MITSUBISHI

Mitsubishi Programmable Controller

MELSEC Q series MELSEC L series

MELSEC-Q/L Structured Programming Manual

Application Functions

QSERIES
LSERIES

● SAFETY PRECAUTIONS ●

(Always read these instructions 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.

REVISIONS

The manual number is written at the bottom left of the back cover.

Print date	Manual number	Revision
Jul., 2008	SH(NA)-080784ENG-A	First edition
Jan., 2009	SH(NA)-080784ENG-B	Model Addition Q00UJCPU, Q00UCPU, Q01UCPU, Q10UDHCPU, Q10UDEHCPU, Q20UDHCPU, Q20UDEHCPU Addition Section 2.1.5, Section 5.5.2, Section 5.5.3 Correction GENERIC TERMS AND ABBREVIATIONS IN THIS MANUAL, Section 3.7.1
Jul., 2009	SH(NA)-080784ENG-C	Model Addition Q00JCPU, Q00CPU, Q01CPU Addition Section 2.2, Section 3.1 Correction Section 1.2, Section 2.1, Section 2.1.1 to Section 2.1.12 are changed to Section 2.2.1 to Section 2.2.12, Section 3.1 is changed to Section 3.2, Chapter 4, Chapter 5 Deletion Section 3.2 to Section 3.7, Appendix 1, Appendix 2
Jan., 2010	SH(NA)-080784ENG-D	Model Addition L02CPU, L26CPU-BT Addition CONDITIONS OF USE FOR THE PRODUCT Correction MANUALS, Section 1.1, Section 1.2, Section 3.1, Chapter 4
Apr., 2010	SH(NA)-080784ENG-E	Model Addition Q50UDEHCPU, Q100UDEHCPU Correction Section 1.1, Section 1.2
Sep., 2010	SH(NA)-080784ENG-F	Addition Section 5.1.39, Section 5.1.40, Section 5.1.41, Section 5.1.42, Section 5.1.43, Section 5.1.44, Section 5.1.45 Correction Section 2.2.1, Section 5.1.30, Section 5.1.31, Section 5.1.33, Section 5.6.1
Jan., 2011	SH(NA)-080784ENG-G	Correction Section 1.1, Section 2.2, Section 5.1.13, Section 5.1.19, Section 5.1.25, Section 5.1.29, Section 5.1.30, Section 5.1.31, Section 5.1.32, Section 5.1.33, Section 5.1.34, Section 5.1.36, Section 5.1.37, Section 5.2.1, Section 5.3.6, Section 5.5.1, Section 5.5.2, Section 5.5.3, Section 5.5.4, Section 5.6.1, Section 5.7.1, Section 5.8.1, Section 5.8.2, Section 5.8.4, Section 5.11.1, Section 5.11.2, Section 5.11.3, Section 5.11.4, Section 5.12.1, Section 5.12.2, Section 5.12.3, Section 5.12.

Print date	Manual number	Revision
Mar., 2011	SH(NA)-080784ENG-H	Addition Section 2.3, Chapter 6 Correction Section 1.1, Section 2.1, Section 2.2.3, Section 2.2.4, Section 2.2.6, Section 5.1.10, Section 5.1.15, Section 5.1.32, Section 5.1.34, Section 5.1.38, Section 5.2.1, Section 5.3.1, Section 5.3.2, Section 5.3.3, Section 5.3.4, Section 5.3.6, Section 5.3.7, Section 5.4.1, Section 5.6.1, Section 5.8.1, Section 5.8.2, Section 5.8.3, Section 5.8.4, Section 5.9.1, Section 5.9.2, Section 5.10.1, Section 5.10.2, Section 5.11.1, Section 5.11.2, Section 5.11.3, Section 5.11.4, Section 5.12.1, Section 5.12.2, Section 5.12.3, Section 5.12.4
Jul., 2011	SH(NA)-080784ENG-I	Model Addition L02CPU-P, L26CPU-PBT Correction Section 1.2, Section 2.3, Chapter 4, Section 5.1.14, Section 5.1.16, Section 5.1.17, Section 5.1.18, Section 5.1.19, Section 5.1.45, Section 5.2.1, Section 5.3.1, Section 5.3.2, Section 5.3.3, Section 5.3.4, Section 5.3.6, Section 5.3.7, Section 5.5.1, Section 5.5.2, Section 5.5.3, Section 5.5.4, Section 5.6.1, Section 5.8.3, Section 5.8.4, Section 5.11.14
Feb., 2013	SH(NA)-080784ENG-J	Descriptions concerning the model additions of a Process CPU, Redundant CPU, Universal model QCPU, and LCPU Model Addition Q02PHCPU, Q06PHCPU, Q12PHCPU, Q25PHCPU, Q12PRHCPU, Q25PRHCPU, Q03UDVCPU, Q04UDVCPU, Q06UDVCPU, Q13UDVCPU, Q26UDVCPU, L02SCPU, L06CPU, L26CPU
Jun., 2013	SH(NA)-080784ENG-K	Model Addition L02SCPU-P, L06CPU-P, L26CPU-P Correction Section 1.2

Japanese manual version SH-080737-O

INTRODUCTION

Thank you for purchasing the Mitsubishi 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.

CONTENTS

5. APPLICATION FUNCTIONS 5 - 1 to 5 - 242

MANUALS

The manuals related to this product are listed below.

Order each manual as needed, referring to the following lists.

(1) Structured programming

Manual name	Manual number (model code)
MELSEC-Q/L/F Structured Programming Manual (Fundamentals) Methods and languages for structured programming (Sold separately)	SH-080782ENG (13JW06)
MELSEC-Q/L Structured Programming Manual (Common Instructions) Specifications and functions of common instructions, such as sequence instructions, basic instructions, and application instructions, that can be used in structured programs (Sold separately)	SH-080783ENG (13JW07)
MELSEC-Q/L Structured Programming Manual (Special Instructions) 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 (Sold separately)	SH-080785ENG (13JW09)

(2) Operation of GX Works2

Manual name	Manual number (model code)
GX Works2 Version 1 Operating Manual (Common) System configuration, parameter settings, and online operations of GX Works2, which are common to Simple projects and Structured projects (Sold separately)	SH-080779ENG (13JU63)
GX Works2 Version 1 Operating Manual (Structured Project) Operations, such as programming and monitoring in Structured projects, of GX Works2 (Sold separately)	SH-080781ENG (13JU65)
GX Works2 Beginner's Manual (Structured Project) 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. (Sold separately)	SH-080788ENG (13JZ23)

POINT

Operating manuals in PDF format are stored on the CD-ROM of the software package. Printed manuals are sold separately. To order manuals, please provide the manual number (model code) listed in the table above.

MEMO

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 list of 'Related manuals'.

(1) Operation of GX Works2

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

(2) Operations in each programming language

For details of instructions used in each programming language, refer to the section 3 on the next page

Purpose		GX Works2 Beginner's Manual		GX Works2 Version 1 Operating Manual	
		Simple Project	Structured Project	Simple Project	Structured Project
Simple Project	Ladder	Outline		Details	
	SFC	*1 Outline		Details	
	ST		Outline		Details
Structured Project/FBD	Ladder	Outline		Details	
	SFC	*1 Outline		Details	
	Structured ladder/ FBD		Outline		Details
	ST		Outline		Details

*1: MELSAP3 and FX series SFC only

(3) Details of instructions in each programming language

Purpose		MELSEC-Q/L/F Structured Programming Manual	MELSEC-Q/L Structured Programming Manual			MELSEC-Q/L Programming Manual	MELSEC-Q/L/QnA Programming Manual		MELSEC-Q Programming /Structured Programming Manual	Manual for module to be used
		Fundamentals	Common Instructions	Special Instructions	Application Functions	Common Instruction	PID Control Instructions	SFC	Process Control Instructions	–
All languages	Learning details of programmable controller CPU error codes, special relays, and special registers									*1 Details
Using ladder language	Learning the types and details of common instructions					Details				
	Learning the types and details of instructions for intelligent function modules									Details
	Learning the types and details of instructions for network modules									Details
	Learning the types and details of instructions for the PID control function						Details			
	Learning the types and details of the process control instructions								Details	
Using SFC language	Learning details of specifications, functions, and instructions of SFC (MELSAP3)							Details		
Using structured ladder/ FBD/ST language	Learning the fundamentals for creating a structured program	Details								
	Learning the types and details of common instructions		Details							
	Learning the types and details of instructions for intelligent function modules			Outline						Details
	Learning the types and details of instructions for network modules			Outline						Details
	Learning the types and details of instructions for the PID control function			Outline			Details			
	Learning the types and details of application functions				Details					
	Learning the types and details of the process control instructions								Details	

*1 Refer to the User's Manual (Hardware Design, Maintenance and Inspection) for the CPU module used.

1.2 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
GX Works2	Product name of the software package for the MELSEC programmable controllers
Basic model QCPU	A generic term for Q00JCPU, Q00CPU, and Q01CPU
High Performance model QCPU	A generic term for Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU, and Q25HCPU
Process CPU	A generic term for the Q02PHCPU, Q06PHCPU, Q12PHCPU, and Q25PHCPU
Redundant CPU	A generic term for the Q12PRHCPU and Q25PRHCPU
Universal model QCPU	A generic term for 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
QCPU (Q mode)	A generic term for the Basic model QCPU, High Performance model QCPU, Process CPU, Redundant CPU, and Universal model QCPU
LCPU	A generic term for the L02SCPU, L02SCPU-P, L02CPU, L02CPU-P, L06CPU, L06CPU-P, L26CPU, L26CPU-P, L26CPU-BT, and L26CPU-PBT
CPU module	A generic term for QCPU (Q mode) and LCPU
Personal computer	A generic term for personal computer on which Windows® operates
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
Special instruction	A generic term for module dedicated instructions, PID control instructions, socket communication function instructions, built-in I/O function instructions, and data logging function instructions
Application function	A generic term for the functions, such as functions and function blocks, defined in IEC61131-3. (The functions are executed with a set of common instructions in a programmable controller.)

MEMO

2 FUNCTION TABLES

2.1 How to Read Function Tables

Function tables in Section 2.2 are shown in the following format.

Function name	Argument	Processing details	Page
ADD_E	ⓢ1 ,ⓢ2 ,...ⓢ28 ,ⓓ (Number of pins variable)	Outputs the sum (ⓢ1 +ⓢ2 +···+ⓢ28) of input values.	5-135
MUL_E	ⓢ1 ,ⓢ2 ,...ⓢ28 ,ⓓ (Number of pins variable)	Outputs the product (ⓢ1 ×ⓢ2 ×···×ⓢ28) of input values.	5-138
SUB_E	ⓢ1 , ⓢ2 , ⓓ	Outputs the difference (ⓢ1 −ⓢ2) between input values.	5-141
DIV_E	ⓢ1 , ⓢ2 , ⓓ	Outputs the quotient (ⓢ1 ÷ⓢ2) of input values.	5-144
			5-147

① ② ③ ④

Description

① .....Indicates the functions used in a program. 'Function name(_E)' is used as a function with EN/ENO.

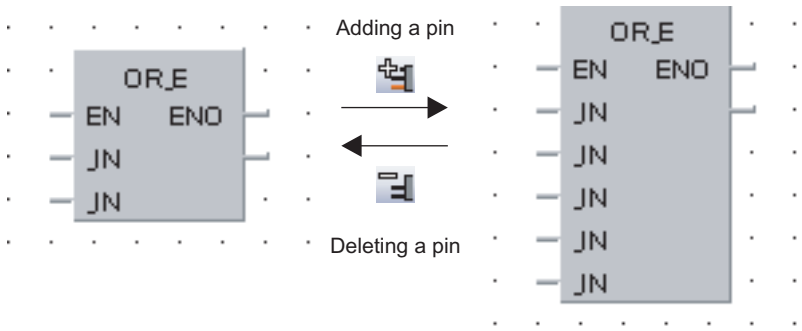
② .....Indicates the arguments of the function.

ⓢ : Source .....Stores data before operation.

ⓓ : Destination.....Indicates the destination of data after operation.

(Number of pins variable).....Allows the number of ⓢ (source) to be changed in the range from 2 to 28.

Changing the number of pins



③ .....Indicates the processing details of each function.

④ .....Indicates the pages on which the functions are explained.

2.2 Function Tables

2.2.1 Type conversion functions

Function name	Argument	Processing details	Page
BOOL_TO_INT(_E)	ⓢ , ⓓ	Converts bit type data into word (signed) or double word (signed) type data.	5-2
BOOL_TO_DINT(_E)	ⓢ , ⓓ		
BOOL_TO_STR(_E)	ⓢ , ⓓ	Converts bit type data into string type data.	5-5
BOOL_TO_WORD(_E)	ⓢ , ⓓ	Converts bit type data into word (unsigned)/16-bit string or double word (unsigned)/32-bit string type date.	5-7
BOOL_TO_DWORD(_E)	ⓢ , ⓓ		
BOOL_TO_TIME(_E)	ⓢ , ⓓ	Converts bit type data into time type data.	5-10
INT_TO_DINT(_E)	ⓢ , ⓓ	Converts word (signed) type data into double word (signed) type data.	5-12
DINT_TO_INT(_E)	ⓢ , ⓓ	Converts double word (signed) type data into word (signed) type data.	5-14
INT_TO_BOOL(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into bit type data.	5-16
DINT_TO_BOOL(_E)	ⓢ , ⓓ		
INT_TO_REAL(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into single-precision real type data.	5-19
DINT_TO_REAL(_E)	ⓢ , ⓓ		
INT_TO_LREAL(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into double-precision real type data.	5-22
DINT_TO_LREAL(_E)	ⓢ , ⓓ		
INT_TO_STR(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into string type data.	5-25
DINT_TO_STR(_E)	ⓢ , ⓓ		
INT_TO_WORD(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into word (unsigned)/16-bit string type data.	5-29
DINT_TO_WORD(_E)	ⓢ , ⓓ		
INT_TO_DWORD(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into double word (unsigned)/32-bit string type data.	5-32
DINT_TO_DWORD(_E)	ⓢ , ⓓ		
INT_TO_BCD(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into BCD type data.	5-35
DINT_TO_BCD(_E)	ⓢ , ⓓ		
INT_TO_TIME(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into time type data.	5-38
DINT_TO_TIME(_E)	ⓢ , ⓓ		
REAL_TO_INT(_E)	ⓢ , ⓓ	Converts single-precision real type data into word (signed) or double word (signed) type data.	5-41
REAL_TO_DINT(_E)	ⓢ , ⓓ		
LREAL_TO_INT(_E)	ⓢ , ⓓ	Converts double-precision real type data into word (signed) or double word (signed) type data.	5-44
LREAL_TO_DINT(_E)	ⓢ , ⓓ		
REAL_TO_LREAL(_E)	ⓢ , ⓓ	Converts single-precision real type data into double-precision real type data.	5-47
LREAL_TO_REAL(_E)	ⓢ , ⓓ	Converts double-precision real type data into single-precision real type data.	5-50
REAL_TO_STR(_E)	ⓢ , ⓓ	Converts single-precision real type data into string type (exponential form) data.	5-53
WORD_TO_BOOL(_E)	ⓢ , ⓓ	Converts word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data into bit type data.	5-57
DWORD_TO_BOOL(_E)	ⓢ , ⓓ		
WORD_TO_INT(_E)	ⓢ , ⓓ	Converts word (unsigned)/16-bit string type data into word (signed) or double word (signed) type data.	5-60
WORD_TO_DINT(_E)	ⓢ , ⓓ		
DWORD_TO_INT(_E)	ⓢ , ⓓ	Converts double word (unsigned)/32-bit string type data into word (signed) or double word (signed) type data.	5-63
DWORD_TO_DINT(_E)	ⓢ , ⓓ		
WORD_TO_DWORD(_E)	ⓢ , ⓓ	Converts word (unsigned)/16-bit string type data into double word (unsigned)/32-bit string type data.	5-66
DWORD_TO_WORD(_E)	ⓢ , ⓓ	Converts double word (unsigned)/32-bit string type data into word (unsigned)/16-bit string type data.	5-69
WORD_TO_STR(_E)	ⓢ , ⓓ	Converts word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data into string type data.	5-72
DWORD_TO_STR(_E)	ⓢ , ⓓ		
WORD_TO_TIME(_E)	ⓢ , ⓓ	Converts word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data into time type data.	5-75
DWORD_TO_TIME(_E)	ⓢ , ⓓ		
STR_TO_BOOL(_E)	ⓢ , ⓓ	Converts string type data into bit type data.	5-78

Function name	Argument	Processing details	Page
STR_TO_INT(_E)	ⓢ , ⓓ	Converts string type data into word (signed) or double word (signed) type data.	5-81
STR_TO_DINT(_E)	ⓢ , ⓓ		
STR_TO_REAL(_E)	ⓢ , ⓓ	Converts string type data into single-precision real type data.	5-84
STR_TO_WORD(_E)	ⓢ , ⓓ	Converts string type data into word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data.	5-88
STR_TO_DWORD(_E)	ⓢ , ⓓ		
STR_TO_TIME(_E)	ⓢ , ⓓ	Converts string type data into time type data.	5-92
STR_TO_BCD(_E)	ⓢ , ⓓ	Converts string type data into BCD type data.	5-95
BCD_TO_INT(_E)	ⓢ , ⓓ	Converts BCD type data into word (signed) or double word (signed) type data.	5-100
BCD_TO_DINT(_E)	ⓢ , ⓓ		
BCD_TO_STR(_E)	ⓢ , ⓓ	Converts BCD type data into string type data.	5-104
TIME_TO_BOOL(_E)	ⓢ , ⓓ	Converts time type data into bit type data.	5-107
TIME_TO_INT(_E)	ⓢ , ⓓ	Converts time type data into word (signed) or double word (signed) type data.	5-109
TIME_TO_DINT(_E)	ⓢ , ⓓ		
TIME_TO_STR(_E)	ⓢ , ⓓ	Converts time type data into string type data.	5-112
TIME_TO_WORD(_E)	ⓢ , ⓓ	Converts time type data into word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data.	5-114
TIME_TO_DWORD(_E)	ⓢ , ⓓ		
BITARR_TO_INT(_E)	ⓢ , n, ⓓ	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.	5-117
BITARR_TO_DINT(_E)	ⓢ , n, ⓓ		
INT_TO_BITARR(_E)	ⓢ , n, ⓓ	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.	5-119
DINT_TO_BITARR(_E)	ⓢ , n, ⓓ		
CPY_BITARR(_E)	ⓢ , n, ⓓ	Copies specified number of bits from bit array.	5-121
GET_BIT_OF_INT(_E)	ⓢ , n, ⓓ	Reads a value of specified bit of word (signed) type data.	5-123
SET_BIT_OF_INT(_E)	ⓢ , n, ⓓ	Writes a value to the specified bit of word (signed) type data.	5-125
CPY_BIT_OF_INT(_E)	ⓢ , n1, n2, ⓓ	Copies a specified bit of word (signed) type data to the specified bit of another word (signed) type data.	5-127
GET_BOOL_ADDR	ⓢ , ⓓ	Converts the type of data into bit type.	5-129
GET_INT_ADDR	ⓢ , ⓓ	Converts the type of data into word (signed) type.	
GET_WORD_ADDR	ⓢ , ⓓ	Converts the type of data to word (unsigned)/16-bit string type.	

2.2.2 Standard functions of one numeric variable

Function name	Argument	Processing details	Page
ABS(_E)	ⓢ , ⓓ	Outputs the absolute value of an input value.	5-131

2.2.3 Standard arithmetic functions

Function name	Argument	Processing details	Page
ADD_E	ⓢ1 ,ⓢ2 ,...ⓢn ,ⓓ (Number of pins variable)	Outputs the sum (ⓢ1 +ⓢ2 +···+ⓢn) of input values.	5-135
MUL_E	ⓢ1 ,ⓢ2 ,...ⓢn ,ⓓ (Number of pins variable)	Outputs the product (ⓢ1 ×ⓢ2 ×···×ⓢn) of input values.	5-138
SUB_E	ⓢ1 , ⓢ2 , ⓓ	Outputs the difference (ⓢ1 −ⓢ2) between input values.	5-141
DIV_E	ⓢ1 , ⓢ2 , ⓓ	Outputs the quotient (ⓢ1 ÷ⓢ2) of input values.	5-144
MOD(_E)	ⓢ1 , ⓢ2 , ⓓ	Outputs the remainder after division of input values (ⓢ1 ÷ⓢ2).	5-147
EXPT(_E)	ⓢ1 , ⓢ2 , ⓓ	Outputs the exponentiation of an input value.	5-150
MOVE(_E)	ⓢ , ⓓ	Moves the input value to ⓓ .	5-153

2.2.4 Standard bitwise Boolean functions

Function name	Argument	Processing details	Page
AND_E	ⓢ1 ,ⓢ2 ,...ⓢn ,ⓓ (Number of pins variable)	Outputs the Boolean AND of input values.	5-157
OR_E	ⓢ1 ,ⓢ2 ,...ⓢn ,ⓓ (Number of pins variable)	Outputs the Boolean OR of input values.	
XOR_E	ⓢ1 ,ⓢ2 ,...ⓢn ,ⓓ (Number of pins variable)	Outputs the Boolean exclusive OR of input values.	
NOT(_E)	ⓢ , ⓓ	Outputs the Boolean NOT of input values.	

2.2.5 Standard selection functions

Function name	Argument	Processing details	Page
SEL(_E)	ⓢ1 , ⓢ2 , ⓢ3 , ⓓ	Outputs the value selected from the input values.	5-162
MAXIMUM(_E)	ⓢ1 ,ⓢ2 ,...ⓢn ,ⓓ (Number of pins variable)	Outputs the maximum value of the input values.	5-165
MINIMUM(_E)	ⓢ1 ,ⓢ2 ,...ⓢn ,ⓓ (Number of pins variable)	Outputs the minimum value of the input values.	
LIMITATION(_E)	ⓢ1 , ⓢ2 , ⓢ3 , ⓓ	Outputs the input value controlled by the upper and lower limit control.	5-168
MUX(_E)	n, ⓢ1 , ⓢ2 ,...ⓢn ,ⓓ (Number of pins variable)	Outputs one of the multiple input values.	5-171

2.2.6 Standard comparison functions

Function name	Argument	Processing details	Page
GT_E	ⓢ1 ,ⓢ2 ,...ⓢ20 ,ⓓ (Number of pins variable)	Outputs the comparison value of an input value.	5-174
GE_E	ⓢ1 ,ⓢ2 ,...ⓢ20 ,ⓓ (Number of pins variable)		
EQ_E	ⓢ1 ,ⓢ2 ,...ⓢ20 ,ⓓ (Number of pins variable)		
LE_E	ⓢ1 ,ⓢ2 ,...ⓢ20 ,ⓓ (Number of pins variable)		
LT_E	ⓢ1 ,ⓢ2 ,...ⓢ20 ,ⓓ (Number of pins variable)		
NE_E	ⓢ1 , ⓢ2 , ⓓ		

2.2.7 Standard character string functions

Function name	Argument	Processing details	Page
MID(_E)	ⓢ, n1, n2, ⓓ	Outputs the specified number of characters, extracted from the specified start position in the input character string.	5-178
CONCAT(_E)	ⓢ1 , ⓢ2 , ⓓ	Concatenates the character strings and outputs the operation result.	5-181
INSERT(_E)	ⓢ1 , ⓢ2 , n, ⓓ (Number of pins variable)	Inserts a character string between other character strings and outputs the operation result.	5-184
DELETE(_E)	ⓢ, n1, n2, ⓓ	Deletes the specified range in a character string and outputs the operation result.	5-187
REPLACE(_E)	ⓢ1 , ⓢ2 , n1, n2, ⓓ	Replaces the specified range in a character string with the specified character string and outputs the operation result.	5-190

2.2.8 Functions of time data types

Function name	Argument	Processing details	Page
ADD_TIME(_E)	ⓢ1 , ⓢ2 , ⓓ	Outputs the sum (ⓢ1 +ⓢ2) of the input values (time type).	5-193
SUB_TIME(_E)	ⓢ1 , ⓢ2 , ⓓ	Outputs the difference (ⓢ1 -ⓢ2) of input values (time type).	5-196
MUL_TIME(_E)	ⓢ1 , ⓢ2 , ⓓ	Outputs the product (ⓢ1 ×ⓢ2) of input values (time type).	5-199
DIV_TIME(_E)	ⓢ1 , ⓢ2 , ⓓ	Outputs the quotient (ⓢ1 ÷ⓢ2) of input values (time type).	5-202

2.2.9 Standard bistable function blocks

Function name	Argument	Processing details	Page
SR(_E)	ⓢ1 , ⓢ2 , ⓓ	Discriminates two input values and outputs 1 (TRUE) or 0 (FALSE). (Set-dominant)	5-204
RS(_E)	ⓢ1 , ⓢ2 , ⓓ	Discriminates two input values and outputs 1 (TRUE) or 0 (FALSE). (Reset-dominant)	5-207

2.2.10 Standard edge detection function blocks

Function name	Argument	Processing details	Page
R_TRIG(_E)	ⓢ , ⓓ	Detects the rising edge of a signal and outputs pulse signals.	5-210
F_TRIG(_E)	ⓢ , ⓓ	Detects the falling edge of a signal and outputs pulse signals.	5-213

2.2.11 Standard counter function blocks

Function name	Argument	Processing details	Page
CTU(_E)	ⓢ1 , ⓢ2 , n, ⓓ1 , ⓓ2	Counts the number of times that the signal turns ON.	5-215
CTD(_E)	ⓢ1 , ⓢ2 , n, ⓓ1 , ⓓ2	Counts down the number of times that the signal turns ON.	5-218
CTUD(_E)	ⓢ1 , ⓢ2 , ⓢ3 , ⓢ4 n, ⓓ1 , ⓓ2 , ⓓ3	Counts/counts down the number of times that the signal turns ON.	5-221
COUNTER_FB_M	ⓢ1 , ⓢ2 , ⓢ3 , ⓓ1 , ⓓ2	Counts the number of times that the signal turns ON from ⓢ3 to ⓢ2 .	5-225

2.2.12 Standard timer function blocks

Function name	Argument	Processing details	Page
TP(_E) TP_HIGH(_E)	ⓢ , n, ⓓ1 , ⓓ2	Holds the signal ON for the specified time.	5-227
TON(_E) TON_HIGH(_E)	ⓢ , n, ⓓ1 , ⓓ2	Turns ON the signal after the specified time.	5-230
TOF(_E) TOF_HIGH(_E)	ⓢ , n, ⓓ1 , ⓓ2	Turns OFF the signal after the specified time.	5-233
TIMER_10_FB_M TIMER_100_FB_M TIMER_HIGH_FB_M TIMER_LOW_FB_M TIMER_CONT_FB_M TIMER_CONTHFB_M	ⓢ1 , ⓢ2 , ⓢ3 , ⓓ1 , ⓓ2	Turns ON the signal after the specified time counted from input value ⓢ3 to ⓢ2 .	5-236

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

2.3.1 Arithmetic operations

Operator name		Argument	Processing details	Page
Structured ladder/FBD	ST			
ADD	+	ⓢ1 ,ⓢ2 ,...ⓢ28 ,ⓓ (Number of pins variable)	Outputs the sum (ⓢ1 +ⓢ2 +···+ⓢ28) of input values.	6-2
MUL	*	ⓢ1 ,ⓢ2 ,...ⓢ28 ,ⓓ (Number of pins variable)	Outputs the product (ⓢ1 ×ⓢ2 ×···×ⓢ28) of input values.	6-4
SUB	-	ⓢ1 , ⓢ2 , ⓓ	Outputs the difference (ⓢ1 −ⓢ2) between input values.	6-6
DIV	/	ⓢ1 , ⓢ2 , ⓓ	Outputs the quotient (ⓢ1 ÷ⓢ2) of input values.	6-8
(Not supported)	MOD	ⓢ1 , ⓢ2 , ⓓ	Outputs the remainder after division of input values (ⓢ1 ÷ⓢ2).	6-10
(Not supported)	**	ⓢ1 , ⓢ2 , ⓓ	Outputs the exponentiation of an input value.	6-11

2.3.2 Logical operations

Operator name		Argument	Processing details	Page
Structured ladder/FBD	ST			
AND	& AND	ⓢ1 ,ⓢ2 ,...ⓢ28 ,ⓓ (Number of pins variable)	Outputs the Boolean AND of input values.	6-13
OR	OR	ⓢ1 ,ⓢ2 ,...ⓢ28 ,ⓓ (Number of pins variable)	Outputs the Boolean OR of input values.	
XOR	XOR	ⓢ1 ,ⓢ2 ,...ⓢ28 ,ⓓ (Number of pins variable)	Outputs the Boolean exclusive OR of input values.	
(Not supported)	NOT	ⓢ , ⓓ	Outputs the Boolean NOT of input values.	

2.3.3 Comparison operations

Operator name		Argument	Processing details	Page
Structured ladder/FBD	ST			
GT	>	ⓢ1 ,ⓢ2 ,...ⓢ28 ,ⓓ (Number of pins variable)	Outputs the comparison value of an input value.	6-16
GE	>=	ⓢ1 ,ⓢ2 ,...ⓢ28 ,ⓓ (Number of pins variable)		
EQ	=	ⓢ1 ,ⓢ2 ,...ⓢ28 ,ⓓ (Number of pins variable)		
LE	<=	ⓢ1 ,ⓢ2 ,...ⓢ28 ,ⓓ (Number of pins variable)		
LT	<	ⓢ1 ,ⓢ2 ,...ⓢ28 ,ⓓ (Number of pins variable)		
NE	<>	ⓢ1 , ⓢ2 , ⓓ		

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.

(1) Source ⓢ

(a) A source is data used in an operation.

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

- 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.

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

(2) Destination ⓓ

(a) Data after the operation are stored to a destination.

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

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

(3) Executing condition (EN)

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

(4) Execution result (ENO)

(a) An output variable ENO outputs an execution result.

POINT

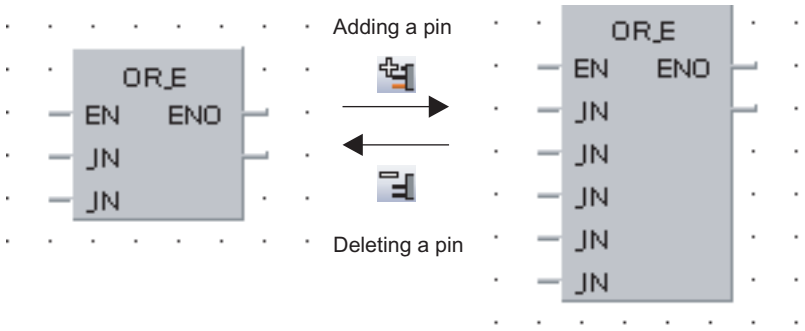
For details of the configuration of functions for labels and structures, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

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.

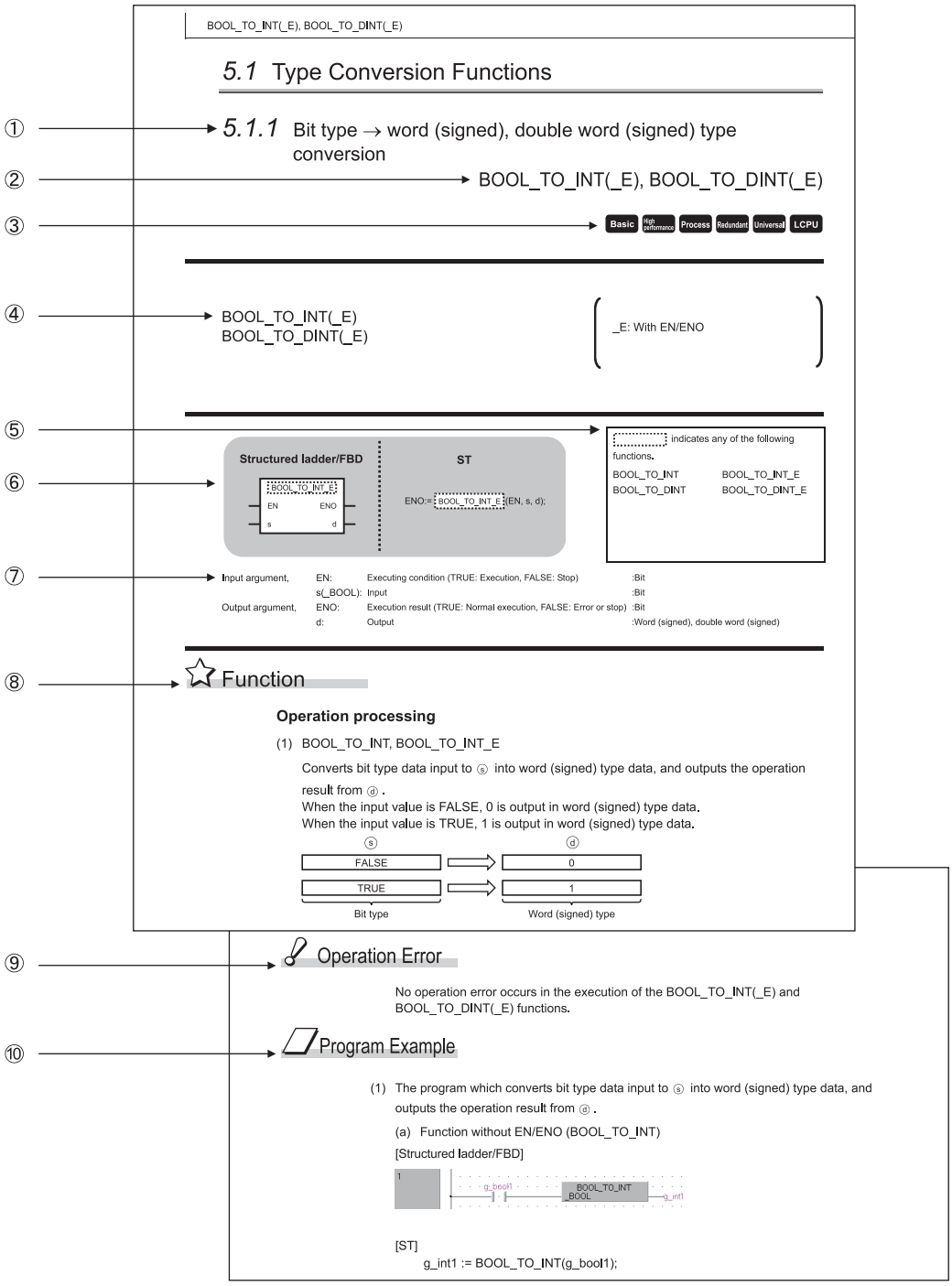
For the number of input pins change operation ☞ GX Works2 Version 1 Operating Manual (Structured Project)



MEMO

4 HOW TO READ FUNCTIONS

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



① Indicates a section number and an outline of a function.

② Indicates a function to be explained.

③ 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	
Basic	High performance	Process	Redundant	Universal	LCPU	The basic icon indicates that the CPU module can use the corresponding function.
Ver. Basic	Ver. High performance	Ver. Process	Ver. Redundant	Ver. Universal	Ver. LCPU	The icon with a Ver. symbol indicates that the CPU module can use the corresponding function under certain restrictions (function version and software version).
Basic	High performance	Process	Redundant	Universal	LCPU	The icon with × indicates that the CPU module cannot use the corresponding function.

④ Indicates the function names.

⑤ Indicates the function names that can be described.

⑥ Indicates the description format of the function in the structured ladder/FBD/ST language.

⑦ Indicates the names of input and output arguments, and the data type of each argument. For details of the data type, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

⑧ Indicates the processing performed by the function.

⑨ Indicates whether to exist the related error. When an error exists, conditions that cause an error are described.

⑩ Indicates program examples in the structured ladder/FBD/ST language .

MEMO

5 APPLICATION FUNCTIONS

5.1 Type Conversion Functions

5.1.1 Bit type → word (signed), double word (signed) type conversion

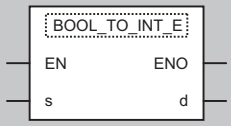
BOOL_TO_INT(_E), BOOL_TO_DINT(_E)

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

BOOL_TO_INT(_E)
BOOL_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

BOOL_TO_INT	BOOL_TO_INT_E
BOOL_TO_DINT	BOOL_TO_DINT_E

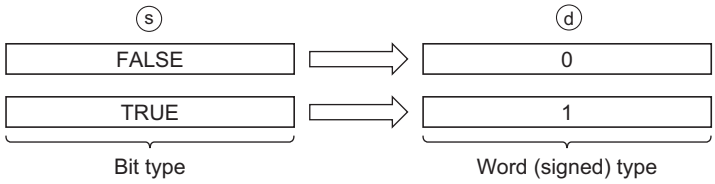
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_BOOL):	Input	:Bit
Output argument,	ENO:	Execution result (TRUE: Normal execution, FALSE: Error or stop)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

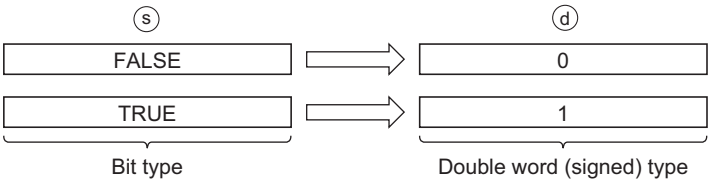
(1) BOOL_TO_INT, BOOL_TO_INT_E

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



(2) BOOL_TO_DINT, BOOL_TO_DINT_E

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

(1) Function without EN/ENO

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

No operation error occurs in the execution of the BOOL_TO_INT(_E) and BOOL_TO_DINT(_E) functions.

Program Example

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

(a) Function without EN/ENO (BOOL_TO_INT)

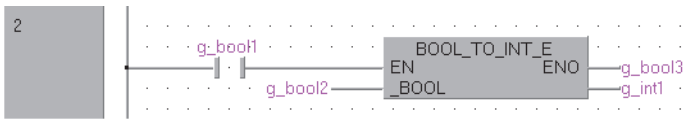
[Structured ladder/FBD]

[ST]

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

(b) Function with EN/ENO (BOOL_TO_INT_E)

[Structured ladder/FBD]



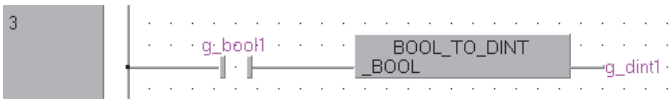
[ST]

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

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

(a) Function without EN/ENO (BOOL_TO_DINT)

[Structured ladder/FBD]



[ST]

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

5.1.2 Bit type → string type conversion

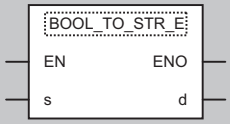
BOOL_TO_STR(_E)

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

BOOL_TO_STR(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

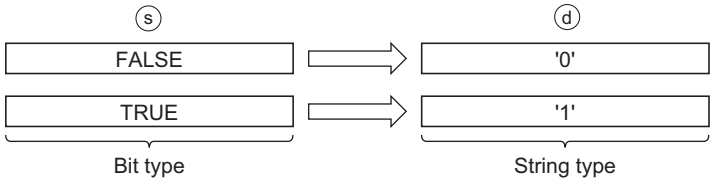
BOOL_TO_STR BOOL_TO_STR_E

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

Function

Operation processing

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

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

No operation error occurs in the execution of the BOOL_TO_STR(_E) function.

