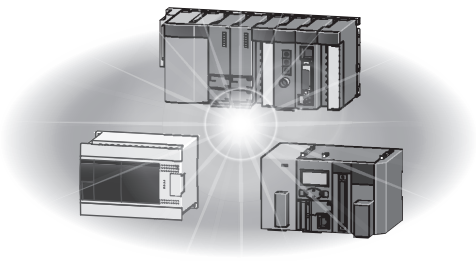
MITSUBISHI ELECTRIC

Programmable Controller

MELSEC Q series MELSEC L series

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



SAFETY PRECAUTIONS

(Read these precautions before using this product.)

Before using MELSEC-Q or -L series programmable controllers, please read the manuals included with each product and the relevant manuals introduced in those manuals carefully, and pay full attention to safety to handle the product correctly.

Make sure that the end users read the manuals included with each product, and keep the manuals in a safe place for future reference.

CONDITIONS OF USE FOR THE PRODUCT

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

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

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

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

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

("Prohibited Application")

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

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

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

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

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 (Common Instructions) <SH-080783ENG>	Specifications and functions of common instructions, such as sequence instructions, basic instructions, and application instructions, 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 IEC61131-3. (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
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
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
Personal computer	A generic term for the personal computers where 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
Redundant CPU	A generic term for the Q12PRHCPU and Q25PRHCPU
Special instruction	A generic term for the module dedicated instructions, PID control instructions, socket communication function instructions, built-in I/O function instructions, and data logging function instructions
Universal model QCPU	A generic term for the Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, Q03UDCPU, Q03UDVCPU, Q03UDECPU, Q04UDHCPU, Q04UDVCPU, Q04UDEHCPU, Q06UDHCPU, Q06UDVCPU, Q06UDEHCPU, Q10UDHCPU, Q10UDEHCPU, Q13UDHCPU, Q13UDVCPU, Q13UDEHCPU, Q20UDHCPU, Q20UDEHCPU, Q26UDHCPU, Q26UDVCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU

1 OVERVIEW

1.1 Purpose of This Manual

This manual explains the application functions 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 following.

☞ Page 8 RELEVANT MANUALS

Operation of GX Works2

Purpose		Overview	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 11 Details of instructions in each programming language

Purpose		OVERVIEW	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/ FBD	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

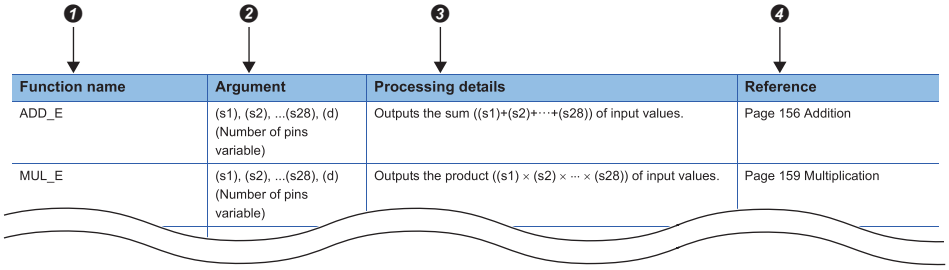
Details of instructions in each programming language

Purpose		Overview	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 FUNCTION TABLES

2.1 How to Read Function Tables

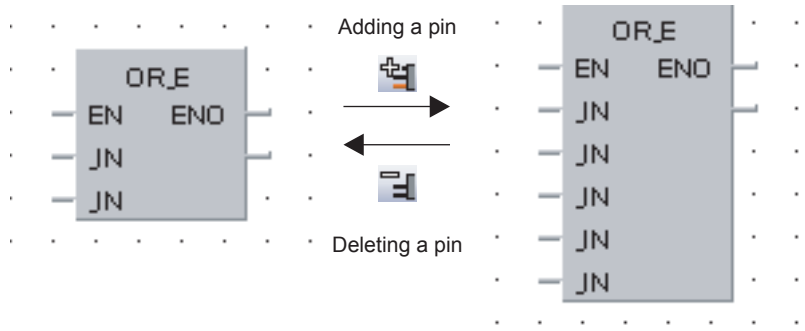
Function tables in Page 13 Function Tables are shown in the following format.



Description

❶···Indicates the functions used in a program. 'Function name(_E)' is used as a function with EN/ENO. Without "_E", it is used as a a function without EN/ENO.

❷···Indicates the arguments of the function.

Symbol	Name	Description
(s)	Source	Stores data before operation.
(d)	Destination	Indicates the destination of data after operation.
(Number of pins variable)		Allows the number of (s) (source) to be changed in the range from 2 to 28. • Changing the number of pins 

❸···Indicates the processing details of each function.

❹···Indicates the references on which the functions are explained.

2.2 Function Tables

Type conversion functions

Function name	Argument	Processing details	Reference
BOOL_TO_INT(_E)	(s), (d)	Converts bit type data into word (signed) or double word (signed) type data.	Page 24 Converting bit type to word (signed), double word (signed) type
BOOL_TO_DINT(_E)	(s), (d)		
BOOL_TO_STR(_E)	(s), (d)	Converts bit type data into string type data.	Page 26 Converting bit type to string type
BOOL_TO_WORD(_E)	(s), (d)	Converts bit type data into word (unsigned)/16-bit string or double word (unsigned)/32-bit string type date.	Page 28 Converting bit type to word (unsigned)/16-bit string, double word (unsigned)/32-bit string type
BOOL_TO_DWORD(_E)	(s), (d)		
BOOL_TO_TIME(_E)	(s), (d)	Converts bit type data into time type data.	Page 30 Converting bit type to time type
INT_TO_DINT(_E)	(s), (d)	Converts word (signed) type data into double word (signed) type data.	Page 32 Converting word (signed) type to double word (signed) type
DINT_TO_INT(_E)	(s), (d)	Converts double word (signed) type data into word (signed) type data.	Page 34 Converting double word (signed) type to word (signed) type
INT_TO_BOOL(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into bit type data.	Page 36 Converting word (signed), double word (signed) type to bit type
DINT_TO_BOOL(_E)	(s), (d)		
INT_TO_REAL(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into single-precision real type data.	Page 38 Converting word (signed), double word (signed) type to single-precision real type
DINT_TO_REAL(_E)	(s), (d)		
INT_TO_LREAL(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into double-precision real type data.	Page 40 Converting word (signed), double word (signed) type to double-precision real type
DINT_TO_LREAL(_E)	(s), (d)		
INT_TO_STR(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into string type data.	Page 42 Converting word (signed), double word (signed) type to string type
DINT_TO_STR(_E)	(s), (d)		
INT_TO_WORD(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into word (unsigned)/16-bit string type data.	Page 45 Converting word (signed), double word (signed) type to word (unsigned)/16-bit string type
DINT_TO_WORD(_E)	(s), (d)		
INT_TO_DWORD(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into double word (unsigned)/32-bit string type data.	Page 47 Converting word (signed), double word (signed) type to double word (unsigned)/32-bit string type
DINT_TO_DWORD(_E)	(s), (d)		
INT_TO_BCD(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into BCD type data.	Page 49 Converting word (signed), double word (signed) type to BCD type
DINT_TO_BCD(_E)	(s), (d)		
INT_TO_TIME(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into time type data.	Page 52 Converting word (signed), double word (signed) type to time type
DINT_TO_TIME(_E)	(s), (d)		
REAL_TO_INT(_E)	(s), (d)	Converts single-precision real type data into word (signed) or double word (signed) type data.	Page 54 Converting single-precision real type to word (signed), double word (signed) type
REAL_TO_DINT(_E)	(s), (d)		
LREAL_TO_INT(_E)	(s), (d)	Converts double-precision real type data into word (signed) or double word (signed) type data.	Page 56 Converting double-precision real type to word (signed), double word (signed) type
LREAL_TO_DINT(_E)	(s), (d)		
REAL_TO_LREAL(_E)	(s), (d)	Converts single-precision real type data into double-precision real type data.	Page 58 Converting single-precision real type to double-precision real type

Function name	Argument	Processing details	Reference
LREAL_TO_REAL(_E)	(s), (d)	Converts double-precision real type data into single-precision real type data.	Page 60 Converting double-precision real type to single-precision real type
REAL_TO_STR(_E)	(s), (d)	Converts single-precision real type data into string type (exponential form) data.	Page 62 Converting single-precision real type to string type
WORD_TO_BOOL(_E)	(s), (d)	Converts word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data into bit type data.	Page 65 Converting word (unsigned)/16-bit string, double word (unsigned)/32-bit string type to bit type
DWORD_TO_BOOL(_E)	(s), (d)		
WORD_TO_INT(_E)	(s), (d)	Converts word (unsigned)/16-bit string type data into word (signed) or double word (signed) type data.	Page 67 Converting word (unsigned)/16-bit string type to word (signed), double word (signed) type
WORD_TO_DINT(_E)	(s), (d)		
DWORD_TO_INT(_E)	(s), (d)	Converts double word (unsigned)/32-bit string type data into word (signed) or double word (signed) type data.	Page 69 Converting double word (unsigned)/32-bit string type to word (signed), double word (signed) type
DWORD_TO_DINT(_E)	(s), (d)		
WORD_TO_DWORD(_E)	(s), (d)	Converts word (unsigned)/16-bit string type data into double word (unsigned)/32-bit string type data.	Page 71 Converting word (unsigned)/16-bit string type to double word (unsigned)/32-bit string type
DWORD_TO_WORD(_E)	(s), (d)	Converts double word (unsigned)/32-bit string type data into word (unsigned)/16-bit string type data.	Page 73 Converting double word (unsigned)/32-bit string type to word (unsigned)/16-bit string type
WORD_TO_STR(_E)	(s), (d)	Converts word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data into string type data.	Page 75 Converting word (unsigned)/16-bit string, double word (unsigned)/32-bit string type to string type
DWORD_TO_STR(_E)	(s), (d)		
WORD_TO_TIME(_E)	(s), (d)	Converts word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data into time type data.	Page 77 Converting word (unsigned)/16-bit string, double word (unsigned)/32-bit string type to time type
DWORD_TO_TIME(_E)	(s), (d)		
STR_TO_BOOL(_E)	(s), (d)	Converts string type data into bit type data.	Page 79 Converting string type to bit type
STR_TO_INT(_E)	(s), (d)	Converts string type data into word (signed) or double word (signed) type data.	Page 81 Converting string type to word (signed), double word (signed) type
STR_TO_DINT(_E)	(s), (d)		
STR_TO_REAL(_E)	(s), (d)	Converts string type data into single-precision real type data.	Page 84 Converting string type to single-precision real type
STR_TO_WORD(_E)	(s), (d)	Converts string type data into word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data.	Page 88 Converting string type to word (unsigned)/16-bit string, double word (unsigned)/32-bit string type
STR_TO_DWORD(_E)	(s), (d)		
STR_TO_TIME(_E)	(s), (d)	Converts string type data into time type data.	Page 91 Converting string type to time type
STR_TO_BCD(_E)	(s), (d)	Converts string type data into BCD type data.	Page 93 Converting string type to BCD type
BCD_TO_INT(_E)	(s), (d)	Converts BCD type data into word (signed) or double word (signed) type data.	Page 96 Converting BCD type to word (signed), double word (signed) type
BCD_TO_DINT(_E)	(s), (d)		
BCD_TO_STR(_E)	(s), (d)	Converts BCD type data into string type data.	Page 99 Converting BCD type to string type
TIME_TO_BOOL(_E)	(s), (d)	Converts time type data into bit type data.	Page 101 Converting time type to bit type
TIME_TO_INT(_E)	(s), (d)	Converts time type data into word (signed) or double word (signed) type data.	Page 103 Converting time type to word (signed), double word (signed) type
TIME_TO_DINT(_E)	(s), (d)		
TIME_TO_STR(_E)	(s), (d)	Converts time type data into string type data.	Page 105 Converting time type to string type
TIME_TO_WORD(_E)	(s), (d)	Converts time type data into word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data.	Page 107 Converting time type to word (unsigned)/16-bit string, double word (unsigned)/32-bit string type
TIME_TO_DWORD(_E)	(s), (d)		

Function name	Argument	Processing details	Reference
BITARR_TO_INT(_E)	(s), n, (d)	Converts specified number of bits from bit array into word (signed) type, word (unsigned)/16-bit string type, double word (signed) type, or double word (unsigned)/32-bit string type data.	Page 109 Converting bit array to word (signed) type, word (unsigned)/16-bit string type, double word (signed) type, double word (unsigned)/32-bit string type
BITARR_TO_DINT(_E)	(s), n, (d)		
INT_TO_BITARR(_E)	(s), n, (d)	Outputs low-order n bits of word (signed) type, word (unsigned)/16-bit string type, double word (signed) type, or double word (unsigned)/32-bit string type data.	Page 111 Converting word (signed) type, word (unsigned)/16-bit string type, double word (signed) type, double word (unsigned)/32-bit string type to bit array
DINT_TO_BITARR(_E)	(s), n, (d)		
CPY_BITARR(_E)	(s), n, (d)	Copies specified number of bits from bit array.	Page 113 Bit array copy
GET_BIT_OF_INT(_E)	(s), n, (d)	Reads a value of specified bit of word (signed) type data.	Page 115 Specified bit read of word (signed) type data
SET_BIT_OF_INT(_E)	(s), n, (d)	Writes a value to the specified bit of word (signed) type data.	Page 117 Specified bit write of word (signed) type data
CPY_BIT_OF_INT(_E)	(s), n1, n2, (d)	Copies a specified bit of word (signed) type data to the specified bit of another word (signed) type data.	Page 119 Specified bit copy of word (signed) type data
GET_BOOL_ADDR	(s), (d)	Converts the type of data into bit type.	Page 121 Nonessential type conversion
GET_INT_ADDR	(s), (d)	Converts the type of data into word (signed) type.	
GET_WORD_ADDR	(s), (d)	Converts the type of data to word (unsigned)/16-bit string type.	

Standard functions of one numeric variable

Function name	Argument	Processing details	Reference
ABS(_E)	(s), (d)	Outputs the absolute value of an input value.	Page 123 Absolute value

Standard arithmetic functions

Function name	Argument	Processing details	Reference
ADD_E	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the sum ((s1)+(s2)+ ··· +(s28)) of input values.	Page 126 Addition
MUL_E	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the product ((s1) × (s2) × ··· × (s28)) of input values.	Page 128 Multiplication
SUB_E	(s1), (s2), (d)	Outputs the difference ((s1)-(s2)) between input values.	Page 130 Subtraction
DIV_E	(s1), (s2), (d)	Outputs the quotient ((s1) ÷ (s2)) of input values.	Page 132 Division
MOD(_E)	(s1), (s2), (d)	Outputs the remainder after division of input values ((s1) ÷ (s2)).	Page 134 Remainder
EXPT(_E)	(s1), (s2), (d)	Outputs the exponentiation of an input value.	Page 136 Exponentiation
MOVE(_E)	(s), (d)	Moves the input value to (d).	Page 138 Move operation

Standard bitwise Boolean functions

Function name	Argument	Processing details	Reference
AND_E	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the Boolean AND of input values.	Page 140 Boolean AND, boolean OR, boolean exclusive OR, and boolean NOT
OR_E	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the Boolean OR of input values.	
XOR_E	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the Boolean exclusive OR of input values.	
NOT(_E)	(s), (d)	Outputs the Boolean NOT of input values.	

Standard selection functions

Function name	Argument	Processing details	Reference
SEL(_E)	(s1), (s2), (s3), (d)	Outputs the value selected from the input values.	Page 144 Selection
MAXIMUM(_E)	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the maximum value of the input values.	Page 146 Maximum/Minimum selection
MINIMUM(_E)	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the minimum value of the input values.	
LIMITATION(_E)	(s1), (s2), (s3), (d)	Outputs the input value controlled by the upper and lower limit control.	Page 148 Upper/Lower limit control
MUX(_E)	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs one of the multiple input values.	Page 150 Multiplexer

Standard comparison functions

Function name	Argument	Processing details	Reference
GT_E	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the comparison value of an input value.	Page 152 Comparison
GE_E	(s1), (s2), ...(s28), (d) (Number of pins variable)		
EQ_E	(s1), (s2), ...(s28), (d) (Number of pins variable)		
LE_E	(s1), (s2), ...(s28), (d) (Number of pins variable)		
LT_E	(s1), (s2), ...(s28), (d) (Number of pins variable)		
NE_E	(s1), (s2), (d)		

Standard character string functions

Function name	Argument	Processing details	Reference
MID(_E)	(s), n1, n2, (d)	Outputs the specified number of characters, extracted from the specified start position in the input character string.	Page 154 Extract mid string
CONCAT(_E)	(s1), (s2), ...(s28), (d) (Number of pins variable)	Concatenates the character strings and outputs the operation result.	Page 156 String concatenation
INSERT(_E)	(s1), (s2), n, (d)	Inserts a character string between other character strings and outputs the operation result.	Page 158 String insertion
DELETE(_E)	(s), n1, n2, (d)	Deletes the specified range in a character string and outputs the operation result.	Page 160 String deletion
REPLACE(_E)	(s1), (s2), n1, n2, (d)	Replaces the specified range in a character string with the specified character string and outputs the operation result.	Page 162 String replacement

Functions of time data types

Function name	Argument	Processing details	Reference
ADD_TIME(_E)	(s1), (s2), (d)	Outputs the sum ((s1)+(s2)) of the input values (time type).	Page 165 Addition
SUB_TIME(_E)	(s1), (s2), (d)	Outputs the difference ((s1)-(s2)) of input values (time type).	Page 167 Subtraction
MUL_TIME(_E)	(s1), (s2), (d)	Outputs the product ((s1) × (s2)) of input values (time type).	Page 169 Multiplication
DIV_TIME(_E)	(s1), (s2), (d)	Outputs the quotient ((s1) ÷ (s2)) of input values (time type).	Page 171 Division

Standard bistable function blocks

Function name	Argument	Processing details	Reference
SR(_E)	(s1), (s2), (d)	Discriminates two input values and outputs 1 (TRUE) or 0 (FALSE). (Set-dominant)	Page 173 Standard bistable function blocks (Set-dominant)
RS(_E)	(s1), (s2), (d)	Discriminates two input values and outputs 1 (TRUE) or 0 (FALSE). (Reset-dominant)	Page 175 Standard bistable function blocks (Reset-dominant)

Standard edge detection function blocks

Function name	Argument	Processing details	Reference
R_TRIG(_E)	(s), (d)	Detects the rising edge of a signal and outputs pulse signals.	Page 177 Rising edge detector
F_TRIG(_E)	(s), (d)	Detects the falling edge of a signal and outputs pulse signals.	Page 179 Falling edge detector

Standard counter function blocks

Function name	Argument	Processing details	Reference
CTU(_E)	(s1), (s2), n, (d1), (d2)	Counts the number of times that the signal turns ON.	Page 181 Up counter
CTD(_E)	(s1), (s2), n, (d1), (d2)	Counts down the number of times that the signal turns ON.	Page 183 Down counter
CTUD(_E)	(s1), (s2), (s3), (s4), n, (d1), (d2), (d3)	Counts/counts down the number of times that the signal turns ON.	Page 185 Up/Down counter
COUNTER_FB_M	(s1), (s2), (s3), (d1), (d2)	Counts the number of times that the signal turns ON from (s3) to (s2).	Page 189 Counter function blocks

Standard timer function blocks

Function name	Argument	Processing details	Reference
TP(_E) TP_HIGH(_E)	(s), n, (d1), (d2)	Holds the signal ON for the specified time.	Page 191 Pulse timer
TON(_E) TON_HIGH(_E)	(s), n, (d1), (d2)	Turns ON the signal after the specified time.	Page 194 On delay timer
TOF(_E) TOF_HIGH(_E)	(s), n, (d1), (d2)	Turns OFF the signal after the specified time.	Page 197 Off delay timer
TIMER_10_FB_M TIMER_100_FB_M TIMER_HIGH_FB_M TIMER_LOW_FB_M TIMER_CONT_FB_M TIMER_CONTHFB_M	(s1), (s2), (s3), (d1), (d2)	Turns ON the signal after the specified time counted from input value (s3) to (s2).	Page 200 Timer function blocks

Point

The function and function block of the application functions execute the operation with the combination of multiple sequence instructions. Therefore, if the interrupt occurs in the application function operations, unintended operation results may occur.
For using an interrupt program, use Disable interrupt/ Enable interrupt (DI/EI instruction) as necessary.

2.3 Operator Tables

Arithmetic operations

Operator name		Argument	Processing details	Reference
Structured ladder/ FBD	ST			
ADD	+	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the sum ((s1)+(s2)+ ··· +(s28)) of input values.	Page 204 Addition
MUL	*	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the product ((s1) × (s2) × ··· × (s28)) of input values.	Page 206 Multiplication
SUB	-	(s1), (s2), (d)	Outputs the difference ((s1)-(s2)) between input values.	Page 207 Subtraction
DIV	/	(s1), (s2), (d)	Outputs the quotient ((s1) ÷ (s2)) of input values.	Page 208 Division
(Not supported)	MOD	(s1), (s2), (d)	Outputs the remainder after division of input values ((s1) ÷ (s2)).	Page 210 Remainder
(Not supported)	**	(s1), (s2), (d)	Outputs the exponentiation of an input value.	Page 211 Exponentiation

Logical operations

Operator name		Argument	Processing details	Reference
Structured ladder/ FBD	ST			
AND	& AND	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the Boolean AND of input values.	Page 212 Boolean AND, boolean OR, boolean exclusive OR, and boolean NOT
OR	OR	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the Boolean OR of input values.	
XOR	XOR	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the Boolean exclusive OR of input values.	
(Not supported)	NOT	(s), (d)	Outputs the Boolean NOT of input values.	

Comparison operations

Operator name		Argument	Processing details	Reference
Structured ladder/ FBD	ST			
GT	>	(s1), (s2), ...(s28), (d) (Number of pins variable)	Outputs the comparison value of an input value.	Page 214 Comparison
GE	>=	(s1), (s2), ...(s28), (d) (Number of pins variable)		
EQ	=	(s1), (s2), ...(s28), (d) (Number of pins variable)		
LE	<=	(s1), (s2), ...(s28), (d) (Number of pins variable)		
LT	<	(s1), (s2), ...(s28), (d) (Number of pins variable)		
NE	<>	(s1), (s2), (d)		

3 CONFIGURATION OF FUNCTIONS

3.1 Configuration of Functions

Instructions available in the CPU module can be divided into a function name and an argument.

The application of a function name and an argument are as follows:

- Function name ··· Indicates the function.
- Argument ··· Indicates the I/O data used in the function.

Arguments are classified into source data, destination data, executing condition, and execution result.

Source (s)

A source is data used in an operation.

The following source types are available depending on the device specified in a function:

Device	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 modification 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 function can be changed.

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

Destination (d)

Data after the operation are stored to a destination.

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

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

Executing condition (EN)

An input variable EN inputs an executing condition of a function.

Execution result (ENO)

An output variable ENO outputs an execution result.

Point

For details of the configuration of functions for labels and structures, refer to the following.

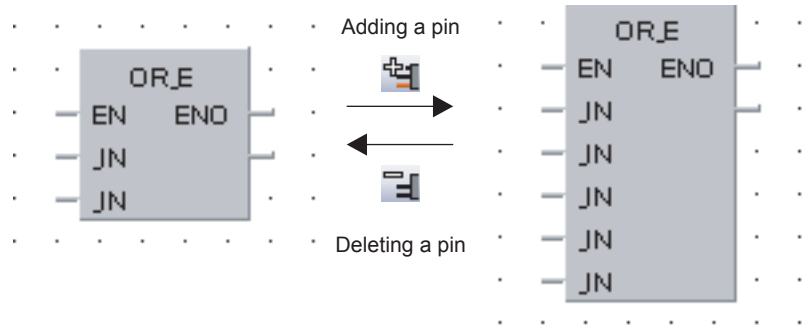
MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

3.2 Input Pins Variable Function

Some functions allow the number of input pins to be changed. To change the number of input pins, select the target function and change the number.

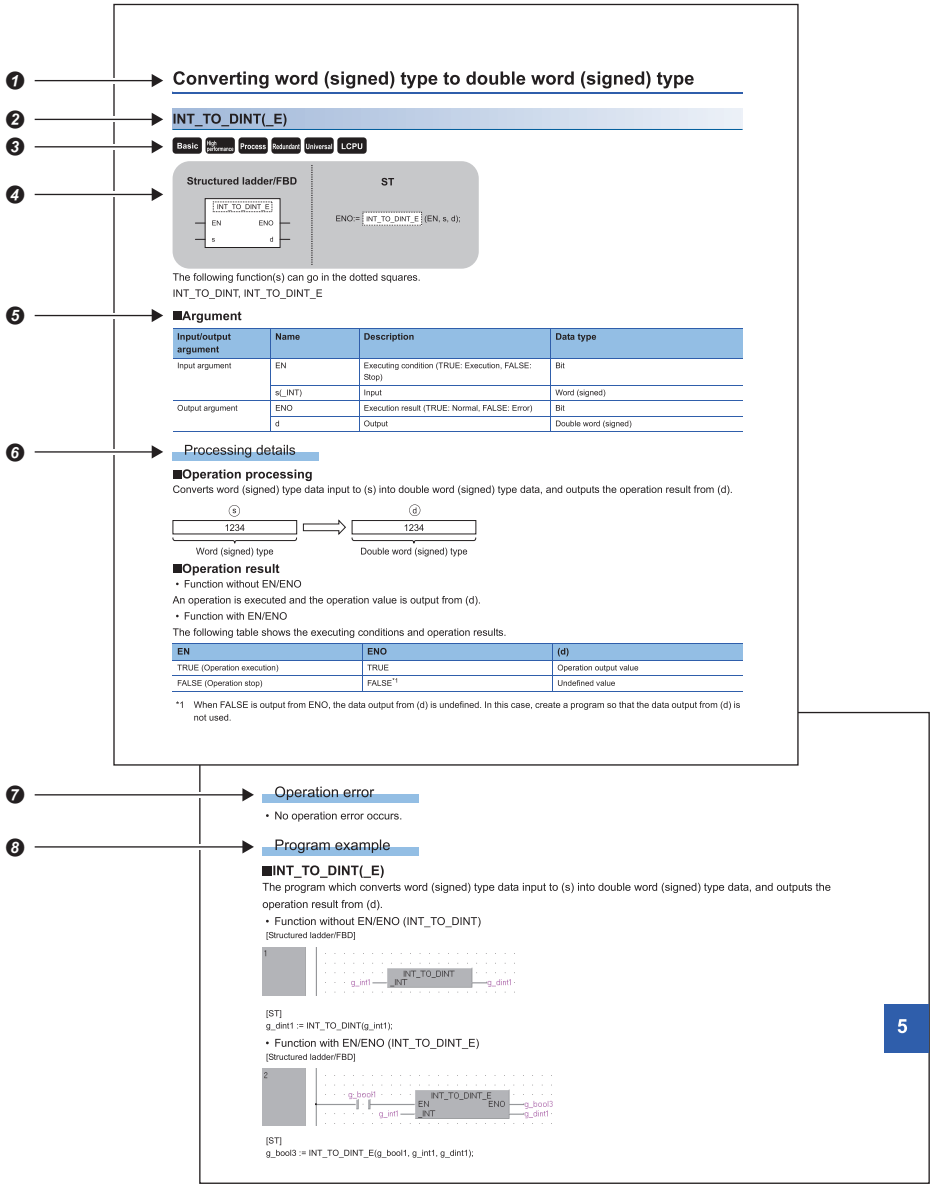
For details, refer to the following.

GX Works2 Version 1 Operating Manual (Structured Project)



4 HOW TO READ FUNCTIONS

Chapter 5 and after provides detailed explanation on each function in the layout as shown below.



❶ Indicates an outline of a function.

❷ Indicates a function to be explained.

❸ Indicates the CPU modules that can use the function.

Icon						Description
Basic model QCPU	High Performance model QCPU	Process CPU	Redundant CPU	Universal model QCPU	LCPU	
						The basic icon indicates that the CPU module can use the corresponding function.
						The icon with a Ver. symbol indicates that the CPU module can use the corresponding function under certain restrictions (function version and software version).
						The icon with × indicates that the CPU module cannot use the corresponding function.

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

❺ Indicates 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)

❻ Indicates the processing performed by the function.

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

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

5 APPLICATION FUNCTIONS

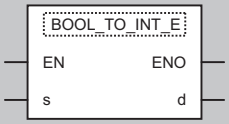
5.1 Type Conversion Functions

Converting bit type to word (signed), double word (signed) type

BOOL_TO_INT(_E), BOOL_TO_DINT(_E)

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

Structured ladder/FBD



ST

```
ENO:= BOOL_TO_INT_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
BOOL_TO_INT, BOOL_TO_INT_E, BOOL_TO_DINT, BOOL_TO_DINT_E

■Argument

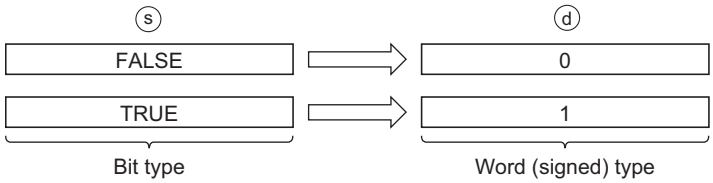
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_BOOL)	Input	Bit
Output argument	ENO	Execution result (TRUE: Normal execution, FALSE: Error or stop)	Bit
	d	Output	Word (signed), double word (signed)

Processing details

■Operation processing

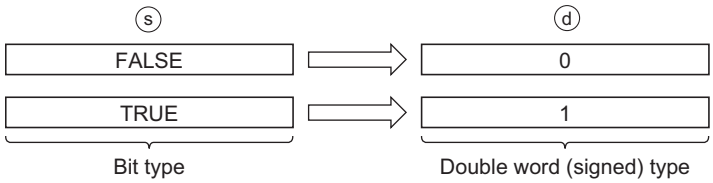
• BOOL_TO_INT, BOOL_TO_INT_E

Converts bit type data input to (s) into word (signed) type data, and outputs the operation result from (d).
When the input value is FALSE, 0 is output in word (signed) type data.
When the input value is TRUE, 1 is output in word (signed) type data.



• BOOL_TO_DINT, BOOL_TO_DINT_E

Converts bit type data input to (s) into double word (signed) type data, and outputs the operation result from (d).
When the input value is FALSE, 0 is output in double word (signed) type data.
When the input value is TRUE, 1 is output in double word (signed) type data.



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

Program example

■BOOL_TO_INT(_E)

The program which converts bit type data input to (s) into word (signed) type data, and outputs the operation result from (d).

• Function without EN/ENO (BOOL_TO_INT)

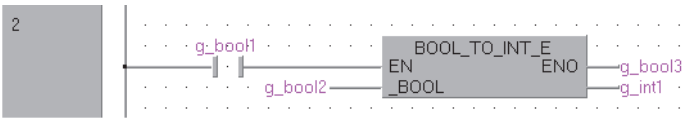
[Structured ladder/FBD]

[ST]

```
g_int1:=BOOL_TO_INT(g_bool1);
```

• Function with EN/ENO (BOOL_TO_INT_E)

[Structured ladder/FBD]



[ST]

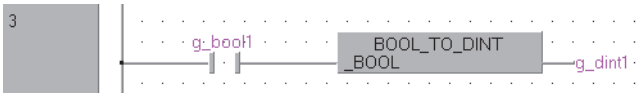
```
g_bool3 := BOOL_TO_INT_E(g_bool1, g_bool2, g_int1);
```

■BOOL_TO_DINT(_E)

The program which converts bit type data input to (s) into double word (signed) type data, and outputs the operation result from (d).

• Function without EN/ENO (BOOL_TO_DINT)

[Structured ladder/FBD]



[ST]

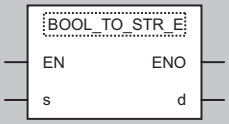
```
g_dint1:=BOOL_TO_DINT(g_bool1);
```

Converting bit type to string type

BOOL_TO_STR(_E)

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

Structured ladder/FBD



ST

```
ENO:= BOOL_TO_STR_E(EN, s, d);
```

The following function(s) can go in the dotted squares.
BOOL_TO_STR, BOOL_TO_STR_E

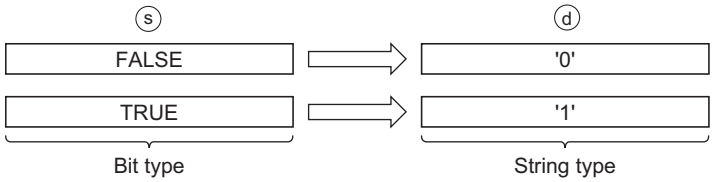
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_BOOL)	Input	Bit
Output argument	ENO	Execution result (TRUE: Normal execution, FALSE: Error or stop)	Bit
	d	Output	String

Processing details

■Operation processing

Converts bit type data input to (s) into string type data, and outputs the operation result from (d).
When the input value is FALSE, 0 is output in string type data.
When the input value is TRUE, 1 is output in string type data.



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

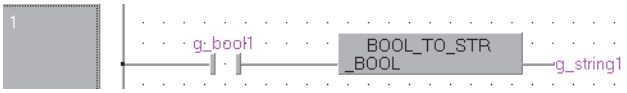
Program example

■BOOL_TO_STR(_E)

The program which converts bit type data input to (s) into string type data, and outputs the operation result from (d).

- Function without EN/ENO (BOOL_TO_STR)

[Structured ladder/FBD]



[ST]

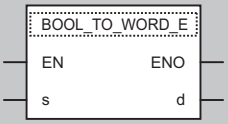
```
g_string1 := BOOL_TO_STR(g_bool1);
```

Converting bit type to word (unsigned)/16-bit string, double word (unsigned)/32-bit string type

BOOL_TO_WORD(_E), BOOL_TO_DWORD(_E)

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

Structured ladder/FBD



ST

```
ENO:= BOOL_TO_WORD_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
BOOL_TO_WORD, BOOL_TO_WORD_E, BOOL_TO_DWORD, BOOL_TO_DWORD_E

■Argument

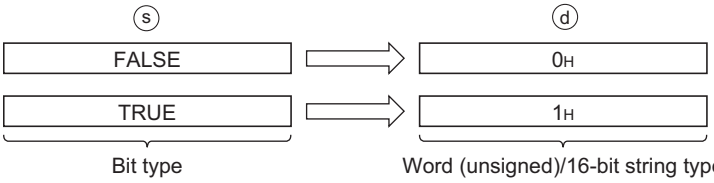
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_BOOL)	Input	Bit
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (unsigned)/16-bit string, double word (unsigned)/32-bit string

Processing details

■Operation processing

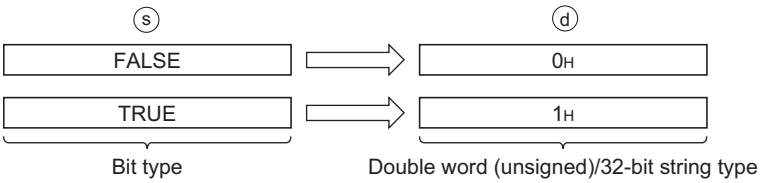
• BOOL_TO_WORD, BOOL_TO_WORD_E

Converts bit type data input to (s) into word (unsigned)/16-bit string type data, and outputs the operation result from (d).
When the input value is FALSE, 0H is output in word (unsigned)/16-bit string type data.
When the input value is TRUE, 1H is output in word (unsigned)/16-bit string type data.



• BOOL_TO_DWORD, BOOL_TO_DWORD_E

Converts bit type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).
When the input value is FALSE, 0H is output in double word (unsigned)/32-bit string type data.
When the input value is TRUE, 1H is output in double word (unsigned)/32-bit string type data.



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

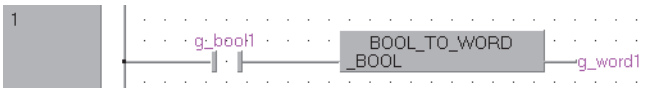
Program example

■BOOL_TO_WORD(_E)

The program which converts bit type data input to (s) into word (unsigned)/16-bit string type data, and outputs the operation result from (d).

• Function without EN/ENO (BOOL_TO_WORD)

[Structured ladder/FBD]

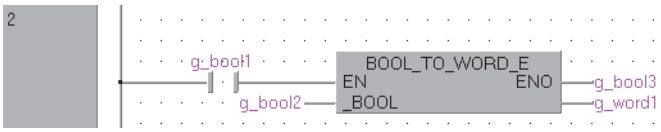


[ST]

```
g_word1 := BOOL_TO_WORD(g_bool1);
```

• Function with EN/ENO (BOOL_TO_WORD_E)

[Structured ladder/FBD]



[ST]

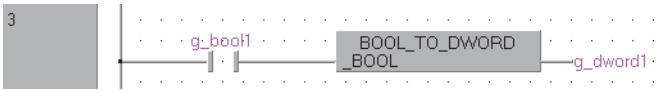
```
g_bool3 := BOOL_TO_WORD_E(g_bool1, g_bool2, g_word1);
```

■BOOL_TO_DWORD(_E)

The program which converts bit type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).

• Function without EN/ENO (BOOL_TO_DWORD)

[Structured ladder/FBD]



[ST]

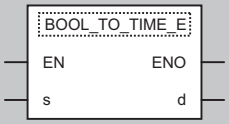
```
g_dword1 := BOOL_TO_DWORD(g_bool1);
```

Converting bit type to time type

BOOL_TO_TIME(_E)

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

Structured ladder/FBD



ST

```
ENO:= BOOL_TO_TIME_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
BOOL_TO_TIME, BOOL_TO_TIME_E

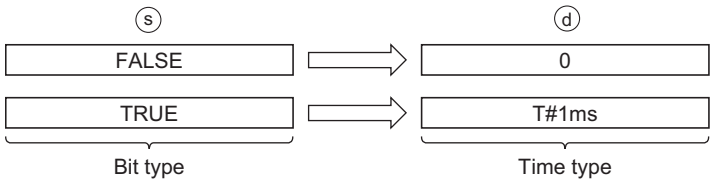
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_BOOL)	Input	Bit
Output argument	ENO	Execution result (TRUE: Normal execution, FALSE: Error or stop)	Bit
	d	Output	Time

Processing details

■Operation processing

Converts bit type data input to (s) into time type data, and outputs the operation result from (d).
When the input value is FALSE, 0 is output in time type data.
When the input value is TRUE, 1 is output in time type data.



■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

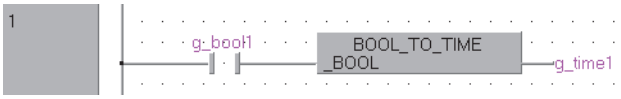
Program example

■BOOL_TO_TIME(_E)

The program which converts bit type data input to (s) into time type data, and outputs the operation result from (d).

- Function without EN/ENO (BOOL_TO_TIME)

[Structured ladder/FBD]



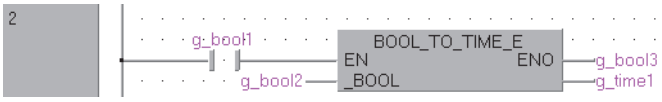
[ST]

```
g_time1 := BOOL_TO_TIME(g_bool1);
```

The program which converts bit type data input to (s) into time type data, and outputs the operation result from (d).

- Function without EN/ENO (BOOL_TO_TIME)

[Structured ladder/FBD]



[ST]

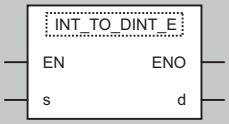
```
g_bool3 := BOOL_TO_TIME_E(g_bool1, g_bool2, g_time1);
```

Converting word (signed) type to double word (signed) type

INT_TO_DINT(_E)

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

Structured ladder/FBD



ST

```
ENO:= INT_TO_DINT_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
INT_TO_DINT, INT_TO_DINT_E

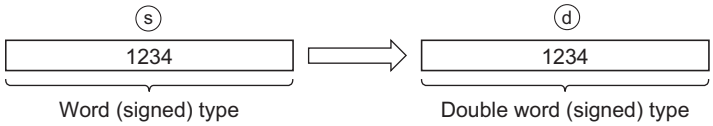
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_INT)	Input	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Double word (signed)

Processing details

■Operation processing

Converts word (signed) type data input to (s) into double word (signed) type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

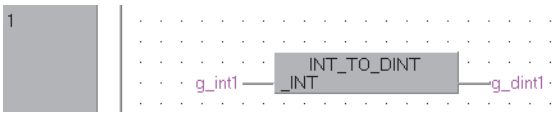
Program example

■INT_TO_DINT(_E)

The program which converts word (signed) type data input to (s) into double word (signed) type data, and outputs the operation result from (d).

- Function without EN/ENO (INT_TO_DINT)

[Structured ladder/FBD]

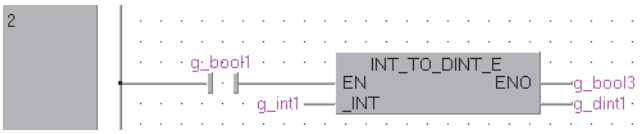


[ST]

```
g_dint1 := INT_TO_DINT(g_int1);
```

- Function with EN/ENO (INT_TO_DINT_E)

[Structured ladder/FBD]



[ST]

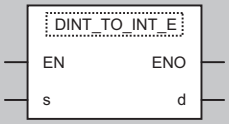
```
g_bool3 := INT_TO_DINT_E(g_bool1, g_int1, g_dint1);
```

Converting double word (signed) type to word (signed) type

DINT_TO_INT(_E)

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

Structured ladder/FBD



ST

```
ENO:= DINT_TO_INT_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
DINT_TO_INT, DINT_TO_INT_E

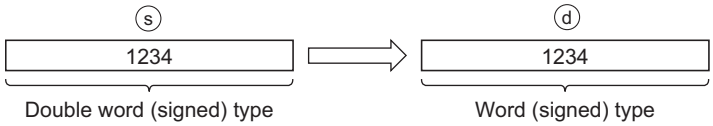
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_DINT)	Input	Double word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (signed)

Processing details

■Operation processing

Converts double word (signed) type data input to (s) into word (signed) type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Point

When the DINT_TO_INT(_E) function is executed, low-order 16-bit data of double word (signed) type data input to (s) are converted into word (signed) type data. High-order 16-bit data are discarded.

Operation error

- No operation error occurs.

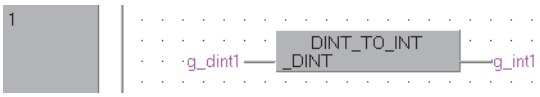
Program example

■DINT_TO_INT(_E)

The program which converts double word (signed) type data input to (s) into word (signed) type data, and outputs the operation result from (d).

- Function without EN/ENO (DINT_TO_INT)

[Structured ladder/FBD]

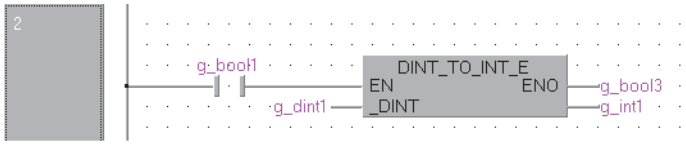


[ST]

```
g_int1 := DINT_TO_INT(g_dint1);
```

- Function with EN/ENO (DINT_TO_INT_E)

[Structured ladder/FBD]



[ST]

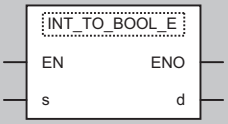
```
g_bool3 := DINT_TO_INT_E(g_bool1, g_dint1, g_int1);
```

Converting word (signed), double word (signed) type to bit type

INT_TO_BOOL(_E), DINT_TO_BOOL(_E)

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

Structured ladder/FBD



ST

```
ENO:= INT_TO_BOOL_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
INT_TO_BOOL, INT_TO_BOOL_E, DINT_TO_BOOL, DINT_TO_BOOL_E

■Argument

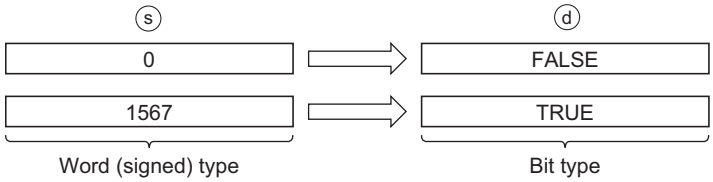
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_INT, _DINT)	Input	Word (signed), double word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Bit

Processing details

■Operation processing

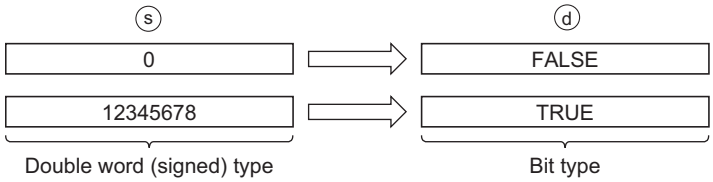
• INT_TO_BOOL, INT_TO_BOOL_E

Converts word (signed) type data input to (s) into bit type data, and outputs the operation result from (d).
When the input value is 0, FALSE is output in bit type data.
When the input value is other than 0, TRUE is output in bit type data.



