, TYPERDP

■Executing condition

Instruction	Executing condition
TYPERD	
TYPERDP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Start I/O number of the module whose module type is read[*1]	ANY16
Output argument	ENO	Execution result	Bit
	d	Instruction execution result and module type • d[0]: instruction execution result • d[1] to d[9]: module type	ANY16(0..9)

[*1] Specified by the upper 3 digits of start I/O number expressed in 4-digit hexadecimal.

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	—	○		—				○	—
(d)	—	○		—				—	—

Setting data

Setting data	Description		Setting range	Setting side	Data type
n	The value obtained by dividing "the start I/O number of the module where a module type is read" by 16		0 to FFH, 3E0 to 3E3H[*1]	User	BIN 16 bits
(d)	(d)[0]	Execution result of the instruction	Within device range	System	BIN 16 bits
	(d)[1] ⋮ (d)[9]	Module type			String

[*1] For Universal model QCPU only

Processing details

- Reads the module type of the slot specified for n, and stores data to (d) and the following devices.

For Universal model QCPU, the target modules are the following six types.

- CPU module
- Input module
- Output module
- I/O combined module
- Intelligent function module
- GOT (when bus is connected)

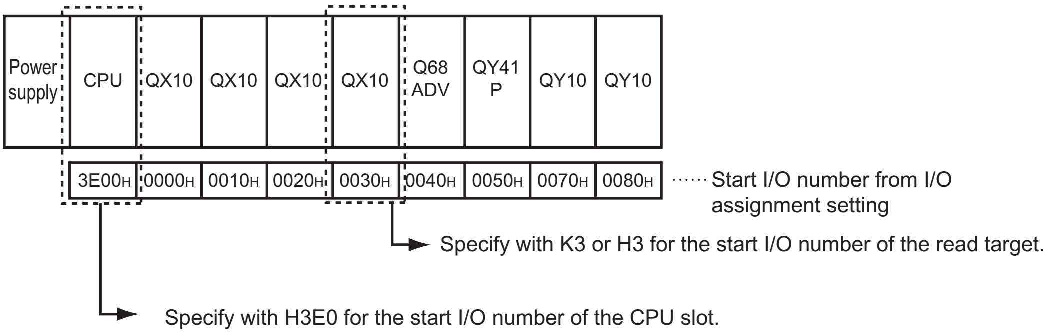
For LCPU, the target modules are the following four types.

- CPU module
- CPU module
- Input module
- Intelligent function module

- Apply the upper 3 digits of the start I/O number of the module where a module type is read (when the start I/O number is written in 4 digit hexadecimal) for n.

When specify the module with 1 slot

< Universal model QCPU >

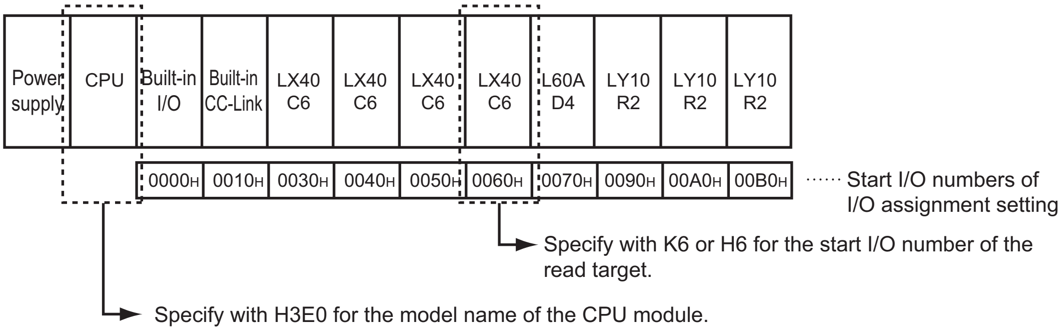


······ Start I/O number from I/O assignment setting

Specify with K3 or H3 for the start I/O number of the read target.

Specify with H3E0 for the start I/O number of the CPU slot.

< LCPU (L26CPU-BT) >



······ Start I/O numbers of I/O assignment setting

Specify with K6 or H6 for the start I/O number of the read target.

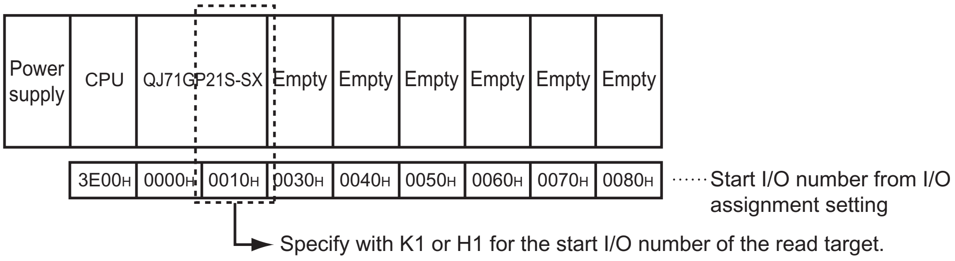
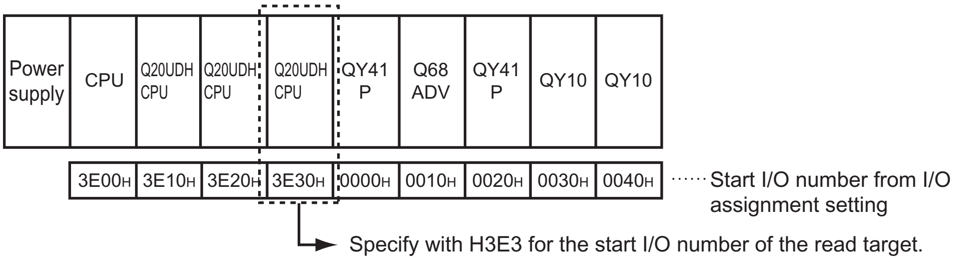
Specify with H3E0 for the model name of the CPU module.

Point

For LCPU, if the start I/O number of built-in I/O or built-in CC-Link module is specified, the CPU module model name is read.

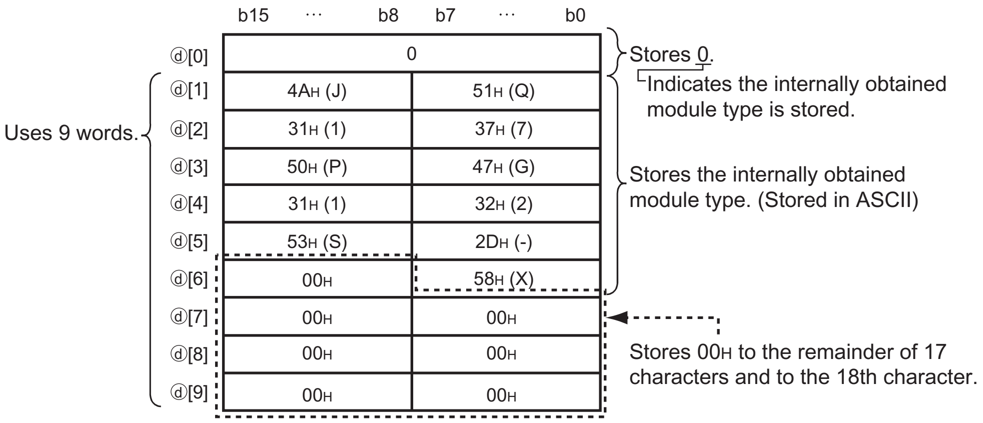
Specifying a module that occupies two slots

The start I/O number specified for read target module may be different from the start I/O number of the mounted slot.
For the I/O start number to be specified, refer to the manual for the module.
Specify the upper 3 digits of the start I/O number of the read target module expressed in 4-digit hexadecimal.

When specifying QJ71GP21S-SX	The start I/O number to be specified is a value to which 0010H of the mounted module is added.  Power supply CPU QJ71GP21S-SX Empty Empty Empty Empty Empty Empty 3E00H 0000H 0010H 0030H 0040H 0050H 0060H 0070H 0080H Start I/O number from I/O assignment setting Specify with K1 or H1 for the start I/O number of the read target.
Reading CPU module type for multiple CPU system configuration	Specify the upper 3 digits of the start I/O number of each CPU expressed in 4-digit hexadecimal.  Power supply CPU Q20UDH CPU Q20UDH CPU Q20UDH CPU QY41 P Q68 ADV QY41 P QY10 QY10 3E00H 3E10H 3E20H 3E30H 0000H 0010H 0020H 0030H 0040H Start I/O number from I/O assignment setting Specify with H3E3 for the start I/O number of the read target. The module type can be read by specifying the start I/O number of the module controlled by the other CPU.

- Stores instruction execution result to (d)[0], and module type to (d)[1] to (d)[9]. The following shows the values to be stored to (d).

  - When the read target module obtains its module type internally (Ex. QJ71GP21-SX)



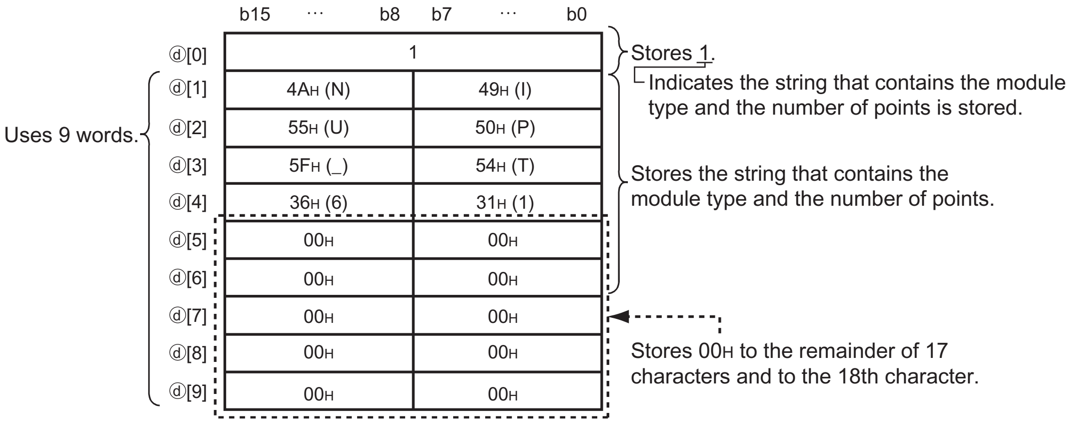
	b15 … b8	b7 … b0
ⓓ[0]	0	
ⓓ[1]	4AH (J)	51H (Q)
ⓓ[2]	31H (1)	37H (7)
ⓓ[3]	50H (P)	47H (G)
ⓓ[4]	31H (1)	32H (2)
ⓓ[5]	53H (S)	2DH (-)
ⓓ[6]	00H	58H (X)
ⓓ[7]	00H	00H
ⓓ[8]	00H	00H
ⓓ[9]	00H	00H

Uses 9 words.
Stores 0.
Indicates the internally obtained module type is stored.
Stores the internally obtained module type. (Stored in ASCII)
Stores 00H to the remainder of 17 characters and to the 18th character.

The following table shows the module type example to be stored to (d)[1] to (d)[9].

Target module	Module type example
CPU module	Q06UDEHCPU
Intelligent function module	QJ71GP21-SX
GOT	GOT1000

  - When the read target module does not obtain its module type internally (Ex. QX40)



	b15 … b8	b7 … b0
ⓓ[0]	1	
ⓓ[1]	4AH (N)	49H (I)
ⓓ[2]	55H (U)	50H (P)
ⓓ[3]	5FH (_)	54H (T)
ⓓ[4]	36H (6)	31H (1)
ⓓ[5]	00H	00H
ⓓ[6]	00H	00H
ⓓ[7]	00H	00H
ⓓ[8]	00H	00H
ⓓ[9]	00H	00H

Uses 9 words.
Stores 1.
Indicates the string that contains the module type and the number of points is stored.
Stores the string that contains the module type and the number of points.
Stores 00H to the remainder of 17 characters and to the 18th character.

In (d)[1] to (d)[9], the string that contains the module type and the number of points is stored.

[Strings that indicate module type]

- Input module: INPUT
- Output module: OUTPUT
- I/O combined module: MIXED
- Intelligent function module (includes the QI60 and GOT): INTELLIGENT

[Strings that indicate the number of points]

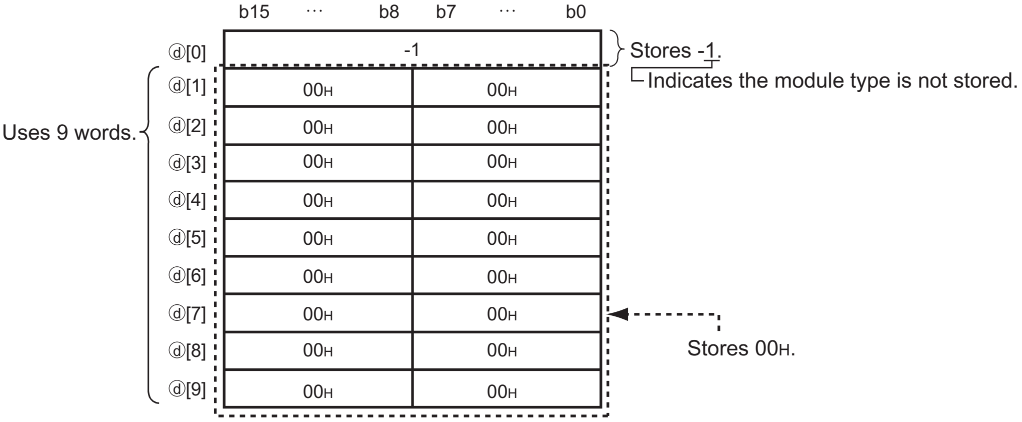
- 16 points: _16
- 32 points: _32
- 64 points: _64
- 128 points: _128
- 256 points: _256
- 512 points: _512
- 1024 points: _1024

The following table shows the string example to be stored to (d)[1] to (d)[9].

Target module	Stored model name
Input module	INPUT_16
Output module	OUTPUT_32
I/O combined module	MIXED_64
Intelligent function module	INTELLIGENT_128

- Others

For an empty slot, or when performing the online program change function
When n is not a start I/O number of module
When n is a value within the setting range, but cannot be set for I/O assignment of PLC parameter



Operation error

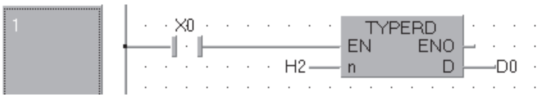
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2110	When the read target module cannot be communicated because of the failure.	—	—	—	—	○	○
4101	When exceeding the device range of 10 words from the device specified for (d).	—	—	—	—	○	○
	When specifying a value other than 0 to FFH, 3E0H to 3E3H for n.	—	—	—	—	○	—
	When specifying a value other than 0 to FFH, 3E0H for n.	—	—	—	—	—	○

Program example

- In the following program, the module type of the module mounted to the slot with the start I/O number 0020H is stored to D0 and the following devices when X10 turns ON.

[Structured ladder/FBD]



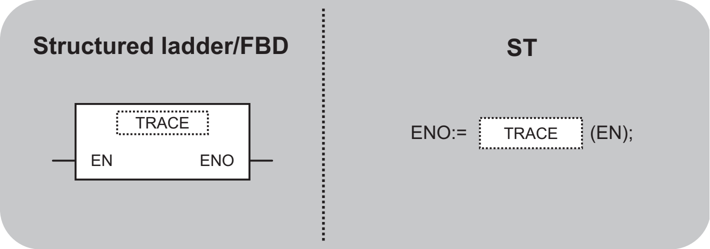
[ST]
TYPERD(X0,H2,D0);

Trace set/reset

TRACE, TRACER

Basic High performance Process Redundant Ver. Universal LCPU

- Universal model QCPU: Not supported for Q00UJCPU



Any of the following instruction can go in the dotted squares.

TRACE, TRACER

■Executing condition

Instruction	Executing condition
TRACE, TRACER	↑

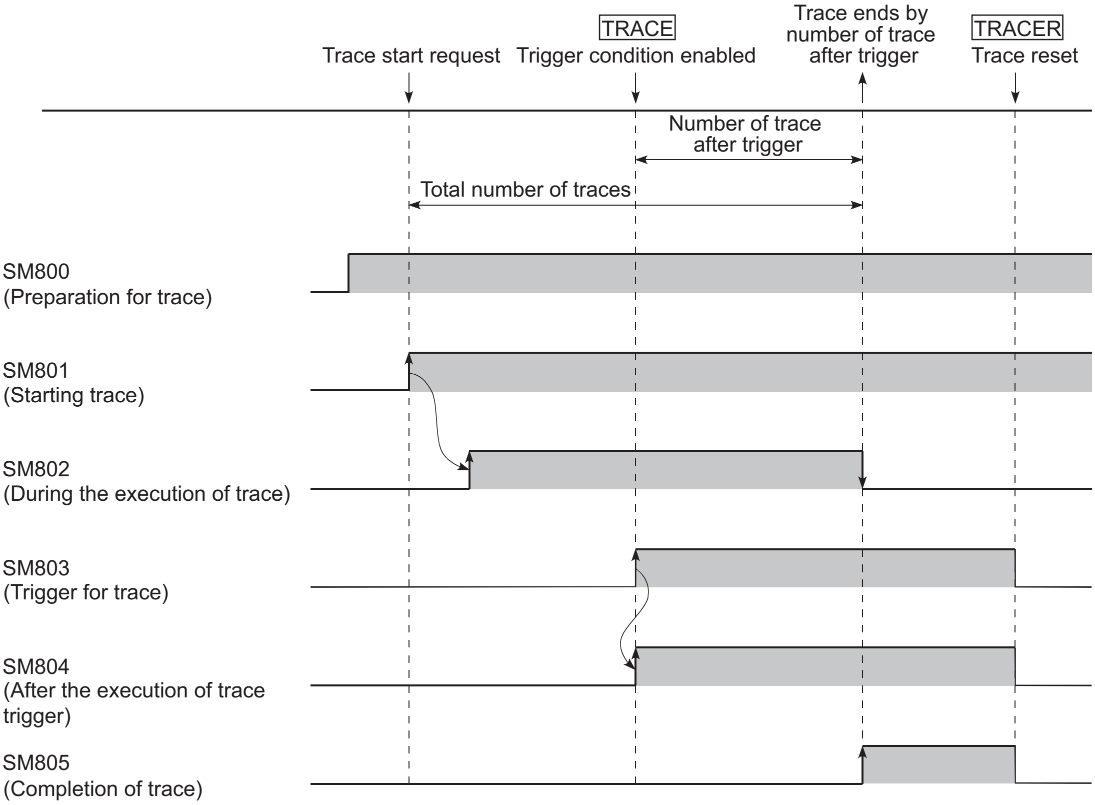
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

- The sampling trace function collects the specified device data of a CPU module consecutively.
- To execute the sampling trace, turn on SM801 while SM800 is turned on.



■TRACE

- The TRACE instruction turns on SM803. In addition, the instruction stops the sampling trace by latching the sampling trace result, after sampling the number of the sampling trace (of after the set TRACE instruction is executed).
- The sampling is stopped if SM801 turns OFF during the trace execution.
- After the TRACE instruction is executed and the sampling trace is stopped, SM805 is turned on.
- Once the TRACE instruction is executed, the second and the subsequent TRACE instructions are ignored. When the TRACER instruction is executed, the TRACE instruction is enabled again.

■TRACER

- The TRACER instruction resets the TRACE instruction. When the TRACER instruction is executed, the TRACE instruction is enabled again.
- When the TRACER instruction is executed, SM803 to SM805 are turned OFF.

Point

- The target devices for the sampling trace and its timing can be set with a programming tool. For details of the sampling trace, refer to the user's manual (Function Explanation, Program Fundamentals) for the CPU module used.
- The sampling trace can be executed with a programming tool. For sampling trace execution with a programming tool, refer to the operating manual for the programming tool used.

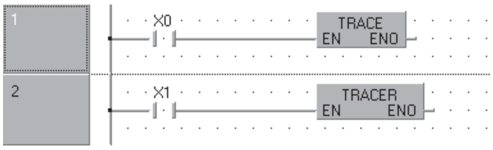
Operation error

- There is no operation error.

Program example

- In the following program, the TRACE instruction is executed when X0 turns ON, and the TRACE instruction is reset with the TRACER instruction when X1 turns ON.

[Structured ladder/FBD]



[ST]

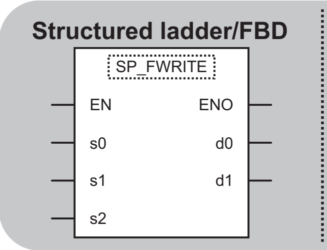
```
TRACE(X0);
TRACER(X1);
```

Writing data to specified file

SP_FWRITE

Basic High performance Process Redundant Ver. Universal Ver. LCPU

- Universal model QCPU: Not supported for Q00UJCPU, Q00UCPU, and Q01UCPU
- Built-in Ethernet port LCPU: Supported
- LCPU: Not supported for L02SCPU and L02SCPU-P



ST

ENO:= SP_FWRITE (EN,s0,s1,s2,d0,d1);

The following instruction can go in the dotted squares.

SP_FWRITE

■Executing condition

Instruction	Executing condition
SP_FWRITE	↑

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s0	Drive specification	ANY16
	s1	File name	String
	s2	Number of data to request writing	ANY16
Output argument	ENO	Execution result	Bit
	d0	Control data	Array of ANY16 (0..7)
	d1	Array to be turned ON at the process completion	Array of bit (0..1)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant		Others
	Bit	Word		Bit	Word			K, H	$	
(s0)	○	○		—				○	—	—
(s1)	—	○		—				—	—	—
(s2)	—	△[*1]		—				—	○	—
(d0)	—	△[*2]		—				—	—	—
(d1)	△[*2]	△[*2]		—				—	—	—

*1 For the Universal model QCPU and LCPU, only when the number of request write data exceeds 1024, local devices and file registers set in each program cannot be used.

*2 Local devices and file registers per program cannot be used as setting data.

Setting data

<table>
<tr><th>Setting data</th><th colspan="3">Description</th><th>Setting range</th><th>Setting side</th><th>Data type</th></tr>
<tr><td>(s0)</td><td colspan="3">Drive specification</td><td>2</td><td>User</td><td rowspan="8">BIN 16 bits</td></tr>
<tr><td rowspan="3">(s1)</td><td colspan="5">Start number of the device that stores data. Written data are expressed as follows.</td></tr>
<tr><td>Device</td><td>Item</td><td>Content/Setting data</td><td>Setting range</td><td>Setting side</td></tr>
<tr><td>(s1) to (s1)+□</td><td>File name character string</td><td>Specify the character string of a file name.
• When omitting an extension, also omit the "." (Period).
• Limit the file name within 8 characters + period + 3 characters.
• When 9 or more characters are used, the extension is ignored regardless of its presence, and "BIN" or "CSV" is automatically assigned as an extension.</td><td>Character string</td><td>User</td></tr>
<tr><td rowspan="4">(s2)</td><td colspan="5">Start number of the device that stores a file name. A file name is expressed as follows.</td></tr>
<tr><td>Device</td><td>Item</td><td>Content/Setting data</td><td>Setting range</td><td>Setting side</td></tr>
<tr><td>(s2)</td><td>Number of request write data</td><td>Specify the number of data to request writing. (Unit: word)
This data should be set in unit of word even when byte is specified for (d0)[7].</td><td>1 to 480
1 to 32707*3</td><td rowspan="2">User</td></tr>
<tr><td>(s2)+1 to (s2)+□</td><td>Write data</td><td>Data to request writing.</td><td>0000H to FFFFH</td></tr>
<tr><td rowspan="9">(d0)</td><td colspan="5">Start number of the device that stores control data. The following control data are required.</td><td rowspan="9">BIN 16 bits</td></tr>
<tr><td>Device</td><td>Item</td><td>Content/Setting data</td><td>Setting range</td><td>Setting side</td></tr>
<tr><td>(d0)[0]</td><td>Execution/ completion type</td><td>Specify the execution type.
0000H: Write binary data
0100H: Write data after CSV format conversion</td><td>0000H
0100H</td><td>User</td></tr>
<tr><td>(d0)[1]</td><td>(Not used)</td><td>Used by system</td><td>—</td><td>System</td></tr>
<tr><td>(d0)[2]</td><td>Writing result (number of written data)</td><td>Contains the number of actually written data against the data specified for (s2) [7]. The unit for the value is determined by the data type specification.</td><td>—</td><td>System</td></tr>
<tr><td>(d0)[3]</td><td>(Not used)</td><td>—</td><td>—</td><td>—</td></tr>
<tr><td>(d0)[4]
(d0)[5]</td><td>File position</td><td>When binary data write is specified for (d0)[0]
• Set the file position.
00000000H: Start of the file.
00000001H to FFFFFFFEH: From the specified position The unit for the value is determined by the data type specification.
FFFFFFFFH: Addition starts from the end of the file.
When CSV format write is specified for (d0)[0]
• For the High Performance model QCPU of which the first 5 digits of the serial number are '01111' or earlier, always set the beginning (0H) of the file.
• For the High Performance model QCPU/ Process CPU/Redundant CPU/Universal model QCPU/LCPU of which the first 5 digits of the serial number are '01112' or later, set the file position.
00000000H to FFFFFFFEH: Start of the file
FFFFFFFFH: Addition from the end of the file</td><td>00000000H to FFFFFFFFH</td><td>User</td></tr>
<tr><td>(d0)[6]</td><td>Number of columns specification</td><td>When binary write is specified for (d0)[0], always set 0.
When CSV format write is specified for (d0)[0], set the number of columns where data will be written.
0: No columns. Regarded as one row.
Other than 0: Set to the specified number of columns.</td><td>0H to FFFFH
(0 to 65535)</td><td>User</td></tr>
<tr><td>(d0)[7]</td><td>Data type specification</td><td>0: Word
1: Byte</td><td>0,1</td><td>User</td></tr>
</table>

<table>
<tr><th>Setting data</th><th colspan="3">Description</th><th>Setting range</th><th>Setting side</th><th>Data type</th></tr>
<tr><td rowspan="4">(d1)</td><td colspan="5">Bit device that turned ON at the completion of the processing. ((d)[1] is also turned ON at error completion.)</td><td rowspan="4">Bit</td></tr>
<tr><td>Device</td><td>Item</td><td>Content/Setting Data</td><td>Setting range</td><td>Setting side</td></tr>
<tr><td>(d1)[0]</td><td>Completion signal</td><td>Indicates the completion of the processing.
ON: Completed
OFF: Not completed</td><td>—</td><td rowspan="2">System</td></tr>
<tr><td>(d1)[1]</td><td>Error completion signal</td><td>Indicates whether the processing is normally completed or abnormally completed.
ON: Error completion
OFF: Normal completion</td><td>—</td></tr>
</table>

*3 Indicates the range applicable for Universal model QCPU and LCPU.

Precautions

- For QCPU (Q mode), only the ATA card drive (2) can be set for (s0) (drive specification). Note that when the Flash card is installed, the SP_FWRITE instruction cannot be used to perform writing. The SRAM card, standard RAM or standard ROM drive cannot be set. For High-speed Universal model QCPU, Universal model Process CPU, and LCPU, only the SD memory card drive (2) can be set for (s0) (drive specification).
- For CSV setting, the data written are decimal values.

Ex.

Character "A" (41H) → 65 is written

Handling range: -32768 to 32767

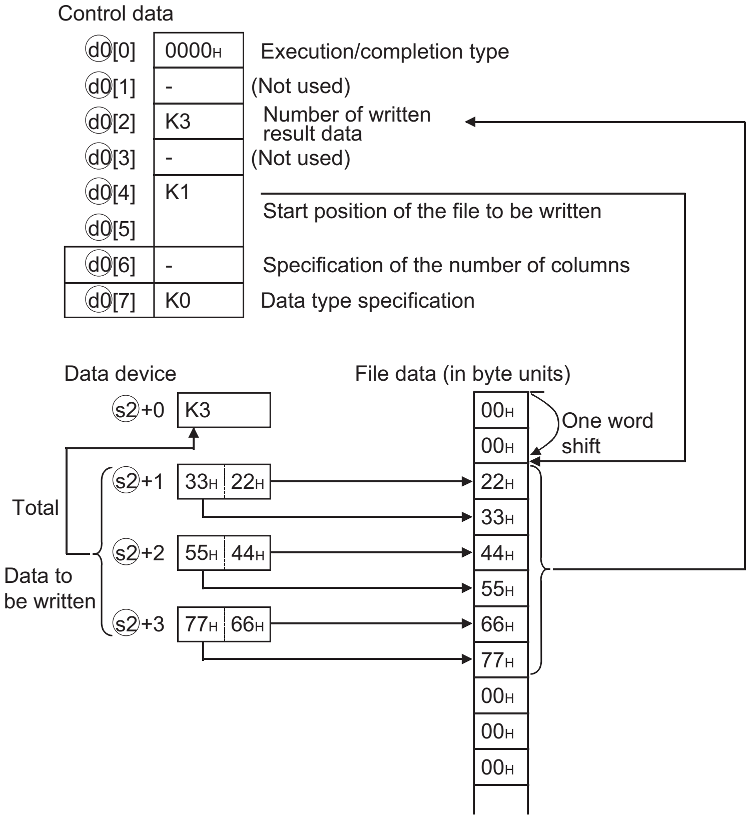
- For binary write, the setting range of the file position with word specification is 00000000H to 7FFFFFFFH and FFFFFFFFH.
- For High-speed Universal model QCPU, Universal model Process CPU, and LCPU, this instruction will not be executed while SM606 (SD memory card forced disable instruction) is on. If executed, no processing is performed.

Processing details

- The specified number of data is written to the specified file. Set the execution/completion type in the control data to specify whether to write binary data without any conversion or to convert binary data to CSV format data before writing it. (The writing target is the ATA card only for QCPU (Q mode), and the SD memory card only for High-speed Universal model QCPU, Universal model Process CPU, and LCPU.)
- The execution completion bit device (d1) is automatically turned on at the END processing after the completion of the instruction is detected. The bit device is turned off at the END processing in the next scan.
  - When this instruction is completed abnormally, the error completion device (d1)[1] is turned ON/OFF in synchronization with the processing complete (d1)[0] device. Use this device as the error completion flag for this instruction.
  - SM721 is turned ON during the execution of the instruction. This instruction cannot be executed while SM721 is ON. (If an attempt is made, no processing is performed.)
  - When an error is detected at the execution of the instruction (before SM721 is turned ON), the processing complete device (d1)[0], the error completion device (d)[1], and SM721 are not turned ON.
- Be sure to use units of word to specify the number of request write data (s2), and the file position ((d0)[4] and (d0)[5]).

■When writing binary data

- If the extension of the target file is omitted, ".BIN" is used as an extension.
- When the specified file does not exist, a new file is created and the data are additionally saved from the beginning of the file. The attributes of this new file are set using the archive attributes.
- When the size of the data exceeds that of the existing area in the file during the writing, the excess data are additionally saved.
- If the file position specified is greater than the existing file size:
  - The High Performance model QCPU of which the first 5 digits of the serial number are '01111' or lower results in an error.
  - The High Performance model QCPU, Process CPU, Redundant CPU, and Universal model QCPU of which the first 5 digits of the serial number are "01112" or later performs writing at point 0 and is completed normally.
- An error occurs when the saving space becomes full while data are additionally saved. In such a case, the data that are additionally saved successfully remains in the medium. The error completion is indicated after additionally saving the data as much as possible.
- The following shows the method for writing binary data when No. of request write data and file position are specified.



■When writing data after CSV format conversion

- If the extension is omitted, ".CSV" is used as an extension.
- When the existing file is specified:

[High Performance model QCPU of which the first 5 digits of the serial number are '01111' or lower]

File content are all deleted and data are saved from the beginning of the file.

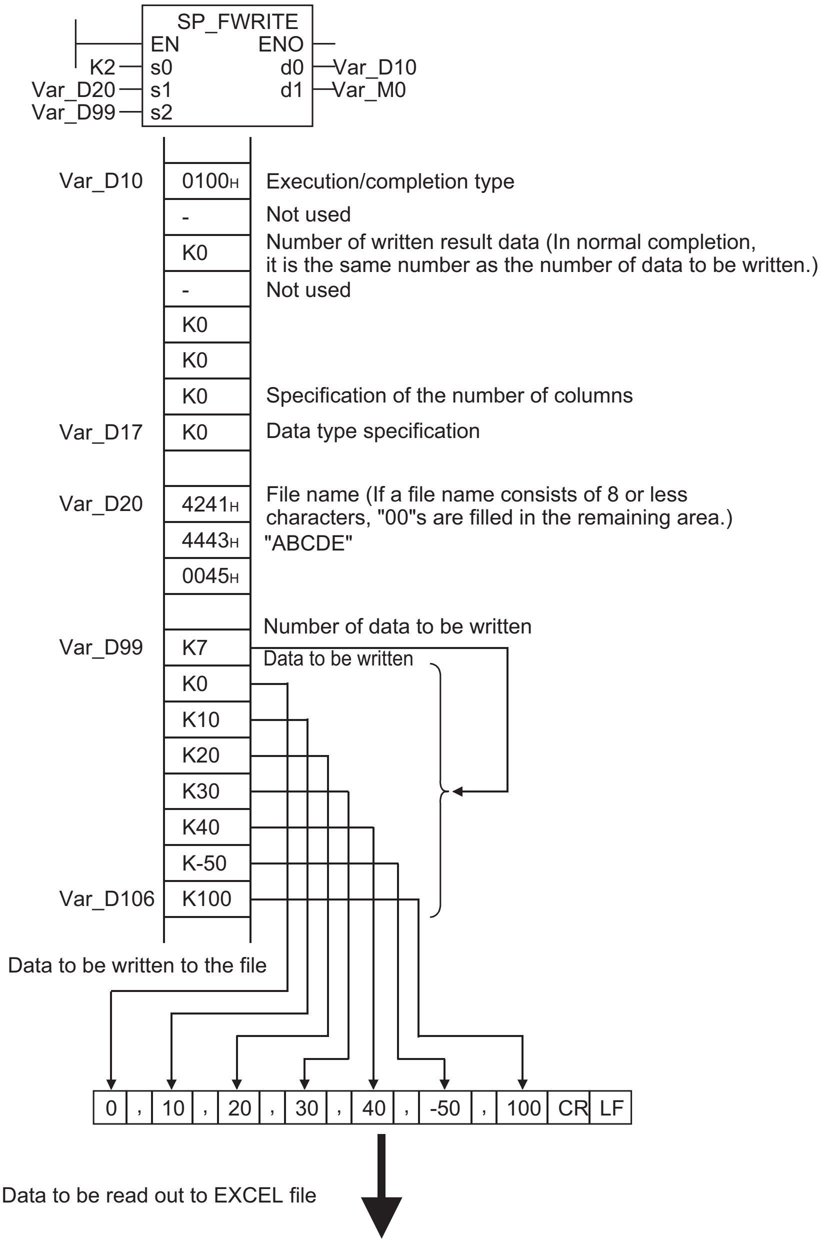
[High Performance model QCPU and Universal model QCPU, Process CPU, Redundant CPU, and LCPU of which the first 5 digits of the serial number are "01112" or later]

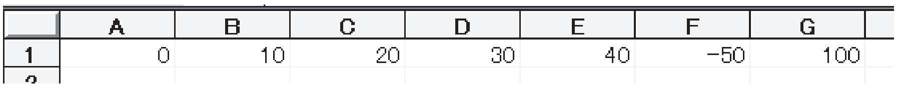
  - When other than FFFFFFFFH is set at ((d0)[4], (d0)[5]), file content are all deleted and data are saved from the beginning of the file.
  - When FFFFFFFFH is set at ((d0)[4], (d0)[5]), data are saved from the end of the file.
- When the specified file does not exist, a new file is created and the data are additionally saved from the beginning of the file. The attributes of this new file are set using the archive attributes.
- An error occurs when the saving space becomes full while data are additionally saved. In such a case, the data that are additionally saved successfully remains in the medium. The error completion is indicated after additionally saving the data as much as possible.

- When the specified number of columns is 0, the data are stored as single-column data in CSV format file.

Ex.

When data are written after CSV format conversion and the specified number of columns is 0.


SP_FWRITE
EN
ENO
K2
s0
d0
Var_D10
Var_D20
s1
d1
Var_M0
Var_D99
s2
Var_D10
0100H
Execution/completion type
-
Not used
K0
Number of written result data (In normal completion, it is the same number as the number of data to be written.)
-
Not used
K0
K0
K0
Specification of the number of columns
Var_D17
K0
Data type specification
Var_D20
4241H
File name (If a file name consists of 8 or less characters, "00"s are filled in the remaining area.)
4443H
"ABCDE"
0045H
Number of data to be written
Var_D99
K7
Data to be written
K0
K10
K20
K30
K40
K-50
Var_D106
K100
Data to be written to the file
0 , 10 , 20 , 30 , 40 , -50 , 100 CR LF
Data to be read out to EXCEL file

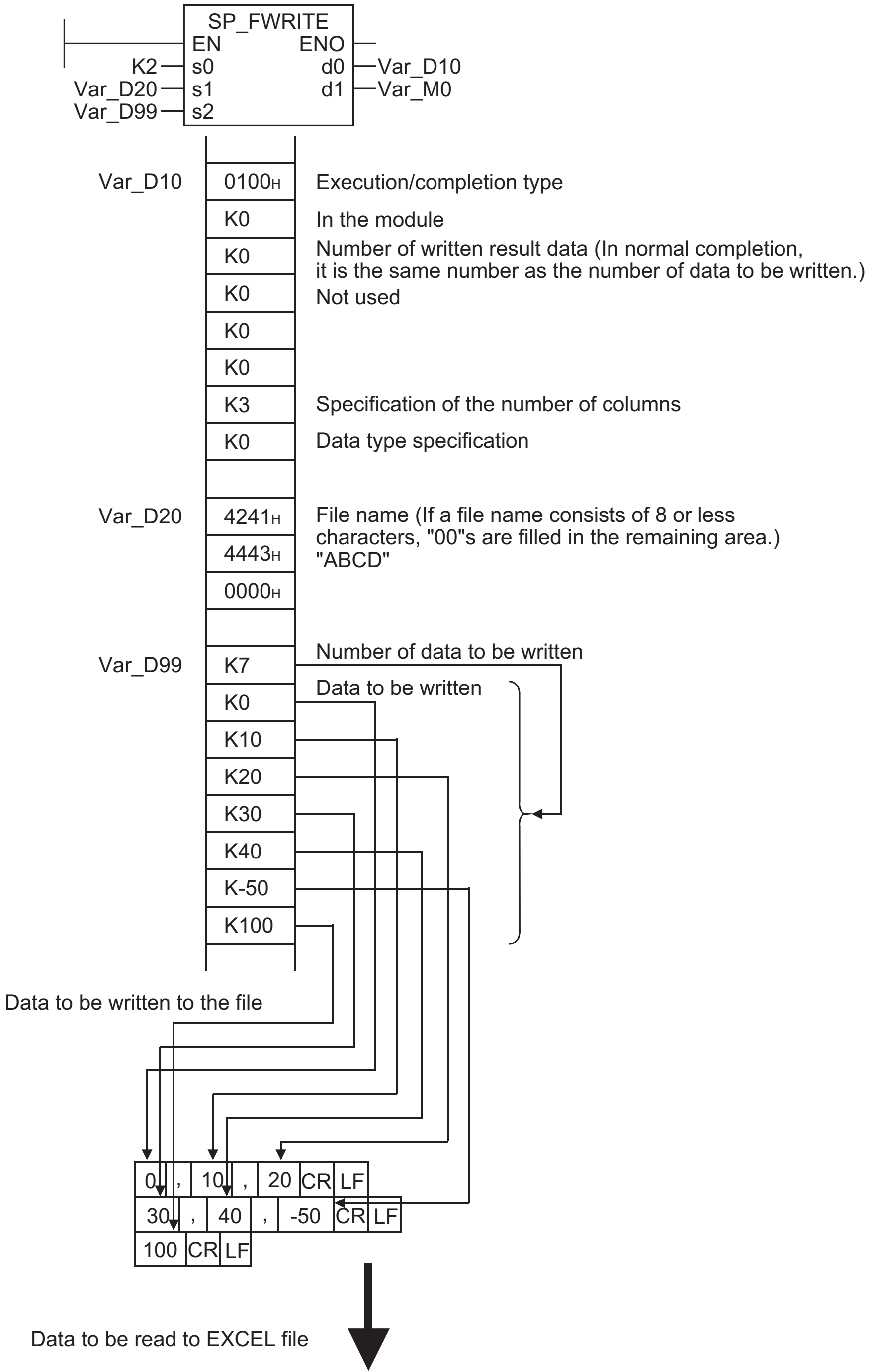


	A	B	C	D	E	F	G
1	0	10	20	30	40	-50	100
2							

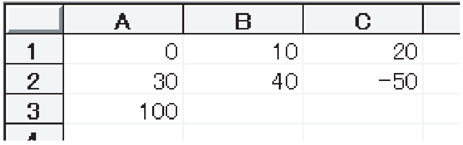
- When data are written after CSV format conversion and the specified number of columns is other than 0, the data are stored as table data with specified number of columns in a CSV format file.

