QCPU

MITSUBISHI

Structured Programming Manual

(Application Functions)



Mitsubishi Programmable Controller





(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

This manual confers no industrial property rights or any rights of any other kind, nor does it confer any patent licenses. Mitsubishi Electric Corporation cannot be held responsible for any problems involving industrial property rights which may occur as a result of using the contents noted in this manual.

© 2008 MITSUBISHI ELECTRIC CORPORATION

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

SAFETY PRECAUTIONS	A - 1
REVISIONS	A - 2
NTRODUCTION	
CONTENTS	
MANUALS	A - 6

1. OVERVIEW	1 - 1 to 1 - 4
1.1 Purpose of This Manual	1 - 2
1.2 Generic Terms and Abbreviations in This Manual	1 - 4

2. FUNCTION TABLES 2 - 1 to 2 - 6 2.1 How to Read Eurotion Tables 2 - 1 to 2 - 6

2.	1 HC	ow to Read Function Tables	2 - 2
	2.1.1	Type conversion functions	2 - 3
	2.1.2	Standard functions of one numeric variable	2 - 4
	2.1.3	Standard arithmetic functions	2 - 4
	2.1.4	Standard bitwise Boolean functions	2 - 4
	2.1.5	Standard selection functions	2 - 5
	2.1.6	Standard comparison functions	2 - 5
	2.1.7	Standard character string functions	2 - 5
	2.1.8	Functions of time data types	2 - 5
	2.1.9	Standard bistable function blocks	2 - 5
	2.1.1) Standard edge detection function blocks	2 - 6
		1 Standard counter function blocks	
	2.1.12	2 Standard timer function blocks	2 - 6

3. FUNCTIONS	3 - 1 to 3 - 12
3.1 Input Pins Variable Function	3 - 2
3.2 Functions with EN	3 - 3
3.3 Labels	3 - 4
3.3.1 Global labels	
3.3.2 Local labels	
3.3.3 Label classes	
3.3.4 Setting labels	
3.4 Data Types	3 - 6
3.5 Device and Address	3 - 8
3.5.1 Device	
3.5.2 Address	
3.6 Expressing Methods of Constants	3 - 10
3.7 Precautions on Programming	3 - 10

3.7.1	Precautions on assigning a name	3 -	1	0
••••		-		-

4. HOW TO READ FUNCTIONS

5.

.3.3	Subtrac
.3.4	Divisior

A-4

5.	APPL	ICATION FUNCTIONS	5 - 1 to 5 -	- 218
5	5.1 Typ	e Conversion Functions		5 - 2
	5.1.1	Bit type \rightarrow word (signed), double word (signed) type conversion		5 - 2
	5.1.2	Bit type \rightarrow string type conversion		5 - 5
	5.1.3	Bit type \rightarrow word (unsigned)/16-bit string, double word (unsigned)/32-bit string type conversion		
	5.1.4	Bit type \rightarrow time type conversion		
	5.1.5	Word (signed) type \rightarrow double word (signed) type conversion		
	5.1.6	Double word (signed) type \rightarrow word (signed) type conversion		
		Word (signed), double word (signed) type \rightarrow bit type conversion		
		Word (signed), double word (signed) type \rightarrow single-precision real type conversion Word (signed), double word (signed) type \rightarrow double precision real type conversion		
		Word (signed), double word (signed) type \rightarrow double-precision real type conversi Word (signed), double word (signed) type \rightarrow string type conversion		
		Word (signed), double word (signed) type \rightarrow string type conversion		
		Word (signed), double word (signed) type \rightarrow word (unsigned)/10-bit string type conversion		
		Word (signed), double word (signed) type \rightarrow double word (unsigned)/52 bit stiming type conversion		
		Word (signed), double word (signed) type \rightarrow time type conversion		
		Single-precision real type \rightarrow word (signed), double word (signed) type conversio		
		Double-precision real type \rightarrow word (signed), double word (signed) type conversion		
		Single-precision real type \rightarrow double-precision real type conversion		
		Double-precision real type \rightarrow single-precision real type conversion		
	5.1.19	Single-precision real type \rightarrow string type conversion		5 - 57
	5.1.20	Word (unsigned)/16-bit string, double word (unsigned)/32-bit string type \rightarrow bit type conversion		5 - 61
	5.1.21	Word (unsigned)/16-bit string type \rightarrow word (signed), double word (signed) type conversion	n	5 - 64
	5.1.22	Double word (unsigned)/32-bit string type \rightarrow word (signed), double word (signed) type conversion		5 - 67
	5.1.23	Word (unsigned)/16-bit string type \rightarrow double word (unsigned)/32-bit string type conversion	n	. 5 - 70
		Double word (unsigned)/32-bit string type \rightarrow word (unsigned)/16-bit string type conversion		
		Word (unsigned)/16-bit string, double word (unsigned)/32-bit string type \rightarrow string type conversion		
		Word (unsigned)/16-bit string, double word (unsigned)/32-bit string type \rightarrow time type conversion		
		String type \rightarrow bit type conversion		
		String type \rightarrow word (signed), double word (signed) type conversion		
		String type \rightarrow single-precision real type conversion		
		String type \rightarrow word (unsigned)/16-bit string, double word (unsigned)/32-bit string type conversion		
		String type \rightarrow time type conversion String type \rightarrow BCD type conversion		
		BCD type \rightarrow BCD type conversion BCD type \rightarrow word (signed), double word (signed) type conversion		
		BCD type \rightarrow string type conversion.		
		Time type \rightarrow bit type conversion		
		Time type \rightarrow word (signed), double word (signed) type conversion		
		Time type \rightarrow string type conversion		
		Time type \rightarrow word (unsigned)/16-bit string, double word (unsigned)/32-bit string type conversion		
5		ndard Functions of One Numeric Variable		5 - 119
	5.2.1	Absolute value		5 - 119
5	5.3 Sta	ndard Arithmetic Functions		5 - 122
	5.3.1	Addition		5 - 122
	5.3.2	Multiplication		5 - 125
	5.3.3	Subtraction		5 - 128
	5.3.4	Division		5 - 131

4 - 1 to 4 - 4

5.3.5 Modulus operation	
5.3.6 Exponentiation	
5.3.7 Move operation	
5.4 Standard Bitwise Boolean Functions	5 - 143
5.4.1 Boolean AND, boolean OR, boolean exclusive OR, and boolean NOT	F 5 - 143
5.5 Standard Selection Functions	5 - 148
5.5.1 Selection	
5.5.2 Multiplexer	
5.6 Standard Comparison Functions	5 - 154
5.6.1 Comparison	
5.7 Standard Character String Functions	5 - 157
5.7.1 Extract mid string	
5.7.2 String concatenation	
5.7.3 String insertion	
5.7.4 String deletion	
5.7.5 String replacement	
5.8 Functions of Time Data Type	5 - 172
5.8.1 Addition	
5.8.2 Subtraction	
5.8.3 Multiplication	
5.8.4 Division	
5.9 Standard Bistable Function Blocks	5 - 184
5.9.1 Standard bistable function blocks (Set-dominant)	
5.9.2 Standard bistable function blocks (Reset-dominant)	
5.10 Standard Edge Detection Function Blocks	5 - 190
5.10.1 Rising edge detector	
5.10.2 Falling edge detector	
5.11 Standard Counter Function Blocks	5 - 194
5.11.1 Up counter	
5.11.2 Down counter	
5.11.3 Up/Down counter	
5.11.4 Counter function blocks	
5.12 Standard Timer Function Blocks	5 - 205
5.12.1 Pulse timer	
5.12.2 On delay timer	
5.12.3 Off delay timer	
5.12.4 Timer function blocks	
APPENDIX	App - 1 to App - 8
Appendix 1 Correspondence between Generic Data Types and Devices	App - 2
Appendix 2 Correspondence between Devices and Addresses	Арр - 6
INDEX	Index - 1 to Index - 4

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	SH-080782ENG
programs.	(13JW06)
(Sold separately)	
QCPU Structured Programming Manual (Common Instructions)	
Explains the specifications and functions of sequence instructions, basic instructions, and application instructions that can be	SH-080783ENG
used in structured programs.	(13JW07)
(Sold separately)	
QCPU Structured Programming Manual (Special Instructions)	
Explains the specifications and functions of instructions for network modules, intelligent function modules, and PID control	SH-080785ENG
functions that can be used in structured programs.	(13JW09)
(Sold separately)	

(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.



1.1	Purpose of This Manual	1-2
1.2	Generic Terms and Abbreviations in This Manual	1-4

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

		GX Works2 Installation Instructions		orks2 's Manual		Works2 Version Works2	
Purŗ	Purpose					-	
		-	Simple Project	Structured Project	Common	Simple Project	Structured Project
Installation	Learning the operating environment and installation method	Details					
	Learning the basic operations and operating procedures		Details		Outline	Outline	
Operation of Simple project	Learning the functions and operation methods for programming				Outline	Details	
	Learning all functions and operation methods except for programming				Details		
	Learning the basic operations and operating procedures			Details	Outline		Outline
Operation of Structured project	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
I								
		Fundamentals	Common Instructions	Special Instructions	Application Functions	Common Instructions	PID Control Instructions	-
	Learning the types and details of common instructions, descriptions of error codes, special relays, and special registers					Details		
Programming in Simple project	Learning the types and details of instructions for intelligent function modules							Details
	Learning the types and details of instructions for network modules							Details
	Learning the types and details of instructions for the PID control function						Details	
	Learning the 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
Programming in Structured project	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

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



FUNCTION TABLES

2.1	How to Read Function	Tables	-2
		140.000	-

Function name	Argument	Processing details	Page
ADD(_E)	€ , , , , , , , , ⊗ , (Number of pins variable)	Outputs the sum () +@ ++@) of input values.	5-139
MUL(_E)	 Image: Strain (Strain Strain S	Outputs the product ((() $\times @ \times \cdots \times @$) of input values.	5-142
SUB(_E)	₫, @, @	Outputs the difference (@ - @) between input values.	5-145
DIV(_E)	\$1, ♥, @	Outputs the quotient $(\textcircled{s} + \textcircled{o})$) of input values.	5-148
† (1)	↑ ②	↑ 3	† (4)

Description

- ①Indicates the functions used in a program. 'Function name(_E)' is used as a function with EN.
- 2Indicates the arguments of the function.
 - (s) : SourceStores data before operation.
 - (d) : Destination......Indicates the destination of data after operation.

(Number of pins variable).....Allows the number of $\circledast\,$ (source) to be changed in the range from 2 to 8.

Changing the number of pins



3Indicates the processing details of each function.

④Indicates the pages on which the functions are explained.

2.1.1 Type conversion functions

Function name	Argument	Processing details	Page
BOOL_TO_INT(_E)	s, d	Converts bit type data into word (signed) or double word (signed) type	5-2
BOOL_TO_DINT(_E)	s, d	data.	5-2
BOOL_TO_STR(_E)	s, d	Converts bit type data into string type data.	5-5
BOOL_TO_WORD(_E)	s, d	Converts bit type data into word (unsigned)/16-bit string or double	5-7
BOOL_TO_DWORD(_E)	s, d	word (unsigned)/32-bit string type date.	5-7
BOOL_TO_TIME(_E)	s, d	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)	s, d	Converts double word (signed) type data into word (signed) type data.	5-16
INT_TO_BOOL(_E)	s, d	Converts word (signed) or double word (signed) type data into bit type	5-19
DINT TO BOOL(E)	s, d	data.	5-19
INT_TO_REAL(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into single-	5-23
DINT_TO_REAL(_E)	(s), (d)	precision real type data.	5-23
INT_TO_LREAL(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into double-	5-26
DINT_TO_LREAL(_E)	s, d	precision real type data.	5-26
	s, d		5-20
INT_TO_STR(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into string	5-29
DINT_TO_STR(_E)	(s), (d)	type data.	
INT_TO_WORD(_E)	s, d	Converts word (signed) or double word (signed) type data into word	5-33
DINT_TO_WORD(_E)	s, d	(unsigned)/16-bit string type data.	5-33
INT_TO_DWORD(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into double	5-36
DINT_TO_DWORD(_E)	(s), (d)	word (unsigned)/32-bit string type data.	5-36
INT_TO_BCD(_E)	(s), (d)	Converts word (signed) or double word (signed) type data into BCD	5-39
DINT_TO_BCD(_E)	(s), (d)	type data.	5-39
INT_TO_TIME(_E)	(s), (d)	Converts word (signed) type data into time type data.	5-42
DINT_TO_TIME(_E)	(s), (d)		5-42
REAL_TO_INT(_E)	s, d	Converts single-precision real type data into word (signed) or double	5-45
REAL_TO_DINT(_E)	s, d	word (signed) type data.	5-45
LREAL_TO_INT(_E)	s, d	Converts double-precision real type data into word (signed) or double	5-48
LREAL_TO_DINT(_E)	s, d	word (signed) type data.	5-48
REAL_TO_LREAL(_E)	s, d	Converts single-precision real type data into double-precision real type data.	5-51
LREAL_TO_REAL(_E)	(s) , (d)	Converts double-precision real type data into single-precision real type data.	5-54
REAL_TO_STR(_E)	s , d	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-	5-61
DWORD_TO_BOOL(_E)	s, d	bit string type data into bit type data.	5-61
WORD_TO_INT(_E)	s, d	Converts word (unsigned)/16-bit string type data into word (signed) or	5-64
WORD_TO_DINT(_E)	(s), (d)	double word (signed) type data.	5-64
DWORD_TO_INT(_E)	(s), (d)	Converts double word (unsigned)/32-bit string type data into word	5-67
	(s), (d)	(signed) or double word (signed) type data.	5-67
DWORD_TO_DINT(_E)	s, d		5-07
WORD_TO_DWORD(_E)	s, d	Converts word (unsigned)/16-bit string type data into double word (unsigned)/32-bit string type data.	5-70
DWORD_TO_WORD(_E)	s, d	Converts double word (unsigned)/32-bit string type data into word (unsigned)/16-bit string type data.	5-73
WORD_TO_STR(_E)	s, d	Converts word (unsigned)/16-bit string or double word (unsigned)/32-	5-76
DWORD_TO_STR(_E)	s, d	bit string type data into string type data.	5-76
WORD_TO_TIME(_E)	s, d	Converts word (unsigned)/16-bit string or double word (unsigned)/32-	5-79
DWORD_TO_TIME(_E)	s, d	bit string type data into time type data.	5-79
STR_TO_BOOL(_E)		Converts string type data into bit type data.	5-82
STR_TO_INT(_E)	s,d	Converts string type data into word (signed) or double word (signed)	5-85
STR_TO_DINT(_E)	s, d	type data.	5-85
STR_TO_REAL(_E)	s, d	Converts string type data into single-precision real type data.	5-88
	(s), (d)	conside daming type data into omgre predicion real type data.	0.00

Function name	Argument	Processing details	Page
STR_TO_WORD(_E)	s, d	Converts string type data into word (unsigned)/16-bit string or double	5-92
STR_TO_DWORD(_E)	s, d	word (unsigned)/32-bit string type data.	5-92
STR_TO_TIME(_E)	s, d	Converts string type data into time type data.	5-95
STR_TO_BCD(_E)	s, d	Converts string type data into BCD type data.	5-98
BCD_TO_INT(_E)	s, d	Converts BCD type data into word (signed) or double word (signed)	5-101
BCD_TO_DINT(_E)	s , d	type data.	5-101
BCD_TO_STR(_E)	s, d	Converts BCD type data into string type data.	5-104
TIME_TO_BOOL(_E)	s, d	Converts time type data into bit type data.	5-107
TIME_TO_INT(_E)	s, d	Converts time type data into word (signed) or double word (signed)	5-110
TIME_TO_DINT(_E)	s, d	type data.	5-110
TIME_TO_STR(_E)	s, d	Converts time type data into string type data.	5-113
TIME_TO_WORD(_E)	s, d	Converts time type data into word (unsigned)/16-bit string or double	5-116
TIME_TO_DWORD(_E)	s, d	word (unsigned)/32-bit string type data.	5-116

2.1.2 Standard functions of one numeric variable

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

2.1.3 Standard arithmetic functions

Function name	Argument	Processing details	Page
ADD(_E)	I ,I	Outputs the sum (ଶ) +֎ +···+֎) of input values.	5-122
MUL(_E)	 (e) , (e) ,	Outputs the product ((() × ((5-125
SUB(_E)	s), s2, d	Outputs the difference (5-128
DIV(_E)	s), ©, d	Outputs the quotient (ⓓ ÷֎) of input values.	5-131
MOD(_E)	s), ©, d	Outputs the remainder after division of input values ((a) \div (b)).	5-134
EXPT(_E)	s), 2, d	Outputs the exponentiation of an input value.	5-137
MOVE(_E)	I ,I ,II ,II ,II ,II ,II ,II ,II ,II ,I	Outputs the substitution of an input value.	5-140

2.1.4 Standard bitwise Boolean functions

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

2.1.5 Standard selection functions

Function name	Argument	Processing details	Page
SEL(_E)	s1, s2, s3, d	Outputs the value selected from the input values.	5-148
MUX(_E)	€1, © ,⊛, @ (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)	I ,I		5-154
GE(_E)	In the second secon		5-154
EQ(_E)	In the second secon	Outputs the comparison value of an input value.	5-154
LE(_E)	I verting and set of the set		5-154
LT(_E)	 Image: Strain (Strain Strain S		5-154
NE(_E)	s), 2, d		5-154

2.1.7 Standard character string functions

Function name	Argument	Processing details	Page	
MID(_E)	s), \$2, \$3, d	Outputs the specified number of characters, extracted from the specified start position in the input character string.	5-157	
CONCAT(_E)	s), 2, 0	Concatenates two character strings and outputs the operation result.	5-160	
INSERT(_E)	€ ,	Inserts a character string between other character strings and outputs the operation result.	5-163	
DELETE(_E)	s), 2, s), d	Deletes the specified range in a character string and outputs the operation result.	5-166	
REPLACE(_E)	SI, Q, SI, G, G, d	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)	s), s2, d	Outputs the sum () +) of the input values (time type).	5-172
SUB_TIME(_E)	s), s2, d	Outputs the difference (lo -lo) of input values (time type).	5-175

2.1.9 Standard bistable function blocks

Function name	Argument	Processing details	Page
SR(_E)	s), 2, d	Discriminates two input values and outputs 1 (TRUE) or 0 (FALSE).	5-184
RS(_E)	€1, 22, 0	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)	st , d	Detects the rising edge of a signal and outputs pulse signals.	5-190
F_TRIG(_E)	s), d	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)	61, 62, 63, 61, 62	Counts the number of times that the signal turns ON.	5-194
CTD(_E)	61, 62, 63, 61, 62	Counts down the number of times that the signal turns ON.	5-197
CTUD(_E)	6), Q, 3, 9 6), A), Q, A)	Counts/counts down the number of times that the signal turns ON.	5-200
COUNTER_FB_M	\$1,\$2,\$3,\$1,\$	Counts the number of times that the signal turns ON from $\circledast\;$ to \circledast .	5-203

2.1.12 Standard timer function blocks

Function name	Argument	Processing details	Page
TP(_E) TP_HIGH(_E)	9, 2, 0, 0	Holds the signal ON for the specified time.	5-205
TON(_E) TON_HIGH(_E)	₫, ֎, ֎, ֎ , ֎	Turns ON the signal after the specified time.	5-208
TOF(_E) TOF_HIGH(_E)	5), &, 3), 9) 5), 1), 12, 13	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	\$), &, \$\$, \$), \$	Turns ON the signal after the specified time counted from input value \circledast to $@$.	5-214



FUNCTIONS

3.1	Input Pins Variable Function
3.2	Functions with EN
3.3	Labels
3.4	Data Types
3.5	Device and Address 3-8
3.6	Expressing Methods of Constants 3-10
3.7	Precautions on Programming 3-10

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 GF^GX Works2 Version1 Operating Manual (Structured Project)



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'.

1	Image: Constraint of the second se
2	Variable_1 2 ADD_E EN ENO 65 JN JN
3	Variable_1 AND 3 ADD_E Image: Non-state state st
4	· · · Variable 1 · · · · · ADD_E · · · · · MUL_E · · · · · ADD_E · · · · · · · · · · · DD_E · · · · · EN EN EN EN EN EN EN DD_E · · · · · · · · · · · · · · · D0 JN JN JN D12 JN D12 JN · · · · · · · · · D1 JN · · · · · · · · · · · · · · · · · · ·

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.

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.