• DINT_TO_BOOL, DINT_TO_BOOL_E

Converts double word (signed) type data input to (s) into bit type data, and outputs the operation result from (d).
When the input value is 0, FALSE is output in bit type data.
When the input value is other than 0, TRUE is output in bit type data.



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

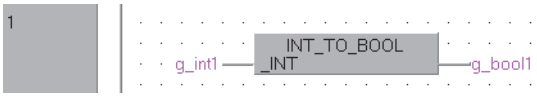
Program example

■INT_TO_BOOL(_E)

The program which converts word (signed) type data input to (s) into bit type data, and outputs the operation result from (d).

• Function without EN/ENO (INT_TO_BOOL)

[Structured ladder/FBD]

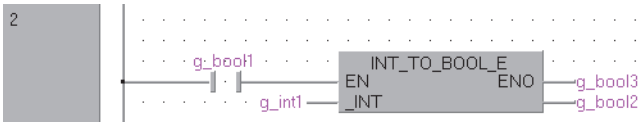


[ST]

```
g_bool1 := INT_TO_BOOL(g_int1);
```

• Function with EN/ENO (INT_TO_BOOL_E)

[Structured ladder/FBD]



[ST]

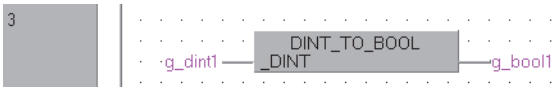
```
g_bool3 := INT_TO_BOOL_E(g_bool1, g_int1, g_bool2);
```

■DINT_TO_BOOL(_E)

The program which converts double word (signed) type data input to (s) into bit type data, and outputs the operation result from (d).

• Function without EN/ENO (DINT_TO_BOOL)

[Structured ladder/FBD]



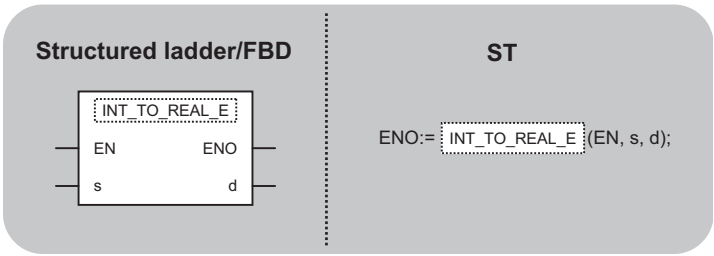
[ST]

```
g_bool1 := DINT_TO_BOOL(g_dint1);
```

Converting word (signed), double word (signed) type to single-precision real type

INT_TO_REAL(_E), DINT_TO_REAL(_E)

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



The following function(s) can go in the dotted squares.
INT_TO_REAL, INT_TO_REAL_E, DINT_TO_REAL, DINT_TO_REAL_E

■Argument

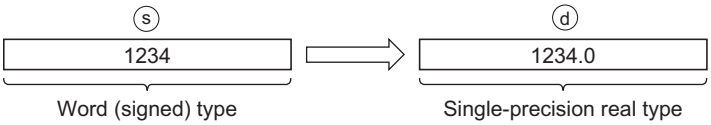
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_INT, _DINT)	Input	Word (signed), double word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Single-precision real number

Processing details

■Operation processing

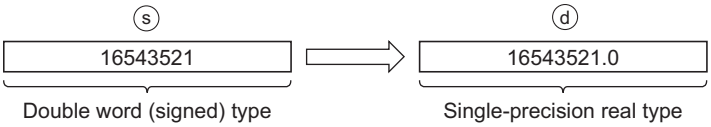
• INT_TO_REAL, INT_TO_REAL_E

Converts word (signed) type data input to (s) into single-precision real type data, and outputs the operation result from (d).



• DINT_TO_REAL, DINT_TO_REAL_E

Converts double word (signed) type data input to (s) into single-precision real type data, and outputs the operation result from (d).



The number of significant figures of single-precision real type data is approximately 7 since the data is processed in 32-bit single precision. Accordingly, the converted data includes an error (rounding error) if an integer value is outside the range of -16777216 to 16777215.

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

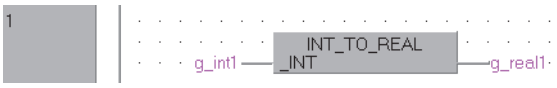
Program example

■INT_TO_REAL(_E)

The program which converts word (signed) type data input to (s) into single-precision real type data, and outputs the operation result from (d).

• Function without EN/ENO (INT_TO_REAL)

[Structured ladder/FBD]

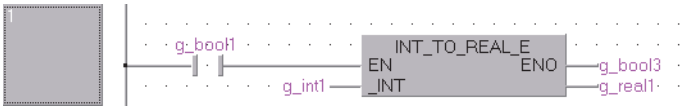


[ST]

```
g_real1 := INT_TO_REAL(g_int1);
```

• Function with EN/ENO (INT_TO_REAL_E)

[Structured ladder/FBD]



[ST]

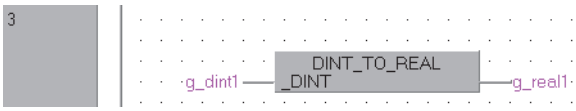
```
g_bool3 := INT_TO_REAL_E(g_bool1, g_int1, g_real1);
```

■DINT_TO_REAL(_E)

The program which converts double word (signed) type data input to (s) into single-precision real type data, and outputs the operation result from (d).

• Function without EN/ENO (DINT_TO_REAL)

[Structured ladder/FBD]



[ST]

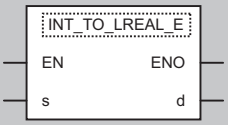
```
g_real1 := DINT_TO_REAL(g_dint1);
```

Converting word (signed), double word (signed) type to double-precision real type

INT_TO_LREAL(_E), DINT_TO_LREAL(_E)

Basic (×) | High performance (×) | Process (×) | Redundant (×) | Universal | LCPU

Structured ladder/FBD



ST

```
ENO:= INT_TO_LREAL_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
INT_TO_LREAL, INT_TO_LREAL_E, DINT_TO_LREAL, DINT_TO_LREAL_E

■Argument

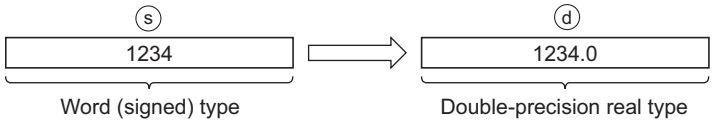
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_INT, _DINT)	Input	Word (signed), double word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Double-precision real

Processing details

■Operation processing

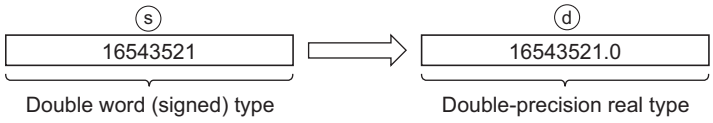
• INT_TO_LREAL, INT_TO_LREAL_E

Converts word (signed) type data input to (s) into double-precision real type data, and outputs the operation result from (d).



• DINT_TO_LREAL, DINT_TO_LREAL_E

Converts double word (signed) type data input to (s) into double-precision real type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

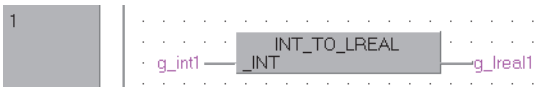
Program example

■INT_TO_LREAL(_E)

The program which converts word (signed) type data input to (s) into double-precision real type data, and outputs the operation result from (d).

• Function without EN/ENO (INT_TO_LREAL)

[Structured ladder/FBD]

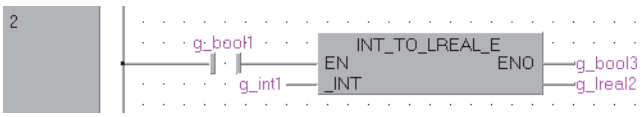


[ST]

```
g_lreal1 := INT_TO_LREAL(g_int1);
```

• Function with EN/ENO

[Structured ladder/FBD]



[ST]

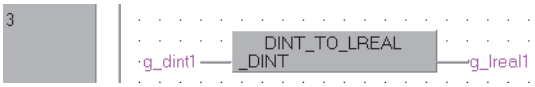
```
g_bool3 := INT_TO_LREAL_E(g_bool1, g_int1, g_lreal2);
```

■DINT_TO_LREAL(_E)

The program which converts double word (signed) type data input to (s) into double-precision real type data, and outputs the operation result from (d).

• Function without EN/ENO (DINT_TO_LREAL)

[Structured ladder/FBD]



[ST]

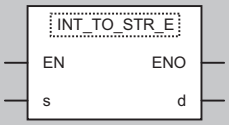
```
g_lreal1 := DINT_TO_LREAL(g_dint1);
```

Converting word (signed), double word (signed) type to string type

INT_TO_STR(_E), DINT_TO_STR(_E)

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

Structured ladder/FBD



ST

```
ENO:= INT_TO_STR_E (EN, s, d);
```

The following function(s) can go in the dotted squares.

INT_TO_STR, INT_TO_STR_E, DINT_TO_STR, DINT_TO_STR_E

■Argument

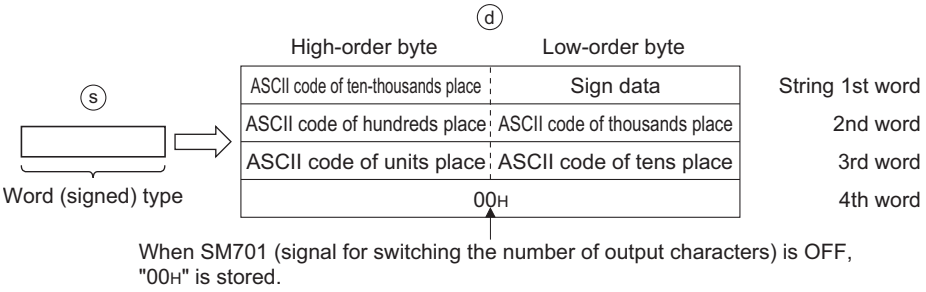
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_INT, _DINT)	Input	Word (signed), double word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	String(6)/(11)

Processing details

■Operation processing

• INT_TO_STR, INT_TO_STR_E

Converts word (signed) type data input to (s) into string type data, and outputs the operation result from (d).

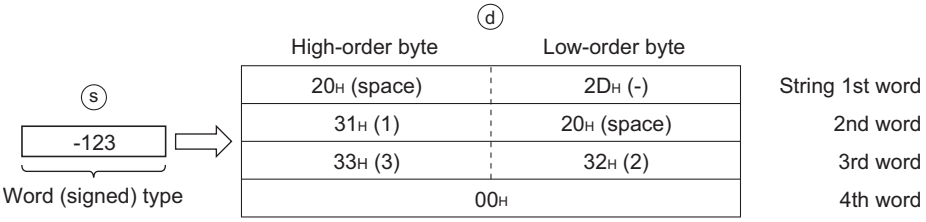


'20H (space)' is stored in 'Sign data' when the input value is positive; '2DH (-)' is stored when negative.

If the number of significant figures is less, '20H (space)' is stored to high-order digits.

Ex.

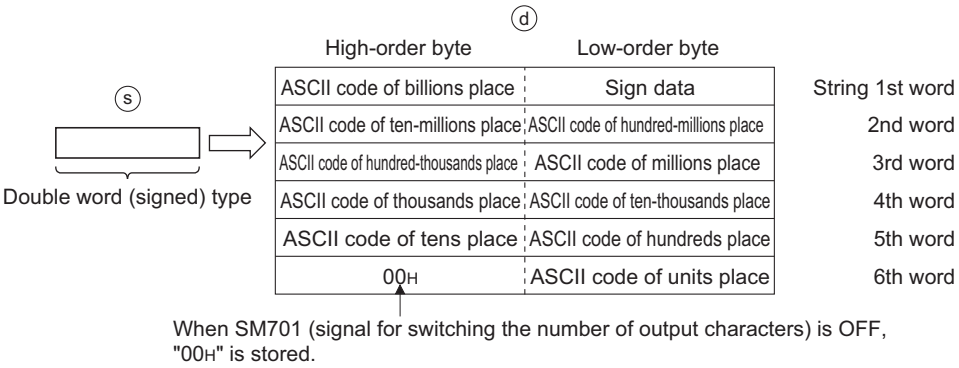
Inputting -123



When SM701 (signal for switching the number of output characters) is OFF, "00H" is stored to the end of the character string.

• DINT_TO_STR, DINT_TO_STR_E

Converts double word (signed) type data input to (s) into string type data, and outputs the operation result from (d).

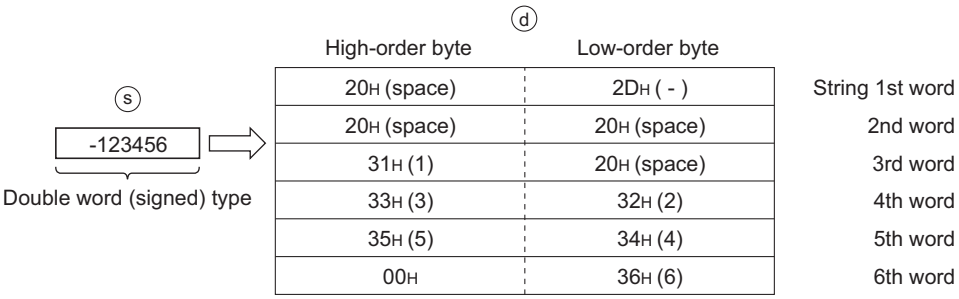


'20H (space)' is stored in 'Sign data' when the input value is positive; '2DH (-)' is stored when negative.

If the number of significant figures is less, '20H (space)' is stored to high-order digits.

Ex.

Inputting -123456



When SM701 (signal for switching the number of output characters) is OFF, "00H" is stored to the end of the character string.

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

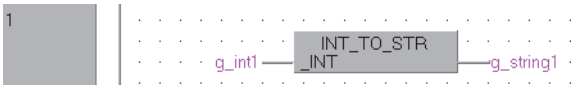
Program example

■INT_TO_STR(_E)

The program which converts word (signed) type data input to (s) into string type data, and outputs the operation result from (d).

- Function without EN/ENO (INT_TO_STR)

[Structured ladder/FBD]

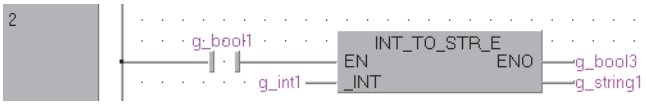


[ST]

```
g_string1 := INT_TO_STR(g_int1);
```

- Function with EN/ENO (INT_TO_STR_E)

[Structured ladder/FBD]



[ST]

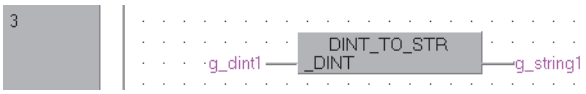
```
g_bool3 := INT_TO_STR_E(g_bool1, g_int1, g_string1);
```

■DINT_TO_STR(_E)

The program which converts double word (signed) type data input to (s) into string type data, and outputs the operation result from (d).

- Function without EN/ENO (DINT_TO_STR)

[Structured ladder/FBD]



[ST]

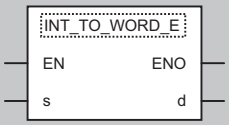
```
g_string1 := DINT_TO_STR(g_dint1);
```

Converting word (signed), double word (signed) type to word (unsigned)/16-bit string type

INT_TO_WORD(_E), DINT_TO_WORD(_E)

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

Structured ladder/FBD



ST

```
ENO:= INT_TO_WORD_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
INT_TO_WORD, INT_TO_WORD_E, DINT_TO_WORD, DINT_TO_WORD_E

■Argument

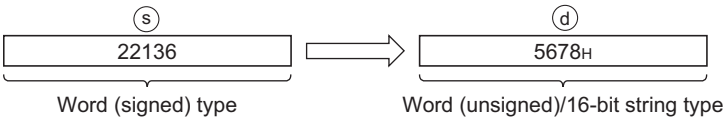
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_INT, _DINT)	Input	Word (signed), double word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (unsigned)/16-bit string

Processing details

■Operation processing

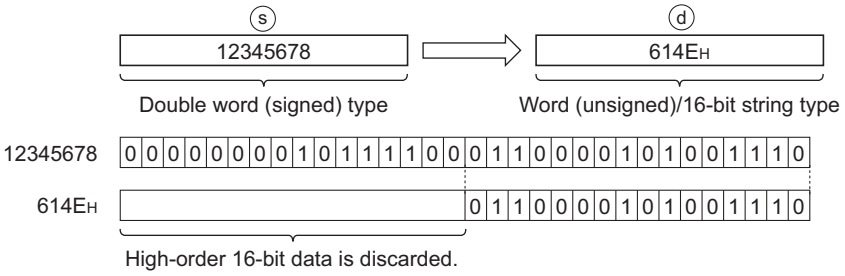
• INT_TO_WORD, INT_TO_WORD_E

Converts word (signed) type data input to (s) into word (unsigned)/16-bit string type data, and outputs the operation result from (d).



• DINT_TO_WORD, DINT_TO_WORD_E

Converts double word (signed) type data input to (s) into word (unsigned)/16-bit string type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Point

When the DINT_TO_WORD(_E) function is executed, low-order 16-bit data of double word (signed) type data input to input variable (s) are converted into word (unsigned)/16-bit string type data. High-order word (unsigned)/16-bit string type data are discarded.

Operation error

• No operation error occurs.

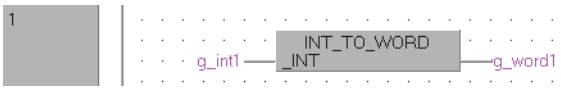
Program example

■INT_TO_WORD(_E)

The program which converts word (signed) type data input to (s) into word (unsigned)/16-bit string type data, and outputs the operation result from (d).

• Function without EN/ENO (INT_TO_WORD)

[Structured ladder/FBD]

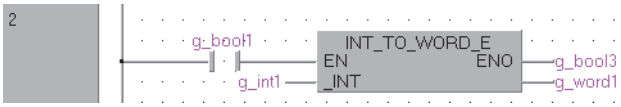


[ST]

```
g_word1 := INT_TO_WORD(g_int1);
```

• Function with EN/ENO (INT_TO_WORD_E)

[Structured ladder/FBD]



[ST]

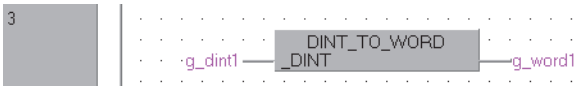
```
g_bool3 := INT_TO_WORD_E(g_bool1, g_int1, g_word1);
```

■DINT_TO_WORD(_E)

The program which converts double word (signed) type data input to (s) into word (unsigned)/16-bit string type data, and outputs the operation result from (d).

• Function without EN/ENO (DINT_TO_WORD)

[Structured ladder/FBD]



[ST]

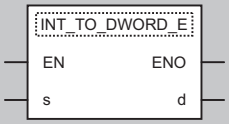
```
g_word1 := DINT_TO_WORD(g_dint1);
```

Converting word (signed), double word (signed) type to double word (unsigned)/32-bit string type

INT_TO_DWORD(_E), DINT_TO_DWORD(_E)

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

Structured ladder/FBD



ST

```
ENO:= INT_TO_DWORD_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
INT_TO_DWORD, INT_TO_DWORD_E, DINT_TO_DWORD, DINT_TO_DWORD_E

■Argument

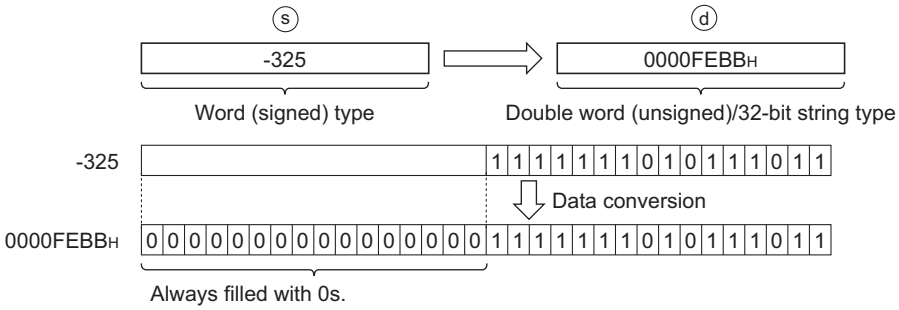
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_INT, _DINT)	Input	Word (signed), double word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Double word (unsigned)/32-bit string

Processing details

■Operation processing

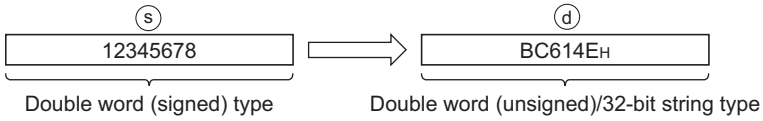
• INT_TO_DWORD, INT_TO_DWORD_E

Converts word (signed) type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).



• DINT_TO_DWORD, DINT_TO_DWORD_E

Converts double word (signed) type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

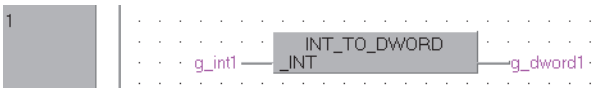
Program example

■INT_TO_DWORD(_E)

The program which converts word (signed) type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).

• Function without EN/ENO (INT_TO_WORD)

[Structured ladder/FBD]

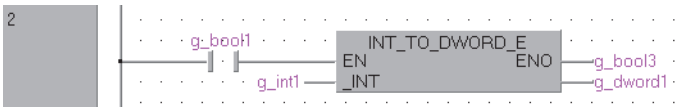


[ST]

```
g_dword1:= INT_TO_DWORD(g_int1);
```

• Function with EN/ENO (INT_TO_DWORD_E)

[Structured ladder/FBD]



[ST]

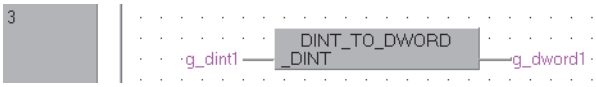
```
g_bool3 := INT_TO_DWORD_E(g_bool1, g_int1, g_dword1);
```

■DINT_TO_DWORD(_E)

The program which converts double word (signed) type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).

• Function without EN/ENO (DINT_TO_DWORD)

[Structured ladder/FBD]



[ST]

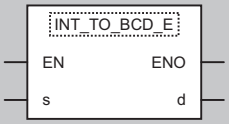
```
g_dword1 := DINT_TO_DWORD(g_dint1);
```

Converting word (signed), double word (signed) type to BCD type

INT_TO_BCD(_E), DINT_TO_BCD(_E)

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

Structured ladder/FBD



ST

```
ENO:= INT_TO_BCD_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
INT_TO_BCD, INT_TO_BCD_E, DINT_TO_BCD, DINT_TO_BCD_E

■Argument

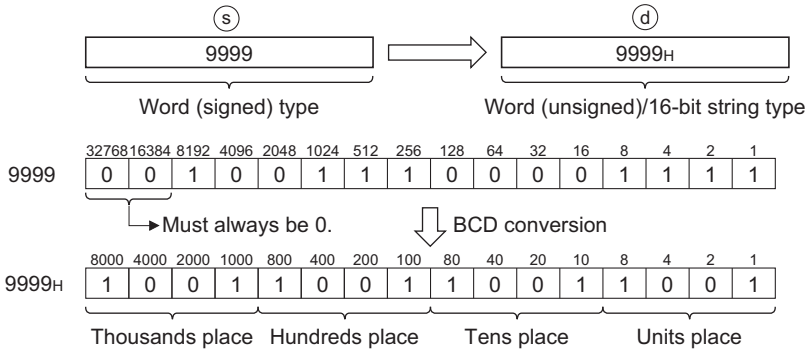
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_INT, _DINT)	Input	Word (signed), double word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (unsigned)/16-bit string, double word (unsigned)/32-bit string

Processing details

■Operation processing

• INT_TO_BCD, INT_TO_BCD_E

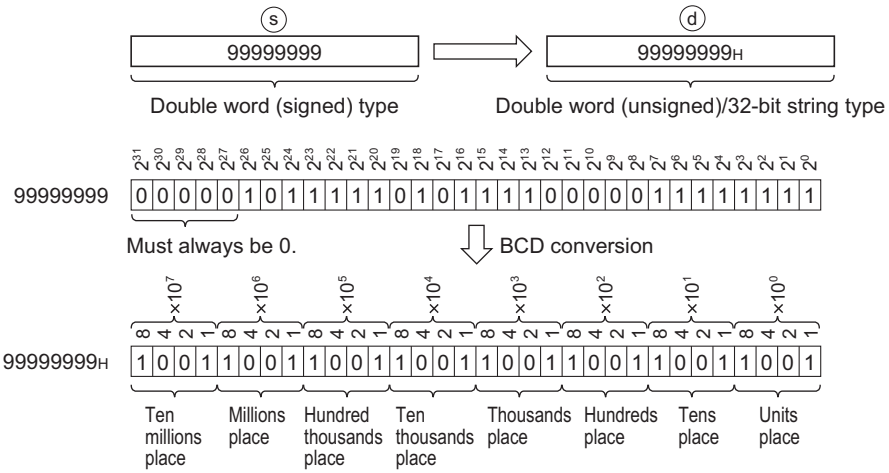
Converts word (signed) type data input to (s) into BCD type data, and outputs the operation result from (d).



The value to be input to (s) is word (signed) type data within the range from 0 to 9999.

• DINT_TO_BCD, DINT_TO_BCD_E

Converts double word (signed) type data input to (s) into BCD type data, and outputs the operation result from (d).

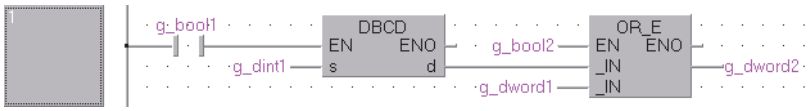


The value to be input to (s) is double word (signed) type data within the range from 0 to 99999999.

Word (unsigned)/16-bit string type, double word (unsigned)/32-bit string type data can be specified for (s). Bit type cannot be specified.

Point

The output from (d) cannot be used with connecting to the input of double word (unsigned)/32-bit string type data. In this case, use the DBCD instruction.



■Operation result

• Function without EN/ENO

The following table shows the operation results.

Operation result	(d)
No operation error	Operation output value
Operation error	Undefined value

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• An operation error occurs in the following cases.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	When the value input exceeds 9999 in the execution of the INT_TO_BCD(_E) When the value input exceeds 99999999 in the execution of the DINT_TO_BCD(_E)	○	○	○	○	○	○

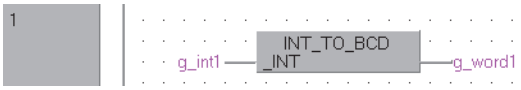
Program example

■INT_TO_BCD(_E)

The program which converts word (signed) type data input to (s) into BCD type data, and outputs the operation result from (d).

• Function without EN/ENO (INT_TO_BCD)

[Structured ladder/FBD]

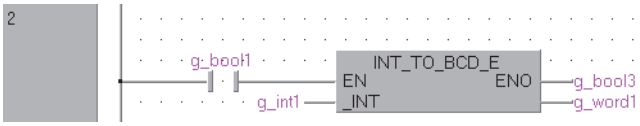


[ST]

```
g_word1 := INT_TO_BCD(g_int1);
```

• Function with EN/ENO (INT_TO_BCD_E)

[Structured ladder/FBD]



[ST]

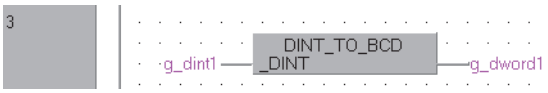
```
g_bool3 := INT_TO_BCD_E(g_bool1, g_int1, g_word1);
```

■DINT_TO_BCD(_E)

The program which converts double word (signed) type data input to (s) into BCD type data, and outputs the operation result from (d).

• Function without EN/ENO (DINT_TO_BCD)

[Structured ladder/FBD]



[ST]

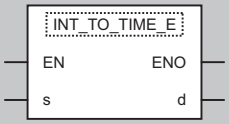
```
g_dword1 := DINT_TO_BCD(g_dint1);
```

Converting word (signed), double word (signed) type to time type

INT_TO_TIME(_E), DINT_TO_TIME(_E)

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

Structured ladder/FBD



ST

```
ENO:= INT_TO_TIME_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
INT_TO_TIME, INT_TO_TIME_E, DINT_TO_TIME, DINT_TO_TIME_E

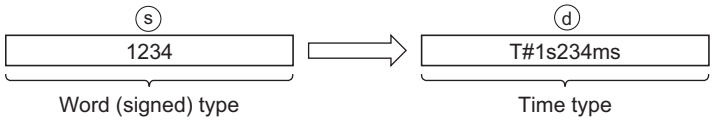
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_INT, _DINT)	Input	Word (signed), double word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Time

Processing details

■Operation processing

Converts word (signed) /double word (signed) type data input to (s) into time type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

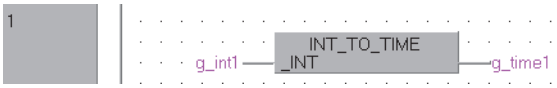
Program example

■INT_TO_TIME(_E)

The program which converts word (signed) type data input to (s) into time type data, and outputs the operation result from (d).

- Function without EN/ENO (INT_TO_TIME)

[Structured ladder/FBD]

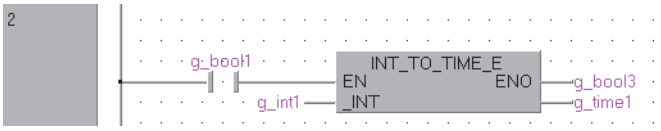


[ST]

```
g_time1:= INT_TO_TIME(g_int1);
```

- Function with EN/ENO (INT_TO_TIME_E)

[Structured ladder/FBD]



[ST]

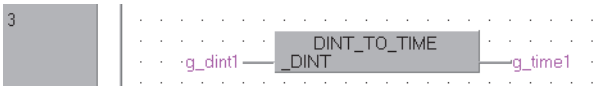
```
g_bool3 := INT_TO_TIME_E(g_bool1, g_int1, g_time1);
```

■DINT_TO_TIME(_E)

The program which converts double word (signed) type data input to (s) into time type data, and outputs the operation result from (d).

- Function without EN/ENO (DINT_TO_TIME)

[Structured ladder/FBD]



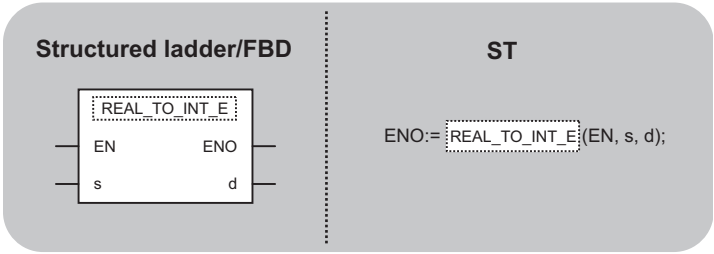
[ST]

```
g_time1 := DINT_TO_TIME(g_dint1);
```

Converting single-precision real type to word (signed), double word (signed) type

REAL_TO_INT(_E), REAL_TO_DINT(_E)

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



The following function(s) can go in the dotted squares.
REAL_TO_INT, REAL_TO_INT_E, REAL_TO_DINT, REAL_TO_DINT_E

■Argument

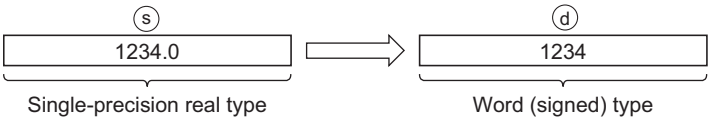
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_REAL)	Input	Single-precision real number
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (signed), double word (signed)

Processing details

■Operation processing

• REAL_TO_INT, REAL_TO_INT_E

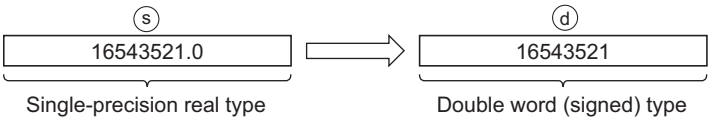
Converts single-precision real type data input to (s) into word (signed) type data, and outputs the operation result from (d).



The value to be input to (s) is single-precision real type data, within the range from -32768 to 32767.
The converted data is the value rounded single-precision real type data to the first digit after the decimal point.

• REAL_TO_DINT, REAL_TO_DINT_E

Converts single-precision real type data input to (s) into double word (signed) type data, and outputs the operation result from (d).



The value to be input to (s) is single-precision real type data within the range from -2147483648 to 2147483647.
However, a rounding error may occur when setting the input value by programming tool. For precautions when setting an input value using a programming tool, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)
The converted data is the value rounded single-precision real type data to the first digit after the decimal point.

■Operation result

• Function without EN/ENO

The following table shows the operation results.

Operation result	(d)
No operation error	Operation output value
Operation error	Undefined value

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• An operation error occurs in the following cases.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	When the REAL_TO_INT(_E) instruction is used, the input value is outside the range of -32768 to 32767. When the REAL_TO_DINT(_E) instruction is used, the input value is outside the range of -2147483648 to 2147483647.	○	○	○	○	○	○

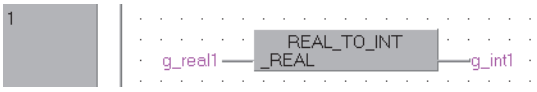
Program example

■REAL_TO_INT(_E)

The program which converts single-precision real type data input to (s) into word (signed) type data, and outputs the operation result from (d).

• Function without EN/ENO (REAL_TO_INT)

[Structured ladder/FBD]

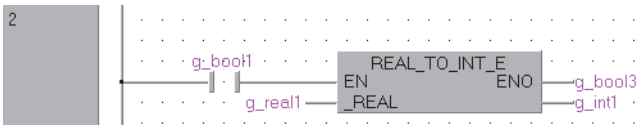


[ST]

```
g_int1:= REAL_TO_INT(g_real1);
```

• Function with EN/ENO (REAL_TO_INT_E)

[Structured ladder/FBD]



[ST]

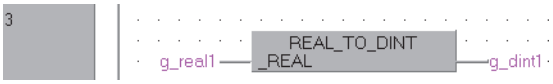
```
g_bool3 := REAL_TO_INT_E(g_bool1, g_real1, g_int1);
```

■REAL_TO_DINT(_E)

The program which converts single-precision real type data input to (s) into double word (signed) type data, and outputs the operation result from (d).

• Function without EN/ENO (REAL_TO_DINT)

[Structured ladder/FBD]



[ST]

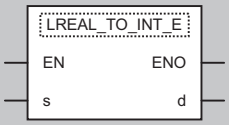
```
g_dint1 := REAL_TO_DINT(g_real1);
```

Converting double-precision real type to word (signed), double word (signed) type

LREAL_TO_INT(_E), LREAL_TO_DINT(_E)

Basic (×) High performance (×) Process (×) Redundant (×) Universal LCPU

Structured ladder/FBD



ST

```
ENO:= LREAL_TO_INT_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
LREAL_TO_INT, LREAL_TO_INT_E, LREAL_TO_DINT, LREAL_TO_DINT_E

■Argument

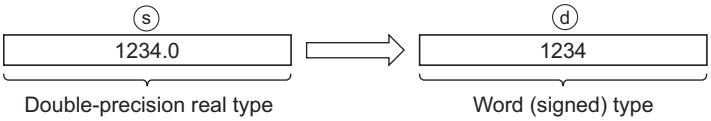
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_LREAL)	Input	Double-precision real
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (signed), double word (signed)

Processing details

■Operation processing

• LREAL_TO_INT, LREAL_TO_INT_E

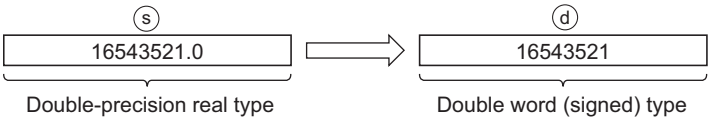
Converts double-precision real type data input to (s) into word (signed) type data, and outputs the operation result from (d).



The value to be input to (s) is double-precision real type data, within the range from -32768 to 32767.
The converted data is the value rounded double-precision real type data to the first digit after the decimal point.

• LREAL_TO_DINT, LREAL_TO_DINT_E

Converts double-precision real type data input to (s) into double word (signed) type data, and outputs the operation result from (d).



The value to be input to (s) is double-precision real type data within the range from -2147483648 to 2147483647. However, rounding error may occur when setting the input value by programming tool. For precautions when setting an input value using a programming tool, refer to the following.

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

The converted data is the value rounded double-precision real type data to the first digit after the decimal point.

■Operation result

• Function without EN/ENO

The following table shows the operation results.

Operation result	(d)
No operation error	Operation output value
Operation error	Undefined value

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• An operation error occurs in the following cases.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The input value is -0 or outside the following range. 0, $2^{-1022} \leq |(s)| < 2^{1024}$ When the LREAL_TO_INT(_E) instruction is used, the input value is outside the range of -32768.0 to 32767.0. When the LREAL_TO_DINT(_E) instruction is used, the input value is outside the range of -2147483648.0 to 2147483647.0.	—	—	—	—	○	○

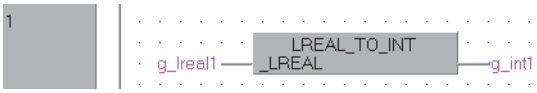
Program example

■LREAL_TO_INT(_E)

The program which converts double-precision real type data input to (s) into word (signed) type data, and outputs the operation result from (d).

• Function without EN/ENO (LREAL_TO_INT)

[Structured ladder/FBD]

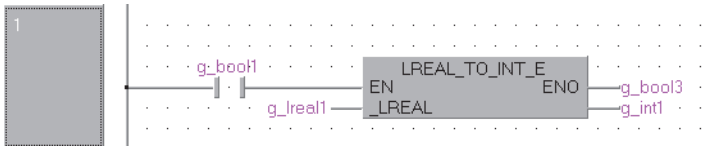


[ST]

```
g_int1:= LREAL_TO_INT(g_lreal1);
```

• Function with EN/ENO (LREAL_TO_INT_E)

[Structured ladder/FBD]



[ST]

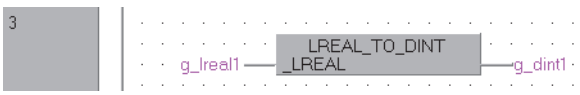
```
g_bool3 := LREAL_TO_INT_E(g_bool1, g_lreal1, g_int1);
```

■LREAL_TO_DINT(_E)

The program which converts double-precision real type data input to (s) into double word (signed) type data, and outputs the operation result from (d).

• Function without EN/ENO (LREAL_TO_DINT)

[Structured ladder/FBD]



[ST]

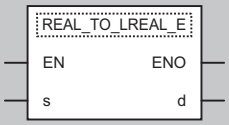
```
g_dint1 := LREAL_TO_DINT(g_lreal1);
```

Converting single-precision real type to double-precision real type

REAL_TO_LREAL(_E)

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

Structured ladder/FBD



ST

```
ENO:= REAL_TO_LREAL_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
REAL_TO_LREAL, REAL_TO_LREAL_E

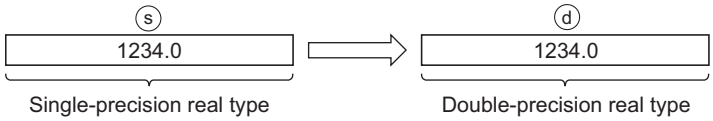
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_REAL)	Input	Single-precision real number
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Double-precision real

Processing details

■Operation processing

- Converts single-precision real type data input to (s) into double-precision real type data, and outputs the operation result from (d).



- Rounding error may occur when specifying the input value to (s) by programming tool. For precautions when setting an input value using a programming tool, refer to the following.

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

■Operation result

• Function without EN/ENO

The following table shows the operation results.

Operation result	(d)
No operation error	Operation output value
Operation error	Undefined value

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• An operation error occurs in the following cases.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The input value is -0 or outside the following range. 0, $2^{-126} \leq \|(s)\| < 2^{128}$	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{1024} \leq$ | operation result |	—	—	—	—	○	○

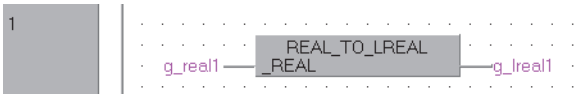
Program example

■REAL_TO_LREAL(_E)

The program which converts single-precision real type data input to (s) into double-precision real type data, and outputs the operation result from (d).

• Function without EN/ENO (REAL_TO_LREAL)

[Structured ladder/FBD]

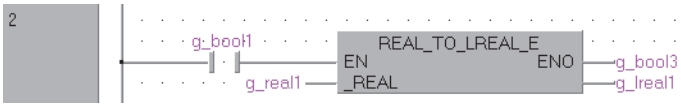


[ST]

```
g_lreal1 := REAL_TO_LREAL(g_real1);
```

• Function with EN/ENO (REAL_TO_LREAL_E)

[Structured ladder/FBD]



[ST]

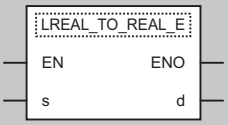
```
g_bool3 := REAL_TO_LREAL_E(g_bool1, g_real1, g_lreal1);
```

Converting double-precision real type to single-precision real type

LREAL_TO_REAL(_E)

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

Structured ladder/FBD



ST

```
ENO:= LREAL_TO_REAL_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
LREAL_TO_REAL, LREAL_TO_REAL_E

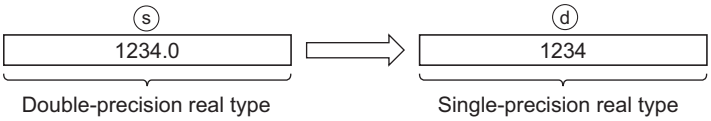
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_LREAL)	Input	Double-precision real
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Single-precision real number

Processing details

■Operation processing

- Converts double-precision real type data input to (s) into single-precision real type data, and outputs the operation result from (d).



