MITSUBISHI

Mitsubishi Programmable Controller

MELSEC Q series MELSEC L series

MELSEC-Q/L Programming Manual

Structured Text

QSERIES

LSERIES

• SAFETY PRECAUTIONS •

(Always read these precautions before use)

Before using the MELSEC-Q series or MELSEC-L series PLC, 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 save the manuals attached to the products carefully to make them accessible when required, and always forward them to the end user.

A - 1

• CONDITIONS OF USE FOR THE PRODUCT •

(1) Mitsubishi programmable controller ("the PRODUCT") shall be used in conditions;
i) where any problem, fault or failure occurring in the PRODUCT, if any, shall not lead to any major or serious accident; and
ii) where the backup and fail-safe function are systematically or automatically provided outside of the PRODUCT for the case of any problem, fault or failure occurring in the PRODUCT.

(2) The PRODUCT has been designed and manufactured for the purpose of being used in general industries.
MITSUBISHI SHALL HAVE NO RESPONSIBILITY OR LIABILITY (INCLUDING, BUT NOT LIMITED TO ANY AND ALL RESPONSIBILITY OR LIABILITY BASED ON CONTRACT, WARRANTY, TORT, PRODUCT LIABILITY) FOR ANY INJURY OR DEATH TO PERSONS OR LOSS OR DAMAGE TO PROPERTY CAUSED BY the PRODUCT THAT ARE OPERATED OR USED IN APPLICATION NOT INTENDED OR EXCLUDED BY INSTRUCTIONS, PRECAUTIONS, OR WARNING CONTAINED IN MITSUBISHI'S USER, INSTRUCTION AND/OR SAFETY MANUALS, TECHNICAL BULLETINS AND GUIDELINES FOR the PRODUCT.
("Prohibited Application")
Prohibited Applications include, but not limited to, the use of the PRODUCT in;
- Nuclear Power Plants and any other power plants operated by Power companies, and/or any other cases in which the public could be affected if any problem or fault occurs in the PRODUCT.
- Railway companies or Public service purposes, and/or any other cases in which establishment of a special quality assurance system is required by the Purchaser or End User.
- Aircraft or Aerospace, Medical applications, Train equipment, transport equipment such as Elevator and Escalator, Incineration and Fuel devices, Vehicles, Manned transportation, Equipment for Recreation and Amusement, and Safety devices, handling of Nuclear or Hazardous Materials or Chemicals, Mining and Drilling, and/or other applications where there is a significant risk of injury to the public or property.

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

REVISIONS

* The manual number is given on the bottom left of the back cover.

Print Date	* Manual Number	Revision
Feb., 2003	SH (NA) 080366E-A	First printing
Jul., 2003	SH (NA) 080366E-B	Correction Section 6.7.1, Section 6.7.2, Section 6.7.3, Section 6.9.1, Section 6.9.2, Section 6.9.3, Section 6.9.4, Section 6.9.5, Section 6.9.6, Chapter 7
Jun., 2004	SH (NA) 080366E-C	Additional models Q12PRHCPU, Q25PRHCPU Correction Abbreviations and Generic Terms in This Manual Section 2.1.1, Section 2.1.3, Chapter 5, Chapter 6, Section 6.1.8, Section 6.1.14, Section 6.8.3, Chapter 7, WARRANTY
Feb., 2006	SH (NA) 080366E-D	Correction Section 6.1.14
May, 2008	SH (NA) 080366E-E	Correction Abbreviations and Generic Terms in This Manual, Section 2.1.1
Oct., 2008	SH (NA) 080366E-F	Additional models Q00UJCPU, Q00UCPU, Q01UCPU, Q10UDHCPU, Q10UDEHCPU, Q20UDHCPU, Q20UDEHCPU Correction Abbreviations and Generic Terms in This Manual, Section 2.1.1, Section 3.2.1, Section 4.2.2
Jan., 2009	SH (NA) 080366E-G	Correction Section 2.1.3, Chapter 5
Jan., 2010	SH (NA) 080366E-H	Additional models L02CPU, L26CPU-BT Addition CONDITIONS OF USE FOR THE PRODUCT, Appendix 2 Correction SAFETY PRECAUTIONS, About Manuals, Abbreviations and Generic Terms in This Manual, Section 1.2, Section 2.1.1, Section 2.1.3, Section 3.1, Section 3.3, Section 3.3.1, Section 3.3.3, Section 4.2.1, Section 4.2.2, Chapter 5, Section 5.16.1 to 5.16.4, Section 5.20.1, Section 5.20.2, Chapter 6, Chapter 7,Appendix 1

Japanese Manual Version SH-080363-I

INTRODUCTION

Thank you for choosing the Mitsubishi MELSOFT series Integrated FA software.
Read this manual and make sure you understand the functions and performance of MELSEC series sequencer thoroughly in advance to ensure correct use.

CONTENTS

About Manuals

The manuals related to this product are shown below.
Refer to the following table when ordering required manuals.

Relevant manuals

Manual Name	Manual Number (Model Code)
GX Developer Version 8 Operating Manual (Startup) Explains the system configuration, installation method and startup method of GX Developer. (Sold separately)	SH-080372E (13JU40)
GX Developer Version 8 Operating Manual Explains the program creation method, printout method, monitoring method, debugging method, etc. using GX Developer.. (Sold separately)	SH-080373E (13JU41)
GX Developer Version 8 Operating Manual (Function Block) Explains the function block creation method, printout method, etc. using GX Developer. (Sold separately)	SH-080376E (13JU44)
GX Developer Version 8 Operating Manual (Structured Text) Explains the structured text (ST) program creation method, printout method, etc. using GX Developer. (Sold separately)	SH-080367E (13JU37)
Structured Text (ST) Programming Guide Book Written for those who will create structured text (ST) programs for the first time. Explains the basic operation methods and functions through sample programs. (Sold separately)	SH-080368E (13JF69)
MELSEC-Q/L Programming Manual (Common Instructions) Explains the methods of using the sequence instructions, basic instructions and application instructions. (Sold separately)	SH-080809ENG (13JW10)

REMARK

The Operating Manuals and Structured Text (ST) Programming Guide Book are included on the CD-ROM of the software package in a PDF file format.
Manuals in printed form are sold separately for single purchase. Order a manual by quoting the manual number (model code) listed in the table above.

A - 12

This Manual ...

Use this manual to perform structured text (hereafter abbreviated to ST) programming with GX Developer. It is suitable for the users who have the knowledge and programming experience of PLC ladder programs and for the users who have the knowledge and programming experience of high-level languages such as the C language.
"CHAPTER 1 OVERVIEW" describes the overview of the ST language, the features of ST programming, and the ST program creation procedure.
"CHAPTER 2 SYSTEM CONFIGURATION" describes the applicable CPUs, ST program specifications, etc.
"CHAPTER 3 HANDLING OF CHARACTERS AND NUMERIC VALUES IN ST PROGRAMS" describes the types and representation methods of data used in ST programs.
"CHAPTER 4 ST PROGRAM EXPRESSIONS" describes the expressions of the operators, control syntaxes, etc. used in ST programs.
"CHAPTER 5 MELSEC FUNCTIONS" and "CHAPTER 6 IEC FUNCTIONS" describe the arguments, return values and description examples of the functions used in ST programs.

Operating Manual ...

The "GX Developer Operating Manual (ST)" consists of in-depth explanations of all menus and menu options used to perform ST programming. Refer to the manual when information on operation details is necessary.
When information on other than ST programming is necessary, refer to the "GX Developer Operating Manual" or "GX Developer Operating Manual (Startup)".

When using the structured text language for the first time ...

Refer to the "First ST", which describes the outline of the ST language, the procedures for creating an ST program using GX Developer and writing it to the PLC CPU, the information necessary for that purpose, and others.

When you already have the knowledge of the ST language and want to start programming immediately ...

Proceed to "CHAPTER 5 MELSEC FUNCTIONS". It describes the necessary items for use of the functions in ST programs. When it is desired to know the data to be used in ST programs, refer to "CHAPTER 3 HANDLING OF CHARACTERS AND NUMERIC VALUES IN ST PROGRAMS". It describes the types and representation methods of the data used in ST programs. When it is desired to use control syntaxes in ST programs, refer to "CHAPTER 4 ST PROGRAM EXPRESSIONS". It describes the formats and description examples of the control syntaxes used in ST programs.

The following explains the symbols and information used in this manual.

Symbol	Description	Example
Point	Gives the section-related knowledge and necessary information.	
Remark	Gives the section-related knowledge and useful information.	REMARK
[ ]	Menu name of menu bar	[Project]

A - 13

Abbreviations and Generic Terms in This Manual

In this manual, the following generic terms and abbreviations are used to represent the GX Developer software package and PLC CPU modules. The module mode name is given when the applicable model name must be pointed out explicitly.

Generic Terms /Abbreviation	Description/Applicable Module
GX Developer	Generic product name of the product types SWnD5C-GPPW-E, SWnD5C-GPPW-EA, SWnD5C-GPPW-EV and SWnD5C-GPPW-EVA. (n denotes the version 8 or later.)
GX Works2	Generic product name of the product types SWnDNC-GXW2-E (n denotes the version).
ST	Abbreviation for structured text.
FB	Abbreviation for function block.
Basic model QCPU	Generic term for Q00JCPU, Q00CPU and Q01CPU of function version B or later.
High Performance model QCPU	Generic term for Q02(H)CPU, Q06CPU, Q12HCPU and Q25HCPU.
Universal model QCPU	Generic term for Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, Q03UDCPU, Q03UDECPU, Q04UDHCPU, Q04UDEHCPU, Q06UDHCPU, Q06UDEHCPU, Q10UDHCPU, Q10UDEHCPU, Q13UDHCPU, Q13UDEHCPU, Q20UDHCPU, Q20UDEHCPU, Q26UDHCPU and Q26UDEHCPU.
Process CPU	Generic term for Q02PHCPU, Q06PHCPU, Q12PHCPU and Q25PHCPU.
Redundant CPU	Generic term for Q12PRHCPU and Q25PRHCPU.
QCPU (Q mode)	Generic term for Q00J, Q00UJ, Q00, Q00U, Q01, Q01U, Q02(H), Q02PH, Q02U, Q03UD, Q03UDE, Q04UDH, Q04UDEH, Q06H, Q06PH, Q06UDH, Q06UDEH, Q10UDH, Q10UDEH, Q12H, Q12PH, Q12PRH, Q13UDH, Q13UDEH, Q20UDH, Q20UDEH, Q25H, Q25PH, Q25PRH, Q26UDH and Q26UDEHCPU.
LCPU	Generic term for L02CPU and L26CPU-BT.

MEMO

A - 15

1 OVERVIEW

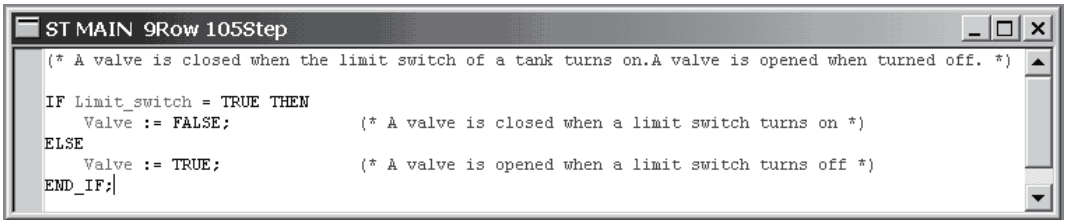
1.1 What Is the ST Language?

The ST language is defined in the International Standard IEC61131-3 that stipulates the logic description system in open controllers.
The ST language supports operators, control syntaxes and functions to permit the following descriptions.

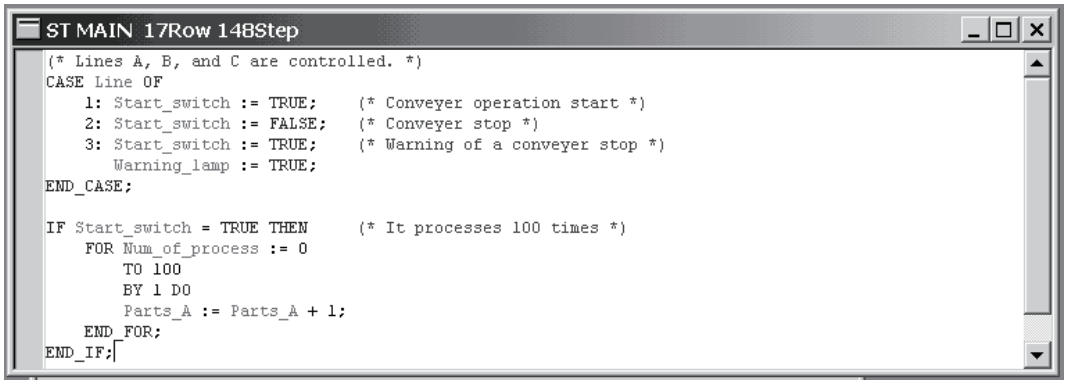
- Control syntaxes such as conditional statement-dependent selective branch and repeated statement-based repetition
- Expressions using operators (*, /, +, -, <, >, =, etc.)
- Call of user-defined function blocks (FB)
- Call of functions (MELSEC functions, IEC functions)
- Description of comments

The main features of the ST language are as described below.

(1) Free description in text format
The ST language is described in text format of alphanumeric characters, comments and labels.



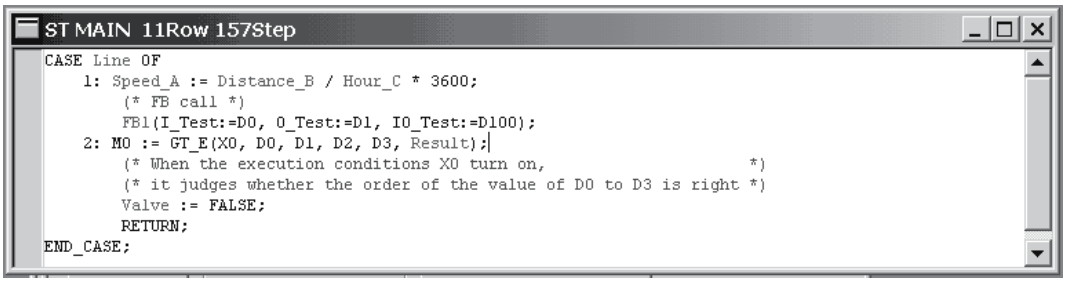
(2) Programming on the same level as those of the C and other high-level languages
Like the high-level languages such as C, the ST language can describe control with control syntaxes such as conditional statement-dependent selective branches and repeated statement-based repetitions. Hence, easy-to-read programs can be written briefly.



1 - 1

(3) Ease of describing operation processing

Capable of briefly describing easy-to-read operation processing that is difficult to describe in lists or ladders, the ST language has a high level of program readability and is suitable for the fields where complex arithmetic operations, comparison operations, etc. are performed.



1 - 2

1.2 Features of ST Program in MELSEC-Q/L Series

ST programs are described in ST language.
Using GX Developer to perform ST programming enables efficient programming to be performed in excellent operation environment. The following provides the main features of ST programs in the MELSEC-Q/L series.

(1) Design efficiency improved by defining processing as parts
With often used processing defined as parts in the form of function blocks (FB) in ST language, they can be used in necessary areas of each program. This not only enhances the efficiency of program development but also reduces program mistakes, improving program quality.
For more information, refer to the "GX Developer Operating Manual (Function Block)" given in Relevant Manuals.

(2) Restoration of ST program read from PLC
In the MELSEC-Q/L series, the created ST program is written to the PLC and executed, and can be read from the PLC and then restored to enable editing in the ST language format.

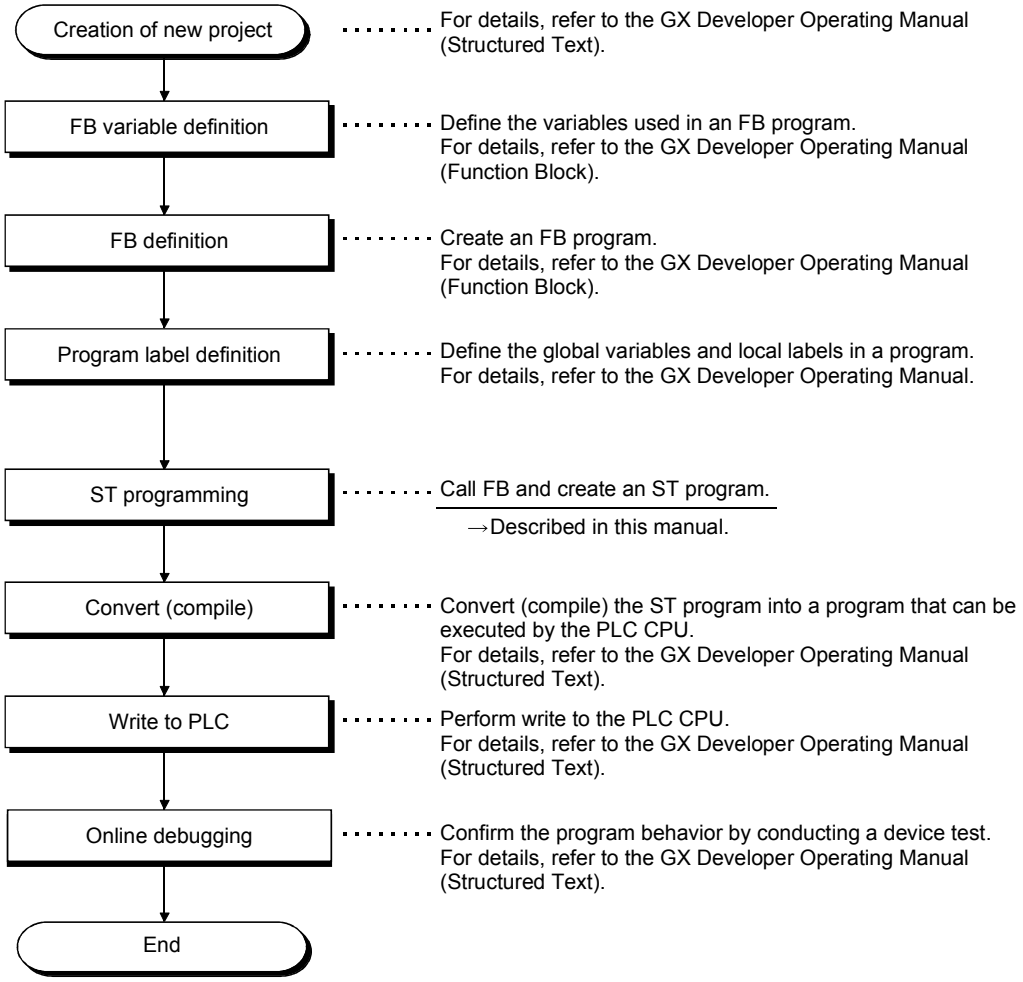
(3) Program change during system operation (online change)
Part of a running program can be changed without the PLC CPU being stopped.

(4) Connection with other language programs
Since the MELSEC-Q/L series also supports languages other than the ST, the language adequate for processing can be used to create efficient programs.
Execution conditions can be set on a file basis in each program, and multiple program files can be written to a single PLC CPU.
Multiple languages support widespread application under optimum control.

(5) A wealth of functions group
The MELSEC functions compatible with various common instructions for the MELSEC-Q/L series and the IEC functions defined in IEC61131-3 are available for ST programs in the MELSEC-Q/L series.

1.3 ST Program Creating Procedure

The following flowchart indicates the general procedure of ST programming.
In the following example, parts were created with the function block function and a main program was then created in ST language.



2 SYSTEM CONFIGURATION

2.1 System Configuration

This section explains the system configuration for use of ST programs.

2.1.1 Applicable CPUs

ST programs are applicable to the following CPU modules.

Basic Model QCPU	High Performance Model QCPU	Universal model QCPU	Process CPU	Redundant CPU	LCPU
Q00JCPU Q00CPU Q01CPU	Q02CPU Q02HCPU Q06HCPU Q12HCPU Q25HCPU	Q00UJCPU Q00UCPU Q01UCPU Q02UCPU Q03UDCPU Q03UDECPU Q04UDHCPU Q04UDEHCPU Q06UDHCPU Q06UDEHCPU Q10UDHCPU Q10UDEHCPU Q13UDHCPU Q13UDEHCPU Q20UDHCPU Q20UDEHCPU Q26UDHCPU Q26UDEHCPU	Q02PHCPU Q06PHCPU Q12PHCPU Q25PHCPU	Q12PRHCPU Q25PRHCPU	L02CPU L26CPU-BT

2.1.2 Programming tool for ST program

Use the following programming tool to create, edit and/or monitor ST programs.

Software Package Name	Operating Environment
GX Developer Version 8.00A or later	Refer to the "GX Developer Version 8 Operating Manual (Startup)".

2.1.3 ST program specifications

This section explains the ST specifications and applicable devices.

(1) Program size

The file size per program is 839680.

Point

- Note the following when counting the number of characters in a file.
  - Carriage return (CR) and Line feed (LF) are handled as two characters.
  - A space is handled as one character.
  - A TAB code is handled as one character.

(2) Applicable devices

The device names that can be used in ST programs are as indicated below. The number of device points can be changed in parameter setting.
Refer to Section "3.3.3 Devices" for details of the device representation methods.

Classification	Type	Device	Representation
Internal user device	Bit	Input	X
		Output	Y
		Internal relay	M
		Latch relay	L
		Annunciator	F
		Link relay	B
		Link special relay	SB
	Word	Data register	D
		Link register	W
		Link special register	SW
Internal system device	Bit	Special relay	SM
	Word	Special register	SD
Link direct device	Bit	Link input	Jn\X
		Link output	Jn\Y
		Link relay	Jn\B
		Link special relay	Jn\SB
	Word	Link register	Jn\W
		Link special register	Jn\SW
Intelligent function module device	Word	Intelligent function module device	Un\G
Index register	Word	Index register	Z*1
File register	Word	File register	R
			ZR
Constant	Bit/word/double word	Decimal constant	K
		Hexadecimal constant	H
	Real number	Real number constant	E
	Character string	Character string constant	"ABC", etc.
Others	Bit	SFC block device	BL
	Bit	SFC transition device	BL\TR
	Bit	SFC step relay	BL\S
	Bit	Direct input	DX
	Bit	Direct output	DY

*1: Z0 and Z1 cannot be used.
For Universal model QCPU/LCPU, Z16 to Z19 cannot be used.

(3) Devices applicable to ST programs only

In ST programs, the contacts, coils and present values of the timers and counters are represented and used as individual devices.
The device representations and types of the contacts, coils and present values of the timers and counters are as indicated below.

Classification	Type	Device	Representation
Internal user device	Bit	Timer contact	TS
		Timer coil	TC
		Retentive timer contact	STS
		Retentive timer coil	STC
		Counter contact	CS
		Counter coil	CC
	Word	Timer present value	TN/T
		Retentive timer present value	STN/ST
		Counter present value	CN/C

Examples of use

(1) [ST program]
M0: = TS0;

[Equivalent list program]
LD T0
OUT M0

(2) [ST program]
COUNTER_M(X0, CC20, 10);

[Equivalent list program]
LD X0
OUT C20 K10

Point

For details of compatible instructions, refer to the following manual:
• MELSEC-Q/L Programming Manual (Common Instructions)

3 HANDLING OF CHARACTERS AND NUMERIC VALUES IN ST PROGRAMS

3.1 Usable Characters

The ST language is a programming language described in text format.
It can be described as in document editing using a general text editor, but the grammar and usable characters and symbols have been defined.

(1) Usable characters

The following characters can be used in ST programs.

Character Type	Locations of Application				Character Examples
	Program statement	Comment	Character string	Label[*1]	
Alphanumeric characters	○	○	○	○	ABC, IF, D0
Symbols + - * / = < > [ ] ( ) . , _ : ; $ # " ' { }	○	○	△[*2]	×	(D0 * D1)
Space	○	○	○	×	
Line feed code	○	○	×	×	
TAB code	○	○	×	×	

○: Can be used. ×: Cannot be used. △: Part cannot be used.

*1: For the characters that cannot be used in labels, refer to "Appendix 1 Character Strings that cannot be Used as Labels and FB Names".

*2: A double quotation (") cannot be used in a character string.
Doing so will result in a conversion error.

3 - 1

(2) Character types

The characters used in ST programs can be classified as indicated below.

Classification		Description	Example
Label name		Character string defined as desired by the user. It includes a function block name, array name, structure name, etc.	Switch_A
Constant		Value written directly to a program. (Integer, real number, character string, etc.)	123, "abc"
Comment		Commentary statement that is not the processing target of control in a program.	(* Turns ON *)
Reserved word	Data type name	Word that represents a data type.	BOOL, DWORD
	Control syntax	Word whose meaning has been defined grammatically for use as a control syntax.	IF, CASE, WHILE, RETURN
	Device name	Data name for MELSEC PLC	X, Y, M, ZR
	Function name	MELSEC function/IEC function name already defined.	OUT_M REAL_TO_STR_E
Operator		Character code whose meaning has been defined for an expression or assignment statement.	+ - < > =
Delimiter		Character code whose meaning has been defined to clarify a program structure.	; ( )
Other symbols		Code for putting a layout in order.	Space Line feed code, TAB

3 - 2

3.2 Data Handling

In ST programs, the types of used data have been defined.
Sections 3.2 and 3.3 indicate the data types and their representation methods in ST programs.

3.2.1 Data types

The following data types can be used in ST programs.

Data Type	Definition	Range	Type in Ladder	Type in C Language
BOOL	Boolean type	TRUE•FALSE,1•0*1	Bit	bool
INT	Integer type	-32768 to 32767	Word	signed short
DINT	Double precision integer type	-2147483648 to 2147483647	Double word	signed long
REAL	Real number type	-3.402823^{+38} to -1.175495^{-38}, 0.0, $+1.175495^{-38}$ to $+3.402823^{+38}$	Real number	float
STRING	Character string type	Up to 50 characters can be defined.	Character string	char
ARRAY	Array data type	Depends on the data type of the specified element.	Array	char[ ], etc.
STRUCT	Structured data type	Depends on the data type of the specified element.	Structure	struct

*1: K0, K1, H0 and H1 for specification of K and H cannot be used as the BOOL type.

Point

- Precautions when an operation result exceeds the data type range
  When an operation result exceeds the data type range, correct result cannot be obtained.

3 - 3

3.2.2 About ANY type

Use the ANY type when multiple data types are permitted for the argument, return value, etc. of a function. The ANY type is a data type that handles any data type and is available in different types indicated in the following table.
For example, when the argument of a function has been defined as ANY_NUM, any data type can be specified as an argument from the word type, double word type and real number type.

[Description example]
REAL EXPT(REAL In1, ANY_NUM In2); (* Function definition of function EXPT *)

→ Word type, double word type or real number type can be specified.

- When a word type device is specified
  RealLabel := EXPT(E1. 0, D0);
- When a double word type label is specified
  RealLabel := EXPT(E1. 0, DWLabel);
- When a real number is specified
  RealLabel := EXPT(E1. 0, E1. 0);

The data types and device types corresponding to the ANY types are as indicated below.

ANY Type	Data Type	BOOL	INT	DINT	REAL	STRING
	Type in ladder	Bit	Word	Double word	Real number	Character string
ANY		○	○	○	○	○
ANY_SIMPLE		○	○	○	○	○
ANY_BIT		○	△	□	－	－
ANY_NUM		－	○	○	○	－
ANY_REAL		－	－	－	○	－
ANY_INT		－	○	○	－	－
ANY16		－	○	－	－	－
ANY32		－	－	○	－	－

○ : Can be specified as corresponding type.
－ : Cannot be specified.
△ : Can be used for device, constant and digit specification, but cannot be used for label.
□ : Can be used for constant and digit specification.

3 - 4

3.2.3 Array and structure

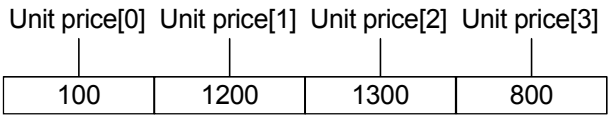
In ST programs, arrays and structures can be used as data.
Arrays and structures are data having a structure that can be handled as one block in a program when their elements are defined with local or global labels before use.

(1) Array

An array is a data type that has been defined by combining multiple data of the same type.
For an array in an ST program, each element can be referred to individually by specifying its element number within [ ] after the variable (label) name defined for the array type.
The specification numbers of the array elements are counted from 0.

[Format]
Array name[specification number of array element]

[Image diagram]
When a word type array having four elements is set to have the array name of Unit price, the specification numbers of the array elements are 0, 1, 2, 3.



For a word type array, word data enters each array element.

[Description example]

```
Unit price [0] := 100;    (* 10 is assigned to the first element of the array *)

(* 1200 is assigned to the second element of the array using device D1 *)
D1 := 1;
Unit price[D1] := 1200;
```

Data type INT can be used as the specification number of the array element.

```
(*Unit price [0] + Unit price [1] is assigned to the third element of the array *)
Unit price [2] := Unit price [0] + Unit price [1];

pen1 := 3;
Unit price[pen1] := 800;
```

A label can be used as the specification number of the array element.

3 - 5

Point

- Precaution for use of the specification numbers of the array elements
  When an array has n elements, the specification numbers of the array elements are 0 to n-1. Hence, if n or more is specified, an error will occur at the time of conversion.

  Example: When an array has four elements
  Unit price [4]: = 100; ← Error occurs.

- Precaution for use of arrays in the specification number of the array element
  Arrays can be used in the specification number of the array element. Up to five arrays can be nested. Using 17 or more arrays will result in a conversion error.

  Example: When five arrays are nested
  Unit price [Unit price [Unit price [Unit price [Unit price [D1]]]]] = 100;

- Precaution for setting the specification number of the array element
  Since there is a possibility that the data of the other devices may be corrupted, be careful so that the value specified as the array element number does not exceed the number of array elements.

- Precaution for setting the number of array elements
  Enter the number of elements on the global (local) variable setting screen. The number of elements that can be entered is 256.

3 - 6

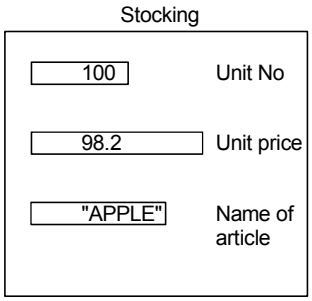
(2) Structure

A structure is a data type defined by combining the data of any types.
Each element can be referred to individually by describing the element name after the variable (label) name defined for the structure type, with a period (.) placed between them.
The element name is also called a member variable.

[Format]
Structure name.structure element name

[Image diagram]
When the setting is as follows
Structure name stocking,
Structure element:

One word type Structure element name Unit No
One real number type Structure element name unit price
One character string type Structure element name name of article



[Description example]

```
(* 100 is assigned to structure element Unit No *)
Stocking.Unit No := 100;

(* 98.2 is assigned to structure element Unit price *)
Stocking.Unit price := E98.2;

(* "APPLE" is assigned to structure element Name of article *)
Stocking.Name of article := “APPLE”;
```

Point

- Precaution for use of the member variables of a structure
  The number of members that can be entered on the structure variable setting screen is 128.

3 - 7

3.3 Data Representation Methods

Constants, labels and devices can be used as data in ST programs.

Item	Description	Representation Example
Constant	Numeric value or character string data written directly to a program. It does not change during program execution.	123, “ABC”
Label	Data whose type and name are defined by the user.	Switch_A
Device	Device used by the QCPU (Q mode)/LCPU. It is identified by the device name and device number.	X0, Y0, D100, J1\X0

3.3.1 Constants

Each constant is represented as described below in ST programs.

Data Type	Numeric Notation	Representation Method	Example
BOOL		TRUE • FALSE 1•0	M0 := TRUE;
	Binary	The used binary number is preceded by "2#".	M0 := 2#0; M0 := 2#1;
	Octal	The used octal number is preceded by "8#".	M0 := 8#0; M0 := 8#1;
	Hexadecimal	The used hexadecimal number is preceded by "16#".	M0 := 16#0; M0 := 16#1;
INT DINT	Binary	The used binary number is preceded by "2#".	D0 := 2#110;
	Octal	The used octal number is preceded by "8#".	D0 := 8#377;
	Decimal	The used decimal number is preceded by "10#". (The numeric value may be preceded by "K".)	D0 := 123; D0 := K123;
	Hexadecimal	The used hexadecimal number is preceded by "16#". (The numeric value may be preceded by "H".)	D0 := 16#FF; D0 := HFF;
REAL		The used real number is directly input. (The numeric value may be preceded by "E".)	ABC := 2.34; Rtest := E2.34;
STRING		A character string is enclosed by ' ' (or " ").	Stest := 'ABC'; Stest := “ABC”;

For the range that can be specified for each constant, refer to Section 3.2.1 Data types.
The following ranges apply to the areas that are not described in Section 3.2.1 Data types.

[K, H representation]

Value Range	IEC Data Type
K-32768 to K32767	INT, ANY16
K-2147483648 to K2147483647	DINT, ANY32
K0 to K32767	ANY_BIT (word)*1
K0 to K2147483647	ANY_BIT (double word)*2
H0 to HFFFF	INT, ANY16, ANY_BIT (word)*1
H0 to HFFFFFFFF	SINT, ANY32, ANY_BIT (double word)*2

[K, H-less representation]

Value Range	IEC Data Type
0 to 1	BOOL
-32768 to 32767	INT
-2147483648 to 2147483647	DINT
0 to 4294967295	ANY_BIT (double word)*2
0 to 65535	ANY_BIT (word)*1
-32768 to 65535	ANY16
-2147483648 to 4294967295	ANY32
2#0 to 2#1 8#0 to 8#1 16#0 to 16#1	BOOL
2#0 to 2#1111_1111_1111_1111 8#0 to 8#177777 16#0 to 16#FFFF	INT ANY16 ANY_BIT (word)*1
2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111 8#0 to 8#37777777777 16#0 to 16#FFFFFFFF	DINT ANY_BIT (double word)*2 ANY32

*1: Indicates when handled as a word device.
<Example> D0 := NOT(K32767);

*2: Indicates when handled as a double word device.
<Example> K8M0 := NOT(K2147483647);

3 - 9

Point

- Precaution for use of the H, 2#, 8# and 16#-specified numeric values in word label and word device operation expressions

  When the value handled in operation is in the range H8000 to HFFFF, the operation result available by ST program conversion differs from the operation result available by the assignment of a value to a device in the PLC CPU.

  Since whether the handled value is a word type or double word type cannot be judged in the operation result available by ST program conversion, it is operated as unsigned, but it is operated as signed in the PLC CPU.

  <Example of use>

  Data1 = -32768;

  Data2 = 16#8000;

  • ST Result := Data1 / Data2; → -32768 / 32768 = -1

  • CPU Result := Data1 / Data2; → -32768 / -32768 = 1

- Precaution for use of "$" and " ' " in character string type data

  "$" is used as an escape sequence.

  Two hexadecimal numbers following "$" are recognized as the ASCII code, and the characters corresponding to the ASCII code are inserted into the character string.

  A conversion error will occur when the two hexadecimal numbers following "$" do not correspond to the ASCII code.

  However, an error will not occur when the characters following "$" are any of the following.

Representation	Symbol/Printer Code Used in Character String
$$	$
$'	'
$L or $1	Line feed
$N or $n	Change line
$P or $p	Page scrolling
$R or $r	Carriage return
$T or $t	Tab

  Example: Value := “$'APPLE$' $$100”;

- Precaution for binary, octal, decimal, hexadecimal and real number representations

  In binary, octal, decimal, hexadecimal or real number representation, "_ (underscore)" can be used for ease of identification. "_" is ignored as a numeric value.