Program Example

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

(a) Function without EN/ENO (BOOL_TO_STR)

[Structured ladder/FBD]

[ST]

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

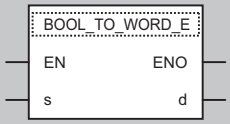
5.1.3 Bit type → word (unsigned)/16-bit string, double word (unsigned)/32-bit string type conversion BOOL_TO_WORD(_E), BOOL_TO_DWORD(_E)

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

BOOL_TO_WORD(_E)
BOOL_TO_DWORD(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

BOOL_TO_WORD BOOL_TO_WORD_E
BOOL_TO_DWORD BOOL_TO_DWORD_E

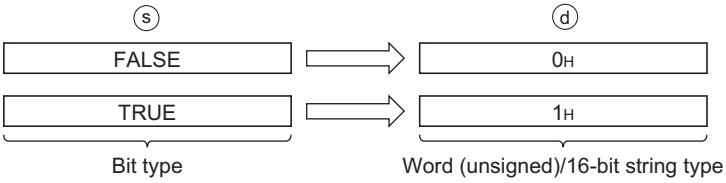
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_BOOL):	Input	:Bit
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string

Function

Operation processing

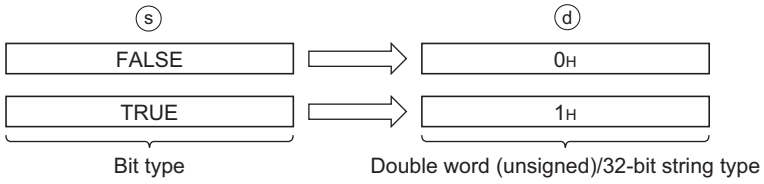
(1) BOOL_TO_WORD, BOOL_TO_WORD_E

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



(2) BOOL_TO_DWORD, BOOL_TO_DWORD_E

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

(1) Function without EN/ENO

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

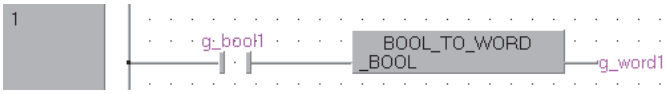
No operation error occurs in the execution of the BOOL_TO_WORD(_E) and BOOL_TO_DWORD(_E) functions.

Program Example

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

(a) Function without EN/ENO (BOOL_TO_WORD)

[Structured ladder/FBD]

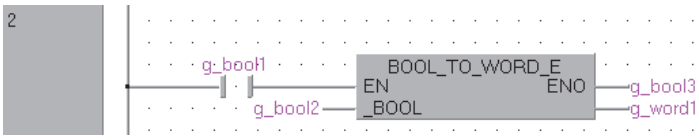


[ST]

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

(b) Function with EN/ENO (BOOL_TO_WORD_E)

[Structured ladder/FBD]



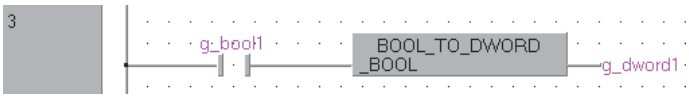
[ST]

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

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

(a) Function without EN/ENO (BOOL_TO_DWORD)

[Structured ladder/FBD]



[ST]

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

5.1.4 Bit type → time type conversion

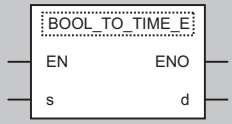
BOOL_TO_TIME(_E)

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

BOOL_TO_TIME(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

BOOL_TO_TIME BOOL_TO_TIME_E

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

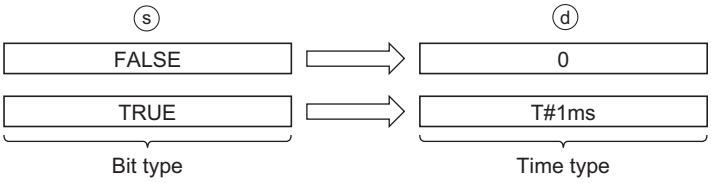
Function

Operation processing

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

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

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

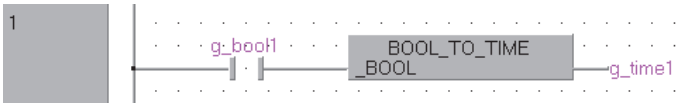
No operation error occurs in the execution of the BOOL_TO_TIME(_E) function.

Program Example

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

(a) Function without EN/ENO (BOOL_TO_TIME)

[Structured ladder/FBD]

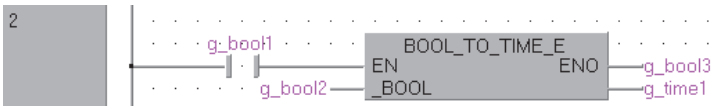


[ST]

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

(b) Function with EN/ENO (BOOL_TO_TIME_E)

[Structured ladder/FBD]



[ST]

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

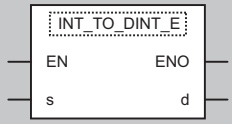
5.1.5 Word (signed) type → double word (signed) type conversion INT_TO_DINT(_E)

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

INT_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

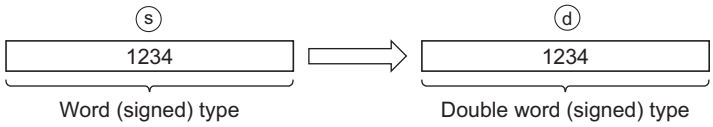
INT_TO_DINT INT_TO_DINT_E

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)

Function

Operation processing

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

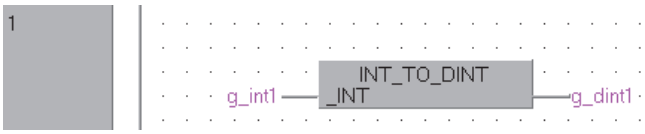
No operation error occurs in the execution of the INT_TO_DINT(_E) function.

Program Example

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

(a) Function without EN/ENO (INT_TO_DINT)

[Structured ladder/FBD]

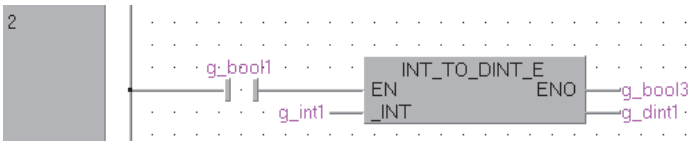


[ST]

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

(b) Function with EN/ENO (INT_TO_DINT_E)

[Structured ladder/FBD]



[ST]

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

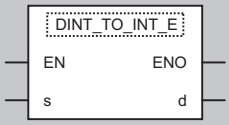
5.1.6 Double word (signed) type → word (signed) type conversion DINT_TO_INT(_E)

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

DINT_TO_INT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

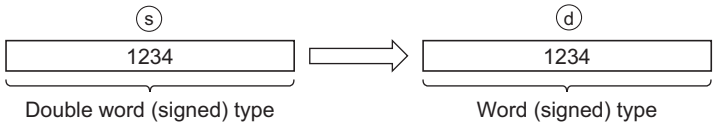
DINT_TO_INT DINT_TO_INT_E

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)

Function

Operation processing

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ 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 ⓢ are converted into word (signed) type data. High-order 16-bit data are discarded.

Operation Error

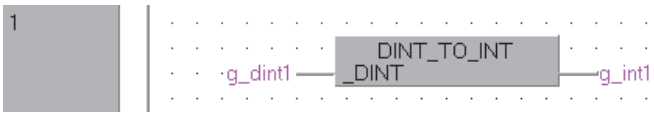
No operation error occurs in the execution of the DINT_TO_INT(_E) function.

Program Example

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

(a) Function without EN/ENO (DINT_TO_INT)

[Structured ladder/FBD]

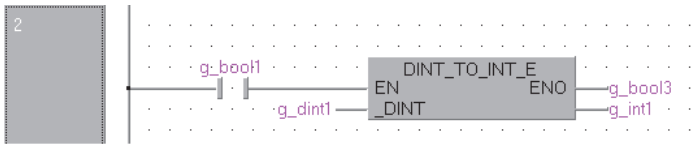


[ST]

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

(b) Function with EN/ENO (DINT_TO_INT_E)

[Structured ladder/FBD]



[ST]

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

5.1.7 Word (signed), double word (signed) type → bit type conversion

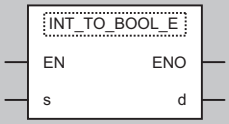
INT_TO_BOOL(_E), DINT_TO_BOOL(_E)

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

INT_TO_BOOL(_E)
DINT_TO_BOOL(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

INT_TO_BOOL	INT_TO_BOOL_E
DINT_TO_BOOL	DINT_TO_BOOL_E

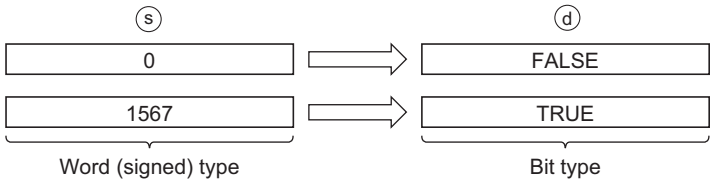
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	;Bit
	s(_INT, _DINT):	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Bit

Function

Operation processing

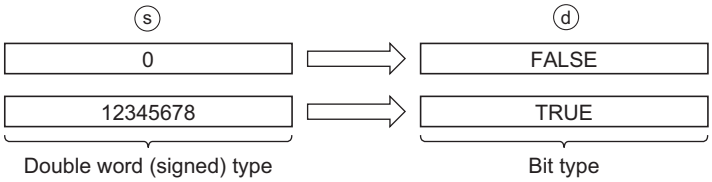
(1) INT_TO_BOOL, INT_TO_BOOL_E

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



(2) DINT_TO_BOOL, DINT_TO_BOOL_E

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

(1) Function without EN/ENO

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

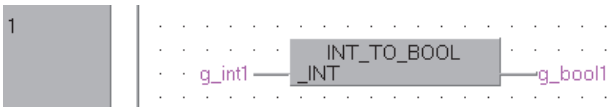
No operation error occurs in the execution of the INT_TO_BOOL(_E) and DINT_TO_BOOL(_E) functions.

Program Example

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

(a) Function without EN/ENO (INT_TO_BOOL)

[Structured ladder/FBD]

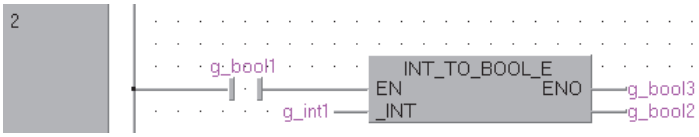


[ST]

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

(b) Function with EN/ENO (INT_TO_BOOL_E)

[Structured ladder/FBD]



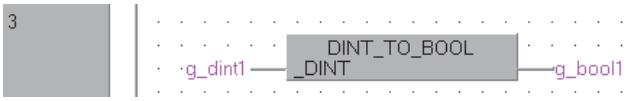
[ST]

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

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

(a) Function without EN/ENO (DINT_TO_BOOL)

[Structured ladder/FBD]



[ST]

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

5.1.8 Word (signed), double word (signed) type → single-precision real type conversion

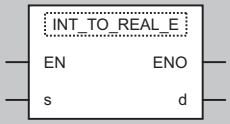
INT_TO_REAL(_E), DINT_TO_REAL(_E)

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

INT_TO_REAL(_E)
DINT_TO_REAL(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:= INT_TO_REAL_E (EN, s, d);

[ ] indicates any of the following functions.

INT_TO_REAL	INT_TO_REAL_E
DINT_TO_REAL	DINT_TO_REAL_E

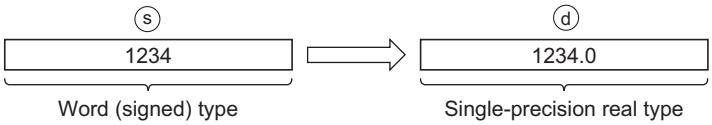
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_INT, _DINT):	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Single-precision real

Function

Operation processing

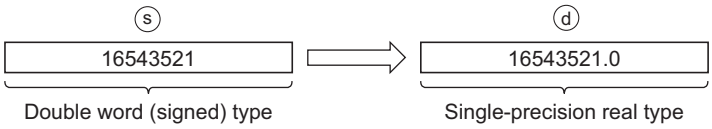
(1) INT_TO_REAL, INT_TO_REAL_E

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



(2) DINT_TO_REAL, DINT_TO_REAL_E

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



(b) 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

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

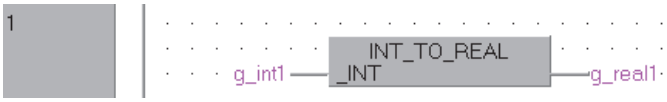
No operation error occurs in the execution of the INT_TO_REAL(_E) and DINT_TO_REAL(_E) functions.

Program Example

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

(a) Function without EN/ENO (INT_TO_REAL)

[Structured ladder/FBD]

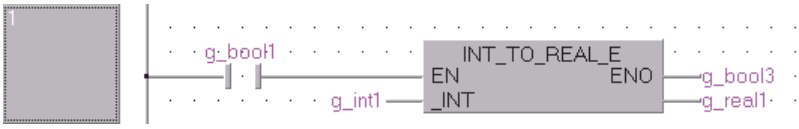


[ST]

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

(b) Function with EN/ENO (INT_TO_REAL_E)

[Structured ladder/FBD]



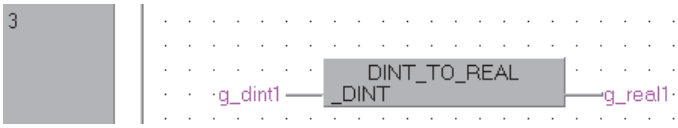
[ST]

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

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

(a) Function without EN/ENO (DINT_TO_REAL)

[Structured ladder/FBD]



[ST]

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

5.1.9 Word (signed), double word (signed) type → double-precision real type conversion

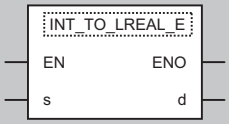
INT_TO_LREAL(_E), DINT_TO_LREAL(_E)

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

INT_TO_LREAL(_E)
DINT_TO_LREAL(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

INT_TO_LREAL	INT_TO_LREAL_E
DINT_TO_LREAL	DINT_TO_LREAL_E

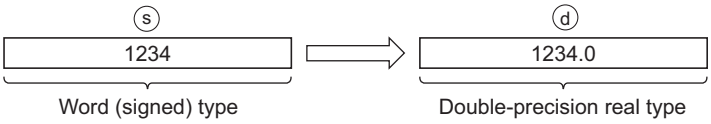
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_INT, _DINT):	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Double-precision real

Function

Operation processing

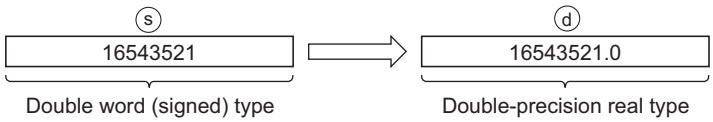
(1) INT_TO_LREAL, INT_TO_LREAL_E

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



(2) DINT_TO_LREAL, DINT_TO_LREAL_E

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

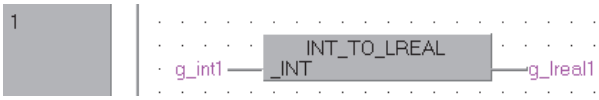
No operation error occurs in the execution of the INT_TO_LREAL(_E) and DINT_TO_LREAL(_E) functions.

Program Example

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

(a) Function without EN/ENO (INT_TO_LREAL)

[Structured ladder/FBD]

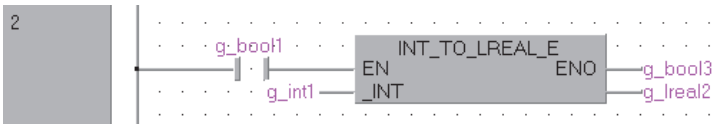


[ST]

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

(b) Function with EN/ENO (INT_TO_LREAL_E)

[Structured ladder/FBD]



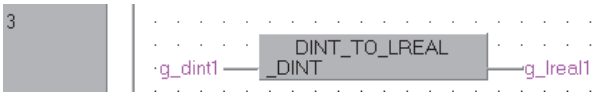
[ST]

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

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

(a) Function without EN/ENO (DINT_TO_LREAL)

[Structured ladder/FBD]



[ST]

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

5.1.10 Word (signed), double word (signed) type → string type conversion

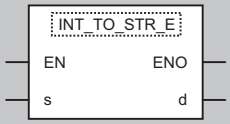
INT_TO_STR(_E), DINT_TO_STR(_E)

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

INT_TO_STR(_E)
DINT_TO_STR(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

INT_TO_STR	INT_TO_STR_E
DINT_TO_STR	DINT_TO_STR_E

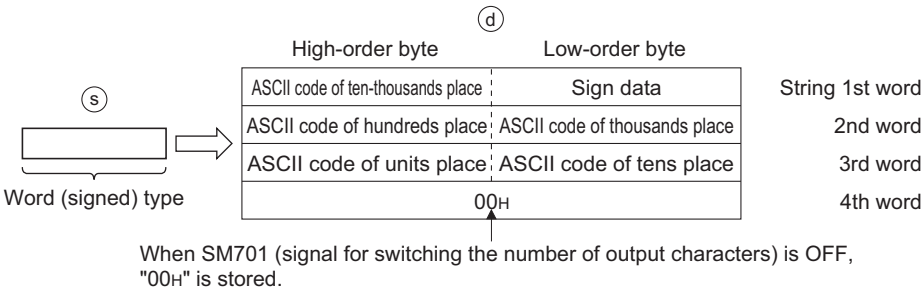
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_INT, _DINT):	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String(6) / (11)

Function

Operation processing

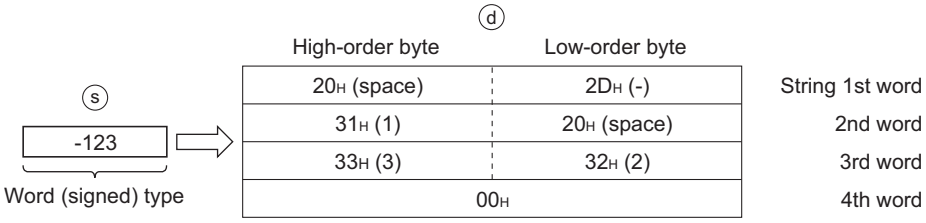
(1) INT_TO_STR, INT_TO_STR_E

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



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

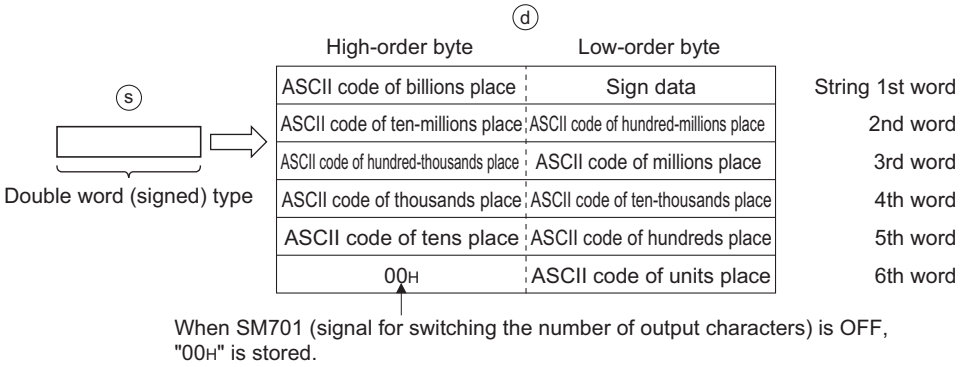
(c) If the number of significant figures is less, '20H (space)' is stored to high-order digits.
(Example) Inputting −123



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

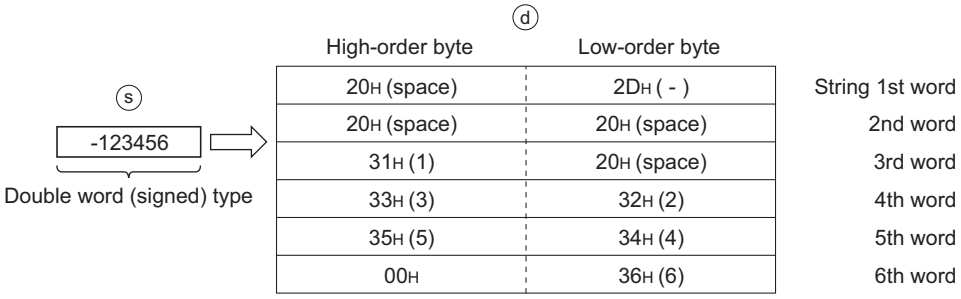
(2) DINT_TO_STR, DINT_TO_STR_E

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



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

(c) If the number of significant figures is less, '20H (space)' is stored to high-order digits.
(Example) Inputting −123456



(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

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

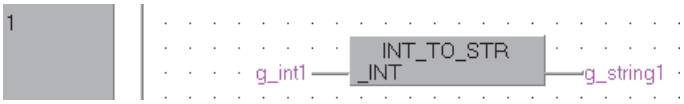
No operation error occurs in the execution of the INT_TO_STR(_E) and DINT_TO_STR(_E) functions.

Program Example

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

(a) Function without EN/ENO (INT_TO_STR)

[Structured ladder/FBD]

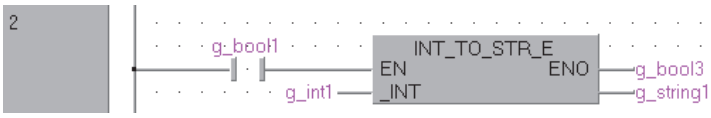


[ST]

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

(b) Function with EN/ENO (INT_TO_STR_E)

[Structured ladder/FBD]



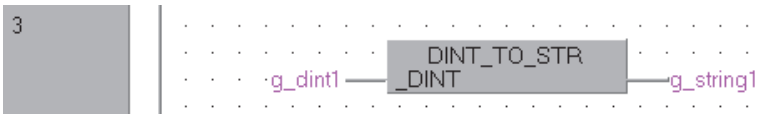
[ST]

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

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

(a) Function without EN/ENO (DINT_TO_STR)

[Structured ladder/FBD]



[ST]

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

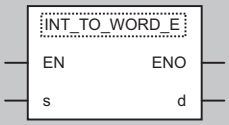
5.1.11 Word (signed), double word (signed) type → word (unsigned)/16-bit string type conversion INT_TO_WORD(_E), DINT_TO_WORD(_E)

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

INT_TO_WORD(_E)
DINT_TO_WORD(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

INT_TO_WORD	INT_TO_WORD_E
DINT_TO_WORD	DINT_TO_WORD_E

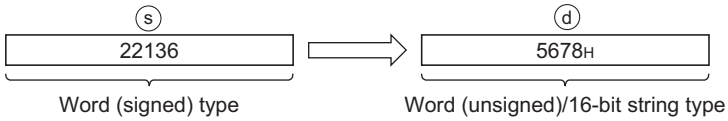
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_INT, _DINT):	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (unsigned)/16-bit string

Function

Operation processing

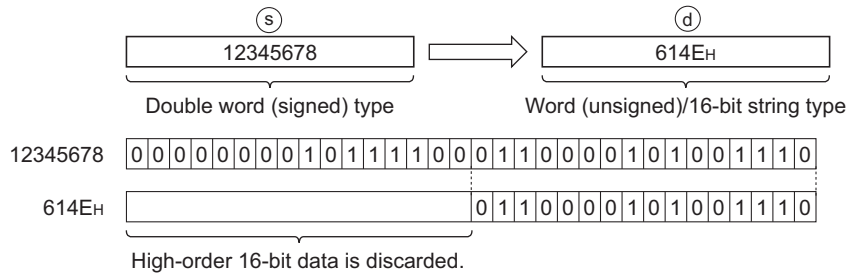
(1) INT_TO_WORD, INT_TO_WORD_E

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



(2) DINT_TO_WORD, DINT_TO_WORD_E

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ 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 ⓢ are converted into word (unsigned)/16-bit string type data. High-order word (unsigned)/16-bit string type data are discarded.

Operation Error

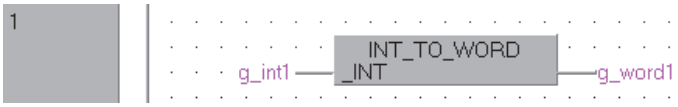
No operation error occurs in the execution of the INT_TO_WORD(_E) and DINT_TO_WORD(_E) functions.

Program Example

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

(a) Function without EN/ENO (INT_TO_WORD)

[Structured ladder/FBD]

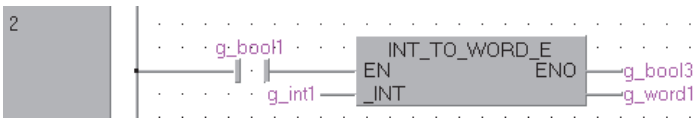


[ST]

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

(b) Function with EN/ENO (INT_TO_WORD_E)

[Structured ladder/FBD]



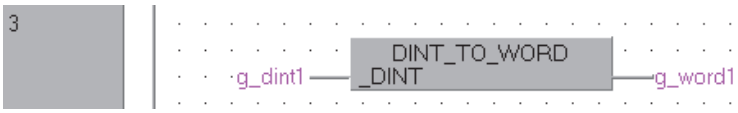
[ST]

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

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

(a) Function without EN/ENO (DINT_TO_WORD)

[Structured ladder/FBD]



[ST]

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

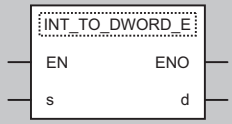
5.1.12 Word (signed), double word (signed) type → double word (unsigned)/32-bit string type conversion INT_TO_DWORD(_E), DINT_TO_DWORD(_E)

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

INT_TO_DWORD(_E)
DINT_TO_DWORD(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

INT_TO_DWORD	INT_TO_DWORD_E
DINT_TO_DWORD	DINT_TO_DWORD_E

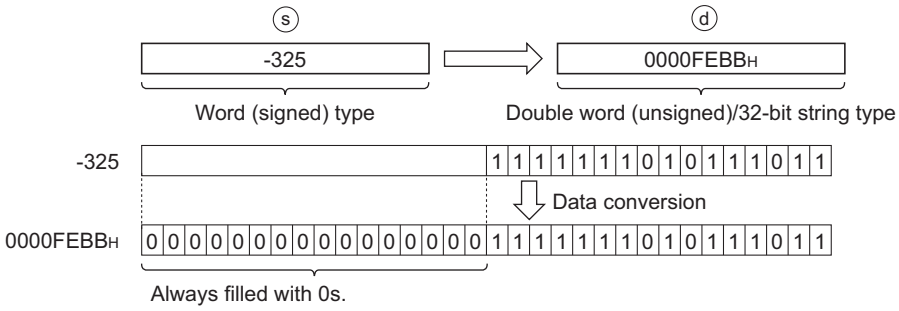
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_INT, _DINT):	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Double word (unsigned)/32-bit string

Function

Operation processing

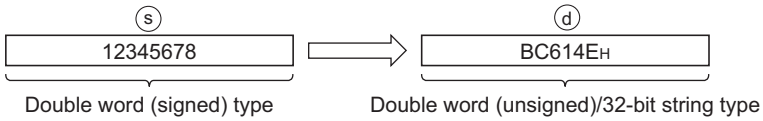
(1) INT_TO_DWORD, INT_TO_DWORD_E

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



(2) DINT_TO_DWORD, DINT_TO_DWORD_E

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

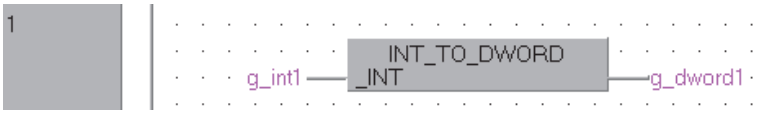
No operation error occurs in the execution of the INT_TO_DWORD(_E) and DINT_TO_DWORD(_E) functions.

Program Example

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

(a) Function without EN/ENO (INT_TO_DWORD)

[Structured ladder/FBD]

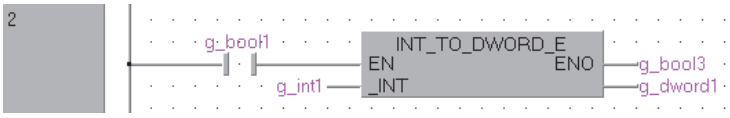


[ST]

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

(b) Function with EN/ENO (INT_TO_DWORD_E)

[Structured ladder/FBD]



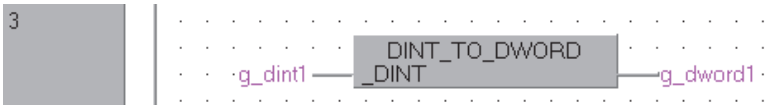
[ST]

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

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

(a) Function without EN/ENO (DINT_TO_DWORD)

[Structured ladder/FBD]



[ST]

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

5.1.13 Word (signed), double word (signed) type → BCD type conversion

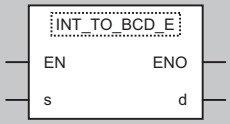
INT_TO_BCD(_E), DINT_TO_BCD(_E)

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

INT_TO_BCD(_E)
DINT_TO_BCD(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

INT_TO_BCD	INT_TO_BCD_E
DINT_TO_BCD	DINT_TO_BCD_E

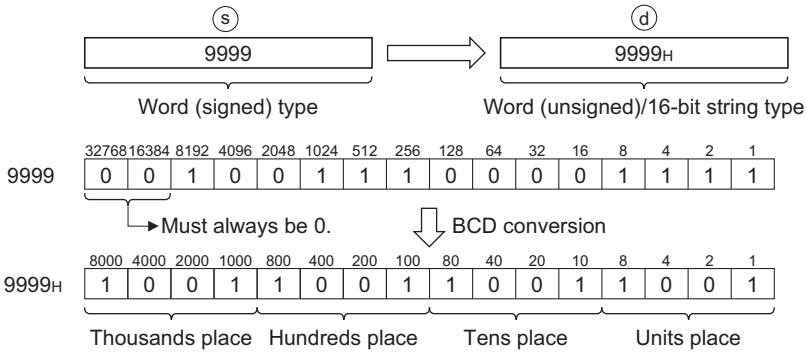
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_INT, _DINT):	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string

Function

Operation processing

(1) INT_TO_BCD, INT_TO_BCD_E

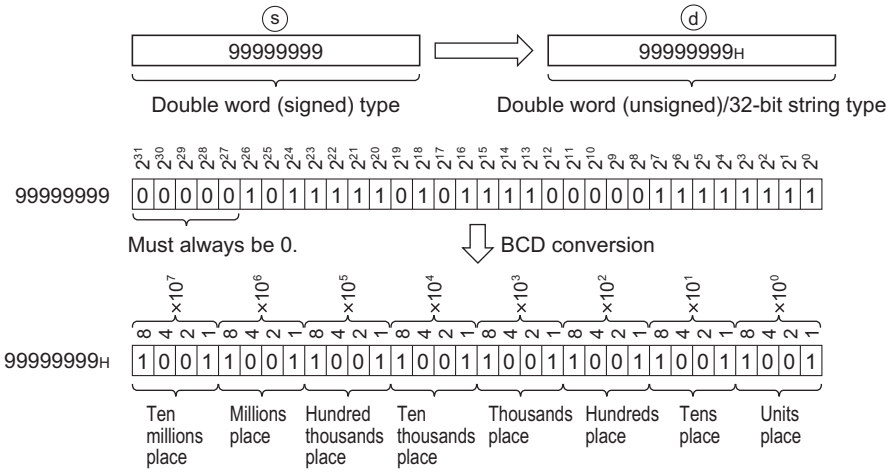
(a) Converts word (signed) type data input to ⓢ into BCD type data, and outputs the operation result from ⓓ .



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

(2) DINT_TO_BCD, DINT_TO_BCD_E

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

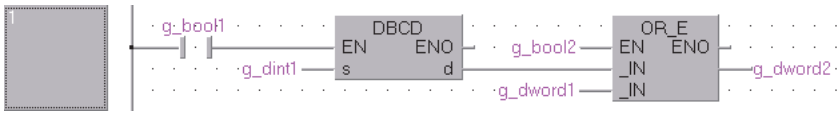


(b) The value to be input to ⓢ is double word (signed) type data within the range from 0 to 99999999.

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

POINT

The output from ⓓ 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

(1) Function without EN/ENO
The following table shows the operation results.

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

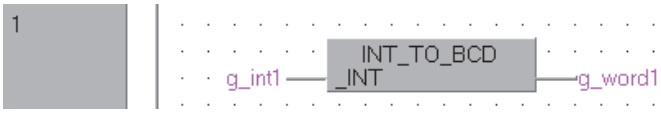
An operation error occurs when the value input exceeds 9999 or 99999999 respectively in the execution of the INT_TO_BCD(_E) or DINT_TO_BCD(_E) function. (Error code: 4100)

Program Example

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

(a) Function without EN/ENO (INT_TO_BCD)

[Structured ladder/FBD]

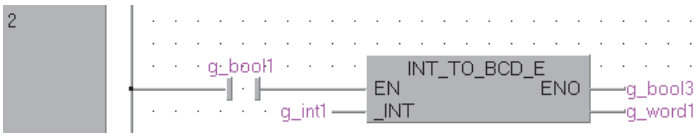


[ST]

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

(b) Function with EN/ENO (INT_TO_BCD_E)

[Structured ladder/FBD]



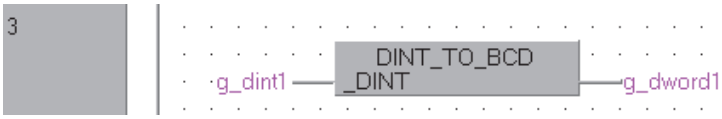
[ST]

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

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

(a) Function without EN/ENO (DINT_TO_BCD)

[Structured ladder/FBD]



[ST]

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

5.1.14 Word (signed), double word (signed) type → time type conversion

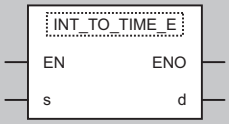
INT_TO_TIME(_E), DINT_TO_TIME(_E)

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

INT_TO_TIME(_E)
DINT_TO_TIME(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

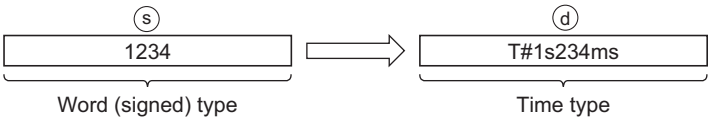
INT_TO_TIME	INT_TO_TIME_E
DINT_TO_TIME	DINT_TO_TIME_E

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

Function

Operation processing

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE*2	Undefined value

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

Operation Error

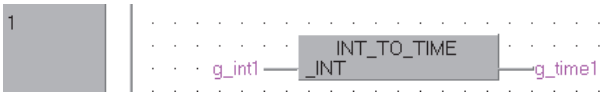
No operation error occurs in the execution of the INT_TO_TIME(_E) and DINT_TO_TIME(_E) functions.

Program Example

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

(a) Function without EN/ENO (INT_TO_TIME)

[Structured ladder/FBD]

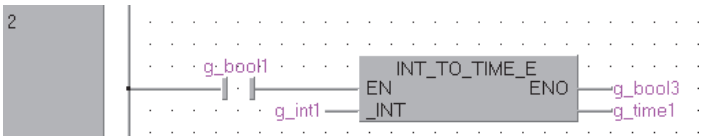


[ST]

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

(b) Function with EN/ENO (INT_TO_TIME_E)

[Structured ladder/FBD]



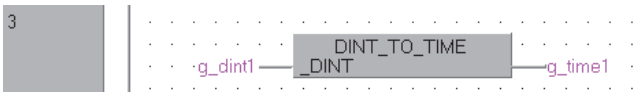
[ST]

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

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

(a) Function without EN/ENO (DINT_TO_TIME)

[Structured ladder/FBD]



[ST]

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

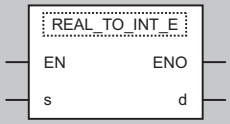
5.1.15 Single-precision real type → word (signed), double word (signed) type conversion REAL_TO_INT(_E), REAL_TO_DINT(_E)

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

REAL_TO_INT(_E)
REAL_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:= REAL_TO_INT_E(EN, s, d);

[ ] indicates any of the following functions.

REAL_TO_INT	REAL_TO_INT_E
REAL_TO_DINT	REAL_TO_DINT_E

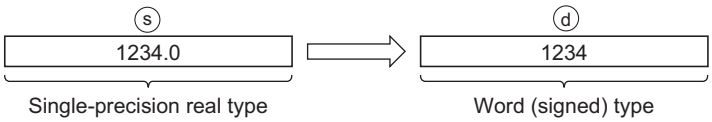
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_REAL):	Input	:Single-precision real
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

(1) REAL_TO_INT, REAL_TO_INT_E

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

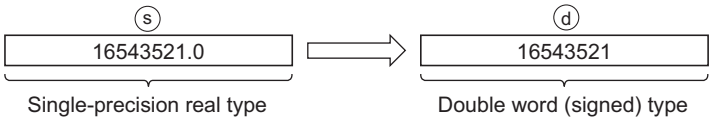


(b) The value to be input to ⓢ is single-precision real type data, within the range from –32768 to 32767.

(c) The converted data is the value rounded single-precision real type data to the first digit after the decimal point.

(2) REAL_TO_DINT, REAL_TO_DINT_E

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



(b) The value to be input to ⓢ 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 the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

(c) The converted data is the value rounded single-precision real type data to the first digit after the decimal point.

Operation result

(1) Function without EN/ENO
The following table shows the operation results.

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

An operation error occurs in the following cases.