- Rounding error may occur when setting the input value to (s) by programming tool. For precautions when setting an input value using a programming tool, refer to the following.

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

■Operation result

• Function without EN/ENO

The following table shows the operation results.

Operation result	(d)
No operation error	Operation output value
Operation error	Undefined value

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• An operation error occurs in the following cases.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4140	The input value is -0 or outside the following range. 0, $2^{-1022} \leq \|(s)\| < 2^{1024}$	—	—	—	—	○	○
4141	The operation result exceeds the following range. (An overflow occurs.) $2^{128} \leq$ | operation result |	—	—	—	—	○	○

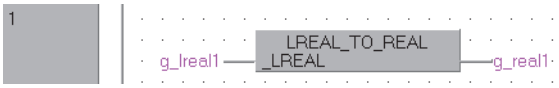
Program example

■LREAL_TO_REAL(_E)

The program which converts double-precision real type data input to (s) into single-precision real type data, and outputs the operation result from (d).

• Function without EN/ENO (LREAL_TO_REAL)

[Structured ladder/FBD]

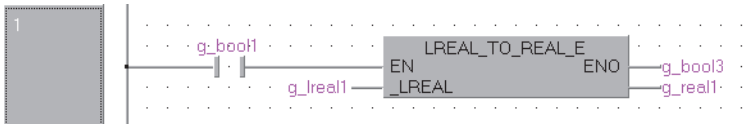


[ST]

```
g_real1 := LREAL_TO_REAL(g_lreal1);
```

• Function with EN/ENO (LREAL_TO_REAL_E)

[Structured ladder/FBD]



[ST]

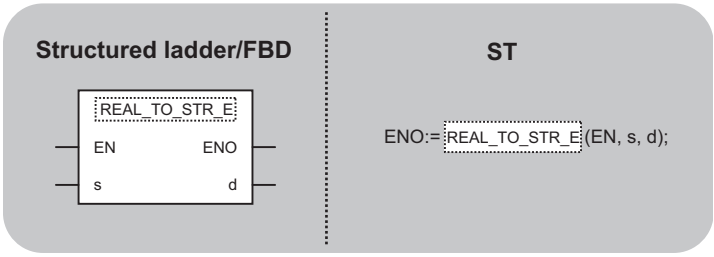
```
g_bool3 := LREAL_TO_REAL_E(g_bool1, g_lreal1, g_real1);
```

Converting single-precision real type to string type

REAL_TO_STR(_E)

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

This function is used in the Basic model QCPU with a serial number (first five digits) of "04122" or later.



The following function(s) can go in the dotted squares.

REAL_TO_STR, REAL_TO_STR_E

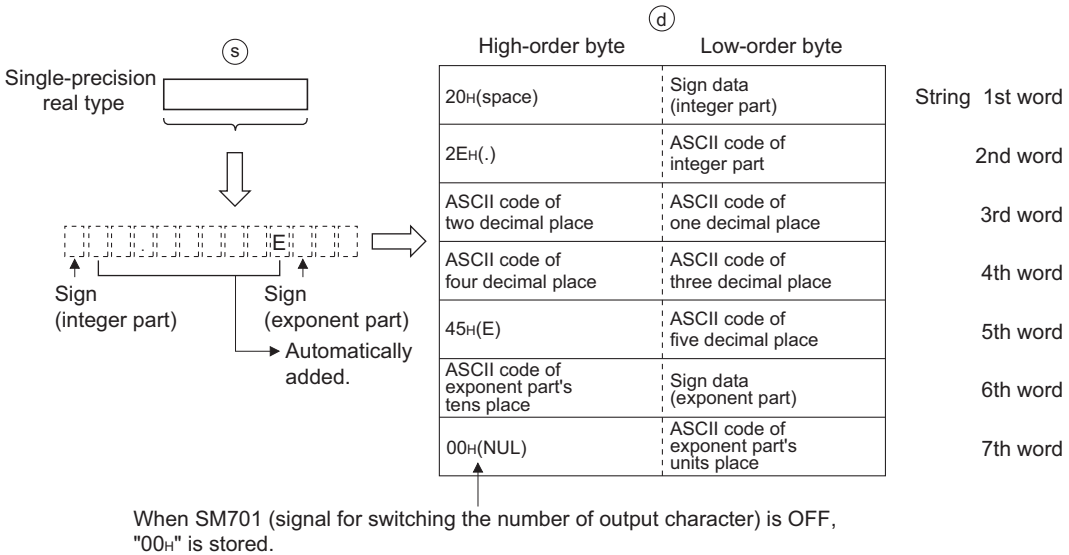
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_LREAL)	Input	Single-precision real number
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	String (13)

Processing details

■Operation processing

- Converts single-precision real type data input to (s) into string type (exponential form) data, and outputs the operation result from (d).

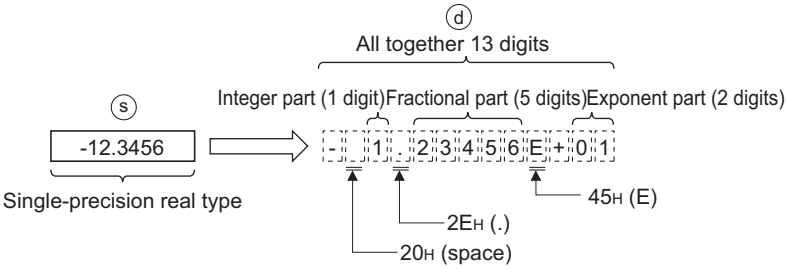


When SM701 (signal for switching the number of output character) is OFF, "00H" is stored.

• The character string data after conversion is output from output variable (d) in the following manner.

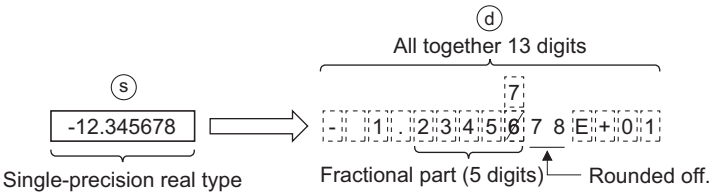
The number of digits is fixed respectively for the integer part, fractional part, and exponent part. (Integer part: 1 digit, fractional part: 5 digits, exponent part: 2 digits)

'20H' (space), '2EH' (.) and '45H' (E) are automatically stored in the 2nd, 4th and 10th bytes, respectively.

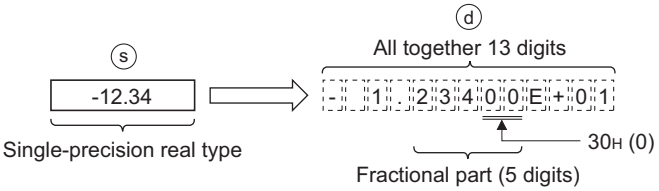


'20H' (space) is stored in 'Sign data' (integer part) when the input value is positive; '2DH' (-) is stored when negative.

Fractional part is rounded to 5 decimal places.

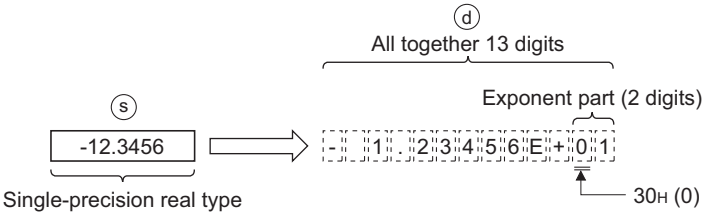


If the number of significant figures is less, '30H' (0) is stored to fractional part.



'2BH' (+) is stored in the 'Sign data' (exponent part) if the exponent is positive; '2DH' (-) is stored when negative.

'30H' (0) is stored to tens place in the exponent part if exponent part has only one digit.



• When SM701 (signal for switching the number of output characters) is OFF, "00H" is stored to the end of the character string (7th word).

• Rounding error may occur when specifying the input value to (s) by programming tool. For precautions when setting an input value using a programming tool, refer to the following.

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

■Operation result

• Function without EN/ENO

The following table shows the operation results.

Operation result	(d)
No operation error	Operation output value
Operation error	Undefined value

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• An operation error occurs in the following cases.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The input value is outside the range of -3.40282^{+38} to -1.17549^{-38}, 0 or 1.17549^{-38} to 3.40282^{+38}	○	○	○	○	○	○

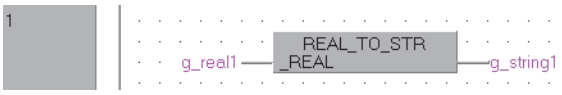
Program example

■REAL_TO_STR(_E)

The program which converts single-precision real type data input to (s) into string type (exponential form) data, and outputs the operation result from (d).

• Function without EN/ENO (REAL_TO_STR)

[Structured ladder/FBD]

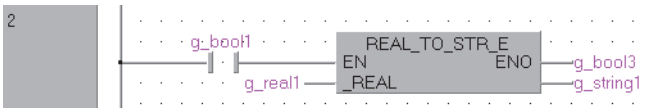


[ST]

```
g_string1 := REAL_TO_STR(g_real1);
```

• Function with EN/ENO (REAL_TO_STR_E)

[Structured ladder/FBD]



[ST]

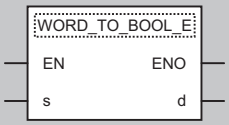
```
g_bool3 := REAL_TO_STR_E(g_bool1, g_real1, g_string1);
```

Converting word (unsigned)/16-bit string, double word (unsigned)/32-bit string type to bit type

WORD_TO_BOOL(_E), DWORD_TO_BOOL(_E)

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

Structured ladder/FBD



ST

```
ENO:=WORD_TO_BOOL_E(EN, s, d);
```

The following function(s) can go in the dotted squares.
WORD_TO_BOOL, WORD_TO_BOOL_E, DWORD_TO_BOOL, DWORD_TO_BOOL_E

■Argument

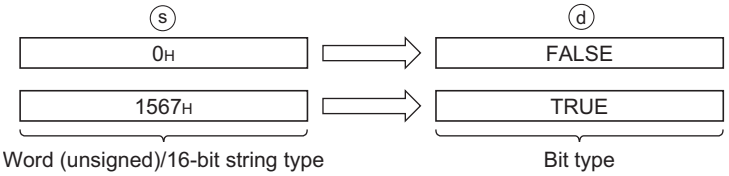
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_WORD, _DWORD)	Input	Word (unsigned)/16-bit string, Double word (unsigned)/32-bit string
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Bit

Processing details

■Operation processing

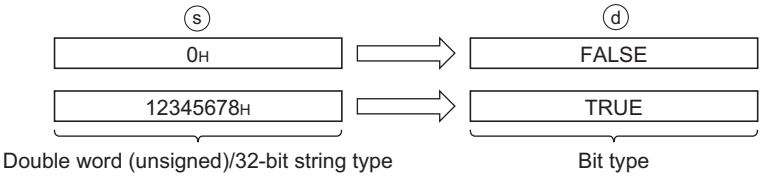
• WORD_TO_BOOL, WORD_TO_BOOL_E

Converts word (unsigned)/16-bit string type data input to (s) into bit type data, and outputs the operation result from (d).
When the input value is 0H, FALSE is output.
When the input value is other than 0H, TRUE is output.



• DWORD_TO_BOOL, DWORD_TO_BOOL_E

Converts double word (unsigned)/32-bit string type data input to (s) into bit type data, and outputs the operation result from (d).
When the input value is 0H, FALSE is output.
When the input value is other than 0H, TRUE is output.



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

Program example

■WORD_TO_BOOL(_E)

The program which converts word (unsigned)/16-bit string type data input to (s) into bit type data, and outputs the operation result from (d).

• Function without EN/ENO (WORD_TO_BOOL)

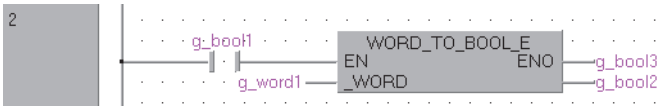
[Structured ladder/FBD]

[ST]

```
g_bool1:= WORD_TO_BOOL(g_word1);
```

• Function with EN/ENO (WORD_TO_BOOL_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := WORD_TO_BOOL_E(g_bool1, g_word1, g_bool2);
```

■DWORD_TO_BOOL(_E)

The program which converts double word (unsigned)/32-bit string type data input to (s) into bit type data, and outputs the operation result from (d).

• Function without EN/ENO (DWORD_TO_BOOL)

[Structured ladder/FBD]

[ST]

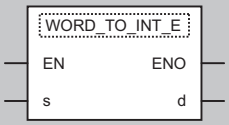
```
g_bool1 := DWORD_TO_BOOL(g_dword1);
```

Converting word (unsigned)/16-bit string type to word (signed), double word (signed) type

WORD_TO_INT(_E), WORD_TO_DINT(_E)

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

Structured ladder/FBD



ST

```
ENO:= WORD_TO_INT_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
WORD_TO_INT, WORD_TO_INT_E, WORD_TO_DINT, WORD_TO_DINT_E

■Argument

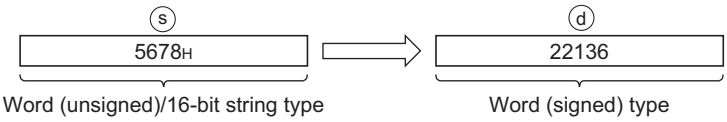
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_WORD)	Input	Word (unsigned)/16-bit string
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (signed), double word (signed)

Processing details

■Operation processing

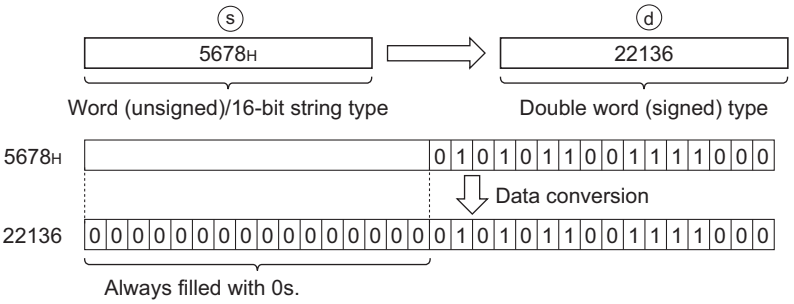
• WORD_TO_INT, WORD_TO_INT_E

Converts word (unsigned)/16-bit string type data input to (s) into word (signed) type data, and outputs the operation result from (d).



• WORD_TO_DINT, WORD_TO_DINT_E

Converts word (unsigned)/16-bit string type data input to (s) into double word (signed) type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

Program example

■WORD_TO_INT(_E)

The program which converts word (unsigned)/16-bit string type data input to (s) into word (signed) type data, and outputs the operation result from (d).

• Function without EN/ENO (WORD_TO_INT)

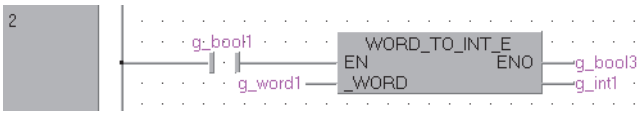
[Structured ladder/FBD]

[ST]

```
g_int1:= WORD_TO_INT(g_word1);
```

• Function with EN/ENO (WORD_TO_INT_E)

[Structured ladder/FBD]



[ST]

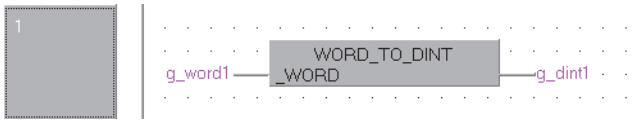
```
g_bool3 := WORD_TO_INT_E(g_bool1, g_word1, g_int1);
```

■WORD_TO_DINT(_E)

The program which converts word (unsigned)/16-bit string type data input to (s) into double word (signed) type data, and outputs the operation result from (d).

• Function without EN/ENO (WORD_TO_DINT)

[Structured ladder/FBD]



[ST]

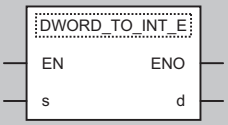
```
g_dint1 := WORD_TO_DINT(g_word1);
```

Converting double word (unsigned)/32-bit string type to word (signed), double word (signed) type

DWORD_TO_INT(_E), DWORD_TO_DINT(_E)

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

Structured ladder/FBD



ST

```
ENO:= DWORD_TO_INT_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
DWORD_TO_INT, DWORD_TO_INT_E, DWORD_TO_DINT, DWORD_TO_DINT_E

■Argument

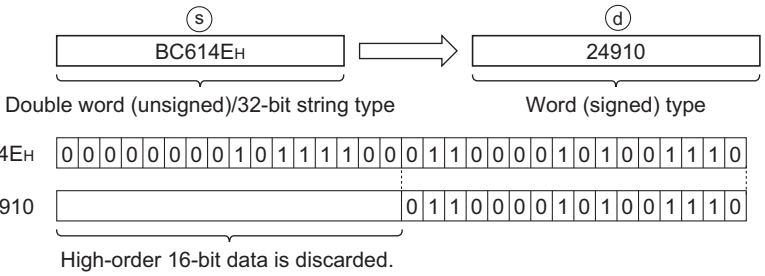
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_DWORD)	Input	Double word (unsigned)/32-bit string
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (signed), double word (signed)

Processing details

■Operation processing

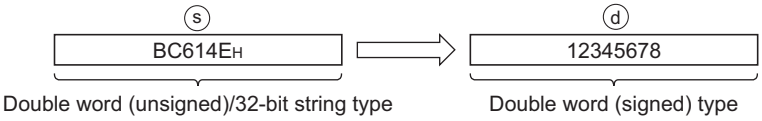
• DWORD_TO_INT, DWORD_TO_INT_E

Converts double word (unsigned)/32-bit string type data input to (s) into word (signed) type data, and outputs the operation result from (d).



• DWORD_TO_DINT, DWORD_TO_DINT_E

Converts double word (unsigned)/32-bit string type data input to (s) into double word (signed) type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Point

When the DINT_TO_INT(_E) function is executed, low-order 16-bit data of double word (unsigned)/32-bit string type data input to (s) are converted into word (signed) type data. High-order 16-bit data are discarded.

Operation error

• No operation error occurs.

Program example

■DWORD_TO_INT(_E)

The program which converts double word (unsigned)/32-bit string type data input to (s) into word (signed) type data, and outputs the operation result from (d).

• Function without EN/ENO (DWORD_TO_INT)

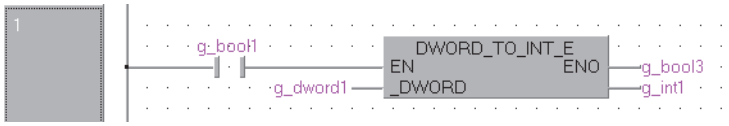
[Structured ladder/FBD]

[ST]

```
g_int1:= DWORD_TO_INT(g_dword1);
```

• Function with EN/ENO (DWORD_TO_INT_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := DWORD_TO_INT_E(g_bool1, g_dword1, g_int1);
```

■DWORD_TO_DINT(_E)

The program which converts double word (unsigned)/32-bit string type data input to (s) into double word (signed) type data, and outputs the operation result from (d).

• Function without EN/ENO (DWORD_TO_DINT)

[Structured ladder/FBD]

[ST]

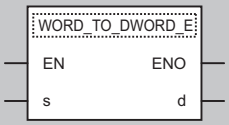
```
g_dint1 := DWORD_TO_DINT(g_dword1);
```

Converting word (unsigned)/16-bit string type to double word (unsigned)/32-bit string type

WORD_TO_DWORD(_E)

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

Structured ladder/FBD



ST

```
ENO:=WORD_TO_DWORD_E(EN, s, d);
```

The following function(s) can go in the dotted squares.
WORD_TO_DWORD, WORD_TO_DWORD_E

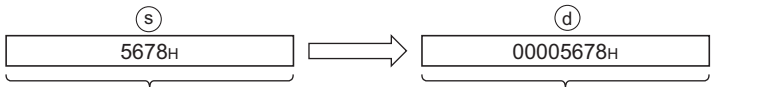
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_WORD)	Input	Word (unsigned)/16-bit string
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Double word (unsigned)/32-bit string

Processing details

■Operation processing

Converts word (unsigned)/16-bit string type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).
After data conversion, high-order 16 bits are filled with 0s.



Word (unsigned)/16-bit string type Double word (unsigned)/32-bit string type

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

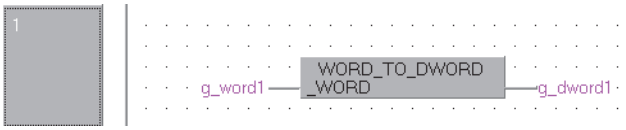
Program example

■WORD_TO_DWORD(_E)

The program which converts word (unsigned)/16-bit string type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).

- Function without EN/ENO (WORD_TO_DWORD)

[Structured ladder/FBD]

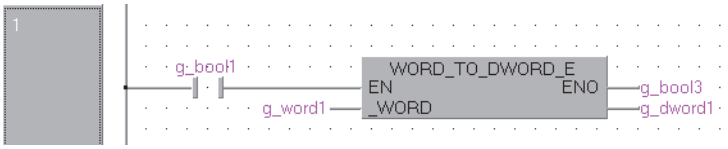


[ST]

```
g_dword1 := WORD_TO_DWORD(g_word1);
```

- Function with EN/ENO (WORD_TO_DWORD_E)

[Structured ladder/FBD]



[ST]

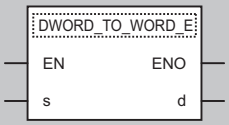
```
g_bool3 := WORD_TO_DWORD_E(g_bool1, g_word1, g_dword1);
```

Converting double word (unsigned)/32-bit string type to word (unsigned)/16-bit string type

DWORD_TO_WORD(_E)

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

Structured ladder/FBD



ST

```
ENO:=DWORD_TO_WORD_E(EN, s, d);
```

The following function(s) can go in the dotted squares.
DWORD_TO_WORD, DWORD_TO_WORD_E

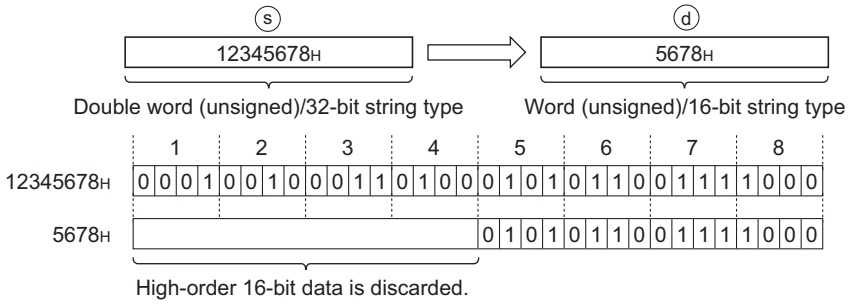
■Argument

Input argument/ Output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_WORD)	Input	Double word (unsigned)/32-bit string
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (unsigned)/16-bit string

Processing details

■Operation processing

Converts double word (unsigned)/32-bit string type data input to (s) into word (unsigned)/16-bit string type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Point

When the DWORD_TO_WORD(_E) function is executed, low-order 16-bit data of double word (unsigned)/32-bit string type data input to (s) are converted into word (unsigned)/16-bit string type data.
High-order 16-bit data are discarded.

Operation error

- No operation error occurs.

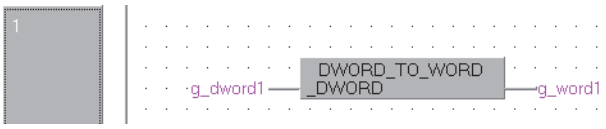
Program example

■DWORD_TO_WORD(_E)

The program which converts double word (unsigned)/32-bit string type data input to (s) into word (unsigned)/16-bit string type data, and outputs the operation result from (d).

- Function without EN/ENO (DWORD_TO_WORD)

[Structured ladder/FBD]

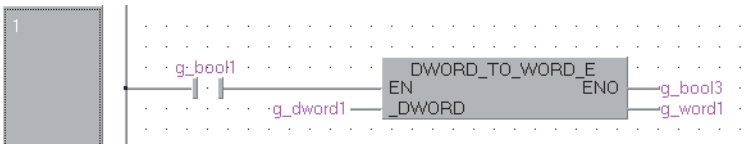


[ST]

```
g_word1 := DWORD_TO_WORD(g_dword1);
```

- Function with EN/ENO (DWORD_TO_WORD_E)

[Structured ladder/FBD]



[ST]

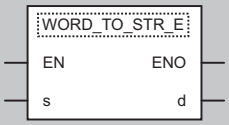
```
g_bool3 := DWORD_TO_WORD_E(g_bool1, g_dword1, g_word1);
```

Converting word (unsigned)/16-bit string, double word (unsigned)/32-bit string type to string type

WORD_TO_STR(_E), DWORD_TO_STR(_E)

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

Structured ladder/FBD



ST

```
ENO:=WORD_TO_STR_E(EN, s, d);
```

The following function(s) can go in the dotted squares.

WWORD_TO_STR, ORD_TO_STR_E, DWORD_TO_STR, DWORD_TO_STR_E

■Argument

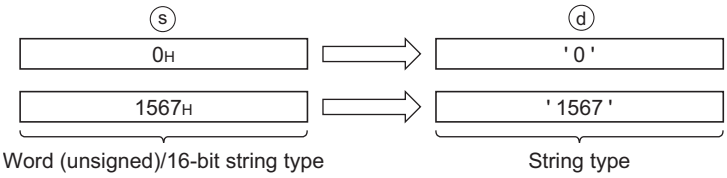
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_WORD, _DWORD)	Input	Word (unsigned)/16-bit string, Double word (unsigned)/32-bit string
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	String(4)/(8)

Processing details

■Operation processing

• WORD_TO_STR, WORD_TO_STR_E

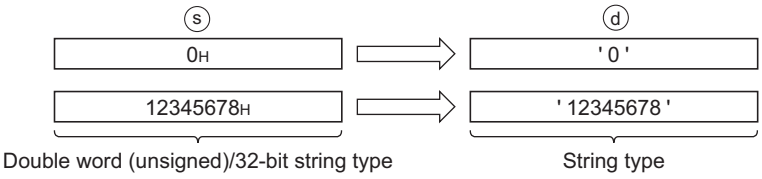
Converts word (unsigned)/16-bit string type data input to (s) into string type data, and outputs the operation result from (d).



When SM701 (signal for switching the number of output characters) is OFF, "00H" is stored to the end of the character string.

• DWORD_TO_STR, DWORD_TO_STR_E

Converts double word (unsigned)/32-bit string type data input to (s) into string type data, and outputs the operation result from (d).



When SM701 (signal for switching the number of output characters) is OFF, "00H" is stored to the end of the character string.

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

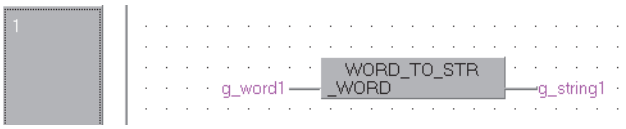
Program example

■WORD_TO_STR(_E)

The program which converts word (unsigned)/16-bit string type data input to (s) into string type data, and outputs the operation result data from (d).

• Function without EN/ENO (WORD_TO_STR)

[Structured ladder/FBD]

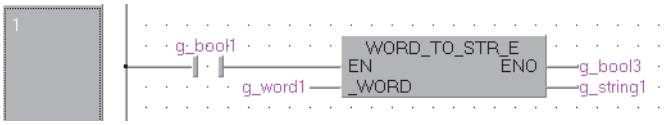


[ST]

```
g_string1 := WORD_TO_STR(g_word1);
```

• Function with EN/ENO (WORD_TO_STR_E)

[Structured ladder/FBD]



[ST]

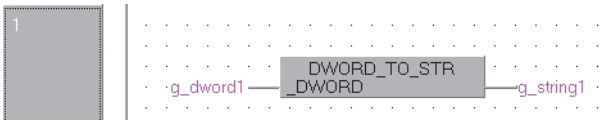
```
g_bool3 := WORD_TO_STR_E(g_bool1, g_word1, g_string1);
```

■DWORD_TO_STR(_E)

The program which converts double word (unsigned)/32-bit string type data input to (s) into string type data, and outputs the operation result data from (d).

• Function without EN/ENO (DWORD_TO_STR)

[Structured ladder/FBD]



[ST]

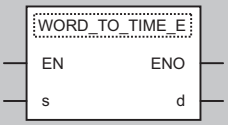
```
g_string1 := DWORD_TO_STR(g_dword1);
```

Converting word (unsigned)/16-bit string, double word (unsigned)/32-bit string type to time type

WORD_TO_TIME(_E), DWORD_TO_TIME(_E)

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

Structured ladder/FBD



ST

```
ENO:= WORD_TO_TIME_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
WORD_TO_TIME, WORD_TO_TIME_E, DWORD_TO_TIME, DWORD_TO_TIME_E

■Argument

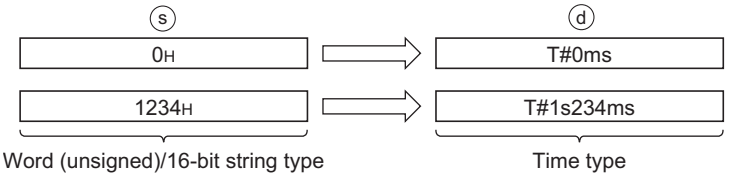
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_WORD, _DWORD)	Input	Word (unsigned)/16-bit string, Double word (unsigned)/32-bit string
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Time

Processing details

■Operation processing

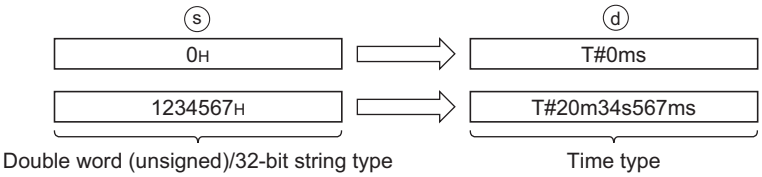
• WORD_TO_TIME, WORD_TO_TIME_E

Converts word (unsigned)/16-bit string type data input to (s) into time type data, and outputs the operation result from (d).



• DWORD_TO_TIME, DWORD_TO_TIME_E

Converts double word (unsigned)/32-bit string type data input to (s) into time type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

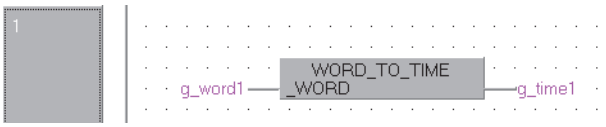
Program example

■WORD_TO_TIME(_E)

The program which converts word (unsigned)/16-bit string type data input to (s) into time type data, and outputs the operation result from (d).

• Function without EN/ENO (WORD_TO_TIME)

[Structured ladder/FBD]

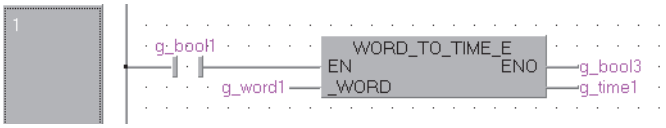


[ST]

```
g_time1 := WORD_TO_TIME(g_word1);
```

• Function with EN/ENO (WORD_TO_TIME_E)

[Structured ladder/FBD]



[ST]

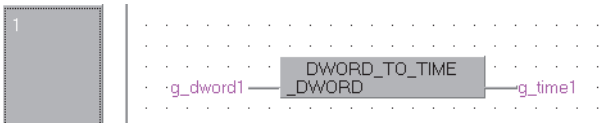
```
g_bool3 := WORD_TO_TIME_E(g_bool1, g_word1, g_time1);
```

■DWORD_TO_TIME(_E)

The program which converts double word (unsigned)/32-bit string type data input to (s) into time type data, and outputs the operation result from (d).

• Function without EN/ENO (DWORD_TO_TIME)

[Structured ladder/FBD]



[ST]

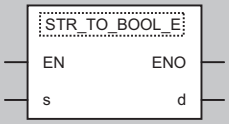
```
g_time1 := DWORD_TO_TIME(g_dword1)
```

Converting string type to bit type

STR_TO_BOOL(_E)

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

Structured ladder/FBD



ST

```
ENO:= STR_TO_BOOL_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
STR_TO_BOOL, STR_TO_BOOL_E

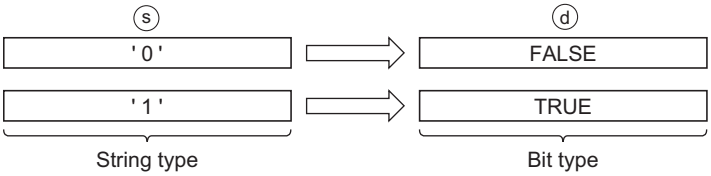
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_STRING)	Input	String (1)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Bit

Processing details

■Operation processing

Converts string type data input to (s) into bit type data, and outputs the operation result from (d).
When the input value is 0, FALSE is output in bit type data.
When the input value is other than 0, TRUE is output in bit type data.



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

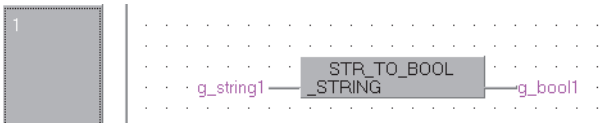
Program example

■STR_TO_BOOL(_E)

The program which converts string type data input to (s) into bit type data, and outputs the operation result from (d).

- Function without EN/ENO (STR_TO_BOOL)

[Structured ladder/FBD]

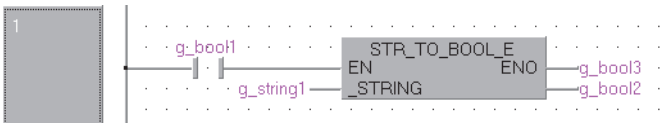


[ST]

```
g_bool1 := STR_TO_BOOL(g_string1);
```

- Function with EN/ENO (STR_TO_BOOL_E)

[Structured ladder/FBD]



[ST]

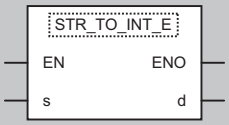
```
g_bool3 := STR_TO_BOOL_E(g_bool1, g_string1, g_bool2);
```

Converting string type to word (signed), double word (signed) type

STR_TO_INT(_E), STR_TO_DINT(_E)

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

Structured ladder/FBD



ST

```
ENO:= STR_TO_INT_E (EN, s, d);
```

The following function(s) can go in the dotted squares.

STR_TO_INT, STR_TO_INT_E, STR_TO_DINT, STR_TO_DINT_E

■Argument

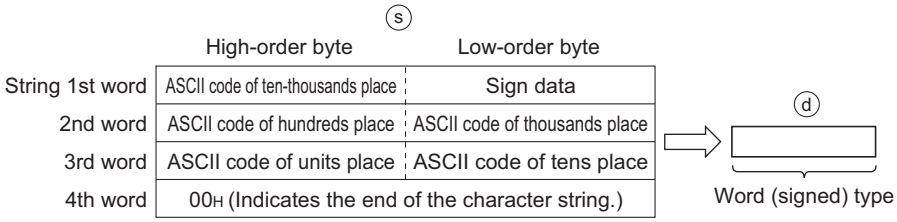
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_STRING)	Input	String(6)/(11)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (signed), double word (signed)

Processing details

■Operation processing

• STR_TO_INT, STR_TO_INT_E

Converts string type data input to (s) into word (signed) type data, and outputs the operation result from (d).



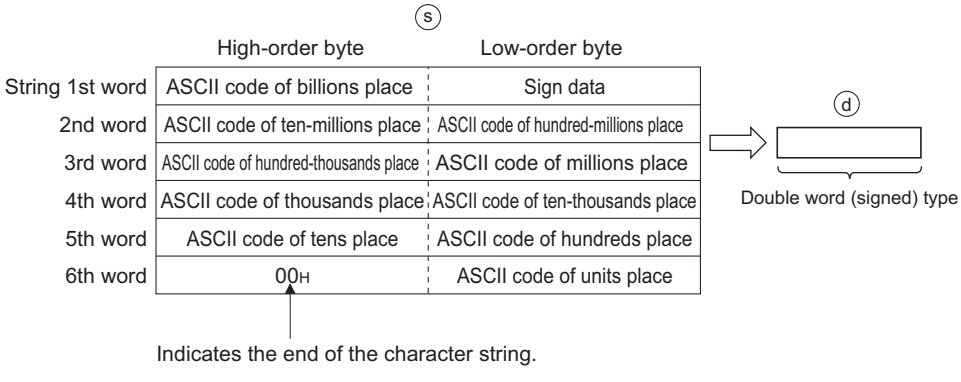
The value to be input to (s) is string type data within the following range.

ASCII code: '30H' to '39H', '20H', '2DH', and '00H'

String type data: '-32768' to '32767'

• STR_TO_DINT, STR_TO_DINT_E

Converts string type data input to (s) into double word (signed) type data, and outputs the operation result from (d).



The value to be input to (s) is string type data within the following range.
ASCII code: '30H' to '39H', '20H', '2DH', and '00H'
String type data: -2147483648 to 2147483647

■Operation result

• Function without EN/ENO

The following table shows the operation results.

Operation result	(d)
No operation error	Operation output value
Operation error	Undefined value

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• An operation error occurs in the following cases.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The input value is other than '30H' to '39H', '20H', '2DH', and '00H' of ASCII code. When the STR_TO_INT(_E) instruction is used, the input value is outside the range of -32768 to 32767. When the STR_TO_DINT(_E) instruction is used, the input value is outside the range of -2147483648 to 2147483647.	—	○	○	○	○	○

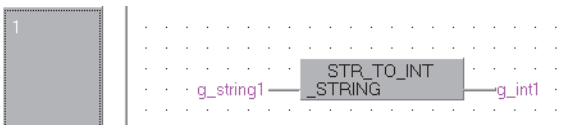
Program example

■STR_TO_INT(_E)

The program which converts string type data input to (s) into word (signed) type data, and outputs the operation result from (d).

• Function without EN/ENO (STR_TO_INT)

[Structured ladder/FBD]

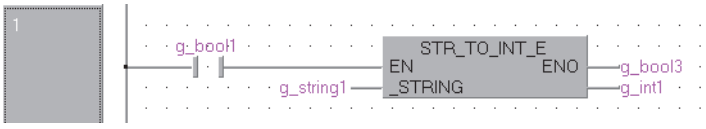


[ST]

```
g_int1 := STR_TO_INT(g_string1);
```

• Function with EN/ENO (STR_TO_INT_E)

[Structured ladder/FBD]



[ST]

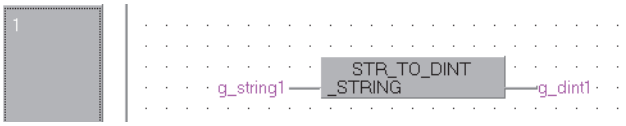
```
g_bool3 := STR_TO_INT_E(g_bool1, g_string1, g_int1);
```

■STR_TO_DINT(_E)

The program which converts string type data input to (s) into double word (signed) type data, and outputs the operation result from (d).

- Function without EN/ENO (STR_TO_DINT)

[Structured ladder/FBD]



[ST]

```
g_dint1 := STR_TO_DINT(g_string1);
```

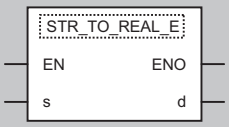
Converting string type to single-precision real type

STR_TO_REAL(_E)

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

This function is used in the Basic model QCPU with a serial number (first five digits) of "04122" or later.