  Example: 2#1101_1111 8#377_1 16#01FF_ABCD 22_323 1.0_1

  (When K, H or E is specified, "_" cannot be used.)

3.3.2 Labels

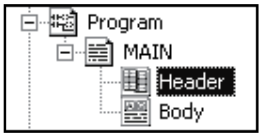
In ST programs, labels can be used with data.
When labels are used in an ST program, label declaration must be made on the local variable setting screen or global variable setting screen before use.
(For the label and structure label declaration methods, refer to the "GX Developer Operating Manual".)
Label representation examples in ST programs are as follows.

```
Example:  Switch_A:= FALSE;            (* FALSE is assigned to Switch_A. *)

Example:  IF INT_TO_BOOL(Unit_No) = FALSE THEN
              Line_No := 2147483647;
          END_IF;
                                       (* IF INT_TO_BOOL (Unit_No) is FALSE *)
                                       (* 2147483647 is assigned to Unit_Number. *)

Example:  Limit_A := E1.0;             (* 1.0 is assigned to Limit_A *)

Example:  Conveyor[4] := Unit_No;      (* The value of Unit_No is assigned to *)
                                       (* the fifth element of Conveyor. *)

Example:  stPressure.Status := TRUE; (* TRUE is assigned to *)
                                       (* element name Status of stPressure. *)

Example:  stPressure.eLimit := E1.0;   (* 1.0 is assigned to *)
                                       (* element name eLimit of stPressure. *)
```

REFERENCE

● Label declaration procedure

Make label declaration on the local variable setting screen or global variable setting screen.
The local variable setting screen can be opened by performing the following operation.

GX Developer start → [Open project] → 

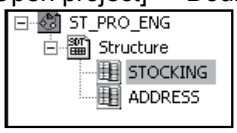
→ Double-click Header icon → Local variable setting screen

The following example shows the label setting made on the local variable setting screen.

	Au	Label	Constant	Device type	
1		Switch_A		BOOL	
2		Unit_No		INT	
3		Line_No		DINT	
4		Limit_A		REAL	
5		Conveyer		INT(20)	

When structure label is to be declared

1) Declare the structure element.

GX Developer start → [Open project] → Double-click the structure tab
→ Add new structure → 

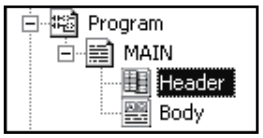
→Double-click "STOCKING" or "ADDRESS".→Structure variable setting screen

The following example shows the structure element label setting made on the structure variable setting screen.

	Label	Device type
1	Unit_No	INT
2	Unit_Price	REAL

2) Declare the structure label.

Make structure label declaration on the local variable setting screen or global variable setting screen.
The local variable setting screen can be opened by performing the following operation.



→ Double-click Header icon
→ Local variable setting screen

The following example shows the structure label setting made on the local variable setting screen.

	Au	Label	Constant	Device type
1		StockingData	Setting detail	STRUCTURE(STOCKI

3 - 12

- Precaution for use of the pointer type, timer type, counter type and retentive timer type labels

  The pointer type, timer type, counter type or retentive timer type label can be declared, but if it is used in an ST program as a label, a conversion error will occur and the label cannot be used.

- Precaution for use of the timer type, counter type and retentive timer type labels

  If the timer type, counter type or retentive timer type label is defined in the member variable of a structure, that member variable cannot be used on the ST edit screen. However, the other member variables of a structure that include the timer type, counter type and retentive timer type labels can be used.

3 - 13

3.3.3 Devices

(1) How to use devices

In an ST program, devices of QCPU (Q mode)/LCPU can be used by directly describing them without labels being used. Devices can be used in the left and right members of an expression and the argument, return value, etc. of a function.

[Description example]

```
M0 := TRUE;                              (* M0 is turned ON. *)

IF INT_TO_BOOL(D0) = FALSE THEN  (* If INT_TO_BOOL(D0) is FALSE *)
W0 := 1000;                              (* 1000 is assigned to W0. *)
END_IF;
```

REMARK

- When devices are to be specified ...
  Devices can be specified in both upper case and lower case.
- What devices are available?
  For available devices, refer to "2.1.3(2) Applicable devices" in this manual.

(2) Other using methods

The following three methods can be used as the device modification and specifying methods.
These can be used in the same usage as when devices are used in ladder programs. The following gives the description examples and explanations for use of devices in ST programs. (For details of each using method, refer to the "MELSEC-Q/L Programming Manual (Common Instructions)".)

(a) Index modification
(b) Bit specification
(c) Digit specification

(a) Index modification

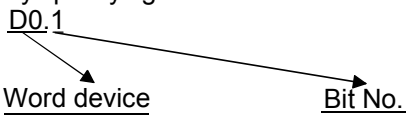
Index modification is indirect address specification using the index register. When the index register is used, the device number is (directly specified device number) + (index register contents).

[Description example]

```
(* The target D device number is changed for the numeric value in Z2      *)
(* When 1 is in Z2, the target device number changes from D(0+1) to D1. *)
Z2 := 1;                (* 1 is assigned to index register Z2 *)
D0Z2 := K0;             (* K0 is assigned to D0Z2             *)
  └►D1
```

(b) Bit specification

By specifying the bit No. of a word device, it can be used as a bit device.

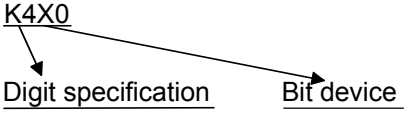


[Description example]

```
D0.0 = TRUE;            (* Bit 0 of D0 device is turned ON.     *)
W0.F = FALSE;           (* Bit 15 of W0 device is turned OFF. *)
```

(c) Digit specification

By specifying the 4 bits, 8 bits, 12 bits, etc. of a bit device as a single digit, word data or double word data can be handled by the bit device.



[Description example]

```
K4X0 := D0;             (* 16 bits are used from X0 device as integer
                                    type (INT) and D0 is assigned. *)

Wtest := K1X0;          (* 4 bits are assigned to word type label
                                    Wtest from X0 device. *)

Dwtest := K5X0;         (* 20 bits are assigned to double word type
                                    label Dwtest from X0 device. *)
```

REMARK

● Data type when digit specification is used ...
When digit specification is used, the data types are as follows.
Example: When X0 is used
Integer type (INT): K1X0, K2X0, K3X0, K4X0
Double precision integer type (DINT): K5X0, K6X0, K7X0, K8X0

Point

● Precaution 1 for use of digit specification
A conversion error will occur if the data type differs between the right member and left member.
Example: D0 := K5X0;
Since K5X0 is a double word type and D0 is a word type, the above program will result in an error.

● Precaution 2 for use of digit specification
If the right member is greater than the left member, data will be transferred to the left member within the range of the applicable number of points.
(For the applicable number of points for digit specification, refer to the MELSEC-Q/L Programming Manual (Common Instructions).)
Example: K5X0 := 2#1011_1101_1111_0111_0011_0001;
K5X0 : Applicable number of points = 20 points
1101_1111_0111_0011_0001 (20 digits) is assigned to K5X0.

3 - 16

4 ST PROGRAM EXPRESSIONS

4.1 Assignment Statement

An assignment statement has a function to assign the result of an expression in the right member to a label or device in the left member.
In the assignment statement, the result of the expression in the right member must be equal to the data type in the left member. If they are different, a conversion error will occur.

[Description example]

• When actual device is used

D0 : = 0;

When this expression is executed, a decimal number of 0 is assigned to D0.

• When label is used

When the character string type label of Stest is used

Stest : = "APPLE";

When this expression is executed, character string "APPLE" is assigned to Stest.

● Precaution for assigning a character string

A character string of up to 32 characters can be assigned. A conversion error will occur if a character string of more than 32 characters is assigned.

● Precaution for use of a device in the left member of an assignment statement

The TS, TC, STS, STC, CS, CC, BL, DX, BL\S, or BL\TR device cannot be used in the left member of an assignment statement. A conversion error will occur if any of the above devices is used in the left member.

4 - 1

4.2 Operators

This section gives a list of operators usable in ST programs and their examples of use.

4.2.1 Operator list

The following table lists the operators used in ST programs and indicates the priorities at the time of operation execution.

Operator	Description	Priority
( )	Parenthesis expression	Highest
Function ( )	Function parameter list	
**	Exponent (exponentiation) tei**shisuu	
NOT	Boolean complement (Bit inverted value)	
* / MOD	Multiplication Division Modulus operation	
+ -	Addition Subtraction	
<, >, <=, >=	Comparison	
= <>	Equality Inequality	
AND, &	Logical product	
XOR	Exclusive logical add	
OR	Logical sum	Lowest

When the priorities are the same, evaluation is made from the left-hand side to the right-hand side operators.

The following table lists the operators, applicable data types and operation result data types.

Operator	Applicable Data Type	Operation Result Data Type
*, /, +, -	ANY_NUM	ANY_NUM
<, >, <=, >=, =, <>	ANY_SIMPLE	BOOL
MOD	ANY_INT	ANY_INT
AND, &, XOR, OR, NOT	ANY_BIT*1	ANY_BIT*1
**	ANY_ REAL (base) ANY_NUM (exponent)	ANY_ REAL

*1: Except the label and constant (negative range).

4 - 2

- Precaution 1 for use of operator
  A conversion error will occur if the applicable data in the right member of an operator is not the same in data type as the applicable data in the left member.

- Precaution 2 for use of operator
  The number of used operators that can be described in a single expression is up to 1024. A conversion error will occur if 1025 or more operators are used.

REMARK

- Explanation of ANY type ...
  For the explanation of the ANY type, refer to "3.2.2 About ANY type".

4 - 3

4.2.2 Examples of using the operators

The following gives the examples of using the operators in ST programs.

(1) Operation of integer type (INT)

(a) When actual devices are used

[Example of use]
D0 : = D1 * (D2 + K3) / K100;

<<Operation order>>
1) D2 + K3
2) (D2 + K3) * D1
3) (D2 + K3) * D1 / K100
4) The result of 3) is assigned to D0.

(b) When labels are used

- When word type labels Dtest1, Dtest2 are used

[Example of use]
Dtest2 : = Dtest1 MOD (D2 + K3) * K100;

<<Operation order>>
1) D2 + K3
2) Dtest1 MOD (D2 + K3)
3) Dtest1 MOD (D2 + K3) * K100
4) The result of 3) is assigned to Dtest2.

- When double word type labels Dwtest1, Dwtest2 are used

[Example of use]
Dwtest2 : = Dwtest1 - Dwtest1 / K100;

<<Operation order>>
1) Dwtest1 / K100
2) Dwtest1 - Dwtest1 / K100
3) The result of 2) is assigned to Dwtest2.

Point

- Precautions when an operation result exceeds the data type range
  When an operation result exceeds the data type range, correct result cannot be obtained.
  For data type range, refer to Section 3.2.1.

4 - 4

(2) Operation of Boolean type (BOOL)

(a) When actual devices are used

[Example of use]

M0 : = X0 AND X1 AND (D1 = 100);

<<Operation order>>

1) Only when the result of X0 AND X1 is ON and D is 100, M0 turns ON.

(b) When labels are used

- When bit type labels Btest1, Btest2 are used

[Example of use]

Btest2 : = Btest2 OR Btest1;

<<Operation order>>

1) When Btest2 or Btest1 is ON, Btest2 turns ON.

4 - 5

4.3 Control Syntaxes

Conditional statements and repeat statements are available for ST programs to perform comparison and repetition.

Conditional statement: When a certain condition is satisfied, the selected statement is executed.
Repeat statement: One or more statements are executed repeatedly according to the state of a certain variable or condition.

4.3.1 Control syntax list

The following table lists the control syntaxes.

Conditional statement	IF conditional statement
	CASE conditional statement
Repeat statement	FOR ... DO syntax
	WHILE ... DO syntax
	REPEAT ... UNTIL syntax
Other control syntaxes	RETURN syntax
	EXIT syntax

Point

- Precaution for use of a hierarchy for a control syntax
  A hierarchy of up to 16 levels is enabled for a control syntax. A conversion error will not occur if 17 or more levels are used. However, since a deep hierarchy may make a program difficult to understand, it is recommended to program a hierarchy up to 4 or 5 levels at the deepest.

4.3.2 Conditional statements

(1) IF THEN conditional statement

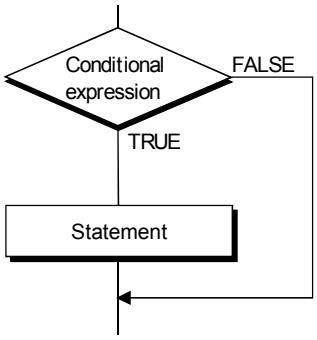
[Format]

```
IF <Boolean expression> THEN
    <Statement ... >
END_IF;
```

[Explanation]

The statement is executed when the Boolean expression (conditional expression) is TRUE. If the Boolean expression is FALSE, the statement is not executed.

Any Boolean expression can be used if it returns TRUE or FALSE as the result of Boolean operation of the condition of a single bit type variable or a complicated expression including many variables.



[Description example]

(a) When actual device is used in Boolean expression

```
IF X0 THEN              (* If X0 is ON, 0 is assigned to D0.              *)
    D0 : = 0;           (* If the X0 area is X0= TRUE, the meaning is *)
                        (* the same.                                      *)
END_IF;
```

(b) When operator is used in Boolean expression

```
IF (D0*D1) <= 200 THEN  (* If D0*D1 is less than or equal to 200  *)
    D0 : = 0;           (* 0 is assigned to D0.                   *)
END_IF;
```

(c) When label is used in Boolean expression

```
IF w_Real > 2.0 THEN    (* If w_Real is greater than 2.0  *)
    D0 : = 0;           (* 0 is assigned to D0.           *)
END_IF;
```

2) When label w_Str is specified as character string type

```
IF w_Str = "ABC" THEN       (* If w_Str is "ABC"                *)
      D0 : = 0;             (* 0 is assigned to D0.             *)
END_IF;
```

3) When label w_Str is specified as character string type

```
IF w_Str = 'ABC' THEN       (* If w_Str is 'ABC'                *)
      D0 : = 0;             (* 0 is assigned to D0.             *)
END_IF;
```

(d) When function block is used in Boolean expression

When function block name w_FB is set to the local variable setting and word type label w_Out is set as the output variable of the function block
After the function block is executed
(For the method of using the function block, refer to the "GX Developer Version 8 Operating Manual".)

```
IF w_FB. w_Out = 100 THEN   (* If w_Out is 100                  *)
      D0 : = 0;             (* 0 is assigned to D0.             *)
END_IF;
```

(e) When function is used in Boolean expression

```
IF INT_TO_BOOL (D0) = FALSE THEN
      D0 : = 0;             (* If INT_T0_BOOL (D0) is FALSE     *)
                            (* 0 is assigned to D0.             *)
END_IF;
```

(2) IF ... ELSE conditional statement

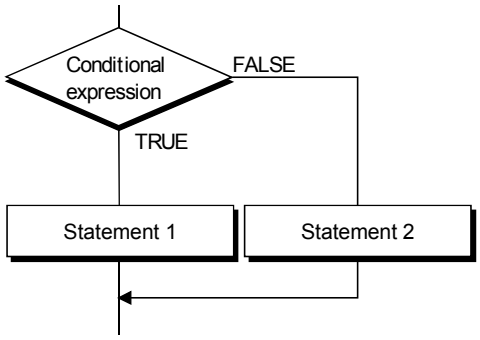
[Format]

```
IF <Boolean expression> THEN
      <Statement1 ... >
ELSE
      <Statement2 ... >
END_IF;
```

[Explanation]

Statement 1 is executed when the Boolean expression (conditional expression) is TRUE.

Statement 2 is executed if the value of the Boolean expression is FALSE.



[Description example]

(a) When actual device is used in Boolean expression

```
IF X0 THEN              (* If the X0 area is X0= TRUE, the meaning is *)
                        (* the same.                                  *)
      D0 : = 0;         (* If X0 is ON, 0 is assigned to D0.          *)
ELSE                    (* If X0 is not ON, 1 is assigned to D0.      *)
      D0 : = 1;
END_IF;
```

(b) When operator is used in Boolean expression

```
IF (D0*D1) <= 200 THEN  (* If D0*D1 is less than or equal to 200     *)
      D0 : = 0;         (* 0 is assigned to D0.                       *)
ELSE                    (* If D0*D1 is not less than or equal to 200 *)
      D0 : = 1;         (* 1 is assigned to D0.                       *)
END_IF;
```

(c) When function is used in Boolean expression

```
IF INT_TO_BOOL (D0) = FALSE THEN
                        (* If INT_T0_BOOL(D0) is FALSE     *)
      D0 : = 0;         (* 0 is assigned to D0.            *)
ELSE                    (* If INT_T0_BOOL(D0) is not FALSE *)
      D0 : = 1;         (* 1 is assigned to D0.            *)
END_IF;
```


4 - 9

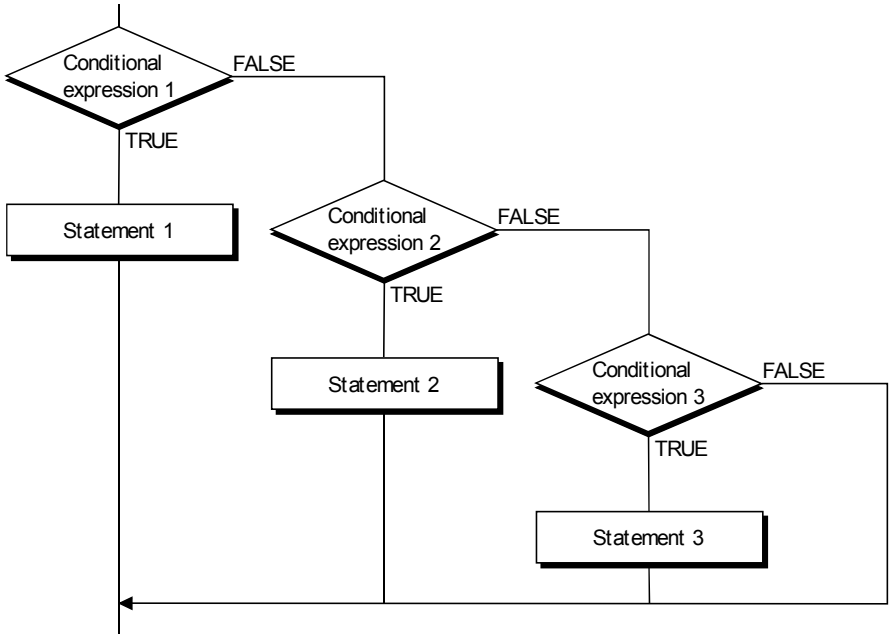
(3) IF ... ELSIF conditional statement

[Format]

```
IF <Boolean expression 1> THEN
      <Statement 1 ... >
ELSIF <Boolean expression 2> THEN
      <Statement 2 ... >
ELSIF <Boolean expression 3> THEN
      <Statement 3 ... >
END_IF;
```

[Explanation]

Statement 1 is executed when Boolean expression (conditional expression) 1 is TRUE. Statement 2 is executed if the value of Boolean expression 1 is FALSE and the value of Boolean expression 2 is TRUE.
Statement 3 is executed if the value of Boolean expression 2 is FALSE and the value of Boolean expression 3 is TRUE.



4 - 10

[Description example]

(a) When actual devices are used in Boolean expressions

```
IF D0 < 100 THEN              (* If D0 is less than 100              *)
    D1 : = 0;                 (* 0 is assigned to D1.                *)
ELSIF D0 <= 200 THEN          (* If D0 is less than or equal to 200  *)
    D1 : = 1;                 (* 1 is assigned to D1.                *)
ELSIF D0 <= 300 THEN          (* If D0 is less than or equal to 300  *)
    D1 : = 2;                 (* 2 is assigned to D1.                *)
END_IF;
```

(b) When operators are used in Boolean expressions

```
IF (D0*D1) < 100 THEN         (* If D0*D1 is less than 100           *)
    D1 : = 0;                 (* 0 is assigned to D1.                *)
ELSIF (D0*D1) <= 200 THEN     (* If D0*D1 is less than or equal to 200  *)
    D1 : = 1;                 (* 1 is assigned to D1.                *)
ELSIF (D0*D1) <= 300 THEN     (* If D0*D1 is less than or equal to 300  *)
    D1 : = 2;                 (* 2 is assigned to D1.                *)
END_IF;
```

(c) When functions are used in Boolean expressions

```
IF INT_TO_BOOL (D0) = TRUE THEN      (* If INT_T0_BOOL (D0) is  *)
                                     (* TRUE                    *)
    D1 : = 0;                        (* 0 is assigned to D1.    *)
ELSIF INT_TO_BOOL (D0) = TRUE THEN   (* If INT_T0_BOOL(D2) is   *)
                                     (* TRUE                    *)
    D1 : = 1;                        (* 1 is assigned to D1.    *)
END_IF;
```


4 - 11

(4) CASE conditional statement

[Format]

```
CASE <Integer expression> OF
        <Integer selection 1> : <Statement 1>
        <Integer selection 2> : <Statement 2>
                .
                .
                .
        <Integer selection n> : <Statement n>
ELSE
      <Statement n+1 ...>
END_CASE;
```

● Specifying method that can be used for <Integer selection *> in CASE conditional statement

One value, multiple values, or a value range can be specified for <Integer selection *> in the CASE conditional statement as indicated below.

Example:

```
1:                          (* When the value of the integer expression is 1 *)
2, 3, 4:        (* When the value of the integer expression is any of 2, 3 and 4 *)
5..10:             (* When the value of the integer expression is any of 5 to 10 *)
```

When ".." is used to specify the range, make the value following ".." greater than the value preceding "..".

Also, multiple values and range specification can be combined to specify values.

```
1, 2..5, 9:     (* When the value of the integer expression is any of 1, 2..5, and 9 *)
```

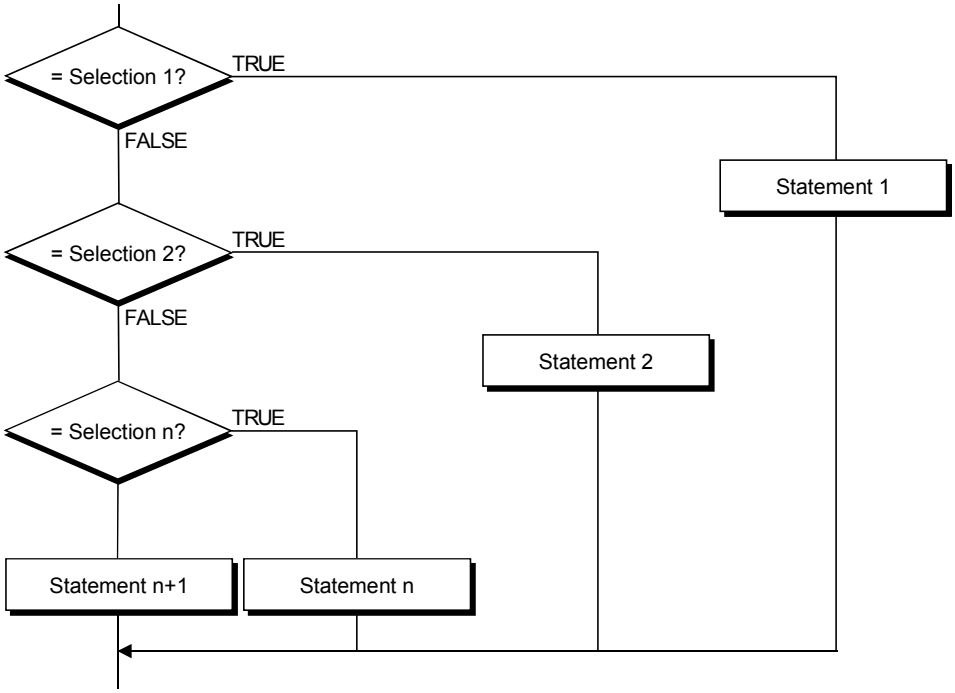
● Data types that can be used in <integer expression> of CASE conditional statement

The data types that can be specified as the <integer expression> in the CASE conditional statement are the integer type (INT) and double precision integer type (DINT). The word devices and word type or double word type labels can be specified.

[Explanation]

The result of the expression in the CASE conditional statement is returned as an integer value. This conditional statement can be used when a selection statement is executed with a single integer value or the integer value of the result of a complicated expression, for example.

The statement having the integer selection that matches the value of the integer expression is executed first, and if there are no matches, the statement that follows ELSE is executed.



[Description example]

(a) When actual device is used in integer expression

```
CASE D0 OF
    1:
        D1 : = 0;  (* If D0 is 1, 0 is assigned to D1.                    *)
    2, 3:
        D1 : = 1;  (* If D0 is 2 or 3, 1 is assigned to D1.               *)
    4..6:
        D1 : = 2;  (* If D0 is any of 4 to 6, 2 is assigned to D1.        *)
    ELSE
        D1 : = 3;  (* If D0 is other than the above, 3 is assigned to D1.  *)
END_CASE;
```

(b) When operation result is used in integer expression

```
CASE D0*D1 OF
    1:
        D1 : = 0;  (* If D0*D1 is 1, 0 is assigned to D1.                  *)
    2, 3:
        D1 : = 1;  (* If D0*D1 is 2 or 3, 1 is assigned to D1.             *)
    4..6:
        D1 : = 2;  (* If D0*D1 is any of 4 to 6, 2 is assigned to D1.      *)
    ELSE
        D1 : = 3;  (* If D0*D1 is other than the above, 3 is assigned to D1. *)
END_CASE;
```

(c) When function is used in integer expression

```
CASE DINT_TO_INT (dData) OF
    1:          (* If DINT_TO_INT (dData) is 1                    *)
        D1 : = 0;  (* 0 is assigned to D1.                        *)
    2, 3:       (* If DINT_TO_INT (dData) is 2 or 3                *)
        D1 : = 1;  (* 1 is assigned to D1.                        *)
    4..6:       (* If DINT_TO-INT(dData) is any of 4 to 6          *)
        D1 : = 2;  (* 2 is assigned to D1.                        *)
    ELSE        (* If DINT_TO_INT (dData) is other than the above  *)
        D1 : = 3;  (* 3 is assigned to D1.                        *)
END_CASE;
```

● Precaution for use of integer selection

When a CASE conditional statement has multiple values of the same integer selection, the statement on the upper line is executed with priority and the latter statement having the same integer selection is not executed. For example, when the D100 value is 3 in the following CASE conditional statement, statement 3 having integer selection 3 is executed and statement 4 having the same integer selection is not executed.

```
CASE D100 OF
    1:      < Statement 1 ...>
    2:      < Statement 2 ...>
    3:      < Statement 3 ...>
    3, 4:   < Statement 4 ...>
ELSE
            < Statement 5 ...>
END_CASE;
```

To specify the <integer selection *>, only a decimal number without K specification can be used.

4 - 14

4.3.3 Repeat statement

(1) FOR...DO syntax

[Format]

```
FOR <Repeat variable initialization>
      TO <Last value expression>
      BY <Incremental expression> DO
      < Statement ...>
END_FOR;
```

Repeat variable initialization: The data used as a repeat variable is initialized.

Last value expression, incremental expression:
The initialized repeat variable is incremented or decremented according to the incremental expression, and repetitive processing is performed until the last value is reached.

● Data types that can be used in <Last value expression, incremental expression> of FOR syntax
Integer values and the integer values of operation expression results can be specified.

[Explanation]

The FOR ... DO syntax repeatedly executes several statements according to the value of the repeat variable.

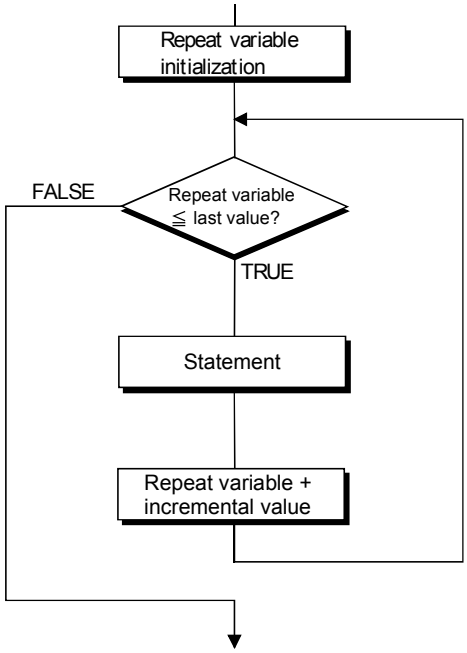
Point

● Precaution for use of repeat variable
The double precision integer type (DINT) and integer type (INT) can be used for a repeat variable, but structure elements and array elements cannot be used.
Also, match the type used for the repeat variable with the types of the <last value expression> and <incremental expression>.

● Precaution for use of incremental expression
The <incremental expression> can be omitted. When omitted, the <incremental expression> is executed as 1.
When "0" is assigned to the <incremental expression>, the FOR syntax and later may not be executed or an endless loop may occur.

● Precaution for use of FOR ... DO syntax
In the FOR ... DO syntax, the count processing of the repeat variable is performed after execution of <Statement ...> in the FOR syntax. An endless loop will occur if the count processing higher than the maximum value or lower than the minimum value of the data type of the repeat variable is executed.

4 - 15



[Description example]

(a) When actual device is used in repeat variable

```
FOR W1 : = 0          (* W1 is initialized with 0.                          *)
    TO 100            (* Processing is repeated until W1 reaches 100.       *)
    BY 1 D0           (* W1 is incremented by 1.                            *)
    W3 : = W3 + 1;    (* During repeat processing, W3 is incremented by 1. *)
END_FOR;
```


4 - 16

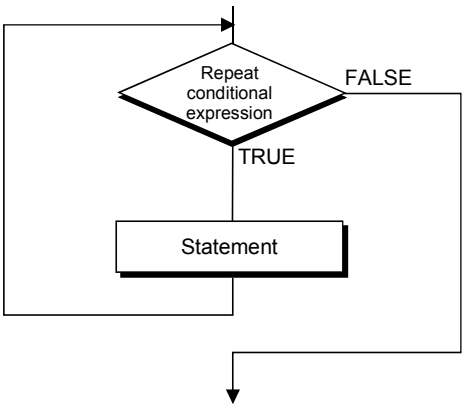
(2) WHILE...DO syntax

[Format]

```
WHILE <Boolean expression> D0
        <Statement ...>
END_WHILE;
```

[Explanation]

The WHILE ... DO syntax executes one or more statements while the Boolean expression (conditional expression) is TRUE.
The Boolean expression is judged before execution of the statement. If the Boolean expression is FALSE, the statement in WHILE ... DO is not executed. Since the <Boolean expression> in the WHILE syntax is only required to return whether the result is true or false, all expressions that can be specified in the <Boolean expression> in the IF conditional statement can be used.



[Description example]

(a) When actual device and operator are used in Boolean expression

```
WHILE W100 < (W2-100) D0   (* While W100<(W2-100) is true          *)
                           (* processing is repeated.              *)
      W100 : = W100 + 1;   (* During repeat processing, W100 is    *)
                           (* incremented by 1                     *)
END_WHILE;
```

(b) When function is used in Boolean expression

```
WHILE BOOL_TO_DINT(M0) < BOOL_TO_DINT(M1) D0
      D4 : = D4 + 1;       (* While BOOL_TO_DINT(M0) <             *)
                           (* BOOL_TO_DINT(M1) is true             *)
                           (* processing is repeated.              *)
                           (* During repeat processing, D4 is      *)
                           (* incremented by 1                     *)
END_WHILE;
```


4 - 17

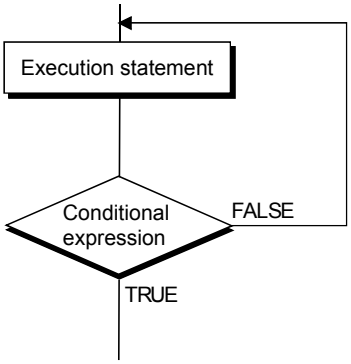
(3) REPEAT...UNTIL syntax

[Format]

```
REPEAT
    <Statement ...>
    UNTIL   <Boolean expression>
END_REPEAT;
```

[Explanation]

The REPEAT ... UNTIL syntax executes one or more statements while the Boolean expression (conditional expression) is FALSE.
The Boolean expression is judged after execution of the statement. If the value is TRUE, the statement in REPEAT ... UNTIL is not executed.
Since the <Boolean expression> in the REPEAT syntax is only required to return whether the result is true or false, all expressions that can be specified in the <Boolean expression> in the IF conditional statement can be used.



[Description example]

(a) When actual device is used in Boolean expression

```
REPEAT
    D1 : = D1 + 1;          (* Until D1 becomes less than 100    *)
    UNTIL D1 < 100          (* D1 is incremented by 1            *)
END_REPEAT;
```

(b) When operator is used in Boolean expression

```
REPEAT
    W1 : = W0*W1 - D0;      (* Until W0*W1 becomes less than 100 *)
                            (* W0*W1 - D0 is                     *)
    UNTIL W0*W1 < 100       (* assigned to W1.                   *)
END_REPEAT;
```


4 - 18

(c) When function is used in Boolean expression

```
REPEAT
    D1 : = D1 + 1;                  (* Until BOOL_TO_DINT(M0)*)
                                    (* than 100              *)
    UNTIL BOOL_TO_DINT(M0) < 100    (* becomes less          *)
                                    (* D1 is incremented by 1 *)
END_REPEAT;
```

Point

- Precaution 1 for use of repeat statements
  When using a repeat statement, be careful not to result in endless loop processing.

- Precaution 2 for use of repeat statements
  If many repeat statements are used, it should be noted that the PLC scan time will increase remarkably.

4 - 19

4.3.4 Other control syntaxes

(1) RETURN syntax

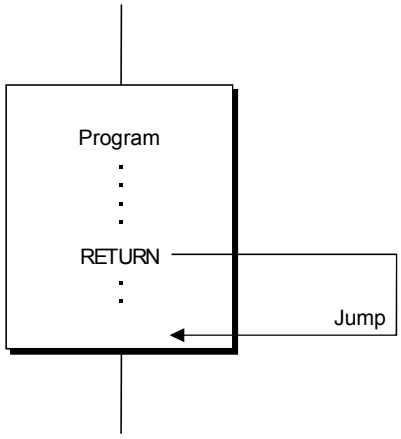
[Format]

```
RETURN;
```

[Explanation]

The RETURN syntax is used to terminate a program in a function block or an ST program.

When the RETURN syntax is used in a program, the processing after the RETURN syntax are all ignored, and a jump occurs from the place where RETURN is executed to the last line of the ST program or the program in the function block.



[Description example]

(a) When actual device is used in IF conditional statement Boolean expression

```
IF X0 THEN          (If X0 is ON, the statement in IF is executed.          *)
    RETURN;         (* The program after the RETURN line is ignored.       *)
END_IF;
```


4 - 20

(2) EXIT syntax

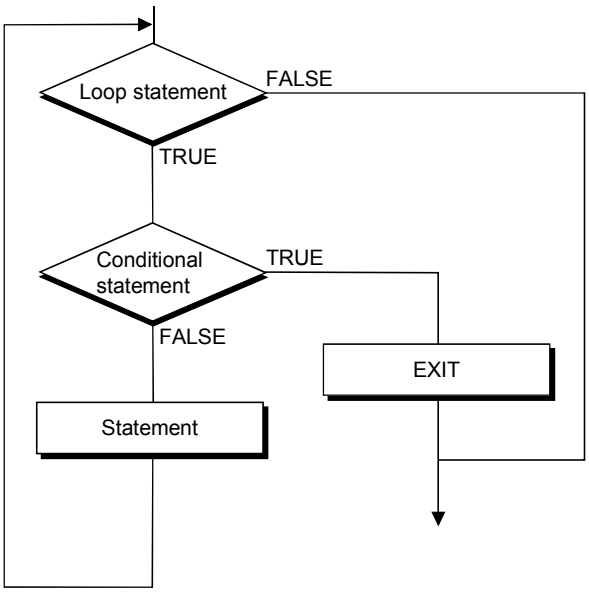
[Format]

```
EXIT;
```

[Explanation]

The EXIT syntax can be used in the repeat statement of an ST program and terminates a repeat loop midway.