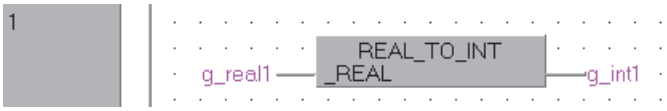
- REAL_TO_INT(_E): The input value is outside the range of –32768 to 32767. (Error code: 4100)
- REAL_TO_DINT(_E): The input value is outside the range of –2147483648 to 2147483647. (Error code: 4100)

Program Example

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

(a) Function without EN/ENO (REAL_TO_INT)

[Structured ladder/FBD]

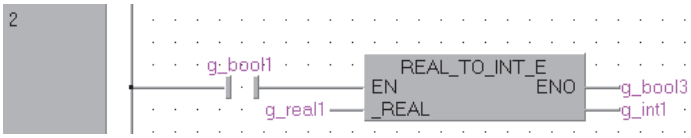


[ST]

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

(b) Function with EN/ENO (REAL_TO_INT_E)

[Structured ladder/FBD]



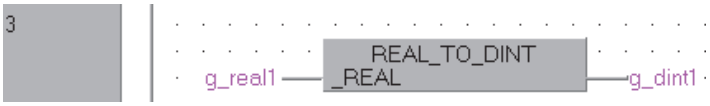
[ST]

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

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

(a) Function without EN/ENO (REAL_TO_DINT)

[Structured ladder/FBD]



[ST]

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

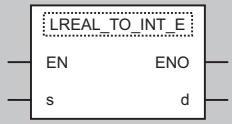
5.1.16 Double-precision real type → word (signed), double word (signed) type conversion LREAL_TO_INT(_E), LREAL_TO_DINT(_E)

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

LREAL_TO_INT(_E)
LREAL_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

LREAL_TO_INT	LREAL_TO_INT_E
LREAL_TO_DINT	LREAL_TO_DINT_E

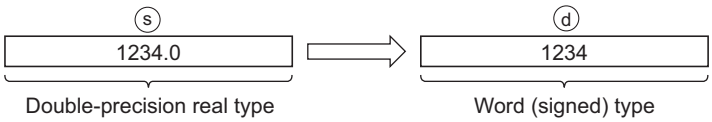
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_LREAL):	Input	:Double-precision real
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

(1) LREAL_TO_INT, LREAL_TO_INT_E

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

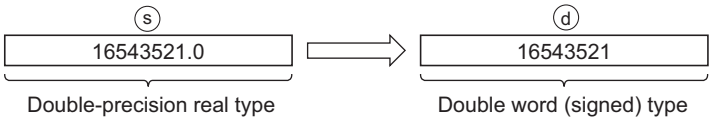


(b) The value to be input to ⓢ is double-precision real type data, within the range from −32768 to 32767.

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

(2) LREAL_TO_DINT, LREAL_TO_DINT_E

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



(b) The value to be input to ⓢ 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 the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

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

Operation result

(1) Function without EN/ENO
The following table shows the operation results.

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

An operation error occurs in the following cases.

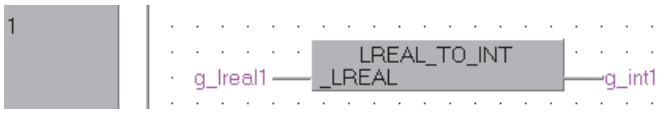
- The input value is -0 or outside the following range. (Error code: 4140)
  $0, 2^{-1022} \leqq |ⓢ| < 2^{1024}$
- LREAL_TO_INT(_E): The input value is outside the range of -32768 to 32767. (Error code: 4140)
- LREAL_TO_DINT(_E): The input value is outside the range of -2147483648 to 2147483647. (Error code: 4140)

Program Example

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

(a) Function without EN/ENO (LREAL_TO_INT)

[Structured ladder/FBD]

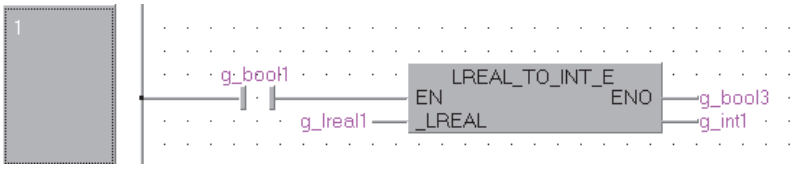


[ST]

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

(b) Function with EN/ENO (LREAL_TO_INT_E)

[Structured ladder/FBD]



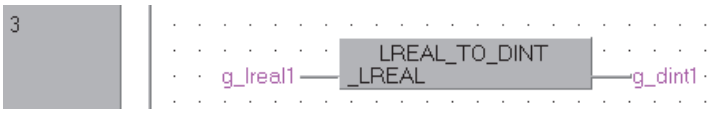
[ST]

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

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

(a) Function without EN/ENO (LREAL_TO_DINT)

[Structured ladder/FBD]



[ST]

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

5.1.17 Single-precision real type → double-precision real type conversion

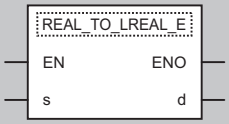
REAL_TO_LREAL(_E)

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

REAL_TO_LREAL(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

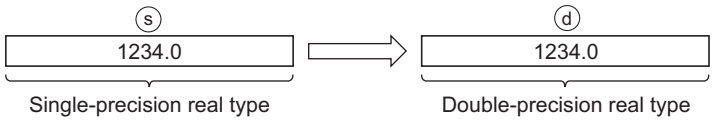
REAL_TO_LREAL REAL_TO_LREAL_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_REAL):	Input	:Single-precision real
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Double-precision real

Function

Operation processing

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



(2) Rounding error may occur when specifying the input value to ⓢ by programming tool. For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

(1) Function without EN/ENO
The following table shows the operation results.

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

An operation error occurs in the following cases.

- The input value is -0 or outside the following range. (Error code: 4140)

  $0, 2^{-126} \leqq |$ ⓢ $| < 2^{128}$

- The operation result is outside the following range (an overflow occurrence). (Error code: 4141)

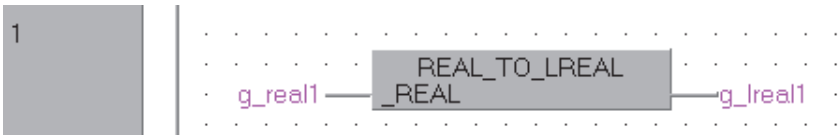
  $2^{1024} \leqq |$ operation result $|$

Program Example

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

(a) Function without EN/ENO (REAL_TO_LREAL)

[Structured ladder/FBD]

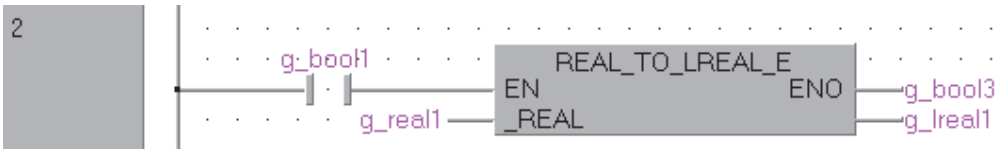


[ST]

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

(b) Function with EN/ENO (REAL_TO_LREAL_E)

[Structured ladder/FBD]



[ST]

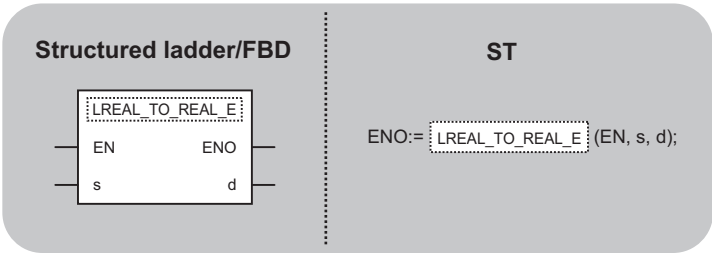
```
g_bool3 := REAL_TO_LREAL_E(g_bool1, g_real1, g_lreal1);
```

5.1.18 Double-precision real type → single-precision real type conversion

LREAL_TO_REAL(_E)

LREAL_TO_REAL(_E)

(_E: With EN/ENO)



indicates any of the following functions.

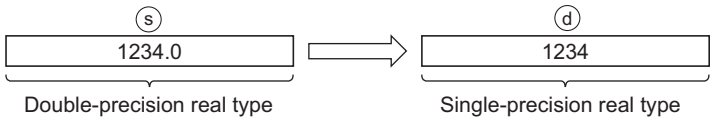
LREAL_TO_REAL LREAL_TO_REAL_E

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

Function

Operation processing

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



(2) Rounding error may occur when setting the input value to ⓢ by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

(1) Function without EN/ENO
The following table shows the operation results.

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

An operation error occurs in the following cases.

- The input value is -0 or outside the following range. (Error code: 4140)

  $0, 2^{-1022} \leqq |$ ⓢ $| < 2^{1024}$

- The operation result is outside the following range (an overflow occurrence). (Error code: 4141)

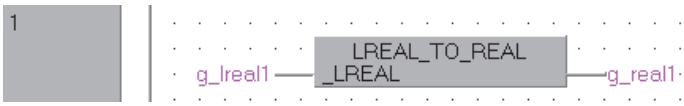
  $2^{128} \leqq |$ operation result $|$

Program Example

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

(a) Function without EN/ENO (LREAL_TO_REAL)

[Structured ladder/FBD]

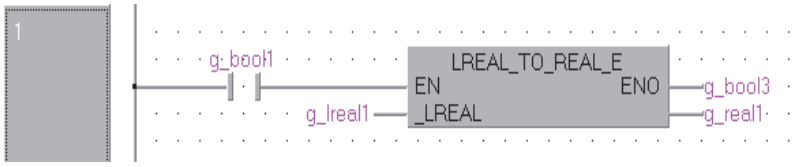


[ST]

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

(b) Function with EN/ENO (LREAL_TO_REAL_E)

[Structured ladder/FBD]



[ST]

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

5.1.19 Single-precision real type → string type conversion REAL_TO_STR(_E)

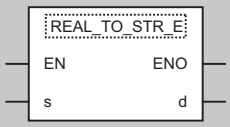
Ver. Basic | High performance | Process | Redundant | Universal | LCPU

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

REAL_TO_STR(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:= REAL_TO_STR_E(EN, s, d);

indicates any of the following functions.

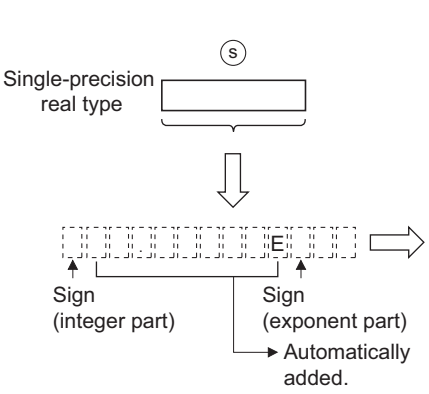
REAL_TO_STR REAL_TO_STR_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_REAL):	Input	:Single-precision real
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String(13)

Function

Operation processing

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



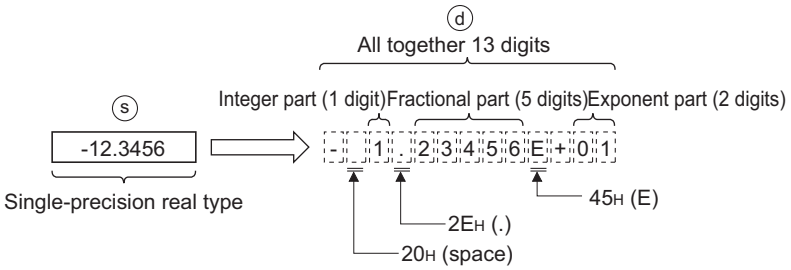
ⓓ

High-order byte	Low-order byte	
20H(space)	Sign data (integer part)	String 1st word
2EH(.)	ASCII code of integer part	2nd word
ASCII code of two decimal place	ASCII code of one decimal place	3rd word
ASCII code of four decimal place	ASCII code of three decimal place	4th word
45H(E)	ASCII code of five decimal place	5th word
ASCII code of exponent part's tens place	Sign data (exponent part)	6th word
00H(NUL)	ASCII code of exponent part's units place	7th word

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

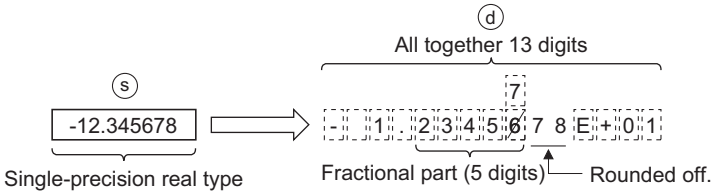
(2) The character string data after conversion is output from output variable ⓓ in the following manner.

(a) 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.

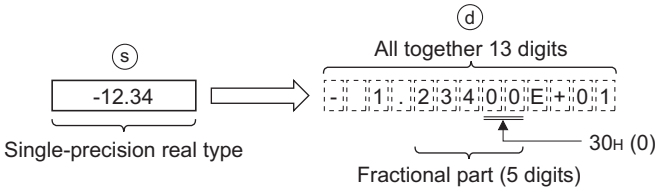


(b) '20H' (space) is stored in 'Sign data' (integer part) when the input value is positive; '2DH' (–) is stored when negative.

(c) Fractional part is rounded to 5 decimal places.

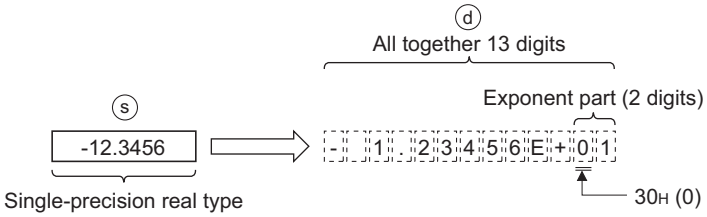


(d) If the number of significant figures is less, '30H' (0) is stored to fractional part.



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

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



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

(4) Rounding error may occur when specifying the input value to ⓢ by programming tool. For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

(1) Function without EN/ENO
The following table shows the operation results.

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

An operation error occurs in the following case.

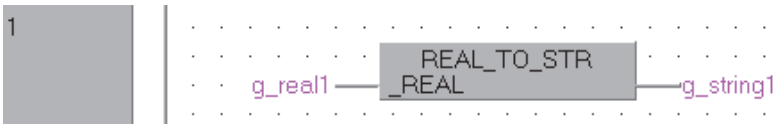
- The input value is outside the range of -3.40282^{+38} to -1.17549^{-38}, 0 or 1.17549^{-38} to 3.40282^{+38} (Error code: 4100)

Program Example

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

(a) Function without EN/ENO (REAL_TO_STR)

[Structured ladder/FBD]

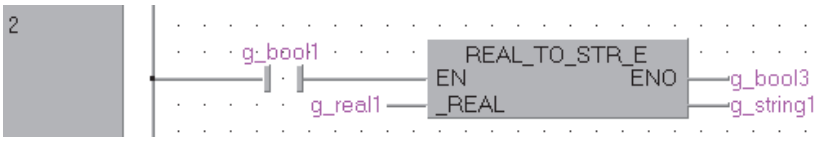


[ST]

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

(b) Function with EN/ENO (REAL_TO_STR_E)

[Structured ladder/FBD]



[ST]

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

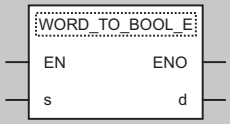
5.1.20 Word (unsigned)/16-bit string, double word (unsigned)/32-bit string type → bit type conversion WORD_TO_BOOL(_E), DWORD_TO_BOOL(_E)

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

WORD_TO_BOOL(_E)
DWORD_TO_BOOL(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

WORD_TO_BOOL	WORD_TO_BOOL_E
DWORD_TO_BOOL	DWORD_TO_BOOL_E

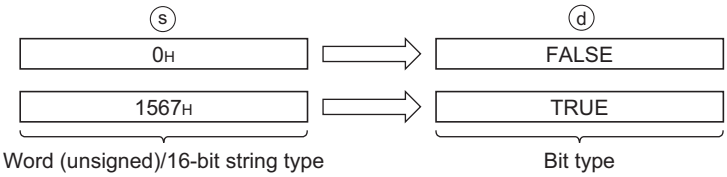
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_WORD, _DWORD):	Input	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Bit

Function

Operation processing

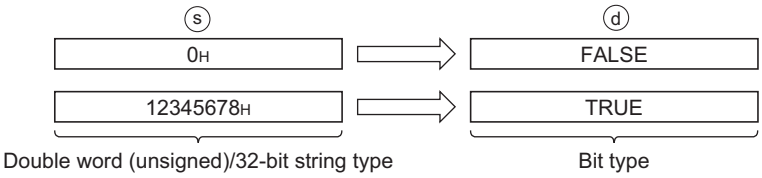
(1) WORD_TO_BOOL, WORD_TO_BOOL_E

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



(2) DWORD_TO_BOOL, DWORD_TO_BOOL_E

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

No operation error occurs in the execution of the WORD_TO_BOOL(_E) and DWORD_TO_BOOL(_E) functions.

Program Example

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

(a) Function without EN/ENO (WORD_TO_BOOL)

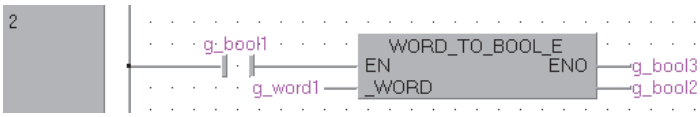
[Structured ladder/FBD]

[ST]

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

(b) Function with EN/ENO (WORD_TO_BOOL_E)

[Structured ladder/FBD]



[ST]

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

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

(a) Function without EN/ENO (DWORD_TO_BOOL)

[Structured ladder/FBD]

[ST]

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

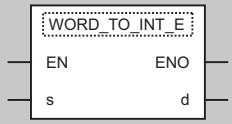
5.1.21 Word (unsigned)/16-bit string type → word (signed), double word (signed) type conversion WORD_TO_INT(_E), WORD_TO_DINT(_E)

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

WORD_TO_INT(_E)
WORD_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

WORD_TO_INT	WORD_TO_INT_E
WORD_TO_DINT	WORD_TO_DINT_E

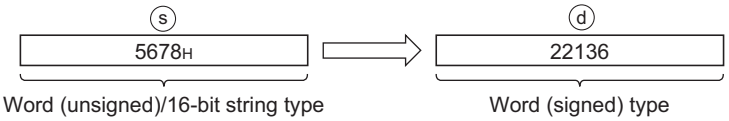
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_WORD):	Input	:Word (unsigned)/16-bit string
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

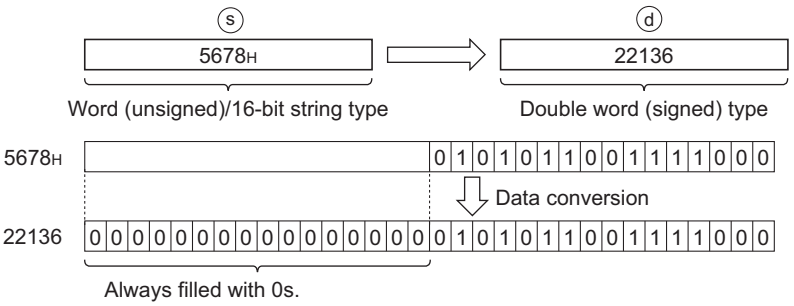
(1) WORD_TO_INT, WORD_TO_INT_E

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



(2) WORD_TO_DINT, WORD_TO_DINT_E

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

No operation error occurs in the execution of the WORD_TO_INT(_E) and WORD_TO_DINT(_E) functions.

Program Example

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

(a) Function without EN/ENO (WORD_TO_INT)

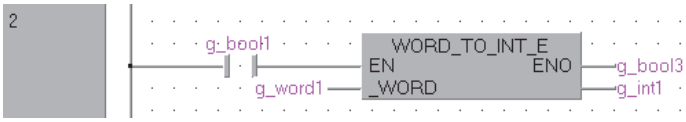
[Structured ladder/FBD]

[ST]

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

(b) Function with EN/ENO (WORD_TO_INT_E)

[Structured ladder/FBD]



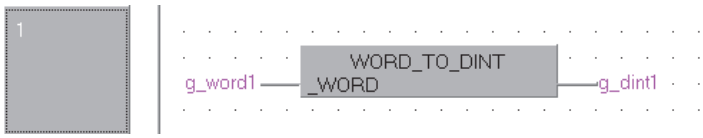
[ST]

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

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

(a) Function without EN/ENO (WORD_TO_DINT)

[Structured ladder/FBD]



[ST]

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

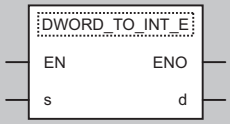
5.1.22 Double word (unsigned)/32-bit string type → word (signed), double word (signed) type conversion DWORD_TO_INT(_E), DWORD_TO_DINT(_E)

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

DWORD_TO_INT(_E)
DWORD_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

(dotted box) indicates any of the following functions.

DWORD_TO_INT	DWORD_TO_INT_E
DWORD_TO_DINT	DWORD_TO_DINT_E

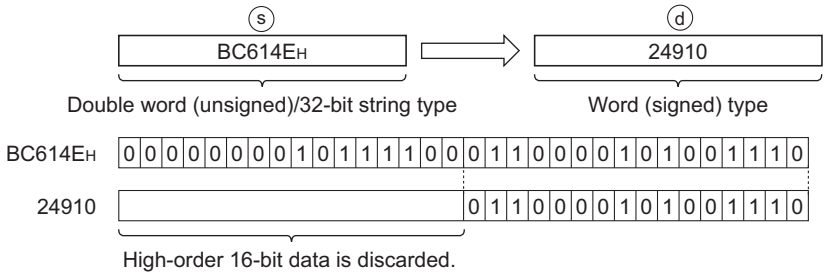
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_DWORD):	Input	:Double word (unsigned)/32-bit string
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

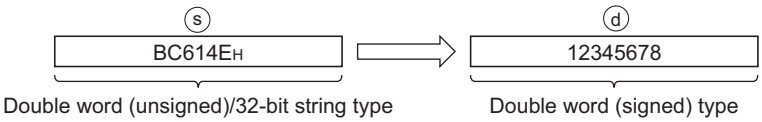
(1) DWORD_TO_INT, DWORD_TO_INT_E

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



(2) DWORD_TO_DINT, DWORD_TO_DINT_E

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ 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 ⓢ are converted into word (signed) type data. High-order 16-bit data are discarded.

Operation Error

No operation error occurs in the execution of the DWORD_TO_INT(_E) and DWORD_TO_DINT(_E) functions.

Program Example

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

(a) Function without EN/ENO (DWORD_TO_INT)

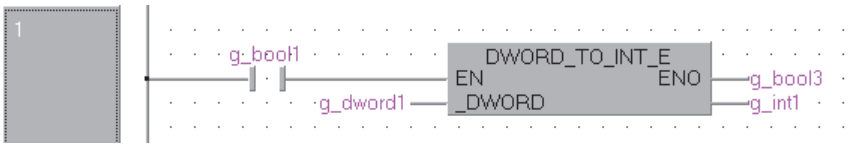
[Structured ladder/FBD]

[ST]

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

(b) Function with EN/ENO (DWORD_TO_INT_E)

[Structured ladder/FBD]



[ST]

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

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

(a) Function without EN/ENO (DWORD_TO_DINT)

[Structured ladder/FBD]

[ST]

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

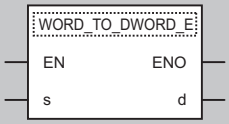
5.1.23 Word (unsigned)/16-bit string type → double word (unsigned)/32-bit string type conversion WORD_TO_DWORD(_E)

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

WORD_TO_DWORD(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

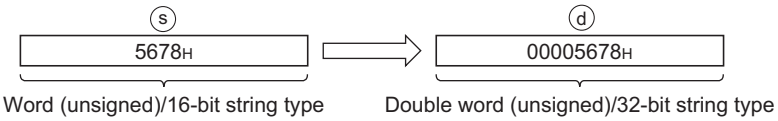
WORD_TO_DWORD WORD_TO_DWORD_E

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

Function

Operation processing

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

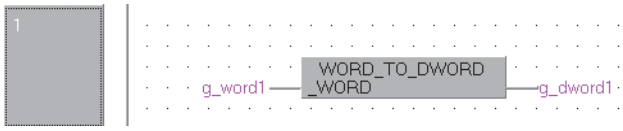
No operation error occurs in the execution of the WORD_TO_DWORD(_E) function.

Program Example

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

(a) Function without EN/ENO (WORD_TO_DWORD)

[Structured ladder/FBD]

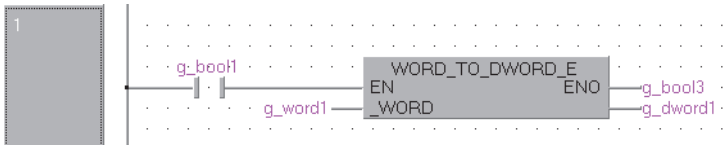


[ST]

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

(b) Function with EN/ENO (WORD_TO_DWORD_E)

[Structured ladder/FBD]



[ST]

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

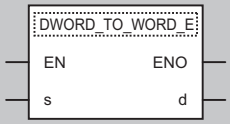
5.1.24 Double word (unsigned)/32-bit string type → word (unsigned)/16-bit string type conversion DWORD_TO_WORD(_E)

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

DWORD_TO_WORD(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

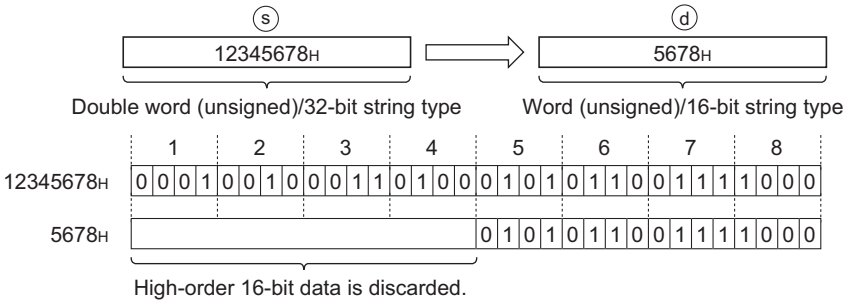
DWORD_TO_WORD DWORD_TO_WORD_E

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 (unsigned)/16-bit string

Function

Operation processing

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ 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 ⓢ are converted into word (unsigned)/16-bit string type data. High-order 16-bit data are discarded.

Operation Error

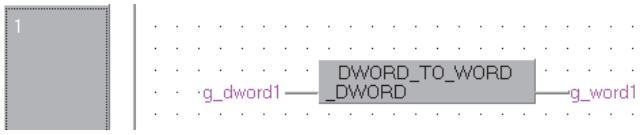
No operation error occurs in the execution of the DWORD_TO_WORD(_E) function.

Program Example

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

(a) Function without EN/ENO (DWORD_TO_WORD)

[Structured ladder/FBD]

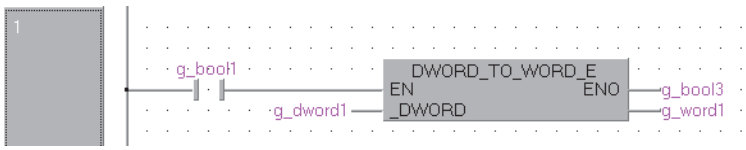


[ST]

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

(b) Function with EN/ENO (DWORD_TO_WORD_E)

[Structured ladder/FBD]



[ST]

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

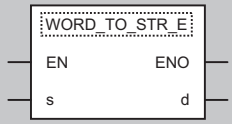
5.1.25 Word (unsigned)/16-bit string, double word (unsigned)/32-bit string type → string type conversion WORD_TO_STR(_E), DWORD_TO_STR(_E)

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

WORD_TO_STR(_E)
DWORD_TO_STR(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

WORD_TO_STR	WORD_TO_STR_E
DWORD_TO_STR	DWORD_TO_STR_E

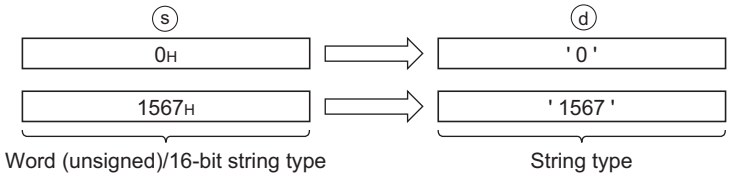
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_WORD, _DWORD):	Input	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String(4)/(8)

Function

Operation processing

(1) WORD_TO_STR, WORD_TO_STR_E

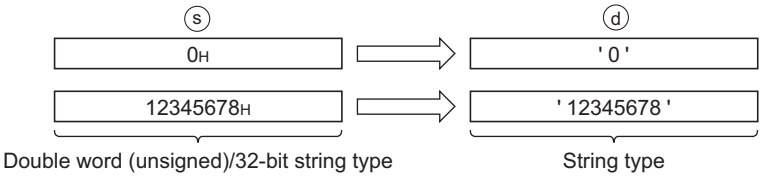
(a) Converts word (unsigned)/16-bit string type data input to ⓢ into string type data, and outputs the operation result from ⓓ .



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

(2) DWORD_TO_STR, DWORD_TO_STR_E

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



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

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

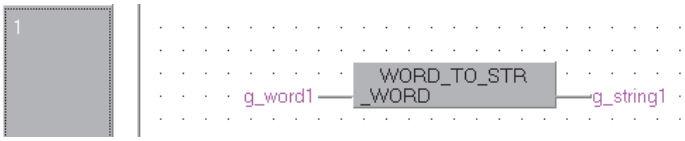
No operation error occurs in the execution of the WORD_TO_STR(_E) and DWORD_TO_STR(_E) functions.

Program Example

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

(a) Function without EN/ENO (WORD_TO_STR)

[Structured ladder/FBD]

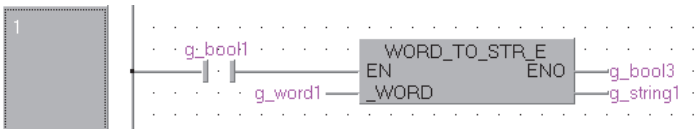


[ST]

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

(b) Function with EN/ENO (WORD_TO_STR_E)

[Structured ladder/FBD]



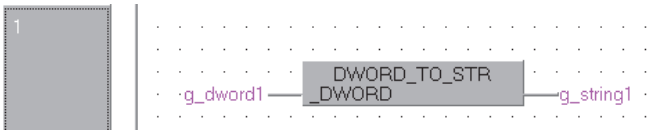
[ST]

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

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

(a) Function without EN/ENO (DWORD_TO_STR)

[Structured ladder/FBD]



[ST]

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

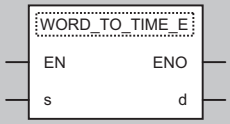
5.1.26 Word (unsigned)/16-bit string, double word (unsigned)/32-bit string type → time type conversion WORD_TO_TIME(_E), DWORD_TO_TIME(_E)

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

WORD_TO_TIME(_E)
DWORD_TO_TIME(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

WORD_TO_TIME	WORD_TO_ TIME _E
DWORD_TO_TIME	DWORD_TO_ TIME _E

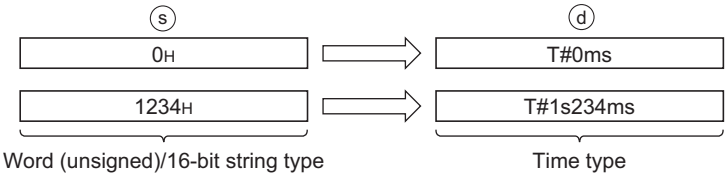
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_WORD, _DWORD):	Input	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Time

Function

Operation processing

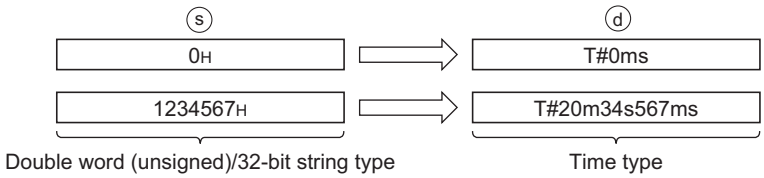
(1) WORD_TO_TIME, WORD_TO_TIME_E

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



(2) DWORD_TO_TIME, DWORD_TO_TIME_E

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

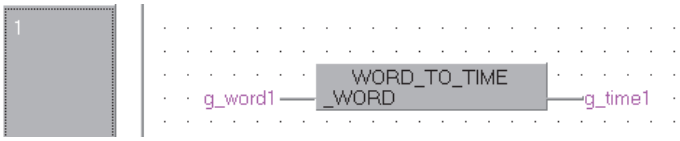
No operation error occurs in the execution of the WORD_TO_TIME(_E) and DWORD_TO_TIME(_E) functions.

Program Example

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

(a) Function without EN/ENO (WORD_TO_TIME)

[Structured ladder/FBD]

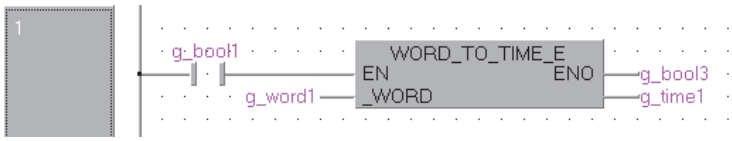


[ST]

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

(b) Function with EN/ENO (WORD_TO_TIME_E)

[Structured ladder/FBD]



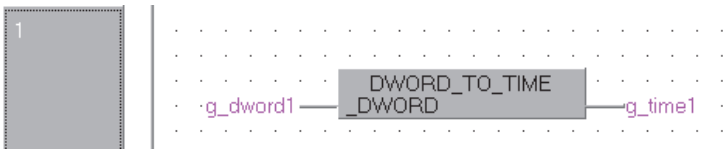
[ST]

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

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

(a) Function without EN/ENO (DWORD_TO_TIME)

[Structured ladder/FBD]



[ST]

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

5.1.27 String type → bit type conversion

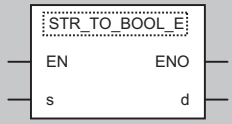
STR_TO_BOOL(_E)

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

STR_TO_BOOL(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

STR_TO_BOOL STR_TO_BOOL_E

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

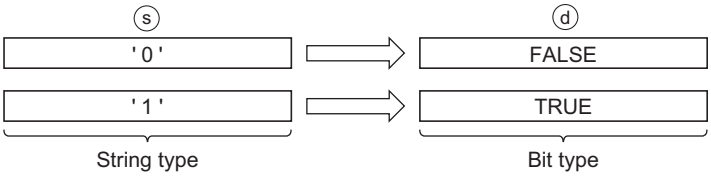
Function

Operation processing

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

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

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

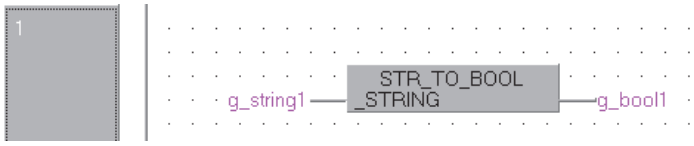
No operation error occurs in the execution of the STR_TO_BOOL(_E) function.

Program Example

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

(a) Function without EN/ENO (STR_TO_BOOL)

[Structured ladder/FBD]

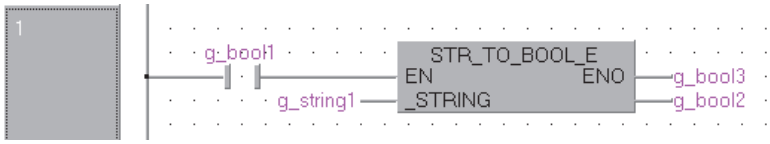


[ST]

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

(b) Function with EN/ENO (STR_TO_BOOL_E)

[Structured ladder/FBD]



[ST]

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

5.1.28 String type → word (signed), double word (signed) type conversion

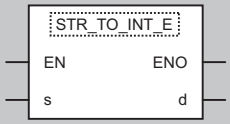
STR_TO_INT(_E), STR_TO_DINT(_E)

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

STR_TO_INT(_E)
STR_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

STR_TO_INT	STR_TO_INT_E
STR_TO_DINT	STR_TO_DINT_E

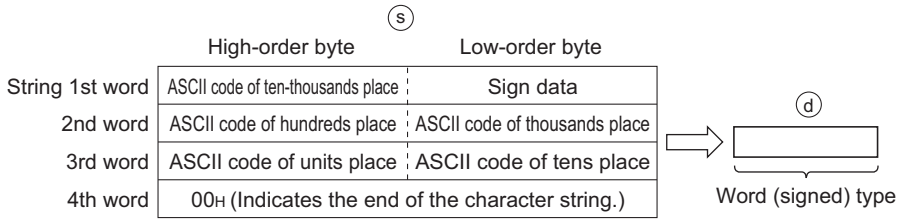
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_STRING):	Input	:String (6)/(11)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

(1) STR_TO_INT, STR_TO_INT_E

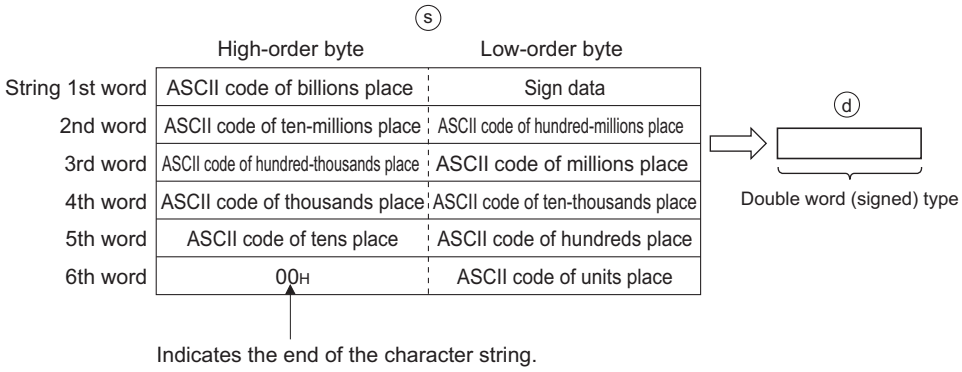
(a) Converts string type data input to ⓢ into word (signed) type data, and outputs the operation result from ⓓ .



(b) The value to be input to ⓢ is string type data within the following range.
ASCII code: '30H' to '39H', '20H', '2DH', and '00H'
String type data: '-32768' to '32767'

(2) STR_TO_DINT, STR_TO_DINT_E

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



(b) The value to be input to ⓢ 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

(1) Function without EN/ENO
The following table shows the operation results.

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

An operation error occurs in the following cases.