Structured ladder/FBD



ST

```
ENO:= STR_TO_REAL_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
STR_TO_REAL, STR_TO_REAL_E

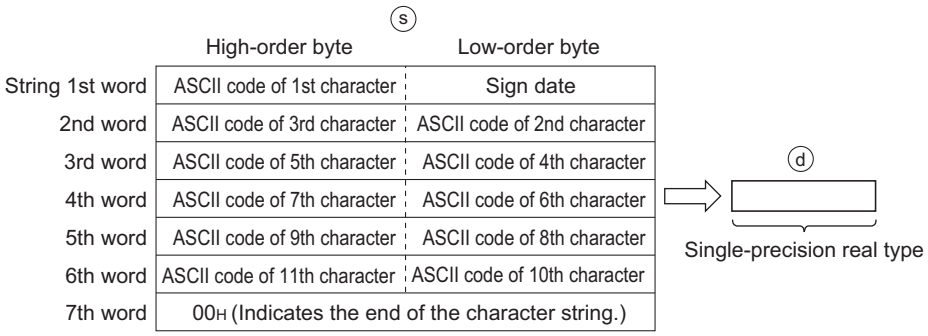
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_STRING)	Input	String (24)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Single-precision real number

Processing details

■Operation processing

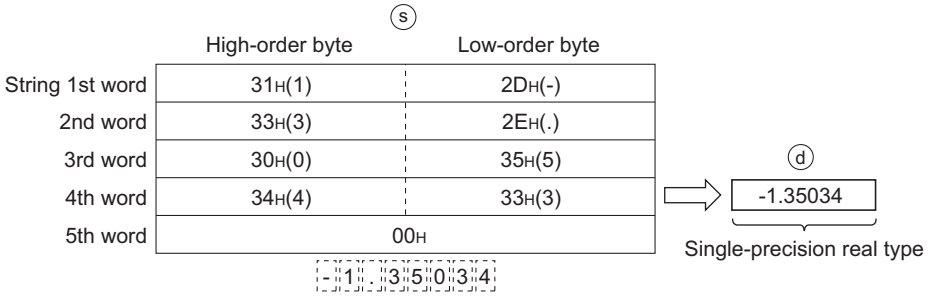
- Converts string type (decimal form/exponential form) data input to (s) into single-precision real type data, and outputs the operation result from (d).



• Both string type data in decimal form and exponential form can be converted to single-precision real type data.

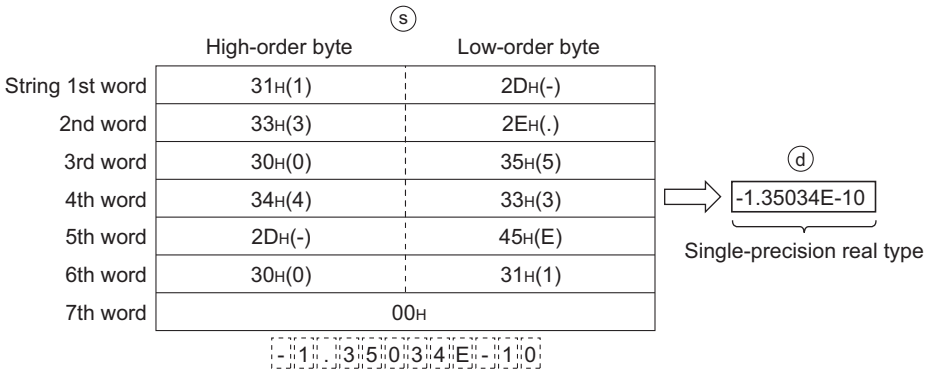
Ex.

Decimal form



Ex.

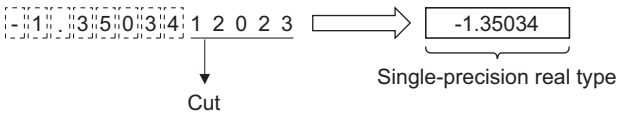
Exponential form



• As the number of significant figures of string type data is 6, the 7th and later digits excluding the sign, decimal point, and exponent part are cut and converted.

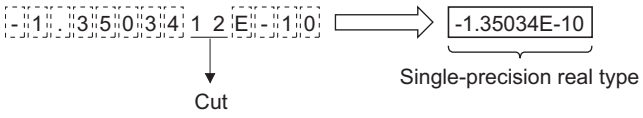
Ex.

Decimal form



Ex.

Exponential form



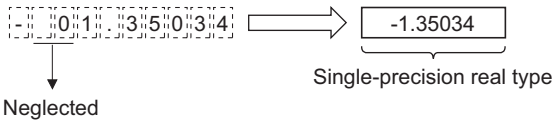
• When a sign is not specified or '2BH' (+) is specified for a sign in decimal form, string type data is converted as a positive value. When '2DH' (-) is specified for a sign, string type data is converted as a negative value.

• When a sign is not specified or '2BH' (+) is specified for a sign of the exponent part in exponential form, string type data is converted as a positive value. When '2DH' (-) is specified for a sign of the exponential part, string type data is converted as a negative value.

• When '20H' (space) or '30H' (0) exists before the first 0 in string type data, the conversion is executed ignoring '20H' and '30H'.

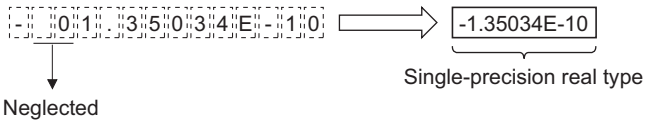
Ex.

Decimal form

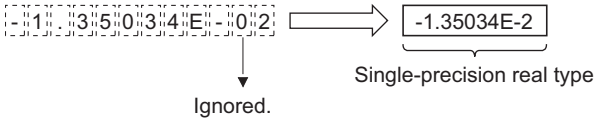


Ex.

Exponential form



• When '30H (0)' exists between 'E' and a numeric value in string type data (exponential form), the conversion is executed ignoring '30H'.



• When '20H' (space) exists in the character string, the conversion is executed ignoring '20H'.

• String type data can contain up to 24 characters. '20H' (space) and '30H' (0) in the character string are counted as one character.

• The value to be input to (s) is string type data within the following range. ASCII code: '30H' to '39H', '45H', '2BH', '2DH', '2EH', '20H', and '00H'

■Operation result

• Function without EN/ENO

The following table shows the operation results.

Operation result	(d)
No operation error	Operation output value
Operation error	Undefined value

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• An operation error occurs in the following cases.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	Any characters other than '30H' to '39H' exist in the integer or fractional part. Two or more "2EH"(.) exist. Any characters other than "45H"(E), "2BH"(+), or "2DH"(-) exist in the exponent part, or more than one exponent parts exist. The data after conversion is outside the range of -3.40282^{+38} to -1.17549^{-38}, 0 or 1.17549^{-38} to 3.40282^{+38} The number of characters is 0 or exceeding 24.	○	○	○	○	○	○

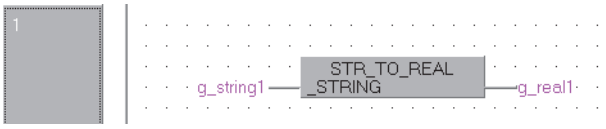
Program example

■STR_TO_REAL(_E)

The program which converts string type data input to (s) into single-precision real type data, and outputs the operation result from (d).

- Function without EN/ENO (STR_TO_REAL)

[Structured ladder/FBD]

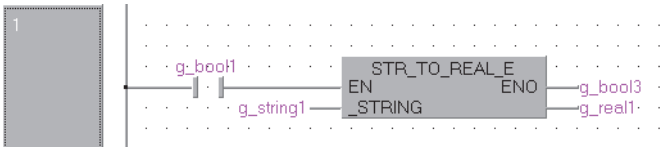


[ST]

```
g_real1 := STR_TO_REAL(g_string1);
```

- Function with EN/ENO (STR_TO_REAL_E)

[Structured ladder/FBD]



[ST]

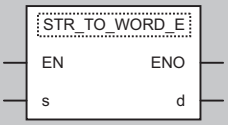
```
g_bool3 := STR_TO_REAL_E(g_bool1, g_string1, g_real1);
```

Converting string type to word (unsigned)/16-bit string, double word (unsigned)/32-bit string type

STR_TO_WORD(_E), STR_TO_DWORD(_E)

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

Structured ladder/FBD



ST

```
ENO:= STR_TO_WORD_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
STR_TO_WORD, STR_TO_WORD_E, STR_TO_DWORD, STR_TO_DWORD_E

■Argument

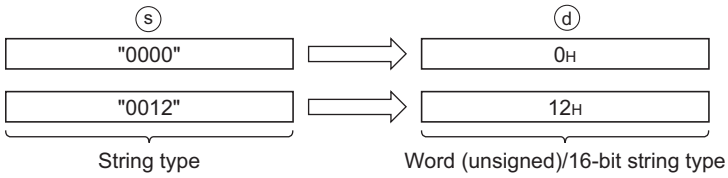
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_STRING)	Input	String(4)/(8)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (unsigned)/16-bit string, Double word (unsigned)/32-bit string

Processing details

■Operation processing

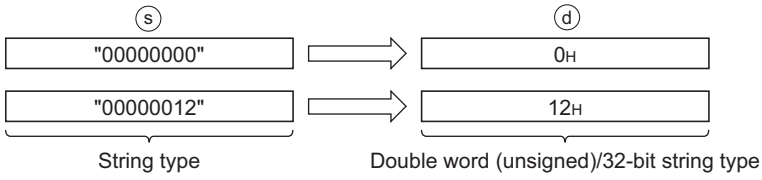
• STR_TO_WORD, STR_TO_WORD_E

Converts string type data input to (s) into word (unsigned)/16-bit string type data, and outputs the operation result from (d).



• STR_TO_DWORD, STR_TO_DWORD_E

Converts the string type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

These functions consist of the following instructions.

STR_TO_WORD(_E): HABIN

STR_TO_DWORD(_E): DHABIN

• 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 number specified for (s) is outside the range of 30H to 39H, 41H to 46H.	—	○	○	○	○	○
4101	The device specified for (s) exceeds the corresponding device range.	—	—	—	—	○	○

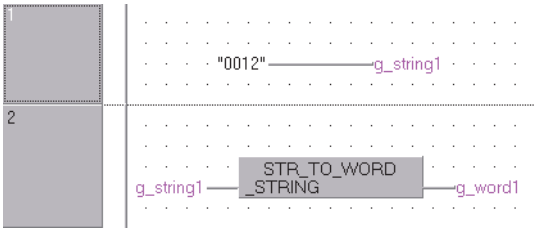
Program example

■STR_TO_WORD(_E)

The program which converts string type data input to (s) into word (unsigned)/16-bit string type data, and outputs the converted data from (d).

• Function without EN/ENO (STR_TO_WORD)

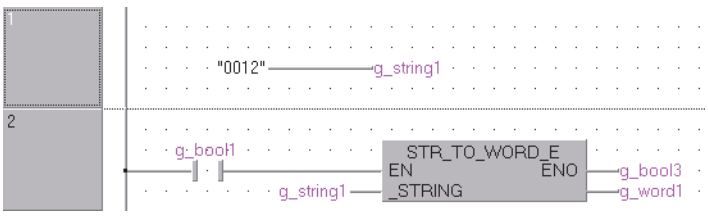
[Structured ladder/FBD]




```
[ST]
g_string1 := "0012"
g_word1 := STR_TO_WORD(g_string1);
```

• Function with EN/ENO (STR_TO_WORD_E)

[Structured ladder/FBD]



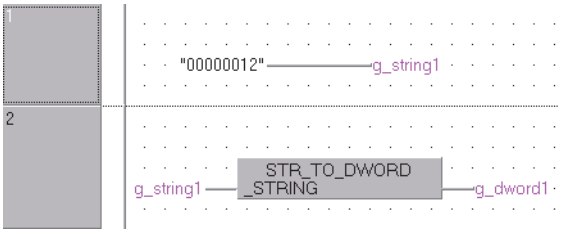

```
[ST]
g_string1 := "0012";
g_bool3 := STR_TO_WORD_E(g_bool1, g_string1, g_word1);
```

■STR_TO_DWORD(_E)

The program which converts string type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).

- Function without EN/ENO (STR_TO_DWORD)

[Structured ladder/FBD]



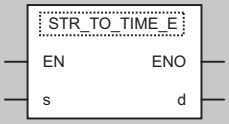

```
[ST]
g_string1 := "00000012";
g_dword1 := STR_TO_DWORD(g_string1);
```

Converting string type to time type

STR_TO_TIME(_E)

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

Structured ladder/FBD



ST

```
ENO:= STR_TO_TIME_E (EN, s, d);
```

The following function(s) can go in the dotted squares.

STR_TO_TIME, STR_TO_TIME_E

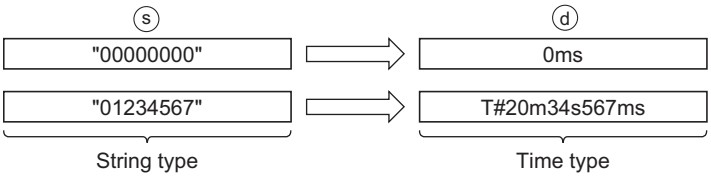
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_STRING)	Input	String (11)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Time

Processing details

■Operation processing

Converts string type data input to (s) into time type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

This function consists of the following instruction.

STR_TO_TIME(_E): DDABIN

- 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 number specified for (s) is outside the range of 30H to 39H, 20H, and 00H. When ASCII data specified for (s) is other than -2147483648 to 4147483647.	—	○	○	○	○	○

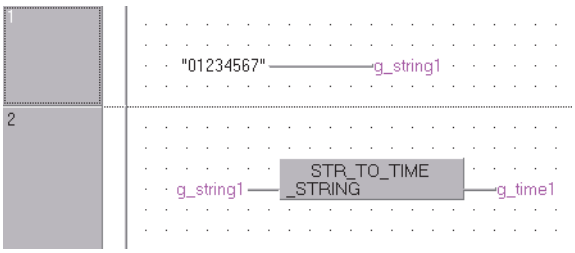
Program example

■STR_TO_TIME(_E)

The program which converts string type data input to (s) into time type data, and outputs the operation result from (d).

- Function without EN/ENO (STR_TO_TIME)

[Structured ladder/FBD]

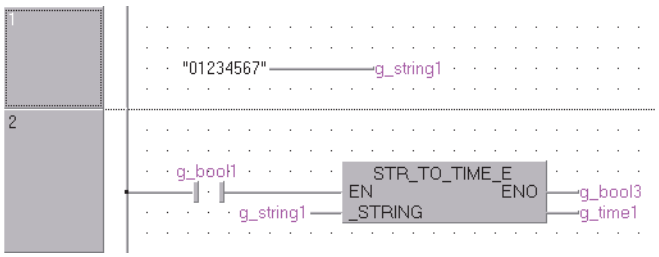


[ST]

```
g_string1 := "01234567";
g_time1 := STR_TO_TIME(g_string1);
```

- Function with EN/ENO (STR_TO_TIME_E)

[Structured ladder/FBD]



[ST]

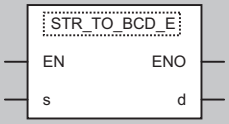
```
g_string1 := "01234567";
g_bool3 := STR_TO_TIME_E(g_bool1, g_string1, g_time1);
```

Converting string type to BCD type

STR_TO_BCD(_E)

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

Structured ladder/FBD



ST

```
ENO:= STR_TO_BCD_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
STR_TO_BCD, STR_TO_BCD_E

■Argument

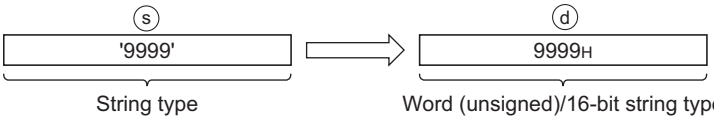
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_STRING)	Input	String (8)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY_BIT

Processing details

■Operation processing

• When word (unsigned)/16-bit string type is specified for output argument.

Converts string type 4-character-string data input to (s) into BCD type data, and outputs the operation result from (d).

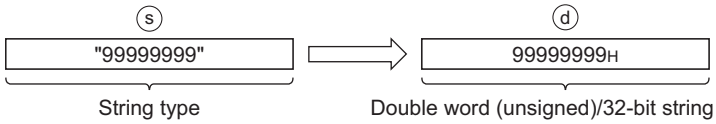


When '20H' (space) exists in the character string, the conversion is executed ignoring '20H'.
'20H' (space) and '30H' (0) in the character string are counted as one character.
The value to be input to (s) is string type data within the following range.
ASCII code: '30H' to '39H', '20H', and '00H'
When input character string has less than 4 letters, convert it with 4 letters supplementing with 0 to the end of the character string. Therefore, when converting character string ("0001" for "1") with less than 4 letters to BCD data, input the zero padding character strings.
When the character string has more than 4 letters, the conversion target is the forth character from the left of the character string data.

Entered character string	Converted character string	Output (BCD type)
"1"	"1000"	1000H(4096)
"12"	"1200"	1200H(4608)
"123"	"1230"	1230H(4656)
"1234"	"1234"	1230H(4656)
"12345"	"1234"	1230H(4656)

• When double word (unsigned)/32-bit string type is specified for output argument.

Converts string type 8-character-string data input to (s) into BCD type data, and outputs the operation result from (d).



When '20H' (space) exists in the character string, the conversion is executed ignoring '20H'.

'20H' (space) and '30H' (0) in the character string are counted as one character.

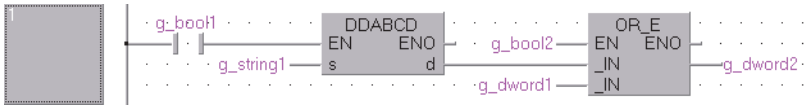
The value to be input to (s) is string type data within the following range.

ASCII code: '30H' to '39H', '20H', and '00H'

• Word (unsigned)/16-bit string, double word (unsigned)/32-bit string type can be specified for (d). Bit type cannot be specified.

Point

Output from (d) cannot be used with connecting to input of function and operator in double word (unsigned)/32-bit string type. In this case, use the DDABCD instruction.



■Operation result

• Function without EN/ENO

The following table shows the operation results.

Operation result	(d)
No operation error	Operation output value
Operation error	Undefined value

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• An operation error occurs in the following cases.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The input character string is outside the range of ASCII code '30H' to '39H', "20H", and "00H".	—	○	○	○	○	○

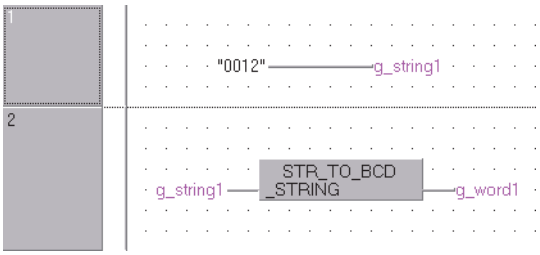
Program example

■STR_TO_BCD(_E) (when word (unsigned)/16-bit string is specified for (d))

The program which converts string type data input to (s) into BCD type data, and outputs the operation result from (d).

- Function without EN/ENO (STR_TO_BCD)

[Structured ladder/FBD]

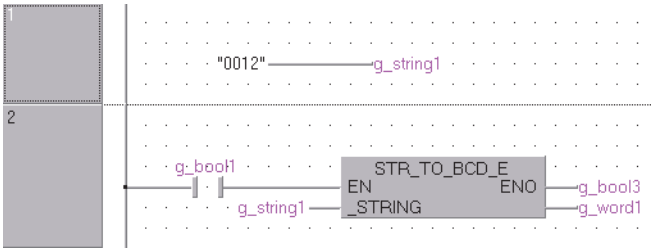


[ST]

```
g_string1 := "0012";
g_word1 := STR_TO_BCD(g_string1);
```

- Function without EN/ENO (STR_TO_BCD_E)

[Structured ladder/FBD]



[ST]

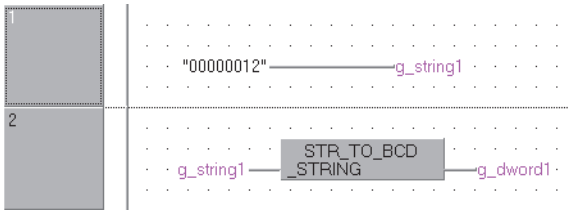
```
g_string1 := "0012";
g_bool3 := STR_TO_BCD_E(g_bool1, g_string1, g_word1);
```

■STR_TO_BCD(_E) (when double word (unsigned)/32-bit string is specified for (d))

The program which converts string type data input to (s) into BCD type data in double word (unsigned)/32-bit string type data, and outputs the operation result from (d).

- Function without EN/ENO (STR_TO_BCD)

[Structured ladder/FBD]

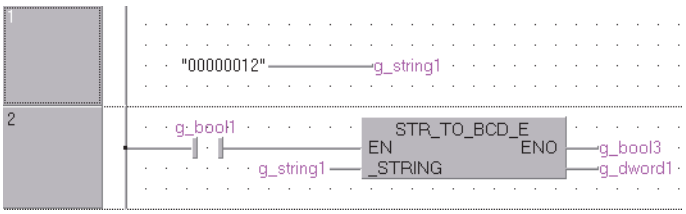


[ST]

```
g_string1:= "00000012";
g_dword1:= STR_TO_BCD(g_string1);
```

- Function without EN/ENO (STR_TO_BCD_E)

[Structured ladder/FBD]



[ST]

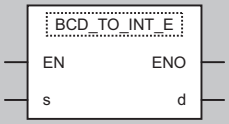
```
g_string1:= "00000012";
g_bool3:= STR_TO_BCD_E(g_bool1, g_string1, g_dword1);
```

Converting BCD type to word (signed), double word (signed) type

BCD_TO_INT(_E), BCD_TO_DINT(_E)

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

Structured ladder/FBD



ST

```
ENO:= BCD_TO_INT_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
BCD_TO_INT, BCD_TO_INT_E, BCD_TO_DINT, BCD_TO_DINT_E

■Argument

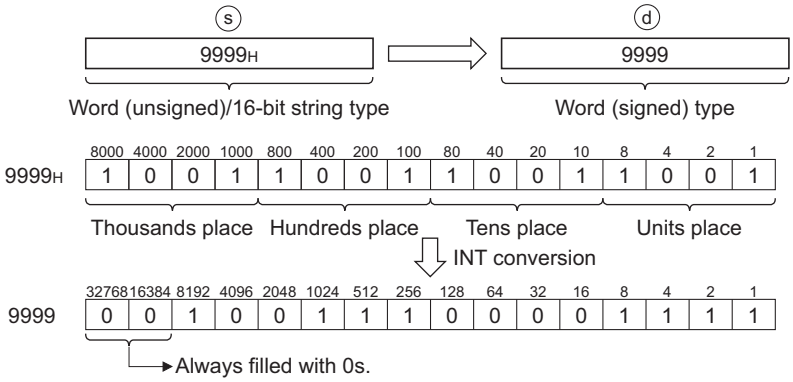
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_BCD)	Input	Word (unsigned)/16-bit string, Double word (unsigned)/32-bit string
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (signed), double word (signed)

Processing details

■Operation processing

• BCD_TO_INT, BCD_TO_INT_E

Converts BCD type data input to (s) into word (signed) type data, and outputs the operation result from (d).

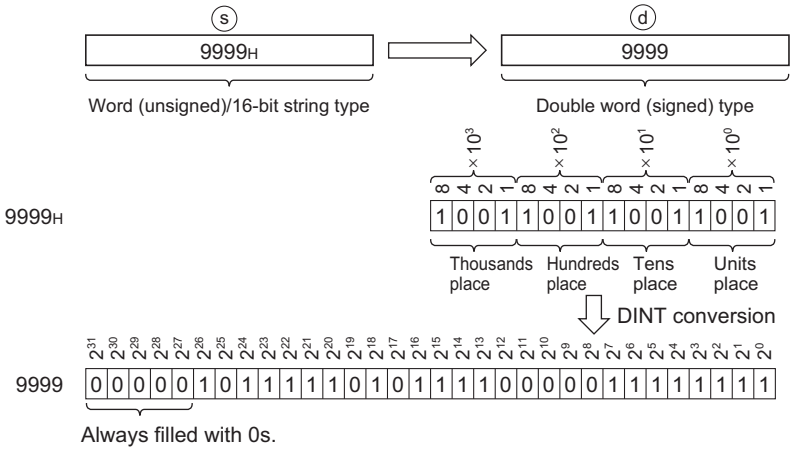


The value to be input to (s) is word (unsigned)/16-bit string type data within the range from 0H to 9999H (0 to 9 for each digit).

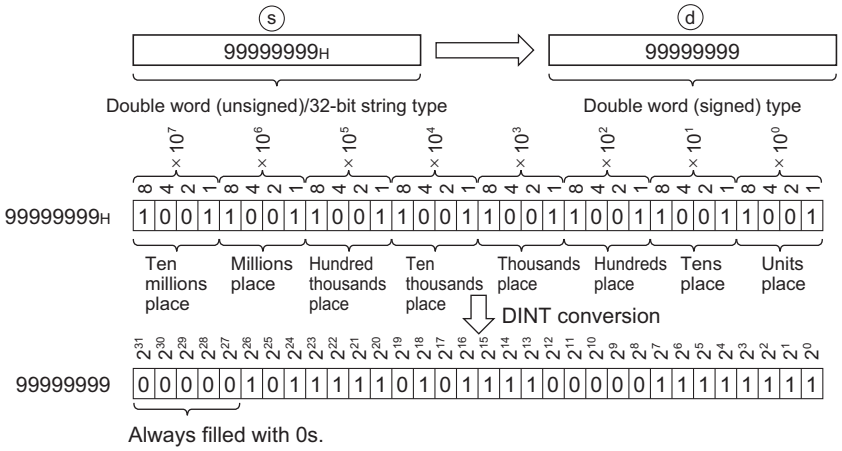
• BCD_TO_DINT, BCD_TO_DINT_E

Converts BCD type data input to (s) into double word (signed) type data, and outputs the operation result from (d).

• When word (unsigned)/16-bit string is specified for (s)



• When double word (unsigned)/32-bit string is specified for (s)



The value to be input to (s) is word (unsigned)/16-bit string type data within the range from 0H to 9999H (0 to 9 for each digit), double word (unsigned)/32-bit string type data within the range from 0H to 99999999H (0 to 9 for each digit).

Word (unsigned)/16-bit string, double word (unsigned)/32-bit string type can be specified for (d). Bit type cannot be specified.

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

These functions consist of the following common instructions.
BCD_TO_INT(_E): BIN
BCD_TO_DINT(_E): BIN, WAND

- 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	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.

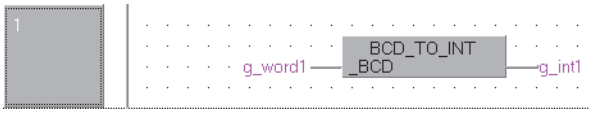
Program example

■BCD_TO_INT(_E)

The program which converts BCD type data input to (s) into word (signed) type data, and outputs the operation result from (d).

- Function without EN/ENO (BCD_TO_INT)

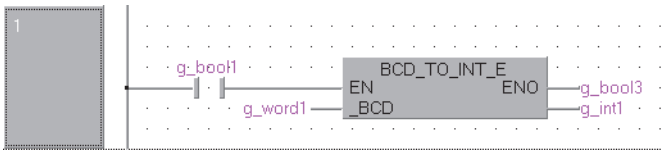
[Structured ladder/FBD]




```
[ST]
g_int1 := BCD_TO_INT(g_word1);
```

- Function with EN/ENO (BCD_TO_INT_E)

[Structured ladder/FBD]



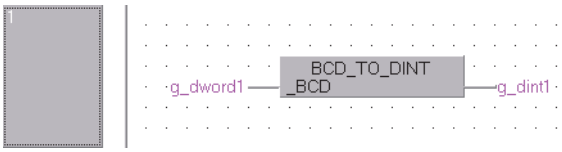

```
[ST]
g_bool3 := BCD_TO_INT_E(g_bool1, g_word1, g_int1);
```

■BCD_TO_DINT(_E)

The program which converts BCD type data input to (s) into double word (signed) type data, and outputs the operation result from (d).

- Function without EN/ENO (BCD_TO_DINT)

[Structured ladder/FBD]



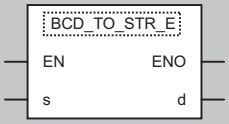

```
[ST]
g_dint1 := BCD_TO_DINT(g_dword1);
```

Converting BCD type to string type

BCD_TO_STR(_E)

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

Structured ladder/FBD



ST

```
ENO:= BCD_TO_STR_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
BCD_TO_STR, BCD_TO_STR_E

■Argument

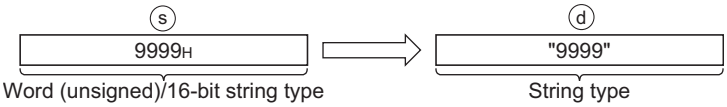
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_BCD)	Input	ANY_BIT
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	String (8)

Processing details

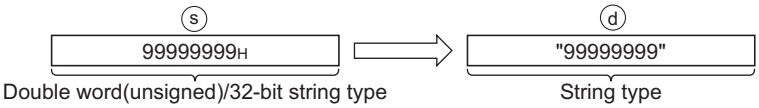
■Operation processing

• Converts BCD type data input to (s) into string type data, and outputs the operation result from (d).

When word (unsigned)/16-bit string type is specified for (s).



When double word (unsigned)/32-bit string type is specified for (s).



• Word (unsigned)/16-bit string type, double word (unsigned)/32-bit string type data can be specified for (d). Bit type cannot be specified.
• When SM701 (signal for switching the number of output characters) is OFF, "00H" is stored to the end of the character string.

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

These functions consist of the following common instructions.
When word (unsigned)/16-bit string type is specified for (s): BCDDA
When double word (unsigned)/32-bit string type is specified for (d): DBCDDA

- 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 word (unsigned)/16-bit string type is specified for (s), (s) is outside the range of 0 to 9999. When double word (unsigned)/32-bit string type is specified for (s), (s) is outside the range of 0 to 99999999.	—	○	○	○	○	○
4101	The device specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

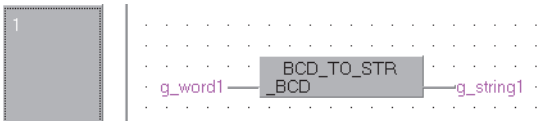
Program example

■BCD_TO_STR(_E) (when word (unsigned)/16-bit string is specified for (s))

The program which converts word (unsigned)/16-bit string type data input to (s) into string type data, and outputs the operation result from (d).

- Function without EN/ENO (BCD _TO_STR)

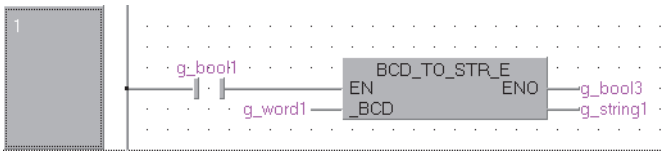
[Structured ladder/FBD]




```
[ST]
g_string1 := BCD_TO_STR(g_word1);
```

- Function with EN/ENO (BCD_TO_STR_E)

[Structured ladder/FBD]



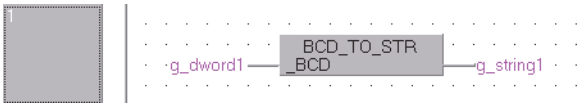

```
[ST]
g_bool3 := BCD_TO_STR_E(g_bool1, g_word1, g_string1);
```

■BCD_TO_STR(_E) (when double word (unsigned)/32-bit string is specified for (s))

The program which converts double word (unsigned)/32-bit string type data input to (s) into string type data, and outputs the operation result from (d).

- Function without EN/ENO (BCD _TO_STR)

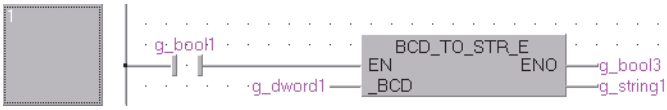
[Structured ladder/FBD]




```
[ST]
g_string1 := BCD_TO_STR(g_dword1);
```

- Function with EN/ENO (BCD_TO_STR_E)

[Structured ladder/FBD]



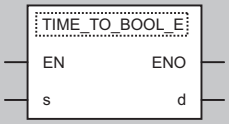

```
[ST]
g_bool3 := BCD_TO_STR_E(g_bool1, g_dword1, g_string1);
```

Converting time type to bit type

TIME_TO_BOOL(_E)

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

Structured ladder/FBD



ST

```
ENO:= TIME_TO_BOOL_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
TIME_TO_BOOL, TIME_TO_BOOL_E

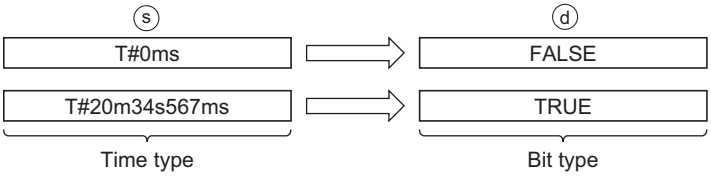
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_TIME)	Input	Time
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Bit

Processing details

■Operation processing

Converts time type data input to (s) into bit type data, and outputs the operation result from (d).
When the input value is 0ms, FALSE is output in bit type data.
When the input value is other than 0ms, TRUE is output in bit type data.



■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

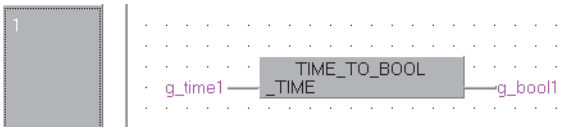
Program example

■TIME_TO_BOOL(_E)

The program which converts time type data input to (s) into bit type data, and outputs the operation result from (d).

- Function without EN/ENO (TIME_TO_BOOL)

[Structured ladder/FBD]

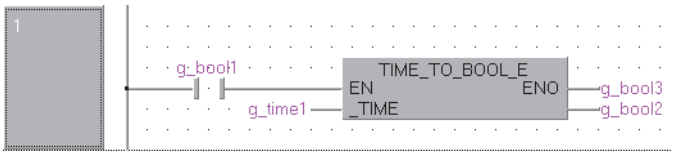


[ST]

```
g_bool1 := TIME_TO_BOOL(g_time1);
```

- Function with EN/ENO (TIME_TO_BOOL_E)

[Structured ladder/FBD]



[ST]

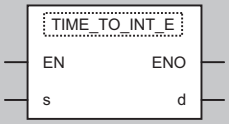
```
g_bool3 := TIME_TO_BOOL_E(g_bool1, g_time1, g_bool2);
```

Converting time type to word (signed), double word (signed) type

TIME_TO_INT(_E), TIME_TO_DINT(_E)

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

Structured ladder/FBD



ST

```
ENO:= TIME_TO_INT_E (EN, s, d);
```

The following function(s) can go in the dotted squares.

TIME_TO_INT, TIME_TO_INT_E, TIME_TO_DINT, TIME_TO_DINT_E

■Argument

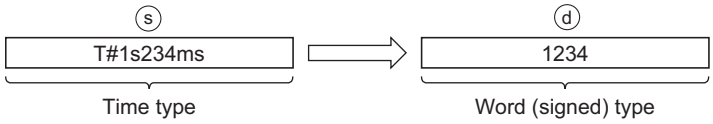
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_TIME)	Input	Time
Output argument	ENO	Execution result (TRUE: Normal execution, FALSE: Error or stop)	Bit
	d	Output	Word (signed), double word (signed)

Processing details

■Operation processing

• TIME_TO_INT, TIME_TO_INT_E

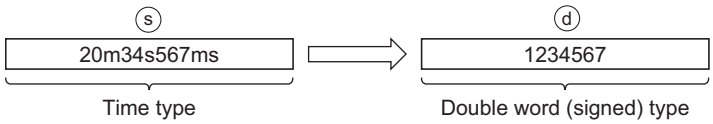
Converts time type data input to (s) into word (signed) type data, and outputs the operation result from (d).



When converting to word (signed) type data, high-order 16-bit (1 word) data of time type is discarded.

• TIME_TO_DINT, TIME_TO_DINT_E

Converts time type data input to (s) into double word (signed) type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

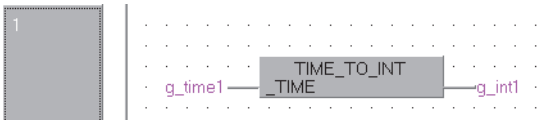
Program example

■TIME_TO_INT(_E)

The program which converts time type data input to (s) into word (signed) type data, and outputs the operation result from (d).

- Function without EN/ENO (TIME _TO_INT)

[Structured ladder/FBD]

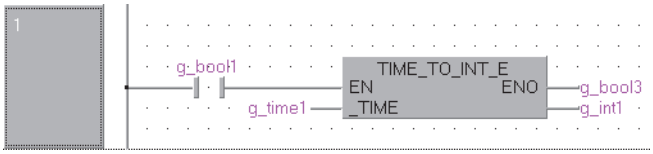


[ST]

```
g_int1 := TIME_TO_INT(g_time1);
```

- Function with EN/ENO (TIME_TO_INT_E)

[Structured ladder/FBD]



[ST]

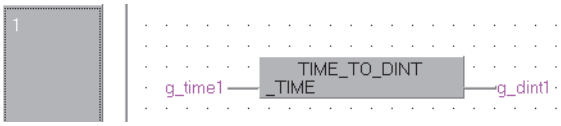
```
g_bool3 := TIME_TO_INT_E(g_bool1, g_time1, g_int1);
```

■TIME_TO_DINT(_E)

The program which converts time type data input to (s) into double word (signed) type data, and outputs the operation result from (d).

- Function without EN/ENO (TIME _TO_DINT)

[Structured ladder/FBD]



[ST]

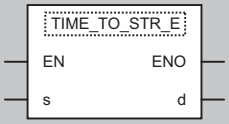
```
g_dint1 := TIME_TO_DINT(g_time1);
```

Converting time type to string type

TIME_TO_STR(_E)

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

Structured ladder/FBD



ST

```
ENO:= TIME_TO_STR_E (EN, s, d);
```

The following function(s) can go in the dotted squares.
TIME_TO_STR, TIME_TO_STR_E

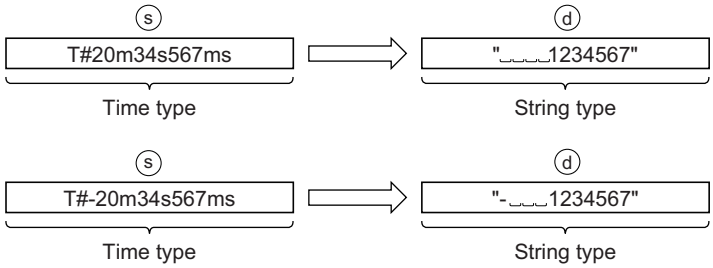
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_BCD)	Input	Time
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	String (11)

Processing details

■Operation processing

- Converts time type data input to (s) into string type data, and outputs the operation result from (d).



- When SM701 (signal for switching the number of output characters) is OFF, "00H" is stored to the end of the character string.
- The operation results stored to (d) are as follows.

For the first character, 20H (space) is stored if the BIN data is positive, and 2DH (-) is stored if it is negative.
20H (space) is stored to the left of significant figures.

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

These functions consist of the following common instructions.

TIME_TO_STR(_E): DBINDA

- 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 specified for (d) exceeds the corresponding device range.	—	—	—	—	○	○

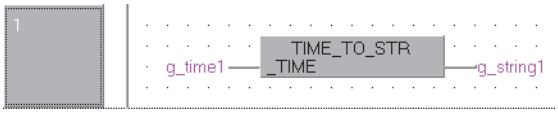
Program example

■TIME_TO_STR(_E)

The program which converts time type data input to (s) into string type data, and outputs the operation result from (d).

- Function without EN/ENO (TIME_TO_STR)

[Structured ladder/FBD]

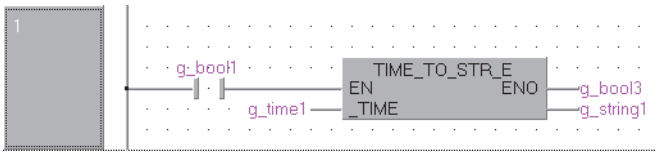


[ST]

```
g_string1 := TIME_TO_STR(g_time1);
```

- Function with EN/ENO (TIME_TO_STR_E)

[Structured ladder/FBD]



[ST]

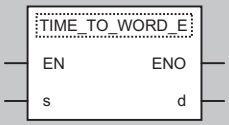
```
g_bool3 := TIME_TO_STR_E(g_bool1, g_time1, g_string1);
```

Converting time type to word (unsigned)/16-bit string, double word (unsigned)/32-bit string type

TIME_TO_WORD(_E), TIME_TO_DWORD(_E)

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

Structured ladder/FBD



ST

```
ENO:=TIME_TO_WORD_E(EN, s, d);
```

The following function(s) can go in the dotted squares.
TIME_TO_WORD, TIME_TO_WORD_E, TIME_TO_DWORD, TIME_TO_DWORD_E

■Argument

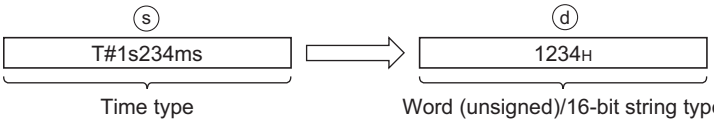
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_TIME)	Input	Time
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Word (unsigned)/16-bit string, double word (unsigned)/32-bit string

Processing details

■Operation processing

• TIME_TO_WORD, TIME_TO_WORD_E

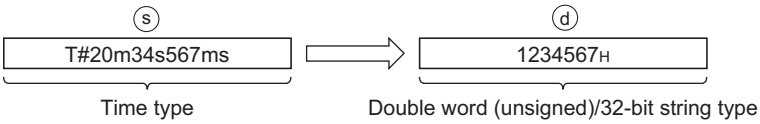
Converts time type data input to (s) into word (unsigned)/16-bit string type data, and outputs the operation result from (d).



• When converting to word (unsigned)/16-bit string type data, high-order 16-bit (1 word) data is discarded.

• TIME_TO_DWORD, TIME_TO_DWORD_E

Converts time type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).



■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

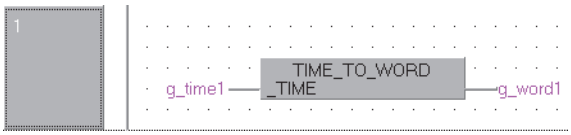
Program example

■TIME_TO_WORD(_E)

The program which converts time type data input to (s) into word (unsigned)/16-bit string type data, and outputs the operation result from (d).

• Function without EN/ENO (TIME _TO_WORD)

[Structured ladder/FBD]

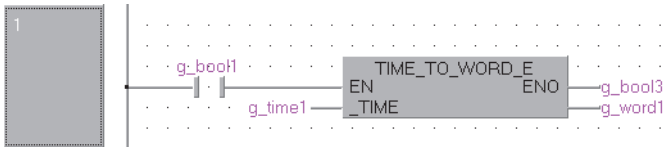


[ST]

```
g_word1 := TIME_TO_WORD(g_time1);
```

• Function with EN/ENO (TIME_TO_WORD_E)

[Structured ladder/FBD]



[ST]

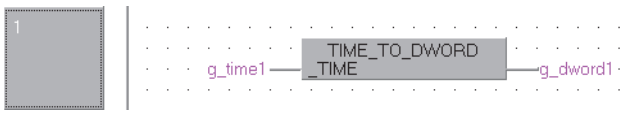
```
g_bool3 := TIME_TO_WORD_E(g_bool1, g_time1, g_word1);
```

■TIME_TO_DWORD(_E)

The program which converts time type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).

• Function without EN/ENO (TIME _TO_DWORD)

[Structured ladder/FBD]



[ST]

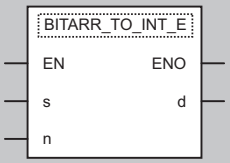
```
g_dword1 := TIME_TO_DWORD(g_time1);
```

Converting bit array to word (signed) type, word (unsigned)/16-bit string type, double word (signed) type, double word (unsigned)/32-bit string type

BITARR_TO_INT(_E), BITARR_TO_DINT(_E)

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

Structured ladder/FBD



ST

```
ENO:= BITARR_TO_INT_E (EN, s, n, d);
```

The following function(s) can go in the dotted squares.
BITARR_TO_INT, BITARR_TO_INT_E, BITARR_TO_DINT, BITARR_TO_DINT_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_BitArr)	Input (Variables are applicable to element specification.)	Bit
	n	Input (Only a constant 4, 8, 12, 16, 20, 24, 28 or 32 can be specified)	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal execution, FALSE: Error or stop)	Bit
	d	Output	ANY16, ANY32

Processing details

■Operation processing

• BITARR_TO_INT, BITARR_TO_INT_E

Converts number of bits specified for n starting from the bit array element input to (s) into word (signed) type or word (unsigned)/16-bit string type data, and outputs the operation result from (d).
Only a constant 4, 8, 12 or 16 can be specified for n.
0 is set for the output bits higher than the specified number of bits.

• BITARR_TO_DINT, BITARR_TO_DINT_E

Converts number of bits specified for n starting from the bit array element input to (s) into double word (signed) type or double word (unsigned)/32-bit string type data, and outputs the operation result from (d).
Only a constant 4, 8, 12, 16, 20, 24, 28 or 32 can be specified for n.
0 is set for the output bits higher than the specified number of bits.

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

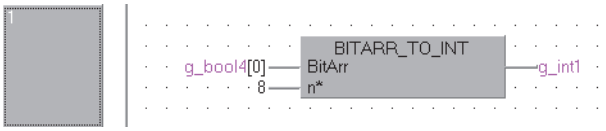
Program example

■BITARR_TO_INT(_E)

The program which converts 8 bits from 0 of bit array input to (s) into word (signed) type data, and outputs the operation result from (d).

- Function without EN/ENO (BITARR_TO_INT)

[Structured ladder/FBD]

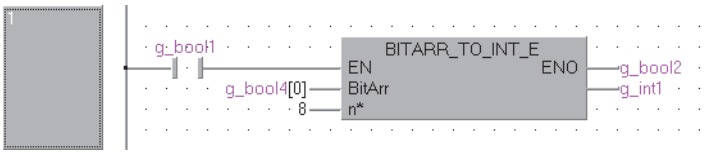


[ST]

```
g_int1 := BITARR_TO_INT(g_bool4[0], 8);
```

- Function with EN/ENO (BITARR_TO_INT_E)

[Structured ladder/FBD]



[ST]

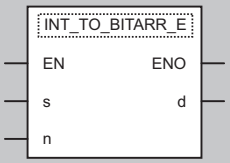
```
g_bool2 := BITARR_TO_INT_E(g_bool1, g_bool4[0], 8, g_int1);
```

Converting word (signed) type, word (unsigned)/16-bit string type, double word (signed) type, double word (unsigned)/32-bit string type to bit array

INT_TO_BITARR(_E), DINT_TO_BITARR(_E)

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

Structured ladder/FBD



ST

```
ENO:= INT_TO_BITARR_E (EN, s, n, d);
```

The following function(s) can go in the dotted squares.

INT_TO_BITARR, INT_TO_BITARR_E, DINT_TO_BITARR, DINT_TO_BITARR_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s	Input	ANY16, ANY32
	n	Input (Only a constant 4, 8, 12, 16, 20, 24, 28 or 32 can be specified)	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal execution, FALSE: Error or stop)	Bit
	d(_BitArr)	Output (Variables are applicable to element specification.)	Bit

Processing details

■Operation processing

• INT_TO_BITARR, INT_TO_BITARR_E

Outputs low-order n bits of word (signed) type or word (unsigned)/16-bit string type data specified for (s) to (d).