When the EXIT syntax is reached during execution of a repeat loop, the repeat loop processing after the EXIT syntax is not executed. The program is continued on the line that follows the one where the repeat loop processing has been terminated.



[Description example]

(a) When actual device is used in IF conditional Boolean expression

```
FOR D0 : = 0 TO 10 D0   (* If the D0 value is less than or equal to 10,  *)
                        (* repeat is executed.                           *)
  IF D1 > 10 THEN       (* Whether the D1 value is greater than 10       *)
                        (* or not is checked.                            *)
    EXIT;               (* If the D1 value is greater than 10,           *)
                        (* repeat processing                             *)
                        (* ends.                                         *)
  END_IF;
END_FOR;
```

4.3.5 Precautions for use of control syntaxes

This section explains the number of used steps, operation processing time and instructions for use of the control syntaxes in an ST program.

(1) Number of used steps and operation processing time for use of control syntaxes

The number of used steps and operation processing time for use of the control syntaxes will be explained.
The operation processing time is calculated by addition of the processing times of the instructions. Use it as reference for program creation.

(a) IF conditional statements

IF conditional statement 1 Unit (μs)

		Number of Steps	Operation Processing Time (Q25H)	Operation Processing Time (Q00J)
ST program	IF X0 THEN D0 : = 100; END_IF;	7	1.534	10.9
List program	LD X0 MOV K100 D0	3	0.134	0.90
[Remarks] In only the conditional statement area, the processing time is shorter than when ST is not used. However, since the comparison target of the IF conditional statement in ST is the Boolean expression, complicated comparison can be made easily.				

IF conditional statement 2 Unit (μs)

		Number of Steps	Operation Processing Time (Q25H)	Operation Processing Time (Q00J)
ST program	IF D0 : = 0 THEN D0 : = 100; END_IF;	9	1.6	11.5
List program	LD X0 MOV K100 D0	5	0.20	1.50
[Remarks] In only the conditional statement area, the processing time is shorter than when ST is not used. However, since the comparison target of the IF conditional statement in ST is the Boolean expression, complicated comparison can be made easily.				

(b) CASE conditional statement

Unit (μs)

		Number of Steps	Operation Processing Time (Q25H)	Operation Processing Time (Q00J)
ST program	CASE D0 OF 1, 2: D0 : = 100; 3..10: D1 : = D1 + 1; END_CASE;	29	5.004	36.1
List program	LD= D0 K1 AND= D0 K2 MOV K100 D0 LD>= D0 K3 AND<= D0 K10 INC D1	16	0.64	4.6
[Remarks] Since CJ, JMP, etc. need not be executed in a list unlike ST, only the times for the compared areas are measured. The time has been calculated on the assumption that the compared areas are conducting.				

(c) FOR...DO statement

Unit (μs)

		Number of Steps	Operation Processing Time (Q25H)	Operation Processing Time (Q00J)
ST program	FOR D0 : = 0 T0 10 BY 1 D0 D1 : = D1 + 1; END_FOR;	16	Initialization: 0.134 Repeat: 3.308 In this case, the repeated area operates 10 times.	Initialization: 0.9 Repeat: 24.0
List program	FOR K10 LD SM400 INC D1 NEXT	6	2.574	21.6
[Remarks] The above operation processing time is taken by the number of repeat times. In a list, only the number of repeat times can be specified. In ST, repeat and other operation processing can be performed by condition comparison.				

4 - 23

(d) WHILE...DO statements

WHILE...DO statement 1 Unit (μs)

		Number of Steps	Operation Processing Time (Q25H)	Operation Processing Time (Q00J)
ST program	WHILE X0 D0 D0 : = 100; END_ WHILE;	10	3.034 Repeat is executed until X0 becomes TRUE.	21.9
List program	As above	As above	As above	As above
[Remarks] If the statement is described in a list, the program is the same as the ST program conversion result. Therefore, the processing time is also the same as that of ST.				

WHILE...DO statement 2 Unit (μs)

		Number of Steps	Operation Processing Time (Q25H)	Operation Processing Time (Q00J)
ST program	WHILE D0= 100 D0 D0 : = 100; END_ WHILE;	15	3.1	22.5
List program	As above	As above	As above	As above
[Remarks] If the statement is described in a list, the program is the same as the ST program conversion result. Therefore, the processing time is also the same as that of ST.				

(e) REPEAT...UNTIL statements

REPEAT...UNTIL statement 1 Unit (μs)

		Number of Steps	Operation Processing Time (Q25H)	Operation Processing Time (Q00J)
ST program	REPEAT D0 : = 100; UNTIL X0 END_ REPEAT;	6	1.534 Repeat is executed until X0 becomes TRUE.	10.9
List program	As above	As above	As above	As above
[Remarks] If the statement is described in a list, the program is the same as the ST program conversion result. Therefore, the processing time is also the same as that of ST.				

REPEAT...UNTIL statement 2 Unit (μs)

		Number of Steps	Operation Processing Time (Q25H)	Operation Processing Time (Q00J)
ST program	REPEAT D0 : = 100; UNTIL X0 END_ REPEAT;	9	1.6 Repeat is executed until X0 becomes TRUE.	11.5
List program	As above	As above	As above	As above
[Remarks] If the statement is described in a list, the program is the same as the ST program conversion result. Therefore, the processing time is also the same as that of ST.				

(f) EXIT statement

Unit (μs)

		Number of Steps	Operation Processing Time (Q25H)	Operation Processing Time (Q00J)
ST program	—	3	1.4	11
List program	As above	As above	As above	As above
[Remarks] Using the JMP instruction, execution moves to the pointer immediately after repeat processing termination. If the statement is described in a list, the operation is the same as in the ST program. Therefore, the processing time is also the same as that of ST.				

(g) RETURN statement

Unit (μs)

		Number of Steps	Operation Processing Time (Q25H)	Operation Processing Time (Q00J)
ST program	—	3	1.4	11
List program	As above	As above	As above	As above
[Remarks] Using the JMP instruction, execution moves to the pointer immediately after repeat processing termination. If the statement is described in a list, the operation is the same as in the ST program. Therefore, the processing time is also the same as that of ST.				

4 - 26

(2) Precautions for use of bit devices

The following explains the precautions to be taken when a program is created using an IF/CASE conditional statement in an ST program.
Once the Boolean expression (conditional expression) is satisfied in the IF condition statement, when a bit device is turned ON in the IF condition statement, that bit device becomes always ON.

[ST program example 1]

```
IF M0 THEN
        Y0 := TRUE;
END_IF;
```

The above program is equivalent to the following.

```
LD M0;
SET Y0;
```

To avoid the bit device being always ON, change the program as shown below.

[ST program example 2]

```
IF M0 THEN
        Y0 := TRUE;
ELSE
        Y0 := FALSE;
END_IF;
```

The above program is equivalent to the following.

(a)
```
LD M0;
OUT Y0;
```

(b)
```
Y0 := M0;
```

(c)
```
OUT_M (M0, Y0);
```

However, when OUT_M() is used in the IF conditional statement, the condition is as in [ST program example 1].

The above precautions also apply to when the CASE conditional statement is used.
Once the integer expression (conditional expression) is satisfied in the CASE condition statement, when a bit device is turned ON in the CASE condition statement, that bit device becomes always ON.

(3) Precautions for use of timers and counters

The following explains the precautions to be taken when a program is created using an IF/CASE conditional statement in an ST program.
In the IF condition statement, the Boolean expression (conditional expression) differs from the execution condition of the timer/counter instruction.

Example: In the case of timer
[ST program example 1]

```
IF M0 THEN
    TIMER_M (M1, TC0, K10);
END_IF;
(* When M0 = ON and M1 = ON, counting starts.                          *)
(* When M0 = ON and M1 = OFF, counting is cleared.                     *)
(* When M0 = OFF and M1 = ON, counting is stopped. The counting        *)
(* value is not cleared.                                              *)
(* When M0 = OFF and M1 = OFF, counting is stopped. The counting       *)
(* value is not cleared.                                              *)
```

Example: In the case of counter
[ST program example 2]

```
IF M0 THEN
    COUNTER_M (M1, CC0, K10);
END_IF;
(* When M0 = ON and M1 = ON/OFF, counting is incremented by 1.        *)
(* When M0 = OFF and M1 = ON/OFF, counting is not executed.           *)
(* M0 = ON/OFF and counting incrementing by 1 are not synchronized.   *)
```

The above occurs since the timer/counter-related statement is not executed if the IF condition statement is not satisfied.
When the AND condition of M0 and M1 is used to operate the timer/counter, do not use the control syntax but use only the MELSEC function.

[Changed ST program example]

- When timer is used — `TIMER_M (M0 & M1, TC0, K10);`
- When counter is used — `COUNTER_M (M0 & M1, CC0, K10);`

Using the new program, the timer/counter can be operated under the AND condition of M0 and M1.

The above precautions also apply to when the CASE conditional statement is used.
In the CASE condition statement, the integer expression (conditional expression) differs from the execution condition of the timer/counter instruction.

4.4 Call of Function Block

In an ST program, a function blocks (FB) can be used.
This section explains the method of using a user-created FB in an ST program. (For the FB creating method, refer to the "GX Developer Version 8 Operating Manual (Function Block)".

(1) Call of function block

When a created FB is to be used in an ST program, an FB name must be defined first on the local variable setting screen. (Refer to REFERENCE.)
The FB can be used by describing the defined FB name (FB call) in the ST program.

When calling the FB, describe all input variables and I/O variables. Also, always specify values for the input variables and I/O variables.
For an output variable, its description can be omitted if the result of the output variable is not needed.

[Description example]
When the following FB is created

FB data name	: LINE1_FB
Input variable	: I_Test
Output variable	: O_Test
I/O variable	: IO_Test
FB label name	: FB1

The description example of a FB call is as given below.

```
FB1( I_Test : = D0, O_Test : = D1, IO_Test : = D100):
                          ↓
          The description of the output variable can be omitted.
```

(2) How to acquire the output result

By providing "." after the FB name to specify the output variable name, the output of the FB can be acquired.

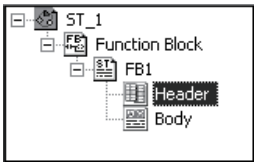
[Description example]
Describe as given below when assigning the result of the output variable to D1.

```
D1 := FB1.O_Test;
```

REFERENCE

● To make label declaration for the input, I/O and output variables of FB ...

GX Developer start → [Open project] → Click the FB tab → Add new FB

→ 

→ Double-click Header icon

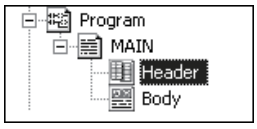
→ FB label setting screen

The following example shows the FB input/output variable label setting made on the FB label setting screen.

	Input/Output	Label	Constant	Device type
1	VAR_INPUT	I_TEST		INT
2	VAR_OUTPUT	O_TEST		INT
3	VAR_IN_OUT	IO_TEST		INT

● To make label declaration for the FB data name ...

Before an FB is called, the label declaration of the used FB must be made.



→ Double-click Header icon

→ Local (or global) variable setting screen

	Au	Label	Constant	Device type
1		BLabel		BOOL
2		DwLabel		DINT
3		label2	Setting detail	FB(FB1)
4				

COUNTER
STOREDTIMER
POINTER
STRUCTURE
FB
FB(FB1)

↓
Select FB in Device type.

The following example shows the FB label definition made on the local variable setting screen.

3		label2	Setting detail	FB(FB1)

4 - 30

Point

● Precaution for acquiring the FB output

Execute FB output acquirement after an FB call. If it is executed before an FB call, an error will occur.

Example: FB name: FB1

Input variable : I_Test

Output variable: O_Test

```
D1 : = FB1.O_Test;                    (* FB output acquirement *)
FB1(I_Test : = D0, O_Test : = D1);    (* FB call               *)
```

An error occurs since this program is written in order of FB output acquirement and FB call.

● Precaution for use of I/O variables

If the result of an I/O variable is used like an output variable, an error will occur.

Like an input variable, the value of an I/O variable must be specified at the time of an FB call.

Example: FB name: FB1

I/O variable : IO_TEST

Output variable: O_Test

[Description example]

```
FB1( IO_Test : = D1);
D1 : FB1.IO_Test; → An error occurs.
```

● Precaution for making an FB call

In an ST program, the FB set on the local variable setting screen can be used only once. (If it is used more than once, an error will occur.) To use the same FB more than once, declare the FB by the number of times it will be used beforehand on the local variable setting screen.

Example: The following example shows that the FB label has been defined more than once on the local variable setting screen.

	Au	Label	Constant	Device type
1		label	Setting detail	FB(FB1)
2		label1	Setting detail	FB(FB1)
3		label2	Setting detail	FB(FB1)

In the program, the FB is used as indicated below.

```
label (I_Test := D0, IO_Test := D100);
label1 (I_Test := D1, IO_Test := D150);
label2 (I_Test := D3, IO_Test := D200);
```


4 - 31

4.5 Comment

In an ST program, comments can be input. An area enclosed by "(*" and "*)" is handled as a comment. If a comment is placed within a comment, an error will occur.

[Description example]

Example 1: (* The pump is activated. *)

Example 2: (**********)

Example 3: (* After the switch is input, the motor is operated. *)

Example 4: (* Flag_A = TRUE control start (* Flag_B = TRUE control stop *)

[Error example]

Example 5: (* Flag_A = TRUE control start *) Flag_A = FALSE control stop *)

Example 6: (* START (* Processing stop *) Restart End *)

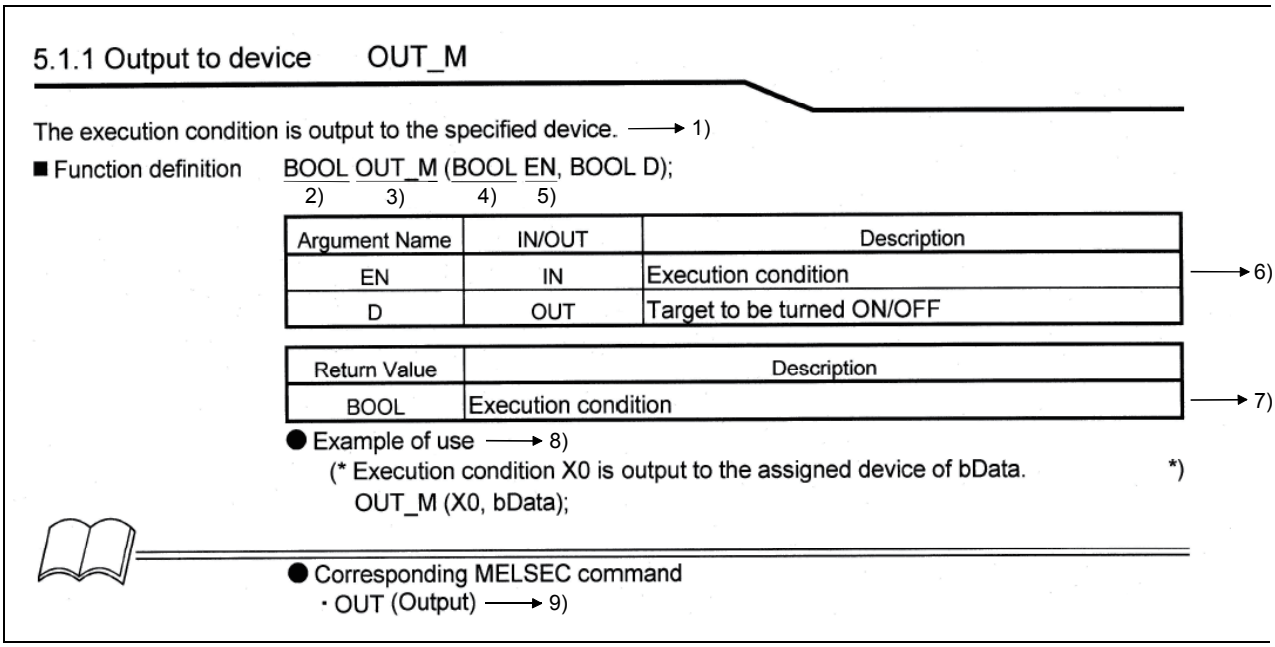
MEMO

4 - 33

5 MELSEC FUNCTIONS

How the functions are described

This manual describes the function definitions, arguments, return values and using examples of the MELSEC functions.
The MELSEC functions are created on the basis of the MELSEC common commands. For the applicable CPU types, basic operations, detailed functions and applicable devices of the functions and the errors that may occur during execution of the functions, refer to the "MELSEC-Q/L Programming Manual (Common Instructions)".
The reference section is the section described in "Corresponding MELSEC command".



5.1.1 Output to device OUT_M

The execution condition is output to the specified device. → 1)

■ Function definition BOOL OUT_M (BOOL EN, BOOL D);
2) 3) 4) 5)

Argument Name	IN/OUT	Description
EN	IN	Execution condition
D	OUT	Target to be turned ON/OFF

→ 6)

Return Value	Description
BOOL	Execution condition

→ 7)

● Example of use → 8)

```
(* Execution condition X0 is output to the assigned device of bData.  *)
  OUT_M (X0, bData);
```

● Corresponding MELSEC command
・OUT (Output) → 9)

1) Indicates the function of the function.
2) Indicates the data type of the function.
3) Indicates the function name.
4) Indicates the data type of the argument. (The STRING type is represented STRING (number of characters). It is represented STRING(6) when the number of characters is 6. The ARRAY type is represented data type(number of elements). It is represented ANY16(3) when the array is of ANY16 type and has three elements.)
5) Indicates the argument name.
6) Indicates the list (argument name, IN/OUT, description) of arguments used with the function. (The STRING type is represented ARRAY [0..Number of elements-1] OF Data type. It is represented ARRAY [0..2] OF ANY16 when the array is of ANY16 type and has three elements.)
7) Indicates the list (return value name, description) of return values used with the function.
8) Indicates the example of using the function. (Indicates the example that uses the actual device/label.)
9) Indicates the QCPU (Q mode)/LCPU MELSEC command corresponding to the function.

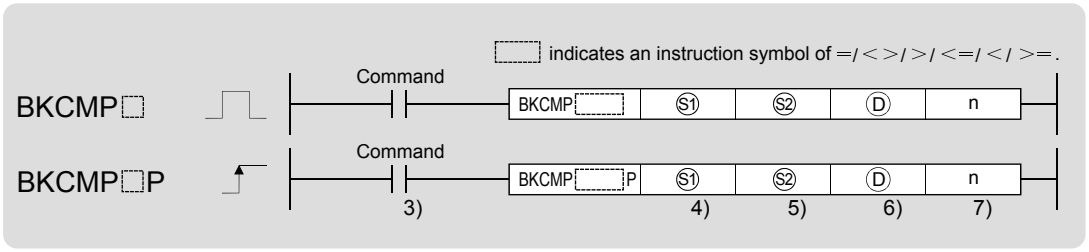
5 - 1

The following indicates the correspondences between the MELSEC command in the "MELSEC-Q/L Programming Manual (Common Instructions)" and the MELSEC function in this manual.

MELSEC-Q/L Programming Manual (Common Instructions) [MELSEC instruction]

6.1.6 BIN block data comparisons (BKCMP□, BKCMP□P)

Basic | High performance | Process | Redundant | Universal | LCPU → 1)



(S1) : Data to be compared or head number of the devices where the data to be compared is stored (BIN 16 bits)

(S2) : Head number of the devices where the comparison data is stored (BIN 16 bits)

(D) : Head number of the devices where the comparison operation result will be stored (bits)

n : Number of comparison data blocks (BIN 16 bits)

Setting Data	Internal Devices		R, ZR	J□\□		U□\G□	Zn	Constants K, H	Other
	Bit	Word		Bit	Word				
(S1)	—	○		—				○	—
(S2)	—	○		—				—	—
(D)	○	○		—				—	—
n	○	○		○				○	—

→ 2)

[MELSEC function] in this manual

5.4.1 Block data comparison (=) BKCMP_EQ_M

n points of BIN 16-bit data (word unit), starting at the specified devices, are compared in terms of "=".

■ Function definition BOOL BKCMP_EQ_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, BOOL D);

8) 9) 10) 11) 12)

5 - 2

1) Applicable CPU types
CPU types that can use the instructions are indicated.
2) Applicable devices
• The correspondences between the arguments of the MELSEC function and MELSEC command are as follows. (The arguments of the same argument names correspond to each other.)

3) ↔ 8)	4) ↔ 9)	5) ↔ 10)	7) ↔ 11)	6) ↔ 12)

Point

● Precaution for use of the arguments of the MELSEC and IEC functions
When the argument is of ANY32 type, the data type that can be specified is the DIN type, and therefore, an actual device cannot be specified. Only the double word type label can be specified. However, digit specification is allowed.

Example: BSQR_MD(BOOL EN, ANY16 s, ANY32 d);
(* Function definition of BSQR_MD *)
BSQR_MD (X0, D0, dData); (* Program example *)
In the MELSEC common command, an actual device can be described as indicated below.
BSQR(D0, W0);
However, it cannot be described in the MELSEC/IEC function.
BSQR_MD(X0, D0, W0); ← An error will occur.

When the argument is of REAL type, the data type that can be specified is the real number type label, or a real number value can be described directly.

An actual device cannot be specified.

Example: ESTR_M(BOOL EN, REAL s1, ANY16(3) s2, STRING d);
(* Function definition of ESTR_M *)
ESTR_M(X0, rData, ArrayData, sData);
(* Program example *)
In the MELSEC common command, the actual device can be described as indicated below.
ESTR(R0, R10, D10);
However, it cannot be described in the MELSEC/IEC function.
ESTR_M(X0, R0, ArrayData, sData); ← An error will occur.

5 - 3

5.1 Output

5.1.1 Output to device OUT_M

The execution condition is output to the specified device.

■ Function definition BOOL OUT_M (BOOL EN, BOOL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition
D	OUT	Target to be turned ON/OFF

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* Execution condition X0 is output to the assigned device of bData.  *)
 OUT_M (X0, bData);
```

● Corresponding MELSEC command
- OUT (Output)

5.1.2 Low-speed timer TIMER_M

When the coil of the timer (low-speed timer, low-speed retentive timer) turns ON, the timer measures up to the set value, and when the timer times out (calculation value (set value), the contact is put in the following status.
N/O contact: Conduction N/C contact: Non-conduction

■ Function definition BOOL TIMER_M (BOOL EN, BOOL TCoil, ANY16 TValue);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
TCoil	IN	TS, TC device or STS, STC device (bit data)
TValue	IN	Timer set value (BIN 16-bit data)

Remarks: When a constant is specified for the timer set value, only a decimal number can be specified.
The timer set value can be specified within the range 0 to 32767.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, TC0 turns ON and the timer calculates  *)
(* up to TValue, and when the timer times out (calculation value ( set value),  *)
(* the contact is put in the following status.                                   *)
(* N/O contact: Conduction N/C contact: Non-conduction                           *)
 TIMER_M (X0, TC0, TValue);
```

● Corresponding MELSEC command
- OUT T (Low-speed timer)

For the usable data type, refer to "3.2.2 About ANY type".

5.1.3 High-speed timer TIMER_H_M

When the coil of the timer (high-speed timer, high-speed retentive timer) turns ON, the timer calculates up to the set value, and when the timer times out (calculation value (set value), the contact is put in the following status.
N/O contact: Conduction N/C contact: Non-conduction

■ Function definition BOOL TIMER_H_M (BOOL EN, BOOL TCoil, ANY16 TValue);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
TCoil	IN	TS, TC device or STS, STC device (bit data)
TValue	IN	Timer set value (BIN 16-bit data)

Remarks: When a constant is specified for the timer set value, only a decimal number can be specified.
The timer set value can be specified within the range 0 to 32767.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, TC0 turns ON and the timer calculates*)
(* up to TValue, and when the timer times out (calculation value( TValue), the  *)
(* contact is put in the following status.                                       *)
(* N/O contact: Conduction N/C contact: Non-conduction                           *)
   TIMER_H_M (X0, TC0, TValue);
```

● Corresponding MELSEC command
・OUTH T (High-speed timer)

5.1.4 Counter COUNTER_M

The present value (count value) of the counter is incremented by 1, and when the counter counts up (present value = set value), the contact is put in the following status.
N/O contact: Conduction N/C contact: Non-conduction

■ Function definition BOOL COUNTER_M (BOOL EN, BOOL CCoil, ANY16 CValue);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
CCoil	IN	CS, CC device number (bit data)
CValue	IN	Counter set value (BIN 16-bit data)

Remarks: When a constant is specified for the counter set value, only a decimal number can be specified.
The timer set value can be specified within the range 0 to 32767.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* After execution condition X0 has turned ON, the present value (count value) is *)
(* incremented by 1 when CC0 changes from OFF to ON, and when the counter  *)
(* counts up (present value = CValue), the contact is put in the following status.  *)
(* N/O contact: Conduction N/C contact: Non-conduction                              *)
   COUNTER_M (X0, CC0, CValue);
```

● Corresponding MELSEC command
・OUT C (Counter)

For the usable data type, refer to "3.2.2 About ANY type".

5.1.5 Set of device SET_M

When the execution condition is satisfied, the specified device is operated as described below.

- Bit device: The coil/contact is turned ON.
- When bit of word device is specified: The specified bit is turned to 1.

■ Function definition BOOL SET_M (BOOL EN, BOOL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	OUT	Data to be set

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the assigned device of bData is  *)
(* turned ON.                                                              *)
  SET_M (X0, bData);
```

● Corresponding MELSEC command

- SET (Set of device)

5.1.6 Reset of device RST_M

When the execution condition is satisfied, the specified device is operated as described below.

- Bit device: The coil/contact is turned OFF.
- Timer, counter: 0 is assigned to the present value and the coil/contact is turned OFF.
- When bit of word device is specified: The specified bit is turned to 0.
- Word device other than timer and counter: 0 is assigned to the device data.

■ Function definition BOOL RST_M (BOOL EN, BOOL ANY_SIMPLE D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	OUT	Data to be reset

Remarks: The DINT/REAL/STRING type cannot be used in argument "D".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the assigned device of bData is  *)
(* turned OFF.                                                             *)
  RST_M (X0, bData);
```

● Corresponding MELSEC command

- RST (Reset of device)

For the usable data type, refer to "3.2.2 About ANY type".

5.1.7 Conversion of direct output into pulse DELTA_M

When the execution condition is satisfied, the specified direct access output (DY) is output as a pulse.

■ Function definition BOOL DELTA_M (BOOL EN, BOOL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	OUT	Data to be output as pulse (DY device)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(*When execution condition X0 turns ON, device DY0 is converted into pulse.  *)
  DELTA_M (X0, DY0);
```

● Corresponding MELSEC command

・DELTA (Conversion of direct output into pulse)

5 - 7

5.2 1-Bit Shift

5.2.1 1-bit shift of device SFT_M

When the execution condition is satisfied, the specified device is operated as described below.

- In the case of bit device:
  The ON/OFF status of the device number preceding the specified device number is shifted to the specified device number, and the preceding device number is turned OFF.
- In the case of word device bit specification:
  The 1/0 status of the bit preceding the bit of the specified device is shifted to the specified bit, and the preceding device number is turned to 0.

■ Function definition BOOL SFT_M (BOOL EN, BOOL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	OUT	Data to be shifted

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, ON/OFF of M10 is shifted to M11   *)
(* and M10 is turned OFF.                                                   *)
   SFT_M (X0, M11);
(* When execution condition X0 turns ON, ON/OFF of W100.1 is shifted to     *)
(* W100.2 and W100.1 is turned OFF.                                         *)
   SFT_M (X0, W100.2);
```

● Corresponding MELSEC command
- SFT (Bit device shift)

5.3 Termination

5.3.1 Stop STOP_M

When the execution condition is satisfied, output Y is reset and the CPU operation is stopped.
(This operation is the same as performed when the RUN/STOP DIP switch is moved to the STOP position.)

■ Function definition BOOL STOP_M (BOOL EN);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the CPU operation is stopped.  *)
  STOP_M (X0);
```

● Corresponding MELSEC command
・STOP (Sequence program stop)

5.4 Comparison Operation

5.4.1 Block data comparison (=) BKCMP_EQ_M

n points of BIN 16-bit data (word unit), starting at the specified devices, are compared in terms of "=".

■ Function definition BOOL BKCMP_EQ_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, BOOL D);

Argument Name	IN/OUT	Description			
EN	IN	Execution condition (Function is executed only when the result is TRUE)			
S1	IN	Compared data (BIN 16-bit data)			
S2	IN	Comparison data (BIN 16-bit data)			
n	IN	Number of data to be compared (BIN 16-bit data)			
D	OUT	Comparison result (bit)	Comparison result	When comparison condition is satisfied	ON
				When comparison condition is not satisfied	OFF

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of points stored  *)
(* in D0, starting at D100, is compared with the data of the number of points stored *)
(* in D0, starting at D200, in terms of "=", and the result stored into M0 and later.  *)
  BKCMP_EQ_M (X0, D100, D200, D0, M0);
```

● Corresponding MELSEC command

・BKCMP= (BIN block data comparison (=))

5.4.2 Block data comparison (<>) BKCMP_NE_M

n points of BIN 16-bit data (word unit), starting at the specified devices, are compared in terms of "<>".

■ Function definition BOOL BKCMP_NE_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, BOOL D);

Argument Name	IN/OUT	Description			
EN	IN	Execution condition (Function is executed only when the result is TRUE)			
S1	IN	Compared data (BIN 16-bit data)			
S2	IN	Comparison data (BIN 16-bit data)			
n	IN	Number of data to be compared (BIN 16-bit data)			
D	OUT	Comparison result (bit)	Comparison result	When comparison condition is satisfied	ON
				When comparison condition is not satisfied	OFF

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of points stored in  *)
(* D0, starting at D100, is compared with the data of the number of points stored     *)
(* D0, starting at D200, in terms of "<>", and the result is stored into M0 and later.  *)
  BKCMP_NE_M (X0, D100, D200, D0 M0);
```

● Corresponding MELSEC command

・BKCMP<> (BIN block data comparison (<>))

For the usable data type, refer to "3.2.2 About ANY type".

5.4.3 Block data comparison (>) BKCMP_GT_M

n points of BIN 16-bit data (word unit), starting at the specified devices, are compared in terms of ">".

■ Function definition BOOL BKCMP_GT_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, BOOL D);

Argument Name	IN/OUT	Description			
EN	IN	Execution condition (Function is executed only when the result is TRUE)			
S1	IN	Compared data (BIN 16-bit data)			
S2	IN	Comparison data (BIN 16-bit data)			
n	IN	Number of data to be compared (BIN 16-bit data)			
D	OUT	Comparison result (bit)	Comparison result	When comparison condition is satisfied	ON
				When comparison condition is not satisfied	OFF

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of points stored  *)
(* in D0, starting at D100, is compared with the data of the number of points stored *)
(* in D0, starting at D200, in terms of ">", and the result is stored into M0 and later. *)
  BKCMP_GT_M (X0, D100, D200, D0, M0);
```

● Corresponding MELSEC command

・BKCMP> (BIN block data comparison (>))

5.4.4 Block data comparison (<=) BKCMP_LE_M

n points of BIN 16-bit data (word unit), starting at the specified devices, are compared in terms of "<=".

■ Function definition BOOL BKCMP_LE_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, BOOL D);

Argument Name	IN/OUT	Description			
EN	IN	Execution condition (Function is executed only when the result is TRUE)			
S1	IN	Compared data (BIN 16-bit data)			
S2	IN	Comparison data (BIN 16-bit data)			
n	IN	Number of data to be compared (BIN 16-bit data)			
D	OUT	Comparison result (bit)	Comparison result	When comparison condition is satisfied	ON
				When comparison condition is not satisfied	OFF

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of points stored  *)
(* in D0, starting at D100, is compared with the data of the number of points stored *)
(* in D0, starting at D200, in terms of "<=", and the result is stored into M0 and later.  *)
  BKCMP_LE_M (X0, D100, D200, D0, M0);
```

● Corresponding MELSEC command

・BKCMP<= (BIN block data comparison (<=))

For the usable data type, refer to "3.2.2 About ANY type".

5.4.5 Block data comparison (<) BKCMP_LT_M

n points of BIN 16-bit data (word unit), starting at the specified devices, are compared in terms of "<".

■ Function definition BOOL BKCMP_LT_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, BOOL D);

Argument Name	IN/OUT	Description			
EN	IN	Execution condition (Function is executed only when the result is TRUE)			
S1	IN	Compared data (BIN 16-bit data)			
S2	IN	Comparison data (BIN 16-bit data)			
n	IN	Number of data to be compared (BIN 16-bit data)			
D	OUT	Comparison result (bit)	Comparison result	When comparison condition is satisfied	ON
				When comparison condition is not satisfied	OFF

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of points stored  *)
(* in D0, starting at D100, is compared with the data of the number of points stored *)
(* in D0, starting at D200, in terms of "<", and the result is stored into M0 and later. *)
  BKCMP_LT_M (X0, D100, D200, D0, M0);
```

● Corresponding MELSEC command

・BKCMP< (BIN block data comparison (<))

5.4.6 Block data comparison (>=) BKCMP_GE_M

n points of BIN 16-bit data (word unit), starting at the specified devices, are compared in terms of ">=".

■ Function definition BOOL BKCMP_GE_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, BOOL D);

Argument Name	IN/OUT	Description			
EN	IN	Execution condition (Function is executed only when the result is TRUE)			
S1	IN	Compared data (BIN 16-bit data)			
S2	IN	Comparison data (BIN 16-bit data)			
n	IN	Number of data to be compared (BIN 16-bit data)			
D	OUT	Comparison result (bit)	Comparison result	When comparison condition is satisfied	ON
				When comparison condition is not satisfied	OFF

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of points stored  *)
(* in D0, starting at D100, is compared with the data of the number of points stored *)
(* in D0, starting at D200, in terms of ">=", and the result is stored into M0 and later. *)
  BKCMP_GE_M (X0, D100, D200, D0, M0);
```

● Corresponding MELSEC command

・BKCMP>= (BIN block data comparison (>=))

For the usable data type, refer to "3.2.2 About ANY type".

5.5 Arithmetic Operation

5.5.1 Addition of BCD 4-digit data (2 devices) BPLUS_M

The specified two BCD 4-digit data are added.

■ Function definition BOOL BPLUS_M (BOOL EN, ANY16 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Addend data (BCD 4-digit data)
D	IN/OUT	Augend data, addition result (BCD 4-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BCD 4-digit data stored in D0      *)
(* and D100 are added, and the addition result is stored into D100.              *)
   BPLUS_M (X0, D0, D100);
```

● Corresponding MELSEC command

・B+ (BCD 4-digit data addition)

5.5.2 Addition of BCD 4-digit data (3 devices) BPLUS_3_M

The specified two BCD 4-digit data are added.

