QCPU

MITSUBISHI

Structured Programming Manual

(Application Functions)

Mitsubishi
Programmable Controller

MELSEC-Q

● SAFETY PRECAUTIONS ●

(Always read these instructions before using this product.)

Before using the MELSEC-Q series programmable controller, thoroughly read the manuals attached to the products and the relevant manuals introduced in the attached manuals. Also pay careful attention to safety and handle the products properly.

Please keep this manual in a place where it is accessible when required and always forward it to the end user.

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

Japanese manual version SH-080737-B

INTRODUCTION

Thank you for purchasing the Mitsubishi MELSEC-Q series programmable controller.
Before using the product, thoroughly read this manual to develop full familiarity with the programming specifications to ensure correct use.
Please forward this manual to the end user.

CONTENTS

MANUALS

Related manuals

The manuals related to this product are shown below.

Refer to the following tables when ordering required manuals.

(1) Structured programming

Manual name	Manual number (Model code)
QCPU Structured Programming Manual (Fundamentals) Explains the programming method, types of programming languages, and other information required to create structured programs. (Sold separately)	SH-080782ENG (13JW06)
QCPU Structured Programming Manual (Common Instructions) Explains the specifications and functions of sequence instructions, basic instructions, and application instructions that can be used in structured programs. (Sold separately)	SH-080783ENG (13JW07)
QCPU Structured Programming Manual (Special Instructions) Explains the specifications and functions of instructions for network modules, intelligent function modules, and PID control functions 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 Version1 Operating Manual (Common) Explains the system configuration of GX Works2 and the functions common to a Simple project and Structured project such as parameter setting, operation method for the online function. (Sold separately)	SH-080779ENG (13JU63)
GX Works2 Version1 Operating Manual (Structured Project) Explains operation methods such as creating and monitoring programs in Structured project of GX Works2. (Sold separately)	SH-080781ENG (13JU65)
GX Works2 Beginner's Manual (Structured Project) Explains fundamental operation methods such as creating, editing, and monitoring programs in Structured project for users inexperienced with GX Works2. (Sold separately)	SH-080788ENG (13JZ23)

POINT

The Operating Manual is included in the CD-ROM with the software package. Manuals in printed form are sold separately. Order a manual by quoting the manual number (model code) listed in the table above.

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 Version1 Operating Manual		
							
		–	Simple Project	Structured Project	Common	Simple Project	Structured Project
Installation	Learning the operating environment and installation method	Details					
Operation of Simple project	Learning the basic operations and operating procedures		Details		Outline	Outline	
	Learning the functions and operation methods for programming				Outline	Details	
	Learning all functions and operation methods except for programming				Details		
Operation of Structured project	Learning the basic operations and operating procedures			Details	Outline		Outline
	Learning the functions and operation methods for programming				Outline	Details	Details
	Learning all functions and operation methods except for programming				Details		

(2) Programming

Purpose		QCPU Structured Programming Manual				QCPU(Q mode)/QnACPU Programming Manual		User's Manual for intelligent function module/ Reference Manual for network module
								
		Fundamentals	Common Instructions	Special Instructions	Application Functions	Common Instructions	PID Control Instructions	–
Programming in Simple project	Learning the types and details of common instructions, descriptions of error codes, special relays, and special registers					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	
Programming in Structured project	Learning the fundamentals for creating a structured program for the first time	Details						
	Learning the types and details of common instructions		Details					
	Learning the types and details of instructions for intelligent function modules			Details				Details
	Learning the types and details of instructions for network modules			Details				Details
	Learning the types and details of instructions for the PID control function			Details			Details	
	Learning the descriptions of error codes, special relays, and special registers					Details		
	Learning the types and details of application functions				Details			

1.2 Generic Terms and Abbreviations in This Manual

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.

Generic term and abbreviation	Description
GX Works2	Generic product name for the SWnDNC-GXW2-E (n: version)
CPU module	Generic term for the High Performance model QCPU and Universal model QCPU
High Performance model QCPU	Generic term for the Q02, Q02H, Q06H, Q12H, and Q25H
Universal model QCPU	Generic term for the Q02U, Q03UD, Q03UDE, Q04UDH, Q04UDEH, Q06UDH, Q06UDEH, Q13UDH, Q13UDEH, Q26UDH, and Q26UDEH
Personal computer	Generic term for personal computer on which Windows® operates
IEC61131-3	Abbreviation for the IEC 61131-3 international standard
Common instruction	Generic term for the sequence instructions, basic instructions, and application instructions
Special instruction	Generic term for the PID control instructions and module dedicated instructions

2 FUNCTION TABLES

2.1 How to Read Function Tables

Function name	Argument	Processing details	Page
ADD(_E)	ⓢ1 ,ⓢ2 ,...ⓢ8 ,ⓓ (Number of pins variable)	Outputs the sum (ⓢ1 +ⓢ2 +···+ⓢ8) of input values.	5-139
MUL(_E)	ⓢ1 ,ⓢ2 ,...ⓢ8 ,ⓓ (Number of pins variable)	Outputs the product (ⓢ1 ×ⓢ2 ×···×ⓢ8) of input values.	5-142
SUB(_E)	ⓢ1 , ⓢ2 , ⓓ	Outputs the difference (ⓢ1 − ⓢ2) between input values.	5-145
DIV(_E)	ⓢ1 , ⓢ2 , ⓓ	Outputs the quotient (ⓢ1 ÷ ⓢ2) of input values.	5-148

① ② ③ ④

Description

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

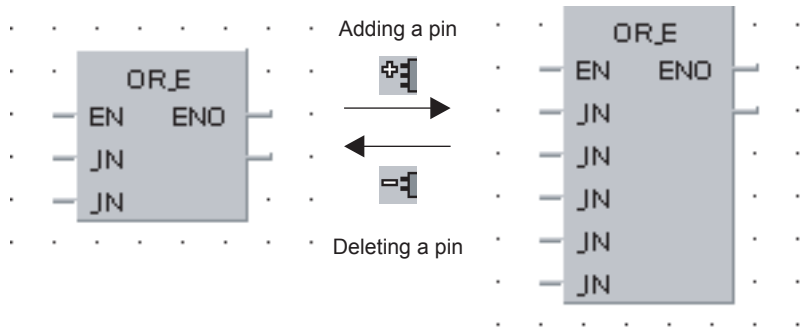
② .....Indicates the arguments of the function.

ⓢ : 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 8.

Changing the number of pins



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

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

2.1.1 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)	ⓢ , ⓓ		5-2
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)	ⓢ , ⓓ		5-7
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.	5-13
DINT_TO_INT(_E)	ⓢ , ⓓ	Converts double word (signed) type data into word (signed) type data.	5-16
INT_TO_BOOL(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into bit type data.	5-19
DINT_TO_BOOL(_E)	ⓢ , ⓓ		5-19
INT_TO_REAL(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into single-precision real type data.	5-23
DINT_TO_REAL(_E)	ⓢ , ⓓ		5-23
INT_TO_LREAL(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into double-precision real type data.	5-26
DINT_TO_LREAL(_E)	ⓢ , ⓓ		5-26
INT_TO_STR(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into string type data.	5-29
DINT_TO_STR(_E)	ⓢ , ⓓ		5-29
INT_TO_WORD(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into word (unsigned)/16-bit string type data.	5-33
DINT_TO_WORD(_E)	ⓢ , ⓓ		5-33
INT_TO_DWORD(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into double word (unsigned)/32-bit string type data.	5-36
DINT_TO_DWORD(_E)	ⓢ , ⓓ		5-36
INT_TO_BCD(_E)	ⓢ , ⓓ	Converts word (signed) or double word (signed) type data into BCD type data.	5-39
DINT_TO_BCD(_E)	ⓢ , ⓓ		5-39
INT_TO_TIME(_E)	ⓢ , ⓓ	Converts word (signed) type data into time type data.	5-42
DINT_TO_TIME(_E)	ⓢ , ⓓ		5-42
REAL_TO_INT(_E)	ⓢ , ⓓ	Converts single-precision real type data into word (signed) or double word (signed) type data.	5-45
REAL_TO_DINT(_E)	ⓢ , ⓓ		5-45
LREAL_TO_INT(_E)	ⓢ , ⓓ	Converts double-precision real type data into word (signed) or double word (signed) type data.	5-48
LREAL_TO_DINT(_E)	ⓢ , ⓓ		5-48
REAL_TO_LREAL(_E)	ⓢ , ⓓ	Converts single-precision real type data into double-precision real type data.	5-51
LREAL_TO_REAL(_E)	ⓢ , ⓓ	Converts double-precision real type data into single-precision real type data.	5-54
REAL_TO_STR(_E)	ⓢ , ⓓ	Converts single-precision real type data into string type (exponent form) data.	5-57
WORD_TO_BOOL(_E)	ⓢ , ⓓ	Converts word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data into bit type data.	5-61
DWORD_TO_BOOL(_E)	ⓢ , ⓓ		5-61
WORD_TO_INT(_E)	ⓢ , ⓓ	Converts word (unsigned)/16-bit string type data into word (signed) or double word (signed) type data.	5-64
WORD_TO_DINT(_E)	ⓢ , ⓓ		5-64
DWORD_TO_INT(_E)	ⓢ , ⓓ	Converts double word (unsigned)/32-bit string type data into word (signed) or double word (signed) type data.	5-67
DWORD_TO_DINT(_E)	ⓢ , ⓓ		5-67
WORD_TO_DWORD(_E)	ⓢ , ⓓ	Converts word (unsigned)/16-bit string type data into double word (unsigned)/32-bit string type data.	5-70
DWORD_TO_WORD(_E)	ⓢ , ⓓ	Converts double word (unsigned)/32-bit string type data into word (unsigned)/16-bit string type data.	5-73
WORD_TO_STR(_E)	ⓢ , ⓓ	Converts word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data into string type data.	5-76
DWORD_TO_STR(_E)	ⓢ , ⓓ		5-76
WORD_TO_TIME(_E)	ⓢ , ⓓ	Converts word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data into time type data.	5-79
DWORD_TO_TIME(_E)	ⓢ , ⓓ		5-79
STR_TO_BOOL(_E)	ⓢ , ⓓ	Converts string type data into bit type data.	5-82
STR_TO_INT(_E)	ⓢ , ⓓ	Converts string type data into word (signed) or double word (signed) type data.	5-85
STR_TO_DINT(_E)	ⓢ , ⓓ		5-85
STR_TO_REAL(_E)	ⓢ , ⓓ	Converts string type data into single-precision real type data.	5-88

Function name	Argument	Processing details	Page
STR_TO_WORD(_E)	ⓢ , ⓓ	Converts string type data into word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data.	5-92
STR_TO_DWORD(_E)	ⓢ , ⓓ		5-92
STR_TO_TIME(_E)	ⓢ , ⓓ	Converts string type data into time type data.	5-95
STR_TO_BCD(_E)	ⓢ , ⓓ	Converts string type data into BCD type data.	5-98
BCD_TO_INT(_E)	ⓢ , ⓓ	Converts BCD type data into word (signed) or double word (signed) type data.	5-101
BCD_TO_DINT(_E)	ⓢ , ⓓ		5-101
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.	5-110
TIME_TO_DINT(_E)	ⓢ , ⓓ		5-110
TIME_TO_STR(_E)	ⓢ , ⓓ	Converts time type data into string type data.	5-113
TIME_TO_WORD(_E)	ⓢ , ⓓ	Converts time type data into word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data.	5-116
TIME_TO_DWORD(_E)	ⓢ , ⓓ		5-116

2.1.2 Standard functions of one numeric variable

Function name	Argument	Processing details	Page
ABS(_E)	ⓢ , ⓓ	Outputs the absolute value of an input value.	5-119

2.1.3 Standard arithmetic functions

Function name	Argument	Processing details	Page
ADD(_E)	ⓢ1 ,ⓢ2 ,...ⓢ8 ,ⓓ (Number of pins variable)	Outputs the sum (ⓢ1 +ⓢ2 +···+ⓢ8) of input values.	5-122
MUL(_E)	ⓢ1 ,ⓢ2 ,...ⓢ8 ,ⓓ (Number of pins variable)	Outputs the product (ⓢ1 ×ⓢ2 ×···×ⓢ8) of input values.	5-125
SUB(_E)	ⓢ1 , ⓢ2 , ⓓ	Outputs the difference (ⓢ1 −ⓢ2) between input values.	5-128
DIV(_E)	ⓢ1 , ⓢ2 , ⓓ	Outputs the quotient (ⓢ1 ÷ⓢ2) of input values.	5-131
MOD(_E)	ⓢ1 , ⓢ2 , ⓓ	Outputs the remainder after division of input values (ⓢ1 ÷ⓢ2).	5-134
EXPT(_E)	ⓢ1 , ⓢ2 , ⓓ	Outputs the exponentiation of an input value.	5-137
MOVE(_E)	ⓢ1 ,ⓢ2 ,...ⓢ8 ,ⓓ (Number of pins variable)	Outputs the substitution of an input value.	5-140

2.1.4 Standard bitwise Boolean functions

Function name	Argument	Processing details	Page
AND(_E)	ⓢ1 ,ⓢ2 ,...ⓢ8 ,ⓓ (Number of pins variable)	Outputs the Boolean AND of input values.	5-143
OR(_E)	ⓢ1 ,ⓢ2 ,...ⓢ8 ,ⓓ (Number of pins variable)	Outputs the Boolean OR of input values.	5-143
XOR(_E)	ⓢ1 ,ⓢ2 ,...ⓢ8 ,ⓓ (Number of pins variable)	Outputs the Boolean exclusive OR of input values.	5-143
NOT(_E)	ⓢ , ⓓ	Outputs the Boolean NOT of input values.	5-143

2.1.5 Standard selection functions

Function name	Argument	Processing details	Page
SEL(_E)	㉑ , ㉒ , ㉓ , ⓓ	Outputs the value selected from the input values.	5-148
MUX(_E)	㉑ ,㉒ ,...㊽ ,ⓓ (Number of pins variable)	Outputs one of the multiple input values.	5-151

2.1.6 Standard comparison functions

Function name	Argument	Processing details	Page
GT(_E)	㉑ ,㉒ ,...㊽ ,ⓓ (Number of pins variable)	Outputs the comparison value of an input value.	5-154
GE(_E)	㉑ ,㉒ ,...㊽ ,ⓓ (Number of pins variable)		5-154
EQ(_E)	㉑ ,㉒ ,...㊽ ,ⓓ (Number of pins variable)		5-154
LE(_E)	㉑ ,㉒ ,...㊽ ,ⓓ (Number of pins variable)		5-154
LT(_E)	㉑ ,㉒ ,...㊽ ,ⓓ (Number of pins variable)		5-154
NE(_E)	㉑ , ㉒ , ⓓ		5-154

2.1.7 Standard character string functions

Function name	Argument	Processing details	Page
MID(_E)	㉑ , ㉒ , ㉓ , ⓓ	Outputs the specified number of characters, extracted from the specified start position in the input character string.	5-157
CONCAT(_E)	㉑ , ㉒ , ⓓ	Concatenates two character strings and outputs the operation result.	5-160
INSERT(_E)	㉑ , ㉒ , ㉓ , ⓓ	Inserts a character string between other character strings and outputs the operation result.	5-163
DELETE(_E)	㉑ , ㉒ , ㉓ , ⓓ	Deletes the specified range in a character string and outputs the operation result.	5-166
REPLACE(_E)	㉑ , ㉒ , ㉓ , ㉔ , ⓓ	Replaces the specified range in a character string with the specified character string and outputs the operation result.	5-169

2.1.8 Functions of time data types

Function name	Argument	Processing details	Page
ADD_TIME(_E)	㉑ , ㉒ , ⓓ	Outputs the sum (㉑ +㉒) of the input values (time type).	5-172
SUB_TIME(_E)	㉑ , ㉒ , ⓓ	Outputs the difference (㉑ -㉒) of input values (time type).	5-175

2.1.9 Standard bistable function blocks

Function name	Argument	Processing details	Page
SR(_E)	㉑ , ㉒ , ⓓ	Discriminates two input values and outputs 1 (TRUE) or 0 (FALSE).	5-184
RS(_E)	㉑ , ㉒ , ⓓ	Discriminates two input values and outputs 1 (TRUE) or 0 (FALSE).	5-187

2.1.10 Standard edge detection function blocks

Function name	Argument	Processing details	Page
RTRIG(_E)	ⓢ1 , ⓓ	Detects the rising edge of a signal and outputs pulse signals.	5-190
F_TRIG(_E)	ⓢ1 , ⓓ	Detects the falling edge of a signal and outputs pulse signals.	5-192

2.1.11 Standard counter function blocks

Function name	Argument	Processing details	Page
CTU(_E)	ⓢ1 , ⓢ2 , ⓢ3 , ⓓ1 , ⓓ2	Counts the number of times that the signal turns ON.	5-194
CTD(_E)	ⓢ1 , ⓢ2 , ⓢ3 , ⓓ1 , ⓓ2	Counts down the number of times that the signal turns ON.	5-197
CTUD(_E)	ⓢ1 , ⓢ2 , ⓢ3 , ⓢ4 ⓢ5 , ⓓ1 , ⓓ2 , ⓓ3	Counts/counts down the number of times that the signal turns ON.	5-200
COUNTER_FB_M	ⓢ1 , ⓢ2 , ⓢ3 , ⓓ1 , ⓓ2	Counts the number of times that the signal turns ON from ⓢ3 to ⓢ2 .	5-203

2.1.12 Standard timer function blocks

Function name	Argument	Processing details	Page
TP(_E) TP_HIGH(_E)	ⓢ1 , ⓢ2 , ⓓ1 , ⓓ2	Holds the signal ON for the specified time.	5-205
TON(_E) TON_HIGH(_E)	ⓢ1 , ⓢ2 , ⓢ3 , ⓓ1 , ⓓ2	Turns ON the signal after the specified time.	5-208
TOF(_E) TOF_HIGH(_E)	ⓢ1 , ⓢ2 , ⓢ3 , ⓢ4 ⓢ5 , ⓓ1 , ⓓ2 , ⓓ3	Turns OFF the signal after the specified time.	5-211
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 .	5-214

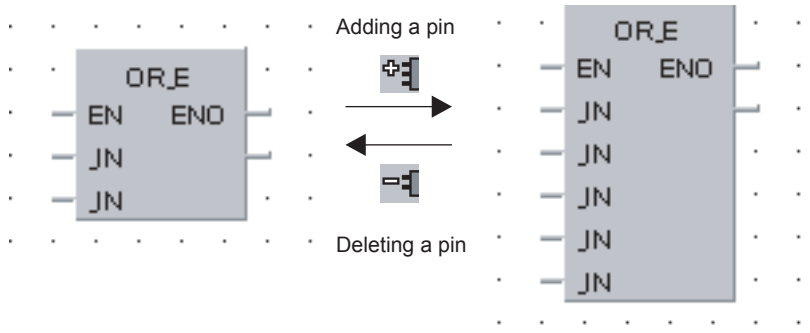
3 FUNCTIONS

3.1 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 Version1 Operating Manual (Structured Project)



3.2 Functions with EN

The function has two types: ordinary functions and functions with EN that have EN/ENO pins.

Functions with EN allow control of function execution.

EN inputs the condition for executing a function.

ENO outputs execution status.

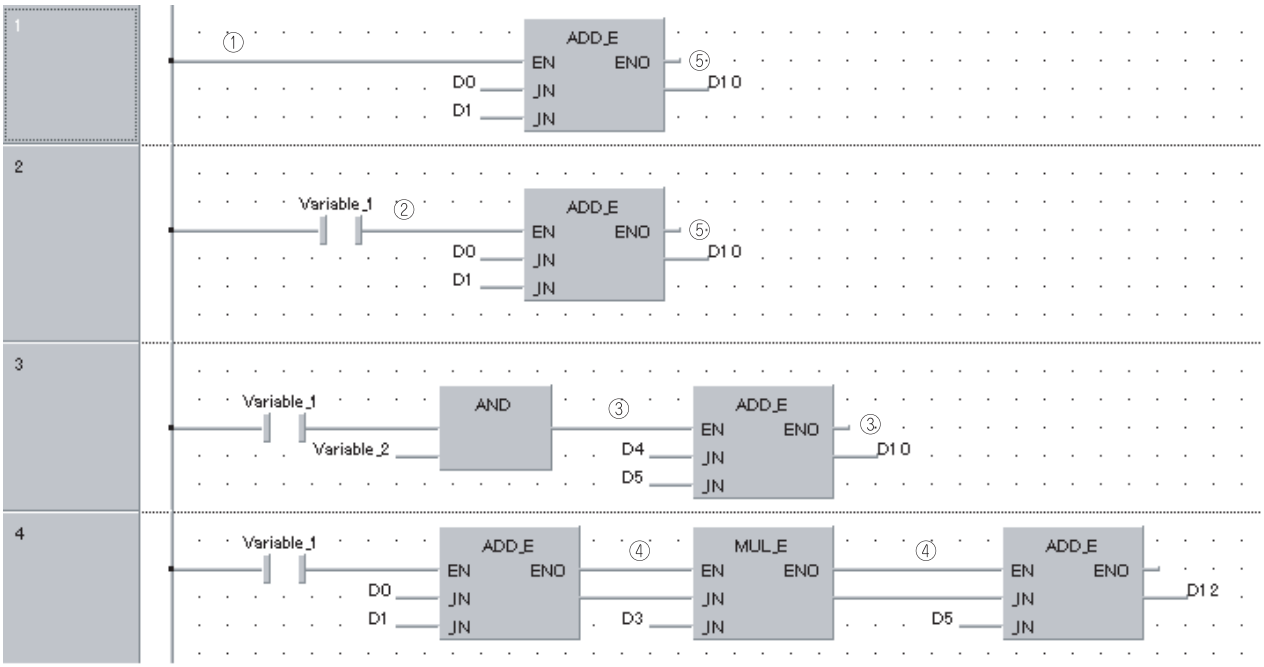
The following table shows the status of EN and ENO and the operation result according to the status of EN.

Table 3.2-1 Status of EN and ENO and the operation result according to the status of EN

EN	ENO	Operation result
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
	FALSE (Operation error)	Undefined value
FALSE (Operation stop)	FALSE	Undefined value

POINT

Functions with EN are expressed as 'Function name_E'.



Examples of use of EN and ENO

No.	Control description
①	When the EN input is directly connected from the left base line, the EN input is always TRUE and the function is always executed. If the ADD_E function is used in this manner, the operation result is the same as the ADD function without the EN input.
②	When Variable_1 is connected to the EN input, the function is executed when Variable_1 is TRUE.
③	When the result of Boolean operation is connected to the EN input, the function is executed when the result is TRUE.
④	When the ENO outputs are connected to the EN inputs, three functions are executed when Variable_1 is TRUE.
⑤	When the ENO outputs are not connected, the execution status of the function is not output.

3.3 Labels

Labels include global labels and local labels.

3.3.1 Global labels

The global labels are labels that can be used in program blocks and function blocks.

In the setting of a global label, a label name, a class, a data type, and a device are associated with each other.

3.3.2 Local labels

The local labels are labels that can be used only in declared POUs. They are individually defined per POU.

In the setting of a local label, a label name, a class, and a data type are set.

For the local labels, the user does not need to specify devices. Devices are assigned automatically at compilation.

3.3.3 Label classes

The label class indicates from which POU and how a label can be used. Different classes can be selected according to the type of POU.

The following table shows label classes.

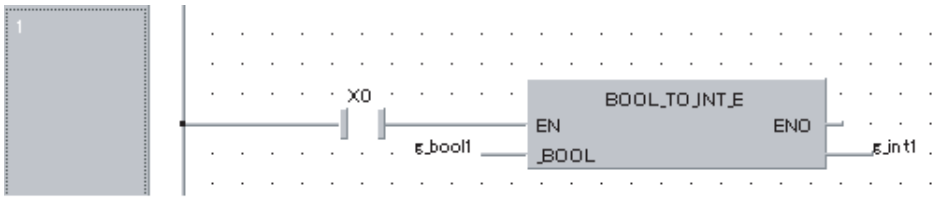
Table 3.3.3-1 Label classes

Class	Description	Applicable POU		
		Program block	Function	Function block
VAR_GLOBAL	Common label that can be used in program blocks and function blocks	○	×	○
VAR_GLOBAL_CONSTANT	Common constant that can be used in program blocks and function blocks	○	×	○
VAR	Label that can be used within the range of declared POUs This label cannot be used in other POUs.	○	○	○
VAR_CONSTANT	Constant that can be used within the range of declared POUs This constant cannot be used in other POUs.	○	○	○
VAR_RETAIN	Latch type label that can be used within the range of declared POUs This label cannot be used in other POUs.	○	×	○
VAR_INPUT	Label that receives a value This label cannot be changed in a POU.	×	○	○
VAR_OUTPUT	Label that outputs a value from a function block	×	×	○
VAR_IN_OUT	Local label that receives a value and outputs the value from a POU This label can be changed in a POU.	×	×	○

3.3.4 Setting labels

Labels used in a program require setting of either global label or local label.

The following describes setting examples of the arguments g_bool1 and g_int1 of the BOOL_TO_INT_E function.



● Using the arguments of the BOOL_TO_INT_E function as global labels
Set the Class, Label Name, Data Type, Device, and Address.

	Class	Label Name	Data Type		Constant	Device	Address	Comment	Remark
1	VAR_GLOBAL	g_bool1	Bit	...		M0	%MX0.0		
2	VAR_GLOBAL	g_int1	Word[Signed]	...		D0	%MW0.0		
3				...					

● Using the arguments of the BOOL_TO_INT_E function as local labels
Set the Class, Label Name, and Data Type.

	Class	Label Name	Data Type		Constant	Device	Address	Comment
1	VAR	g_bool1	Bit	...				
2	VAR	g_int1	Word[Signed]	...				
3				...				

3.4 Data Types

Data types of labels include the elementary data types and the generic data types.

Generic data type is the data type of labels covering some elementary data types. The data type name starts with 'ANY'.

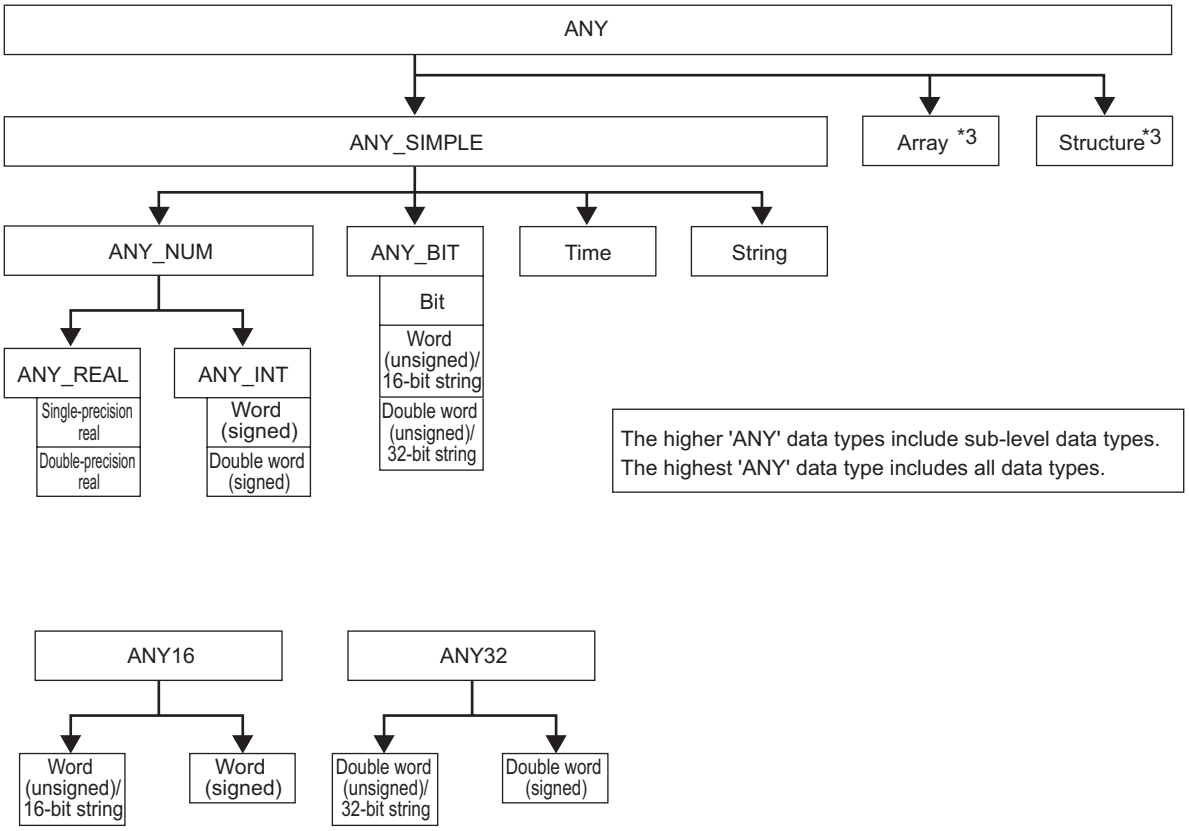
The following shows the list of elementary data types and generic data types.