Only a constant 4, 8, 12 or 16 can be specified for n.

The output bits higher than the specified number of bits do not change.

• DINT_TO_BITARR, DINT_TO_BITARR_E

Outputs low-order n bits of double word (signed) type or double word (unsigned)/32-bit string type data specified for (s) to (d).

Only a constant 4, 8, 12, 16, 20, 24, 28 or 32 can be specified for n.

The output bits higher than the specified number of bits do not change.

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

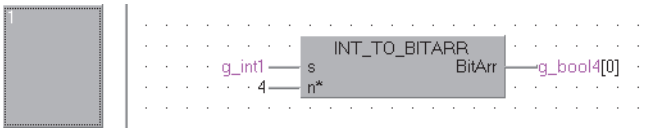
Program example

■INT_TO_BITARR(_E)

The program which outputs low-order 4 bits of word (signed) type data input to (s) to (d).

- Function without EN/ENO (INT_TO_BITARR)

[Structured ladder/FBD]

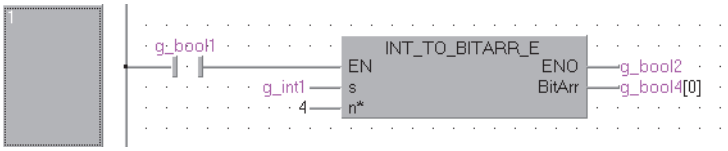


[ST]

```
g_bool4[0] := INT_TO_BITARR(g_int1, 4);
```

- Function with EN/ENO (INT_TO_BITARR_E)

[Structured ladder/FBD]



[ST]

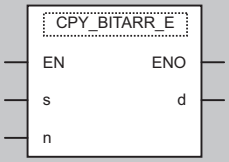
```
g_bool2 := INT_TO_BITARR_E(g_bool1, g_int1, 4, g_bool4[0]);
```

Bit array copy

CPY_BITARR(_E)

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

Structured ladder/FBD



ST

```
ENO:= CPY_BITARR_E (EN, s, n, d);
```

The following function(s) can go in the dotted squares.
CPY_BITARR, CPY_BITARR_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_BitArrIn)	Input	Bit
	n	Input (Only a constant 4, 8, 12, 16, 20, 24, 28 or 32 can be specified)	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal execution, FALSE: Error or stop)	Bit
	d(_BitArrOut)	Output (Variables are applicable to element specification.)	Bit

Processing details

■Operation processing

Outputs n bits of bit array input to (s) to (d).

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

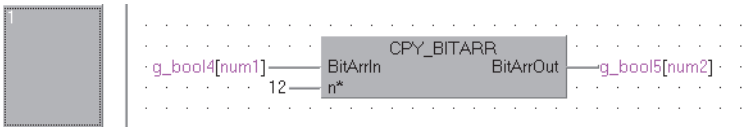
Program example

■CPY_BITARR(_E)

The program which outputs 12 bits from num1 element of bit string input to (s) to num2 and the following bits of (d).

- Function without EN/ENO (CPY_BITARR)

[Structured ladder/FBD]

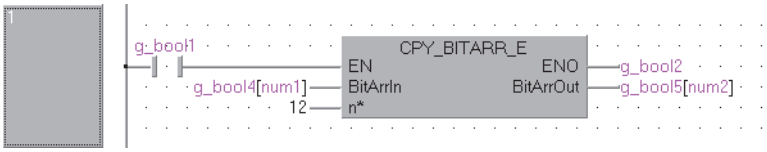


[ST]

```
g_bool5[num2] := CPY_BITARR(g_bool4[num1], 12);
```

- Function with EN/ENO (CPY_BITARR_E)

[Structured ladder/FBD]



[ST]

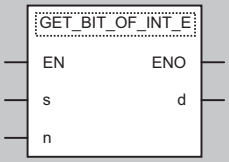
```
g_bool2 := CPY_BITARR_E(g_bool1, g_bool4[num1], 12, g_bool5[num2]);
```

Specified bit read of word (signed) type data

GET_BIT_OF_INT(_E)

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

Structured ladder/FBD



ST

```
ENO:=GET_BIT_OF_INT_E(EN, s, n, d);
```

The following function(s) can go in the dotted squares.
GET_BIT_OF_INT, GET_BIT_OF_INT_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s	Input	Word (signed)
	n	Input (Only a constant between 0 and 15 can be specified)	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal execution, FALSE: Error or stop)	Bit
	d	Output	Bit

Processing details

■Operation processing

Reads a value of nth bit of (s), and outputs the operation result from (d).

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

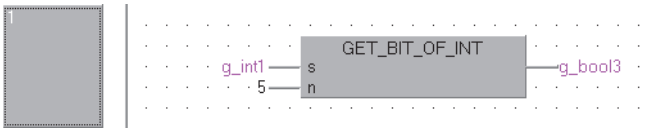
Program example

■GET_BIT_OF_INT(_E)

The program which reads a value of 5th bit of data input to (s), and outputs the operation result from (d).

- Function without EN/ENO (GET_BIT_OF_INT)

[Structured ladder/FBD]

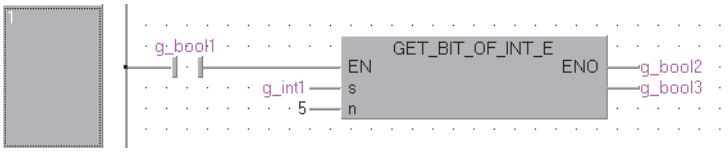


[ST]

```
g_bool3 := GET_BIT_OF_INT(g_int1, 5);
```

- Function with EN/ENO (GET_BIT_OF_INT_E)

[Structured ladder/FBD]



[ST]

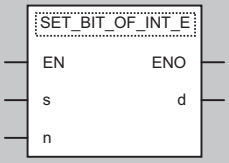
```
g_bool2 := GET_BIT_OF_INT_E(g_bool1, g_int1, 5, g_bool3);
```

Specified bit write of word (signed) type data

SET_BIT_OF_INT(_E)

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

Structured ladder/FBD



ST

```
ENO:= SET_BIT_OF_INT_E (EN, s, n, d);
```

The following function(s) can go in the dotted squares.
SET_BIT_OF_INT, SET_BIT_OF_INT_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s	Input	Bit
	n	Input (Only a constant between 0 and 15 can be specified)	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal execution, FALSE: Error or stop)	Bit
	d	Output	Word (signed)

Processing details

■Operation processing

Writes a value specified for (s) to the nth bit of (d).

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

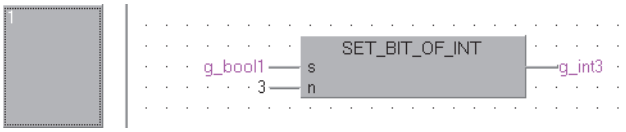
Program example

■SET_BIT_OF_INT(_E)

The program which writes a value specified for (s) to the 3rd bit of (d).

- Function without EN/ENO (SET_BIT_OF_INT)

[Structured ladder/FBD]

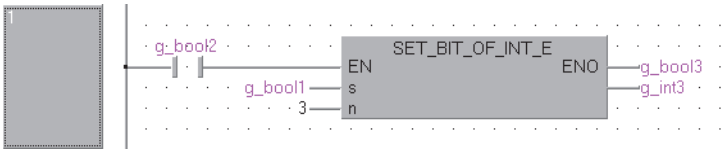


[ST]

```
g_int3 := SET_BIT_OF_INT(g_bool1. 3);
```

- Function with EN/ENO (SET_BIT_OF_INT_E)

[Structured ladder/FBD]



[ST]

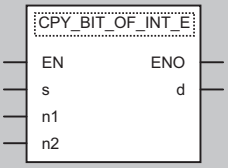
```
g_bool3 := SET_BIT_OF_INT_E(g_bool2, g_bool1, 3, g_int3);
```

Specified bit copy of word (signed) type data

CPY_BIT_OF_INT(_E)

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

Structured ladder/FBD



ST

```
ENO:=CPY_BIT_OF_INT_E(EN, s, n1, n2, d);
```

The following function(s) can go in the dotted squares.

CPY_BIT_OF_INT, CPY_BIT_OF_INT_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s	Input	Word (signed)
	n1	Input (Only a constant between 0 and 15 can be specified)	Word (signed)
	n2	Input (Only a constant between 0 and 15 can be specified)	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal execution, FALSE: Error or stop)	Bit
	d	Output	Word (signed)

Processing details

■Operation processing

Copies a value of (n1)th bit of input (s) to the (n2)th bit of output (d).

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

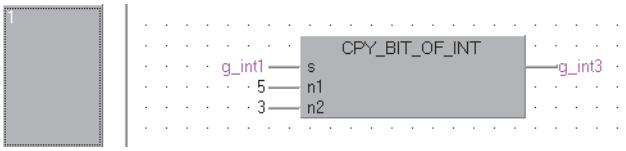
Program example

■CPY_BIT_OF_INT(_E)

The program which writes a value of 5th bit of (s) to the 3rd bit of (d).

- Function without EN/ENO (CPY_BIT_OF_INT)

[Structured ladder/FBD]

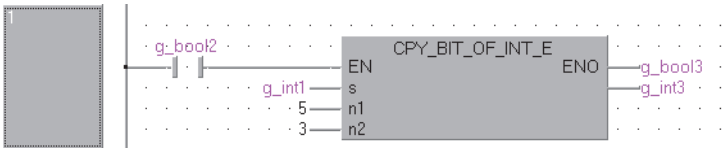


[ST]

```
g_int3 := CPY_BIT_OF_INT(g_int1, 5, 3);
```

- Function with EN/ENO (CPY_BIT_OF_INT_E)

[Structured ladder/FBD]



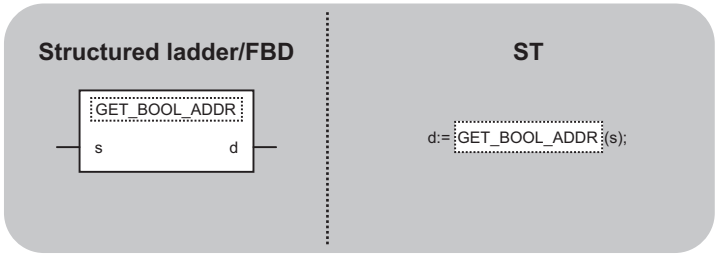
[ST]

```
g_bool3 := CPY_BIT_OF_INT_E(g_bool2, g_int1, 5, 3, g_int3);
```

Nonessential type conversion

GET_BOOL_ADDR, GET_INT_ADDR, GET_WORD_ADDR

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



The following function(s) can go in the dotted squares.
GET_BOOL_ADDR, GET_INT_ADDR, GET_WORD_ADDR

■Argument

Input/output argument	Name	Description	Data type
Input argument	s	Input	ANY
Output argument	d	Output	Bit, word (signed), Word (unsigned)/16-bit string

Processing details

■Operation processing

- Outputs data type of (s) as data type of (d).

Function name	Input data type	Output data type
GET_BOOL_ADDR	Bit Array of bit	Bit
GET_INT_ADDR	Word (signed) Double word (signed) Word (unsigned)/16-bit string Single-precision real number String Time type	Word (signed)
GET_WORD_ADDR	Array of word (signed) Array of double word (signed) Array of word (unsigned)/16-bit string Array of double word (unsigned)/32-bit string Array of real number Array of time type	Word (unsigned)/16-bit string

- Rounding error may occur when specifying the input value to (s) by programming tool. For precautions when setting an input value using a programming tool, refer to the following.

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

■Operation result

An operation is executed and the operation value is output from (d).

Operation error

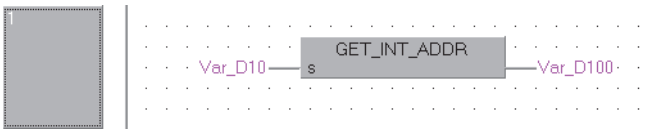
- No operation error occurs.

Program example

■GET_INT_ADDR

The program which directly handles 32-bit input variable Var_D10 as 16-bit input data without the type conversion.

[Structured ladder/FBD]



[ST]

```
Var_D100 := GET_INT_ADDR(Var_D10);
```

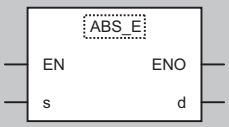
5.2 Standard Functions of One Numeric Variable

Absolute value

ABS(_E)

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

Structured ladder/FBD



ST

```
ENO:= ABS_E (EN, s, d);
```

The following function(s) can go in the dotted squares.

ABS, ABS_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_IN)	Input	ANY_NUM
Output argument	ENO	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	d	Output	ANY_NUM

Processing details

■Operation processing

- Outputs the absolute value of word (signed), double word (signed), single-precision real or double-precision real type data input to (s) from (d) in the same data type as that of (s).

Assuming that the input value is A and the operation output value is B, the relationship is expressed by the following equality.

$B = |A|$

- The value to be input to (s) is word (signed), double word (signed), single-precision real or double-precision real type data.
- When the data type of (s) is word (signed) type and the input value is -32768, -32768 is output from (d).

When the data type of (s) is double word (signed) type and the input value is -2147483648, -2147483648 is output from (d).
(No operation error occurs. In case of ABS_E, TRUE is output from ENO.)

- Rounding error may occur when specifying single-precision real or double-precision real type data to (s) by programming tool.

For precautions when setting an input value using a programming tool, refer to the following.

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

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

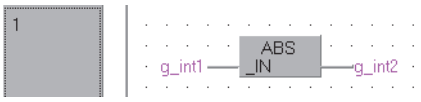
Program example

■ABS(_E) (when word (signed) is specified for (s))

The program which outputs the absolute value of word (signed) type data input to (s) from (d) in the same data type as that of (s).

• Function without EN/ENO (ABS)

[Structured ladder/FBD]

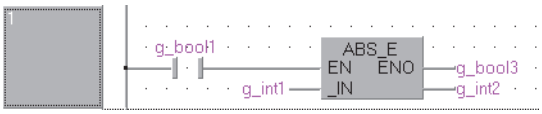


[ST]

```
g_int2:= ABS(g_int1);
```

• Function with EN/ENO (ABS_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := ABS_E(g_bool1, g_int1, g_int2);
```

■ABS(_E) (when double word (signed) is specified for (s))

The program which outputs the absolute value of double word (signed) type data input to (s) from (d) in the same data type as that of (s).

• Function without EN/ENO (ABS)

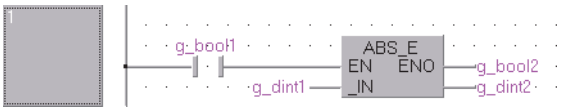
[Structured ladder/FBD]

[ST]

```
g_dint2:= ABS(g_dint1);
```

• Function with EN/ENO (ABS_E)

[Structured ladder/FBD]



[ST]

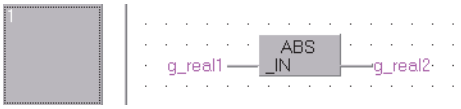
```
g_bool2 := ABS_E(g_bool1, g_dint1, g_dint2);
```

■ABS(_E) (when single-precision real type data is specified for (s))

The program which outputs the absolute value of single-precision real type data input to (s) from (d) in the same data type as that of (s).

- Function without EN/ENO (ABS)

[Structured ladder/FBD]

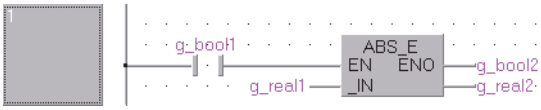


[ST]

```
g_real2:= ABS(g_real1);
```

- Function with EN/ENO (ABS_E)

[Structured ladder/FBD]



[ST]

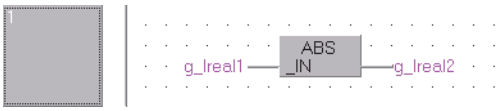
```
g_bool2 := ABS_E(g_bool1, g_real1, g_real2);
```

■ABS(_E) (when double-precision real type data is specified for (s))

The program which outputs the absolute value of double-precision real type data input to (s) from (d) in the same data type as that of (s).

- Function without EN/ENO (ABS)

[Structured ladder/FBD]

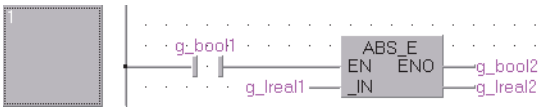


[ST]

```
g_lreal2:= ABS(g_lreal1);
```

- Function with EN/ENO (ABS_E)

[Structured ladder/FBD]



[ST]

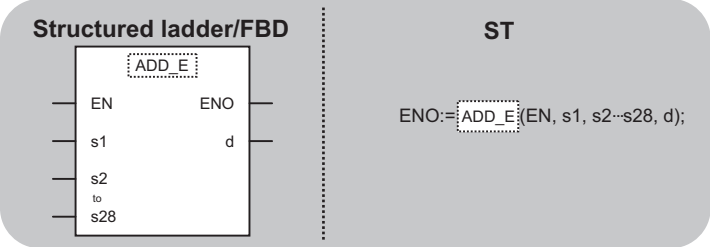
```
g_bool2 := ABS_E(g_bool1, g_lreal1, g_lreal2);
```

5.3 Standard Arithmetic Functions

Addition

ADD_E

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



The following function(s) can go in the dotted squares.

ADD_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1 to s28(_IN)	Input	ANY_NUM
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY_NUM

Processing details

■Operation processing

- Performs addition ((s1)+(s2)+ ··· +(s28)) on word (signed), double word (signed), single-precision real or double-precision real type data input to (s1) to (s28), and outputs the operation result from (d) in the same data type as that of (s1) to (s28).

Ex.

Word (signed) type data

- The values to be input to (s1) to (s28) are word (signed), double word (signed), single-precision real or double-precision real type data.
- The number of pins of (s) can be changed in the range from 2 to 28.
- If an underflow/overflow occurs in the operation result, data is output from (d) as follows.

Word (signed) type data

No operation error occurs even if an underflow/overflow occurs.

In case of ADD_E, TRUE is output from ENO.

32767 + 2 = -32767 (7FFFH) (0002H) (8001H) — Since the highest-order bit is 1, the result value is negative.

-32768 + (-2) = 32766 (8000H) (FFFEH) (7FFEH) — Since the highest-order bit is 0, the result value is positive.

Double word (signed) type data

No operation error occurs even if an underflow/overflow occurs.

In case of ADD_E, TRUE is output from ENO.

2147483647 + 2 = -2147483647 Since the highest-order bit is 1, the result value is negative.
(7FFFFFFFH) (0002H) (80000001H)

-2147483648 + (-2) = 2147483646 Since the highest-order bit is 0, the result value is positive.
(80000000H) (FFFEH) (7FFFFFFEH)

- Rounding error may occur when specifying single-precision real or double-precision real type data to (s1) through (s28) by programming tool.

For precautions when setting an input value using a programming tool, refer to the following.

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

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

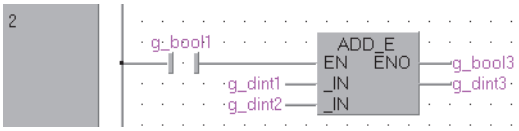
- No operation error occurs.

Program example

■ADD_E

The program which performs addition ((s1) + (s2)) on double word (signed) type data input to (s1) and (s2), and outputs the operation result from (d) in the same data type as that of (s1) and (s2).

[Structured ladder/FBD]



[ST]

```
g_bool3 := ADD_E(g_bool1, g_dint1, g_dint2, g_dint3);
```

Multiplication

MUL_E

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

The following function(s) can go in the dotted squares.
MUL_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1 to s28(_IN)	Input	ANY_NUM
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY_NUM

Processing details

■Operation processing

- Performs multiplication ((s1) × (s2) ×···× (s28)) on word (signed), double word (signed), single-precision real or double-precision real type data input to (s1) to (s28), and outputs the operation result from (d) in the same data type as that of (s1) to (s28).

Ex.
Word (signed) type data

- The values to be input to (s1) to (s28) are word (signed), double word (signed), single-precision real or double-precision real type data.
- The number of pins of (s) can be changed in the range from 2 to 28.
- If an underflow/overflow occurs in the operation result, data is output from (d) as follows.

Word (signed) type data

- No operation error occurs even if an underflow/overflow occurs. In case of MUL_E, TRUE is output from ENO.
- Even if the operation result exceeds the word (signed) type data range, data is output in word (signed) type. (Although the operation result is 32-bit data, data is output in word (signed) type with the high-order 16 bits discarded.)
- If the operation result exceeds the word (signed) type data range, convert the input values to the double word (signed) type data by the INT_TO_DINT function and perform the operation using the converted data.

Double word (signed) type data

- No operation error occurs even if an underflow/overflow occurs. In case of MUL_E, TRUE is output from ENO.
- Even if the operation result exceeds the double word (signed) data range, data is output in double word (signed) type. (Although the operation result is 64-bit data, data is output in double word (signed) type with the high-order 32 bits discarded.)
- If the operation result exceeds the double word (signed) type data range, convert the input values to the single-precision real type data by the DINT_TO_REAL function and perform the operation using the converted data.

Rounding error may occur when specifying single-precision real or double-precision real type data to (s1) through (s28) by programming tool.
For precautions when setting an input value using a programming tool, refer to the following.
MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Point

If the operation result exceeds the data type range, convert the data type of the input data before the operation.

Operation error

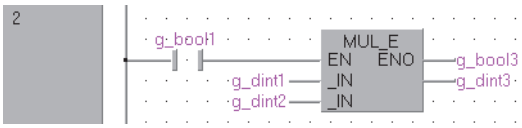
- No operation error occurs.

Program example

■MUL_E

The program which performs multiplication ((s1) × (s2)) on double word (signed) type data input to (s1) and (s2), and outputs the operation result from (d) in the same data type as that of (s1) and (s2).

[Structured ladder/FBD]



[ST]

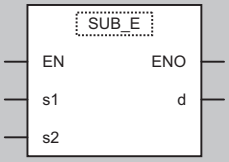
```
g_bool3 := MUL_E(g_bool1, g_dint1, g_dint2, g_dint3);
```

Subtraction

SUB_E

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

Structured ladder/FBD



ST

```
ENO:=SUB_E(EN, s1, s2, d);
```

The following function(s) can go in the dotted squares.
SUB_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_IN1)	Input	ANY_NUM
	s2(_IN2)		
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY_NUM

Processing details

■Operation processing

- Performs subtraction ((s1) - (s2)) on word (signed), double word (signed), single-precision real or double-precision real type data input to (s1) and (s2), and outputs the operation result from (d) in the same data type as that of (s1) and (s2).

Ex.
Word (signed) type data

- The values to be input to (s1) and (s2) are word (signed), double word (signed), single-precision real or double-precision real type data.
- If an underflow/overflow occurs in the operation result, data is output from (d) as follows.

Word (signed) type data

No operation error occurs even if an underflow/overflow occurs.

In case of SUB_E, TRUE is output from ENO.

32767- (-2)= -32767 — Since the highest-order bit is 1, the result value is negative.
(7FFFH) (FFFEH) (8001H)

-32768 - 2 = 32766 — Since the highest-order bit is 0, the result value is positive.
(8000H) (0002H) (7FFEH)

Double word (signed) type data

No operation error occurs even if an underflow/overflow occurs.

In case of SUB_E, TRUE is output from ENO.

2147483647 - (-2) = -2147483647 — Since the highest-order bit is 1, the result value is negative.
(7FFFFFFFH) (FFFEH) (80000001H)

-2147483648 - 2 = 2147483646 — Since the highest-order bit is 0, the result value is positive.
(80000000H) (0002H) (7FFFFFFEH)

• Rounding error may occur when specifying single-precision real or double-precision real type data to (s1), (s2) by programming tool.

For precautions when setting an input value using a programming tool, refer to the following.

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

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

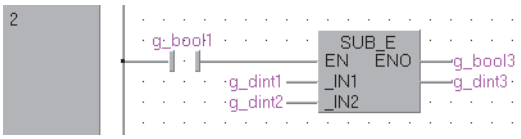
• No operation error occurs.

Program example

■SUB_E

The program which performs subtraction ((s1) - (s2)) on double word (signed) type data input to (s1) and (s2), and outputs the operation result from (d) in the same data type as that of (s1) and (s2).

[Structured ladder/FBD]



[ST]

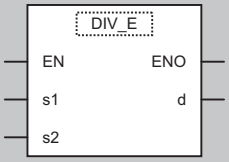
```
g_bool3 := SUB_E(g_bool1, g_dint1, g_dint2, g_dint3);
```

Division

DIV_E

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

Structured ladder/FBD



ST

```
ENO:=DIV_E(EN, s1, s2, d);
```

The following function(s) can go in the dotted squares.
DIV_E

■Argument

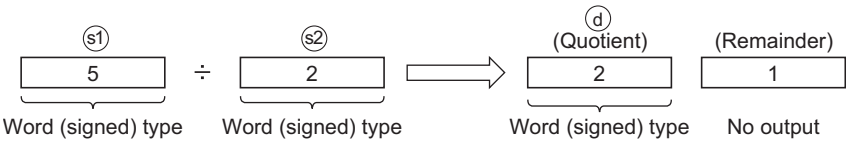
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_IN1)	Input	ANY_NUM
	s2(_IN2)		
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY_NUM

Processing details

■Operation processing

- Performs division ((s1) ÷ (s2)) on word (signed), double word (signed), single-precision real or double-precision real type data input to (s1) and (s2), and outputs the quotient of the operation result from (d) in the same data type as that of (s1) and (s2).

Ex.
Word (signed) type data



- The values to be input to (s1) and (s2) are word (signed), double word (signed), single-precision real or double-precision real type data. (The value to be input to (s2) must be other than 0.)
- Rounding error may occur when specifying single-precision real or double-precision real type data to (s1), (s2) by programming tool.

For precautions when setting an input value using a programming tool, refer to the following.
📖 MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

■Operation result

• Function without EN/ENO

The following table shows the operation results.

Operation result	(d)
No operation error	Operation output value
Operation error	Undefined value

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• An operation error occurs in the following cases.

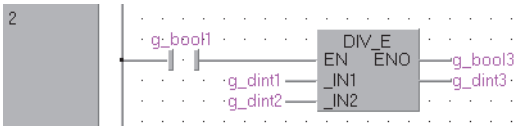
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value to be input to (s2) is 0. (Division by 0)	○	○	○	○	○	○

Program example

■DIV_E

The program which performs division ((s1) ÷ (s2)) on double word (signed) type data input to (s1) and (s2), and outputs the quotient of the operation result from (d) in the same data type as that of (s1) and (s2).

[Structured ladder/FBD]



[ST]

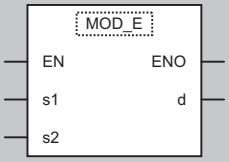
```
g_bool3 := DIV_E(g_bool1, g_dint1, g_dint2, g_dint3);
```

Remainder

MOD(_E)

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

Structured ladder/FBD



ST

```
ENO:=MOD_E(EN, s1, s2, d);
```

The following function(s) can go in the dotted squares.
MOD, MOD_E

■Argument

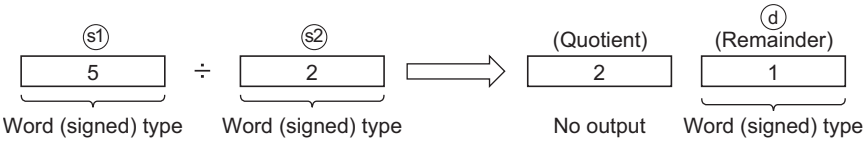
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_IN1)	Input	ANY_INT
	s2(_IN2)		
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY_INT

Processing details

■Operation processing

- Performs division ((s1) ÷ (s2)) on word (signed) or double word (signed) type data input to (s1) and (s2), and outputs the remainder of the operation result from (d) in the same data type as that of (s1) and (s2).

Ex.
Word (signed) type data



- The values to be input to (s1) and (s2) are word (signed) or double word (signed) type data. (Note that the value to be input to (s2) must be other than 0.)

■Operation result

- Function without EN/ENO

The following table shows the operation results.

Operation result	(d)
No operation error	Operation output value
Operation error	Undefined value

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
FALSE (Operation stop)	FALSE *1	Undefined value

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

Operation error

• An operation error occurs in the following case.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value to be input to (s2) is 0. (Division by 0)	○	○	○	○	○	○

Program example

■MOD(_E)

The program which performs division ((s1) ÷ (s2)) on double word (signed) type data input to (s1) and (s2), and outputs the remainder of the operation result from (d) in the same data type as that of (s1) and (s2).

• Function without EN/ENO (MOD)

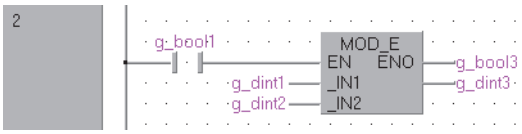
[Structured ladder/FBD]

[ST]

```
g_dint3:= (g_dint1) MOD (g_dint2);
```

• Function with EN/ENO (MOD_E)

[Structured ladder/FBD]



[ST]

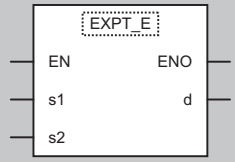
```
g_bool3 := MOD_E(g_bool1, g_dint1, g_dint2, g_dint3);
```

Exponentiation

EXPT(_E)

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

Structured ladder/FBD



ST

```
ENO:=EXPT_E(EN, s1, s2, d);
```

The following function(s) can go in the dotted squares.
EXPT, EXPT _E

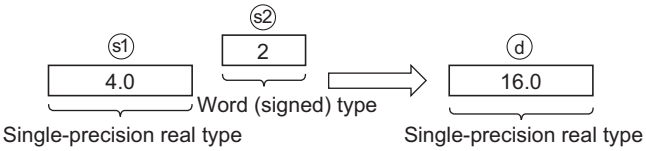
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_IN1)	Input	ANY_REAL
	s2(_IN2)	Input	ANY_NUM
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY_REAL

Processing details

■Operation processing

- Performs exponentiation (s2) on single-precision real or double-precision real type data input to (s2) and word (signed), double word (signed), single-precision real or double-precision real type data input to (s1), and outputs the operation result from (d).



- Rounding error may occur when specifying single-precision real or double-precision real type data to (s1), (s2) by programming tool.

For precautions when setting an input value using a programming tool, refer to the following.
MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE *1	Undefined value

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

Operation error

These functions consist of the following common instructions depending on the data type of (s1) and (s2).

Data type of (s1)	Data type of (s2)	Common instruction used
Single-precision real type	Word (signed) type	LOG, FLT
	Double word (signed) type	LOG, DFLT
	Single-precision real type	LOG
	Double-precision real type	LOGD, DFLTD
Double-precision real type	Word (signed) type	LOGD
	Double word (signed) type	LOGD, FLTD
	Single-precision real type	LOGD, DFLTD
	Double-precision real type	LOGD

- For details of an error which occurs when the function is executed, refer to the following.

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

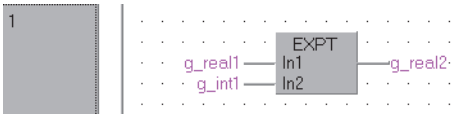
Program example

■EXPT(_E)

The program which performs exponentiation and outputs the operation result from (d) in the same data type as that of (s1) and (s2).

- Function without EN/ENO (EXPT)

[Structured ladder/FBD]

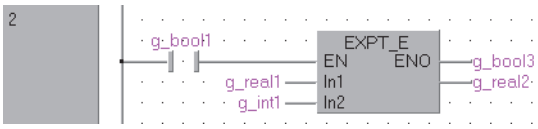


[ST]

```
g_real2:= EXPT(g_real1, g_int1);
```

- Function with EN/ENO (EXPT_E)

[Structured ladder/FBD]



[ST]

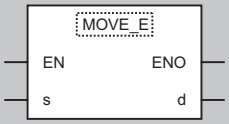
```
g_bool3 := EXPT_E(g_bool1, g_real1, g_int1, g_real2);
```

Move operation

MOVE(_E)

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

Structured ladder/FBD



ST

```
ENO:=MOVE_E(EN, s, d);
```

The following function(s) can go in the dotted squares.
MOVE, MOVE_E

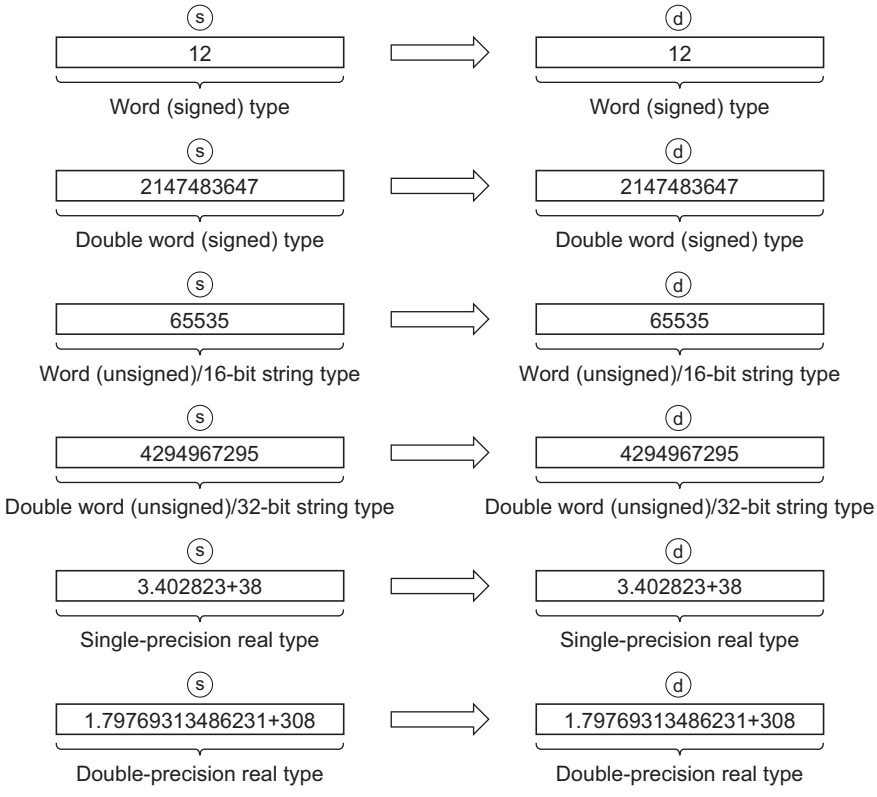
■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_IN)	Input	ANY
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY

Processing details

■Operation processing

- Moves the data input for (s) from (d) in the same data type as that of (s).
- The values to be specified to (s) and (d) are word (signed), double word (signed), word(unsigned)/16-bit string, double word (unsigned)/32-bit string, single-precision real, double-precision real, string, or time type data. Only the same data type can be specified for (s) and (d).



• Rounding error may occur when specifying single-precision real or double-precision real type data to (s) by programming tool.

For precautions when setting an input value using a programming tool, refer to the following.

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

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

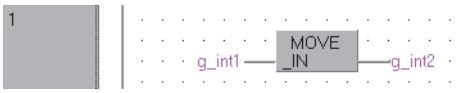
Program example

■MOVE(_E)

The program which moves the word (signed) type data input to (s) to (d).

• Function without EN/ENO (MOVE)

[Structured ladder/FBD]

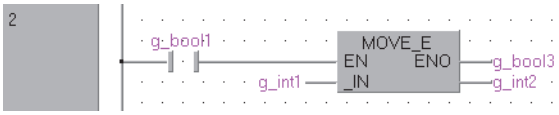


[ST]

```
g_int2:= MOVE(g_int1);
```

• Function with EN/ENO (MOVE_E)

[Structured ladder/FBD]



[ST]

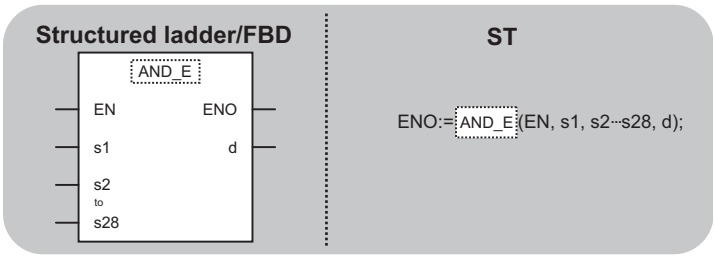
```
g_bool3 := MOVE_E(g_bool1, g_int1, g_int2);
```

5.4 Standard Bitwise Boolean Functions

Boolean AND, boolean OR, boolean exclusive OR, and boolean NOT

AND_E, OR_E, XOR_E, NOT(_E)

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



The following function(s) can go in the dotted squares.
AND_E, OR_E, XOR_E, NOT, NOT_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1 to s28(_IN) (s1 only for NOT(_E))	Input	ANY_BIT
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY_BIT

Processing details

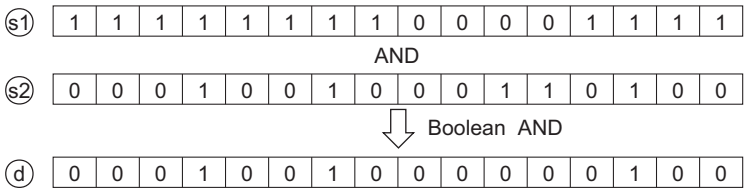
■Operation processing

• AND_E

Performs Boolean AND on bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data input to (s1) to (s28) bit by bit, and outputs the operation result from (d) in the same data type as that of (s1) to (s28).

Ex.

Word (unsigned)/16-bit string type data



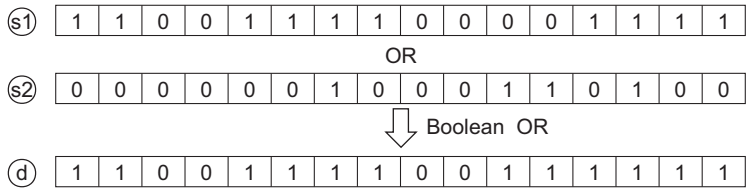
The number of pins of variable 's' can be changed in the range from 2 to 28.

• OR_E

Performs Boolean OR on bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data input to (s1) to (s28) bit by bit, and outputs the operation result from (d) in the same data type as that of (s1) to (s28).

Ex.

Word (unsigned)/16-bit string type data



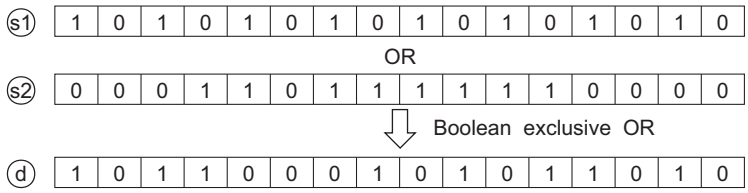
The number of pins of variable 's' can be changed in the range from 2 to 28.

• XOR_E

Performs Boolean exclusive OR on bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data input to (s1) to (s28) bit by bit, and outputs the operation result from (d) in the same data type as that of (s1) to (s28).

Ex.

Word (unsigned)/16-bit string type data



The number of pins of variable 's' can be changed in the range from 2 to 28.

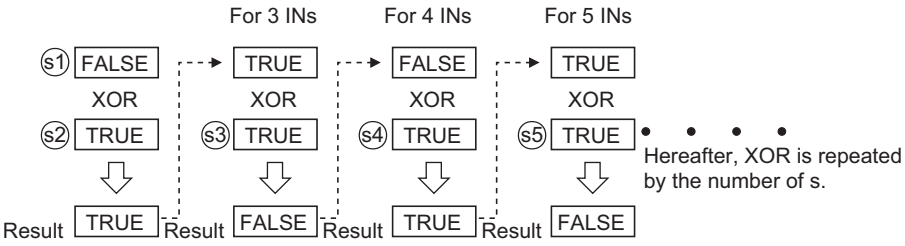
When three or more variables 's' exist, XOR is performed between (s1) and (s2) first, and XOR is successively performed between the result and (s3).

When the expression includes (s4), XOR is performed between the result of XOR with (s3) and (s4).

In this manner, XOR is repeated by the number of variables 's' in the order with (s5), (S6) and so on.

Ex.

Bit type data

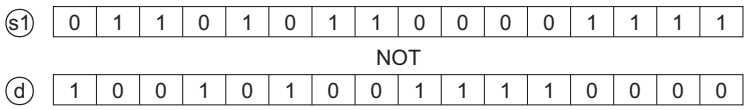


• NOT, NOT_E

Performs Boolean NOT on bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data input to (s1) bit by bit, and outputs the operation result from (d) in the same data type as that of (s1).

Ex.

Word (unsigned)/16-bit string type data



The value to be input to variables (s1) to (s28) is bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data.

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

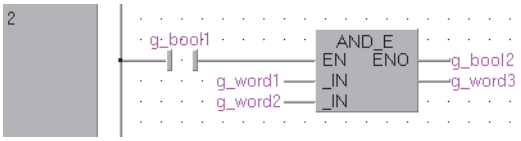
• No operation error occurs.

Program example

■AND_E

The program which performs Boolean AND on bit, word (unsigned)/16-bit string type data input to (s1) to (s28) bit by bit, and outputs the operation result from (d) in the same data type as that of (s1) to (s28).

[Structured ladder/FBD]



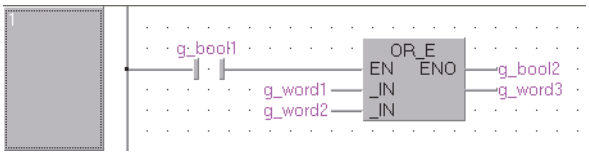
[ST]

```
g_bool2 := AND_E(g_bool1, g_word1, g_word2, g_word3);
```

■OR_E

The program which performs Boolean OR on bit, word (unsigned)/16-bit string type data input to (s1) to (s28) bit by bit, and outputs the operation result from (d) in the same data type as that of (s1) to (s28).

[Structured ladder/FBD]



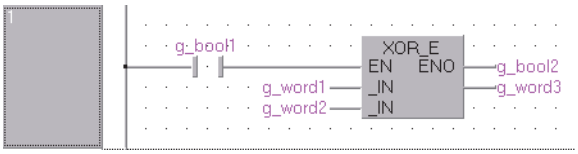
[ST]

```
g_bool2 := OR_E(g_bool1, g_word1, g_word2, g_word3);
```

■XOR_E

The program which performs Boolean XOR on bit, word (unsigned)/16-bit string type data input to (s1) to (s28) bit by bit, and outputs the operation result from (d) in the same data type as that of (s1) to (s28).

[Structured ladder/FBD]



[ST]

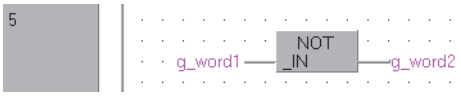
```
g_bool2 := XOR_E(g_bool1, g_word1, g_word2, g_word3);
```

■NOT(_E)

The program which performs Boolean NOT on bit, word (unsigned)/16-bit string type data input to (s1) bit by bit, and outputs the operation result from (d) in the same data type as that of (s1).

- Function without EN/ENO (NOT)

[Structured ladder/FBD]

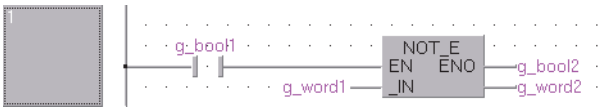


[ST]