Ex.

When data are written after CSV format conversion and the specified number of columns is other than 0.


SP_FWRITE
EN
ENO
K2
s0
d0
Var_D10
Var_D20
s1
d1
Var_M0
Var_D99
s2
Var_D10
0100H
Execution/completion type
K0
In the module
K0
Number of written result data (In normal completion, it is the same number as the number of data to be written.)
K0
Not used
K0
K0
K3
Specification of the number of columns
K0
Data type specification
Var_D20
4241H
4443H
0000H
File name (If a file name consists of 8 or less characters, "00"s are filled in the remaining area.) "ABCD"
Var_D99
K7
Number of data to be written
K0
Data to be written
K10
K20
K30
K40
K-50
K100
Data to be written to the file
0 , 10 , 20 CR LF
30 , 40 , -50 CR LF
100 CR LF

Data to be read to EXCEL file



	A	B	C
1	0	10	20
2	30	40	−50
3	100		
4			

- When data are added by the High Performance model QCPU, Process CPU, Redundant CPU, Universal model QCPU, and LCPU of which the first 5 digits of the serial number are "01112" or later.

Ex.

Example when data are added by the High Performance model QCPU, Process CPU, Redundant CPU, Universal model QCPU, and LCPU with a serial number (first five digits) of "01112" or later

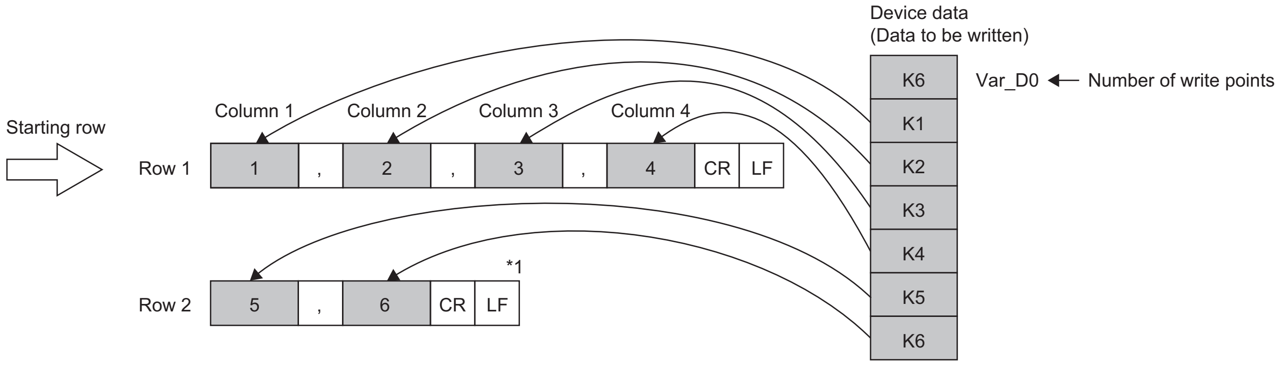
[First write]

Writing VAR_D0 data to a CSV file with the following setting

Argument	Description	Set value	
(s2)	Number of request write data	6H	6 points[*1]

Control data	Description	Set value	
(d0)[0]	Execution/completion type	0100H	Writing data after CSV format conversion
(d0)[4] (d0)[5]	File position	0H	Writing data from the start of a file
(d0)[6]	Number of columns specification	4H	4 columns[*1*3]
(d0)[7]	Data type specification	0	Word

The following figure shows the operation when the SP_FWRITE instruction is executed.



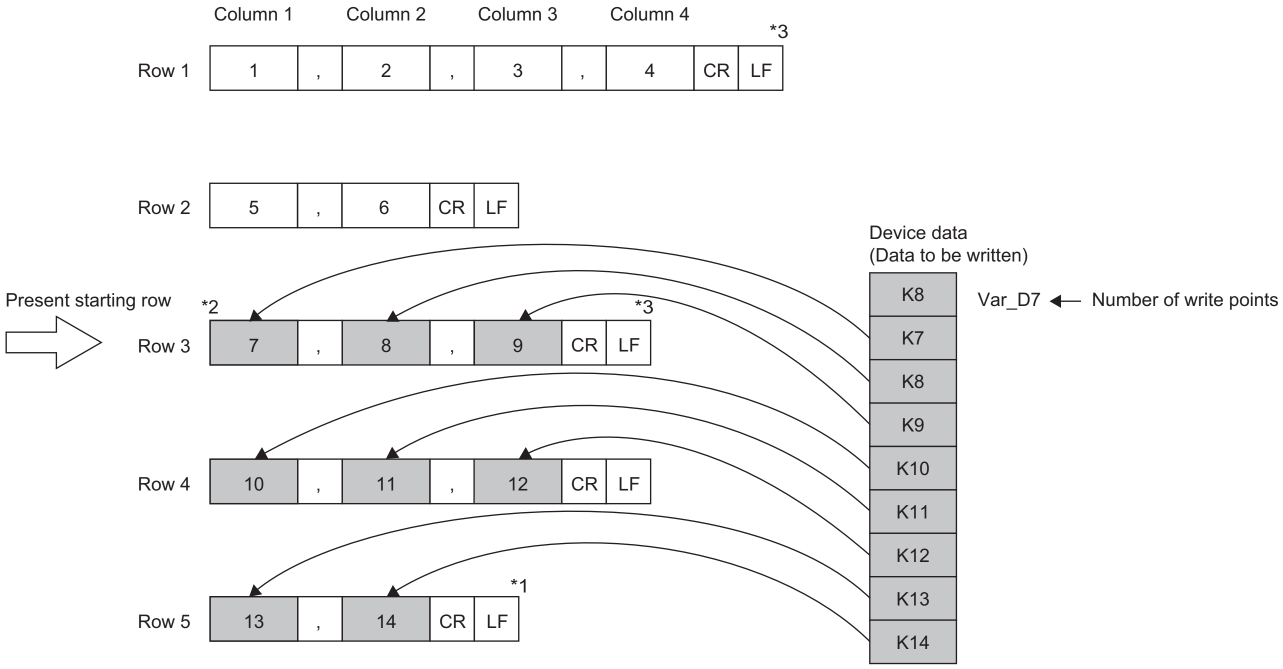
[Second write]

Writing VAR_D7 data to a CSV file with the following setting

Argument	Description	Set value	
(s2)	Number of request write data	8H	8 points*1

Control data	Description	Set value	
(d0)[0]	Execution/completion type	0100H	Data write after CSV format conversion
(d0)[4] (d0)[5]	File position	FFFFFFFFH	Adding data from the end of a file
(d0)[6]	Number of columns specification	3H	3 columns*1*3
(d0)[7]	Data type specification	0	Word

The following figure shows the operation when the SP_FWRITE instruction is executed.



*1 Unless the "number of write points" is set to an integral multiple of "column specification", the column numbers will be random.

*2 Since the last datum is always followed by the row feed code, addition normally starts at the beginning of the new row in the addition mode.

*3 If, in the addition mode, "column specification" is changed from that in the previous writing, the column numbers are shifted.

- Do not execute the SP_FWRITE instruction in an interrupt program. (Otherwise, a malfunction may result.)
- The following shows the calculation method of the file size (total bytes) when writing CSV format file to ATA card.

Total bytes = total bytes except the last line + number of bytes of the last line

(Number of bytes of each line = number of columns*4 + 1 + total bytes of each data value in a line*5)

*4 Lines except the last line are specified number of columns. The number of columns of the last line may differ from the number of columns specified in the number of data to be written, therefore, calculated as follows.

(1) Calculate lines except the last line. Lines except the last line = required number of data to be written ÷ number of columns (reminder is rounded down)

(2) Calculate the number of columns of the last line. Number of columns of the last line = required number of data to be written - (lines except the last line × number of columns)

*5 The number of bytes of each data value is calculated as follows.

Sign of data value	Number of bytes of each data value	Range of number of bytes	Example
Positive	Number of digits	1 to 5 (When word is specified) 1 to 3 (When byte is specified)	12345...5 bytes 67...2 bytes
Negative	Digits+1	2 to 6 (When word is specified) 2 to 4 (When byte is specified)	-12345...6 bytes -67...3 bytes

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

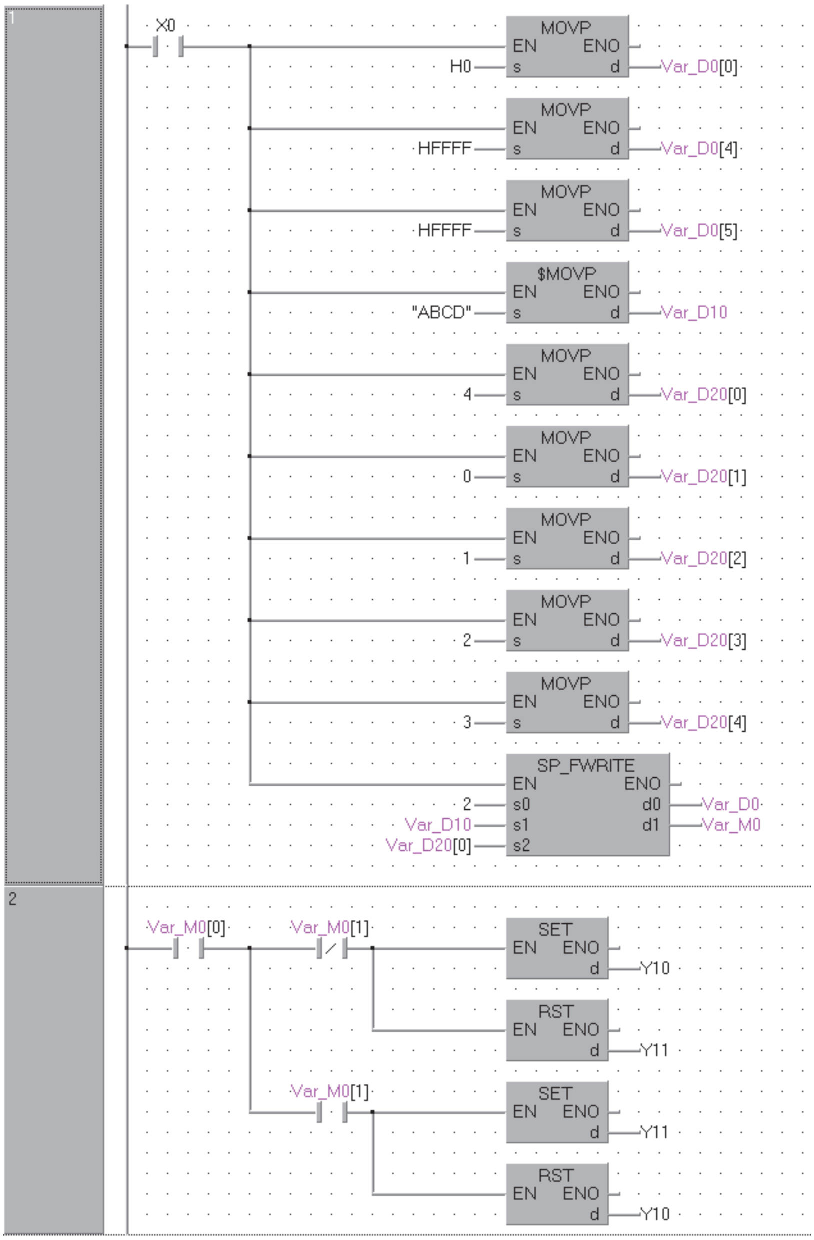
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4004	When the device which cannot be specified is specified	—	○	○	○	○	○
4100	Values specified for control data (d0) and the following devices are out of the setting range. No free space is found when an attempt is made to create a new file. An unusable value is set for a file name (s1). The attribute of a file name (s1) is "read only".	—	○	○	○	○	○
	Drive specified for drive specification device (s0) contains the medium other than the ATA card. Empty space in the ATA card is insufficient. Access error occurred in the ATA card.	—	○	○	○	○	—
	Drive specified for drive specification device (s0) contains the medium other than the SD memory card. Empty space in the SD memory card is insufficient. Access error occurred in the SD memory card.	—	—	—	—	○	○
	The file being used other functions is accessed by the SP_FWRITE instruction.	—	—	—	—	○	○
	Data are written to the SD memory card by the SP_FWRITE instruction, when the write-protect switch on the card is set to lock (write inhibited).	—	—	—	—	○	○
4101	Value specified for "number of request write data" (s2) is out of the setting range, or exceeds the device range specified for (s2)+1 and the following devices.	—	○	○	○	○	○
	The device specified for (d0) or (d1) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, 4 bytes of binary data (00H, 01H, 02H, and 03H) are added to the file "ABCD.BIN" in the memory card inserted to drive 2 when X10 turns ON.

• Assume that 8 points from (d0) are reserved for the control data devices.

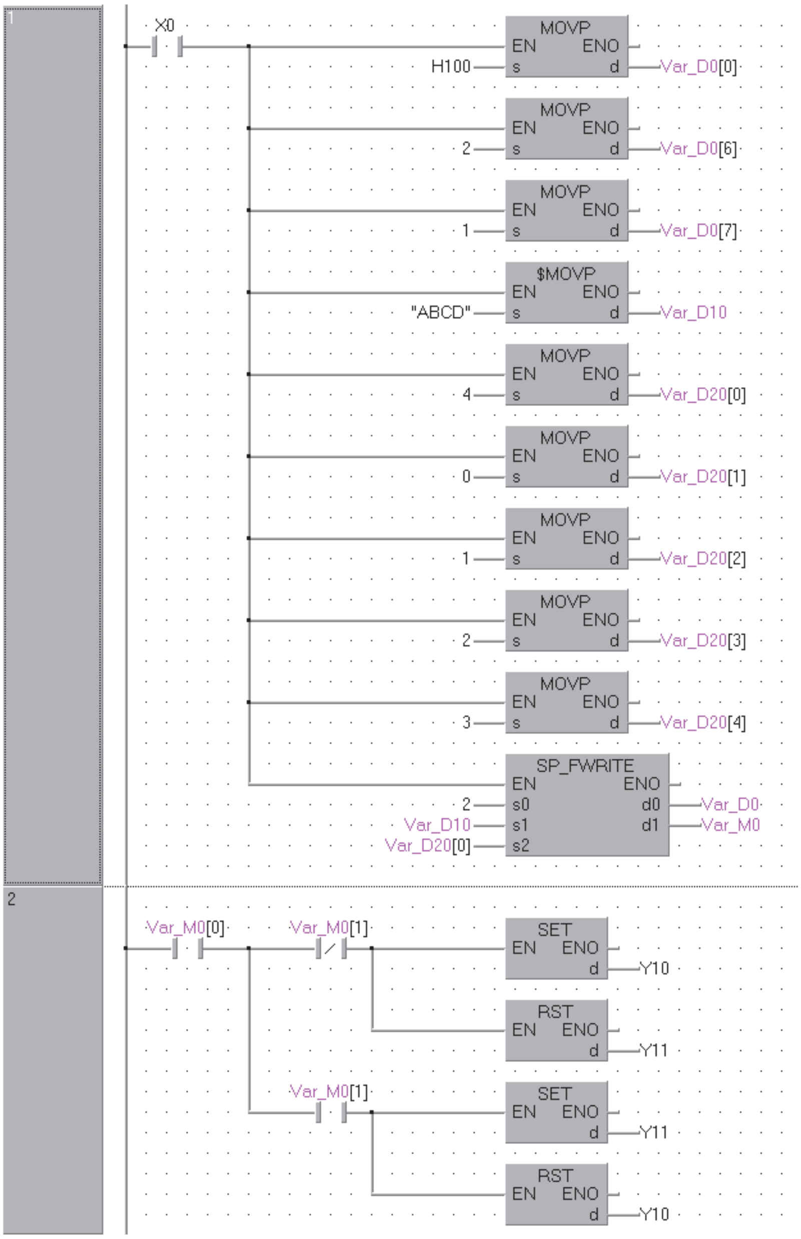
[Structured ladder/FBD]



```
[ST]
IF X10 THEN
    MOVP(TRUE,H0,Var_D0[0]);
    MOVP(TRUE,HFFFF,Var_D0[4]);
    MOVP(TRUE,HFFFF,Var_D0[5]);
    Var_D10:="ABCD";
    MOVP(TRUE,4,Var_D20[0]);
    MOVP(TRUE,0,Var_D20[1]);
    MOVP(TRUE,1,Var_D20[2]);
    MOVP(TRUE,2,Var_D20[3]);
    MOVP(TRUE,3,Var_D20[4]);
    SP_FWRITE(TRUE,2,Var_D10,Var_D20[0],Var_D0,Var_M0);
END_IF;
IF Var_M0[0] THEN
    IF(Var_M0[1]=FALSE)THEN
        SET(TRUE, Y10);
        RST(TRUE, Y11);
    ELSE
        SET(TRUE, Y11);
        RST(TRUE, Y10);
    END_IF;
END_IF;
```

- In the following program, 4 bytes of data (00H, 01H, 02H, and 03H) are created as a file name "ABCD.CSV" in the memory card inserted to drive 2 as two-column table data in CSV format, when X10 turns ON.
  - Assume that 8 points from (d0) are reserved for the control data devices.

[Structured ladder/FBD]


1
X0
MOVP EN ENO H100 s d Var_D0[0]
MOVP EN ENO 2 s d Var_D0[6]
MOVP EN ENO 1 s d Var_D0[7]
$MOVP EN ENO "ABCD" s d Var_D10
MOVP EN ENO 4 s d Var_D20[0]
MOVP EN ENO 0 s d Var_D20[1]
MOVP EN ENO 1 s d Var_D20[2]
MOVP EN ENO 2 s d Var_D20[3]
MOVP EN ENO 3 s d Var_D20[4]
SP_FWRITE EN ENO 2 s0 d0 Var_D0 Var_D10 s1 d1 Var_M0 Var_D20[0] s2
2
Var_M0[0]
Var_M0[1]
SET EN ENO d Y10
RST EN ENO d Y11
Var_M0[1]
SET EN ENO d Y11
RST EN ENO d Y10

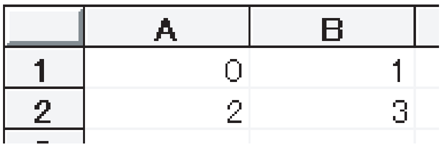
```
[ST]
IF X10 THEN
     MOVP(TRUE,H0,Var_D0[0]);
     MOVP(TRUE,2,Var_D0[6]);
     MOVP(TRUE,1,Var_D0[7]);
     Var_D10:="ABCD";
     MOVP(TRUE,4,Var_D20[0]);
     MOVP(TRUE,0,Var_D20[1]);
     MOVP(TRUE,1,Var_D20[2]);
     MOVP(TRUE,2,Var_D20[3]);
     MOVP(TRUE,3,Var_D20[4]);
     SP_FWRITE(TRUE,2,Var_D10,Var_D20[0],Var_D0,Var_M0);
END_IF;
IF Var_M0[0] THEN
     IF(Var_M0[1]=FALSE)THEN
        SET(TRUE, Y10);
        RST(TRUE, Y11);
     ELSE
        SET(TRUE, Y11);
        RST(TRUE, Y10);
     END_IF;
END_IF;
```

- The written file is displayed as follows.

0	,	1	,	CR	LF
2	,	3	,	CR	LF

Content of the file to be written

Data to be read to the EXCEL file



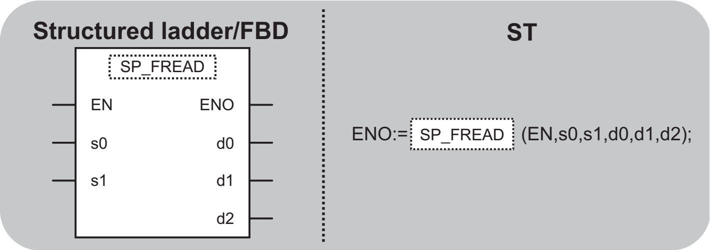
	A	B
1	0	1
2	2	3

Reading data from specified file

SP_FREAD

~~Basic~~ High performance Process Redundant Ver. Universal Ver. LCPU

- Universal model QCPU: Not supported for Q00UJCPU, Q00UCPU, and Q01UCPU
- Built-in Ethernet port LCPU: Supported
- LCPU: Not supported for L02SCPU and L02SCPU-P



The following instruction can go in the dotted squares.

SP_FREAD

■Executing condition

Instruction	Executing condition
SP_FREAD	↑

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s0	Drive specification	ANY16
	s1	File name	String
Output argument	ENO	Execution result	Bit
	d0	Control data	Array of ANY16 (0..7)
	d1	Start number of the device that stores read data	ANY16
	d2	Array to be turned ON at the process completion	Array of bit (0..1)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant		Others
	Bit	Word		Bit	Word			K, H	$	
(s0)	○	○		—				○	—	—
(s1)	—	○		—				—	○	—
(d0)	—	○		—				—	—	—
(d1)	—	△*1		—				—	—	—
(d2)	△*1	△*1		—				—	—	—

*1 Local devices and file registers per program cannot be used as setting data.

Setting data

<table>
<tr><th>Setting data</th><th colspan="3">Description</th><th>Setting range</th><th>Setting side</th><th>Data type</th></tr>
<tr><td>(s0)</td><td colspan="3">Drive specification</td><td>2</td><td>User</td><td rowspan="4">BIN 16 bits</td></tr>
<tr><td rowspan="3">(s1)</td><td colspan="5">Start number of the device that stores a file name. A file name is expressed as follows.</td></tr>
<tr><td>Device</td><td>Item</td><td>Content/Setting data</td><td>Setting range</td><td>Setting side</td></tr>
<tr><td>(s1) to (s1)+□</td><td>File name character string</td><td>Specify the character string of a file name.
• When omitting an extension, also omit the "." (Period).
• Limit the file name within 8 characters + period + 3 characters.
• When 9 or more characters are used, the extension is ignored regardless of its presence, and "BIN" or "CSV" is automatically assigned as an extension.</td><td>Character string</td><td>User</td></tr>
</table>

Setting data	Description			Setting range	Setting side	Data type
(d0)	Start number of the device that stores control data The following control data are required.					BIN 16 bits
	Device	Item	Content/Setting data	Setting range	Setting side	
	(d0)[0]	Execution/ completion type	Specify the execution type. 0000H: Read binary data 0100H: Read data after CSV format conversion	0000H, 0100H	User	
	(d0)[1]	(Not used)	Used by system	—	System	
	(d0)[2]	Number of request read data	Specify the number of data to request reading. (Unit: word) This data should be set in unit of word even when byte is specified for (d0)[7].	1 to 480 1 to 32767[*2]	User	
	(d0)[3]	(Not used)	—	—	—	
	(d0)[4] (d0)[5]	File position	When binary data read is specified for (d0)[0] • Set the file position. 00000000H: Start of the file 00000001H to FFFFFFFEH: From the specified position The unit for the value is determined by the data type specification. FFFFFFFFH: Setting disabled When CSV format read is specified for (d0)[0] • For the High Performance model QCPU of which the first 5 digits of the serial number are '01111' or lower, always set the beginning (0H) of the file. • For the High Performance model QCPU/ Process CPU/Redundant CPU/Universal model QCPU/LCPU of which the first 5 digits of the serial number are '01112' or later, set the file position (Row). 00000000H: Read from the beginning of the file. 00000001H to FFFFFFFEH: Read from the specified row. FFFFFFFFH: Read continues from the previous read position.	00000000H to FFFFFFFFH	User	
	(d0)[6]	Number of columns specification	When binary read is specified for (d0)[0], always set 0. When CSV format read is specified for (d0)[0], set the number of columns from where data will be read. 0: No columns. Regarded as one row. Other than 0: Regarded as the specified number of columns.	0H to FFFFH (0 to 65535)	User	
	(d0)[7]	Data type specification	0: Word 1: Byte	0,1	User	
(d1)	Start number of the device for storing the read data					
	Device	Item	Content/Setting data	Setting range	Setting side	
	(d1)	Reading result (number of read data)	Contains the number of actually read data against the data specified for (d0)[2]. The unit for the value is determined by the data type specification.	—	System	
	(d1)[1] to (d1)+□	Reading data	Read data	—	System	
(d2)	Bit device that turned ON at the completion of the processing ((d2)[1] is also turned ON at error completion.)					Bit
	Device	Item	Content/Setting Data	Setting range	Setting side	
	(d2)[0]	Completion signal	Indicates the completion of the processing. ON: Completed OFF: Not completed	—	System	
	(d2)[1]	Error completion signal	Indicates whether the processing is normally completed or abnormally completed. ON: Error completion OFF: Normal completion	—		

*2 Indicates the range applicable for Universal model QCPU and LCPU.

Precautions

- For QCPU (Q mode), only the ATA card drive (2) can be set for (s0) (drive specification). Note that when the Flash card is installed, the SP_FREAD instruction cannot be used to perform reading. The SRAM card, standard RAM or standard ROM drive cannot be set. For High-speed Universal model QCPU, Universal model Process CPU, and LCPU, only the SD memory card drive (2) can be set for (s0) (drive specification).
- For CSV setting, the data written are decimal values.

Ex.

Character "A" (41H) → 65 is written.

Handling range: -32768 to 32767

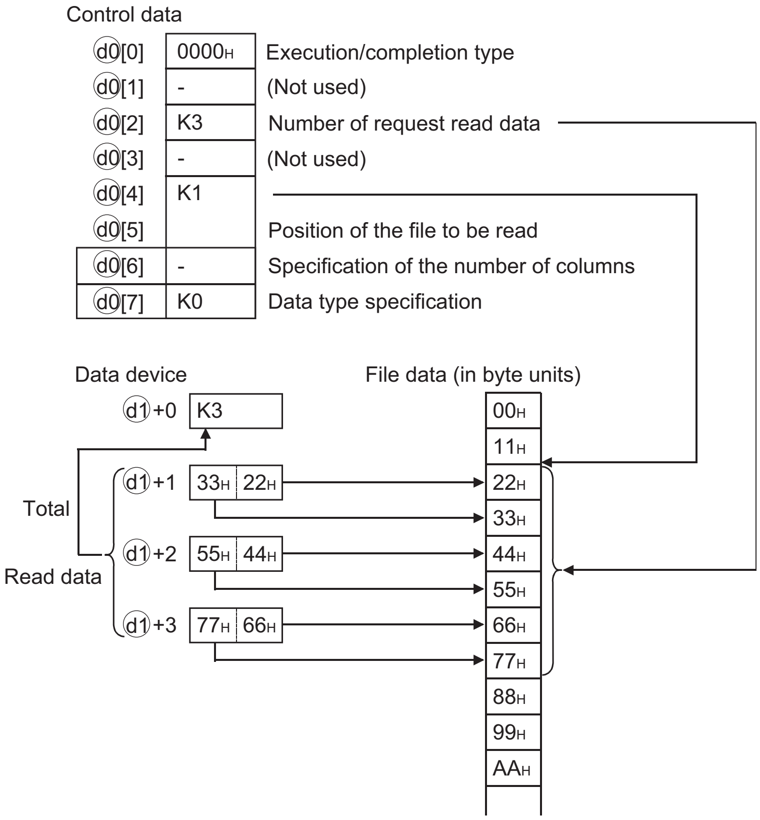
- For binary read, the setting range of the file position with word specification is 00000000H to 7FFFFFFFH.
- For High-speed Universal model QCPU, Universal model Process CPU, and LCPU, this instruction will not be executed while SM606 (SD memory card forced disable instruction) is on. If executed, no processing is performed.

Processing details

- Data are read from the specified file. Set the execution/completion type in the control data to specify whether to read binary data without any conversion or to convert binary data to CSV format data before reading it. (The reading target is the ATA card only for QCPU (Q mode), and the SD memory card only for High-speed Universal model QCPU, Universal model Process CPU, and LCPU.)
- The execution completion bit device (d2) is automatically turned on at the END processing after the completion of the instruction is detected. The bit device is turned off at the END processing in the next scan.
  - When this instruction is completed abnormally, the error completion device (d2)[1] is turned ON/OFF in synchronization with the execution completion (d2)[0] device. Use this device as the error completion flag for this instruction.
  - SM721 is turned ON during the execution of the instruction. This instruction cannot be executed while SM721 is ON. (If an attempt is made, no processing is performed.)
  - When an error is detected at the execution of the instruction (before SM721 is turned ON), the processing complete device (d1)[0], the error completion device (d)[1], and SM721 are not turned ON.
- Be sure to use units of word to specify the number of request read data (d0)[2], file position ((d0)[4], (d0)[5]), and read data device size (d1).

■When reading binary data

- If the extension of the target file is omitted, ".BIN" is used as an extension.
- When the specified file does not exist, an error occurs.
- If the position specified is greater than the existing file size:
  - The High Performance model QCPU of which the first 5 digits of the serial number are '01111' or lower results in an error.
  - The High Performance model QCPU, Universal model QCPU, Process CPU, Redundant CPU, and LCPU of which the first 5 digits of the serial number are "01112" or later performs reading at point 0 and is completed normally.
- The following shows how the data is read in binary data reading operation.



■When reading data after CSV format conversion

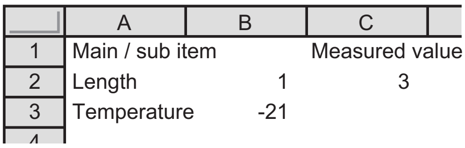
- The elements in CSV format file (cells for EXCEL) are read by each row. The numeric values and character strings are converted to binary data and stored to the device.
- If the extension is omitted, ".CSV" is used as an extension.
- When the specified file does not exist, an error occurs.
- The elements specified by the number of request read data (d0)[2] are read from the beginning of the file. When the last datum of the file is reached before the specified number of data are read:
  - The High Performance model QCPU of which the first 5 digits of the serial number are '01111' or lower results in an error.
  - The High Performance model QCPU, Process CPU, Redundant CPU, Universal model QCPU, and LCPU of which the first 5 digits of the serial number are "01112" or later reads the data up to the point where the reading is possible.

- When the specified number of columns is 0, the data are read by ignoring the rows in CSV format file.

Ex.

When data are read after CSV format conversion and the specified number of columns is 0.

Data created by EXCEL

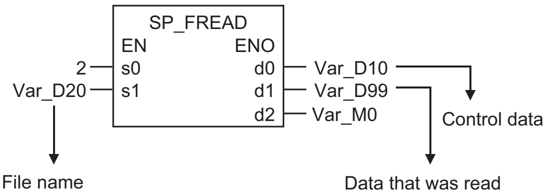


	A	B	C
1	Main / sub item		Measured value
2	Length	1	3
3	Temperature	-21	
4			

Data saved in the CSV format

Main / sub item	,	,	Measured value	CR	LF	
Length	,	1	,	3	CR	LF
Temperature	,	-21	,	CR	LF	

Data to be read into devices



Control data

Var_D10	H0100	Execution/completion type
	-	Not used
	K9	Number of request read data
	-	Not used
	K0	File position
	K0	
	K0	Specification of the number of columns
	K0	Data type specification
Var_D20	H4241	File name
	H4443	"ABCDE"
	H0045	

Data that was read

Read data				
Stores the number of read data	Var_D99	K9	·····	Reading result (number of read data)
Main/sub item		K0	·····	Conversion data (0) is stored since "Main/sub item" is nonnumeric data.
Data between , and ,		K0	·····	Conversion data (0) is stored since " " is nonnumeric data.
Measured value		K0	·····	Conversion data (0) is stored since "Measured value" is nonnumeric data.
Length		K0	·····	Conversion data (0) is stored since "Length" is nonnumeric data.
1		K1	·····	Since "1" is a numeric value, it is converted to a binary value.
3		K3	·····	Since "3" is a numeric value, it is converted to a binary value.
Temperature		K0	·····	Conversion data (0) is stored since "Temperature" is nonnumeric data.
K-21		K-21	·····	Since "-21" is a numeric value, it is converted to a binary value.
Data between , and CR		K0	·····	Conversion data (0) is stored since " " is nonnumeric data.

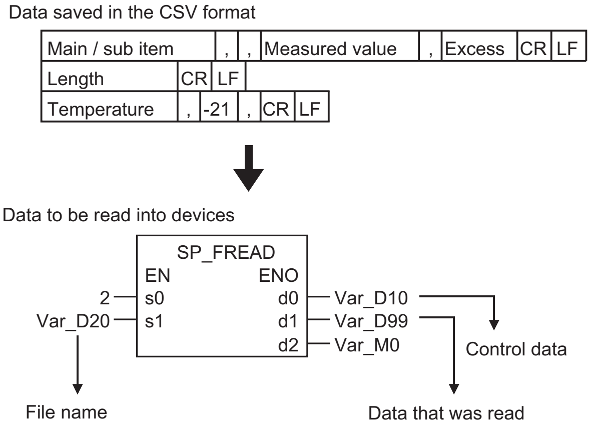
- If the number of columns varies in each row, the data are also read by ignoring the rows.

Point

Such file cannot be created using EXCEL. This happens when CSV file is modified by a user.

Ex.

When the number of columns varies in each row when the data are read.



Data saved in the CSV format
Main / sub item | , | , | Measured value | , | Excess | CR | LF
Length | CR | LF
Temperature | , | -21 | , | CR | LF
Data to be read into devices
SP_FREAD
EN
ENO
2
s0
d0
Var_D10
Var_D20
s1
d1
Var_D99
d2
Var_M0
Control data
File name
Data that was read

Control data

Var_D10	H0100	Execution/completion type
	-	Not used
	K7	Number of request read data
	-	Not used
	K0	File position
	K0	
	K0	Specification of the number of columns
	K0	Data type specification
Var_D20	H4241	File name
	H4443	"ABCD"
	H0000	

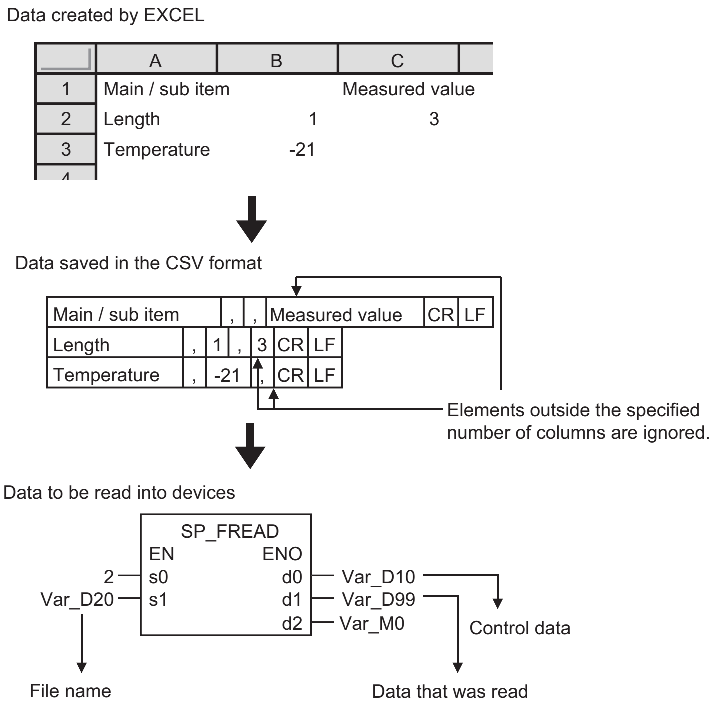
Data that was read

Read data			
Stores the number of read data	Var_D99	K7	····· Reading result (number of read data)
Main/sub item		K0	····· Conversion data (0) is stored since "Main/sub item" is nonnumeric data.
Data between , and ,		K0	····· Conversion data (0) is stored since " " is nonnumeric data.
Measured value		K0	····· Conversion data (0) is stored since "Measured value" is nonnumeric data.
Excess		K0	····· Conversion data (0) is stored since "Excess" is nonnumeric data.
Length		K0	····· Conversion data (0) is stored since "Length" is nonnumeric data.
Temperature		K0	····· Conversion data (0) is stored since "Temperature" is nonnumeric data.
K-21		K-21	····· Since "-21" is a numeric value, it is converted to a binary value.

- When data are read after CSV format conversion and the specified number of columns is other than 0, the data are read as the table with specified number of columns in CSV format file. The elements outside of the specified columns are ignored.

Ex.

When data are read after CSV format conversion and the specified number of columns is other than 0.


Data created by EXCEL
Data saved in the CSV format
Elements outside the specified number of columns are ignored.
Data to be read into devices
SP_FREAD
EN
ENO
2
s0
d0
Var_D10
Var_D20
s1
d1
Var_D99
d2
Var_M0
Control data
File name
Data that was read

	A	B	C
1	Main / sub item		Measured value
2	Length	1	3
3	Temperature	-21	
4			

Main / sub item	,	,	Measured value	CR	LF	
Length	,	1	,	3	CR	LF
Temperature	,	-21	,	CR	LF	

Control data

Var_D10	H0100	Execution/completion type
	-	Not used
	K6	Number of request read data
	-	Not used
	K0	File position
	K0	
	K2	Specification of the number of columns
	K0	Data type specification
Var_D20	H4241	File name
	H4443	"ABCD"
	H0000	

Data that was read

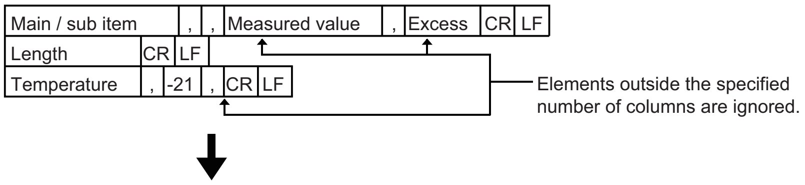
Stores the number of read data	Var_D99	K6	····· Reading result (number of read data)
Read data: Main/sub item		K0	····· Conversion data (0) is stored since "Main/sub item" is nonnumeric data.
Data between , and ,		K0	····· Conversion data (0) is stored since " " is nonnumeric data.
Length		K0	····· Conversion data (0) is stored since "Length" is nonnumeric data.
1		K1	····· Since "1" is a numeric value, it is converted to a binary value.
Temperature		K0	····· Conversion data (0) is stored since "Temperature" is nonnumeric data.
K-21		K-21	····· Since "-21" is a numeric value, it is converted to a binary value.