■ Function definition BOOL BPLUS_3_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Augend data (BCD 4-digit data)
S2	IN	Addend data (BCD 4-digit data)
D	OUT	Addition result (BCD 4-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BCD 4-digit data stored in D1      *)
(* and D2 are added, and the addition result is stored into D100.                *)
   BPLUS_3_M (X0, D1, D2, D100);
```

● Corresponding MELSEC command

・B+ (BCD 4-digit data addition)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 13

5.5.3 Subtraction of BCD 4-digit data (2 devices) BMINUS_M

Subtraction is performed between the specified two BCD 4-digit data.

■ Function definition BOOL BMINUS_M (BOOL EN, ANY16 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Subtrahend data (BCD 4-digit data)
D	IN/OUT	Minuend data, subtraction result (BCD 4-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, subtraction is performed between the  *)
(* BCD 4-digit data stored in D0 and D100, and the subtraction result is stored  *)
(* into D100.                                                                    *)
  BMINUS_M (X0, D0, D100);
```

● Corresponding MELSEC command

・B- (BCD 4-digit data subtraction)

5.5.4 Subtraction of BCD 4-digit data (3 devices) BMINUS_3_M

Subtraction is performed between the specified two BCD 4-digit data.

■ Function definition BMINUS_3_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Minuend data (BCD 4-digit data)
S2	IN	Subtrahend data (BCD 4-digit data)
D	OUT	Subtraction result (BCD 4-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, subtraction is performed between the  *)
(* BCD 4-digit data stored in D1 and D2, and the subtraction result is stored   *)
(* into D100.                                                                    *)
  BMINUS_3_M (X0, D1, D2, D100);
```

● Corresponding MELSEC command

・B- (BCD 4-digit data subtraction)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 14

5.5.5 Addition of BCD 8-digit data (2 devices) DBPLUS_M

The specified two BCD 8-digit data are added.

■ Function definition BOOL DBPLUS_M (BOOL EN, ANY16 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Addend data (BCD 8-digit data)
D	IN/OUT	Augend data, addition result (BCD 8-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BCD 8-digit data stored in      *)
(* dwData1 and Result are added, and the addition result is stored into Result. *)
   DBPLUS_M (X0, dwData1, Result);
```

● Corresponding MELSEC command

・DB+ (BCD 8-digit data addition)

5.5.6 Addition of BCD 8-digit data (3 devices) DBPLUS_3_M

The specified two BCD 8-digit data are added.

■ Function definition DBPLUS_3_M (BOOL EN, ANY32 S1, ANY32 S2, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Augend data (BCD 8-digit data)
S2	IN	Addend data (BCD 8-digit data)
D	OUT	Addition result (BCD 8-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BCD 8-digit data stored in      *)
(* dwData1 and dwData2 are added, and the addition result is stored into Result. *)
   DBPLUS_3_M (X0, dwData1, dwData2, Result);
```

● Corresponding MELSEC command

・DB+ (BCD 8-digit data addition)

For the usable data type, refer to "3.2.2 About ANY type".

5.5.7 Subtraction of BCD 8-digit data (2 devices) DBMINUS_M

Subtraction is performed between the specified two BCD 8-digit data.

■ Function definition BOOL DBMINUS_M (BOOL EN, ANY32 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Subtrahend data (BCD 8-digit data)
D	IN/OUT	Minuend data, subtraction result (BCD 8-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, subtraction is performed between the *)
(* BCD 8-digit data stored in dwData1 and Result, and the subtraction result is *)
(* stored into Result. *)
  DBMINUS_M (X0, dwData1, Result);
```

● Corresponding MELSEC command

・DB- (BCD 8-digit data subtraction)

5.5.8 Subtraction of BCD 8-digit data (3 devices) DBMINUS_3_M

Subtraction is performed between the specified two BCD 8-digit data.

■ Function definition BOOL DBMINUS_3_M (BOOL EN, ANY32 S1, ANY32 S2, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Minuend data (BCD 8-digit data)
S2	IN	Subtrahend data (BCD 8-digit data)
D	OUT	Subtraction result (BCD 8-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, subtraction is performed between the *)
(* BCD 8-digit data stored in dwData1 and dwData2, and the subtraction result *)
(* is stored into Result. *)
  DBMINUS_3_M (X0, dwData1, dwData2, Result);
```

● Corresponding MELSEC command

・DB- (BCD 8-digit data subtraction)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 16

5.5.9 Multiplication of BCD 4-digit data BMULTI_M

The specified two BCD 4-digit data are multiplied.

■ Function definition BOOL BMULTI_M (BOOL EN, ANY16 S1, ANY16 S2, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Multiplicand data (BCD 4-digit data)
S2	IN	Multiplier data (BCD 4-digit data)
D	OUT	Multiplication result (BCD 4-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BCD 4-digit data stored in D1   *)
(* and D2 are multiplied, and the multiplication result is stored into Result. *)
  BMULTI_M (X0, D1, D2, Result);
```

● Corresponding MELSEC command

- B* (BCD 4-digit data multiplication)

5.5.10 Division of BCD 4-digit data BDIVID_M

Division is performed between the specified two BCD 4-digit data.

■ Function definition BOOL BDIVID_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16(2) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Dividend data (BCD 4-digit data)		
S2	IN	Divisor data (BCD 4-digit data)		
D	OUT	Division result (ARRAY [0..1] OF ANY16)	D[0]	Quotient
			D[1]	Remainder

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, division is performed between the   *)
(* BCD 4-digit data stored in D1 and D2, and the division result is stored into *)
(* array ArrayResult.                                                         *)
  BDIVID_M (X0, D1, D2, ArrayResult);
```

● Corresponding MELSEC command

- B/ (BCD 4-digit data division)

For the usable data type, refer to "3.2.2 About ANY type".

5.5.11 Multiplication of BCD 8-digit data DBMULTI_M

The specified two BCD 8-digit data are multiplied.

■ Function definition BOOL DBMULTI_M (BOOL EN, ANY32 S1, ANY32 S2, ANY16(4) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Multiplicand data (BCD 8-digit data)		
S2	IN	Multiplier data (BCD 8-digit data)		
D	OUT	Multiplication result (ARRAY [0..3] OF ANY16)	D[0]	Lower 4 digits ↓ Upper 4 digits
			D[1]	
			D[2]	
			D[3]	

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BCD 8-digit data stored in      *)
(* dwData1 and dwData2 are multiplied, and the multiplication result is stored *)
(* into array ArrayResult.                                                    *)
   DBMULTI_M (X0, dwData1, dwData2, ArrayResult);
```

● Corresponding MELSEC command

・DB* (BCD 8-digit data multiplication)

5.5.12 Division of BCD 8-digit data DBDIVID_M

Division is performed between the specified two BCD 8-digit data.

■ Function definition BOOL DBDIVID_M (BOOL EN, ANY32 S1, ANY32 S2, ANY32(2) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Dividend data (BCD 8-digit data)		
S2	IN	Divisor data (BCD 8-digit data)		
D	OUT	Division result (ARRAY [0..1] OF ANY32)	D[0]	Quotient
			D[1]	Remainder

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, division is performed between the  *)
(* BCD 8-digit data stored in dwData1 and dwData2, and the division result is *)
(* stored into array ArrayResult.                                            *)
   DBDIVID_M (X0, dwData1, dwData2, ArrayResult);
```

● Corresponding MELSEC command

・DB/ (BCD 8-digit data division)

For the usable data type, refer to "3.2.2 About ANY type".

5.5.13 Character string data connection (2 devices) STRING_PLUS_M

The specified character string data are connected.

■ Function definition BOOL STRING_PLUS_M (BOOL EN, STRING S1, STRING D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to connect (character string data)
D	IN/OUT	Data to be connected, connection result (character string data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, character string "ABC" is connected   *)
(* to the end of the character string stored in StrResult and the connected     *)
(* character strings are stored into StrResult.                                 *)
  STRING_PLUS_M (X0, "ABC" StrResult);
```

● Corresponding MELSEC command

・$+ (Character string connection)

5.5.14 Character string data connection (3 devices) STRING_PLUS_3_M

The specified character string data are connected.

■ Function definition BOOL STRING_PLUS_3_M (BOOL EN, STRING S1, STRING S2, STRING D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be connected (character string data)
S2	IN	Data to connect (character string data)
D	OUT	Connection result (character string data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string stored in        *)
(* StrData2 is connected to the end of the character string stored in StrData1  *)
(* and the connected character strings are stored into StrResult.               *)
  STRING_PLUS_3_M (X0, StrData1, StrData2, StrResult);
```

● Corresponding MELSEC command

・$+ (Character string connection)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 19

5.5.15 BIN block addition BKPLUS_M

n points of BIN 16-bit data, starting at the specified devices, are added.

■ Function definition BOOL BKPLUS_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Augend data (BIN 16-bit data)
S2	IN	Addend data (BIN 16-bit data)
n	IN	Number of addition data (BIN 16-bit data)
D	OUT	Addition result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of points      *)
(* stored in D0, starting at D100, and the data of the number of points stored in  *)
(* D0, starting at D200, are added, and the result is stored into D1000 and later.  *)
  BKPLUS_M (X0, D100, D200, D0, D1000);
```

● Corresponding MELSEC command

・BK+ (Block data addition)

5.5.16 BIN block subtraction BKMINUS_M

Subtraction is performed between n points of BIN 16-bit data, starting at the specified devices.

■ Function definition BOOL BKMINUS_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Minuend data (BIN 16-bit data)
S2	IN	Subtrahend data (BIN 16-bit data)
n	IN	Number of subtraction data (BIN 16-bit data)
D	OUT	Subtraction result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, subtraction is performed between      *)
(* the data of the number of points stored in D0, starting at D100, and the data  *)
(* of the number of points stored in D0, starting at D200, and the result is      *)
(* stored into D1000 and later.                                                   *)
  BKMINUS_M (X0, D100, D200, D0, D1000);
```

● Corresponding MELSEC command

・BK- (Block data subtraction)

For the usable data type, refer to "3.2.2 About ANY type".

5.5.17 Increment INC_M

The specified BIN 16-bit data is incremented (by 1).

■ Function definition BOOL INC_M (BOOL EN, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	IN/OUT	Increment data, incrementing result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data stored in D0 is incremented by 1. *)
  INC_M (X0, D0);
```

● Corresponding MELSEC command

・INC (BIN 16-bit increment)

5.5.18 Decrement DEC_M

The specified BIN 16-bit data is decremented (by 1).

■ Function definition BOOL DEC_M (BOOL EN, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	IN/OUT	Decrement data, decrementing result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data stored in D0 is decremented by 1. *)
  DEC_M (X0, D0);
```

● Corresponding MELSEC command

・DEC (BIN 16-bit decrement)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 21

5.5.19 32-bit BIN increment DINC_M

The specified BIN 32-bit data is incremented (by 1).

■ Function definition BOOL DINC_M (BOOL EN, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	IN/OUT	Increment data, incrementing result (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data stored in dwData1 is  *)
(* incremented by 1.                                                     *)
  DINC_M (X0, dwData1);
```

● Corresponding MELSEC command

・DINC (BIN 32-bit increment)

5.5.20 32-bit BIN decrement DDEC_M

The specified BIN 32-bit data is decremented (by 1).

■ Function definition BOOL DDEC_M (BOOL EN, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	IN/OUT	Decrement data, decrementing result (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data stored in dwData1 is  *)
(* decremented by 1.                                                     *)
  DDEC_M (X0, dwData1);
```

● Corresponding MELSEC command

・DDEC (BIN 32-bit decrement)

For the usable data type, refer to "3.2.2 About ANY type".

5.6 Data Conversion

5.6.1 BIN→BCD conversion BCD_M

The specified BIN 16-bit data (0 to 9999) is converted into BCD 4-digit data.

■ Function definition BOOL BCD_M (BOOL EN, ANY16 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 16-bit data)
D	OUT	Conversion result (BCD 4-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BIN data stored in D0 is      *)
(* converted into BCD, and the result is stored into D100.                 *)
  BCD_M (X0, D0, D100);
```

● Corresponding MELSEC command

- BCD (Conversion from BIN data to 4-digit BCD data)

5.6.2 32-bit BIN→BCD conversion DBCD_M

The specified BIN 32-bit data (0 to 99999999) is converted into BCD 8-digit data.

■ Function definition BOOL DBCD_M (BOOL EN, ANY32 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 32-bit data)
D	OUT	Conversion result (BCD 8-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BIN data stored in dwData1 is *)
(* converted into BCD, and the result is stored into Result.               *)
  DBCD_M (X0, dwData1, Result);
```

● Corresponding MELSEC command

- DBCD (Conversion from BIN data to 8-digit BCD data)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 23

5.6.3 BCD→BIN conversion BIN_M

The specified BCD 4-digit data (0 to 9999) is converted into BIN 16-bit data.

■ Function definition BOOL BIN _M (BOOL EN, ANY16 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BCD 4-digit data)
D	OUT	Conversion result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BCD data stored in D0 is      *)
(* converted into BIN, and the result is stored into D100.                 *)
  BIN_M (X0, D0, D100);
```

● Corresponding MELSEC command

・BIN (Conversion from BCD 4-digit data to BIN data)

5.6.4 32-bit BCD→BIN conversion DBIN_M

The specified BCD 8-digit data (0 to 99999999) is converted into BIN 32-bit data.

■ Function definition DBIN_M (BOOL EN, ANY32 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BCD 8-digit data)
D	OUT	Conversion result (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BCD data stored in dwData1 is  *)
(* converted into BIN, and the result is stored into Result.               *)
  DBIN_M (X0, dwData1, Result);
```

● Corresponding MELSEC command

・DBIN (Conversion from BCD 8-digit data to BIN data)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 24

5.6.5 Floating-point→BIN conversion INT_E_MD

The specified real number data is converted into BIN 16-bit data.

■ Function definition BOOL INT_E_MD (BOOL EN, REAL S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (real number data)
D	OUT	Conversion result (BIN 16-bit data)

Remarks: The real number data specified in argument "S1" can be specified within the range -32768 to 32767.
The data after conversion is the value obtained by rounding off the real number in the first decimal place.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the real number data in RealData1 is  *)
(* converted into BIN 16-bit data, and the result is stored into D0.            *)
  INT_E_MD (X0, RealData1, D0);
```

● Corresponding MELSEC command

・INT (Conversion from floating decimal point data to BIN16-bit data (Single precision))

5.6.6 32-bit floating-point→BIN conversion DINT_E_MD

The specified real number data is converted into BIN 32-bit data.

■ Function definition BOOL DINT_E_MD (BOOL EN, REAL S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (real number data)
D	OUT	Conversion result (BIN 32-bit data)

Remarks: The real number data specified in argument "S1" can be specified within the range -2147483648 to 2147483647.
The data after conversion is the value obtained by rounding off the real number in the first decimal place.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, real number data E2.6 is converted    *)
(* into BIN 32-bit data, and the result is stored into Result.                  *)
  DINT_E_MD (X0, E2.6, Result);
```

● Corresponding MELSEC command

・DINT (Conversion from floating decimal point data to BIN32-bit data (Single precision))

For the usable data type, refer to "3.2.2 About ANY type".

5.6.7 BIN→floating-point conversion FLT_M

The specified BIN 16-bit data is converted into real number data.

■ Function definition BOOL FLT_M (BOOL EN, ANY16 S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 16-bit data)
D	OUT	Conversion result (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(*When execution condition X0 turns ON, the BIN 16-bit data in D100 is      *)
(* converted into real number data, and the result is stored into Result.    *)
  FLT_M (X0, D100, Result);
```

● Corresponding MELSEC command

・FLT (Conversion from BIN 16-bit data to floating decimal point (Single precision))

5.6.8 32-bit BIN→floating-point conversion DFLT_M

The specified BIN 32-bit data is converted into real number data.

■ Function definition BOOL DFLT_M (BOOL EN, ANY32 S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 32-bit data)
D	OUT	Conversion result (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BIN 32-bit data in dwData1 is   *)
(* converted into real number data, and the result is stored into Result.    *)
  DFLT_M (X0, dwData1, RealResult);
```

● Corresponding MELSEC command

・DFLT (Conversion from BIN 32-bit data to floating decimal point (Single precision))

For the usable data type, refer to "3.2.2 About ANY type".

5 - 26

5.6.9 16-bit BIN→32-bit BIN conversion DBL_M

The specified BIN 16-bit data is converted into signed BIN 32-bit data.

■ Function definition BOOL DBL_M (BOOL EN, ANY16 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 16-bit data)
D	OUT	Conversion result (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BIN 16-bit data in D0 is      *)
(* converted into signed BIN 32-bit data, and the result is stored into Result. *)
  DBL_M (X0, D0, Result);
```

● Corresponding MELSEC command

・DBL (Conversion from BIN 16-bit to BIN 32-bit data)

5.6.10 32-bit BIN→16-bit BIN conversion WORD_M

The specified BIN 32-bit data is converted into signed BIN 16-bit data.

■ Function definition BOOL WORD_M (BOOL EN, ANY32 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 32-bit data)
D	OUT	Conversion result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BIN 32-bit data stored in dwData1 *)
(* is converted into signed BIN 16-bit data, and the result is stored into D0.  *)
  WORD_M (X0, dwData1, D0);
```

● Corresponding MELSEC command

・WORD (Conversion from BIN 32-bit to BIN 16-bit data)

For the usable data type, refer to "3.2.2 About ANY type".

5.6.11 BIN→gray code conversion GRY_M

The specified BIN 16-bit data is converted into gray code 16-bit data.

■ Function definition BOOL GRY_M (BOOL EN, ANY16 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 16-bit data)
D	OUT	Conversion result (gray code 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BIN 16-bit data in D0 is      *)
(* converted into gray code 16-bit data, and the result is stored into D100. *)
  GRY_M (X0, D0, D100);
```

● Corresponding MELSEC command

・GRY (Conversion from BIN 16-bit data to Gray code)

5.6.12 32-bit BIN→gray code conversion DGRY_M

The specified BIN 32-bit data is converted into gray code 32-bit data.

■ Function definition BOOL DGRY_M (BOOL EN, ANY32 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 32-bit data)
D	OUT	Conversion result (gray code 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BIN 32-bit data in dwData1 is  *)
(* converted into gray code 32-bit data, and the result is stored into Result. *)
  DGRY_M (X0, dwData1 Result);
```

● Corresponding MELSEC command

・DGRY (Conversion from BIN 32-bit data to Gray code)

For the usable data type, refer to "3.2.2 About ANY type".

5.6.13 Gray code→BIN conversion GBIN_M

The specified gray code 16-bit data is converted into BIN 16-bit data.

■ Function definition BOOL GBIN_M (BOOL EN, ANY16 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (gray code 16-bit data)
D	OUT	Conversion result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the gray code 16-bit data in D100 is  *)
(* converted into BIN 16-bit data, and the result is stored into D200.          *)
  GBIN_M (X0, D100, D200);
```

● Corresponding MELSEC command

・GBIN (Conversion of Gray code to BIN 16-bit data)

5.6.14 32-bit gray code→BIN conversion DGBIN_M

The specified gray code 32-bit data is converted into BIN 32-bit data.

■ Function definition BOOL DGBIN_M (BOOL EN, ANY32 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (gray code 32-bit data)
D	OUT	Conversion result (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the gray code 32-bit data in dwData1  *)
(* is converted into BIN 32-bit data, and the result is stored into Result.     *)
  DGBIN_M (X0, dwData1, Result);
```

● Corresponding MELSEC command

・DGBIN (Conversion of Gray code to BIN 32-bit data)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 29

5.6.15 2' complement of 16-bit BIN NEG_M

The sign of the specified BIN 16-bit data is inverted. (2's complement)

■ Function definition BOOL NEG_M (BOOL EN, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	IN/OUT	Data whose sign will be inverted, sign inversion result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the sign of the BIN 16-bit data in D0  *)
(* is inverted, and the result is stored into D0.                                *)
  NEG_M (X0, D0);
```

● Corresponding MELSEC command

・NEG (Complement of 2 of BIN 16-bit data (sign reversal))

5.6.16 2' complement of 32-bit BIN DNEG_M

The sign of the specified BIN 32-bit data is inverted. (2's complement)

■ Function definition BOOL DNEG_M (BOOL EN, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	IN/OUT	Data whose sign will be inverted, sign inversion result (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the sign of the BIN 32-bit data in  *)
(* Result is inverted, and the result is stored into Result.                  *)
  DNEG_M (X0, Result);
```

● Corresponding MELSEC command

・DNEG (Complement of 2 of BIN 32-bit data (sign reversal))

For the usable data type, refer to "3.2.2 About ANY type".

5.6.17 2' complement of floating-point ENEG_M

The sign of the specified real number data is inverted. (2's complement)

■ Function definition BOOL ENEG_M (BOOL EN, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	IN/OUT	Data whose sign will be inverted, sign inversion result (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the sign of the real number data in  *)
(* Result is inverted, and the result is stored into Result.                   *)
  ENEG_M (X0, Result);
```

● Corresponding MELSEC command

- ENEG (Floating-point sign inversion (Single precision))

5.6.18 Block BIN→BCD conversion BKBCD_M

n points of BIN 16-bit data (0 to 9999), starting at the specified device, is converted into BCD 4-digit data.

■ Function definition BOOL BKBCD_M (BOOL EN, ANY16 S1, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 16-bit data)
n	IN	Number of converted data
D	OUT	Conversion result (BCD 4-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BIN 16-bit data of the number  *)
(* of points stored in W0, starting at D0, is converted into BCD, and the result is  *)
(* stored into D100 and later.                                                *)
  BKBCD_M (X0, D0, W0, D100);
```

● Corresponding MELSEC command

- BKBCD (Conversion from block BIN 16-bit data to BCD 4-digit data)

For the usable data type, refer to "3.2.2 About ANY type".

5.6.19 Block BCD→BIN conversion BKBIN_M

n points of BCD 4-digit data (0 to 9999), starting at the specified device, is converted into BIN 16-bit data.

■ Function definition BOOL BKBIN_M (BOOL EN, ANY16 S1, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BCD 4-digit data)
n	IN	Number of converted data
D	OUT	Conversion result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BCD data of the number of points *)
(* stored in W0, starting at D0, is converted into BIN, and the result is stored into *)
(* D100 and later. *)
  BKBIN_M (X0, D0, W0, D100);
```

● Corresponding MELSEC command

・BKBIN (Conversion from block BCD 4-digit data to block BIN 16-bit data)

For the usable data type, refer to "3.2.2 About ANY type".

5.7 Data Transfer

5.7.1 16-bit data NOT transfer CML_M

The specified BIN 16-bit data are inverted bit by bit.

■ Function definition BOOL CML_M (BOOL EN, ANY16 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data whose bits will be inverted (BIN 16-bit data)
D	OUT	Inversion result transfer destination (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of M0 to M7 are inverted,  *)
(* and the result is transferred to D0.                                       *)
  CML_M (X0, K2M0, D0);
```

● Corresponding MELSEC command

・CML (16-bit NOT transfer)

5.7.2 32-bit data NOT transfer DCML_M

The specified BIN 32-bit data are inverted bit by bit.

■ Function definition BOOL DCML_M (BOOL EN, ANY32 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data whose bits will be inverted (BIN 32-bit data)
D	OUT	Inversion result transfer destination (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in dwData1 are inverted    *)
(* bit by bit, and the result is transferred to Result.                       *)
  DCML_M (X0, dwData1, Result);
```

● Corresponding MELSEC command

・DCML (32-bit NOT transfer)

For the usable data type, refer to "3.2.2 About ANY type".

5.7.3 Block transfer BMOV_M

n points of BIN 16-bit data, starting at the specified device, are batch-transferred.

■ Function definition BOOL BMOV_M (BOOL EN, ANY16 S1, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be transferred (BIN 16-bit data)
n	IN	Number of data to be transferred (BIN 16-bit data)
D	OUT	Transfer destination (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 16-bit data of the number of      *)
(* points stored in W0, starting at the device specified in D0, are transferred to *)
(* the number of points stored in W0, starting at D100.                         *)
  BMOV_M (X0, D0, W0, D100);
```

● Corresponding MELSEC command

- BMOV (Block 16-bit transfer)

5.7.4 Same data block transfer FMOV_M

The 16-bit data of the specified device are transferred to the number of points, starting at the specified device.

■ Function definition BOOL FMOV_M (BOOL EN, ANY16 S1, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be transferred (BIN 16-bit data)
n	IN	Number of data to be transferred (BIN 16-bit data)
D	OUT	Transfer destination (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 16-bit data of D0 are transferred *)
(* to the number of points stored in W0, starting at D100.                      *)
  FMOV_M (X0, D0, W0, D100);
```

● Corresponding MELSEC command

- FMOV (Block 16-bit data transfer)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 34

5.7.5 16-bit data exchange XCH_M

The specified two BIN 16-bit data are exchanged.

■ Function definition BOOL XCH_M (BOOL EN, ANY16 D1, ANY16 D2);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN/OUT	Data to be exchanged, exchange result (BIN 16-bit data)
D2	IN/OUT	Data to be exchanged, exchange result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 16-bit data in D100 and D200  *)
(* are exchanged.                                                           *)
  XCH_M (XO, D100, D200);
```

● Corresponding MELSEC command

- XCH (16-bit data exchange)

5.7.6 32-bit data exchange DXCH_M

The specified two BIN 32-bit data are exchanged.

■ Function definition BOOL DXCH_M (BOOL EN, ANY32 D1, ANY32 D2);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D1	IN/OUT	Data to be exchanged, exchange result (BIN 32-bit data)
D2	IN/OUT	Data to be exchanged, exchange result (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 32-bit data in dwData1 and  *)
(* dwData2 are exchanged.                                                 *)
  DXCH_M (X0, dwData1, dwData2);
```

● Corresponding MELSEC command

- DXCH (32-bit data exchange)

For the usable data type, refer to "3.2.2 About ANY type".

5.7.7 Block data exchange BXCH_M

n points of BIN 16-bit data, starting at the specified devices, are exchanged.

■ Function definition BOOL BXCH_M (BOOL EN, ANY16 n, ANY16 D1, ANY16 D2);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of data to be exchanged (BIN 16-bit data)
D1	IN/ OUT	Data to be exchanged, exchange result (BIN 16-bit data)
D2	IN/ OUT	Data to be exchanged, exchange result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, 3 points of 16-bit data, starting at  *)
(* D100, and 3 points of 16-bit data, starting at D200, are exchanged.          *)
  BXCH_M (X0, K3, D100, D200);
```

● Corresponding MELSEC command

- BXCH (Block 16-bit data exchange)

5.7.8 First/last byte exchange SWAP_MD

The first 8 bits and last 8 bits of the specified device are exchanged.

■ Function definition BOOL SWAP_MD (BOOL EN, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	IN/OUT	Data to be exchanged, exchange result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the first 8 bits and last 8 bits of D0  *)
(* are exchanged.                                                                 *)
  SWAP_MD (X0, D0);
```

● Corresponding MELSEC command

- SWAP (First/last byte exchange)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 36

5.8 Program Execution Control

5.8.1 Interrupt disable DI_M

If the interrupt factor of an interrupt program occurs, the execution of the interrupt program is disabled until EI_M is executed.

■ Function definition BOOL DI_M (BOOL EN);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Only value TRUE indicating that the result is always valid or normally ON device SM400 can be specified.)

Return Value	Description
BOOL	Execution condition (always TRUE)

● Example of use

```
(* The execution of the interrupt program is disabled until until EI_M is executed. *)
DI_M (TRUE);
```

● Corresponding MELSEC command

・DI (Interrupt disable)

5.8.2 Interrupt enable EI_M

The interrupt disable status during DI_M execution is reset, and the execution of the interrupt program of the interrupt pointer number enabled by IMASK is enabled.

■ Function definition BOOL EI_M (BOOL EN);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Only value TRUE indicating that the result is always valid or normally ON device SM400 can be specified.)

Return Value	Description
BOOL	Execution condition (always TRUE)

● Example of use

```
(* The interrupt disable status during DI_M execution is reset.        *)
EI_M (TRUE);
```

● Corresponding MELSEC command

・EI (Interrupt enable)

5.9 I/O Refresh

5.9.1 I/O refresh RFS_M

n points of I/O devices, starting at the specified device, are refreshed.

■ Function definition BOOL RFS_M (BOOL EN, BOOL S1, ANY16 n);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Devices to be refreshed (bit data)
n	IN	Number of data to be refreshed (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, 32 points of devices, starting at      *)
(* X100, are refreshed.                                                          *)
   RFS_M (M0, X100, H20);
```

● Corresponding MELSEC command

- RFS (I/O refresh)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 38

5.10 Logical Operation Commands

5.10.1 Logical product (2 devices) WAND_M

The specified two BIN 16-bit data are ANDed bit by bit.

■ Function definition BOOL WAND_M (BOOL EN, ANY16 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to AND (BIN 16-bit data)
D	IN/OUT	Data to be ANDed, operation result (BIN 16-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 16-bit data in D0 and D10 are  *)
(* ANDed bit by bit, and the result is stored into D10.                      *)
  WAND_M (X0, D0, D10);
```

● Corresponding MELSEC command

- WAND (16-bit data logical product)

5.10.2 Logical product (3 devices) WAND_3_M

The specified two BIN 16-bit data are ANDed bit by bit.

■ Function definition BOOL WAND_3_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 D1);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be ANDed (BIN 16-bit data)
S2	IN	Data to AND (BIN 16-bit data)
D1	OUT	Operation result (BIN 16-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 16-bit data in D0 and D10 are  *)
(* ANDed bit by bit, and the result is stored into D100.                     *)
  WAND_3_M (X0, D0, D10, D100);
```

● Corresponding MELSEC command

- WAND (16-bit data logical product)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 39

5.10.3 32-bit data logical product (2 devices) DAND_M

The specified two BIN 32-bit data are ANDed bit by bit.

■ Function definition BOOL DAND_M (BOOL EN, ANY32 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to AND (BIN 32-bit data)
D	IN/OUT	Data to be ANDed, operation result (BIN 32-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 24-bit data in dwData1 and    *)
(* X30 to X47 are ANDed, and the result is stored into dwData1.             *)
  DAND_M (X0, K6X30, dwData1);
```

● Corresponding MELSEC command

- DAND (32-bit data logical product)

5.10.4 32-bit data logical product (3 devices) DAND_3_M

The specified two BIN 32-bit data are ANDed bit by bit.

■ Function definition BOOL DAND_3_M (BOOL EN, ANY32 S1, ANY32 S2, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be ANDed (BIN 32-bit data)
S2	IN	Data to AND (BIN 32-bit data)
D	OUT	Operation result (BIN 32-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 32-bit data in dwData1 and    *)
(* dwData2 are ANDed, and the result is stored into Result.                 *)
  DAND_3_M (X0, dwData1, dwData2, Result);
```

● Corresponding MELSEC command

- DAND (32-bit data logical product)

For the usable data type, refer to "3.2.2 About ANY type".

5.10.5 Block data logical product BKAND_M

n points of 16-bit data, starting at the specified two devices, are ANDed bit by bit.

■ Function definition BOOL BKAND_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be ANDed, operation result (BIN 16-bit data)
S2	IN	Data to AND (BIN 16-bit data)
n	IN	Number of data to be processed (BIN 16-bit data)
D	OUT	Operation result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of points   *)
(* stored in D0, starting at D100, and the data of the number of points stored  *)
(* in D0, starting at D200, are ANDed, and the result is stored into D1000   *)
(* and later.                                                                *)
  BKAND_M (X0, D100, D200, D0, D1000);
```

● Corresponding MELSEC command

- BKAND (Block logical product)

5.10.6 Logical sum (2 devices) WOR_M

The specified two BIN 16-bit data are ORed bit by bit.

■ Function definition BOOL WOR_M (BOOL EN, ANY16 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE))
S1	IN	Data to OR (BIN 16-bit data)
D	IN/OUT	Data to be ORed, operation result (BIN 16-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in D10 and D20 are ORed,  *)
(* and the result is stored into D10.                                        *)
  WOR_M (X0, D10, D20);
```

● Corresponding MELSEC command

- WOR (16-bit data logical sum)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 41

5.10.7 Logical sum (3 devices) WOR_3_M

The specified two BIN 16-bit data are ORed bit by bit.

■ Function definition BOOL WOR_3_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 D1);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be ORed (BIN 16-bit data)
S2	IN	Data to OR (BIN 16-bit data)
D1	OUT	Operation result (BIN 16-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in X10 to X1B and the data *)
(* in D0 are ORed, and the result is output to Y10 - Y1B.                     *)
  WOR_3_M (M0, K3X10, D0, K3Y10);
```

● Corresponding MELSEC command

- WOR (16-bit data logical sum)

5.10.8 32-bit data logical sum (2 devices) DOR_M

The specified two BIN 32-bit data are ORed bit by bit.

■ Function definition BOOL DOR_M (BOOL EN, ANY32 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to OR (BIN 32-bit data)
D	IN/OUT	Data to be ORed, operation result (BIN 32-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in dwData1 and Result are *)
(* ORed, and the result is output to Result.                                 *)
  DOR _M (X0, dwData1, Result);
```

● Corresponding MELSEC command

- DOR (32-bit data logical sum)

For the usable data type, refer to "3.2.2 About ANY type".

5.10.9 32-bit data logical sum (3 devices) DOR_3_M

The specified two BIN 32-bit data are ORed bit by bit.

■ Function definition BOOL DOR_3_M (BOOL EN, ANY32 S1, ANY32 S2, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be ORed (BIN 32-bit data)
S2	IN	Data to OR (BIN 32-bit data)
D	OUT	Operation result (BIN 32-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 32-bit data in dwData1 and the  *)
(* 32-bit data in X20 to X3F are ORed, and the result is output to Result.    *)
 DOR_3_M (X0, dwData1, K8X20, Result);
```

● Corresponding MELSEC command

- DOR (32-bit data logical sum)

5.10.10 Block data logical sum BKOR_M

n points of 16-bit data, starting at the specified two devices, are ORed bit by bit.

■ Function definition BOOL BKOR_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be ORed (BIN 16-bit data)
S2	IN	Data to OR (BIN 16-bit data)
n	IN	Number of data to be processed (BIN 16-bit data)
D	OUT	Operation result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of points    *)
(* stored in D0, starting at D10, and the data of the number of points stored in  *)
(* D0, starting at D20, are ORed, and the result is stored into D100 and later.   *)
 BKOR_M (X0, D10, D20, D0, D100);
```

● Corresponding MELSEC command

- BKOR (Block logical sum)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 43

5.10.11 Exclusive OR (2 devices) WXOR_M

The specified two BIN 16-bit data are EXCLUSIVE ORed bit by bit.

■ Function definition BOOL WXOR_M (BOOL EN, ANY16 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to EXCLUSIVE OR (BIN 16-bit data)
D	IN/OUT	Data to be EXCLUSIVE ORed, operation result (BIN 16-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 16-bit data in D10 and D20 are  *)
(* EXCLUSIVE ORed, and the result is stored into D20.                         *)
  WXOR_M (X0, D10, D20);
```

● Corresponding MELSEC command

- WXOR (16-bit data exclusive OR)

5.10.12 Exclusive OR (3 devices) WXOR_3_M

The specified two BIN 16-bit data are EXCLUSIVE ORed bit by bit.