		A	pplicable PO	U
Class	Description	Program block	Function	Function block
VAR_GLOBAL	Common label that can be used in program blocks and function blocks	0	×	0
VAR_GLOBAL_CONSTANT	Common constant that can be used in program blocks and function blocks	0	×	0
VAR	Label that can be used within the range of declared POUs This label cannot be used in other POUs.	0	0	0
VAR_CONSTANT	Constant that can be used within the range of declared POUs This constant cannot be used in other POUs.	0	0	0
VAR_RETAIN	Latch type label that can be used within the range of declared POUs This label cannot be used in other POUs.	0	×	0
VAR_INPUT	Label that receives a value This label cannot be changed in a POU.	×	0	0
VAR_OUTPUT	Label that outputs a value from a function block	×	×	0
Local label that receives a value and outputs the valu VAR_IN_OUT from a POU This label can be changed in a POU.		×	×	0

Table 3.3.3-1 Label classes

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		мо	%MX0.0		
	2	VAR_GLOBAL	g_int1	Word[Signed]		DO	%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		Class		Label Name	Data Type	Constant	Device	Address	Comment
1	VAR	•	g_bool1	Bit						
2	VAR	٠	g_int1	Word[Signed]						
3		٩								

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.

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

Table 3.4-1 Elementary data types

*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 CPU Structured Programming Manual (Common Instructions)

For data specification method, refer to the following manual.

CPU Structured Programming Manual (Common Instructions)

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

Fragmendix 1 Correspondence between Generic Data Types and Devices

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.

W35F X0 Device name Device number

CPU 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.

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

Table 3.5.2-1 Address definition specifications

Position

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

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)

Example)

%IX0 %MX1. 863 Position Data Classification Number size

3

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

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

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

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"

Table 3.6-1	Constant	expressing	method
-------------	----------	------------	--------

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	Application instructions such as DMOD, PCHK, INC(P)
Application instruction in GX Works2	CF 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, TRANCOJ, 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	$", \ \%, \ ', \ \sim, \ ^, \ !, \ @, \ [, \], \ \{, \ \}, \ ;, \ , \ , \ , \ ?, \ \backslash, \ !, \ \#, \ \$, \ ', \ _, \ \ast, \ /, \ +, \ <, \ >, \ =, \ \&, \ (, \), \ -$	
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	



OVERVIEW

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.
- Indicates a function to be explained.

3 Indicates the CPU modules that can use the function.

lc	on			
Universal model	High Performance model	Description		
QCPU	QCPU			
Universal	High Performance	The normal icon indicates that the CPU module can use the corresponding functions.		
Universal	High Performance	The icon with \triangle symbol indicates that the CPU module can use the corresponding functions under certain restrictions. (Function version and software version)		
Universal	High Performance	The icon with \times 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	Туре	Description	Direct device setting
Bit	Bool	Uses bit data or bit device. (Data range: 0 (FALSE), 1 (TRUE))	0
Word (signed)	Integer	Uses signed word data. (Data range: -32768 to 32767)	0
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)	0
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.
- $(\!0\!)$ Indicates program examples in the structured ladder and ST languages.

APPLICATION FUNCTIONS

5

5.1	Type Conversion Functions	5-2
5.2	Standard Functions of One Numeric Variable.	5-119
5.3	Standard Arithmetic Functions	5-122
5.4	Standard Bitwise Boolean Functions	5-143
5.5	Standard Selection Functions	5-148
5.6	Standard Comparison Functions	5-154
5.7	Standard Character String Functions	5-157
5.8	Functions of Time Data Type	5-172
5.9	Standard Bistable Function Blocks	5-184
5.10	Standard Edge Detection Function Blocks	5-190
5.11	Standard Counter Function Blocks	5-194
5.12	Standard Timer Function Blocks	5-205

OVERVIEW

FUNCTION TABLES

3

5.1 Type Conversion Functions

5.1.1 Bit type \rightarrow word (signed), double word (signed) type conversion

BOOL_TO_INT(_E), BOOL_TO_DINT(_E)



BOOL_TO_INT(_E) BOOL_TO_DINT(_E)

_E: With EN/ENO

Structure			functions.	s any of the following
Structured BOOL TO EN s		ST ENO:= BOOL_TO_INT_E (EN, s, d);	BOOL_TO_INT BOOL_TO_DINT	BOOL_TO_INT_E BOOL_TO_DINT_E
Input argument, Output argument,	EN: s: ENO: d:	Executing condition (TRUE: Execution, FALSE: Stop) Input Output status (TRUE: Normal execution, FALSE: Error or stop Output	:Bit :Bit) :Bit :Word (signed), dou	uble word (signed)

Grant Function

Operation processing

(1) BOOL_TO_INT, BOOL_TO_INT_E

Converts bit type data input to \odot into word (signed) type data, and outputs the operation result from \bigcirc .

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.

FALSE	
TRUE	
Bit type	Word (signed) type
(2) BOOL_TO_DINT, BOOL_TO_DINT_E

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

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.

FALSE		0
TRUE		1
Bit type	ر	Double word (signed) type

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from $\, \mathbb{d}$.

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

EN	ENO	b
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

5

Coperation 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 (s) into word (signed) type data, and outputs the operation result from (d).

(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]

1	.	•	:	•	•	•	•	•	•	•	•	•	•		•	•		•	•	•	•	•		
1		•	ده 	ool1	ı. ├─	•	هاه	ool	2 _	•	EN	1		00	L_T	ונס	J.T₽	3	EN	10	•	_	boc jnt1	
	.																•							

[ST]

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

- (2) The program which converts bit type data input to \odot into double word (signed) type data, and outputs the operation result from @.
 - (a) Function without EN/ENO (BOOL_TO_DINT)

[Structured ladder]

1	.																						
	- ·		•		1	•		1		÷	1			1			1	1	•				
	•	•	ta I	bool 	1·	•	•	•	•		001		800	IL_T	0.0	NN	Г			•	5	_din 1	H
											001	-											
	- ·		·			·		1	·			·	•		•	·			·			·	•

[ST]

g_dint1 := BOOL_TO_DINT(g_bool1);



☆ Function

Operation processing

Converts bit type data input to ${\scriptstyle{(s)}}$ into string type data, and outputs the operation result from

d .

When the input value is FALSE, 0 is output in string type data. When the input value is TRUE, 1 is output in string type data.

FALSE] '0'
TRUE] '1'
Bit type	String type

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

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

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

Operation Error

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

Program Example

The program which converts bit type data input to ${}_{\odot}$ into string type data, and outputs the operation result from ${}_{\odot}$.

(a) Function without EN/ENO (BOOL_TO_STR)

[Structured ladder]

1	:			 			•	•					•		•		
	+	•	t.a ∥—	•	•	•	·	_В	001	OL	TO	UST	R.	•	.	 stri	ng1

[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)



C Function

Operation processing

(1) BOOL_TO_WORD, BOOL_TO_WORD_E

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

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.

FALSE] Он
TRUE] 1н
Bit type	Word (unsigned)/16-bit string type

(2) BOOL_TO_DWORD, BOOL_TO_DWORD_E

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

outputs the operation result from $\ensuremath{\textcircled{}}$.

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.

FALSE	$ \longrightarrow \begin{tabular}{ c c c c } \hline \end{tabular} \end{tabular}$	0н	
TRUE	$ \longrightarrow \ \ \Box$	1н	
Bit type	Double word ((unsigned)/32-bit string t	ype

Operation result

(1) Function without EN/ENO

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

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

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

*1 When FALSE is output from ENO, the data output from (a) is undefined. In this case, create a program so that the data output from (a) 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 (s) into word (unsigned)/16-bit string type data, and outputs the operation result from (d).

(a) Function without EN/ENO (BOOL_TO_WORD)

[Structured ladder]

1		•	•		•	•		•	•		•	•	•	•		•	•	•	•	•		
	┝	•	رء 	000	1 · ⊫− ·	•	•	•	•	_B			TC	0_₩	ORI	D .			· ·	5_	wor	d1

[ST]

g_word1 := BOOL_TO_WORD (g_bool1);

(b) Function with EN/ENO (BOOL_TO_WORD_E)

[Structured ladder]

1																									
			·			·	÷		·	÷		·	·		·	·		·	·		·	·			·
	•	•	t.a	ool1	•	•	•	•	•	•	EN		В	00	L_T	o y	VOF	RD JE	•	EN		•	5.	boc	13
				. "			e'p	ool	2 _	_	EN JB(_							EN	U	_		wo	

[ST]

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

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



[Structured ladder]



[ST]

g_dword1 := BOOL_TO_DWORD (g_bool1);



Grant Function

Operation processing

Converts bit type data input to ${\scriptstyle (s)}$ into time type data, and outputs the operation result from

(d) .

When the input value is FALSE, 0 is output in time type data. When the input value is TRUE, 1 is output in time type data.



Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ${\,\rm (d)}$.

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

EN	ENO	ð
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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

Program Example

The program which converts bit type data input to ${}_{\textcircled{S}}$ into time type data, and outputs the operation result from ${}_{\textcircled{G}}$.

(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]

1	•	•	•	•	•	•	•	•	•	:	•	•	•	•	•	•	•	•	•	•	•	:		•	:
	•	•	5	; bool	11 ·	•	•	•	•	•	EN		E	300	DL.	то	тім	E,E		EN	In	•	5.	boc	ol3
							e ja	ool	2 _			DOL	L										_s.	tim	e1

[ST]

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

5.1.5 Word (signed) type \rightarrow double word (signed) type conversion INT_TO_DINT(_E)



Grant Function

Operation processing

Converts word (signed) type data input to ${}_{\textcircled{S}}$ into double word (signed) type data, and outputs the operation result from ${}_{\textcircled{G}}$.



INT_TO_DINT(_E)

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	٥
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 (a) is undefined. In this case, create a program so that the data output from (a) 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 s into double word (signed) type data, and outputs the operation result from d.

(a) Function without EN/ENO (INT_TO_DINT)

[Structured ladder]

1	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
	•	5_	int1	= 5	592	3 _	•	٩L	11	IN	1T_1	roj	DIN.	Т			•		din f	1 =	59	23
			·			·			·			·			•			·			·	

[ST]

g_dint1 := INT_TO_DINT (g_int1);

(b) Function with EN/ENO (INT_TO_DINT_E)

[Structured ladder]

		·	·	·	·	·	·	·	·		·	·		·	·	·	•	·	·	·	·	•	·	•
	•	•	٤J	bool I	1.	•	•	•	•	•	EN		I	NT.	TO.	DIN	IT E		EN		•		boo	13
Γ							. 5	jnt	1_	_	EN JN								EN			_	din t	
															·						·			

[ST]

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

5.1.6 Double word (signed) type \rightarrow word (signed) type conversion DINT_TO_INT(_E)

Universal

				UD	
DINT_TO_IN	ντ(_E)		_E: With EN/ENO		
Structure DiNT_TC EN s	ed ladde D_INT_E ENO d	r ST ENO:= DINT_TO_INT_E (EN, s, d);	functions. DINT_TO_INT	any of the fol	-
Input argument, Output argument,	EN: s: ENO: d:	Executing condition (TRUE: Execution, FALSE: Stop) Input Output status (TRUE: Normal, FALSE: Error) Output	:Bit :Double word (signed :Bit :Word (signed))	

Geration processing

Converts double word (signed) type data input to ${}_{\textcircled{S}}$ into word (signed) type data, and outputs the operation result from ${}_{\textcircled{G}}$.



Double word (signed) type

Word (signed) type

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ${\,\rm (d)}$.

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

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

When the DINT_TO_WORD(_E) function is executed, high-order 16-bit data

of double word (signed) type data input to $\, \circledast \,$ 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 \odot into word (signed) type data, and outputs the operation result from @.

(a) Function without EN/ENO (DINT_TO_INT) [Structured ladder]

1	•		•	•	•	•		•	•		•	•	•	•	:	•			•	•	
	•	€_di	nt1	= 5	92:		• •	JDI	NT	DI	INT.	_TO	ЛИ,	г	•	•	5	int1	= !	592	3

[ST]

g_int1 := DINT_TO_INT(g_dint1);

(b) Function with EN/ENO (DINT_TO_INT_E)

[Structured ladder]

1		•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	•		•	•	•
	$\left \right $	•	•	вJ —	boo 	∥ ⊩	•	5_	dint	1 _	•	EN JDI	I INT	[DIN.	т_то	ИLC	IT_E		EN	0	•	_	boc jnt1	

[ST]

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

5.1.7 Word (signed), double word (signed) type \rightarrow bit type conversion

INT_TO_BOOL(_E), DINT_TO_BOOL(_E)



Operation processing

(1) INT_TO_BOOL, INT_TO_BOOL_E

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

When the input value is 0, FALSE is output in bit type data.

When the input value is other than 0, TRUE is output in bit type data.

0	FALSE
1567	TRUE

Word (signed) type

(2) DINT_TO_BOOL, DINT_TO_BOOL_E

Converts double word (signed) type data input to $\, \mathrm{s} \,$ into bit type data, and outputs the

operation result from (d).

When the input value is 0, FALSE is output in bit type data. When the input value is other than 0, TRUE is output in bit type data.



Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from $\operatorname{\underline{o}}$.

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

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

Operation Error

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

Program Example

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

(a) Function without EN/ENO (INT_TO_BOOL)

[Structured ladder]

1	.																			
		5.	int1	= 5	592:	3		JN	T	IN	T_T	0,в	:00	L	•			5.	boo	11
	·	·		·	·	·	·	·	÷	·	·	·	·	·	÷	·	·	·	·	·

[ST]

g_bool1 := INT_TO_BOOL(g_int1);

(b) Function with EN/ENO (INT_TO_BOOL_E)

[Structured ladder]

			5	صر	· ol1 ·				÷	•	•	•		1T_T	D_E		DLJE			•	•		boo	
t				1.	5		. E.	jn t1	_	_	EN JN	т							EN	0			boo	
	·	·					•	·		·	÷		·	÷	·	·		·	·		÷	·		•



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

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



1																					
	· ·	·			·	÷		·			·			·	·			·	·		·
	•	·		•	·	•	•			D	INT	тото	,в0	οοι	-			·			
	•	·	6_C	lin t1	=	0_		JD	NT									-	5.	рос	ol1
	•	·	·	·	·	·	•	·	·	·	·	·	·	·	·	·	·	·	·	·	·

[ST]

g_bool1 := DINT_TO_BOOL(g_dint1);

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

INT_TO_REAL(_E), DINT_TO_REAL(_E)



Gran Function

Operation processing

(1) INT_TO_REAL, INT_TO_REAL_E

Converts word (signed) type data input to ${}_{\textcircled{S}}$ into single-precision real type data, and outputs the operation result from ${}_{\textcircled{G}}$.



- (2) DINT_TO_REAL, DINT_TO_REAL_E
 - (a) Converts double word (signed) type data input to ${}_{\mbox{\scriptsize $\$$}}$ into single-precision real type data, and outputs the operation result from ${}_{\mbox{\scriptsize d}}$.



(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 a .

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	Ø
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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 s into single-precision real type data, and outputs the operation result from d.
 - (a) Function without EN/ENO (INT_TO_REAL)

[Structured ladder]

1	•		•			•	•	•	•	•		•	•		•	•	•	•	•		•		
	•	5.	jnt1	= !	592	3	•	aj	nt	IN	IT_T	оŗ	REA	L			•		rea	1 =	59	23.0).
	·	·	·	·	·	÷	·	•		•	•		•	•	•	•	•	÷	·	·	·	·	·

[ST]

g_real1 := INT_TO_REAL(g_int1);

(b) Function with EN/ENO (INT_TO_REAL_E)

[Structured ladder]

.	•	•	•		•	:		•	•		•	•		:	•		:		•	•		•	:
	•	ta I	oool	1·	•	•	•	•	•	EN		II	NT.	тој	REA	NL JE		EN		•	ج	boc	513
						. 5	jnt	1_	_	aJ								EN			<u></u> .	rea	12
·										•													



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

- (2) The program which converts double word (signed) type data input to s into single-precision real type data, and outputs the operation result from d.
 - (a) Function without EN/ENO (DINT_TO_REAL)

[Structured ladder]

1																				
	 _din	tl	= 6!	500	•	•	aj	Din t	ЛИ	г_то	J.RI	EAL	•	•	•	Ŀ	 jrea	.11 =	= 65).0

- [ST]
- g_real1 := DINT_TO_REAL(g_dint1);

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

INT_TO_LREAL(_E), DINT_TO_LREAL(_E)



Grant Function

Operation processing

(1) INT_TO_LREAL, INT_TO_LREAL_E

Converts word (signed) type data input to ${}_{\textcircled{S}}$ into double-precision real type data, and outputs the operation result from ${}_{\textcircled{G}}$.



(2) DINT_TO_LREAL, DINT_TO_LREAL_E

Converts double word (signed) type data input to \odot 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	٥
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 (a) is undefined. In this case, create a program so that the data output from (d) is not used.

INT_TO_LREAL(_E), DINT_TO_LREAL(_E)

✓ 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 \odot into double-precision real type data, and outputs the operation result from \bigcirc .

(a) Function without EN/ENO (INT_TO_LREAL) [Structured ladder]

 1																																
	·	·						·]	INT	T T	οт	LRE	EAI	_		•														
	·	g	Lin	t1	=	123	3 –		a,	In	t	- 1	-						-g	In	eal	1 =	= 1	.20	300	000)e+	+0()2	·	·	·
	·	•	•	•	•	·		·		·		·	•	• •		•	·	·		·	·	·	·	·	·	·	·	·	·	·	·	·

[ST]

g_lreal1 := INT_TO_LREAL(g_int1);

(b) Function with EN/ENO (INT_TO_LREAL_E)[Structured ladder]

1	
	EN EN EN EN gbool3 · · · ·
	····gint1 — a_INTg_Ireal2 ····

[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 \odot into double-precision real type data, and outputs the operation result from .
 - (a) Function without EN/ENO (DINT_TO_LREAL)

[Structured ladder]

1	
	g_lreal1 = 65000a_Dintg_lreal1 = 6.500000e+004
ļ	

[ST] g_lreal1 := DINT_TO_LREAL(g_dint1);

5.1.10 Word (signed), double word (signed) type \rightarrow string type conversion

INT_TO_STR(_E), DINT_TO_STR(_E)



Grant Function

Operation processing

- (1) INT_TO_STRING, INT_TO_STRING_E
 - (a) Converts word (signed) type data input to \odot into string type data, and outputs the operation result from \bigcirc .



Automatically stored at the end of the character string.

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

5-29

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

	High-order byte	Low-order byte	_
	20н (space)	2Dн (-)	String 1st word
-123	31н (1)	20н (space)	2nd word
	33н (3)	32н (2)	3rd word
Word (signed) type	0	0н	4th word

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



Automatically stored at the end of the character string.

- (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 ${\,\rm (d)}$.

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

EN	ENO	(b)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

5

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 (s) into string type data, and outputs the operation result from (d).
 - (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]

	•		•			•	•		•	:	:	Ì	:		:	:	:	:		•	•	•	•	
-	•	•	Б) —]	bool 	⊢		. 5	jnt	1 _	•	AB JN		INT.	,то,	_STI	R,E		ΕŅ	10	•		boc stri		

[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 (s) into string type data, and outputs the operation result from (d).
 - (a) Function without EN/ENO (DINT_TO_STR)

[Structured ladder]

1	. .	•	:	:	•	•	•	•	:	•	•	:	:	•	:	•	•	:	•	•	•	:	•	•	•	•	:
	5.0	din t1	= ·	-12	345	67:	з	•	JD	INT	DI	NT.	.то	STI	R			·	ह	stri	n 151	= '	- 1	234	4567	78'	
	·		÷					·	•	•	•		•	•	÷	•	·			·			·		·		·

[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)



Grant Function

Operation processing

(1) INT_TO_WORD, INT_TO_WORD_E

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



INT_TO_WORD(_E), DINT_TO_WORD(_E)

(2) DINT_TO_WORD, DINT_TO_WORD_E

Converts double word (signed) type data input to \odot into word (unsigned)/16-bit string type data, and outputs the operation result from 0.



High-order 16-bit data is discarded.

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from $\operatorname{\underline{o}}$.

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

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

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

When the DINT_TO_WORD(_E) function is executed, high-order 16-bit data

of double word (signed) type data input to input variable $\,{}_{\textcircled{}}\,$ is discarded.