Table 3.4-1 Elementary data types

Data type	Description	Value range	Bit length
Bit	Bool	0(FALSE), 1(TRUE)	1 bit
Word (signed)	Integer	-32768 to 32767	16 bits
Double word (signed)	Double-precision integer	-2147483648 to 2147483647	32 bits
Word (unsigned)/ 16-bit string	16-bit string	0 to 65535	16 bits
Double word (unsigned)/ 32-bit string	32-bit string	0 to 4294967295	32 bits
Single-precision real	Real	-2^{128} to -2^{-126}, 0, 2^{-126} to 2^{128}	32 bits
Double-precision real*1	Double-precision real	-2^{1024} to -2^{-1022}, 0, 2^{-1022} to 2^{1024}	64 bits
String	Character string	Maximum 255 characters	Variable
Time*2	Time value	T#-24d-0h31m23s648ms to T#24d20h31m23s647ms	32 bits

*1 Can be used for the Universal model QCPU only.
*2 The time type is used in time type operation instructions of application function.

Generic data types



*3 For details ☞ QCPU Structured Programming Manual (Common Instructions)

For data specification method, refer to the following manual.

☞ QCPU Structured Programming Manual (Common Instructions)

For the format of devices that correspond to generic data types, refer to the following section.

☞ Appendix 1 Correspondence between Generic Data Types and Devices

3.5 Device and Address

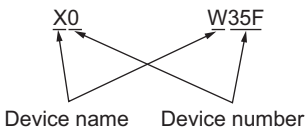
This section explains the method for expressing programmable controller CPU devices. The following two types of format are available.

- Device: This format consists of a device name and a device number.
- Address: A format defined in IEC61131-3. In this format, a device name starts with %.

3.5.1 Device

Device is a format that uses a device name and a device number.

Example)



For details of devices used in the QCPU, refer to the following manual.

☞ QCPU User's Manual (Function Explanation, Program Fundamentals)

3.5.2 Address

Address is a format defined in IEC61131-3.

The following table shows details of format that conforms to IEC61131-3.

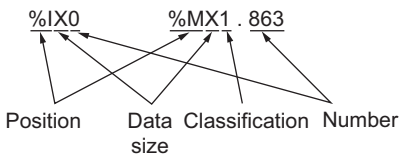
Table 3.5.2-1 Address definition specifications

Start	1st character: position		2nd character: data size		3rd character and later: classification	Number
%	I	Input	(Omitted)	Bit	Numerics used for detailed classification Use '.' (period) to delimit the numbers from the subsequent numbers. A period may be omitted.	Number corresponding to the device number (decimal notation)
	Q	Output	X	Bit		
	M	Internal	W	Word (16 bits)		
			D	Double word (32 bits)		
			L	Long word (64 bits)		

● **Position**

Position is a major class indicating the position to which data are allocated in three types: input, output, and internal.

Example)



The following shows the format rules corresponding to the device format.

- X, J\X (X device) : I (input)
- Y, J\Y (Y device) : Q (output)
- Other devices : M (internal)

● **Data size**

Data size is a class indicating the size of data.

The following shows the format rules corresponding to the device format.

- Bit device : X (bit)
- Word device : W (word), D (double word), L (long word)

● **Classification**

Classification is a minor class indicating the type of a device that cannot be identified only by its position and size.

Devices X and Y do not support classification.

For the format corresponding to the device format, refer to the following section.

☞ Appendix 2 Correspondence between Devices and Addresses

POINT

Long words are used in double-precision real operation instructions of the Universal model QCPU.

3.6 Expressing Methods of Constants

The following table shows the expressing method for setting a constant to a label.

Table 3.6-1 Constant expressing method

Constant type	Expressing method	Example
Bool	Input FALSE or TRUE, or input 0 or 1.	TRUE, FALSE
Binary	Append '2#' in front of a binary number.	2#0010, 2#01101010
Octal notation	Append '8#' in front of an octal number.	8#0, 8#337
Decimal	Directly input a decimal number, or append 'K' in front of a decimal number.	123, K123
Hexadecimal	Append '16#' or 'H' in front of a hexadecimal number. When a lowercase letter 'h' is appended, it is converted to uppercase automatically.	16#FF, HFF
Real number	Directly input a real number or append 'E' in front of a real number.	2.34, E2.34
Character string	Enclose a character string with single quotations (') or double quotations (").	'ABC', "ABC"

3.7 Precautions on Programming

Reserved words cannot be used for a name that is used in a program (label name, function block instance name, structured data type name, etc.).

3.7.1 Precautions on assigning a name

This section explains the conditions for assigning a name and shows the list of reserved words.

Conditions

(1) Specify a name by a character string of up to 32 characters.

(2) Do not use a reserved word.
For reserved words, refer to Table 3.7-1 Reserved words.

(3) Use alphanumeric and underscore (_).

(4) Do not use an underscore at the end of a name.
Do not use two or more underscores in succession.

(5) Do not use a space.

(6) Do not use a number at the initial character.

(7) A constant cannot be used.
(An identifier that begins with 'H' or 'h' and an expression where a hexadecimal (0 to F) immediately follows 'H' or 'h' (maximum 9 digits including 'H' or 'h' (excluding 0 that immediately follows 'H' or 'h')) are also treated as a constant. (Example: 'hab0'))

(8) An elementary data type name cannot be used.

(9) Part names of function/FB cannot be used.

Reserved words list

Table 3.7-1 Reserved words (1/2)

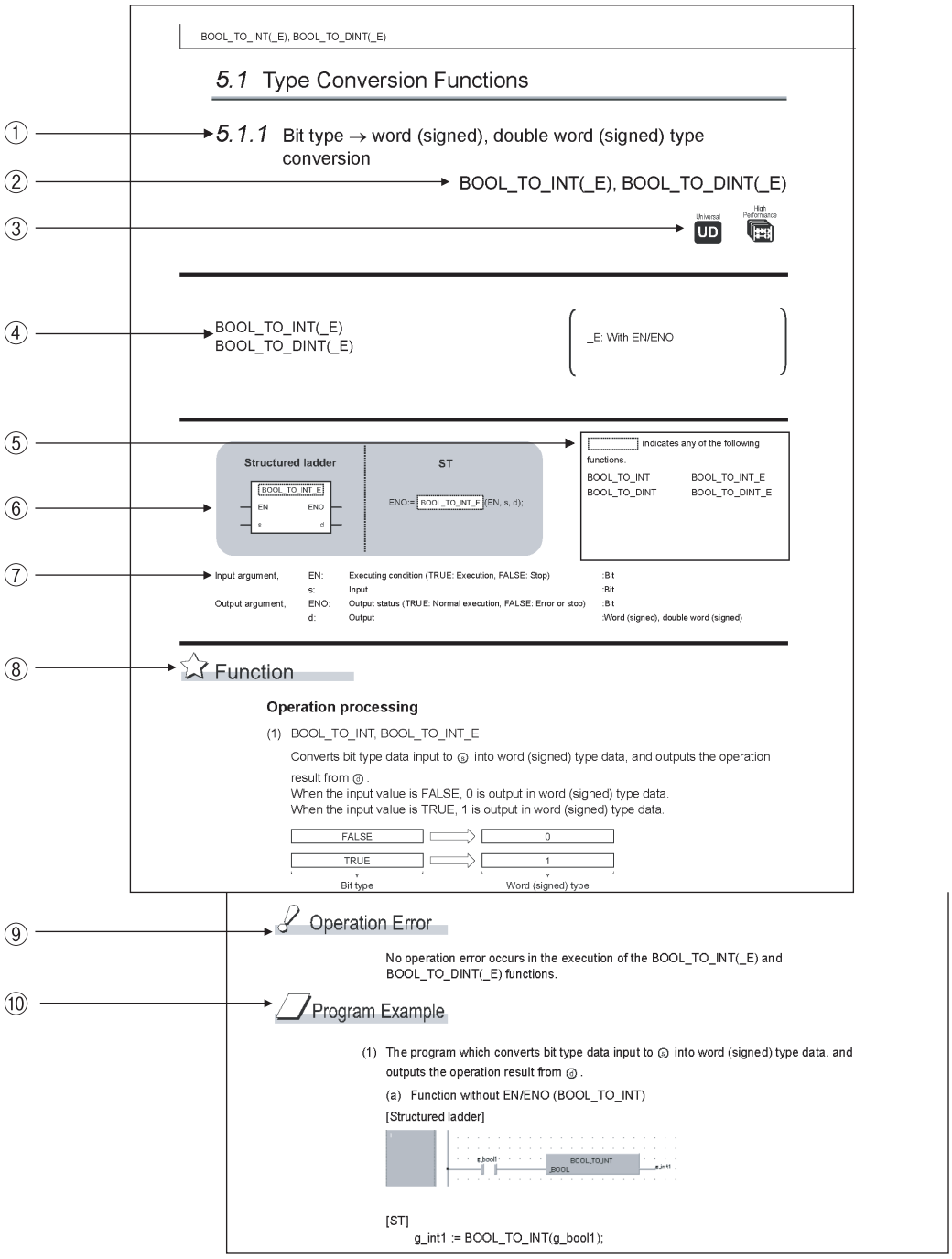
Category	Character string
Class identifier	VAR, VAR_RETAIN, VAR_ACCESS, VAR_CONSTANT, VAR_CONSTANT_RETAIN, VAR_INPUT, VAR_INPUT_RETAIN, VAR_OUTPUT, VAR_OUTPUT_RETAIN, VAR_IN_OUT, VAR_IN_EXT, VAR_EXTERNAL, VAR_EXTERNAL_CONSTANT, VAR_EXTERNAL_CONSTANT_RETAIN, VAR_EXTERNAL_RETAIN, VAR_GLOBAL, VAR_GLOBAL_CONSTANT, VAR_GLOBAL_CONSTANT_RETAIN, VAR_GLOBAL_RETAIN
Data type	BOOL, BYTE, INT, SINT, DINT, LINT, UINT, USINT, UDINT, ULINT, WORD, DWORD, LWORD, ARRAY, REAL, LREAL, TIME, STRING
Data type hierachy	ANY, ANY_NUM, ANY_BIT, ANY_REAL, ANY_INT, ANY_DATE, ANY_SIMPLE, ANY16, ANY32
Device name	X, Y, D, M, T, B, C, F, L, P, V, Z, W, I, N, U, J, K, H, E, A, SD, SM, SW, SB, FX, FY, DX, DY, FD, TR, BL, SG, VD, ZR, ZZ
Character string recognized as device (Device name + Numeral)	Such as X0
ST operator	NOT, MOD
IL operator	LD, LDN, ST, STN, S, S1, R, R1, AND, ANDN, OR, ORN, XOR, XORN, ADD, SUB, MUL, DIV, GT, GE, EQ, NE, LE, LT, JMP, JMPC, JMPCN, CAL, CALC, CALCN, RET, RETC, RETCN, LDI, LDP, LDF, ANI, ANDP, ANDF, ANB, ORI, ORP, ORF, ORB, MPS, MRD, MPP, INV, MEP, MEF, EGP, EGF, OUT(H), SET, RST, PLS, PLF, FF, DELTA(P), SFT(P), MC, MCR, STOP, PAGE, NOP, NOPLF
Application instruction in GX Works2	Application instructions such as DMOD, PCHK, INC(P) ☞ QCPU (Q Mode)/QnACPU Programming Manual (Common Instructions), QCPU Structured Programming Manual (Common Instructions)
SFC instruction	SFCP, SFCPEND, BLOCK, BEND, TRANL, TRANO, TRANA, TRANC, TRANCA, TRANOA, SEND, TRANOC, TRANOCA, TRANCO, TRANCOC, STEPN, STEPD, STEPSC, STEPSE, STEPST, STEPR, STEPC, STEPG, STEPI, STEPID, STEPISC, STEPISE, STEPIST, STEPIR, TRANJ, TRANOJ, TRANOCJ, TRANCJ, TRANCOJ, TRANCOCJ
ST code body	RETURN, IF, THEN, ELSE, ELSIF, END_IF, CASE, OF, END_CASE, FOR, TO, BY, DO, END_FOR, WHILE, END_WHILE, REPEAT, UNTIL, END_REPEAT, EXIT, TYPE, END_TYPE, STRUCT, END_STRUCT, RETAIN, VAR_ACCESS, END_VAR, FUNCTION, END_FUNCTION, FUCTION_BLOCK, END_FUCTION_BLOCK, STEP, INITIAL_STEP, END_STEP, TRANSITION, END_TRANSITION, FROM, TO, UNTILWHILE
Standard function name	Function names in application functions such as AND_E, NOT_E

Table 3.7-2 Reserved words (2/2)

Category	Character string
Standard function block name	Function block names in application functions such as CTD, CTU
Symbol	", %, ', ~, ^, ¦, @, [,] , {, }, ;, :, , , ., ?, \, !, #, $, ', _, *, /, +, <, >, =, &, (,), -
Date and time literal	DATE, DATE_AND_TIME, DT, TIME, TIME_OF_DAY, TOD
Others	ACTION, END_ACTION, CONFIGURATION, END_CONFIGURATION, CONSTANT, F_EDGE, R_EDGE, AT, PROGRAM, WITH, END_PROGRAM, TRUE, FALSE, READ_ONLY, READ_WRITE, RESOURCE, END_RESOURCE, ON, TASK, EN, ENO, BODY_CCE, BODY_FBD, BODY_IL, BODY_LD, BODY_SFC, BODY_ST, END_BODY, END_PARAMETER_SECTION, PARAM_FILE_PATH, PARAMETER_SECTION, SINGLE, TRUE, FALSE, RETAIN, INTERVAL, L, P
String that starts with K1 to K8	Such as K1AAA
Address	Such as %IX0
Statement in ladder language	;FB BLK START, ;FB START, ;FB END, ;FB BLK END, ;FB IN, ;FB OUT, ;FB_NAME;,INSTANCE_NAME, ;FB, ;INSTANCE
Common instruction	Such as MOV
Windows reserved word	COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9, AUX, CON, PRN, NUL

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
Universal model QCPU	High Performance model QCPU	
		The normal icon indicates that the CPU module can use the corresponding functions.
		The icon with △ symbol indicates that the CPU module can use the corresponding functions under certain restrictions. (Function version and software version)
		The icon with × symbol indicates that the CPU module cannot use the corresponding functions.

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 and ST languages.

⑦ Indicates the names of input and output arguments, the data type of each argument, and the availability of direct device setting for each function.

Elementary data type	Type	Description	Direct device setting
Bit	Bool	Uses bit data or bit device. (Data range: 0 (FALSE), 1 (TRUE))	○
Word (signed)	Integer	Uses signed word data. (Data range: -32768 to 32767)	○
Double word (signed)	Double-precision integer	Uses signed double-word data. (Data range: -2147483648 to 2147483647)	×
Word (unsigned)/ 16-bit string	16-bit string	Uses unsigned word data/16-bit BIN data. (Data range: 0 to 65535)	○
Double word (unsigned)/ 32-bit string	32-bit string	Uses unsigned double-word data/32-bit BIN data. (Data range: 0 to 4294967295)	×
Single-precision real	Real	Uses floating-point data. (Data range: -2^{128} to -2^{-126}, 0, 2^{-126} to 2^{128})	×
Double-precision real*1	Double-precision real	Uses floating-point data. (Data range: -2^{1024} to -2^{-1022}, 0, 2^{-1022} to 2^{1024})	×
String	Character string	Uses character string data. (Data range: max. 255 characters)	×
Time*2	Time value	Uses the time type data. (Data range: T#-24d-0h31m23s648ms to T#24d20h31m23s647ms)	×

*1 Can be used for the Universal model QCPU only.
*2 The time type is used in time type operation instructions of application function.

Generic data type	Description
ANY	Can use all elementary data types.
ANY_NUM	Can use elementary data type of single-precision real, double-precision real, word (signed), and double word (signed).
ANY_REAL	Can use elementary data type of single-precision real and double-precision real.
ANY_INT	Can use elementary data type of word (signed) and double word (signed).
ANY_BIT	Can use elementary data type of bit, word (unsigned)/16-bit string, and double word (unsigned)/32-bit string. With word (unsigned)/16-bit string, and double word (unsigned)/32-bit string, bit is specified.
ANY16	Can use elementary data type of word (unsigned)/16-bit string and word (signed).
ANY32	Can use elementary data type of double word (unsigned)/32-bit string and double word (signed).

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 and ST languages.

5 APPLICATION FUNCTIONS

5.1 Type Conversion Functions

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

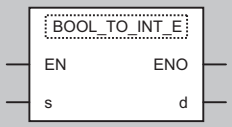
BOOL_TO_INT(_E), BOOL_TO_DINT(_E)

Universal UD High Performance

BOOL_TO_INT(_E)
BOOL_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

BOOL_TO_INT_E indicates any of the following functions.

BOOL_TO_INT	BOOL_TO_INT_E
BOOL_TO_DINT	BOOL_TO_DINT_E

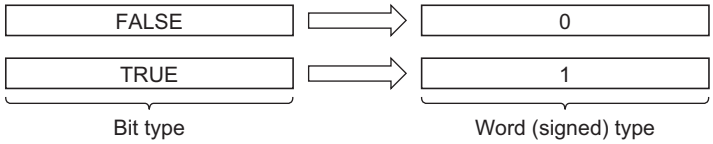
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Bit
Output argument,	ENO:	Output status (TRUE: Normal execution, FALSE: Error or stop)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

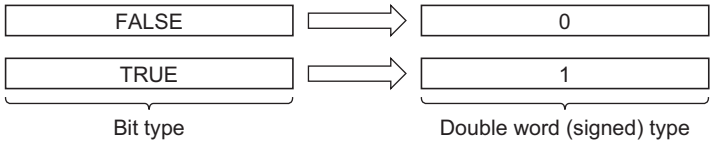
(1) BOOL_TO_INT, BOOL_TO_INT_E

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

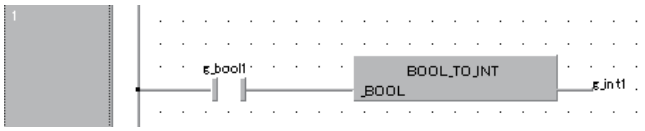
No operation error occurs in the execution of the BOOL_TO_INT(_E) and BOOL_TO_DINT(_E) functions.

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]

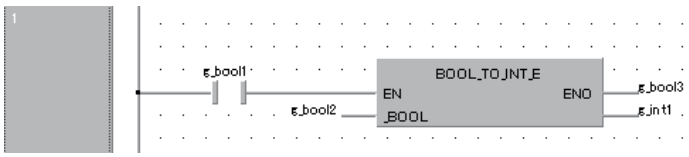


[ST]

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

(b) Function with EN/ENO (BOOL_TO_INT_E)

[Structured ladder]



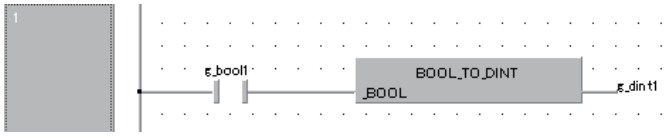
[ST]

```
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]



[ST]

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

5.1.2 Bit type → string type conversion

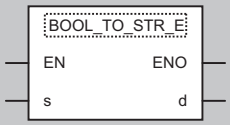
BOOL_TO_STR(_E)

Universal UD High Performance

BOOL_TO_STR(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

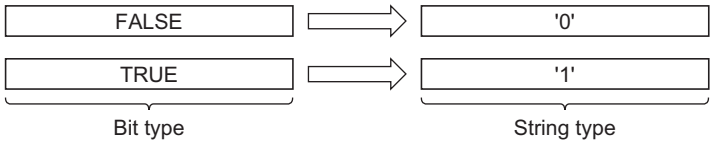
BOOL_TO_STR BOOL_TO_STR_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Bit
Output argument,	ENO:	Output status (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

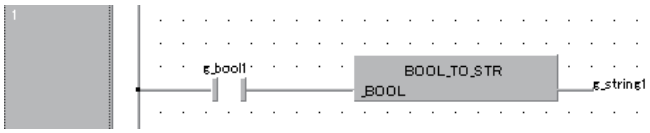
No operation error occurs in the execution of the BOOL_TO_STR(_E) function.

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]



[ST]

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

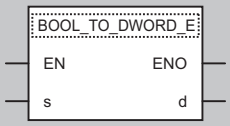
5.1.3 Bit type → word (unsigned)/16-bit string, double word (unsigned)/32-bit string type conversion BOOL_TO_WORD(_E), BOOL_TO_DWORD(_E)

Universal UD High Performance

BOOL_TO_WORD(_E)
BOOL_TO_DWORD(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

[ ] indicates any of the following functions.

BOOL_TO_WORD BOOL_TO_WORD_E
BOOL_TO_DWORD
BOOL_TO_DWORD_E

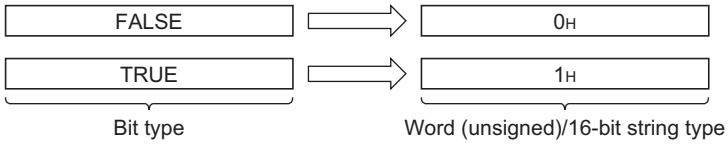
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Bit
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string

Function

Operation processing

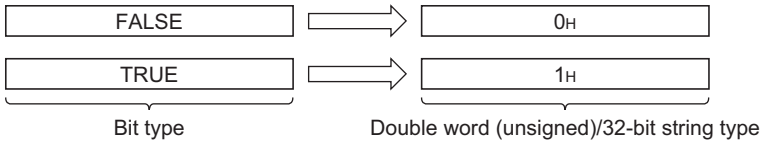
(1) BOOL_TO_WORD, BOOL_TO_WORD_E

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

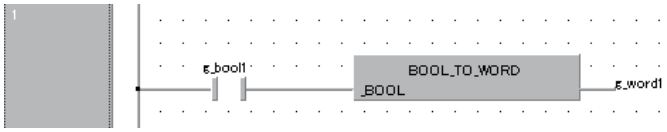
No operation error occurs in the execution of the BOOL_TO_WORD(_E) and BOOL_TO_DWORD(_E) functions.

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]

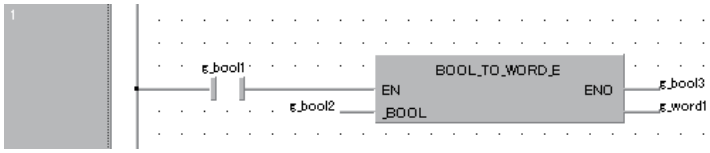


[ST]

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

(b) Function with EN/ENO (BOOL_TO_WORD_E)

[Structured ladder]



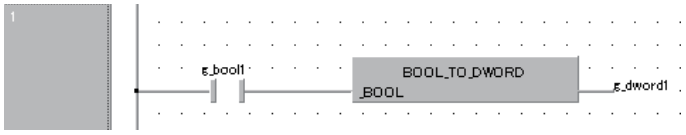
[ST]

```
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]



[ST]

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

5.1.4 Bit type → time type conversion

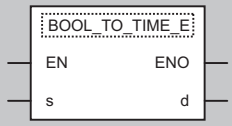
BOOL_TO_TIME(_E)

Universal UD High Performance

BOOL_TO_TIME(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

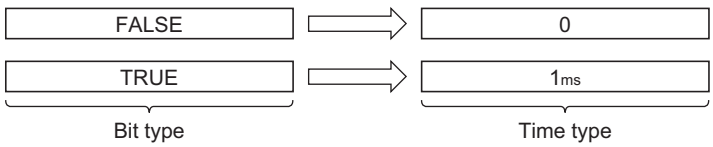
BOOL_TO_TIME BOOL_TO_TIME_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Bit
Output argument,	ENO:	Output status (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

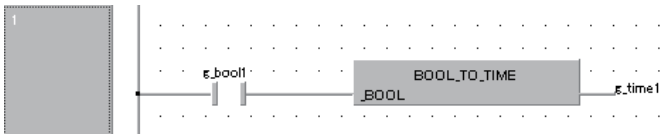
No operation error occurs in the execution of the BOOL_TO_TIME(_E) function.

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]

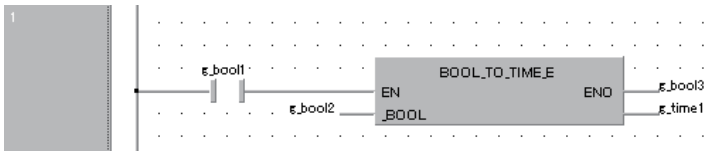


[ST]

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

(b) Function with EN/ENO (BOOL_TO_TIME_E)

[Structured ladder]



[ST]

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

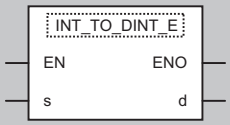
5.1.5 Word (signed) type → double word (signed) type conversion INT_TO_DINT(_E)

Universal UD High Performance

INT_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

[ ] indicates any of the following functions.

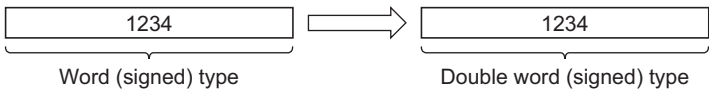
INT_TO_DINT INT_TO_DINT_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (signed)
Output argument,	ENO:	Output status (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

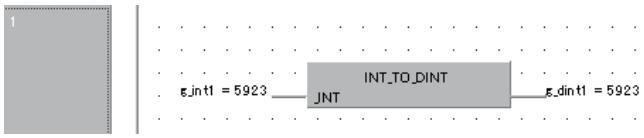
No operation error occurs in the execution of the INT_TO_DINT(_E) function.

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]

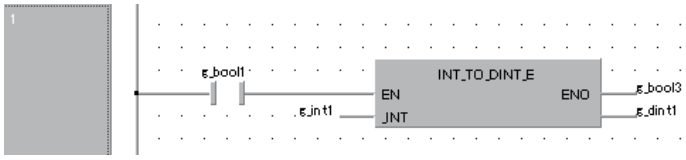


[ST]

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

(b) Function with EN/ENO (INT_TO_DINT_E)

[Structured ladder]



[ST]

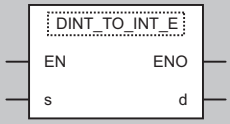
```
g_bool3 := INT_TO_DINT_E (g_bool1, g_int1, g_dint1);
```

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

Universal UD High Performance

DINT_TO_INT(_E) (_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

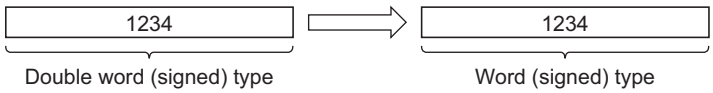
DINT_TO_INT DINT_TO_INT_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Double word (signed)
Output argument,	ENO:	Output status (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_WORD(_E) function is executed, high-order 16-bit data of double word (signed) type data input to ⓢ is discarded.

Operation Error

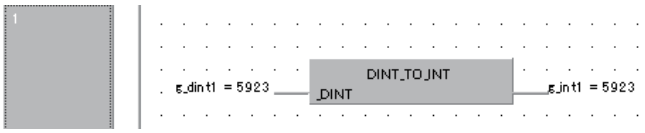
No operation error occurs in the execution of the DINT_TO_INT(_E) function.

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]

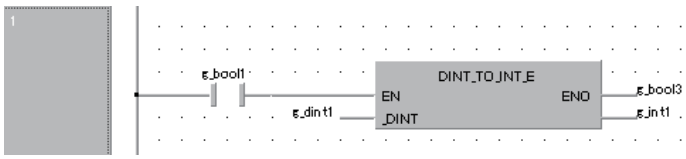


[ST]

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

(b) Function with EN/ENO (DINT_TO_INT_E)

[Structured ladder]



[ST]

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

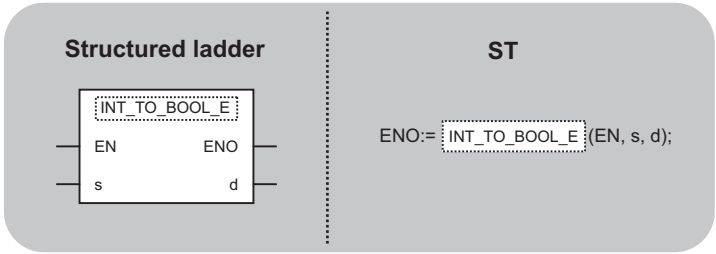
5.1.7 Word (signed), double word (signed) type → bit type conversion

INT_TO_BOOL(_E), DINT_TO_BOOL(_E)

Universal UD High Performance

INT_TO_BOOL(_E)
DINT_TO_BOOL(_E)

(_E: With EN/ENO)



[ ] indicates any of the following functions.

INT_TO_BOOL	INT_TO_BOOL_E
DINT_TO_BOOL	DINT_TO_BOOL_E

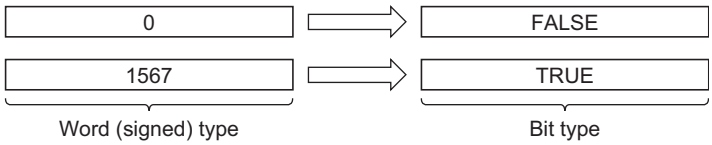
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	;Bit
	s:	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Bit

Function

Operation processing

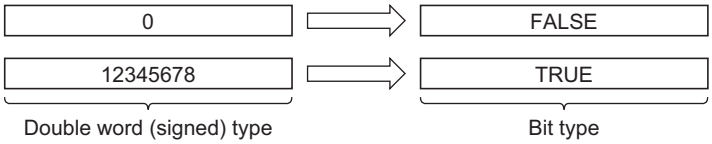
(1) INT_TO_BOOL, INT_TO_BOOL_E

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

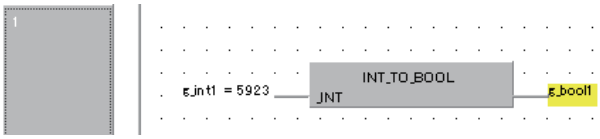
No operation error occurs in the execution of the INT_TO_BOOL(_E) and DINT_TO_BOOL(_E) functions.