```
g_word2 := NOT(g_word1);
```

- Function with EN/ENO (NOT_E)

[Structured ladder/FBD]



[ST]

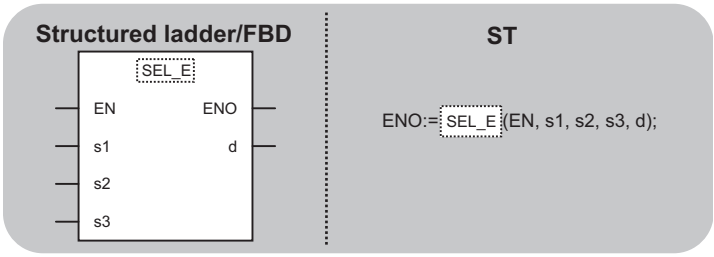
```
g_bool2 := NOT_E(g_bool1, g_word1, g_word2);
```

5.5 Standard Selection Functions

Selection

SEL(_E)

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



The following function(s) can go in the dotted squares.
SEL, SEL_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_G)	Output condition (TRUE: s3 output, FALSE: s2 output)	Bit
	s2(_IN0) s3(_IN1)	Input	ANY
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY

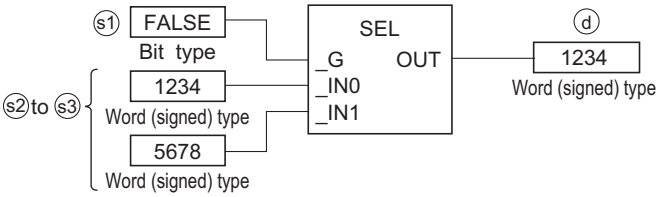
Processing details

■Operation processing

- Selects either of values input to (s2) and (s3) according to the bit type data input to (s1), and outputs the operation result from (d) in the same data type as that of (s2) and (s3).

When the input value of (s1) is FALSE, the value input to (s2) is output from (d).
When the input value of (s1) is TRUE, the value input to (s3) is output from (d).

Ex.
(s2) and (s3) are word (signed) type data



- The input value to (s1) is data value of bit type.
- The input value to (s2), (s3) is data value of bit type/word (signed) type/double word (signed) type/word (unsigned) type/16-bit string type/double word (unsigned) type/32-bit string type/ single-precision real number type/double-precision real number type/string type/time type/structured data type/array type.
- Rounding error may occur when specifying single-precision real or double-precision real type data to (s2), (s3) by programming tool.

For precautions when setting an input value using a programming tool, refer to the following.
MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

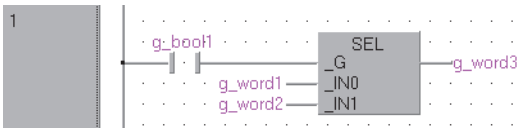
Program example

■SEL(_E)

The program which selects either of values input to (s2) and (s3) according to the value input to (s1), and outputs the operation result from (d) in the same data type as that of (s2) and (s3).

• Function without EN/ENO (SEL)

[Structured ladder/FBD]

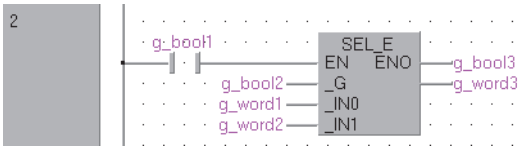


[ST]

```
g_word3 := SEL(g_bool1, g_word1, g_word2);
```

• Function with EN/ENO (SEL_E)

[Structured ladder/FBD]



[ST]

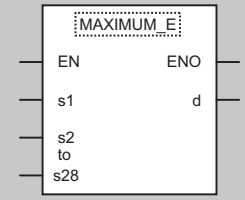
```
g_bool3 := SEL_E(g_bool1, g_bool2, g_word1, g_word2, g_word3);
```

Maximum/Minimum selection

MAXIMUM(_E), MINIMUM(_E)

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

Structured ladder/FBD



ST

```
ENO:=MAXIMUM_E(EN, s1, s2…s28, d);
```

The following function(s) can go in the dotted squares.
MAXIMUM, MAXIMUM_E, MINIMUM, MINIMUM_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1 to s28(_IN)	Input	ANY_SIMPLE
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY_SIMPLE

Processing details

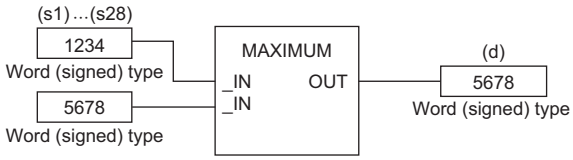
■Operation processing

• MAXIMUM, MAXIMUM_E

Selects the maximum value to be output among the bit, word (signed), double word (signed), word(unsigned)/16-bit string, double word(unsigned)/32-bit string, single-precision real, double-precision real, string, or time type data input to (s1) to (s28), and outputs the operation result from (d) in the same data type as that of (s1) to (s28).

Ex.

Word (signed) type data

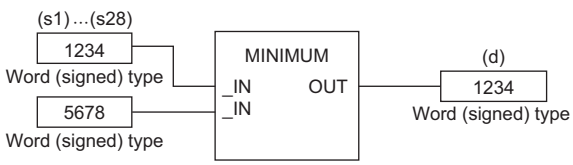


• MINIMUM, MINIMUM_E

Selects the minimum value to be output among the word (signed), double word (signed) or single-precision real type data input to (s1) to (s28), and outputs the operation result from (d) in the same data type as that of (s1) to (s28).

Ex.

Word (signed) type data



- The values to be input to (s1) to (s28) are bit, word (signed), double word (signed), word(unsigned)/16-bit string, double word (unsigned)/32-bit string, single-precision real, double-precision real, string, or time type data.
- Rounding error may occur when specifying single-precision real or double-precision real type data to (s1) through (s28) by programming tool.

For precautions when setting an input value using a programming tool, refer to the following.

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

- The number of pins of (s) can be changed in the range from 2 to 28.
- If word (unsigned) type/16-bit string type/double word (unsigned) type/32-bit string type is specified for (d), warning C9026 occurs.

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

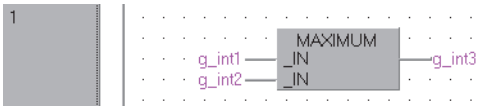
Program example

■MAXIMUM(_E)

The program which outputs the maximum value of the word (signed) data input to variables (s1) to (s28) from (d) in the same data type as that of (s1) to (s28).

- Function without EN/ENO (MAXIMUM)

[Structured ladder/FBD]

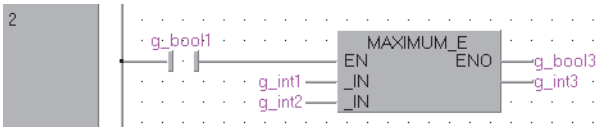


[ST]

```
g_int3 := MAXIMUM(g_int1, g_int2);
```

- Function with EN/ENO (MAXIMUM_E)

[Structured ladder/FBD]



[ST]

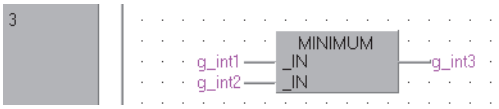
```
g_bool3 := MAXIMUM_E(g_bool1, g_int1, g_int2, g_int3);
```

■MINIMUM(_E)

The program which outputs the minimum value of the word (signed) data input to variables (s1) to (s28) from (d) in the same data type as that of (s1) to (s28).

- Function without EN/ENO (MINIMUM)

[Structured ladder/FBD]



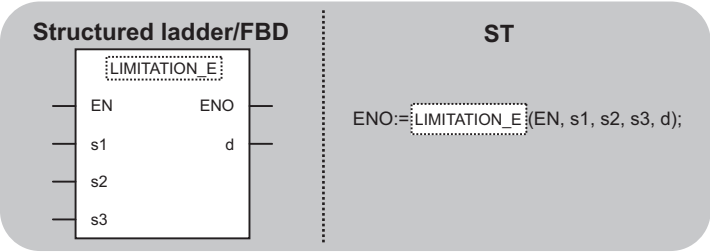
[ST]

```
g_int3 := MINIMUM(g_int1, g_int2);
```

Upper/Lower limit control

LIMITATION(_E)

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



The following function(s) can go in the dotted squares.
LIMITATION, LIMITATION_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_MN)	Lower limit value (minimum output limit value)	ANY_SIMPLE
	s2(_IN)	Input controlled by the upper/lower limit control	ANY_SIMPLE
	s3(_MX)	Upper limit value (maximum output limit value)	ANY_SIMPLE
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY_SIMPLE

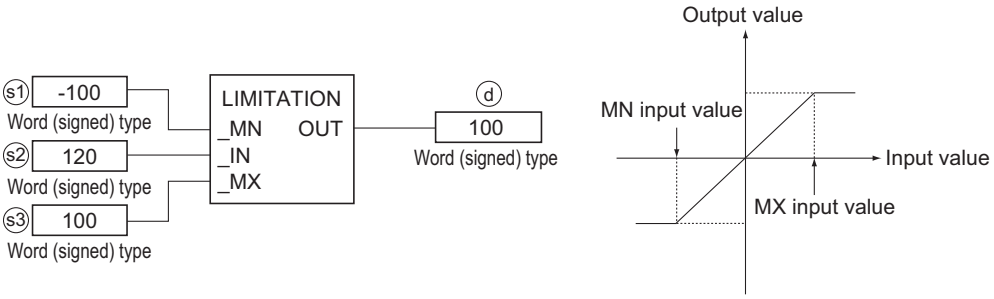
Processing details

■Operation processing

- Selects the value to be output among the bit, word (signed), double word (signed), word (unsigned)/16-bit string, double word (unsigned)/32-bit string, or single-precision real type, double-precision real, string, or time type data input to (s1), (s2), and (s3) according to their values, and outputs the operation result from (d) in the same data type as that of (s1) to (s3).

When the input value of (s2) > the input value of (s3), outputs the input value (s3) from (d).
When the input value of (s2) < the input value of (s1), outputs the input value (s1) from (d).
When the input value of (s1) ≤ the input value of (s2) ≤ the input value of (s3), outputs the input value of (s2) from (d).

Ex.
Word (signed) type data



- The values to be input to (s1), (s2), and (s3) are bit, word (signed), double word (signed), word (unsigned)/16-bit string, double word (unsigned)/32-bit string, single-precision real, double-precision real, string, or time type data. (the input value of (s1) < the input value of (s3))
- Rounding error may occur when specifying single-precision real or double-precision real type data to (s1), (s2), or (s3) by programming tool.

For precautions when setting an input value using a programming tool, refer to the following.

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

- If word (unsigned) type/16-bit string type/double word (unsigned) type/32-bit string type is specified for (d), warning C9026 occurs at compilation.

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

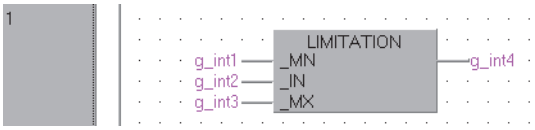
Program example

■LIMITATION(_E)

The program which outputs the values input to variables (s1), (s2), and (s3) according to the word (signed) data from (d) in the same data type as that of (s1), (s2), and (s3).

- Function without EN/ENO (LIMITATION)

[Structured ladder/FBD]

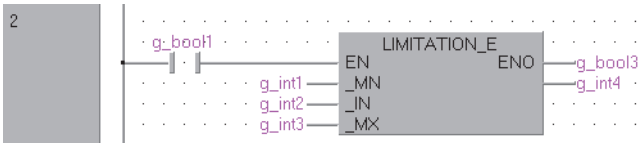


[ST]

```
g_int4 := LIMITATION(g_int1, g_int2, g_int3);
```

- Function with EN/ENO (LIMITATION_E)

[Structured ladder/FBD]



[ST]

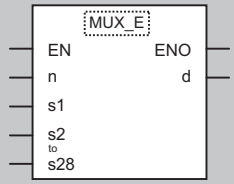
```
g_bool3 := LIMITATION_E(g_bool1, g_int1, g_int2, g_int3, g_int4);
```

Multiplexer

MUX(_E)

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

Structured ladder/FBD



ST

```
ENO:=MUX_E(EN, n, s1, s2…s28, d);
```

The following function(s) can go in the dotted squares.
MUX, MUX_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	n(_K)	Output value selection	Word (signed)
	s1 to s28(_IN)	Input	ANY
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	ANY

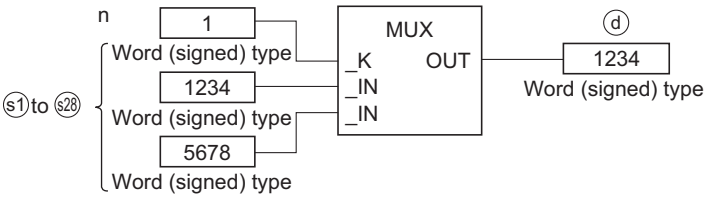
Processing details

■Operation processing

- Selects the value to be output among the values input to variables (s1) to (s28) according to the value input to n, and outputs the operation result from (d) in the same data type as that of variables (s1) to (s28).

When the input value of n is 1, the value input to (s1) is output from (d).
When the input value of n is n, the value input to (sn) is output from (d).

Ex.
Word (signed) type data



- If a value input to n is outside the range of number of pins of variable 's', an undefined value is output from (d).

(No operation error occurs. In case of MUX_E, FALSE is output from ENO.)

- The value to be input to n is word (signed) type data within the range from 1 to 28. (within the range of the number of pins of variable 's')
- The value to be input to variable 's' is bit, word (signed), double word (signed), word (unsigned)/16-bit string, double word (unsigned)/32-bit string, single-precision real, double-precision real, string, time, structure, or array type data.
- Rounding error may occur when specifying single-precision real or double-precision real type data to (s1) through (s28) by programming tool.

For precautions when setting an input value using a programming tool, refer to the following.
MELSEC-Q/L/F Structured Programming Manual (Fundamentals)

- The number of pins of (s) can be changed in the range from 2 to 28.

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

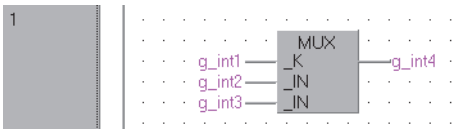
Program example

■MUX(_E)

The program which selects the value to be output among the values input to variables (s1) and (s2) according to the value input to n, and outputs the operation result from (d) in the same data type as that of (s1) or (s2).

• Function without EN/ENO (MUX)

[Structured ladder/FBD]

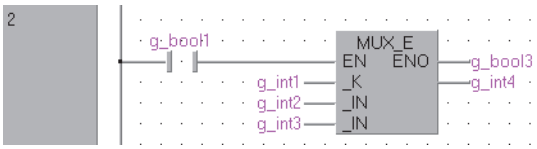


[ST]

```
g_int4 := MUX(g_int1, g_int2, g_int3);
```

• Function with EN/ENO (MUX_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := MUX_E(g_bool1, g_int1, g_int2, g_int3, g_int4);
```

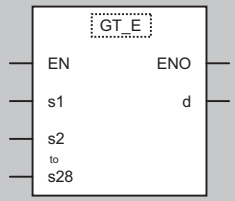
5.6 Standard Comparison Functions

Comparison

GT_E, GE_E, EQ_E, LE_E, LT_E, NE_E

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

Structured ladder/FBD



ST

Not supported

The following function(s) can go in the dotted squares.

GT_E, GE_E, EQ_E, LE_E, LT_E, NE_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1 to s28(_IN) (s1 and s2 only for NE(_E))		ANY_SIMPLE
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output (TRUE: True value, FALSE: False value)	Bit

Processing details

■Operation processing

- Performs comparison operation between the values input to variables (s1) to (s28), and outputs the operation result from (d) in bit type as that of variables (s1) to (s28).

GT_E

Performs comparison of $[(s1)>(s2)]\&[(s2)>(s3)]\&\cdots\&[(s)_{(n-1)}>(s)_{(n)}]$.

- Outputs TRUE if all of comparisons satisfy $(s)_{(n-1)} > (s)_{(n)}$.
- Outputs FALSE if any of comparisons satisfies $(s)_{(n-1)} \leq (s)_{(n)}$.

GE_E

Performs comparison of $[(s1) \geq (s2)]\&[(s2) \geq (s3)]\&\cdots\&[(s)_{(n-1)} \geq (s)_{(n)}]$.

- Outputs TRUE if all of comparisons satisfy $(s)_{(n-1)} \geq (s)_{(n)}$.
- Outputs FALSE if any of comparisons satisfies $(s)_{(n-1)} < (s)_{(n)}$.

EQ_E

Performs comparison of $[(s1) = (s2)]\&[(s2) = (s3)]\&\cdots\&[(s)_{(n-1)} = (s)_{(n)}]$.

- Outputs TRUE if all of comparisons satisfy $(s)_{(n-1)} = (s)_{(n)}$.
- Outputs FALSE if any of comparisons satisfies $(s)_{(n-1)} \neq (s)_{(n)}$.

LE_E

Performs comparison of $[(s1) \leq (s2)]\&[(s2) \leq (s3)]\&\cdots\&[(s)_{(n-1)} \leq (s)_{(n)}]$.

- Outputs TRUE if all comparisons satisfy $(s)_{(n-1)} \leq (s)_{(n)}$.
- Outputs FALSE if any of comparisons satisfies $(s)_{(n-1)} > (s)_{(n)}$.

LT_E

Performs comparison of $[(s1) < (s2)]\&[(s2) < (s3)]\&\cdots\&[(s)_{(n-1)} < (s)_{(n)}]$.

- Outputs TRUE if all comparisons satisfy $(s)_{(n-1)} < (s)_{(n)}$.
- Outputs FALSE if any of comparisons satisfies $(s)_{(n-1)} \geq (s)_{(n)}$.

NE_E

Performs comparison of $[(s1) \neq (s2)]$.

- Outputs TRUE if $(s1) \neq (s2)$.
- Outputs FALSE if $(s1) = (s2)$.

- The values to be input to (s) is bit, word (signed), double word (signed), word (unsigned), 16-bit string, double word (unsigned), 32-bit string, single-precision real, double-precision real, string, time type data.
- Rounding error may occur when specifying single-precision real or double-precision real type data to (s1) through (s28) by programming tool.

For precautions when setting an input value using a programming tool, refer to the following.

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

- The number of pins of (s) can be changed in the range from 2 to 28. (The number of pins of (s) for comparison operator NE(_E)) is fixed at (s1) and (s2).

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

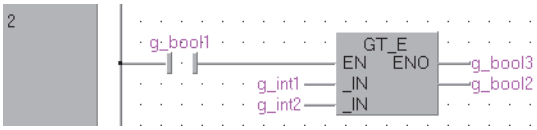
- No operation error occurs.

Program example

■GT_E

The program which performs comparison operation between the values input to (s1) and (s2), and outputs the operation result from (d).

[Structured ladder/FBD]



[ST]

```
g_bool3 := GT_E(g_bool1, g_int1, g_int2, g_bool2);
```

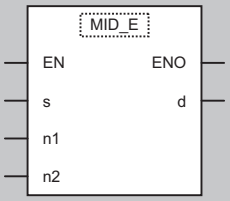
5.7 Standard Character String Functions

Extract mid string

MID(_E)

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

Structured ladder/FBD



ST

```
ENO:= MID_E (EN, s, n1, n2, d);
```

The following function(s) can go in the dotted squares.
MID, MID_E

■Argument

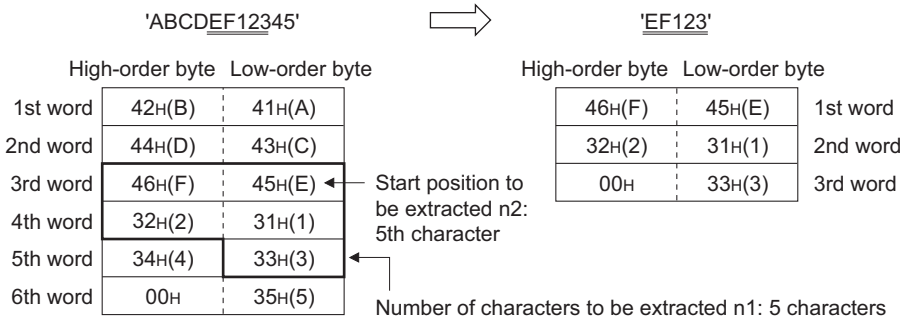
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_IN)	Input	String (255)
	n1(_L)	Number of characters to be extracted	Word (signed)
	n2(_P)	Start position to be extracted	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	String (255)

Processing details

■Operation processing

- Extracts the specified number of characters from the specified start position in the character string input to (s), and outputs the operation result from (d). The number of characters to be extracted is specified by the value input to n1. The start position of the characters to be extracted is specified by the value input to n2.

Ex.
Values input to n1 and n2 are 5



- The value to be input to (s) is string type data within the range from 0 to 255 bytes.
- The value to be input to n1 is word (signed) type data within the range from 0 to 255. (The input value must not exceed the number of characters of character string input to (s).)
- The value to be input to n2 is word (signed) type data within the range from 1 to 255. (The input value must not exceed the number of characters of character string input to (s).)

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

These functions consist of the following instructions.

MID(_E): MIDR

• 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 value of n2 exceeds the number of characters specified for (s). The number of characters from the start position of (d) to n1 exceeds the device range of (d). The value of n2 is 0. "00H" does not exist at the device number or later specified by (s) in the range of the corresponding device.	—	○	○	○	○	○

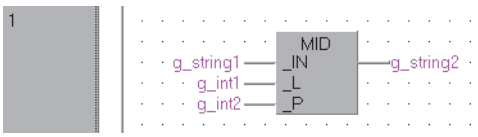
Program example

■MID(_E)

The program which extracts the specified number of characters from the specified start position in the character string input to (s), and outputs the operation result from (d).

• Function without EN/ENO (MID)

[Structured ladder/FBD]

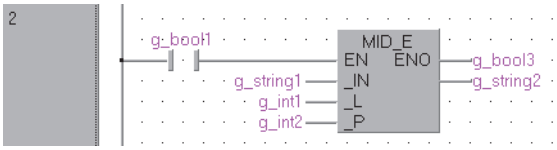


[ST]

```
g_string2 := MID(g_string1, g_int1, g_int2);
```

• Function with EN/ENO (MID_E)

[Structured ladder/FBD]



[ST]

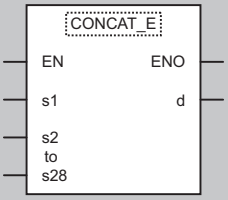
```
g_bool3 := MID_E(g_bool1, g_string1, g_int1, g_int2, g_string2);
```

String concatenation

CONCAT(_E)

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

Structured ladder/FBD



ST

```
ENO:=CONCAT_E(EN, s1, s2…n28, d);
```

The following function(s) can go in the dotted squares.
CONCAT, CONCAT_E

■Argument

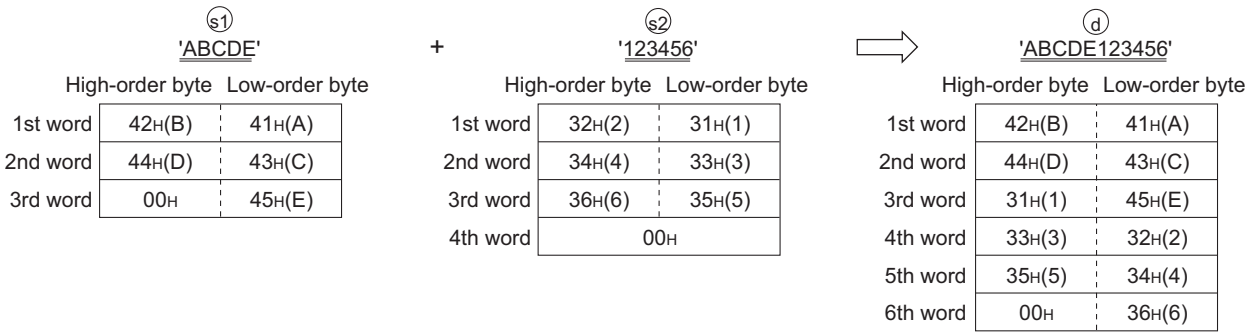
Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1 to s28(_IN)	Input	String (255)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	String (255)

Processing details

■Operation processing

- Concatenates the character string input to (s2) to (s28) following the one input to (s1), and outputs the operation result from (d).

This function concatenates character string (s2) to (s28) with ignoring '00H', which indicates the end of character string (s1).
If the concatenated character string has over 255 bytes, the character string up to 255 bytes is output.



	High-order byte	Low-order byte
1st word	42H(B)	41H(A)
2nd word	44H(D)	43H(C)
3rd word	00H	45H(E)

	High-order byte	Low-order byte
1st word	32H(2)	31H(1)
2nd word	34H(4)	33H(3)
3rd word	36H(6)	35H(5)
4th word	00H	

	High-order byte	Low-order byte
1st word	42H(B)	41H(A)
2nd word	44H(D)	43H(C)
3rd word	31H(1)	45H(E)
4th word	33H(3)	32H(2)
5th word	35H(5)	34H(4)
6th word	00H	36H(6)

- The values to be input to (s1) and (s2) to (s28) are string type data within the range from 0 to 255 bytes.
- The number of pins of (s) can be changed in the range from 2 to 28.

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

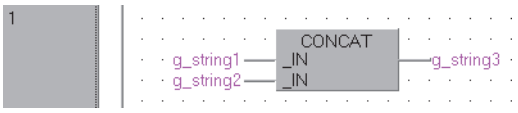
Program example

■CONCAT(_E)

The program which concatenates the character string input to (s2) following the one input to (s1), and outputs the operation result from (d).

- Function without EN/ENO (CONCAT)

[Structured ladder/FBD]

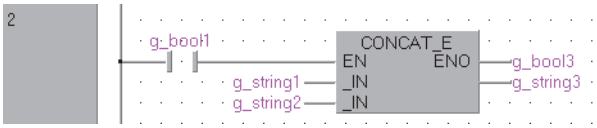


[ST]

```
g_string3 := CONCAT(g_string1, g_string2);
```

- Function with EN/ENO (CONCAT_E)

[Structured ladder/FBD]



[ST]

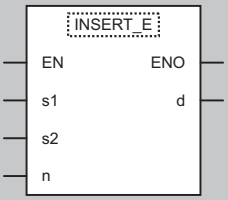
```
g_bool3 := CONCAT_E(g_bool1, g_string1, g_string2, g_string3);
```

String insertion

INSERT(_E)

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

Structured ladder/FBD



ST

```
ENO:= INSERT_E (EN, s1, s2, n, d);
```

The following function(s) can go in the dotted squares.
INSERT, INSERT_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_IN1)	Input	String (255)
	s2(_IN2)		
	n(_P)	Start position to be inserted	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	String (255)

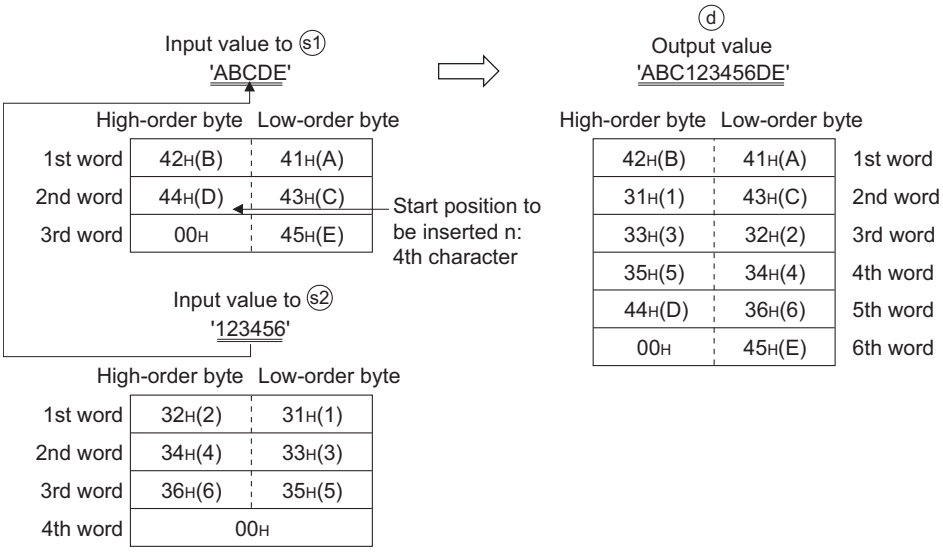
Processing details

■Operation processing

- Inserts the character string input to (s2) to the specified position in the character string input to (s1), and outputs the operation result from (d).

Specify the start position of the character string to be inserted by the value input to n.
After the insertion of character string (s2) to character string (s1), '00H' that indicates the end of character string (s2) is ignored. If the character string after insertion has over 255 bytes, the character string up to 255 bytes is output.

Ex.
Value input to n is 4



• The values to be input to (s1) and (s2) are string type data within the range from 0 to 255 bytes.
• The value to be input to n is word (signed) type data within the range from 1 to 255.
(The input value must not exceed the number of characters of character string input to input variable (s1).)

■Operation result

• Function without EN/ENO
An operation is executed and the operation value is output from (d).
• Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

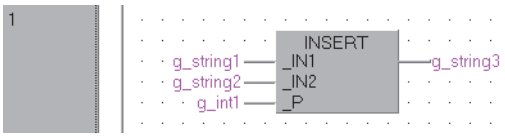
• No operation error occurs.

Program example

■INSERT(_E)

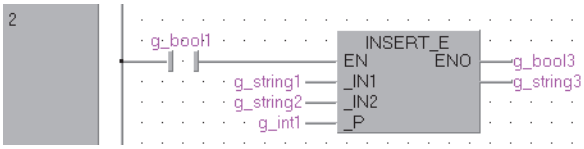
The program which inserts the character string input to (s2) to the specified position in the character string input to (s1), and outputs the operation result from (d).

• Function without EN/ENO (INSERT)
[Structured ladder/FBD]



[ST]
g_string3 := INSERT(g_string1, g_string2, g_int1);

• Function with EN/ENO (INSERT_E)
[Structured ladder/FBD]



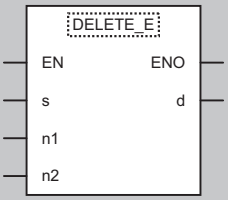
[ST]
g_bool3 := INSERT_E(g_bool1, g_string1, g_string2, g_int1, g_string3);

String deletion

DELETE(_E)

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

Structured ladder/FBD



ST

```
ENO:= DELETE_E (EN, s, n1, n2, d);
```

The following function(s) can go in the dotted squares.
DELETE, DELETE_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_IN)	Input	String (255)
	n1(_L)	Number of characters to be deleted	Word (signed)
	n2(_P)	Start position to be deleted	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	String (255)

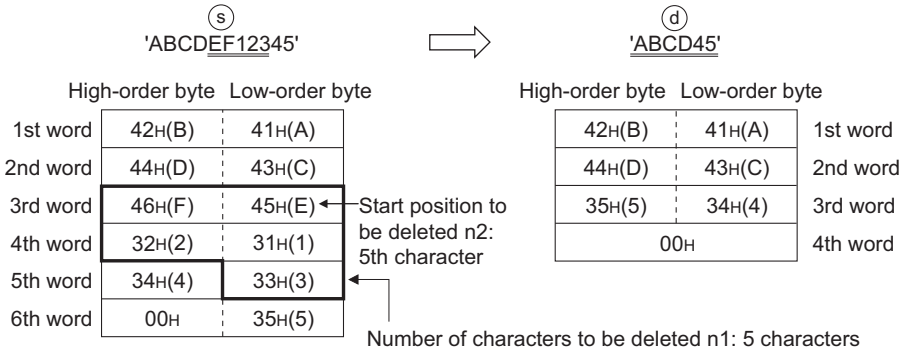
Processing details

■Operation processing

- Deletes the specified number of characters from the specified position in the character string input to (s), and outputs the remaining character string from (d).

The number of characters to be deleted is specified by the value input to n1.
The start position to be deleted in the character string is specified by the value input to n2.

Ex.
Values input to n1 and n2 are 5



- The value to be input to (s) is string type data within the range from 0 to 255 bytes.
- The value to be input to n1 is word (signed) type data within the range from 0 to 255.

(The input value must not exceed the number of characters of character string input to (s).)

- The value to be input to n2 is word (signed) type data within the range from 1 to 255.

(The input value must not exceed the number of characters of character string input to (s).)

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

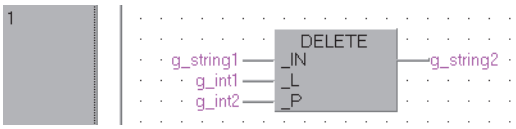
Program example

■DELETE(_E)

The program which deletes the specified number of characters from the specified position in the character string input to (s), and outputs the remaining character string from (d).

• Function without EN/ENO (DELETE)

[Structured ladder/FBD]

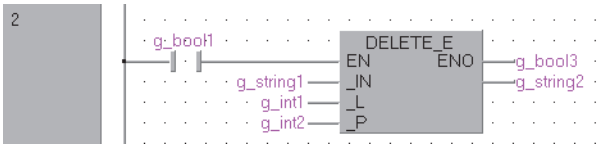


[ST]

```
g_string2 := DELETE(g_string1, g_int1, g_int2);
```

• Function with EN/ENO (DELETE_E)

[Structured ladder/FBD]



[ST]

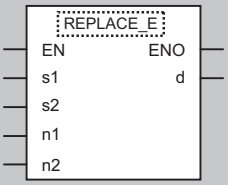
```
g_bool3 := DELETE_E(g_bool1, g_string1, g_int1, g_int2, g_string2);
```

String replacement

REPLACE(_E)

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

Structured ladder/FBD



ST

```
ENO:=REPLACE_E(EN, s1, s2, n1, n2, d);
```

The following function(s) can go in the dotted squares.

REPLACE, REPLACE_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_IN1)	Input	String (255)
	s2(_IN2)		
	n1(_L)	Number of characters to be replaced	Word (signed)
	n2(_P)	Start position to be replaced	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	String (255)

Processing details

■Operation processing

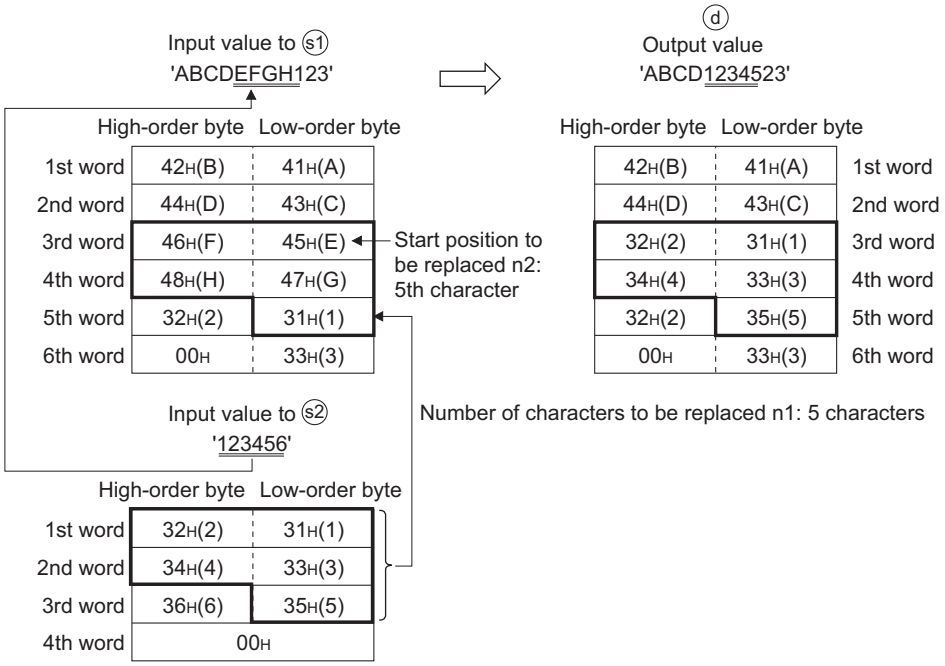
- Replaces the specified number of characters from the specified position in the character string input to (s1) with the character string input to (s2), and outputs the operation result from (d).

The number of characters to be replaced is specified by the value input to n1.

The start position to be replaced in the character string is specified by the value input to n2.

Ex.

Values input to n1 and n2 are 5



- The values to be input to (s1) and (s2) are string type data within the range from 0 to 255 bytes.
- The value to be input to n1 is word (signed) type data within the range from 0 to 255.

(The input value must not exceed the number of characters of character string input to input variable (s1).)

- The value to be input to n2 is word (signed) type data within the range from 1 to 255.

(The input value must not exceed the number of characters of character string input to input variable (s1).)

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

- Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

- No operation error occurs.

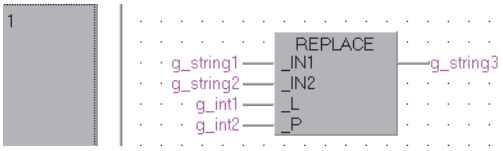
Program example

■REPLACE(_E)

The program which replaces the specified number of characters from the specified position in the character string input to (s1) with the character string input to (s2), and outputs the operation result from (d).

- Function without EN/ENO (REPLACE)

[Structured ladder/FBD]

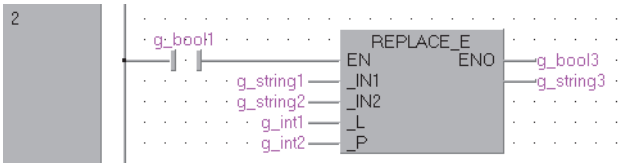


[ST]

```
g_string3 := REPLACE(g_string1, g_string2, g_int1, g_int2);
```

- Function with EN/ENO (REPLACE_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := REPLACE_E(g_bool1, g_string1, g_string2, g_int1, g_int2, g_string3);
```

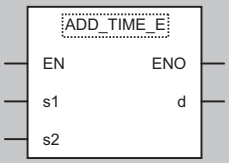
5.8 Functions of Time Data Type

Addition

ADD_TIME(_E)

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

Structured ladder/FBD



ST

```
ENO:=ADD_TIME_E(EN, s1, s2, d);
```

The following function(s) can go in the dotted squares.
ADD_TIME, ADD_TIME_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_IN1)	Input	Time
	s2(_IN2)		
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Time

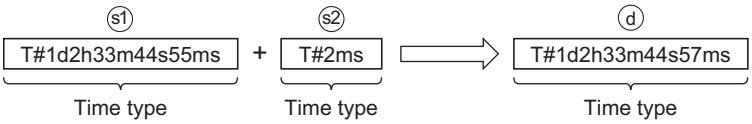
Processing details

■Operation processing

- Performs addition ((s1) + (s2)) on time type data input to (s1) and (s2), and outputs the operation result from (d) in time type.

Ex.

When the input value to (s1) and (s2) are T#1d2h33m44s55ms (1 day 2 hours 33 minutes 44 seconds 55 milliseconds) and T#2ms (2 milliseconds).

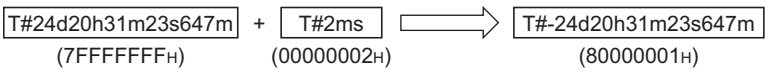


- The value to be input to (s1), (s2) are time type data.
- No operation error occurs even if an underflow/overflow occurs. Data is output from (d) as follows.

In case of ADD_TIME_E, TRUE is output from ENO.

Ex.

Overflow



Since the highest-order bit is 1, the result value is negative.

Ex.

Underflow

T#-24d20h31m23s648ms (80000000H) + T#-2ms (FFFFFFFEH) ⟹ T#24d20h31m23s646ms (7FFFFFFEH)

Since the highest-order bit is 0, the result value is positive.

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

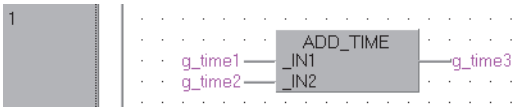
Program example

■ADD_TIME(_E)

The program which performs addition ((s1) + (s2)) on time type data input to (s1) and (s2), and outputs the operation result from (d) in time type.

• Function without EN/ENO (ADD_TIME)

[Structured ladder/FBD]

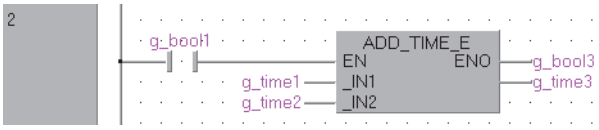


[ST]

```
g_time3 := ADD_TIME(g_time1, g_time2);
```

• Function with EN/ENO (ADD_TIME_E)

[Structured ladder/FBD]



[ST]

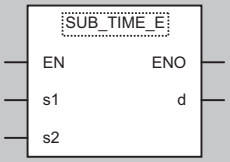
```
g_bool3 := ADD_TIME_E(g_bool1, g_time1, g_time2, g_time3);
```

Subtraction

SUB_TIME(_E)

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

Structured ladder/FBD



ST

```
ENO:=SUB_TIME_E(EN, s1, s2, d);
```

The following function(s) can go in the dotted squares.
SUB_TIME, SUB_TIME_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_IN1)	Input	Time
	s2(_IN2)		
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Time

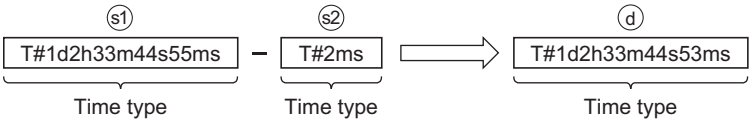
Processing details

■Operation processing

- Performs subtraction ((s1) - (s2)) on time type data input to (s1) and (s2), and outputs the operation result from (d) in time type.

Ex.

When the input value to (s1) and (s2) are T#1d2h33m44s55ms (1 day 2 hours 33 minutes 44 seconds 55 milliseconds) and T#2ms (2 milliseconds).

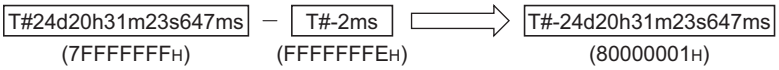


- The value to be input to (s1), (s2) are time type data.
- No operation error occurs even if an underflow/overflow occurs. Data is output from (d) as follows.

In case of SUB_TIME_E, TRUE is output from ENO.

Ex.

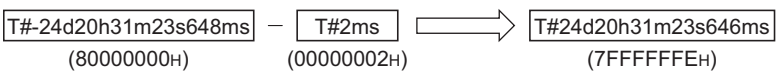
Overflow



Since the highest-order bit is 1, the result value is negative.

Ex.

Underflow



Since the highest-order bit is 0, the result value is positive.

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

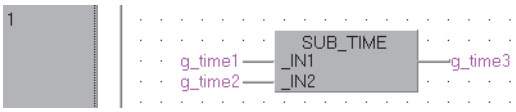
Program example

■SUB_TIME(_E)

The program which performs subtraction ((s1) - (s2)) on time type data input to (s1) and (s2), and outputs the operation result from (d) in time type.

• Function without EN/ENO (SUB_TIME)

[Structured ladder/FBD]

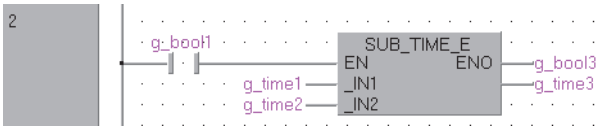


[ST]

```
g_time3 := SUB_TIME(g_time1, g_time2);
```

• Function with EN/ENO (SUB_TIME_E)

[Structured ladder/FBD]



[ST]

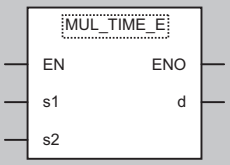
```
g_bool3 := SUB_TIME_E(g_bool1, g_time1, g_time2, g_time3);
```

Multiplication

MUL_TIME(_E)

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

Structured ladder/FBD



ST

```
ENO:=MUL_TIME_E(EN, s1, s2, d);
```

The following function(s) can go in the dotted squares.
MUL_TIME, MUL_TIME_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_IN1)	Input	Time
	s2(_IN2)	Input	ANY_NUM
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Time

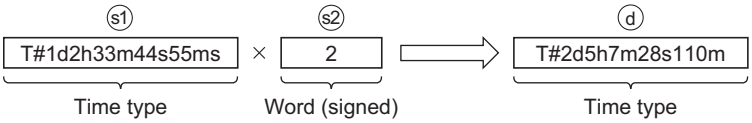
Processing details

■Operation processing

- Performs multiplication ((s1) × (s2)) on time type data input to (s1) and the word (signed), double word (signed), single-precision real or double-precision real type data input to (s2), and outputs the operation result from (d) in time type.

Ex.

When the input value to (s1) and (s2) are T#1d2h33m44s55ms (1 day 2 hours 33 minutes 44 seconds 55 milliseconds) and 2.



- The value to be input to (s1) is time type data.
- The value to be input to (s2) is word (signed), double word (signed), single-precision real or double-precision real type data.
- Rounding error may occur when specifying single-precision real or double-precision real type data to (s2) by programming tool.

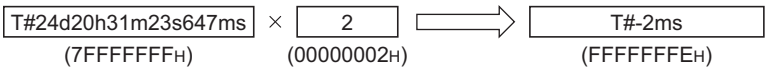
For precautions when setting an input value using a programming tool, refer to the following.

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

No operation error occurs even if an underflow/overflow occurs. Data is output from (d) as follows. In case of MUL_TIME_E, TRUE is output from ENO. (Although the operation result is 64-bit data, data is output in time type with the high-order 32 bits discarded.)

Ex.

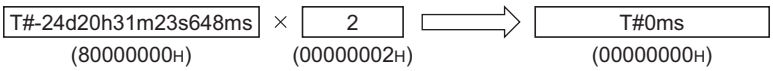
Overflow



Since the highest-order bit is 1, the result value is negative.

Ex.

Underflow



Since the highest-order bit is 0, the result value is positive.

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• No operation error occurs.

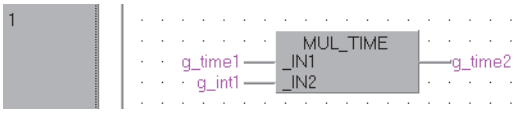
Program example

■MUL_TIME(_E)

The program which performs multiplication ((s1) × (s2)) on time type data input to (s1) and the word (signed) type data input to (s2), and outputs the operation result from (d) in time type.

• Function without EN/ENO (MUL_TIME)

[Structured ladder/FBD]

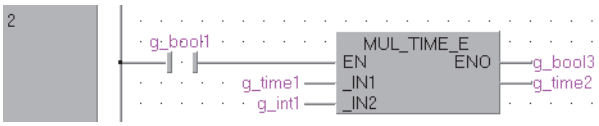


[ST]

```
g_time2 := MUL_TIME(g_time1, g_int1);
```

• Function with EN/ENO (MUL_TIME_E)

[Structured ladder/FBD]



[ST]

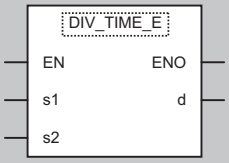
```
g_bool3 := MUL_TIME_E(g_bool1, g_time1, g_int1, g_time2);
```

Division

DIV_TIME(_E)

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

Structured ladder/FBD



ST

```
ENO:=DIV_TIME_E(EN, s1, s2, d);
```

The following function(s) can go in the dotted squares.
DIV_TIME, DIV_TIME_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_IN1)	Input	Time
	s2(_IN2)	Input	:ANY_NUM
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error)	Bit
	d	Output	Time

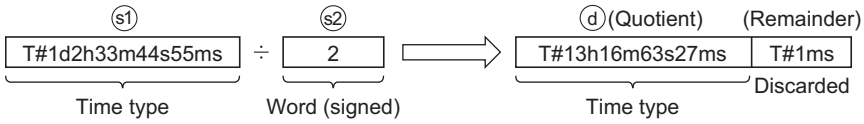
Processing details

■Operation processing

- Performs division ((s1) ÷ (s2)) on time type data input to (s1) and the word (signed), double word (signed), single-precision real or double-precision real type data input to (s2), and outputs the quotient of the operation result from (d) in time type. Remainder is rounded down. Remainder is rounded down.

Ex.

When the input value to (s1) and (s2) are T#1d2h33m44s55ms (1 day 2 hours 33 minutes 44 seconds 55 milliseconds) and 2.



- The value to be input to (s1) is time type data.
- The value to be input to (s2) is word (signed), double word (signed), single-precision real or double-precision real type data. (The value to be input to (s2) must be other than 0.)
- Rounding error may occur when specifying single-precision real or double-precision real type data to (s2) by programming tool.

For precautions when setting an input value using a programming tool, refer to the following.

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

■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d).

• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*1	Undefined value

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

Operation error

• An operation error occurs in the following case.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value to be input to (s2) is 0. (Division by 0)	○	○	○	○	○	○

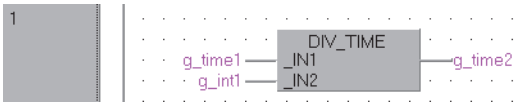
Program example

■DIV_TIME(_E)

The program which performs division ((s1) ÷ (s2)) on time type data input to (s1) and the word (signed) type data input to (s2), and outputs the quotient of the operation result from (d) in time type.

• Function without EN/ENO (DIV_TIME)

[Structured ladder/FBD]

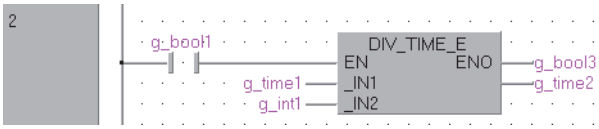


[ST]

```
g_time2 := DIV_TIME(g_time1, g_int1);
```

• Function with EN/ENO (DIV_TIME_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := DIV_TIME_E(g_bool1, g_time1, g_int1, g_time2);
```

5.9 Standard Bistable Function Blocks

Standard bistable function blocks (Set-dominant)

SR(_E)

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

Structured ladder/FBD

ST

```
SR_E (EN, s1, s2, ENO, d);
```

The following function(s) can go in the dotted squares.