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

(a) Function without EN/ENO (INT_TO_WORD)

[Structured ladder]

1	·	•	•	:	•	•	•	•	•	•	•	•	•	•	•	•	•	÷	•	•	•	•	•	•
	.	5.	jnt1	= (592:	3	•	JN	IT	IN	T_TI	0_W	/OR	D			•	5.	wor	d1	= 1 (5#1	723	
	·	•		·	·	·	·	•		•	•		•	•		•	•	·		·	·	•	·	·

[ST]

g_word1 := INT_TO_WORD(g_int1);

(b) Function with EN/ENO (INT_TO_WORD_E)

[Structured ladder]

	·		·		·	·	·	·			·	·		·	·		·		·	·	·	·	·
	•	•	¢,	000	11 ·	•	•	•	•	•	EN	IN	IT.T	ю.у	MOR	RDJ	=	EN		•		boo	13
							. 5	jnt	1_	_	JN JN							EN	10		<u>5</u> .	wor	d1

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

(a) Function without EN/ENO (DINT_TO_WORD)

[Structured ladder]

1		
	.ɛ_dint1 = 12345678DINT_TO_WORD	g_word1 = 16#614E

[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)



Grant Function

Operation processing

(1) INT_TO_DWORD, INT_TO_DWORD_E

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

	-325	0000FEBBн
	Word (signed) type	Double word (unsigned)/32-bit string type
-325		1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1
		Data conversion
0000FEBBн	00000000000000000	0 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 1
	Always filled with 0s.	

(2) DINT_TO_DWORD, DINT_TO_DWORD_E

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



Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ${\scriptstyle (\ensuremath{\underline{o}})}$.

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

EN	ENO	Ø
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

✓ Operation Error

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

Program Example

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

(a) Function without EN/ENO (INT_TO_DWORD)

[Structured ladder]

1	:	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
	.	•	gjr	it1	= 1	•	•	٩ل -	1T		чт_:	roj	DW0	DRE	>			•	6	dwo	ord1	= 1	6#	000	1000	00A	

[ST]

g_dword1:= INT_TO_DWORD(g_int1);

(b) Function with EN/ENO (INT_TO_DWORD_E)

[Structured ladder]

1																									
		•	·	•		·			·	•	•	•	•		•	•		•	•		•	·		•	·
	•	·	۶J	bool I	1 · .	·	•	•	÷	•			1	NT.	TO.	D₩	ORI	D,E			_	·	Б	boa	13
							. 5	jnti	_	_	JN									EN					ord1
		·	·	·	·	·		·	·	·	·	·	·	·	·	·		·	·	·	·	·		·	•

[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 \odot into double word (unsigned)/32-bit string type data, and outputs the operation result from \bigcirc .
 - (a) Function without EN/ENO (DINT_TO_DWORD)

[Structured ladder]

1	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
	. Б	_din	t1 :	= 7	456	5	· ·	JDI	INT	DI	NT.	.то,	DW	'O RI	D .	•		•	5	dwo	ord1	= 1	6#	000)1 2 3	345	•

[ST]

g_dword1:= DINT_TO_DWORD(g_dint1);
5.1.13 Word (signed), double word (signed) type \rightarrow BCD type conversion

INT_TO_BCD(_E), DINT_TO_BCD(_E)



Section 7

Operation processing

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



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

5.1 Type Conversion Functions 5.1.13 Word (signed), double word (signed) type \rightarrow BCD type conversion

5

INT_TO_BCD(_E), DINT_TO_BCD(_E)

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



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

Operation result

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

Operation result	Ø
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	Ø
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 (d) is undefined. In this case, create a program so that the data output from (d) is not used.

✓ Operation Error

An operation error occurs when the value input exceeds 9999 or 999999999 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 s into BCD type data, and outputs the operation result from (d).
 - (a) Function without EN/ENO (INT_TO_BCD)

[Structured ladder]

1	.	•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	
	. . .	5.	int1	= {	592:	3	·	JN	IT	INT	_TO	.BC	D.			•	5.	woi	rd1	= 1	5#5	923	•

[ST]

g_word1:= INT_TO_BCD(g_int1);

(b) Function with EN/ENO (INT_TO_BCD_E)

[Structured ladder]

	·	·	÷		·	÷	·	÷	÷	·		÷	·	÷		·		÷	·	÷.,	·	·	÷	·
	•	•	5. 	bool I	1.	•	•	•	•	•	-		I	INT,	TO.	,BC	D,E		-		•	д	boo	13
Γ							. 5	jnt	_		UN UN								EN	10		_	wor	
											•										•			

[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 (s) into BCD type data, and outputs the operation result from (d).
 - (a) Function without EN/ENO (DINT_TO_BCD)

[Structured ladder]

1																									
	•	·	•	•	•	·		•	•		•	•	•	·	•	•		•	•	·	·	•	•	•	·
		s_din	tt :	= 20	000	o	•	م	INT	DI	INT.	.то,	BC	D			·	g.	dwo	ord1	= 1	6#	000	200	
	•	·	·	·	·	·	•	·	·	·	•	·	·	·	·	·	·	·	·	·	·	·	·	·	·

[ST]

s_dword1:= DINT_TO_BCD(g_dint1);

5.1.14 Word (signed), double word (signed) type \rightarrow time type conversion

INT TO TIME(E), DINT TO TIME(E)



Grant Function

Operation processing

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

FFFFH	1m5s535ms
·	·
Word (signed) type	Time type

Word (signed) type

(1) Function without EN/ENO

An operation is executed and the operation value is output from $\ensuremath{\mathbbm d}$.

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

EN	ENO	(b)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

5

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

(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]

																		÷					
		÷	·	÷		÷		·			·			·			·		·	·	÷		·
		•	¢J	600	11 ·	•	•	•	•	EN		I	NT_	TO.	TIN	IE_E		EN	0	•	5.	boc	013
						. 5	jnt	1_		JN								EN			_s.	time	a1
		÷		·							·					•	÷		•				

[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 (s) into time type data, and outputs the operation result from (d).

(a) Function without EN/ENO (DINT_TO_TIME)

[Structured ladder]

		·	·			·	·		·	·		·	·	·	·	·	• .	·		
			÷			·	·			D	INT	г_тс)_TI	ME					÷	
		·	÷	£"q	lin t1	_	_	JUI	NT									 _5.1	time	1
							•				•									

[ST]

g_time1:= DINT_TO_TIME(g_dint1);

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

REAL_TO_INT(_E), REAL_TO_DINT(_E)



Operation processing

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



- (b) The value to be input to \odot 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 (s) into double word (signed) type data, and outputs the operation result from (d).



- (b) The value to be input to \odot 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.

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

Operation result	٥
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	Ø
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 (d) is undefined. In this case, create a program so that the data output from (d) is not used.

✓ Operation Error

An operation error occurs in the following cases.

• 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 \odot into word (signed) type data, and outputs the operation result from \bigcirc .
 - (a) Function without EN/ENO (REAL_TO_INT)

[Structured ladder]

	·	·		•	·	•		·	·		·	•	·	·	•	·	·	•	•	·	·	
	. Б.	Jes	11 =	= 59	23.	5 _	•	a 1	real	R	EAL	тс	ЛЛ	т			ŀ		jnt1	=	592	4
										•	•	•	•	•	•	•	· .					

[ST]

g_int1:= REAL_TO_INT(g_real1);

(b) Function with EN/ENO (REAL_TO_INT_E)

[Structured ladder]

		•	·	·		·			·			·		·			·	·		•	·		•
		ta I	ool	11	•	•	•	•	•	EN		F	REA	L_TI	٩LO	IT E	•	EN		•	5	boc	13
						5.	real	1_		_	eal							EN	U			jnt1	

[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 s into double word (signed) type data, and outputs the operation result from d.
 - (a) Function without EN/ENO (REAL_TO_DINT)

[Structured ladder]

1	.																						
	gjre	 al1 =	650	100.!	5		aj	eal	F	:EAI	L_TI	ס_ס	INT	•			•	5.	din f	t1 =	65	001	•
	.				•	·		•	•	•	•	•	•	•	•	•			·			·	·

[ST]				
	g_dint1:= REAL_	TO	_DINT(g_	_real1);

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

LREAL_TO_INT(_E), LREAL_TO_DINT(_E)

Universal

					UD
LREAL_TO_ LREAL_TO_				_E: With EN/ENO	
Structured LREAL TO EN s		ST ENO:= <u>[</u>	I	functions. LREAL_TO_INT L	v of the following .REAL_TO_INT_E .REAL_TO_DINT_E
Input argument, Output argument,	EN: s: ENO: d:	Executing condition (TRUE: Execution, FALSE: Stop) Input Output status (TRUE: Normal, FALSE: Error) Output	_	:Bit :Double-precision real :Bit :Word (signed), double w	vord (signed)

Grant Function

Operation processing

- (1) LREAL_TO_INT, LREAL_TO_INT_E
 - (a) Converts double-precision real type data input to ${}_{\circledast}$ into word (signed) type data, and

outputs the operation result from \bigcirc .



- (b) The value to be input to \odot 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 s into double word (signed) type data, and outputs the operation result from d.



- (b) The value to be input to \odot 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.

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

Operation result	Ø
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	(b)
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 (d) is undefined. In this case, create a program so that the data output from (d) is not used.

Operation Error

An operation error occurs in the following cases.

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

(Error code: 4140)

Program Example

- (1) The program which converts double-precision real type data input to \odot into word (signed) type data, and outputs the operation result from .
 - (a) Function without EN/ENO (LREAL_TO_INT)

[Structured ladder]

1	· · · · · · · · · · · · · · · · · · ·
	LREAL_TO_INT
	g_lreal1=5.923500e=003

[ST]

- g_int1:= LREAL_TO_INT(g_lreal1);
- (b) Function with EN/ENO (LREAL_TO_INT_E)

[Structured ladder]

1	.			g_k	000	þ							L	_RE	EAL	T	ΟJ	NT	Ē					·				·		
				-	.	ŀ	g	Ire	aľ	1 –		El a	N ĮR∉	eal					E	NC.			-g_	bo: int		3				
	·	•	·	·	•	·	•	·	·		•	•	•	•	• •		•		•	•	•	•	·	•	•	·	•	·	•	·

[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 \odot 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 \rightarrow double-precision real type conversion

REAL TO LREAL(E)



Grant Function

Operation processing

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



REAL_TO_LREAL(_E

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

Operation result	(b)
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	ð					
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 (a) is undefined. In this case, create a program so that the data output from (a) 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, $2^{-126} \le | \otimes | < 2^{128}$
- The operation result is outside the following range (an overflow occurrence).

(Error code: 4141)

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

Program Example

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

(a) Function without EN/ENO (REAL_TO_LREAL)

[Structured ladder]

1	
	REAL_TO_LREAL
	g_real1 = 5923.5

[ST]

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

(b) Function with EN/ENO (REAL_TO_LREAL_E)

[Structured ladder]

1	
	<u></u>
	g bool1 REAL_TO_LREAL_E
•	
	· · · g_real1 ── <mark>a_Real </mark>

[ST]

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

REAL_TO_LREAL(_E

5.1.18 Double-precision real type \rightarrow single-precision real type conversion





Grant Function

Operation processing

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



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

Operation result	Ø
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	(d)
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 (d) is undefined. In this case, create a program so that the data output from (d) is not used.

5

Operation Error

An operation error occurs in the following cases.

- The input value is -0 or outside the following range. (Error code: 4140)
 - $0, 2^{-1022} \leq | \leq | < 2^{1024}$
- The operation result is outside the following range (an overflow occurrence).

(Error code: 4141)

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

Program Example

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

(a) Function without EN/ENO (LREAL_TO_REAL)

[Structured ladder]

1		
	I BEAL TO BEAL	
	g_lreal1 = 1.234000e+000 — a_lRealg_real1 = 1.234	

[ST]

- g_real1:= LREAL_TO_REAL(g_lreal1);
- (b) Function with EN/ENO (LREAL_TO_REAL_E)

[Structured ladder]

1	
	LREAL_TO_REAL_E
	EN EN EN EN glreal
	· · · · · glreal1 — a_IRealg_real1 · · · · ·
	1

[ST]

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

5.1.19 Single-precision real type \rightarrow string type conversion REAL_TO_STR(_E) Universa UD REAL_TO_STR(_E) _E: With EN/ENO indicates any of the following functions. Structured ladder ST REAL_TO_STR REAL_TO_STR_E REAL_TO_STR_E ENO:= REAL_TO_STR_E (EN, s, d); ΕN ENO d Input argument, EN: Executing condition (TRUE: Execution, FALSE: Stop) :Bit :Single-precision real s: Input Output argument, ENO: Output status (TRUE: Normal, FALSE: Error) :Bit d: Output :String(12)

Grant Function

Operation processing

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

		High-order byte	Low-order byte	
Single-precision		ASCII code of integral part	Sign data (integral part)	String 1st word
	7	ASCII code of one decimal place	ASCII code of (.) decimal place (2Eн)	2nd word
א ארארארארארארו		ASCII code of three decimal place	ASCII code of two decimal place	3rd word
i_i_i_i_i_i_i. .▲ Sign	▲ ▲ Sign	ASCII code of five decimal place	ASCII code of four decimal place	4th word
(integral part)	(exponent part)	Sign data (exponential part)	45н(E)	5th word
	added.	ASCII code of exponent part's units place	ASCII code of exponent part's tens place	6th word
			00н Ф	7th word

Automatically stored at the end of the character string.

- (2) The character string data after conversion is output from output variable (a) 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).

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

Operation result	(b)
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	(d)
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 (d) is undefined. In this case, create a program so that the data output from (d) is not used.

Coperation 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 \odot into string type (exponent form) data, and outputs the operation result from @.

(a) Function without EN/ENO (REAL_TO_STR)

[Structured ladder]

		 eal1	= -1	23	456	7			RE	AL.	.TO_\$	STR	•	•	•	<tri< th=""><th>· ·</th><th>- 1.3</th><th>57F</th><th>+01</th><th></th></tri<>	· ·	- 1.3	57F	+01	
	00					·	 RI	EAL								 			 		

[ST]

g_string1:= REAL_TO_STR(g_real1);

(b) Function with EN/ENO (REAL_TO_STR_E) [Structured ladder]

	.																							
	·	÷	÷	·	÷	÷				·			·	·	÷	·	·		·	·	÷	·	·	
		•	t.a	ool	11	•		•	•	•	EN		R	EAL	тс)_S1	IR J	•	EN		•	ج	boc	513
							E.J	real	1_	_		i EAL							EN					in s1
	.	·	·	·	·	·	·	·	·	·	·	·	·	·	÷	·	·	÷	·	·	·	·	·	•

[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 \rightarrow bit type conversion WORD_TO_BOOL(_E), DWORD_TO_BOOL(_E)

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

Operation processing

(1) WORD_TO_BOOL, WORD_TO_BOOL_E

Converts word (unsigned)/16-bit string type data input to $\, \mathrm{s} \,$ into bit type data, and outputs

the operation result from \bigcirc .

When the input value is 0H, FALSE is output.

When the input value is other than 0H, TRUE is output.



WORD_TO_BOOL(_E), DWORD_TO_BOOL(_E) (2) DWORD_TO_BOOL, DWORD_TO_BOOL_E

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

outputs the operation result from (d).

When the input value is 0H, FALSE is output.

When the input value is other than 0H, TRUE is output.



Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from \bigcirc .

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	٥
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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 s into bit type data, and outputs the operation result from d.

(a) Function without EN/ENO (WORD_TO_BOOL)

[Structured ladder]

1																						
		•	÷		÷	·	·	÷	÷	÷	÷	·	÷	÷	·	÷	•		÷	·	÷	•
	5.9	NOI	rd1	= 1	6#(000	1 _	·	.w	ORI		ORI	D_T(),В	00	L		•		, boc	ol1	
		÷	÷				·	÷	÷			÷			÷		·	·		·	÷	

[ST]

g_bool1:= WORD_TO_BOOL(g_word1);

(b) Function with EN/ENO (WORD_TO_BOOL_E)

[Structured ladder]

1	. .	•	:		•	•		•	:	•	•	:	•	•	:	•		•	•	•	:	•	•	•	•
	ŀ	•	ta 	ooli	1 · 			•	•	•	EN	1	۷	VOF	RD_T	0,	300)LJ	ŧ	EN	10	ŀ		boc	13
	ŀ	÷	1	•		·	6.W	ord	-		.w	ΰR	D									_	_5.	boc	ol2

[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 (s) into bit type data, and outputs the operation result from (d).

(a) Function without EN/ENO (DWORD_TO_BOOL)

[Structured ladder]

1																							
	· ·	·	·		·	•		·			·	·		·		·	·		·	·		·	•
	E_dv	vord	11 =	16	#OC	0000	200	1 _	•	م	woi		VOR	נסמ	ro je	300	DL			•	5.	boc	ol1
	·	·	•					•	•		•	·	·	·	·	·	·	·	·	·	·	·	•

[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)

UD WORD TO INT(E) E: With EN/ENO WORD TO DINT(E) indicates any of the following functions. Structured ladder ST WORD_TO_INT WORD_TO_INT_E WORD_TO_INT_E WORD_TO_DINT WORD_TO_DINT_E ENO:= WORD_TO_INT_E (EN, s, d); ENO ΕN d s Input argument, EN: Executing condition (TRUE: Execution, FALSE: Stop) :Bit s: Input :Word (unsigned)/16-bit string Output status (TRUE: Normal, FALSE: Error) Output argument, ENO: :Bit d: Output :Word (signed), double word (signed)

Grant Function

Operation processing

(1) WORD_TO_INT, WORD_TO_INT_E

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



Operation result

(1) Function without EN/ENO

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

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

EN	ENO	đ
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 (a) is undefined. In this case, create a program so that the data output from (a) 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 (s) into word (signed) type data, and outputs the operation result from (d).

(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]

.												•									·	
· ·	•		• •		•	•		•	<u> </u>	•	·		•	·	•	•	•		÷	•		•
·	•	٤J	ool1 ·	•	•	•	•	•	-		W	/OR	D_T	IL O	AT 1	=	-		•	ج	boc	13
					6.W	ord	1 _		EN LW	ı ORI	D						EN	U		_	in t1	
·			• •		•		÷	·			·	÷	·	·	·	·	·		·	·		·

[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 s into double word (signed) type data, and outputs the operation result from d.
 - (a) Function without EN/ENO (WORD_TO_DINT)

[Structured ladder]

1	•	•			•			•	•	•		•	•		•	•	•	•	•			•	
	. 5	_wo	rd1	= 1	6#1	1 2 3	4	•	 ORI		OR	:D_T		INK	г			•		din f	t1 =	46	60

[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 Universal Performance
DWORD_TC DWORD_TC			_E: With EN/ENO
Structured DWORD_T EN s		ST ENO:= DWORD_TO_INT_E (EN, s, d);	indicates any of the following functions. DWORD_TO_INT DWORD_TO_INT_E DWORD_TO_DINT DWORD_TO_DINT_E
Input argument, Output argument,	EN: s: ENO: d:	Executing condition (TRUE: Execution, FALSE: Stop) Input Output status (TRUE: Normal, FALSE: Error) Output	:Bit :Double word (unsigned)/32-bit string :Bit :Word (signed), double word (signed)

Grant Function

Operation processing

(1) DWORD_TO_INT, DWORD_TO_INT_E

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



High-order 16-bit data is discarded.

DWORD_TO_INT(_E), DWORD_TO_DINT(_E)

(2) DWORD_TO_DINT, DWORD_TO_DINT_E

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

(signed) type data, and outputs the operation result from ${\scriptstyle (\ensuremath{\textbf{d}})}$.



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	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

⊠POINT -

When the DINT_TO_WORD(_E) function is executed, high-order 16-bit data of double word (unsigned)/32-bit string type data input to (s) 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 (s) into word (signed) type data, and outputs the operation result from (d).

(a) Function without EN/ENO (DWORD_TO_INT)

[Structured ladder]

1																										
											·										•					
	÷	g_dv	vorc	11 =	16	#00	001 :	234	5 _	•	ر	WOI		WO	RD,	TO,	JΝ.	r			•	 jn t1	•	902	9	:
			·		·	·			·		•	÷	•	•	·	•	·	•	•	•	•	·				•

[ST]

g_int1:= DWORD_TO_INT(g_dword1);

(b) Function with EN/ENO (DWORD_TO_INT_E)

[Structured ladder]

1	·	•	•	· ·	•	•	•	:	:	•	•	•	•	•	•	•	•	•	•	•	•	•	:	•
	ŀ		€.bo —		•			•	•	EN		[DWO	ORE	тс	ИU	T_E		EN	10	ŀ		boc	13
	.	•				s_dw	ordi	' <u>—</u>		ים, י	NO F	RD .									-	_5.	jnt1	· .