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]

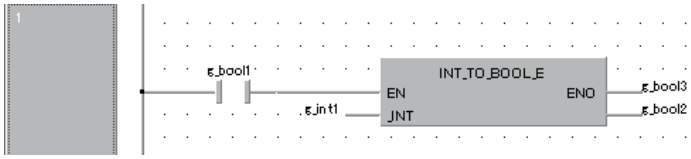


[ST]

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

(b) Function with EN/ENO (INT_TO_BOOL_E)

[Structured ladder]



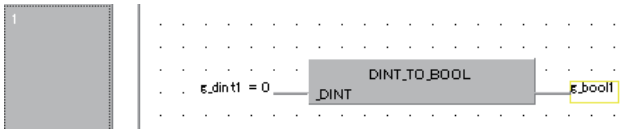
[ST]

```
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]



[ST]

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

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

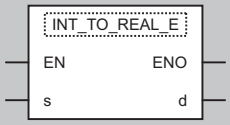
INT_TO_REAL(_E), DINT_TO_REAL(_E)

Universal UD High Performance

INT_TO_REAL(_E)
DINT_TO_REAL(_E)

(_E: With EN/ENO)

Structured ladder



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

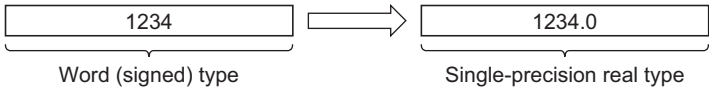
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Single-precision real

Function

Operation processing

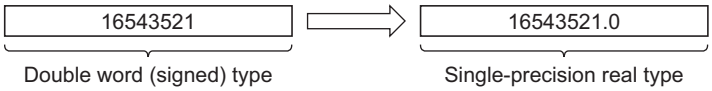
(1) INT_TO_REAL, INT_TO_REAL_E

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

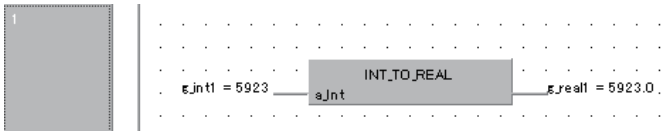
No operation error occurs in the execution of the INT_TO_REAL(_E) and DINT_TO_REAL(_E) functions.

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]

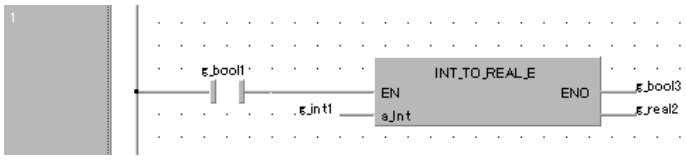


[ST]

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

(b) Function with EN/ENO (INT_TO_REAL_E)

[Structured ladder]



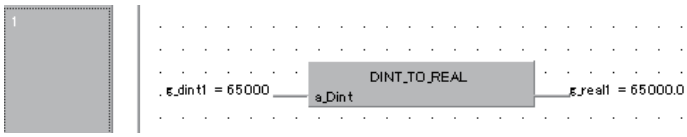
[ST]

```
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]



[ST]

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

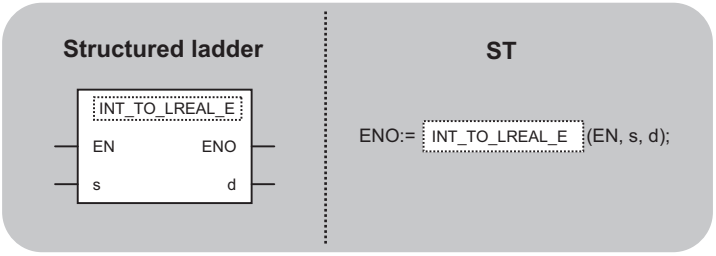
5.1.9 Word (signed), double word (signed) type → double-precision real type conversion

INT_TO_LREAL(_E), DINT_TO_LREAL(_E)

Universal
UD

INT_TO_LREAL(_E)
DINT_TO_LREAL(_E)

(_E: With EN/ENO)



INT_TO_LREAL_E indicates any of the following functions.

INT_TO_LREAL	INT_TO_LREAL_E
DINT_TO_LREAL	DINT_TO_LREAL_E

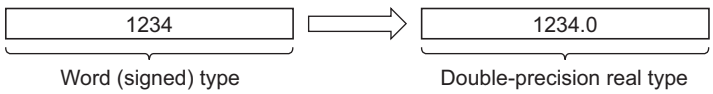
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Double-precision real

Function

Operation processing

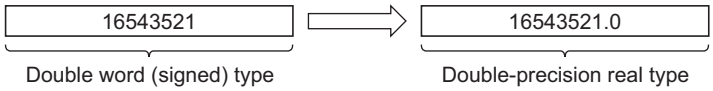
(1) INT_TO_LREAL, INT_TO_LREAL_E

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

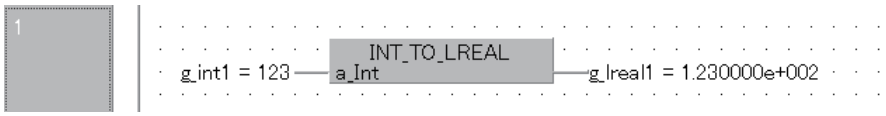
No operation error occurs in the execution of the INT_TO_LREAL(_E) and DINT_TO_LREAL(_E) functions.

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]

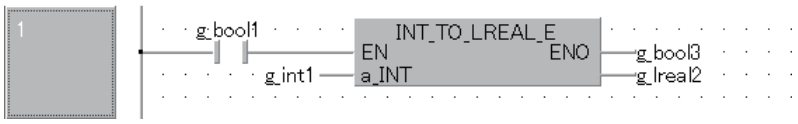


[ST]

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

(b) Function with EN/ENO (INT_TO_LREAL_E)

[Structured ladder]



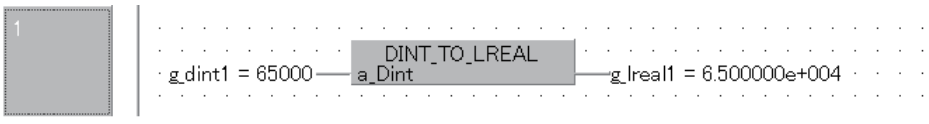
[ST]

```
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]



[ST]

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

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

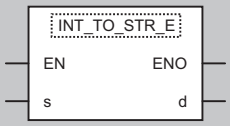
INT_TO_STR(_E), DINT_TO_STR(_E)

Universal UD High Performance

INT_TO_STR(_E)
DINT_TO_STR(_E)

(_E: With EN/ENO)

Structured ladder



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

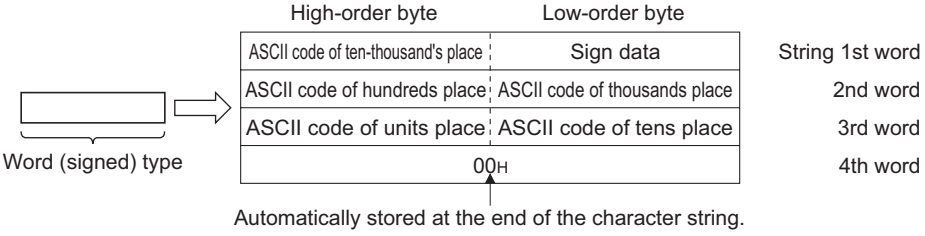
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String(6)

Function

Operation processing

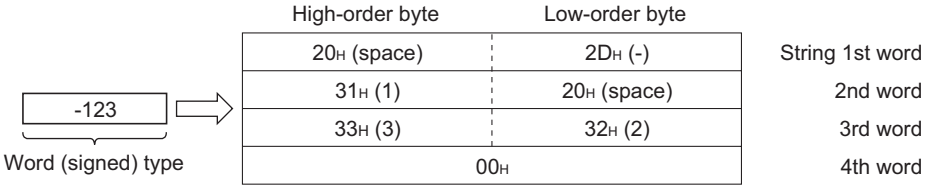
(1) INT_TO_STRING, INT_TO_STRING_E

(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.

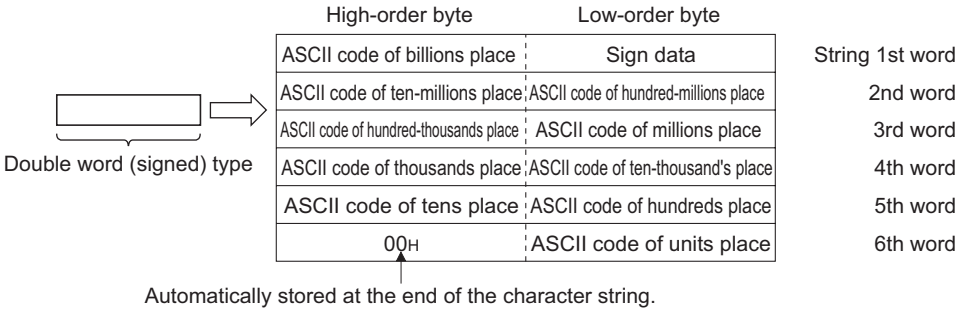
(c) If the number of significant figures is less, '20H (space)' is stored to high-order digits.
(Example) Inputting −123



(d) '00H' is automatically stored at the end of the character string (4th word).

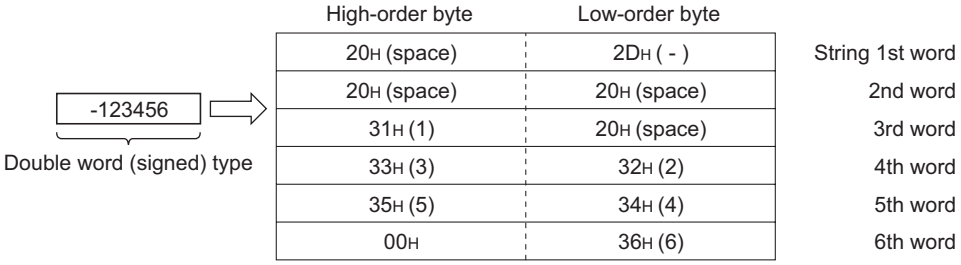
(2) DINT_TO_STRING, DINT_TO_STRING_E

(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) '00H' is automatically stored at the end of the character string (high-order byte in 6th word).

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

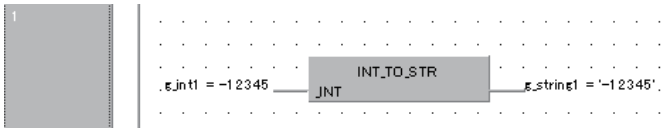
No operation error occurs in the execution of the INT_TO_STR(_E) and DINT_TO_STR(_E) functions.

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]

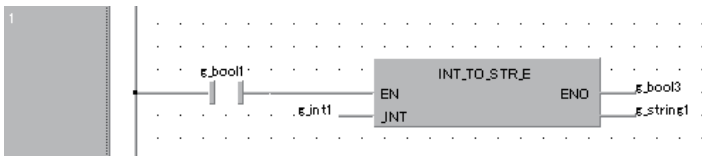


[ST]

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

(b) Function with EN/ENO (INT_TO_STR_E)

[Structured ladder]



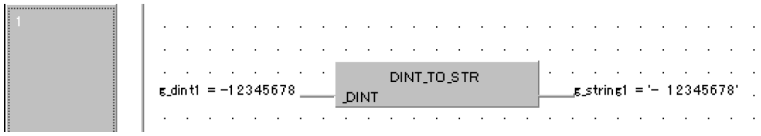
[ST]

```
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]



[ST]

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

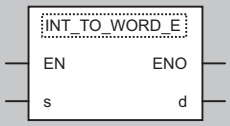
5.1.11 Word (signed), double word (signed) type → word (unsigned)/16-bit string type conversion INT_TO_WORD(_E), DINT_TO_WORD(_E)

Universal UD High Performance

INT_TO_WORD(_E)
DINT_TO_WORD(_E)

(_E: With EN/ENO)

Structured ladder



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

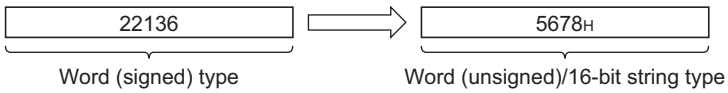
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (unsigned)/16-bit string

Function

Operation processing

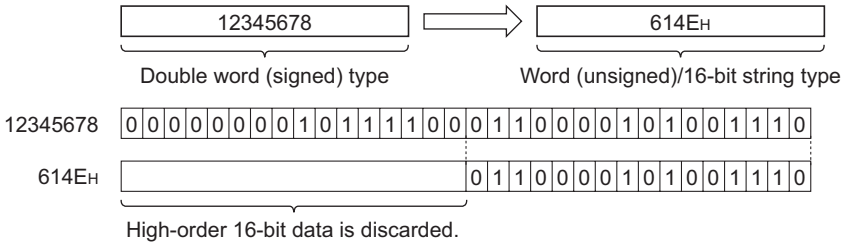
(1) INT_TO_WORD, INT_TO_WORD_E

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, high-order 16-bit data of double word (signed) type data input to input variable ⓢ is discarded.

Operation Error

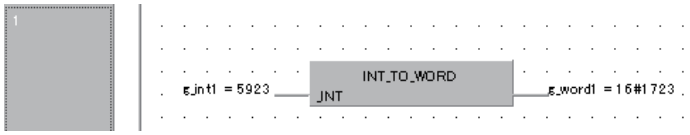
No operation error occurs in the execution of the INT_TO_WORD(_E) and DINT_TO_WORD(_E) functions.

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]

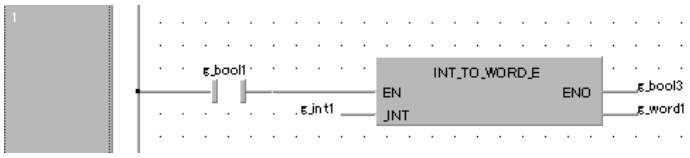


[ST]

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

(b) Function with EN/ENO (INT_TO_WORD_E)

[Structured ladder]



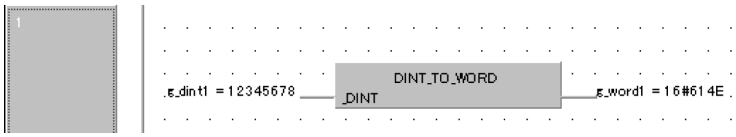
[ST]

```
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]



[ST]

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

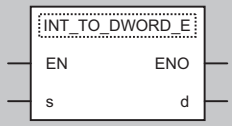
5.1.12 Word (signed), double word (signed) type → double word (unsigned)/32-bit string type conversion INT_TO_DWORD(_E), DINT_TO_DWORD(_E)

Universal UD High Performance

INT_TO_DWORD(_E)
DINT_TO_DWORD(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

INT_TO_DWORD_E indicates any of the following functions.

INT_TO_DWORD	INT_TO_DWORD_E
DINT_TO_DWORD	DINT_TO_DWORD_E

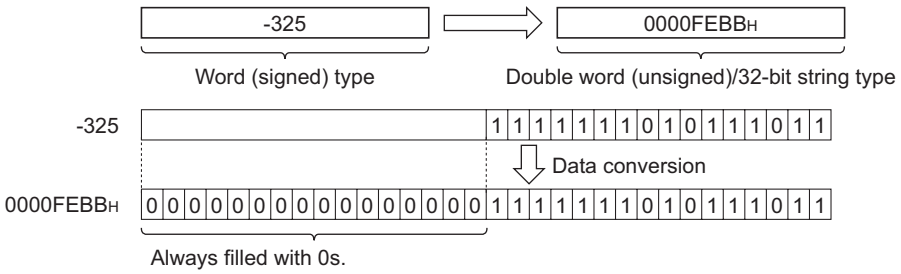
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Double word (unsigned)/32-bit string

Function

Operation processing

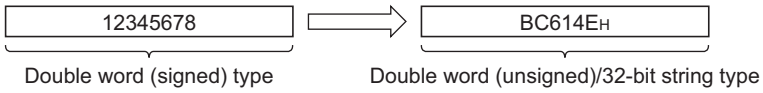
(1) INT_TO_DWORD, INT_TO_DWORD_E

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

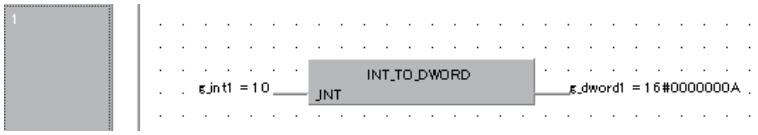
No operation error occurs in the execution of the INT_TO_DWORD(_E) and DINT_TO_DWORD(_E) functions.

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]

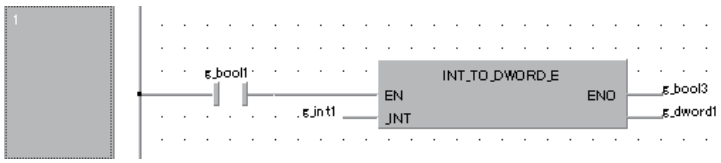


[ST]

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

(b) Function with EN/ENO (INT_TO_DWORD_E)

[Structured ladder]



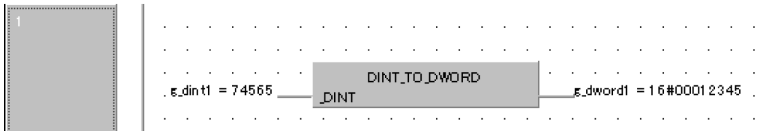
[ST]

```
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]



[ST]

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

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

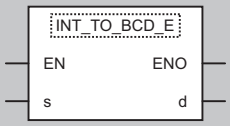
INT_TO_BCD(_E), DINT_TO_BCD(_E)

Universal UD High Performance

INT_TO_BCD(_E)
DINT_TO_BCD(_E)

(_E: With EN/ENO)

Structured ladder



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

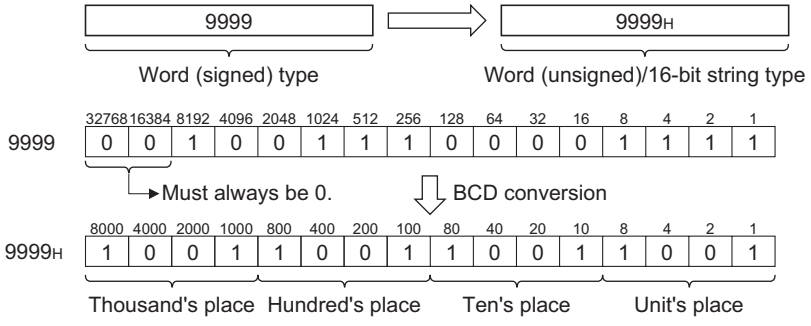
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (signed), double word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string

Function

Operation processing

(1) INT_TO_BCD, INT_TO_BCD_E

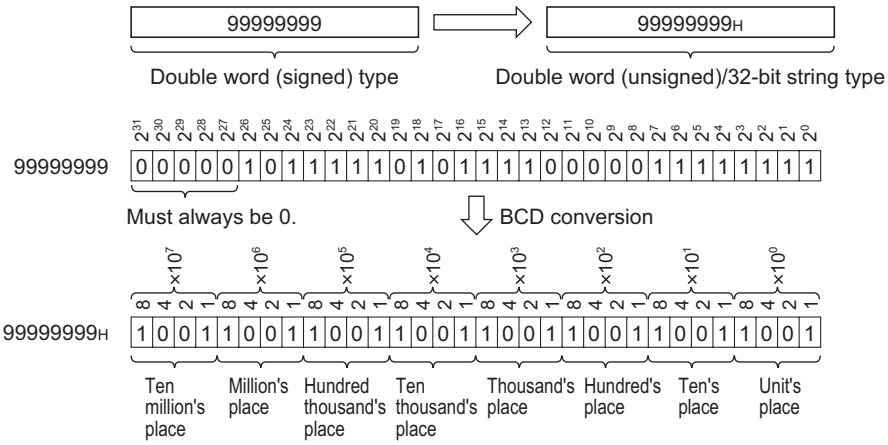
(a) Converts word (signed) type data input to ⓢ into BCD type data, and outputs the operation result from ⓓ.



(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.

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 error)*1	Undefined 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

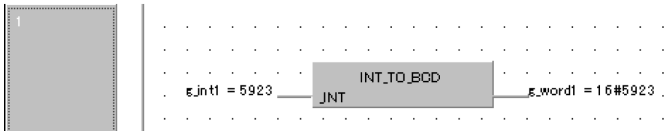
An operation error occurs when the value input exceeds 9999 or 99999999 respectively in the execution of the INT_TO_BCD(_E) or DINT_TO_BCD(_E) function. (Error code: 4100)

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]

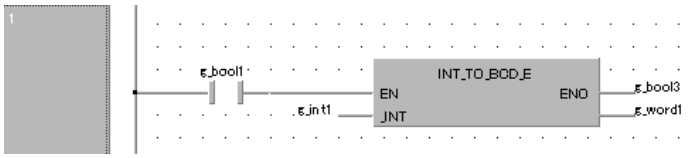


[ST]

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

(b) Function with EN/ENO (INT_TO_BCD_E)

[Structured ladder]



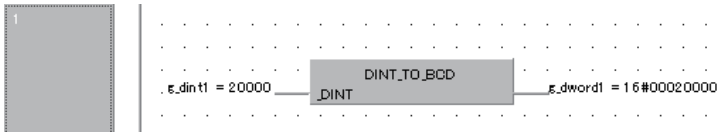
[ST]

```
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]



[ST]

```
s_dword1:= DINT_TO_BCD(g_dint1);
```

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

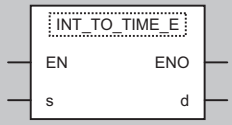
INT_TO_TIME(_E), DINT_TO_TIME(_E)

Universal UD High Performance

INT_TO_TIME(_E)
DINT_TO_TIME(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

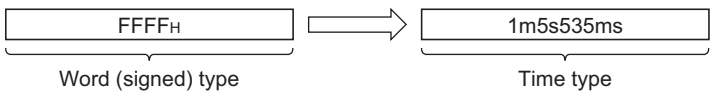
INT_TO_TIME	INT_TO_TIME_E
DINT_TO_TIME	DINT_TO_TIME_E

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

Function

Operation processing

Converts 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*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

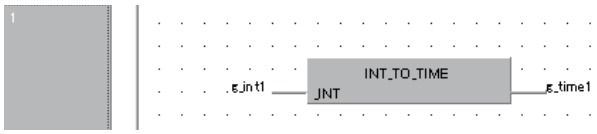
No operation error occurs in the execution of the INT_TO_TIME(_E) and DINT_TO_TIME(_E) functions.

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]

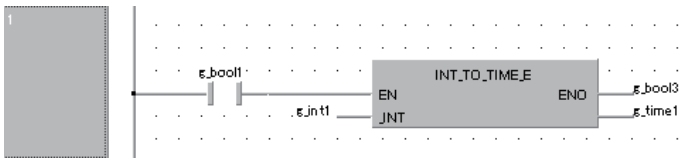


[ST]

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

(b) Function with EN/ENO (INT_TO_TIME_E)

[Structured ladder]



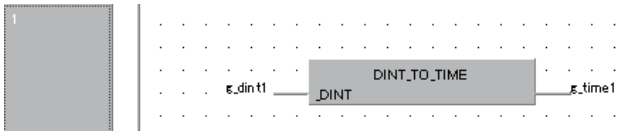
[ST]

```
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]



[ST]

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

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

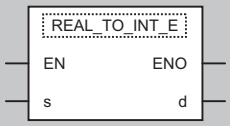
REAL_TO_INT(_E), REAL_TO_DINT(_E)

Universal UD High Performance

REAL_TO_INT(_E)
REAL_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder



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

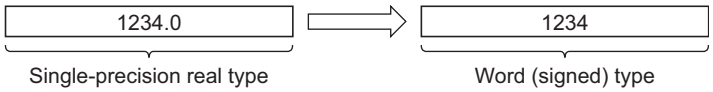
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Single-precision real
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (signed), double word (signed)

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 ⓓ.

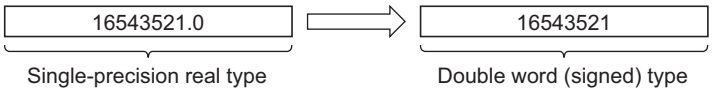


(b) The value to be input to ⓢ is single-precision real type data, within the range from −32768 to 32767.

(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.

(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 error)*1	Undefined 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.

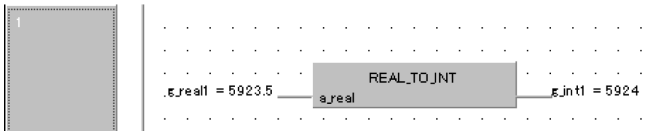
- REAL_TO_INT(_E): The input value is outside the range of −32768 to 32767. (Error code: 4100)
- REAL_TO_DINT(_E): The input value is outside the range of −2147483648 to 2147483647. (Error code: 4100)

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]

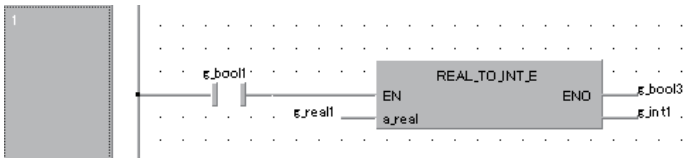


[ST]

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

(b) Function with EN/ENO (REAL_TO_INT_E)

[Structured ladder]



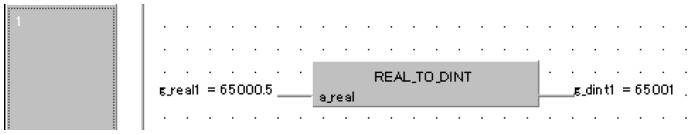
[ST]

```
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]



[ST]

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

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

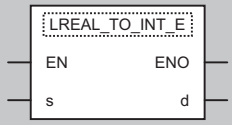
LREAL_TO_INT(_E), LREAL_TO_DINT(_E)

Universal
UD

LREAL_TO_INT(_E)
LREAL_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder



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

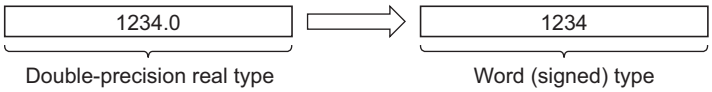
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Double-precision real
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (signed), double word (signed)

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 ⓓ.

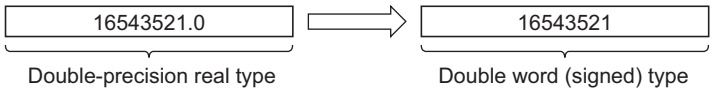


(b) The value to be input to ⓢ is double-precision real type data, within the range from −32768 to 32767.

(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.

(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 error)*1	Undefined 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.

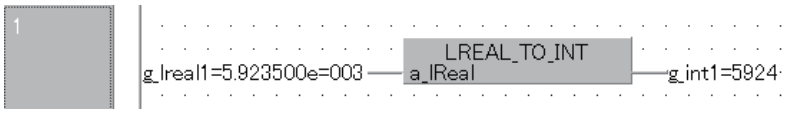
- The input value is -0 or outside the following range. (Error code: 4140)
  0, $2^{-1022} \leqq |$ ⓢ , ⓓ $| < 2^{1024}$
- LREAL_TO_INT(_E): The input value is outside the range of -32768 to 32767. (Error code: 4140)
- LREAL_TO_DINT(_E): The input value is outside the range of -2147483648 to 2147483647.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

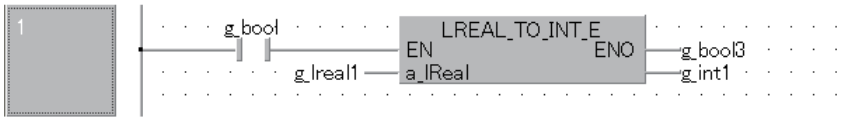


[ST]

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

(b) Function with EN/ENO (LREAL_TO_INT_E)

[Structured ladder]



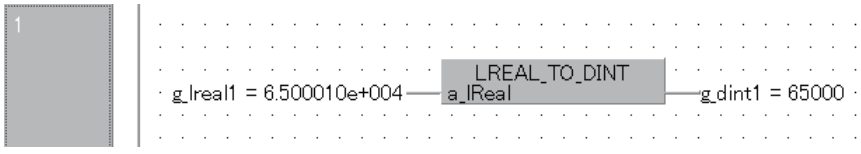
[ST]

```
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]



[ST]

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

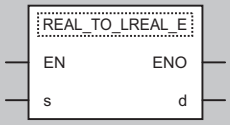
5.1.17 Single-precision real type → double-precision real type conversion

REAL_TO_LREAL(_E)

Universal
UD

REAL_TO_LREAL(_E) (_E: With EN/ENO)

Structured ladder



ST

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

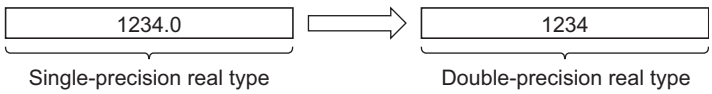
REAL_TO_LREAL_E indicates any of the following functions.
REAL_TO_LREAL REAL_TO_LREAL_E

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

Function

Operation processing

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



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 error)*1	Undefined 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.