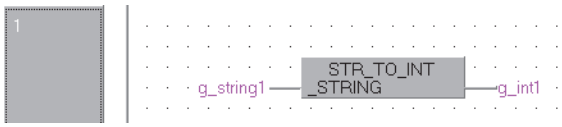
- The input value is other than '30H' to '39H', '20H', '2DH', and '00H' of ASCII code. (Error code: 4100)
- The input value is outside the following ranges of ASCII code. (Error code: 4100)
  STR_TO_INT(_E): –32768 to 32767
  STR_TO_DINT(_E): –2147483648 to 2147483647

Program Example

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

(a) Function without EN/ENO (STR_TO_INT)

[Structured ladder/FBD]

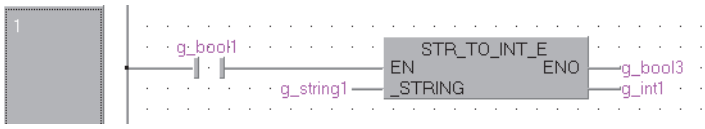


[ST]

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

(b) Function with EN/ENO (STR_TO_INT_E)

[Structured ladder/FBD]



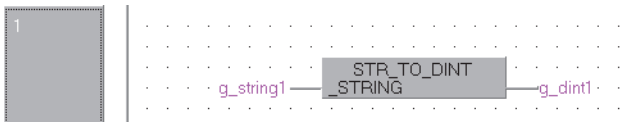
[ST]

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

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

(a) Function without EN/ENO (STR_TO_DINT)

[Structured ladder/FBD]



[ST]

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

5.1.29 String type → single-precision real type conversion STR_TO_REAL(_E)

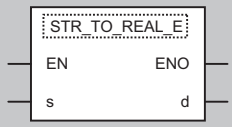
Ver. Basic | High performance | Process | Redundant | Universal | LCPU

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

STR_TO_REAL(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

STR_TO_REAL STR_TO_REAL_E

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

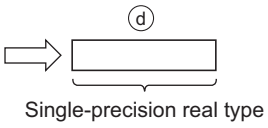
Function

Operation processing

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

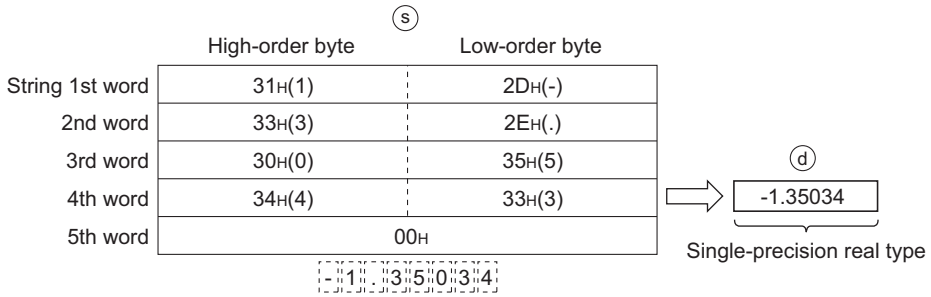
ⓢ

	High-order byte	Low-order byte
String 1st word	ASCII code of 1st character	Sign date
2nd word	ASCII code of 3rd character	ASCII code of 2nd character
3rd word	ASCII code of 5th character	ASCII code of 4th character
4th word	ASCII code of 7th character	ASCII code of 6th character
5th word	ASCII code of 9th character	ASCII code of 8th character
6th word	ASCII code of 11th character	ASCII code of 10th character
7th word	00H (Indicates the end of the character string.)	

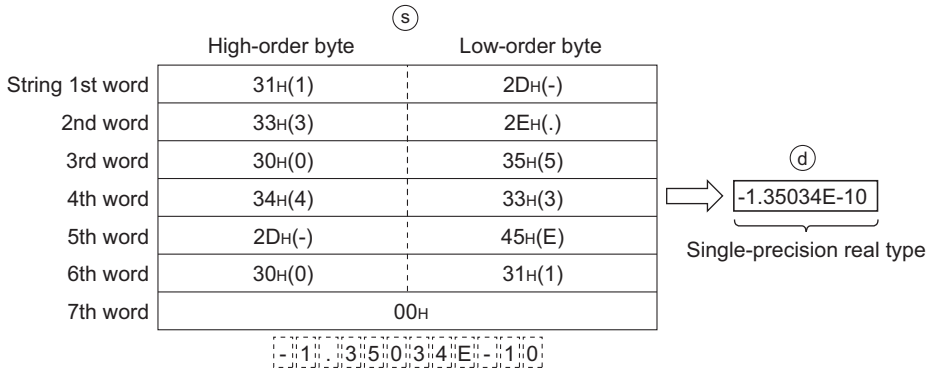


(2) Both string type data in decimal form and exponential form can be converted to single-precision real type data.

(a) Decimal form

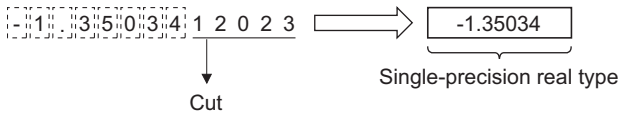


(b) Exponential form

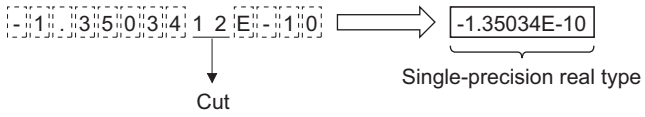


(3) As the number of significant figures of string type data is 6, the 7th and later digits excluding the sign, decimal point, and exponent part are cut and converted.

(a) Decimal form



(b) Exponential form



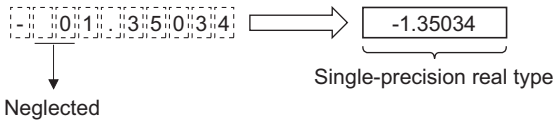
(4) When a sign is not specified or '2BH' (+) is specified for a sign in decimal form, string type data is converted as a positive value. When '2DH' (−) is specified for a sign, string type data is converted as a negative value.

(5) 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.

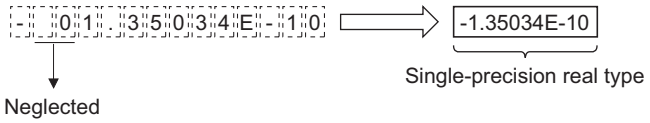
When '2DH' (−) is specified for a sign of the exponential part, string type data is converted as a negative value.

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

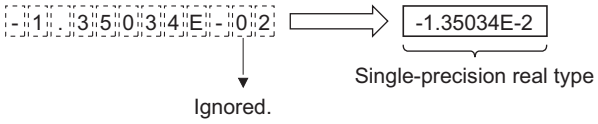
(a) Decimal form



(b) Exponential form



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



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

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

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

Operation result

(1) Function without EN/ENO
The following table shows the operation results.

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

An operation error occurs in the following cases.

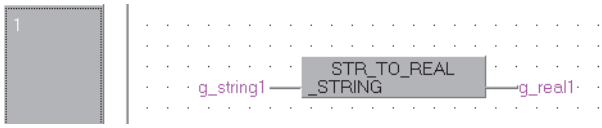
- Any characters other than '30H' to '39H' exist in the integer or fractional part. (Error code: 4100)
- Two or more 2EH exist. (Error code: 4100)
- Any characters other than '45H'(E), '2BH'(+) or '45H'(E), '2DH'(−) exist in the exponent part, or more than one exponent parts exist. (Error code: 4100)
- 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} (Error code: 4100)
- The number of characters is 0 or exceeding 24. (Error code: 4100)

Program Example

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

(a) Function without EN/ENO (STR_TO_REAL)

[Structured ladder/FBD]

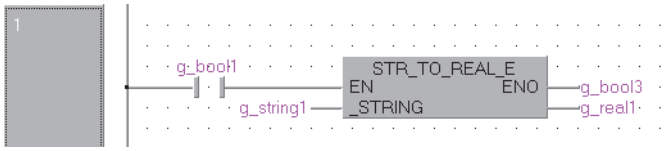


[ST]

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

(b) Function with EN/ENO (STR_TO_REAL_E)

[Structured ladder/FBD]



[ST]

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

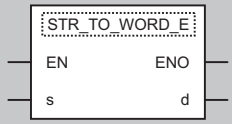
5.1.30 String type → word (unsigned)/16-bit string, double word (unsigned)/32-bit string type conversion STR_TO_WORD(_E), STR_TO_DWORD(_E)

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

STR_TO_WORD(_E)
STR_TO_DWORD(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

(dotted box) indicates any of the following functions.

STR_TO_WORD	STR_TO_WORD_E
STR_TO_DWORD	STR_TO_DWORD_E

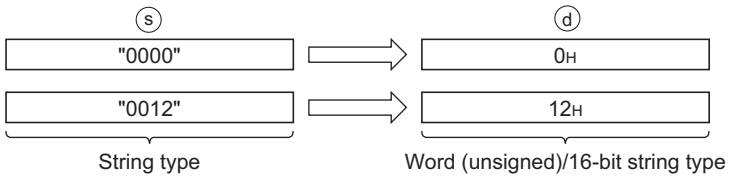
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_STRING):	Input	:String (4)/(8)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string

Function

Operation processing

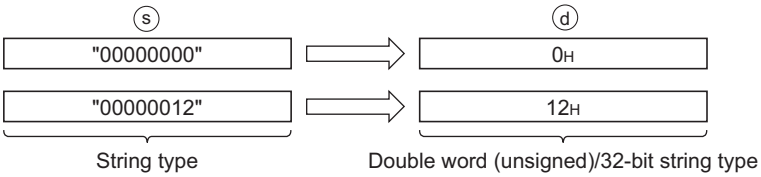
(1) STR_TO_WORD, STR_TO_WORD_E

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



(2) STR_TO_DWORD, STR_TO_DWORD_E

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ 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.

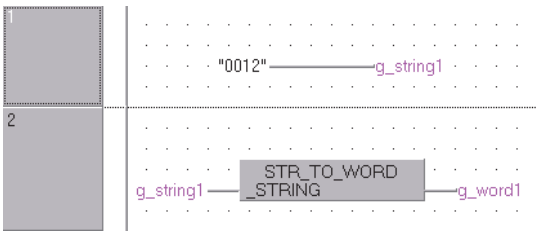
- The ASCII code for each number specified for ⓢ is outside the range of 30H to 39H, 41H to 46H. (Error code: 4100)
- The device specified ⓢ for exceeds the corresponding device range. (For Universal model QCPU and LCPU) (Error code: 4101)

Program Example

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

(a) Function without EN/ENO (STR_TO_WORD)

[Structured ladder/FBD]

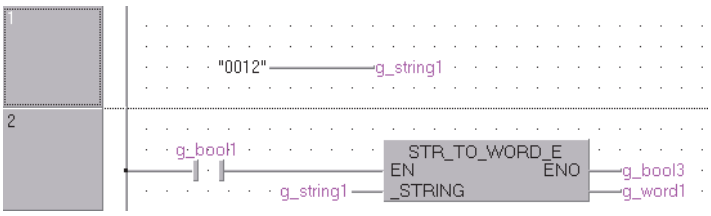


[ST]

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

(b) Function with EN/ENO (STR_TO_WORD_E)

[Structured ladder/FBD]



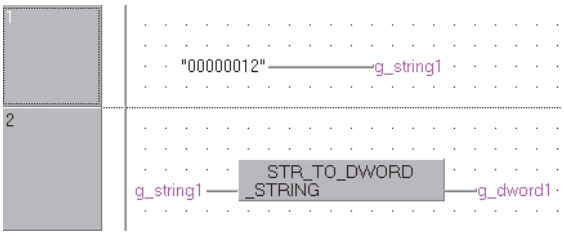
[ST]

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

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

(a) Function without EN/ENO (STR_TO_DWORD)

[Structured ladder/FBD]



[ST]

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

5.1.31 String type → time type conversion

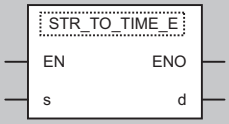
STR_TO_TIME(_E)

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

STR_TO_TIME(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

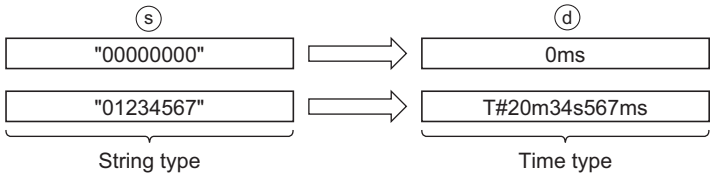
STR_TO_TIME　　STR_TO_ TIME _E

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

Function

Operation processing

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ 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.

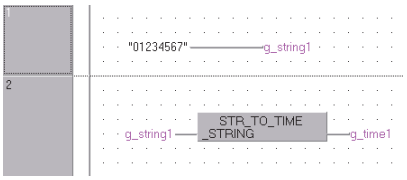
- The ASCII code for each number specified for ⓢ is outside the range of 30H to 39H, 20H, and 00H. (Error code: 4100)
- The ASCII data specified for ⓢ is outside of the range shown below.
  -2147483648 to 4147483647 (Error code: 4100)

Program Example

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

(a) Function without EN/ENO (STR_TO_TIME)

[Structured ladder/FBD]

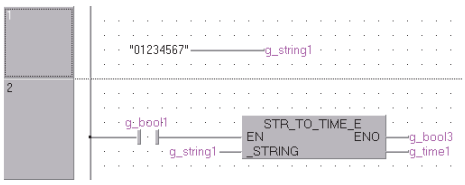


[ST]

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

(b) 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);
```

5.1.32 String type → BCD type conversion

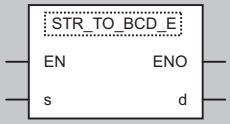
STR_TO_BCD(_E)

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

STR_TO_BCD(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

STR_TO_BCD STR_TO_ BCD _E

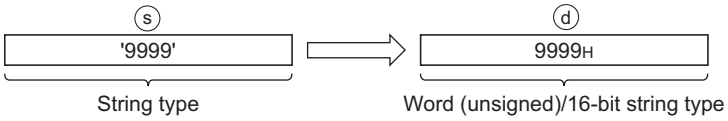
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_STRING):	Input	:String (8)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:ANY BIT

Function

Operation processing

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

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



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

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

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

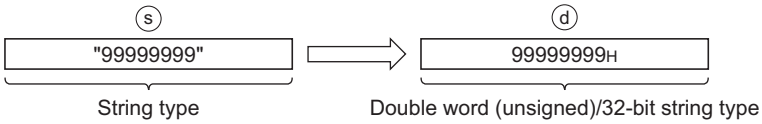
(e) When input character string has less than 4 letters, convert it with 4 letters supplementing with 0 to the end of the character string. Therefore, when converting character string ("0001" for "1") with less than 4 letters to BCD data, input the zero padding character strings.

(f) 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)

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

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



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

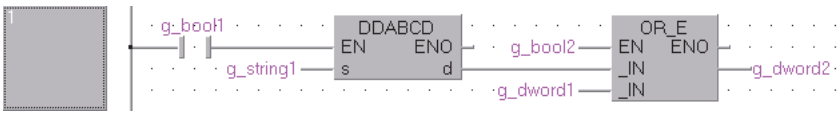
(c) '20H' (space) and '30H' (0) in the character string are counted as one character.

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

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

POINT

Output from ⓓ 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

(1) Function without EN/ENO
The following table shows the operation results.

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

An operation error occurs in the following cases.

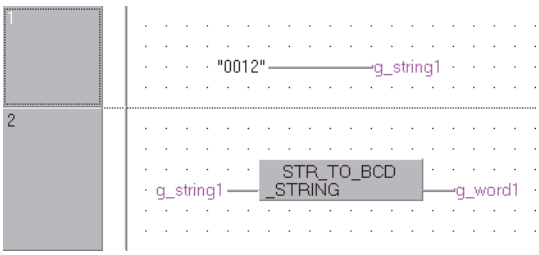
- The input character string is outside the range of ASCII code '30H' to '39H', "20H", and "00H". (Error code 4100)

Program Example

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

(a) Function without EN/ENO (STR_TO_BCD)

[Structured ladder/FBD]

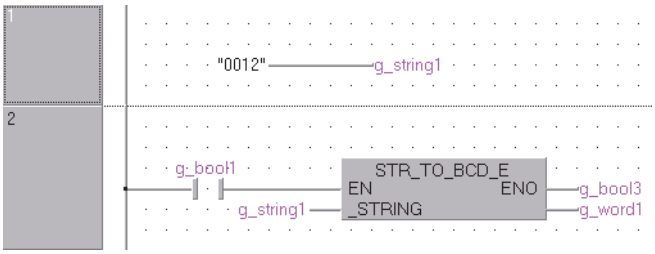


[ST]

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

(b) Function without EN/ENO (STR_TO_BCD_E)

[Structured ladder/FBD]



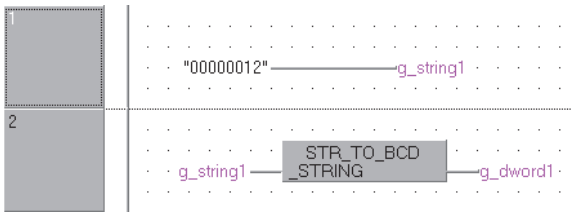
[ST]

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

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

(a) Function without EN/ENO (STR_TO_BCD)

[Structured ladder/FBD]

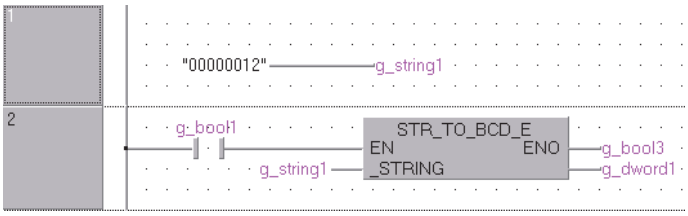


[ST]

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

(b) 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);
```

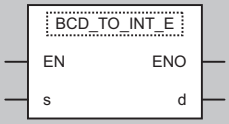
5.1.33 BCD type → word (signed), double word (signed) type conversion BCD_TO_INT(_E), BCD_TO_DINT(_E)

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

BCD_TO_INT(_E)
BCD_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

BCD_TO_INT	BCD_TO_INT_E
BCD_TO_DINT	BCD_TO_DINT_E

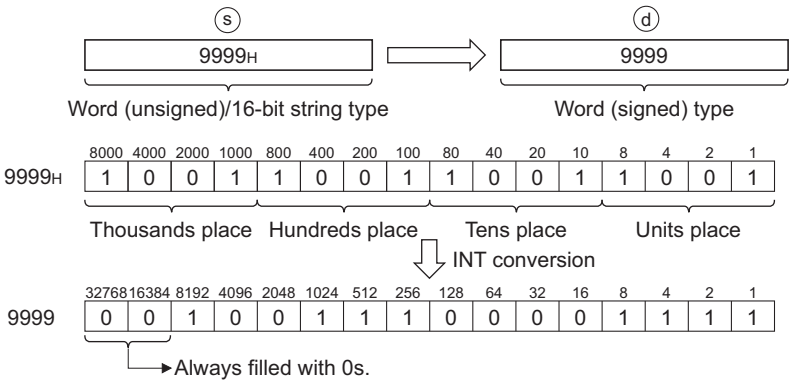
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_BCD):	Input	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

(1) BCD_TO_INT, BCD_TO_INT_E

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

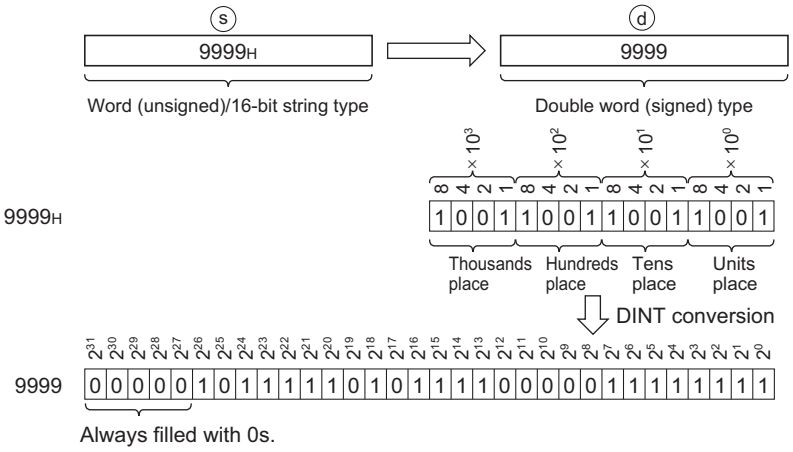


(b) The value to be input to ⓢ is word (unsigned)/16-bit string type data within the range from 0H to 9999H (0 to 9 for each digit).

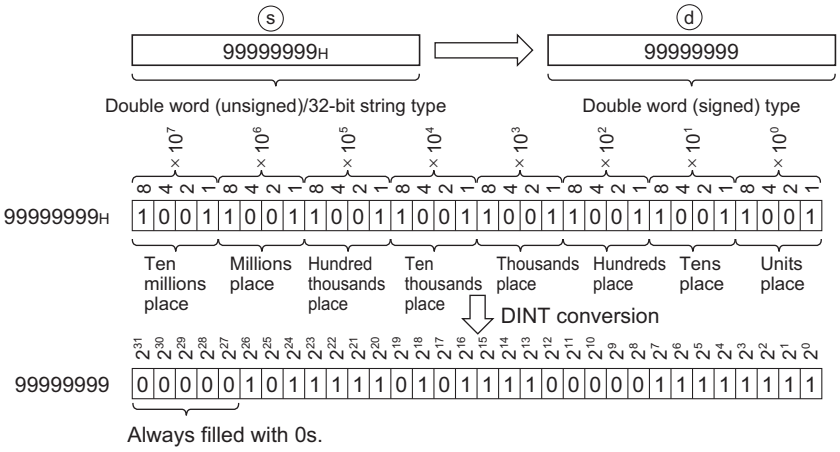
(2) BCD_TO_DINT, BCD_TO_DINT_E

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

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



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



(b) The value to be input to ⓢ 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).

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

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ 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.

- Values other than 0 to 9 are specified for each digit of ⓢ .
  (Error code: 4100)

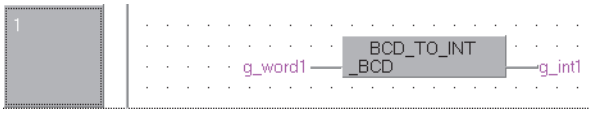
  The error above can be suppressed by turning ON SM722.
  However, the instruction is not executed regardless of whether SM722 is turned ON or OFF if the specified value is out of the available range.
  For the BINP and DBINP instruction, the next operation will not be performed until the command (executing condition) is turned from OFF to ON regardless of the presence or absence of an error.

Program Example

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

(a) Function without EN/ENO (BCD_TO_INT)

[Structured ladder/FBD]

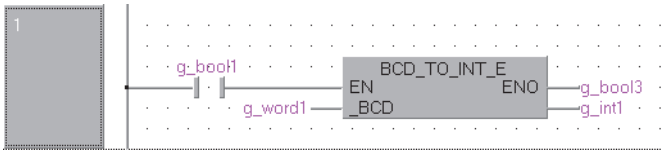


[ST]

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

(b) Function with EN/ENO (BCD_TO_INT_E)

[Structured ladder/FBD]



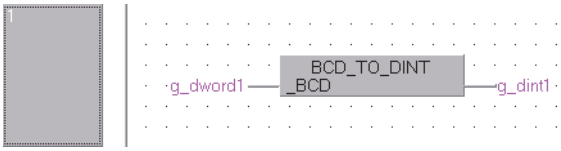
[ST]

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

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

(a) Function without EN/ENO (BCD_TO_DINT)

[Structured ladder/FBD]



[ST]

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

5.1.34 BCD type → string type conversion

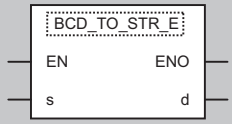
BCD_TO_STR(_E)

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

BCD_TO_STR(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

BCD_TO_STR BCD_TO_STR_E

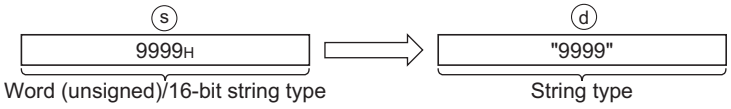
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_BCD):	Input	ANY_BIT
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String (8)

Function

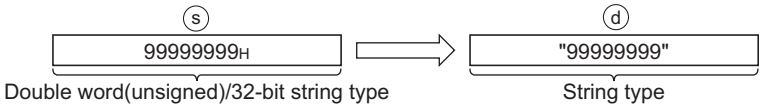
Operation processing

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

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



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



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

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

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

These functions consist of the following common instructions.

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

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

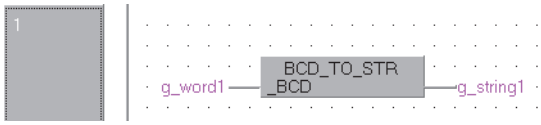
- When word (unsigned)/16-bit string type is specified for ⓢ , ⓢ is outside the range of 0 to 9999. (Error code: 4100)
- When double word (unsigned)/32-bit string type is specified for ⓢ , ⓢ is outside the range of 0 to 99999999. (Error code: 4100)
- The device specified for ⓓ exceeds the corresponding device range. (For Universal model QCPU and LCPU) (Error code: 4101)

Program Example

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

(a) Function without EN/ENO (BCD _TO_STR)

[Structured ladder/FBD]

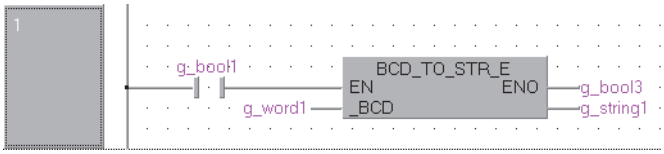


[ST]

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

(b) Function with EN/ENO (BCD_TO_STR_E)

[Structured ladder/FBD]



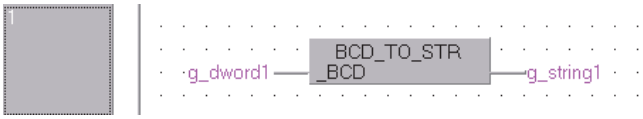
[ST]

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

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

(a) Function without EN/ENO (BCD _TO_STR)

[Structured ladder/FBD]

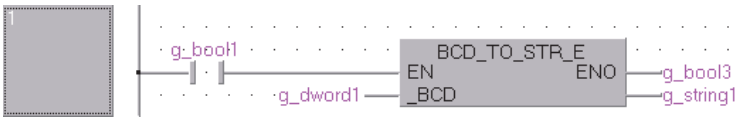


[ST]

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

(b) Function with EN/ENO (BCD_TO_STR_E)

[Structured ladder/FBD]



[ST]

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

5.1.35 Time type → bit type conversion

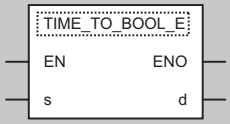
TIME_TO_BOOL(_E)

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

TIME_TO_BOOL(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

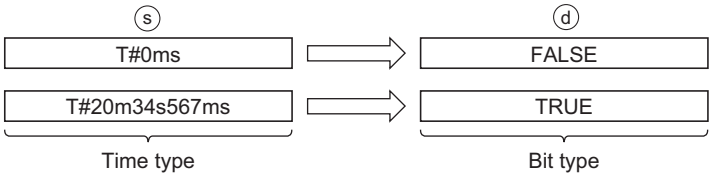
TIME_TO_BOOL TIME_TO_BOOL_E

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

Function

Operation processing

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

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

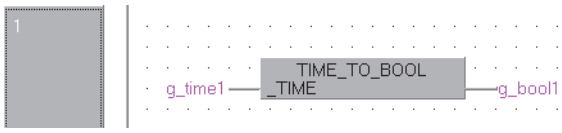
No operation error occurs in the execution of the TIME_TO_BOOL(_E) function.

Program Example

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

(a) Function without EN/ENO (TIME_TO_BOOL)

[Structured ladder/FBD]

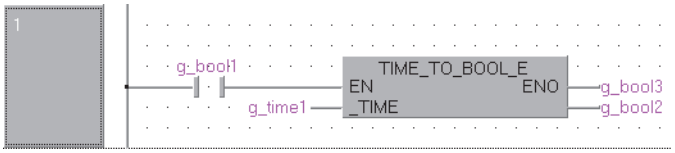


[ST]

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

(b) Function with EN/ENO (TIME_TO_BOOL_E)

[Structured ladder/FBD]



[ST]

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

5.1.36 Time type → word (signed), double word (signed) type conversion

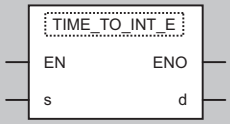
TIME_TO_INT(_E), TIME_TO_DINT(_E)

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

TIME_TO_INT(_E)
TIME_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

TIME_TO_INT	TIME _TO_INT_E
TIME _TO_DINT	TIME _TO_DINT_E

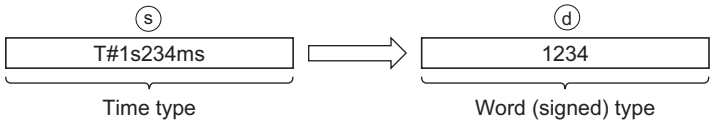
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_TIME):	Input	:Time
Output argument,	ENO:	Execution result (TRUE: Normal execution, FALSE: Error or stop)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

(1) TIME_TO_INT, TIME _TO_INT_E

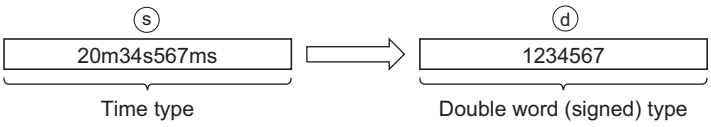
(a) Converts time type data input to ⓢ into word (signed) type data, and outputs the operation result from ⓓ .



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

(2) TIME _TO_DINT, TIME _TO_DINT_E

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

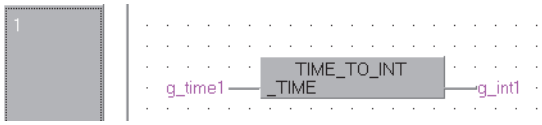
No operation error occurs in the execution of the TIME_TO_INT(_E) and TIME_TO_DINT(_E) functions.

Program Example

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

(a) Function without EN/ENO (TIME _TO_INT)

[Structured ladder/FBD]

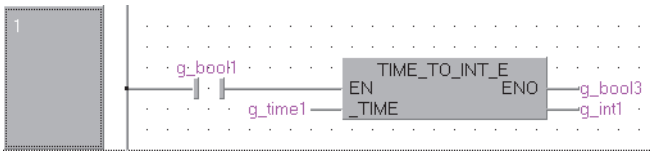


[ST]

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

(b) Function with EN/ENO (TIME_TO_INT_E)

[Structured ladder/FBD]



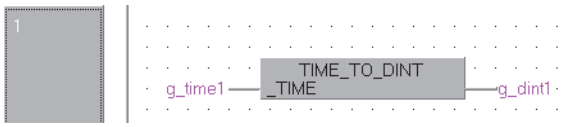
[ST]

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

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

(a) Function without EN/ENO (TIME _TO_DINT)

[Structured ladder/FBD]



[ST]

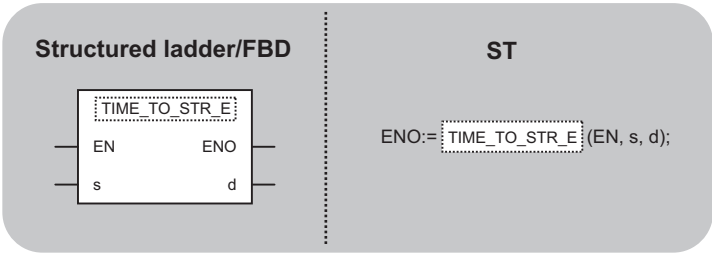
```
g_dint1 := TIME_TO_DINT (g_time1);
```

5.1.37 Time type → string type conversion

TIME_TO_STR(_E)

TIME_TO_STR(_E)

(_E: With EN/ENO)



indicates any of the following functions.

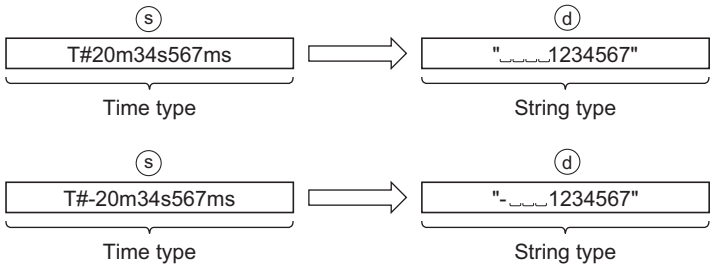
TIME_TO_STR TIME _TO_STR_E

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	:String (11)

Function

Operation processing

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



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

(3) The operation results stored to ⓓ are as follows.

(a) For the first character, 20H (space) is stored if the BIN data is positive, and 2DH (-) is stored if it is negative.

(b) 20H (space) is stored to the left of significant figures.

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ 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.

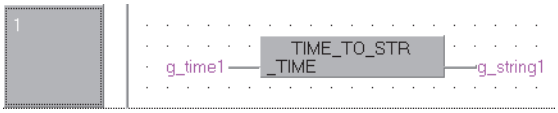
- The device specified for exceeds the corresponding device range.
  (For Universal model QCPU and LCPU) (Error code: 4101)

Program Example

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

(a) Function without EN/ENO (TIME_TO_STR)

[Structured ladder/FBD]

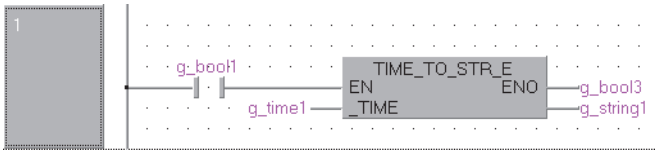


[ST]

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

(b) Function with EN/ENO (TIME_TO_STR_E)

[Structured ladder/FBD]



[ST]

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

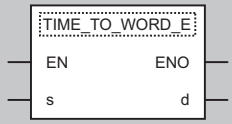
5.1.38 Time type → word (unsigned)/16-bit string, double word (unsigned)/32-bit string type conversion TIME_TO_WORD(_E), TIME_TO_DWORD(_E)

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

TIME_TO_WORD(_E)
TIME_TO_DWORD(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

TIME_TO_WORD	TIME _TO_WORD_E
TIME _TO_DWORD	TIME _TO_DWORD_E

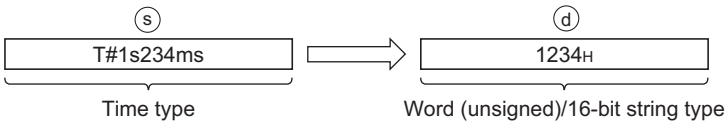
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_TIME):	Input	:Time
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string

Function

Operation processing

(1) TIME_TO_WORD, TIME _TO_WORD_E

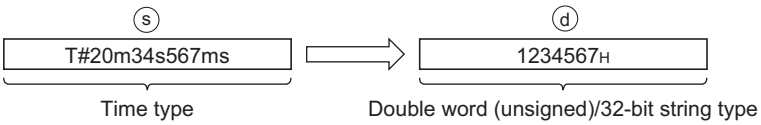
Converts time type data input to ⓢ into word (unsigned)/16-bit string type data, and outputs the operation result from ⓓ .



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

(3) TIME _TO_DWORD, TIME _TO_DWORD_E

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



Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

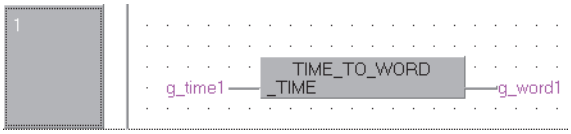
No operation error occurs in the execution of the TIME_TO_WORD(_E) and TIME_TO_DWORD(_E) functions.

Program Example

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

(a) Function without EN/ENO (TIME _TO_WORD)

[Structured ladder/FBD]

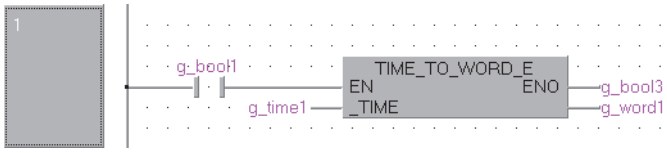


[ST]

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

(b) Function with EN/ENO (TIME_TO_WORD_E)

[Structured ladder/FBD]



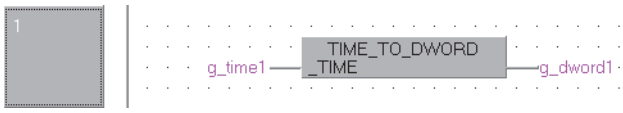
[ST]

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

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

(a) Function without EN/ENO (TIME _TO_DWORD)

[Structured ladder/FBD]



[ST]

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

5.1.39 Bit array → word (signed) type, word (unsigned)/16-bit string type, double word (signed) type, double word (unsigned)/32-bit string type conversion

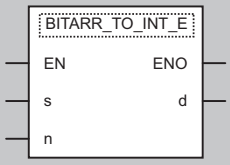
BITARR_TO_INT(_E), BITARR_TO_DINT(_E)

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

BITARR_TO_INT(_E)
BITARR_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

BITARR_TO_INT	BITARR_TO_INT_E
BITARR_TO_DINT	BITARR_TO_DINT_E

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

Function

Operation processing

(1) BITARR_TO_INT, BITARR_TO_INT_E

Converts number of bits specified for n starting from the bit array element input to ⓢ into word (signed) type or word (unsigned)/16-bit string type data, and outputs the operation result from ⓓ .
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.

(2) BITARR_TO_DINT, BITARR_TO_DINT_E

Converts number of bits specified for n starting from the bit array element input to ⓢ into double word (signed) type or double word (unsigned)/32-bit string type data, and outputs the operation result from ⓓ .
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

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

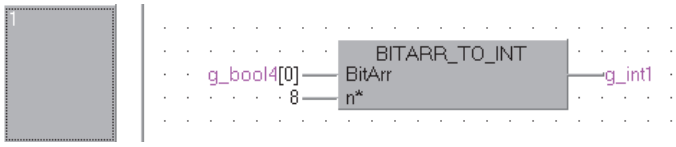
No operation error occurs in the execution of the BITARR_TO_INT(_E) and BITARR_TO_DINT(_E) functions.