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

(a) Function without EN/ENO (DWORD_TO_DINT)

[Structured ladder]

1																												
	•	·	·		·	÷		·			·		÷	·	÷		·	÷		•	·		•	·		·	÷	•
		. e	s_dv	orc	1 =	16	#00	101 2	234	5 _	•	لر	woi		MOR	RD_	τо.	DIN	Т			•	5	din f	1 =	74	565	
	•	•	•	•	·	·	·	·	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	·	•	•	•	•

[ST] g_dint1:= DWORD_TO_DINT(g_dword1);

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

WORD TO DWORD(E) WORD_TO_DWORD(_E) E: With EN/ENO indicates any of the following functions. Structured ladder ST WORD_TO_DWORD WORD_TO_DWORD_E WORD_TO_DWORD_E ENO:= WORD_TO_DWORD_E (EN, s, d); ENO ΕN d S Input argument, EN: Executing condition (TRUE: Execution, FALSE: Stop) :Bit s: Input :Word (unsigned)/16-bit string Output status (TRUE: Normal, FALSE: Error) Output argument, ENO: :Bit d: Output :Double word (unsigned)/32-bit string

Grant Function

Operation processing

Converts word (unsigned)/16-bit string type data input to (s) into double word (unsigned)/32-

bit string type data, and outputs the operation result from ${}_{\textcircled{}}$. After data conversion, high-order 16 bits are filled with 0s.

	5678н		00005678н
--	-------	--	-----------

Word (unsigned)/16-bit string type

Double word (unsigned)/32-bit string type

(1) Function without EN/ENO

An operation is executed and the operation value is output from ${\,\rm (d)}$.

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

EN	ENO	Ø
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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

Program Example

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

(a) Function without EN/ENO (WORD_TO_DWORD)

[Structured ladder]

1	.																														
	·	·			·			÷							·	·			·		·	·		·			·		÷		
	·											wo	RD	то	DY	VOF															
	.	. 5 .	woi	rd1	= 1	6#1	234	4_		.w	ΰRI										_s.	dwo	ord1	= 1	6#	000	013	234			
	·	•	·	•	·	·		·	·	•	•	•		•	•	•	•	•	•	•	·	·	÷	·	·	·	•	·	·	·	•

[ST]

g_dword1:= WORD_TO_DWORD(g_word1);

(b) Function with EN/ENO (WORD_TO_DWORD_E)

[Structured ladder]

1			•		•	•			•		•	•		•	•	•	•	•		•	•			•		•	•
		•	•	۶.	bool	1 ·	•	•	•	•	·	EN			WOI	RD_	TO.	D₩	ΌR	D,E		EN		ŀ		boc	513
	Ī							E-W	ord	1_		_	' ORI	D								EN	10		5 .	dwo	ord1
	- 1																										

[ST]

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

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

DWORD TO WORD(E) DWORD_TO_WORD(_E) E: With EN/ENO indicates any of the following functions. **Structured ladder** ST DWORD_TO_WORD DWORD_TO_WORD_E DWORD_TO_WORD_E ENO:= DWORD_TO_WORD_E (EN, s, d); ENO ΕN d s Input argument, EN: Executing condition (TRUE: Execution, FALSE: Stop) :Bit s: Input :Double word (unsigned)/32-bit string Output status (TRUE: Normal, FALSE: Error) Output argument, ENO: ·Bit d: Output :Word (unsigned)/16-bit string

Grant Function

Operation processing

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



High-order 16-bit data is discarded.

DWORD_TO_WORD(_E)

(1) Function without EN/ENO

An operation is executed and the operation value is output from \bigcirc .

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	٥					
TRUE (Operation execution)	TRUE	Operation output value					
FALSE (Operation stop)	FALSE ^{*1}	Undefined value					

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

POINT -

When the DWORD_TO_WORD(_E) function is executed, high-order 16-bit data

of double word (unsigned)/32-bit string type data input to $\,\,\mathrm{(s)}\,$ 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 s into word (unsigned)/16-bit string type data, and outputs the operation result from d.

(a) Function without EN/ENO (DWORD_TO_WORD)

[Structured ladder]

1	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•			•	•	•	•	•	•	•		•	•	•	•	
			g_dv	vorc	11 =	16	#12	345	567	8	•	_ل ر	WOI	DW RD	OR	D_T	ю_V	YO F	RD			· -		woi	rd1	- = 1	6#5	678		•

[ST]

g_word1:= DWORD_TO_WORD(g_dword1);

(b) Function with EN/ENO (DWORD_TO_WORD_E)

[Structured ladder]

1		•	•			•			•		•	•	•	•	•	•	•	•	•	•	•	•	•		•	•	•	
	L	•	•	ta 	ool	1 · 	•	•	•	•	•	EN		C	DW/C	ORD	та	UW.	ORI	D,E		EN		•		boc	13	:
				-			. 5	_dw	ord1	_			MOR	۶D									-		_5.	wor	d1	
		÷	·	·		·	·		·	·		·	·	÷	·	·		·	·	•	·	·		·	·	÷	·	•

[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 \rightarrow string type conversion WORD TO STR(E), DWORD TO STR(E)



Grant Function

Operation processing

(1) WORD_TO_STRING, WORD_TO_STRING_E

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



(2) DWORD_TO_STRING, DWORD_TO_STRING_E

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



Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ${\,\rm (d)}$.

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

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

Coperation 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 s into string type data, and outputs the operation result data from a.

(a) Function without EN/ENO (WORD_TO_STR)

[Structured ladder]

1	.	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
	.		s_wc	ord1	= 1	6#1	123-	• •	•	_w	ORI)RD	о_тс)_S1	rR			.	 	stri	ng1	='	123	4'	•

[ST]

g_string1 := WORD_TO_STR (g_word1);

(b) Function with EN/ENO (WORD_TO_STR_E)

[Structured ladder]

L	•		•			·			•			•			·			•		÷.,	•			•	
	•	•	ta I	ool	1 ·	·	·	·	•	·	-		W	DRI	р_т(o_s	TR	E		_	·	ج	boo	13	÷
				·			67W	ord	1 _		EN JW	DRI	>						EN			5.	stri	n s1	

[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 (s) into string type data, and outputs the operation result data from (d).
 - (a) Function without EN/ENO (DWORD_TO_STR)

[Structured ladder]

1	.																			
	·				÷	·									÷					
	· .	. 5	_dw	ord		•	ر	WOI		wo	RD.	.то	ST	٦			•	,5.	stri	Ingl
	·	·	·	·	·	·	•		•	•	•	•	•	•	•	•	•	·	•	•

[ST]

g_string1:= DWORD_TO_STR (g_dword1);

High Performance

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

		Universal Performance
WORD_TO_ DWORD_TO		_E: With EN/ENO
Structure WORD_TO EN s		indicates any of the following functions. WORD_TO_TIME WORD_TO_TIME_E DWORD_TO_TIME DWORD_TO_TIME DWORD_TO_TIME_E
Input argument, Output argument,	EN:Executing condition (TRUE: Execution s:s:InputENO:Output status (TRUE: Normal, FALSE: Output	:Word (unsigned)/16-bit string, double word (unsigned)/32-bit string

Grant Function

Operation processing

(1) WORD_TO_TIME, WORD_TO_TIME_E

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



(2) DWORD_TO_TIME, DWORD_TO_TIME_E

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



Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ${\ensuremath{\, \mathrm{d}}}$.

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

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

✓ Operation Error

No operation error occurs in the execution of the 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 s into time type data, and outputs the operation result from (d).

(a) Function without EN/ENO (WORD_TO_TIME)

[Structured ladder]

1	.	•	•	•	•	•	•	•	•		•	•	•	•	•		•	•	
		6_W	ordi	1	•	_W	ORI		/OR	D_T	ю, т	TIME	•			•		tim	≥1

[ST]

g_time1 := WORD_TO_TIME (g_word1);

(b) Function with EN/ENO (WORD_TO_TIME_E)

[Structured ladder]

	-		ŀ	•	6-W	ord	1	_	EN JW	I ORI	D						EN	10			time.	
·	۶.	bool	t:		•	·	·	·				WOR	RD.	TO.	TIM	IE_E			·	•	boq	
·						÷						÷	·		·	·			·	·		
.																						

[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 (s) into time type data, and outputs the operation result from (d).

(a) Function without EN/ENO (DWORD_TO_TIME)

[Structured ladder]

1	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	
	. 6	_dw	ord	1 _	•	م	WO			RD.	тο.	TIM.	IE			ŀ		tim	e1
		•	•	•	•	•	•	•	•	•	•	•	•	•	•		•	•	

[ST]

g_time1 := DWORD_TO_TIME (g_dword1)



Operation processing

Converts string type data input to $\,\,\mathrm{(s)}\,$ into bit type data, and outputs the operation result from

(d) .

When the input value is 0, FALSE is output in bit type data. When the input value is other than 0, TRUE is output in bit type data.

'0'	FALSE
'1'	
String type	Bit type

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from $\, \mathbb{d}$.

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

EN	ENO	Ø
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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

Program Example

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

(a) Function without EN/ENO (STR_TO_BOOL) [Structured ladder]

		•	•	:	•	:	:	•	:	:	•	:	:	•	:	:	•	•
	.	E_st	tring	1 _		_s	TRI		R_T	Ο,E	300)L			•	5.	boc	ol1
	·			·												·	·	

[ST]

g_bool1 := STR_TO_BOOL (g_string1);

(b) Function with EN/ENO (STR_TO_BOOL_E)

[Structured ladder]

.																					
·	•	• •	•	•		•	•	·		•		•	•		•		· .	÷	•		
•	رء ا	ool1 ·	•	•	1	•	•	EN		S	TR_	то.	вo	DLJ	E	EN		•	ج	boc	13
			. (s_sti	ring	1_		_	' TRII	NG						EN			<u>5</u> .	boc	12
·																					

[ST]

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

5.1.28 String type \rightarrow word (signed), double word (signed) type conversion

STR_TO_INT(_E), STR_TO_DINT(_E)



Grant Function

Operation processing

- (1) STRING_TO_INT, STRING_TO_INT_E
 - (a) Converts string type data input to ${}_{\textcircled{S}}$ into word (signed) type data, and outputs the operation result from ${}_{\textcircled{G}}$.



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

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

	High-order byte	Low-order byte	
String 1st word	ASCII code of billions place	Sign data	
2nd word	ASCII code of ten-millions place	ASCII code of hundred-millions place	
3rd word	ASCII code of hundred-thousands place	ASCII code of millions place	
4th word	ASCII code of thousands place	ASCII code of ten-thousands place	Double word (signed) type
5th word	ASCII code of tens place	ASCII code of hundreds place	
6th word	ООн	ASCII code of units place	

Indicates the end of the character string.

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

Operation result

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

Operation result	٩
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	Ø
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 (a) is undefined. In this case, create a program so that the data output from (a) 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 s into word (signed) type data, and outputs the operation result from d.
 - (a) Function without EN/ENO (STR_TO_INT)

[Structured ladder]

1											·												
	•	·	÷	·	·	÷	·	÷	÷		÷	÷	·	·	·	÷	÷	÷	·	÷	·	·	÷
	.5	stri	nel	= '	-12	345		•	_s'	TRI		R_T	U)	ЧT			•		jnt1	= ·	-12	345	; .
	•	·	·	•	·	·	•	·	•	•	•	•	•	•	•	•	•		·	·	•	·	÷

[ST]

g_int1 := STR_TO_INT (g_string1);

(b) Function with EN/ENO (STR_TO_INT_E)

[Structured ladder]

1		:	•	•	•	:	•	•	•	•	•	:	•	:	:	•	•	:	:	:	:	•	:	•	•
	Ļ			6. 	600 	∥· ⊩–			•	•		EN	1	ŝ	STR	сто	ЛГ	T,E		EN	ю				513
		:	:	:	•	:		s_sti	ring	1_		_S'	TRI	NG									5.	jn t1	

[ST]

- g_bool3 := STR_TO_INT_E (g_bool1, g_string1, g_int1);
- (2) The program which converts string type data input to \odot into double word (signed) type data, and outputs the operation result from a.
 - (a) Function without EN/ENO (STR_TO_DINT)

[Structured ladder]

1			•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
		5.5	tring	1 =	'	_65				_S	TRI		TR_	то_	DIN	т					din f	1 =	65	000),
	·	•	·		·	÷	÷	·	·	•	·	·	•	•	÷	÷	·		•	·	·	÷	·	·	÷

[ST]

g_dint1 := STR_TO_DINT (g_string1);

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

Universal

		Ľ	
STR_TO_REAL(_E)	_E: With EN/ENO	
Structured lade	ENO:= <u>STR_TO_REAL_E</u> (EN, s, d);	functions. STR_TO_REAL ST	the following
Input argument, EN: s: Output argument, EN0 d:	Input	:Bit :String :Bit :Single-precision real	

Operation processing

(1) Converts string type (decimal point form/exponent form) data input to ${}_{\mbox{\scriptsize $\$$}}$ into single-precision real type data, and outputs the operation result from ${}_{\mbox{\scriptsize d}}$.

	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	Single-precision real type
6th word	ASCII code of 11th character	ASCII code of 10th character	
7th word	00н (Indicates the end	of the character string.)	

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

(a) Decimal point form

	High-order byte	Low-order byte	
String 1st word	31н(1)	2Dн(-)	
2nd word	33н(3)	2Ен(.)	
3rd word	30н(0)	35н(5)	
4th word	34н(4)	33н(3)	-1.35034
5th word	0	Юн	Single-precision real type
	- [1].]3	5 0 3 4	

(b) Exponent form

	High-order byte	Low-order byte	
String 1st word	31н(1)	2Dн(-)	
2nd word	33н(3)	2Ен(.)	
3rd word	30н(0)	35н(5)	
4th word	34н(4)	33н(3)	-1.35034E-10
5th word	2Dн(-)	45н(E)	Single-precision real type
6th word	30н(0)	31н(1)	
7th word		00н	
_	- 1. 35)34E-110	_

- (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





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

Single-precision real type



- (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 $_{\odot}$ is string type data within the following range. ASCII code: '30H' to '39H', '45H', '2BH', '2DH', '2EH', '20H', and '00H'

Operation result

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

Operation result	đ
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	ð
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 (a) is undefined. In this case, create a program so that the data output from (a) 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.
- 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) (Error code: 4100)

(Error code: 4100)

The number of characters is 0 or exceeding 24.

Program Example

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

(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]

1																								
	·	·	·	·		·		÷	·	·	÷			÷	·	·	÷		·	÷	·	·		·
	•	•	εJ	ool	1.	·	·	·	•	·	-		s	TR.	TO.	RE,	ALJ	E	EN		•	ج	boc	13
						. (;sti	ring	1_		EN "S	i TRI	NG						EN	U			rea	
																					•			

[ST]

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

STR_TO_REAL(_E)

5.1.30 String type \rightarrow word (unsigned)/16-bit string, double word (unsigned)/32-bit string type conversion STR TO WORD(E), STR TO DWORD(E)



Grant Function

Operation processing

(1) STR_TO_WORD, STR_TO_WORD_E

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



(2) STR_TO_DWORD, STR_TO_DWORD_E

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

and outputs the operation result from $\ensuremath{\textcircled{}}$.

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 \bigcirc .

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	٥
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

✓ Operation Error

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

Program Example

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

(a) Function without EN/ENO (STR_TO_WORD)

[Structured ladder]

		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
	5	_str	ring	I _		_s	TRI		R_T	0./	NOF	8D			•	5.	woi	rd1
						•			•			•		÷				

[ST]

g_word1 := STR_TO_WORD (g_string1);

(b) Function with EN/ENO (STR_TO_WORD_E)

[Structured ladder]

1			•	•		•	•				:	:	:		•	:		:			•			
		•	s.bi	ool1			•	•		•	EN		SI	IR_1	ro_	WOI	RD_	E	EN		•	"Б.	boc	13
				-		. 6	:_str	ingi	_		EN "S	ı TRIİ	NG						EN			_	wor	
		•	·	·	•	·	÷	·	·	·	·	•	÷	·	÷	·	·	·	·	·	÷	·	·	·

[ST]

- g_bool3 := STR_TO_WORD_E(g_bool1, g_string1, g_word1);
- (2) The program which converts string type data input to \odot into double word (unsigned)/32-bit string type data, and outputs the operation result from \bigcirc .

(a) Function without EN/ENO (STR_TO_DWORD)

[Structured ladder]

	:	÷	:	÷	÷	:	:	÷	:	÷	÷	:	÷	:	:	:	:	:	÷
.	sti	rin gʻ	1					TR_	тο.	D₩	ORI	D			.	ह	dwo	ord1	·
					_ <u>s</u>	TRI	NG	•		•	•		•	•					

[ST] g_dword1 := STR_TO_DWORD (g_string1);



Grant Function

Operation processing

Converts string type data input to ${}_{\textcircled{S}}$ into time type data, and outputs the operation result from ${}_{\textcircled{O}}$.



STR_TO_TIME(_E)

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from \bigcirc .

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	٥
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 (a) is undefined. In this case, create a program so that the data output from (a) 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 ${}_{\textcircled{s}}$ into time type data, and outputs the operation result from ${}_{\textcircled{d}}$.

(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]

1																									
	•	•	•	·		•	·		·	·		·	·		·	·		·	·		·	·		·	·
			ta	000	11 ·	•	•	•	•	•	EN		s	TR.	то	TIN	∕IE_E	•	EN		•		boc	13	•
						. (stı,	ring	_			' TRII	NG						EN			_s.	tim	e1	
	•		·	·	•	·	·	·	·	·	•	·	·	·	·	·	·	·	·	·	·	·	•	·	•

[ST]

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

5.1.32 String type \rightarrow BCD type conversion



\overleftrightarrow Function

Operation processing

(1) Converts string type data input to \odot 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 \odot is string type data within the following range. ASCII code: '30H' to '39H', '45H', '2BH', '2DH', '2EH', '20H', and '00H'

Operation result

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

Operation result	Ø
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	(b)
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 (d) is undefined. In this case, create a program so that the data output from (d) is not used.

Operation Error

An operation error occurs in the following cases.

• 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 s into BCD type data, and outputs the operation result from d.

(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]

1	.																							
	·		·	·	÷	·	·	÷	·	·	÷		•	•	÷	•	·	÷	•	·	·	·	·	
	ŀ	•	Б.]	bool	1 ·	•	•	•	•	•	EN		S	TR	то	,B0	D_E		EN		•	_5.	boc	513
				•		. 6	s_str	ring	1_			' TRII	NG						EN			_s.	wor	rd1
	·	·	·	·	1	·	·	÷	·	·	÷	·	·	·	÷	·	·	·	·	·	·	·	·	÷

[ST]

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

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



Operation processing

- (1) BCD_TO_INT, BCD_TO_INT_E
 - (a) Converts BCD type data input to ${}_{\textcircled{S}}$ into word (signed) type data, and outputs the operation result from ${}_{\textcircled{O}}$.



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

5-101

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



(b) The value to be input to \odot 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	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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 (s) into word (signed) type data, and outputs the operation result from (d).
 - (a) Function without EN/ENO (BCD_TO_INT)

[Structured ladder]

1	.	•	:	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
	.	. б.	wo	rd1	= 1	6#1	23	4	•	в	CD	BC	D_T	οJI	NT			•	Ę.	int1	= '	123	4	
	·	•	•	·	•		·	•	•	·	•		·	•		·	•		·	·		·	•	•

[ST]

g_int1 := BCD_TO_INT (g_word1);

(b) Function with EN/ENO (BCD_TO_INT_E)

[Structured ladder]

	t		-		ŀ	6.W	ord	1 _	 EN JBI						EN	10		joc jnt1	
			٤.	bool	l1 ·					E	BCD	то) IN	ΤЕ			.		
1																			

[ST]

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

- (2) The program which converts BCD type data input to \odot into double word (signed) type data, and outputs the operation result from a.
 - (a) Function without EN/ENO (BCD_TO_DINT)

[Structured ladder]

1																									
				·								÷			·			÷			·			·	
		. 5	.wo	Ird1	= 1	6#(000	o	•	в	CD	B	DD_	TO	DIN	IT			•		din f	1 =	0		:
	•	·	•	•	·	•		•	•	•	•	•	•	·	•	•	•	•	•	·	•	·	·		·

[ST]

g_dint1 := BCD_TO_DINT (g_word1);

BCD_TO_INT(_E), BCD_TO_DINT(_E)



Operation processing

Converts BCD type data input to ${}_{\textcircled{S}}$ into string type data, and outputs the operation result from ${}_{\textcircled{G}}$.



Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from $\, \mathbb{d}$.