■ Function definition BOOL WXOR_3_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 D1);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be EXCLUSIVE ORed (BIN 16-bit data)
S2	IN	Data to EXCLUSIVE OR (BIN 16-bit data)
D	OUT	Operation result (BIN 16-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 16-bit data in D10 and D20 are  *)
(* EXCLUSIVE ORed, and the result is stored into D100.                        *)
  WXOR_3_M (X0, D10, D20, D100);
```

● Corresponding MELSEC command

- WXOR (16-bit data exclusive OR)

For the usable data type, refer to "3.2.2 About ANY type".

5.10.13 32-bit data exclusive OR (2 devices) DXOR_M

The specified two BIN 32-bit data are EXCLUSIVE ORed bit by bit.

■ Function definition BOOL DXOR_M (BOOL EN, ANY32 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to EXCLUSIVE OR (BIN 32-bit data)
D	IN/OUT	Data to be EXCLUSIVE ORed, operation result (BIN 32-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 32-bit data in dwData1 and      *)
(* Result are EXCLUSIVE ORed, and the result is stored into Result.          *)
 DXOR_M (X0, dwData1, Result);
```

● Corresponding MELSEC command

- DXOR (32-bit data exclusive OR)

5.10.14 32-bit data exclusive OR (3 devices) DXOR_3_M

The specified two BIN 32-bit data are EXCLUSIVE ORed bit by bit.

■ Function definition BOOL DXOR_3_M (BOOL EN, ANY32 S1, ANY32 S2, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be EXCLUSIVE ORed (BIN 32-bit data)
S2	IN	Data to EXCLUSIVE OR (BIN 32-bit data)
D	OUT	Operation result (BIN 32-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 32-bit data in dwData1 and      *)
(* dwData2 are EXCLUSIVE ORed, and the result is stored into Result.         *)
 DXOR_3_M (X0, dwData1, dwData2, Result);
```

● Corresponding MELSEC command

- DXOR (32-bit data exclusive OR)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 45

5.10.15 Block data exclusive OR BKXOR_M

n points of 16-bit data, starting at the specified two devices, are EXCLUSIVE ORed bit by bit.

■ Function definition BOOL BKXOR_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be EXCLUSIVE ORed, operation result (BIN 16-bit data)
S2	IN	Data to EXCLUSIVE OR (BIN 16-bit data)
n	IN	Number of data to be processed (BIN 16-bit data)
D	OUT	Operation result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of points   *)
(* stored in D0, starting at D10, and the data of the number of points stored *)
(* in D0, starting at D20, are EXCLUSIVE ORed, and the result is stored into  *)
(* D100 and later.                                                           *)
  BKXOR_M (X0, D10, D20, D0, D100);
```

● Corresponding MELSEC command

- BKXOR (Block exclusive OR)

5.10.16 NOT exclusive OR (2 devices) WXNR_M

The specified two BIN 16-bit data are NOT EXCLUSIVE ORed bit by bit.

■ Function definition BOOL WXNR_M (BOOL EN, ANY32 S1, ANY32 S2, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to NOT EXCLUSIVE OR (BIN 16-bit data)
D	IN/OUT	Data to be NOT EXCLUSIVE ORed, operation result (BIN 16-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 16-bit data in X20 to X2F and  *)
(* the 16-bit data in D10 are NOT EXCLUSIVE ORed, and the result is stored   *)
(* into D10.                                                                 *)
  WXNR_M (X0, K4X20, D10);
```

● Corresponding MELSEC command

- WXNR (16-bit data NOT exclusive OR)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 46

5.10.17 NOT exclusive OR (3 devices) WXNR_3_M

The specified two BIN 16-bit data are NOT EXCLUSIVE ORed bit by bit.

■ Function definition BOOL WXNR_3_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be NOT EXCLUSIVE ORed (BIN 16-bit data)
S2	IN	Data to NOT EXCLUSIVE OR (BIN 16-bit data)
D	OUT	Operation result (BIN 16-bit data)

Remarks: The same device can be specified in arguments "S1" and "D", and in "S2" and "D".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 16-bit data in X20 to X2F and  *)
(* the 16-bit data in D0 are NOT EXCLUSIVE ORed, and the result is stored  *)
(* into D100.                                                               *)
  WXNR_3_M (X0, K4X20, D0, D100);
```

● Corresponding MELSEC command

- WXNR (16-bit data NOT exclusive OR)

5.10.18 32-bit data NOT exclusive OR (2 devices) DXNR_M

The specified two BIN 32-bit data are NOT EXCLUSIVE ORed bit by bit.

■ Function definition BOOL DXNR_M (BOOL EN, ANY32 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to NOT EXCLUSIVE OR (BIN 32-bit data)
D	IN/OUT	Data to be NOT EXCLUSIVE ORed, operation result (BIN 32-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 32-bit data in dwData1 and the  *)
(* 32-bit data in Result are NOT EXCLUSIVE ORed, and the result is stored     *)
(* into Result.                                                               *)
  DXNR_M (X0, dwData1, Result);
```

● Corresponding MELSEC command

- DXNR (32-bit data NOT exclusive OR)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 47

5.10.19 32-bit data NOT exclusive OR (3 devices) DXNR_3_M

The specified two BIN 32-bit data are NOT EXCLUSIVE ORed bit by bit.

■ Function definition BOOL DXNR_3_M (BOOL EN, ANY32 S1, ANY32 S2, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be NOT EXCLUSIVE ORed (BIN 32-bit data)
S2	IN	Data to NOT EXCLUSIVE OR (BIN 32-bit data)
D	OUT	Operation result (BIN 32-bit data)

Remarks: For bit devices, the bits greater than in the digit specification are processed as "0 (zero)".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 32-bit data in dwData1 and the  *)
(* 32-bit data in dwData2 are NOT EXCLUSIVE ORed, and the result is stored  *)
(* into Result.                                                              *)
  DXNR_3_M (X0, dwData1, dwData2, Result);
```

● Corresponding MELSEC command

- DXNR (32-bit data NOT exclusive OR)

5.10.20 Block data NOT exclusive OR BKXNR_M

n points of 16-bit data, starting at the specified two devices, are NOT EXCLUSIVE ORed bit by bit.

■ Function definition BOOL BKXNR_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be NOT EXCLUSIVE ORed (BIN 16-bit data)
S2	IN	Data to NOT EXCLUSIVE OR (BIN 16-bit data)
n	IN	Number of data to be processed (BIN 16-bit data)
D	OUT	Operation result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of points  *)
(* stored in D0, starting at D100, and the data of the number of points stored  *)
(* in D0, starting at W100, are NOT EXCLUSIVE ORed, and the result is  *)
(* stored into D200 and later.                                        *)
  BKXNR_M (X0, D100, W100, D0, D200);
```

● Corresponding MELSEC command

- BKXNR (Block NOT exclusive OR)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 48

5.11 Rotation

5.11.1 Right rotation (carry flag not included) ROR_M

The specified BIN 16-bit data are rotated n bits to the right, without the carry flag being included.

■ Function definition BOOL ROR_M (BOOL EN, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of rotations (0 to 15) (BIN 16-bit data)
D	IN/OUT	Data to be rotated, rotation result (BIN 16-bit data)

Remarks: When a bit device is specified in "D", the data in the specified number of digits are rotated.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in D0 are rotated 3 bits to  *)
(* the right, without the carry flag being included.                            *)
  ROR_M (X0, K3, D0);
```

● Corresponding MELSEC command

- ROR (Right rotation of 16-bit data)

5.11.2 Right rotation (carry flag included) RCR_M

The specified BIN 16-bit data are rotated n bits to the right, with the carry flag being included.

■ Function definition BOOL RCR_M (BOOL EN, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of rotations (0 to 15) (BIN 16-bit data)
D	IN/OUT	Data to be rotated, rotation result (BIN 16-bit data)

Remarks: When a bit device is specified in "D", the data in the specified number of digits are rotated.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in D0 are rotated 3 bits to  *)
(* the right, with the carry flag being included.                               *)
  RCR_M (X0, K3, D0);
```

● Corresponding MELSEC command

- RCR (Right rotation of 16-bit data)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 49

5.11.3 Left rotation (carry flag not included) ROL_M

The specified BIN 16-bit data are rotated n bits to the left, without the carry flag being included.

■ Function definition BOOL ROL_M (BOOL EN, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of rotations (0 to 15) (BIN 16-bit data)
D	IN/OUT	Data to be rotated, rotation result (BIN 16-bit data)

Remarks: When a bit device is specified in "D", the data in the specified number of digits are rotated.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in D0 are rotated 3 bits to  *)
(* the left, without the carry flag being included.                             *)
  ROL_M (X0, K3, D0);
```

● Corresponding MELSEC command

- ROL (Left rotation of 16-bit data)

5.11.4 Left rotation (carry flag included) RCL_M

The specified BIN 16-bit data are rotated n bits to the left, with the carry flag being included.

■ Function definition BOOL RCL_M (BOOL EN, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of rotations (0 to 15) (BIN 16-bit data)
D	IN/OUT	Data to be rotated, rotation result (BIN 16-bit data)

Remarks: When a bit device is specified in "D", the data in the specified number of digits are rotated.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in D0 are rotated 3 bits to  *)
(* the left, with the carry flag being included.                                *)
  RCL_M (X0, K3, D0);
```

● Corresponding MELSEC command

- RCL (Left rotation of 16-bit data)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 50

5.11.5 32-bit data right rotation (carry flag not included) DROR_M

The specified BIN 32-bit data are rotated n bits to the right, without the carry flag being included.

■ Function definition BOOL DROR_M (BOOL EN, ANY16 n, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of rotations (0 to 31) (BIN 16-bit data)
D	IN/OUT	Data to be rotated, rotation result (BIN 32-bit data)

Remarks: When a bit device is specified in "D", the data in the specified number of digits are rotated.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 32-bit data in dwData1 are      *)
(* rotated to the right by the number of bits stored in D0, without the carry flag  *)
(* being included.                                                           *)
  DROR_M (X0, D0, dwData1);
```

● Corresponding MELSEC command

- DROR (Right rotation of 32-bit data)

5.11.6 32-bit data right rotation (carry flag included) DRCR_M

The specified BIN 32-bit data are rotated n bits to the right, with the carry flag being included.

■ Function definition BOOL DRCR_M (BOOL EN, ANY16 n, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of rotations (0 to 31) (BIN 16-bit data)
D	IN/OUT	Data to be rotated, rotation result (BIN 32-bit data)

Remarks: When a bit device is specified in "D", the data in the specified number of digits are rotated.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 32-bit data in dwData1 are      *)
(* rotated to the right by the number of bits stored in D0, with the carry flag  *)
(* being included.                                                           *)
  DRCR_M (X0, D0, dwData1);
```

● Corresponding MELSEC command

- DRCR (Right rotation of 32-bit data)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 51

5.11.7 32-bit data left rotation (carry flag not included) DROL_M

The specified BIN 32-bit data are rotated n bits to the left, without the carry flag being included.

■ Function definition BOOL DROL_M (BOOL EN, ANY16 n, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of rotations (0 to 31) (BIN 16-bit data)
D	IN/OUT	Data to be rotated, rotation result (BIN 32-bit data)

Remarks: When a bit device is specified in "D", the data in the specified number of digits are rotated.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 32-bit data in dwData1 are   *)
(* rotated 4 bits to the left, without the carry flag being included.     *)
  DROL_M (X0, K4, dwData1);
```

● Corresponding MELSEC command

- DROL (Left rotation of 32-bit data)

5.11.8 32-bit data left rotation (carry flag included) DRCL_M

The specified BIN 32-bit data are rotated n bits to the left, with the carry flag being included.

■ Function definition BOOL DRCL_M (BOOL EN, ANY16 n, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of rotations (0 to 31) (BIN 16-bit data)
D	IN/OUT	Data to be rotated, rotation result (BIN 32-bit data)

Remarks: When a bit device is specified in "D", the data in the specified number of digits are rotated.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 32-bit data in dwData1 are   *)
(* rotated 4 bits to the left, with the carry flag being included.        *)
  DRCL_M (X0, K4, dwData1);
```

● Corresponding MELSEC command

- DRCL (Left rotation of 32-bit data)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 52

5.12 Shift

5.12.1 n-bit right shift SFR_M

The specified BIN 16-bit data are shifted n bits to the right.

■ Function definition BOOL SFR_M (BOOL EN, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of shifts (0 to 15) (BIN 16-bit data)
D	IN/OUT	Data to be shifted, shift result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in D100 are shifted 4 bits  *)
(* to the right.                                                               *)
   SFR_M (X0, K4, D100);
```

● Corresponding MELSEC command

・SFR (n-bit right shift of 16-bit data)

5.12.2 n-bit left shift SFL_M

The specified BIN 16-bit data are shifted n bits to the left.

■ Function definition BOOL SFL_M (BOOL EN, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of shifts (0 to 15) (BIN 16-bit data)
D	IN/OUT	Data to be shifted, shift result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in D100 are shifted 4 bits  *)
(* to the left.                                                                *)
   SFL_M (X0, K4, D100);
```

● Corresponding MELSEC command

・SFL (n-bit left shift of 16-bit data)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 53

5.12.3 n-bit data 1-bit right shift BSFR_M

n points of bit data, starting at the specified device, are shifted one bit to the right.

■ Function definition BOOL BSFR_M (BOOL EN, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of devices to be shifted (BIN 16-bit data)
D	IN/OUT	Data to be shifted, shift result (bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in M100 to M104 are      *)
(* shifted 1 bit to the right.                                              *)
   BSFR_M (X0, K5, M100);
```

● Corresponding MELSEC command

- BSFR (1-bit right shift of n-bit data)

5.12.4 n-bit data 1-bit left shift BSFL_M

n points of bit data, starting at the specified device, are shifted one bit to the left.

■ Function definition BOOL BSFL_M (BOOL EN, ANY16 n, BOOL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of devices to be shifted (BIN 16-bit data)
D	IN/OUT	Data to be shifted, shift result (bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in M100 to M104 are      *)
(* shifted 1 bit to the left.                                               *)
   BSFL_M (X0, K5, M100);
```

● Corresponding MELSEC command

- BSFL (1-bit left shift of n-bit data)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 54

5.12.5 1-word right shift DSFR_M

n points of 16-bit data, starting at the specified device, are shifted one word to the right.

■ Function definition BOOL DSFR_M (BOOL EN, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of devices to be shifted (BIN 16-bit data)
D	IN/OUT	Data to be shifted, shift result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in D100 to D106 are      *)
(* shifted 1 word to the right.                                            *)
   DSFR_M (X0, K7, D100);
```

● Corresponding MELSEC command

- DSFR (1-word right shift of n-word data)

5.12.6 1-word left shift DSFL_M

n points of 16-bit data, starting at the specified device, are shifted one word to the left.

■ Function definition BOOL DSFL_M (BOOL EN, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Number of devices to be shifted (BIN 16-bit data)
D	OUT	Data to be shifted, shift result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in D100 to D106 are      *)
(* shifted 1 word to the left.                                             *)
   DSFL_M (X0, K7, D100);
```

● Corresponding MELSEC command

- DSFL (1-word left shift of n-word data)

For the usable data type, refer to "3.2.2 About ANY type".

5.13 Bit Processing

5.13.1 Bit set of word device BSET_M

Bit n of the specified word device is set.

■ Function definition BOOL_BSET_M (BOOL EN, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Bit number to be set (BIN 16-bit data)
D	IN/OUT	Data to be set, bit set result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, bit 8 of D100 is set.        *)
  BSET_M (X0, K8, D100);
```

● Corresponding MELSEC command

・BSET (Bit set of word device)

5.13.2 Bit reset of word device BRST_M

Bit n of the specified word device is reset.

■ Function definition BOOL_BRST_M (BOOL EN, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Bit number to be reset (BIN 16-bit data)
D	IN/OUT	Data to be reset, bit reset result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, bit 8 of D100 is reset.      *)
  BRST_M (X0, K8, D100);
```

● Corresponding MELSEC command

・BRST (Bit reset of word device)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 56

5.13.3 Bit test of word device TEST_MD

The bit status in the specified position of the specified word device is written to the specified bit device.

■ Function definition BOOL TEST_MD (BOOL EN, ANY16 S1, ANY 16 S2, BOOL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be extracted (BIN 16-bit data)
S2	IN	Position of bit to be extracted (BIN 16-bit data)
D	OUT	Extracted data (bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, M0 is turned ON/OFF      *)
(* according to the status of bit 10 of D100.                     *)
   TEST_MD (X0, D100, K10, M0);
```

● Corresponding MELSEC command

・TEST (Bit set)

5.13.4 Bit test of 32-bit data DTEST_MD

The bit in the specified position of the specified BIN 32-bit data is written to the specified bit device.

■ Function definition BOOL DTEST_MD (BOOL EN, ANY32 S1, ANY16 S2,BOOL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be extracted (BIN 32-bit data)
S2	IN	Position of bit to be extracted (BIN 16-bit data)
D	OUT	Extracted data (bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, bit 10 in dData is fetched and  *)
(* written to M0.                                                        *)
   DTEST_MD (X0, dData, K10, M0);
```

● Corresponding MELSEC command

・DTEST (Bit set)

For the usable data type, refer to "3.2.2 About ANY type".

5.13.5 Bit device batch reset BKRST_M

n points, starting at the specified bit device, are reset.

■ Function definition BOOL BKRST_M (BOOL EN, BOOL S1, ANY16 n);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Head of data to be reset (bit data)
n	IN	Number of bits to be reset (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the number of points stored in D100,  *)
(* starting at M10, are reset.                                                  *)
  BKRST_M (X0, M10, D100);
```

● Corresponding MELSEC command

- BKRST (Batch reset of bit devices)

For the usable data type, refer to "3.2.2 About ANY type".

5.14 Data Processing

5.14.1 Data search SER_M

n points of data, starting at the specified BIN 16-bit data, are searched for the specified BIN 16-bit data.

■ Function definition BOOL SER_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, ANY16(2) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Data to be searched for (BIN 16-bit data)		
S2	IN	Data to be searched (BIN 16-bit data)		
n	IN	Number of data to be searched (BIN 16-bit data)		
D	OUT	Search result (ARRAY [0..1] OF ANY16)	D[0]	Match position
			D[1]	Number of matches

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, D300 points of data, starting at D200, *)
(* are searched for D100.                                                        *)
(* The number of data that matched the search target is stored into D[1], and the *)
(* relative value indicating the number of points from D200 is stored into D[0]. *)
  SER_M (X0, D100, D200, D300, D);
```

● Corresponding MELSEC command
- SER (16-bit data search)

5.14.2 32-bit data search DSER_M

2n points of data, starting at the specified BIN 32-bit data, are searched for the specified BIN 32-bit data.

■ Function definition BOOL SER_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 n, ANY16(2) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Data to be searched for (BIN 32-bit data)		
S2	IN	Data to be searched (BIN 32-bit data)		
n	IN	Number of data to be searched (BIN 16-bit data)		
D	OUT	Search result (ARRAY [0..1] OF ANY16)	D[0]	Match position
			D[1]	Number of matches

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the number of points stored in D100, *)
(* starting at dData2, are searched on a 32-bit basis for dData1 and dData1+1. *)
(* The number of data that matched the search target is stored into             *)
(* ArrayResult[1], and the relative value indicating the number of points from  *)
(* dData2 is stored into ArrayResult[0].                                        *)
  DSER_M (X0, dData1, dData2, D100, ArrayResult);
```

● Corresponding MELSEC command
- DSER (32-bit data search)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 59

5.14.3 Bit check SUM_M

The number of bits having 1 in the specified BIN 16-bit data is counted.

■ Function definition BOOL SUM_M (BOOL EN, ANY16 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be counted (BIN 16-bit data)
D	OUT	Count result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the number of bits having 1 in dData  *)
(* is stored into Result.                                                       *)
   SUM_M (X0, iData, Result);
```

● Corresponding MELSEC command

・SUM (16-bit data bit check)

5.14.4 32-bit data bit check DSUM_M

The number of bits having 1 in the specified BIN 32-bit data is counted.

■ Function definition BOOL DSUM_M (BOOL EN);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be counted (BIN 32-bit data)
D	OUT	Count result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the number of bits having 1  *)
(* in iData is stored into Result.                                     *)
   DSUM_M (X0, dData, Result);
```

● Corresponding MELSEC command

・DSUM (32-bit data bit check)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 60

5.14.5 Decode DECO_M

The lower n bits of the specified data are decoded.

■ Function definition BOOL DECO_M (BOOL EN, ANY_SIMPLE S1, ANY16 n, ANY_SIMPLE D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be decoded
n	IN	Valid bit length (1 to 8) *0: No processing (BIN 16-bit data)
D	OUT	Decode result

Remarks: The DINT, REAL and STRING types cannot be used in arguments S1" and "D".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the lower Bit Size bits of D100 are  *)
(* decoded, and the decode result is stored into 2^BitSize bits, starting at Result.  *)
  DECO_M (X0, D100, BitSize, Result)
```

● Corresponding MELSEC command

- DECO (8 → 256 bits decode)

For the usable data type, refer to "3.2.2 About ANY type".

5.14.6 Encode ENCO_M

2^n bits of data, starting at the specified data, are encoded.

■ Function definition BOOL ENCO_M (BOOL EN, ANY_SIMPLE S1, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be encoded
n	IN	Valid bit length (1 to 8) *0: No processing (BIN 16-bit data)
D	OUT	Encode result

Remarks: The DINT, REAL and STRING types cannot be used in argument S1".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, 2^BitSize bits, starting at D100, are  *)
(* encoded, and the result is stored into Result.  *)
  ENCO (X0, D100, BitSize, Result);
```

● Corresponding MELSEC command

- ENCO (256 → 8 bits decode)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 61

5.14.7 7-segment decode SEG_M

The lower 4 bits (0 to F) of the specified data are decoded into 7-segment display data.

■ Function definition BOOL SEG_M (BOOL EN, ANY16 S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be decoded
D	OUT	Decode result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the lower 4 bits of D100 are decoded  *)
(* into 7-segment display data, and the result is stored into Result.           *)
  SEG_M (X0, D100, Result);
```

● Corresponding MELSEC command

・SEG (7-segment decode)

5.14.8 4-bit disconnection of 16-bit data DIS_M

The data in the lower n digits of the specified BIN 16-bit data are disconnected and stored into the lower 4 bits of n points, starting at the specified device.

■ Function definition BOOL DIS_M (BOOL EN, ANY16 S1, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be disconnected (BIN 16-bit data)
n	IN	Number of disconnected data (1 to 4) *0: No processing (BIN 16-bit data)
D	OUT	Disconnection result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in the lower D200 digits     *)
(* (1 digit = 4 bits) of D100 are stored into the lower 4 bits of D200 points,  *)
(* starting at Result.                                                          *)
  DIS_M (X0, D100, D200, Result);
```

● Corresponding MELSEC command

・DIS (4-bit disconnection of 16-bit data)

For the usable data type, refer to "3.2.2 About ANY type".

5.14.9 4-bit connection of 16-bit data UNI_M

The lower 4 bits of n points of BIN 16-bit data, starting at the specified device, are connected to the specified device.

■ Function definition BOOL UNI_M (BOOL EN, ANY16 S1, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be connected (BIN 16-bit data)
n	IN	Number of connected data (1 to 4) *0: No processing (BIN 16-bit data)
D	OUT	Connection result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the lower 4 bits of 3 points of 16-bit  *)
(* data, starting at D100 are connected to Result.                                *)
  UNI_M (X0, D100, K3, Result);
```

● Corresponding MELSEC command

・UNI (4-bit connection of 16-bit data)

5.14.10 Bit disconnection of any data NDIS_M

The bits of the data stored in and after the specified device are disconnected in units of the specified bits.

■ Function definition BOOL NDIS_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be disconnected (BIN 16-bit data)
S2	IN	Disconnection unit (number of bits to be disconnected) (BIN 16-bit data)
D	OUT	Disconnection result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the bits of the data stored in and      *)
(* after iData1 are disconnected in units of iData2 bits, and the result is stored *)
(* into Result and later.                                                         *)
  NDIS_M (X0, iData1, iData2, Result);
```

● Corresponding MELSEC command

・NDIS (Disconnection of any bit data)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 63

5.14.11 Bit connection of any data NUNI_M

The bits of the data stored in and after the specified device are connected in units of the specified bits.

■ Function definition BOOL NUNI_M (BOOL EN, ANY16 S1, ANY16 S2, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be connected (BIN 16-bit data)
S2	IN	Connection unit (number of bits to be connected) (BIN 16-bit data)
D	OUT	Connection result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the bits of the data stored in and after *)
(* iData1 are connected in units of iData2 bits, and the result is stored into     *)
(* Result and later.                                                               *)
   NUNI_M (X0, iData1, iData2, Result);
```

● Corresponding MELSEC command

・NUNI (Connection of any bit data)

5.14.12 Byte unit data disconnection WTOB_MD

The BIN 16-bit data stored in and after the specified device are disconnected into n bytes.

■ Function definition BOOL WTOB_MD (BOOL EN, ANY16 S1, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be disconnected in byte units (BIN 16-bit data)
n	IN	Number of resultant byte data (BIN 16-bit data)
D	OUT	Disconnection result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the 16-bit data stored in and after *)
(* iData1 is disconnected in iData2 bytes, and the result is stored into Result *)
(* and later.                                                                   *)
   WTOB_MD (X0, iData1, iData2, Result);
```

● Corresponding MELSEC command

・WTOB (Disconnection into byte unit data)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 64

5.14.13 Byte unit data connection BTOW_MD

The lower 8 bits of n points of BIN 16-bit data in and after the specified device are connected in word units.

■ Function definition BOOL BTOW_MD (BOOL EN, ANY16 S1, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be connected in byte units (BIN 16-bit data)
n	IN	Number of byte data to be connected (BIN 16-bit data)
D	OUT	Connection result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the lower 8 bits of iData2 words of   *)
(* 16-bit data in and after iData1 are connected in word units, and the result is  *)
(* stored into Result and later.                                                *)
   BTOW_MD (X0, iData1, iData2, Result);
```

● Corresponding MELSEC command

・BTOW (Connection of byte unit data)

5.14.14 Data maximum value retrieval MAX_M

The maximum value is retrieved from n points of BIN 16-bit data, starting at the specified device.

■ Function definition BOOL MAX_M (BOOL EN, ANY16 S1, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Head of data to be retrieved (BIN 16-bit data)
n	IN	Number of data to be retrieved (BIN 16-bit data)
D	OUT	Maximum value retrieval result (BIN 16-bit data)

Remarks: When a constant is specified for the timer set value, only a decimal number can be specified.
The timer set value can be specified within the range 0 to 32767.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the maximum value is retrieved   *)
(* from iData2 points of 16-bit BIN data in and after iData1,and the result is  *)
(* stored into Result. *                                                  *)
   MAX_M (X0, iData1, iData2, Result)
```

● Corresponding MELSEC command

・MAX (16-bit data maximum value retrieval)

For the usable data type, refer to "3.2.2 About ANY type".

5.14.15 32-bit data maximum value retrieval DMAX_M

The maximum value is retrieved from n points of BIN 32-bit data, starting at the specified device.

■ Function definition BOOL DMAX_M (BOOL EN, ANY32 S1, ANY16 n, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Head of data to be retrieved (BIN 32-bit data)
n	IN	Number of data to be retrieved (BIN 16-bit data)
D	OUT	Maximum value retrieval result (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the maximum value is retrieved      *)
(* from iData points of 32-bit BIN data in and after dData, and the result   *)
(* is stored into Result.                                                    *)
  DMAX_M (X0, dData, iData, Result);
```

● Corresponding MELSEC command

・DMAX (32-bit data maximum value retrieval)

5.14.16 Data minimum value retrieval MIN_M

The minimum value is retrieved from n points of BIN 16-bit data, starting at the specified device.

■ Function definition BOOL MIN_M (BOOL EN, ANY16 S1, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Head of data to be retrieved (BIN 16-bit data)
n	IN	Number of data to be retrieved (BIN 16-bit data)
D	OUT	Minimum value retrieval result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the minimum value is retrieved from  *)
(* iData2 points of 16-bit BIN data in and after iData1, and the result is stored *)
(* into Result. Execution condition X0 is output to the assigned device of bData. *)
  MIN_M (X0, iData1, iData2, Result);
```

● Corresponding MELSEC command

・MIN (16-bit data minimum value retrieval)

For the usable data type, refer to "3.2.2 About ANY type".

5.14.17 32-bit data minimum value retrieval DMIN_M

The minimum value is retrieved from n points of BIN 32-bit data, starting at the specified device.

■ Function definition BOOL DMIN_M (BOOL EN, ANY32 S1, ANY16 n, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Head of data to be retrieved (BIN 32-bit data)
n	IN	Number of data to be retrieved (BIN 16-bit data)
D	OUT	Minimum value retrieval result (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the minimum value is retrieved from  *)
(* iData points of 32-bit BIN data in and after dData, and the result is stored  *)
(* into Result and Result+1.                                                     *)
  DMIN_M (X0, dData, iData, Result);
```

● Corresponding MELSEC command

- DMIN (32-bit data minimum value retrieval)

5.14.18 Data sort SORT_M

n points of BIN 16-bit data, starting at the specified device, are sorted in ascending or descending order.

■ Function definition BOOL SORT_M (BOOL EN, ANY16 S1, ANY16 n, ANY16 S2, BOOL D1, ANY16 D2);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Head of data to be sorted (BIN 16-bit data)
n	IN	Number of data to be sorted (BIN 16-bit data)
S2	IN	Number of data to be compared at one execution (BIN 16-bit data)
D1	OUT	Bit device to be turned ON at sort completion (bit data)
D2	OUT	System used device (BIN 16-bit data)

Remarks: Specify the sort order by turning ON/OFF SM703. When SM703 is OFF: Ascending order, when SM703 is ON: Descending order

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, iData2 points of BIN 16-bit data,  *)
(* starting at iData1, are sorted in ascending or descending order.          *)
SORT_M (X0, iData1, iData2, iData3, bData, iData4);
```

● Corresponding MELSEC command

- SORT (16-bit data sort)

For the usable data type, refer to "3.2.2 About ANY type".

5.14.19 32-bit data sort DSORT_M

n points of BIN 32-bit data, starting at the specified device, are sorted in ascending or descending order.

■ Function definition BOOL DSORT_M (BOOL EN, ANY32 S1, ANY16 n, ANY16 S2, BOOL D1, ANY16 D2);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Head of data to be sorted (BIN 32-bit data)
n	IN	Number of data to be sorted (BIN 16-bit data)
S2	IN	Number of data to be compared at one execution (BIN 16-bit data)
D1	OUT	Bit device to be turned ON at sort completion (bit data)
D2	OUT	System used device (BIN 16-bit data)

Remarks: Specify the sort order by turning ON/OFF SM703. When SM703 is OFF: Ascending order, when SM703 is ON: Descending order

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, iData1 points of BIN 32-bit data,  *)
(* starting at dData, are sorted in ascending or descending order.           *)
   DSORT_M (X0, dData, iData1, iData2, bData, iData3);
```

● Corresponding MELSEC command

- DSORT (32-bit data sort)

5.14.20 Total value calculation WSUM_M

n points of BIN 16-bit data, starting at the specified device, are all added.

■ Function definition BOOL WSUM_M (BOOL EN, ANY16 S1, ANY16 n, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data from which total value will be calculated (BIN 16-bit data)
n	IN	Number of data (BIN 16-bit data)
D	OUT	Total value storage destination (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, iData2 points of 16-bit BIN data,  *)
(* starting at iData1, are all added, and the result is stored into Result.  *)
   WSUM_M (X0, iData1, iData2, Result);
```

● Corresponding MELSEC command

- WSUM (16-bit total value calculation)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 68

5.14.21 32-bit total value calculation DWSUM_M

n points of BIN 32-bit data, starting at the specified device, are all added.

■ Function definition BOOL DWSUM_M (BOOL EN, ANY32 S1, ANY16 n, ANY16(4) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Data from which total value will be calculated (BIN 32-bit data)		
n	IN	Number of data (BIN 16-bit data)		
D	OUT	Total value storage destination (ARRAY [0..3] OF ANY16)	D[0]	Upper 4 digits to Lower 4 digits
			D[1]	
			D[2]	
			D[3]	

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, iData points of 32-bit BIN data,   *)
(* starting at dData, are all added, and the result is stored into Result.   *)
   DWSUM_M (X0, dData, iData, Result);
```

● Corresponding MELSEC command

・DWSUM (32-bit total value calculation)

For the usable data type, refer to "3.2.2 About ANY type".

5.15 Structuring

5.15.1 Refresh COM_M

The I/O refresh and general data processing of the intelligent function module are performed.

■ Function definition BOOL COM_M (BOOL EN);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Only value TRUE indicating that the result is always valid or normally ON device SM400 can be specified.)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When SM755 is OFF: I/O refresh and general data processing of intelligent   *)
(* function module, when SM755 is ON: Only general data processing is          *)
(* performed.                                                                  *)
  COM_M (TRUE);
```

● Corresponding MELSEC command

- COM (Refresh command)

5 - 70

5.16 Buffer Memory Access

5.16.1 Intelligent function module 1-word data read FROM_M

The specified points of data are read from the specified address and later of the buffer memory in the specified intelligent function module or special function module.

■ Function definition BOOL FROM_M (BOOL EN, ANY16 n1, ANY16 n2, ANY16 n3, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n1	IN	Head input number of specified intelligent function module/special function module (BIN 16-bit data)
n2	IN	Head address of data to be read (BIN 16-bit data)
n3	IN	Number of data to be read (BIN 16-bit data)
D	OUT	Read data (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, 1 word of data are read to D0 from     *)
(* address 10 and later of the buffer memory in the intelligent function module  *)
(* mounted at I/O numbers 040 to 05F.                                            *)
   FROM_M (X0, H4, K10, K1, D0);
```

● Corresponding MELSEC command

・FROM (Reading 1-word data from the intelligent function module)

5.16.2 Intelligent function module 2-word data read DFRO_M

The specified points ×2 of data are read from the specified address and later of the buffer memory in the specified intelligent function module or special function module.

■ Function definition BOOL DFRO_M (BOOL EN, ANY16 n1, ANY16 n2, ANY16 n3, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n1	IN	Head input number of specified intelligent function module/special function module (BIN 16-bit data)
n2	IN	Head address of data to be read (BIN 16-bit data)
n3	IN	Number of data to be read (BIN 16-bit data)
D	OUT	Read data (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, 2 words of data are read to DwResult  *)
(* from addresses 602, 603 and later of the buffer memory in the intelligent    *)
(* function module mounted at I/O numbers 040 to 05F.                           *)
   DFRO_M (X0, H4, K602, K1, DwResult);
```

● Corresponding MELSEC command

・DFRO (Reading 2-word data from the intelligent function module)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 71

5.16.3 Intelligent function module 1-word data write TO_M

n3 points of data, starting at the specified device, are written to the specified address and later of the buffer memory in the specified intelligent function module or special function module.

■ Function definition BOOL TO_M (BOOL EN, ANY16 S1, ANY16 n1, ANY16 n2, ANY16 n3);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be written (BIN 16-bit data)
n1	IN	Head input number of specified intelligent function module/special function module (BIN 16-bit data)
n2	IN	Head address where data will be written (BIN 16-bit data)
n3	IN	Number of data to be written (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, 3 is written to address 0 of the buffer  *)
(* memory in the intelligent function module mounted at I/O numbers 040 to 05F. *)
  TO_M (X0, K3, H4, K0, K1);
```

● Corresponding MELSEC command

・TO (Writing 1-word data to intelligent function module)

5.16.4 Intelligent function module 2-word data write DTO_M

n3×2 points of data, starting at the specified device, are written to the specified address and later of the buffer memory in the specified intelligent function module or special function module.

■ Function definition BOOL DTO_M (BOOL EN, ANY32 S1, ANY16 n1, ANY16 n2, ANY16 n3);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be written (BIN 32-bit data)
n1	IN	Head input number of specified intelligent function module/special function module (BIN 16-bit data)
n2	IN	Head address where (3×2) points of data will be written (BIN 16-bit data)
n3	IN	Number of data to be written (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, 0 is written to addresses 41, 42      *)
(* of the buffer memory in the intelligent function module mounted at I/O       *)
(* numbers 040 to 05F.                                                          *)
  DTO_M (X0, K0, H4, K41, K1);
```

● Corresponding MELSEC command

・DTO (Writing 2-word data to intelligent function module)

For the usable data type, refer to "3.2.2 About ANY type".