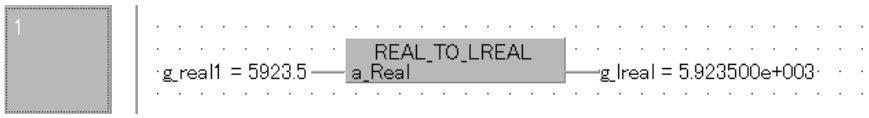
- The input value is -0 or outside the following range. (Error code: 4140)
  0, $2^{-126} \leqq$ | ⓢ | $< 2^{128}$
- The operation result is outside the following range (an overflow occurrence). (Error code: 4141)
  $2^{1024} \leqq$ | operation result |

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]

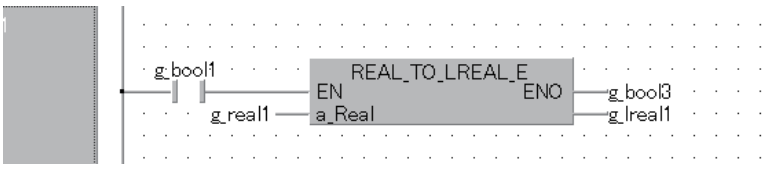


[ST]

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

(b) Function with EN/ENO (REAL_TO_LREAL_E)

[Structured ladder]



[ST]

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

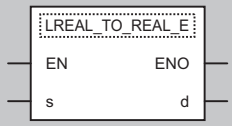
5.1.18 Double-precision real type → single-precision real type conversion

LREAL_TO_REAL(_E)

Universal
UD

LREAL_TO_REAL(_E) (_E: With EN/ENO)

Structured ladder



ST

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

[ ] indicates any of the following functions.

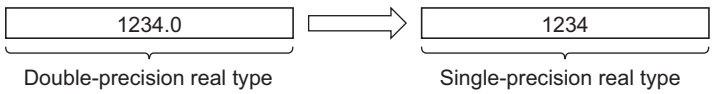
LREAL_TO_REAL LREAL_TO_REAL_E

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

Function

Operation processing

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



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 error)*1	Undefined 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)

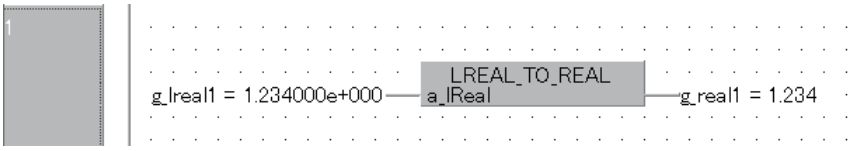
  $2^{128} \leqq$ | operation result |

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]

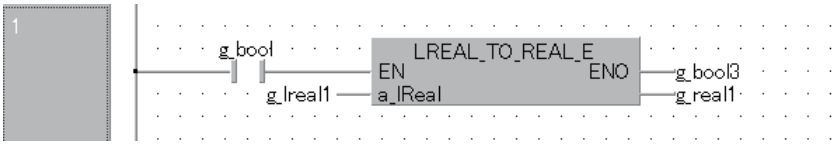


[ST]

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

(b) Function with EN/ENO (LREAL_TO_REAL_E)

[Structured ladder]



[ST]

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

5.1.19 Single-precision real type → string type conversion

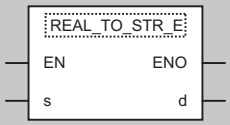
REAL_TO_STR(_E)

Universal UD High Performance

REAL_TO_STR(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

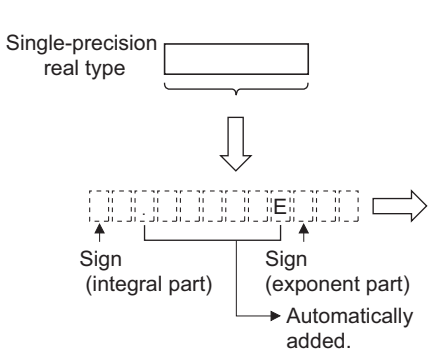
REAL_TO_STR REAL_TO_STR_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Single-precision real
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String(12)

Function

Operation processing

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

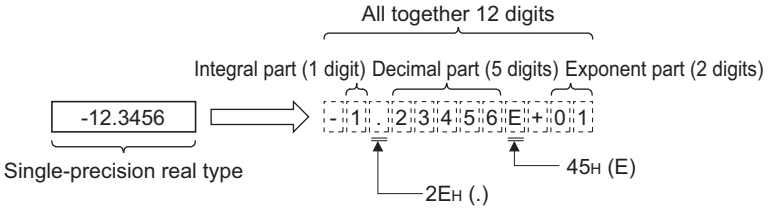


	High-order byte	Low-order byte	
	ASCII code of integral part	Sign data (integral part)	String 1st word
	ASCII code of one decimal place	ASCII code of (.) decimal place (2EH)	2nd word
	ASCII code of three decimal place	ASCII code of two decimal place	3rd word
	ASCII code of five decimal place	ASCII code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

Automatically stored at the end of the character string.

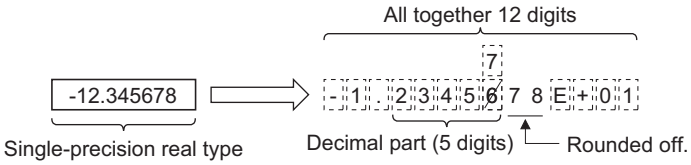
(2) The character string data after conversion is output from output variable ⓓ in the following manner.

(a) The number of digits is fixed respectively for the integral part, decimal part, and exponent part. (Integral part: 1 digit, decimal part: 5 digits, exponent part: 2 digits) '2EH' (.) and '45H' (E) are automatically stored in the 3rd and 9th bytes, respectively.

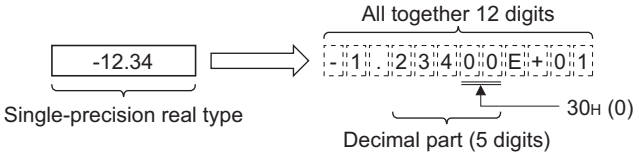


(b) '20H' (space) is stored in 'Sign data' (integral part) when the input value is positive; '2DH' (−) is stored when negative

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

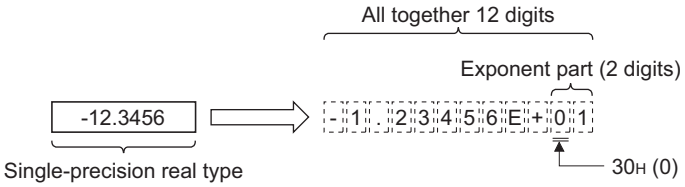


(d) If the number of significant figures is less, '30H' (0) is stored to decimal part.



(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) '00H' is automatically stored to the end of the character string (7th word).

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 error)*1	Undefined 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.

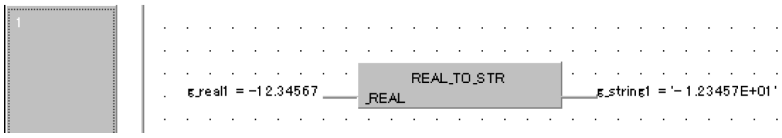
- The input value is outside the range of $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$. (Error code: 4100)

Program Example

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

(a) Function without EN/ENO (REAL_TO_STR)

[Structured ladder]

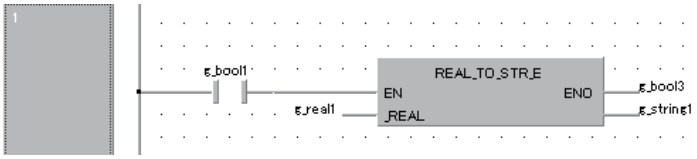


[ST]

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

(b) Function with EN/ENO (REAL_TO_STR_E)

[Structured ladder]



[ST]

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

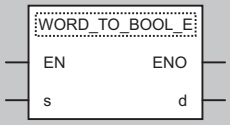
5.1.20 Word (unsigned)/16-bit string, double word (unsigned)/32-bit string type → bit type conversion WORD_TO_BOOL(_E), DWORD_TO_BOOL(_E)

Universal UD High Performance

WORD_TO_BOOL(_E)
DWORD_TO_BOOL(_E)

(_E: With EN/ENO)

Structured ladder



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

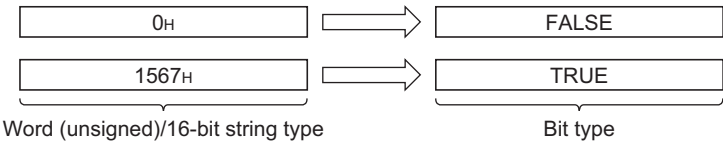
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Bit

Function

Operation processing

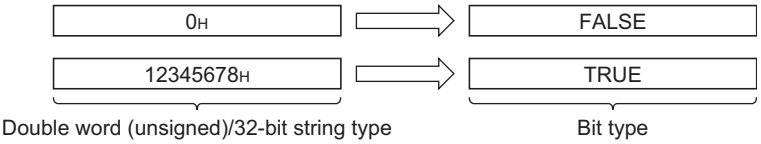
(1) WORD_TO_BOOL, WORD_TO_BOOL_E

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

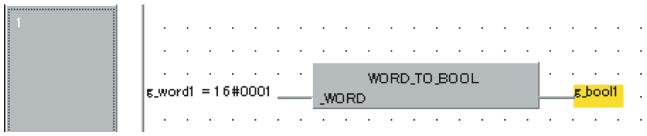
No operation error occurs in the execution of the WORD_TO_BOOL(_E) and DWORD_TO_BOOL(_E) functions.

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]

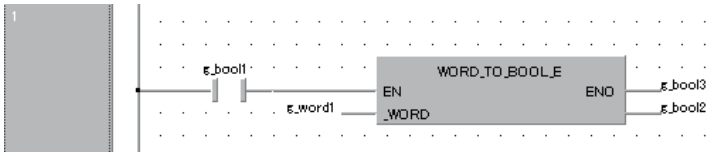


[ST]

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

(b) Function with EN/ENO (WORD_TO_BOOL_E)

[Structured ladder]



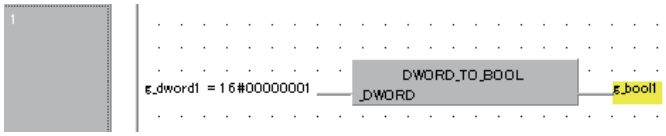
[ST]

```
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]



[ST]

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

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

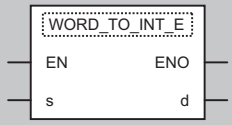
WORD_TO_INT(_E), WORD_TO_DINT(_E)

Universal UD High Performance

WORD_TO_INT(_E)
WORD_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder



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

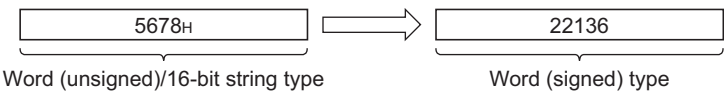
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (unsigned)/16-bit string
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

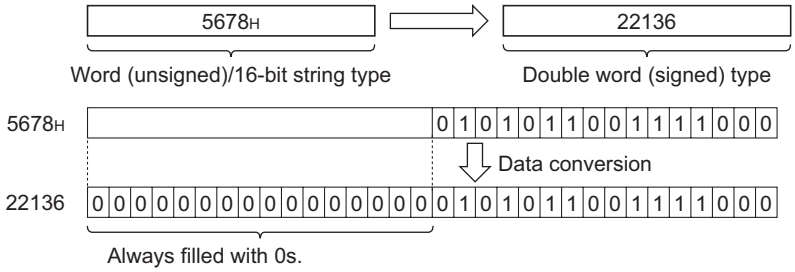
(1) WORD_TO_INT, WORD_TO_INT_E

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

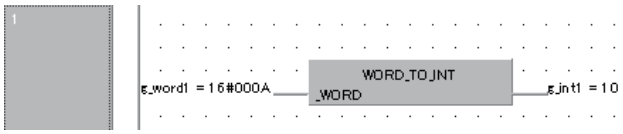
No operation error occurs in the execution of the WORD_TO_INT(_E) and WORD_TO_DINT(_E) functions.

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]

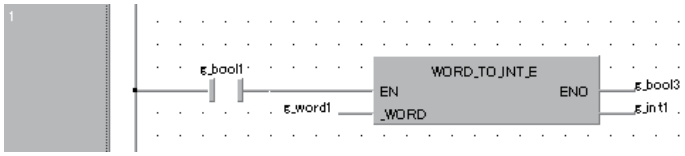


[ST]

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

(b) Function with EN/ENO (WORD_TO_INT_E)

[Structured ladder]



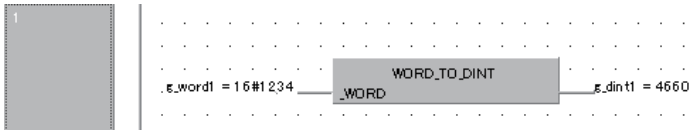
[ST]

```
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]



[ST]

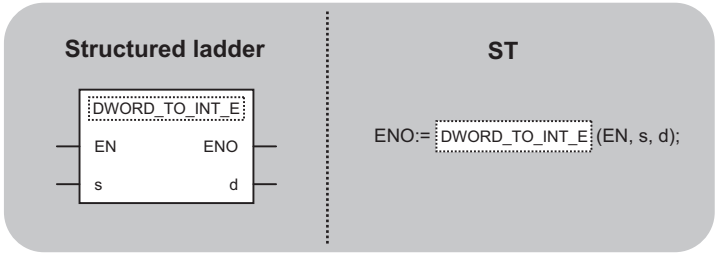
```
g_dint1:= WORD_TO_DINT(g_word1);
```

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)

Universal UD High Performance

DWORD_TO_INT(_E)
DWORD_TO_DINT(_E)

(_E: With EN/ENO)



DWORD_TO_INT_E indicates any of the following functions.

DWORD_TO_INT	DWORD_TO_INT_E
DWORD_TO_DINT	DWORD_TO_DINT_E

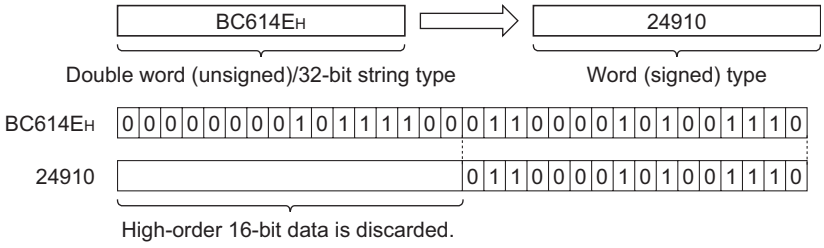
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Double word (unsigned)/32-bit string
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

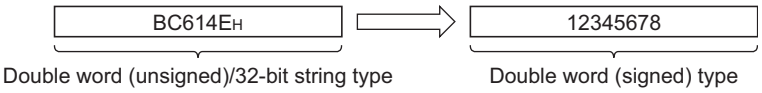
(1) DWORD_TO_INT, DWORD_TO_INT_E

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_WORD(_E) function is executed, high-order 16-bit data of double word (unsigned)/32-bit string type data input to ⓢ is discarded.

Operation Error

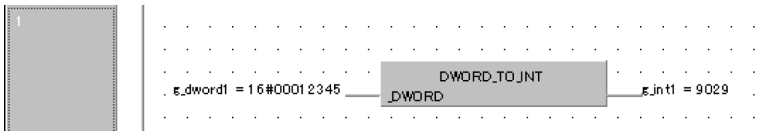
No operation error occurs in the execution of the DWORD_TO_INT(_E) and DWORD_TO_DINT(_E) functions.

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]

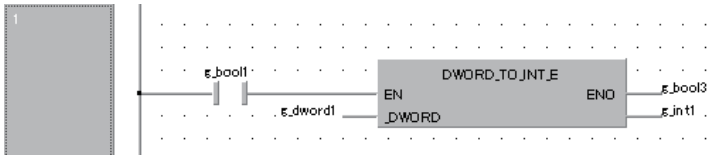


[ST]

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

(b) Function with EN/ENO (DWORD_TO_INT_E)

[Structured ladder]



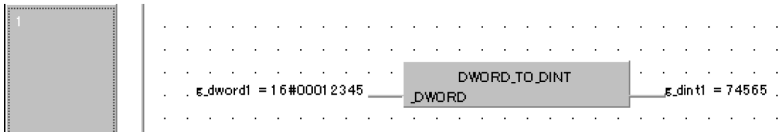
[ST]

```
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]



[ST]

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

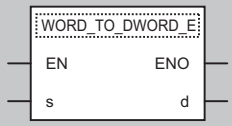
5.1.23 Word (unsigned)/16-bit string type → double word (unsigned)/32-bit string type conversion

WORD_TO_DWORD(_E)

Universal UD High Performance

WORD_TO_DWORD(_E)	_E: With EN/ENO

Structured ladder



ST

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

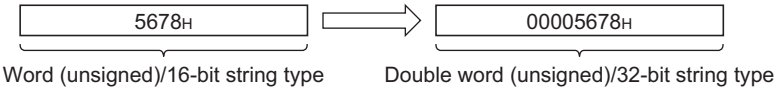
[ ] indicates any of the following functions.
WORD_TO_DWORD
WORD_TO_DWORD_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (unsigned)/16-bit string
Output argument,	ENO:	Output status (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

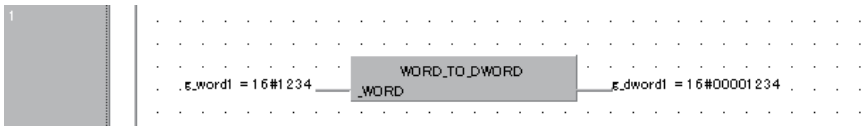
No operation error occurs in the execution of the WORD_TO_DWORD(_E) function.

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]

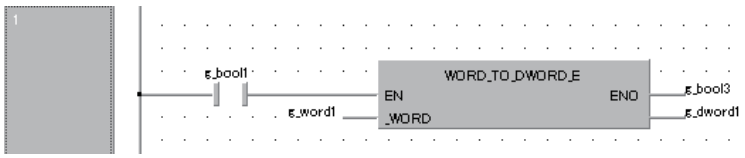


[ST]

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

(b) Function with EN/ENO (WORD_TO_DWORD_E)

[Structured ladder]



[ST]

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

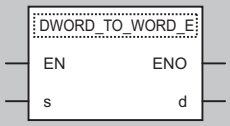
5.1.24 Double word (unsigned)/32-bit string type → word (unsigned)/16-bit string type conversion

DWORD_TO_WORD(_E)

Universal UD High Performance

DWORD_TO_WORD(_E) (_E: With EN/ENO)

Structured ladder



ST

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

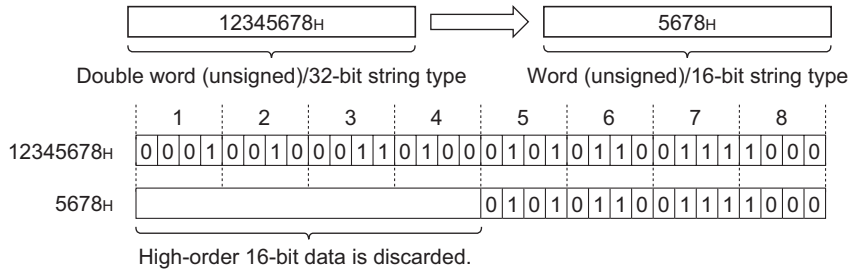
[ ] indicates any of the following functions.
DWORD_TO_WORD
DWORD_TO_WORD_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Double word (unsigned)/32-bit string
Output argument,	ENO:	Output status (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, high-order 16-bit data of double word (unsigned)/32-bit string type data input to ⓢ is discarded.

Operation Error

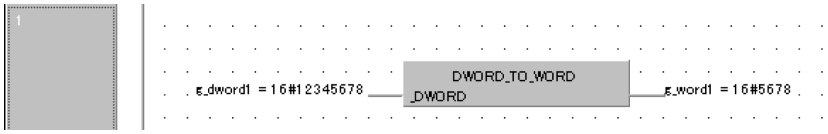
No operation error occurs in the execution of the DWORD_TO_WORD(_E) function.

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]

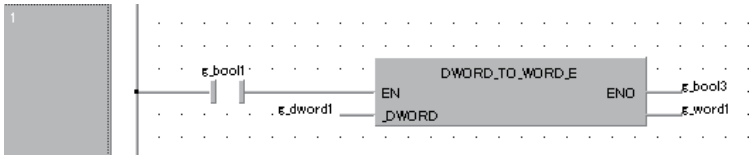


[ST]

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

(b) Function with EN/ENO (DWORD_TO_WORD_E)

[Structured ladder]



[ST]

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

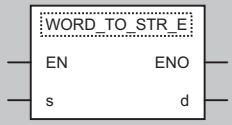
5.1.25 Word (unsigned)/16-bit string, double word (unsigned)/32-bit string type → string type conversion WORD_TO_STR(_E), DWORD_TO_STR(_E)

Universal UD High Performance

WORD_TO_STR(_E)
DWORD_TO_STR(_E)

(_E: With EN/ENO)

Structured ladder



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

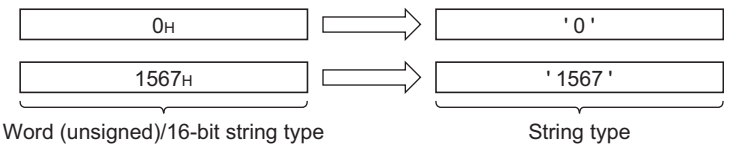
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String

Function

Operation processing

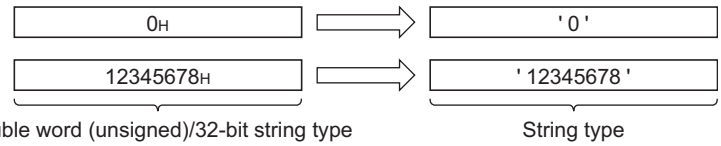
(1) WORD_TO_STRING, WORD_TO_STRING_E

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



(2) DWORD_TO_STRING, DWORD_TO_STRING_E

Converts double word (unsigned)/32-bit string type data input to ⓢ into 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

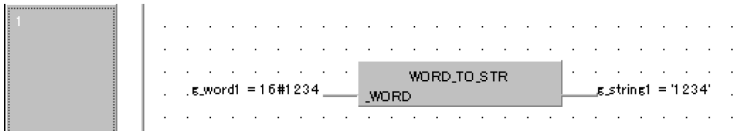
No operation error occurs in the execution of the WORD_TO_STR(_E) and DWORD_TO_STR(_E) functions.

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]

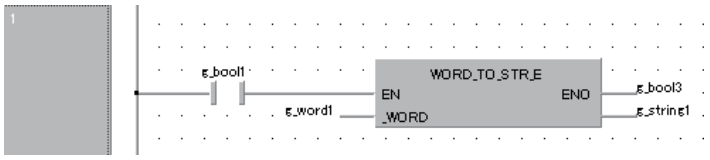


[ST]

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

(b) Function with EN/ENO (WORD_TO_STR_E)

[Structured ladder]



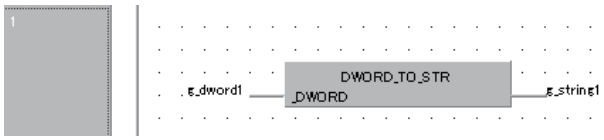
[ST]

```
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]



[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

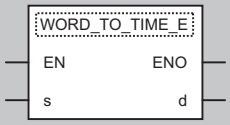
WORD_TO_TIME(_E), DWORD_TO_TIME(_E)

Universal UD High Performance

WORD_TO_TIME(_E)
DWORD_TO_TIME(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

WORD_TO_TIME_E indicates any of the following functions.

WORD_TO_TIME
WORD_TO_ TIME _E
DWORD_TO_TIME
DWORD_TO_ TIME _E

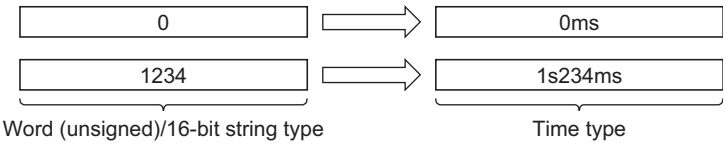
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Time

Function

Operation processing

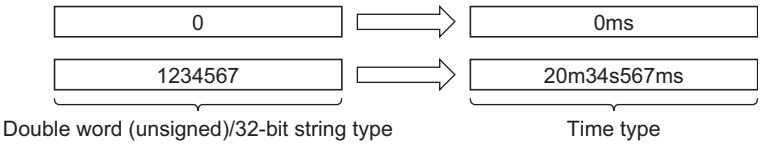
(1) WORD_TO_TIME, WORD_TO_TIME_E

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

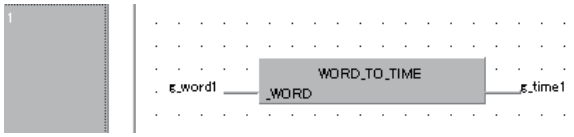
No operation error occurs in the execution of the WORD_TO_TIME(_E) and DWORD_TO_TIME(_E) functions.