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

EN	ENO	٥
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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

Program Example

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

(a) Function without EN/ENO (BCD _TO_STR) [Structured ladder]

1	.																	
	·			·			·		·	·		·	·	÷	÷	·		
	·			·	÷			BC	р_т(o_s	TR			·	•			÷
	·	67A	vord1	-	_	_B0	D								_£.	stri	ng1	•
	·	•	·	·	•	·	·		·	·	•	·	·			·	·	

[ST]

g_string1 := BCD_TO_STR (g_word1);

(b) Function with EN/ENO (BCD_TO_STR_E)

[Structured ladder]

1			•	•	•	•	•	•	•	•	•	•	•	•	•	•		•	•	•	•	•		•
		•	ta 	000	11 ·	•	•	•	•	•	EN		E	30D	то	LST	R,E		EN	0	•		boc	513
			-				E-W	ord	۱_			CD										_s.	stri	ng1
	•	·	·	÷	·	·	·	·	·	·	·	·	÷	·		·	·	÷	·	·	÷	·	·	

[ST]

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



Grant Function

Operation processing

Converts time type data input to ${\scriptstyle \textcircled{s}}$ into bit type data, and outputs the operation result from

(d) .

When the input value is 0ms, FALSE is output in bit type data. When the input value is other than 0ms, TRUE is output in bit type data.



Operation result

(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	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

Poperation 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 $\, {\rm (s)}\,$ into bit type data, and outputs the operation result from $\, {\rm (d)}\,$.

(a) Function without EN/ENO (TIME_TO_BOOL) [Structured ladder]

1	.	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
	. .	s_t	ime'	I	•	_TI	ME		TME	Е,ТС) "В(οοι	-				5.	boc	11
	·	•			·	•		•	•		·	•		•	•		•	•	

[ST]

g_bool1 := TIME_TO_BOOL (g_time1);

(b) Function with EN/ENO (TIME_TO_BOOL_E)

[Structured ladder]

1	•	•		•			•			•			•	•					•	•		•			•
	•	•	t.a	000	11 ·	•	•	•	•	•	EN			тім	E_TO	р,в	00	L JE		EN	0	•		boc	ol3
							e_ti	me1		_		ME								EIN	.0		<u>5</u> .	boc	012
	•	·	·	·	·	·	·	·	·	÷	·	·	÷	·	·	÷	·	·	·	·	·	·	·	·	·



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

TIME_TO_BOOL(_E)

5.1.36 Time type \rightarrow word (signed), double word (signed) type conversion

TIME_TO_INT(_E), TIME_TO_DINT(_E)



Grant Function

Operation processing

(1) TIME_TO_INT, TIME _TO_INT_E

Converts time type data input to ${\ensuremath{\scriptstyle (s)}}$ into word (signed) type data, and outputs the operation result from ${\ensuremath{_{(s)}}}$.


(2) TIME _TO_DINT, TIME _TO_DINT_E

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



Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ${\ensuremath{\, \mathrm{d}}}$.

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	Ø
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

TIME_TO_INT(_E), TIME_TO_DINT(_E)