SR, SR_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_S1)	Input	Bit
	s2(RESET)		
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error or stop)	Bit
	d(Q1)	Output	Bit

Processing details

■Operation processing

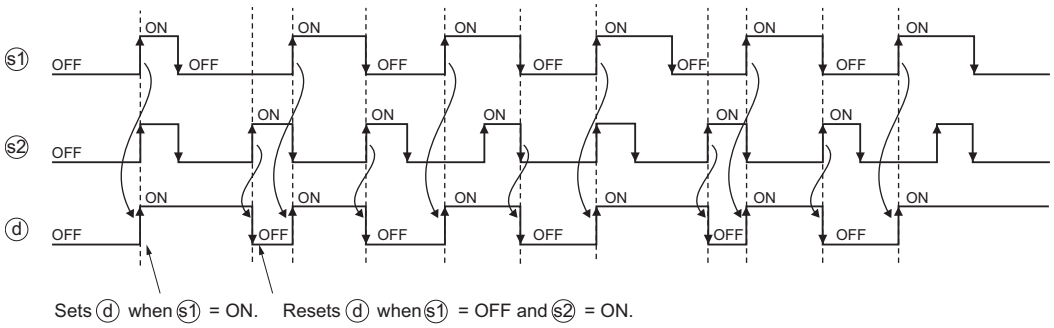
Sets (d) when (s1) is turned ON and resets (d) when (s2) is turned ON while (s1) is OFF. (d) is not reset even when (s2) is turned ON while (s1) is ON.

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

[Timing chart]

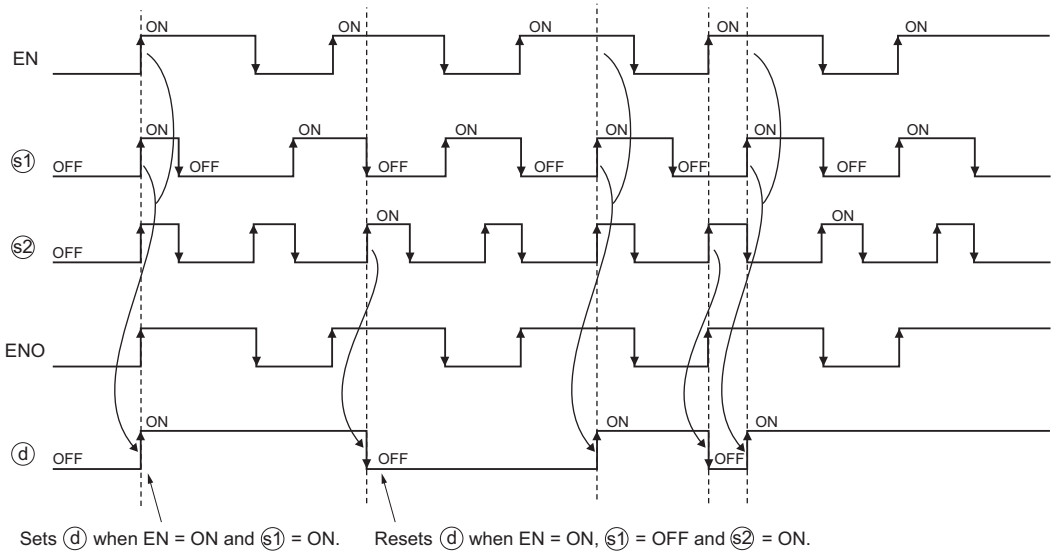


• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]



Operation error

• No operation error occurs.

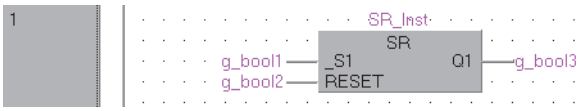
Program example

■SR(_E)

The program which outputs bit type data input to (s1) from (d) and holds the output, and resets the value of (d) only when bit type data input to (s2) is 1 and the data input to (s1) is 0.

• Function without EN/ENO (SR)

[Structured ladder/FBD]

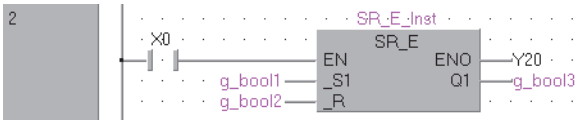


[ST]

```
SR_Inst(_S1:=g_bool1, RESET:=g_bool2, Q1:=g_bool3);
```

• Function with EN/ENO (SR_E)

[Structured ladder/FBD]



[ST]

```
SR_E_Inst(EN:= X0, _S1:=g_bool1, _R:=g_bool2, Q1:=g_bool3, ENO:=Y20);
```

Standard bistable function blocks (Reset-dominant)

RS(_E)

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

Structured ladder/FBD

ST

```
RS_E (EN, s1, s2, ENO, d);
```

The following function(s) can go in the dotted squares.

RS, RS_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(_S)	Input	Bit
	s2(_R1)		
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error or stop)	Bit
	d(Q1)	Output	Bit

Processing details

■Operation processing

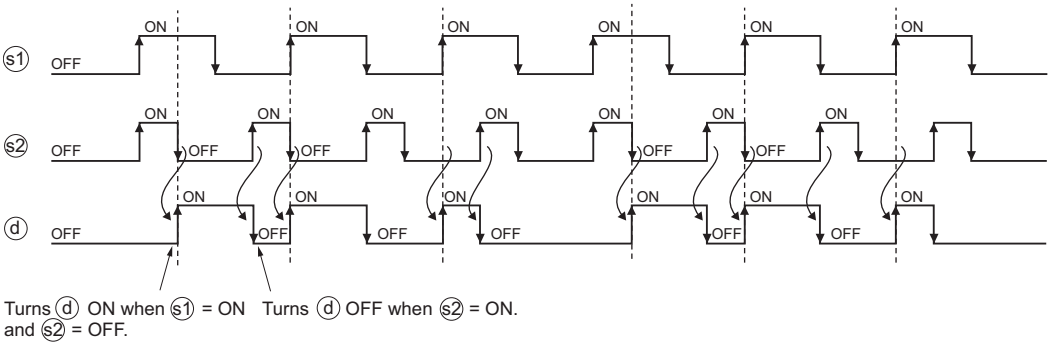
Sets (d) when (s1) is turned ON, and resets (d) when (s2) is turned ON. (d) is not set even when (s1) is turned ON while (s2) is ON.

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

[Timing chart]

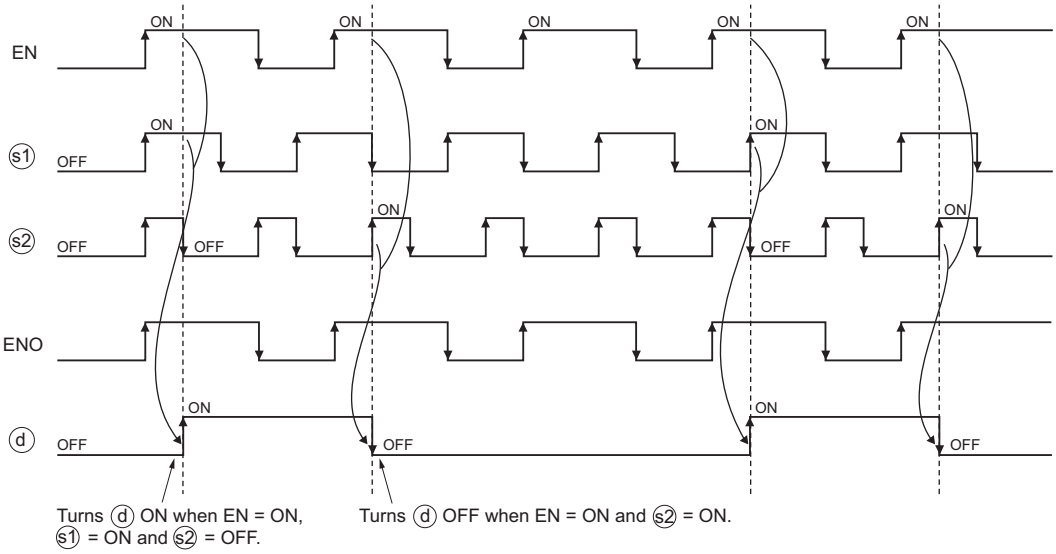


• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]



Operation error

• No operation error occurs.

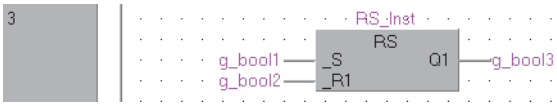
Program example

■RS(_E)

The program which outputs bit type data input to (s1) from (d) and holds the output, and resets forcibly the value of (d) when bit type data input to (s2) is 1.

• Function without EN/ENO (RS)

[Structured ladder/FBD]

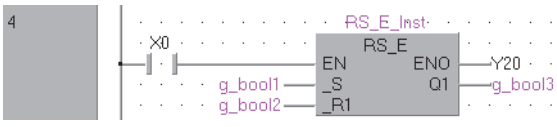


[ST]

```
RS_Inst(_S:=g_bool1, _R1:=g_bool2, Q1:=g_bool3);
```

• Function with EN/ENO (RS_E)

[Structured ladder/FBD]



[ST]

```
RS_E_Inst(EN:=X0, _S:=g_bool1, _R1:=g_bool2, Q1:=g_bool3, ENO:=Y20);
```

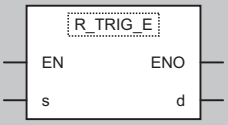
5.10 Standard Edge Detection Function Blocks

Rising edge detector

R_TRIG(_E)

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

Structured ladder/FBD



ST

```
R_TRIG_E (EN, s, ENO, d);
```

The following function(s) can go in the dotted squares.

R_TRIG, R_TRIG_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_CLK)	Input	Bit
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error or stop)	Bit
	d(Q)	Output	Bit

Processing details

■Operation processing

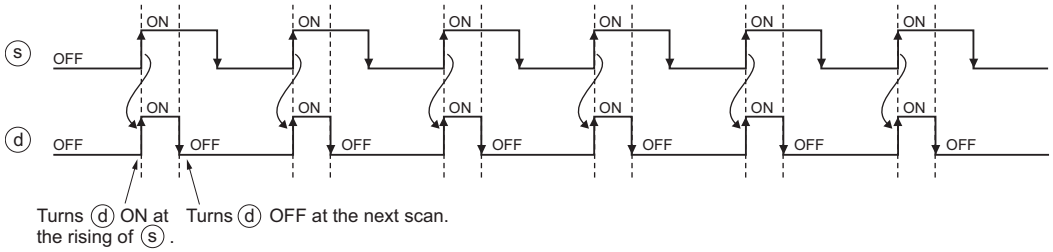
Turns ON (d) for one scan when (s) is turned ON.

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

[Timing chart]

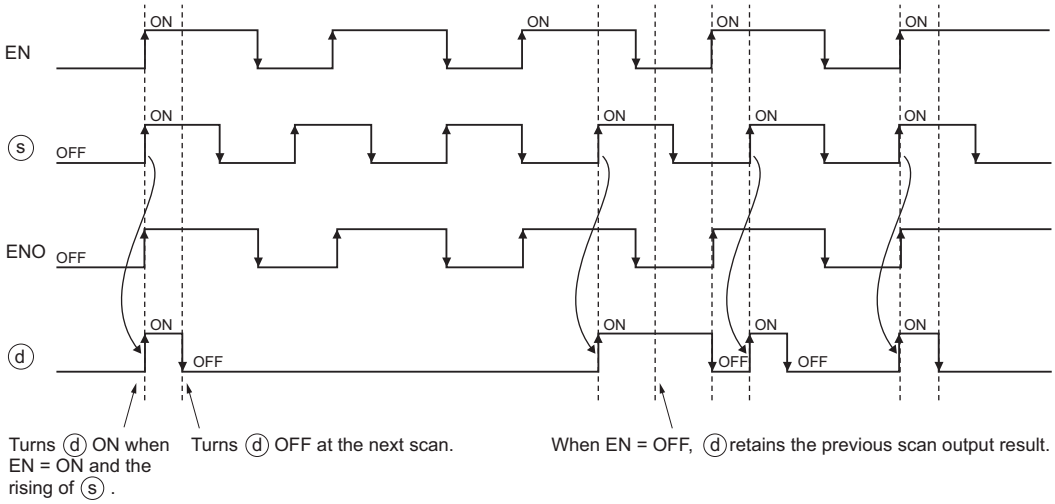


• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]



Operation error

• No operation error occurs.

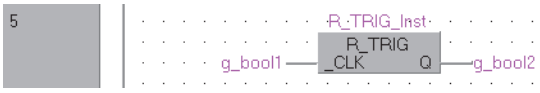
Program example

■R_TRIG(_E)

The program which turns ON (d) for one scan when bit type data input to (s) is turned from OFF to ON.

• Function without EN/ENO (R_TRIG)

[Structured ladder/FBD]

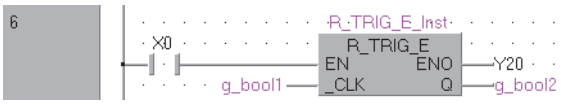


[ST]

```
R_TRIG_Inst(_CLK:=g_bool1, Q:=g_bool2);
```

• Function with EN/ENO (R_TRIG_E)

[Structured ladder/FBD]



[ST]

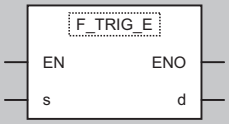
```
R_TRIG_E_Inst(EN:=X0, _CLK:=g_bool1, Q:=g_bool2, ENO:=Y20);
```

Falling edge detector

F_TRIG(_E)

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

Structured ladder/FBD



ST

```
F_TRIG_E (EN, s, ENO, d);
```

The following function(s) can go in the dotted squares.

F_TRIG, F_TRIG_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(_CLK)	Input	Bit
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error or stop)	Bit
	d(Q)	Output	Bit

Processing details

■Operation processing

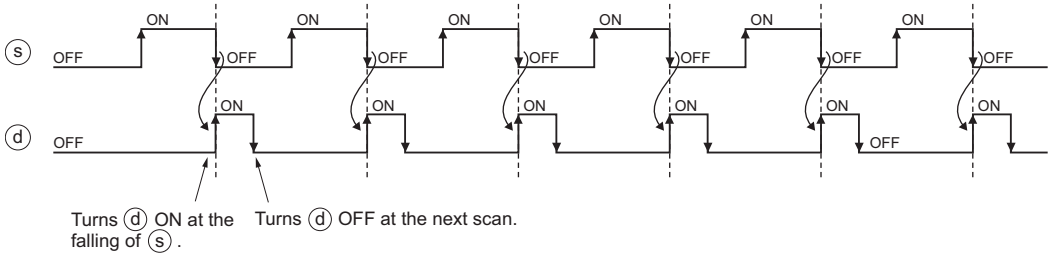
Turns ON (d) for one scan when (s) is turned OFF.

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d).

[Timing chart]

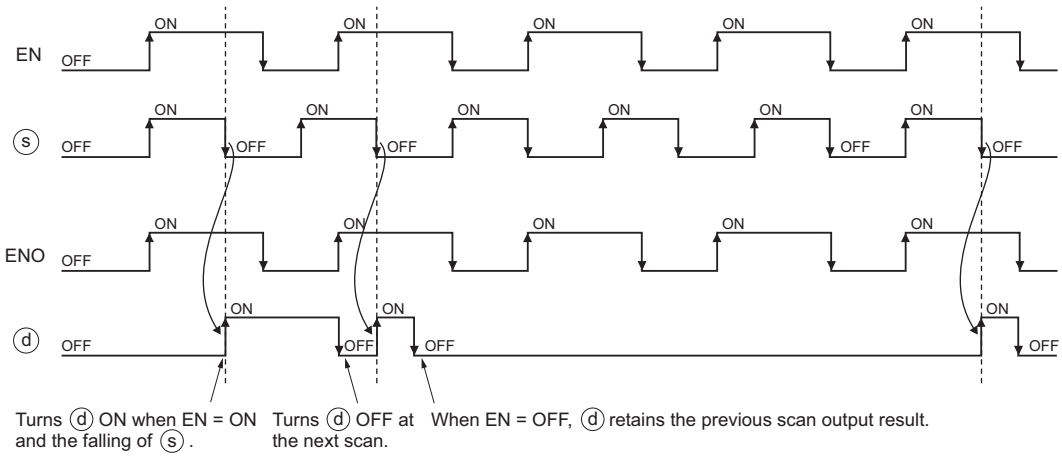


• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]



Operation error

• No operation error occurs.

Program example

■F_TRIG(_E)

The program which turns ON (d) for one scan when bit type data input to (s) is turned from ON to OFF.

• Function without EN/ENO (F_TRIG)

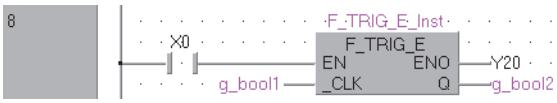
[Structured ladder/FBD]

[ST]

```
F_TRIG_Inst(_CLK:=g_bool1, Q:=g_bool2);
```

• Function with EN/ENO (F_TRIG_E)

[Structured ladder/FBD]



[ST]

```
F_TRIG_E_Inst(EN:=X0, _CLK:=g_bool1, Q:=g_bool2, ENO:=Y20);
```

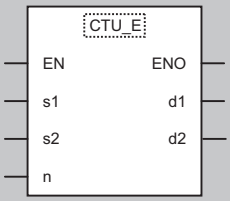
5.11 Standard Counter Function Blocks

Up counter

CTU(_E)

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

Structured ladder/FBD



ST

```
ENO:= CTU_E (EN, s1, s2, n, d1, d2);
```

The following function(s) can go in the dotted squares.

CTU, CTU_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(CU)	Count signal input	Bit
	s2(RESET)	Count reset	Bit
	n(PV)	Maximum count value	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error or stop)	Bit
	d1(Q)	Count match output	Bit
	d2(CV)	Count value	Word (signed)

Processing details

■Operation processing

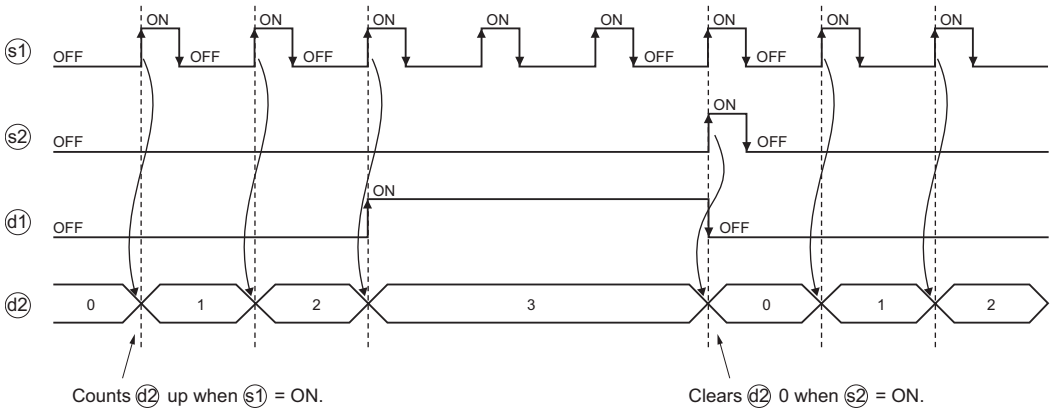
- Counts (d2) when (s1) is turned ON. When the count value (d2) reaches the value input to n, (d1) turns ON. When (s2) is turned ON, (d1) turns OFF and count value (d2) is reset.
- Valid setting range for n is -32768 to 32767. However, if 0 or less is set, (d1) is turned on regardless of the count value of (d2).

■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d1) and (d2).

[Timing chart]

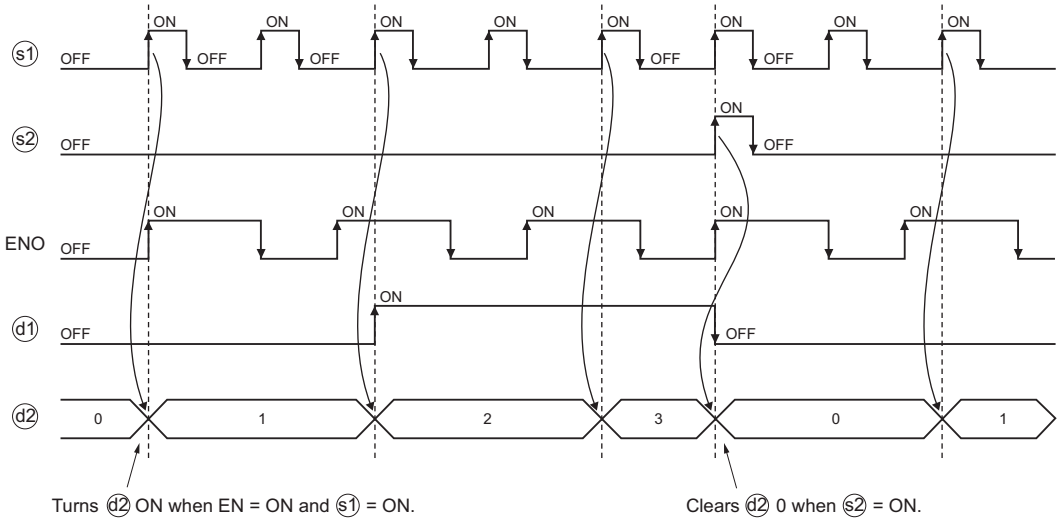


• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]



Operation error

• No operation error occurs.

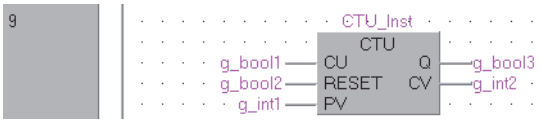
Program example

■CTU(_E)

The program which counts the number of times that bit type data input to (s1) is turned from OFF to ON, and outputs the count value from (d2).

• Function without EN/ENO (CTU)

[Structured ladder/FBD]

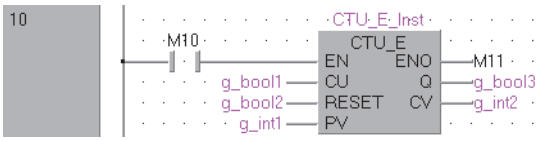


[ST]

```
CTU_Inst(CU:=g_bool1, RESET:=g_bool2, PV:=g_int1, Q:=g_bool3, CV:=g_int2);
```

• Function with EN/ENO (CTU_E)

[Structured ladder/FBD]



[ST]

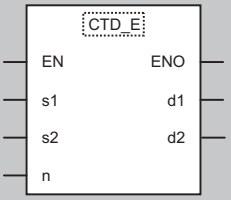
```
CTU_E_Inst(EN:=M10, CU:=g_bool1, RESET:=g_bool2, PV:=g_int1, Q:=g_bool3, CV:=g_int2, ENO:=M11);
```

Down counter

CTD(_E)

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

Structured ladder/FBD



ST

```
CTD_E (EN, s1, s2, n, ENO, d1, d2);
```

The following function(s) can go in the dotted squares.
CTD, CTD_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(CD)	Count signal input	Bit
	s2(LOAD)	Count reset	Bit
	n(PV)	Count start value	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error or stop)	Bit
	d1(Q)	Count match output	Bit
	d2(CV)	Count value	Word (signed)

Processing details

■Operation processing

- Counts down (-1) (d2) when (s1) is turned ON. n sets the initial value for subtraction. (d1) turns ON when count value (d2) reaches 0. When (s2) is turned ON, (d1) turns OFF and initial value for subtraction n is set for count value (d2).
- Valid setting range for n is -32768 to 32767. However, if 0 or less is set, (d1) is turned on regardless of the count value of (d2).

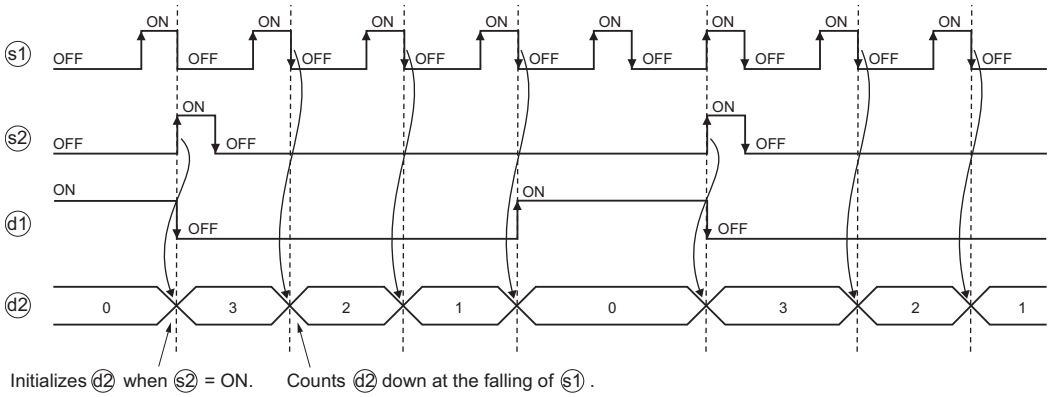
■Operation result

- Function without EN/ENO

An operation is executed and the operation value is output from (d1) and (d2).

[Timing chart]

When n=3



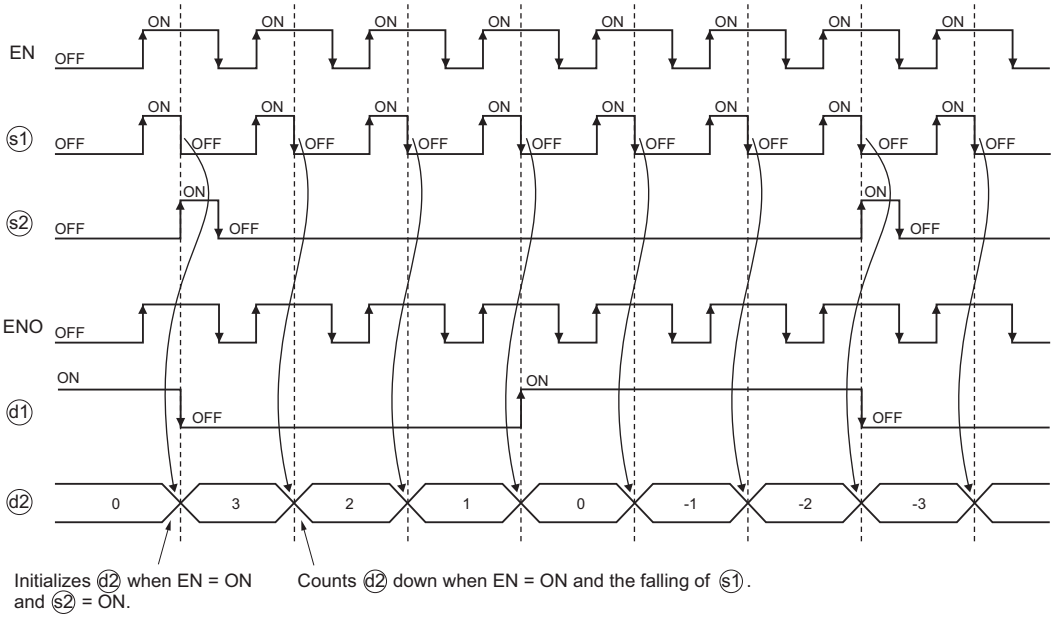
• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d1), (d2)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]

When n=3



Operation error

• No operation error occurs.

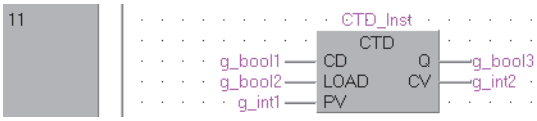
Program example

■CTD(_E)

The program which counts the number of times that bit type data input to (s1) is turned from OFF to ON, and turns ON (d1) when the value of (d2) reaches 0.

• Function without EN/ENO (CTD)

[Structured ladder/FBD]

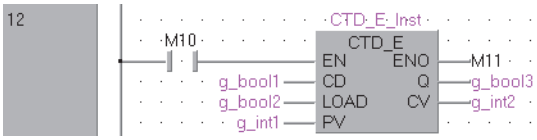


[ST]

```
CTD_Inst(CD:=g_bool1, LOAD:=g_bool2, PV:=g_int1, Q:=g_bool3, CV:=g_int2 );
```

• Function with EN/ENO (CTD_E)

[Structured ladder/FBD]



[ST]

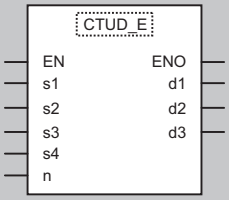
```
CTD_E_Inst(EN:=M10, CD:=g_bool1, LOAD:=g_bool2, PV:=g_int1, Q:=g_bool3, CV:=g_int2, ENO:=M11);
```

Up/Down counter

CTUD(_E)

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

Structured ladder/FBD



ST

```
CTUD_E (EN, s1, s2, s3, s4, n,
ENO, d1, d2, d3);
```

The following function(s) can go in the dotted squares.
CTUD, CTUD_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s1(CU)	Count-up signal input	Bit
	s2(CD)	Count-down signal input	Bit
	s3(RESET)	Count-up reset	Bit
	s4(LOAD)	Count-down reset	Bit
	n(PV)	Maximum count value	Word (signed)
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error or stop)	Bit
	d1(QU)	Count-up match output	Bit
	d2(QD)	Count-down match output	Bit
	d3(CV)	Current count value	Word (signed)

Processing details

■Operation processing

- Counts up (+1) (D3) when (s1) is turned ON, and counts down (-1) (D3) when (s2) is turned ON. n sets the maximum value of counter.

(d2) turns ON when (D3) reaches 0.
(d1) turns ON when (D3) reaches the maximum value n.
Resets (D3) when (s3) turns ON.
The value of n is set to (D3) when (s4) is turned ON.

- Valid setting range for n is -32768 to 32767. However, if 0 or less is set, (d1), (d2) are turned on regardless of the count value of (D3).

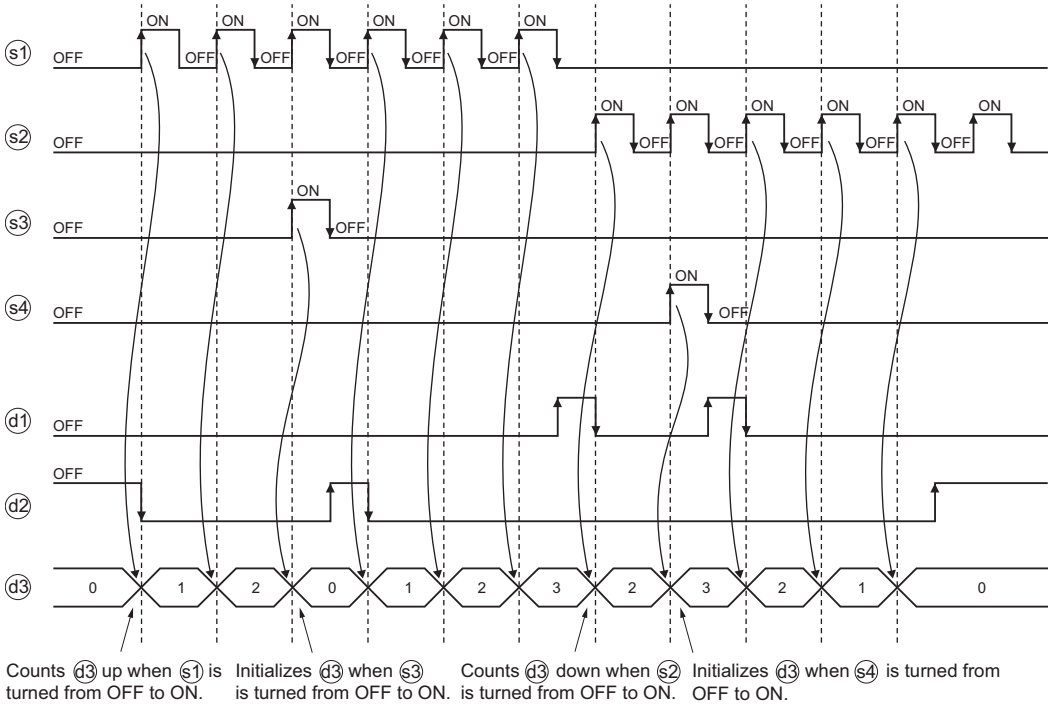
■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d1), (d2), and (D3).

[Timing chart]

When n=3



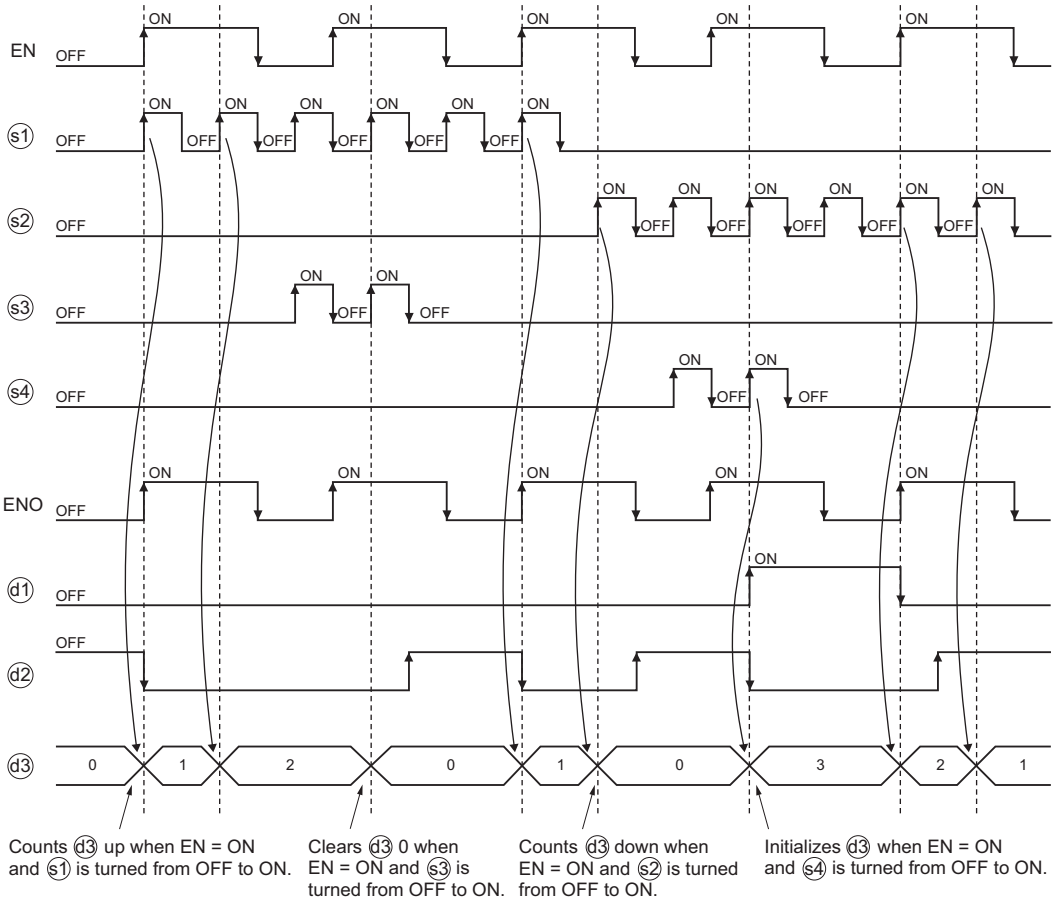
• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d1), (d2), (d3)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]

When n=3



Operation error

- No operation error occurs.

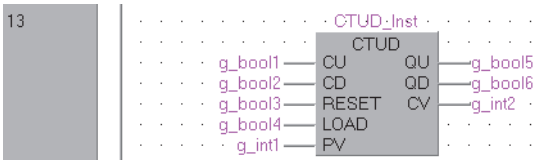
Program example

■CTUD(_E)

The program which counts the number of times that bit type data input to (s1) is turned from OFF to ON, and turns ON (d1) when the value of (D3) reaches the value set at (n). Simultaneously, it counts the number of times that bit type data input to (s2) is turned from OFF to ON, and turns ON (d2) when the value of (D3) reaches 0.