5.17 Character string processing

5.17.1 BIN→decimal ASCII conversion BINDA_S_MD

The numeric value in each digit of the specified BIN 16-bit data represented in decimal is converted into ASCII code data.

■ Function definition BOOL BINDA_S_MD (BOOL EN, ANY16 S1, STRING(8) D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 16-bit data)
D	OUT	Conversion result (decimal ASCII code data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the numeric value in each digit of the  *)
(* BIN data stored in iData and represented in decimal is converted into ASCII    *)
(* code, and the result is stored into sData.                                     *)
  BINDA_S_MD (X0, iData, sData);
```

● Corresponding MELSEC command

・BINDA (BIN 16-bit to decimal ASCII conversion)

5.17.2 32-bit BIN→decimal ASCII conversion DBINDA_S_MD

The numeric value in each digit of the specified BIN 32-bit data represented in decimal is converted into ASCII code data.

■ Function definition BOOL DBINDA_S_MD (BOOL EN, ANY32 S1, STRING(12) D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 32-bit data)
D	OUT	Conversion result (decimal ASCII code data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the numeric value in each digit of     *)
(* the BIN data stored in dData and represented in decimal is converted into     *)
(* ASCII code, and the result is stored into sData.                              *)
  DBINDA_S_MD (X0, dData, sData);
```

● Corresponding MELSEC command

・DBINDA (BIN 32-bit → decimal ASCII conversion)

For the usable data type, refer to "3.2.2 About ANY type".

5.17.3 BIN→hexadecimal ASCII conversion BINHA_S_MD

The numeric value in each digit of the specified BIN 16-bit data represented in hexadecimal is converted into ASCII code data.

■ Function definition BOOL BINHA_S_MD (BOOL EN, ANY 16S1, STRING(6) D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 16-bit data)
D	OUT	Conversion result (hexadecimal ASCII code data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the numeric value in each digit of  *)
(* the BIN data stored in iData and represented in hexadecimal is converted  *)
(* into ASCII code, and the result is stored into sData.                     *)
  BINHA_S_MD (X0, iData, sData);
```

● Corresponding MELSEC command

・BINHA (BIN 16-bit → hexadecimal ASCII conversion);

5.17.4 32-bit BIN→hexadecimal ASCII conversion DBINHA_S_MD

The numeric value in each digit of the specified BIN 32-bit data represented in hexadecimal is converted into ASCII code data.

■ Function definition BOOL DBINHA_S_MD (BOOL EN, ANY32 S1, STRINGS (10) D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 32-bit data)
D	OUT	Conversion result (hexadecimal ASCII code data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the numeric value in each digit of  *)
(* the BIN data stored in dData and represented in hexadecimal is converted  *)
(* into ASCII code, and the result is stored into sData.                     *)
  DBINHA_S_MD (X0, dData, sData);
```

● Corresponding MELSEC command

・DBINHA (BIN 32-bit → hexadecimal ASCII conversion)

For the usable data type, refer to "3.2.2 About ANY type".

5.17.5 BCD 4-digit→decimal ASCII conversion BCDDA_S_MD

The numeric value in each digit of the specified BCD 4-digit data is converted into ASCII code.

■ Function definition BOOL BCDDA_S_MD (BOOL EN, ANY16 S1, STRING(6) D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BCD 4-digit data)
D	OUT	Conversion result (decimal ASCII code data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the numeric value in each digit of  *)
(* the BCD data stored in iData and represented in decimal is converted into  *)
(* ASCII code, and the result is stored into sData.                           *)
  BCDDA_S_MD (X0, iData, sData);
```

● Corresponding MELSEC command

・BCDDA (BCD 4-digit → decimal ASCII conversion)

5.17.6 BCD 8-digit→decimal ASCII conversion DBCDDA_S_MD

The numeric value in each digit of the specified BCD 8-digit data is converted into ASCII code.

■ Function definition BOOL DBCDDA_S_MD (BOOL EN, ANY32 S1, STRING (10) D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BCD 8-digit data)
D	OUT	Conversion result (decimal ASCII code data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the numeric value in each digit of  *)
(* the BCD data stored in dData and represented in decimal is converted into  *)
(* ASCII code, and the result is stored into sData.                           *)
  DBCDDA_S_MD (X0, dData, sData);
```

● Corresponding MELSEC command

・DBCDDA (BCD 8-digit → decimal ASCII conversion)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 75

5.17.7 Decimal ASCII→BIN conversion DABIN_S_MD

The specified decimal ASCII code data is converted into BIN 16-bit data.

■ Function definition BOOL DABIN_S_MD (BOOL EN, STRING (6) S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (decimal ASCII code data)
D	OUT	Conversion result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the decimal ASCII data stored      *)
(* in sData is converted into BIN 16-bit data, and the result is stored into iData.  *)
  DABIN_S_MD (X0, sData, iData);
```

● Corresponding MELSEC command

- DABIN (Decimal ASCII → BIN 16-bit conversion)

5.17.8 Decimal ASCII→32-bit BIN conversion DDABIN_S_MD

The specified decimal ASCII code data is converted into BIN 32-bit data.

■ Function definition BOOL DDABIN_S_MD (BOOL EN, STRING (11) S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (decimal ASCII code data)
D	OUT	Conversion result (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the decimal ASCII data stored in   *)
(* sData is converted into BIN 32-bit data, and the result is stored into dData.  *)
  DDABIN_S_MD (X0, sData, dData);
```

● Corresponding MELSEC command

- DDABIN (Decimal ASCII → BIN 32-bit conversion)

For the usable data type, refer to "3.2.2 About ANY type".

5.17.9 Hexadecimal ASCII→BIN conversion HABIN_S_MD

The specified hexadecimal ASCII code data is converted into BIN 16-bit data.

■ Function definition BOOL HABIN_S_MD (BOOL EN, STRING(4) S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (hexadecimal ASCII code data)
D	OUT	Conversion result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the hexadecimal ASCII data stored   *)
(* in sData is converted into BIN 16-bit data, and the result is stored into iData.  *)
 HABIN_S_MD (X0, sData, iData);
```

● Corresponding MELSEC command

- HABIN (Hexadecimal ASCII → BIN 16-bit conversion)

5.17.10 Hexadecimal ASCII→32-bit BIN conversion DHABIN_S_MD

The specified hexadecimal ASCII code data is converted into BIN 32-bit data.

■ Function definition BOOL DHABIN_S_MD (BOOL EN, STRING (8) S1, ANY32 D) ;

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (hexadecimal ASCII code data)
D	OUT	Conversion result (BIN 32-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the hexadecimal ASCII data stored in  *)
(* sData is converted into BIN 32-bit data, and the result is stored into dData.  *)
 DHABIN_S_MD (X0, sData, dData);
```

● Corresponding MELSEC command

- DHABIN (Hexadecimal ASCII → BIN 32-bit conversion)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 77

5.17.11 Decimal ASCII→BCD 4-digit conversion DABCD_S_MD

The specified decimal ASCII code data is converted into BCD 4-digit data.

■ Function definition BOOL DABCD_S_MD (BOOL EN, STRING(4) S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (decimal ASCII code data)
D	OUT	Conversion result (BCD 4-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the decimal ASCII data stored in      *)
(* sData is converted into BCD 4-digit data, and the result is stored into iData. *)
 DABCD_S_MD (X0, sData, iData);
```

● Corresponding MELSEC command

- DABCD (Decimal ASCII → BCD 4-digit conversion)

5.17.12 Decimal ASCII→BCD 8-digit conversion DDABCD_S_MD

The specified decimal ASCII code data is converted into BCD 8-digit data.

■ Function definition BOOL DDABCD_S_MD (BOOL EN, STRING(8) S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (decimal ASCII code data)
D	OUT	Conversion result (BCD 8-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the decimal ASCII data stored in      *)
(* sData is converted into BCD 8-digit data, and the result is stored into dData. *)
 DDABCD_S_MD (X0, sData, dData);
```

● Corresponding MELSEC command

- DDABCD (Decimal ASCII → BCD 8-digit conversion)

For the usable data type, refer to "3.2.2 About ANY type".

5.17.13 Device comment data read COMRD_S_MD

The comment of the specified device is read as ASCII code data.

■ Function definition BOOL COMRD_S_MD (BOOL EN, ANY_SIMPLE S1, STRING (32) D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data from which comment will be read
D	OUT	Comment read result (ASCII code data)

Remarks: The DINT, REAL and STRING types cannot be used in argument "S1".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the comment set in D100 is read,     *)
(* and stored into sData in ASCII code.                                        *)
   COMRD_S_MD (X0, D100, sData);
```

● Corresponding MELSEC command

- COMRD (Device comment data read)

5.17.14 Character string length detection LEN_S_MD

The length of the specified character string is obtained.

■ Function definition BOOL LEN_S_MD (BOOL EN, STRING S1, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data whose character string length will be detected (character string data)
D	OUT	Detection result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the length of the character string    *)
(* specified in sData is detected,  and stored into iData.                     *)
   LEN_S_MD (X0, sData, iData);
```

● Corresponding MELSEC command

- LEN (Character string length detection)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 79

5.17.15 BIN→character string conversion STR_S_MD

The specified BIN 16-bit data is converted into a character string with a decimal point added in the specified position.

■ Function definition BOOL STR_S_MD (BOOL EN, ANY32 S1, ANY16 S2, STRING(9) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Number of digits of numeric value to be converted (BIN 32-bit data)	S1	Total number of digits (2 to 8 digits)
			S1+1	Number of fraction part digits (0 to 5 digits)
S2	IN	Data to be converted (BIN 16-bit data)		
D	OUT	Conversion result (character string data)		

Remarks: The digits of a bit device cannot be specified in "S1".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BIN 16-bit data specified in  *)
(* iData is converted into a character string with a decimal point added in the  *)
(* position specified in dData, and the result is stored into sData.  *)
  STR_S_MD (X0, dData, iData, sData);
```

● Corresponding MELSEC command

- STR (BIN 16-bit → character string conversion)

5.17.16 32-bit BIN→character string conversion DSTR_S_MD

The specified BIN 32-bit data is converted into a character string with a decimal point added in the specified position.

■ Function definition BOOL DSTR_S_MD (BOOL EN, ANY32 S1, ANY32 S2 STRING(14) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Number of digits of numeric value to be converted (BIN 32-bit data)	S1	Total number of digits (2 to 8 digits)
			S1+1	Number of fraction part digits (0 to 5 digits)
S2	IN	Data to be converted (BIN 32-bit data)		
D	OUT	Conversion result (character string data)		

Remarks: The digits of a bit device cannot be specified in "S1".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BIN 32-bit data specified in  *)
(* dData1 is converted into a character string with a decimal point added in  *)
(* the position specified in dData2, and the result is stored into sData.  *)
  DSTR_S_MD (X0, dData1, dData2, sData);
```

● Corresponding MELSEC command

- DSTR (BIN 32-bit → character string conversion)

For the usable data type, refer to "3.2.2 About ANY type".

5.17.17 Character string→BIN conversion VAL_S_MD

The specified character string is converted into BIN 16-bit data, and its number of digits and BIN 16-bit data are obtained.

■ Function definition BOOL VAL_S_MD (BOOL EN, STRING (8) S1, ANY32 D1, ANY16 D2);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (character string data) Remarks: The number of characters in the fraction part of the character string specified in S1 is 0 to 5 characters. Note that the character string should be specified within the (total number of digits - 3) or less.
D1	OUT	Conversion result (number of digits) (BIN 32-bit data)
D2	OUT	Conversion result (BIN 16-bit data)

Remarks: The digits of a bit device cannot be specified in "D1".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string specified in      *)
(* sData is converted into BIN 16-bit data, the number of digits is stored      *)
(* into dData, and the BIN data is stored into iData.                           *)
   VAL_S_MD (X0, sData, dData, iData);
```

● Corresponding MELSEC command

・VAL (Character string → BIN 16-bit conversion)

5.17.18 Character string→32-bit BIN conversion DVAL_S_MD

The specified character string is converted into BIN 32-bit data, and its number of digits and BIN 32-bit data are obtained.

■ Function definition BOOL DVAL_S_MD (BOOL EN, STRING(13) S1, ANY32 D1, ANY32 D2);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (character string data) Remarks: The number of characters in the fraction part of the character string specified in S1 is 0 to 5 characters. Note that the character string should be specified within the (total number of digits - 3) or less.
D1	OUT	Conversion result (number of digits) (BIN 32-bit data)
D2	OUT	Conversion result (BIN 32-bit data)

Remarks: The digits of a bit device cannot be specified in "D1".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string specified        *)
(* in sData is converted into BIN 32-bit data,the number of digits is stored into *)
(* dData1, and the BIN data is stored into dData2.                              *)
   DVAL_S_MD (X0, sData, dData1, dData2);
```

● Corresponding MELSEC command

・DVAL (Character string → BIN 32-bit conversion)

For the usable data type, refer to "3.2.2 About ANY type".

5.17.19 Floating-point→character string conversion ESTR_M

The specified real number data is converted into a character string according to the specified display instruction.

■ Function definition BOOL ESTR_M (BOOL EN, REAL S1, ANY16 (3) S2, STRING (24) D);

Argument Name	IN/OUT	Description	
EN	IN	Execution condition (Function is executed only when the result is TRUE)	
S1	IN	Data to be converted (real number data)	
S2	IN	Display specification of numeric value to be converted	
		S2 [0]	Display format (0: decimal point format, 1: exponent format)
		S2 [1]	Total number of digits (2 to 24 digits) When the number of fraction part digits is "0" ..... Number of digits (max.: 24) ≥ 2 When the number of fraction part digits is other than "0" ..... Number of digits (max.: 24) ≥ (number of fraction part digits + 3)
		S2 [2]	Number of fraction part digits (0 to 7 digits)
D	OUT	Conversion result (character string data)	

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the real number data specified in      *)
(* rData is converted into a character string according to the display instruction *)
(* specified in ArrayData, and the result is stored into sData.                  *)
  ESTR_M (X0, rData, ArrayData, sData,);
```

● Corresponding MELSEC command

- ESTR (Floating-point data → character string conversion)

5.17.20 Character string→floating-point conversion EVAL_M

The specified character string is converted into real number data.

■ Function definition BOOL EVAL_M (BOOL EN. STRING (24) S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (character string data)
D	OUT	Conversion result (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string specified in      *)
(* sData is converted into real number data, and the result is stored into rData. *)
  EVAL_M (X0, sData, rData);
```

● Corresponding MELSEC command

- EVAL (Character string data _ floating-point conversion)

For the usable data type, refer to "3.2.2 About ANY type".

5.17.21 BIN→ASCII conversion ASC_S_MD

The specified BIN 16-bit data is converted into the hexadecimal ASCII data of the specified number of characters.

■ Function definition BOOL ASC_S_MD (BOOL EN, ANY16 S1, ANY16 n, STRING D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 16-bit data)
n	IN	Number of characters to be stored (BIN 16-bit data)
D	OUT	Conversion result (ASCII data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BIN 16-bit data specified in      *)
(* iData1 is converted into hexadecimal ASCII, and the result is stored into the *)
(* range of the number of characters specified in iData2, starting at the device *)
(* number specified in sData.                                                   *)
  ASC_S_MD (X0, iData1, iData2, sData);
```

● Corresponding MELSEC command

・ASC (BIN 16-bit data → ASCII conversion)

5.17.22 ASCII→BIN conversion HEX_S_MD

The hexadecimal ASCII data stored in the specified number of characters is converted into BIN 16-bit data.

■ Function definition BOOL HEX_S_MD (BOOL EN, STRING S1, ANY16n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (hexadecimal ASCII data)
n	IN	Number of characters to be converted (BIN 16-bit data)
D	OUT	Conversion result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the hexadecimal ASCII data stored     *)
(* in the number of characters specified in iData1, starting at the device number *)
(* specified in sData, is converted into a BIN value, and the result is stored into *)
(* iData2.                                                                      *)
  HEX_S_MD (X0, sData, iData1, iData2);
```

● Corresponding MELSEC command

・HEX (ASCII → BIN 16-bit conversion)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 83

5.17.23 Fetch from character string right side RIGHT_M

n characters of data, starting at the right of the specified character string data (end of the character string), are acquired.

■ Function definition BOOL RIGHT_M (BOOL EN, STRING S1, ANY16 n, STRING D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be acquired (character string data)
n	IN	Number of characters to be acquired (BIN 16-bit data)
D	OUT	Acquisition result (n characters of character string data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, iData characters of data, starting at  *)
(* the right of the character string specified in sData (end of the character string), *)
(* are stored into Result.                                                       *)
   RIGHT_M (X0, sData, iData, Result);
```

● Corresponding MELSEC command

- RIGHT (Fetch from right side of character string)

5.17.24 Fetch from character string left side LEFT_M

n characters of data, starting at the left of the specified character string data (head of the character string), are acquired.

■ Function definition BOOL LEFT_M (BOOL EN, STRING S1, ANY16 n, STRING D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be acquired (character string data)
n	IN	Number of characters to be acquired (BIN 16-bit data)
D	OUT	Acquisition result (n characters of character string data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, iData characters of data, starting at  *)
(* the left of the character string specified in sData (head of the character string), *)
(* are stored into Result.                                                       *)
   LEFT_M (X0, sData, iData, Result);
```

● Corresponding MELSEC command

- LEFT (Fetch from left side of character string)

For the usable data type, refer to "3.2.2 About ANY type".

5.17.25 Any data fetch in character string MIDR_M

S2[1] characters of data, starting at S2[0] of the specified character data, are acquired.

■ Function definition BOOL MIDR_M (BOOL EN, STRING S1, ANY16(2) S2 , STRING D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Data to be acquired (character string data)		
S2	IN	Position of first character and storage destination of characters to be acquired (ARRAY [0..1] OF ANY16)	S2 [0]	Position of first character
			S2 [1]	Number of acquired characters
D	OUT	Acquisition result (character string data)		

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of      *)
(* characters specified in StrArray [1] from the position specified in StrArray [0],  *)
(* starting at the left of the character specified in sData (head of the character  *)
(* string), are stored into Result.                                      *)
  MIDR_M (X0, sData, StrArray, Result);
```

● Corresponding MELSEC command

- MIDR (Any data fetch in character string)

5.17.26 Any data replacement in character string MIDW_M

The data of the number of characters specified in S2[1] are stored into the position, starting at S2[0], of the specified character string data.

■ Function definition BOOL MIDW_M (BOOL EN, STRING S1, ANY16(S) S2, STRING D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Data to be acquired (character string data)		
S2	IN	Position of first character and storage destination of characters to be acquired (ARRAY [0..1] OF ANY16)	S2 [0]	Position of first character of replacement destination
			S2 [1]	Number of acquired characters
D	IN/OUT	Data to be replaced, replacement result (character string data)		

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data of the number of characters  *)
(* specified in StrArray [1], starting at the left of the character specified in sData  *)
(* (head of the character string), are stored into the position specified in StrArray  *)
(* [0], starting at the left of the character string data stored in sData2.  *)
  MIDW_M (X0, sData1, StrArray, sData2);
```

● Corresponding MELSEC command

- MIDW (Any data replacement in character string)

For the usable data type, refer to "3.2.2 About ANY type".

5.17.27 Character string search INSTR_M

A searched for the specified character string data is performed, starting at the "n"th character from the left of the specified character string data.

■ Function definition BOOL INSTR_M (BOOL EN, STRING S1, STRING S2, ANY16 n, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be searched for (character string data)
S2	IN	Data to be searched (character string data)
n	IN	Search start position (at the "n"th character from left) (BIN 16-bit data)
D	OUT	Search result (character position from head of character string data specified in S2) (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, a search for the character string      *)
(* specified in sData1 is performed, starting at the iData character from the left of *)
(* the character string specified in sData2 (head of the character string),and the  *)
(* search result is stored into Result.                                          *)
   INSTR_M (X0, sData1, sData2, iData, Result) ;
```

● Corresponding MELSEC command

- INSTR (Character string search)

5.17.28 Floating-point→BCD decomposition EMOD_M

The specified real number data is decomposed into the BCD type floating-point format based on the specified fraction part digits.

■ Function definition BOOL EMOD_M (BOOL EN, REAL S1, ANY16 S2, ANY16(5) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Data to be decomposed (real number data)		
S2	IN	Fraction part digit data (BIN 16-bit data)		
D	OUT	BCD-decomposed data storage destination (ARRAY[0..4] OF ANY16)	D[0]	Sign (positive: 0, negative: 1)
			D[1]	BCD 7 digits
			D[2]	
			D[3]	Exponent part sign (positive: 0, negative: 1)
			D[4]	BCD exponent

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the real number data specified in      *)
(* rData is decomposed into the BCD type floating-point format based on the      *)
(* fraction part digits specified in iData, and the result is stored into Result.  *)
   EMOD_M (X0, rData, iData, Result);
```

● Corresponding MELSEC command

- EMOD (Floating-point data → BCD decomposition);

For the usable data type, refer to "3.2.2 About ANY type".

5.17.29 BCD format data→floating-point EREXP_M

The specified BCD type floating-point format data is converted into real number data based on the specified fraction part digits.

■ Function definition BOOL EREXP_M (BOOL EN, ANY16 S1, ANY16 S2, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BCD type floating-point format data)
S2	IN	Fraction part digit data (BIN 16-bit data)
D	OUT	Conversion result (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the BCD type floating-point format   *)
(* data specified in iData1 is converted into real number data based on the    *)
(* fraction part digits specified in iData2, and the result is stored into Result.  *)
(* real number data based on the fraction part digits specified in iData2, and  *)
(* the result is stored into Result.                                           *)
  EREXP_M (X0, iData1, iData2, Result);
```

● Corresponding MELSEC command

- EREXP (BCD format data → floating-point)

For the usable data type, refer to "3.2.2 About ANY type".

5 - 87

5.18 Special Functions

5.18.1 Floating-point SIN operation SIN_E_MD

The SIN (sine) value of the specified angle is operated.

■ Function definition BOOL SIN_E_MD (BOOL EN, REAL S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Angle data to be SIN (sine) operated (real number data) Remarks: Set the specified angle in radian unit (angle×π/180).
D	OUT	Operation result (SIN value) (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the SIN value of the angle specified   *)
(* in rData is calculated, and the result is stored into Result.                 *)
  SIN_E_MD (X0, rData, Result);
```

● Corresponding MELSEC command

・SIN (SIN operation on floating-point data (Single precision))

5.18.2 Floating-point COS operation COS_E_MD

The COS (cosine) value of the specified angle is operated.

■ Function definition BOOL COS_E_MD (BOOL EN, REAL S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Angle data to be COS (cosine) operated (real number data) Remarks: Set the specified angle in radian unit (angle×π/180).
D	OUT	Operation result (COS value) (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the COS value of the angle             *)
(* specified in rData is calculated, and the result is stored into Result.       *)
  COS_E_MD (X0, rData, Result)
```

● Corresponding MELSEC command

・COS (COS operation on floating-point data (Single precision))

5.18.3 Floating-point TAN operation TAN_E_MD

The TAN (tangent) value of the specified angle is operated.

■ Function definition BOOL TAN_E_MD (BOOL EN, REAL S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Angle data to be TAN (tangent) operated (real number data) Remarks: Set the specified angle in radian unit (angle × π /180).
D	OUT	Operation result (TAN value) (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the TAN value of the angle specified  *)
(* in rData is calculated, and the result is stored into Result.                *)
  TAN_E_MD (X0, rData, Result);
```

● Corresponding MELSEC command

・TAN (TAN operation on floating-point data (Single precision))

5.18.4 Floating-point SIN^{-1} operation ASIN_E_MD

The SIN^{-1} (arcsine) operation of the specified SIN value is performed.

■ Function definition BOOL ASIN_E_MD (BOOL EN, REAL S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be operated, SIN value (-1.0 to 1.0) (real number data)
D	OUT	Operation result (angle data in radian unit) (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the angle is operated from the SIN  *)
(* value specified in rData, and the result is stored into Result.           *)
  ASIN_E_MD (X0, rData, Result);
```

● Corresponding MELSEC command

・ASIN (SIN^{-1} operation on floating-point data (Single precision))

5.18.5 Floating-point COS^{-1} operation ACOS_E_MD

The COS^{-1} (arccosine) operation of the specified COS value is performed.

■ Function definition BOOL ACOS_E_MD (BOOL EN, REAL S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be operated, COS value (-1.0 to 1.0) (real number data)
D	OUT	Operation result (angle data in radian unit) (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the angle is operated from the COS   *)
(* value specified in rData, and the result is stored into Result.            *)
  ACOS_E_MD (X0, rData, Result);
```

● Corresponding MELSEC command

・ACOS (COS^{-1} operation on floating-point data (Single precision))

5.18.6 Floating-point TAN^{-1} operation ATAN_E_MD

The TAN^{-1} (arctangent) operation of the specified TAN value is performed.

■ Function definition BOOL ATAN_E_MD (BOOL EN, REAL S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be operated, TAN value (real number data)
D	OUT	Operation result (angle data in radian unit) (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the angle is operated from the TAN   *)
(* value specified in rData, and the result is stored into Result.            *)
  ATAN_E_MD (X0, rData, Result);
```

● Corresponding MELSEC command

・ATAN (TAN^{-1} operation on floating-point data (Single precision))

5.18.7 Floating-point angle→radian RAD_E_MD

The unit of magnitude of the specified angle is converted from the degree unit to the radian unit.

■ Function definition BOOL RAD_E_MD (BOOL EN, REAL S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted, angle data in degree unit (real number data)
D	OUT	Conversion result (radian unit) (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the angle data of degree unit specified  *)
(* in rData is converted into the radian unit, and the result is stored into Result.  *)
  RAD_E_MD (X0, rData, Result);
```

● Corresponding MELSEC command

・RAD (Conversion from floating-point angle to radian (Single precision))

5.18.8 Floating-point radian→angle conversion DEG_E_MD

The unit of magnitude of the specified angle is converted from the radian unit to the degree unit.

■ Function definition BOOL DEG_E_MD (BOOL EN, REAL S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted, radian value data (real number data)
D	OUT	Conversion result (degree unit) (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the unit of magnitude of the angle is  *)
(* converted from the radian unit to the degree unit, and the result is stored into Result. *)
  DEG_E_MD (X0, rData, Result);
```

● Corresponding MELSEC command

・DEG (Conversion from floating-point radian to angle (Single precision))

5.18.9 Floating-point square root SQR_E_MD

The square root of the specified value is operated.

■ Function definition BOOL SQR_E_MD (BOOL EN, REAL S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be operated (only positive value can be specified) (real number data)
D	OUT	Operation result (real number data)

Remarks: The value to be specified in "S1" is a positive number only.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the square root of the value specified  *)
(* in rData is operated, and the result is stored into Result.                    *)
  SQR_E_MD (X0, rData, Result);
```

● Corresponding MELSEC command

・SQR (Square root operation for floating-point data (Single precision))

5.18.10 Floating-point natural exponential operation EXP_E_MD

The base e natural exponent of the specified value is operated.

■ Function definition BOOL EXP_E_MD (BOOL EN, REAL S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Exponent part data to be operated (real number data)
D	OUT	Operation result (e^{S1}) (real number data)

Remarks: Operation is performed on the assumption that the base (e) is "2.71828".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, natural exponential operation  *)
(* performed using rData as an exponent is, and the result is stored into Result.  *)
  EXP_E_MD (X0, rData, Result);
```

● Corresponding MELSEC command

・EXP (Exponent operation on floating-point data (Single precision))

5.18.11 Floating-point natural logarithm operation LOG_E_MD

The base e logarithm (natural logarithm) of the specified value is operated.

■ Function definition BOOL LOG_E_MD (BOOL EN, REAL S1, REAL D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be operated (only positive value can be specified) (real number data)
D	OUT	Operation result ($\log_e$S1) (real number data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the base e logarithm (natural logarithm) *)
(* of the value specified in rData is operated, and the result is stored into Result. *)
  LOG_E_MD (X0, rData, Result);
```

● Corresponding MELSEC command

・LOG (Natural logarithm operation on floating-point data (Single precision))

5.18.12 Random number generation RND_M

Random numbers of 0 to 32767 are generated.

■ Function definition BOOL RND_M (BOOL EN, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
D	OUT	Random number generation result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, random numbers of 0 to 32767 are    *)
(* generated and stored into Result.                                          *)
  RND_M (X0, Result);
```

● Corresponding MELSEC command

・RND (Random number generation)

For the usable data type, refer to "3.2.2 About ANY type".

5.18.13 Sequence change SRND_M

The random number sequence is changed according to the specified 16-bit BIN data.

■ Function definition BOOL SRND_M (BOOL EN, ANY16 S1);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Random number sequence change result (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the random number sequence is changed *)
(* according to the 16-bit BIN data stored in the device specified in iData.    *)
  SRND_M (X0, iData);
```

● Corresponding MELSEC command

・SRND (Series updates)

5.18.14 BCD 4-digit square root BSQR_MD

The square root of the specified BCD 4-digit data is operated.

■ Function definition BOOL BSQR_MD (BOOL EN, ANY16 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	BCD 4-digit data to be operated (BIN 16-bit data)
D	OUT	Operation result (BIN 32-bit data)

Remarks: The digits of a bit device cannot be specified in "D".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the square root of the value specified *)
(* in iData is operated, and the result is stored into Result.                   *)
  BSQR_MD (X0, iData, dData);
```

● Corresponding MELSEC command

・BSQR (BCD 4-digit square root)

For the usable data type, refer to "3.2.2 About ANY type".

5.18.15 BCD 8-digit square root BDSQR_MD

The square root of the specified BCD 8-digit data is operated.

■ Function definition BOOL BDSQR_MD (BOOL EN, ANY32 S1, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	BCD 8-digit data to be operated (BIN 32-bit data)
D	OUT	Operation result (BIN 32-bit data)

Remarks: The digits of a bit device cannot be specified in "D".

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the square root of the value specified in  *)
(* dData is operated, and the result is stored into Result.                          *)
  BDSQR_MD (X0, dData, Result);
```

● Corresponding MELSEC command

・BDSQR (BCD 8-digit square root)

5.18.16 BCD type SIN operation BSIN_MD

The BCD 4-digit data of the specified angle is SIN (sine) operated.

■ Function definition BOOL BSIN_MD (BOOL EN, ANY16 S1, ANY16(3) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Data to be operated (BCD 4-digit data)		
D	OUT	Operation result (ARRAY [0..2] OF ANY16)	D [0]	Sign (positive: 0, negative: 1)
			D [1]	Integer part (BCD 4-digit data)
			D [2]	Fraction part (BCD 4-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the SIN value of the angle specified in iData is *)
(* operated, the sign of the operation result is stored into ArrayData [0], the integer part of *)
(* the operation result into ArrayData [1], and the fraction part into ArrayData [2]. *)
  BSIN_MD (X0, iData, ArrayData);
```

● Corresponding MELSEC command

・BSIN (BCD type SIN operation)

For the usable data type, refer to "3.2.2 About ANY type".

5.18.17 BCD type COS operation BCOS_MD

The BCD 4-digit data of the specified angle is COS (cosine) operated.

■ Function definition BOOL BCOS_MD (BOOL EN, ANY16 S1, ANY16(3) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Data to be operated (BCD 4-digit data)		
D	OUT	Operation result (ARRAY [0..2] OF ANY16)	D [0]	Sign (positive: 0, negative: 1)
			D [1]	Integer part (BCD 4-digit data)
			D [2]	Fraction part (BCD 4-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the COS value of the angle specified in iData is *)
(* operated, the sign of the operation result is stored into ArrayData [0], the integer part of *)
(* the operation result into ArrayData [1], and the fraction part into ArrayData [2]. *)
  BCOS_MD (X0, iData, ArrayData);
```

● Corresponding MELSEC command

・BCOS (BCD type COS operation)

5.18.18 BCD type TAN operation BTAN_MD

The BCD 4-digit data of the specified angle is TAN (tangent) operated.

■ Function definition BOOL BTAN_MD (BOOL EN, ANY16 S1, ANY16(3) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Data to be operated (BCD 4-digit data)		
D	OUT	Operation result (ARRAY [0..2] OF ANY16)	D [0]	Sign (positive: 0, negative: 1)
			D [1]	Integer part (BCD 4-digit data)
			D [2]	Fraction part (BCD 4-digit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the TAN value of the angle specified in iData is *)
(* operated, the sign of the operation result is stored into ArrayData [0], the integer part of *)
(* the operation result into ArrayData [1], and the fraction part into ArrayData [2]. *)
  BTAN_MD (X0, iData, ArrayData);
```

● Corresponding MELSEC command

・BTAN (BCD type TAN operation)

For the usable data type, refer to "3.2.2 About ANY type".

5.18.19 BCD type SIN^{-1} operation BASIN_MD

The SIN^{-1} (arcsine) value of the specified BCD value is operated.

■ Function definition BOOL BASIN_MD (BOOL EN, ANY16(3) S1, ANY16 D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Operation result (ARRAY [0..2] OF ANY16)	S [0]	Sign (positive: 0, negative: 1)
			S [1]	Integer part (BCD 4-digit data)
			S [2]	Fraction part (BCD 4-digit data)
D	OUT	Operation result (head number of device) (BCD 4-digit data)		

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the SIN-1 value of the value specified in  *)
(* BasinArrayData is operated, and the result is stored into Result.                *)
  BASIN_MD (X0, BasinArrayData, Result);
```

● Corresponding MELSEC command

・BASIN (BCD type SIN^{-1} operation)

5.18.20 BCD type COS^{-1} operation BACOS_MD

The COS^{-1} (arccosine) value of the specified BCD value is operated.

■ Function definition BOOL BACOS_MD (BOOL EN, ANY16(3) S1, ANY16 D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Data to be COS^{-1} (arccosine) operated (ARRAY [0..2] OF ANY16)	S [0]	Sign (positive: 0, negative: 1)
			S [1]	Integer part (BCD 4-digit data)
			S [2]	Fraction part (BCD 4-digit data)
D	OUT	Operation result (head number of device) (BCD 4-digit data)		

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the COS-1 value of the value specified  *)
(* in BacosArrayData is operated, and the result is stored into Result.          *)
  BACOS_MD (X0, BacosArrayData, Result);
```

● Corresponding MELSEC command

・BACOS (BCD type COS^{-1} operation)

For the usable data type, refer to "3.2.2 About ANY type".

5.18.21 BCD type TAN^{-1} operation BATAN_MD

The TAN^{-1} (arctangent) value of the specified BCD value is operated.

■ Function definition BOOL BATAN_MD (BOOL EN, ANY16(3) S1, ANY16 D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Head number of device that stores data to be operated (ARRAY [0..2] OF ANY16)	S [0]	Sign (positive: 0, negative: 1)
			S [1]	Integer part (BCD 4-digit data)
			S [2]	Fraction part (BCD 4-digit data)
D	OUT	Operation result (BCD 4-digit data)		

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the TAN-1 value of the value specified in  *)
(* BatanArrayData is operated, and the result is stored into Result.                *)
  BATAN_MD (X0, BatanArrayData, Result);
```

● Corresponding MELSEC command

・BATAN (BCD type TAN^{-1} operation)

For the usable data type, refer to "3.2.2 About ANY type".

5.19 Data Control

5.19.1 Upper/lower limit control LIMIT_MD

The output value is controlled depending on whether the specified BIN 16-bit data is within the upper/lower limit value range or not.