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

```
(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]

	·	•	•		•	•	•	•	•	•	•	:	•	•	:	•	•	•	•	•	•	•	•	:
			ده 	ool	1 ·	•	•	•	•		EN	J	-	ГIМ	E_TO	лГс	IT_E		EN		•	5.	boc	13
	.		e,				e_t	ime	-			ME										_s.	jn t1	
	·			·							·		·			·				·		·	·	

[ST]

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

- (2) The program which converts time type data input to \odot into double word (signed) type data, and outputs the operation result from a.
 - (a) Function without EN/ENO (TIME _TO_DINT)

[Structured ladder]

1	•	•	•	•	•	•	•		•		•	•	•	•	•	•	•	•	•
		s_ti	ime 1		•	_TI	ME		тім	E_T	סכ	INT				·		din t	1
	•	•		·	·	·	·	÷	·	·	·	·	·	·	·	·	·	·	•

[ST]

g_dint1 := TIME_TO_DINT (g_time1);

5.1.37	Time type \rightarrow string typ	e conversion TIME_TO_STR(_E)
TIME_TO_S	TR(_E)	_E: With EN/ENO
Structured TIME_TO EN s		indicates any of the following functions. TIME_TO_STR TIME_TO_STR_E
Input argument, Output argument,	EN: Executing condition (TRUE: Executing condition (TRUE: Executing condition (TRUE: Executing condition) (TRUE: Normal, FAILENO: Output status (TRUE: Normal, FAILeno) (TRUE: TRUE: TRUE) (TRUE: TRUE) (TRUE: TRUE) (TRUE: TRUE) (TRUE: TRUE) (TRUE: TRUE) (TRUE) (:Time

Grant Function

Operation processing

Converts time type data input to ${}_{\textcircled{S}}$ into string type data, and outputs the operation result from ${}_{\textcircled{G}}$.



5

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	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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 ${}_{\textcircled{S}}$ into string type data, and outputs the operation result from ${}_{\textcircled{O}}$.

(a) Function without EN/ENO (TIME_TO_STR) [Structured ladder]

1	:	•	•	•	•	•	•	•	•	•	·	•	•	•	•	•	•	•	•
	. . .	s_ti	ime1		•	_TI	ME		ME,	.то,	_STI	R			·	6.	stri	ng1	

[ST]

g_string1 := TIME_TO_STR (g_time1);

(b) Function with EN/ENO (TIME_TO_STR_E)

[Structured ladder]

1																									
		·	·	·	·	·	·	·	·	·		·	·	·	·	·		·	·	·	·	·		·	•
		•	•	۶.	boo 	11 ·	•	1	•	•	•	EN		Т	IME	.то	_S1	IR J	•	EN		•		boo	13
								e_t	ime	1_		_	ME							EN	0		<u>5</u> .	stri	ng1



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)



C Function

Operation processing

(1) TIME_TO_WORD, TIME _TO_WORD_E

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



(2) TIME _TO_DWORD, TIME _TO_DWORD_E

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



Operation result

(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	Ø
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

> **APPLICATION** FUNCTIONS

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

(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]

	·	·		·		·	·	·	·		·	·	·			·	·	·	·		•	·			·
	•	•	٤.	boo I	11 ·	•	•	•	•	•	EN		٦	ГIМ	E_TO	D_W	ORI	D,E		EN		•	ج	boo	13
							e_t	ime	1_											EN				wor	
	·			·		·	·	·	·	·	•	•	÷	•		•	•	•	•	·	•		·	÷	•

[ST]

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

- (2) The program which converts time type data input to (s) into double word (unsigned)/32-bit string type data, and outputs the operation result from (d).
 - (a) Function without EN/ENO (TIME _TO_DWORD)

[Structured ladder]

1	•	•		•	•	•		•	•	•	•	•	•	•	•	•	•	•	
	•	s_ti	me1		•	JT	ME	TI	ME,	TO,	JD W	/OR	Ð			•	5.	dwo	ord1
	•	•	·	·	•	·	·	•	·	·	•	·	·	·	·	·	·	·	·

[ST]

g_dword1 := TIME_TO_DWORD (g_time1);

5.2 Standard Functions of One Numeric Variable

5.2.1 Absolute value



Grant Function

Operation processing

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

B= A

- (2) The value to be input to ${}_{\odot}$ is word (signed), double word (signed), single-precision real or double-precision real type data.
- (3) When the data type of \odot is word (signed) type and the input value is -32768, 32768 is output from \bigcirc .

When the data type of \bigcirc is double word (signed) type and the input value is -2147483648, 2147483648 is output from \bigcirc .

(No operation error occurs. In case of ABS_E, TRUE is output from ENO.)

ABS(_E)

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	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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 s from d in the same data type as that of s.

(a) Function without EN/ENO (ABS)

[Structured ladder]



[ST]

g_int2:= ABS(g_int1);

(b) Function with EN/ENO (ABS_E)

[Structured ladder]

1		•	•		•	•	•			•						•	•		
	•	•	ta I	oool	1.	•	•	•	•	•	EN	AJ	BS.	E EN		ŀ	Б.	, boo	13
							. 5	jnt	1 _	_	JN			EN	10			jn t2	



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

5-121

5.3 Standard Arithmetic Functions

5.3.1 Addition



Grant Function

Operation processing

(Example) Word (signed) type data



- (2) The values to be input to (a) to (a) are word (signed), double word (signed), single-precision real or double-precision real type data.
- (3) The number of pins for \odot can be changed in the range from 2 to 27.

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

II a	in undernow/overnow occurs in the operation	result, data is output norm @ as follows.										
(a)	 (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 (7FFFн) (0002н) (8001н)	Since the highest-order bit is 1, the result value is negative.										
	–32767 + (–2) = 32766 (8000н) (FFFEн) (7FFEн)	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 under In case of ADD_E, TRUE is output from EN											
	2147483647 + 2 = -2147483647 (7FFFFFFH) (0002н) (80000001н)	Since the highest-order bit is 1, the result value is negative.										
	–2147483648 + (–2) = 2147483646 (8000000н) (FFFEн) (7FFFFFEн)	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 ${\scriptstyle (\ensuremath{\textbf{d}})}$.

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

EN	ENO	Ø
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

*1 When FALSE is output from ENO, the data output from (d) is undefined. In this case, create a program so that the data output from (d) is not used. 5

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

Program Example

The program which performs addition (s + s) on double word (signed) type data input to s and

0 , and outputs the operation result from 0 in the same data type as that of 0 .

(a) Function without EN/ENO (ADD)

[Structured ladder]



[ST]

g_dint3:= g_dint1+g_dint2;

(b) Function with EN/ENO (ADD_E)

[Structured ladder]

:	•	•	• •	•	•	•	•	:	:	•	•	:	•	•	:	:	•
ŀ	•	٤J	bool1 ·		•	•	•	•	EN	A	DD_	E EN			ج	boc	13
						din t	_		JN			EN			_s.	din f	t3
·	•	·	• •		Б.	dint	2 _		JN					•	·	·	
1 .	•	•		•			·			·	•	1	•	•			·

[ST]

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

MUL(_E)

5.3.2 Multiplication

		Universal Hig UD
MUL(_E)		_E: With EN/ENO
Structured EN s1 s2 s27		functions. MUL MUL_E
Input argument, Output argument,	EN: Executing condition (TRUE: Execution, FALSE: Stop s1 to s27: Input ENO: Output status (TRUE: Normal, FALSE: Error) d: Output	pp) :Bit :ANY_NUM :Bit :ANY_NUM

Grant Function

Operation processing

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

`	- I / -	- (- J			
	100) ×	15		1500
	~	,	<u> </u>		\frown
IN1(word (signed) t	ype) ll	N2(word (signed) type)	Word (signed) type

- (2) The values to be input to (a) to (a) are word (signed), double word (signed), single-precision real or double-precision real type data.
- (3) The number of pins for ${}_{\circledast}$ can be changed in the range from 2 to 27.

- (4) If an underflow/overflow occurs in the operation result, data is output from (d) 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 ${\scriptstyle (\ensuremath{\textbf{d}})}$.

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(b)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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

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

Program Example

The program which performs multiplication ($\mathfrak{S} \times \mathfrak{D}$) on double word (signed) type data input to \mathfrak{S} and \mathfrak{D} , and outputs the operation result from \mathfrak{D} in the same data type as that of \mathfrak{S} .

(a) Function without EN/ENO (MUL)

[Structured ladder]

1																		
	•	·	·		·	·	·	·		·	·		·	·		·	·	·
								MU	IL		·							
		€_di	nt1	= 5	567	8 _						_5.	din '	t3 =	:70	066	52	
		€_di	n t2	= 1	23	4 _												

[ST]

g_dint3:= g_dint1*g_dint2;

(b) Function with EN/ENO (MUL_E)

[Structured ladder]

1			•		•	:			:		•			•	•	•	•	•	•
			tء -	ooli	. 	•	•	•	•	•	EN	MU	JLJ	E EN	0	•		boc	ol3
	·	•		•		·		lin t1		_	JN						_5.	din f	t3
	·	•	·	·	·	·	£_0	lin t2	2	_	JN					·	·	·	·
	۱.																		

[ST]

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

5.3.3 Subtraction SUB(E) UD SUB(_E) _E: With EN/ENO indicates any of the following Structured ladder ST functions. SUB_E SUB SUB E ΕN ENO ENO:= SUB_E (EN, s1, s2, d); d s1 s2 Input argument, EN: Executing condition (TRUE: Execution, FALSE: Stop) :Bit s1: :ANY NUM Input s2: ENO: Output status (TRUE: Normal, FALSE: Error) Output argument, :Bit :ANY_NUM d: Output

Operation processing

- (1) Performs subtraction (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 (
 <
 - ${}_{\textcircled{}}$ in the same data type as that of ${}_{\textcircled{}}$.
 - (Example) Word (signed) type data



(2) The values to be input to (s) and (and (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 \bigcirc 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 (7FFFн) (FFFEн) (8001н)	Since the highest-order bit is 1, the result value is negative.
	–32767 – 2 = 32766 (8000н) (0002н) (7FFEн)	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 under In case of SUB_E, TRUE is output from EN	
	2147483647 – (–2) = –2147483647 (7FFFFFFH) (FFFEH) (8000001н)	Since the highest-order bit is 1, the result value is negative.
	–2147483648 – 2 = 2147483646 (8000000н) (0002н) (7FFFFFEн)	Since the highest-order bit is 0, the result value is positive.

Operation result

- Function without EN/ENO
 An operation is executed and the operation value is output from (d).
- (2) Function with EN/ENO The following table shows the executing conditions and operation results.

EN	ENO	٥
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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

Program Example

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

(a) Function without EN/ENO (SUB)

[Structured ladder]



[ST]

g_dint3:= g_dint1-g_dint2;

(b) Function with EN/ENO (SUB_E)

[Structured ladder]

	.																		
	· ·		·		·	·		·		·		÷			·	·			
		•	ta 	ool	1 ·	•	•	•	•	•	EN		UB.	E EN	10	•	5.	boc	ol3
	.		- 1				6.C	din ti	1_		JN	1					_5.	din f	t3
	.		÷		·	·	6.Q	lin tâ	2 _		JN	12					·	·	÷
	· ·							·			·								

[ST]

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

5-130

DIV(_E)

5.3.4 Division

		Universal Perform
DIV(_E)		_E: With EN/ENO
Structured EN s1 s2	·	2, d);
Input argument, Output argument,	EN: Executing condition (TRUE: Execution s1: s2: ENO: Output status (TRUE: Normal, FALSE: d: Output	:ANY_NUM

Grant Function

Operation processing

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



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

(The value to be input to must be other than 0.)

Operation result

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

Operation result	(d)
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	(b)
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 (a) is undefined. In this case, create a program so that the data output from (a) is not used.

An operation error occurs in the following case.

• The value to be input to @ is 0. (Division by 0) (Error code: 4100)

Program Example

The program which performs division ((\mathfrak{S}) $\div \mathfrak{Q}$) on double word (signed) type data input to (\mathfrak{S}) and (\mathfrak{Q}), and outputs the quotient of the operation result from (\mathfrak{Q}) in the same data type as that of (\mathfrak{S}).

(a) Function without EN/ENO (DIV)

[Structured ladder]

.	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	
.	g_di	int1	= १	567:	8			DI	v		ŀ		din f	t3 =	4	•	
.	€_di	in t2	= 1	23	4						·	•	•	•	·	•	
		 	 . g_dint1		. s_dint1 = 567;									. s_dint1 = 5678DIVs_din		. g_dint1 = 5678 DIVg_dint3 = 4	. g_dint1 = 5678 DIVg_dint3 = 4 .

[ST]

g_dint3:= g_dint1/g_dint2;

(b) Function with EN/ENO (DIV_E)

[Structured ladder]

1		
	· g:bool1 · · · · · DIV E	
	g_dint1 — _IN1	g_bool3 g_dint3·
	· · · · g_dint2 — _IN2	

[ST]

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

DIV(_E)

5.3.5 Modulus operation MOD(E) UD MOD(_E) E: With EN/ENO indicates any of the following Structured ladder ST functions. MOD_E MOD MOD E ΕN ENO ENO:= MOD_E (EN, s1, s2, d); d s1 s2 Input argument, EN: Executing condition (TRUE: Execution, FALSE: Stop) :Bit s1: :ANY INT Input s2: Output status (TRUE: Normal, FALSE: Error) Output argument, ENO: :Bit :ANY_INT d: Output

Operation processing

(1) Performs division (S) ÷ S
 (2) on word (signed) or double word (signed) type data input to S
 (3) and S
 (4) and S
 (5) and S
 (5) and S
 (6) and S
 (7) and S
 (8) and S
 (8) and S
 (9) and S
 (9) and S
 (1) and S
 (1) and S
 (1) and S
 (1) and S
 (2) and S
 (3) and S
 (4) and S
 (5) and S
 (1) and S
 (2) and S
 (3) and S
 (4) and S
 (5) and S
 (5) and S
 (1) and S
 (1) and S
 (1) and S
 (2) and S
 (3) and S
 (4) and S
 (4) and S
 (5) and S
 (5) and S
 (5) and S
 (5) and S
 (6) and S
 (7) and S
 (8) and S
 (9) and S
 (9) and S
 (1) and S
 (

(Example) Word (signed) type data



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

Operation result

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

Operation result	٥
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	(d)
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 (d) is undefined. In this case, create a program so that the data output from (d) is not used.

An operation error occurs in the following case.

• The value to be input to @ is 0. (Division by 0) (Error code: 4100)

Program Example

The program which performs division ($\mathfrak{S} \div \mathfrak{Q}$) on double word (signed) type data input to \mathfrak{S} and \mathfrak{Q} , and outputs the remainder of the operation result from \mathfrak{G} in the same data type as that of

 \odot .

(a) Function without EN/ENO (MOD)

[Structured ladder]

.	·	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
. .	g_di	int1	= 5	567:	8	•	JN		10 C	•				din f	t3 =	74	2.	•
.	€_di	int2	= 1	23	4		JN					·	•	•	•	•	•	:

[ST]

g_dint3:= g_dint1 MOD g_dint2;

(b) Function with EN/ENO (MOD_E)

[Structured ladder]

.																		
·		·	·	·	·		·			·	·		·	·				·
		ta	lood	1 ·	•	•	•	•	•	EN	M	DD.	E EN	0	•	_5.	boc	ol3
			•				din t'			JN	1		EN	.0		_s.	din f	t3
·		·	·	·	·	6_(din t	2 _	_	JN	2				•	·	·	·
·		·	·	·		·			·	·		·	·		·	·		

[ST]

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

5.3.6 Exponentiation EXPT(_E) Universal UD EXPT(_E) E: With EN/ENO indicates any of the following **Structured ladder** ST functions. EXPT_E EXPT_E EXPT ΕN ENO ENO:= EXPT_E (EN, s1, s2, d); s1 d s2 Input argument, EN: Executing condition (TRUE: Execution, FALSE: Stop) :Bit s1: Input :ANY_REAL s2: :ANY_NUM Input ENO: Output status (TRUE: Normal, FALSE: Error) Output argument, :Bit Output :ANY_REAL d:

Grant Function

Operation processing

Performs exponentiation @ on single-precision real or double-precision real type data input

to ${\ensuremath{\scriptsize \ensuremath{\scriptsize \ensuremath{\ensuremath{\scriptsize \ensuremath{\scriptsize \ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\smalemath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\smalemath{\ensuremath{\smalemath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\smalemath{\smalemath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\smalemath{\ensuremath{\smalemath{\smalemath{\ensuremath{\ensuremath{\ensuremath{\ensurem$



EXPT(_E)

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	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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 and .

(a) Function without EN/ENO (EXPT)

[Structured ladder]

1													
	•	·	•	•	۰.	1		·	· .	•	·	•	
	·	·	•	•	•		EXP	РΤ			·		
			e al 1	_	_	In1					_s.	real	12
			eal1 jnt1	_	_	In1 In2					_5.	real	

[ST]

g_real2:= EXPT(g_real1, g_int1);

(b) Function with EN/ENO (EXPT_E)

[Structured ladder]

1	·																	
	·	·		• •			·				÷				·			
		•	ta	ool1 ·	•	•	•	•	•	EN	E	EXP	T_E	EN	•	ج	boc	13
							real			In1				EN			rea	
	·		·	• •	÷	.5	jnt	1_		In2					.	·	÷	•
	·			· ·		·	·	·	·	•	·	·	·	•		·		·

[ST]

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



Grant Function

Operation processing



5-140

Operation result

(1) Function without EN/ENO

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

EN	ENO	(b)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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

Program Example

(a) Function without EN/ENO (MOVE)

[Structured ladder]



[ST]

g_int2:= MOVE(g_int1);

(b) Function with EN/ENO (MOVE_E)

[Structured ladder]

. .	•	•	•	•	:	•	•	:	•	•	•	•	•	:	•	•	•	•	•
•	•	6.	bool	1.	•	•	•	•	•	EN	ł	NO۱	Æ,E	EN		ŀ	ج	boc	J3
						. 6	jnt	1_		JN				EN	10		<u>5</u> .	jn t2	2.
·																			

[ST]

g_bool3 := MOVE_E(g_bool1, g_int1, g_int2);

5-142

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)



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 in the word in the operation result from in the same data type as that of variables in to in the same data type as that of variables in the word (unsigned)/16-bit string type data



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

- (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 it to it by bit, and outputs the operation result from (a) in the same data type as that of variables it to (b). (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 ⑤ to ⑳ bit by bit, and outputs the operation result from ⓓ in the same data type as that of variables ⑥ to ⑳ . (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 ⊚ and ⊗ first, and XOR is successively performed between the result and ⊗.

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

In this manner, XOR is repeated by the number of variables 's' in the order with \circledast , \circledast 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)/32bit string type data input to variable (s) bit by bit, and outputs the operation result from (d) in the same data type as that of variable (s) . (Example) Word (unsigned)/16-bit string type data



(5) The value to be input to variables (5) 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 ${\,\rm (d)}$.

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

EN	ENO	(b)
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.

5

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 is to in bit by bit, and outputs the operation result from in the same data type as that of variables it to it.
 - (a) Function without EN/ENO (AND)

[Structured ladder]

1	.																	
	.																	
	.									AN	ID.							
1			шo	rd1	= 1	6#F	FO	F		~	~		mor	ch3	= 1 1	5#1	204	
	1 .		 -110			~		_	_			_	 ,					
	1:	÷					23	_										

[ST]

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

(b) Function with EN/ENO (AND_E)

[Structured ladder]

1	.			•						•	•			•		•			
		•	¢J	000	11 ·	•	•	•	•	•	EN	AJ	VD_	E EN	0	•	_5.	bod	ol3
							e'w				JN			EN	.0		_5.	woi	rd3
		•	•	·		5	E'W	ord	2 -	_	JN					•	•	•	
	·	•	·	1	•	·			·			•			•			·	•

[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 is to in bit by bit, and outputs the operation result from in the same data type as that of variables is to in to in the same data type as that of variables is to interval.

(a) Function without EN/ENO (OR)

[Structured ladder]

1	Ι.																			
	.										0		.							
	.		.5	_wo	rd1	= 1	6#	5FO	3 _		0	R		_5	_woi	rd3	= 1	6 #C	FO	F.
	.		۶.	wor	'd2	= 1	6#9	CC	D_											
										_		_								

[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 is to in bit by bit, and outputs the operation result from (a) in the same data type as that of variables (c) to (a).
 - (a) Function without EN/ENO (XOR)

[Structured ladder]

	L		·			·			·					·		·			·		
	L :		•			•			•			•		· · .		•			•		
	L .												XC								
1					mor	d1 - :	- 1.6	A # 2	44	۵.					-	mor	ch2	= 1.6	5 # 🖻	15.	Δ.
	·			5.	wor	d1 =	= 1 6	5#A	AA,	۹_	_				_5.	wor	'd3	= 1 (5#B	15,	Α.
	L .	•	•					5#A 6#1		_	_				_5.	wor	rd3	= 1 (5#B	15,	A.

[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 is bit by bit, and outputs the operation result from in the same data type as that of variable is.

(a) Function without EN/ENO (NOT)

[Structured ladder]

NOT s_word1 = 16#AAAA _____JN ____s_word2 = 16#5555 . .

[ST]

g_word2 :=NOT (g_word1);

5.5 Standard Selection Functions

5.5.1 Selection



Grant Function

Operation processing

Selects either of values input to
 and
 and
 according to the value input to
 s), and outputs
 the operation result from
 d) in the same data type as that of
 and
 and
 s).

When the input value of \bigcirc is FALSE, the value input to \oslash is output from \bigcirc . When the input value of \bigcirc is TRUE, the value input to \bigcirc is output from \bigcirc . (Example) and are word (signed) type data



Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from $\, \mathbb{d}$.

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	٥
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

5

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

Program Example

The program which selects either of values input to 0 and 0 according to the value input to 0, and outputs the operation result from 0 in the same data type as that of 0 and 0.

(a) Function without EN/ENO (SEL)

[Structured ladder]

1			•						•	•			•		•	•		•		
	Ļ	•	•	с) —	ooli		•	•	•	•	•	_G	9	EL			•		wor	d3
		·	÷	÷	· - "			6-W			_	JNC)					·	÷	
		·	·	·	·		1	67M	ord2		_	JN1					·	·	·	·
		•		·	•	•	•	•	•	•	·	•	•	·	•		•	·		•

[ST]

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

(b) Function with EN/ENO (SEL_E)

[Structured ladder]

1	•	•	:	:		•	•	•		•			:	:		•	•		•
		•	t.a	ool	1. 	•	•	•	•	•	EN	SI	ELJ	EN	0	•		boo	13
			- 1	•		·		ool2		_	_G						_S.	wor	d3
	.	•	·	·	·		67M)			_	JN	0				·	·	·	·
	· ·	•	·	·	·	·	ETM:	ord2	-	_	JN	1				·	·	·	·
	· ·	•	·	·	·	·		·	·	•	·	·	÷	·	·	·	·	·	·

[ST]

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

MUX(E)

5.5.2 Multiplexer

		Universal Perform
MUX(_E)		_E: With EN/ENO
Structured EN n s1 s2 to s27		junctions. MUX MUX_E
Input argument, Output argument,	EN: Executing condition (TRUE: Execution, FALSE: 5 n: Output value selection s1 to s27: Input ENO: Output status (TRUE: Normal, FALSE: Error) d: Output	Stop) :Bit :Word (signed) :ANY :Bit :ANY

Grant Function

Operation processing

Selects the value to be output among the values input to variables in to a according to the value input to n, and outputs the operation result from in the same data type as that of variables in to a.

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

When the input value of n is n , the value input to sn is output from @ . (Example) Word (signed) type data



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

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

- (3) The value to be input to n is word (signed) type data within the range from 1 to 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 ${\ensuremath{\, \mathrm{d}}}$.

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

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

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

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 s and s according to the value input to m, and outputs the operation result from m in the same data type as that of variable s or s.

(a) Function without EN/ENO (MUX)

[Structured ladder]

1																					
	•	·	·	·	·	•	·	·	·	·	·	·	·	·	·	•	·	·	·	·	•
	•			·			·	·		·	÷		MU	ж		·	÷		·		
	•		÷		÷				= ;		_	к				_	_5.	jn t4	1 = !	567	8
						БĴ	nt2	= 1	23	4	_	JN	1			•					
						ĘĴ	nt3	= 5	567	3_	_	JN	ł								
	•		·	·	÷	·	·	·	·	·		•	÷		·	•	·	·	·		

[ST]