- If the number of columns varies in each row, the elements outside of the specified columns are ignored and "0" is added to the places where elements do not exist.

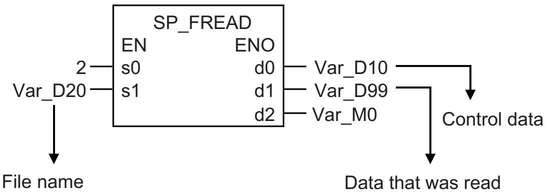
Ex.

When the number of columns varies in each row when the data are read.

Data saved in the CSV format



Data to be read into devices



Control data

Var_D10	H0100	Execution/completion type
	-	Not used
	K6	Number of request read data
	-	Not used
	K0	File position
	K0	
	K2	Specification of the number of columns
	K0	Data type specification
Var_D20	H4241	File name
	H4443	"ABCD"
	H0000	

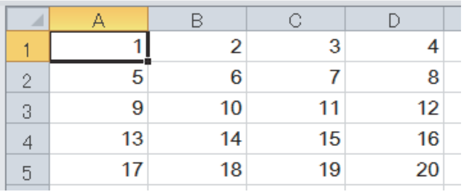
Data that was read

	Stores the number of read data	Var_D99	K6	····· Reading raesult (number of read data)
Read data	Main/sub item		K0	····· Conversion data (0) is stored since "Main/sub item" is nonnumeric data.
	Data between , and ,		K0	····· Conversion data (0) is stored since " " is nonnumeric data.
	Length		K0	····· Conversion data (0) is stored since "Length" is nonnumeric data.
	No data		K0	····· Since no element exists here, conversion data (0) is added.
	Temperature		K0	····· Conversion data (0) is stored since "Temperature" is nonnumeric data.
	K-21		K-21	····· Since "-21" is a numeric value, it is converted to a binary value.

- With the High Performance model QCPU, Process CPU/Redundant CPU, Universal model QCPU, and LCPU whose first 5 digits of the serial number are "01112" or later, they divides read operation into multiple times.

Ex.

When a CSV file as follows is read by multiple read operations



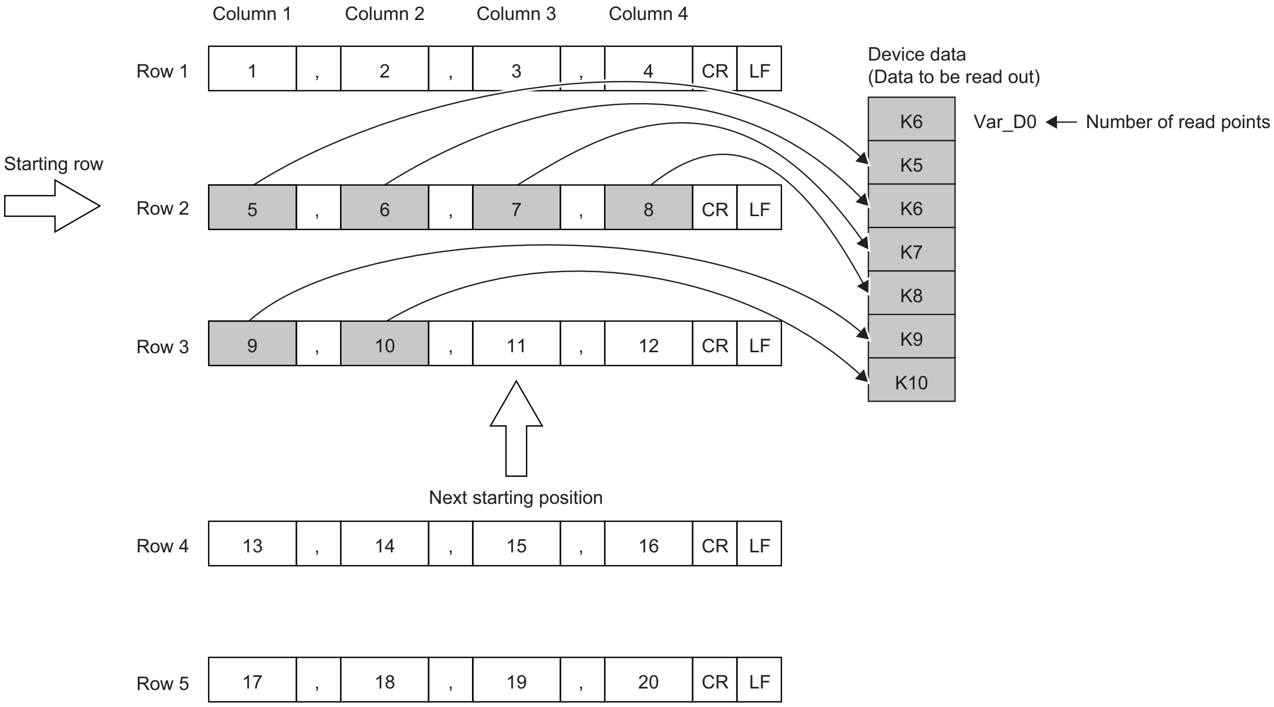
	A	B	C	D
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12
4	13	14	15	16
5	17	18	19	20

[First read]

Reading VAR_D0 data with the following setting

Control data	Description	Set value	
(d0)[0]	Execution/completion type	0100H	CSV format
(d0)[2]	Number of request read data	6	6 words
(d0)[4] (d0)[5]	File position	2H	Reading data from the second row
(d0)[6]	Number of columns specification	4H	4 columns
(d0)[7]	Data type specification	0	Word

The following figure shows the operation when the SP_FREAD instruction is executed.

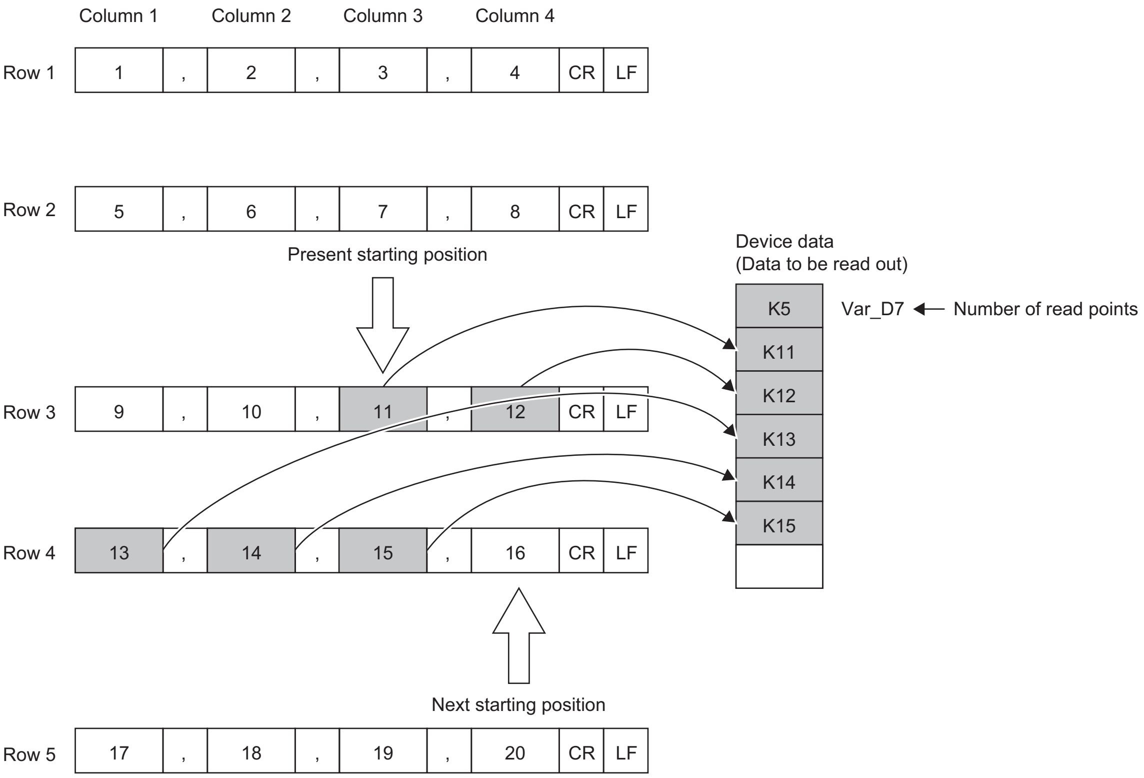


[Second read]

Reading VAR_D7 data with the following setting

Control data	Description	Set value	
(d0)[0]	Execution/completion type	0100H	CSV format
(d0)[2]	Number of request read data	5	5 words
(d0)[4] (d0)[5]	File position	FFFFFFFFH	Continuing read from the end of the previous read position
(d0)[6]	Number of columns specification	4H	4 columns
(d0)[7]	Data type specification	0	Word

The following figure shows the operation when the SP_FREAD instruction is executed.



- When read is performed in the continuation mode, the previous addition cannot be made normally if the 'execution type', 'column specification' and 'data type specification' settings differ from those at the previous time.
- The previous addition cannot be made normally if the SP_FREAD instruction or SP_FWRITE instruction with another setting is executed while data are being read continuously in the continuation mode.

- When data are read after CSV format conversion, the numeric values that are out of range or the elements other than numeric values in the CSV format file to be read are converted to 0H.
- When data are read after CSV format conversion, numeric values are read and converted as follows.

n CSV format		-32768 to -1	0 to 32767	32768 to 65535
Word device	Without a sign	32768 to 65535	0 to 32767	32768 to 65535
	With a sign	-32768 to -1	0 to 32767	-32768 to -1

- Do not execute this instruction in an interrupt program. (Otherwise, a malfunction may result.)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

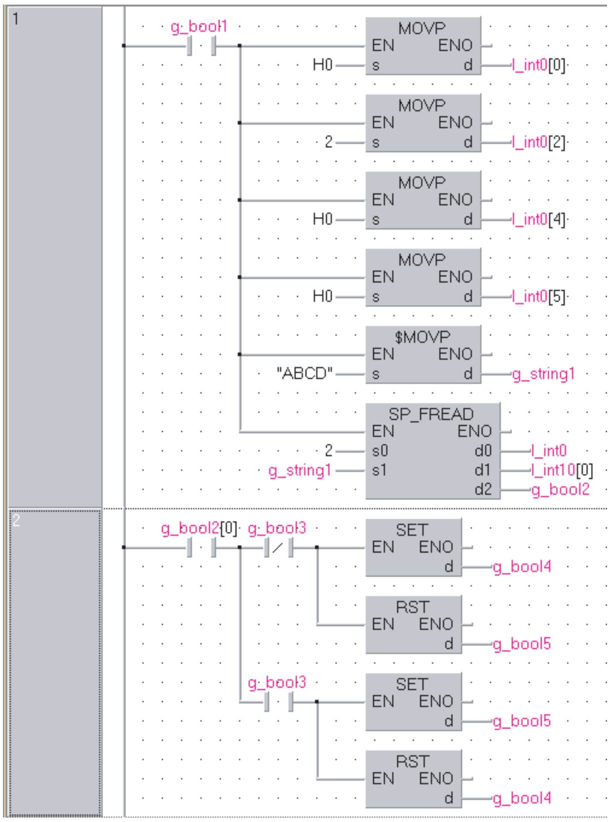
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2410	File name specified for file name character string (s1) and the following devices does not exist in the specified drive.	—	○	○	○	○	○
4004	When the device which cannot be specified is specified	—	○	○	○	○	○
4100	Values specified for control data (d0) and the following devices are out of the setting range. (Excluding (d0)+2)	—	○	○	○	○	○
	Drive specified for drive specification device (s0) contains the medium other than the ATA card. Access error occurred in the ATA card.	—	○	○	○	○	—
	When binary data are read, the number of data in the file is less than the size specified for the number of request read data (d0)+2	—	○	—	—	—	—
	Drive specified for drive specification device (s0) contains the medium other than the SD memory card. Access error occurred in the SD memory card.	—	—	—	—	○	○
4101	Value specified for number of data blocks to be read (d0)+2 is out of the setting range. Size of read data exceeds the size of reading device.	—	○	○	○	○	○
	The device specified for (d0) or (d1) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, 4 bytes of binary data are read from the beginning of the file "ABCD.BIN" in the memory card inserted to drive 2 when g_bool1 turns ON.

- Assume that 8 points from l_int0 are reserved for the control data devices.
- Assume that 100 bytes from l_int10[0] are reserved for the reading devices.

[Structured ladder/FBD]



Sets the execution/completion type.

Sets the number of request read data.

Sets the file position.

Sets the file name.

Normal completion display

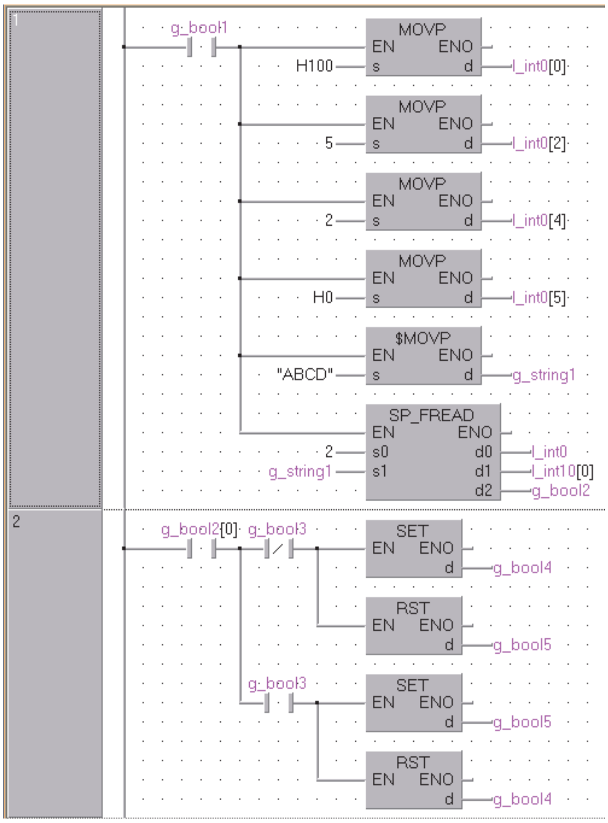
Error completion display

[ST]

```
IF g_bool1 THEN
    MOVP(TRUE, H0, l_int0[0]);
    MOVP(TRUE, 2, l_int0[2]);
    MOVP(TRUE, H0, l_int0[4]);
    MOVP(TRUE, H0, l_int0[5]);
    g_string1: ="ABCD";
    SP_FREAD(TRUE,2, g_string1, l_int0, l_int10[0], g_bool2);
END_IF;
IF g_bool2 THEN
    IF(g_bool3=FALSE)THEN
      SET(TRUE, g_bool4);
      RST(TRUE,g_bool5);
    ELSE
      SET(TRUE, g_bool5);
      RST(TRUE, g_bool4);
    END_IF;
END_IF;
```

- In the following program, the file "ABCD.CSV" is read to the PC card inserted to slot 0 as two-column table data in CSV format when g_bool1 turns ON.
- Assume that 8 points from l_int0 are reserved for the control data devices.
- Assume that 100 bytes from l_int10[0] are reserved for the reading devices.
- Assume that the target CSV format file contains numeric values only.

[Structured ladder/FBD]



Sets the execution/completion type.

Sets the number of request read data.

Sets the designation of number of columns.

Sets the file name.

Normal completion display

Error completion display

[ST]

```
IF g_bool1 THEN
    MOVP(TRUE, H100, l_int0[0]);
    MOVP(TRUE, 5, l_int0[2]);
    MOVP(TRUE, 2, l_int0[4]);
    MOVP(TRUE, H0, l_int0[5]);
    g_string1:="ABCD";
    SP_FREAD(TRUE,2, g_string1, l_int0, l_int10[0], g_bool2);
END_IF;
IF g_bool2 THEN
    IF(g_bool3=FALSE)THEN
      SET(TRUE, g_bool4);
      RST(TRUE,g_bool5);
    ELSE
      SET(TRUE, g_bool5);
      RST(TRUE, g_bool4);
    END_IF;
END_IF;
```

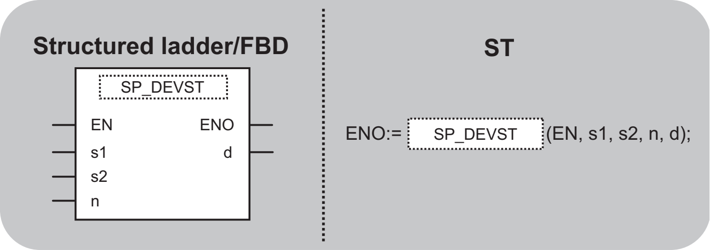
Precautions

When using the SP_FREAD instruction, do not branch a line from (d1). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

Writing data to standard ROM

SP_DEVST

✗ Basic ✗ High performance ✗ Process ✗ Redundant Universal LCPU



The following instruction can go in the dotted squares.

SP_DEVST

■Executing condition

Instruction	Executing condition
SP_DEVST	↑

■Argument

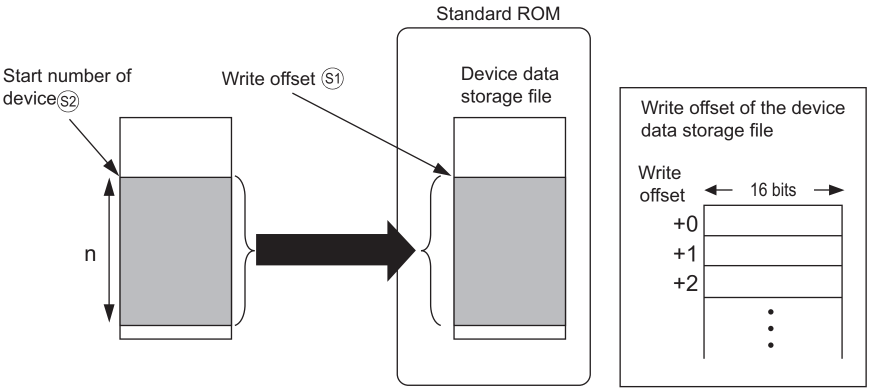
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Write offset of the device data storage file (Specified in unit of 16 bits per one point)	ANY32
	s2	Start number of the device to be written to the standard ROM	ANY16
	n	Number of device points to be written	ANY16
Output argument	ENO	Execution result	Bit
	d	Array to be turned ON at the execution completion • d[0]: Completion • d[1]: Error completion	Array of bit (0..1)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	—	○	○	—				○	—
(s2)	—	○	○	—				—	—
n	—	○	○	—				○	—
(d)	△[*1]	—	△[*1]	—				—	—

*1 Local devices and file registers per program cannot be used as setting data.

Processing details

- Writes n points of device data specified for (s2) to the write offset specified for (s1) of the device data storage file in the standard ROM. (s1) is the offset from the start of device data storage file and specified by word offset (in unit that increments by 1 for each 16 bits).



- Since the device data write position completion (d)[0] in the standard ROM automatically turns on at execution of the END processing, which detects the completion of this instruction, and turns off at the END processing of next scan, it is used as an execution completion flag of this instruction.
- When this instruction is completed in error, the error completion (d)[1] turns ON/OFF at the same timing with the completion (d)[0]. This is used as an error completion flag of this instruction.
- SM721 turns ON during execution of this instruction. When SM721 has already turned ON, this instruction cannot be executed. (If executed, no processing is performed.)
- When an error is detected at execution of this instruction, the completion (d)[0], error completion (d)[1], and SM721 do not turn ON.

Operation error

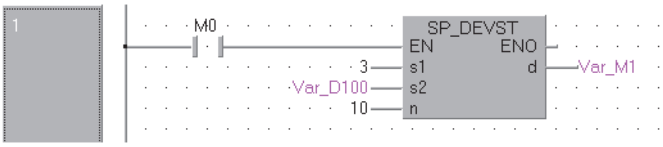
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
1610	The write frequency into standard ROM exceeds 100,000 times during execution of SP_DEVST instruction.	—	—	—	—	○	○
2410	The device data storage file is not set in the PLC file setting of the PLC parameter.	—	—	—	—	○	○
4100	The write offset specified for (s1) is out of the device data storage file range. The n2 points from the write offset specified for (s1) is out of the device data storage file range. The value specified for n is 0 or a negative number.	—	—	—	—	○	○
4101	The range of n points from the device specified for (s2) exceeds the corresponding device range. The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○
4113	The value out of the range is specified in the SD695 during execution of SP_DEVST instruction. The write frequency into standard ROM at the current day exceeds the value specified by the SD695 during execution of SP_DEVST instruction.	—	—	—	—	○	○

Program example

- In the following program, 10 points of data from Var_D100 are written to the device data storage file in the standard ROM when M0 turns ON.

[Structured ladder/FBD]



[ST]

```
SP_DEVST(M0,3,Var_D100,10,Var_M1);
```

Precautions

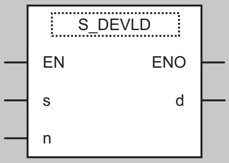
- The value written to the standard ROM is the value at execution of this instruction.
- The standard ROM data write count (SD687, SD688) increases by the execution of the SP_DEVST instruction. If the standard ROM data write count exceeds 100,000, a FLASH ROM ERROR occurs. (Error code: 1610)
- In order to prevent the increase of the standard ROM data write count by the improper instruction execution, set an instruction execution count for the standard ROM data write (SD695) to limit the number of times that data are written to the standard ROM in a day. If the set data write count (default: 36) is exceeded, an OPERATION ERROR occurs (Error code: 4113).

Reading data from standard ROM

S(P)_DEVLD

~~Basic~~ ~~High performance~~ ~~Process~~ ~~Redundant~~ Universal LCPU

Structured ladder/FBD



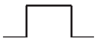
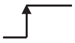
ST

ENO:= S_DEVLD (EN, s, n, d);

Any of the following instruction can go in the dotted squares.

S_DEVLD, SP_DEVLD

■Executing condition

Instruction	Executing condition
S_DEVLD	
SP_DEVLD	

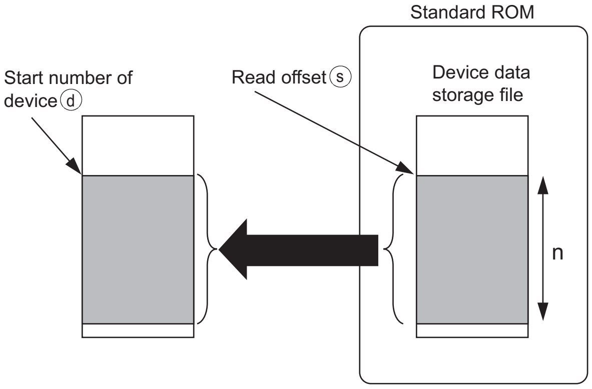
■Argument

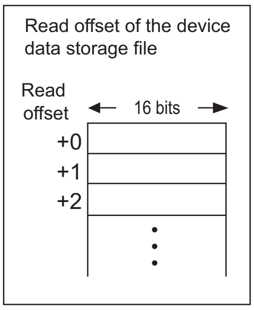
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Read offset of the device data storage file (Specified in unit of 16 bits per point)	ANY32
	n	The number of read points	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device to be read from the standard ROM	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
n	—	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Reads n points of device data from read offset specified for (s) of the device data storage file in the standard ROM, and stores the data to the device specified for (d).
- (s) is the offset from the start of device data storage file and specified by word offset (in unit that increments by 1 for each 16 bits).





Operation error

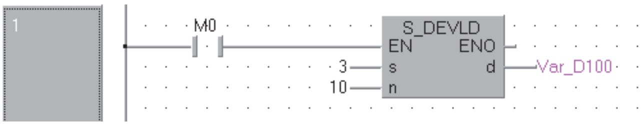
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2410	The device data storage file is not set in the PLC file setting of the PLC parameter.	—	—	—	—	○	○
4100	The address specified for (s) is out of the standard ROM range. The n2 points from the address specified for (s) is out of the standard ROM range. The value specified for n is 0 or a negative number.	—	—	—	—	○	○
4101	The range of n points from the device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

Program example

- In the following program, 10 points of device data are read from the device data storage file in the standard ROM to from Var_D100 when M0 turns ON.

[Structured ladder/FBD]



[ST]

S_DEVLD(M0,3,10,Var_D100);

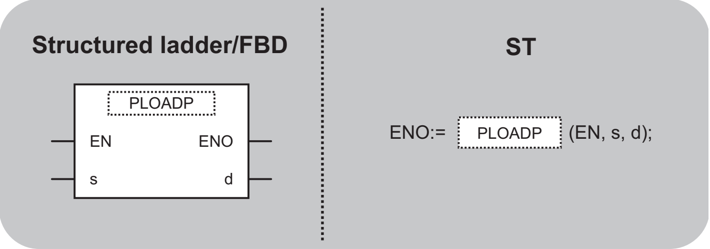
Precautions

When using the SP_DEVLD instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

Program load from memory card

PLOADP

~~Basic~~ High performance Process ~~Redundant~~ ~~Universal~~ ~~LCPU~~



The following instruction can go in the dotted squares.

PLOADP

■Executing condition

Instruction	Executing condition
PLOADP	↑

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Character string data of the driver number file name that stores programs to be loaded, or start number of the device that stores character string data[*1]	String
Output argument	ENO	Execution result	Bit
	d	Device to be turned ON for one scan at the instruction completion	Bit

*1 Specified by "<Drive number>:<File Name>". Example 1: MAIN

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	△[*2]	—		—				—	—

*2 Local devices cannot be used.

Processing details

- The program stored in the memory card or standard ROM is transferred to the program memory (drive 0). If the transferred program is not registered to the program setting of the PLC parameter dialog box, its program setting in the CPU module is set to the standby type. At this time, the program setting of the PLC parameter dialog box does not change. (To transfer a program with the PLOADP instruction, consecutive free spaces are required in the program memory.)
- The program added using the PLOADP instruction is assigned by the lowest number among the unused program numbers. (To assign a program number manually, store the program number to be assigned to SD720.)

The following example shows when "MAIN6" is added by the PLOADP instruction.

- When the program numbers have been set consecutively, the new program is added at the end of the preset program numbers.

When programs number 1 to 5 have been set, a new program is added as program number 6.

Program No.	Program name
1	MAIN1
2	MAIN2
3	MAIN3
4	MAIN4
5	MAIN5

Adds"MAIN6"by the PLOADP instruction.

Program No.	Program name
1	MAIN1
2	MAIN2
3	MAIN3
4	MAIN4
5	MAIN5
6	MAIN6

←Added at the end.

- When there are multiple open program numbers, the program specified by the PLOADP instruction is added to the lowest number among them to be added.

(The empty program numbers are made when programs are deleted by the PUNLOADP instruction.)

When programs number 2 and 4 are empty, a new program is added as program number 2.

Program No.	Program name
1	MAIN1
2	Empty
3	MAIN3
4	Empty
5	MAIN5

Adds"MAIN6"by the PLOADP instruction.

Program No.	Program name
1	MAIN1
2	MAIN6
3	MAIN3
4	Empty
5	MAIN5

←Added to the smallest program number which is empty.

- Drive numbers 1, 2, and 4 can be specified. (Drive 3 cannot be specified.)
  - Drive 1: Memory card (RAM)
  - Drive 2: Memory card (ROM)
  - Drive 4: Standard ROM
- An extension (.QPG) does not need to be specified for the file name.
- The bit device specified for (d) is turned ON during the END processing of the scan where this instruction is completed. The bit device is turned OFF at the next END processing.

■The PLC file settings of the loaded program are set as follows.

- File usage for each program

All the usage of file registers, device initial values, comments, and local devices of the program transferred by this instruction are set based on PLC file settings. However, an error will be returned if both of the conditions below are met when the program is transferred using this instruction.

- Setting is made so that local devices are used in the PLC file setting.
- The number of programs in the program memory exceeds the number of programs set in the parameters.

To use local devices in the program transferred by this instruction, register a dummy program file in the parameter, delete the dummy file with the PUNLOADP instruction, and then load the program with the PLOADP instruction.

- I/O refresh setting

No settings are applied to both input and output data of the I/O refresh setting for the program transferred by this instruction.

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2401	The file size of the local devices cannot be reserved.	—	○	○	—	—	—
2410	File name does not exist at the drive number specified for (s). The program file which has the same name as the program file to be loaded already exists.	—	○	○	—	—	—
2413	There is not enough memory to load the specified program in drive 0.	—	○	○	—	—	—
4100	The drive number specified for (s) is invalid.	—	○	○	—	—	—
4101	The number of files indicated in the table below are registered in the program memory. The program number stored in SD720 is already used, or larger than the largest program number shown in the table below.	—	○	○	—	—	—

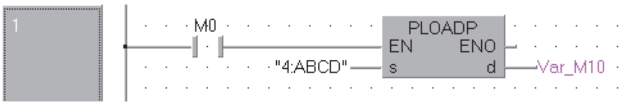
The following table lists the number of files and the largest program number of each CPU module.

CPU model name	Program memory (number of files)	Largest program No.
Q02(H)CPU, Q02PHCPU	28	28
Q06HCPU, Q06PHCPU	60	60
Q12HCPU	124	124
Q25HCPU	124	124
Q12PHCPU	124	124
Q25PHCPU	124	124

Program example

- In the following program, "ABCD.QPG" stored in the drive 4 is transferred to drive 0 and set the program in standby type when M0 turns ON.

[Structured ladder/FBD]



[ST]
PLOADP(Var_M0,"4:ABCD",Var_M10);

Precautions

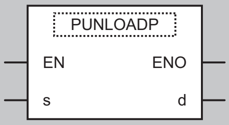
- The PLOADP, PUNLOADP and PSWAPP instructions cannot be executed simultaneously. If two or more of the above instructions are executed simultaneously, the instruction executed later will not be executed. When using the above instructions, provide interlocks manually to avoid simultaneous execution.
- Do not execute this instruction in an interrupt program. (Otherwise, a malfunction may result.)
- To execute the program that was transferred to the program memory with the PLOADP instruction, execute the scan execution type with the PSCAN instruction (☞ Page 710 Registering program as scan execution type).
- The "PLOADP instruction" and "Write during RUN" processing cannot be executed simultaneously.
  - When a write during RUN request is given during processing of the PLOADP instruction, write during RUN is delayed. Write during RUN is started after the processing of the PLOADP instruction is completed.
  - When the PLOADP instruction is executed during write during RUN, the processing of the PLOADP instruction is delayed. The processing of the PLOADP instruction is started after completion of write during RUN.
- Do not execute "Read from PLC" or "Verify with PLC" and the instruction simultaneously. If the instruction is executed, "Read from PLC" or "Verify with PLC" is not complete normally because the program file status stored in the program memory is changed. If executed, execute "Read from PLC" or "Verify with PLC" again after the instruction completion.

Program unload from program memory

PUNLOADP

~~Basic~~ High performance Process ~~Redundant~~ ~~Universal~~ ~~LCPU~~

Structured ladder/FBD



ST

```
ENO:= PUNLOADP (EN, s, d);
```

The following instruction can go in the dotted squares.

PUNLOADP

■Executing condition

Instruction	Executing condition
PUNLOADP	↑

■Argument

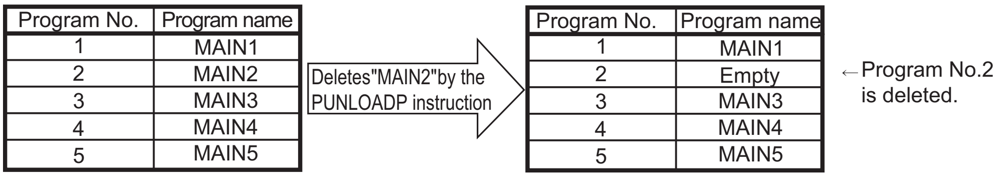
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Character string data of the program file name to be unloaded, or start number of the device that stores character string data	String
Output argument	ENO	Execution result	Bit
	d	Device to be turned ON for one scan at the instruction completion	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	△[*1]	—		—				—	—

*1 Local devices cannot be used.

Processing details

- The standby type program stored in the program memory (drive 0) is deleted from the program memory. (The program set as the 'scan execution type' with the PSCAN instruction or the program set as the 'low speed execution type' with the PLOW instruction cannot be deleted.)
- The program number deleted by the PUNLOADP instruction becomes 'Empty'. When programs number 1 to 5 have been set in the program setting of the PLC parameter dialog box, deleting program number 2 with this instruction makes program number 2 empty.



Program No.	Program name
1	MAIN1
2	MAIN2
3	MAIN3
4	MAIN4
5	MAIN5

Deletes"MAIN2"by the PUNLOADP instruction

Program No.	Program name
1	MAIN1
2	Empty
3	MAIN3
4	MAIN4
5	MAIN5

←Program No.2 is deleted.

- An extension (.QPG) does not need to be specified for the file name.
- The bit device specified for (d) is turned ON during the END processing of the scan where this instruction is completed. The bit device is turned OFF at the next END processing.
- When the programmable controller CPU is powered ON or the CPU module is reset after execution of the PUNLOADP instruction, the following operation is performed.
  - When boot setting has been set in the PLC parameter dialog box, the program where the boot setting has been set is transferred to the program memory. When the program deleted by the PUNLOADP instruction is not to be executed, delete the corresponding program name from the boot setting and program setting of the PLC parameter dialog box.
  - When boot setting has not been set in the PLC parameter dialog box, "FILE SET ERROR (error code: 2400)" occurs. When the program deleted by the PUNLOADP instruction is not to be executed, delete the corresponding program name from the program setting of the PLC parameter dialog box. When the program deleted by the PUNLOADP instruction is to be executed again, write the corresponding program to the CPU module.

Operation error

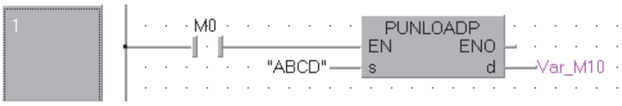
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
2410	The file name specified for (s) does not exist.	—	○	○	—	—	—
4101	The program specified for (s) is not a standby program, or is a program in execution.	—	○	○	—	—	—

Program example

- In the following program, "ABCD.QPG" stored in the drive 0 is deleted from the memory when M0 turns from OFF to ON.

[Structured ladder/FBD]



[ST]
PUNLOADP(M0,"ABCD",Var_M10);

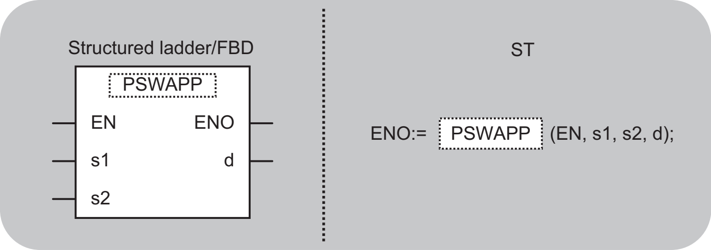
Precautions

- The PLOADP, PUNLOADP and PSWAPP instructions cannot be executed simultaneously. If two or more of the above instructions are executed simultaneously, the instruction executed later will not be executed. When using the above instructions, provide interlocks manually to avoid simultaneous execution.
- Do not execute this instruction in an interrupt program. (Otherwise, a malfunction may result.)
- The program to be deleted from the program memory by this instruction should be set to the "standby execution type" with the PSTOP instruction beforehand. (☞ Page 706 Program standby)
- The "PUNLOADP instruction" and "Write during RUN" processing cannot be executed simultaneously.
  - When a write during RUN request is given during processing of the PUNLOADP instruction, write during RUN is delayed. Write during RUN is started after the processing of the PUNLOADP instruction is completed.
  - When the PUNLOADP instruction is executed during write during RUN, the processing of the PUNLOADP instruction is delayed. The processing of the PUNLOADP instruction is started after completion of write during RUN.
- Do not execute "Read from PLC" or "Verify with PLC" and the instruction simultaneously. If the instruction is executed, "Read from PLC" or "Verify with PLC" is not complete normally because the program file status stored in the program memory is changed. If executed, execute "Read from PLC" or "Verify with PLC" again after the instruction completion.

Load and unload

PSWAPP

Basic High performance Process Redundant Universal LCPU



The following instruction can go in the dotted squares.

PSWAPP

■Executing condition

Instruction	Executing condition
PSWAPP	↑

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Character string data of the program file name to be unloaded, or start number of the device that stores character string data	String
	s2	Character string data of the driver number file name that stores programs to be loaded, or start number of the device that stores character string data[*1]	String
Output argument	ENO	Execution result	Bit
	d	Device to be turned ON for one scan at the instruction completion	Bit

*1 Specified by "<Drive number>:<File Name>". Example 1: MAIN

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				○	—
(s2)	—	○		—				○	—
(d)	△[*2]	—		—				—	—

*2 Local devices cannot be used.

Processing details

- The standby type program stored in the program memory (drive 0) specified for (s1) is deleted from the program memory, and at the same time, the program stored in the memory card or standard ROM specified for (s2) is transferred to the program memory and placed in standby type. (When the program is transferred to the program memory, the program must have a continuous free space.)
- The program set as the 'scan execution type' with the PSCAN instruction or the program set as the 'low speed execution type' with the PLOW instruction cannot be deleted.
- The program to be transferred to the program memory by the PSWAPP instruction will have the program number of the program to be deleted from the program memory. (If there is an empty program number before the program to be deleted from the program memory, the program to be transferred to the program memory will not have the empty program number.)
- When program number 2 is "Empty", the program transferred to the program memory is registered as program number 3 by the program swapping of program number with this instruction.

Program No.	Program name
1	MAIN1
2	Empty
3	MAIN3
4	MAIN4
5	MAIN5

Swaps"MAIN3"with"MAIN6" by the PSWAPP instruction.

Program No.	Program name	
1	MAIN1	
2	Empty	
3	MAIN6	←MAIN6 enters
4	MAIN4	
5	MAIN5	

- Drive numbers 1, 2, and 4 can be specified. (Drive 3 cannot be specified.)
  - Drive 1: Memory card (RAM)
  - Drive 2: Memory card (ROM)
  - Drive 4: Standard ROM
- An extension (.QPG) does not need to be specified for the file name.
- The bit device specified for (d) is turned ON during the END processing of the scan where this instruction is completed. The bit device is turned OFF at the next END processing.
- When the programmable controller CPU is powered ON or the CPU module is reset after execution of the PSWAPP instruction, the following operation is performed.
  - When boot setting has been set in the PLC parameter dialog box, the program where the boot setting has been set is transferred to the program memory. When the program replaced by the PSWAPP instruction is to be executed, change the boot setting and program setting of the PLC parameter dialog box for the corresponding program name.
  - When boot setting has not been set in the PLC parameter dialog box, "FILE SET ERROR (error code: 2400)" occurs. When the program replaced by the PSWAPP instruction is to be executed, change the program setting of the PLC parameter dialog box for the corresponding program name. To execute the program set in the program setting of the PLC parameter dialog box, write the corresponding program to the CPU module again.
- The PLC file settings of the program on which the PSWAPP instruction has been executed are set as follows.
  - File usage for each program

All the usage of file register, device initial value, comment, and local device of the program after the execution of the PSWAPP instruction are set based on PLC file settings.

- I/O refresh setting

No settings are applied to both input and output data of the I/O refresh setting for the program after the PSWAPP instruction has been executed.

Operation error

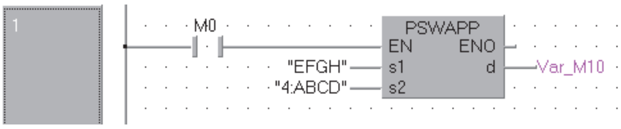
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2410	The drive number or the file name specified for (s1) or (s2) does not exist.	—	○	○	—	—	—
2413	There is not enough memory to load the specified program in drive 0.	—	○	○	—	—	—
4100	The drive number specified for (s1) is invalid.	—	○	○	—	—	—
4101	The program specified for (s1) is not a standby program, or is a program in execution.	—	○	○	—	—	—

Program example

- In the following program, as "EFGH.QPG" stored in drive 0 is deleted from the memory, "ABCD.QPG" stored in drive 4 is transferred to drive 0, and set the program in standby type when M0 turns from OFF to ON.