■ Function definition BOOL LIMIT_MD (BOOL EN, ANY16 S1, ANY16 S2, ANY16 S3, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Lower limit value (BIN 16-bit data)
S2	IN	Upper limit value (BIN 16-bit data)
S3	IN	Input value (BIN 16-bit data)
D	OUT	Output value (BIN 16-bit data)

Remarks: The output value is controlled as described below.

When S1 (lower limit value) > S3 (input value)
..... S1 (lower limit value) → D (output value)
When S2 (upper limit value) < S3 (input value)
.....S2 (upper limit value) → D (output value)
When S1 (lower limit value) ≤ S3 (input value) ≤ S2 (upper limit value)
.....S3 (input value) → D (output value)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the output value is stored into Result  *)
(* depending on whether or not the input value specified in iData3 is within the  *)
(* per/lower limit value range specified in iData1 and iData 2.                   *)
  LIMIT_MD (X0, iData1, iData2, iData3, Result);
```

● Corresponding MELSEC command

・LIMIT (16-bit upper/lower limit control)

For the usable data type, refer to "3.2.2 About ANY type".

5.19.2 32-bit data upper/lower limit control DLIMIT_MD

The output value is controlled depending on whether the specified BIN 32-bit data is within the upper/lower limit value range or not.

■ Function definition BOOL DLIMIT_MD (BOOL EN, ANY32 S1, ANY32 S2, ANY32 S3, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Lower limit value (BIN 32-bit data)
S2	IN	Upper limit value (BIN 32-bit data)
S3	IN	Input value (BIN 32-bit data)
D	OUT	Output value (BIN 32-bit data)

Remarks: The output value is controlled as described below.

When S1 (lower limit value) > S3 (input value)
..... S1 (lower limit value) → D (output value)
When S2 (upper limit value) < S3 (input value)
.....S2 (upper limit value) → D (output value)
When S1 (lower limit value) ≤ S3 (input value) ≤ S2 (upper limit value)
.....S3 (input value) → D (output value)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the output value is stored into Result   *)
(* depending on whether or not the input value specified in dData3 is within the  *)
(* upper/lower limit value range specified in dData1 and dData 2.                 *)
  DLIMIT_MD (X0, dData1, dData2, dData3, Result);
```

● Corresponding MELSEC command

・DLIMIT (32-bit upper/lower limit control)

For the usable data type, refer to "3.2.2 About ANY type".

5.19.3 Dead band control BAND_MD

The output value is controlled depending on whether the specified BIN 16-bit data is within the upper/lower limit range of the specified dead band or not.

■ Function definition BOOL BAND_MD (BOOL EN, ANY16 S1, ANY16 S2, ANY16 S3, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Lower limit value data of dead band (BIN 16-bit data)
S2	IN	Upper limit value data of dead band (BIN 16-bit data)
S3	IN	Input value (BIN 16-bit data)
D	OUT	Output value (BIN 16-bit data)

Remarks: The output value is controlled as described below.

When S1 (lower limit value) > S3 (input value)

..... S3 (input value) - S1 (lower limit value) → D (output value)

When S2 (upper limit value) < S3 (input value)

.....S3 (input value) - S2 (upper limit value) → D (output value)

When S1 (lower limit value) ≤ S3 (input value) ≤ S2 (upper limit value)

.....0 → D (output value)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the output value is stored into Result  *)
(* depending on whether or not the input value specified in iData3 is within the  *)
(* upper/lower limit range of the dead band specified in iData1 and iData2.        *)
  BAND_MD (X0, iData1, iData2, iData3, Result);
```

● Corresponding MELSEC command

・BAND (16-bit dead band control)

For the usable data type, refer to "3.2.2 About ANY type".

5.19.4 32-bit data dead band control DBAND_MD

The output value is controlled depending on whether the specified BIN 32-bit data is within the upper/lower limit range of the specified dead band or not.

■ Function definition BOOL DBAND_MD (BOOL EN, ANY32 S1, ANY32 S2, ANY32 S3, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Lower limit value data of dead band (BIN 32-bit data)
S2	IN	Upper limit value data of dead band (BIN 32-bit data)
S3	IN	Input value (BIN 32-bit data)
D	OUT	Output value (BIN 32-bit data)

Remarks: The output value is controlled as described below.

When S1 (lower limit value) > S3 (input value)

..... S3 (input value) - S1 (lower limit value) → D (output value)

When S2 (upper limit value) < S3 (input value)

.....S3 (input value) - S2 (upper limit value) → D (output value)

When S1 (lower limit value) ≤ S3 (input value) ≤ S2 (upper limit value)

.....0 → D (output value)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the output value is stored into Result  *)
(* depending on whether or not the input value specified in iData3 is within the  *)
(* upper/lower limit range of the dead band specified in iData1 and iData2.        *)
  DBAND_MD (X0, dData1, dData2, dData3, Result);
```

● Corresponding MELSEC command

・DBAND (32-bit dead band control)

For the usable data type, refer to "3.2.2 About ANY type".

5.19.5 Bit zone control ZONE_MD

The output value is zone-controlled with a bias value added to the specified BIN 16-bit data.

■ Function definition BOOL ZONE_MD (BOOL EN, ANY16 S1, ANY16 S2, ANY16 S3, ANY16 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Negative bias value added to input value (BIN 16-bit data)
S2	IN	Positive bias value added to input value (BIN 16-bit data)
S3	IN	Input value (BIN 16-bit data)
D	OUT	Output value (BIN 16-bit data)

Remarks: The output value is controlled as described below.

When S3 (input value) < 0

.....S3 (input value) + S1 (negative bias value) → D (output value)

When S3 (input value) = 0.....0 → D (output value)

When S3 (input value) > 0

..... S3 (input value) + S1 (positive bias value) → D (output value)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the bias value specified in iData1 or  *)
(* iData2 is added to the input value specified in iData3, and the result is stored  *)
(* into Result.                                                                    *)
  ZONE_MD (X0, iData1, iData2, iData3, Result);
```

● Corresponding MELSEC command

・ZONE (16-bit zone control)

For the usable data type, refer to "3.2.2 About ANY type".

5.19.6 32-bit data bit zone control DZONE_MD

The output value is zone-controlled with a bias value added to the specified BIN 32-bit data.

■ Function definition BOOL DZONE_MD (BOOL EN, ANY32 S1, ANY32 S2, ANY32 S3, ANY32 D);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Negative bias value added to input value (BIN 32-bit data)
S2	IN	Positive bias value added to input value (BIN 32-bit data)
S3	IN	Input value (BIN 32-bit data)
D	OUT	Output value (BIN 32-bit data)

Remarks: The output value is controlled as described below.

When S3 (input value) < 0

.....S3 (input value) + S1 (negative bias value) → D (output value)

When S3 (input value) = 0.....0 → D (output value)

When S3 (input value) > 0

..... S3 (input value) + S1 (positive bias value) → D (output value)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the bias value specified in iData1 or  *)
(* iData2 is added to the input value specified in iData3, and the result is stored  *)
(* into Result.                                                                    *)
  DZONE_MD (X0, dData1, dData2, dData3, Result);
```

● Corresponding MELSEC command

・DZONE (32-bit zone control)

For the usable data type, refer to "3.2.2 About ANY type".

5.19.7 File register block No. switching RSET_MD

The block No. of the file registers used in a program is changed into the specified block No.

■ Function definition BOOL RSET_MD (BOOL EN, ANY16 S1);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	New block No. data (BIN 16-bit data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the block No. of the file registers used *)
(* in the program is changed into the block No. stored in the device specified in iData. *)
  RSET_MD (X0, iData);
```

● Corresponding MELSEC command

・RSET (File register block No. switching)

5.19.8 Set of file register file QDRSET_M

The file name of the file registers used in a program is changed into the specified file name.

■ Function definition BOOL QDRSET_M (BOOL EN, STRING S1);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	"Drive No.: File name" of target file registers (character string data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the file name of the file registers of     *)
(* drive No. 1 is changed into "ABS.QDR".                                           *)
  QDRSET_M (X0, "1: ABC");
```

● Corresponding MELSEC command

・QDRSET (Set of file register file)

For the usable data type, refer to "3.2.2 About ANY type".

5- 105

5.19.9 Set of comment file QCDSET_M

The file name of the comment file used in a program is changed into the specified file name.

■ Function definition BOOL QDRSET_M (BOOL EN, STRING S1);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	"Drive No.: File name" of target comment file (character string data)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the file name of the comment file of  *)
(* drive No. 3 is changed into "DEF.QCD".                                       *)
  QCDSET_M (X0, "3: DEF");
```

● Corresponding MELSEC command

・QCDSET (Set of comment file)

5.20 Clock

5.20.1 Read of clock data DATERD_MD

The "year, month, day, hour, minute, second, day of week" is read from the clock element of the QCPU/LCPU. They are stored into the specified destination as BIN values.

■ Function definition BOOL DATERD_MD (BOOL EN, ANY16(7) S);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
D	OUT	Read clock data (ARRAY [0..6] OF ANY16)	D [0]	Year (1980 to 2079)
			D [1]	Month (1 to 12)
			D [2]	Day (1 to 31)
			D [3]	Hour (0 to 23)
			D [4]	Minute (0 to 59)
			D [5]	Second (0 to 59)
			D [6]	Day of week (0 to 6)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the "year, month, day, hour, minute,  *)
(* second, day of week" are read from the clock element of the QCPU/LCPU,       *)
(* and stored into the device specified in TimeData as BIN values.              *)
  DATERD_MD (X0, TimeData);
```

● Corresponding MELSEC command

・DATERD (Read of clock data)

For the usable data type, refer to "3.2.2 About ANY type".

5.20.2 Write of clock data DATEWR_MD

The clock data "year, month, day, hour, minute, second, day of week" are written to the clock element of the QCPU/LCPU.

■ Function definition BOOL DATEWR_MD (BOOL EN, ANY16(7) S);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S	IN	Clock data to be written (ARRAY [0..6] OF ANY16)	S [0]	Year (1980 to 2079)
			S [1]	Month (1 to 12)
			S [2]	Day (1 to 31)
			S [3]	Hour (0 to 23)
			S [4]	Minute (0 to 59)
			S [5]	Second (0 to 59)
			S [6]	Day of week (0 to 6)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the clock data stored in TimeData are *)
(* written to the clock element of the QCPU/LCPU.                               *)
  DATEWR_MD (X0, TimeData);
```

● Corresponding MELSEC command

・DATEWR (Write of clock data)

For the usable data type, refer to "3.2.2 About ANY type".

5.20.3 Addition of clock data DATEPLUS_M

The specified time data is added to the specified time-of-day data.

■ Function definition BOOL DATEPLUS_M (BOOL EN, ANY16(3) S1, ANY16(3) S2, ANY16(3) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Time-of-day data to which data will be added (ARRAY [0..2] OF ANY16)	S1 [0]	Hour (0 to 23)
			S1 [1]	Minute (0 to 59)
			S1 [2]	Second (0 to 59)
S2	IN	Time data that will be added to data (ARRAY [0..2] OF ANY16)	S2 [0]	Hour (0 to 23)
			S2 [1]	Minute (0 to 59)
			S2 [2]	Second (0 to 59)
D	OUT	Addition result time-of-day data (ARRAY [0..2] OF ANY16)	D [0]	Hour (0 to 23)
			D [1]	Minute (0 to 59)
			D [2]	Second (0 to 59)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the time data specified in TimeData2  *)
(* is added to the time-of-day data specified in TimeData1, and the addition    *)
(* result is stored into Result.                                                *)
  DATEPLUS_M (X0, TimeData1, TimeData2, Result);
```

● Corresponding MELSEC command

・DATE+ (Addition of clock data)

For the usable data type, refer to "3.2.2 About ANY type".

5- 109

5.20.4 Subtraction of clock data DATEMINUS_M

The specified time data is subtracted from the specified time-of-day data.

■ Function definition BOOL DATEMINUS_M (BOOL EN, ANY16(3) S1, ANY16(3) S2, ANY16(3) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Time-of-day data from which data will be subtracted (ARRAY [0..2] OF ANY16)	S1 [0]	Hour (0 to 23)
			S1 [1]	Minute (0 to 59)
			S1 [2]	Second (0 to 59)
S2	IN	Time data that will be subtracted from data (ARRAY [0..2] OF ANY16)	S2 [0]	Hour (0 to 23)
			S2 [1]	Minute (0 to 59)
			S2 [2]	Second (0 to 59)
D	OUT	Subtraction result time-of-day data (ARRAY [0..2] OF ANY16)	D [0]	Hour (0 to 23)
			D [1]	Minute (0 to 59)
			D [2]	Second (0 to 59)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the time data specified in TimeData2  *)
(* is subtracted from the time-of-day data specified in TimeData1, and the      *)
(* subtraction result is stored into Result.                                    *)
  DATEMINUS_M (X0, TimeData1, TimeData2, Result);
```

● Corresponding MELSEC command

・DATE- (Subtraction of clock data)

For the usable data type, refer to "3.2.2 About ANY type".

5.20.5 Clock data format conversion (hour, minute, second→second) SECOND_M

The specified time data is converted into second.

■ Function definition BOOL SECOND_M (BOOL EN, ANY16(3) S, ANY32 D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S	IN	Clock data to be converted (ARRAY [0..2] OF ANY16)	S [0]	Hour (0 to 23)
			S [1]	Minute (0 to 59)
			S [2]	Second (0 to 59)
D	OUT	Conversion result clock data (second) (BIN 32-bit data)		

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the time data specified in TimeData  *)
(* is converted into second, and the result is stored into Result.             *)
  SECOND_M (X0, TimeData, Result);
```

● Corresponding MELSEC command

・SECOND (Clock data format conversion)

5.20.6 Clock data format conversion (second→hour, minute, second) HOUR_M

The specified data in second is converted into hour, minute, second.

■ Function definition BOOL HOUR_M (BOOL EN, ANY32 S1, ANY16(3) D);

Argument Name	IN/OUT	Description		
EN	IN	Execution condition (Function is executed only when the result is TRUE)		
S1	IN	Clock data to be converted (second) (BIN 32-bit data)		
D	OUT	Conversion result clock data (ARRAY [0..2] OF ANY16)	D [0]	Hour (0 to 23)
			D [1]	Minute (0 to 59)
			D [2]	Second (0 to 59)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data in second specified in dData  *)
(* is converted into hour, day, second, and the result is stored into Result.   *)
  HOUR_M (X0, dData, TimeData);
```

● Corresponding MELSEC command

・HOUR (Clock data format conversion)

For the usable data type, refer to "3.2.2 About ANY type".

5.21 Program Control

5.21.1 Program standby PSTOP_M

The program of the specified file name is put in a standby status.

■ Function definition BOOL PSTOP_M (BOOL EN, STRING S1);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	File name of program to be put in standby status (character string data)

Remarks: Only the program stored in the program memory (drive No.: 0) can be placed in a standby status.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the program whose file name is  *)
(* "ABC" is placed in a standby status.                                   *)
  PSTOP_M (X0, "ABC");
```

● Corresponding MELSEC command

・PSTOP (Program standby command)

5.21.2 Program output OFF standby POFF_M

The program of the specified file name is brought into non-execution and put in a standby status.

■ Function definition BOOL POFF_M (BOOL EN, STRING S1);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	File name of program to be brought into non-execution and put in a standby status (character string data)

Remarks: Only the program stored in the program memory (drive No.: 0) can be brought into non-execution and placed in a standby status.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the program whose file name is  *)
(* "ABC" is brought into non-execution and placed in a standby status.    *)
  POFF_M (X0, sData);
```

● Corresponding MELSEC command

・POFF (Program output OFF standby command)

5.21.3 Program scan execution registration PSCAN_M

The program of the specified file name is put in a scan execution status.

■ Function definition BOOL PSCAN_M (BOOL EN, STRING S1);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	File name of program to be put in scan execution status (character string data)

Remarks: Only the program stored in the program memory (drive No.: 0) can be placed in a scan execution status.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the program whose file name is    *)
(* "ABC" is placed in a scan execution status.                              *)
  PSCAN_M (X0, sData);
```

● Corresponding MELSEC command

・PSCAN (Program scan execution registration command)

5.21.4 Program low-speed execution registration PLOW_M

The program of the specified file name is put in a low-speed execution status.

■ Function definition BOOL PLOW_M (BOOL EN, STRING S1);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	File name of program to be put in low-speed execution status (character string data)

Remarks: Only the program stored in the program memory (drive No.: 0) can be placed in a low-speed execution status.

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the program whose file name is    *)
(* "ABC" is placed in a low-speed execution status.                         *)
  PLOW_M (X0, "ABC");
```

● Corresponding MELSEC command

・PLOW (Program low-speed execution registration instruction)

5.22 Others

5.22.1 WDT reset WDT_M

The watchdog timer is reset in a sequence program.

■ Function definition BOOL WDT _M (BOOL EN);

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the watchdog timer is reset in the *)
(* sequence program. *)
  WDT_M (X0);
```

● Corresponding MELSEC command

・WDT (Watchdog timer reset)

6 IEC FUNCTIONS

How the functions are described

This manual describes the function definitions, arguments, return values and using examples of the IEC functions.
The IEC functions are created by combining the MELSEC common instructions. For the applicable devices of the IEC functions, the errors that may occur during execution of the functions, and the applicable CPU types, refer to the "MELSEC-Q/L Programming Manual (Common Instructions)". The reference section is the section described in "Used Instructions" in the "● Example of use" table field.

6.1.6 Double precision integer type (DINT)→real number type (REAL) conversion DINT_TO_REAL DINT_TO_REAL_E

Double precision integer type (DINT) data is converted into real number type (REAL) data. ⟶ 1)

■ Function definition

REAL DINT_TO_REAL (DINT S1);
2) 3) 4) 5)

● Argument ⟶ 6)

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (BIN 32-bit data)

● Return value ⟶ 7)

Return Value	Description
REAL	Conversion result (real number data)

● Example of use ⟶ 8)

Argument Type	ST Program	Conversion Result	Used Instruction
DINT	r_data1 := DINT_TO_REAL(di_data1);	LD SM400 DFLT di_data1 r_data1	LD,DFLT

9) 10) 11)

1) Indicates the function of the function.
2) Indicates the data type of the function.
3) Indicates the function name.
4) Indicates the data type of the argument. (The STRING type is represented STRING (number of characters). It is represented STRING(6) when the number of characters is 6.)
5) Indicates the argument name.
6) Indicates the list (argument name, IN/OUT, description) of arguments used with the function.
7) Indicates the list (return value name, description) of return values used with the function.
8) Indicates the example of using the function. (Indicates the example that uses the actual device/label.)
9) This example is the one that uses a REAL type (real number type) label.
10) This example is the one that uses a DINT type (double word type) label.
11) Indicates the QCPU (Q mode)/L MELSEC common instruction corresponding to the function.

6 - 1

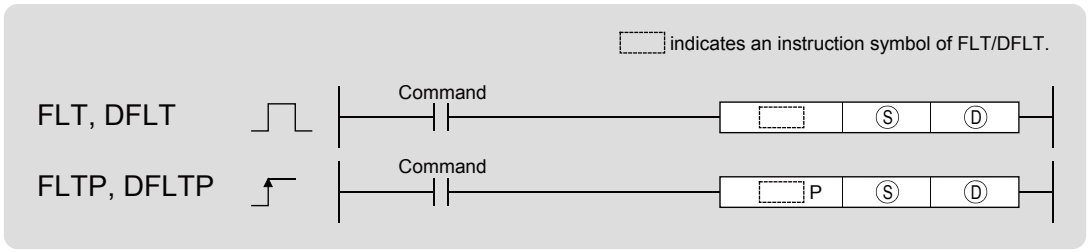
The following indicates the correspondences between the MELSEC instruction in the "MELSEC-Q/L Programming Manual (Common Instructions)" and the IEC function in this manual.

MELSEC-Q/L Programming Manual (Common Instructions) [MELSEC instruction]

6.3.3 Conversion from BIN 16 and 32-bit data to floating decimal point (Single precision) (FLT(P),DFLT(P)) →1)

Ver. Basic | High performance | Process | Redundant | Universal | LCPU →2)

Basic model QCPU: The upper five digits of the serial No. are "04122" or larger.



Ⓢ : Integer data to be converted to 32-bit floating decimal point data or head number of the devices where the integer data is stored (BIN 16/32 bits)

Ⓓ : Head number of the devices where the converted 32-bit floating decimal point data will be stored (real number)

Setting Data	Internal Devices		R, ZR	J□\□		U□\G□	Zn	Constants K, H	Other
	Bit	Word		Bit	Word				
Ⓢ	○	○		○	○		○		—
Ⓓ	—	○		—	○		○*1		—

→3)

*1:Available only in multiple Universal model QCPU and LCPU

[IEC function] in this manual

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
DINT	r_data1 := DINT_TO_REAL(di_data1);	LD	SM400	LD,DFLT ↓ 4)
		DFLT	di_data1	
			r_data1	

1) MELSEC instruction reference destination
2) Applicable CPU types
CPU types that can use the instructions are indicated.
3) Applicable devices
4) MELSEC common instructions to be referred to

6 - 2

6.1 Type Conversion Functions

6.1.1 Boolean type (BOOL)→double precision integer type (DINT) conversion BOOL_TO_DINT BOOL_TO_DINT_E

The specified Boolean type (BOOL) data is converted into double precision integer type (DINT) data.

■ Function definition

DINT BOOL_TO_DINT(BOOL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (bit data)

● Return value

Return Value	Description
DINT	Conversion result (BIN 32-bit data)

Remarks: The data to be converted (bit data) is stored into the least significant bit of the return value.

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
BOOL	di_data1 := BOOL_TO_DINT(b_data1);	LD b_data1 DMOV K1 di_data1 LDI b_data1 DMOV K0 di_data1	LD, DMOV, LDI

■ Function definition

BOOL BOOL_TO_DINT_E(BOOL EN, BOOL S1, DINT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (bit data)
D1	OUT	Conversion result (BIN 32-bit data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the Boolean type data in bData is  *)
(* converted into double precision integer type (DINT) data, and the result is  *)
(* stored into Result.  *)
    M0 := BOOL_TO _DINT_E ( X0, bData, Result ) ;
```

6.1.2 Boolean type (BOOL)→integer type (INT) conversion BOOL_TO_INT BOOL_TO_INT_E

Boolean type (BOOL) data is converted into integer type (INT) data.

■ Function definition

INT BOOL_TO_INT (BOOL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (bit data)

● Return value

Return Value	Description
INT	Conversion result (BIN 16-bit data)

Remarks: The data to be converted (bit data) is stored into the least significant bit of the return value.

● Example of use

Argument Type	ST Program	Conversion Result			Used Instruction
INT	D50 := BOOL_TO_INT(M100);	LD	M100		LD, MOV, LDI
		MOV	K1	D50	
		LDI	M100		
		MOV	K0	D50	

■ Function definition

BOOL BOOL_TO_INT_E(BOOL EN, BOOL S1, INT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (bit data)
D1	OUT	Conversion result (BIN 16-bit data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the Boolean type (BOOL) in bData is  *)
(* converted into the integer type (INT), and the result is stored into Result.  *)
    M0 := BOOL_TO _INT_E ( X0, bData, Result ) ;
```

6.1.3 Boolean type (BOOL)→character string type (STRING) conversion BOOL_TO_STR BOOL_TO_STR_E

Boolean type (BOOL) data is converted into character string type (STRING) data.

■ Function definition

STRING(2) BOOL_TO_STR (BOOL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (bit data)

● Return value

Return Value	Description
STRING (2)	Conversion result (character string data)

Remarks: When the data to be converted (bit data) is 0, the return value is "0".
When the data to be converted (bit data) is 1, the return value is "1".

● Example of use

Argument Type	ST Program	Conversion Result			Used Instruction
BOOL	s_ary1 :=	LD	b_data1		LD, MOV, LDI
	BOOL_TO_STR(b_data1);	MOV	K49	s_ary1	
		LDI	b_data1		
		MOV	K48	s_ary1	

■ Function definition

BOOL BOOL_TO_STR_E(BOOL EN, BOOL S1, STRING(2) D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (bit data)
D1	OUT	Conversion result (character string data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the Boolean type (BOOL) data in      *)
(* bData is converted into the character string type, and the result is stored into  *)
(* Result.                                                                    *)
    M0 := BOOL_TO _STR _E ( X0, bData, Result ) ;
```

6.1.4 Double precision integer type (DINT)→Boolean type (BOOL) conversion DINT_TO_BOOL DINT_TO_BOOL_E

Double precision integer type (DINT) data is converted into Boolean type (BOOL) data.

■ Function definition

BOOL DINT_TO_BOOL (DINT S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (BIN 32-bit data)

● Return value

Return Value	Description
BOOL	Conversion result (bit data)

Remarks: When the data to be converted (BIN 32-bit data) is 0, the return value is "0".
When the data to be converted (BIN 32-bit data) is other than 0, the return value is "1".

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
DINT	M100 := DINT_TO_BOOL(di_data1);	LDD<> OUT	di_data1 K0 M100	LDD<>, OUT

■ Function definition

BOOL DINT_TO_BOOL _E(BOOL EN, DINT S1, BOOL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 32-bit data)
D1	OUT	Conversion result (bit data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the double precision integer type     *)
(* (DINT) data in dData is converted into the Boolean type (BOOL), and the     *)
(* result is stored into Result.                                               *)
    M0 := DINT_TO _BOOL_E ( X0, dData, Result ) ;
```

6.1.5 Double precision integer type (DINT)→integer type (INT) conversion DINT_TO_INT DINT_TO_INT_E

Double precision integer type (DINT) data is converted into integer type (INT) data.

■ Function definition

INT DINT_TO_INT (DINT S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (BIN 32-bit data)

● Return value

Return Value	Description
INT	Conversion result (BIN 16-bit data)

Remarks: The lower 16 bits of the data to be converted (BIN 32-bit data) is stored in the return value.
The upper 16 bits are discarded.

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
DINT	i_data1 := DINT_TO_INT(di_data1);	LD MOV	SM400 di_data1 i_data1	LD,MOV

■ Function definition

BOOL DINT_TO_INT_E(BOOL EN, DINT S1, INT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 32-bit data)
D1	OUT	Conversion result (BIN 16-bit data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the double precision integer type   *)
(* (DINT) data in dData is converted into integer type (INT) data, and the result  *)
(* is stored into Result.                                                    *)
    M0 := DINT_TO _INT_E ( X0, dData, Result ) ;
```

6.1.6 Double precision integer type (DINT)→real number type (REAL) conversion DINT_TO_REAL DINT_TO_REAL_E

Double precision integer type (DINT) data is converted into real number type (REAL) data.

■ Function definition

REAL DINT_TO_REAL (DINT S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (BIN 32-bit data)

● Return value

Return Value	Description
REAL	Conversion result (real number data)

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
DINT	r_data1 := DINT_TO_REAL(di_data1);	LD DFLT	SM400 di_data1 r_data1	LD,DFLT

■ Function definition

BOOL DINT_TO_REAL_E(BOOL EN, DINT S1, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 32-bit data)
D1	OUT	Conversion result (real number data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the double precision integer type      *)
(* (DINT) data in dData is converted into real number type (REAL) data, and the *)
(* result is stored into Result.                                                 *)
    M0 := DINT_TO _REAL_E ( X0, dData, Result ) ;
```


6 - 8

6.1.7 Double precision integer type (DINT)→character string type (STRING) conversion DINT_TO_STR DINT_TO_STR_E

Double precision integer type (DINT) data is converted into character string type (STRING) data.

■ Function definition

STRING(12) DINT_TO_STR (DINT S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (BIN 32-bit data)

● Return value

Return Value	Description
STRING (12)	Conversion result (character string data)

Remarks: This function cannot be used with the Basic model QCPU.

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
DINT	s_ary1 := DINT_TO_STR(K65535);	LD SM400 DBINDA K65535 s_ary1	LD,DBINDA

■ Function definition

BOOL DINT_TO_STR _E(BOOL EN, DINT S1, STRING(12) D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 32-bit data)
D1	OUT	Conversion result (character string data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the double precision integer type   *)
(* (DINT) data in dData is converted into character string type data, and the *)
(* result is stored into Result.                                              *)
    M0 := DINT_TO_STR_E( X0, dData, Result ) ;
```

6.1.8 Integer type (INT)→Boolean type (BOOL) conversion INT_TO_BOOL INT_TO_BOOL_E

Integer type (INT) data is converted into Boolean type (BOOL) data.

■ Function definition

BOOL INT_TO_BOOL (INT S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (BIN 16-bit data)

● Return value

Return Value	Description
BOOL	Conversion result (bit data)

Remarks: When the data to be converted (BIN 16-bit data) is 0, the return value is "0".
When the data to be converted (BIN 16-bit data) is other than 0, the return value is "1".

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
INT	b_data1 := INT_TO_BOOL(i_data1);	LD<>	i_data K0	LD<>, OUT
		OUT	b_data1	

■ Function definition

BOOL INT_TO_BOOL _E(BOOL EN, INT S1, BOOL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 16-bit data)
D1	OUT	Conversion result (bit data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the integer type (INT) data in iData is  *)
(* converted into Boolean type (BOOL) data, and the result is stored into Result. *)
    M0 := INT_TO_BOOL_E( X0, iData, Result ) ;
```

6.1.9 Integer type (INT)→double precision integer type (DINT) conversion INT_TO_DINT INT_TO_DINT_E

Integer type (INT) data is converted into double precision integer type (DINT) data.

■ Function definition

DINT INT_TO_DINT (INT S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (BIN 16-bit data)

● Return value

Return Value	Description
DINT	Conversion result (BIN 32-bit data)

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
INT	di_data1 := INT_TO_DINT(D500);	LD DBL	SM400 D500 di_data1	LD,DBL

■ Function definition

BOOL INT_TO_DINT _E(BOOL EN, INT S1, DINT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 16-bit data)
D1	OUT	Conversion result (BIN 32-bit data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the integer type (INT) data in iData is  *)
(* converted into double precision integer type (DINT) data, and the result is     *)
(* stored into Result.                                                             *)
      M0 := INT_TO_DINT_E( X0, iData, Result ) ;
```

6.1.10 Integer type (INT)→real number type (REAL) conversion INT_TO_REAL INT_TO_REAL_E

Integer type (INT) data is converted into real number type (REAL) data.

■ Function definition

REAL INT_TO_REAL (INT S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (BIN 16-bit data)

● Return value

Return Value	Description
REAL	Conversion result (real number data)

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
INT	w_Real1:= INT_TO_REAL(D0);	LD FLT	SM400 D0 w_Real1	LD,FLT

■ Function definition

BOOL INT_TO_REAL_E(BOOL EN, INT S1, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 16-bit data)
D1	OUT	Conversion result (real number data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the integer type (INT) data in iData is  *)
(* converted into real number type (REAL) data, and the result is stored into      *)
(* Result.                                                                         *)
    M0 := INT_TO_REAL_E( X0, iData, Result ) ;
```

6.1.11 Integer type (INT)→character string type (STRING) conversion INT_TO_STR INT_TO_STR_E

Integer type (INT) data is converted into character string type (STRING) data.

■ Function definition

STRING(8) INT_TO_STR (INT S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (BIN 16-bit data)

● Return value

Return Value	Description
STRING (8)	Conversion result (character string data)

Remarks: This function cannot be used with the Basic model QCPU.

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
INT	w_Str1 := INT_TO_STR(D0);	LD SM400 BINDA D0 w_Str1	LD,BINDA

■ Function definition

BOOL INT_TO_STR _E(BOOL EN, INT S1, STRING(8) D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (BIN 16-bit data)
D1	OUT	Conversion result (character string data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the integer type (INT) data in iData is  *)
(* converted into character string type data, and the result is stored into Result.  *)
      M0 := INT_TO_STR_E( X0, iData, Result ) ;
```

6.1.12 Real number type (REAL)→double precision integer type (DINT) conversion REAL_TO_DINT REAL_TO_DINT_E

The specified real number type (REAL) data is converted into double precision integer type (DINT) data.

■ Function definition

DINT REAL_TO_DINT(REAL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (real number data)

● Return value

Return Value	Description
DINT	Conversion result (BIN 32-bit data)

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	w_DWord1:= REAL_TO_DINT(w_Real1);	LD DINT	SM400 w_Real1 w_DWord1	LD,DINT

■ Function definition

BOOL REAL_TO_DINT_E(BOOL EN, REAL S1, DINT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (real number data)
D1	OUT	Conversion result (BIN 32-bit data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the real number type (REAL) data in  *)
(* rData is converted into double precision integer type (DINT) data, and the  *)
(* result is stored into Result.                                               *)
    M0 := REAL_TO_DINT_E( X0, rData, Result ) ;
```


6 - 14

6.1.13 Real number type (REAL)→integer type (INT) conversion REAL_TO_INT REAL_TO_INT_E

Real number type (REAL) data is converted into integer type (INT) data.

■ Function definition

INT REAL_TO_INT (REAL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (real number data)

● Return value

Return Value	Description
INT	Conversion result (BIN 16-bit data)

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	w_Word1:= REAL_TO_INT(w_Real1);	LD INT	SM400 w_Real1 w_Word1	LD,INT

■ Function definition

BOOL REAL_TO_INT_E(BOOL EN, REAL S1, INT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (real number data)
D1	OUT	Conversion result (BIN 16-bit data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the real number type (REAL) data in   *)
(* rData is converted into integer type (INT) data, and the result is stored into   *)
(* Result.                                                                          *)
      M0 := REAL_TO_INT_E( X0, rData, Result ) ;
```

6.1.14 Real number type (REAL)→character string type (STRING) conversion REAL_TO_STR REAL_TO_STR_E

Real number type (REAL) data is converted into character string type data.

■ Function definition

STRING(14) REAL_TO_STR (REAL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (real number data)

● Return value

Return Value	Description
STRING (14)	Conversion result (character string data)

Note: The display format of the ESTR instruction is the Exponent format, the total number of digits is 13, and the number of fraction part digits is 5.

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	w_Str1:= REAL_TO_STR(w_Real1);	LD	SM400	LD,MOV,ESTR
		MOV	K1 D10237	
		MOV	K13 D10238	
		MOV	K5 D10239	
		ESTR	w_Real1 D10237	
			w_Str1	

■ Function definition

BOOL REAL_TO_STR _E(BOOL EN, REAL S1, STRING(14) D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (real number data)
D1	OUT	Conversion result (character string data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the real number type (REAL) data in  *)
(* rData is converted into character string type data, and the result is stored into  *)
(* Result.                                                                    *)
    M0 := REAL_TO_STR_E( X0, rData, Result ) ;
```

6.1.15 Character string type (STRING)→Boolean type (BOOL) conversion STR_TO_BOOL STR_TO_BOOL_E

Character string type (STRING) data is converted into Boolean type (BOOL) data.

■ Function definition

BOOL STR_TO_BOOL (STRING(2) S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (character string data)

● Return value

Return Value	Description
BOOL	Conversion result (bit data)

Remarks: When the data to be converted (character string data) is 0, the return value is "0". When the data to be converted (character string data) is other than 0, the return value is "1".