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]

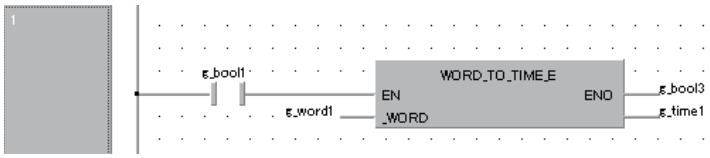


[ST]

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

(b) Function with EN/ENO (WORD_TO_TIME_E)

[Structured ladder]



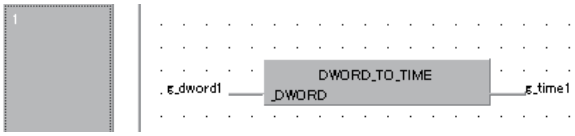
[ST]

```
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]



[ST]

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

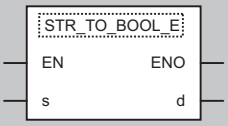
5.1.27 String type → bit type conversion

STR_TO_BOOL(_E)

Universal UD High Performance

STR_TO_BOOL(_E) (_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

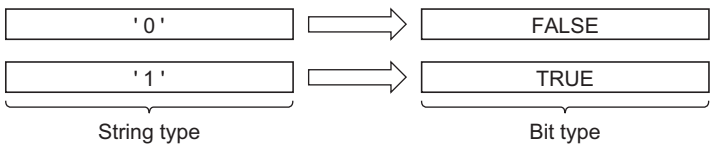
STR_TO_BOOL	STR_TO_BOOL_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:String
Output argument,	ENO:	Output status (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

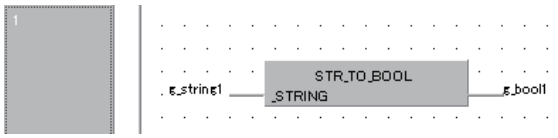
No operation error occurs in the execution of the STR_TO_BOOL(_E) function.

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]

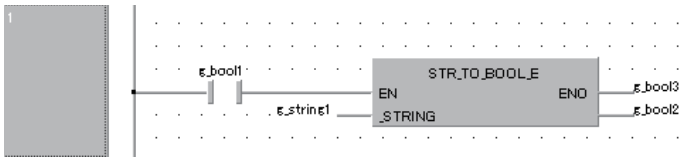


[ST]

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

(b) Function with EN/ENO (STR_TO_BOOL_E)

[Structured ladder]



[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

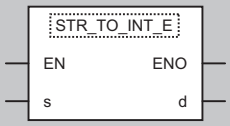
STR_TO_INT(_E), STR_TO_DINT(_E)

Universal UD High Performance

STR_TO_INT(_E)
STR_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder



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:	Input	:String
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (signed), double word (signed)

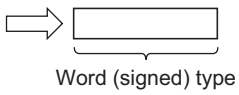
Function

Operation processing

(1) STRING_TO_INT, STRING_TO_INT_E

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

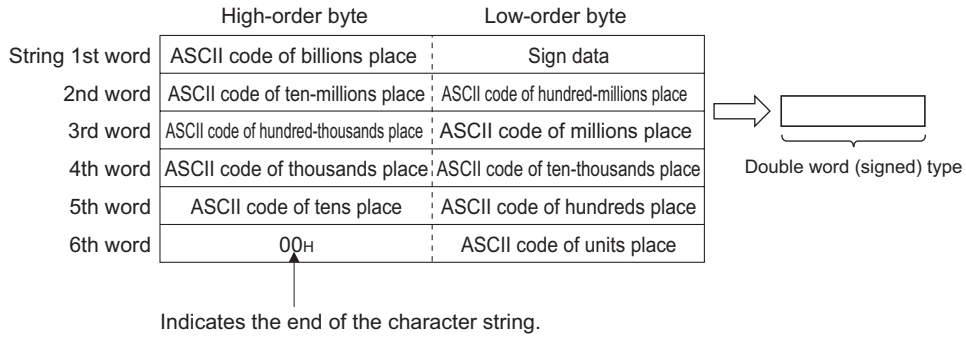
	High-order byte	Low-order byte
String 1st word	ASCII code of ten-thousands place	Sign data
2nd word	ASCII code of hundreds place	ASCII code of thousands place
3rd word	ASCII code of units place	ASCII code of tens place
4th word	00H (Indicates the end of the character string.)	



(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) STRING_TO_DINT, STRING_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 error)*1	Undefined 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.

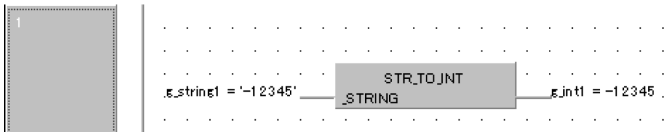
- The input value is other than '30H' to '39H', '20H', '2DH', and '00H' of ASCII code. (Error code: 4100)
- The input value is outside the following ranges of ASCII code. (Error code: 4100)
  STR_TO_INT(_E): '−32768 to 32767'
  STR_TO_DINT(_E): '−2147483648 to 2147483647'

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]

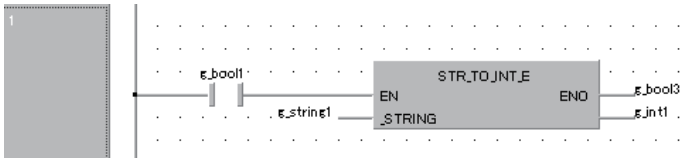


[ST]

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

(b) Function with EN/ENO (STR_TO_INT_E)

[Structured ladder]



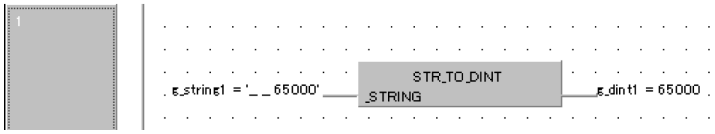
[ST]

```
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]



[ST]

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

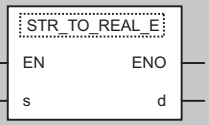
5.1.29 String type → single-precision real type conversion

STR_TO_REAL(_E)

Universal UD High Performance

STR_TO_REAL(_E) (_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

STR_TO_REAL STR_TO_REAL_E

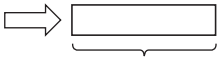
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:String
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Single-precision real

Function

Operation processing

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

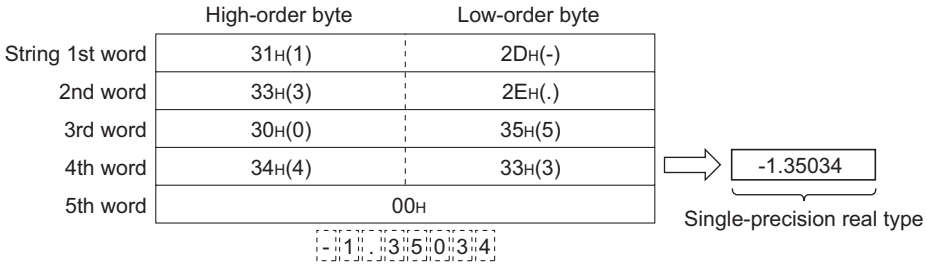
	High-order byte	Low-order byte
String 1st word	ASCII code of 1st character	Sign date
2nd word	ASCII code of 3rd character	ASCII code of 2nd character
3rd word	ASCII code of 5th character	ASCII code of 4th character
4th word	ASCII code of 7th character	ASCII code of 6th character
5th word	ASCII code of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



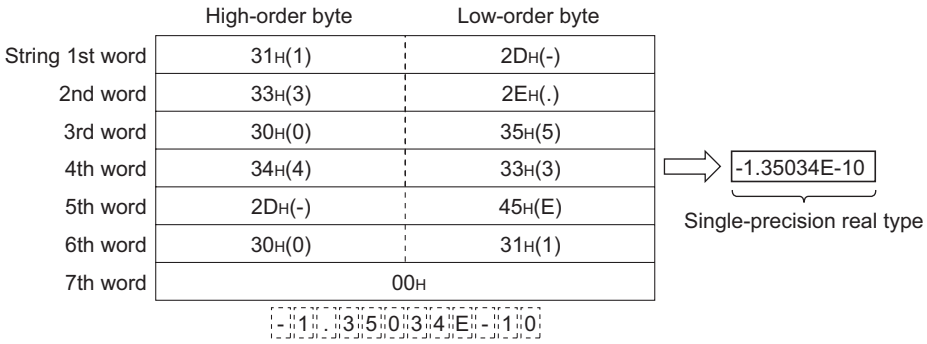
Single-precision real type

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

(a) Decimal point form

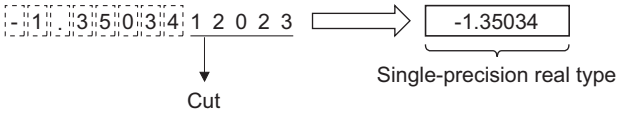


(b) Exponent form

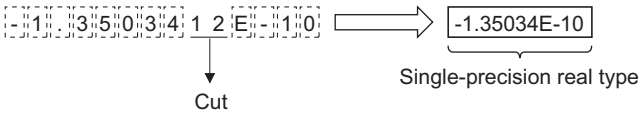


(3) As the number of significant figures of string type data is 6, the 7th and later digits excluding the sign, decimal point, and exponent part are cut and converted.

(a) Decimal point form



(b) Exponent form



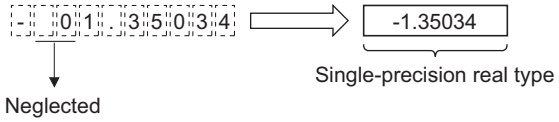
(4) When a sign is not specified or '2BH' (+) is specified for a sign in decimal point form, string type data is converted as a positive value. When '2DH' (−) is specified for a sign, string type data is converted as a negative value.

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

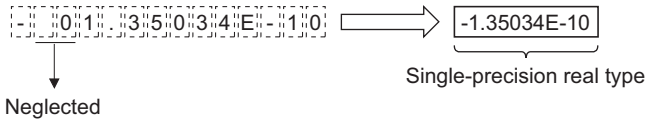
When '2DH' (−) is specified for a sign of the exponential part, string type data is converted as a negative value.

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

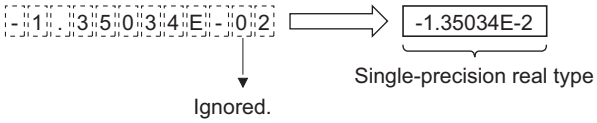
(a) Decimal point form



(b) Exponent form



(7) When '30H (0)' exists between 'E' and a numeric value in string type data (exponent 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 error)*1	Undefined 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.

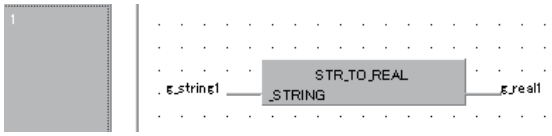
- Any characters other than 30H to 39H exist in the integral or decimal part. (Error code: 4100)
- Two or more 2EH exist. (Error code: 4100)
- Any characters other than '45H(E), 2CH(+)' or '45H(E), 2DH(−)' exist in the exponent part, or more than one exponential parts exist. (Error code: 4100)
- The data after conversion is 0 or outside the range of $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$ (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]

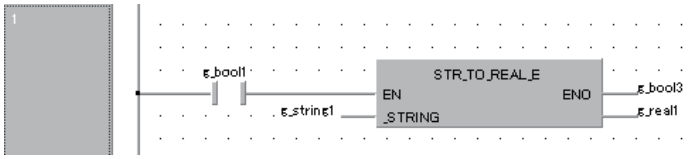


[ST]

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

(b) Function with EN/ENO (STR_TO_REAL_E)

[Structured ladder]



[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

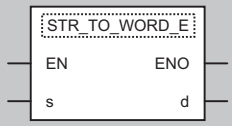
STR_TO_WORD(_E), STR_TO_DWORD(_E)

Basic High Performance

STR_TO_WORD(_E)
STR_TO_DWORD(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

[ ] indicates any of the following functions.

STR_TO_WORD	STR_TO_WORD_E
STR_TO_DWORD	STR_TO_DWORD_E

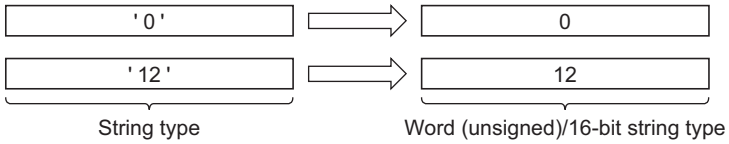
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:String
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string

Function

Operation processing

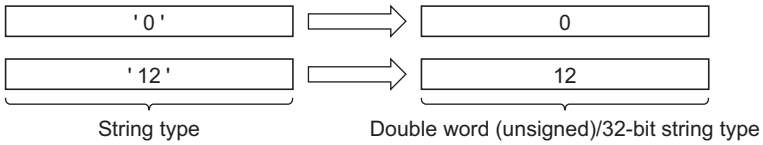
(1) STR_TO_WORD, STR_TO_WORD_E

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 ⓓ .
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

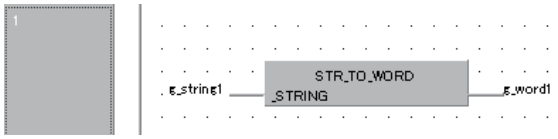
No operation error occurs in the execution of the STR_TO_WORD(_E) and STR_TO_DWORD(_E) functions.

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]

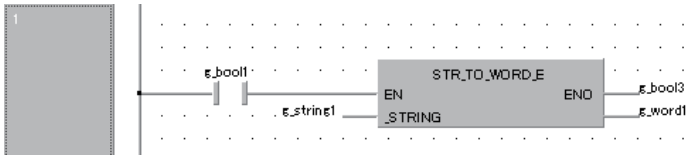


[ST]

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

(b) Function with EN/ENO (STR_TO_WORD_E)

[Structured ladder]



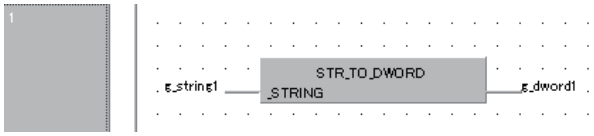
[ST]

```
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]



[ST]

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

5.1.31 String type → time type conversion

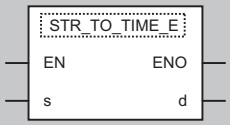
STR_TO_TIME(_E)

Universal UD High Performance

STR_TO_TIME(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

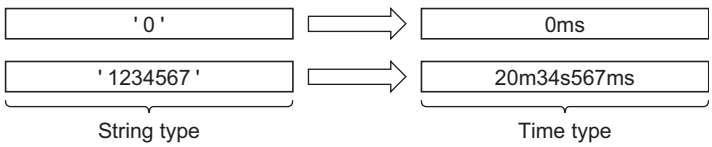
STR_TO_TIME STR_TO_ TIME _E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:String
Output argument,	ENO:	Output status (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

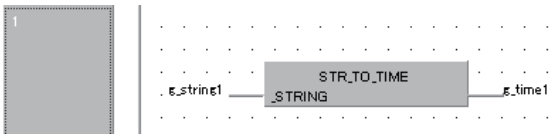
No operation error occurs in the execution of the STR_TO_TIME(_E) function.

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]

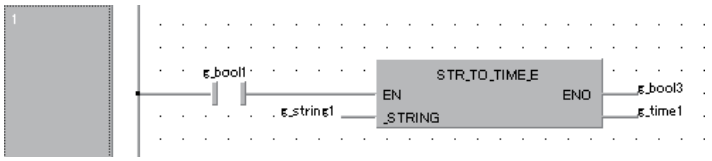


[ST]

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

(b) Function with EN/ENO (STR_TO_TIME_E)

[Structured ladder]



[ST]

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

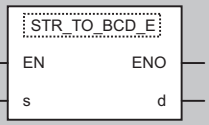
5.1.32 String type → BCD type conversion

STR_TO_BCD(_E)

Universal UD High Performance

STR_TO_BCD(_E) (_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

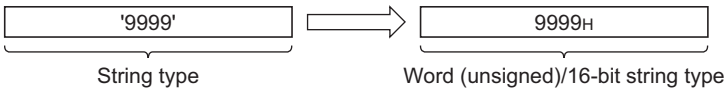
STR_TO_BCD STR_TO_ BCD _E

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

Function

Operation processing

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



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

(3) String type data can contain up to 50 characters.
'20H' (space) and '30H' (0) in the character string are counted as one character.

(4) 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 error)*1	Undefined 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.

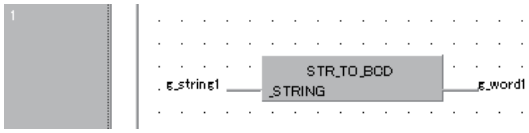
- The input character string is outside the range of ASCII code '30H' to '39H'. (Error code 4100)
- The input character string is within the range of ASCII code '30H' to '39H' and exceeding '9999'. (Error code 4100)

Program Example

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]

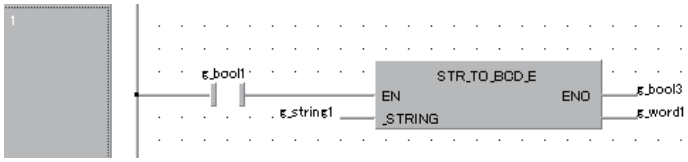


[ST]

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

(b) Function without EN/ENO (STR_TO_BCD_E)

[Structured ladder]



[ST]

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

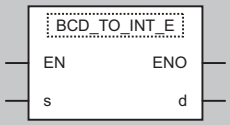
5.1.33 BCD type → word (signed), double word (signed) type conversion BCD_TO_INT(_E), BCD_TO_DINT(_E)

Universal UD High Performance

BCD_TO_INT(_E)
BCD_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder



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

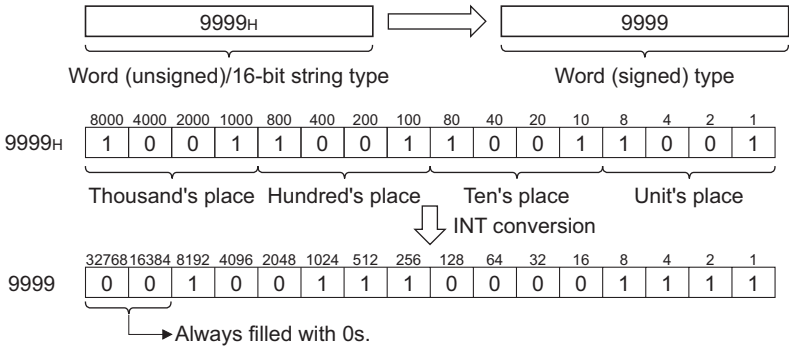
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

(1) BCD_TO_INT, BCD_TO_INT_E

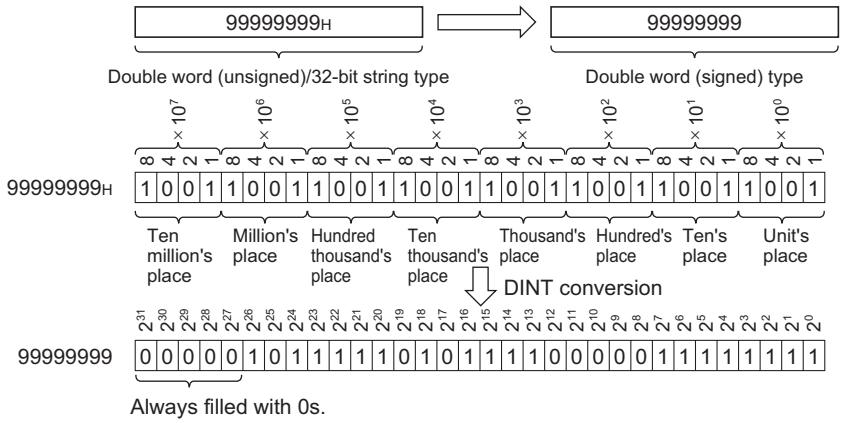
(a) Converts BCD type data input to ⓢ into word (signed) type data, and outputs the operation result from ⓓ .



(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 ⓓ .



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

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

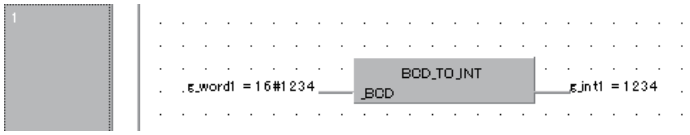
No operation error occurs in the execution of the BCD_TO_INT(_E) and BCD_TO_DINT(_E) functions.

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]

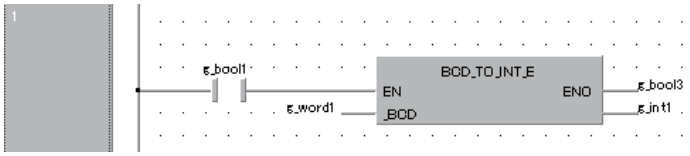


[ST]

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

(b) Function with EN/ENO (BCD_TO_INT_E)

[Structured ladder]



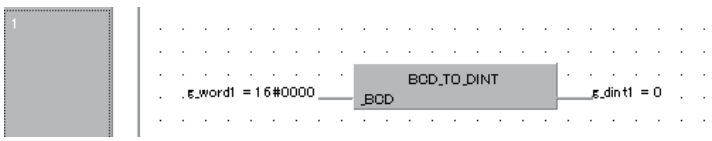
[ST]

```
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]



[ST]

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

5.1.34 BCD type → string type conversion

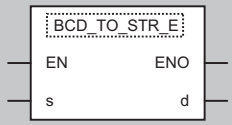
BCD_TO_STR(_E)

Universal UD High Performance

BCD_TO_STR(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

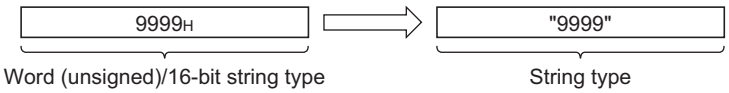
BCD_TO_STR BCD_TO_STR_E

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

Function

Operation processing

Converts BCD type data input to ⓢ into 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

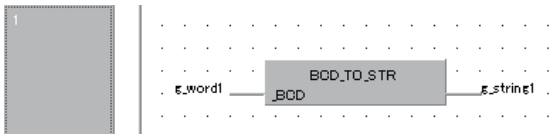
No operation error occurs in the execution of the BCD_TO_STR(_E) function.

Program Example

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]

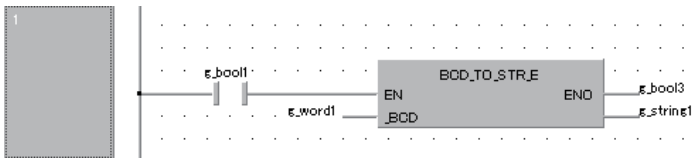


[ST]

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

(b) Function with EN/ENO (BCD_TO_STR_E)

[Structured ladder]



[ST]

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

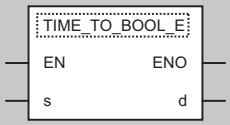
5.1.35 Time type → bit type conversion

TIME_TO_BOOL(_E)

Universal UD High Performance

TIME_TO_BOOL(_E) (_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

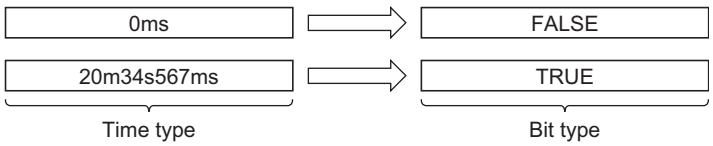
TIME_TO_BOOL TIME_TO_BOOL_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Time
Output argument,	ENO:	Output status (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

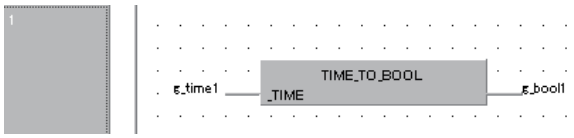
No operation error occurs in the execution of the TIME_TO_BOOL(_E) function.

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]

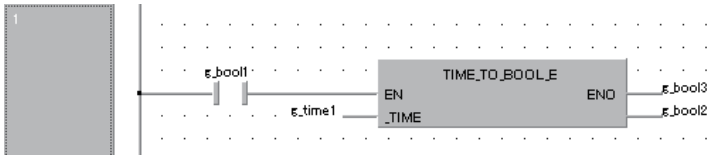


[ST]

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

(b) Function with EN/ENO (TIME_TO_BOOL_E)

[Structured ladder]



[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

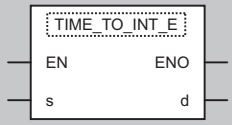
TIME_TO_INT(_E), TIME_TO_DINT(_E)

Universal UD High Performance

TIME_TO_INT(_E)
TIME_TO_DINT(_E)

(_E: With EN/ENO)

Structured ladder



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

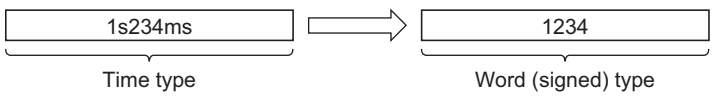
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Time
Output argument,	ENO:	Output status (TRUE: Normal execution, FALSE: Error or stop)	:Bit
	d:	Output	:Word (signed), double word (signed)

Function

Operation processing

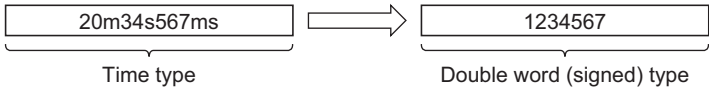
(1) TIME_TO_INT, TIME _TO_INT_E

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



(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

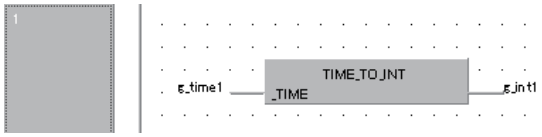
No operation error occurs in the execution of the TIME_TO_INT(_E) and TIME_TO_DINT(_E) functions.

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]

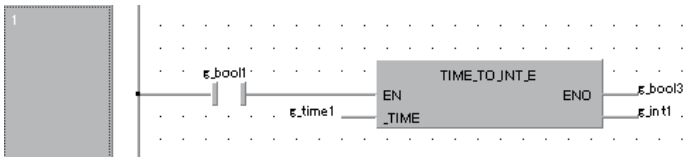


[ST]

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

(b) Function with EN/ENO (TIME_TO_INT_E)

[Structured ladder]



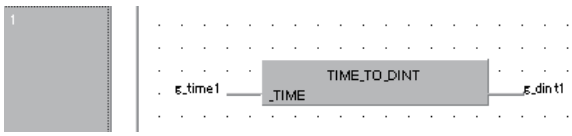
[ST]

```
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]



[ST]

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

5.1.37 Time type → string type conversion

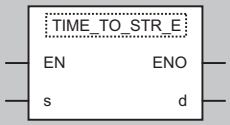
TIME_TO_STR(_E)

Universal UD High Performance

TIME_TO_STR(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

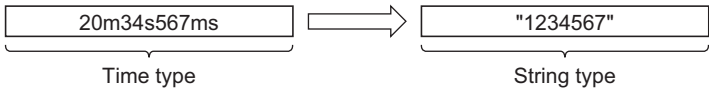
TIME_TO_STR TIME _TO_STR_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Time
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String

Function

Operation processing

Converts time type data input to ⓢ into 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

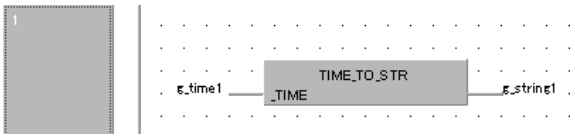
No operation error occurs in the execution of the TIME_TO_STR(_E) function.

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]

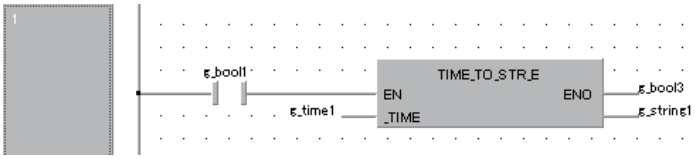


[ST]

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

(b) Function with EN/ENO (TIME_TO_STR_E)

[Structured ladder]



[ST]

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

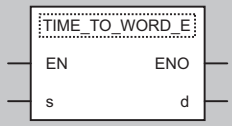
5.1.38 Time type → word (unsigned)/16-bit string, double word (unsigned)/32-bit string type conversion TIME_TO_WORD(_E), TIME_TO_DWORD(_E)

Universal UD High Performance

TIME_TO_WORD(_E)
TIME_TO_DWORD(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

TIME_TO_WORD_E indicates any of the following functions.

TIME_TO_WORD TIME _TO_WORD_E
TIME _TO_DWORD
TIME _TO_DWORD_E

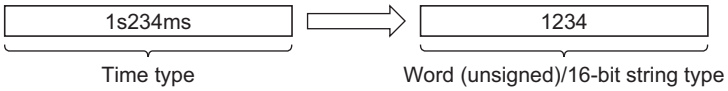
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Time
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string

Function

Operation processing

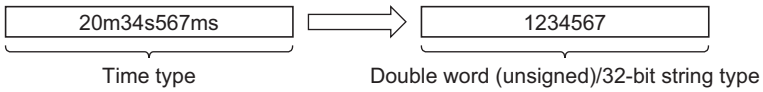
(1) TIME_TO_WORD, TIME _TO_WORD_E

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



(2) 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

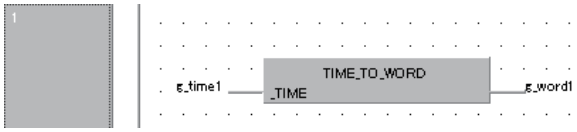
No operation error occurs in the execution of the TIME_TO_WORD(_E) and TIME_TO_DWORD(_E) functions.

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]

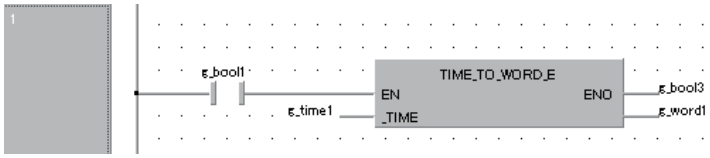


[ST]

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

(b) Function with EN/ENO (TIME_TO_WORD_E)

[Structured ladder]



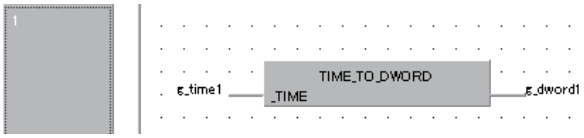
[ST]

```
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]



[ST]

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

5.2 Standard Functions of One Numeric Variable

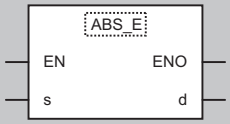
5.2.1 Absolute value

ABS(_E)

Universal UD High Performance

ABS(_E) (_E: With EN/ENO)

Structured ladder



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:	Input	:ANY_NUM
Output argument,	ENO:	Output status (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.)

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

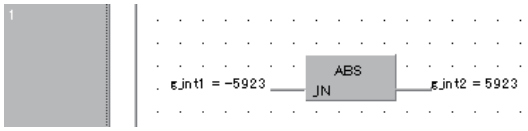
No operation error occurs in the execution of the ABS(_E) function.

Program Example

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

(a) Function without EN/ENO (ABS)

[Structured ladder]

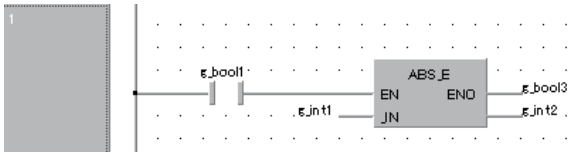


[ST]

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

(b) Function with EN/ENO (ABS_E)

[Structured ladder]



[ST]

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

5.3 Standard Arithmetic Functions

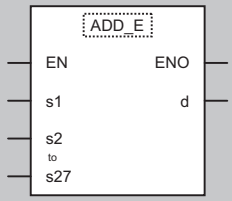
5.3.1 Addition

ADD(_E)

Universal UD High Performance

ADD(_E) (_E: With EN/ENO)

Structured ladder



ST

ENO:=ADD_E(EN, s1, s2…s27, d);

: indicates any of the following functions.

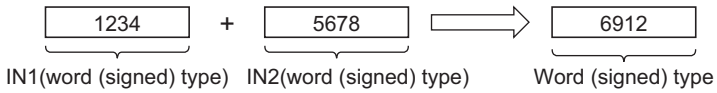
ADD ADD_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1 to s27: (s1 only	Input for NOT(_E)	:ANY_NUM
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:ANY_NUM

Function

Operation processing

(1) Performs addition (ⓢ1 +ⓢ2 +…+ⓢ27) on word (signed), double word (signed), single-precision real or double-precision real type data input to ⓢ1 to ⓢ27 , and outputs the operation result from ⓓ in the same data type as that of ⓢ .
(Example) Word (signed) type data



(2) The values to be input to ⓢ1 to ⓢ27 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 27.

(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.

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

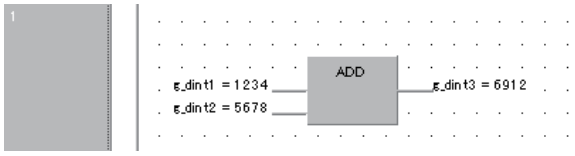
No operation error occurs in the execution of the ADD(_E) function.

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 ⓢ .