Program Example

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

(a) Function without EN/ENO (BITARR_TO_INT)

[Structured ladder/FBD]

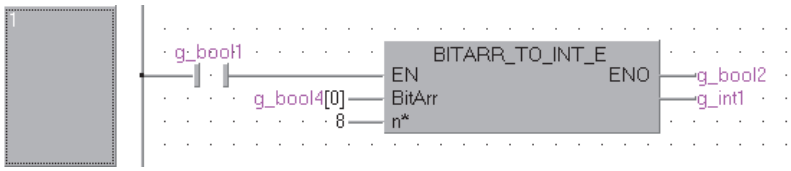


[ST]

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

(b) 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);
```

5.1.40 Word (signed) type, word (unsigned)/16-bit string type, double word (signed) type, double word (unsigned)/32-bit string type → bit array conversion

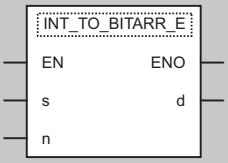
INT_TO_BITARR(_E), DINT_TO_BITARR(_E)

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

INT_TO_BITARR(_E)
DINT_TO_BITARR(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

INT_TO_BITARR	INT_TO_BITARR_E
DINT_TO_BITARR	DINT_TO_BITARR_E

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

Function

Operation processing

(1) 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 ⓢ to ⓓ.
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.

(2) 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 ⓢ to ⓓ.
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

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

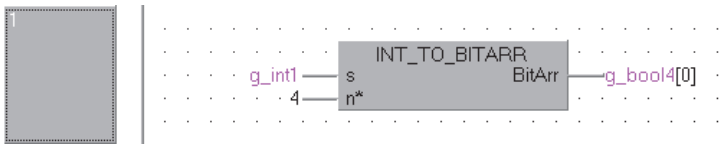
No operation error occurs in the execution of the INT_TO_BITARR(_E) and DINT_TO_BITARR(_E) functions.

Program Example

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

(a) Function without EN/ENO (INT_TO_BITARR)

[Structured ladder/FBD]

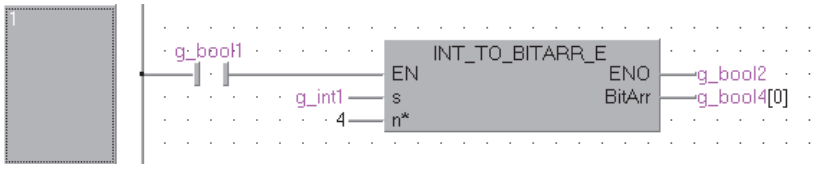


[ST]

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

(b) 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]);
```

5.1.41 Bit array copy

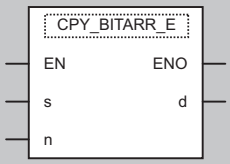
CPY_BITARR(_E)

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

CPY_BITARR(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

CPY_BITARR CPY_BITARR_E

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	:Bit

Function

Operation processing

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

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

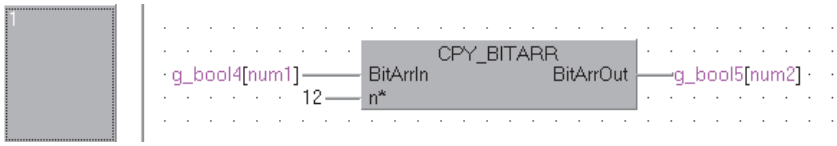
No operation error occurs in the execution of the CPY_BITARR(_E) function.

Program Example

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

(a) Function without EN/ENO (CPY_BITARR)

[Structured ladder/FBD]

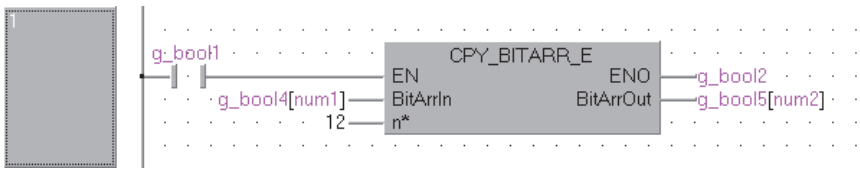


[ST]

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

(b) 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]);
```

5.1.42 Specified bit read of word (signed) type data

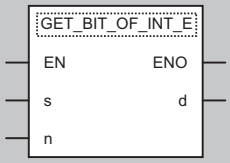
GET_BIT_OF_INT(_E)

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

GET_BIT_OF_INT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

GET_BIT_OF_INT GET_BIT_OF_INT_E

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

Function

Operation processing

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

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

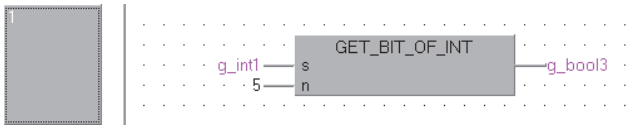
No operation error occurs in the execution of the GET_BIT_OF_INT(_E) function.

Program Example

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

(a) Function without EN/ENO (GET_BIT_OF_INT)

[Structured ladder/FBD]

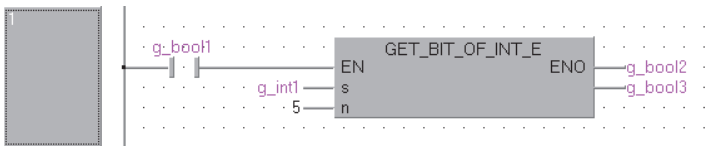


[ST]

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

(b) 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);
```

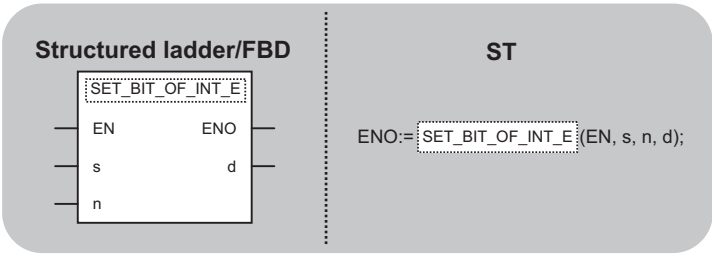
5.1.43 Specified bit write of word (signed) type data

SET_BIT_OF_INT(_E)

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

SET_BIT_OF_INT(_E)

(_E: With EN/ENO)



indicates any of the following functions.

SET_BIT_OF_INT SET_BIT_OF_INT_E

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)

Function

Operation processing

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

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

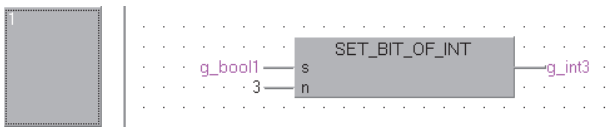
No operation error occurs in the execution of the SET_BIT_OF_INT(_E) function.

Program Example

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

(a) Function without EN/ENO (SET_BIT_OF_INT)

[Structured ladder/FBD]

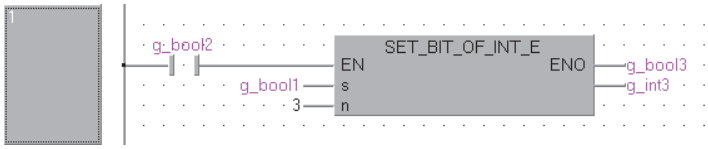


[ST]

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

(b) 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);
```

5.1.44 Specified bit copy of word (signed) type data

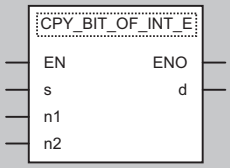
CPY_BIT_OF_INT(_E)

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

CPY_BIT_OF_INT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

[ ] indicates any of the following functions.

CPY_BIT_OF_INT CPY_BIT_OF_INT_E

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)

Function

Operation processing

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

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

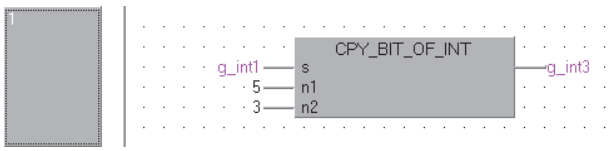
No operation error occurs in the execution of the CPY_BIT_OF_INT(_E) function.

Program Example

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

(a) Function without EN/ENO (CPY_BIT_OF_INT)

[Structured ladder/FBD]

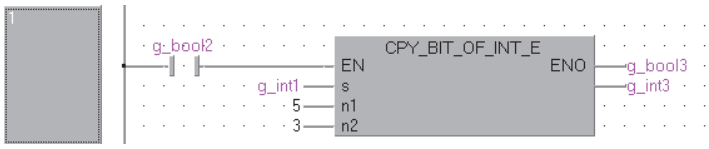


[ST]

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

(b) Function with EN/ENO (CPY_BIT_OF_INT_E)

[Structured ladder/FBD]



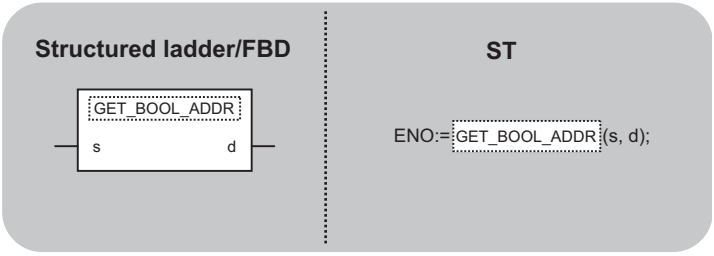
[ST]

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

5.1.45 Nonessential type conversion GET_BOOL_ADDR, GET_INT_ADDR, GET_WORD_ADDR

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

GET_BOOL_ADDR
GET_INT_ADDR
GET_WORD_ADDR



indicates any of the following functions.
GET_BOOL_ADDR
GET_INT_ADDR
GET_WORD_ADDR

Input argument,	s:	Input	: ANY
Output argument,	d:	Output	: Bit, word (signed), word (unsigned)/16-bit string

Function

Operation processing

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

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

(2) Rounding error may occur when specifying the input value to ⓢ by programming tool. For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

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

Operation Error

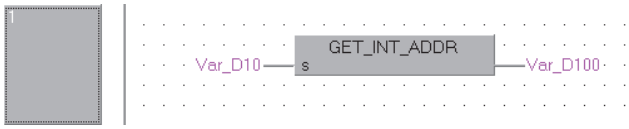
No operation error occurs in the execution of the GET_BOOL_ADDR, GET_INT_ADDR, and GET_WORD_ADDR functions.

Program Example

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

GET_INT_ADDR

[Structured ladder/FBD]



[ST]

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

5.2 Standard Functions of One Numeric Variable

5.2.1 Absolute value

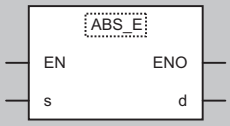
ABS(_E)

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

ABS(_E)

_E: With EN/ENO

Structured ladder/FBD



ST

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

ABS_E indicates any of the following functions.

ABS ABS_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_IN):	Input	:ANY_NUM
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:ANY_NUM

Function

Operation processing

(1) Outputs the absolute value of word (signed), double word (signed), single-precision real or double-precision real type data input to ⓢ from ⓓ in the same data type as that of ⓢ. Assuming that the input value is A and the operation output value is B, the relationship is expressed by the following equality.

$B = |A|$

(2) The value to be input to ⓢ is word (signed), double word (signed), single-precision real or double-precision real type data.

(3) When the data type of ⓢ is word (signed) type and the input value is -32768, -32768 is output from ⓓ.

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

(4) Rounding error may occur when specifying single-precision real or double-precision real type data to ⓢ by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

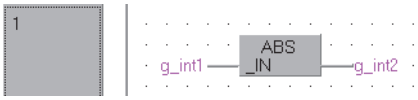
No operation error occurs in the execution of the ABS(_E) function.

Program Example

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

(a) Function without EN/ENO (ABS)

[Structured ladder/FBD]

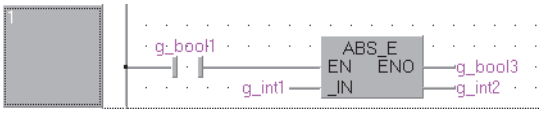


[ST]

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

(b) Function with EN/ENO (ABS_E)

[Structured ladder/FBD]



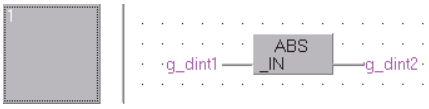
[ST]

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

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

(a) Function without EN/ENO (ABS)

[Structured ladder/FBD]

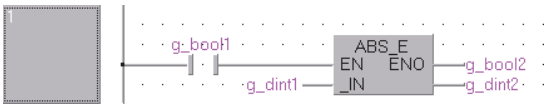


[ST]

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

(b) Function with EN/ENO (ABS_E)

[Structured ladder/FBD]



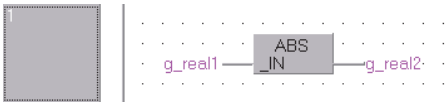
[ST]

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

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

(a) Function without EN/ENO (ABS)

[Structured ladder/FBD]

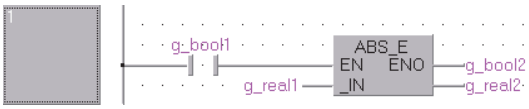


[ST]

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

(b) Function with EN/ENO (ABS_E)

[Structured ladder/FBD]



[ST]

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

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

(a) Function without EN/ENO (ABS)

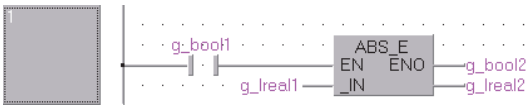
[Structured ladder/FBD]

[ST]

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

(b) 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

5.3.1 Addition

ADD_E

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

ADD_E

_E: With EN/ENO

indicates any of the following functions.

ADD_E

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

Function

Operation processing

(1) Performs addition (ⓢ1 +ⓢ2 +···+ⓢ28) on word (signed), double word (signed), single-precision real or double-precision real type data input to ⓢ1 to ⓢ28 , and outputs the operation result from ⓓ in the same data type as that of ⓢ1 to ⓢ28.
(Example) Word (signed) type data

(2) The values to be input to ⓢ1 to ⓢ28 are word (signed), double word (signed), single-precision real or double-precision real type data.

(3) The number of pins for ⓢ can be changed in the range from 2 to 28.

(4) If an underflow/overflow occurs in the operation result, data is output from ⓓ as follows.

(a) 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.

–32767 + (–2) = 32766
(8000H) (FFFEH) (7FFEH)

Since the highest-order bit is 0, the result value is positive.

(b) 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
(7FFFFFFFH) (0002H) (80000001H)

Since the highest-order bit is 1, the result value is negative.

–2147483648 + (–2) = 2147483646
(80000000H) (FFFEH) (7FFFFFFEH)

Since the highest-order bit is 0, the result value is positive.

(5) Rounding error may occur when specifying single-precision real or double-precision real type data to ⓢ1 through ⓢ28 by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

(1) Function without EN/ENO
An operation is executed and the operation value is output from ⓓ .

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

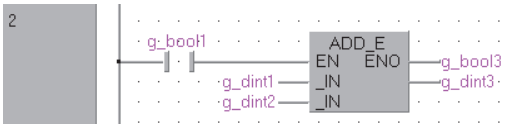
Operation Error

No operation error occurs in the execution of the ADD_E function.

Program Example

The program which performs addition (ⓢ1 +ⓢ2) on double word (signed) type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

[Structured ladder/FBD]



[ST]

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

5.3.2 Multiplication

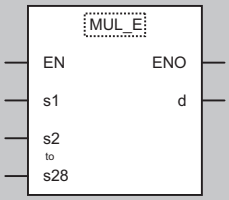
MUL_E

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

MUL_E

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:=MUL_E(EN, s1, s2···s28, d);

indicates any of the following functions.

MUL_E

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

Function

Operation processing

(1) Performs multiplication (ⓢ1 × ⓢ2 × ··· × ⓢ28) on word (signed), double word (signed), single-precision real or double-precision real type data input to ⓢ1 to ⓢ28 , and outputs the operation result from ⓓ in the same data type as that of ⓢ1 to ⓢ28.

(Example) Word (signed) type data

(2) The values to be input to ⓢ1 to ⓢ28 are word (signed), double word (signed), single-precision real or double-precision real type data.

(3) The number of pins for ⓢ can be changed in the range from 2 to 28.

(4) If an underflow/overflow occurs in the operation result, data is output from ⓓ as follows.

(a) 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.

(b) 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.

(5) Rounding error may occur when specifying single-precision real or double-precision real type data to ⓢ1 through ⓢ28 by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

(1) Function without EN/ENO
An operation is executed and the operation value is output from ⓓ .

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

POINT

If the operation result exceeds the data type range, convert the data type of the input data before the operation.

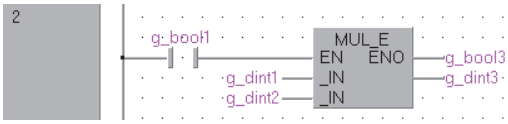
Operation Error

No operation error occurs in the execution of the MUL_E function.

Program Example

The program which performs multiplication (ⓢ1 × ⓢ2) on double word (signed) type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

[Structured ladder/FBD]



[ST]

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

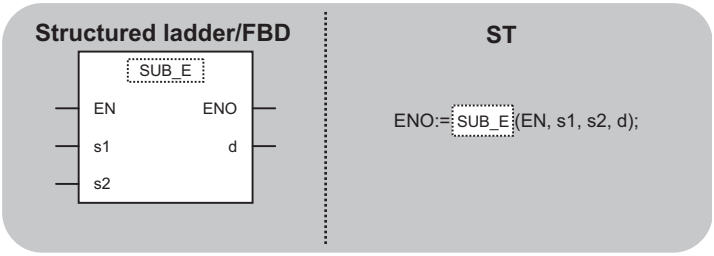
5.3.3 Subtraction

SUB_E

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

SUB_E

(_E: With EN/ENO)



indicates any of the following functions.

SUB_E

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

Function

Operation processing

(1) Performs subtraction (ⓢ1 − ⓢ2) on word (signed), double word (signed), single-precision real or double-precision real type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

(Example) Word (signed) type data

(2) The values to be input to ⓢ1 and ⓢ2 are word (signed), double word (signed), single-precision real or double-precision real type data.

(3) If an underflow/overflow occurs in the operation result, data is output from ⓓ as follows.

(a) 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
(7FFFH) (FFFEH) (8001H)
Since the highest-order bit is 1, the result value is negative.

−32767 − 2 = 32766
(8000H) (0002H) (7FFEH)
Since the highest-order bit is 0, the result value is positive.

(b) 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
(7FFFFFFFH) (FFFEH) (80000001H)
Since the highest-order bit is 1, the result value is negative.

−2147483648 − 2 = 2147483646
(80000000H) (0002H) (7FFFFFFEH)
Since the highest-order bit is 0, the result value is positive.

(4) Rounding error may occur when specifying single-precision real or double-precision real type data to ⓢ1, ⓢ2 by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

(1) Function without EN/ENO
An operation is executed and the operation value is output from ⓓ .

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

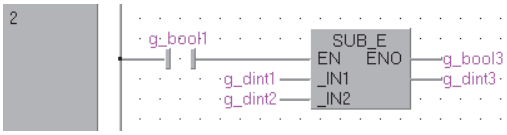
Operation Error

No operation error occurs in the execution of the SUB_E function.

Program Example

The program which performs subtraction (ⓢ1 − ⓢ2) on double word (signed) type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

[Structured ladder/FBD]



[ST]

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

5.3.4 Division

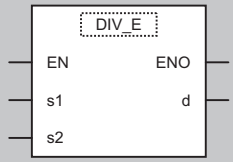
DIV_E

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

DIV_E

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:= DIV_E (EN, s1, s2, d);

indicates any of the following functions.

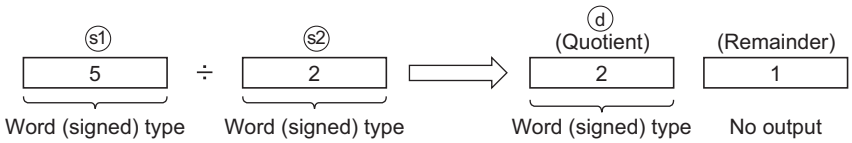
DIV_E

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

Function

Operation processing

(1) Performs division (ⓢ1 ÷ⓢ2) on word (signed), double word (signed), single-precision real or double-precision real type data input to ⓢ1 and ⓢ2 , and outputs the quotient of the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .
(Example) Word (signed) type data



(2) The values to be input to ⓢ1 and ⓢ2 are word (signed), double word (signed), single-precision real or double-precision real type data.
(The value to be input to ⓢ2 must be other than 0.)

(3) Rounding error may occur when specifying single-precision real or double-precision real type data to ⓢ1, ⓢ2 by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

(1) Function without EN/ENO
The following table shows the operation results.

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

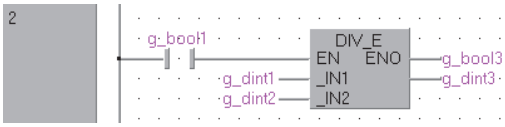
An operation error occurs in the following case.

- The value to be input to ⓢ2 is 0. (Division by 0) (Error code: 4100)

Program Example

The program which performs division (ⓢ1 ÷ⓢ2) on double word (signed) type data input to ⓢ1 and ⓢ2 , and outputs the quotient of the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

[Structured ladder/FBD]



[ST]

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

5.3.5 Modules operation

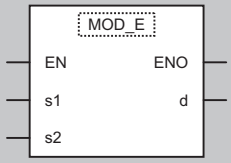
MOD(_E)

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

MOD(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

```
ENO:=MOD_E(EN, s1, s2, d);
```

indicates any of the following functions.

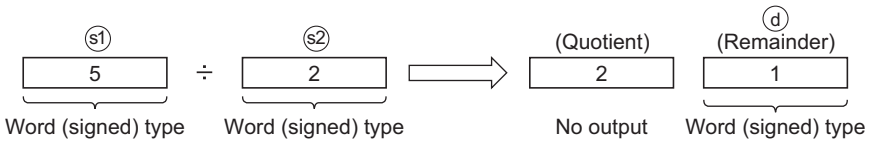
MOD	MOD_E

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

Function

Operation processing

(1) Performs division (ⓢ1 ÷ ⓢ2) on word (signed) or double word (signed) type data input to ⓢ1 and ⓢ2 , and outputs the remainder of the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .
(Example) Word (signed) type data



(2) The values to be input to ⓢ1 and ⓢ2 are word (signed) or double word (signed) type data. (Note that the value to be input to ⓢ2 must be other than 0.)

Operation result

(1) Function without EN/ENO
The following table shows the operation results.

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

An operation error occurs in the following case.

- The value to be input to ⓢ2 is 0. (Division by 0) (Error code: 4100)

Program Example

The program which performs division (ⓢ1 ÷ⓢ2) on double word (signed) type data input to ⓢ1 and ⓢ2 , and outputs the remainder of the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

(a) Function without EN/ENO (MOD)

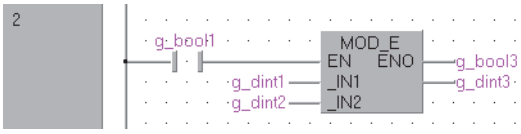
[Structured ladder/FBD]

[ST]

```
g_dint3:= g_dint1 MOD g_dint2;
```

(b) Function with EN/ENO (MOD_E)

[Structured ladder/FBD]



[ST]

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

5.3.6 Exponentiation

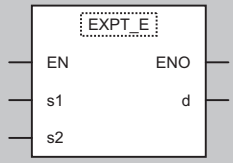
EXPT(_E)

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

EXPT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

```
ENO:= EXPT_E (EN, s1, s2, d);
```

[ ] indicates any of the following functions.

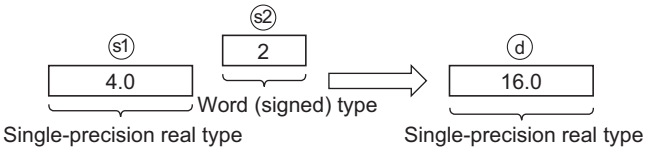
EXPT EXPT _E

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

Function

Operation processing

(1) Performs exponentiation ⓢ2 on single-precision real or double-precision real type data input to ⓢ2 and word (signed), double word (signed), single-precision real or double-precision real type data input to ⓢ1, and outputs the operation result from ⓓ .



(2) Rounding error may occur when specifying single-precision real or double-precision real type data to ⓢ1, ⓢ2 by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

These functions consist of the following common instructions.

When ⓢ1 is single-precision real number, ⓢ2 is word (signed): LOG, FLT

When ⓢ1 is single-precision real number, ⓢ2 is double word (signed): LOG, DFLT

When ⓢ1 is single-precision real number, ⓢ2 is single-precision real number: LOG

When ⓢ1 is single-precision real number, ⓢ2 is double-precision real number: LOGD, DFLTD

When ⓢ1 is double-precision real number, ⓢ2 is word (signed): LOGD

When ⓢ1 is double-precision real number, ⓢ2 is double word (signed): LOGD, FLTD

When ⓢ1 is double-precision real number, ⓢ2 is single-precision real number: LOGD, DFLTD

When ⓢ1 is double-precision real number, ⓢ2 is double-precision real number: LOGD

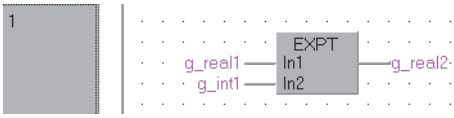
For details of an error which occurs when the function is executed, refer to MELSEC-Q/L Structured Programming Manual (Common Instructions).

Program Example

The program which performs exponentiation and outputs the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

(a) Function without EN/ENO (EXPT)

[Structured ladder/FBD]

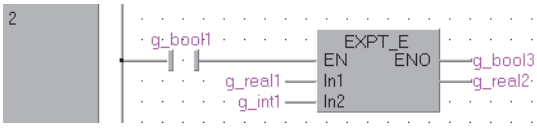


[ST]

```
g_real2:= EXPT(g_real1, g_int1);
```

(b) Function with EN/ENO (EXPT_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := EXPT_E(g_bool1, g_real1, g_int1, g_real2);
```

5.3.7 Move operation

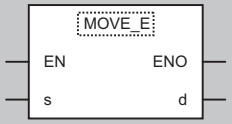
MOVE(_E)

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

MOVE(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:= MOVE_E (EN, s, d);

(dotted box) indicates any of the following functions.

MOVE MOVE_E

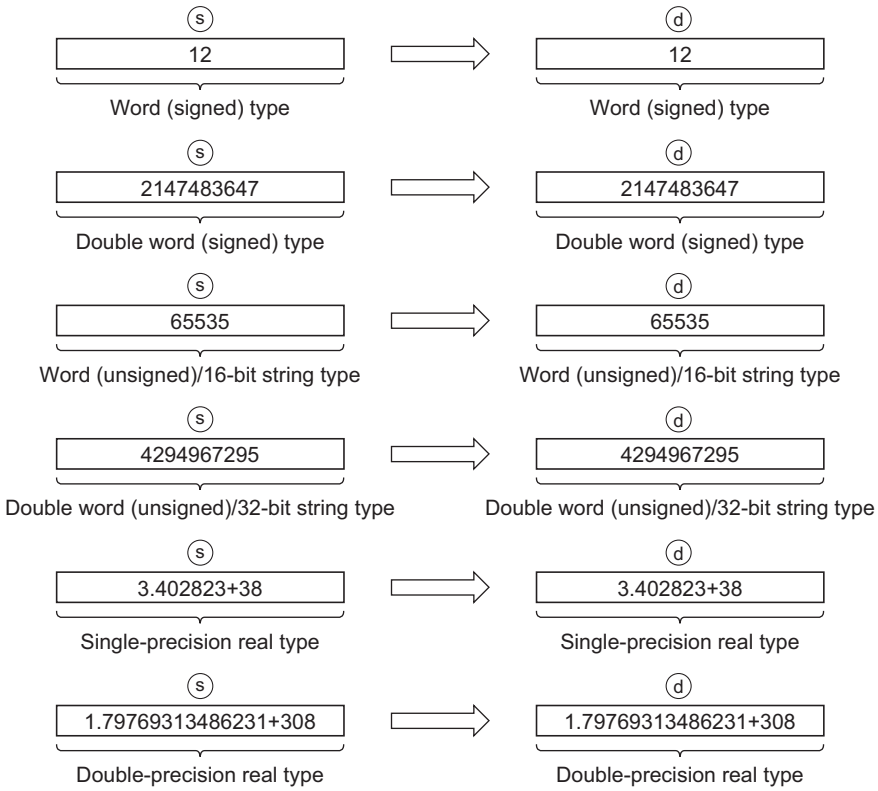
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1(_IN1):	Input	:ANY
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:ANY

Function

Operation processing

(1) Moves the data input for ⓢ from ⓓ in the same data type as that of ⓢ.

(2) The values to be specified to ⓢ and ⓓ 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 ⓢ and ⓓ.



(3) Rounding error may occur when specifying single-precision real or double-precision real type data to ⓢ by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

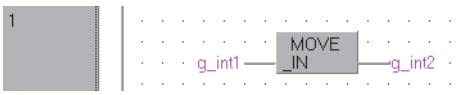
No operation error occurs in the execution of the MOVE(_E) function.

Program Example

The program which moves the word (signed) type data input to ⓢ to ⓓ .

(a) Function without EN/ENO (MOVE)

[Structured ladder/FBD]



[ST]

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

(b) 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

5.4.1 Boolean AND, boolean OR, boolean exclusive OR, and boolean NOT

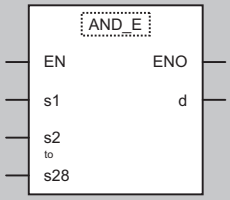
AND_E, OR_E, XOR_E, NOT(_E)

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

AND_E
OR_E
XOR_E
NOT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:=AND_E(EN, s1, s2···s28, d);

indicates any of the following functions.

AND_E
OR_E
XOR_E
NOT NOT_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1 to s28(_IN1): (s1 only for NOT(_E))	Input	:ANY_BIT
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:ANY_BIT

Function

Operation processing

(1) AND_E

(a) 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.
(Example) Word (unsigned)/16-bit string type data

s1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1
								AND								
s2	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0
									⇩ Boolean AND							
d	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0

(b) The number of pins of variable 's' can be changed in the range from 2 to 28.

(2) OR_E

(a) 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.

(Example) Word (unsigned)/16-bit string type data

s1	1	1	0	0	1	1	1	1	0	0	0	0	1	1	1	1

OR

s2	0	0	0	0	0	0	1	0	0	0	1	1	0	1	0	0

⇩ Boolean OR

d	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1	1

(b) The number of pins of variable 's' can be changed in the range from 2 to 28.

(3) XOR_E

(a) 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.

(Example) Word (unsigned)/16-bit string type data

s1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

OR

s2	0	0	0	1	1	0	1	1	1	1	1	1	0	0	0	0

⇩ Boolean exclusive OR

d	1	0	1	1	0	0	0	1	0	1	0	1	1	0	1	0

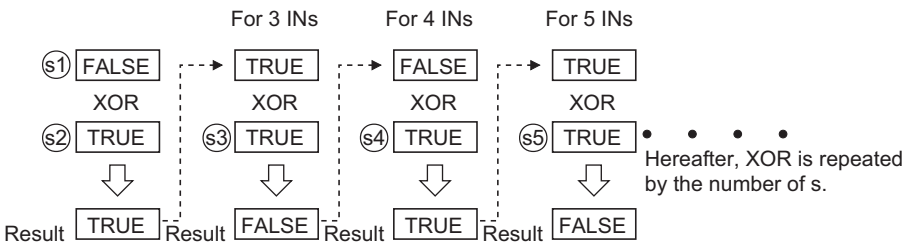
(b) The number of pins of variable 's' can be changed in the range from 2 to 28.

(c) 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.

(Example) Bit type data



(4) NOT, NOT_E

(a) Performs Boolean NOT on bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data input to ⓢ1 bit by bit, and outputs the operation result from ⓓ in the same data type as that of ⓢ1 .
(Example) Word (unsigned)/16-bit string type data

ⓢ1	0	1	1	0	1	0	1	1	0	0	0	0	1	1	1	1

NOT

ⓓ	1	0	0	1	0	1	0	0	1	1	1	1	0	0	0	0

(b) The value to be input to variables ⓢ1 to ⓢ28 is bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data.

Operation result

(1) Function without EN/ENO
An operation is executed and the operation value is output from ⓓ .

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

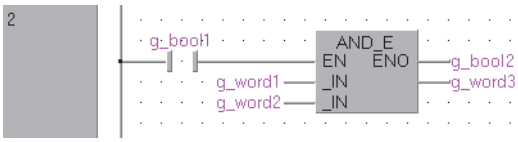
Operation Error

No operation error occurs in the execution of the AND_E, OR_E, XOR_E, and NOT(_E) functions.

Program Example

(1) The program which performs Boolean AND on bit, word (unsigned)/16-bit string type data input to ⓢ1 to ⓢ28 bit by bit, and outputs the operation result from ⓓ in the same data type as that of ⓢ1 to ⓢ28 .

[Structured ladder/FBD]

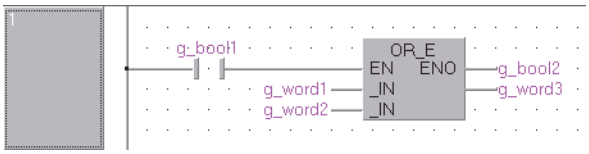


[ST]

```
g_bool2 := AND_E (g_bool1, g_word1, g_word2, g_word3);
```

(2) The program which performs Boolean OR on bit, word (unsigned)/16-bit string type data input to ⓢ1 to ⓢ28 bit by bit, and outputs the operation result from ⓓ in the same data type as that of ⓢ1 to ⓢ28 .

[Structured ladder/FBD]

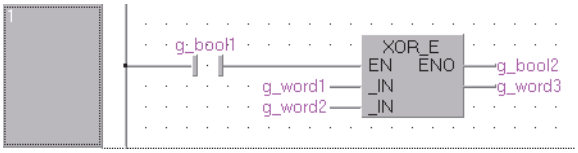


[ST]

```
g_bool2 := OR_E(g_bool1, g_word1, g_word2, g_word3);
```

(3) The program which performs Boolean XOR on bit, word (unsigned)/16-bit string type data input to ⓢ1 to ⓢ28 bit by bit, and outputs the operation result from ⓓ in the same data type as that of ⓢ1 to ⓢ28 .

[Structured ladder/FBD]



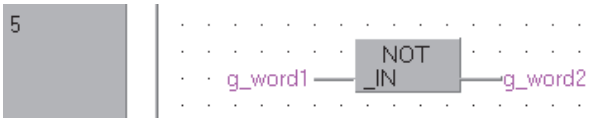
[ST]

```
g_bool2 := XOR_E(g_bool1, g_word1, g_word2, g_word3);
```

(4) The program which performs Boolean NOT on bit, word (unsigned)/16-bit string type data input to ⓢ1 bit by bit, and outputs the operation result from ⓓ in the same data type as that of ⓢ1 .

(a) Function without EN/ENO (NOT)

[Structured ladder/FBD]

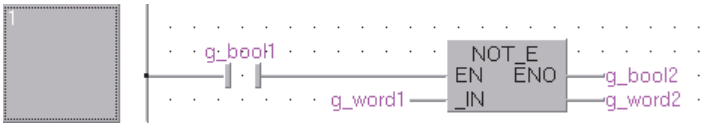


[ST]

```
g_word2 :=NOT (g_word1);
```

(b) 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

5.5.1 Selection

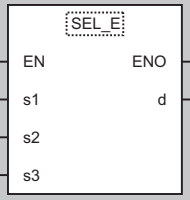
SEL(_E)

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

SEL(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

```
ENO:= SEL_E (EN, s1, s2, s3, d);
```

[ ] indicates any of the following functions.

SEL SEL_E

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

Function

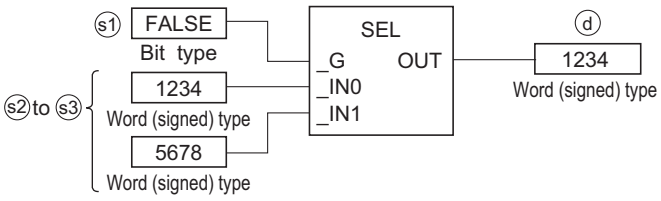
Operation processing

(1) 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.

(Example) s2 and s3 are word (signed) type data



(2) The input value to ⓢ1 is data value of bit type.

(3) The input value to ⓢ2 , ⓢ3 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.

(4) Rounding error may occur when specifying single-precision real or double-precision real type data to ⓢ2, ⓢ3 by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

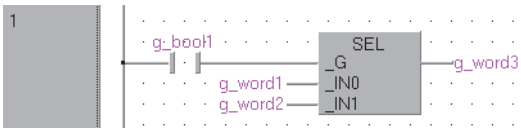
No operation error occurs in the execution of the SEL(_E) function.

Program Example

The program which selects either of values input to ⓢ2 and ⓢ3 according to the value input to ⓢ1 , and outputs the operation result from ⓓ in the same data type as that of ⓢ2 and ⓢ3 .