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

(b) Function with EN/ENO (MUX_E)

[Structured ladder]

1	.			•	•		•	•			:	•	•	•	•			
		•	5.	bool 	1· 	•	•		•	•	EN	MU	JXJ	E EN	 ·		boo	13
				<u>.</u>				jnti			ЪК.			EN		5.	jn t4	
	·		·		÷			jnt			JN					÷	·	·
	·	•	•		·	·	.5	jn t:	3 _	_	JN				•	÷	·	·

[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)

UБ

GT(_E) GE(_E) EQ(_E) LE(_E) LT(_E) NE(_E)		_E: With EN/ENO
Structured		functions.
EN EN		GT GT_E GE GE_E
— s1	ENO:= GT_E (EN, s1, s2-s27, d	EQ EQ_E
s2		LE LE_E
		LT LT_E
		NE NE_E
Input argument,	EN: Executing condition (TRUE: Execution, FAL	SE: Stop) :Bit
	s1 to s27: Input	:ANY_SIMPLE
Output argument,	ENO: Output status (TRUE: Normal, FALSE: Error	
	d: Output (TRUE: True value, FALSE: False va	lue) :Bit

Grant Function

Operation processing

- (a) GT, GT_E(>) Performs comparison of $[s_1 > s_2]$ $[s_2 > s_3]$ $[s_1 1) > s_2$ (n)].
 - Outputs TRUE if all of comparisons satisfy (s) (n-1) > (s) (n).
 - Outputs FALSE if any of comparisons satisfies (s) (n-1) \leq (s) (n).
- (b) GE, GE_E(\geq) Performs comparison of [$\mathfrak{S} \geq \mathfrak{Q}$]&[$\mathfrak{Q} \geq \mathfrak{S}$]&...&[\mathfrak{S} (n-1) $\geq \mathfrak{S}$ (n)].
 - Outputs TRUE if all of comparisons satisfy $(n-1) \ge (n)$.
 - Outputs FALSE if any of comparisons satisfies (n−1) < (s) (n).
- (c) EQ, EQ_E(=) Performs comparison of $[s_1 = s_2]\&[s_2 = s_3]\&\cdots\&[s_n(n-1) = s_n(n)].$
 - Outputs TRUE if all of comparisons satisfy (n-1) = (n).
 - Outputs FALSE if any of comparisons satisfies (s) (n-1) $^{\pm}$ (s) (n).
- (d) LE, LE_E(\leq) Performs comparison of [$\mathfrak{S} \leq \mathfrak{Q}$]&[$\mathfrak{Q} \leq \mathfrak{S}$]&...&[\mathfrak{S} (n-1) $\leq \mathfrak{S}$ (n)].
 - Outputs TRUE if all comparisons satisfy (s) $(n-1) \leq (s)$ (n).
 - Outputs FALSE if any of comparisons satisfies (s) (n−1) > (s) (n).
- (e) LT, LT_E(<) Performs comparison of $[s_1 < s_2]\&[s_2 < s_3]\& \dots \&[s_n(n-1) < s_n(n)].$
 - Outputs TRUE if all comparisons satisfy (n-1) < (n).
 - Outputs FALSE if any of comparisons satisfies (s) (n–1) \ge (s) (n).
- (f) NE, NE_E(<>) Performs comparison of [$\mathfrak{G} \neq \mathfrak{Q}$].
 - Outputs TRUE if s \neq s.
 - 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 (s) and (2).

Operation result

- Function without EN/ENO
 An operation is executed and the operation value is output from (d).
- (2) Function with EN/ENO The following table shows the executing conditions and operation results.

EN	ENO	đ
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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 \odot and \odot , and outputs the operation result from \odot .

(a) Function without EN/ENO (GT)

[Structured ladder]

1																			
											·		G	г					
						٤J	nt1	= 5	678	3_	_						_ 5 _	boo	11
						٤j	nt2	= 1	234	4_	_								
	·	·	·	·	·		·		·	·	•	•	•	•	•	·	·	·	

[ST]

g_bool1:= (g_int1) > (g_int2);

(b) Function with EN/ENO (GT_E)

[Structured ladder]

					÷	·			·			·	·					
	•	•	٤J	bool	1.	•	•	•	•	•	EN	GT	je EN		•	ج	boo	513
								jnt			JN		EN	0			boo	
	•			·	·	`	. 5	jn t	2 _		JN	ı			.			
	· ·	÷	÷			·			·	÷		·				÷		

[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)

Grant Function

Operation processing

(1) Extracts the specified number of characters from the specified start position in the character string input to \odot , and outputs the operation result from \bigcirc .

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

(Example) Values input to n and 2 are 5



- (2) The value to be input to \odot is string type data within the range from 0 to 255 bytes.
- (3) The value to be input to n is word (signed) type data within the range from 0 to 255.
 (The input value must not exceed the number of characters of character string input to s.)
- (4) 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
 .)

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ${\scriptstyle (\ensuremath{\mathfrak{g}})}$.

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(b)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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 s, and outputs the operation result from @.

(a) Function without EN/ENO (MID)

[Structured ladder]

1																										
	·	·		·						·		·	·		·	·		·		·		·	·			•
	·			·						·				MI	5					·	·		·			
	Б.	strir	ng1	= '7	ABC	DEF	-12	345	· _		JN						M	D		_£.	stri	ng2	= '	EF1	23'	
						E	in t1	= 5	5																	
				•	•	.0.			· —	_	L.								•		•	•		•	•	•
										_	ь Р									:	•					

[ST]

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

(b) Function with EN/ENO (MID_E)

[Structured ladder]

1			•		•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	
			•	1.3 	ool	n∙ ⊫	•	1	•	•	•	EN		IDJ	EN	10	Ŀ		boc	ol3	
				. "		÷.,	. 6		ring			JN						_s.	stri	in g2	
			·	·	·	·	5		jnt		_	Т					·	·	·	·	
			·	·	·	·	·	. 5	jnt	- 2	_	Р					ŀ	·	·	•	
1	- 1																				

[ST]

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

MID(_E)

5.7.2 String concatenation

CONCAT(_E)		_E: With EN/ENO
Structured EN s1 s2		ST ENO:= CONCAR_E (EN, s1, s2, d);	indicates any of the following functions.
Input argument, Output argument,	EN: s1: s2: ENO:	Executing condition (TRUE: Execution, FALSE: Stop) Input Output status (TRUE: Normal, FALSE: Error)	:Bit :String :Bit

CONCAT(E)

Grant Function

Operation processing

(1) Concatenates the character string input to @ following the one input to I , and outputs the operation result from ().

This function concatenates character string O with ignoring 'OOH', which indicates the end of character string O.

If the concatenated character string has over 255 bytes, the character string up to 255 bytes is output.



(2) The values to be input to i and a 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 ${\,\rm (d)}$.

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

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

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

Program Example

The program which concatenates the character string input to following the one input to , and outputs the operation result from .

(a) Function without EN/ENO (CONCAT)

[Structured ladder]

1	.																											
	·	·		·	·		·	·	•	·	·	·		·	·		•	·	·	·	•		·	·		·	·	•
	·										0	nΝ	CAI	г														
	.					ABC			 ٦N	1	-				100	NCA	л		_5.	stri	ing3	= '	ABI	DDE	F1 2	2345	5'	
	·		€_st	rine	:2 =	12	345	5' <u> </u>	 JN										÷		·				÷		÷	
	.																											

[ST]

g_string3:=CONCAT(g_string1, g_string2);

(b) Function with EN/ENO (CONCAT_E)

[Structured ladder]

1	.										·										·
	·				·			·	·	·	·	·	·	·	·		·	·		·	
	·	•	s.b	ool	1 ·	•	•	•	•	÷	EN		201	ICA				•	ج	boa	13
				. '		. е	_str	rinst	1	_	JN					EN					ng3
	·		÷	·	·	, Ε	_str	ringă	2 _	_	JN							•	÷		·
	- I																				

[ST]

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

5.7.3 String insertion **INSERT(E)** Universa UD INSERT(_E) E: With EN/ENO indicates any of the following **Structured ladder** ST functions. INSERT_E INSERT INSERT E EN ENO ENO:= INSERT_E (EN, s1, s2, n, d); s1 d s2 n Input argument, EN: Executing condition (TRUE: Execution, FALSE: Stop) :Bit s1: Input :String s2: n: Start position to be inserted :Word (signed) ENO: Output status (TRUE: Normal, FALSE: Error) :Bit Output argument, d: Output :String

Grant Function

Operation processing

(1) Inserts the character string input to 0 to the specified position in the character string input to 0, and outputs the operation result from 0.

Specify the start position of the character string to be inserted by the value input to .

After the insertion of character string 0 to character string 0, '00H' that indicates the end of character string 0 is ignored. If the character string after insertion has over 255 bytes, the character string up to 255 bytes is output.

(Example) Value input to n is 4

	Input val <u>'ABC</u>				Output <u>'ABC12</u> :	value 3456DE'	
High	n-order byte	Low-order b	yte	Hig	h-order byte	Low-order b	oyte
1st word	42н(В)	41н(А)			42н(В)	41н(A)	1st word
2nd word	44н(D)	43н(С)	Start position to		31н(1)	43н(С)	2nd word
3rd word	00н	45н(E)	be inserted (n):		33н(3)	32н(2)	3rd word
			4th character		35н(5)	34н(4)	4th word
	•	ue to IN2			44н(D)	36н(6)	5th word
	123	<u>456</u> '			00н	45н(E)	6th word
High	n-order byte	Low-order b	yte				_
1st word	32н(2)	31н(1)					
2nd word	34н(4)	33н(3)					
3rd word	36н(6)	35н(5)					
4th word	00	Эн					

- (2) The values to be input to \bigcirc and \oslash are string type data within the range from 0 to 255 bytes.
- (3) The value to be input to n is word (signed) type data within the range from 1 to 255.

(The input value must not exceed the number of characters of character string input to 3 .)

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from $\ensuremath{\textcircled{}}$.

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

EN	ENO	ð
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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

Program Example

The program which inserts the character string input to 0 to the specified position in the character string input to 0, and outputs the operation result from 0.

(a) Function without EN/ENO (INSERT)

[Structured ladder]

1		
	INSERT .	
	JNI INSERT	g_string3 = 'AB12345CDEF'
	g_string2 = '12345' JN2 .	
	gjnt1 = 3p	

[ST]

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

(b) Function with EN/ENO (INSERT_E)

[Structured ladder]

1	.																				
	·																·				·
	·	•	t a	oool	11		•	•		·			INS	ER.	T_E	-	_	ह	boc	ol3	•
							s_str				EN JN					EN	U			in g3	
	·					. E	s_str	ing	2 _		JN	2									
	·						. 5.	jn t1	_		Р										
	.																				

[ST]

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

5.7.4 String deletion

				Universal
DELETE(_E)		_E: With EN/ENO	
Structured DELET EN s n1 n2		ST ENO:= DELETE_E (EN, s, n1, n2, d);	functions.	y of the following
Input argument,	EN: s: n1: n2:	Executing condition (TRUE: Execution, FALSE: Stop) Input Number of characters to be deleted Start position to be deleted	:Bit :String :Word (signed) :Word (signed) :Bit	

DELETE(_E)

Function

Operation processing

(1) Deletes the specified number of characters from the specified position in the character string input to \odot , and outputs the remaining character string from \bigcirc .

The number of characters to be deleted is specified by the value input to $\widehat{\mbox{ }}$.

The start position to be deleted in the character string is specified by the value input to 0 .

(Example) Values input to n and 2 are 5



- (2) The value to be input to \odot is 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 0 to 255.
 (The input value must not exceed the number of characters of character string input to s.)
- (4) 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
 [∞].)

Operation result

- Function without EN/ENO
 An operation is executed and the operation value is output from (a).
- (2) Function with EN/ENO The following table shows the executing conditions and operation results.

EN	ENO	٥
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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 \odot , and outputs the remaining character string from \bigcirc .

(a) Function without EN/ENO (DELETE)

[Structured ladder]

1	. .									•	•				•	•		•		•	•						•			
	·				÷	·		·	·			·			C	ELI	ЕТЕ					·			·	·	÷	·	·	
	.	Ę	,st	rin	61	= ',	ABC	DE	F1 2	345	5°		JN						DE	LEI	E		_5.	stri	ng2	= '.	AB1	234	45'	
	.							.5.	jn ti	=	4_		L																	
	.							.5	jn ti	2 = 2	3 _		Р																	
	.																													

[ST]

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

(b) Function with EN/ENO (DELETE_E)

[Structured ladder]

	•	•	•		•	•			•		•	•		•	•	•		•	•		
		•	زع 		1 ·	•	•	•	•	•	EN		DEL	ETE.	E,E	EN	0	•		boc	13
		·	÷	•		. 6	sstı F	ring jint		_	JN								_s.	stri	ng2
						÷		jnt		_	ь Р							•			

[ST]

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

5.7.5 String replacement REPLACE(E) Universa UD REPLACE(_E) E: With EN/ENO indicates any of the following **Structured ladder** ST functions. REPLACE_E REPLACE REPLACE E ΕN ENO ENO:= REPLACE_E (EN, s1, s2, n1, n2, d); s1 d s2 n1 n2 Input argument, EN: Executing condition (TRUE: Execution, FALSE: Stop) :Bit s1: Input :String s2: n1: Number of characters to be replaced :Word (signed) :Word (signed) n2: Start position to be replaced 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 input t

The number of characters to be replaced is specified by the value input to $\widehat{\mbox{ }}$.

The start position to be replaced in the character string is specified by the value input to 0 .



- (2) The values to be input to (s) and (a) 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 (3).)
- (4) The value to be input to 0 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 <a>[i].)

Operation result

(1) Function without EN/ENO

An operation is executed and the operation value is output from ${\ensuremath{\scriptstyle @}}$.

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(b)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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 with the character string input to , and outputs the operation result from .

(a) Function without EN/ENO (REPLACE)

[Structured ladder]

1	.																														
	·															·		·			·				·						
	·		·	·		÷	·	·	·	·				RI	EPL	ACI	E					÷	·	·	·			·		·	•
	·			ng1					_		J١	41						F	REPI	LAC	E		_5_	stri	ng3	= '.	ABC	12:	345	•	
	·	Б.	stri	ng2	= 'I	01 2	345	67	в. —		J١	42																			
											L.																				
	·		·		·	.5	jn t	2 =	3 _		Р											·	·		·	·	·	÷	·	·	•
	·				·					•					÷	÷		·			·		·	·			·				

[ST]

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

(b) Function with EN/ENO (REPLACE_E)

[Structured ladder]

1																							
		·		·	·	·	·	·	·	·	·		·	·	·	·	·	·	•	•	·	·	•
		•	•	¢,	000	11 · 	•	•	•	•	•	EN		RE	PLA	VCE.	JE.	EN		÷	Б.	boo	13
								str.			_	JN						EN			<u>5</u> .	stri	ng3
		•	·	·	·	·	. E	str.			_	JN	12							•	·	·	•
		·	·	·	·	·	·		jnti 		_	Т								1	·	·	·
		•		•	·		•	. 5	jn ti		_	Р								÷	•	·	•
	- 1	•			•		•	•		·			•			·			·		•	·	

[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



Grant Function

5-172

Operation processing

- Performs addition ((s) + (s)) on time type data input to (s) and (s), and outputs the operation result from (a) 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 ${\,\rm (d)}$.

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

EN	ENO	(b)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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

Program Example

The program which performs addition ($(+ \otimes)$) on time type data input to $(+ \otimes)$ and $(+ \otimes)$, and outputs the operation result from (-) in time type.

(a) Function without EN/ENO (ADD_TIME)

[Structured ladder]

1																						
	•	•		•	•			•			•			•	•		•	•		•	•	•
	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	•	·	·	·	·	·
											·				ти	ME						
								e_ti	me1	_	_	IN	1							_5_	time	≥3
	÷	•	•	:	•	:				_	_	JN								_5_	time	
	•	•	:			•		€_tii 6_tii		_	_	Л Л								_5_	time	≥3

[ST]

g_time3:= ADD_TIME(g_time1, g_time2);

(b) Function with EN/ENO (ADD_TIME_E)

[Structured ladder]

		·	÷	·			·	·		·	·		·				·			·	÷	÷	
			÷	٤J	bool	l1 ·	·	÷	÷	·	·			AD	D_T	IME	E			.	÷	÷	
	-			-	[ŀ-						EN	1					EN	ю		_5.	bod	ol3
				÷	· .	۰.			ime 1			JN	11								_5.	tim	e3
								€_ti	ime:	2 _		JN	12										
																				۰.			

[ST]

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



Grant Function

Operation processing

Performs subtraction (() -() on time type data input to () and (), 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	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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

Program Example

The program which performs subtraction ($\[mathbf{shift} - \[mathbf{shift}]$) on time type data input to $\[mathbf{shift}$ and $\[mathbf{shift}$, and outputs the operation result from () in time type.

(a) Function without EN/ENO (SUB_TIME)

[Structured ladder]

1	.																				
	·	·		÷			·	·		·				·		÷	·	•	·	·	
	·				·		·	·	·	·	·			su	B_TI	ME		·	·	·	
	·				·		÷		ime			J١	11					⊢	_5.	tim	e3
	·		·		÷	·	÷	€_t	ime)	2 _		J١	42					· .	÷	·	
	1																				

[ST]

g_time3:= SUB_TIME(g_time1, g_time2);

(b) Function with EN/ENO (SUB_TIME_E)

[Structured ladder]

1	•		•		:	•	•	:	•	•	:	•	•	:	•		:	•	•	:	•	•
			tء -	000	1 · 	•	•	•	•	•	EN		su	IB_T	IME	JE.	EN	ю	•		boo	
	•		:	:		•		ime ime			JN JN									_5.	tim	e3

[ST]

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

5.8.3 Multiplication

Universa UD MUL_TIME(_E) _E: With EN/ENO indicates any of the following functions. **Structured ladder** ST MUL_TIME MUL_TIME_E MUL_TIME_E ΕN ENO ENO:= MUL_TIME_E (EN, s1, s2, d); d s1 s2 Input argument, EN: Executing condition (TRUE: Execution, FALSE: Stop) :Bit s1: :Time Input s2: :ANY_NUM Input ENO: Output status (TRUE: Normal, FALSE: Error) Output argument, :Bit Output :Time d:

MUL_TIME(_E)

Grant Function

Operation processing

- Performs multiplication ((s) × (s)) on time type data input to (s) and (s), and outputs the operation result from (d) in time type.
- (2) The value to be input to s is time type data.

The value to be input to 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 $\, \mathbb{d}$.

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

EN	ENO	٥
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE ^{*1}	Undefined value

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

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

Program Example

The program which performs multiplication ($\mathfrak{T} \times \mathfrak{D}$) on time type data input to \mathfrak{T} and \mathfrak{D} , and outputs the operation result from \mathfrak{D} in time type.

(a) Function without EN/ENO (MUL_TIME)

[Structured ladder]

1	·							·												
	·	•	•	•			•			•		1	•		•		·	·		•
	·	·	•	·		·			·	·			MUL	TI	ME		·	·	tim	
	·	•	•	·	•	·	€_ti			_	JNI							_6.	um	22
	·	·		·		÷	. 5.	jn t1	-		JN	2					·	·		•
	·	·	·	·	÷				·	·		÷	·		·	·	·	·		

[ST]

g_time2:= MUL_TIME(g_time1, g_int1);

(b) Function with EN/ENO (MUL_TIME_E)

[Structured ladder]

1																						
						•			•			•			•			•			•	
	•	•	t a	lood	ŀ L	•	•	•	•	•	EN		ML	JL_T	IME	JE.	EN		•	5.	boc	513
								ime1		_	JN						EN	0			tim.	
		·	·	÷	·	`	.5	jn ti	_	_	JN	12							·	÷	·	·
	· ·	·	·		·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·

[ST]

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

5-180

5.8.4 Division DIV_TIME(_E) Universa UD DIV_TIME(_E) E: With EN/ENO indicates any of the following functions. **Structured ladder** ST DIV_TIME DIV_TIME_E DIV_TIME_E ΕN ENO ENO:= DIV_TIME_E (EN, s1, s2, d); d s1 s2 Input argument, EN: Executing condition (TRUE: Execution, FALSE: Stop) :Bit s1: :Time Input :ANY_NUM s2: Input Output status (TRUE: Normal, FALSE: Error) Output argument, ENO: :Bit Output :Time d:

☆ Function

Operation processing

- Performs division ((() ÷ (2)) on time type data input to (() and ((2)), and outputs the quotient of the operation result from (() in time type.
- (2) The value to be input to $\, \mathrm{st}\,$ is time type data.

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

(The value to be input to $\circledast\;$ 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	٥
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 (a) is undefined. In this case, create a program so that the data output from (a) is not used.
No operation error occurs in the execution of the $DIV_TIME(_E)$ function.

Program Example

The program which performs division ($(\mathfrak{S} \div \mathfrak{D})$) on time type data input to (\mathfrak{S}) and (\mathfrak{D}) , and outputs the quotient of the operation result from () in time type.

(a) Function without EN/ENO (DIV_TIME)

[Structured ladder]

1	.																					
	·	·	•	·	·	÷	·	·	·		·		·	•	·	·	·	·	·		·	
	·	·	•	·	·		·	·	•	·	·			DIV	TIN.	ΛE			÷	·	·	÷
	·	·	÷	·	·	÷	÷		me1		-	JN	1							_5_	time	2
		:	:	:	:	:	•		me1 jnt1		_	л Л								_5_	time	2

[ST]

g_time2:= DIV_TIME(g_time1, g_int1);

(b) Function with EN/ENO (DIV_TIME_E)

[Structured ladder]

1																					
					•	•			·	·	·	·		•	·	÷		·	·	·	·
		6	; bod -	511 · 		•	•		•	EN		DI	V_TI	IME	JE	EN	10	·		boc	513
			٠.	٠.			ime			JN									_s.	tim	e2
						.5	jnt	1_	_	JN	12							·	·	·	

[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) UD SR(E)_E: With EN/ENO indicates any of the following functions. Structured ladder ST SR SR E SR_E ΕN ENO ENO:= SR_E (EN, s1, s2, d); s1 d s2 :Bit Input argument, EN: Executing condition (TRUE: Execution, FALSE: Stop) s1: :Bit Input s2: ENO: Output status (TRUE: Normal, FALSE: Error or stop) :Bit Output argument, d: Output :Bit

Grant Function

Operation processing

Sets (d) when (s) is turned ON, and resets (d) when (e) is turned ON while (s) is OFF.

0 is not reset even when 0 is turned ON while 0 is ON.

(1) Function without EN/ENO

An operation is executed and the operation value is output from $\, \mathbb{d} \, .$

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(b)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value ^{*1}

*1 The previous operation output value is held at (a) even if EN is FALSE.

PPLICATION **C**

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 1 and holds the output, and resets the value of 1 only when bit type data input to 2 is 1 and the data input to 3 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]

1		
	X0 SR_E Y10 Image: bool1 EN EN0 Y10 Image: bool2 _S1 Q1 _g.bool2	 0 0

[ST]

SR_E_Instance (g_bool1, g_bool2, g_bool3);

5.9.2 Standard bistable function blocks (Reset-dominant)



Grant Function

Operation processing

Sets when is turned ON, and resets when is turned ON.

(d) is not set even when (s) is turned ON while (a) is ON.

(1) Function without EN/ENO

An operation is executed and the operation value is output from ${\ensuremath{\, \mathrm{d}}}$.

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	٩
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value ^{*1}

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

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

Program Example

The program which outputs bit type data input to a from a and holds the output, and resets forcibly the value of a when bit type data input to a 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]

1	· · · · · · · · · Instance · · · · ·
	RSE ····
	EN ENO Y10 · · ·
	····g_bool1 — _S Q1 — g_bool3
	····g_bool2 — · · · · · ·

[ST]

RS_E_Instance (g_bool1, g_bool2, g_bool3);

PPLICATION UNCTIONS

5.10 Standard Edge Detection Function Blocks

5.10.1 Rising edge detector



Grunction

Operation processing

Turns ON for one scan when is turned ON.

(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	٩
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value ^{*1}

*1 The previous operation output value is held at (a) 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 ${}_{\textcircled{}}$ for one scan when bit type data input to ${}_{\textcircled{}}$ 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)		_E: With EN/ENO
Structured F TRI EN s		functions. F_TRIG F_TRIG_E
Input argument, Output argument,	EN: Executing condition (TRUE: Execution, FALSE: s: Input ENO: Output status (TRUE: Normal, FALSE: Error or d: Output	:Bit

F_TRIG(_E)

Universa

Grant 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 ${\ensuremath{\, \mathrm{d}}}$.

(2) Function with EN/ENO

The following table shows the executing conditions and operation results.

EN	ENO	(d)
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE	Previous output value ^{*1}

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

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

Program Example

The program which turns ON ${}_{\textcircled{}}$ for one scan when bit type data input to ${}_{\textcircled{}}$ is turned from ON to OFF.

(a) Function without EN/ENO (R_TRIG)