(a) Function without EN/ENO (ADD)

[Structured ladder]

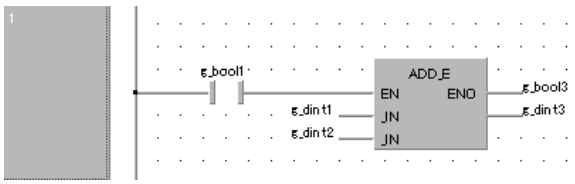


[ST]

```
g_dint3:= g_dint1+g_dint2;
```

(b) Function with EN/ENO (ADD_E)

[Structured ladder]



[ST]

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

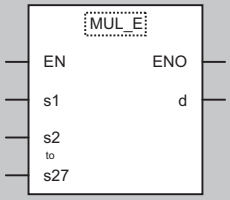
5.3.2 Multiplication

MUL(_E)

Universal UD | High Performance

MUL(_E) (_E: With EN/ENO)

Structured ladder



ST

```
ENO:=MUL_E(EN, s1, s2…s27, d);
```

MUL_E : indicates any of the following functions.

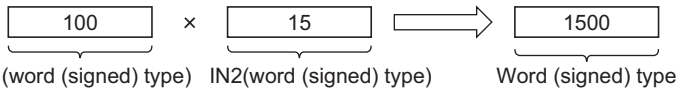
MUL	MUL_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1 to s27:	Input	:ANY_NUM
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:ANY_NUM

Function

Operation processing

(1) Performs multiplication (ⓢ1 × ⓢ2 ×…× ⓢ27) on word (signed), double word (signed), single-precision real or double-precision real type data input to ⓢ1 to ⓢ27 , and outputs the operation result from ⓓ in the same data type as that of ⓢ .
(Example) Word (signed) type data



(2) The values to be input to ⓢ1 to ⓢ27 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 27.

(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.

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

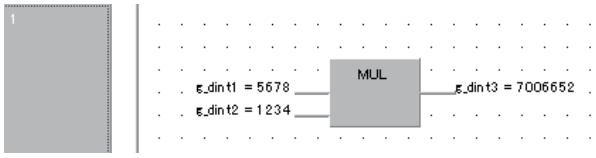
No operation error occurs in the execution of the MUL(_E) function.

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 ⓢ .

(a) Function without EN/ENO (MUL)

[Structured ladder]

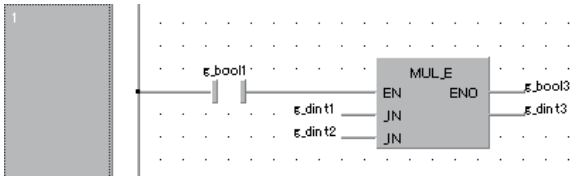


[ST]

```
g_dint3:= g_dint1*g_dint2;
```

(b) Function with EN/ENO (MUL_E)

[Structured ladder]



[ST]

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

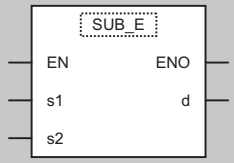
5.3.3 Subtraction

SUB(_E)

Universal UD | High Performance

SUB(_E) (_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

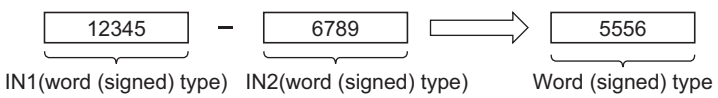
SUB	SUB_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1: s2:	Input	:ANY_NUM
Output argument,	ENO:	Output status (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 ⓢ .
(Example) Word (signed) type data



IN1(word (signed) type) IN2(word (signed) type) Word (signed) type

(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.

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

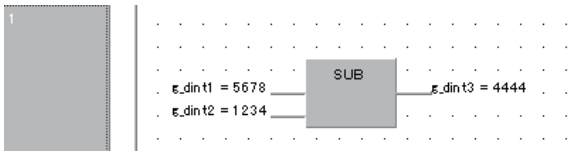
No operation error occurs in the execution of the SUB(_E) function.

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 ⓢ .

(a) Function without EN/ENO (SUB)

[Structured ladder]

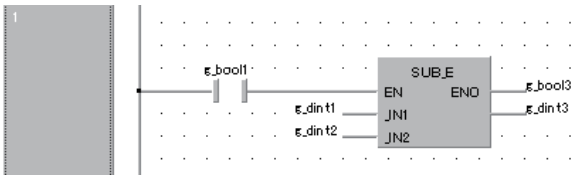


[ST]

```
g_dint3:= g_dint1-g_dint2;
```

(b) Function with EN/ENO (SUB_E)

[Structured ladder]



[ST]

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

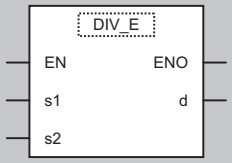
5.3.4 Division

DIV(_E)

Universal UD High Performance

DIV(_E) (_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

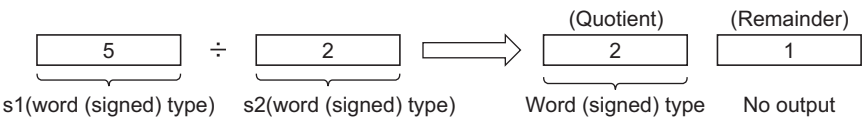
DIV DIV_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1: s2:	Input	:ANY_NUM
Output argument,	ENO:	Output status (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 ⓢ .
(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.)

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 error)*1	Undefined 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.

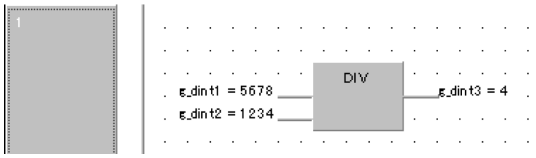
- The value to be input to ⓢ2 is 0. (Division by 0) (Error code: 4100)

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 ⓢ .

(a) Function without EN/ENO (DIV)

[Structured ladder]

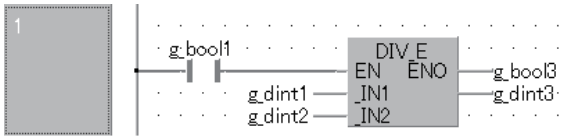


[ST]

```
g_dint3:= g_dint1/g_dint2;
```

(b) Function with EN/ENO (DIV_E)

[Structured ladder]



[ST]

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

5.3.5 Modulus operation

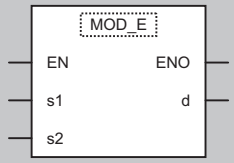
MOD(_E)

Universal UD High Performance

MOD(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

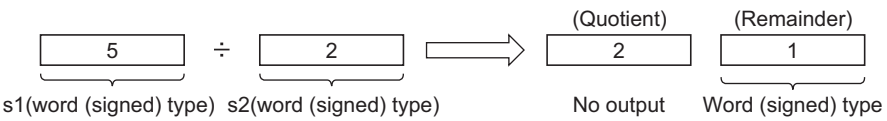
MOD	MOD_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1: s2:	Input	:ANY_INT
Output argument,	ENO:	Output status (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 ⓢ .
(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 error)*1	Undefined 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.

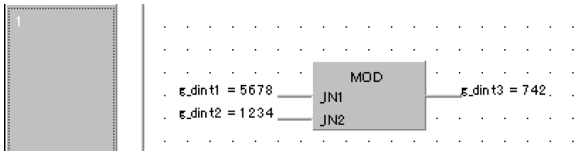
- The value to be input to ⓢ2 is 0. (Division by 0) (Error code: 4100)

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 ⓢ .

(a) Function without EN/ENO (MOD)

[Structured ladder]

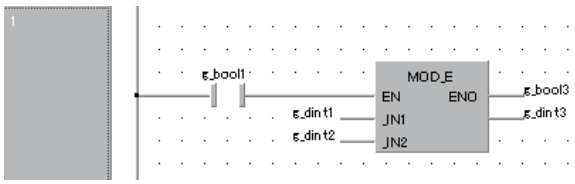


[ST]

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

(b) Function with EN/ENO (MOD_E)

[Structured ladder]



[ST]

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

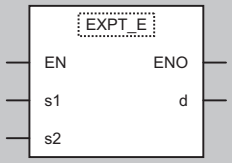
5.3.6 Exponentiation

EXPT(_E)

Universal UD High Performance

EXPT(_E) (_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

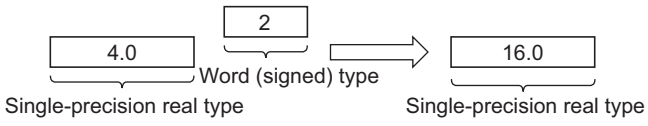
EXPT EXPT _E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1:	Input	:ANY_REAL
	s2:	Input	:ANY_NUM
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:ANY_REAL

Function

Operation processing

Performs exponentiation ⓢ2 on single-precision real or double-precision real type data input to ⓢ1, 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

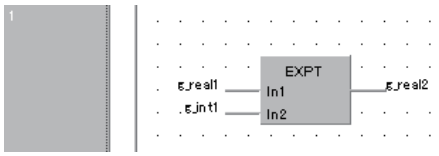
No operation error occurs in the execution of the EXPT(_E) function.

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]

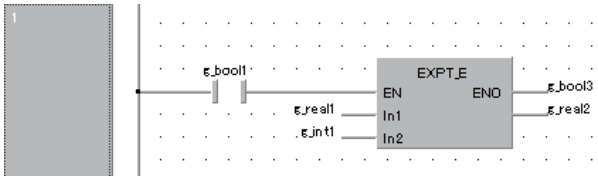


[ST]

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

(b) Function with EN/ENO (EXPT_E)

[Structured ladder]



[ST]

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

5.3.7 Move operation

MOVE(_E)

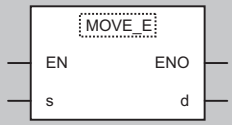
Universal UD

High Performance

MOVE(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

MOVE_E indicates any of the following functions.

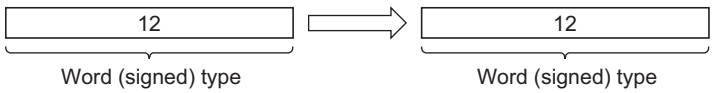
MOVE MOVE_E

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

Function

Operation processing

(1) Moves the data from ⓢ to ⓓ.



Word (signed) type — Word (signed) type

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

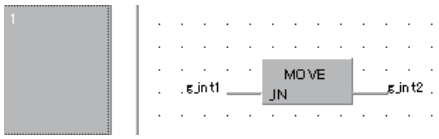
No operation error occurs in the execution of the MOVE(_E) function.

Program Example

The program which moves the word (signed) type data input to ⓢ to ⓓ .

(a) Function without EN/ENO (MOVE)

[Structured ladder]

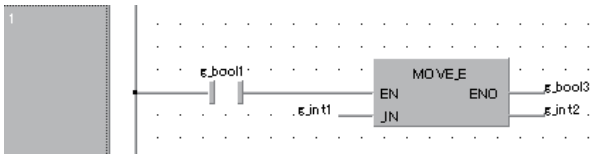


[ST]

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

(b) Function with EN/ENO (MOVE_E)

[Structured ladder]



[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

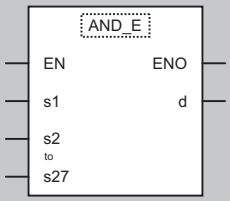
AND(_E), OR(_E), XOR(_E), NOT(_E)

Universal UD High Performance

AND(_E)
OR(_E)
XOR(_E)
NOT(_E)

(_E: With EN/ENO)

Structured ladder



ST

ENO:= AND_E (EN, s1, s2…s27, d);

AND_E indicates any of the following functions.

AND	AND_E
OR	OR_E
XOR	XOR_E
NOT	NOT_E

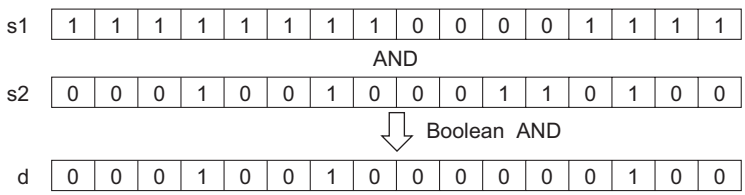
Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1 to s27: (s1 only for NOT(_E))	Input	:ANY_BIT
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:ANY_BIT

Function

Operation processing

(1) AND, AND_E

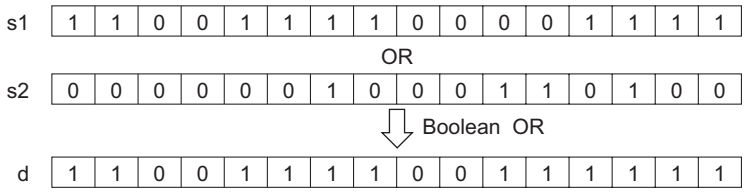
(a) Performs Boolean AND on bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data input to variables ⓢ1 to ⓢ27 bit by bit, and outputs the operation result from ⓓ in the same data type as that of variables ⓢ1 to ⓢ27.
(Example) Word (unsigned)/16-bit string type data



(b) The number of pins of variable 's' can be changed in the range from 2 to 27.

(2) OR, OR_E

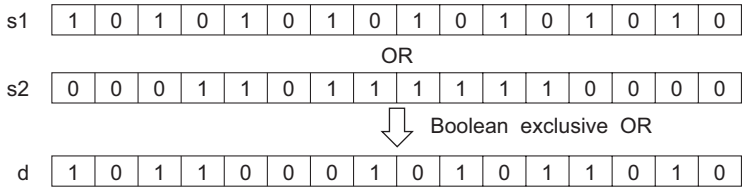
(a) Performs Boolean OR on bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data input to variables ⓢ1 to ⓢ27 bit by bit, and outputs the operation result from ⓓ in the same data type as that of variables ⓢ1 to ⓢ27.
(Example) Word (unsigned)/16-bit string type data



(b) The number of pins of variable 's' can be changed in the range from 2 to 27.

(3) XOR, 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 variables ⓢ1 to ⓢ27 bit by bit, and outputs the operation result from ⓓ in the same data type as that of variables ⓢ1 to ⓢ27.
(Example) Word (unsigned)/16-bit string type data

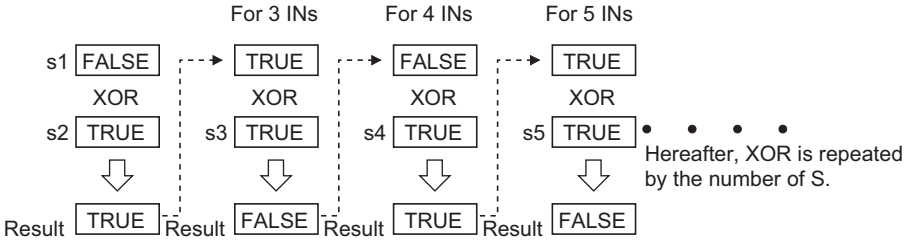


(b) The number of pins of variable 's' can be changed in the range from 2 to 27.

(c) When three or more variables 's' exist, XOR is performed between ⓢ1 and ⓢ2 first, and XOR is successively performed between the result and ⓢ3.

When the expression includes ⓢ4, XOR is performed between the result of XOR with ⓢ3 and ⓢ4.

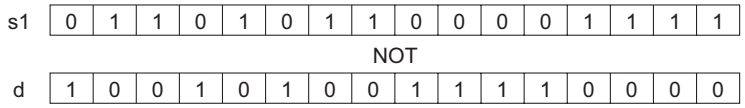
In this manner, XOR is repeated by the number of variables 's' in the order with ⓢ5, ⓢ6 and so on.
(Example) Bit type data



(4) NOT, NOT_E

Performs Boolean NOT on bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data input to variable ⓢ1 bit by bit, and outputs the operation result from ⓓ in the same data type as that of variable ⓢ1.

(Example) Word (unsigned)/16-bit string type data



(5) The value to be input to variables ⓢ1 to ⓢ27 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

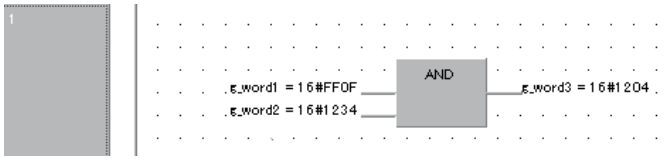
No operation error occurs in the execution of the AND(_E), OR(_E), XOR(_E), and NOT(_E) functions.

Program Example

(1) The program which performs Boolean AND on bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data input to variables ⓢ1 to ⓢ27 bit by bit, and outputs the operation result from ⓓ in the same data type as that of variables ⓢ1 to ⓢ27.

(a) Function without EN/ENO (AND)

[Structured ladder]

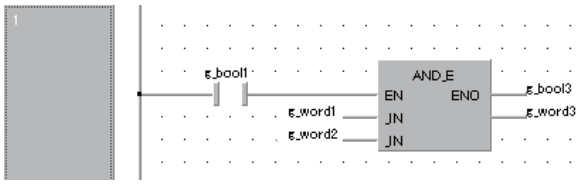


[ST]

```
g_word3 :=(g_word1) AND (g_word2);
or
g_word3 :=(g_word1) & (g_word2);
```

(b) Function with EN/ENO (AND_E)

[Structured ladder]



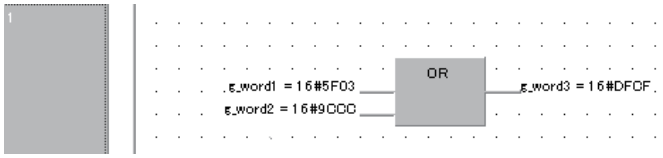
[ST]

```
g_bool3 := AND_E (g_bool1, g_word1, g_word2, g_word3);
```

(2) The program which performs Boolean OR on bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data input to variables ⓢ1 to ⓢ27 bit by bit, and outputs the operation result from ⓓ in the same data type as that of variables ⓢ1 to ⓢ27.

(a) Function without EN/ENO (OR)

[Structured ladder]



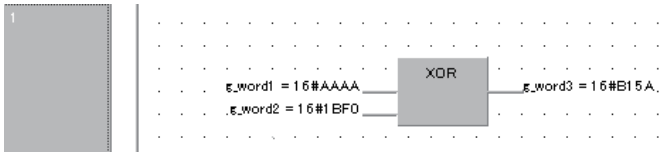
[ST]

```
g_word3 :=(g_word1) OR (g_word2);
```

(3) The program which performs Boolean XOR on bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data input to variables ⓢ1 to ⓢ27 bit by bit, and outputs the operation result from ⓓ in the same data type as that of variables ⓢ1 to ⓢ27.

(a) Function without EN/ENO (XOR)

[Structured ladder]



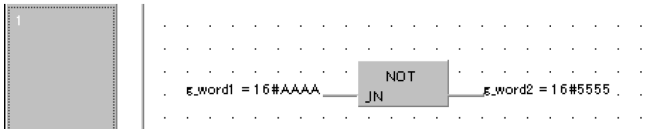
[ST]

```
g_word3 :=(g_word1) XOR (g_word2);
```

(4) The program which performs Boolean NOT on bit, word (unsigned)/16-bit string or double word (unsigned)/32-bit string type data input to variable ⓢ1 bit by bit, and outputs the operation result from ⓓ in the same data type as that of variable ⓢ1.

(a) Function without EN/ENO (NOT)

[Structured ladder]



[ST]

```
g_word2 :=NOT (g_word1);
```

5.5 Standard Selection Functions

5.5.1 Selection

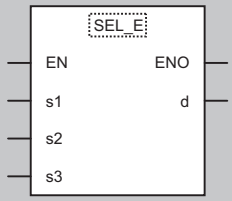
SEL(_E)

Universal UD High Performance

SEL(_E)

(_E: With EN/ENO)

Structured ladder



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:	Output condition (TRUE: s3 output, FALSE: s2 output)	:Bit
	s2: s3:	Input	:ANY
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:ANY

Function

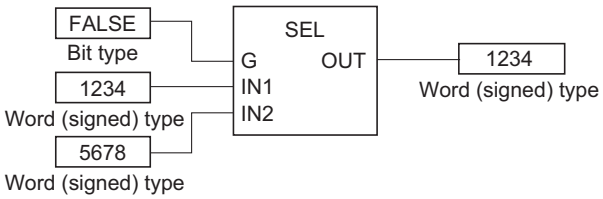
Operation processing

(1) 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.

When the input value of ⓢ1 is FALSE, the value input to ⓢ2 is output from ⓓ.

When the input value of ⓢ1 is TRUE, the value input to ⓢ3 is output from ⓓ.

(Example) ⓢ2 and ⓢ3 are 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

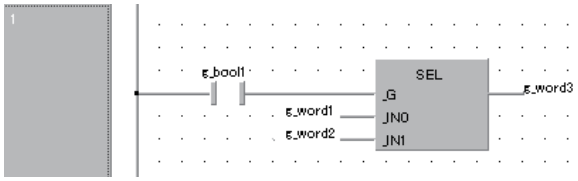
No operation error occurs in the execution of the SEL(_E) function.

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]

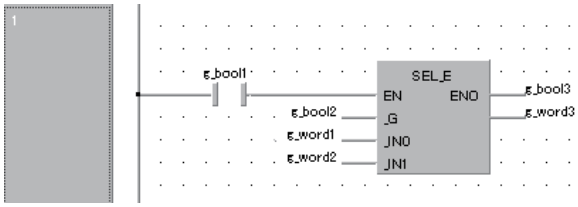


[ST]

```
g_word3 := SEL (g_bool1, g_word1, g_word2);
```

(b) Function with EN/ENO (SEL_E)

[Structured ladder]



[ST]

```
g_bool3 := SEL_E (g_bool1, g_bool2, g_word1, g_word2, g_word3);
```

5.5.2 Multiplexer

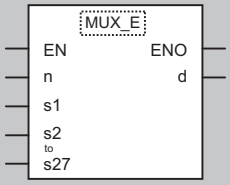
MUX(_E)

Universal UD High Performance

MUX(_E)

(_E: With EN/ENO)

Structured ladder



ST

```
ENO:=MUX_E(EN, n, s1, s2···s27, d);
```

dotted box indicates any of the following functions.

MUX	MUX_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	n:	Output value selection	:Word (signed)
	s1 to s27:	Input	:ANY
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:ANY

Function

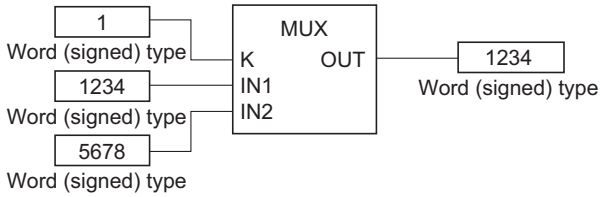
Operation processing

(1) Selects the value to be output among the values input to variables ⓢ1 to ⓢ27 according to the value input to ⓝ, and outputs the operation result from ⓓ in the same data type as that of variables ⓢ1 to ⓢ27.

When the input value of ⓝ is 1, the value input to ⓢ1 is output from ⓓ.

When the input value of ⓝ is ⓝ, the value input to ⓢn is output from ⓓ.

(Example) Word (signed) type data



(2) If a value input to ⓝ is outside the range of number of pins of variable 's', an undefined value is output from ⓓ.

(No operation error occurs. In case of MUX_E, FALSE is output from ENO.)

(3) The value to be input to ⓝ is word (signed) type data within the range from 1 to 27 (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 or string type data.

(5) The number of pins of variable 's' can be changed in the range from 2 to 27.

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

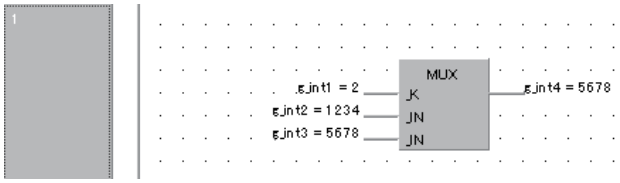
No operation error occurs in the execution of the MUX(_E) function.

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 ⓝ , and outputs the operation result from ⓓ in the same data type as that of variable ⓢ1 or ⓢ2 .

(a) Function without EN/ENO (MUX)

[Structured ladder]

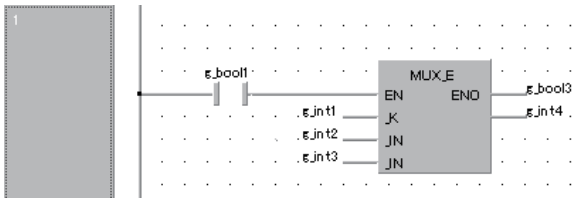


[ST]

```
g_int4 := MUX (g_int1, g_int2, g_int3);
```

(b) Function with EN/ENO (MUX_E)

[Structured ladder]



[ST]

```
g_bool3 := MUX_E (g_bool1, g_int1, g_int2, g_int3, g_int4);
```

5.6 Standard Comparison Functions

5.6.1 Comparison

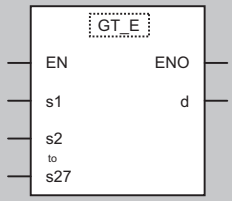
GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E), NE(_E)

Universal UD High Performance

GT(_E)
GE(_E)
EQ(_E)
LE(_E)
LT(_E)
NE(_E)

(_E: With EN/ENO)

Structured ladder



ST

ENO:= GT_E (EN, s1, s2···s27, d);

[ ] indicates any of the following functions.

GT	GT_E
GE	GE_E
EQ	EQ_E
LE	LE_E
LT	LT_E
NE	NE_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1 to s27:	Input	:ANY_SIMPLE
Output argument,	ENO:	Output status (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 ⓢ1 to ⓢ27, and outputs the operation result from ⓓ in the same data type as that of variables ⓢ1 to ⓢ27.

(a) GT, 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, 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, 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, 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, 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, NE_E(< >) Performs comparison of [ⓢ1 ≠ ⓢ2].

- Outputs TRUE if ⓢ1 ≠ ⓢ2 .
- Outputs FALSE if =ⓢ1 ⓢ2 .

(2) The number of pins of variable 's' can be changed in the range from 2 to 27. (The number of pins of variable 's' 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

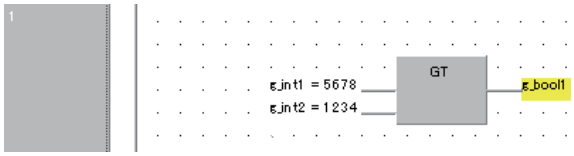
No operation error occurs in the execution of the GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E), and NE(_E) functions.

Program Example

The program which performs comparison operation between the values input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ .

(a) Function without EN/ENO (GT)

[Structured ladder]

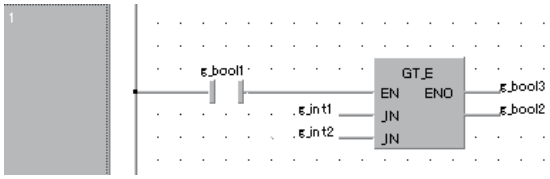


[ST]

```
g_bool1:= (g_int1) > (g_int2);
```

(b) Function with EN/ENO (GT_E)

[Structured ladder]



[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)

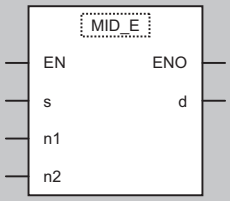
Universal UD

High Performance

MID(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

[ ] indicates any of the following functions.

MID MID_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:String
	n1:	Number of characters to be extracted	:Word (signed)
	n2:	Start position to be extracted	:Word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String

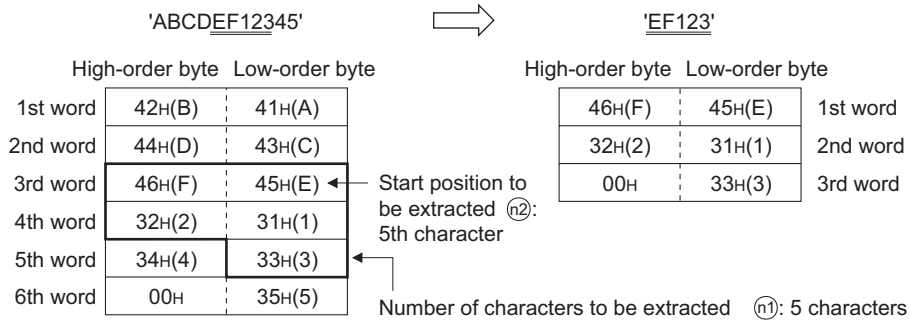
Function

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 ⓝ1 .
The start position of the characters to be extracted is specified by the value input to P.

(Example) Values input to ⓝ1 and ⓝ2 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 ⓝ1 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 ⓝ2 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

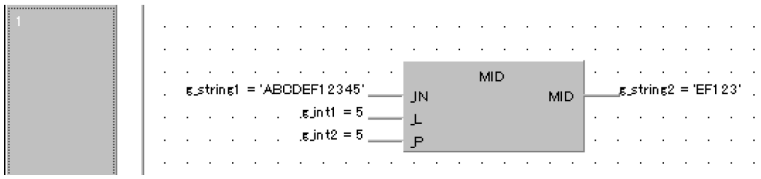
No operation error occurs in the execution of the MID(_E) function.

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]

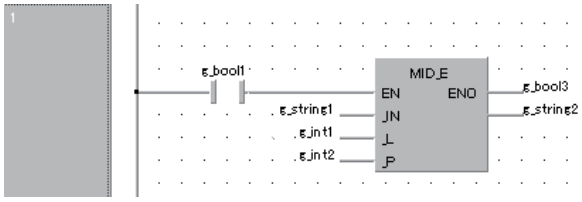


[ST]

```
g_string2:=MID(g_string1, g_int1, g_int2);
```

(b) Function with EN/ENO (MID_E)

[Structured ladder]



[ST]

```
g_bool3 := MID_E(g_bool1, g_string1, g_int1, g_int2, g_string2);
```

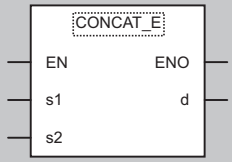
5.7.2 String concatenation

CONCAT(_E)

Universal UD | High Performance

CONCAT(_E) (_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

CONCAT CONCAT_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1: s2:	Input	:String
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String

Function

Operation processing

(1) Concatenates the character string input to ⓢ2 following the one input to ⓢ1 , and outputs the operation result from ⓓ .