[Structured ladder/FBD]



[ST]

```
PSWAPP(M0,"EFGH","4:ABCD",Var_M10);
```

Precautions

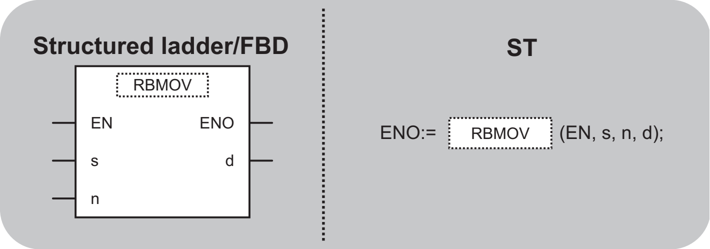
- The PLOADP, PUNLOADP and PSWAPP instructions cannot be executed simultaneously. When using the above instructions, provide interlocks manually to avoid simultaneous execution. If two or more of the above instructions are executed simultaneously, the instruction executed later will not be executed.
- Do not execute this instruction in an interrupt program. (Execution of this instruction in an interrupt program can cause a malfunction.)
- The "PSWAPP instruction" and "Write during RUN" processing cannot be executed simultaneously.
  - When a write during RUN request is given during processing of the PSWAPP instruction, write during RUN is delayed. Write during RUN is started after the processing of the PSWAPP instruction is completed.
  - When the PSWAPP instruction is executed during write during RUN, the processing of the PSWAPP instruction is delayed. The processing of the PSWAPP instruction is started after completion of write during RUN.
- Do not execute "Read from PLC" or "Verify with PLC" and the instruction simultaneously. If the instruction is executed, "Read from PLC" or "Verify with PLC" is not complete normally because the program file status stored in the program memory is changed. If executed, execute "Read from PLC" or "Verify with PLC" again after the instruction completion.

File register high-speed block transfer

RBMOV(P)

~~Basic~~ High performance Process Redundant Ver. Universal ~~LCPU~~

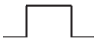
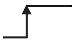
- Universal model QCPU: Not supported for Q00UJCPU



Any of the following instruction can go in the dotted squares.

RBMOV, RBMOVP

■Executing condition

Instruction	Executing condition
RBMOV	
RBMOVP	

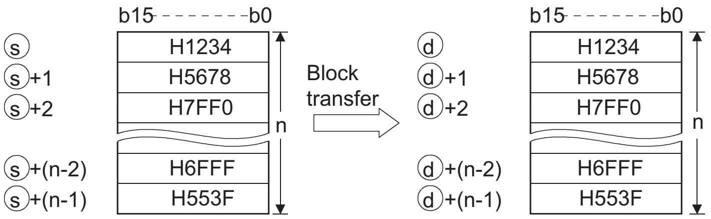
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Start number of the device that stores data to be transferred	ANY16
	n	Number of transfers	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device at the transfer destination	ANY16

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	○						—		—
n	○						○		—
(d)	○						—		—

Processing details

- Batch transfers the 16-bit data n points from the device specified for (s) to the n points of devices from the one specified for (d).



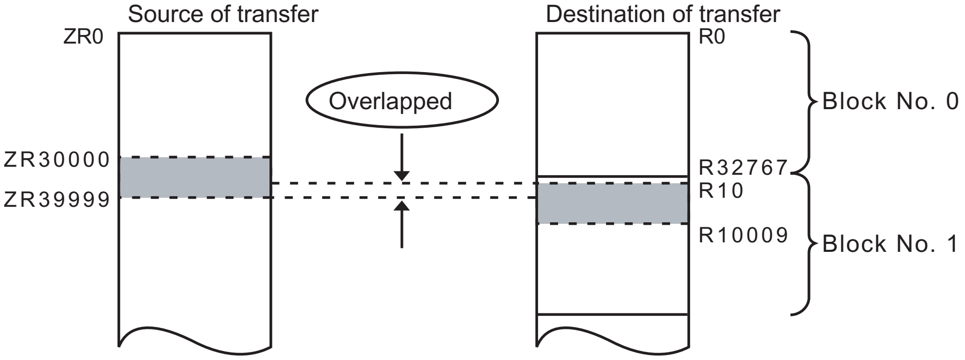
- Transfers can be accomplished even in cases where there is an overlap between the source and destination device. For the transmission to the smaller number of device, the data are transferred from (s). For the transmission to the larger number of device, the data are transferred from (s)+(n-1). However, as shown in the example below, when transferring data from R to ZR, or from ZR to R, the range to be transferred and the range of destination must not overlap.
  - ZR transfer range ((specified start number of ZR) to (specified start number of ZR + the number of transfers -1))
  - R transfer range ((specified start number of R + file register block number × 32768) to (specified start number of R + file register block number × 32768 + the number of transfers -1))

Ex.

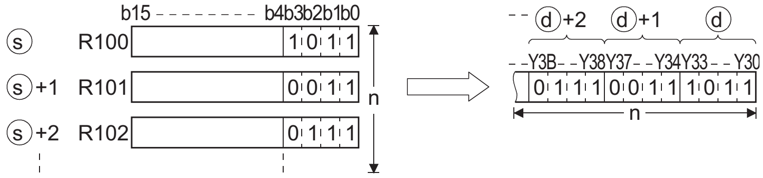
Transfer ranges of ZR and R overlap when transferring 10000 points of data from ZR30000 (source) to R10 (block No. 1 of the destination).

- ZR transfer range → (30000) to (30000+10000-1) (30000) to (39999)
- R transfer range → (10 + (1 × 32768)) to (10 + (1 × 32768) + 10000 - 1) → (32778) to (42777)

Therefore, the range from 32778 to 39999 overlaps.



- When (s) is a word device and (d) is a bit device, the target for the word device will be the number of bits specified for digit-specified bit device. If K1Y30 has been specified for (d), the lower four bits of the word device specified for (s) will become the object.



- The RBMOV(P) instruction is useful when batch transferring a large quantity of file register data with QnHCPU/QnPHCPU/QnPRHCPU. With QnUCPU, processing speed is equivalent to the BMOV instruction. The comparison of processing speed with the BMOV instruction is as follows.

■Transferring data from file registers to internal devices or from internal devices to file registers

CPU	Instruction	Target memory where file register is stored	1 word		1000 words		10000 words	
			Min	Max	Min	Max	Min	Max
QnHCPU QnPHCPU QnPRHCPU	RBMOV	Standard RAM	20.0μs		91.0μs		775.0μs	
		SRAM card	22.0μs		305.0μs		2900.0μs	
		Flash card*1	22.5μs		405.0μs		3950.0μs	
	BMOV	Standard RAM	7.5μs		76.2μs		720.0μs	
		SRAM card	8.0μs		384.0μs		3900.0μs	
		Flash card*1			418.0μs		4250.0μs	
QnCPU	RBMOV	Standard RAM	45.5μs		215.0μs		1850.0μs	
		SRAM card	49.5μs		540.0μs		5150.0μs	
		Flash card*1						
	BMOV	Standard RAM	17.5μs		177.0μs		1700.0μs	
		SRAM card	18.0μs		500.0μs		5050.0μs	
		Flash card*1			572.0μs		5800.0μs	
Q00UCPU Q01UCPU	RBMOV	Standard RAM	12.2μs	34.9μs	121.5μs	145.1μs	1111.5μs	1135.1μs
		SRAM card*2	—	—	—	—	—	—
		Flash card*2	—	—	—	—	—	—
	BMOV	Standard RAM	7.3μs	13.8μs	116.5μs	124.2μs	1106.5μs	1114.2μs
		SRAM card*2	—	—	—	—	—	—
		Flash card*2	—	—	—	—	—	—
Q02UCPU	RBMOV	Standard RAM	9.4μs	31.3μs	118.5μs	141.3μs	1108.5μs	1131.3μs
		SRAM card	9.4μs	31.4μs	178.5μs	201.3μs	1708.5μs	1731.3μs
		Flash card*1	9.4μs	32.1μs	278.5μs	301.3μs	2708.5μs	2731.3μs
	BMOV	Standard RAM	5μs	11.6μs	114.5μs	122.3μs	1104.5μs	1112.3μs
		SRAM card	5.1μs	11.7μs	174.5μs	182.3μs	1704.5μs	1712.3μs
		Flash card*1	5μs	11.6μs	274.5μs	282.3μs	2704.5μs	2712.3μs
Q03UD(E)CPU	RBMOV	Standard RAM	11.3μs	16.8μs	120.7μs	127.1μs	1110.7μs	1117.1μs
		SRAM card	11.2μs	16.7μs	180.7μs	187.1μs	1710.7μs	1717.1μs
		Flash card*1	11.3μs	16.8μs	280.7μs	287.1μs	2710.7μs	2717.1μs
	BMOV	Standard RAM	4.8μs	6.6μs	114.7μs	117.1μs	1104.7μs	1107.1μs
		SRAM card	4.8μs	6.6μs	174.7μs	177.1μs	1704.7μs	1707.1μs
		Flash card*1	4.8μs	6.5μs	274.7μs	277.1μs	2704.7μs	2707.1μs
Q04UD(E)HCPU Q06UD(E)HCPU Q10UD(E)HCPU Q13UD(E)HCPU Q20UD(E)HCPU Q26UD(E)HCPU Q50UDEHCPU Q100UDEHCPU	RBMOV	Standard RAM	9.2μs	15.1μs	61.0μs	68.6μs	531.0μs	538.6μs
		SRAM card	9.4μs	15.6μs	165.0μs	172.6μs	1576.0μs	1583.6μs
		Flash card*1	9.4μs	15.7μs	260.0μs	267.6μs	2526.0μs	2533.6μs
	BMOV	Standard RAM	4.1μs	5.6μs	56.0μs	58.6μs	526.0μs	528.6μs
		SRAM card	4.5μs	6.1μs	160.0μs	162.6μs	1571.0μs	1573.6μs
		Flash card*1	4.3μs	6.2μs	255.0μs	257.6μs	2521.0μs	2523.6μs
Q03UDVCPU	RBMOV	Standard RAM	3.7μs	21.0μs	80.6μs	89.3μs	822.2μs	831.4μs
		Extended SRAM cassette	3.7μs	21.0μs	102.6μs	118.1μs	1056.4μs	1072.0μs
	BMOV	Standard RAM	1.9μs	7.9μs	79.5μs	82.0μs	820.6μs	823.1μs
		Extended SRAM cassette	1.9μs	7.9μs	102.6μs	107.9μs	1055.5μs	1057.5μs
Q04UDVCPU, Q04UDPVCPU, Q06UDVCPU, Q06UDPVCPU, Q13UDVCPU, Q13UDPVCPU, Q26UDVCPU, Q26UDPVCPU	RBMOV	Standard RAM	3.7μs	21.0μs	42.1μs	57.7μs	413.0μs	428.6μs
		Extended SRAM cassette	3.7μs	21.0μs	102.6μs	118.1μs	1056.4μs	1072.0μs
	BMOV	Standard RAM	1.9μs	7.9μs	41.0μs	47.1μs	411.6μs	417.7μs
		Extended SRAM cassette	1.9μs	7.9μs	102.6μs	107.9μs	1055.4μs	1060.9μs

*1 When file registers are stored in the Flash card, no processing is performed for transfer from internal devices to file registers

*2 Memory card cannot be used on Q00UCPU and Q01UCPU.

■Transferring data from file registers to file registers

CPU	Instruction	Target memory where file register is stored	1 word		1000 words		10000 words	
			Min	Max	Min	Max	Min	Max
QnHCPU QnPHCPU QnPRHCPU	RBMOV	Standard RAM	20.0µs		91.0µs		775.0µs	
		SRAM card	22.5µs		545.0µs		5300.0µs	
	BMOV	Standard RAM	7.5µs		77.0µs		720.0µs	
		SRAM card	8.5µs		692.0µs		7050.0µs	
QnCPU	RBMOV	Standard RAM	45.5µs		215.0µs		1850.0µs	
		SRAM card	50.0µs		870.0µs		8350.0µs	
	BMOV	Standard RAM	17.5µs		179.0µs		1700.0µs	
		SRAM card	18.5µs		839.0µs		8600.0µs	
Q00UCPU Q01UCPU	RBMOV	Standard RAM	12.6µs	35.3µs	232.5µs	256.1µs	2211.5µs	2235.1µs
		SRAM card[*1]	—	—	—	—	—	—
	BMOV	Standard RAM	7.7µs	14.2µs	227.5µs	234.2µs	2206.5µs	2214.2µs
		SRAM card[*1]	—	—	—	—	—	—
Q02UCPU	RBMOV	Standard RAM	9.6µs	31.5µs	228.5µs	252.3µs	2208.5µs	2231.3µs
		SRAM card	9.6µs	31.5µs	378.5µs	401.3µs	3708.5µs	3731.3µs
	BMOV	Standard RAM	5.2µs	11.8µs	224.5µs	232.3µs	2204.5µs	2212.3µs
		SRAM card	5.2µs	11.8µs	374.5µs	382.3µs	3704.5µs	3712.3µs
Q03UD(E)CPU	RBMOV	Standard RAM	11.2µs	16.7µs	230.7µs	237.1µs	2210.7µs	2217.1µs
		SRAM card	11.6µs	16.7µs	380.7µs	387.1µs	3710.7µs	3717.1µs
	BMOV	Standard RAM	4.9µs	6.7µs	224.7µs	227.1µs	2204.7µs	2207.1µs
		SRAM card	5.2µs	6.7µs	374.7µs	377.1µs	3704.7µs	3707.1µs
Q04UD(E)HCPU Q06UD(E)HCPU Q10UD(E)HCPU Q13UD(E)HCPU Q20UD(E)HCPU Q26UD(E)HCPU Q50UDEHCPU Q100UDEHCPU	RBMOV	Standard RAM	9.3µs	15.5µs	118.0µs	124.6µs	1102.0µs	1107.6µs
		SRAM card	9.7µs	15.5µs	365.0µs	371.6µs	3571.0µs	3578.6µs
	BMOV	Standard RAM	4.3µs	6.2µs	113.0µs	115.6µs	1096.0µs	1098.6µs
		SRAM card	4.5µs	6.1µs	360.0µs	362.6µs	3566.0µs	3568.6µs
Q03UDVCPU	RBMOV	Standard RAM	3.7µs	20.7µs	162.0µs	171.2µs	1637.7µs	1646.4µs
		Extended SRAM cassette	3.7µs	20.7µs	216.7µs	232.1µs	2197.4µs	2212.5µs
	BMOV	Standard RAM	1.9µs	8.0µs	161.1µs	163.7µs	1636.2µs	1638.8µs
		Extended SRAM cassette	1.9µs	8.3µs	216.4µs	221.7µs	2197.4µs	2201.7µs
Q04UDVCPU, Q04UDPVCPU, Q06UDVCPU, Q06UDPVCPU, Q13UDVCPU, Q13UDPVCPU, Q26UDVCPU, Q26UDPVCPU	RBMOV	Standard RAM	3.5µs	20.7µs	84.6µs	99.5µs	836.3µs	851.7µs
		Extended SRAM cassette	3.6µs	20.7µs	216.7µs	232.1µs	2197.4µs	2212.5µs
	BMOV	Standard RAM	1.8µs	8.0µs	83.1µs	89.0µs	835.0µs	840.9µs
		Extended SRAM cassette	1.8µs	8.3µs	216.4µs	221.7µs	2197.4µs	2201.7µs

[*1] Memory card cannot be used on Q00UCPU and Q01UCPU.

Operation error

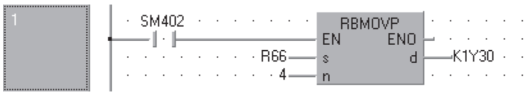
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4101	The device range of n points from (s) or (d) exceeds the corresponding device range. The file register is not specified for either (s) or (d).	—	○	○	○	○	—

Program example

- In the following program, the lower 4 bits of data in R66 to R69 are output to the devices from Y30 to Y3F in units of 4 points.

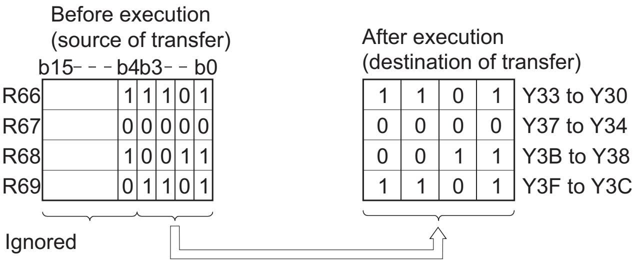
[Structured ladder/FBD]



[ST]

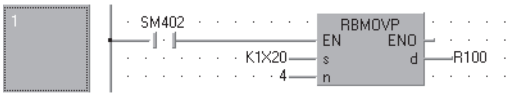
```
RBMOVP(SM402,R66,4,K1Y30);
```

[Operation]



- In the following program, the data in X20 to X2F are output to the devices from R100 to R103 in units of 4 points.

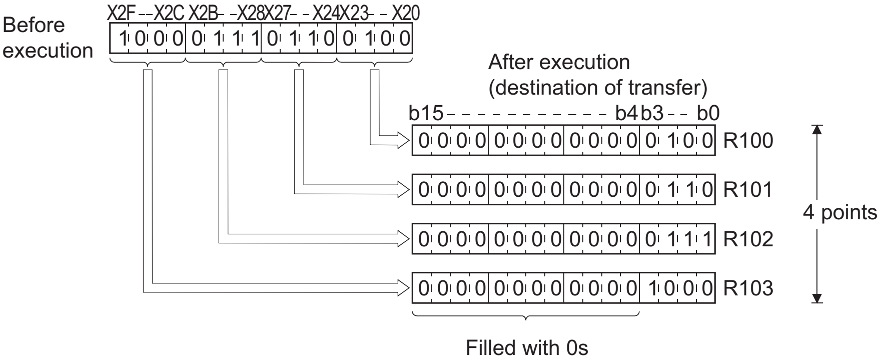
[Structured ladder/FBD]



[ST]

```
RBMOVP(SM402,K1X20,4,R100);
```

[Operation]


X2F--X2C X2B- -X28X27- -X24X23- - X20
Before execution
1 0 0 0 0 1 1 1 0 1 1 0 0 1 0 0
After execution
(destination of transfer)
b15- - - - - - - - - - - - b4b3- - b0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 R100
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 R101
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 R102
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 R103
4 points
Filled with 0s

Precautions

- When using the RBMOV instruction, do not branch a line from (d). If branched, the operation result is not output to the devices or labels of branch destination correctly. For details of branch point of line from destination, refer to ☞ Page 63 Precautions on Programming.

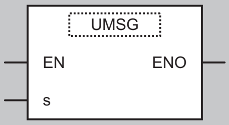
User message

UMSG

✕ Basic ✕ High performance ✕ Process ✕ Redundant ✕ Universal Ver. LCPU

- Built-in Ethernet port LCPU: Supported
- LCPU: Not supported for L02SCPU and L02SCPU-P

Structured ladder/FBD



ST

```
ENO:= UMSG (EN, s);
```

The following instruction can go in the dotted squares.

UMSG

■Executing condition

Instruction	Executing condition
UMSG	⎍

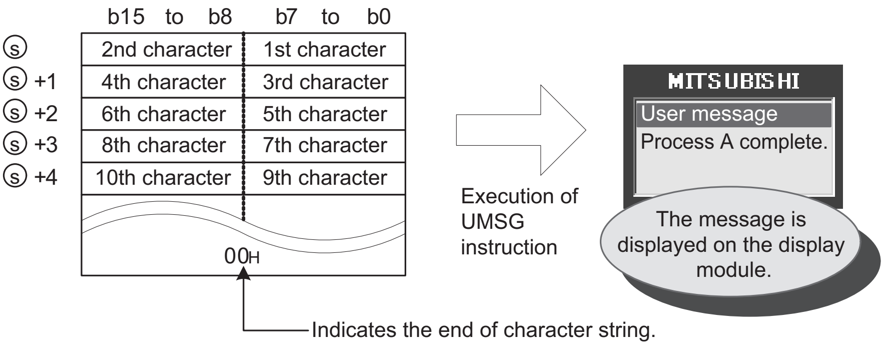
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Character string data to be displayed on the display module, or start number of the device that stores character string data	String
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant $	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—

Processing details

- Character string data specified for (s) are displayed on the display module as a user message. Character string data directly specified for (s) (specified by enclosing with " " (double quotation)), or character string data between the device number specified for (s) and the device number that stores 00H are displayed.

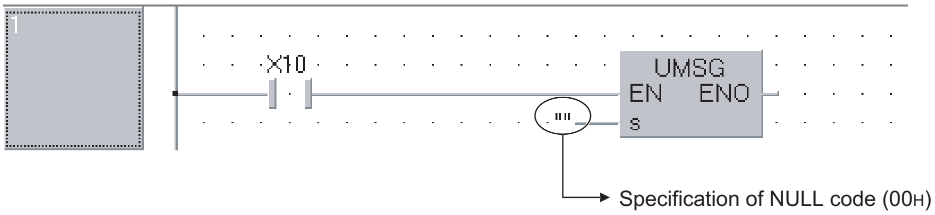


- Up to 128 characters can be displayed on the display module.

Point

The upper bytes and the lower bytes are reversed when character string data are stored to devices by using the character string data transfer instruction ($MOV).
For the $MOV instruction, refer to ☞ Page 249 Character string data transfer.

- The user message is displayed at the rising edge of the UMSG instruction command. If the character string data are changed while the command is ON, the changed user message is displayed on the display module.
- The specified character string data for the UMSG instruction are displayed at the END processing. When the multiple UMSG instructions are executed, the UMSG instruction executed directly prior to the End processing is validated. When the multiple programs are running, the UMSG instruction executed at last is validated.
- No processing is performed if the instruction is executed without the display module mounted.
- Press "ESC" on the display module to clear the displayed message. Execute "User Message" on the menu screen of the display module to redisplay the message.
- The displayed message is cleared if the NULL code (00H) is specified for the argument of the instruction. The following explains the method for specifying a NULL code (00H) for the argument of the instruction.



- For details on the display unit, refer to the following.

📖 MELSEC-L CPU Module User's Manual (Function Explanation, Program Fundamentals)

Operation error

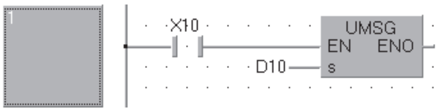
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The number of characters of (s) exceeds 128 characters.	—	—	—	—	—	○
4101	The NULL code (00H) does not exist within the specified range of the devices following the device number specified for (s).	—	—	—	—	—	○

Program example

- In the following program, the character string data in D10 and the following devices are displayed when X10 turns ON.

[Structured ladder/FBD]

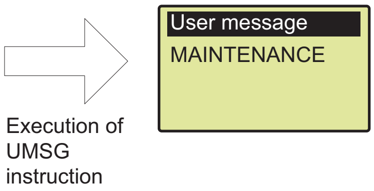


[ST]

UMSG(X10,D10);

[Operation]

	B15 to b8	B7 to b0
D10	41_H (A)	$4D_H$ (M)
D11	$4E_H$ (N)	49_H (I)
D12	45_H (E)	54_H (T)
D13	41_H (A)	$4E_H$ (N)
D14	43_H (C)	$4E_H$ (N)
D15	20_H (space)	45_H (E)
D16	0000_H	

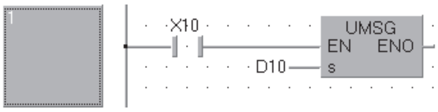


User message

MAINTENANCE

- In the following program, "A line in operation" is displayed when M0 turns ON.

[Structured ladder/FBD]

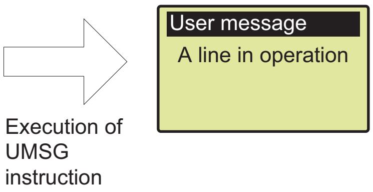


[ST]

UMSG(M0,"A line in operation");

[Operation]

B15 to b8	B7 to b0
20_H (space)	41_H (A)
69_H (i)	$6C_H$ (l)
65_H (e)	$6E_H$ (n)
69_H (i)	20_H (space)
20_H (space)	$6E_H$ (n)
70_H (p)	$6F_H$ (o)
72_H (r)	65_H (e)
74_H (t)	61_H (a)
$6F_H$ (o)	69_H (i)
20_H (space)	$6E_H$ (n)
0000_H	



User message

A line in operation

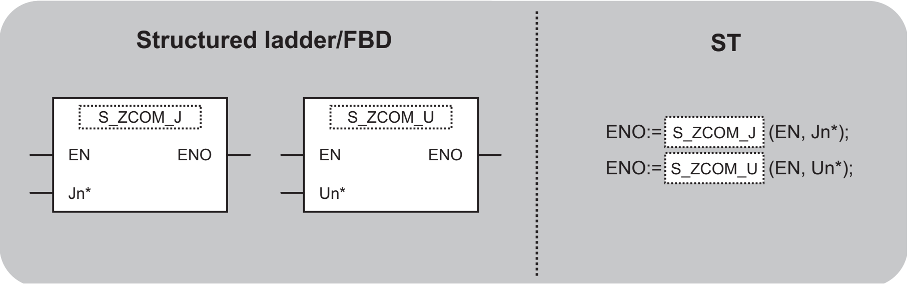
8 DATA LINK INSTRUCTIONS

8.1 Network Refresh Instructions

Refresh instruction for specified module

S(P)_ZCOM_J, S(P)_ZCOM_U

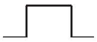
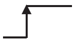
Basic | High performance | Process | Redundant | Universal | LCPU



Any of the following instruction can go in the dotted squares.

S_ZCOM_J, S_ZCOM_U, SP_ZCOM_J, SP_ZCOM_U

■Executing condition

Instruction	Executing condition
S_ZCOM_J S_ZCOM_U	
SP_ZCOM_J SP_ZCOM_U	

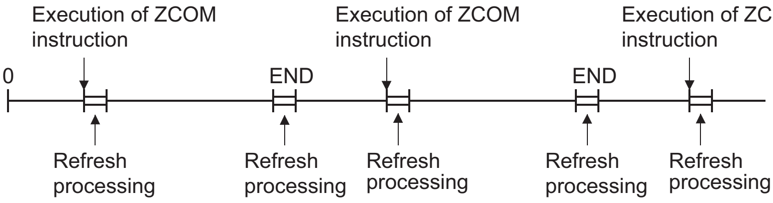
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	Jn	Network number of the host station (For QCPU (Q mode) only)	ANY16
	Un	Start I/O number of the network module of the host station (00 to FE: Higher two digits when expressing the I/O number in three digits)	ANY16
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
—	—								

Processing details

- The ZCOM instruction is used to perform refresh at any timing during execution of a sequence program. The targets of refresh performed by the ZCOM instruction are indicated below.
  - Refresh of CC-Link IE Controller Network (when refresh parameters are set) (For QCPU (Q mode) only)
  - Refresh of CC-Link IE Field Network (when refresh parameters are set) (For Universal model QCPU with a serial number whose first five digits are '12012' or higher and LCPU with a serial number whose first five digits are '13012' or higher only)
  - Refresh of MELSECNET/H (when refresh parameters are set) (For QCPU (Q mode) only)
  - Auto refresh of CC-Link (when refresh device is set)
  - Auto refresh of intelligent function module (when auto refresh is set)
- When the ZCOM instruction is executed, the CPU module temporarily suspends processing of the sequence program and conducts refresh processing of the network modules specified by Jn/Un. (For LCPU with a serial number whose first five digits are "13011" or lower, a module cannot be specified by Jn.)

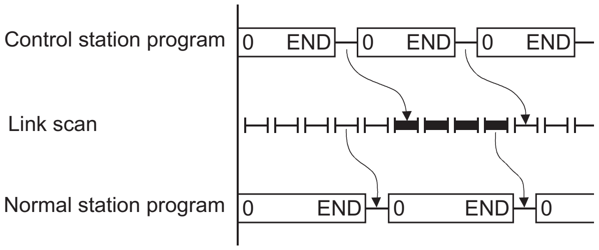


- The ZCOM instruction does not perform the following processing.
  - Communication processing between CPU module and programming tool
  - Monitor processing of other station
  - Read processing of buffer memory of other intelligent function module by serial communication module
  - Low-speed cyclic data transmission of MELSECNET/H
- The ZCOM instruction can be used as many times as desired in sequence programs. However, note that each execution of a refresh operation will lengthen the sequence program scan time by the amount of time required for the refresh operation.
- Designating "Un" in the argument enables the target designation of the intelligent function as well as the network modules. In this case, the auto refresh is performed for the buffer memory of the intelligent function modules. (It replaces the FROM/TO instructions.)
- For Universal model QCPU and LCPU, an interrupt is permitted during the execution of the ZCOM instruction. However, note that the data may be separated if the refresh data are used in an interrupt program.

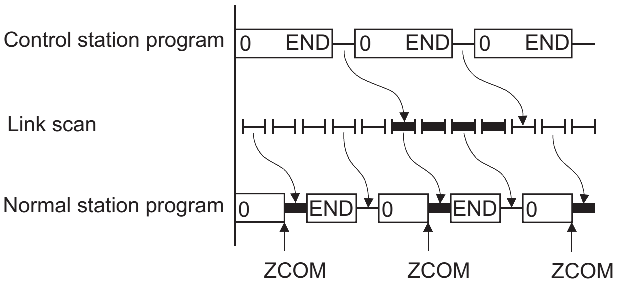
■CC-Link IE Controller Network and MELSECNET/H (PLC to PLC network)

When the scan time for the sequence program of host station is longer than the scan time for the other station, the ZCOM instruction is used to ensure the data reception from the other station.

- Example of data communications when the ZCOM instruction is not used



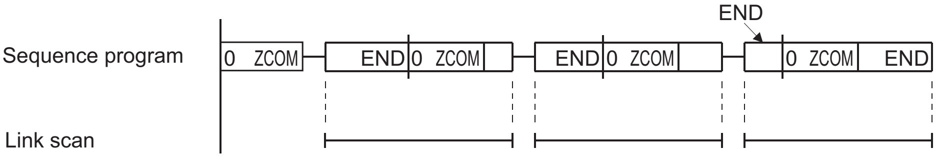
- Example of data communications when the ZCOM instruction is used



For details of the transmission delay time on the CC-Link IE Controller Network and MELSECNET/H (PLC to PLC network), refer to the following manuals.

- CC-Link IE Controller Network Reference Manual
- Q Corresponding MELSECNET/H Network System Reference Manual (PLC to PLC network)

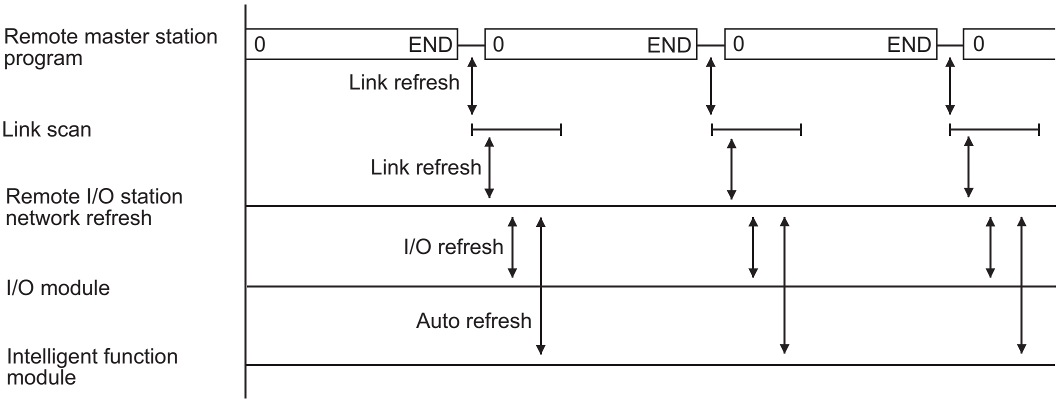
When the link scan time is longer than the sequence program scan time, data communications will not be faster even if the ZCOM instruction is used.



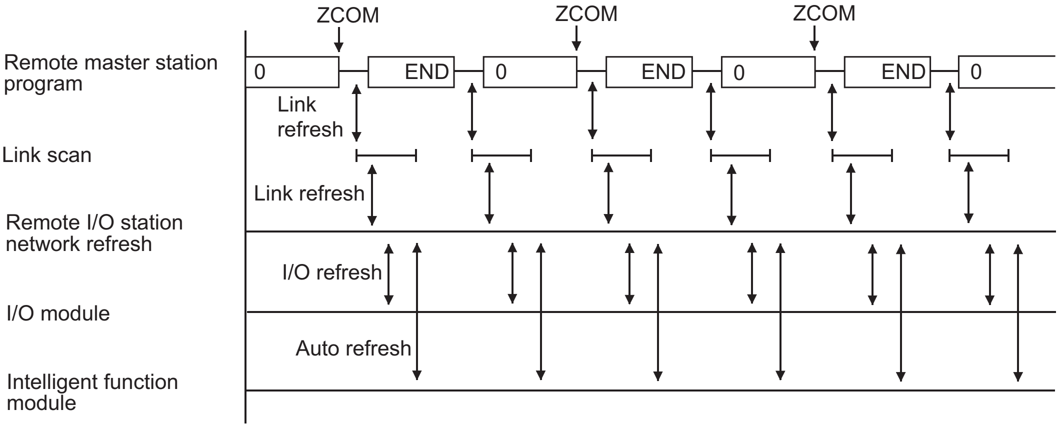
■MELSECNET/H (remote I/O network)

The link refresh of the remote master station is performed by the "END processing" of the CPU module. Since link scan is performed at completion of link refresh, link scan 'synchronizes' with the program of the CPU module. When the ZCOM instruction is used at the remote master station, link refresh is performed at the point of ZCOM instruction execution, and link scan is performed at completion of link refresh. Hence, use of the ZCOM instruction at the remote master station speeds up send/receive processing to/from the remote I/O station.

- When the ZCOM instruction is not used


Remote master station program
0 END 0 END 0 END 0
Link refresh
Link scan
Link refresh
Remote I/O station network refresh
I/O refresh
I/O module
Auto refresh
Intelligent function module

- When the ZCOM instruction is used



For details of the transmission delay time on the MELSECNET/H (remote I/O network), refer to the following manual.

- Q Corresponding MELSECNET/H Network System Reference Manual (Remote I/O network)

Point

- The ZCOM instruction cannot be used in a fixed cycle execution type program or interrupt program.
- For Redundant CPU, there are restrictions on the use of ZCOM instruction. For details on the restrictions, refer to the QnPRHCPU User's Manual (Redundant System).

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2111	When the module specified with the start I/O number is not a network module or intelligent function module.	○	○	○	○	—	—
4102	When the specified network number is not connected to the host station.	○	○	○	○	○	○*1
	When the module specified with the start I/O number is not a network module or intelligent function module.	—	—	—	—	○	○

*1 Modules with serial number (first five digits) is "13012" or later

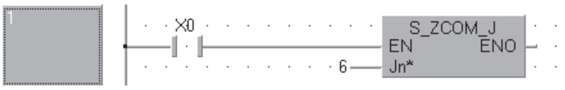
Point

To conduct only service processing, use the COM instruction. (Refer to ☞ Page 430 Refresh and Page 433 Selection of refresh(COM).)

Program example

- In the following program, a link refresh is executed on the network module of network No. 6 while X0 is ON.

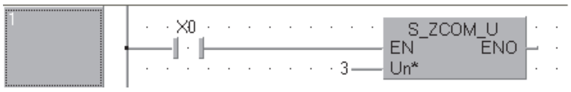
[Structured ladder/FBD]



[ST]
S_ZCOM_J(X0,6);

- In the following program, a link refresh is executed on the network module mounted to the position whose start I/O number is X/Y30 to X/Y4F while X0 is ON.

[Structured ladder/FBD]



[ST]
S_ZCOM_U(X0,3);

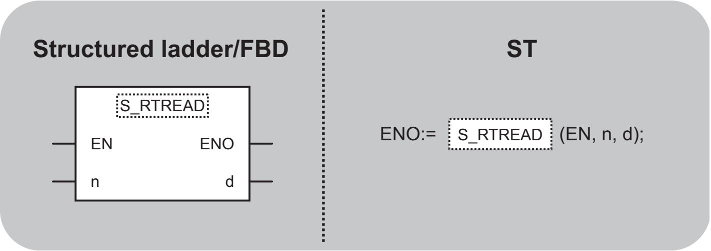
8.2 Reading/Registering Routing Information

Reading routing information

S(P)_RTREAD

Basic High performance Process Redundant Universal Ver. LCPU

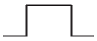
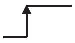
- LCPU: Supported if first 5 digits of the serial number are "13012" or later



Any of the following instruction can go in the dotted squares.

S_RTREAD, SP_RTREAD

■Executing condition

Instruction	Executing condition
S_RTREAD	
SP_RTREAD	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Transfer destination network number (1 to 239)	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number or the array of the device that stores the read data	Array of ANY16 (0..2)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○	○		—				○	—
(d)	—	○		—				—	—

Processing details

- Reads data from transfer destination network number specified for n, using routing information set by the routing parameters, and stores it to (d) and the following devices.
- If no data for the transfer destination network number specified for n is set at the routing parameters, stores 0 to (d) and the following devices.
- The content of the data stored in the area starting from (d) is as indicated below.

		(Data ranges)
ⓓ[0]	Relay network number	(1 to 239)
ⓓ[1]	Relay station number	Refer to the following table
ⓓ[2]	Dummy	

[Relay station number]

Network Type	Specified range
MELSECNET/H	1 to 64
CC-Link IE Controller Network	1 to 120
CC-Link IE Field Network	Master Station: 125 Fixed (Fixed value is stored) Local station: 1 to 120 (Station number is stored)

Operation error

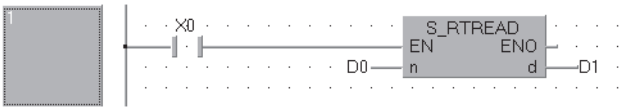
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4004	When the device which cannot be used the argument is specified	—	○	○	○	○	○
4100	When data specified for n is other than 1 to 239.	—	○	○	○	○	○
4101	The device specified for n and (d) exceeds the range of the corresponding device.	—	—	—	—	○	○

Program example

- In the following program, the routing information of the network number specified for D0 is read when X0 turns ON.

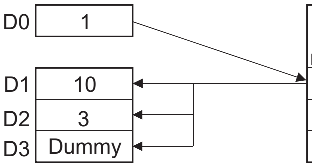
[Structured ladder/FBD]



[ST]

S_RTREAD(X0,D0,D1);

[Operation]



[Setting details of routing parameter]

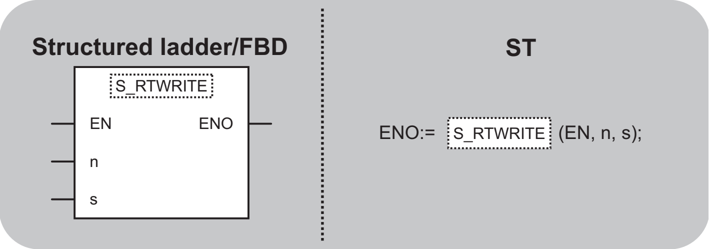
Transfer destination network number	Relay network number	Relay station number
1	10	3
2	10	2
3	10	1

Registering routing information

S(P)_RTWRITE

~~Basic~~ High performance Process Redundant Universal Ver. LCPU

- LCPU: Supported if first 5 digits of the serial number are "13012" or later

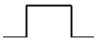
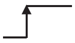


```
ENO:= S_RTWRITE (EN, n, s);
```

Any of the following instruction can go in the dotted squares.