(a) Function without EN/ENO (SEL)

[Structured ladder/FBD]

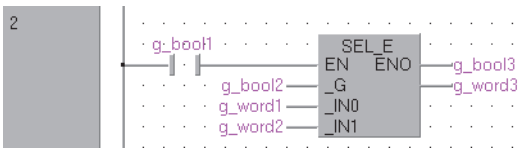


[ST]

```
g_word3 := SEL (g_bool1, g_word1, g_word2);
```

(b) Function with EN/ENO (SEL_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := SEL_E (g_bool1, g_bool2, g_word1, g_word2, g_word3);
```

5.5.2 Maximum/Minimum selection

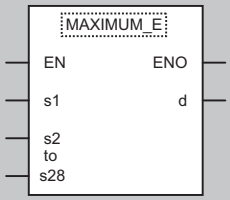
MAXIMUM(_E), MINIMUM(_E)

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

MAXIMUM(_E)
MINIMUM(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:= MAXIMUM_E (EN, s1, s2…s28, d);

[ ] indicates any of the following functions.

MAXIMUM	MAXIMUM_E
MINIMUM	MINIMUM_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1 to s28(_IN1):	Input	:ANY_SIMPLE
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:ANY_SIMPLE

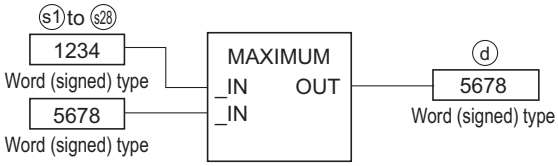
Function

Operation processing

(1) 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.

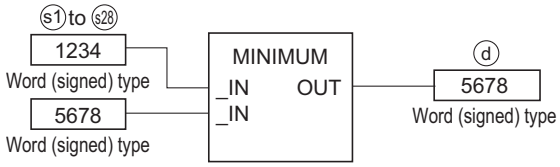
(Example) Word (signed) type data



(2) 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 ⓢ1 to ⓢ28 , and outputs the operation result from ⓓ in the same data type as that of ⓢ1 to ⓢ28 .

(Example) Word (signed) type data



(3) The values to be input to ⓢ1 to ⓢ28 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.

(4) Rounding error may occur when specifying single-precision real or double-precision real type data to ⓢ1 through ⓢ28 by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

(5) The number of pins of variable 's' can be changed in the range from 2 to 28.

(6) If word (unsigned) type/16-bit string type/double word (unsigned) type/32-bit string type is specified for ⓓ , warning C9026 occurs.

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

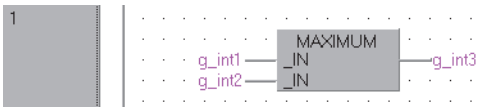
No operation error occurs in the execution of the MAXIMUM(_E) and MINIMUM(_E) function.

Program Example

(1) The program which outputs the maximum value of the word (signed) data input to variables ⓢ1 to ⓢ28 from ⓓ in the same data type as that of ⓢ1 to ⓢ28 .

(a) Function without EN/ENO (MAXIMUM)

[Structured ladder/FBD]

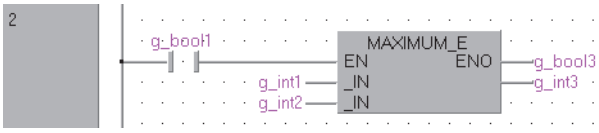


[ST]

```
g_int3 := MAXIMUM(g_int1, g_int2);
```

(b) Function with EN/ENO (MAXIMUM_E)

[Structured ladder/FBD]



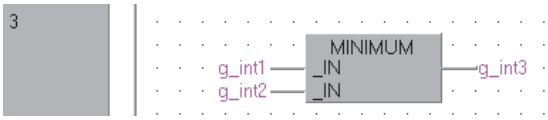
[ST]

```
g_bool3 := MAXIMUM_E (g_bool1, g_int1, g_int2, g_int3);
```

(2) The program which outputs the minimum value of the word (signed) data input to variables ⓢ1 to ⓢ28 from ⓓ in the same data type as that of ⓢ1 to ⓢ28 .

(a) Function without EN/ENO (MINIMUM)

[Structured ladder/FBD]



[ST]

```
g_int3 := MINIMUM(g_int1, g_int2);
```

5.5.3 Upper/Lower limit control

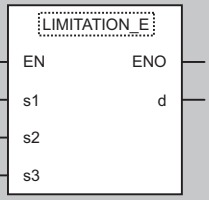
LIMITATION(_E)

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

LIMITATION(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

```
ENO:=LIMITATION_E(EN, s1, s2, s3, d);
```

[ ] indicates any of the following functions.

LIMITATION LIMITATION_E

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

Function

Operation processing

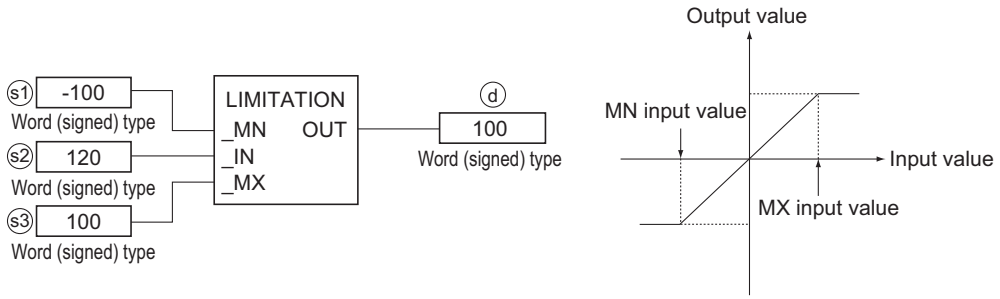
(1) Selects the value to be output among the bit, word (signed), double word (signed), word (unsigned)/16-bit string, double word (unsigned)/32-bit string, or single-precision real type, double-precision real, string, or time type data input to ⓢ1 , ⓢ2 , and ⓢ3 according to their values, and outputs the operation result from ⓓ in the same data type as that of ⓢ1 to ⓢ3 .

(a) When the input value of ⓢ2 > the input value of ⓢ3 , outputs the input value ⓢ3 from ⓓ .

(b) When the input value of ⓢ2 < the input value of ⓢ1 , outputs the input value ⓢ1 from ⓓ .

(c) When the input value of ⓢ1 ≦ the input value of ⓢ2 ≦ the input value of ⓢ3 , outputs the input value of ⓢ2 from ⓓ .

(Example) Word (signed) type data



(2) 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)

(3) Rounding error may occur when specifying single-precision real or double-precision real type data to s1, s2, or s3 by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

(4) 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

(1) Function without EN/ENO
An operation is executed and the operation value is output from d .

(2) 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

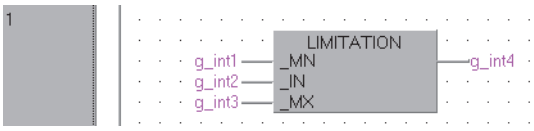
No operation error occurs in the execution of the LIMITATION(_E) function.

Program Example

The program which outputs the values input to variables ⓢ1 , ⓢ2 , and ⓢ3 according to the word (signed) data from ⓓ in the same data type as that of ⓢ1 , ⓢ2 , and ⓢ3 .

(a) Function without EN/ENO (LIMITATION)

[Structured ladder/FBD]

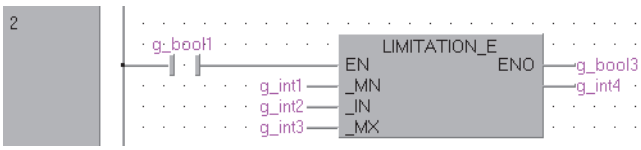


[ST]

```
g_int4 := LIMITATION (g_int1, g_int2, g_int3);
```

(b) Function with EN/ENO (LIMITATION_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := LIMITATION_E (g_bool1, g_int1, g_int2, g_int3, g_int4);
```

5.5.4 Multiplexer

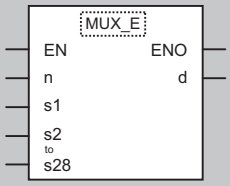
MUX(_E)

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

MUX(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:=MUX_E(EN, n, s1, s2···s28, d);

indicates any of the following functions.

MUX MUX_E

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

Function

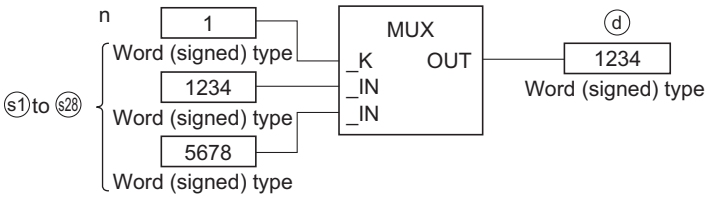
Operation processing

(1) 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.

(Example) Word (signed) type data



(2) 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.)

(3) 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').

(4) 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.

(5) Rounding error may occur when specifying single-precision real or double-precision real type data to ⓢ1 through ⓢ28 by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

(6) The number of pins of variable 's' can be changed in the range from 2 to 28.

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

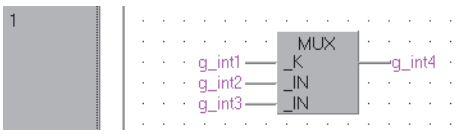
No operation error occurs in the execution of the MUX(_E) function.

Program Example

The program which selects the value to be output among the values input to variables ⓢ1 and ⓢ2 according to the value input to n, and outputs the operation result from ⓓ in the same data type as that of ⓢ1 or ⓢ2 .

(a) Function without EN/ENO (MUX)

[Structured ladder/FBD]

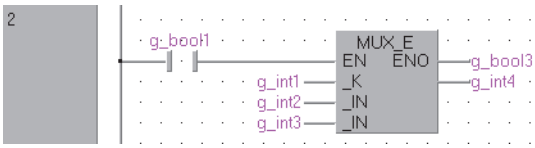


[ST]

```
g_int4 := MUX (g_int1, g_int2, g_int3);
```

(b) 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

5.6.1 Comparison

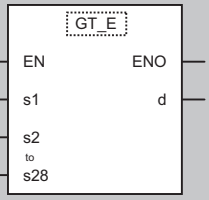
GT_E, GE_E, EQ_E, LE_E, LT_E, NE_E

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

GT_E
GE_E
EQ_E
LE_E
LT_E
NE_E

(_E: With EN/ENO)

Structured ladder/FBD



ST

Not supported

[ ] indicates any of the following functions.

GT_E
GE_E
EQ_E
LE_E
LT_E
NE_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1 to s28(_IN):	Input	:ANY_SIMPLE
	(s1 and s2 only for NE(_E))		
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output (TRUE: True value, FALSE: False value)	:Bit

Function

Operation processing

(1) 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.

(a) GT_E(>) Performs comparison of [ⓢ1 > ⓢ2]&[ⓢ2 > ⓢ3]&···&[ⓢ (n–1) > ⓢ (n)].

- Outputs TRUE if all of comparisons satisfy ⓢ (n–1) > ⓢ (n).
- Outputs FALSE if any of comparisons satisfies ⓢ (n–1) ≦ ⓢ (n).

(b) GE_E(≧) Performs comparison of [ⓢ1 ≧ ⓢ2]&[ⓢ2 ≧ ⓢ3]&···&[ⓢ (n–1) ≧ ⓢ (n)].

- Outputs TRUE if all of comparisons satisfy ⓢ (n–1) ≧ ⓢ (n).
- Outputs FALSE if any of comparisons satisfies ⓢ (n–1) < ⓢ (n).

(c) EQ_E(=) Performs comparison of [ⓢ1 = ⓢ2]&[ⓢ2 = ⓢ3]&···&[ⓢ (n–1) = ⓢ (n)].

- Outputs TRUE if all of comparisons satisfy ⓢ (n–1) = ⓢ (n).
- Outputs FALSE if any of comparisons satisfies ⓢ (n–1) ≠ ⓢ (n).

(d) LE_E(≦) Performs comparison of [ⓢ1 ≦ ⓢ2]&[ⓢ2 ≦ ⓢ3]&···&[ⓢ (n–1) ≦ ⓢ (n)].

- Outputs TRUE if all comparisons satisfy ⓢ (n–1) ≦ ⓢ (n).
- Outputs FALSE if any of comparisons satisfies ⓢ (n–1) > ⓢ (n).

(e) LT_E(<) Performs comparison of [ⓢ1 < ⓢ2]&[ⓢ2 < ⓢ3]&···&[ⓢ (n–1) < ⓢ (n)].

- Outputs TRUE if all comparisons satisfy ⓢ (n–1) < ⓢ (n).
- Outputs FALSE if any of comparisons satisfies ⓢ (n–1) ≧ ⓢ (n).

(f) NE_E(< >) Performs comparison of [ⓢ1 ≠ ⓢ2].

- Outputs TRUE if ⓢ1 ≠ ⓢ2 .
- Outputs FALSE if ⓢ1 = ⓢ2 .

(2) The values to be input to ⓢ 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.

(3) Rounding error may occur when specifying single-precision real or double-precision real type data to ⓢ1 through ⓢ28 by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

(4) The number of pins of ⓢ can be changed in the range from 2 to 28. (The number of pins of ⓢ for comparison operator NE(_E)) is fixed at ⓢ1 and ⓢ2 .

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

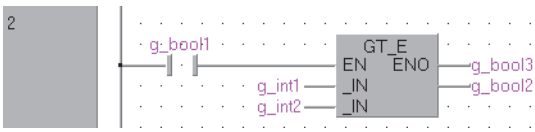
Operation Error

No operation error occurs in the execution of the GT_E, GE_E, EQ_E, LE_E, LT_E, and NE_E functions.

Program Example

The program which performs comparison operation between the values input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ .

[Structured ladder/FBD]



[ST]

```
g_bool3 := GT_E (g_bool1, g_int1, g_int2, g_bool2);
```

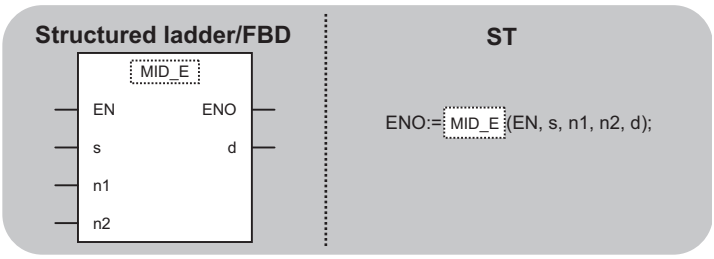
5.7 Standard Character String Functions

5.7.1 Extract mid string

MID(_E)

MID(_E)

_E: With EN/ENO



indicates any of the following functions.

MID MID_E

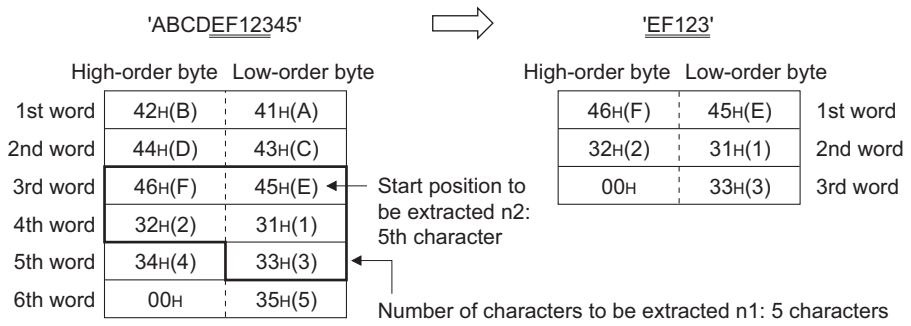
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_IN):	Input	:String (255)
	n1(_L):	Number of characters to be extracted	:Word (signed)
	n2(_P):	Start position to be extracted	:Word (signed)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String (255)

Function

Operation processing

(1) Extracts the specified number of characters from the specified start position in the character string input to ⓢ , and outputs the operation result from ⓓ .
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.

(Example) Values input to n1 and n2 are 5



(2) The value to be input to ⓢ is string type data within the range from 0 to 255 bytes.

(3) 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 ⓢ .)

(4) 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 ⓢ .)

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

These functions consist of the following instructions.
MID(_E): MIDR

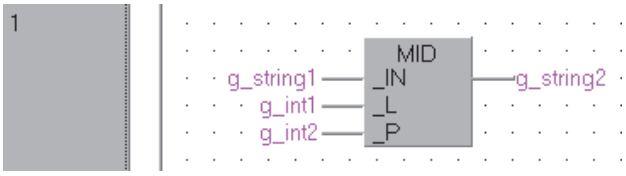
For details of an error which occurs when the function is executed, refer to MELSEC-Q/L Structured Programming Manual (Common Instructions).

Program Example

The program which extracts the specified number of characters from the specified start position in the character string input to ⓢ , and outputs the operation result from ⓓ .

(a) Function without EN/ENO (MID)

[Structured ladder/FBD]

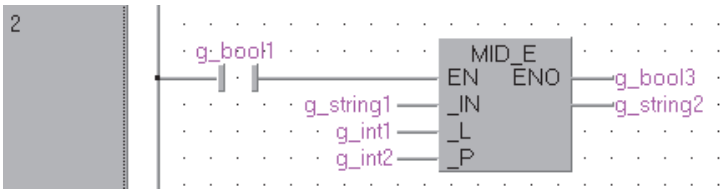


[ST]

```
g_string2:=MID(g_string1, g_int1, g_int2);
```

(b) Function with EN/ENO (MID_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := MID_E(g_bool1, g_string1, g_int1, g_int2, g_string2);
```

5.7.2 String concatenation

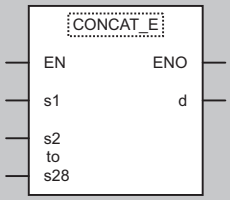
CONCAT(_E)

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

CONCAT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:= CONCAT_E (EN, s1, s2…n28, d);

indicates any of the following functions.

CONCAT CONCAT_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1 to s28(_IN):	Input	:String (255)
	s2:		
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String (255)

Function

Operation processing

(1) 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.

s1 'ABCDE' + s2 '123456' ⇨ d 'ABCDE123456'

	s1 High-order byte	s1 Low-order byte
1st word	42H(B)	41H(A)
2nd word	44H(D)	43H(C)
3rd word	00H	45H(E)

	s2 High-order byte	s2 Low-order byte
1st word	32H(2)	31H(1)
2nd word	34H(4)	33H(3)
3rd word	36H(6)	35H(5)
4th word	00H	

	d High-order byte	d 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)

(2) The values to be input to s1 and s2 to s28 are string type data within the range from 0 to 255 bytes.

(3) The number of pins for s can be changed in the range from 2 to 28.

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

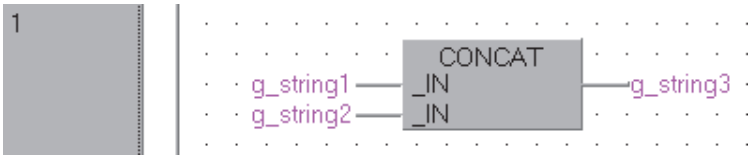
No operation error occurs in the execution of the CONCAT(_E) function.

Program Example

The program which concatenates the character string input to ⓢ2 following the one input to ⓢ1 , and outputs the operation result from ⓓ .

(a) Function without EN/ENO (CONCAT)

[Structured ladder/FBD]

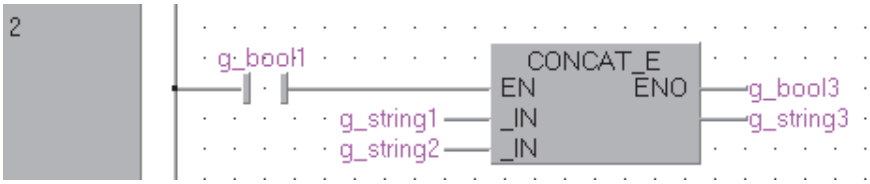


[ST]

```
g_string3:=CONCAT(g_string1, g_string2);
```

(b) Function with EN/ENO (CONCAT_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := CONCAT_E(g_bool1, g_string1, g_string2, g_string3);
```

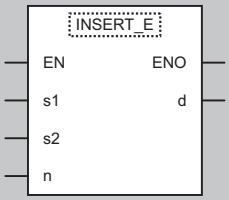
5.7.3 String insertion

INSERT(_E)

INSERT(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:= INSERT_E (EN, s1, s2, n, d);

[ ] indicates any of the following functions.

INSERT	INSERT_E

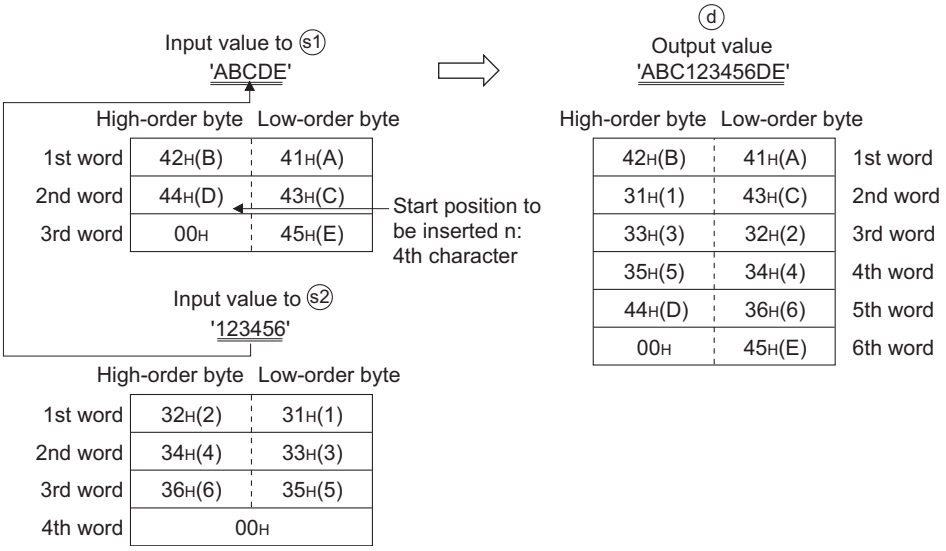
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1(_IN1): s2(_IN2):	Input	:String (255)
	n(_P):	Start position to be inserted	:Word (signed)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String (255)

Function

Operation processing

(1) Inserts the character string input to ⓢ2 to the specified position in the character string input to ⓢ1 , and outputs the operation result from ⓓ .
Specify the start position of the character string to be inserted by the value input to n.
After the insertion of character string ⓢ2 to character string ⓢ1 , '00H' that indicates the end of character string ⓢ2 is ignored. If the character string after insertion has over 255 bytes, the character string up to 255 bytes is output.

(Example) Value input to n is 4



(2) The values to be input to s1 and s2 are string type data within the range from 0 to 255 bytes.

(3) 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 s1 .)

Operation result

(1) Function without EN/ENO

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

(2) 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

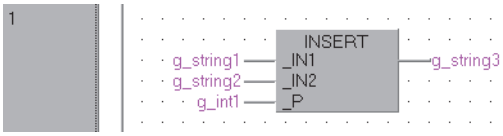
No operation error occurs in the execution of the INSERT(_E) function.

Program Example

The program which inserts the character string input to ⓢ2 to the specified position in the character string input to ⓢ1 , and outputs the operation result from ⓓ .

(a) Function without EN/ENO (INSERT)

[Structured ladder/FBD]

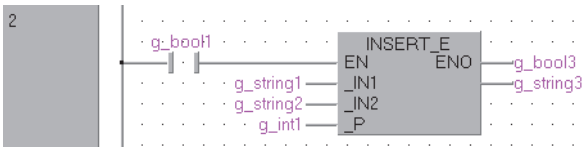


[ST]

```
g_string3:=INSERT(g_string1, g_string2, g_int1);
```

(b) Function with EN/ENO (INSERT_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := INSERT_E(g_bool1, g_string1, g_string2, g_int1, g_string3);
```

5.7.4 String deletion

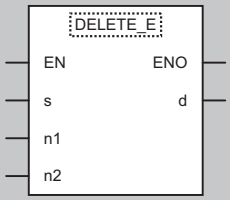
DELETE(_E)

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

DELETE(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

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

indicates any of the following functions.

DELETE DELETE_E

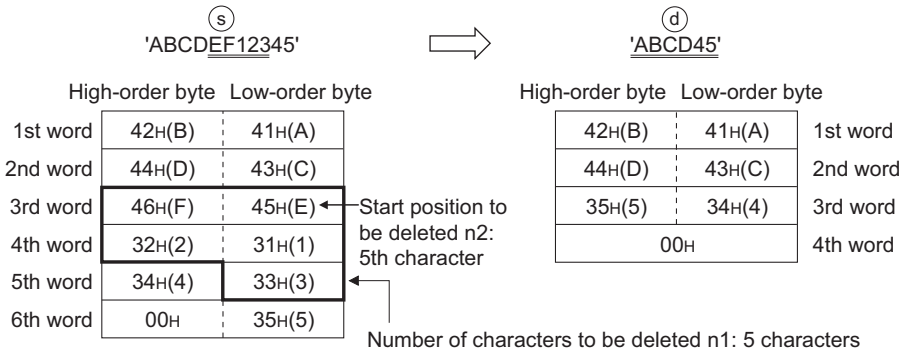
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s(_IN):	Input	:String (255)
	n1(_L):	Number of characters to be deleted	:Word (signed)
	n2(_P):	Start position to be deleted	:Word (signed)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String (255)

Function

Operation processing

(1) Deletes the specified number of characters from the specified position in the character string input to ⓢ , and outputs the remaining character string from ⓓ .
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.

(Example) Values input to n1 and n2 are 5



(2) The value to be input to ⓢ is string type data within the range from 0 to 255 bytes.

(3) 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 ⓢ .)

(4) 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 ⓢ .)

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

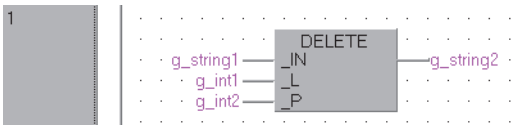
No operation error occurs in the execution of the DELETE(_E) function.

Program Example

The program which deletes the specified number of characters from the specified position in the character string input to ⓢ , and outputs the remaining character string from ⓓ .

(a) Function without EN/ENO (DELETE)

[Structured ladder/FBD]

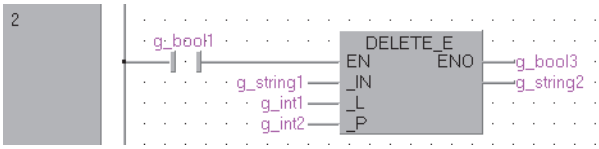


[ST]

```
g_string2:=DELETE(g_string1, g_int1, g_int2);
```

(b) Function with EN/ENO (DELETE_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := DELETE_E(g_bool1, g_string1, g_int1, g_int2, g_string2);
```

5.7.5 String replacement

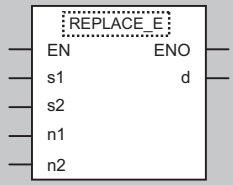
REPLACE(_E)

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

REPLACE(_E)

_E: With EN/ENO

Structured ladder/FBD



ST

```
ENO:= REPLACE_E (EN, s1, s2, n1, n2, d);
```

indicates any of the following functions.

REPLACE REPLACE_E

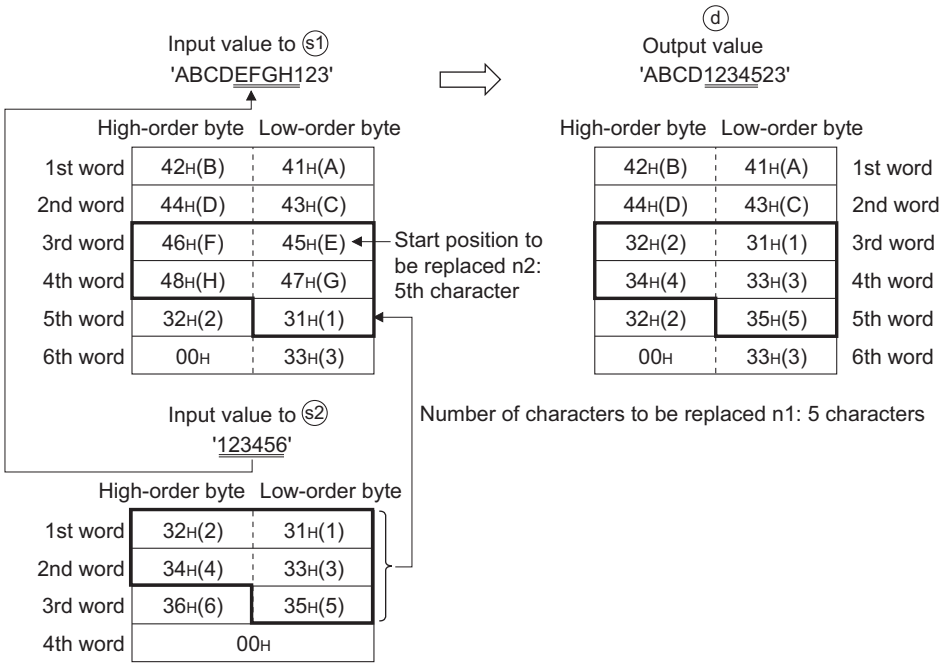
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1(_IN1): s2(_IN2):	Input	:String (255)
	n1(_L):	Number of characters to be replaced	:Word (signed)
	n2(_P):	Start position to be replaced	:Word (signed)
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String (255)

Function

Operation processing

(1) Replaces the specified number of characters from the specified position in the character string input to ⓢ1 with the character string input to ⓢ2 , and outputs the operation result from ⓓ .
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.

(Example) Values input to n1 and n2 are 5



(2) The values to be input to s1 and s2 are string type data within the range from 0 to 255 bytes.

(3) 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 .)

(4) 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 s1 .)

Operation result

(1) Function without EN/ENO
An operation is executed and the operation value is output from d .

(2) 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

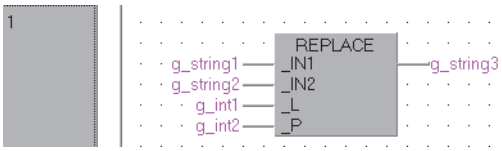
No operation error occurs in the execution of the REPLACE(_E) function.

Program Example

The program which replaces the specified number of characters from the specified position in the character string input to ⓢ1 with the character string input to ⓢ2 , and outputs the operation result from ⓓ .

(a) Function without EN/ENO (REPLACE)

[Structured ladder/FBD]

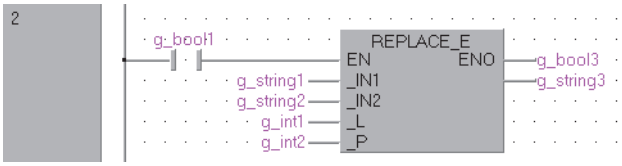


[ST]

```
g_string3:=REPLACE(g_string1, g_string2, g_int1, g_int2);
```

(b) 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

5.8.1 Addition

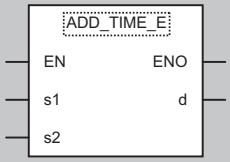
ADD_TIME(_E)

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

ADD_TIME(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:=ADD_TIME_E(EN, s1, s2, d);

[ ] indicates any of the following functions.

ADD_TIME ADD_TIME_E

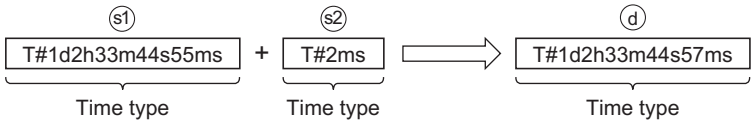
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1(_IN1): s2(_IN2):	Input	:Time
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Time

Function

Operation processing

(1) Performs addition (s1 + s2) on time type data input to s1 and s2, and outputs the operation result from d in time type.

(Example) 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).

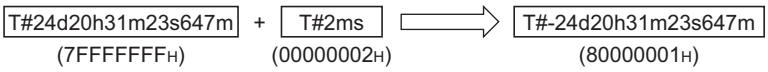


(2) The value to be input to s1, s2 are time type data.

(3) No operation error occurs even if an underflow/overflow occurs. Data is output from ⓓ as follows.

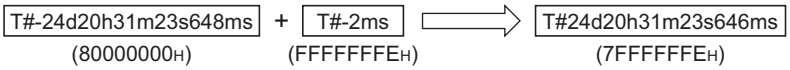
In case of ADD_TIME_E, TRUE is output from ENO.

(Example) Overflow



Since the highest-order bit is 1, the result value is negative.

(Example) Underflow



Since the highest-order bit is 0, the result value is positive.

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

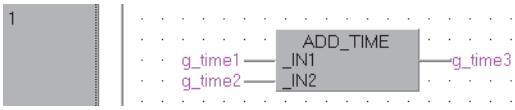
No operation error occurs in the execution of the ADD_TIME(_E) function.

Program Example

The program which performs addition (ⓢ1 +ⓢ2) on time type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in time type.

(a) Function without EN/ENO (ADD_TIME)

[Structured ladder/FBD]

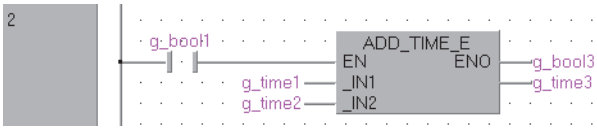


[ST]

```
g_time3:= ADD_TIME(g_time1, g_time2);
```

(b) Function with EN/ENO (ADD_TIME_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := ADD_TIME_E(g_bool1, g_time1, g_time2, g_time3);
```

5.8.2 Subtraction

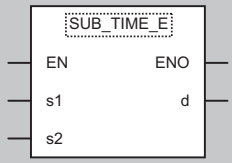
SUB_TIME(_E)

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

SUB_TIME(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:=SUB_TIME_E(EN, s1, s2, d);

indicates any of the following functions.

SUB_TIME SUB_TIME_E

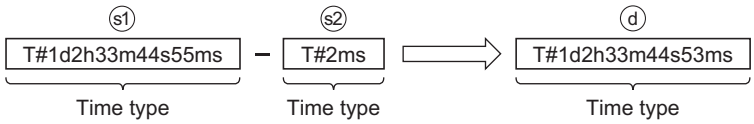
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1(_IN1): s2(_IN2):	Input	:Time
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Time

Function

Operation processing

(1) Performs subtraction (ⓢ1 −ⓢ2) on time type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in time type.

(Example) When the input value to ⓢ1 and ⓢ2 are T#1d2h33m44s55ms (1 day 2 hours 33 minutes 44 seconds 55 milliseconds) and T#2ms (2 milliseconds).

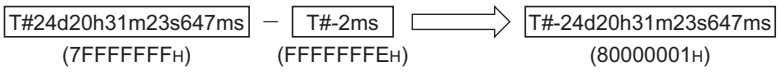


(2) The value to be input to ⓢ1 , ⓢ2 are time type data.

(3) No operation error occurs even if an underflow/overflow occurs. Data is output from ⓓ as follows.

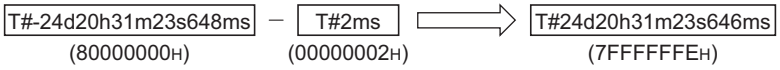
In case of SUB_TIME_E, TRUE is output from ENO.

(Example) Overflow



Since the highest-order bit is 1, the result value is negative.

(Example) Underflow



Since the highest-order bit is 0, the result value is positive.

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

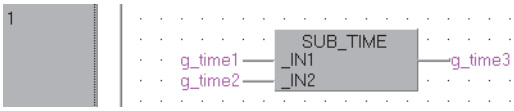
No operation error occurs in the execution of the SUB_TIME(_E) function.

Program Example

The program which performs subtraction (ⓢ1 −ⓢ2) on time type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in time type.

(a) Function without EN/ENO (SUB_TIME)

[Structured ladder/FBD]

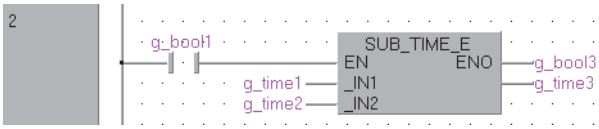


[ST]

```
g_time3:= SUB_TIME(g_time1, g_time2);
```

(b) Function with EN/ENO (SUB_TIME_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := SUB_TIME_E(g_bool1, g_time1, g_time2, g_time3);
```

5.8.3 Multiplication

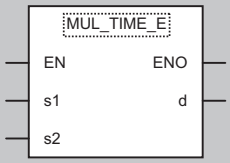
MUL_TIME(_E)

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

MUL_TIME(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:=MUL_TIME_E(EN, s1, s2, d);

indicates any of the following functions.

MUL_TIME MUL_TIME_E

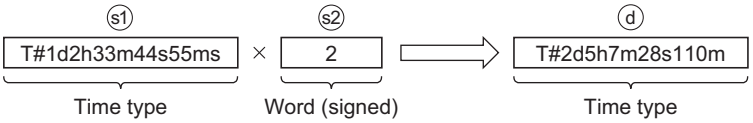
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1(_IN1):	Input	:Time
	s2(_IN2):	Input	:ANY_NUM
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Time

Function

Operation processing

(1) 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.

(Example) When the input value to s1 and s2 are T#1d2h33m44s55ms (1 day 2 hours 33 minutes 44 seconds 55 milliseconds) and 2.



(2) The value to be input to s1 is time type data.

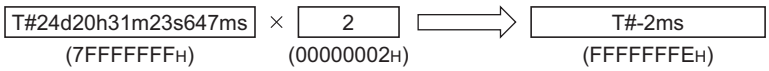
(3) The value to be input to s2 is word (signed), double word (signed), single-precision real or double-precision real type data.

(4) Rounding error may occur when specifying single-precision real or double-precision real type data to s2 by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

(5) No operation error occurs even if an underflow/overflow occurs. Data is output from ⓓ as follows.

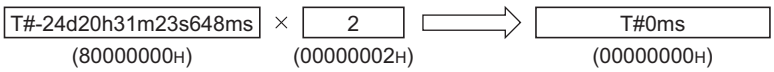
In case of MUL_TIME_E, TRUE is output from ENO.
(Although the operation result is 64-bit data, data is output in time type with the high-order 32 bits discarded.)

(Example) Overflow



Since the highest-order bit is 1, the result value is negative.

(Example) Underflow



Since the highest-order bit is 0, the result value is positive.

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

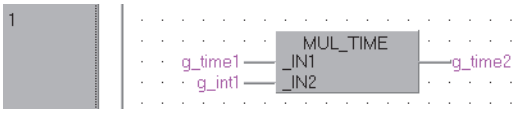
No operation error occurs in the execution of the MUL_TIME(_E) function.

Program Example

The program which performs multiplication (ⓢ1 × ⓢ2) on time type data input to ⓢ1 and the word (signed) type data input to ⓢ2 , and outputs the operation result from ⓓ in time type.

(a) Function without EN/ENO (MUL_TIME)

[Structured ladder/FBD]

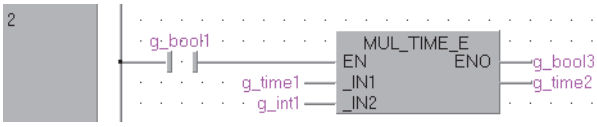


[ST]

```
g_time2:= MUL_TIME(g_time1, g_int1);
```

(b) Function with EN/ENO (MUL_TIME_E)

[Structured ladder/FBD]



[ST]

```
g_bool3 := MUL_TIME_E(g_bool1, g_time1, g_int1, g_time2);
```

5.8.4 Division

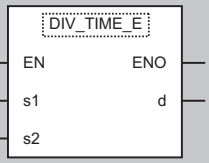
DIV_TIME(_E)

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

DIV_TIME(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:=DIV_TIME_E(EN, s1, s2, d);

indicates any of the following functions.

DIV_TIME DIV_TIME_E

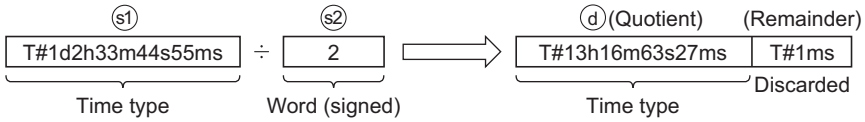
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1(_IN1):	Input	:Time
	s2(_IN2):	Input	:ANY_NUM
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Time

Function

Operation processing

(1) Performs division (ⓢ1 ÷ⓢ2) on time type data input to ⓢ1 and the word (signed), double word (signed), single-precision real or double-precision real type data input to ⓢ2 , and outputs the quotient of the operation result from ⓓ in time type. Remainder is rounded down.

(Example) When the input value to ⓢ1 and ⓢ2 are T#1d2h33m44s55ms (1 day 2 hours 33 minutes 44 seconds 55 milliseconds) and 2.



(2) The value to be input to ⓢ1 is time type data.