This function concatenates character string ⓢ2 with ignoring '00H', which indicates the end of character string ⓢ1 .
If the concatenated character string has over 255 bytes, the character string up to 255 bytes is output.

'ABCDE' + '123456' ⇨ 'ABCDE123456'

	High-order byte	Low-order byte
1st word	42H(B)	41H(A)
2nd word	44H(D)	43H(C)
3rd word	00H	45H(E)

	High-order byte	Low-order byte
1st word	32H(2)	31H(1)
2nd word	34H(4)	33H(3)
3rd word	36H(6)	35H(5)
4th word	00H	

	High-order byte	Low-order byte
1st word	42H(B)	41H(A)
2nd word	44H(D)	43H(C)
3rd word	31H(1)	45H(E)
4th word	33H(3)	32H(2)
5th word	35H(5)	34H(4)
6th word	00H	36H(6)

(2) The values to be input to ⓢ1 and ⓢ2 are string type data within the range from 0 to 255 bytes.

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

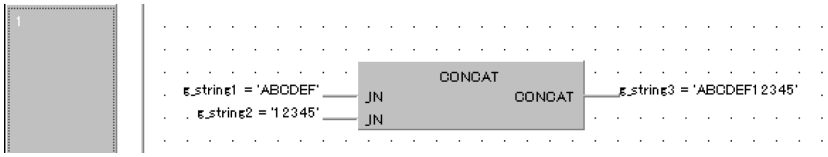
No operation error occurs in the execution of the CONCAT(_E) function.

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]

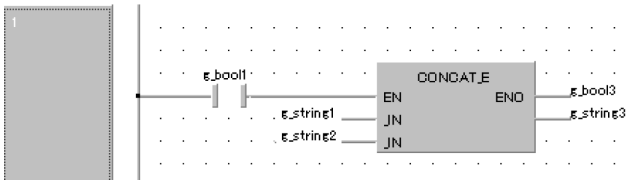


[ST]

```
g_string3:=CONCAT(g_string1, g_string2);
```

(b) Function with EN/ENO (CONCAT_E)

[Structured ladder]



[ST]

```
g_bool3 := CONCAT_E(g_bool1, g_string1, g_string2, g_string3);
```

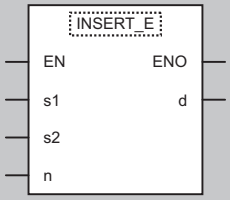
5.7.3 String insertion

INSERT(_E)

Universal UD High Performance

INSERT(_E) (_E: With EN/ENO)

Structured ladder



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: s2:	Input	:String
	n:	Start position to be inserted	:Word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String

Function

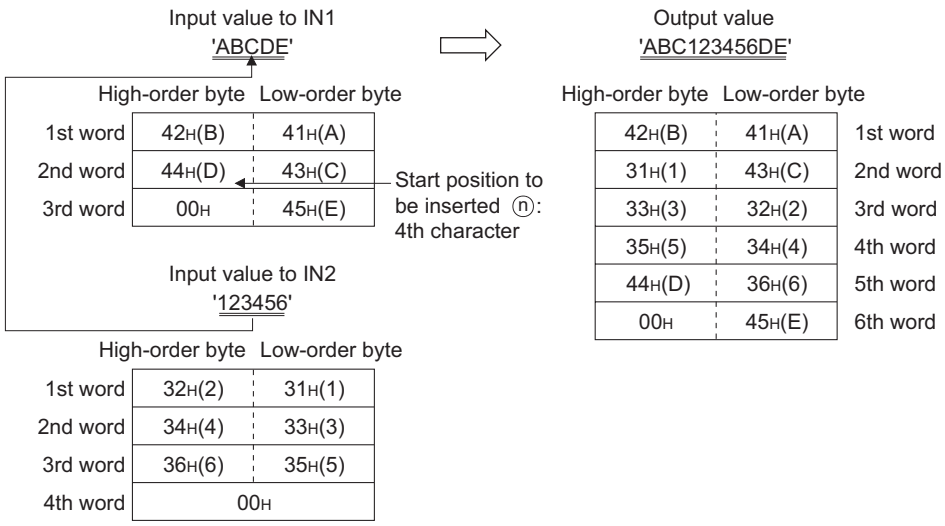
Operation processing

(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 ⓝ .

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 ⓝ is 4



(2) The values to be input to ⓢ1 and ⓢ2 are string type data within the range from 0 to 255 bytes.

(3) The value to be input to ⓝ 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 ⓢ1 .)

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

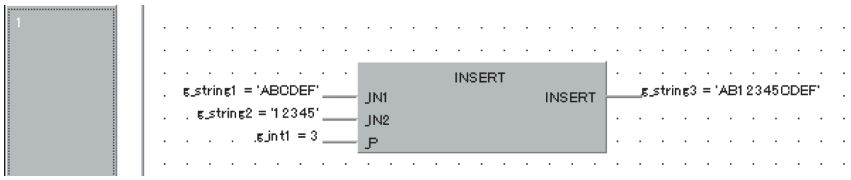
No operation error occurs in the execution of the INSERT(_E) function.

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]

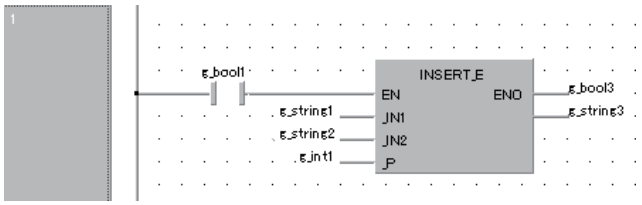


[ST]

```
g_string3:=INSERT(g_string1, g_string2, g_int1);
```

(b) Function with EN/ENO (INSERT_E)

[Structured ladder]



[ST]

```
g_bool3 := INSERT_E(g_bool1, g_string1, g_string2, g_int1, g_string3);
```

5.7.4 String deletion

DELETE(_E)

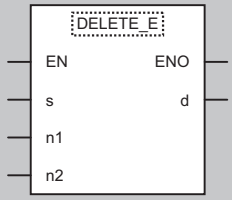
Universal UD

High Performance

DELETE(_E)

(_E: With EN/ENO)

Structured ladder



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:	Input	:String
	n1:	Number of characters to be deleted	:Word (signed)
	n2:	Start position to be deleted	:Word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String

Function

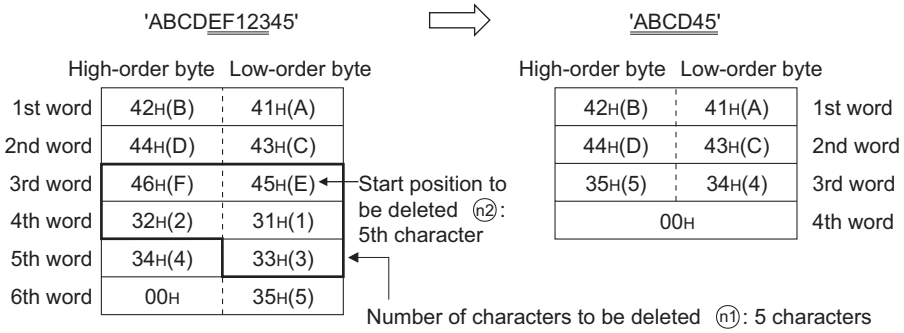
Operation processing

(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 ⓝ1 .

The start position to be deleted in the character string is specified by the value input to ⓝ2 .

(Example) Values input to ⓝ1 and ⓝ2 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 ⓝ1 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 ⓝ2 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

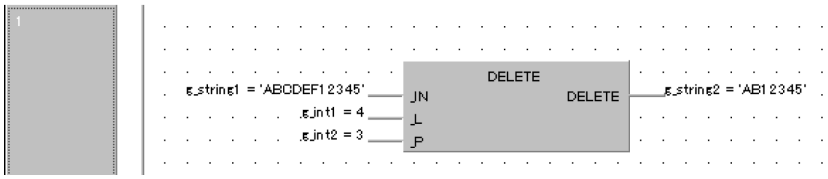
No operation error occurs in the execution of the DELETE(_E) function.

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]

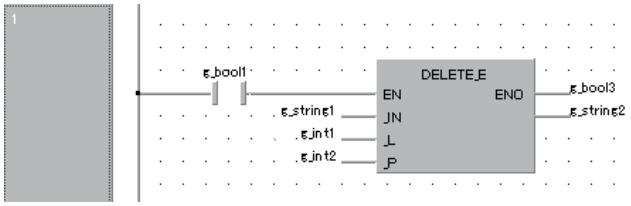


[ST]

```
g_string2:=DELETE(g_string1, g_int1, g_int2);
```

(b) Function with EN/ENO (DELETE_E)

[Structured ladder]



[ST]

```
g_bool3 := DELETE_E(g_bool1, g_string1, g_int1, g_int2, g_string2);
```

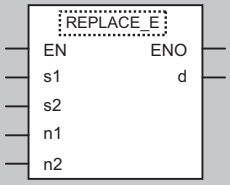
5.7.5 String replacement

REPLACE(_E)

Universal UD | High Performance

REPLACE(_E) (_E: With EN/ENO)

Structured ladder



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: s2:	Input	:String
	n1:	Number of characters to be replaced	:Word (signed)
	n2:	Start position to be replaced	:Word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:String

Function

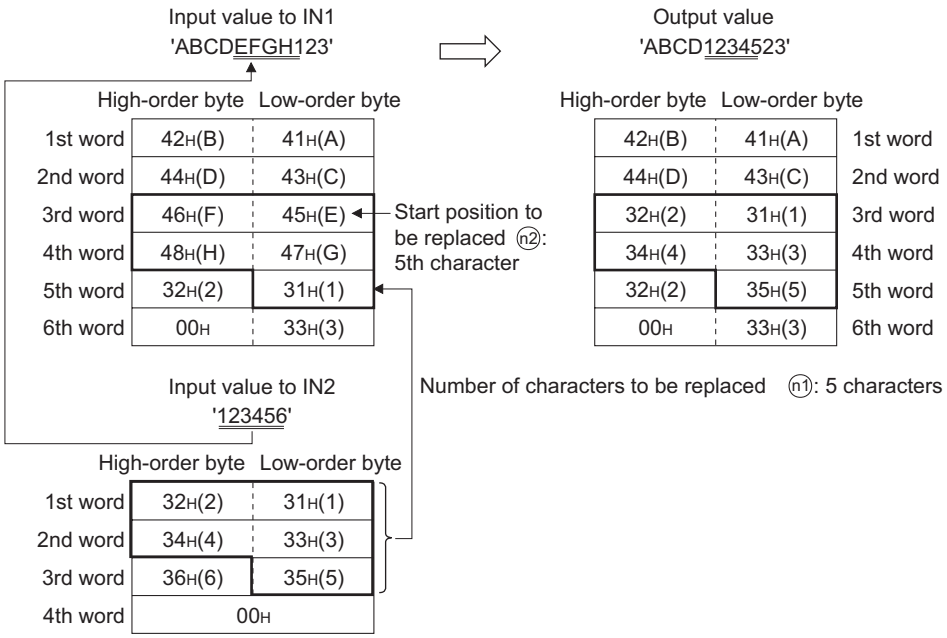
Operation processing

(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 ⓝ1 .

The start position to be replaced in the character string is specified by the value input to ⓝ2 .

(Example) Values input to ⓝ1 and ⓝ2 are 5



(2) The values to be input to ⓢ1 and ⓢ2 are string type data within the range from 0 to 255 bytes.

(3) The value to be input to ⓝ1 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 ⓢ1 .)

(4) The value to be input to ⓝ2 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 ⓢ1 .)

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

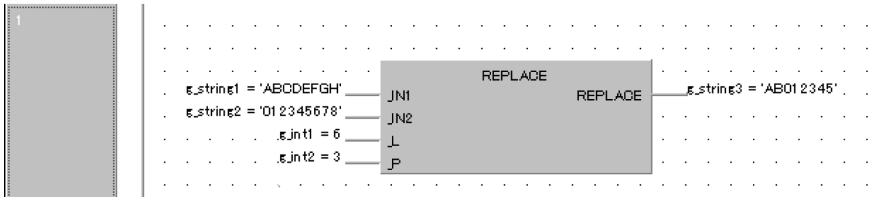
No operation error occurs in the execution of the REPLACE(_E) function.

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]

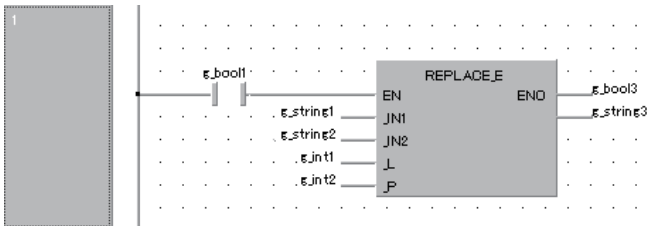


[ST]

```
g_string3:=REPLACE(g_string1, g_string2, g_int1, g_int2);
```

(b) Function with EN/ENO (REPLACE_E)

[Structured ladder]



[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

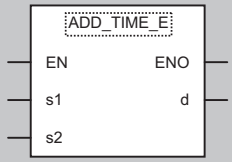
ADD_TIME(_E)

Universal UD High Performance

ADD_TIME(_E)

(_E: With EN/ENO)

Structured ladder



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: s2:	Input	:Time
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Time

Function

Operation processing

(1) Performs addition (ⓢ1 +ⓢ2) on time type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in time type.

(2) After data conversion, high-order 16 bits are filled with 0.

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

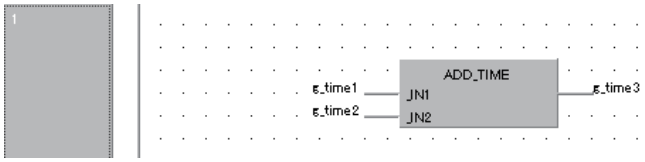
No operation error occurs in the execution of the ADD_TIME(_E) function.

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]

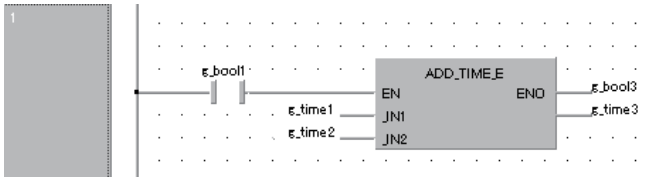


[ST]

```
g_time3:= ADD_TIME(g_time1, g_time2);
```

(b) Function with EN/ENO (ADD_TIME_E)

[Structured ladder]



[ST]

```
g_bool3 := ADD_TIME_E(g_bool1, g_time1, g_time2, g_time3);
```

5.8.2 Subtraction

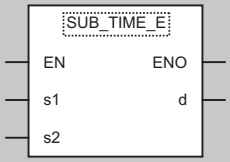
SUB_TIME(_E)

Universal UD High Performance

SUB_TIME(_E)

(_E: With EN/ENO)

Structured ladder



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: s2:	Input	:Time
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Time

Function

Operation processing

Performs subtraction (ⓢ1 −ⓢ2) on time type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in time type.

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

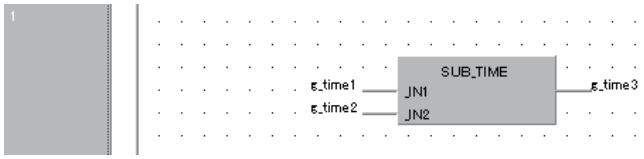
No operation error occurs in the execution of the SUB_TIME(_E) function.

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]

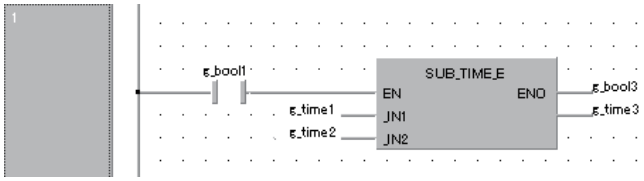


[ST]

```
g_time3:= SUB_TIME(g_time1, g_time2);
```

(b) Function with EN/ENO (SUB_TIME_E)

[Structured ladder]



[ST]

```
g_bool3 := SUB_TIME_E(g_bool1, g_time1, g_time2, g_time3);
```

5.8.3 Multiplication

MUL_TIME(_E)

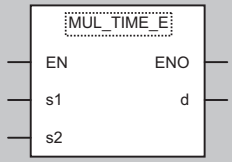
Universal UD

High Performance

MUL_TIME(_E)

(_E: With EN/ENO)

Structured ladder



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:	Input	:Time
	s2:	Input	:ANY_NUM
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Time

Function

Operation processing

(1) Performs multiplication (ⓢ1 × ⓢ2) on time type data input to ⓢ1 and ⓢ2 , and outputs the operation result from ⓓ in time type.

(2) The value to be input to ⓢ1 is time type data.

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

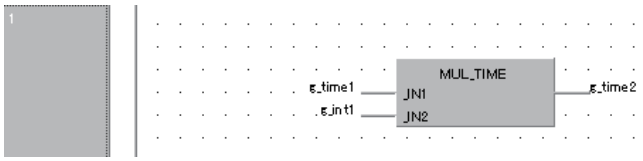
No operation error occurs in the execution of the MUL_TIME(_E) function.

Program Example

The program which performs multiplication (ⓢ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 (MUL_TIME)

[Structured ladder]

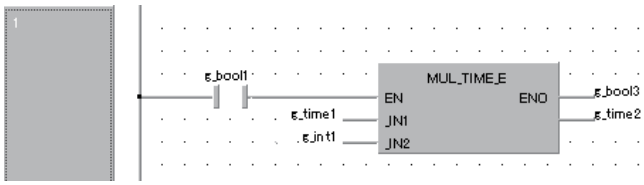


[ST]

```
g_time2:= MUL_TIME(g_time1, g_int1);
```

(b) Function with EN/ENO (MUL_TIME_E)

[Structured ladder]



[ST]

```
g_bool3 := MUL_TIME_E(g_bool1, g_time1, g_int1, g_time2);
```

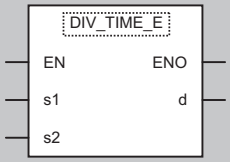
5.8.4 Division

DIV_TIME(_E)

Universal UD High Performance

DIV_TIME(_E) (_E: With EN/ENO)

Structured ladder



ST

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

DIV_TIME_E indicates any of the following functions.

DIV_TIME DIV_TIME_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1:	Input	:Time
	s2:	Input	:ANY_NUM
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error)	:Bit
	d:	Output	:Time

Function

Operation processing

(1) Performs division (ⓢ1 ÷ⓢ2) on time type data input to ⓢ1 and ⓢ2 , and outputs the quotient of the operation result from ⓓ in time type.

(2) The value to be input to ⓢ1 is time type data.

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.)

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

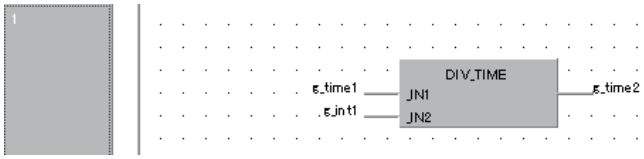
No operation error occurs in the execution of the DIV_TIME(_E) function.

Program Example

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

(a) Function without EN/ENO (DIV_TIME)

[Structured ladder]

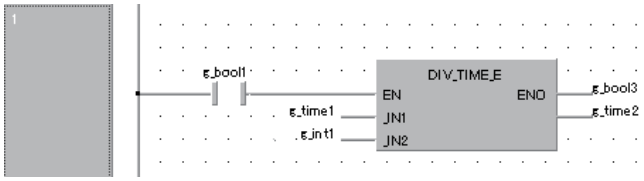


[ST]

```
g_time2:= DIV_TIME(g_time1, g_int1);
```

(b) Function with EN/ENO (DIV_TIME_E)

[Structured ladder]



[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)

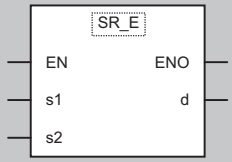
SR(_E)

Universal UD | High Performance

SR(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

SR_E indicates any of the following functions.

SR SR_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1:	Input	:Bit
	s2:		
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error or stop)	:Bit
	d:	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 ⓓ.

(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*1

*1 The previous operation output value is held at ⓓ even if EN is FALSE.

Operation Error

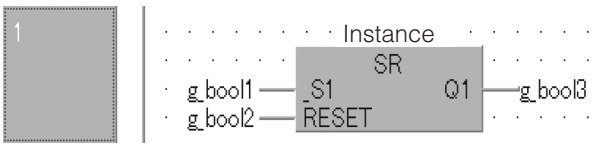
No operation error occurs in the execution of the SR (_E) function.

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]

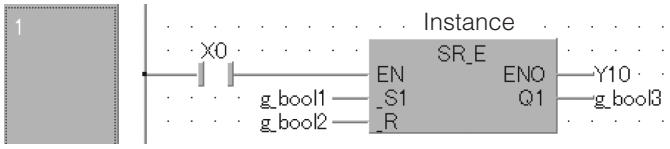


[ST]

```
SR_Instance( g_bool1, g_bool2, g_bool3 );
```

(b) Function with EN/ENO (SR_E)

[Structured ladder]



[ST]

```
SR_E_Instance ( g_bool1, g_bool2, g_bool3 );
```

5.9.2 Standard bistable function blocks (Reset-dominant)

RS(_E)

Universal UD High Performance

RS(_E)

(_E: With EN/ENO)

Structured ladder

ST

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

RS_E indicates any of the following functions.

RS	RS_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1: s2:	Input	:Bit
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error or stop)	:Bit
	d:	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 ⓓ .

(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*1

*1 The previous operation output value is held at ⓓ even if EN is FALSE.

Operation Error

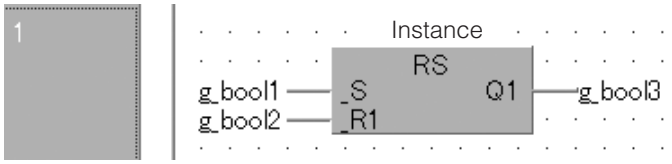
No operation error occurs in the execution of the RS (_E) function.

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]

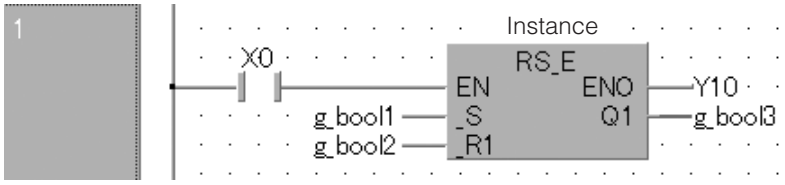


[ST]

```
RS_Instance( g_bool1, g_bool2, g_bool3 );
```

(b) Function with EN/ENO (RS_E)

[Structured ladder]



[ST]

```
RS_E_Instance ( g_bool1, g_bool2, g_bool3 );
```

5.10 Standard Edge Detection Function Blocks

5.10.1 Rising edge detector

R_TRIG(_E)

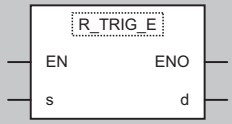
Universal UD

High Performance

R_TRIG(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

[ ] indicates any of the following functions.

R_TRIG R_TRIG_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Bit
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error or stop)	:Bit
	d:	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 ⓓ.

(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*1

*1 The previous operation output value is held at ⓓ even if EN is FALSE.

Operation Error

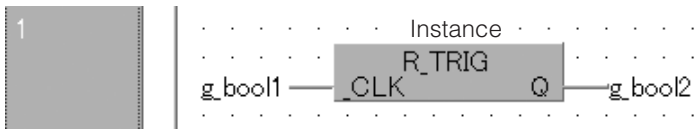
No operation error occurs in the execution of the R_TRIG (_E) function.

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]

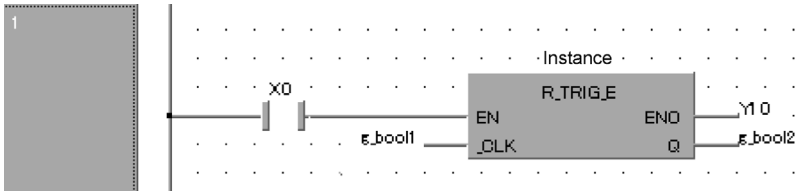


[ST]

```
R_TRIG_Inst ( g_bool1, g_bool2 );
```

(b) Function with EN/ENO (R_TRIG_E)

[Structured ladder]



[ST]

```
R_TRIG_E_Inst ( g_bool1, g_bool2 );
```

5.10.2 Falling edge detector

F_TRIG(_E)

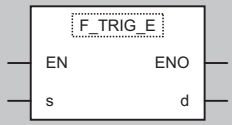
Universal UD

High Performance

F_TRIG(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

: indicates any of the following functions.

F_TRIG F_TRIG_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s:	Input	:Bit
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error or stop)	:Bit
	d:	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 ⓓ.

(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*1

*1 The previous operation output value is held at ⓓ even if EN is FALSE.

Operation Error

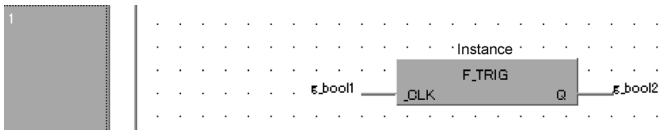
No operation error occurs in the execution of the F_TRIG (_E) function.

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 (R_TRIG)

[Structured ladder]

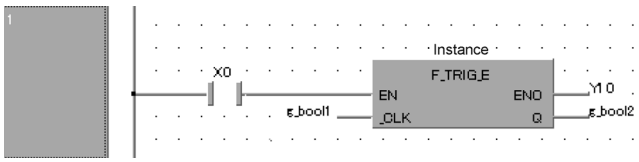


[ST]

```
F_TRIG_Inst ( g_bool1, g_bool2 );
```

(b) Function with EN/ENO (F_TRIG_E)

[Structured ladder]



[ST]

```
F_TRIG_E_Inst ( g_bool1, g_bool2 );
```

5.11 Standard Counter Function Blocks

5.11.1 Up counter

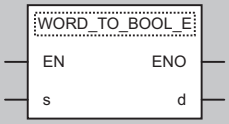
CTU(_E)

Universal UD High Performance

CTU(_E)

(_E: With EN/ENO)

Structured ladder



ST

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

[ ] indicates any of the following functions.

CTU CTU_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1:	Count signal input	:Bit
	s2:	Count reset	:Bit
	n:	Maximum count value	:Word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error or stop)	:Bit
	d1:	Count match output	:Bit
	d2:	Count value	:Word (signed)

Function

Operation processing

Counts ⓓ2 when ⓢ1 is turned ON.

When the count value reaches the value input to ⓝ , ⓓ1 turns ON.

When ⓢ2 is turned ON, ⓓ1 turns OFF and count value ⓓ2 is reset.

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ1 and ⓓ2 .

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ1, ⓓ2
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value*1

*1 The previous operation output value is held at ⓓ1 and ⓓ2 even if EN is FALSE.

Operation Error

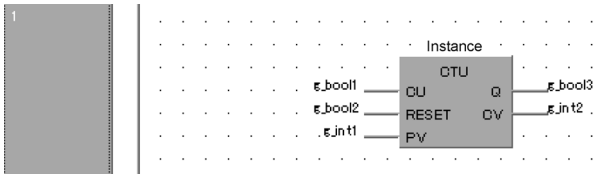
No operation error occurs in the execution of the CTU (_E) function.

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]

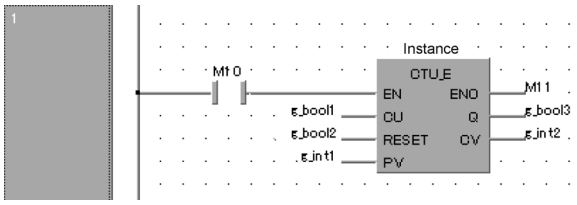


[ST]

```
CTU_Inst ( g_bool1, g_bool2, g_int1, g_bool3, g_int2 );
```

(b) Function with EN/ENO (CTU_E)

[Structured ladder]



[ST]

```
CTU_E_Inst ( g_bool1, g_bool2 , g_int1, g_bool3,  g_int2 );
```

5.11.2 Down counter

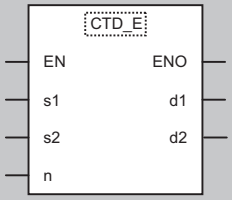
CTD(_E)

Universal UD High Performance

CTD(_E)

(_E: With EN/ENO)

Structured ladder



ST

```
ENO:= CUD_E (EN, s1, s2, n, d1, d2);
```

: indicates any of the following functions.

CTD CTD_E

Input argument,	EN:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s1:	Count signal input	:Bit
	s2:	Count reset	:Bit
	n:	Count start value	:Word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error or stop)	:Bit
	d1:	Count match output	:Bit
	d2:	Count value	:Word (signed)

Function

Operation processing

Counts down (-1) ⓓ2 when ⓢ1 is turned ON.

ⓝ sets the initial value for subtraction.

ⓓ1 turns ON when count value reaches 0.

When ⓢ2 is turned ON, ⓓ1 turns OFF and initial value for subtraction ⓝ is set for count value ⓓ2 .

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ1 and ⓓ2.