S_RTWRITE, SP_RTWRITE

■Executing condition

Instruction	Executing condition
S_RTWRITE	
SP_RTWRITE	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n	Transfer destination network number (1 to 239)	ANY16
	s	Start number or the array of the device that stores the data to be written	Array of ANY16 (0..2)
Output argument	ENO	Execution result	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n	○	○		—				○	—
(s)	—	○		—				—	—

Processing details

- Registers routing data of (s) and the following devices in the area for the transfer destination network number specified for n in the routing parameters.

CPU module	Registerable data
The High Performance model QCPU, Process CPU, Redundant CPU, Universal model QCPU whose serial number (first five digits) is "14111" or earlier, High-speed Universal model QCPU whose serial number (first five digits) is "15041" or earlier, Universal model Process CPU and LCPU	Up to 64
• The Universal model QCPU whose serial number (first five digits) is "14112" or later, (except the High-speed Universal model QCPU and Universal model Process CPU) • High-speed Universal model QCPU whose serial number (first five digits) is "15042" or later	Up to 238

- The following shows the content of data to be set for (s) and the following devices.

		(Data ranges)
ⓢ [0]	Relay network number	(1 to 239)
ⓢ [1]	Relay station number	Refer to the following table.
ⓢ [2]	Dummy	

[Relay station number]

Network Type	Specified range
MELSECNET/H	1 to 64
CC-Link IE Controller Network	1 to 120
CC-Link IE Field Network	Master Station: 125 Fixed (Fixed value is stored) Local station: 1 to 120 (Station number is stored)

- If data of the transfer destination network number specified for n are set in the routing parameters, they are changed to the data in (s) and the following devices.
- When data in both of (s) + 0 and (s) + 1 are 0, the data of the transfer destination network number specified for n is deleted from the routing parameters.

Operation error

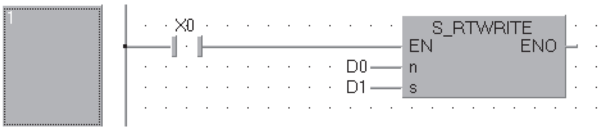
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4004	When the device which cannot be used the argument is specified	—	○	○	○	○	○
4100	When data specified for n is other than 1 to 239. When the data of (s) and the following devices exceed each setting ranges. When the total number of routing data registered in the routing parameter of the network parameters and routing data registered with the RTWRITE instruction exceeds the maximum number of registerable data. When the transfer destination network number which is not registered in the routing parameter is attempted to delete.	—	○	○	○	○	○
4101	The device specified for n and (s) exceeds the range of the corresponding device.	—	—	—	—	○	○

Program example

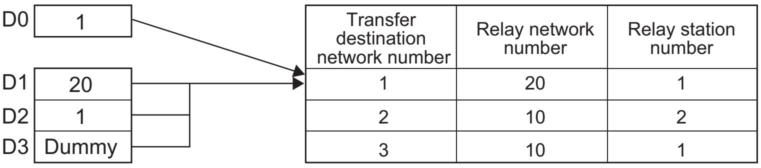
- In the following program, the routing information specified for D1 to D3 is written to the network module whose network number is specified for D0 when X0 turns ON.

[Structured ladder/FBD]



[ST]

```
S_RTWRITE(X0,D0,D1);
```



Transfer destination network number	Relay network number	Relay station number
1	20	1
2	10	2
3	10	1

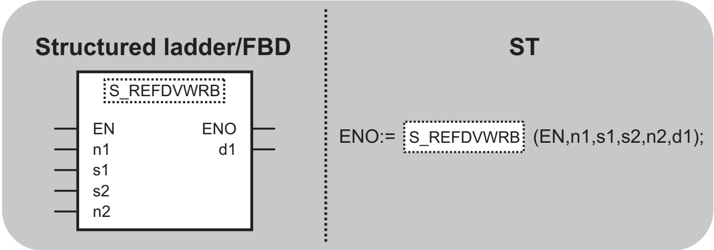
8.3 Refresh Device Write/Read Instruction

Refresh device write (in 1-bit units)

S(P)_REFDVWRB

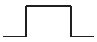
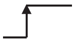
Basic ✗ High performance ✗ Process ✗ Redundant ✗ Ver. Universal Ver. LCPU

- QnUD(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "14072" or later
- Built-in Ethernet port LCPU: Supported
- Not supported for Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, QnUDVCPU, QnUDPVCPU, L02SCPU, and L02SCPU-P



Any of the following instruction can go in the dotted squares.
S_REFDVWRB, SP_REFDVWRB

■Executing condition

Instruction	Executing condition
S_REFDVWRB	
SP_REFDVWRB	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Start I/O number (0H to FEH) (BIN 16-bit)[*1]1 of the master station controlling the station assigned the refresh device which writes data	ANY16
	s1	Start number or the array of the device stored control data (device name)	Array of ANY16 (0..3)
	s2	Start number of the device stored write data to the refresh device assigned the device specified in (s1)+0 and (s1)+1 (device name)	Bit
	n2	Number of write points (1 to 2147483647)	ANY32
Output argument	ENO	Execution result	Bit
	d1	Start number or the array of the bit device which turns on for 1 scan by the instruction completion. (d1)+1 also turns on at the error completion (bit).	Array of Bit (0..1)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n1	—	○		—				○	—
(s1)	—	○		—				—	—
(s2)	△[*2]	—		—				—	—
n2	—	○		—				○	—
(d1)	△[*2]	—		—				—	—

[*1] The first 3 digits of the hexadecimal 4 digits which represent the start I/O number.

[*2] Local devices and file registers per program cannot be used as setting data.

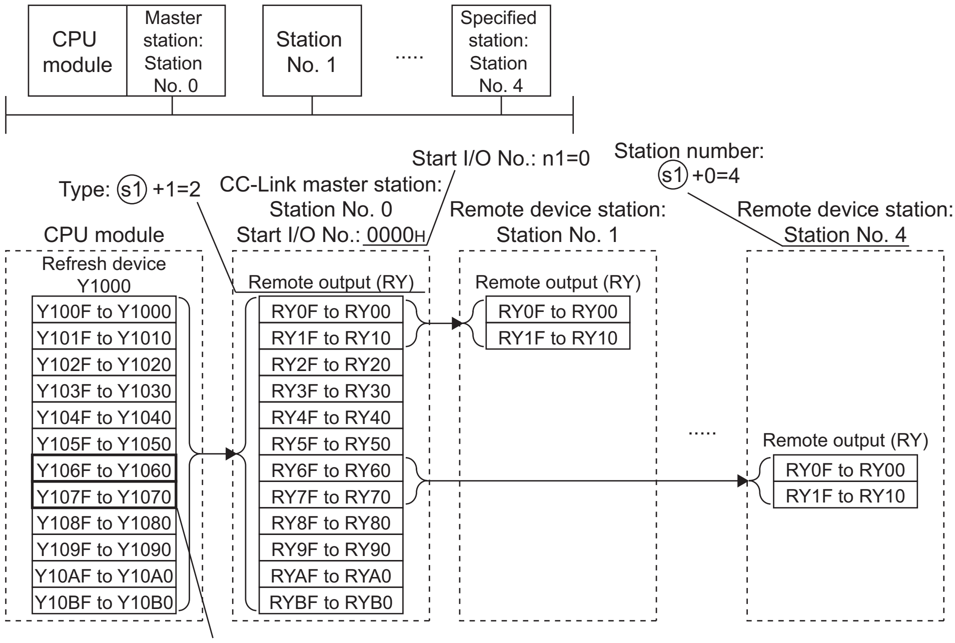
Processing details

- The contents of the data stored in the area starting from (s1) are as indicated below.

Device	Item	Setting contents	Setting range
(s1)+0	Station number	Station number for the station assigned the refresh device which writes data. When the link special relay (SB) is specified as type of (s1)+1, the setting is disabled.	1 to 120
(s1)+1	Type	Type of the refresh device which writes data • 1: Remote input (RX) • 2: Remote output (RY) • 3: Link special relay (SB)	1 to 3
(s1)+2 (s1)+3	Offset	Offset from the head of the refresh device assigned the device specified in (s1)+0 and (s1)+1	0 to 2147483647

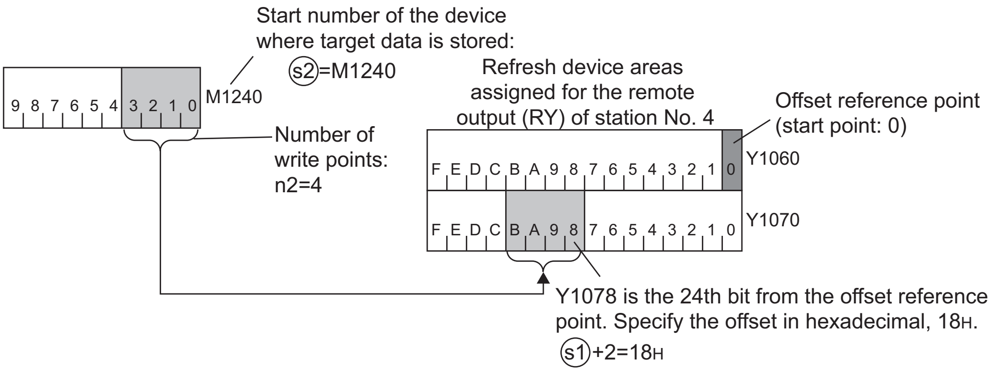
- Instruction execution possibility of an execution type for each program.
  - Enabled: Initial program and scan execution type program
  - Disabled: Fixed scan execution type program and interrupt program
- To write data to a refresh device, the data reflection to the station number specified by the instruction is executed at the timing for the auto refresh.
- Number of points specified in n2 is written from the device specified in (s2) to the offset specified in (s1)+2 of the refresh device assigned for the device specified in (s1)+1 of the target station specified in n1 and (s1)+0.

[Structure]



Refresh device areas assigned for station No. 4

At the above configuration, number of points specified in n2 is written from the device specified in (s2) to the offset (Y1078) specified in (s1)+2 of the device assigned for the station number 4.


Start number of the device where target data is stored:
(s2)=M1240
9 8 7 6 5 4 3 2 1 0
M1240
Number of write points: n2=4
Refresh device areas assigned for the remote output (RY) of station No. 4
Offset reference point (start point: 0)
F E D C B A 9 8 7 6 5 4 3 2 1 0
Y1060
Y1070
Y1078 is the 24th bit from the offset reference point. Specify the offset in hexadecimal, 18H.
(s1)+2=18H

Point

When a refresh range per station is assigned in transfer settings, specify number of write points so that the range written data from the specified offset is within the range assigned in the same transfer setting. An error occurs if the number of write points over the range assigned in each transfer setting is specified.

- The station type which can and cannot specify with the start I/O number is as follows.

Specification possibility	Station type
Enabled	CC-Link master station, CC-Link master station (compatible with redundant function), CC-Link IE Field Network master station
Disabled	CC-Link local station, CC-Link standby master station, CC-Link IE Field Network local station, CC-Link IE Field Network submaster station

- Because the available range of the station number is 1 to 120, the station number for the master station in n1 cannot be specified to (s1)+0. If the station number is specified, the "OPERATION ERROR" (error code: 4102) occurs.
- SM739 (Refresh device write/read instruction in execution flag) turns on during the instruction execution. When SM739 is on, the following instructions cannot be executed. If these instructions are executed, no processing is performed. When an error is detected at the instruction execution (before SM739 turns on), the completion device ((d1)+0), the completion device ((d1)+1), and SM739 do not turn on.
  - S(P)_REFDVWRB
  - S(P)_REFDVWRW
  - S(P)_REFDVRDB
  - S(P)_REFDVRDW
- The instruction completion can be checked in the completion device ((d1)+0 and (d1)+1).
  - Completion device ((d1)+0)

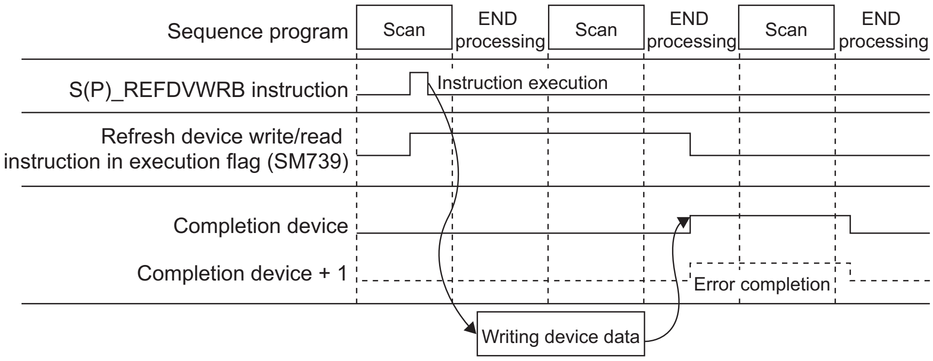
The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.

- Completion device ((d1)+1)

The device turns on or off by the status when the instruction is completed.

Normal completion: No change from off

Error completion: The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.



- A module set parameters by the dedicated instruction and a CC-Link module operating by the automatic CC-Link startup cannot be specified with the instruction.
- The write source (points in n2 from (s2)) and write destination (points in n2 from a device specified in control data) are overlapped, data can be written. Write data starting from (s2) when data are written to the smaller device number. Write data starting from (s2)+((n2)-1) when data are written to the larger device number.

Operation error

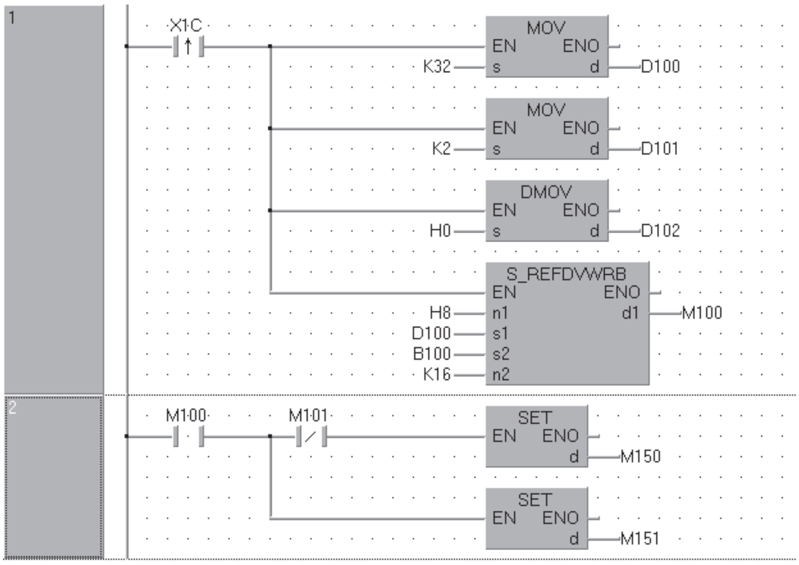
- In any of the following cases, an operation error occurs, the error flag (SM0) turns on, and an error code is stored into SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4002	When the instruction is used for the CPU module that is supported.	—	—	—	—	○	○
	When the module cannot be specified the start I/O number in n1						
4004	When the device which cannot be specified is specified	—	—	—	—	○	○
4101	When the specified device exceeds the range of the number of device points	—	—	—	—	○	○
	When the start I/O number in n1 is out of the specified range						
	When the device type number in (s1)+1 is out of the specified range						
	When the write offset in (s1)+2 is out of the specified range						
	When the number of write points in n2 is out of the specified range						
	When the number of write points in n2 exceeds the range of the number of device points						
4102	When the station number in (s1)+0 is out of the specified range	—	—	—	—	○	○
	When the station number in (s1)+0 does not exist						
	When the station number in (s1)+0 is the master station specified in n1						
4150	When the start I/O number in n1 is the station type which cannot be specified	—	—	—	—	○	○
	When the start I/O number in n1 does not exist in the network parameter						
4151	When the device in (s1)+1 of the station number specified in (s1)+0 is not assigned the refresh device	—	—	—	—	○	○
	When the write offset in (s1)+2 exceeds the refresh device range assigned for the device in (s1)+1 of the station number specified in (s1)+0						
	When the number of write points in n2 exceeds the assignment range of the setting for one transfer from the write offset in (s1)+2						

Program example

- The following program writes 16 device values in B100 to the head of the refresh device (offset: 0) assigned for the remote output (RY) in the remote I/O station on the station number 32 controlled by the CC-Link master station of the start I/O number 0080H when X1C is turned on.

[Structured ladder/FBD]



[ST]

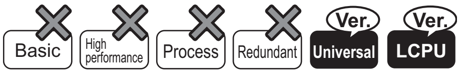
```
MOV(LDP(TRUE, X1C), K32, D100);
MOV(LDP(TRUE, X1C), K2, D101);
DMOV(LDP(TRUE, X1C), H0, D102);
SP_REFDVWRB(LDP(TRUE, X1C), H8, D100, B100, K16,M100);
IF M100=TRUE THEN
    IF M101=FALSE THEN
       SET(TRUE, M150);
    ELSE
       SET(TRUE, M151);
    END_IF;
END_IF;
```

Precautions

- Do not execute the instruction in an interrupt program. If the instruction is executed, no processing is performed. In addition, the completion device ((d1)+0), the completion device ((d1)+1), and SM739 do not turn on. If the instruction is executed in a fixed scan execution type program, they also do not turn on.
- When the instruction is executed, do not rewrite the device data in (s2) until the completion device turns on.

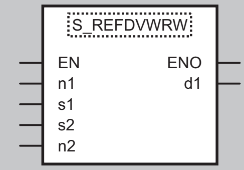
Refresh device write (in 16-bit units)

S(P)_REFDVWRW



- QnUD(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "14072" or later
- Built-in Ethernet port LCPU: Supported
- Not supported for Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, QnUDVCPU, QnUDPVCPU, L02SCPU, and L02SCPU-P

Structured ladder/FBD



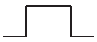
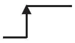
ST

ENO:= S_REFDVWRW (EN,n1,s1,s2,n2,d1);

Any of the following instruction can go in the dotted squares.

S_REFDVWRW, SP_REFDVWRW

■Executing condition

Instruction	Executing condition
S_REFDVWRW	
SP_REFDVWRW	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Start I/O number (0H to FEH) (BIN 16-bit)*11 of the master station controlling the station assigned the refresh device which writes data	ANY16
	s1	Start number or the array of the device stored control data (device name)	Array of ANY16 (0..3)
	s2	Start number of the device stored write data to the refresh device assigned the device specified in (s1)+0 and (s1)+1 (device name)	ANY16
	n2	Number of write points (1 to 2147483647)	ANY32
Output argument	ENO	Execution result	Bit
	d1	Start number or the array of the bit device which turns on for 1 scan by the instruction completion. (d1)+1 also turns on at the error completion (bit).	Array of Bit (0..1)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n1	—	○		—				○	—
(s1)	—	○		—				—	—
(s2)	—	△*2		—				—	—
n2	—	○		—				○	—
(d1)	△*2	—		—				—	—

*1 The first 3 digits of the hexadecimal 4 digits which represent the start I/O number.

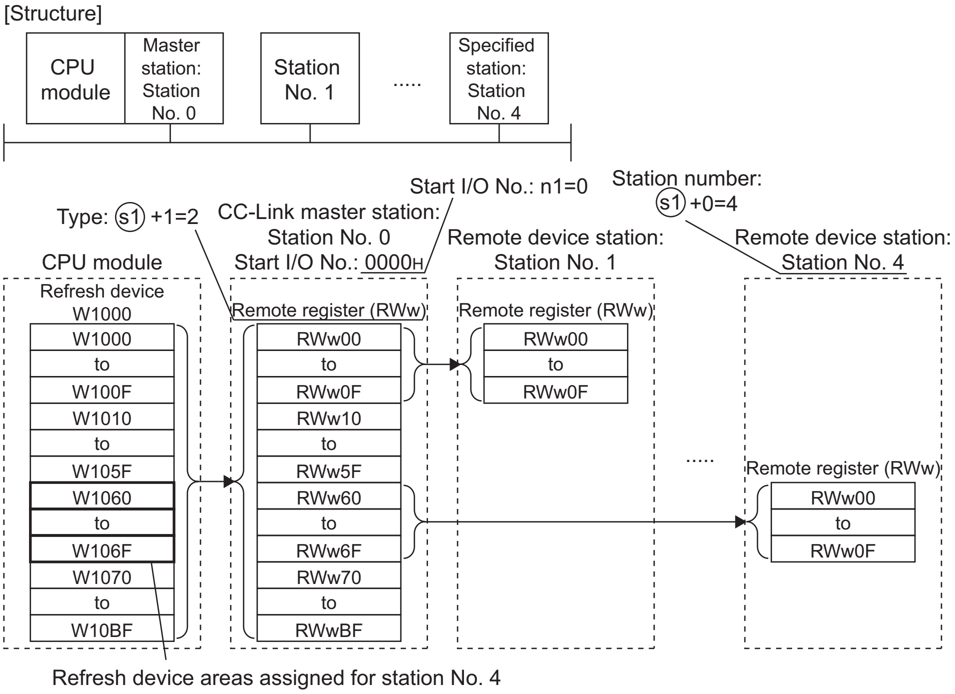
*2 Local devices and file registers per program cannot be used as setting data.

Processing details

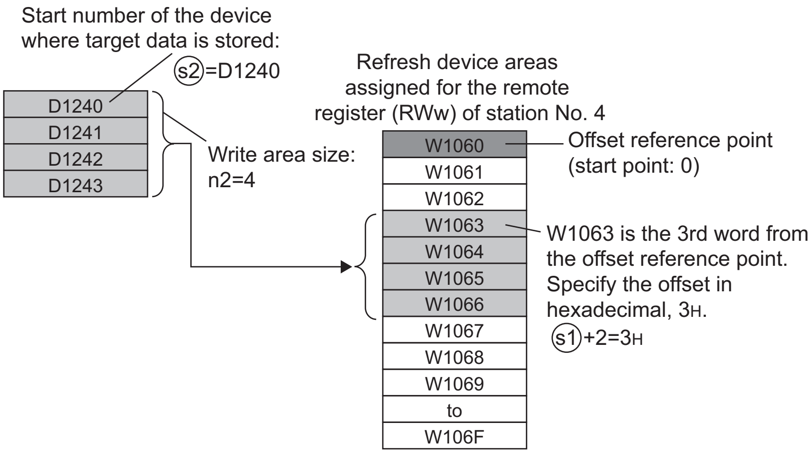
- The contents of the data stored in the area starting from (s1) are as indicated below.

Device	Item	Setting contents	Setting range
(s1)+0	Station number	Station number for the station assigned the refresh device which writes data. When the link special register (SW) is specified as type of (s1)+1, the setting is disabled.	1 to 120
(s1)+1	Type	Type of the refresh device which writes data • 1: Remote register (RWr) • 2: Remote register (RWw) • 3: Link special register (SW)	1 to 3
(s1)+2 (s1)+3	Offset	Offset from the head of the refresh device assigned the device specified in (s1)+0 and (s1)+1	0 to 2147483647

- Instruction execution possibility of an execution type for each program.
  - Enabled: Initial program and scan execution type program
  - Disabled: Fixed scan execution type program and interrupt program
- To write data to a refresh device, the data reflection to the station number specified by the instruction is executed at the timing for the auto refresh.
- Number of points specified in n2 is written from the device specified in (s2) to the offset specified in (s1)+2 of the refresh device assigned for the device specified in (s1)+1 of the target station specified in n1 and (s1)+0.



At the above configuration, number of points specified in n2 is written from the device specified in (s2) to the offset (W1063) specified in (s1)+2 of the device assigned for the station number 4.



Point

When a refresh range per station is assigned in transfer settings, specify number of write points so that the range written data from the specified offset is within the range assigned in the same transfer setting. An error occurs if the number of write points over the range assigned in each transfer setting is specified.

- The station type which can and cannot specify with the start I/O number is as follows.

Specification possibility	Station type
Enabled	CC-Link master station, CC-Link master station (compatible with redundant function), CC-Link IE Field Network master station
Disabled	CC-Link local station, CC-Link standby master station, CC-Link IE Field Network local station, CC-Link IE Field Network submaster station

- Because the available range of the station number is 1 to 120, the station number for the master station in n1 cannot be specified to (s1)+0. If the station number is specified, the "OPERATION ERROR" (error code: 4102) occurs.
- SM739 (Refresh device write/read instruction in execution flag) turns on during the instruction execution. When SM739 is on, the following instructions cannot be executed. If these instructions are executed, no processing is performed. When an error is detected at the instruction execution (before SM739 turns on), the completion device ((d1)+0), the completion device ((d1)+1), and SM739 do not turn on.
  - S(P)_REFDVWRB
  - S(P)_REFDVWRW
  - S(P)_REFDVRDB
  - S(P)_REFDVRDW
- The instruction completion can be checked in the completion device ((d1)+0 and (d1)+1).
  - Completion device ((d1)+0)

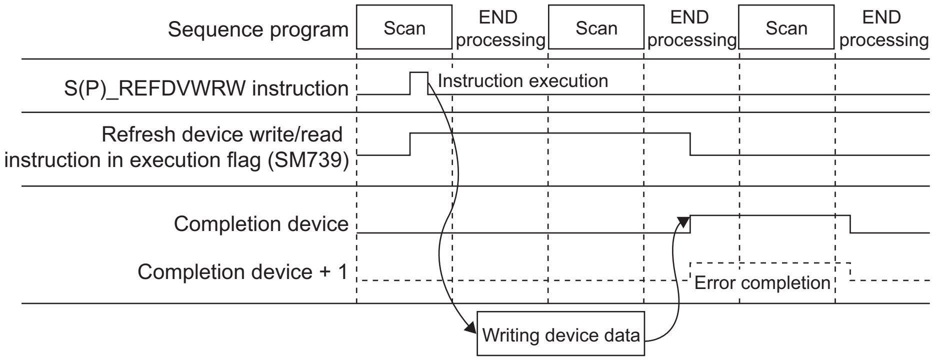
The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.

- Completion device ((d1)+1)

The device turns on or off by the status when the instruction is completed.

Normal completion: No change from off

Error completion: The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.



- A module set parameters by the dedicated instruction and a CC-Link module operating by the automatic CC-Link startup cannot be specified with the instruction.
- The write source (points in n2 from (s2)) and write destination (points in n2 from a device specified in control data) are overlapped, data can be written. Write data starting from (s2) when data are written to the smaller device number. Write data starting from (s2)+((n2)-1) when data are written to the larger device number.

Operation error

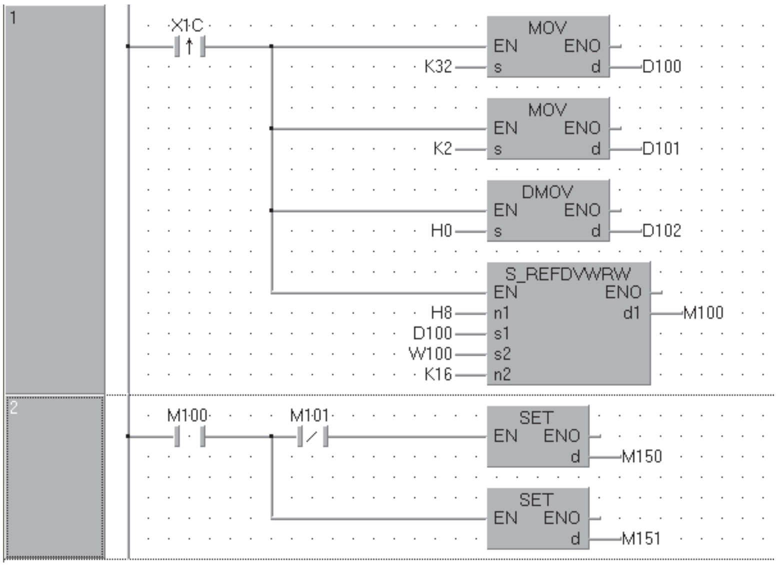
- In any of the following cases, an operation error occurs, the error flag (SM0) turns on, and an error code is stored into SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4002	When the instruction is used for the CPU module that is supported.	—	—	—	—	○	○
	When the module cannot be specified the start I/O number in n1						
4004	When the device which cannot be specified is specified	—	—	—	—	○	○
4101	When the specified device exceeds the range of the number of device points	—	—	—	—	○	○
	When the start I/O number in n1 is out of the specified range						
	When the device type number in (s1)+1 is out of the specified range						
	When the write offset in (s1)+2 is out of the specified range						
	When the number of write points in n2 is out of the specified range						
	When the number of write points in n2 exceeds the range of the number of device points						
4102	When the station number in (s1)+0 is out of the specified range	—	—	—	—	○	○
	When the station number in (s1)+0 does not exist						
	When the station number in (s1)+0 is the master station specified in n1						
4150	When the start I/O number in n1 is the station type which cannot be specified	—	—	—	—	○	○
	When the start I/O number in n1 does not exist in the network parameter						
4151	When the device in (s1)+1 of the station number specified in (s1)+0 is not assigned the refresh device	—	—	—	—	○	○
	When the write offset in (s1)+2 exceeds the refresh device range assigned for the device in (s1)+1 of the station number specified in (s1)+0						
	When the number of write points in n2 exceeds the assignment range of the setting for one transfer from the write offset in (s1)+2						

Program example

- The following program writes 16 device values in W100 to the head of the refresh device (offset: 0) assigned for the remote register (RWw) in the remote device station on the station number 32 controlled by the CC-Link master station of the start I/O number 0080H when X1C is turned on.

[Structured ladder/FBD]



[ST]

```
MOV(LDP(TRUE, X1C), K32, D100);
MOV(LDP(TRUE, X1C), K2, D101);
DMOV(LDP(TRUE, X1C), H0, D102);
S_REFDVWRW(LDP(TRUE, X1C), H8, D100, W100, K16, M100);
IF M100=TRUE THEN
    IF M101=FALSE THEN
        SET(TRUE, M150);
    ELSE
        SET(TRUE, M151);
    END_IF;
END_IF;
```

Precautions

- Do not execute the instruction in an interrupt program. If the instruction is executed, no processing is performed. In addition, the completion device ((d1)+0), the completion device ((d1)+1), and SM739 do not turn on. If the instruction is executed in a fixed scan execution type program, they also do not turn on.
- When the instruction is executed, do not rewrite the device data in (s2) until the completion device turns on.
- Specifying digit for the bit device can be used only when the following conditions (a) and (b) are met.
  - Digit specification: K4
  - Head of device: multiple of 16

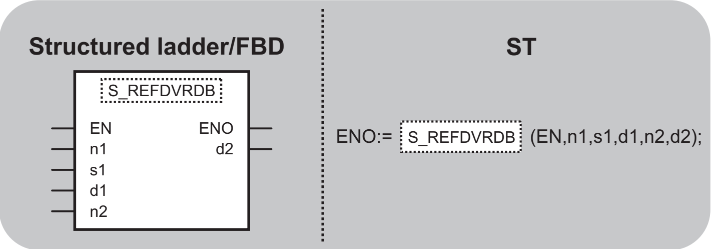
When the above conditions (a) and (b) are not met, INSTRCT CODE ERR. (error code: 4004) will occur.

Refresh device read (in 1-bit units)

S(P)_REFDVRDB

✗ Basic ✗ High performance ✗ Process ✗ Redundant Ver. Universal Ver. LCPU

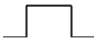
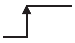
- QnUD(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "14072" or later
- Built-in Ethernet port LCPU: Supported
- Not supported for Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, QnUDVCPU, QnUDPVCPU, L02SCPU, and L02SCPU-P



Any of the following instruction can go in the dotted squares.

S_REFDVRDB, SP_REFDVRDB

■Executing condition

Instruction	Executing condition
S_REFDVRDB	
SP_REFDVRDB	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Start I/O number (0H to FEH) (BIN 16-bit)[*1] of the master station controlling the station assigned the refresh device which reads data	ANY16
	s1	Start number or the array of the device stored control data (device name)	Array of ANY16 (0..3)
	d1	Start number of the device stored read data to the refresh device assigned the device specified in (s1)+0 and (s1)+1 (device name)	Bit
	n2	Number of read points (1 to 2147483647)	ANY32
Output argument	ENO	Execution result	Bit
	d2	Start number or the array of the bit device which turns on for 1 scan by the instruction completion. (d1)+1 also turns on at the error completion (bit).	Array of Bit (0..1)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n1	—	○		—				○	—
(s1)	—	○		—				—	—
(d1)	△[*2]	—		—				—	—
n2	—	○		—				○	—
(d2)	△[*2]	—		—				—	—

[*1] The first 3 digits of the hexadecimal 4 digits which represent the start I/O number.

[*2] Local devices and file registers per program cannot be used as setting data.

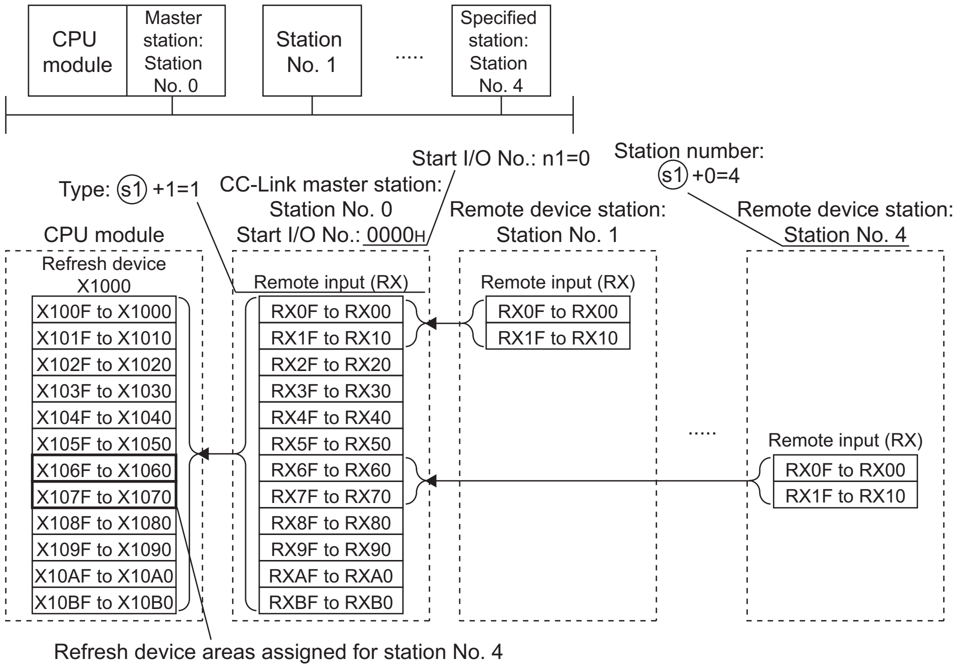
Processing details

- The contents of the data stored in the area starting from (s1) are as indicated below.

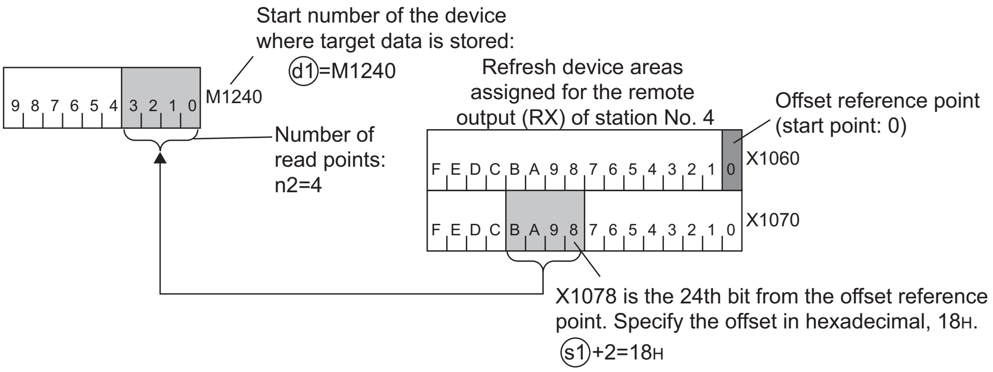
Device	Item	Setting contents	Setting range
(s1)+0	Station number	Station number for the station assigned the refresh device which reads data. When the link special relay (SB) is specified as type of (s1)+1, the setting is disabled.	1 to 120
(s1)+1	Type	Type of the refresh device which reads data • 1: Remote input (RX) • 2: Remote output (RY) • 3: Link special relay (SB)	1 to 3
(s1)+2 (s1)+3	Offset	Offset from the head of the refresh device assigned the device specified in (s1)+0 and (s1)+1	0 to 2147483647

- Instruction execution possibility of an execution type for each program.
  - Enabled: Initial program and scan execution type program
  - Disabled: Fixed scan execution type program and interrupt program
- To read data to a refresh device, the data reflection to the station number specified by the instruction is executed at the timing for the auto refresh.
- Number of points specified in n2 is read from the device specified in (s2) to the offset specified in (s1)+2 of the refresh device assigned for the device specified in (s1)+1 of the target station specified in n1 and (s1)+0.

[Structure]



At the above configuration, number of points specified in n2 is read starting from the offset (X1078) specified in (s1)+2 of the device assigned for the station number 4, then the number of points is read to devices starting from the device specified in (d1).


Start number of the device where target data is stored:
(d1)=M1240
9 8 7 6 5 4 3 2 1 0
M1240
Number of read points: n2=4
Refresh device areas assigned for the remote output (RX) of station No. 4
Offset reference point (start point: 0)
F E D C B A 9 8 7 6 5 4 3 2 1 0
X1060
F E D C B A 9 8 7 6 5 4 3 2 1 0
X1070
X1078 is the 24th bit from the offset reference point. Specify the offset in hexadecimal, 18H.
(s1)+2=18H

Point

When a refresh range per station is assigned in transfer settings, specify number of read points so that the range read data from the specified offset is within the range assigned in the same transfer setting. An error occurs if the number of read points over the range assigned in each transfer setting is specified.

- The station type which can and cannot specify with the start I/O number is as follows.

Specification possibility	Station type
Enabled	CC-Link master station, CC-Link master station (compatible with redundant function), CC-Link IE Field Network master station
Disabled	CC-Link local station, CC-Link standby master station, CC-Link IE Field Network local station, CC-Link IE Field Network submaster station

- Because the available range of the station number is 1 to 120, the station number for the master station in n1 cannot be specified to (s1)+0. If the station number is specified, the "OPERATION ERROR" (error code: 4102) occurs.
- SM739 (Refresh device write/read instruction in execution flag) turns on during the instruction execution. When SM739 is on, the following instructions cannot be executed. If these instructions are executed, no processing is performed. When an error is detected at the instruction execution (before SM739 turns on), the completion device ((d2)+0), the completion device ((d2)+1), and SM739 do not turn on.
  - S(P)_REFDVWRB
  - S(P)_REFDVWRW
  - S(P)_REFDVRDB
  - S(P)_REFDVRDW
- The instruction completion can be checked in the completion device ((d2)+0 and (d2)+1).
  - Completion device ((d2)+0)

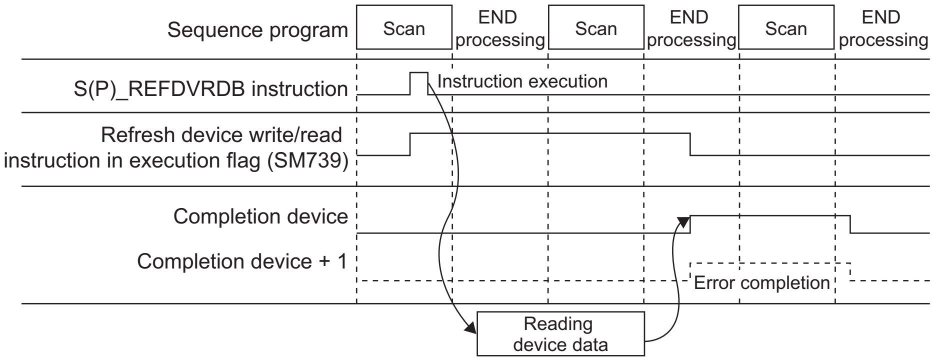
The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.