(3) The value to be input to ⓢ2 is word (signed), double word (signed), single-precision real or double-precision real type data.

(The value to be input to ⓢ2 must be other than 0.)

(4) Rounding error may occur when specifying single-precision real or double-precision real type data to ⓢ2 by programming tool.
For the precautions on setting the input value by the programming tool, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO
The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
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 ⓓ is undefined.
In this case, create a program so that the data output from ⓓ is not used.

Operation Error

An operation error occurs in the following cases.

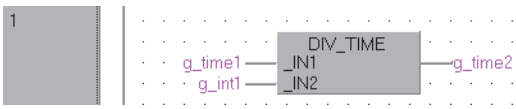
- The value input to ⓢ2 is 0. (Division by zero) (Error code: 4100)

Program Example

The program which performs division (ⓢ1 ÷ⓢ2) on time type data input to ⓢ1 and the word (signed) type data input to ⓢ2 , and outputs the quotient of the operation result from ⓓ in time type.

(a) Function without EN/ENO (DIV_TIME)

[Structured ladder/FBD]

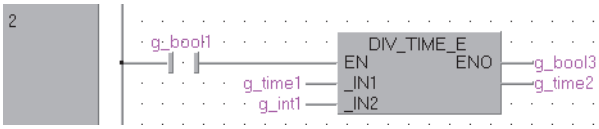


[ST]

```
g_time2:= DIV_TIME(g_time1, g_int1);
```

(b) 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

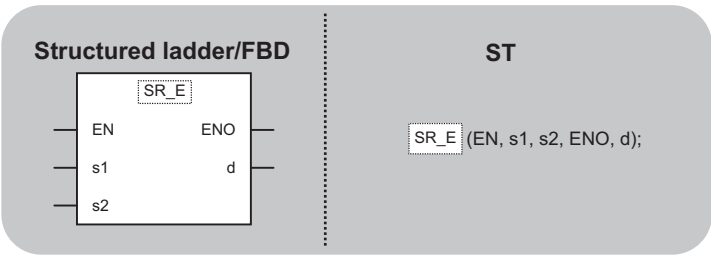
5.9.1 Standard bistable function blocks (Set-dominant)

SR(_E)

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

SR(_E)

(_E: With EN/ENO)



[ ] indicates any of the following functions.

SR SR_E

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

Function

Operation processing

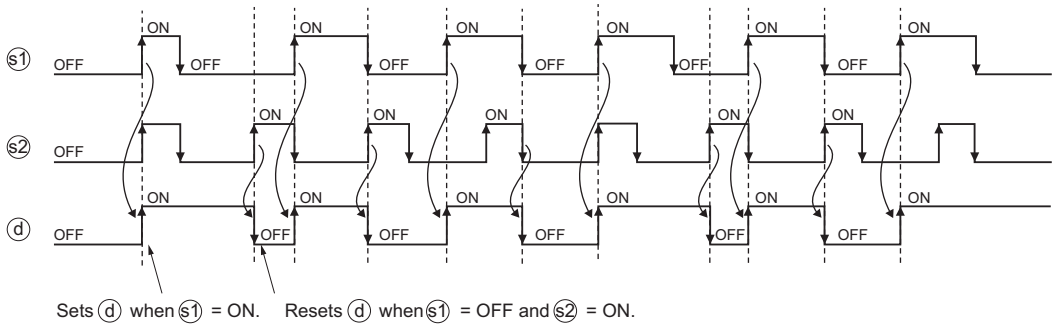
Sets ⓓ when ⓢ1 is turned ON, and resets ⓓ when ⓢ2 is turned ON while ⓢ1 is OFF.

ⓓ is not reset even when ⓢ2 is turned ON while ⓢ1 is ON.

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ .
[Timing chart]

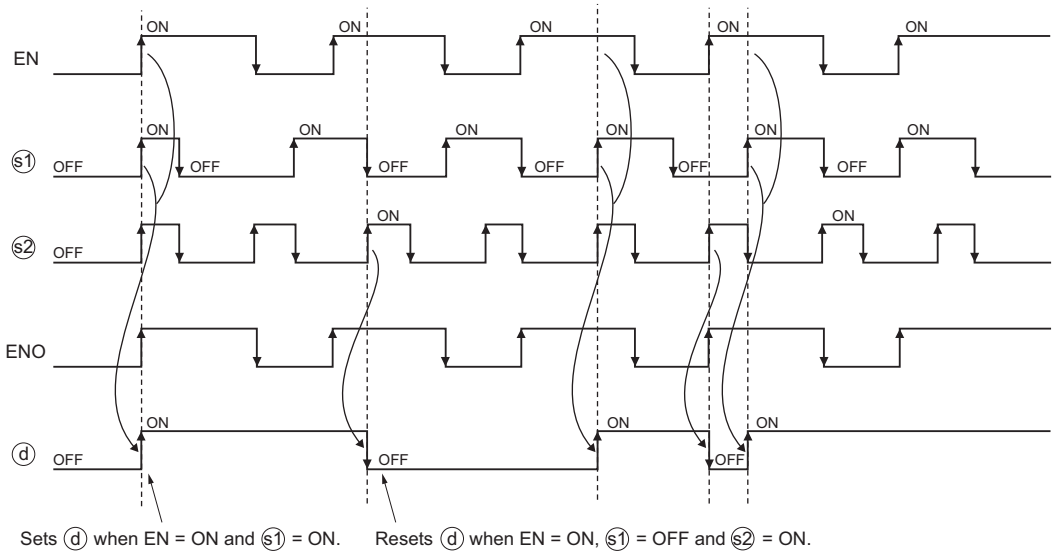


(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]



Operation Error

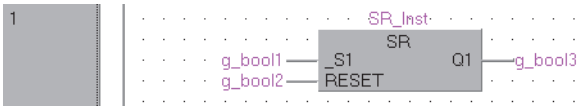
No operation error occurs in the execution of the SR (_E) function.

Program Example

The program which outputs bit type data input to ⓢ1 from ⓓ and holds the output, and resets the value of ⓓ only when bit type data input to ⓢ2 is 1 and the data input to ⓢ1 is 0.

(a) Function without EN/ENO (SR)

[Structured ladder/FBD]

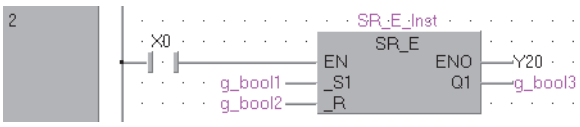


[ST]

```
SR_Inst(_S1:= g_bool1 ,RESET:= g_bool2 ,Q1:= g_bool3 );
```

(b) 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 );
```

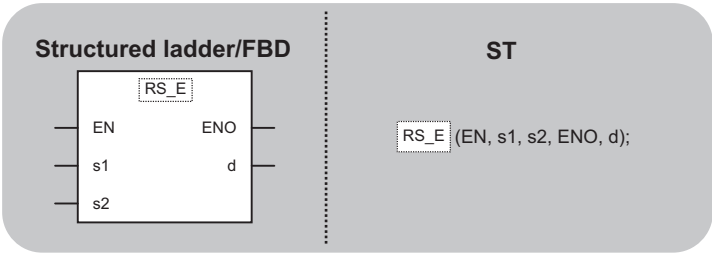
5.9.2 Standard bistable function blocks (Reset-dominant)

RS(_E)

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

RS(_E)

(_E: With EN/ENO)



[ ] indicates any of the following functions.

RS RS_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1(_S): s2(_R1):	Input	:Bit
Output argument,	ENO:	Execution result (TRUE: Normal, FALSE: Error or stop)	:Bit
	d(Q1):	Output	:Bit

Function

Operation processing

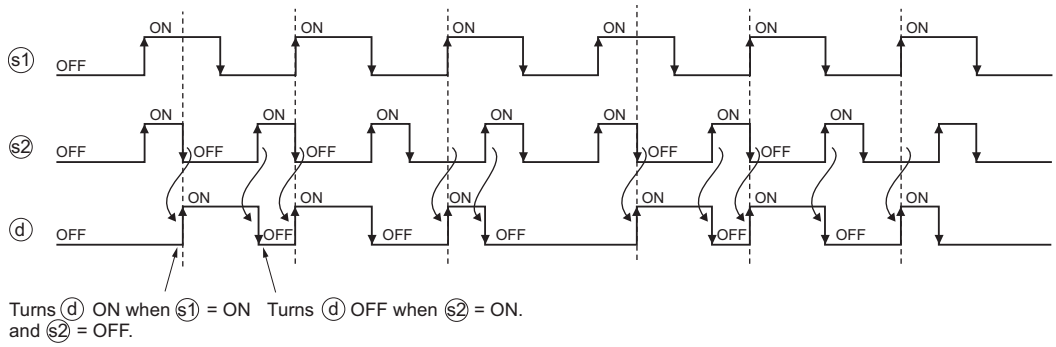
Sets ⓓ when ⓢ1 is turned ON, and resets ⓓ when ⓢ2 is turned ON.

ⓓ is not set even when ⓢ1 is turned ON while ⓢ2 is ON.

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ .
[Timing chart]

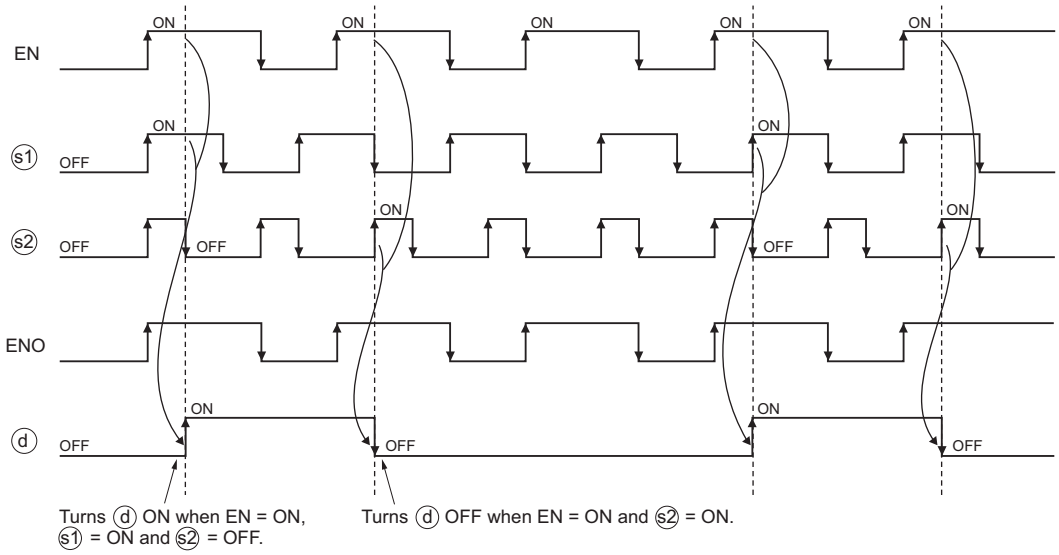


(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]



Operation Error

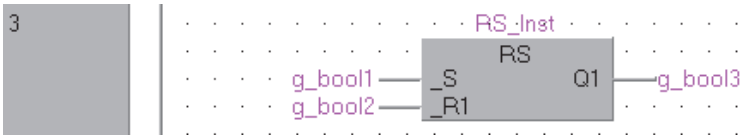
No operation error occurs in the execution of the RS (_E) function.

Program Example

The program which outputs bit type data input to ⓢ1 from ⓓ and holds the output, and resets forcibly the value of ⓓ when bit type data input to ⓢ2 is 1.

(a) Function without EN/ENO (RS)

[Structured ladder/FBD]

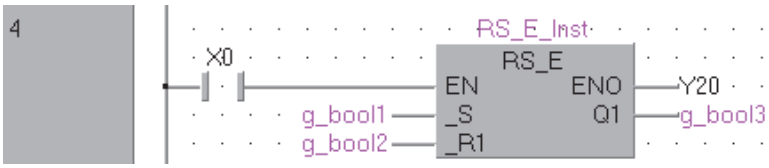


[ST]

```
RS_Inst(_S:= g_bool1 ,_R1:= g_bool2 ,Q1:= g_bool3 );
```

(b) 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

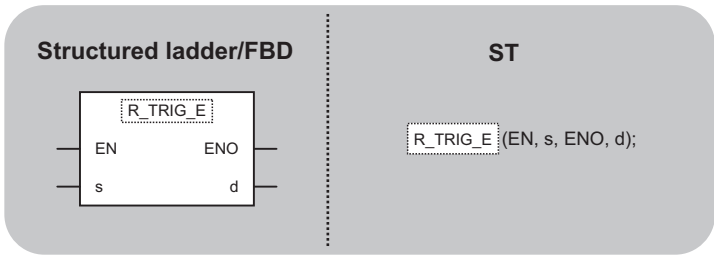
5.10.1 Rising edge detector

R_TRIG(_E)

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

R_TRIG(_E)

(_E: With EN/ENO)



[ ] indicates any of the following functions.

R_TRIG R_TRIG_E

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

Function

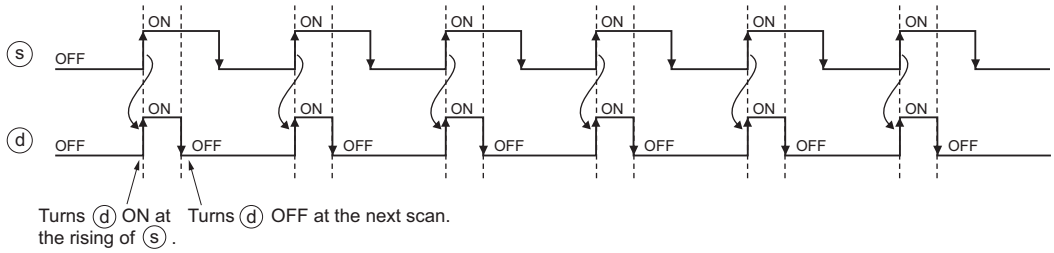
Operation processing

Turns ON ⓓ for one scan when ⓢ is turned ON.

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ .
[Timing chart]

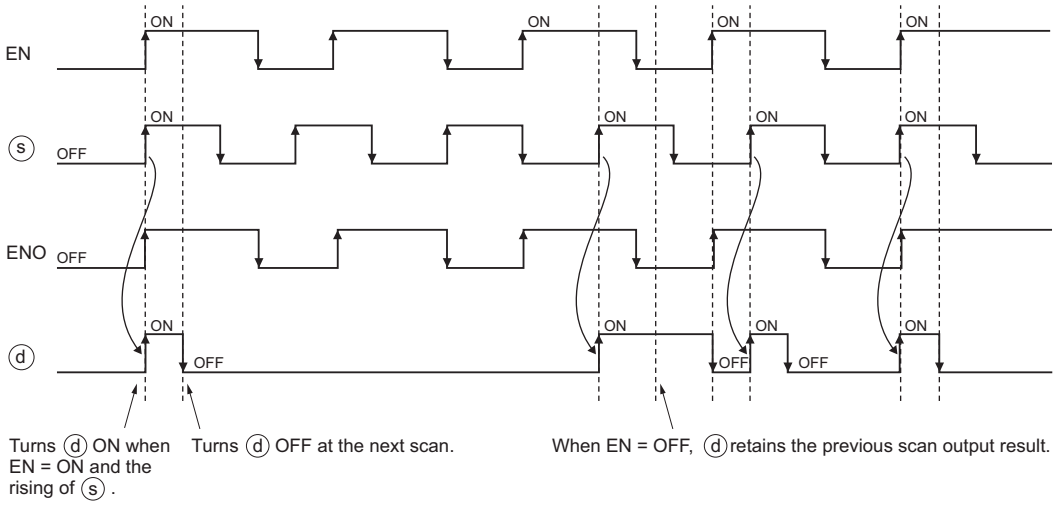


(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]



Operation Error

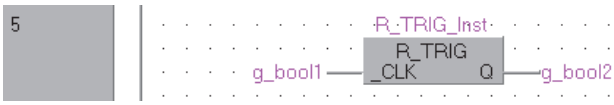
No operation error occurs in the execution of the R_TRIG (_E) function.

Program Example

The program which turns ON ⓓ for one scan when bit type data input to ⓢ is turned from OFF to ON.

(a) Function without EN/ENO (R_TRIG)

[Structured ladder/FBD]

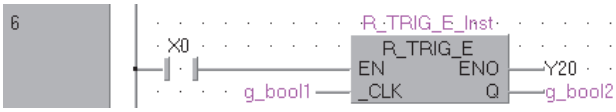


[ST]

```
R_TRIG_Inst(_CLK:= g_bool1 ,Q:= g_bool2 );
```

(b) 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 );
```

5.10.2 Falling edge detector

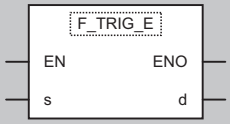
F_TRIG(_E)

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

F_TRIG(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

F_TRIG_E (EN, s, ENO, d);

[ ] indicates any of the following functions.

F_TRIG F_TRIG_E

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

Function

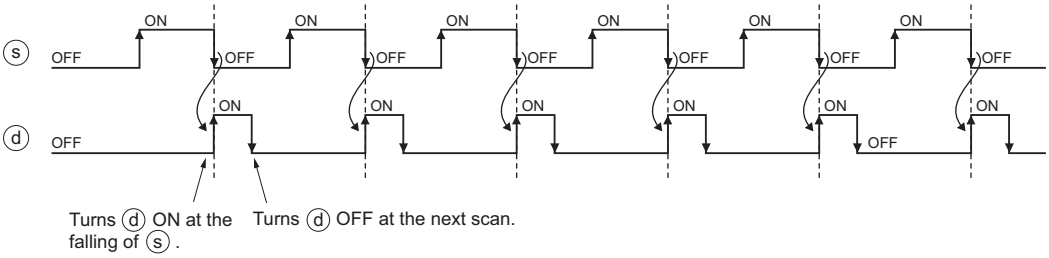
Operation processing

Turns ON ⓓ for one scan when ⓢ is turned OFF.

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ.
[Timing chart]

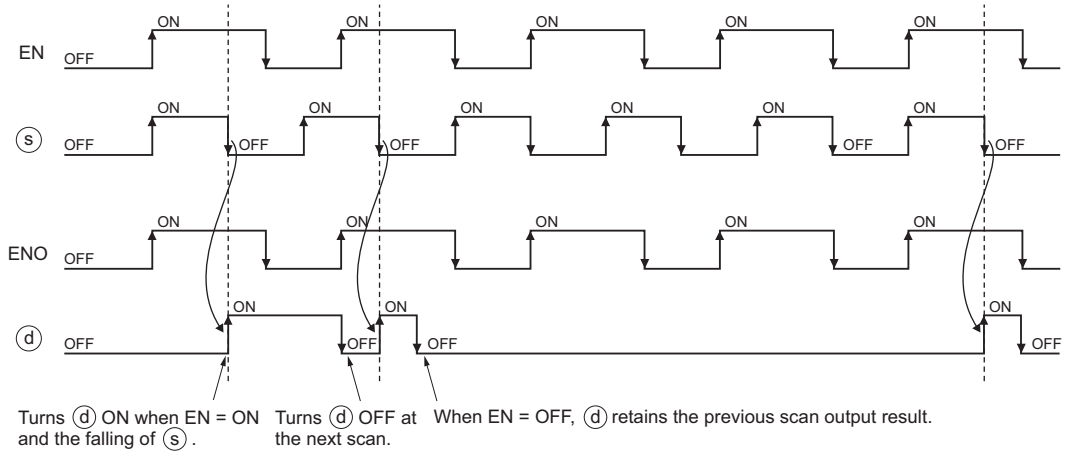


(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]



Operation Error

No operation error occurs in the execution of the F_TRIG (_E) function.

Program Example

The program which turns ON ⓓ for one scan when bit type data input to ⓢ is turned from ON to OFF.

(a) Function without EN/ENO (F_TRIG)

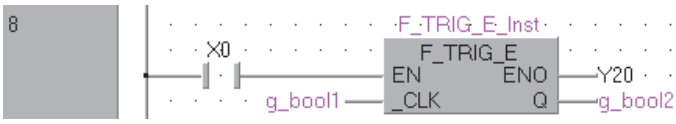
[Structured ladder/FBD]

[ST]

```
F_TRIG_Inst(_CLK:= g_bool1 ,Q:= g_bool2 );
```

(b) 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

5.11.1 Up counter

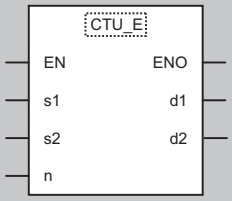
CTU(_E)

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

CTU(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:= CTU_E (EN, s1, s2, n, d1, d2);

[ ] indicates any of the following functions.

CTU CTU_E

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)

Function

Operation processing

(1) Counts ⓓ2 when ⓢ1 is turned ON.

When the count value ⓓ2 reaches the value input to n, ⓓ1 turns ON.

When ⓢ2 is turned ON, ⓓ1 turns OFF and count value ⓓ2 is reset.

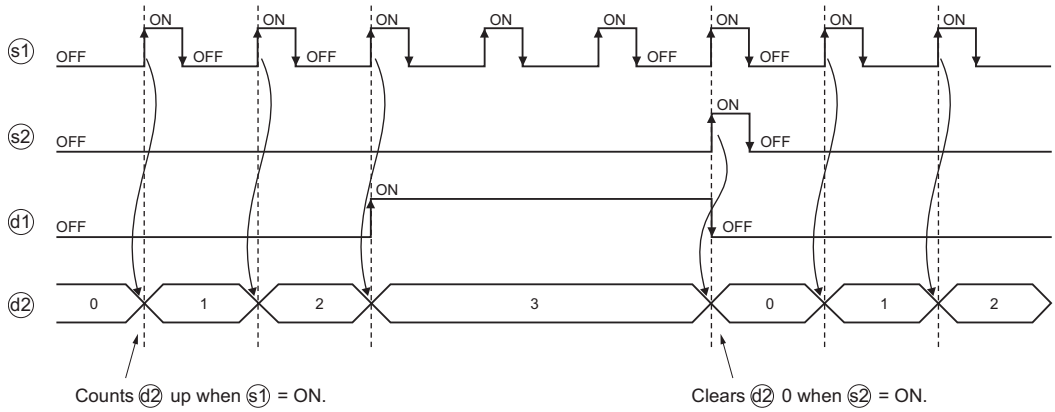
(2) Valid setting range for n is -32768 to 32767.

However, if 0 or less is set, ⓓ1 is turned on regardless of the count value of ⓓ2 .

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ1 and ⓓ2 .
[Timing chart]

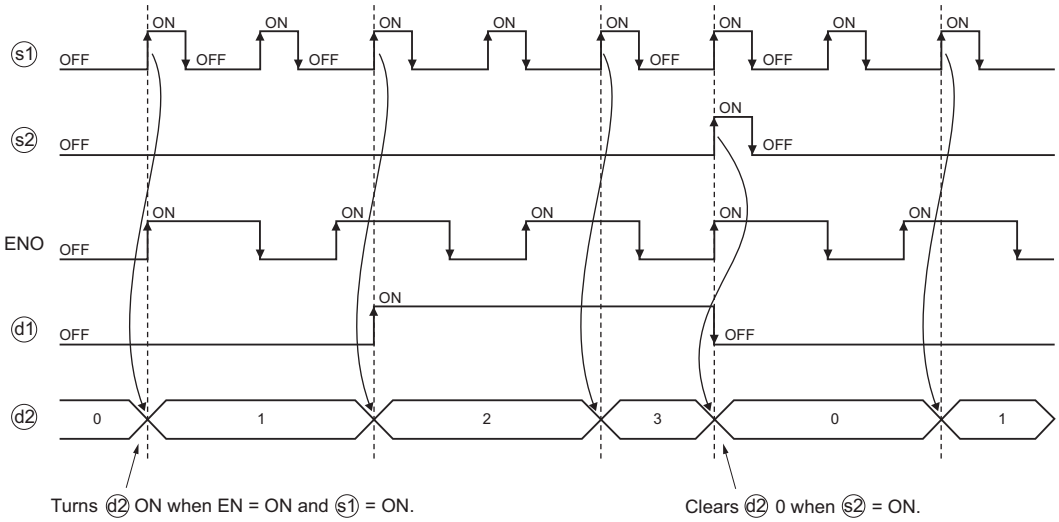


(2) Function with EN/ENO

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]



Operation Error

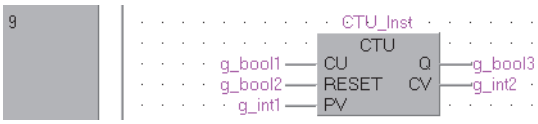
No operation error occurs in the execution of the CTU (_E) function.

Program Example

The program which counts the number of times that bit type data input to ⓢ1 is turned from OFF to ON, and outputs the count value from ⓓ2 .

(a) Function without EN/ENO (CTU)

[Structured ladder/FBD]

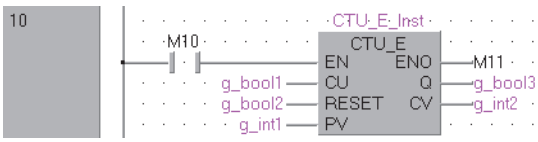


[ST]

```
CTU_Inst(CU:= g_bool1 ,RESET:= g_bool2 ,PV:= g_int1 ,Q:= g_bool3 ,CV:= g_int2 );
```

(b) 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 );
```

5.11.2 Down counter

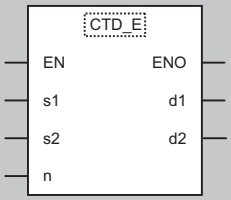
CTD(_E)

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

CTD(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

```
CTD_E (EN, s1, s2, n, ENO, d1, d2);
```

[ ] indicates any of the following functions.

CTD CTD_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1(CU):	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)

Function

Operation processing

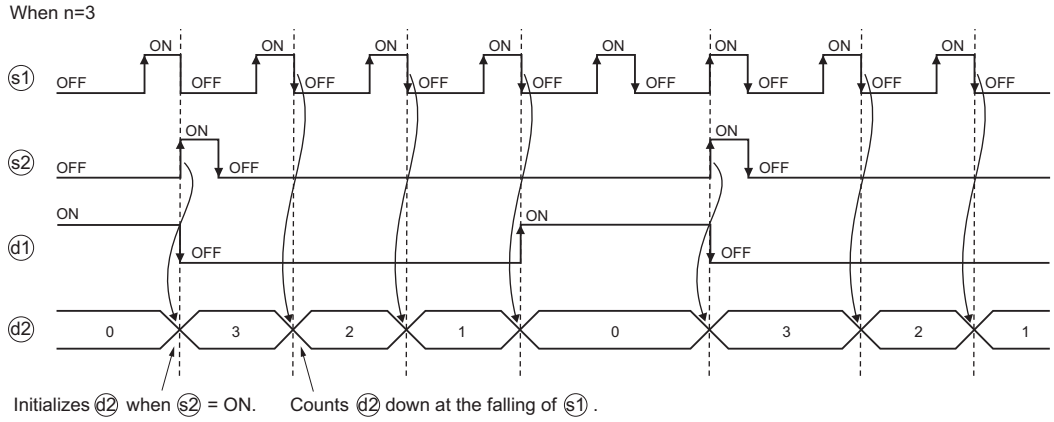
(1) Counts down (-1) ⓓ2 when ⓢ1 is turned ON.
n sets the initial value for subtraction.
ⓓ1 turns ON when count value ⓓ2 reaches 0.
When ⓢ2 is turned ON, ⓓ1 turns OFF and initial value for subtraction n is set for count value ⓓ2.

(2) Valid setting range for n is -32768 to 32767.
However, if 0 or less is set, ⓓ1 is turned on regardless of the count value of ⓓ2.

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ1 and ⓓ2 .
[Timing chart]

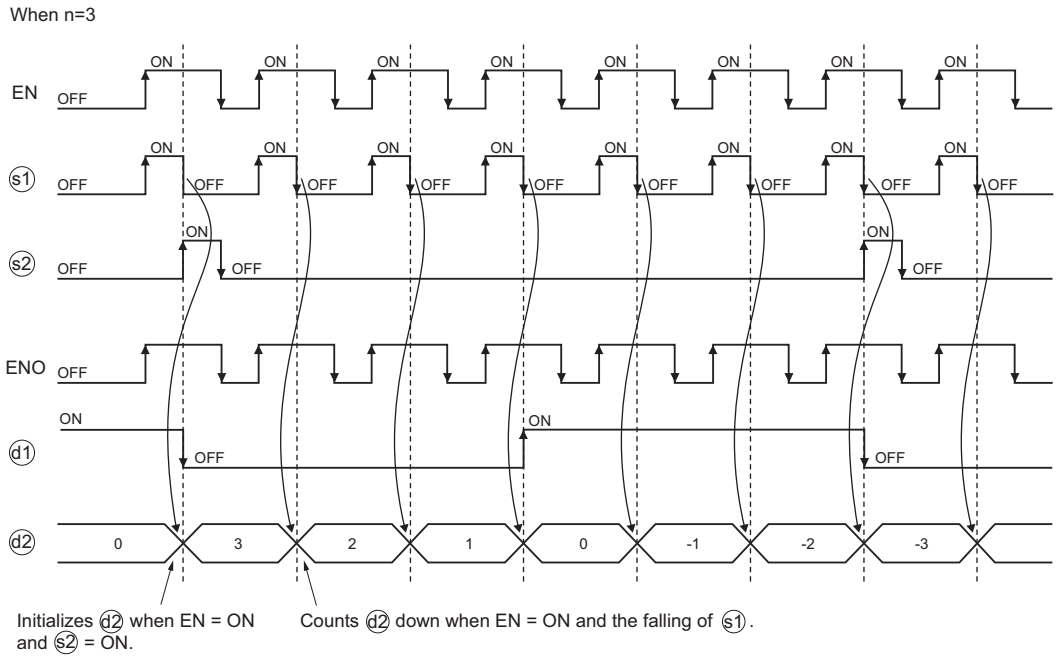


(2) Function with EN/ENO

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]



Operation Error

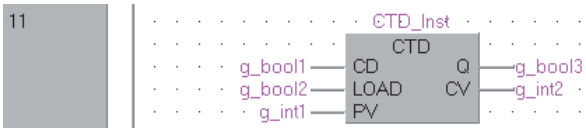
No operation error occurs in the execution of the CTD (_E) function.

Program Example

The program which counts the number of times that bit type data input to ⓢ1 is turned from OFF to ON, and turns ON ⓓ1 when the value of ⓓ2 reaches 0.

(a) Function without EN/ENO (CTD)

[Structured ladder/FBD]

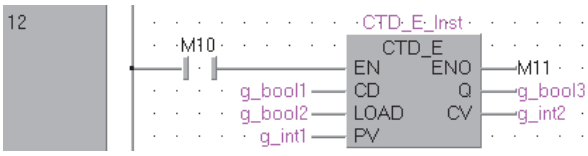


[ST]

```
CTD_Inst(CD:= g_bool1 ,LOAD:= g_bool2 ,PV:= g_int1 ,Q:= g_bool3, CV:= g_int2 );
```

(b) 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 );
```

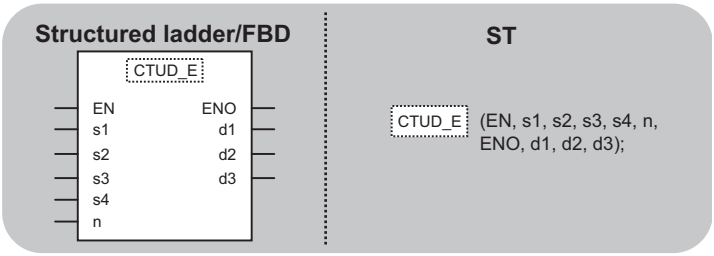
5.11.3 Up/Down counter

CTUD(_E)

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

CTUD(_E)

(_E: With EN/ENO)



indicates any of the following functions.

CTUD	CTUD_E

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)

Function

Operation processing

(1) 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.

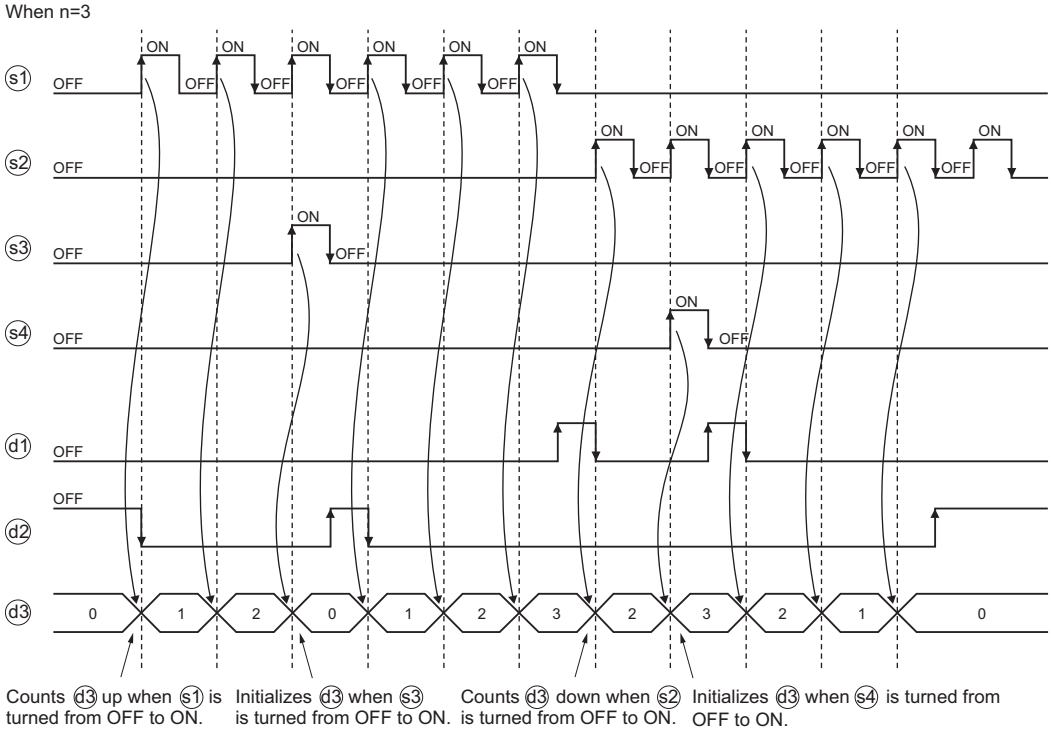
(2) Valid setting range for n is -32768 to 32767.

However, if 0 or less is set, d1 , d2 are turned on regardless of the count value of d3 .

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ1 , ⓓ2 , and ⓓ3 .
[Timing chart]

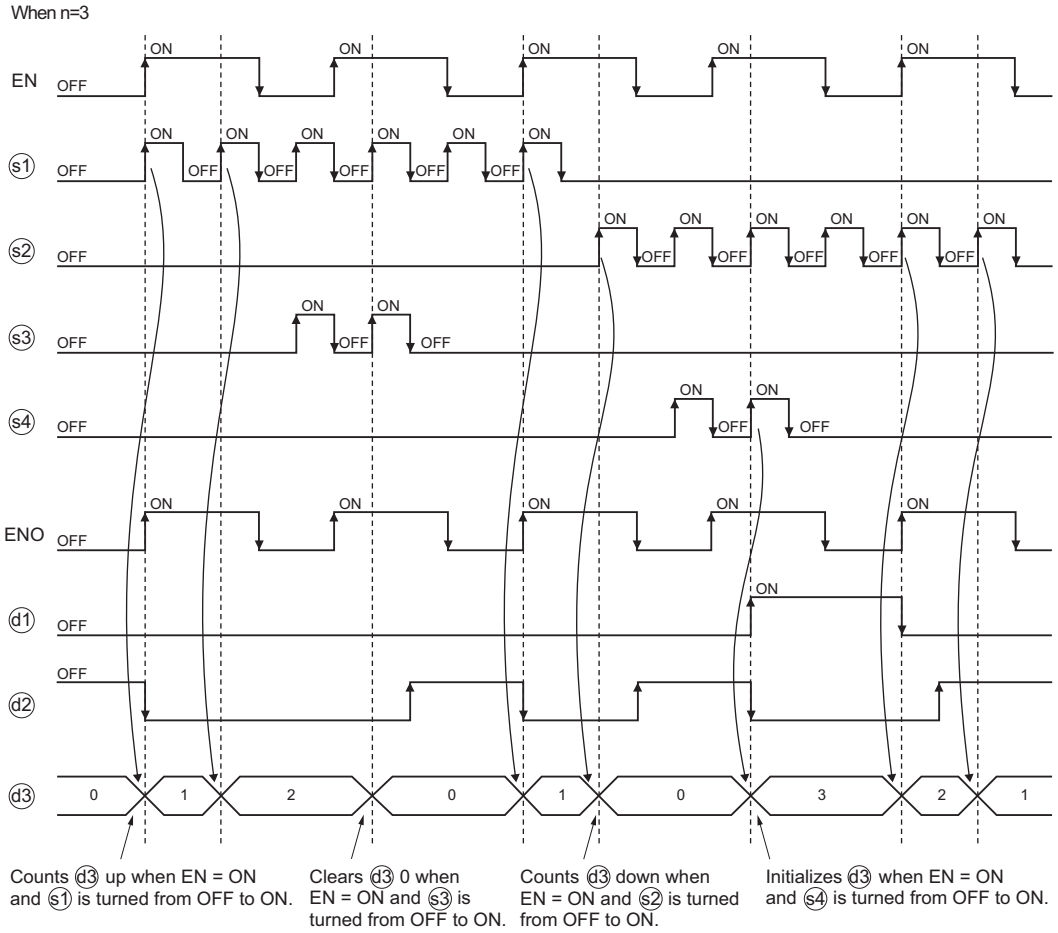


(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ1, ⓓ2, ⓓ3
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value

[Timing chart]



Operation Error

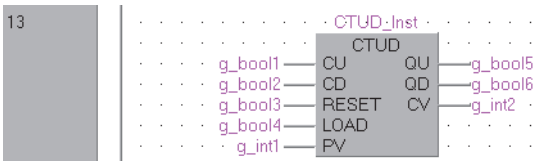
No operation error occurs in the execution of the CTUD (_E) function.

Program Example

The program which counts the number of times that bit type data input to ⓢ1 is turned from OFF to ON, and turns ON ⓓ1 when the value of ⓓ3 reaches the value set at ⓝ . Simultaneously, it counts the number of times that bit type data input to ⓢ2 is turned from OFF to ON, and turns ON ⓓ2 when the value of ⓓ3 reaches 0.