- Function without EN/ENO (CTD)

[Structured ladder/FBD]

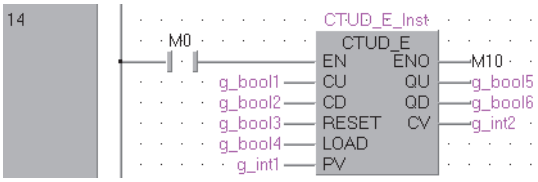


[ST]

```
CTUD_Inst(CU:=g_bool1, CD:=g_bool2, RESET:=g_bool3, LOAD:=g_bool4, PV:=g_int1, QU:=g_bool5, QD:=g_bool6, CV:=g_int2);
```

- Function with EN/ENO (CTD_E)

[Structured ladder/FBD]



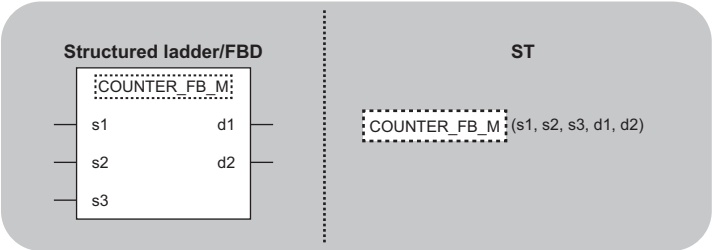
[ST]

```
CTUD_E_Inst(EN:=M0, CU:=g_bool1, CD:=g_bool2, RESET:=g_bool3, LOAD:=g_bool4, PV:=g_int1, QU:=g_bool5, QD:=g_bool6, CV:=g_int2, ENO:=M10);
```

Counter function blocks

COUNTER_FB_M

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



The following function(s) can go in the dotted squares.
COUNTER_FB_M

■Argument

Input/output argument	Name	Description	Data type
Input argument	s1(Coil)	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s2(Preset)	Counter setting value	Word (signed)
	s3(ValueIn)	Counter initial value	Word (signed)
Output argument	d1(ValueOut)	Counter current value	ANY16
	d2(Status)	Output	Bit

Processing details

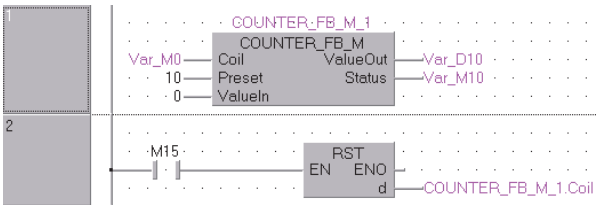
■Operation processing

- Counts the detected rising edge (from OFF to ON) of (s1). It is not counted when (s1) stays ON. The count starts from the value input to (s3) and when the count value reaches the value input to (s2), (d2) turns ON. The current value is stored in (d1).
- Valid setting range for (s2) is 0 to 32767.
- Valid setting range for (s3) is -32768 to 32767. however, if negative value is specified, the initial value is 0.
- When resetting the current value of the counter, reset (s1).

Ex.

When instance name is COUNTER_FB_M_1.

[Structured ladder/FBD]



Resets (s1) (Coil)

[ST]

```
COUNTER_FB_M_1(Coil:=Var_M0,Preset:=10,ValueIn:=0,ValueOut:=Var_D10,Status:=Var_M10);
RST(M15,COUNTER_FB_M_1.Coil);
```

Operation error

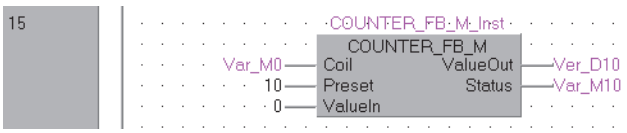
- No operation error occurs.

Program example

■COUNTER_FB_M

The program which counts the number of times that bit type data input to (s1) is turned from OFF to ON, and outputs the count value from (d1).

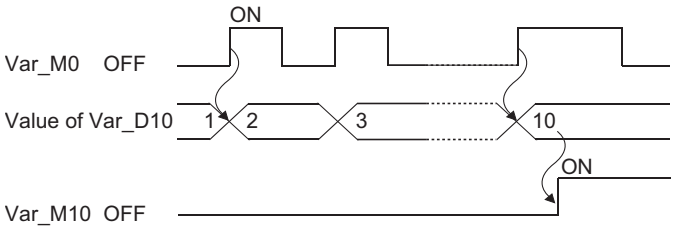
[Structured ladder/FBD]



[ST]

```
COUNTER_FB_M_Inst(Coil:=Var_M0, Preset:=10, ValueIn:=0, ValueOut:=Var_D10, Status:=Var_M10);
```

[Timing chart]

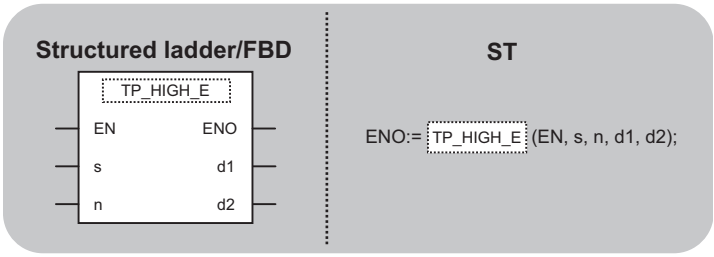


5.12 Standard Timer Function Blocks

Pulse timer

TP(_E), TP_HIGH(_E)

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



The following function(s) can go in the dotted squares.

TP, TP_E, TP_HIGH, TP_HIGH_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(IN)	Input	Bit
	n(PT)	Output time setting value	Time
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error or stop)	Bit
	d1(Q)	Output	Bit
	d2(ET)	Elapsed time	Time

Processing details

■Operation processing

Turns ON (d1) for the duration set to n after (s) is turned ON. The duration (elapsed time) during which (d1) stays ON is set to (d2). When the elapsed time reaches the preset time, (d1) turns OFF.

The elapsed time is not reset even when (d1) turns OFF.

After (d1) turns OFF, it is reset when (s) is OFF.

• TP(_E)

Uses a low-speed timer to count the elapsed time.

Output time can be set between 1ms and 1000ms. The unit is set in Timer limit setting on the PLC system of PLC parameter.

Valid setting range for n is T#0ms to T#3276700ms.

• TP_HIGH(_E)

Uses a high-speed timer to count the elapsed time.

Output time can be set within the following range. The unit is set in Timer limit setting on the PLC system of PLC parameter.

CPU module	Setting range
Basic model QCPU, High Performance model QCPU, Process CPU, Redundant CPU	0.1ms to 100ms
Universal model QCPU, LCPU	0.01ms to 100ms

Valid setting range for n is T#0ms to T#327670ms.

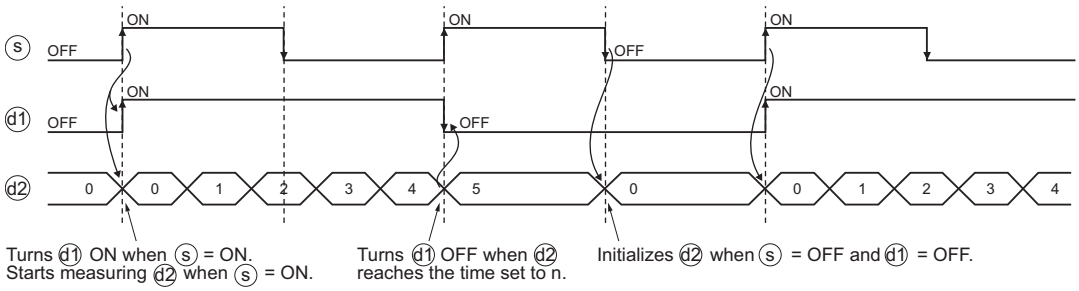
■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d1) and (d2).

[Timing chart]

When n = T#5s (5 seconds)



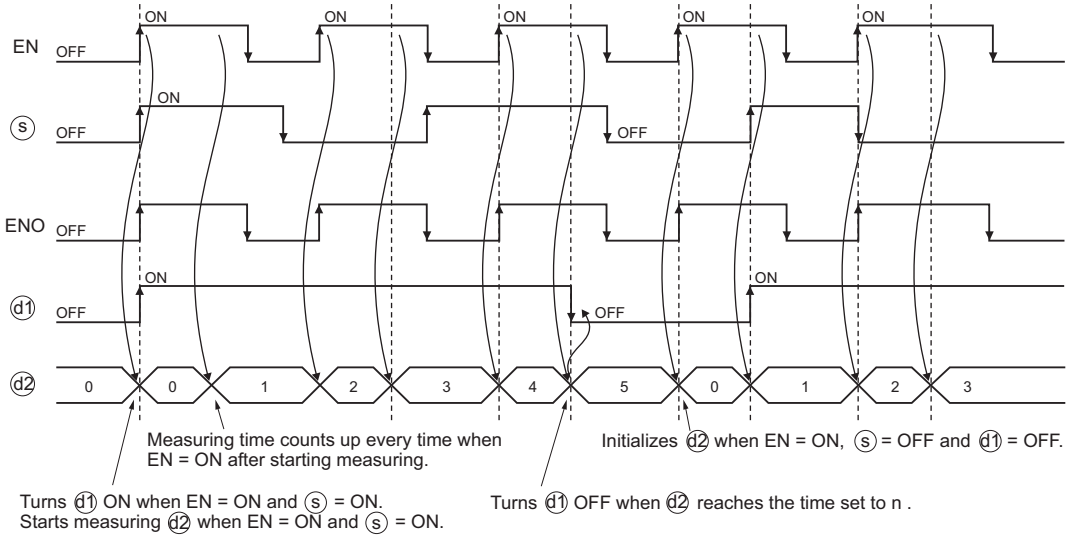
• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d1), (d2)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]

When n = T#5s (5 seconds)



Operation error

• No operation error occurs.

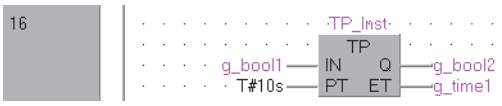
Program example

■TP(_E)

The program which turns ON bit type data of (d1) for 10 seconds after bit type data input to (s) is turned ON.

- Function without EN/ENO (TP)

[Structured ladder/FBD]

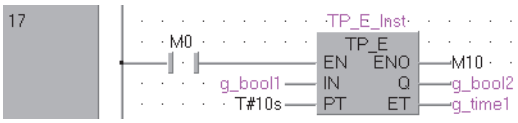


[ST]

```
TP_Inst(IN:=g_bool1, PT:=T#10s, Q:=g_bool2, ET:=g_time1);
```

- Function with EN/ENO (TP_E)

[Structured ladder/FBD]



[ST]

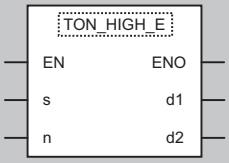
```
TP_E_Inst(EN:=M0, IN:=g_bool1, PT:=T#10s, Q:=g_bool2, ET:=g_time1, ENO:=M10);
```

On delay timer

TON(_E), TON_HIGH(_E)

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

Structured ladder/FBD



ST

```
ENO:= TON_HIGH_E (EN, s, n, d1, d2);
```

The following function(s) can go in the dotted squares.
TON, TON_E, TON_HIGH, TON_HIGH_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(IN)	Input	Bit
	n(PT)	Delay time setting value	Time
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error or stop)	Bit
	d1(Q)	Output	Bit
	d2(ET)	Elapsed time	Time

Processing details

■Operation processing

Turns ON (d1) when (s) is turned ON after the elapse of the time set to n. Elapsed delay time until (d1) is turned ON is set to (d2).

When (s) is turned OFF, (d1) turns OFF and the elapsed delay time is reset.

• TON(_E)

Uses a low-speed timer to count the elapsed time.

Output time can be set between 1ms and 1000ms. The unit is set in Timer limit setting on the PLC system of PLC parameter.

Valid setting range for n is T#0ms to T#3276700ms.

• TON_HIGH(_E)

Uses a high-speed timer to count the elapsed time.

Output time can be set within the following range. The unit is set in Timer limit setting on the PLC system of PLC parameter.

CPU module	Setting range
Basic model QCPU, High Performance model QCPU, Process CPU, Redundant CPU	0.1ms to 100ms
Universal model QCPU, LCPU	0.01ms to 100ms

Valid setting range for n is T#0ms to T#327670ms.

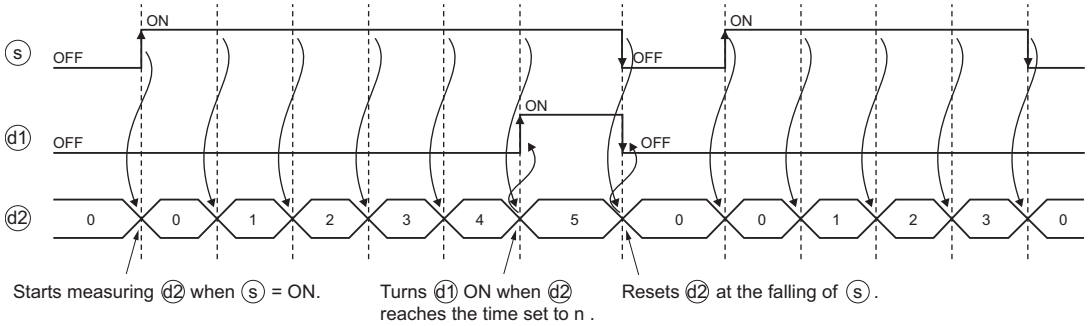
■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d1) and (d2).

[Timing chart]

When n = T#5s (5 seconds)



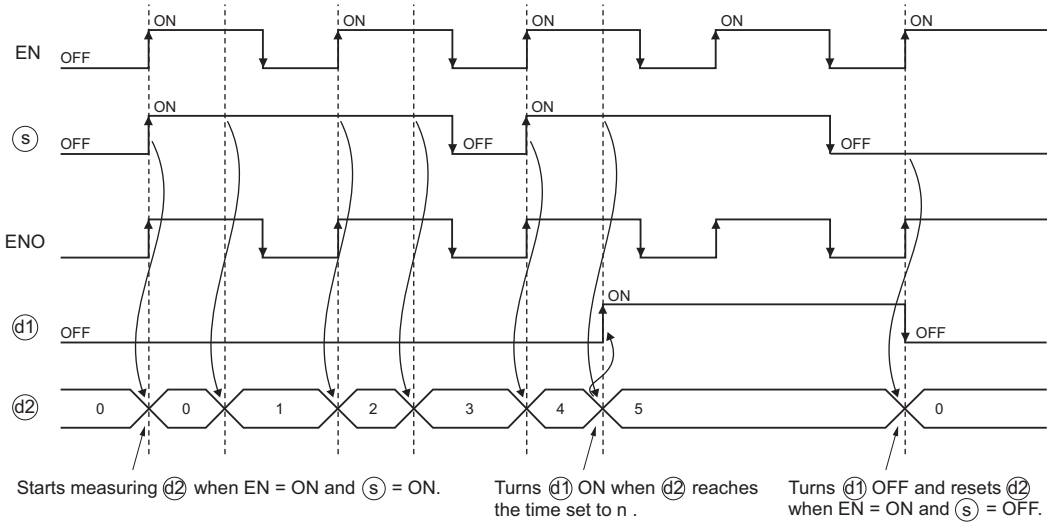
• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d1), (d2)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]

When n = T#5s (5 seconds)



Operation error

- No operation error occurs.

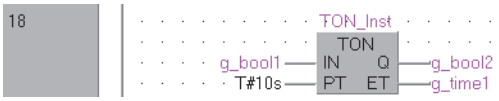
Program example

■TON(_E)

The program which turns ON bit type data of (d1) 10 seconds after bit type data input to (s) is turned ON.

- Function without EN/ENO (TON)

[Structured ladder/FBD]

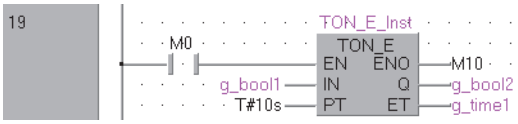


[ST]

```
TON_Inst(IN:=g_bool1, PT:=T#10s, Q:=g_bool2, ET:=g_time1);
```

- Function with EN/ENO (TON_E)

[Structured ladder/FBD]



[ST]

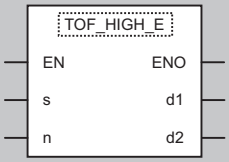
```
TON_E_Inst(EN:=M0, IN:=g_bool1, PT:=T#10s, Q:=g_bool2, ET:=g_time1, ENO:=M10);
```

Off delay timer

TOF(_E), TOF_HIGH(_E)

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

Structured ladder/FBD



ST

```
ENO:= TOF_HIGH_E (EN, s, n, d1, d2);
```

The following function(s) can go in the dotted squares.
TOF, TOF_E, TOF_HIGH, TOF_HIGH_E

■Argument

Input/output argument	Name	Description	Data type
Input argument	EN	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s(IN)	Input	Bit
	n(PT)	Delay time setting value	Time
Output argument	ENO	Execution result (TRUE: Normal, FALSE: Error or stop)	Bit
	d1(Q)	Output	Bit
	d2(ET)	Elapsed time	Time

Processing details

■Operation processing

Turns ON (d1) when (s) is turned ON.
Turns OFF (d1) when (s) is turned from ON to OFF after the elapse of the time set to n. Elapsed time until (d1) is turned OFF is set to (d2).
When (s) is turned ON again, (d1) turns ON and the elapsed time is reset.

• TOF(_E)

Uses a low-speed timer to count the elapsed time.
Output time can be set between 1ms and 1000ms. The unit is set in Timer limit setting on the PLC system of PLC parameter.
Valid setting range for n is T#0ms to T#3276700ms.

• TOF_HIGH(_E)

Uses a high-speed timer to count the elapsed time.
Output time can be set within the following range. The unit is set in Timer limit setting on the PLC system of PLC parameter.

CPU module	Setting range
Basic model QCPU, High Performance model QCPU, Process CPU, Redundant CPU	0.1ms to 100ms
Universal model QCPU, LCPU	0.01ms to 100ms

Valid setting range for n is T#0ms to T#327670ms.

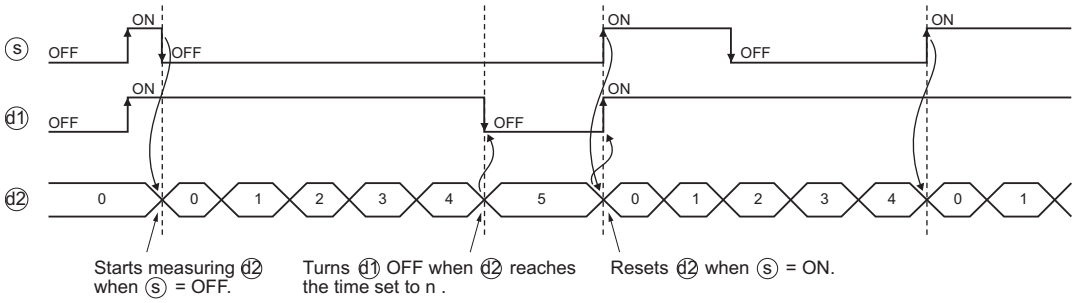
■Operation result

• Function without EN/ENO

An operation is executed and the operation value is output from (d1) and (d2).

[Timing chart]

When n = T#5s (5 seconds)



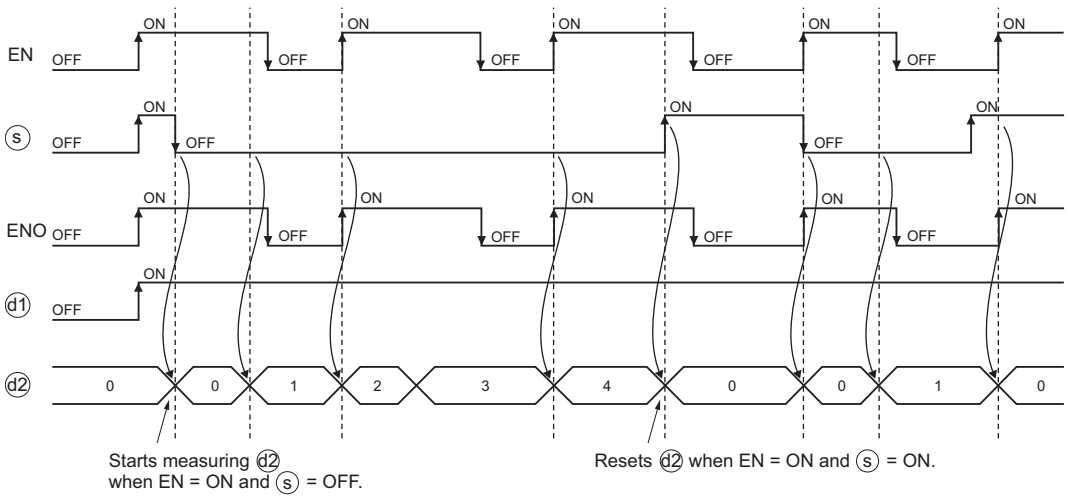
• Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d1), (d2)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]

When n = T#5s (5 seconds)



Operation error

- No operation error occurs.

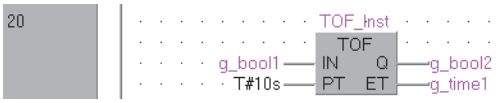
Program example

■TOF(_E)

The program which turns ON bit type data of (d1) when bit type data input to (s) is turned ON, and turns (d1) OFF 10 seconds after (s) is turned OFF.

- Function without EN/ENO (TOF)

[Structured ladder/FBD]

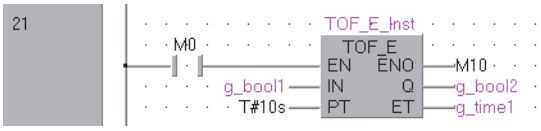


[ST]

```
TOF_Inst(IN:=g_bool1, PT:=T#10s, Q:=g_bool2, ET:=g_time1);
```

- Function with EN/ENO (TOF_E)

[Structured ladder/FBD]



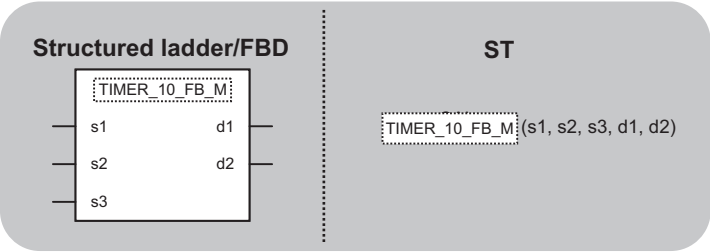
[ST]

```
TOF_E_Inst(EN:=M0, IN:=g_bool1, PT:=T#10s, Q:=g_bool2, ET:=g_time1, ENO:=M10);
```

Timer function blocks

TIMER_□_M

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



The following function(s) can go in the dotted squares.
TIMER_10_FB_M, TIMER_100_FB_M, TIMER_HIGH_FB_M, TIMER_LOW_FB_M, TIMER_CONT_FB_M, TIMER_CONTHFB_M

■Argument

Input/output argument	Name	Description	Data type
Input argument	s1(Coil)	Executing condition (TRUE: Execution, FALSE: Stop)	Bit
	s2(Preset)	Timer setting value	Word (signed)
	s3(ValueIn)	Timer initial value	Word (signed)
Output argument	d1(ValueOut)	Timer current value	ANY16
	d2(Status)	Output	Bit

Processing details

■Operation processing

[TIMER_10_FB_M]

- Starts measuring the current value when the executing condition of (s1) turns ON.
- Starts measuring from the value input to (s3) × 10ms, and when the measuring value reaches to the value input to (s2) × 10ms, (d2) turns ON.
- The current value is output from (d1).
- When the executing condition of (s1) turns OFF, the current value is set to the value input to (s3), and (d2) turns OFF.
- When the unit of measurement (time period) for the high-speed timer is changed from default value of PLC parameter, warning C9047 occurs in compilation.
- Valid setting range for (s2) is 0 to 32767.
- Valid setting range for (s3) is -32768 to 32767. However, if negative value is specified, the initial value is 0.

[TIMER_100_FB_M]

- Starts measuring the current value when the executing condition of (s1) turns ON. Starts measuring from the value input to (s3) × 100ms, and when the measuring value reaches to the value input to (s2) × 100ms, (d2) turns ON.
- The current value is output from (d1).
- When the executing condition of (s1) turns OFF, the current value is set to the value input to (s3), and (d2) turns OFF.
- When the unit of measurement (time period) for the low-speed timer is changed from default value of PLC parameter, warning C9047 occurs in compilation.
- Valid setting range for (s2) is 0 to 32767.
- Valid setting range for (s3) is -32768 to 32767. However, if negative value is specified, the initial value is 0.

[TIMER_HIGH_FB_M]

- The high-speed timer with the unit of measurement from 0.1 to 100ms. Starts measuring the current value when the executing condition of (s1) turns ON. Starts measuring from the value input to (s3) × 0.1 to 100ms, and when the measuring value reaches to the value input to (s2) × 0.1 to 100ms, (d2) turns ON.
- The current value is output from (d1).
- When the executing condition of (s1) turns OFF, the current value is set to the value input to (s3), and (d2) turns OFF.
- The default value of the unit of measurement (time period) for the high-speed timer is 10ms.
- The unit of measurement can be changed within the following range.
- This setting is set in the PLC system setting of the PLC parameter.

CPU module	Setting range
Basic model QCPU, High Performance model QCPU, Process CPU, Redundant CPU	0.1ms to 100ms
Universal model QCPU, LCPU	0.01ms to 100ms

- Valid setting range for (s2) is 0 to 32767.
- Valid setting range for (s3) is -32768 to 32767. However, if negative value is specified, the initial value is 0.

[TIMER_LOW_FB_M]

- The low-speed timer with the unit of measurement from 1 to 1000ms. Starts measuring the current value when the executing condition of (s1) turns ON. Starts measuring from the value input to (s3) × 1 to 1000ms, and when the measuring value reaches to the value input to (s2) × 1 to 1000ms, (d2) turns ON.
- The current value is output from (d1).
- When the executing condition of (s1) turns OFF, the current value is set to the value input to (s3), and (d2) turns OFF.
- The default value of the unit of measurement (time period) for the low-speed timer is 100ms.
- The unit of measurement is from 1 to 1000ms and it can be changed by unit of 1ms.
- This setting is set in the PLC system setting of the PLC parameter.
- Valid setting range for (s2) is 0 to 32767.
- Valid setting range for (s3) is -32768 to 32767. However, if negative value is specified, the initial value is 0.

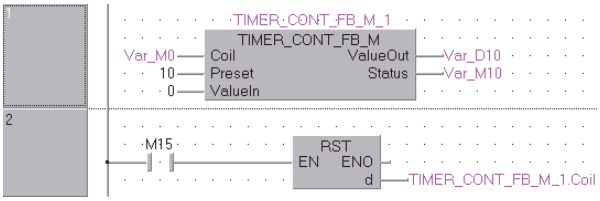
[TIMER_CONT_FB_M,TIMER_CONTHFB_M]

- The retentive timer that measures the time during variable is ON. Starts measuring the current value when the executing condition of (s1) turns ON. The low-speed retentive timer (TIMER_CONT_FB_M) and the high-speed retentive timer (TIMER_CONTHFB_M) are the two types of retentive timer.
- Starts measuring from the value input to (s3) × 1 to 1000ms, and when the count value reaches to the value input to (s2) × 1 to 1000ms, (d2) turns ON. The current value is output from (d1).
- Even when the executing condition of (s1) turns OFF, the ON/OFF statuses of measuring value (d1) and (d2) are retained. When the executing condition of (s1) turns ON again, restarts measuring from the values that are retained.
- The unit of measurement (time period) for retentive timer is same as the low-speed timer (TIMER_LOW_FB_M) and the high-speed timer (TIMER_HIGH_FB_M).
  - Low-speed retentive timer: Low-speed timer
  - High-speed retentive timer: High-speed timer
- Valid setting range for (s2) is 0 to 32767.
- Valid setting range for (s3) is -32768 to 32767. However, if negative value is specified, the initial value is 0.
- When resetting the current value of the retentive timer, reset (s1).

Ex.

When instance name is TIMER_CONT_FB_M_1.

[Structured ladder/FBD]



Resets (s1) (Coil)

[ST]

```
TIMER_CONT_FB_M_1(Coil:=Var_M0,Preset:=10,ValueIn:=0,ValueOut:=Var_D10,Status:=Var_M10);
RST(M15,TIMER_CONT_FB_M_1.Coil);
```

Operation error

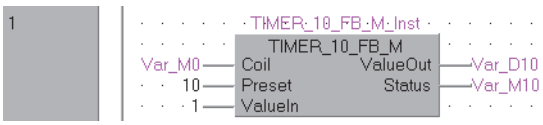
- No operation error occurs.

Program example

■TIMER_10_FB_M

The program which starts measuring from (s3) × 10ms when the executing condition of (s1) turns ON, and when the measuring value reaches to the value input to (s2) × 10ms, (d2) turns ON.

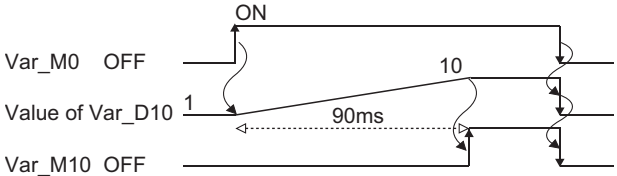
[Structured ladder/FBD]



[ST]

```
TIMER_10_FB_M_Inst(Coil:=Var_M0, Preset:=10, ValueIn:=1, ValueOut:=Var_D10, Status:=Var_M10);
```

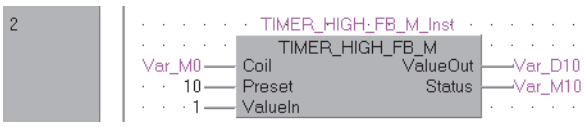
[Timing chart]



■TIMER_HIGH_FB_M

The program which starts measuring from (s3) × 10ms when the executing condition of (s1) turns ON, and when the measuring value reaches to the value input to (s2) × 10ms, (d2) turns ON.

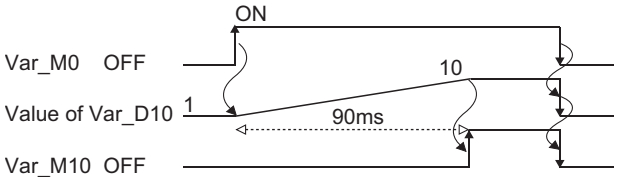
[Structured ladder/FBD]



[ST]

```
TIMER_HIGH_FB_M_Inst(Coil:=Var_M0, Preset:=10, ValueIn:=1, ValueOut:=Var_D10, Status:=Var_M10);
```

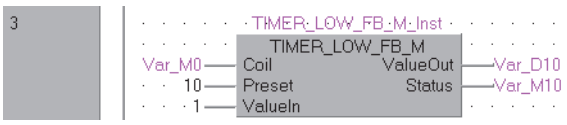
[Timing chart]



■TIMER_LOW_FB_M

The program which starts measuring from (s3) × 10ms when the executing condition of (s1) turns ON, and when the measuring value reaches to the value input to (s2) × 100ms, (d2) turns ON.

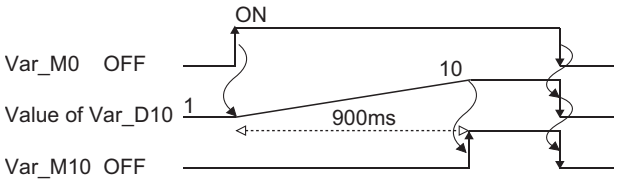
[Structured ladder/FBD]



[ST]

```
TIMER_LOW_FB_M_Inst(Coil:=Var_M0, Preset:=10, ValueIn:=1, ValueOut:=Var_D10, Status:=Var_M10);
```

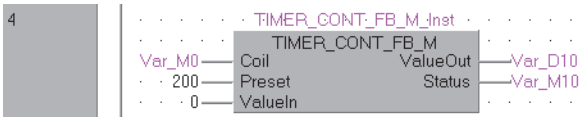
[Timing chart]



■TIMER_CONT_FB_M

The program which measures from (s3) × 10ms, and when the measuring value reaches to the value input to (s2) × 100ms, (d2) turns ON.

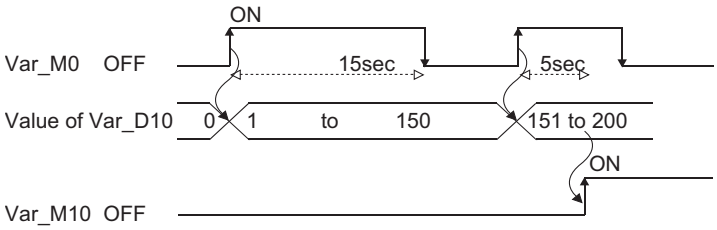
[Structured ladder/FBD]



[ST]

```
TIMER_CONT_FB_M_Inst(Coil:=Var_M0, Preset:=200, ValueIn:=0, ValueOut:=Var_D10, Status:=Var_M10);
```

[Timing chart]



5

6 OPERATOR

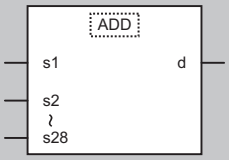
6.1 Arithmetic Operations

Addition

ADD, +

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

Structured ladder/FBD



ST

Not supported

The following operator(s) can go in the dotted squares.

ADD

Structured ladder/FBD

Not supported

ST

d:=s1 + s2…s28;

The following operator(s) can go in the dotted squares.

+

■Argument

Input/output argument	Name	Description	Data type
Input argument	s1 to s28	Input	ANY_NUM
Output argument	d	Output	ANY_NUM

Processing details

■Operation processing

For details of the operation processing, refer to the following.

☞ Page 126 Addition

Operation error

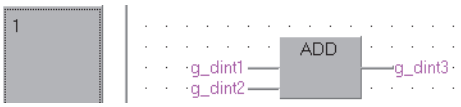
- No operation error occurs.

Program example

■ADD, +

The program which performs addition ((s1) + (s2)) on double word (signed) type data input to (s1) and (s2), and outputs the operation result from (d) in the same data type as that of (s1) and (s2).

[Structured ladder/FBD]



[ST]

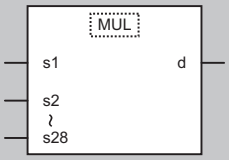
```
g_dint3:=(g_dint1)+(g_dint2);
```

Multiplication

MUL, *

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

Structured ladder/FBD



ST

Not supported

The following operator(s) can go in the dotted squares.

MUL

Structured ladder/FBD

Not supported

ST

d:=s1 * s2…s28;

The following operator(s) can go in the dotted squares.

*

■Argument

Input/output argument	Name	Description	Data type
Input argument	s1 to s28	Input	ANY_NUM
Output argument	d	Output	ANY_NUM

Processing details

■Operation processing

For details of the operation processing, refer to the following.

☞ Page 128 Multiplication

Operation error

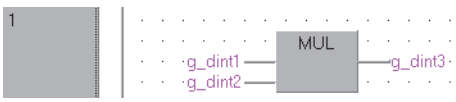
- No operation error occurs.

Program example

■MUL, *

The program which performs multiplication ((s1) × (s2)) on double word (signed) type data input to (s1) and (s2), and outputs the operation result from (d) in the same data type as that of (s1) and (s2).

[Structured ladder/FBD]



[ST]

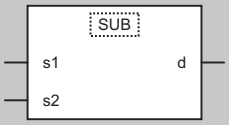
```
g_dint3:= (g_dint1)*(g_dint2);
```

Subtraction

SUB, -

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

Structured ladder/FBD



ST

Not supported

The following operator(s) can go in the dotted squares.

SUB

Structured ladder/FBD

Not supported

ST

d:=s1 - s2;

The following operator(s) can go in the dotted squares.

-

■Argument

Input/output argument	Name	Description	Data type
Input argument	s1	Input	ANY_NUM
	s2		
Output argument	d	Output	ANY_NUM

Processing details

■Operation processing

For details of the operation processing, refer to the following.

☞ Page 130 Subtraction

Operation error

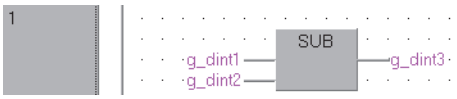
- No operation error occurs.

Program example

■SUB, -

The program which performs subtraction ((s1) - (s2)) on double word (signed) type data input to (s1) and (s2), and outputs the operation result from (d) in the same data type as that of (s1) and (s2).

[Structured ladder/FBD]



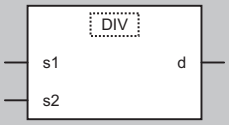

```
[ST]
g_dint3:= (g_dint1) - (g_dint2);
```

Division

DIV, /

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

Structured ladder/FBD



ST

Not supported

The following operator(s) can go in the dotted squares.

DIV

Structured ladder/FBD

Not supported

ST

d:=s1 / s2;

The following operator(s) can go in the dotted squares.

/

■Argument

Input/output argument	Name	Description	Data type
Input argument	s1	Input	ANY_NUM
	s2		
Output argument	d	Output	ANY_NUM

Processing details

■Operation processing

For details of the operation processing, refer to the following.

☞ Page 132 Division

Operation error

• An operation error occurs in the following case.

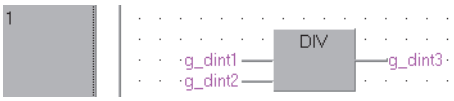
Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value to be input to (s2) is 0. (Division by 0)	○	○	○	○	○	○

Program example

■DIV, /

The program which performs division ((s1) ÷ (s2)) on double word (signed) type data input to (s1) and (s2), and outputs the quotient of the operation result from (d) in the same data type as that of (s1) and (s2).

[Structured ladder/FBD]



[ST]

```
g_dint3:=(g_dint1)/(g_dint2);
```

Remainder

MOD

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

Structured ladder/FBD	ST
Not supported	d:=s1 MOD s2;

The following operator(s) can go in the dotted squares.
MOD

■Argument

Input/output argument	Name	Description	Data type
Input argument	s1	Input	ANY_INT
	s2		
Output argument	d	Output	ANY_INT

Processing details

■Operation processing

For details of the operation processing, refer to the following.

☞ Page 134 Remainder

Operation error

- An operation error occurs in the following cases.

Error code	Description	Q00J/ Q00/ Q01	QnH	QnPH	QnPRH	QnU	LCPU
4100	The value to be input to (s2) is 0. (Division by 0)	○	○	○	○	○	○

Program example

■MOD

The program which performs division ((s1) ÷ (s2)) on double word (signed) type data input to (s1) and (s2), and outputs the remainder of the operation result from (d) in the same data type as that of (s1) and (s2).

```
[ST]
g_dint3:= (g_dint1) MOD (g_dint2);
```

Exponentiation

**

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

Structured ladder/FBD	ST
Not supported	d:=s1 ** s2;

The following operator(s) can go in the dotted squares.

**

■Argument

Input/output argument	Name	Description	Data type
Input argument	s1	Input	ANY_REAL
	s2	Input	ANY_NUM
Output argument	d	Output	ANY_REAL

Processing details

■Operation processing

For details of the operation processing, refer to the following.

☞ Page 136 Exponentiation

Operation error

These functions consist of the following common instructions depending on the data type of (s1) and (s2).

Data type of (s1)	Data type of (s2)	Common instruction used
Single-precision real type	Word (signed) type	LOG, FLT
	Double word (signed) type	LOG, DFLT
	Single-precision real type	LOG
	Double-precision real type	LOGD, DFLTD
Double-precision real type	Word (signed) type	LOGD
	Double word (signed) type	LOGD, FLTD
	Single-precision real type	LOGD, DFLTD
	Double-precision real type	LOGD

• For details of an error which occurs when the operator is executed, refer to the following.

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

Program example

■**

The program which performs exponentiation and outputs the operation result from (d) in the same data type as that of (s1) and (s2).

```
[ST]
g_real2:= (g_real1)**(g_int1);
```

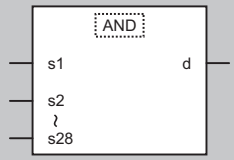
6.2 Logical Operations

Boolean AND, boolean OR, boolean exclusive OR, and boolean NOT

AND, &, OR, XOR, NOT

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

Structured ladder/FBD



ST

Not supported

The following operator(s) can go in the dotted squares.
AND, OR, XOR

Structured ladder/FBD

Not supported

ST

d:=s1 AND s2…s28,d);

The following operator(s) can go in the dotted squares.
AND, &, OR, XOR, NOT

■Argument

Input/output argument	Name	Description	Data type
Input argument	s1 to s28 (s1 only for NOT)	Input	ANY_BIT
Output argument	d	Output	ANY_BIT

Processing details

■Operation processing

For details of the operation processing, refer to the following.

☞ Page 140 Boolean AND, boolean OR, boolean exclusive OR, and boolean NOT

Operation error

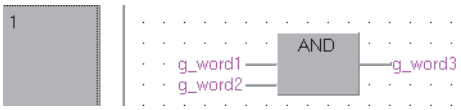
- No operation error occurs.

Program example

■AND, &

The program which performs Boolean AND on bit, word (unsigned)/16-bit string type data input to (s1) and (s2) bit by bit, and outputs the operation result from (d) in the same data type as that of (s1) and (s2).

[Structured ladder/FBD]



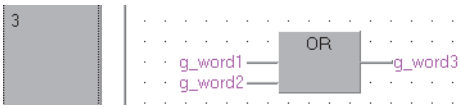
[ST]

```
g_word3:= (g_word1) AND (g_word2);
or
g_word3:= (g_word1) & (g_word2);
```

■OR

The program which performs Boolean OR on bit, word (unsigned)/16-bit string type data input to (s1) and (s2) bit by bit, and outputs the operation result from (d) in the same data type as that of (s1) and (s2).

[Structured ladder/FBD]



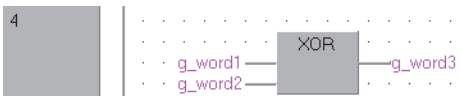
[ST]

```
g_word3:= (g_word1) OR (g_word2);
```

■XOR

The program which performs Boolean XOR on bit, word (unsigned)/16-bit string type data input to (s1) and (s2) bit by bit, and outputs the operation result from (d) in the same data type as that of (s1) and (s2).

[Structured ladder/FBD]



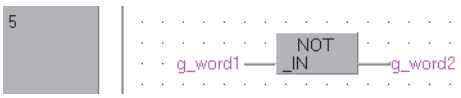
[ST]

```
g_word3:= (g_word1) XOR (g_word2);
```

■NOT

The program which performs Boolean NOT on bit, word (unsigned)/16-bit string type data input to (s1) bit by bit, and outputs the operation result from (d) in the same data type as that of (s1).

[Structured ladder/FBD]



[ST]

```
g_word2:= NOT(g_word1);
```

6.3 Comparison Operations

Comparison

GT, GE, EQ, LE, LT, NE, >, >=, =, <=, <, <>

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

Structured ladder/FBD

ST

Not supported

The following operator(s) can go in the dotted squares.

GT, GE, EQ, LE, LT, NE

Structured ladder/FBD

Not supported

ST

d:=s1 > s2…s28;

The following operator(s) can go in the dotted squares.

>, >=, =, <=, <, <>

■Argument

Input/output argument	Name	Description	Data type
Input argument	s1 to s28 (s1 and s2 only for NE and <>)	Input	ANY_SIMPLE
Output argument	d	Output (TRUE: True value, FALSE: False value)	Bit

Processing details

■Operation processing

For details of the operation processing, refer to the following.

☞ Page 152 Comparison

Operation error

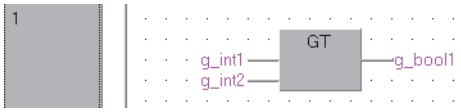
- No operation error occurs.

Program example

■GT, >

The program which performs comparison operation between the values input to (s1) and (s2), and outputs the operation result from (d).

[Structured ladder/FBD]



[ST]

```
g_bool1:=(g_int1)>(g_int2);
```

MEMO

INDEX

MEMO

INSTRUCTION INDEX

Symbols

A

B

C

D

E

F

G

I

L

M

N

O

R

S

T

W

X

I

REVISIONS

*The manual number is given on the bottom left of the back cover.

Revision date	*Manual number	Description
July, 2008 ⋮ June, 2013	SH(NA)-080784ENG-A ⋮ SH(NA)-080784ENG-K	Due to the transition to the e-Manual, the details of revision have been deleted.
January, 2017	SH(NA)-080784ENG-L	Complete revision (layout change)

Japanese manual version SH-080737-P

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

Ethernet is a registered trademark of Fuji Xerox Co., Ltd. in Japan.
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.
The company names, system names and product names mentioned in this manual are either registered trademarks or trademarks of their respective companies.
In some cases, trademark symbols such as '™' or '®' are not specified in this manual.

SH(NA)-080784ENG-L(1701)KWIX
MODEL: Q-KP-OK-E
MODEL CODE: 13JW08

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