- Completion device ((d2)+1)

The device turns on or off by the status when the instruction is completed.

Normal completion: No change from off

Error completion: The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.



- A module set parameters by the dedicated instruction and a CC-Link module operating by the automatic CC-Link startup cannot be specified with the instruction.
- The read source (points in n2 from a device specified in control data) and read destination (points in n2 from (d1)) are overlapped, data can be read. Read data starting from a device specified in control data when data are read to the smaller device number. Read data starting from a device specified in control data (d1)+ ((n2)-1) when data are read to the larger device number.

Operation error

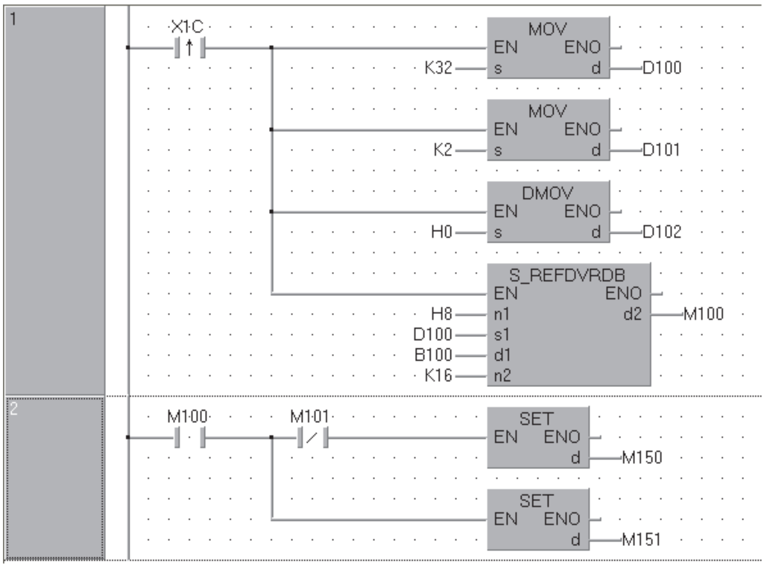
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4002	When the instruction is used for the CPU module that is supported.	—	—	—	—	○	○
	When the module cannot be specified the start I/O number in n1						
4004	When the device which cannot be specified is specified	—	—	—	—	○	○
4101	When the specified device exceeds the range of the number of device points	—	—	—	—	○	○
	When the start I/O number in n1 is out of the specified range						
	When the device type number in (s1)+1 is out of the specified range						
	When the read offset in (s1)+2 is out of the specified range						
	When the number of read points in n2 is out of the specified range						
	When the number of read points in n2 exceeds the range of the number of device points						
4102	When the station number in (s1)+0 is out of the specified range	—	—	—	—	○	○
	When the station number in (s1)+0 does not exist						
	When the station number in (s1)+0 is the master station specified in n1						
4150	When the start I/O number in n1 is the station type which cannot be specified	—	—	—	—	○	○
	When the start I/O number in n1 does not exist in the network parameter						
4151	When the device in (s1)+1 of the station number specified in (s1)+0 is not assigned the refresh device	—	—	—	—	○	○
	When the read offset in (s1)+2 exceeds the refresh device range assigned for the device in (s1)+1 of the station number specified in (s1)+0						
	When the number of read points in n2 exceeds the assignment range of the setting for one transfer from the read offset in (s1)+2						

Program example

- The following program reads 16 device values from the head of the refresh device (offset: 0) assigned for the remote input (RX) in the remote I/O station on the station number 32 controlled by the CC-Link master station of the start I/O number 0080H to B100 when X1C is turned on.

[Structured ladder/FBD]



[ST]

```
MOV(LDP(TRUE, X1C), K32, D100);
MOV(LDP(TRUE, X1C), K1, D101);
DMOV(LDP(TRUE, X1C), H0, D102);
S_REFDVRDB(LDP(TRUE, X1C), H8, D100, B100, K16, M100);
IF M100=TRUE THEN
    IF M101=FALSE THEN
        SET(TRUE, M150);
    ELSE
        SET(TRUE, M151);
    END_IF;
END_IF;
```

Precautions

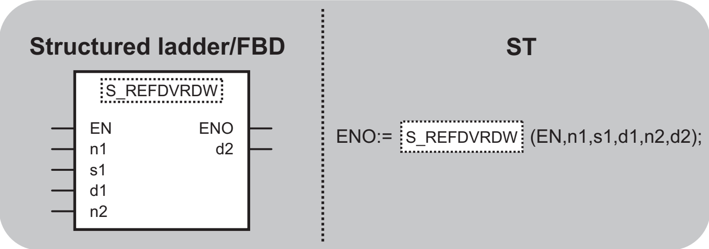
- Do not execute the instruction in an interrupt program. If the instruction is executed, no processing is performed. In addition, the completion device ((d2)+0), the completion device ((d2)+1), and SM739 do not turn on. If the instruction is executed in a fixed scan execution type program, they also do not turn on.

Refresh device read (in 16-bit units)

S(P)_REFDVRDW

✕ Basic ✕ High performance ✕ Process ✕ Redundant Ver. Universal Ver. LCPU

- QnUD(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "14072" or later
- Built-in Ethernet port LCPU: Supported
- Not supported for Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, QnUDVCPU, QnUDPVCPU, L02SCPU, and L02SCPU-P

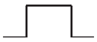
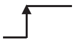


```
ENO:= S_REFDVRDW (EN,n1,s1,d1,n2,d2);
```

Any of the following instruction can go in the dotted squares.

S_REFDVRDW, SP_REFDVRDW

■Executing condition

Instruction	Executing condition
S_REFDVRDW	
SP_REFDVRDW	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Start I/O number (0H to FEH) (BIN 16-bit)[*1]1 of the master station controlling the station assigned the refresh device which reads data	ANY16
	s1	Start number or the array of the device stored control data (device name)	Array of ANY16 (0..3)
	d1	Start number of the device stored read data to the refresh device assigned the device specified in (s1)+0 and (s1)+1 (device name)	ANY16
	n2	Number of read points (1 to 2147483647)	ANY32
Output argument	ENO	Execution result	Bit
	d2	Start number or the array of the bit device which turns on for 1 scan by the instruction completion. (d2)+1 also turns on at the error completion (bit).	Array of Bit (0..1)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
n1	—	○		—				○	—
(s1)	—	○		—				—	—
(d1)	—	△[*2]		—				—	—
n2	—	○		—				○	—
(d2)	△[*2]	—		—				—	—

[*1] The first 3 digits of the hexadecimal 4 digits which represent the start I/O number.

[*2] Local devices and file registers per program cannot be used as setting data.

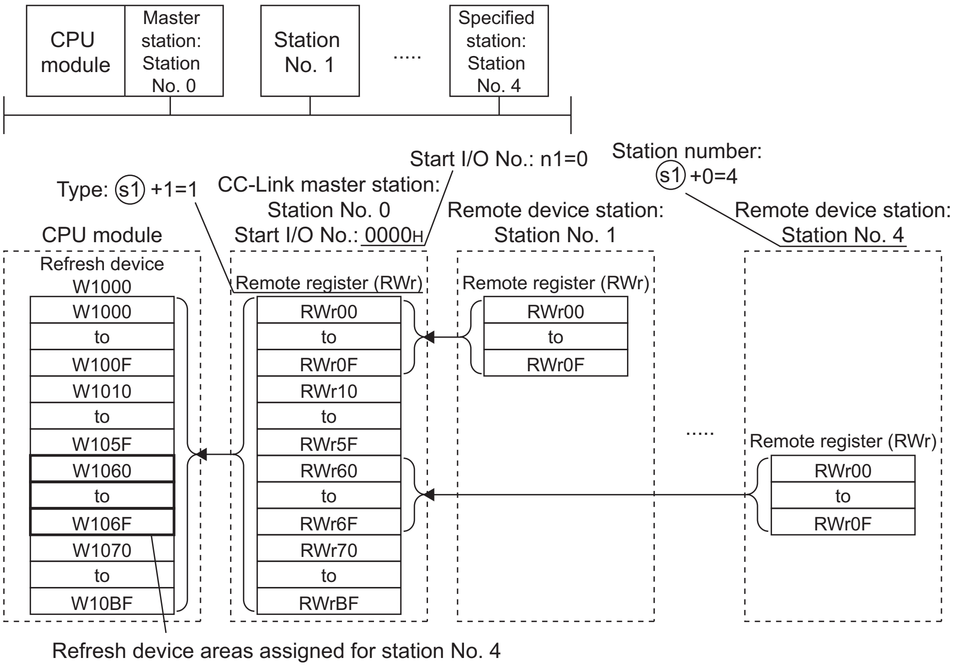
Processing details

- The contents of the data stored in the area starting from (s1) are as indicated below.

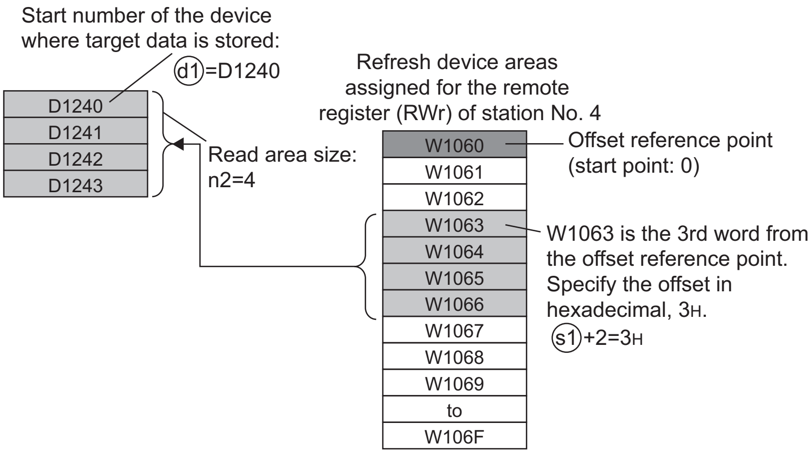
Device	Item	Setting contents	Setting range
(s1)+0	Station number	Station number for the station assigned the refresh device which reads data. When the link special register (SW) is specified as type of (s1)+1, the setting is disabled.	1 to 120
(s1)+1	Type	Type of the refresh device which reads data • 1: Remote register (RWr) • 2: Remote register (RWw) • 3: Link special register (SW)	1 to 3
(s1)+2 (s1)+3	Offset	Offset from the head of the refresh device assigned the device specified in (s1)+0 and (s1)+1.	0 to 2147483647

- Instruction execution possibility of an execution type for each program.
  - Enabled: Initial program and scan execution type program
  - Disabled: Fixed scan execution type program and interrupt program
- To read data to a refresh device, the data reflection to the station number specified by the instruction is executed at the timing for the auto refresh.
- Number of points specified in n2 is read from the device specified in (s2) to the offset specified in (s1)+2 of the refresh device assigned for the device specified in (s1)+1 of the target station specified in n1 and (s1)+0.

[Structure]



At the above configuration, number of points specified in n2 is read starting from the offset (W1063) specified in (s1)+2 of the device assigned for the station number 4, then the number of points is read to devices starting from the device specified in (d1).



Point

When a refresh range per station is assigned in transfer settings, specify number of read points so that the range read data from the specified offset is within the range assigned in the same transfer setting. An error occurs if the number of read points over the range assigned in each transfer setting is specified.

- The station type which can and cannot specify with the start I/O number is as follows.

Specification possibility	Station type
Enabled	CC-Link master station, CC-Link master station (compatible with redundant function), CC-Link IE Field Network master station
Disabled	CC-Link local station, CC-Link standby master station, CC-Link IE Field Network local station, CC-Link IE Field Network submaster station

- Because the available range of the station number is 1 to 120, the station number for the master station in n1 cannot be specified to (s1)+0. If the station number is specified, the "OPERATION ERROR" (error code: 4102) occurs.
- SM739 (Refresh device write/read instruction in execution flag) turns on during the instruction execution. When SM739 is on, the following instructions cannot be executed. If these instructions are executed, no processing is performed. When an error is detected at the instruction execution (before SM739 turns on), the completion device ((d2)+0), the completion device ((d2)+1), and SM739 do not turn on.
  - S(P)_REFDVWRB
  - S(P)_REFDVWRW
  - S(P)_REFDVRDB
  - S(P)_REFDVRDW
- The instruction completion can be checked in the completion device ((d2)+0 and (d2)+1).
  - Completion device ((d2)+0)

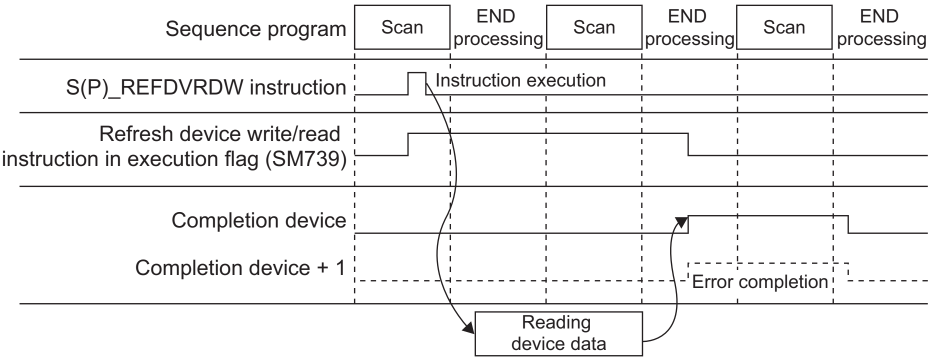
The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.

- Completion device ((d2)+1)

The device turns on or off by the status when the instruction is completed.

Normal completion: No change from off

Error completion: The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.



- A module set parameters by the dedicated instruction and a CC-Link module operating by the automatic CC-Link startup cannot be specified with the instruction.
- The read source (points in n2 from a device specified in control data) and read destination (points in n2 from (d1)) are overlapped, data can be read. Read data starting from a device specified in control data when data are read to the smaller device number. Read data starting from a device specified in control data (d1)+ ((n2)-1) when data are read to the larger device number.

Operation error

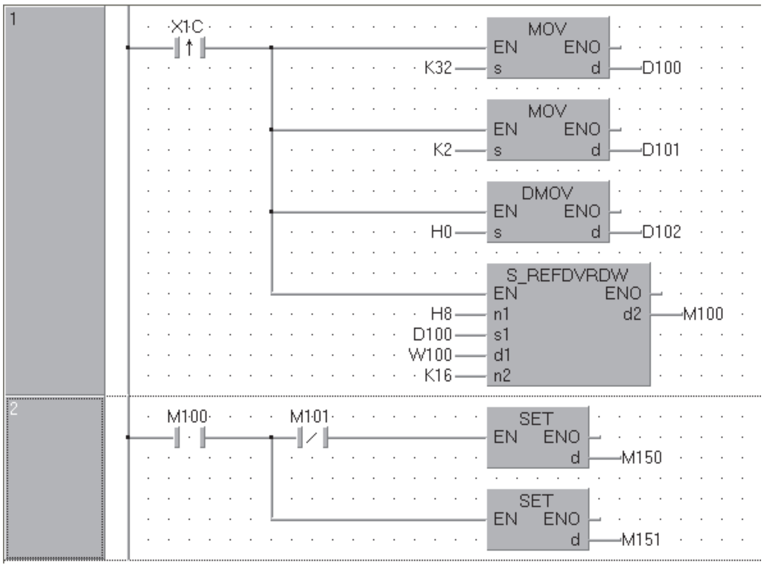
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
4002	When the instruction is used for the CPU module that is supported.	—	—	—	—	○	○
	When the module cannot be specified the start I/O number in n1						
4004	When the device which cannot be specified is specified	—	—	—	—	○	○
4101	When the specified device exceeds the range of the number of device points	—	—	—	—	○	○
	When the start I/O number in n1 is out of the specified range						
	When the device type number in (s1)+1 is out of the specified range						
	When the read offset in (s1)+2 is out of the specified range						
	When the number of read points in n2 is out of the specified range						
	When the number of read points in n2 exceeds the range of the number of device points						
4102	When the station number in (s1)+0 is out of the specified range	—	—	—	—	○	○
	When the station number in (s1)+0 does not exist						
	When the station number in (s1)+0 is the master station specified in n1						
4150	When the start I/O number in n1 is the station type which cannot be specified	—	—	—	—	○	○
	When the start I/O number in n1 does not exist in the network parameter						
4151	When the device in (s1)+1 of the station number specified in (s1)+0 is not assigned the refresh device	—	—	—	—	○	○
	When the read offset in (s1)+2 exceeds the refresh device range assigned for the device in (s1)+1 of the station number specified in (s1)+0						
	When the number of read points in n2 exceeds the assignment range of the setting for one transfer from the read offset in (s1)+2						

Program example

- The following program reads 16 device values from the head of the refresh device (offset: 0) assigned for the remote register (RWr) in the remote device station on the station number 32 controlled by the CC-Link master station of the start I/O number 0080H to W100 when X1C is turned on.

[Structured ladder/FBD]



[ST]

```
MOV(LDP(TRUE, X1C), K32, D100);
MOV(LDP(TRUE, X1C), K1, D101);
DMOV(LDP(TRUE, X1C), H0, D102);
S_REFDVRDW(LDP(TRUE, X1C), H8, D100, W100, K16, M100);
IF M100=TRUE THEN
    IF M101=FALSE THEN
        SET(TRUE, M150);
    ELSE
        SET(TRUE, M151);
    END_IF;
END_IF;
```

Precautions

- Do not execute the instruction in an interrupt program. If the instruction is executed, no processing is performed. In addition, the completion device ((d2)+0), the completion device ((d2)+1), and SM739 do not turn on. If the instruction is executed in a fixed scan execution type program, they also do not turn on.
- Specifying digit for the bit device can be used only when the following conditions (a) and (b) are met.
  - Digit specification: K4
  - Head of device: multiple of 16

When the above conditions (a) and (b) are not met, INSTRCT CODE ERR. (error code: 4004) will occur.

9 MULTIPLE CPU DEDICATED INSTRUCTIONS

9.1 Writing Data to Host CPU Shared Memory

The S(P)_TO or TO(P) instruction is used to write to the host CPU shared memory in the multiple CPU system.

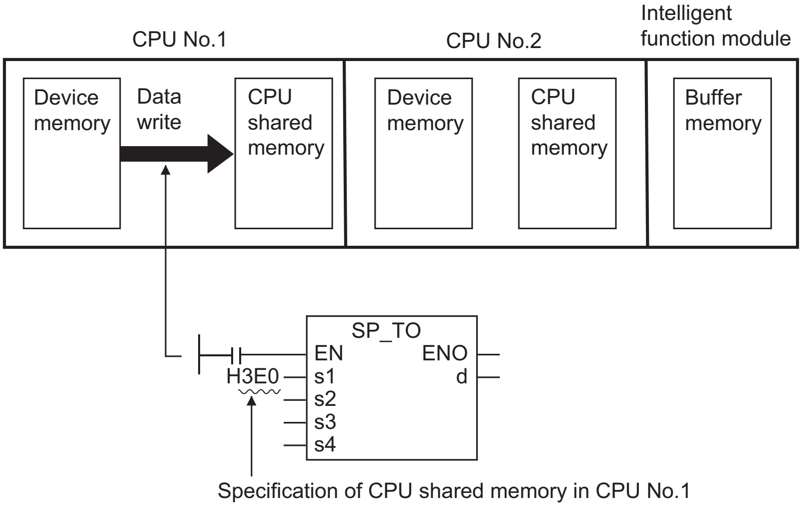
The following table indicates the usability of the S(P)_TO and TO(P) instructions.

CPU module model		S(P) TO instruction	TO(P) instruction
Basic model QCPU	Q00JCPU	Unusable	Unusable
	Q00CPU and Q01CPU	Usable	Usable
High Performance model QCPU	Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU, and Q25HCPU	Usable	Unusable
Process CPU	Q02PHCPU, Q06PHCPU, Q12PHCPU and Q25PHCPU	Usable	Unusable
Redundant CPU	Q12PRHCPU and Q25PRHCPU	Unusable	Unusable
Universal model QCPU	Q00UJCPU	Unusable	Unusable
	Q00UCPU, Q01UCPU, Q02UCPU, Q03UDCPU, Q03UDVCPU, Q03UDECPU, Q04UDHCPU, Q04UDVCPU, Q04UDPVCPU, Q04UDEHCPU, Q06UDHCPU, Q06UDVCPU, Q06UDPVCPU, Q06UDEHCPU, Q10UDHCPU, Q10UDEHCPU, Q13UDHCPU, Q13UDVCPU, Q13UDPVCPU, Q13UDEHCPU, Q20UDHCPU, Q20UDEHCPU, Q26UDHCPU, Q26UDVCPU, Q26UDPVCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU	Usable	Usable
LCPU	L02SCPU, L02SCPU-P, L02CPU, L02CPU-P, L06CPU, L06CPU-P, L26CPU, L26CPU-P, L26CPU-BT, L26CPU-PBT	Unusable	Unusable

Operation of S(P)_TO instruction

The S(P)_TO instruction can write data to the CPU shared memory of the host CPU module.

The following figure shows the processing performed when the S(P)_TO instruction is executed in CPU No. 1.

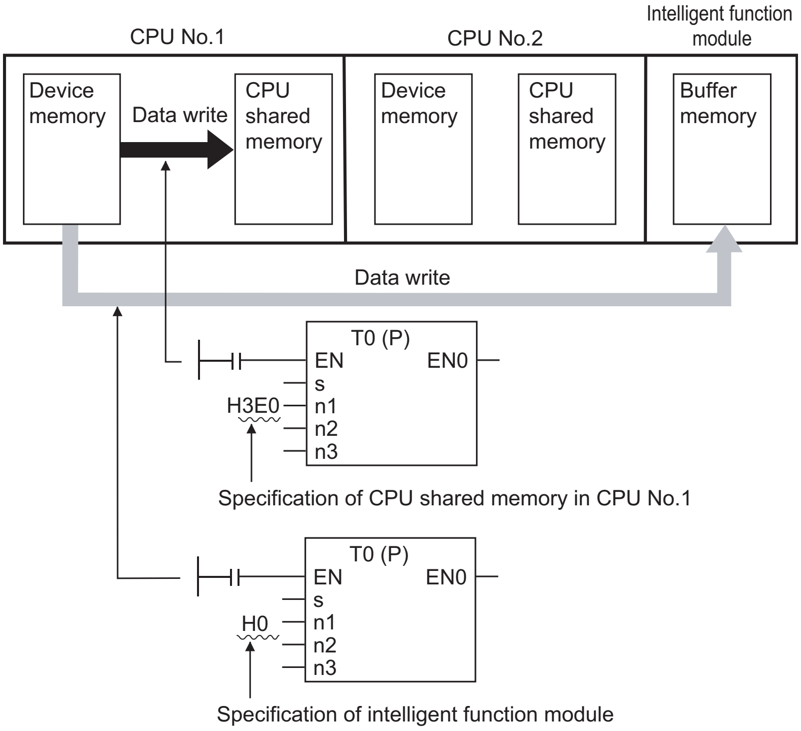


Operation of TO(P) instruction

The TO(P) instruction can write device memory data to the following memory.

- CPU shared memory of the host CPU module
- Buffer memory of the intelligent function module

The following figure shows the processing performed when the TO(P) instruction is executed in CPU No. 1.



Point

For Basic model QCPU (Q00CPU, Q01CPU) and Universal model QCPU, both S(P)_TO and TO(P) instructions can be used to write data to the CPU shared memory. However, use of the TO(P) instruction is recommended in order to reduce the number of steps and shorten the processing time for writing data to the host CPU shared memory.

For writing to the buffer memory of the intelligent function module by the TO instruction, refer to the following.

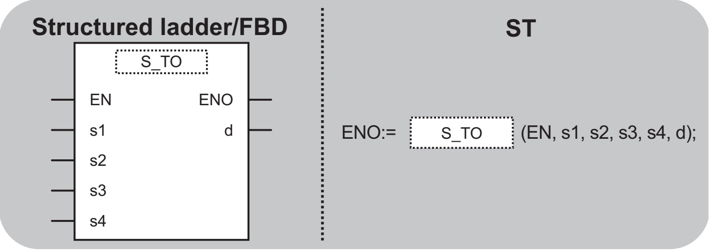
☞ Page 454 Writing 1-/2-word data to intelligent function module

Writing data to host CPU shared memory(S(P)_TO)

S(P)_TO

Ver. Basic | Ver. High performance | Process | Redundant ✗ | Universal | LCPU ✗

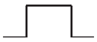
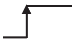
- Q00CPU and Q01CPU: Supported if first 5 digits of the serial number are "04122" or later
- High Performance model QCPU: Supported if the function version B or later



Any of the following instruction can go in the dotted squares.

S_TO, SP_TO

■Executing condition

Instruction	Executing condition
S_TO	
SP_TO	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s1	Start I/O number of the host CPU*1	ANY16
	s2	Host CPU shared memory addresses of the write destination (BIN 16 bits) Basic model QCPU: 0 to 511 High Performance model QCPU, Process CPU, and Universal model QCPU: 0 to 4095	ANY16
	s3	Start number of the device that stores write data	ANY16
	s4	Number of write points Basic model QCPU: 1 to 320 High Performance model QCPU, Process CPU: 1 to 255 Universal model QCPU: 1 to 2048	ANY16
Output argument	ENO	Execution result	Bit
	d	Host CPU device to be turned ON for one scan at the write completion	Bit

*1 Specified by the upper 3 digits of start I/O number expressed in 4-digit hexadecimal.

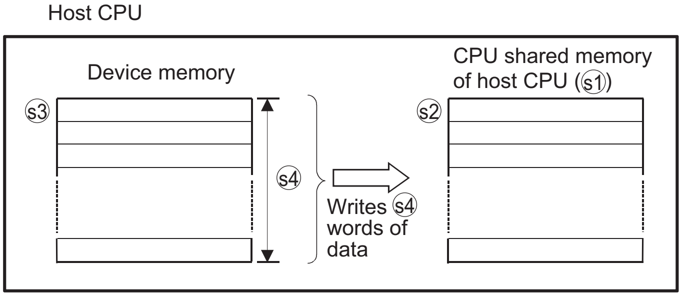
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s1)	—	○		—				○	—
(s2)	—	○		—				○	—
(s3)	—	○		—				—	—
(s4)	—	○		—				○	—
(d)	○	○		—				—	—

The (s1) is specified by the first 3 digits of the hexadecimal 4 digits which represent the start I/O number of the slot on which the CPU module is mounted.

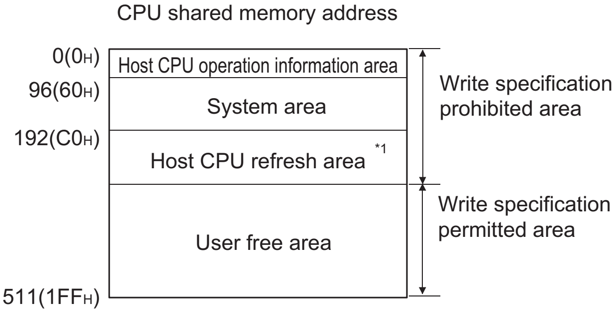
Slot No.	Start I/O number	(s1)
CPU Slot	3E00	3E0
Slot 0	3E10	3E1
Slot 1	3E20	3E2
Slot 2	3E30	3E3

Processing details

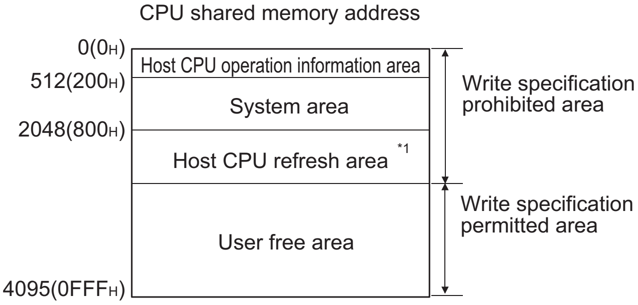
- Writes (s4) words of device data from host CPU module (s3) to the CPU shared memory address specified for (s2) and the following addresses of host CPU module. When writing is completed, the completion bit specified for (d) turns ON.



- CPU shared memory address of the Basic model QCPU


CPU shared memory address
0(0H)
96(60H)
192(C0H)
511(1FFH)
Host CPU operation information area
System area
Host CPU refresh area *1
User free area
Write specification prohibited area
Write specification permitted area

- CPU shared memory address of the High Performance model QCPU, Process CPU, and Universal model QCPU*2


CPU shared memory address
0(0H)
512(200H)
2048(800H)
4095(0FFFH)
Host CPU operation information area
System area
Host CPU refresh area *1
User free area
Write specification prohibited area
Write specification permitted area

*1 Usable as a user setting area when auto refresh setting is not set. In addition, even when auto refresh setting is set, the auto refresh send range or later is usable as a user setting area.

*2 Data cannot be written to the multiple CPU high speed transmission area of the Universal model QCPU using the S(P)_TO instruction.

- When the number of write points is 0, no processing is performed and the completion device does not turn ON.
- The S(P)_TO instruction can be executed once per one scan for each CPU. When execution condition is established at two or more places at the same time, the S(P)_TO instruction executed later is not processed since handshake is established automatically.
- The number of data that can be written varies depending on the target CPU module

CPU module	Number of write points
Basic model QCPU	1 to 320
High Performance model QCPU, Process CPU	1 to 256
Universal model QCPU	1 to 2048

Point

Data Write to CPU shared memory can be performed using the intelligent function module device.
For the intelligent function module device, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

Operation error

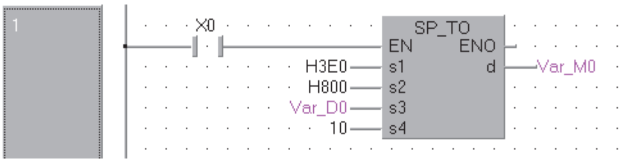
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2107	When the start I/O number (s1) of the host CPU is other than that of the host CPU.	—	○	○	—	—	—
2110	The CPU module does not exist in the position specified by the start I/O number of the CPU module.	○	○	○	—	○	—
4002	When the specified instruction is improper.	○	○	○	—	○	—
4003	When the specified number of devices is wrong.	○	○	○	—	○	—
4004	When the unusable device is specified.	○	○	○	—	○	—
4100	When the start I/O number (s1) of the host CPU is other than 3E0H, 3E1H, 3E2H or 3E3H.	○	○	○	—	○	—
4101	When the host CPU operation information area, restricted system area or host CPU refresh area is specified to the CPU shared memory address (s2) of the write destination.	—	○	○	—	—	—
	When the number of write points (s4) is outside the specified range of set data. When the start of the CPU shared memory address (s2) of the write destination host CPU exceeds the CPU shared memory address range. When the CPU shared memory address (s2) + the number of write points (s4) exceeds the CPU shared memory address range. When the start number of the devices (s3) where the data to be written is stored + the number of write points (s4) exceeds the device range.	○	○	○	—	○	—
4111	When the host CPU operation information area, restricted system area or host CPU refresh area is specified to the CPU shared memory address (s2) of the write destination.	○	—	—	—	○	—
4112	When the start I/O number (s1) of the host CPU is other than that of the host CPU.	○	—	—	—	○	—

Program example

- In the following program, 10 points of data from Var_D0 are stored to the address 800H of the CPU shared memory of CPU No. 1 when X0 turns ON.

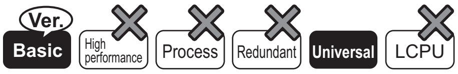
[Structured ladder/FBD]



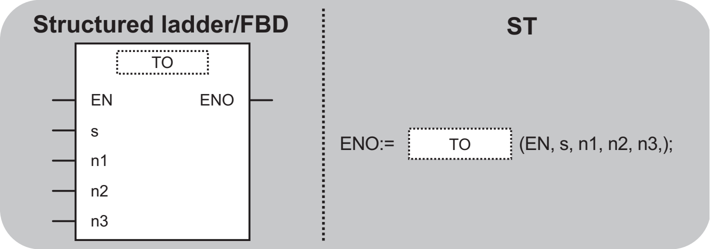
[ST]
SP_TO(X0,H3E0,H800,Var_D0,10,Var_M0);

Writing data to host CPU shared memory(TO(P), DTO(P))

TO(P), DTO(P)



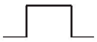
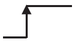
- Q00CPU and Q01CPU: Supported if first 5 digits of the serial number are "04122" or later



Any of the following instruction can go in the dotted squares.

TO, TOP, DTO, DTOP

■Executing condition

Instruction	Executing condition
TO, DTO	
TOP, DTOP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Write data, or start number of the device that stores write data	ANY16/32
	n1	Start I/O number of the host CPU[*1] Basic model QCPU: 3E0H Universal model QCPU: 3E0H to 3E3H	ANY16
	n2	Host CPU shared memory address of the write destination Basic model QCPU: 192 to 511 Universal model QCPU: 2048 to 4095, 10000 to 24335[*2]	ANY16
	n3	Number of write points 1 to 320 Basic model QCPU: FROM(P): 1 to 512, DFRO(P): 1 to 256 Universal model QCPU: FROM(P): 1 to 14336[*2], DFRO(P): 1 to 7168[*2]	ANY16
Output argument	ENO	Execution result	Bit

*1 Specified by the upper 3 digits of start I/O number expressed in 4-digit hexadecimal.

*2 The setting range varies depending on the auto refresh setting range of the multiple CPU high speed transmission function.

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others U
	Bit	Word		Bit	Word				
(s)	○			—				○	—
n1	○			○				○	○
n2	○			○				○	—
n3	○			○				○	—

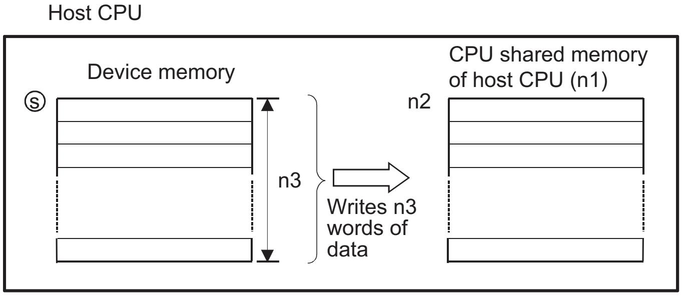
The n1 is specified by the first 3 digits of the hexadecimal 4 digits which represent the start I/O number of the slot on which the CPU module is mounted.

	CPU Slot	Slot 0	Slot 1	Slot 2
Start I/O number	3E00	3E10	3E20	3E30
n1	3E0	3E1	3E2	3E3

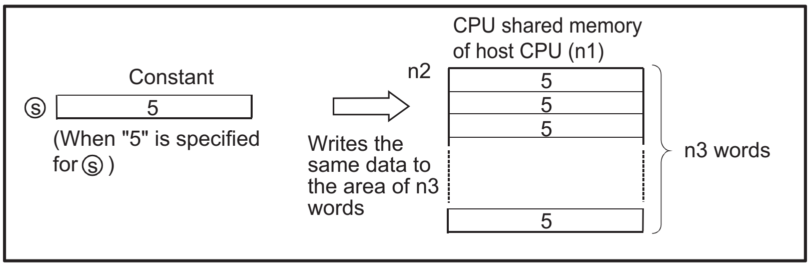
Processing details

■TO

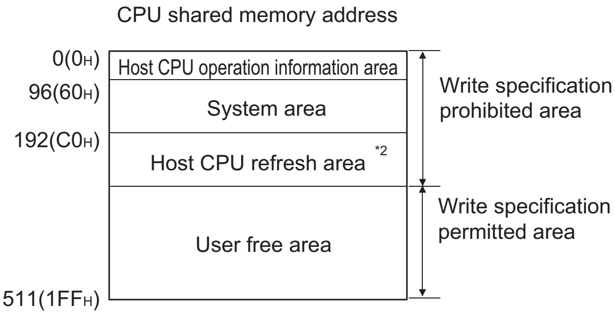
- Writes n3 words of device data from host CPU module (s) to the CPU shared memory address specified for n2 and the following addresses of host CPU module.



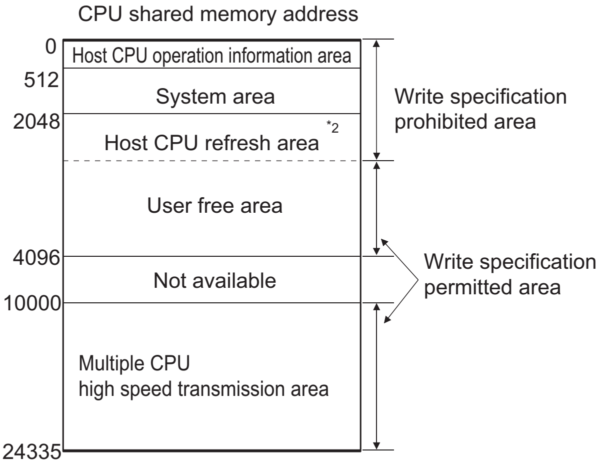
- When a constant is specified for (s), writes the same data (value specified for (s)) to the area of n3 words from the specified CPU shared memory.


Constant
ⓢ
5
(When "5" is specified for ⓢ)
Writes the same data to the area of n3 words
n2
CPU shared memory of host CPU (n1)
5
5
5
5
n3 words

- CPU shared memory address of the Basic model QCPU


CPU shared memory address
0(0H)
96(60H)
192(C0H)
511(1FFH)
Host CPU operation information area
System area
Host CPU refresh area *2
User free area
Write specification prohibited area
Write specification permitted area

- CPU shared memory address of the Universal model QCPU[2]


CPU shared memory address
0
512
2048
4096
10000
24335
Host CPU operation information area
System area
Host CPU refresh area *2
User free area
Not available
Multiple CPU high speed transmission area
Write specification prohibited area
Write specification permitted area

[1] Usable as a user setting area when auto refresh setting is not set. In addition, even when auto refresh setting is set, the auto refresh send range or later is usable as a user setting area.

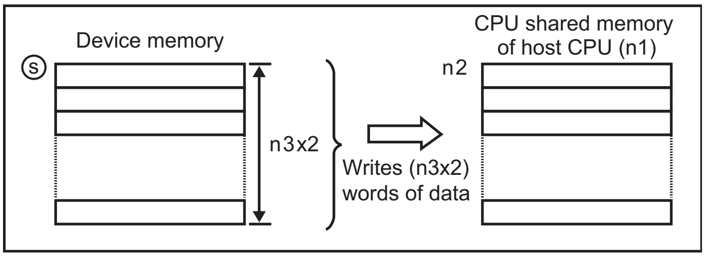
[2] With Q02UCPU, data cannot be written to the multiple CPU high speed transmission area.

- No processing is performed when the number of write points is 0.
- The number of data that can be written varies depending on the target CPU module.