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ1, ⓓ2
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value*1

*1 The previous operation output value is held at ⓓ1 and ⓓ2 even if EN is FALSE.

Operation Error

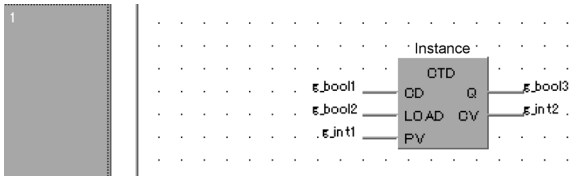
No operation error occurs in the execution of the CTD (_E) function.

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]

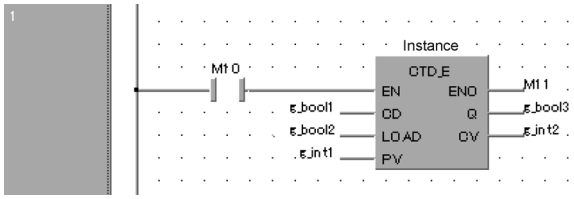


[ST]

```
CTD_Inst ( g_bool1, g_bool2, g_int1, g_bool3, g_int2 );
```

(b) Function with EN/ENO (CTD_E)

[Structured ladder]



[ST]

```
CTD_E_Inst ( g_bool1, g_bool2, g_int1, g_bool3, g_int2)
```

5.11.3 Up/Down counter

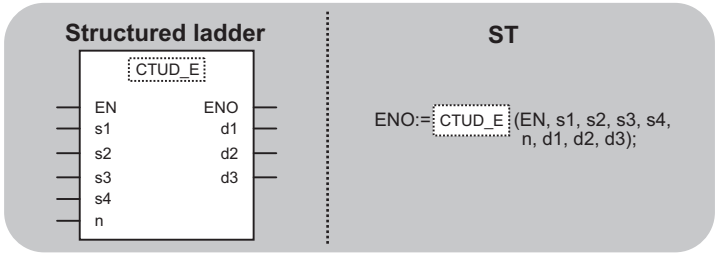
CTUD(_E)

Universal UD

High Performance

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:	Count-up signal input	:Bit
	s2:	Count-down signal input	:Bit
	s3:	Count-up reset	:Bit
	s4:	Count-down reset	:Bit
	n:	Maximum count value	:Word (signed)
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error or stop)	:Bit
	d1:	Count-up match output	:Bit
	d2:	Count-down match output	:Bit
	d3:	Current count value	:Word (signed)

Function

Operation processing

Counts (+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).

The value of (n) is set to (d3) when (s4) is turned ON.

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ1 , ⓓ2 , and ⓓ3 .

(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*1

*1 The previous operation output value is held at ⓓ1 , ⓓ2 , and ⓓ3 even if EN is FALSE.

Operation Error

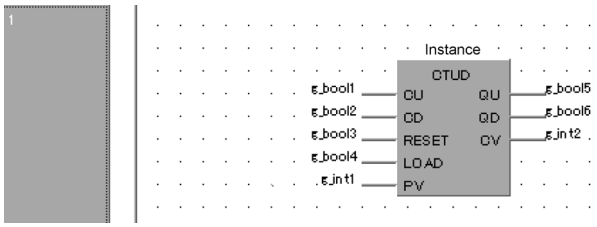
No operation error occurs in the execution of the CTUD (_E) function.

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]

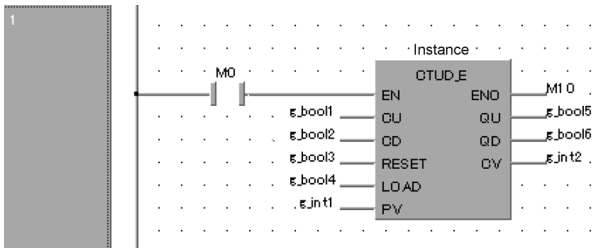


[ST]

```
CTUD_Inst (g_bool1, g_bool2, g_bool3, g_bool4, g_int1, g_bool5, g_bool6, g_int2);
```

(b) Function with EN/ENO (CTD_E)

[Structured ladder]



[ST]

```
CTUD_E_Inst ( g_bool1, g_bool2, g_bool3, g_bool4, g_int1, g_bool5, g_bool6, g_int2);
```

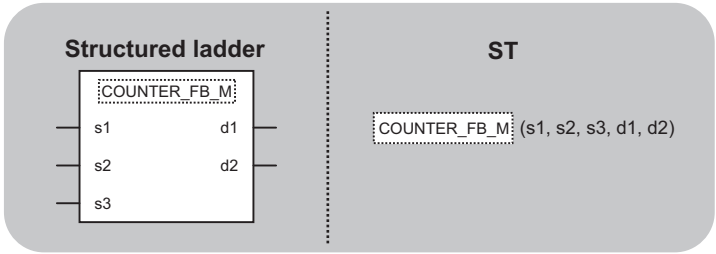
5.11.4 Counter function blocks

COUNTER_FB_M

Universal UD | High Performance *1

*1: The first five digits of the serial number are '04012' or higher.

COUNTER_FB_M



indicates the following function.

COUNTER_FB_M

Input argument,	s1:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s2:	Counter setting value	:Word (signed)
	s3:	Counter initial value	:Word (signed)
Output argument,	d1:	Counter current value	:ANY16
	d2:	Output	:Bit

Function

Operation processing

Counts the detected rising edge (from OFF to ON) of ⓢ1 . It is not counted when ⓢ1 stays ON. The count starts from the value input to ⓢ3 and when the count value reaches the value input to ⓢ2 , ⓓ2 turns ON. The current value is stored in ⓓ1 .

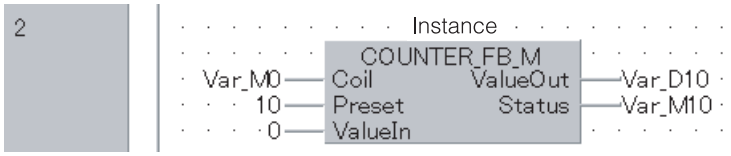
Operation Error

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.

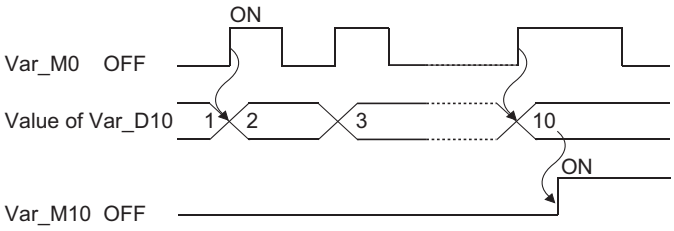
[Structured ladder]



[ST]

```
COUNTER_FB_M_Inst ( Var_M0, 10, 0, Var_D10, Var_M10 );
```

[Timing chart]



5.12 Standard Timer Function Blocks

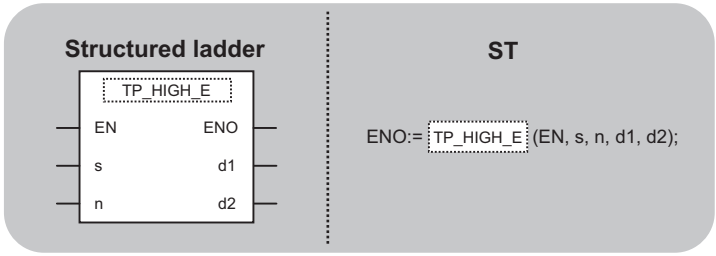
5.12.1 Pulse timer

TP(_E), TP_HIGH(_E)

Universal UD High Performance

TP(_E), TP_HIGH(_E)

(_E: With EN/ENO)



TP_HIGH_E 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:	Input	:Bit
	n:	Output time setting value	:Time
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error or stop)	:Bit
	d1:	Output	:Bit
	d2:	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. It is reset and (d1) turns ON again when (s) is turned from OFF to ON next.

(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.

(2) TP_HIGH(_E)

Uses a high-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.

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ1 and ⓓ2 .

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ1, ⓓ2
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value*1

*1 The previous operation output value is held at ⓓ1 and ⓓ2 even if EN is FALSE.

Operation Error

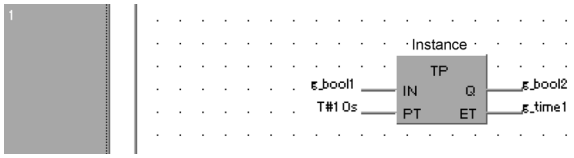
No operation error occurs in the execution of the TP (_E) function.

Program Example

The program which turns ON bit type data of ⓓ1 for 10 seconds after bit type data input to ⓢ is turned ON.

(a) Function without EN/ENO (TP)

[Structured ladder]

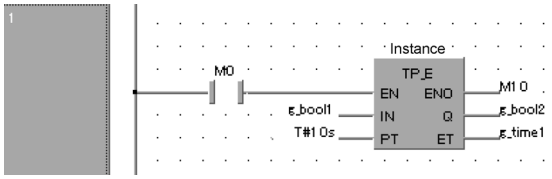


[ST]

```
TP_Inst (g_bool1, T#10s, g_bool2, g_time1);
```

(b) Function with EN/ENO (TP_E)

[Structured ladder]



[ST]

```
TP_E_Inst (g_bool1, T#10s, g_bool2, g_time1);
```

5.12.2 On delay timer

TON(_E), TON_HIGH(_E)

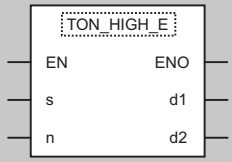
Universal UD

High Performance

TON(_E), TON_HIGH(_E)

(_E: With EN/ENO)

Structured ladder



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:	Input	:Bit
	n:	Delay time setting value	:Time
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error or stop)	:Bit
	d1:	Output	:Bit
	d2:	Elapsed time	:Time

Function

Operation processing

Turns ON ⓓ1 when ⓢ is turned ON after the elapse of the time set to ⓝ . 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.

(2) TON_HIGH(_E)

Uses a high-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.

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ1 and ⓓ2 .

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

EN	ENO	ⓓ1, ⓓ2
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value*1

*1 The previous operation output value is held at ⓓ1 and ⓓ2 even if EN is FALSE.

Operation Error

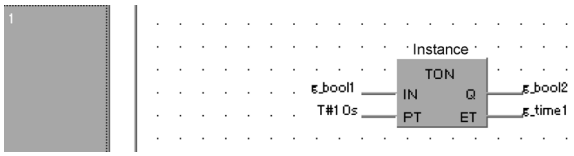
No operation error occurs in the execution of the TON (_E) function.

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]

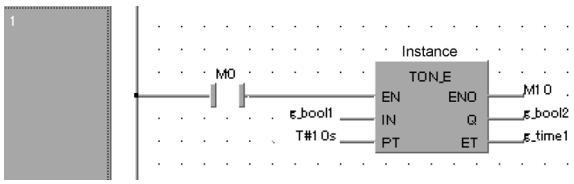


[ST]

```
TON_Inst (g_bool1, T#10s, g_bool2, g_time1);
```

(b) Function with EN/ENO (TON_E)

[Structured ladder]



[ST]

```
TON_E_Inst (g_bool1, T#10s, g_bool2, g_time1);
```

5.12.3 Off delay timer

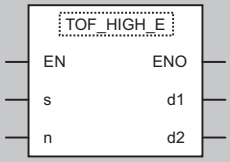
TOF(_E), TOF_HIGH(_E)

Universal UD High Performance

TOF(_E), TOF_HIGH(_E)

(_E: With EN/ENO)

Structured ladder



ST

ENO:= TOF_HIGH_E (EN, s, n, d1, d2);

TOF_HIGH_E 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:	Input	:Bit
	n:	Delay time setting value	:Time
Output argument,	ENO:	Output status (TRUE: Normal, FALSE: Error or stop)	:Bit
	d1:	Output	:Bit
	d2:	Elapsed time	:Time

Function

Operation processing

Turns ON ⓓ1 when ⓢ is turned ON.

Turns OFF ⓓ1 when ⓢ is turned from ON to OFF after the elapse of the time set to ⓝ . Elapsed time until ⓓ1 is turned OFF is set to ⓓ2 .

When ⓢ is turned ON again, ⓓ1 turns ON and the elapsed time is reset.

(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.

(2) TP_HIGH(_E)
Uses a high-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.

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ⓓ1 and ⓓ2 .

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	ⓓ1, ⓓ2
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value*1

*1 The previous operation output value is held at ⓓ1 and ⓓ2 even if EN is FALSE.

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 ⓓ1 when bit type data input to ⓢ is turned ON, and turns ⓓ1 OFF 10 seconds after ⓢ is turned OFF.

(a) Function without EN/ENO (TOF)

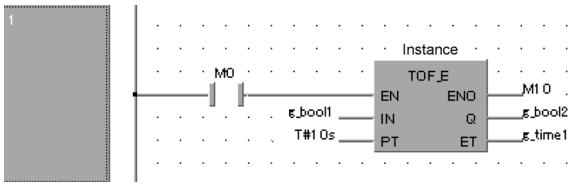
[Structured ladder]

[ST]

```
TOF_Inst (g_bool1, T#10s, g_bool2, g_time1);
```

(b) Function with EN/ENO (TOF_E)

[Structured ladder]



[ST]

```
TOF_E_Inst (g_bool1, T#10s, g_bool2, g_time1);
```

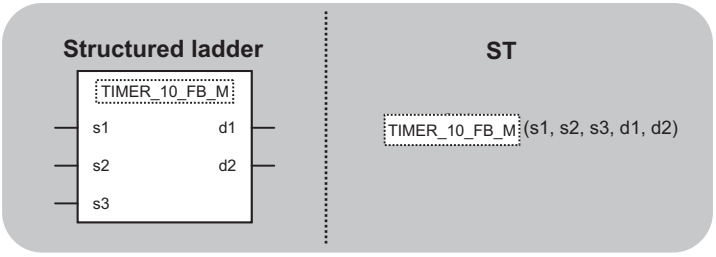
5.12.4 Timer function blocks

TIMER_10_FB_M

Universal UD | High Performance *1

*1: The first five digits of the serial number are '04012' or higher.

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:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit
	s2:	Timer setting value	:Word (signed)
	s3:	Timer initial value	:Word (signed)
Output argument,	d1:	Timer current value	:ANY16
	d2:	Output	:Bit

Function

Operation processing

(1) TIMER_10_FB_M

(a) Starts measuring the current value when the executing condition of ⓢ1 turns ON.
Starts measuring from the value input to ⓢ3 × 10ms, and when the measuring value reaches to the value input to ⓢ2 × 10ms, ⓓ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.

(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.

(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 is from 0.1 to 100ms and it can be changed by unit of 0.1ms.
This setting is set in the PLC system setting of the PLC parameter.

(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.

(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.

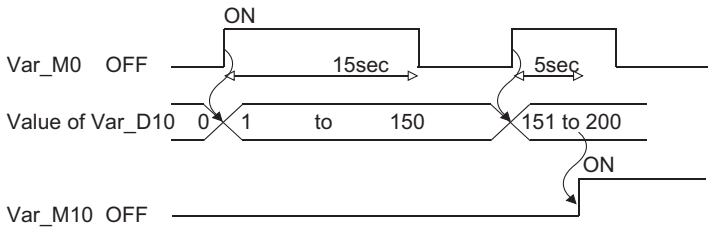
The current value is output from ⓓ1 .

(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_HIGHFB_M).

- Low-speed retentive timer : Low-speed timer
- High-speed retentive timer : High-speed timer

[Timing chart]



Operation Error

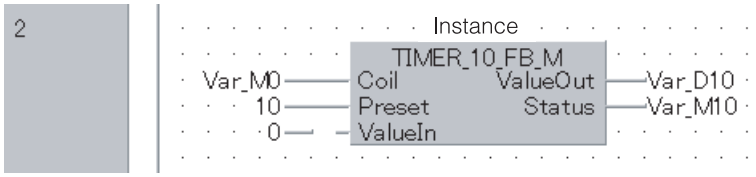
No operation error occurs in the execution of the timer function blocks.

Program Example

(1) TIMER_10_FB_M

The program which starts measuring from 0 when the executing condition of ⓢ1 turns ON, and when the measuring value reaches to the value input to ⓢ2 × 10ms, ⓓ2 turns ON.

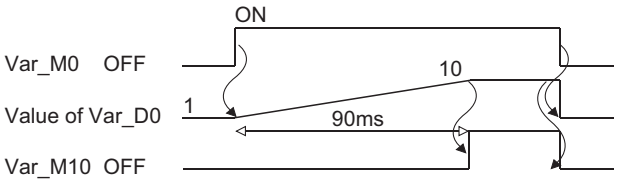
[Structured ladder]



[ST]

TIMER_10_FB_M_Inst (Var_M0, 10, 0, Var_D10, Var_M10);

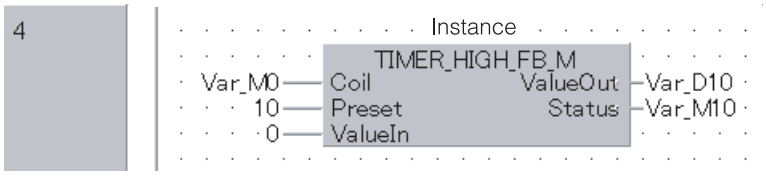
[Timing chart]



(2) TIMER_HIGH_FB_M

The program which starts measuring from 0 when the executing condition of ⓢ1 turns ON, and when the measuring value reaches to the value input to ⓢ2 × 0.1 to 100ms, ⓓ2 turns ON.

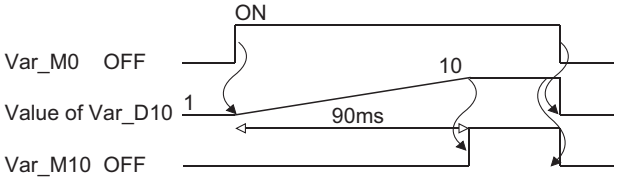
[Structured ladder]



[ST]

TIMER_HIGH_FB_M_Inst (Var_M0, 10, 0, Var_D10, Var_M10);

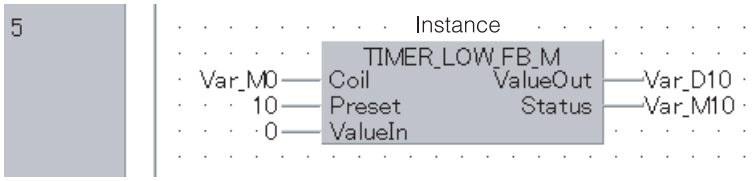
[Timing chart]



(3) TIMER_LOW_FB_M

The program which starts measuring from 0 when the executing condition of ⓢ1 turns ON, and when the measuring value reaches to the value input to ⓢ2 × 1 to 1000ms, ⓓ2 turns ON.

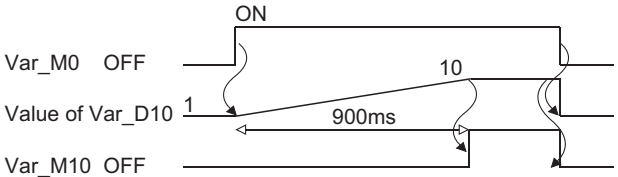
[Structured ladder]



[ST]

```
TIMER_LOW_FB_M_Inst ( Var_M0, 10, 0, Var_D10, Var_M10 );
```

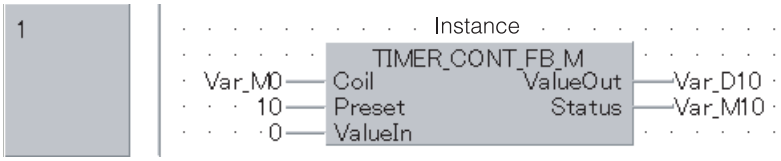
[Timing chart]



(4) TIMER_CONT_FB_M

The program which measures the time during ⓢ1 is ON, and when the measuring value reaches to the value input to ⓢ2 × 1 to 1000ms, ⓓ2 turns ON.

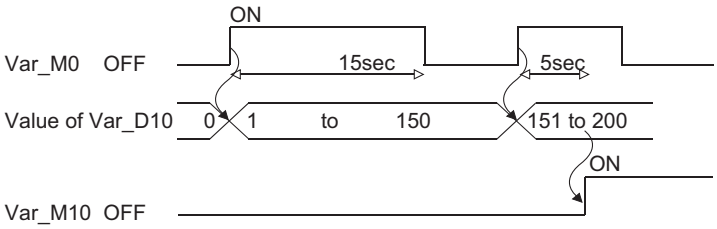
[Structured ladder]



[ST]

```
TIMER_CONT_FB_M_Inst ( Var_M0, 10, 0, Var_D10, Var_M10 );
```

[Timing chart]



APPENDIX

Appendix 1 Correspondence between Generic Data Types and Devices

The following table shows the correspondence between generic data types and devices.

Table App. 1-1 Correspondence between generic data types and devices

Device				
Classification	**Type**	**Device name**	**Device symbol**	
Internal user device	Bit device	Input	X	
		Output	Y	
		Internal relay	M	
		Latch relay	L	
		Annunciator	F	
		Edge relay	V	
		Step relay	S	
		Link special relay	SB	
		Link relay	B	
		Timer contact *1	TS	
		Timer coil*1	TC	
		Retentive timer contact*1	STS	
		Retentive timer coil*1	STC	
		Counter contact*1	CS	
		Counter coil	CC	
	Word device	Timer current value	T or TN*1	
		Retentive timer current value	ST or STN*1	
		Counter current value	C or CN*1	
		Data register	D	
		Link register	W	
		Link special register	SW	
Internal system device	Bit device	Function input	FX	
		Function output	FY	
		Special relay	SM	
	Word device	Function register	FD	
		Special register	SD	

*1 : Can be used for digit specification.
*2 : Can be used for bit specification.

Generic data type												
ANY											ANY	
ANY_SIMPLE									Array	Structure		
ANY_BIT			ANY_NUM				Time	String				
			ANY_INT		ANY_REAL							
Bit	Word (unsigned)/ 16-bit string	Double word (unsigned)/ 32-bit string	Word (signed)	Double word (signed)	Single-precision real	Double-precision real					ANY16	ANY32
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	×	×	×	×	×	×	×	×	×	×	×	×
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
×	○	×	○	×	×	×	×	×	×	×	○	×
×	○	×	○	×	×	×	×	×	×	×	○	×
×	○	×	○	×	×	×	×	×	×	×	○	×
○*2	○	×	○	×	×	×	×	×	×	×	○	×
○*2	○	×	○	×	×	×	×	×	×	×	○	×
○*2	○	×	○	×	×	×	×	×	×	×	○	×
–	–	–	–	–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–	–	–	–	–
○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
–	–	–	–	–	×	×	×	×	×	×		–
○*2	○	×	○	×	×	×	×	×	×	×	○	×

A

APPENDIX

Device				
Classification	**Type**	**Device name**	**Device symbol**	
Link direct device	Bit device	Link input	Jn\X	
		Link output	Jn\Y	
		Link relay	Jn\B	
		Link special relay	Jn\SB	
	Word device	Link register	Jn\W	
		Link special register	Jn\SW	
Intelligent function module device	Word device	Intelligent function module device	Un\G	
Index register	Word device	Index register	Z	
File register	Word device	File register	R or ZR	
Nesting	–	Nesting	N	
Pointer	–	Pointer	P	
		Interrupt pointer	I	
Constant	–	–	K, H	
			E	
Character string constant	–	–	'Character string' or "Character string"	

*1 : Can be used for digit specification.
*2 : Can be used for bit specification.

	Generic data type												
	ANY											ANY	
	ANY_SIMPLE												
	ANY_BIT			ANY_NUM									
				ANY_INT		ANY_REAL							
	Bit	Word (unsigned)/ 16-bit string	Double word (unsigned)/ 32-bit string	Word (signed)	Double word (signed)	Single-precision real	Double-precision real	Time	String	Array	Structure	ANY16	ANY32
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	×	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	–	–	–	–	–	–	–	–	–	–	–	–	–
	–	–	–	–	–	–	–	–	–	–	–	–	–
	–	–	–	–	–	–	–	–	–	–	–	–	–
	○	○	○	○	○	○	×	×	×	×	×	○	○
	×	×	×	×	×	○	○	×	×	×	×	×	×
	×	×	×	×	×	×	×	×	○	×	×	×	×

Appendix 2 Correspondence between Devices and Addresses

The following table shows the correspondence between devices and addresses.

Table App. 2-1 Correspondence between devices and addresses

Device			Expressing method		Example of correspondence between device and address	
			Device	Address	Device	Address
Input		X	Xn	%IXn	X7FF	%IX2047
Output		Y	Yn	%QXn	Y7FF	%QX2047
Internal relay		M	Mn	%MX0.n	M2047	%MX0.2047
Latch relay		L	Ln	%MX8.n	L2047	%MX8.2047
Annunciator		F	Fn	%MX7.n	F1023	%MX7.1023
Special relay		SM	SMn	%MX10.n	SM1023	%MX10.1023
Function input		FX	FXn	None	FX10	None
Function output		FY	FYn	None	FY10	None
Edge relay		V	Vn	%MX9.n	V1023	%MX9.1023
Direct access input		DX	DXn	%IX1.n	DX7FF	%IX1.2047
Direct access output		DY	DYn	%QX1.n	DY7FF	%QX1.2047
Timer	Contact	TS	Tn	%MX3.n	TS511	%MX3.511
	Coil	TC	Tn	%MX5.n	TC511	%MX5.511
	Current value	TN	Tn	%MW3.n %MD3.n	TN511 T511	%MW3.511 %MD3.511
Counter	Contact	CS	Cn	%MX4.n	CS511	%MX4.511
	Coil	CC	Cn	%MX6.n	CC511	%MX6.511
	Current value	CN	Cn	%MW4.n %MD4.n	CN511 C511	%MW4.511 %MD4.511
Retentive timer	Contact	SS	STn	%MX13.n	STS511	%MX13.511
	Coil	SC	STn	%MX15.n	STC511	%MX15.511
	Current value	SN	STn	%MW13.n %MD13.n	STN511 ST511	%MW13.511 %MD13.511
Data register		D	Dn	%MW0.n %MD0.n	D11135	%MW0.11135 %MD0.11135
Special register		SD	SDn	%MW10.n %MD10.n	SD1023	%MW10.1023 %MD10.1023
Function register		FD	FDn	None	FD0	None
Link relay		B	Bn	%MX1.n	B7FF	%MX1.2047
Link special relay		SB	SBn	%MX11.n	SB3FF	%MX11.1023
Link register		W	Wn	%MW1.n %MD1.n	W7FF	%MW1.2047 %MD1.2047
Link special register		SW	SWn	%MW11.n %MD11.n	SW3FF	%MW11.1023 %MD11.1023
Intelligent function module device		G	Ux\Gn	%MW14.x.n %MD14.x.n	U0\G65535	%MW14.0.65535 %MD14.0.65535
File register		R	Rn	%MW2.n %MD2.n	R32767	%MW2.32767 %MD2.32767
Pointer		P	Pn	"" (null character)	P299	None
Interrupt pointer		I	In	None	–	–
Nesting		N	Nn	None	–	–
Index register		Z	Zn	%MW7.n %MD7.n	Z9	%MW7.9 %MD7.9

Table App. 2-2 Correspondence between devices and addresses

Device		Expressing method		Example of correspondence between device and address	
		Device	Address	Device	Address
Step relay	S	Sn	%MX2.n	S127	%MX2.127
SFC transition device	TR	TRn	%MX18.n	TR3	%MX18.3
SFC block device	BL	BLn	%MX17.n	BL3	%MX17.3
Link input	J	Jx\Xn	%IX16.x.n	J1\X1FFF	%IX16.1.8191
Link output		Jx\Yn	%QX16.x.n	J1\Y1FFF	%QX16.1.8191
Link relay		Jx\Bn	%MX16.x.1.n	J2\B3FFF	%MX16.2.1.16383
Link register		Jx\Wn	%MW16.x.1.n %MD16.x.1.n	J2\W3FFF	%MW16.2.1.16383 %MD16.2.1.16383
Link special relay		Jx\SBn	%MX16.x.11.n	J2\SB1FF	%MX16.2.11.511
Link special register		Jx\SWn	%MW16.x.11.n %MD16.x.11.n	J2\SW1FF	%MW16.2.11.511
File register	ZR	ZRn	%MW12.n %MD12.n	ZR32767	%MW12.32767 %MD12.32767

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.

6. Product application

(1) In using the Mitsubishi MELSEC programmable controller, the usage conditions shall be that the application will not lead to a major accident even if any problem or fault should occur in the programmable controller device, and that backup and fail-safe functions are systematically provided outside of the device for any problem or fault.

(2) The Mitsubishi MELSEC programmable controller has been designed and manufactured for applications in general industries, etc. Thus, applications in which the public could be affected such as in nuclear power plants and other power plants operated by respective power companies, and applications in which a special quality assurance system is required, such as for Railway companies or Public service purposes shall be excluded from the programmable controller applications. In addition, applications in which human life or property that could be greatly affected, such as in aircraft, medical applications, incineration and fuel devices, manned transportation, equipment for recreation and amusement, and safety devices, shall also be excluded from the programmable controller range of applications.
However, in certain cases, some applications may be possible, providing the user consults their local Mitsubishi representative outlining the special requirements of the project, and providing that all parties concerned agree to the special circumstances, solely at the users discretion.

QCPU

Structured Programming Manual (Application Functions)

MODEL	Q-KP-OK-E
MODEL CODE	13JW08
SH(NA)-080784ENG-A(0807)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