[Structured ladder]

1	:		•	•	•	•			•			•	• h	nsta	nce	•	•	•	•	•	•
		•	•		•	•	ولع	ool1		•	_01	_ĸ	I	F_TF	RIG			Q	•	 boc	12
	·			·	·		·	·								•			·		·

[ST]

F_TRIG_Inst (g_bool1, g_bool2);

(b) Function with EN/ENO (F_TRIG_E)

[Structured ladder]

1																					
				•			•	÷		·	·			· Instand	ce ·	·	۰.		·		
		·	·	•	X0	È	•	•	•	•	•			F_TRIG	JE			•	. M	0	•
	T			- ·				۶Þ	ooli	_	_	EN _CL				EN	0 Q		_	boo	12
		·	·	·	·	·	`	·	·	·	·	·	·		·	·	·	·	·	·	·

[ST]

F_TRIG_E_Inst (g_bool1, g_bool2);

5.11 Standard Counter Function Blocks

5.11.1 Up counter



Grant Function

Operation processing

Counts $\textcircled{}_{2}$ when $\textcircled{}_{3}$ is turned ON.

When the count value reaches the value input to , turns ON.

When 0 is turned ON, 0 turns OFF and count value 0 is reset.

(1) Function without EN/ENO

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

(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 \bigcirc and \oslash even if EN is FALSE.

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 S is turned from OFF to ON, and outputs the count value from D.

(a) Function without EN/ENO (CTU)

[Structured ladder]

	•	·	·		·	·		· · · · · Instance · · · · ·	
	•	·	•	•	·	·	•	сти сти	
	•		·		·			spool1 CUspool3	
	•							s_bool2 RESET OVs_int2 .	
	•	·	·		·	÷		.sjnt1 PV	
	•	÷	÷		÷		÷		

[ST]

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

(b) Function with EN/ENO (CTU_E)

[Structured ladder]

	·	·	·	·	·	÷	·	·	·	·	·	Inst	anc	e	·	·	÷	·	·
	·	•	-1	0 110	È	·	·	·	·	·			UJE		_	•	М	11	:
							٤Þ	ool1	_	_	EN			EN	u Q		_s.	boo	
						5		ool2	_	_	RE	SET		C	v		_5.	jn t2	
	·	·			·	·	. E.	jn t1	_	_	P٧	(.		·	·
	·			•	·		·	·			·		•	·	·			·	÷

[ST]

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

CTD(_E)

5.11.2 Down counter

		Universal Perform
CTD(_E)		_E: With EN/ENO
Structured EN s1 s2 n		indicates any of the following functions. CTD CTD_E
Input argument,	EN: Executing condition (TRUE: Execution, FALSE: s1: Count signal input s2: Count reset n: Count start value	Stop) :Bit :Bit :Bit :Bit :Word (signed)
Output argument,	ENO: Output status (TRUE: Normal, FALSE: Error or d1: Count match output d2: Count value	

Grant Function

Operation processing

Counts down (-1) when is turned ON.

n sets the initial value for subtraction.

 \bigcirc turns ON when count value reaches 0.

When $\circledast\,$ is turned ON, $\oplus\,$ turns OFF and initial value for subtraction $\oplus\,$ is set for count value

d2 ·

CTD(_E)

5

TION NS

(1) Function without EN/ENO

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

(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 and even if EN is FALSE.

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 *s* is turned from OFF to ON, and turns ON *d* when the value of *d* reaches 0.

(a) Function without EN/ENO (CTD)

[Structured ladder]

1		•	•	•	•		•	•		•	•	 [.] Insta	 nce [.]		•	•	
	·	•	·		·	·		•	·	•	·	ОТ	D	•	•	·	·
	.							£"po	pol1	_	_	CD	Q		_5_	boo	13
	.							£"po			_	LOAD	οv		_5_	in t2	
	.						·	.Ej	in t1	_	_	PV					
	·	•	·	·	·	·	·	•	•	·	•	• •	• •		•	·	·

[ST]

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

(b) Function with EN/ENO (CTD_E)

[Structured ladder]

1		
	· · · · · · · · · · · Instance	
	· · · Mt0 · · · · · · CTD_E	 M11 .
	LOAD OV	g_int2 .

[ST]

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



5.11.3 Up/Down counter

		Universal High Universal Deformance
CTUD(_E)		_E: With EN/ENO
Structured EN s1 s2 s3 s4 n		indicates any of the following functions. CTUD CTUD_E
Input argument, Output argument,	EN: Executing condition (TRUE: Execution, FALSE: Stores 1: s1: Count-up signal input s2: Count-down signal input s3: Count-up reset s4: Count-down reset n: Maximum count value ENO: Output status (TRUE: Normal, FALSE: Error or stored1:	:Bit :Bit :Bit :Bit :Word (signed)
	d1:Count-up match outputd2:Count-down match outputd3:Current count value	:Bit :Bit :Word (signed)

CTUD(_E)

Grant Function

Operation processing

Counts (+1) $_{(3)}$ when $_{(3)}$ is turned ON, and counts down (-1) $_{(3)}$ when $_{(2)}$ is turned ON.

- n sets the maximum value of counter.
- $_{\tiny{(1)}}$ turns ON when $_{\tiny{(3)}}$ reaches 0.
- $_{(\mathrm{cl})}\,$ turns ON when $_{(\mathrm{cl})}\,$ reaches the maximum value $\,\mathrm{\bar{n}}\,.$

The value of n is set to $\underset{\textbf{(3)}}{\textbf{(3)}}$ when A is turned ON.

(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	@,@,@
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.

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 S is turned from OFF to ON, and turns ON O when the value of O reaches the value set at O. Simultaneously, it counts the number of times that bit type data input to O is turned from OFF to ON, and turns ON O when the value of O reaches 0.

(a) Function without EN/ENO (CTD)

[Structured ladder]

		•			·	· · · · Instance ·	
	•		·	·		стир стир	
						s_bool1 CU QUs_bool5	į.
						s_bool2 CD QDs_bool6	į.
						s_bool3 RESET	
						s_bool4 LOAD	
						.sjnt1 PV	
						· · · · · · · · · · · · · · ·	
							CTUD

[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]

.																			
·				·					·	÷	۰I	nsta	nce		•	·	·	·	·
·	•		MO JI	È	•	•	•	·	÷	EN		сти	D,E	ENI		•	м	10	•
			Ϊ.			٤Þ	ool1	_						EN Q	υ		_5_	boa	15
.							ool2			OD					D		_	boa	
·				·			ool3	_	_	RE	SE	т		C	N		_5_	in t2	
·		•	•	·	·		ool4 jn t1	_	_	LO		>				·	·	·	·
·		•	•		·	. 6.	Jinti	_	_	P٧	(·	•	·	•

[ST]

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

COUNTER_FB_M

5.11.4 Counter function blocks



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

COUNTER_FB_M

			indicates the following function.
Structured s1 s2 s3		ST COUNTER_FB_M (s1, s2, s3, d1, d2) 	
Input argument, Output argument,	s1: s2: s3: d1: d2:	Executing condition (TRUE: Execution, FALSE: Stop) Counter setting value Counter initial value Counter current value Output	:Bit :Word (signed) :Word (signed) :ANY16 :Bit

Grant Function

Operation processing

Counts the detected rising edge (from OFF to ON) of (a). It is not counted when (b) stays ON. The count starts from the value input to (c) and when the count value reaches the value input to (c), (c) turns ON. The current value is stored in (c).

5-203

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 \mathfrak{S} is turned from OFF to ON, and outputs the count value from \mathfrak{A} .

[Structured ladder]

2	
	COUNTER FB_M Var M0-Coil ValueOutVar D10
	· · · 10— Preset Status Var_M10 · · · · · · · · · · · · · · · · · · ·

[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



Grant Function

Operation processing

Turns ON (a) for the duration set to (n) after (s) is turned ON. The duration (elapsed time) during which (a) stays ON is set to (a).

When the elapsed time reaches the preset time, d) turns OFF.

The elapsed time is not reset even when turns OFF. It is reset and turns ON again when

(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 ${}_{\tiny{(1)}}$ and ${}_{\tiny{(2)}}$.

(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 (d) and (d) even if EN is FALSE.

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

Program Example

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

(a) Function without EN/ENO (TP)

[Structured ladder]

1	.																
	·				·	·	·	·		·	·In	stan	ce ·		·		·
	·	• •		·		·	·	·	·	•		TP		·	·	·	·
	·						e٦p	ool1	_	_	IN		Q	L	_5_	bool	2
	·				·		Τŧ	‡ 1 O≤	- ⁻	_	PT		ET	Ŀ	_5_	time	1
	·	• •	•	·	·	·	·	·	·	·	·	• •	•	·	·	·	·

[ST]

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

(b) Function with EN/ENO (TP_E)

[Structured ladder]

	.																	
	·	·	·	•	·	·	·	·	·	·	. In	sta	ince	э.		·	·	·
	·	·	•	M0 ⊫	i.	·	÷	·	·			TΡ	-		·	M	11 O	·
				L	ŀ-		в b	ool	1		EN		EN				boc	12
	:		÷		÷	÷		#1 O			IN PT			Q ET		_	tim	
	.												÷		· .			



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

(PPLICATION UNCTIONS

5.12.2 On delay timer

TON(_E), TON_HIGH(_E)



TON(_E), T(DN_HI	GH(_E)	E: With EN/I	ENO
Structure TON_HI EN s n		ST ENO:= TON_HIGH_E (EN, s, n, d1, d2);	functions. TON TON_HIGH	dicates any of the following TON_E TON_HIGH_E
Input argument, Output argument,	EN: s: n: ENO: d1: d2:	Executing condition (TRUE: Execution, FALSE: Stop) Input Delay time setting value Output status (TRUE: Normal, FALSE: Error or stop) Output Elapsed time	:Bit :Bit :Time :Bit :Bit :Time	

Grant Function

Operation processing

Turns ON (a) when (s) is turned ON after the elapse of the time set to (n). Elapsed delay time until (d) is turned ON is set to (e).

When (s) is turned OFF, (d) 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.

(1) Function without EN/ENO

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

(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 \bigcirc and \bigcirc even if EN is FALSE.

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

Program Example

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

(a) Function without EN/ENO (TON)

[Structured ladder]

1																			
												· Ir	nsta	ance	э.				
													то			.			
		·	·	·		·	÷	£"p			_	IN			Q			poo	
		·	·	·	·	·	÷	T#	10:	5 _	_	PT		E	ЕΤ		_5.	time	1
																•			

[ST]

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

(b) Function with EN/ENO (TON_E)

[Structured ladder]

	1																			
		•	·	•	·	·	·	·	•	•	·	•	·	•		·	•	•	•	·
		•	·	·	·	·	·	•	·	·	•	•	Inst	ance	Э	•	·	·	·	·
		•	·	•	мо	ċ.	·	·	·	·	·			NJE			·	м	10	·
	⊢	-	-	-		ŀ-	-				_	EN	1	E	ENC	D	_	_		
		·	·	·		·	·	E D	ool	۱ <u>–</u>		IN			- 0	2			boc	
		·	·			·		T	#10	s		PT	-		E	Т		_5.	tim	≥1
		•		·		÷	·				·					÷	·	÷		÷

[ST]

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

5-210



Grant Function

Operation processing

Turns ON \bigcirc when \bigcirc is turned ON.

Turns OFF (d) when (s) is turned from ON to OFF after the elapse of the time set to (n). Elapsed

time until 0 is turned OFF is set to 0.

When (s) is turned ON again, (d) 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.

5-211

(1) Function without EN/ENO

An operation is executed and the operation value is output from ${}_{\tiny{\tiny{(1)}}}$ and ${}_{\tiny{(2)}}$.

(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 and even if EN is FALSE.

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

Program Example

The program which turns ON bit type data of when bit type data input to is turned ON, and turns OFF 10 seconds after is turned OFF.

(a) Function without EN/ENO (TOF)

[Structured ladder]

1	·															
	·	·	÷	÷	÷	·	÷	· ·	·	۰.	· Instanc	æ.	·	·		·
	·	·	·	•	÷	·	÷	• •	·	•	TOF		·	۰.	۰.	÷
	·	·	·		÷	·	÷	g_bool1		_	IN	Q			lood	
	·	·	·	÷		·	·	T#1 0:	5	_	PT	ET		_5.1	time	1
	·	·	·	•	·	·	·	• •	·	•			·	·	·	•

[ST]

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

(b) Function with EN/ENO (TOF_E)

[Structured ladder]

1	.																	
	·	•	·	·		•	·	·	•	·	•	Insta	nce	•	•	·	·	·
	·	·	•	мо	÷	·	·	·	·	·		TOF	E		•	•		
	-		-		ŀ-						EN		EN	ю		_	10	·
	·			•			e ja	ool	1_		IN			Q		_5.	boc	42
	·	·	·	÷			Τŧ	#10	s	_	PT		E	т		_5.	tim	e1
	·																	



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

5.12.4 Timer function blocks

TIMER_10_FB_M



*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

Structure	d ladder	ST	functions.	any of the following
		TIMER_10_FB_M (s1, s2, s3, d1, d2)	TIMER_10_FB_M TIMER_HIGH_FB_M TIMER_CONT_FB_M	TIMER_100_FB_M TIMER_LOW_FB_M TIMER_CONTHFB_
Input argument,	s1:	Executing condition (TRUE: Execution, FALSE: Stop)	:Bit	
	s2:	Timer setting value	:Word (signed)	
	s3:	Timer initial value	:Word (signed)	
Output argument,	s3: d1:	Timer initial value Timer current value	:Word (signed) :ANY16	

Grant Function

Operation processing

- (1) TIMER_10_FB_M

 - (b) When the executing condition of (a) turns OFF, the current value is set to the value input to (a), and (a) turns OFF.

- (2) TIMER_100_FB_M
 - (a) Starts measuring the current value when the executing condition of si turns ON.
 Starts measuring from the value input to si × 100ms, and when the measuring value reaches to the value input to si × 100ms, de turns ON.
 The current value is output from d .
- (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 is turns ON. Starts measuring from the value input to is × 0.1 to 100ms, and when the measuring value reaches to the value input to is × 0.1 to 100ms, is turns ON. The current value is output from in .
 - (b) When the executing condition of (s) turns OFF, the current value is set to the value input to (s), and (a) 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.

(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 in turns ON.

Starts measuring from the value input to $\circledast \times 1$ to 1000ms, and when the measuring

value reaches to the value input to $\otimes \times 1$ to 1000ms, \otimes turns ON.

This setting is set in the PLC system setting of the PLC parameter.

The current value is output from \bigcirc .

- (b) When the executing condition of (a) turns OFF, the current value is set to the value input to (a), and (a) 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 ⊕ 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 $\odot \times 1$ to 1000ms, and when the count value

reaches to the value input to $\textcircled{} \times$ 1 to 1000ms, turns ON.

The current value is output from d) .

(b) Even when the executing condition of sturns OFF, the ON/OFF statuses of measuring

value (d) and (d) are retained. When the executing condition of (s) 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]

ON		
Var_M0 OFF	15sec	<u>5seç</u>
Value of Var_D10_0	to 150	151 to 200
		ON
Var_M10 OFF		4



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 O turns ON, and when the measuring value reaches to the value input to $\textcircled{O} \times 10$ ms, O turns ON. [Structured ladder]

2	
	TIMER_10_FB_M Var_M0——— Coil ValueOut ——Var_D10
	Var_M0 — Coil ValueOut — Var_D10
	Var_M10
	· · · · · · · · · · · · · · · · · · ·

[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 O turns ON, and when the measuring value reaches to the value input to $\textcircled{O} \times 0.1$ to 100ms, O turns ON.

[Structured ladder]

4	
	TIMER_HIGH_FB_M
	Var_M0 — Coil ValueOut – Var_D10
	Var_M10
	ValueIn

[ST]

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

[Timing chart]



5

(3) TIMER_LOW_FB_M

The program which starts measuring from 0 when the executing condition of S turns ON, and when the measuring value reaches to the value input to $\textcircled{S} \times 1$ to 1000ms, D turns ON.

[Structured ladder]

5		
	TIMER_LOW_FB_M	·
	Var_M0 — Coil ValueOut — Var_D10	
	Var_M10 View Status Var_M10	ŀ.
	· · · · O— ValueIn	·
		·

[ST]

	A loot ()/	MO 1	0 0 1/0-		M40 \.
FIMER_LOW_FB_M	VI IIISL (Vä	al IVIU, I	0, 0, vai	DIU, Var	IVIIU),

[Timing chart]

ON		
Var_M0 OFF	10	
Value of Var_D10 1	900ms	
Var_M10 OFF	Ľ.	

(4) TIMER_CONT_FB_M

The program which measures the time during G is ON, and when the measuring value reaches to the value input to $\textcircled{O} \times 1$ to 1000ms, O turns ON. [Structured ladder]

1	· · · · · · · · · · Instance ·	
		B_M lueOutVar_D10 StatusVar_M10

[ST]

TIMER_CONT	FB M	Inst (Var	M0.	10, (0, Var	D10,	Var	M10):

[Timing chart]

	ON				
Var_M0 OFF		15s	ec b	<u>5sec</u>	_
Value of Var_D10	0 1	to	150	151 to 200	
Var_M10 OFF					


Appendix 1	Correspondence between Generic Data Types and Devices App-2
Appendix 2	Correspondence between Devices and Addresses App-6

Appendix 1 Correspondence between Generic Data Types and Devices

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

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

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

*1: Can be used for digit specification.

*2 : Can be used for bit specification.

	Generic data type											
					ANY							
			A	NY_SIMPL							ANY	
	ANY_BIT			ANY_	-	DEAL						
		Double		_INT	ANY_	REAL			Array	Structure		
Bit	Word (unsigned)/ 16-bit string	word (unsigned)/ 32-bit string	Word (signed)	Double word (signed)	Single- precision real	Double- precision real	Time	String	Анау	otructure	ANY16	ANY32
0	O*1	O*1	O*1	O*1	×	×	×	×	×	×	○*1	O*1
0	O*1	O*1	O*1	O*1	×	×	×	×	×	×	O*1	O*1
0	O*1	O*1	O*1	○*1	×	×	×	×	×	×	○*1	O*1
0	O*1	O*1	O*1	O*1	×	×	×	×	×	×	○*1	○*1
0	O*1	O*1	O*1	* 1	×	×	×	×	×	×	O*1	O*1
0	×	×	×	×	×	×	×	×	×	×	×	×
\bigcirc	O*1	O*1	O*1	O*1	×	×	×	×	×	×	○*1	O*1
0	O*1	O*1	O*1	O*1	×	×	×	×	×	×	○*1	O*1
0	O*1	O*1	O*1	O*1	×	×	Х	×	×	×	○*1	O*1
0	O*1	O*1	O*1	O*1	×	×	×	×	×	×	○*1	O*1
0	O*1	O*1	O*1	O*1	×	×	×	×	×	×	○*1	O*1
0	O*1	O*1	O*1	O*1	×	×	×	×	×	×	O*1	O*1
0	O*1	O*1	O*1	O*1	×	×	×	×	×	×	O*1	O*1
0	O*1	O*1	O*1	O*1	×	×	×	×	×	×	O*1	O*1
0	O*1	O*1	O*1	O*1	×	×	×	×	×	×	O*1	O*1
×	0	×	0	×	×	×	×	×	×	×	0	×
×	0	×	0	×	×	×	×	×	×	×	0	×
×	0	×	0	×	×	×	×	×	Х	×	0	×
○*2	! 0	×	0	×	×	×	×	×	Х	×	0	×
○*2	? O	×	0	×	×	×	×	×	×	×	0	×
○*2	2 0	×	0	×	×	×	×	×	×	×	\bigcirc	×
-	-	-	-	-	_	-	_			-	_	
-	-	-	-	-	-	-	-	-	-	-	-	-
0	O*1	O*1	O*1	O*1	×	×	×	×	×	×	○*1	O*1
-	-	-	-	-	×	×	×	×	Х	×		-
○*2	2 0	×	0	×	×	×	×	×	×	×	\bigcirc	×

Device						
Classification	Туре	Device name	Device symbol			
		Link input	Jn\X			
	Bit device	Link output	Jn\Y			
Link direct device	Dir device	Link relay	Jn\B			
		Link special relay	Jn\SB			
	Word device	Link register	Jn\W			
	word device	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	Р			
FUILLEI	-	Interrupt pointer	1			
Constant			К, Н			
Constant	-	-	E			
Character string constant	-	_	'Character string' or "Character string"			

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

							neric data	type					
Ĩ						ANY							
				AN	NY_SIMPL							ANY	
		ANY_BIT		ANY_NUM									
		1	Double	ANY	_INT	ANY_	REAL			Array	Structure		1
	Bit	Word (unsigned)/ 16-bit string	word (unsigned)/ 32-bit string	Word (signed)	Double word (signed)	Single- precision real	Double- precision real	Time	String	Alldy	Structure	ANY16	ANY32
Ī	\bigcirc	O*1	O*1	○*1	○*1	×	×	×	×	×	×	○*1	O*1
Ĩ	0	○*1	O*1	O*1	O*1	×	×	×	×	×	×	O*1	O*1
Ī	0	O*1	O*1	O*1	O*1	×	×	×	×	×	×	O*1	O*1
Ĩ	\bigcirc	O*1	O*1	O*1	O*1	×	×	×	×	×	×	O*1	O*1
Ĩ	○*2	0	×	0	×	×	×	×	×	×	×	0	×
Ĩ	○*2	0	×	0	×	×	×	×	×	×	×	\bigcirc	×
	○*2	0	×	0	×	×	×	×	×	×	×	0	×
Ĩ	×	0	×	0	×	×	×	×	×	×	×	0	×
Ĩ	○*2	0	×	0	×	×	×	×	×	×	×	\bigcirc	×
Ĩ	_	-	-	-	_	-	-	_	-	-	-	-	-
Ī	-	-	—	-	-	-	—	-	-	-	-	_	-
Ī	-	-	-	-	-	-	-	-	-	-	-	-	-
Ī	0	0	0	0	0	0	×	×	×	×	×	0	0
Ī	×	×	×	×	×	0	0	×	×	×	×	×	×
Ĩ	×	×	×	×	×	×	×	×	0	×	×	×	×

Appendix 2 Correspondence between Devices and Addresses

The following table shows the correspondence between devices and addresses.

Device		Expres	ssing method	Example of correspondence between device and address		
			Device	Address	Device	Address
	Input	Х	Xn	%IXn	X7FF	%IX2047
	Output	Y	Yn	%QXn	Y7FF	%QX2047
	Internal relay	М	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
D	Direct access output	DY	DYn	%QX1.n	DY7FF	%QX1.2047
	Contact	TS	Tn	%MX3.n	TS511	%MX3.511
er	Coil	TC	Tn	%MX5.n	TC511	%MX5.511
Timer				%MW3.n	TN511	%MW3.511
	Current value	TN	Tn	%MD3.n	T511	%MD3.511
	Contact	CS	Cn	%MX4.n	CS511	%MX4.511
nter	Coil	CC	Cn	%MX6.n	CC511	%MX6.511
Counter	Ourseturation	011	0	%MW4.n	CN511	%MW4.511
0	Current value	CN	Cn	%MD4.n	C511	%MD4.511
er	Contact	SS	STn	%MX13.n	STS511	%MX13.511
Retentive timer	Coil	SC	STn	%MX15.n	STC511	%MX15.511
ıtive	Current value	SN	STn	%MW13.n	STN511	%MW13.511
eter				%MD13.n	ST511	%MD13.511
Ř						
	Data register	D	Dn	%MW0.n	D11135	%MW0.11135
				%MD0.n %MW10.n		%MD0.11135 %MW10.1023
	Special register	SD	SDn	%MW10.n %MD10.n	SD1023	%MVV10.1023 %MD10.1023
	Function register	FD	FDn	None	FD0	None
	Link relay	B	Bn	%MX1.n	B7FF	%MX1.2047
		SB				
	Link special relay	3B	SBn	%MX11.n %MW1.n	SB3FF	%MX11.1023 %MW1.2047
	Link register	W	Wn	%MD1.n	W7FF	%MVV1.2047 %MD1.2047
				%MW11.n		%MW11.1023
L	ink special register	SW	SWn	%MD11.n	SW3FF	%MD11.1023
Intel	lligent function module			%MW14.x.n		%MW14.0.65535
	device	G	Ux∖Gn	%MD14.x.n	U0\G65535	%MD14.0.65535
File register		_		%MW2.n		%MW2.32767
		R	Rn	%MD2.n	R32767	%MD2.32767
Pointer		Р	Pn	"" (null character)	P299	None
	Interrupt pointer	I	In	None	-	_
	Nesting	N	Nn	None	-	-
		7	7	%MW7.n	70	%MW7.9
	Index register	Z	Zn	%MD7.n	Z9	%MD7.9

Table App. 2-1 Correspondence between devices and addresses

Device		Expre	ssing 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		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	J	Jx\Wn	%MW16.x.1.n	J2\W3FFF	%MW16.2.1.16383		
Link register	J	JX\VVN	%MD16.x.1.n	JZ\W3FFF	%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	J2\SW1FF	%MW16.2.11.511		
Link special register		JX/2001	%MD16.x.11.n	JZ/SVVIFF	%1/1// 10.2.11.511		
Filo register	ZR	ZRn	%MW12.n	ZR32767	%MW12.32767		
File register	ZR		%MD12.n	2032101	%MD12.32767		

Table App. 2-2 Correspondence between devices and addresses

MEMO



1

[A]

• •	
ABS(_E)	5-119
ADD(_E)	5-122
Address	
ADD_TIME(_E)	5-172
AND(_E)	5-143

[B]

BCD_TO_DINT(_E)	5-101
BCD_TO_INT(_E)	5-101
BCD_TO_STR(_E)	5-104
BOOL_TO_DINT(_E)	5-2
BOOL_TO_DWORD(_E)	5-7
BOOL_TO_INT(_E)	5-2
BOOL_TO_STR(_E)	5-5
BOOL_TO_TIME(_E)	5-10
BOOL_TO_WORD(_E)	5-7

[C]

CONCAT(_E) 5-160
Correspondence between Devices and
Addresses App-6
Correspondence between Generic Data Types and
Devices App-2
CTD(_E)
CTU(_E)
CTUD(_E) 5-200,5-203

[D]

Data Types	
DELETE(_E)	5-166
Device	
DINT_TO_BCD(_E)	5-39
DINT_TO_BOOL(_E)	5-19
DINT_TO_DWORD(_E)	5-36
DINT_TO_INT(_E)	5-16
DINT_TO_REAL(_E)	5-23,5-26
DINT_TO_STR(_E)	5-29
DINT_TO_TIME(_E)	5-42
DINT_TO_WORD(_E)	5-33
DIV(_E)	5-131
DIV_TIME(_E)	5-181
DWORD_TO_BOOL(_E)	
DWORD_TO_DINT(_E)	5-67
DWORD_TO_INT(_E)	5-67
DWORD_TO_STR(_E)	5-76
DWORD_TO_TIME(_E)	5-79
DWORD_TO_WORD(_E)	5-73

[E]

Elementary data types	3-6
EQ(_E)	5-154
Expressing Methods of Constants	3-10

EXPT(_E) 5-137

F_TRIG(_E)	5-192
FUNCTION TABLES	2-1
Functions of time data types table	2-5
Functions with EN	3-3

[G]

GE(_E)	5-154
Generic data types	3-7
Global labels	3-4
GT(_E)	5-154

[H]

How to Read Function Tables	. 2-2
HOW TO READ FUNCTIONS	. 4-1

[I]

Input Pins Variable Function	
INSERT(_E)	5-163
INT_TO_BCD(_E)	5-39
INT_TO_BOOL(_E)	5-19
INT_TO_DINT(_E)	5-13
INT_TO_DWORD(_E)	5-36
INT_TO_REAL(_E)	5-23,5-26
INT_TO_STR(_E)	5-29
INT_TO_TIME(_E)	5-42
INT_TO_WORD(_E)	5-33

[L]

Label classes	
Labels	
LE(_E)	5-154
Local labels	
LT(_E)	5-154

[M]

MID(_E)	5-157
MOD(_E)	5-134
MOVE(_E)	5-140
MUL(_E)	5-125
MUL_TIME(_E)	5-178
MUX(_E)	5-151

[N]

NE(_E)	. 5-154
NOT(_E)	. 5-143

[0]

OR(_E)5-14	.3
------------	----

[P]

Precautions on assigning a name	. 3-10
Precautions on Programming	. 3-10

[R]

5-45,5-48,5-51,5-54
5-57
5-169
5-187
5-190

[S]

SEL(_E)	5-148
Setting labels	
SR(_E)	5-184
Standard arithmetic functions table	2-4
Standard bistable function blocks table	2-5
Standard bitwise Boolean functions table	2-4
Standard character string functions table	2-5
Standard comparison functions table	2-5
Standard counter function blocks table	2-6
Standard edge detection function blocks table	2-6
Standard functions of one numeric variable table	2-4
Standard selection functions table	2-5
Standard timer function blocks table	2-6
STR_TO_BCD(_E)	5-98
STR_TO_BOOL(_E)	5-82
STR_TO_DINT(_E)	5-85
STR_TO_DWORD(_E)	5-92
STR_TO_INT(_E)	5-85
STR_TO_REAL(_E)	5-88
STR_TO_TIME(_E)	
STR_TO_WORD(_E)	5-92
SUB(_E)	
SUB_TIME(_E)	5-175

[T]

TIME_TO_BOOL(_E)	5-107
TIME_TO_DINT(_E)	5-110
TIME_TO_DWORD(_E)	5-116
TIME_TO_INT(_E)	5-110
TIME_TO_STR(_E)	5-113
TIME_TO_WORD(_E)	
TOF(_E)	
TOF_HIGH(_E)	
TON(_E)	
TON_HIGH(_E)	5-208
TP(_E)	5-205
TP_HIGH(_E)	
Type conversion functions table	

[W]

WORD_TO_BOOL(_E)	5-61
WORD_TO_DINT(_E)	5-64
WORD_TO_DWORD(_E)	5-70

WORD_TO_INT(_E)	5-64
WORD_TO_STR(_E)	5-76
WORD_TO_TIME(_E)	5-79
[X]	

XOR(_E)	43
---------	----

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.

Microsoft, Windows are registered trademarks of Microsoft Corporation in the United States and other countries. Other company names and product names used in this document are trademarks or registered trademarks of respective companies.

QCPU

Structured Programming Manual (Application Functions)

Q-KP-OK-E

MODEL

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

When exported from Japan, this manual does not require application to the Ministry of Economy, Trade and Industry for service transaction permission.