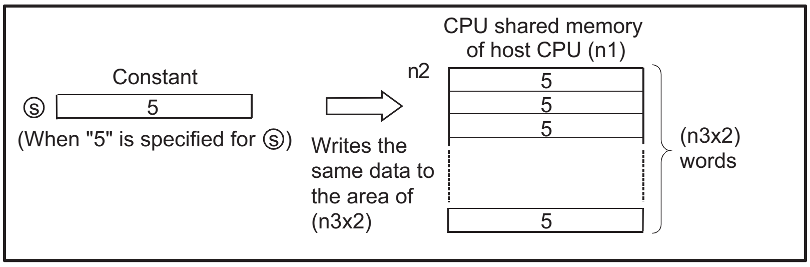
CPU module	Number of write points
Basic model QCPU	1 to 320
Universal model QCPU	1 to 14336

■DTO

- Writes (n3 × 2) words of device data from host CPU module (s) to the CPU shared memory address specified for n2 and the following addresses of host CPU module.


Device memory
Ⓢ
n3x2
Writes (n3x2) words of data
CPU shared memory of host CPU (n1)
n2

When a constant is specified for (s), writes the same data (value specified for (s)) to the area of (n3 × 2) words from the specified CPU shared memory.


Constant
Ⓢ
5
(When "5" is specified for Ⓢ)
Writes the same data to the area of (n3x2)
CPU shared memory of host CPU (n1)
n2
5
5
5
5
(n3x2) words

- No processing is performed when the number of write points is 0.
- The number of data that can be written varies depending on the target CPU module

CPU module	Number of write points
Basic model QCPU	1 to 160
Universal model QCPU	1 to 7168

Point

Data Write to CPU shared memory can be performed using the intelligent function module device.
For the intelligent function module device, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

Operation error

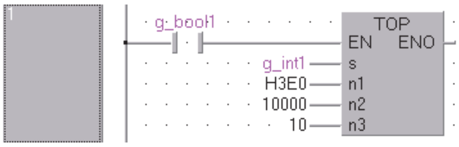
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2110	The CPU module does not exist in the position specified by the start I/O number of the CPU module.	○	—	—	—	○	—
4101	When the number of write points (n3) is outside the specified range of set data. When the CPU shared memory address (n2) of the write destination host CPU + the number of write points (n3) exceeds the CPU shared memory range. When the start number of the devices that stores the data to be written (s) + the number of write points (n3) exceeds the device range. When the value that exceeds the write specification permitted area is specified for the start CPU shared memory address (n2) of the write destination.	○	—	—	—	○	—
4111	When the value for the start of the CPU shared memory address (n2) of the write destination is not valid.	○	—	—	—	○	—
4112	When the I/O number specified for (n1) is other than that of the host station. (Except when the multiple CPU high speed transmission area of the other station is specified.)	○	—	—	—	○	—

Program example

- In the following program, 10 points of data from g_int1 are stored to the address 10000 of the CPU shared memory of CPU No. 1 when g_bool1 turns ON.

[Structured ladder/FBD]

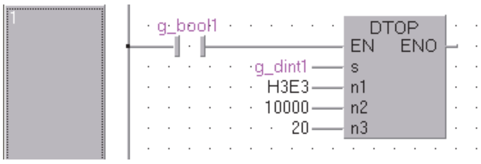


[ST]

TOP(g_bool1, g_int1, H3E0, 10000, 10);

- In the following program, 20 points of data from g_dint1 are stored to the address 10000 or later of the CPU shared memory of CPU No. 4 when g_bool1 turns ON.

[Structured ladder/FBD]



[ST]

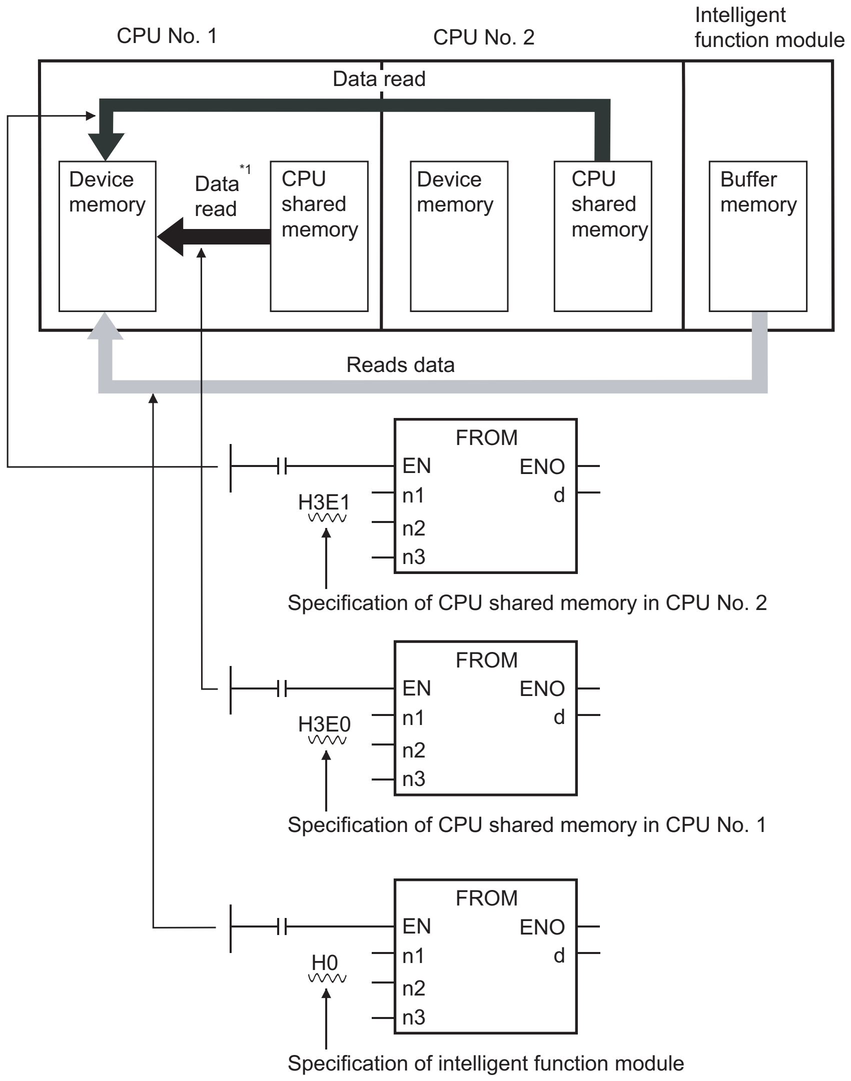
DTOP(g_bool1, g_dint1, H3E3, 10000, 20);

9.2 Reading data from other CPU shared memory

Data can be read using the FROM(P)/DFRO(P) instruction of multiple CPU system from the following memories.

- Buffer memory of the intelligent function module
- CPU shared memory of other CPU module
- CPU shared memory of host CPU module (Executable in the Basic model QCPU and the Universal model QCPU)

The following figure shows the processing performed when the FROM(P) instruction is executed in CPU No. 1.



*1 Can be executed in the Basic model QCPU and the Universal model QCPU.

Point

For reading the buffer memory of the intelligent function module with the FROM(P)/DFRO(P) instruction, refer to the following.

☞ Page 451 Reading 1-/2-word data from intelligent function module

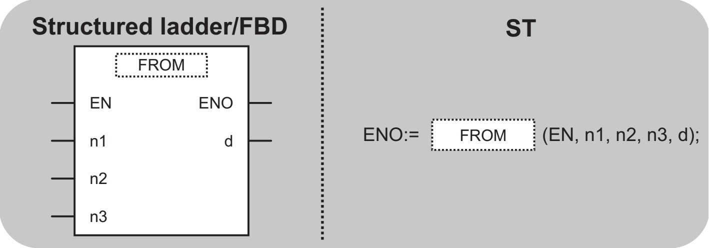
Reading data from other CPU shared memory

FROM(P), DFRO(P)

Ver. Basic | Ver. High performance | Process | Redundant | Universal | LCPU

- Q00CPU and Q01CPU: Supported if first 5 digits of the serial number are "04122" or later
- High Performance model QCPU: Supported if the function version B or later

■For Basic model QCPU and Universal model QCPU

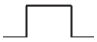
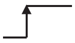


ENO:= FROM (EN, n1, n2, n3, d);

Any of the following instruction can go in the dotted squares.

FROM, FROMP, DFRO, DFROP

■Executing condition

Instruction	Executing condition
FROM, DFRO	
FROMP, DFROP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Start I/O number of the read target CPU module[*1] Basic model QCPU: 3E0H to 3E2H Universal model QCPU: 3E0H to 3E2H	ANY16
	n2	CPU shared memory address of the read destination Basic model QCPU: 0 to 511 Universal model QCPU: 0 to 4095, 10000 to 24335[*2]	ANY16
	n3	Number of data to be read Basic model QCPU: FROM(P): 1 to 512, DFRO(P): 1 to 256 Universal model QCPU: FROM(P): 1 to 14336[*2], DFRO(P): 1 to 7168[*2]	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores read data	ANY16/32

[*1] Specified by the upper 3 digits of start I/O number expressed in 4-digit hexadecimal.

[*2] The setting range varies depending on the auto refresh setting range of the multiple CPU high speed transmission function.

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others (U)
	Bit	Word		Bit	Word				
n1	—	○		○				○	○
n2	—	○		○				○	—
n3	—	○		○				○	—
(d)	—	○		—				—	—

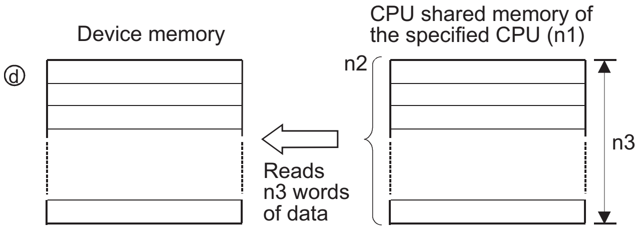
The n1 is specified by the first 3 digits of the hexadecimal 4 digits which represent the start I/O number of the slot on which the CPU module is mounted.

	CPU Slot	Slot 0	Slot 1	Slot 2
Start I/O number	3E00	3E10	3E20	3E30
n1	3E0	3E1	3E2	3E3

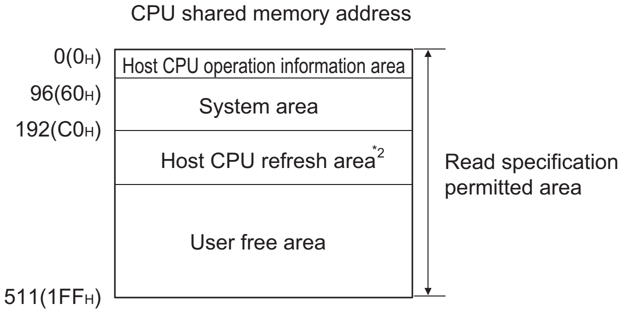
Processing details

■FROM

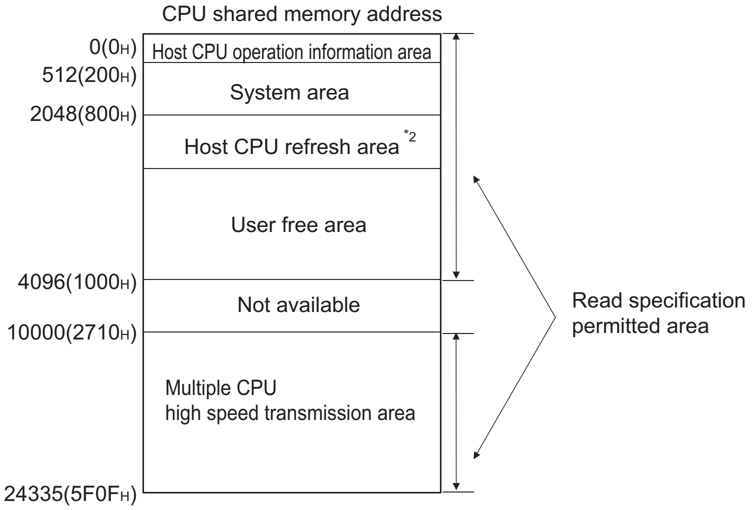
- Reads n3 words of data from the CPU shared memory address specified for n2 of the CPU module specified for n1, and stores the data to (d) and the following devices.



- CPU shared memory address of the Basic model QCPU


CPU shared memory address
0(0H)
96(60H)
192(C0H)
511(1FFH)
Host CPU operation information area
System area
Host CPU refresh area*2
User free area
Read specification permitted area

- CPU shared memory address of the Universal model QCPU*2


CPU shared memory address
0(0H)
512(200H)
2048(800H)
4096(1000H)
10000(2710H)
24335(5F0FH)
Host CPU operation information area
System area
Host CPU refresh area *2
User free area
Not available
Multiple CPU high speed transmission area
Read specification permitted area

[1] Usable as a user setting area when auto refresh setting is not set. When auto refresh setting is set, the auto refresh send range and later are usable as a user setting area.

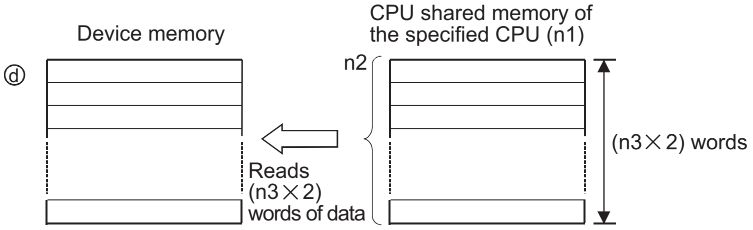
[2] With Q02UCPU, data cannot be read from the multiple CPU high speed transmission area.

- When 0 is specified for n3 as the number of data to be read, no processing is performed.
- The number of data that can be read varies depending on the target CPU module.

CPU module	Number of read points
Basic model QCPU	1 to 512
Universal model QCPU	1 to 14336

■DFRO

- Reads (n3 × 2) words of data from the CPU shared memory address specified for n2 of the CPU module specified for n1, and stores the data to (d) and the following devices.


Device memory
CPU shared memory of the specified CPU (n1)
ⓓ
n2
Reads (n3×2) words of data
(n3×2) words

- No processing is performed when read data n3 is 0.
- The number of data that can be read varies depending on the target CPU module.

CPU module	Number of read points
Basic model QCPU	1 to 256
Universal model QCPU	1 to 7168

Point

Data read from the CPU shared memory can also be performed using the intelligent function module devices. For the intelligent function module device, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

Operation error

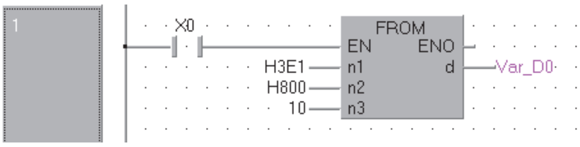
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
2110	The CPU module does not exist in the position specified by the start I/O number of the CPU module.	○	—	—	—	○	—
4101	The head of the CPU shared memory address (n2) which performs reading is outside the CPU shared memory range. The address of the CPU shared memory (n2) from which data are read plus the number of read points (n3) is outside of the CPU shared memory range. The read data storage device number (d) plus the number of read points (n3) is outside of the specified device range. When the start of the CPU shared memory address (n2) which performs reading is an invalid value. (4097 to 9999)	○	—	—	—	○	—

Program example

- In the following program, 10 points of data from the address 800H of the CPU shared memory of CPU No. 2 are stored to Var_D0 and the following devices when X0 turns ON.

[Structured ladder/FBD]

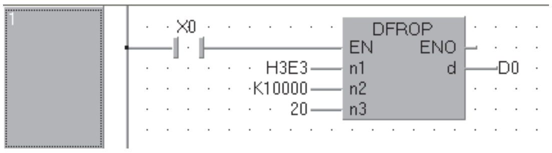


[ST]

FROM(X0,H3E1,H800,10,Var_D0);

- In the following program, 20 points of data from the address 10000 of the CPU shared memory of CPU No. 4 are stored to D0 and the following devices when X0 turns ON.

[Structured ladder/FBD]

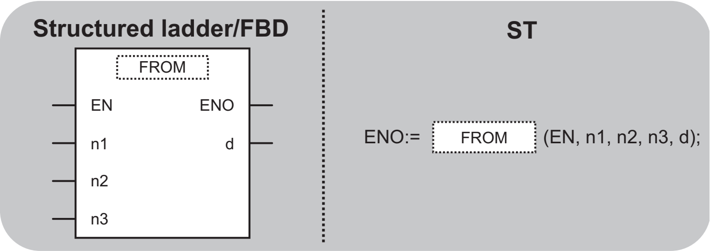


[ST]

DFROP(X0,H3E3,K10000,20,D0);

FROM(P)

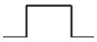
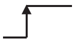
■For High Performance model QCPU and Process CPU



Any of the following instruction can go in the dotted squares.

FROM, FROMP

■Executing condition

Instruction	Executing condition
FROM	
FROMP	

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	n1	Start I/O number of the read target CPU module[*1] 3E0H to 3E3H	ANY16
	n2	CPU shared memory address of the read destination 0 to 4095	ANY16
	n3	Number of data to be read 1 to 4096	ANY16
Output argument	ENO	Execution result	Bit
	d	Start number of the device that stores read data	ANY16

[*1] Specified by the upper 3 digits of start I/O number expressed in 4-digit hexadecimal.

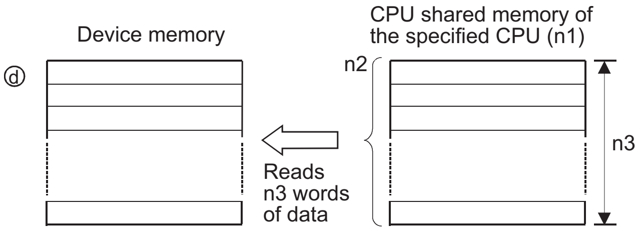
Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others (U)
	Bit	Word		Bit	Word				
n1	—	○		○				○	○
n2	—	○		○				○	—
n3	—	○		○				○	—
(d)	—	○		—				—	—

The n1 is specified by the first 3 digits of the hexadecimal 4 digits which represent the start I/O number of the slot on which the CPU module is mounted.

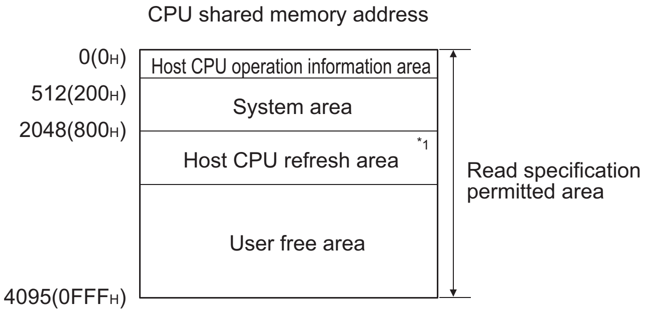
	CPU Slot	Slot 0	Slot 1	Slot 2
Start I/O number	3E00	3E10	3E20	3E30
n1	3E0	3E1	3E2	3E3

Processing details

- Reads n3 words of data from the CPU shared memory address specified for n2 of the CPU module specified for n1, and stores the data to (d) and the following devices.



- CPU shared memory address of the High Performance model QCPU and Process CPU


CPU shared memory address
0(0H)
512(200H)
2048(800H)
4095(0FFFH)
Host CPU operation information area
System area
Host CPU refresh area
*1
User free area
Read specification permitted area

*1 Usable as a user setting area when auto refresh setting is not set. When auto refresh setting is set, the auto refresh send range and later are usable as a user setting area.

- When 0 is specified for n3 as the number of data to be read, no processing is performed.
- The number of data that can be read varies depending on the target CPU module.

CPU module	Number of read points
High Performance model QCPU Process CPU	1 to 4096

Point

Data read from the CPU shared memory can also be performed using the intelligent function module devices. For the intelligent function module device, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

Operation error

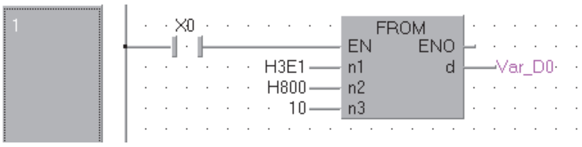
- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/Q00/Q01	QnH	QnPH	QnPRH	QnU	LCPU
2110	The CPU module does not exist in the position specified by the start I/O number of the CPU module.	—	○	○	—	—	—
4101	The head of the CPU shared memory address (n2) which performs reading is outside the CPU shared memory range. The address of the CPU shared memory (n2) from which data are read plus the number of read points (n3) is outside of the CPU shared memory range. The read data storage device number (d) plus the number of read points (n3) is outside of the specified device range. When the start of the CPU shared memory address (n2) which performs reading is an invalid value. (4097 to 9999)	—	○	○	—	—	—

Program example

- In the following program, 10 points of data from the address 800H of the CPU shared memory of CPU No. 2 are stored to Var_D0 and the following devices when X0 turns ON.

[Structured ladder/FBD]



[ST]

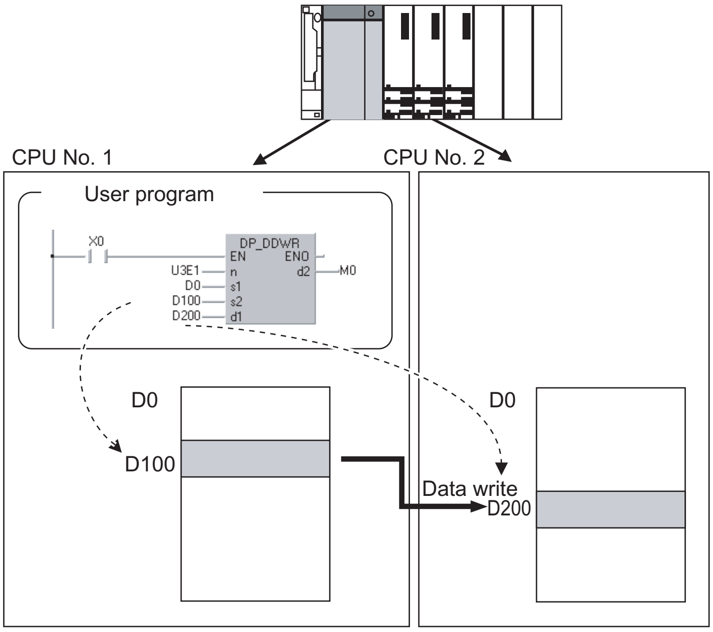
FROM(X0,H3E1,H800,10,Var_D0);

10 MULTIPLE CPU HIGH SPEED TRANSMISSION DEDICATED INSTRUCTIONS

10.1 Overview

The multiple CPU high speed transmission dedicated instruction is an instruction used to read and write device data between the multiple Universal model QCPUs.

The following figure shows the operation when the data are read from the CPU No. 1 and written to the CPU No. 2 using the multiple CPU high speed transmission dedicated instruction.



Point

This instruction can be used for the following CPU modules only, including the ones used as host CPU and other CPU (target CPU for instruction execution).

- Q3UDCPU, Q4UDHCPU or Q06UDHCPU with a serial number whose first five digits are '10012' or higher
- Q10UDHCPU, Q13UDHCPU, Q20UDHCPU, Q26UDHCPU
- QnUDVCPU
- QnUDPVCPU
- QnUDE(H)CPU

System configuration and parameter settings for executing the instruction

This instruction can be executed under the following system configuration and parameter settings.

- QnUD(H)CPU, QnUDVCPU, QnUDPVCPU, or QnUDE(H)CPU is used as the CPU No.1.
- Multiple CPU high speed main base unit (Q3□DB) is used.
- "Multiple CPU High Speed Transmission Area Setting is valid." is selected in the multiple CPU setting of the PLC parameter.

Devices that can be written/read

■Names of devices that can be written/read

The following table shows the devices that can be written to/read from the other Universal model QCPUs using the multiple CPU high transmission dedicated instruction.

Classification	Type	Device name	Target device applicable/not applicable (○: Applicable, △: Applicable with required conditions)	Remarks
Internal user device	Bit device	X, Y, M, L, B, F, SB	△	Required conditions at the setting • Digit specification of bit devices (K4/16 bits). • The start number of bit device is a multiple of 16 (10H).
	Word device	T, ST, C, D, W, SW	○	—
Internal system device	Bit device	SM	△	Required conditions at the setting • Digit specification of bit devices (K4/16 bits). • The start number of bit device is a multiple of 16 (10H).
	Word device	SD	○	—
File register	Word device	R, ZR	○	—

Point

Devices SB, SW, SM, and SD contain the system information area.

When writing data to devices described above using the multiple CPU high speed transmission dedicated instruction D(P)_DDWR, be careful not to destroy the system information.

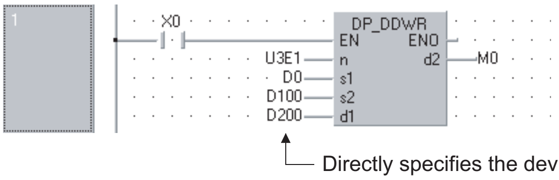
Device specification method and applicable device range for writing/reading data

The device specification and character string specification are the two types of device specification method for other CPUs. The applicable device range for writing/reading data to/from the other CPU is different between the device specification and character string specification.

■Device specification

The device specification is a method for specifying devices of the other CPU directly.

Device specification of the DP_DDWR instruction

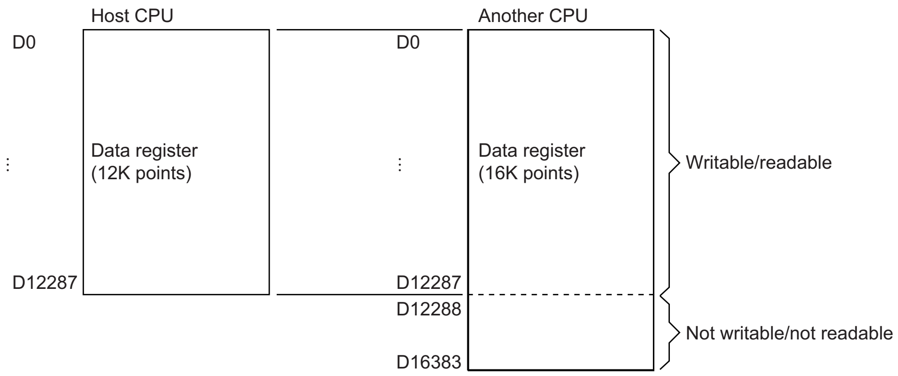


Directly specifies the device 'D200' of the other CPU to be written.

By using the device specification method, data can be written/read within the device range of the host CPU.

For example, when the data register of the host CPU is 12K points, and the data register of other CPU is 16K points, data of 12K points can be written/read from the start of the data register of the other CPU.

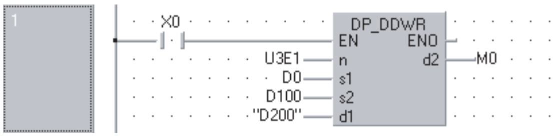
The range for writing/reading data when the device is specified



■Character string specification

The character string specification is a method for specifying devices of the other CPU using the character string.

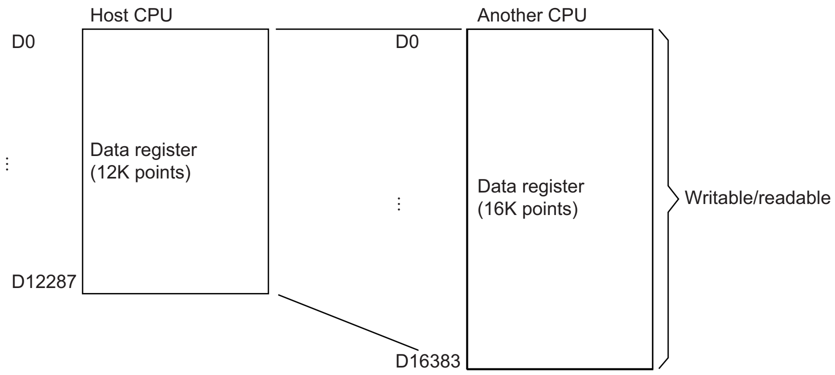
Character string specification of the DP_DDWR instruction



Specifies the device 'D200' of the other CPU to be written using the character string.

By using the character string specification method, data can be written/read within the whole device range of the other CPU. For example, when the data register of the host CPU is 12K points, and the data register of other CPU is 16K points, data of 16K points can be written/read from the start of the data register of the other CPU.

The range for writing/reading data when the character string is specified



Point

The following describes the precautions on the character string specification.

- Up to 32 characters can be specified.
- Any 0s which are appended to upper digits of device number are ignored. The devices with the same lower digits are processed as the same device. For example, 'D1' and 'D0001' are processed as 'D1'.
- Devices specified with uppercase and lowercase of the same letter are processed as a same device. For example, 'D1' and 'd1' are processed as 'D1'.
- If devices that do not exist in other CPUs are specified by the character string, the instruction completes with an error.

Managing the multiple CPU high speed transmission area

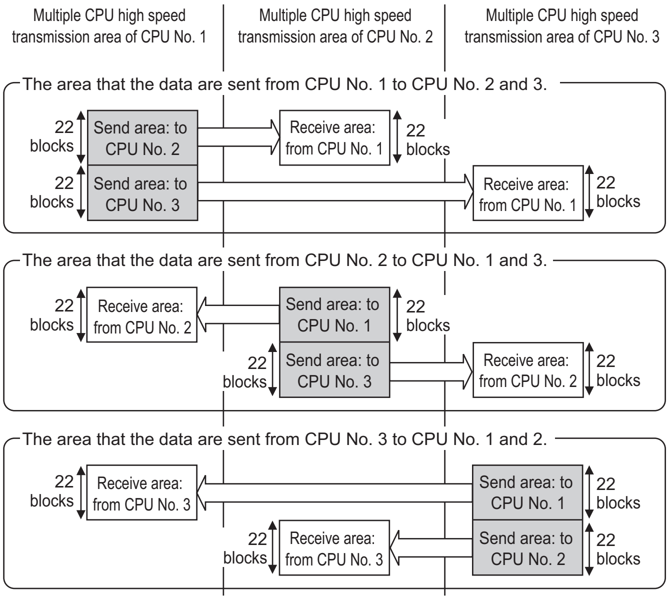
The multiple CPU high speed transmission area is managed in blocks and each block contains minimum of 16 words.

The following table shows the number of blocks that can be used in each CPU and the number of blocks used for the instruction.

Number of CPUs	System area[*1]	
	1K point	2K point
2	46	110
3	22	54
4	14	35

[*1] For details of the system area settings, refer to QCPU User's Manual (Multiple CPU System).

The following figure shows the configuration of the multiple CPU high speed transmission area when the system area size is 1k word in the configuration of multiple CPU system with three CPU modules.


Multiple CPU high speed transmission area of CPU No. 1
Multiple CPU high speed transmission area of CPU No. 2
Multiple CPU high speed transmission area of CPU No. 3
The area that the data are sent from CPU No. 1 to CPU No. 2 and 3.
22 blocks
Send area: to CPU No. 2
Receive area: from CPU No. 1
22 blocks
22 blocks
Send area: to CPU No. 3
Receive area: from CPU No. 1
22 blocks
The area that the data are sent from CPU No. 2 to CPU No. 1 and 3.
22 blocks
Receive area: from CPU No. 2
Send area: to CPU No. 1
22 blocks
22 blocks
Send area: to CPU No. 3
Receive area: from CPU No. 2
22 blocks
The area that the data are sent from CPU No. 3 to CPU No. 1 and 2.
22 blocks
Receive area: from CPU No. 3
Send area: to CPU No. 1
22 blocks
22 blocks
Receive area: from CPU No. 3
Send area: to CPU No. 2
22 blocks

Number of blocks used for the instruction

The number of blocks used for the instruction differs by the points to be written.

The following table shows the number of blocks used for the instruction.

Number of write/read points specified by the instruction	D(P)_DDWR instruction	D(P)_DDRD instruction
1 to 4	1	1
5 to 20	2	
21 to 36	3	
37 to 52	4	
53 to 68	5	
69 to 84	6	
85 to 100	7	

Number of instructions that can be executed simultaneously

For Universal model QCPU, the multiple CPU high speed transmission dedicated instructions can be executed simultaneously within the following range.

$$\left[\text{Number of blocks that can be used for each CPU}\right] \geqq \left[\text{Total of blocks used by the simultaneously executed instructions}\right]$$

By executing the multiple CPU high speed transmission dedicated instruction, if the number of blocks used by the multiple CPU high speed transmission dedicated instruction exceeds the total number of blocks of the multiple CPU high speed transmission area, this instruction is not executed (not processed) in the scan but it is executed again in the next scan. Note that, if the number of empty blocks of the multiple CPU high speed transmission area is fewer than the set value of the special registers SD796 to SD799 (for setting the maximum number of blocks for the multiple CPU high speed transmission dedicated instruction), this instruction completes with an error.

The following table shows the instruction execution applicability by the amount of blocks used for the multiple CPU high speed transmission dedicated instruction, and when the number of empty blocks of multiple CPU high speed transmission area is fewer than the set value of the special registers SD796 to SD799.

Relations between the SD set value and the number of empty blocks	Relation between the number of blocks used by the instruction[1] and the number of empty blocks	
	Number of blocks used by the instruction[1] ≤ Number of empty blocks[2]	Number of blocks used by the instruction[1] > Number of empty blocks[2]
SD set value[3] ≤ Number of empty blocks[2]	Executed	Not executed (not processed)
SD set value[3] > Number of empty blocks[2]	Completes with an error	

*1 Number of blocks used by the multiple CPU high speed transmission dedicated instruction

*2 Number of empty blocks of the multiple CPU high speed transmission area

*3 Set value of the special registers SD796 to SD799

Interlocks when the instruction is used

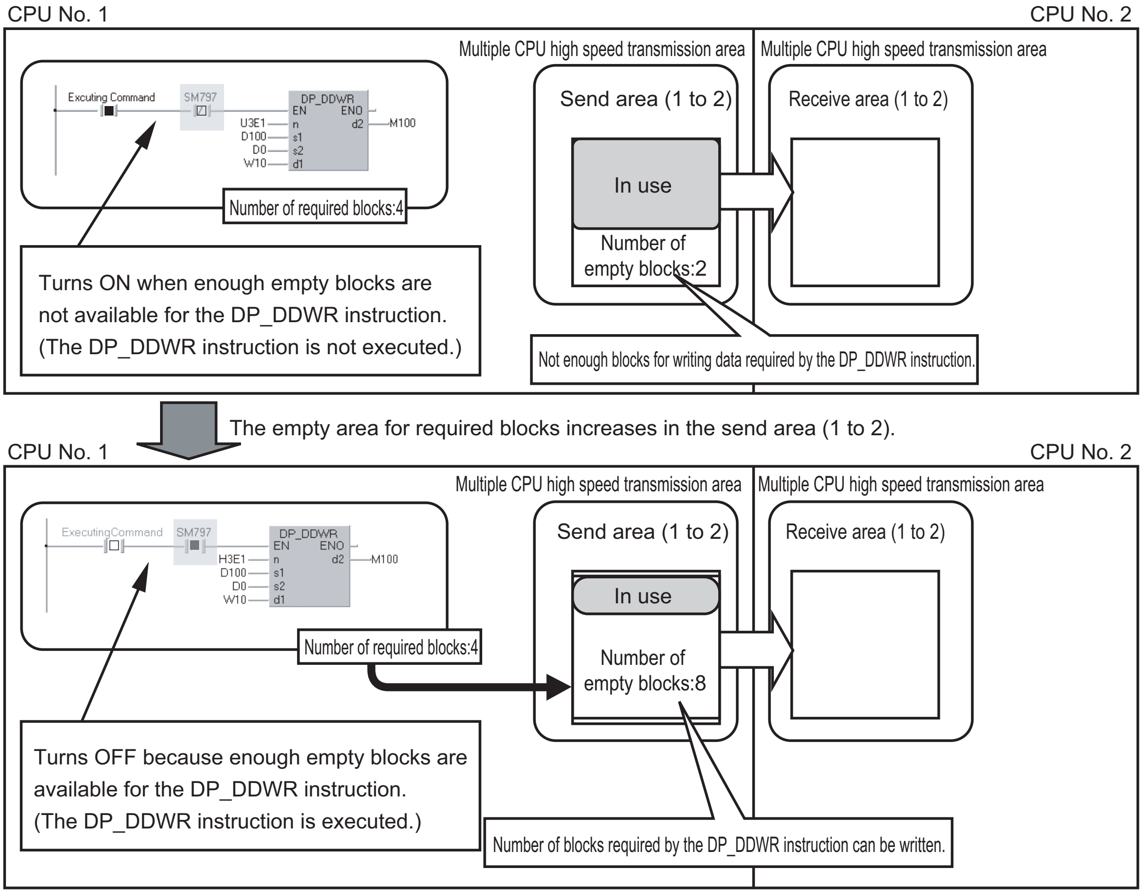
The special relays SM796 to SM799 (information of blocks used for the multiple CPU high speed transmission dedicated instruction) can be used as the interlocks for the multiple CPU high speed transmission dedicated instruction.

When executing two or more multiple CPU high speed transmission dedicated instructions simultaneously, use the special relays SM796 to SM799 as the interlocks for the instruction.

Point

When using the special relays SM796 to SM799, set the maximum number of blocks for the instruction used in each CPU to the special registers SD796 to SD799. (For example, when the maximum number of blocks for the multiple CPU high speed transmission dedicated instruction that is executed to CPU No. 3 is 5, set 5 to SD798.)

If the multiple CPU high speed transmission area becomes less than the number of blocks set in the special registers SD796 to SD799, the corresponding special relay (SM796 to SM799) turns ON.



Turns ON when enough empty blocks are not available for the DP_DDWR instruction. (The DP_DDWR instruction is not executed.)

Not enough blocks for writing data required by the DP_DDWR instruction.

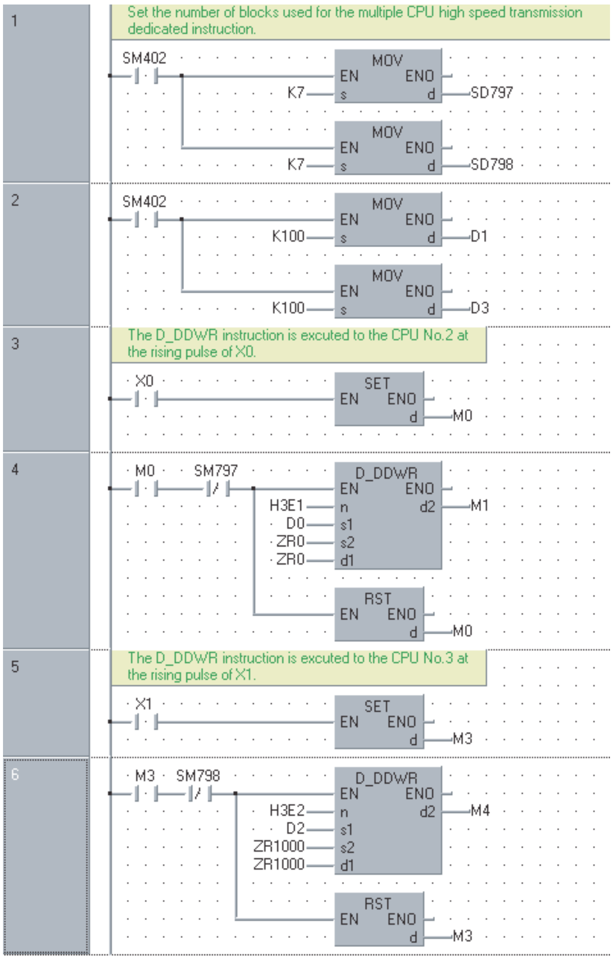
The empty area for required blocks increases in the send area (1 to 2).

Turns OFF because enough empty blocks are available for the DP_DDWR instruction. (The DP_DDWR instruction is executed.)

Number of blocks required by the DP_DDWR instruction can be written.

■Program example when the special relays SM796 to SM799 are used as the interlocks

In the following program, the D_DDWR instruction is executed to the CPU No. 2 at the rising edge of X0, and to the CPU No. 3 at the rising edge of X1.