(a) Function without EN/ENO (CTD)

[Structured ladder/FBD]

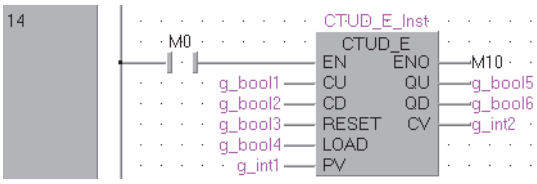


[ST]

```
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 );
```

(b) Function with EN/ENO (CTD_E)

[Structured ladder/FBD]



[ST]

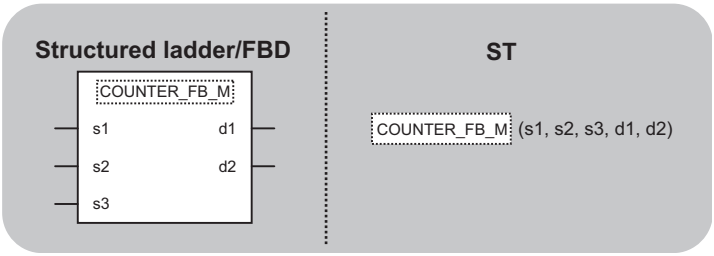
```
CTUD_E_Inst(EN:= M0 ,CU:= g_bool1 ,CD:= g_bool2 ,RESET:= g_bool3,
LOAD:= g_bool4 ,PV:= g_int1 ,QU:= g_bool5 ,QD:= g_bool6 ,CV:= g_int2,
ENO:= M10 );
```

5.11.4 Counter function blocks

COUNTER_FB_M

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

COUNTER_FB_M



indicates the following function.

COUNTER_FB_M

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

Function

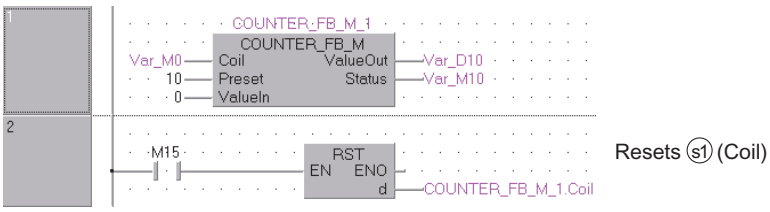
Operation processing

(1) 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 .

(2) Valid setting range for s2 is 0 to 32767.

(3) Valid setting range for s3 is -32768 to 32767. however, if negative value is specified, the initial value is 0.

(4) When resetting the current value of the counter, reset ⓢ1 .
(Example) When instance name is COUNTER_FB_M_1.
[Structured ladder/FBD]



[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);

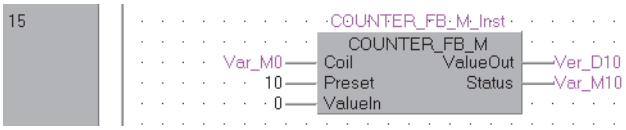
Operation Error

No operation error occurs in the execution of the counter function blocks.

Program Example

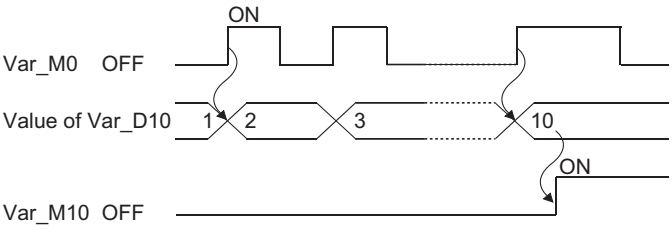
The program which counts the number of times that bit type data input to ⓢ1 is turned from OFF to ON, and outputs the count value from ⓓ1 .

[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

5.12.1 Pulse timer

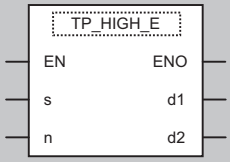
TP(_E), TP_HIGH(_E)

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

TP(_E)
TP_HIGH(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:= TP_HIGH_E (EN, s, n, d1, d2);

[ ] indicates any of the following functions.

TP	TP_E
TP_HIGH	TP_HIGH_E

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

Function

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.

(1) 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.

(2) 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

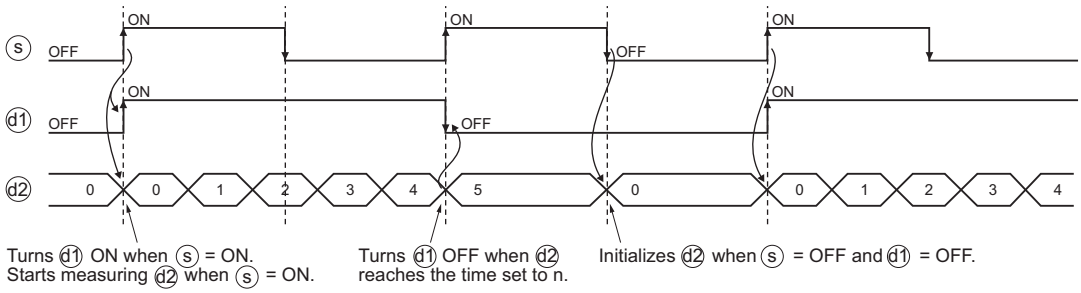
Valid setting range for n is T#0ms to T#327670ms.

Operation result

(1) 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)



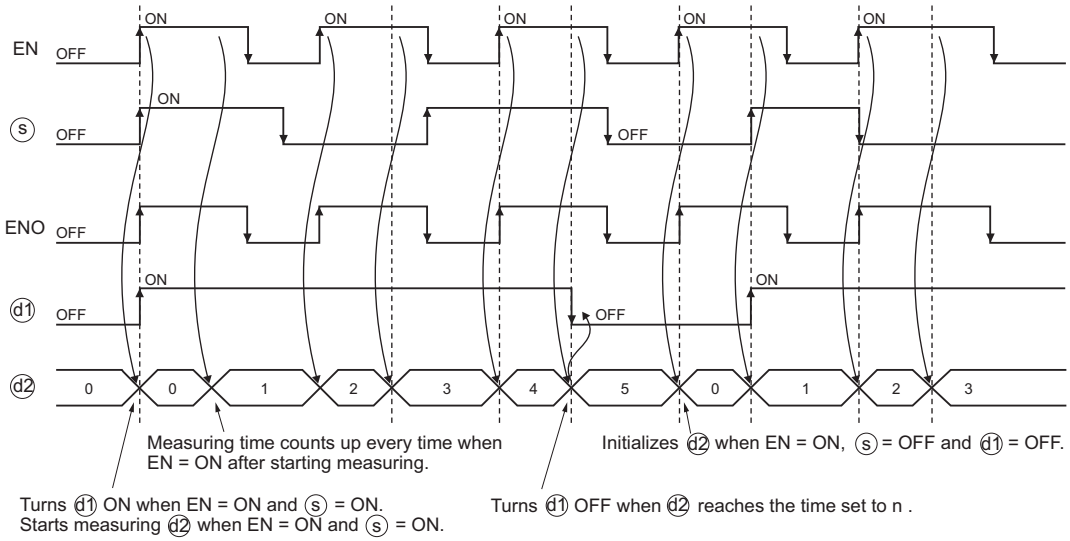
(2) Function with EN/ENO

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

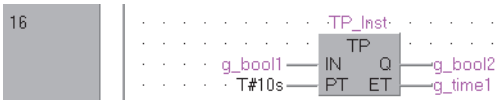
No operation error occurs in the execution of the TP (_E) function.

Program Example

The program which turns ON bit type data of d1 for 10 seconds after bit type data input to s is turned ON.

(a) Function without EN/ENO (TP)

[Structured ladder/FBD]

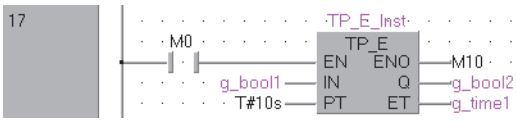


[ST]

```
TP_Inst(IN:= g_bool1 ,PT:= T#10s ,Q:= g_bool2 ,ET:= g_time1 );
```

(b) 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 );
```

5.12.2 On delay timer

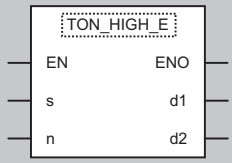
TON(_E), TON_HIGH(_E)

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

TON(_E)
TON_HIGH(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:= TON_HIGH_E (EN, s, n, d1, d2);

indicates any of the following functions.

TON	TON_E
TON_HIGH	TON_HIGH_E

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

Function

Operation processing

Turns ON ⓓ1 when ⓢ is turned ON after the elapse of the time set to n. Elapsed delay time until ⓓ1 is turned ON is set to ⓓ2 .

When ⓢ is turned OFF, ⓓ1 turns OFF and the elapsed delay time is reset.

(1) 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.

(2) 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.

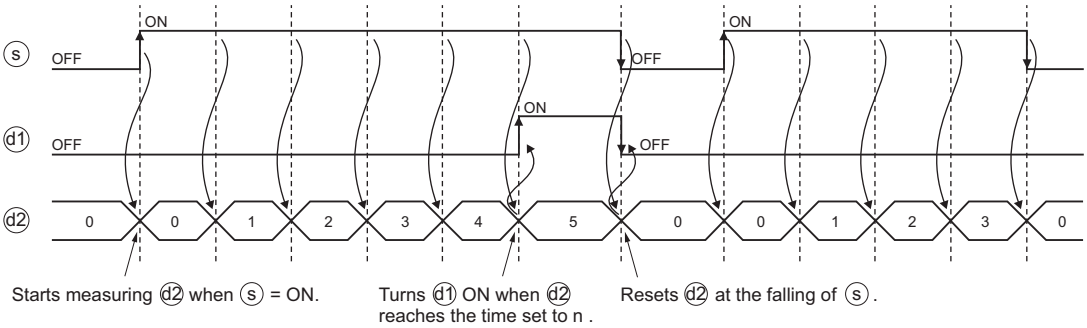
Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ1 and ⓓ2 .

[Timing chart]

When n = T#5s (5 seconds)



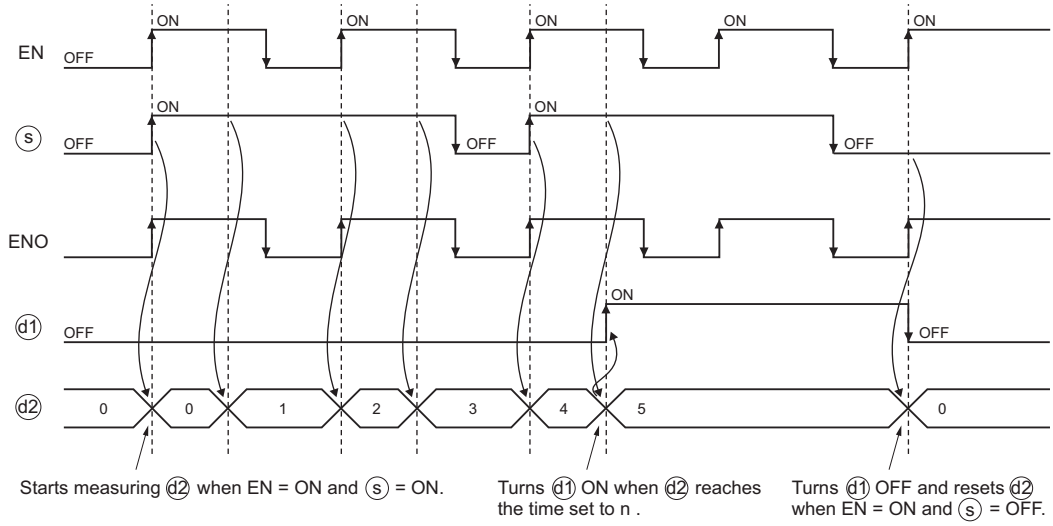
(2) Function with EN/ENO

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

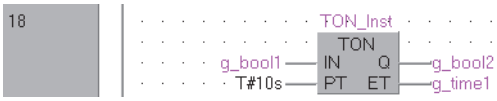
No operation error occurs in the execution of the TON (_E) function.

Program Example

The program which turns ON bit type data of ⓓ1 10 seconds after bit type data input to ⓢ is turned ON.

(a) Function without EN/ENO (TON)

[Structured ladder/FBD]

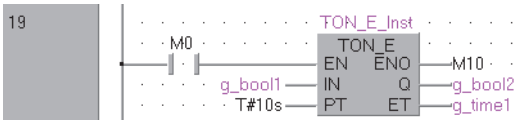


[ST]

```
TON_Inst(IN:= g_bool1 ,PT:= T#10s ,Q:= g_bool2 ,ET:= g_time1 );
```

(b) 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 );
```

5.12.3 Off delay timer

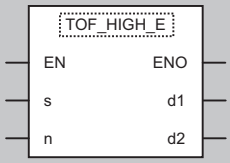
TOF(_E), TOF_HIGH(_E)

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

TOF(_E)
TOF_HIGH(_E)

(_E: With EN/ENO)

Structured ladder/FBD



ST

ENO:= TOF_HIGH_E (EN, s, n, d1, d2);

[ ] indicates any of the following functions.

TOF	TOF_E
TOF_HIGH	TOF_HIGH_E

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

Function

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.

(1) 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.

(2) 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

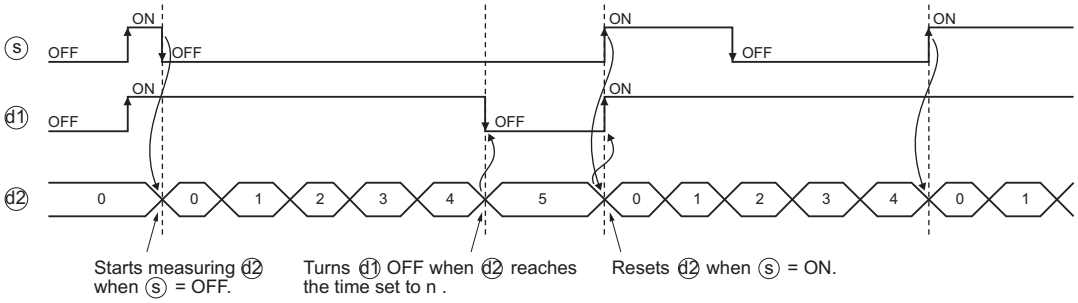
Valid setting range for n is T#0ms to T#327670ms.

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ1 and ⓓ2 .
[Timing chart]

When n = T#5s (5 seconds)



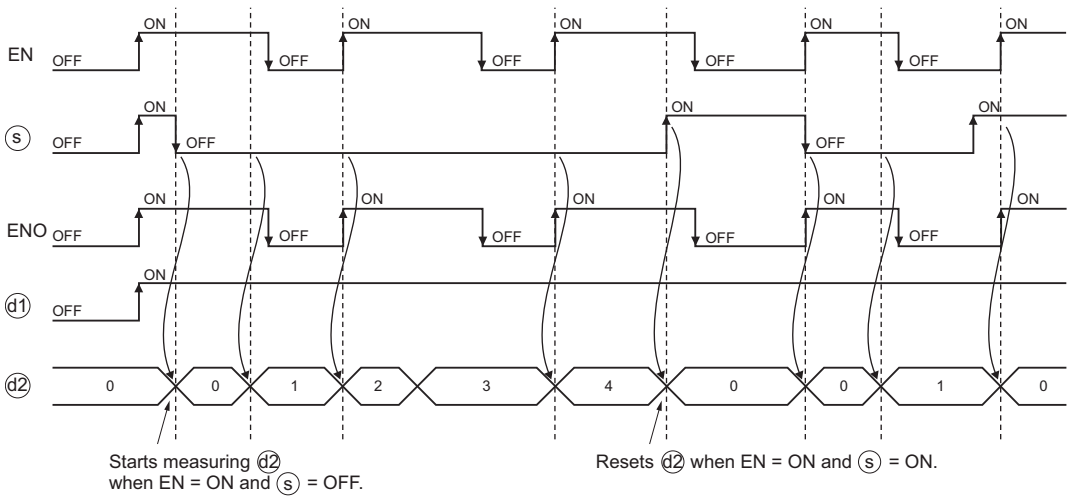
(2) Function with EN/ENO

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

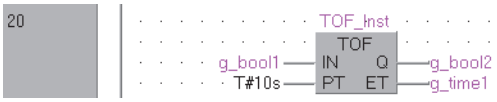
No operation error occurs in the execution of the TOF(_E) function.

Program Example

The program which turns ON bit type data of ⓓ when bit type data input to ⓢ is turned ON, and turns ⓓ OFF 10 seconds after ⓢ is turned OFF.

(a) Function without EN/ENO (TOF)

[Structured ladder/FBD]

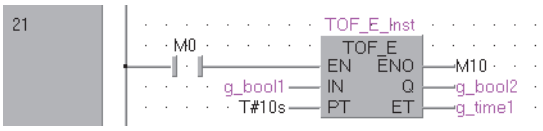


[ST]

```
TOF_Inst(IN:= g_bool1 ,PT:= T#10s ,Q:= g_bool2 ,ET:= g_time1 );
```

(b) Function with EN/ENO (TOF_E)

[Structured ladder/FBD]



[ST]

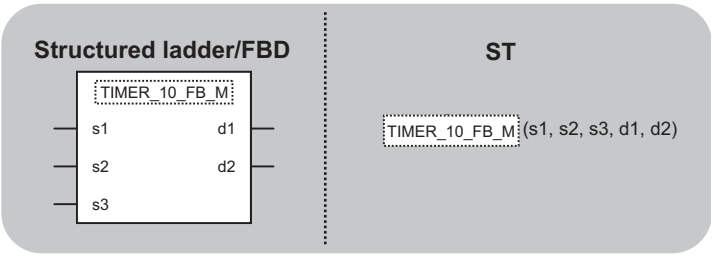
```
TOF_E_Inst(EN:= M0 ,IN:= g_bool1 ,PT:= T#10s ,Q:= g_bool2 ,ET:= g_time1,
ENO:= M10 );
```

5.12.4 Timer function blocks

TIMER_10_FB_M

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

TIMER_10_FB_M
TIMER_100_FB_M
TIMER_HIGH_FB_M
TIMER_LOW_FB_M
TIMER_CONT_FB_M
TIMER_CONTHFB_M



indicates any of the following functions.

TIMER_10_FB_M	TIMER_100_FB_M
TIMER_HIGH_FB_M	TIMER_LOW_FB_M
TIMER_CONT_FB_M	TIMER_CONTHFB_M

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

Function

Operation processing

(1) TIMER_10_FB_M

(a) 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 .

(b) When the executing condition of s1 turns OFF, the current value is set to the value input to s3 , and d2 turns OFF.

(c) 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.

(d) Valid setting range for s2 is 0 to 32767.

(e) Valid setting range for s3 is -32768 to 32767. However, if negative value is specified, the initial value is 0.

(2) TIMER_100_FB_M

(a) Starts measuring the current value when the executing condition of ⓢ1 turns ON.

Starts measuring from the value input to ⓢ3 × 100ms, and when the measuring value reaches to the value input to ⓢ2 × 100ms, ⓓ2 turns ON.

The current value is output from ⓓ1 .

(b) When the executing condition of ⓢ1 turns OFF, the current value is set to the value input to ⓢ3 , and ⓓ2 turns OFF.

(c) 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.

(d) Valid setting range for ⓢ2 is 0 to 32767.

(e) Valid setting range for ⓢ3 is -32768 to 32767. However, if negative value is specified, the initial value is 0.

(3) TIMER_HIGH_FB_M

(a) The high-speed timer with the unit of measurement from 0.1 to 100ms. Starts measuring the current value when the executing condition of ⓢ1 turns ON.

Starts measuring from the value input to ⓢ3 × 0.1 to 100ms, and when the measuring value reaches to the value input to ⓢ2 × 0.1 to 100ms, ⓓ2 turns ON.

The current value is output from ⓓ1 .

(b) When the executing condition of ⓢ1 turns OFF, the current value is set to the value input to ⓢ3 , and ⓓ2 turns OFF.

(c) 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

(d) Valid setting range for ⓢ2 is 0 to 32767.

(e) Valid setting range for ⓢ3 is -32768 to 32767. However, if negative value is specified, the initial value is 0.

(4) TIMER_LOW_FB_M

(a) The low-speed timer with the unit of measurement from 1 to 1000ms. Starts measuring the current value when the executing condition of ⓢ1 turns ON.

Starts measuring from the value input to ⓢ3 × 1 to 1000ms, and when the measuring value reaches to the value input to ⓢ2 × 1 to 1000ms, ⓓ2 turns ON.

The current value is output from ⓓ1 .

(b) When the executing condition of ⓢ1 turns OFF, the current value is set to the value input to ⓢ3 , and ⓓ2 turns OFF.

(c) 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.

(d) Valid setting range for ⓢ2 is 0 to 32767.

(e) Valid setting range for ⓢ3 is -32768 to 32767. However, if negative value is specified, the initial value is 0.

(5) TIMER_CONT_FB_M, TIMER_CONTHFB_M

(a) The retentive timer that measures the time during variable is ON. Starts measuring the current value when the executing condition of ⓢ1 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 ⓢ3 × 1 to 1000ms, and when the count value reaches to the value input to ⓢ2 × 1 to 1000ms, ⓓ2 turns ON.

The current value is output from ⓓ1 .

(b) Even when the executing condition of ⓢ1 turns OFF, the ON/OFF statuses of measuring value ⓓ1 and ⓓ2 are retained. When the executing condition of ⓢ1 turns ON again, restarts measuring from the values that are retained.

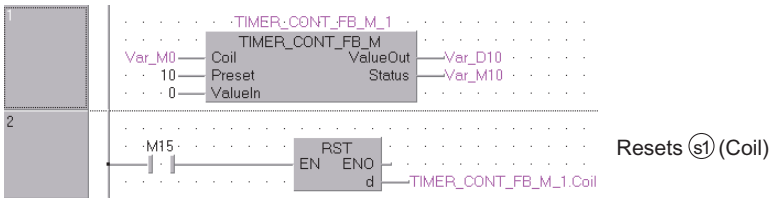
(c) The unit of measurement (time period) for retentive timer is same as the low-speed timer (TIMER_LOW_FB_M) and the high-speed timer (TIMER_HIGH_FB_M).

- Low-speed retentive timer : Low-speed timer
- High-speed retentive timer : High-speed timer

(d) Valid setting range for ⓢ2 is 0 to 32767.

(e) Valid setting range for ⓢ3 is -32768 to 32767. However, if negative value is specified, the initial value is 0.

(f) When resetting the current value of the retentive timer, reset ⓢ1 .
(Example) When instance name is TIMER_CONT_FB_M_1.
[Structured ladder/FBD]



[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

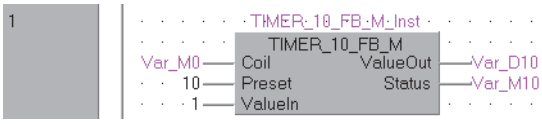
No operation error occurs in the execution of the timer function blocks.

Program Example

(1) TIMER_10_FB_M

The program which starts measuring from ⓢ3 × 10ms when the executing condition of ⓢ1 turns ON, and when the measuring value reaches to the value input to ⓢ2 × 10ms, ⓓ2 turns ON.

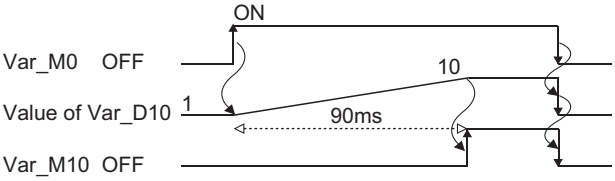
[Structured ladder/FBD]



[ST]

```
TIMER_10_FB_M_Inst(Coil:= Var_M0 ,Preset:= 10 ,ValueIn:= 1 ,ValueOut:=
Var_D10 ,Status:= Var_M10 );
```

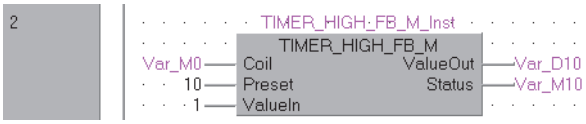
[Timing chart]



(2) TIMER_HIGH_FB_M

The program which starts measuring from ⓢ3 × 10ms when the executing condition of ⓢ1 turns ON, and when the measuring value reaches to the value input to ⓢ2 × 10ms, ⓓ2 turns ON.

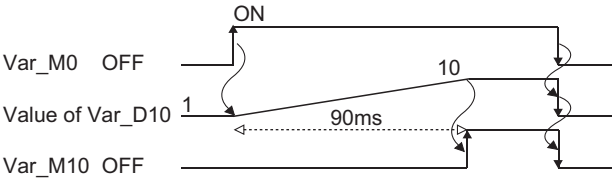
[Structured ladder/FBD]



[ST]

```
TIMER_HIGH_FB_M_Inst(Coil:= Var_M0 ,Preset:= 10 ,ValueIn:= 1 ,ValueOut:=
Var_D10 ,Status:= Var_M10);
```

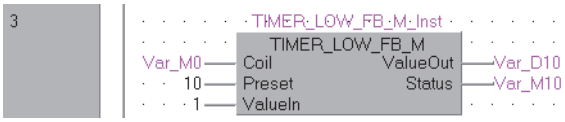
[Timing chart]



(3) TIMER_LOW_FB_M

The program which starts measuring from ⓢ3 × 10ms when the executing condition of ⓢ1 turns ON, and when the measuring value reaches to the value input to ⓢ2 × 100ms, ⓓ2 turns ON.

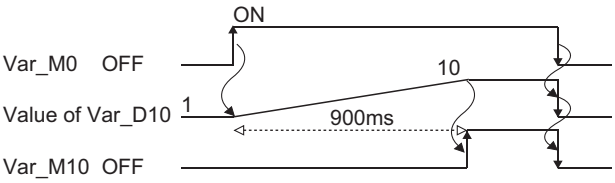
[Structured ladder/FBD]



[ST]

```
TIMER_LOW_FB_M_Inst(Coil:= Var_M0 ,Preset:= 10 ,ValueIn:= 1 ,ValueOut:=
Var_D10 ,Status:= Var_M10);
```

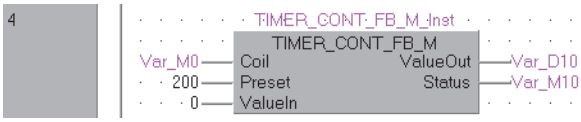
[Timing chart]



(4) TIMER_CONT_FB_M

The program which measures from ⓢ3 × 10ms, and when the measuring value reaches to the value input to ⓢ2 × 100ms, ⓓ2 turns ON.

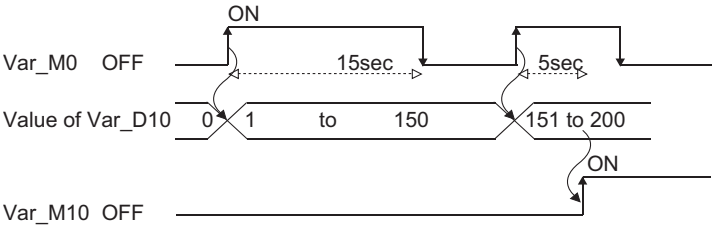
[Structured ladder/FBD]



[ST]

```
TIMER_CONT_FB_M_Inst(Coil:= Var_M0 ,Preset:= 200 ,ValueIn:= 0 ,Value
Out:= Var_D10 ,Status:= Var_M10);
```

[Timing chart]



MEMO

6 OPERATOR

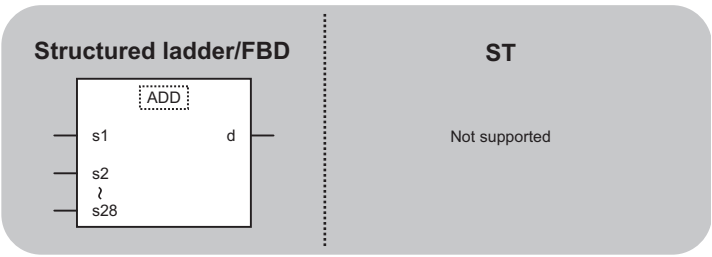
6.1 Arithmetic Operations

6.1.1 Addition

ADD, +

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

ADD
+



indicates any of the following operator.
ADD

indicates any of the following operator.
+

Input argument,	s1 to s28:	Input	:ANY_NUM
Output argument,	d:	Output	:ANY_NUM

Function

Operation processing

For details of the operation processing, refer to Section 5.3.1.

Operation Error

No operation error occurs in the execution of the ADD and + operations.

Program Example

The program which performs addition (ⓢ1 +ⓢ2) on double word (signed) type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

[Structured ladder/FBD]

[ST]

```
g_dint3:= (g_dint1) + (g_dint2);
```

6.1.2 Multiplication

MUL, *

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

MUL
*

indicates any of the following operator.

MUL

indicates any of the following operator.

*

Input argument,	s1 to s28:	Input	:ANY_NUM
Output argument,	d:	Output	:ANY_NUM

Function

Operation processing

For details of the operation processing, refer to Section 5.3.2.

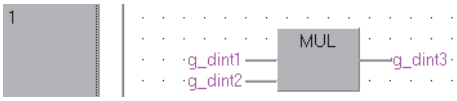
Operation Error

No operation error occurs in the execution of the MUL and * operations.

Program Example

The program which performs multiplication (ⓢ1 × ⓢ2) on double word (signed) type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

[Structured ladder/FBD]



[ST]

```
g_dint3:= (g_dint1) * (g_dint2);
```

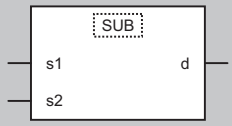
6.1.3 Subtraction

SUB, -

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

SUB
-

Structured ladder/FBD



ST

Not supported

indicates any of the following operator.
SUB

Structured ladder/FBD

Not supported

ST

d:=s1 - s2;

indicates any of the following operator.
-

Input argument,	s1:	Input	:ANY_NUM
	s2:		
Output argument,	d:	Output	:ANY_NUM

Function

Operation processing

For details of the operation processing, refer to Section 5.3.3.

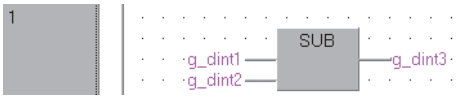
Operation Error

No operation error occurs in the execution of the SUB and - operations.

Program Example

The program which performs subtraction (ⓢ1 − ⓢ2) on double word (signed) type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

[Structured ladder/FBD]



[ST]

```
g_dint3:= (g_dint1) - (g_dint2);
```

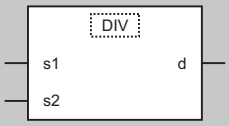
6.1.4 Division

DIV, /

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

DIV
/

Structured ladder/FBD



ST

Not supported

indicates any of the following operator.

DIV

Structured ladder/FBD

Not supported

ST

d:=s1 / s2;

indicates any of the following operator.

/

Input argument,	s1:	Input	:ANY_NUM
	s2:		
Output argument,	d:	Output	:ANY_NUM

Function

Operation processing

For details of the operation processing, refer to Section 5.3.4.

Operation Error

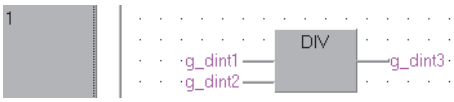
An operation error occurs in the following case.

- The value to be input to ⓢ2 is 0. (Division by 0) (Error code: 4100)

Program Example

The program which performs division (ⓢ1 ÷ⓢ2) on double word (signed) type data input to ⓢ1 and ⓢ2 , and outputs the quotient of the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

[Structured ladder/FBD]



[ST]

```
g_dint3:= (g_dint1) / (g_dint2);
```

6.1.5 Modules operation

MOD

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

MOD

Structured ladder/FBD	ST
Not supported	d:=s1 MOD s2;

indicates any of the following operator.

MOD

Input argument,	s1:	Input	:ANY_INT
	s2:		
Output argument,	d:	Output	:ANY_INT

Function

Operation processing

For details of the operation processing, refer to Section 5.3.5.

Operation Error

An operation error occurs in the following case.

- The value to be input to ⓢ2 is 0. (Division by 0) (Error code: 4100)

Program Example

The program which performs division (ⓢ1 ÷ⓢ2) on double word (signed) type data input to ⓢ1 and ⓢ2 , and outputs the remainder of the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

[ST]

```
g_dint3:= (g_dint1) MOD (g_dint2);
```

6.1.6 Exponentiation

**

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

**

Structured ladder/FBD	ST
Not supported	d:=s1 ** s2;

indicates any of the following operator.
**

Input argument,	s1:	Input	:ANY_REAL
	s2:	Input	:ANY_NUM
Output argument,	d:	Output	:ANY_REAL

Function

Operation processing

For details of the operation processing, refer to Section 5.3.6.

Operation Error

These operator consist of the following common instructions.

When ⓢ1 is single-precision real number, ⓢ2 is word (signed): LOG, FLT

When ⓢ1 is single-precision real number, ⓢ2 is double word (signed): LOG, DFLT

When ⓢ1 is single-precision real number, ⓢ2 is single-precision real number: LOG

When ⓢ1 is single-precision real number, ⓢ2 is double-precision real number: LOGD, DFLTD

When ⓢ1 is double-precision real number, ⓢ2 is word (signed): LOGD

When ⓢ1 is double-precision real number, ⓢ2 is double word (signed): LOGD, FLTD

When ⓢ1 is double-precision real number, ⓢ2 is single-precision real number: LOGD, DFLTD

When ⓢ1 is double-precision real number, ⓢ2 is double-precision real number: LOGD

For details of an error which occurs when the operation is executed, refer to MELSEC-Q/L Structured Programming Manual (Common Instructions).

Program Example

The program which performs exponentiation and outputs the operation result from ⓓ in the same data type as that of ⓢ1 and ⓢ2 .

[ST]

```
g_real2:= (g_real1) ** ( g_int1);
```

6.2 Logical Operations

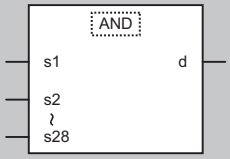
6.2.1 Boolean AND, boolean OR, boolean exclusive OR, and boolean NOT

AND, &, OR, XOR, NOT

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

AND
$
OR
XOR
NOT

Structured ladder/FBD



ST

Not supported

indicates any of the following operators.

AND
OR
XOR

Structured ladder/FBD

Not supported

ST

d:=s1 AND s2…s28,d);

indicates any of the following operators.

AND &
OR
XOR
NOT

Input argument,	s1 to s28: (s1 only for NOT)	Input	:ANY_BIT
Output argument,	d:	Output	:ANY_BIT

Function

Operation processing

For details of the operation processing, refer to Section 5.4.1.

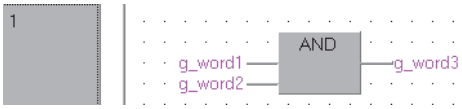
Operation Error

No operation error occurs in the execution of the AND, &, OR, XOR, and NOT operations.

Program Example

(1) The program which performs Boolean AND on bit, word (unsigned)/16-bit string type data input to ⓢ1 to ⓢ28 bit by bit, and outputs the operation result from ⓓ in the same data type as that of ⓢ1 to ⓢ28 .

[Structured ladder/FBD]

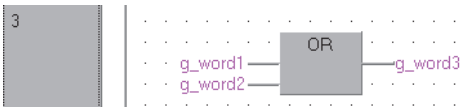


[ST]

```
g_word3 :=(g_word1) AND (g_word2);
or
g_word3 :=(g_word1) & (g_word2);
```

(2) The program which performs Boolean OR on bit, word (unsigned)/16-bit string type data input to ⓢ1 to ⓢ28 bit by bit, and outputs the operation result from ⓓ in the same data type as that of ⓢ1 to ⓢ28 .

[Structured ladder/FBD]

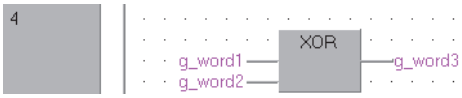


[ST]

```
g_word3 :=(g_word1) OR (g_word2);
```

(3) The program which performs Boolean XOR on bit, word (unsigned)/16-bit string type data input to ⓢ1 to ⓢ28 bit by bit, and outputs the operation result from ⓓ in the same data type as that of ⓢ1 to ⓢ28 .

[Structured ladder/FBD]

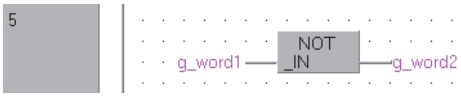


[ST]

```
g_word3 :=(g_word1) XOR (g_word2);
```

(4) The program which performs Boolean NOT on bit, word (unsigned)/16-bit string type data input to ⓢ1 bit by bit, and outputs the operation result from ⓓ in the same data type as that of ⓢ1 .

[Structured ladder/FBD]



[ST]

```
g_word2 :=NOT (g_word1);
```

6.3 Comparison Operations

6.3.1 Comparison

GT, GE, EQ, LE, LT, NE,
>, >=, =, <=, <, <>

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

GT	>
GE	>=
EQ	=
LE	<=
LT	<
NE	<>

Structured ladder/FBD

ST

Not supported

[ ] indicates any of the following operators.

GT
GE
EQ
LE
LT
NE

Structured ladder/FBD

Not supported

ST

d:=s1 [>] s2…s28;

[ ] indicates any of the following operators.

>
>=
=
<=
<
<>

Input argument,	s1 to s28(_IN):	Input	:ANY_SIMPLE
	(s1 and s2 only for NE and <>)		
Output argument,	d:	Output (TRUE: True value, FALSE: False value)	:Bit

Function

Operation processing

For details of the operation processing, refer to Section 5.6.1.

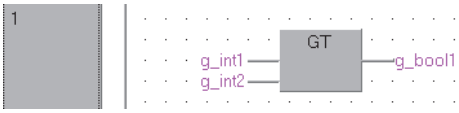
Operation Error

No operation error occurs in the execution of the GT, GE, EQ, LE, LT, NE, >, >=, =, <=, < and <> operations.

Program Example

The program which performs comparison operation between the values input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ .

[Structured ladder/FBD]



[ST]

```
g_bool1:= (g_int1) > (g_int2);
```

MEMO

I INDEX

MEMO

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 onsite 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 damages caused by any cause found not to be the responsibility of Mitsubishi, loss in opportunity, lost profits incurred to the user by Failures of Mitsubishi products, special damages and secondary damages whether foreseeable or not, compensation for accidents, and compensation for damages to products other than Mitsubishi products, replacement by the user, maintenance of on-site equipment, start-up test run and other tasks.

5. Changes in product specifications

The specifications given in the catalogs, manuals or technical documents are subject to change without prior notice.

SH(NA)-080784ENG-K

MELSEC-Q/L Structured Programming Manual

Application Functions

MODEL	Q-KP-OK-E
MODEL CODE	13JW08
SH(NA)-080784ENG-K(1306)KWIX	

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