Set the number of blocks used for the multiple CPU high speed transmission dedicated instruction.

The D_DDWR instruction is excuted to the CPU No.2 at the rising pulse of X0.

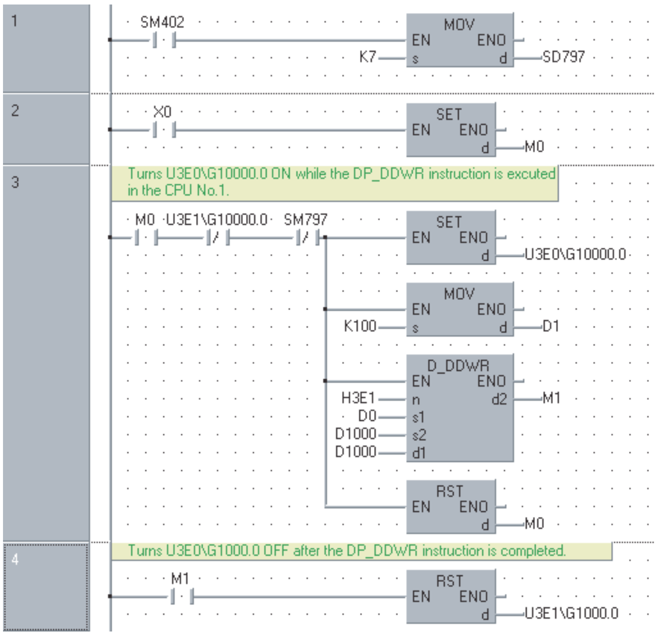
The D_DDWR instruction is excuted to the CPU No.3 at the rising pulse of X1.

■Program example when the multiple CPU high speed transmission dedicated instructions are executed from one CPU to another CPU between the multiple CPU modules

When executing the multiple CPU high speed transmission dedicated instructions from one CPU to another CPU between the Universal model QCPUs, set the interlocks to avoid the instructions to be executed simultaneously. The cyclic transmission area devices (from U3En\G10000) are used as the interlocks.

In the following program, the multiple CPU high speed transmission dedicated instructions are executed from one CPU to another CPU between the CPU No. 1 and CPU No. 2.

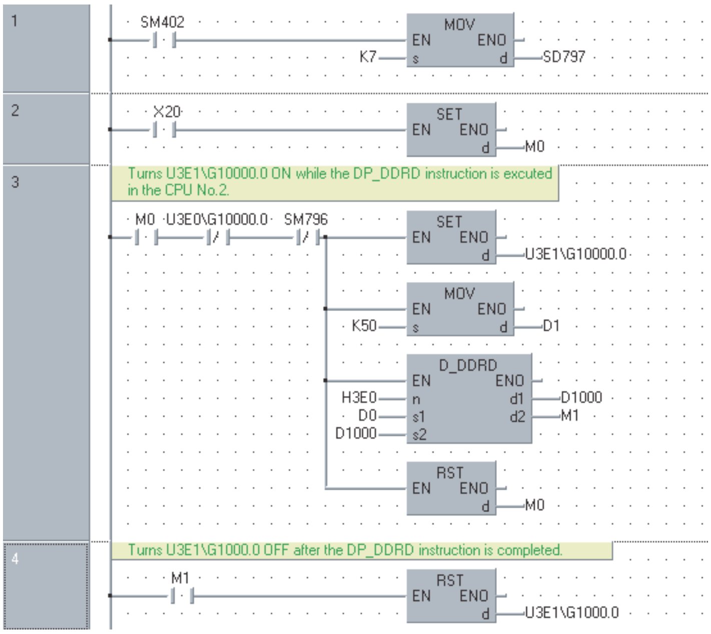
- Program example when the multiple CPU high speed transmission dedicated instruction is executed in the CPU No. 1



Turns U3E0\G10000.0 ON while the DP_DDWR instruction is excuted in the CPU No.1.

Turns U3E0\G1000.0 OFF after the DP_DDWR instruction is completed.

- Program example when the multiple CPU high speed transmission dedicated instruction is executed in the CPU No. 2



Turns U3E1\G10000.0 ON while the DP_DDRD instruction is excuted in the CPU No.2.

Turns U3E1\G1000.0 OFF after the DP_DDRD instruction is completed.

■Program example when data exceeding 100 words are written/read by the multiple CPU high speed transmission dedicated instruction

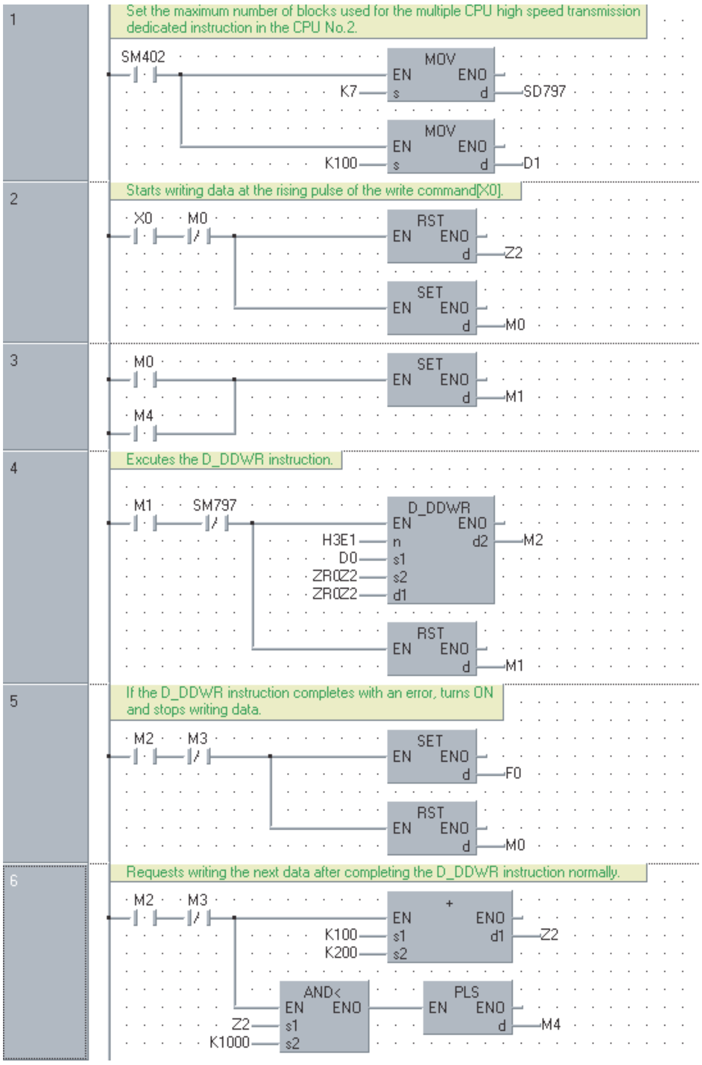
Up to 100 words of data can be used for the multiple CPU high speed transmission dedicated instruction. In order to write/read data exceeding 100 words, execute the instruction multiple times.

Note that the D(P)_DDWR instruction of the multiple CPU high speed transmission dedicated instruction is used in the following program examples, however, the same program configuration can be used for the D(P)_DDRD instruction.

- Program example in which only one D(P)_DDWR instruction is executed

In the following program, data in the devices ZR0 to ZR999 (1000 points) of the CPU No. 1 are written to the devices ZR0 to ZR999 of the CPU No.2 using the D_DDWR instruction. Only one D_DDWR instruction is executed in order to start the next D_DDWR instruction by turning ON the completion device (M2) of the D_DDWR instruction.

- Program example in which only one D(P)_DDWR instruction is executed



1 Set the maximum number of blocks used for the multiple CPU high speed transmission dedicated instruction in the CPU No.2.

2 Starts writing data at the rising pulse of the write command[X0].

4 Excutes the D_DDWR instruction.

5 If the D_DDWR instruction completes with an error, turns ON and stops writing data.

6 Requests writing the next data after completing the D_DDWR instruction normally.

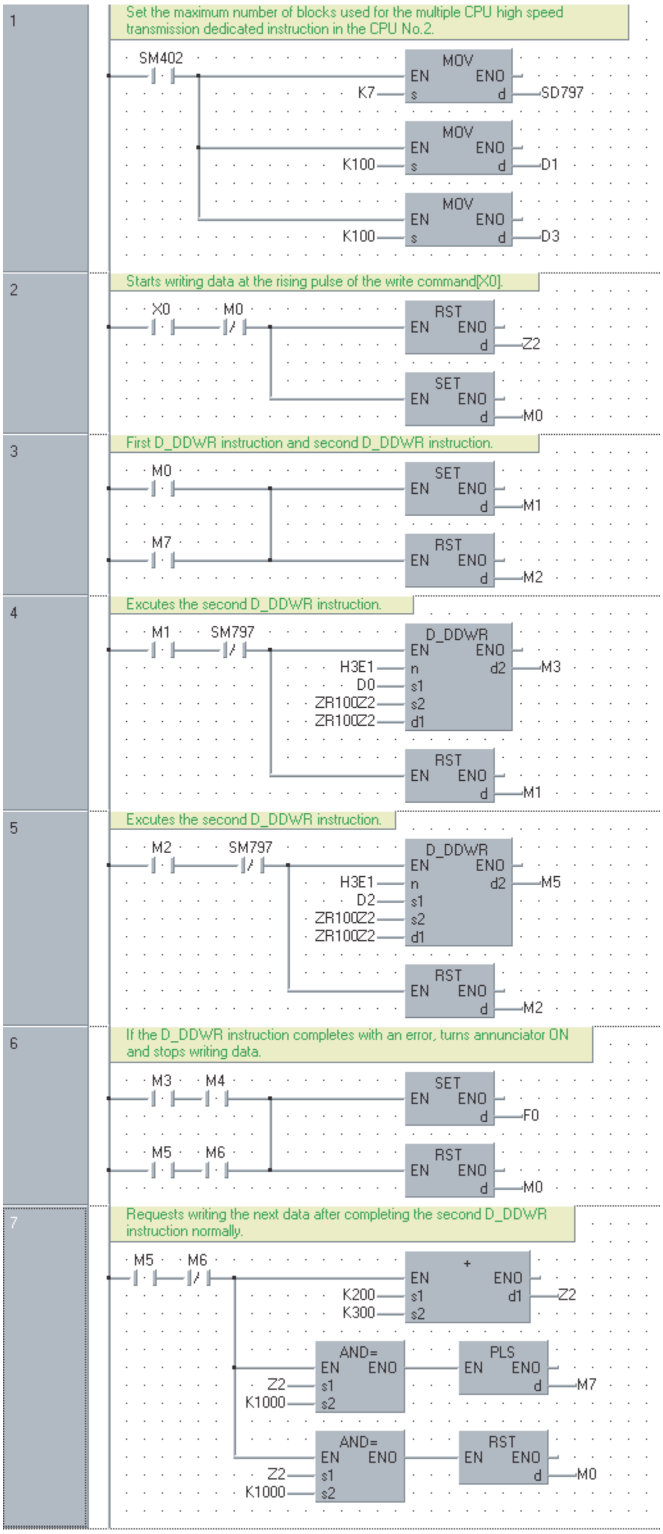
- Program example in which two or more D(P)_DDWR instructions are executed simultaneously

In the following program, data in the devices ZR0 to ZR999 (1000 points) of the CPU No. 1 are written to the devices ZR0 to ZR999 of the CPU No.2 using the D_DDWR instruction.

Two or more instructions to write/read devices between the multiple CPUs can be executed simultaneously as shown in the following program example.

When executing two or more multiple CPU high speed transmission dedicated instructions simultaneously to write/read devices, the higher total number of blocks of the multiple CPU high speed transmission area (send area) shortens the processing time of the instruction.

- Program example in which two or more D(P)_DDWR instructions are executed simultaneously



1 Set the maximum number of blocks used for the multiple CPU high speed transmission dedicated instruction in the CPU No.2.

2 Starts writing data at the rising pulse of the write command[X0].

3 First D_DDWR instruction and second D_DDWR instruction.

4 Excutes the second D_DDWR instruction.

5 Excutes the second D_DDWR instruction.

6 If the D_DDWR instruction completes with an error, turns annunciator ON and stops writing data.

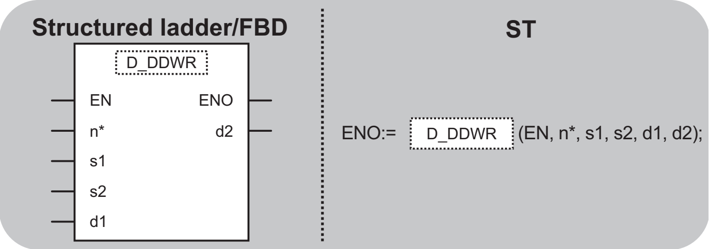
7 Requests writing the next data after completing the second D_DDWR instruction normally.

10.2 Writing Device Data to Other CPUs

D(P)_DDWR

Basic ✕ High performance ✕ Process ✕ Redundant ✕ Ver. Universal LCPU ✕

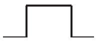
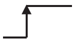
- Universal model QCPU: Supported if first 5 digits of the serial number are "10012" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and Q02UCPU: Not supported



ENO:= D_DDWR (EN, n*, s1, s2, d1, d2);

Any of the following instruction can go in the dotted squares.
D_DDWR, DP_DDWR

■Executing condition

Instruction	Executing condition
D_DDWR	
DP_DDWR	

■Argument

Input/output argument	Name	Description	Data type
Input argument	n	Start I/O number of the other CPU divided by 16 CPU No. 1: 3E0H CPU No. 2: 3E1H CPU No. 3: 3E2H CPU No. 4: 3E3H	ANY16
	s1	Start number or arrays of device in the host CPU that stores control data	Array of ANY16 (0..1)
	s2	Start number of device in the host CPU that stores data to be written	ANY16
	d1	Start number of device in the other CPU that stores data to be written	ANY[*1] String[*2, *3]
Output argument	d2	A bit device that turns on when the process is completed	Array of bit (0..1)

[*1] When file registers (R, ZR) are specified, data that are outside the range in the host CPU can be written to devices of the other CPUs.

[*2] By specifying the start device as a character string " ", data that are outside the range in the host CPU can be written to devices of the other CPUs.

[*3] Device that indexed cannot be specified. (Example: D0Z0, etc.)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word[*7]		Bit	Word				
n[*4]	—	○	○	—				○	—
(s1)[*5]	—	△[*6]	△[*6]	—				—	—
(s2)[*5]	—	○	○	—				—	—
(d1)[*5]	—	○	○	—				—	—
(d2)[*5]	△[*8]	—	△[*6]	—				—	—

[*4] Indexes cannot be set to the setting data n.

[*5] Indexes can be set to the setting data s1 to d2.

[*6] Local devices and file registers per program cannot be used as setting data.

[*7] FD cannot be used.

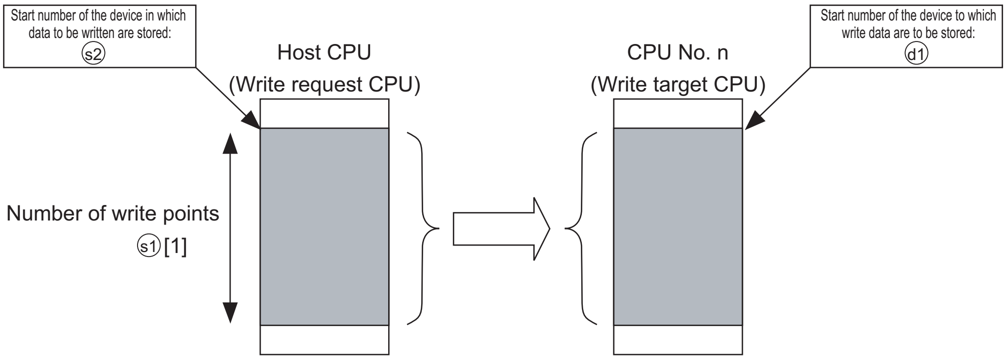
[*8] FX and FY cannot be used.

Setting data

Device	Item	Setting data	Setting range	Setting side
(s1)[0]	Completion status	Execution result of the instruction completion is stored. 0000(H):No error (normal completion) Other than 0000(H):Error code (error completion)	—	System
(s1)[1]	Number of data to be written	Set the number of data to be written in unit of word.	1 to 100	User

Processing details

- Writes data stored in the device (d1) and the following devices specified in the host CPU at the time of multiple CPU system configuration, to the device (s2) and the following devices specified in the CPU No. n for the number of points specified for (s)[1].


Start number of the device in which data to be written are stored: s2
Host CPU
(Write request CPU)
CPU No. n
(Write target CPU)
Start number of the device to which write data are to be stored: d1
Number of write points
s1[1]

- The status of the D(P)_DDWR instruction completion can be checked by the completion device ((d2)[0]) and the completion status display device ((d2)[1]).
  - Completion device ((d2)[0])

    Turns ON at the END process of the scan in which the instruction is completed, and turns OFF at the next END process.
  - Completion status display device ((d2)[1])

    Turns ON/OFF by the status of the instruction completion.

    Normal completion: OFF

    Error completion: Turns ON at the END process of the scan in which the instruction is completed, and turns OFF at the next END process. (The error code is stored to the control data (((s1)[0]): completion status) at the error completion.)
- The number of blocks used for the instruction is specified by the number of data to be written. (☞ Page 846 Overview)

The following table shows the number of blocks used for the instruction.

Write points to be specified in the instruction	D(P)_DDWR instruction
1 to 4	1
5 to 20	2
21 to 36	3
37 to 52	4
53 to 68	5
69 to 84	6
85 to 100	7

- When empty blocks are not available in the multiple CPU high speed transmission area, the instruction completes with an error. This error completion is prevented by setting the number of blocks used for the instruction to the special registers (SD796 to SD799), and using the special relays (SM796 to SM799) as the interlocks. (☞ Page 846 Overview)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4350	When the specified other CPU is wrong, or when the settings are not set correctly to use the multiple CPU high speed transmission instruction. • A reserved CPU is specified. • A CPU number that is not mounted is specified. • The value, start I/O number of the other CPU divided by 16n, is outside the range of 3E0H to 3E3H. • The instruction was executed when the module is set to "Do not use multiple CPU high speed transmission". • The instruction was executed with the CPU module that cannot use this instruction. • The host CPU is specified. • The CPU module where the instruction cannot be executed has been specified.	—	—	—	—	○	—
4351	The instruction is not supported by the other CPU.	—	—	—	—	○	—
4352	The number of devices specified is incorrect.	—	—	—	—	○	—
4353	A device that cannot be used is specified.	—	—	—	—	○	—
4354	A device is specified with the character string that is not applicable.	—	—	—	—	○	—
4355	Number of data to be written ((s1)+1) is outside the range of 0 to 100.	—	—	—	—	○	—

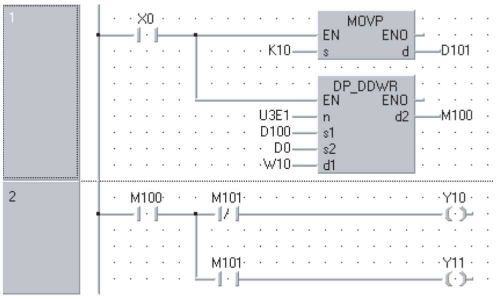
- In any of the following cases, an error completion occurs, and the error code is stored to the device specified for the completion status storing device ((s1)+0).

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
0010H	The amount of instruction requests to a target CPU is exceeding the allowable value. (Empty blocks are not available in the multiple CPU high speed transmission area.)	—	—	—	—	○	—
1001H	The device of the other CPU specified for (d1) is a device that cannot be used for the other CPU, or the device is out of the range.	—	—	—	—	○	—
1003H	The response of the instruction cannot be returned from the other CPU module. (Empty blocks are not available in the multiple CPU high speed transmission area.)	—	—	—	—	○	—
1080H	The number of data to be written set for the D(P)_DDWR instruction is 0.	—	—	—	—	○	—

Program example

- In the following program, 10 words of data from D0 in the host CPU are written to W10 and the following devices in the CPU No. 2 when X0 turns ON.

[Structured ladder/FBD]



The number of write data '10' is stored to the write data points storing device of control data D101 ((s1)[1]).

Data from D0 to D9 in the host CPU are stored to W10 to W19 in the CPU No. 2.

[ST]

```
MOVP(X0,10,D101);
D_DDWR(X0,H3E1,D100,D0,W10,M100);
IF M100=TRUE THEN
    IF M101=FALSE THEN
        Y10:=TRUE;
    ELSE
        Y11:=TRUE;
    END_IF;
END_IF;
```

Precautions

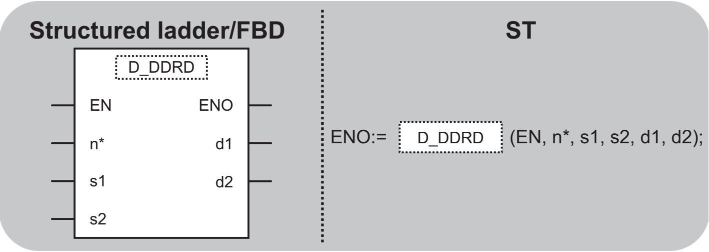
- Digit specification of bit devices can be set for n, (s2) and (d1). However, the following conditions need to be satisfied when the digit specification of bit devices is set for (s2) and (d1).
  - Digit specification of bit devices (K4/16 bits)
  - The start number of bit device is a multiple of 16 (10H).
- Execute this instruction as the write target CPU is powered ON. If the write target CPU is not powered ON and this instruction is executed, the instruction is not processed.
- After the execution of this instruction, data stored by the system (completion status, completion device) cannot be stored normally, if the range of the device specified for the setting data is changed before the completion device turns ON.
- Devices SB, SW, SM, and SD contain the system information area. When writing data to devices described above using the multiple CPU high speed transmission dedicated instruction D(P)_DDWR, be careful not to destroy the system information.

10.3 Reading Device Data from Other CPUs

D(P)_DDRD

Basic High performance Process Redundant Ver. Universal LCPU

- Universal model QCPU: Supported if first 5 digits of the serial number are "10012" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and Q02UCPU: Not supported

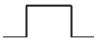
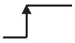


```
ENO:= D_DDRD (EN, n*, s1, s2, d1, d2);
```

Any of the following instruction can go in the dotted squares.

D_DDRD, DP_DDRD

■Executing condition

Instruction	Executing condition
D_DDRD	
DP_DDRD	

■Argument

Input/output argument	Name	Description	Data type
Input argument	n	Start I/O number of the other CPU divided by 16 CPU No. 1: 3E0H, CPU No. 2: 3E1H, CPU No. 3: 3E2H, CPU No. 4: 3E3H	ANY16
	s1	Start number or arrays of device in the host CPU that stores control data	Array of ANY16 (0..1)
	s2	Start number of device in the other CPU that stores data to be read	ANY[1] String[2, 3]
Output argument	d1	Start number of device in the host CPU that stores data to be read	ANY16
	d2	A bit device that turns on when the process is completed	Array of bit (0..1)

*1 When file registers (R, ZR) are specified, data that are outside the range in the host CPU can be written to devices of the other CPUs.

*2 By specifying the start device as a character string " ", data that are outside the range in the host CPU can be read to devices of the other CPUs.

*3 Device that indexed cannot be specified. (Example: D0Z0, etc.)

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word[7]		Bit	Word				
n[4]	—	○	○	—				○	—
(s1) [5]	—	△[6]	△[6]	—				—	—
(s2) [5]	—	○	○	—				—	—
(d1) [5]	—	○	○	—				—	—
(d2) [5]	△[8]	—	△[6]	—				—	—

*4 Indexes cannot be set to the setting data n.

*5 Indexes can be set to the setting data s1 to d2.

*6 Local devices and file registers per program cannot be used as setting data.

*7 FD cannot be used.

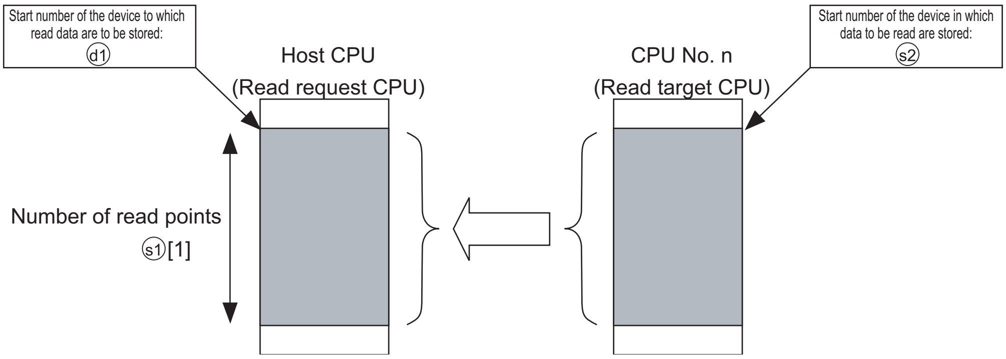
*8 FX and FY cannot be used.

Setting data

Device	Item	Setting data	Setting range	Setting side
(s1)[0]	Completion status	Execution result of the instruction completion is stored. 0000(H):No error (normal completion) Other than 0000(H):Error code (error completion)	—	System
(s1)[1]	Number of data to be read	Set the number of data to be read in unit of word.	1 to 100	User

Processing details

- Stores data read from the device (s2) and the following devices specified in the CPU No. n for the number of points specified for (s1)[1], to the device (d1) and the following devices specified in the host CPU at the time of multiple CPU system configuration.


Start number of the device to which read data are to be stored: ⓓ1
Host CPU
(Read request CPU)
CPU No. n
(Read target CPU)
Start number of the device in which data to be read are stored: ⓢ2
Number of read points
ⓢ1[1]

- The status of the D(P)_DDRD instruction completion can be checked by the completion device ((d2)[0]) and the completion status display device ((d2)[1]).
  - Completion device ((d2)[0])
    Turns ON at the END process of the scan in which the instruction is completed, and turns OFF at the next END process.
  - Completion status display device ((d2)[1])
    Turns ON/OFF by the status of the instruction completion.
    Normal completion: OFF
    Error completion: Turns ON at the END process of the scan in which the instruction is completed, and turns OFF at the next END process. (The error code is stored to the control data (((s1)[0]): completion status) at the error completion.)
- The number of blocks used for the instruction is specified by the number of data to be read. (☞ Page 846 Overview)

The following table shows the number of blocks used for the instruction.

Read points to be specified in the instruction	D(P)_DDRD instruction
1 to 100	1

- When empty blocks are not available in the multiple CPU high speed transmission area, the instruction completes with an error. This error completion is prevented by setting the number of blocks used for the instruction to the special registers (SD796 to SD799), and using the special relays (SM796 to SM799) as the interlocks. (☞ Page 846 Overview)

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4350	When the specified other CPU is wrong, or when the settings are not set correctly to use the multiple CPU high speed transmission instruction. • A reserved CPU is specified. • A CPU number that is not mounted is specified. • The value, start I/O number of the other CPU divided by 16n, is outside the range of 3E0H to 3E3H. • The instruction was executed when the module is set to "Do not use multiple CPU high speed transmission". • The instruction was executed with the CPU module that cannot use this instruction. • The host CPU is specified. • The CPU where the instruction cannot be executed has been specified.	—	—	—	—	○	—
4351	The instruction is not supported by the other CPU.	—	—	—	—	○	—
4352	The number of devices specified is incorrect.	—	—	—	—	○	—
4353	A device that cannot be used is specified.	—	—	—	—	○	—
4354	A device is specified with the character string that is not applicable.	—	—	—	—	○	—
4355	Number of data to be read ((s1)+1) is outside the range of 0 to 100.	—	—	—	—	○	—

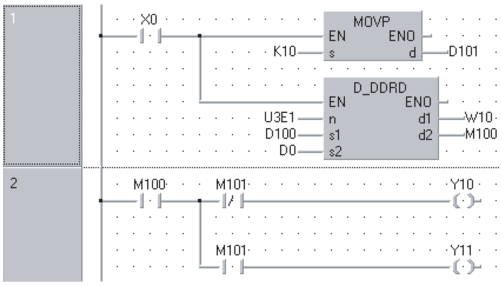
- In any of the following cases, an error completion occurs, and the error code is stored to the device specified for the completion status storing device ((s1)+0).

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
0010H	The amount of instruction requests to a target CPU is exceeding the allowable value. (Empty blocks are not available in the multiple CPU high speed transmission area.)	—	—	—	—	○	—
1001H	The device of the other CPU specified for (s2) is a device that cannot be used for the other CPU, or the device is out of the range.	—	—	—	—	○	—
1003H	The response of the instruction cannot be returned from the other CPU module. (Empty blocks are not available in the multiple CPU high speed transmission area.)	—	—	—	—	○	—
1081H	The number of data to be read set for the D(P)_DDRD instruction is 0.	—	—	—	—	○	—

Program example

- In the following program, 10 words of data from D0 in the CPU No. 2 are read and stored to W10 and the following devices in the host CPU when X0 turns ON.

[Structured ladder/FBD]



The number of read data '10' is stored to the read data points storing device of control data D101 (ⓢ1[1]).

Data from D0 to D9 in the CPU No. 2 are stored to W10 to W19 in the host CPU.

[ST]

```
MOVP(X0,10,D101);
D_DDRD(X0,H3E1,D100,D0,W10,M100);
IF M100=TRUE THEN
    IF M101=FALSE THEN
       Y10:=TRUE;
    ELSE
       Y11:=TRUE;
    END_IF;
END_IF;
```

Precautions

- Digit specification of bit devices can be set for n, (s2) and (d1). However, the following conditions need to be satisfied when the digit specification of bit devices is set for (s2) and (d1).
    - Digit specification of bit devices (K4/16 bits)
    - The start number of bit device is a multiple of 16 (10H).
- Execute this instruction as the read target CPU is powered ON. If the read target CPU is not powered ON and this instruction is executed, the instruction is not processed.
- After the execution of this instruction, data stored by the system (completion status, completion device) cannot be stored normally, if the range of the device specified for the setting data is changed before the completion device turns ON.

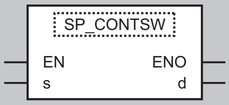
11 REDUNDANT SYSTEM INSTRUCTIONS (FOR REDUNDANT CPU)

11.1 System Switching

SP_CONTSW

Basic ✗ High performance ✗ Process ✗ Redundant Universal ✗ LCPU ✗

Structured ladder/FBD



ST

ENO:= SP_CONTSW (EN,s,d);

The following instruction can go in the dotted squares.

SP_CONTSW

■Executing condition

Instruction	Executing condition
SP_CONTSW	↑

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition	Bit
	s	Value other than 0 and used to identify the processing that issued the system switching request	ANY16
Output argument	ENO	Execution result	Bit
	d	Error completion device number	Bit

Setting data	Internal device		R, ZR	J□\□		U□\G□	Zn	Constant K, H	Others
	Bit	Word		Bit	Word				
(s)	—	○		—				○	—
(d)	○	○[*1]		—				—	—

*1 The bit specification for the word device is available.

Processing details

- Switches between the control system and standby system at the END processing of the scan executed with the SP_CONTSW instruction.
- When using the SP_CONTSW instruction for system switching, the "manual switching enable flag (SM1592)" must have been turned on (enabled) in advance.
- (s) is provided to identify the processing block of the program where system switching occurred when multiple SP_CONTSW instructions are used.
- At (s), specify a value within the ranges -32768 to -1 and 1 to 32767 (1_H to $FFFF_H$).
- The (s) value specified by the SP_CONTSW instruction is stored into the "system switching instruction argument (SD6)" of the error common information when the system switching is normally completed.[*2] When multiple SP_CONTSW instructions are executed during the same scan, the argument of the SP_CONTSW instruction executed first is stored into the system switching instruction argument (SD6).

[*2] The value (s) specified for the SP_CONTSW instruction can be confirmed in the error common information of the PLC diagnostics dialog box on GX Works2.

- The (s) value specified by the SP_CONTSW instruction is stored into the "system switching instruction argument (SD1602)" of the new control system CPU module when system switching is normally completed.[*3] By reading the SD1602 value from the new control system CPU module, which the SP_CONTSW instruction was used for system switching can be confirmed.

[*3] The new control system CPU module means the CPU module that was switched from the standby system to the control system by the SP_CONTSW instruction.

- The error completion device is turned on by the control system CPU module when system switching by the SP_CONTSW instruction was unsuccessful.
- When OPERATION ERROR is detected due to any of the following reasons at the execution of the SP_CONTSW instruction, the error completion device is turned on during the instruction execution.
  - 0 is specified at (s) of the executed SP_CONTSW instruction.
  - The "manual switching enable flag (SM1592)" is off.
  - The SP_CONTSW instruction was executed by the standby system in the separate mode.
  - The SP_CONTSW instruction was executed in the debug mode.
- If systems could not be switched due to any of the reasons given in the following table, the error completion device turns on when system switching is executed in the END processing.

Reason No.	Reason for System Switching Failure
0	Normally completed
1	Tracking cable is disconnected or faulty.
2	Hardware fault, power-off, reset or watchdog timer error occurred in the standby system.
3	Watchdog timer error occurred in the control system.
4	Preparations being made for tracking transfer.
5	Communication time-out.
6	Stop error occurred in the standby system. (Excluding watchdog timer error)
7	Operating status different between the control system and standby system.
8	Memory copy being executed from the control system to the standby system.
9	Write during RUN being executed.
10	Network fault detected by the standby system.

When the error completion device was turned on due to unsuccessful system switching, 16 is stored into the "reason(s) for system switching (SD1588)" and the reason No. of the above table into the "reason(s) for system switching failure (SD1589)".

- Use a user program or GX Works2 to turn off the error completion bit that has turned on. If normal system switching is performed by the execution of the SP_CONTSW instruction with the error completion device on, the error completion device of the new standby system CPU module is also turned off. When system switching is performed due to a factor other than the SP_CONTSW instruction, however, the error completion device is not turned off.

Operation error

- In any of the following cases, an operation error occurs, the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4110	The value specified in (s) is 0 at execution of the SP_CONTSW instruction.	—	—	—	○	—	—
4120	The manual switching enable flag (SM1592) is off (disabled) at the execution of the SP_CONTSW instruction.	—	—	—	○	—	—
4121	The SP_CONTSW instruction was executed by the standby system CPU module in the separate mode. The SP_CONTSW instruction was executed in the debug mode.	—	—	—	○	—	—

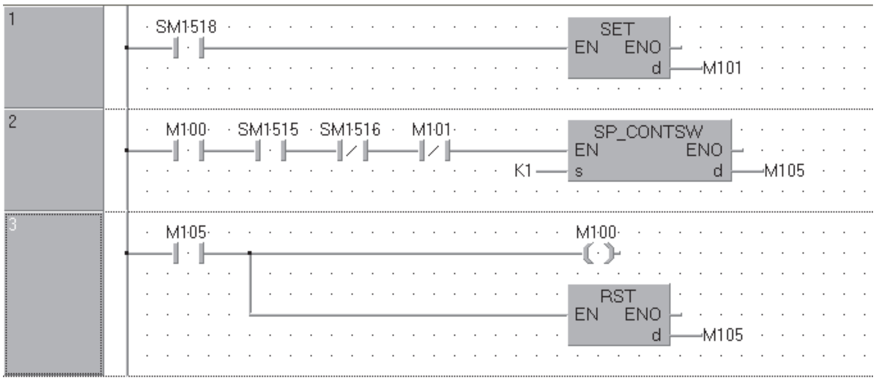
If system switching was unsuccessful, the error flag (SM0) is turned on and an error code is stored into SD0.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
6220	The tracking cable is disconnected or faulty. Hardware fault, power-off, reset or watchdog timer error occurred in the standby system. Watchdog timer error occurred in the control system. Preparations are being made for tracking transfer. Communication time-out occurred. A stop error, excluding watchdog timer error, occurred in the standby system. The operating status differs between the control system and standby system. Memory copy is being executed from the control system to the standby system. Write during RUN is being executed. Network fault was detected by the standby system.	—	—	—	○	—	—

Program example

- The following program executes system switching on the leading edge of the system switching command (M100). If the system switching command (M100) remains on, the SP_CONTSW instruction is also executed by the new control system CPU module after system switching. Therefore, M101 is added to the execution conditions as a consecutive switching prevention flag.

[Structured ladder/FBD]



[ST]

```
SET(LDP( TRUE , SM1518 ) , M101 );
IF M100 AND SM1515 AND NOT(1516) AND NOT(M101) THEN
    SP_CONTSW( TRUE , K1 , M105 );
END_IF;
OUT( M105 , M100 );
RST( M105 , M105 );
```

MEMO

APPENDIX

Appendix 1 Added/Changed Instructions with Version Upgrade of GX Works2

The following table shows instructions added/changed with version upgrade of GX Works2.

Version	Instruction name	Addition/change	Reference
Version 1.15R	LDP	• Symbolic description is added for structured ladder.	Page 86 Edge operation start, edge series connection, edge parallel connection
	LDF		
	ANDP		
	ANDF		
	ORP		
	ORF		
	LDPI	• Added for structured ladder and ST.	Page 89 Negated edge operation start, negated edge series connection, negated edge pulse parallel connection
	LDFI		
	ANDPI		
	ANDFI		
	ORPI		
	ORFI		
Version 1.24A	UMSG	• Added for structured ladder and ST.	Page 796 User message
Version 1.91V	S_REFDVWRB	• Added for structured ladder and ST.	Page 808 Refresh device write (in 1-bit units)
	SP_REFDVWRB		
	S_REFDVWRW		Page 813 Refresh device write (in 16-bit units)
	SP_REFDVWRW		
	S_REFDVRDB		Page 818 Refresh device read (in 1-bit units)
	SP_REFDVRDB		
	S_REFDVRDW		Page 823 Refresh device read (in 16-bit units)
	SP_REFDVRDW		
	SP_CONTSW	• Added for ST.	Page 863 System Switching
Version 1.98C	SP_CONTSW	• Added for structured ladder	Page 863 System Switching

MEMO

INDEX

INSTRUCTION INDEX

REVISIONS

The manual number is written at the bottom left of the back cover.

Revision date	Manual number	Revision
Jul., 2008 ⋮ Sep., 2013	SH(NA)-080783ENG-A ⋮ SH(NA)-080783ENG-N	Due to the transition to the e-Manual, the details of revision have been deleted.
Dec., 2016	SH(NA)-080783ENG-O	Complete revision (layout change)
September 2018	SH(NA)-080783ENG-P	Descriptions regarding the QnUDPVCPU is added.

Japanese manual version SH-080736-T

This manual confers no industrial property rights or any rights of any other kind, nor does it confer any patent licenses. Mitsubishi Electric Corporation cannot be held responsible for any problems involving industrial property rights which may occur as a result of using the contents noted in this manual.

WARRANTY

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

Microsoft, Microsoft Access, Excel, SQL Server, Visual Basic, Visual C++, Visual Studio, Windows, Windows NT, Windows Server, Windows Vista, and Windows XP are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Celeron, Intel, and Pentium are either registered trademarks or trademarks of Intel Corporation in the United States and/or other countries.

Ethernet is a registered trademark of Fuji Xerox Co., Ltd. in Japan.

The SD and SDHC logos are trademarks of SD-3C, LLC.

The company names, system names and product names mentioned in this manual are either registered trademarks or trademarks of their respective companies.

SH(NA)-080783ENG-P(1809)KWIX
MODEL: Q-KP-KM-E
MODEL CODE: 13JW07

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

When exported from Japan, this manual does not require application to the Ministry of Economy, Trade and Industry for service transaction permission.

Specifications subject to change without notice.