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
STRING	w_Bit1:= STR_TO_BOOL(w_Str1);	LD<> OUT	w_Str1 K48 w_Bit1	LD<>,OUT

■ Function definition

BOOL STR_TO_BOOL_E(BOOL EN, STRING(2) S1, BOOL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (character string data)
D1	OUT	Conversion result (bit data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string type data in      *)
(* sData is converted into Boolean type data, and the result is stored into Result. *)
    M0 := STR_TO_BOOL _E( X0, sData, Result ) ;
```

6.1.16 Character string type (STRING)→double precision integer type (DINT) conversion STR_TO_DINT STR_TO_DINT_E

Character string type (STRING) data is converted into double precision integer type (DINT) data.

■ Function definition

DINT STR_TO_DINT (STRING(12) S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (character string data)

● Return value

Return Value	Description
DINT	Conversion result (BIN 32-bit data)

Remarks: This function cannot be used with the Basic model QCPU.

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
STRING	w_DWord1:= STR_TO_DINT("123");	LD SM400 DDABIN "123" w_DWord1	LD,DDABIN

■ Function definition

BOOL STR_TO_DINT _E(BOOL EN, STRING(12) S1, DINT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (character string data)
D1	OUT	Conversion result (BIN 32-bit data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string type data in      *)
(* sData is converted into double precision integer type (DINT) data, and the   *)
(* result is stored into Result.                                                *)
    M0 := STR_TO_DINT_E( X0, sData, Result ) ;
```

6.1.17 Character string type (STRING)→integer type (INT) conversion STR_TO_INT STR_TO_INT_E

Character string type (STRING) data is converted into integer type (INT) data.

■ Function definition

INT STR_TO_INT (STRING(6) S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (character string data)

● Return value

Return Value	Description
INT	Conversion result (BIN 16-bit data)

Remarks: This function cannot be used with the Basic model QCPU.

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
STRING	w_Word1:= STR_TO_INT(w_Str1);	LD DABIN	SM400 w_Str1 w_Word1	LD,DABIN

■ Function definition

BOOL STR_TO_INT _E(BOOL EN, STRING(6) S1, INT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (character string data)
D1	OUT	Conversion result (BIN 16-bit data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string type data in      *)
(* sData is converted into integer type (INT) data, and the result is stored into *)
(* Result.                                                                       *)
      M0 := STR_TO_INT_E( X0, sData, Result ) ;
```

6.1.18 Character string type (STRING)→real number type (REAL) conversion STR_TO_REAL STR_TO_REAL_E

Character string type (STRING) data is converted into real number type (REAL) data.

■ Function definition

REAL STR_TO_REAL (STRING(24) S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be converted (character string data)

● Return value

Return Value	Description
REAL	Conversion result (real number data)

Remarks:

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
STRING	w_Real1:= STR_TO_REAL(w_Str1);	LD SM400 EVAL w_Str1 w_Real1	LD,EVAL

■ Function definition

BOOL STR_TO_REAL _E(BOOL EN, STRING(24) S1, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be converted (character string data)
D1	OUT	Conversion result (real number data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string type data in      *)
(* sData is converted into real number type (REAL) data, and the result is stored *)
(* into Result.                                                                  *)
      M0 := STR_TO_REAL_E( X0, sData, Result ) ;
```

6.2 Numerical Functions (General Functions)

6.2.1 Absolute value ABS
ABS_E

The absolute value of the specified data is operated.

■ Function definition

ANY_NUM ABS (ANY_NUM S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data whose absolute value will be found

● Return value

Return Value	Description
ANY_NUM	Absolute value operation result

● Example of use

Argument Type	ST Program	Conversion Result			Used Instruction
REAL	r_data1 := ABS(r_data2);	LD	SM400		LD,EMOV, LDE<, E*
		EMOV	r_data2		
			r_data1		
		LDE<	r_data2	E0	
		E*	E-1		
			r_data2		
			r_data1		
INT	D0 := ABS(D1);	LD	SM400		LD,MOV, LD<, NEG
		MOV	D1	D0	
		LD<	D1	K0	
		NEG	D0		
DINT	di_data1 := ABS(di_data2);	LD	SM400		LD,DMOV, LDD<, DCML D+
		DMOV	di_data2		
			di_data1		
		LDD<	di_data2	K0	
		DCML	di_data2		
			di_data1		
		D+	K1		
			di_data1		

■ Function definition

BOOL ABS_E(BOOL EN, ANY_NUM S1, ANY_NUM D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data whose absolute value will be found
D1	OUT	Absolute value operation result

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the absolute value of the data stored  *)
(* in iData is found, and the result is stored into Result.                      *)
      M0 := ABS_E( X0, iData, Result ) ;
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 21

6.2.2 Square root SQRT
SQRT_E

The square root of the specified data is operated.

■ Function definition

REAL SQRT (REAL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data whose square root will be found

● Return value

Return Value	Description
REAL	Square root operation result (real number data)

Remarks:

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	r_data1 := SQRT(r_data2);	LD	SM400	LD,SQR
		SQR	r_data2	
			r_data1	

■ Function definition

BOOL SQRT_E(BOOL EN, REAL S1, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data whose square root will be found (real number data)
D1	OUT	Square root operation result (real number data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the square root of the data stored in   *)
(* rData is found, and the result is stored into Result.                          *)
    M0 := SQRT_E( X0, rData, Result ) ;
```


6 - 22

6.3 Numeric Functions (Logarithm Functions)

6.3.1 Natural logarithm LN
LN_E

The natural logarithm of the specified data is operated.

■ Function definition

REAL LN(REAL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data whose natural logarithm will be found (real number data)

● Return value

Return Value	Description
REAL	Natural logarithm operation result (real number data)

Remarks:

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	r_data1 := LN(1.23456);	LD	SM400	LD,LOG
		LOG	E1.23456	
			r_data1	

■ Function definition

BOOL LN_E(BOOL EN, REAL S1, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data whose natural logarithm will be found (real number data)
D1	OUT	Natural logarithm operation result (real number data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the natural logarithm of the data    *)
(* stored in rData is found, and the result is stored into Result.             *)
    M0 := LN_E( X0, rData, Result ) ;
```


6 - 23

6.3.2 Natural exponent EXP
EXP_E

The natural exponent of the specified data is operated.

■ Function definition

REAL EXP(REAL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data whose natural exponent will be found (real number data)

● Return value

Return Value	Description
REAL	Natural exponent operation result (real number data)

Remarks:

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	r_data1 := EXP(r_data2);	LD	SM400	LD,EXP
		EXP	r_data2	
			r_data1	

■ Function definition

BOOL EXP_E(BOOL EN, REAL S1, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data whose natural exponent will be found (real number data)
D1	OUT	Natural exponent operation result (real number data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the natural exponent of the data      *)
(* stored in rData is found, and the result is stored into Result.              *)
    M0 := EXP_E( X0, rData, Result ) ;
```


6 - 24

6.4 Numerical Functions (Trigonometric Functions)

6.4.1 Floating-point SIN operation SIN
SIN_E

The SIN (sine) value of the specified angle is operated.

■ Function definition

REAL SIN(REAL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Angle data to be SIN (sine) operated (real number data)

Remarks: Set the specified angle in radian unit (angle × π / 180).

● Return value

Return Value	Description
REAL	SIN operation result (real number data)

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	r_data1 := SIN(1.23456);	LD	SM400	LD,SIN
		SIN	E1.23456	
			r_data1	

■ Function definition

BOOL SIN_E(BOOL EN, REAL S1, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Angle data to be SIN (sine) operated (real number data) Remarks: Set the specified angle in radian (unit angle × π / 180).
D1	OUT	SIN operation result (real number data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the SIN value of the angle data      *)
(* stored in rData is calculated, and the result is stored into Result.        *)
    M0 := SIN_E( X0, rData, Result ) ;
```

6.4.2 Floating-point COS operation COS COS_E

The COS (cosine) value of the specified angle is operated.

■ Function definition

REAL COS(REAL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Angle data to be COS (cosine) operated (real number data)

Remarks: Set the specified angle in radian unit (angle × π / 180).

● Return value

Return Value	Description
REAL	COS operation result (real number data)

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	w_Real1 := COS(w_Real2);	LD	SM400	LD,COS
		COS	w_Real2	
			w_Real1	

■ Function definition

BOOL COS_E(BOOL EN, REAL S1, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Angle data to be COS (cosine) operated (real number data) Remarks: Set the specified angle in radian unit (angle × π / 180).
D1	OUT	COS operation result (real number data)

Remarks: Set the specified angle in radian unit (angle × π / 180).

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the COS value of the angle data      *)
(* stored in rData is calculated, and the result is stored into Result.        *)
    M0 := COS_E( X0, rData, Result ) ;
```


6 - 26

6.4.3 Floating-point TAN operation TAN
TAN_E

The TAN (tangent) value of the specified angle is operated.

■ Function definition

REAL TAN(REAL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Angle data to be TAN (tangent) operated (real number data)

Remarks: Set the specified angle in radian unit (angle × π / 180).

● Return value

Return Value	Description
REAL	TAN operation result (real number data)

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	w_Real1 := TAN(w_Real2);	LD	SM400	LD,TAN
		TAN	w_Real2	
			w_Real1	

■ Function definition

BOOL TAN_E(BOOL EN, REAL S1, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Angle data to be TAN (tangent) operated (real number data) Remarks: Set the specified angle in radian unit (angle × π / 180).
D1	OUT	TAN operation result (real number data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the TAN value of the angle data      *)
(* stored in rData is calculated, and the result is stored into Result.        *)
    M0 := TAN_E( X0, rData, Result ) ;
```

6.4.4 Floating-point SIN^{-1} operation ASIN
ASIN_E

The SIN^{-1} (arcsine) of the specified SIN value is operated.

■ Function definition

REAL ASIN(REAL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	SIN value to be SIN^{-1} (arcsine) operated (-1.0 to 1.0) (real number data)

● Return value

Return Value	Description
REAL	SIN^{-1} operation result (real number data)

Remarks: This function cannot be used with the Basic model QCPU.
The operation result is the angle data in radian unit.

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	w_Real1 := ASIN(w_Real2);	LD	SM400	LD,ASIN
		ASIN	w_Real2	
			w_Real1	

■ Function definition

BOOL ASIN_E(BOOL EN, REAL S1, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	SIN value to be SIN^{-1} (arcsine) operated (-1.0 to 1.0) (real number data)
D1	OUT	SIN^{-1} operation result (real number data)

Remarks: The operation result is the angle data in radian unit.

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the angle is operated from the SIN    *)
(* value stored in rData, and the result is stored into Result.                 *)
    M0 := ASIN_E( X0, rData, Result ) ;
```

6.4.5 Floating-point COS^{-1} operation ACOS ACOS_E

The COS^{-1} (arccosine) of the specified COS value is operated.

■ Function definition

REAL ACOS(REAL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	COS value to be COS^{-1} (arccosine) operated (-1.0 to 1.0) (real number data)

● Return value

Return Value	Description
REAL	COS^{-1} operation result (real number data)

Remarks: This function cannot be used with the Basic model QCPU.
The operation result is the angle data in radian unit.

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	w_Real1 := ACOS(w_Real2);	LD	SM400	LD,ACOS
		ACOS	w_Real2	
			w_Real1	

■ Function definition

BOOL ACOS_E(BOOL EN, REAL S1, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	COS value to be COS^{-1} (arccosine) operated (-1.0 to 1.0) (real number data)
D1	OUT	COS^{-1} operation result (real number data)

Remarks: The operation result is the angle data in radian unit.

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the angle is operated from the COS   *)
(* value stored in rData, and the result is stored into Result.               *)
    M0 := ACOS_E( X0, rData, Result ) ;
```

6.4.6 Floating-point TAN^{-1} operation ATAN ATAN_E

The TAN^{-1} (arctangent) of the specified TAN value is operated.

■ Function definition

REAL ATAN(REAL S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	TAN value to be TAN^{-1} (arctangent) operated (real number data)

● Return value

Return Value	Description
REAL	TAN^{-1} operation result (real number data)

Remarks: This function cannot be used with the Basic model QCPU.
The operation result is the angle data in radian unit.

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	w_Real1 := ATAN(w_Real2);	LD	SM400	LD,ATAN
		ATAN	w_Real2	
			w_Real1	

■ Function definition

BOOL ATAN_E(BOOL EN, REAL S1, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	TAN value to be TAN^{-1} (arctangent) operated (real number data)
D1	OUT	TAN^{-1} operation result (real number data)

Remarks: The operation result is the angle data in radian unit.

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the angle is operated from the TAN   *)
(* value stored in rData, and the result is stored into Result.               *)
    M0 := ATAN_E( X0, rData, Result ) ;
```

6.5 Arithmetic Operation Functions

6.5.1 Addition ADD_E

The specified multiple data are added.

■ Function definition

BOOL ADD_E(BOOL EN, ANY_NUM S1, ANY_NUM S2,....,ANY_NUM Sn, ANY_NUM D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Data to be added
D1	OUT	Addition operation result

● Return value

Return Value	Description
BOOL	Execution condition (bit data)

● Example of use

Argument Type	ST Program	Conversion Result			Used Instruction
REAL	b_result := ADD_E(b_select, r_data1, r_data2, r_data3);	LD E+ LD OUT	b_select r_data1 r_data2 r_data3 b_select b_result		LD, E+, OUT
INT	b_result := ADD_E(b_select, D10, D20, D30, D40);	LD + + LD OUT	b_select D10 D40 D30 b_select b_result	 D20 D40 	LD, +, OUT
DINT	b_result := ADD_E(b_select, di_data1, di_data2, di_data3);	LD D+ LD OUT	b_select di_data1 di_data2 di_data3 b_select b_result		LD,D+,OUT

For the usable data type, refer to "3.2.2 About ANY type".

6 - 31

6.5.2 Multiplication MUL_E

The specified multiple data are multiplied.

■ Function definition

BOOL MUL_E(BOOL EN, ANY_NUM S1, ANY_NUM S2,....,ANY_NUM Sn, ANY_NUM D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Data to be multiplied
D1	OUT	Multiplication operation result

● Return value

Return Value	Description
BOOL	Execution condition (bit data)

● Example of use

Argument Type	ST Program	Conversion Result			Used Instruction
REAL	b_result := MUL_E(b_select, r_data1, r_data2, r_data3);	LD E* LD OUT	b_select r_data1 r_data2 r_data3 b_select b_result		LD, E*, OUT
INT	b_result := MUL_E(b_select, D10, D20, D30, D40);	LD * * MOV LD OUT	b_select D10 D10238 D10238 D10236 D10236 b_select b_result	 D20 D30 D40 	LD, *, MOV, OUT
DINT	b_result := MUL_E(b_select, di_data1, di_data2, di_data3);	LD D* DMOV LD OUT	b_select di_data1 di_data2 D10236 D10236 di_data3 b_select b_result		LD, D*, DMOV, OUT

For the usable data type, refer to "3.2.2 About ANY type".

6 - 32

6.5.3 Subtraction SUB_E

Subtraction is performed between the specified data.

■ Function definition

BOOL SUB_E(BOOL EN, ANY_NUM S1, ANY_NUM S2, ANY_NUM D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Minuend data
S2	IN	Subtrahend data
D1	OUT	Subtraction operation result

● Return value

Return Value	Description
BOOL	Execution condition (bit data)

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	b_result := SUB_E(b_select, r_data1, r_data2, r_data3);	LD E- LD OUT	b_select r_data1 r_data2 r_data3 b_select b_result	LD, E-, OUT
INT	b_result := SUB_E(b_select, 32767, D100, i_data1);	LD - LD OUT	b_select K32767 D100 i_data1 b_select b_result	LD, -, OUT
DINT	b_result := SUB_E(b_select, di_data1, di_data2, di_data3);	LD D- LD OUT	b_select di_data1 di_data2 di_data3 b_select b_result	LD, D-, OUT

For the usable data type, refer to "3.2.2 About ANY type".

6 - 33

6.5.4 Division DIV_E

Division is performed between the specified data.

■ Function definition

BOOL DIV_E(BOOL EN, ANY_NUM S1, ANY_NUM S2, ANY_NUM D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Dividend data
S2	IN	Divisor data
D1	OUT	Division operation result

● Return value

Return Value	Description
BOOL	Execution condition (bit data)

● Example of use

Argument Type	ST Program	Conversion Result			Used Instruction
REAL	b_result := DIV_E(b_select, r_data1, r_data2, r_data3);	LD E/ LD OUT	b_select r_data1 r_data2 r_data3 b_select b_result		LD, E/, OUT
INT	b_result := DIV_E(b_select, D10, D20, D30);	LD / MOV LD OUT	b_select D10 D10238 D10238 b_select b_result	 D20 D30 	LD, /, MOV, OUT
DINT	b_result := DIV_E(b_select, di_data1, di_data2, di_data3);	LD D/ DMOV LD OUT	b_select di_data1 di_data2 D10236 D10236 di_data3 b_select b_result		LD, D/, DMOV, OUT

For the usable data type, refer to "3.2.2 About ANY type".

6 - 34

6.5.5 Modulus operation MOD MOD_E

Division is performed between the specified data, and its remainder is operated.

■ Function definition

BOOL MOD_E(BOOL EN, ANY_INT S1, ANY_INT S2, ANY_INT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Dividend data
S2	IN	Divisor data
D1	OUT	Modulus operation result

● Return value

Return Value	Description
BOOL	Execution condition (bit data)

● Example of use

Argument Type	ST Program	Conversion Result			Used Instruction
INT	B100 := MOD_E(M1, D10, D20, D30);	LD / MOV LD OUT	M1 D10 D10238 D10239 M1 B100	 D20 D30 	LD, /, MOV, OUT
DINT	b_result := MOD_E(b_select, di_data1, di_data2, di_data3);	LD D/ DMOV LD OUT	b_select di_data1 di_data2 D10236 D10238 di_data3 b_select b_result		LD, D/, DMOV, OUT

∗ MOD can be used as an operator only.

For the usable data type, refer to "3.2.2 About ANY type".

6 - 35

6.5.6 Natural exponential EXPT EXPT_E

Natural exponential is operated from the specified data used as a base and data used as an exponent.

■ Function definition

REAL EXPT (REAL S1, ANY_NUM S2);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data used as base
S2	IN	Data used as exponent

● Return value

Return Value	Description
REAL	Operation result (real number data)

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	r_data1 := EXPT(r_data2, r_data3);	LD	SM400	LD, LOG, E*, EXP
		LOG	r_data2	
			r_data1	
		E*	r_data1	
			r_data3	
			r_data1	
		EXP	r_data1	
			r_data1	
INT	r_data1 := EXPT(1.123, k32767);	LD	SM400	LD, LOG, FLT, E*, EXP
		LOG	E1.123	
			r_data1	
		FLT	K32767	
			D10238	
		E*	r_data1	
			D10238	
			r_data1	
		EXP	r_data1	
			r_data1	
DINT	r_data1 := EXPT(r_data2, di_data1);	LD	SM400	LD,LOG, DFLT, E*, EXP
		LOG	r_data2	
			r_data1	
		DFLT	di_data1	
			D10238	
		E*	r_data1	
			D10238	
			r_data1	
		EXP	r_data1	
			r_data1	

6 - 36

■ Function definition

BOOL EXPT_E(BOOL EN, REAL S1, ANY_NUM S2, REAL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data used as base
S2	IN	Data used as exponent
D1	OUT	Operation result

Remarks: The operation result is the angle data in radian unit.

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data stored in rData is natural   *)
(* exponential-operated with the data stored in iData, and the result is stored *)
(* into Result.                                                                 *)
    M0 := EXPT_E( X0, rData, iData, Result ) ;
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 37

6.5.7 Assignment MOVE MOVE_E

The specified data is assigned to the specified storage destination.

■ Function definition

ANY MOVE (ANY S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be assigned

● Return value

Return Value	Description
ANY	Assignment result data

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	W_Real1:= MOVE(W_Real2);	LD EMOV	SM400 w_Real2 w_Real1	LD,EMOV
INT	D1 :=MOVE(D0);	LD MOV	SM400 D0 D1	LD,MOV
DINT	w_DWord1:= MOVE(2147483647);	LD DMOV	SM400 K2147483647 w_DWord1	LD,DMOV
BOOL	w_Bit1:= MOVE(w_Bit2);	LD MPS AND SET MRD ANI RST MPP OUT	SM400 w_Bit2 w_Bit1 w_Bit2 w_Bit1 M8191	LD,MPS,AND,SET,MRD, ANI,RST,MPP,OUT
STRING	w_Str1 := MOVE("ABCDEFG");	LD $MOV	SM400 "ABCDEFG" w_Str1	LD,$MOV

■ Function definition

BOOL MOVE_E(BOOL EN, ANY S1, ANY D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be assigned
D1	OUT	Assignment result data

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data stored in iData is stored into  *)
(* Result.                                                                         *)
      M0 := MOVE_E( X0, iData, Result ) ;
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 38

6.6 Bit Shift Functions

6.6.1 Bit left shift SHL
SHL_E

The specified data is shifted n bits to the left.

■ Function definition ANY_BIT SHL (ANY_BIT S1, ANY_BIT n);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be shifted
n	IN	Number of bits to be shifted Remarks: Only a constant can be specified as the number of bits to be shifted.

● Return value

Return Value	Description
ANY_BIT	Shifted data Remarks: n bits of data from the least significant bit are 0.

Remarks:

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
INT	D0 := SHL(D1,1);	LD MOV SFL	SM400 D1 D0 D0 K1	LD,MOV,SFL

■ Function definition BOOL SHL_E(BOOL EN, ANY_BIT S1, ANY_BIT n, ANY_BIT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be shifted
n	IN	Number of bits to be shifted Remarks: Only a constant can be specified as the number of bits to be shifted.
D1	OUT	Shifted data Remarks: n bits of data from the least significant bit are 0.

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data stored in D0 is shifted 2 bits  *)
(* to the left, and the result is stored into Result.                              *)
    M0:=SHL_E( X0, D0, 2, D100 );
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 39

6.6.2 Bit right shift SHR
SHR_E

The specified data is shifted n bits to the right.

■ Function definition

ANY_BIT SHR (ANY_BIT S1, ANY_BIT n);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be shifted
n	IN	Number of bits to be shifted Remarks: Only a constant can be specified as the number of bits to be shifted.

● Return value

Return Value	Description
ANY_BIT	Shifted data Remarks: n bits of data from the most significant bit are 0.

Remarks:

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
INT	D0 := SHR(D1,1);	LD	SM400	LD,MOV,SFR
		MOV	D1 D0	
		SFR	D0 K1	

■ Function definition

BOOL SHR_E(BOOL EN, ANY_BIT S1, ANY_BIT n, ANY_BIT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be shifted
n	IN	Number of bits to be shifted Remarks: Only a constant can be specified as the number of bits to be shifted.
D1	OUT	Shifted data Remarks: n bits of data from the most significant bit are 0.

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data stored in D0 is shifted 2 bits  *)
(* to the right, and the result is stored into Result.                            *)
    M0:=SHR_E(X0, D0, 2, D100);
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 40

6.6.3 Right rotation ROR
ROR_E

Data is rotated n bits to the right in a circle.

■ Function definition

ANY_BIT ROR (ANY_BIT S1, ANY_BIT n);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be rotated
n	IN	Number of bits to be rotated Remarks: Only a constant can be specified as the number of bits to be rotated.

● Return value

Return Value	Description
ANY_BIT	Rotation result data

Remarks:

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
INT	D0 := ROR(D1,1);	LD SM400 MOV D1 D0 ROR D0 K1	LD,MOV,ROR

■ Function definition

BOOL ROR_E(BOOL EN, ANY_BIT S1, ANY_BIT n, ANY_BIT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be rotated
n	IN	Number of bits to be rotated Remarks: Only a constant can be specified as the number of bits to be rotated.
D1	OUT	Rotation result data

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data stored in D0 is rotated 1 bit   *)
(* to the right, and the result is stored into D100.                               *)
    M0:=ROR_E( X0, D0, 1, D100 );
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 41

6.6.4 Left rotation ROL
ROL_E

Data is rotated n bits to the left in a circle.

■ Function definition

ANY_BIT ROL (ANY_BIT S1, ANY_BIT n);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be rotated
n	IN	Number of bits to be rotated Remarks: Only a constant can be specified as the number of bits to be rotated.

● Return value

Return Value	Description
ANY_BIT	Rotation result data

Remarks:

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
INT	D0 := ROL(D1,1);	LD SM400 MOV D1 D0 ROL D0 K1	LD,MOV,ROL

■ Function definition

BOOL ROL_E(BOOL EN, ANY_BIT S1, ANY_BIT n, ANY_BIT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be rotated
n	IN	Number of bits to be rotated Remarks: Only a constant can be specified as the number of bits to be rotated.
D1	OUT	Rotation result data

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data stored in D0 is rotated 1 bit   *)
(* to the left, and the result is stored into D100.                                *)
      M0:=ROL_E( X0, D0, 1, D100 );
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 42

6.7 Bit Type Boolean Functions

6.7.1 Logical product AND_E

The logical product of the specified multiple data is operated.

■ Function definition

BOOL AND_E(BOOL EN, ANY_BIT S1, ANY_BIT S2,…., ANY_BIT Sn, ANY_BIT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Data to be ANDed
D1	OUT	AND operation result

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
BOOL	b_result := AND_E(b_select, b_data1, b_data2, b_data3, b_data4);	LD b_data1 AND b_data2 AND b_data3 OUT M8191 LD b_select AND M8191 SET b_data4 LD b_select ANI M8191 RST b_data4 LD b_select OUT b_result	LD,AND,OUT,SET, ANI,RST
Word device	b_result := AND_E(b_select, d0, d1, d2, d3);	LD b_select WAND D0 D1 D10239 WAND D10239 D2 D3 LD b_select OUT b_result	LD, WAND, OUT

For the usable data type, refer to "3.2.2 About ANY type".

6 - 43

6.7.2 Logical sum OR_E

The logical sum of the specified multiple data is operated.

■ Function definition

BOOL OR_E(BOOL EN, ANY_BIT S1, ANY_BIT S2,…., ANY_BIT Sn, ANY_BIT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Data to be ORed
D1	OUT	OR operation result

● Return value

Return Value	Description
BOOL	Execution condition

Remarks:

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
BOOL	b_result := OR_E(TRUE, b_data1, b_data2, b_data3);	LD	b_data1	LD,OR,OUT,AND,SET, ANI,RST
		OR	b_data2	
		OUT	M8191	
		LD	SM400	
		AND	M8191	
		SET	b_data3	
		LD	SM400	
		ANI	M8191	
		RST	b_data3	
		LD	SM400	
		OUT	b_result	
Word device	B1 := OR_E(TRUE, D0, D1, D2);	LD	SM400	LD, WOR, OUT
		WOR	D0 D1 D2	
		LD	SM400	
		OUT	B1	

For the usable data type, refer to "3.2.2 About ANY type".

6 - 44

6.7.3 Exclusive logical sum XOR_E

The exclusive logical sum of the specified multiple data is operated.

■ Function definition

BOOL XOR_E(BOOL EN, ANY_BIT S1, ANY_BIT S2,…., ANY_BIT Sn, ANY_BIT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Data to be EXCLUSIVE ORed
D1	OUT	EXCLUSIVE OR operation result

● Return value

Return Value	Description
BOOL	Execution condition

Remarks:

● Example of use

Argument Type	ST Program	Conversion Result			Used Instruction
BOOL	b_result := XOR_E(b_select, b_data1, b_data2, b_data3);	LD	b_data1		LD,ANI,LDI,AND,ORB, OUT,SET,RST
		ANI	b_data2		
		LDI	b_data1		
		AND	b_data2		
		ORB			
		OUT	M8191		
		LD	b_select		
		AND	M8191		
		SET	b_data3		
		LD	b_select		
		ANI	M8191		
		RST	b_data3		
		LD	b_select		
		OUT	b_result		
Word device	b_result := XOR_E(TRUE, d0z2, d1z3, d2z4);	LD	SM400		LD,WXOR, OUT
		WXOR	D0Z2	D1Z3	
			D2Z4		
		LD	SM400		
		OUT	b_result		

For the usable data type, refer to "3.2.2 About ANY type".

6 - 45

6.7.4 Logical NOT NOT
NOT_E

The logical NOT of the specified data is operated.

■ Function definition

ANY_BIT NOT(ANY_BIT S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be logical NOT operated

● Return value

Return Value	Description
ANY_BIT	Logical NOT operation result

Remarks:

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
BOOL	b_result := NOT(b_data1);	LDI OUT	b_data1 b_result	LDI, OUT
Word device	d0z2 := NOT(d1z3);	LD CML	SM400 D1Z3 D0Z2	LD, CML

■ Function definition

BOOL NOT_E(BOOL EN, ANY_BIT S1, ANY_BIT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be logical NOT operated
D1	OUT	Logical NOT operation result

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the logical NOT of the data stored in  *)
(* D0 is found, and the result is stored into D100.                              *)
    M0:=NOT_E( X0, D0, D100 );
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 46

6.8 Selection Functions

6.8.1 Binary selection SEL
SEL_E

One data is selected from among the specified two data according to the selection condition.

■ Function definition

ANY SEL(BOOL S1, ANY S2, ANY S3);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Selection condition
S2	IN	Data to be selected when S1 is FALSE
S3	IN	Data to be selected when S1 is TRUE

● Return value

Return Value	Description
ANY	Selection result When S1 is FALSE ..... Return value = S2 When S1 is TRUE ..... Return value = S3

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
REAL	r_data1 := SEL(b_select, r_data2, r_data3);	LDI b_select EMOV r_data2 r_data1 LD b_select EMOV r_data3 r_data1	LDI, EMOV, LD,
INT	D1 := SEL(X1, D2, D3);	LDI X1 MOV D2 D1 LD X1 MOV D3 D1	LDI, MOV, LD
DINT	K8X100 := SEL(X1, K8X10, K2147483647);	LDI X1 DMOV K8X10 K8X100 LD X1 DMOV K2147483647 K8X100	LDI, DMOV, LD
BOOL	b_result := SEL(b_select, b_data1, b_data2);	LDI b_select MPS AND b_data1 SET b_result MPP ANI b_data1 RST b_result LD b_select MPS AND b_data2 SET b_result MPP ANI b_data2 RST b_result	LDI, MPS,AND, SET, MPP, ANI, RST,LD
STRING	s_result := SEL(b_select, s_ary1, s_ary2);	LDI b_select $MOV s_ary1 s_result LD b_select $MOV s_ary2 s_result	LDI, $MOV,LD

■ Function definition

BOOL SEL_E(BOOL EN, BOOL S1, ANY S2, ANY S3, ANY D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Selection condition
S2	IN	Data to be selected when S1 is FALSE
S3	IN	Data to be selected when S1 is TRUE
D1	OUT	Selection result When S1 is FALSE ..... Return value = S2 When S1 is TRUE ..... Return value = S3

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data stored in iData1 is stored     *)
(* into Result if the bit data in bData is FALSE, or the data stored in iData2 is *)
(* stored into Result if the bit data in bData is TRUE.                           *)
    M0 := SEL_E( X0, bData, iData1, iData2, Result ) ;
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 48

6.8.2 Maximum value MAX
MAX_E

The specified data are searched for the maximum value.

■ Function definition

ANY_SIMPLE MAX(ANY_SIMPLE S1, ANY_SIMPLE S2,…., ANY_SIMPLE Sn);

● Argument

Argument Name	IN/OUT	Description
S1 to Sn	IN	Search target data

● Return value

Return Value	Description
ANY_SIMPLE	Search result

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	w_Real4 := MAX(w_Real1,w_Real2,w_Real3);	LD EMOV LDE< EMOV LDE< EMOV 	SM400 w_Real1 w_Real4 w_Real4 w_Real2 w_Real2 w_Real4 w_Real4 w_Real3 w_Real3 w_Real4	LD,EMOV,LDE<
INT	D0 := MAX(D1,D2,D3);	LD MOV LD< MOV LD< MOV	SM400 D1 D0 D0 D2 D2 D0 D0 D3 D3 D0	LD,MOV,LD<
DINT	w_DWord4 := MAX(-2147483648,0,2147483647);	LD DMOV	SM400 K2147483647 w_DWord4	LD,DMOV
BOOL	w_Bit4 := MAX(w_Bit1,w_Bit2,w_Bit3);	LD OR OR OUT	w_Bit1 w_Bit2 w_Bit3 w_Bit4	LD,OR,OUT
STRING	w_Str4 := MAX("ABC","DEF","GHI");	LD $MOV LD$< $MOV LD$< $MOV	SM400 "ABC" w_Str4 w_Str4 "DEF" "DEF" w_Str4 w_Str4 "GHI" "GHI" w_Str4	LD,MOV,LD<

6 - 49

■ Function definition

BOOL MAX_E(BOOL EN, ANY_SIMPLE S1, ANY_SIMPLE S2,…., ANY_SIMPLE Sn, ANY_SIMPLE D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Search target data
D1	OUT	Search result

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the data stored in iData1, iData2 and  *)
(* iData3 are searched for the maximum value, and the result is stored into      *)
(* Result.                                                                       *)
      M0 := MAX_E( X0, iData1, iData2, iData3, Result ) ;
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 50

6.8.3 Minimum value MIN
MIN_E

The specified data are searched for the minimum value.

■ Function definition

ANY_SIMPLE MIN(ANY_SIMPLE S1, ANY_SIMPLE S2,…., ANY_SIMPLE Sn);

● Argument

Argument Name	IN/OUT	Description
S1 to Sn	IN	Search target data

● Return value

Return Value	Description
ANY_SIMPLE	Search result

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
REAL	Real4:= MIN(Real1,Real2,Real3);	LD EMOV LDE> EMOV LDE> EMOV	SM400 Real1 Real4 Real4 Real2 Real2 Real4 Real4 Real3 Real3 Real4	LD,EMOV,LDE>
INT	Int4:= MIN(Int1,Int2,Int3);	LD MOV LD> MOV LD> MOV	SM400 Int1 Int4 Int4 Int2 Int2 Int4 Int4 Int3 Int3 Int4	LD,MOV,LD>
DINT	Dint4:= MIN(Dint1,Dint2,Dint3);	LD DMOV LDD> DMOV LDD> DMOV	SM400 Dint1 Dint4 Dint4 Dint2 Dint2 Dint4 Dint4 Dint3 Dint3 Dint4	LD,DMOV,LDD>
BOOL	bBit4:= MIN(bBit1,bBit2,bBit3);	LD AND AND OUT	bBit1 bBit2 bBit3 bBit4	LD,AND,OUT
STRING	Str4:= MIN(Str1,Str2,Str3);	LD $MOV LD$> $MOV LD$> $MOV	SM400 Str1 Str4 Str4 Str2 Str2 Str4 Str4 Str3 Str3 Str4	LD,MOV,LD>

6 - 51

■ Function definition BOOL MIN_E (BOOL EN, ANY_SIMPLE S1, ANY_SIMPLE S2, ANY_SIMPLE S2,…., ANY_SIMPLE Sn, ANY_SIMPLE D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Search target data
D1	OUT	Search result

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* BOOL MIN_E(BOOL EN, ANY_SIMPLE S1, ANY_SIMPLE S2,….,          *)
(* ANY_SIMPLE Sn, ANY_SIMPLE D1 );                                *)
      M0 := MIN_E( X0, iData1, iData2, iData3, Result ) ;
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 52

6.8.4 Limiter LIMIT
LIMIT_E

The output value is controlled depending on whether the specified data is within the upper/lower limit value (minimum/maximum output limit value) range or not.

■ Function definition

ANY_SIMPLE LIMIT(ANY_SIMPLE MIN, ANY_SIMPLE S1, ANY_SIMPLE MAX);

● Argument

Argument Name	IN/OUT	Description
MIN	IN	Minimum output limit value
S1	IN	Input value
MAX	IN	Maximum output limit value

● Return value

Return Value	Description
ANY_SIMPLE	Output value When MIN (lower limit value) > S1 (input value) ..... Return value = MIN (lower limit value) When MAX (upper limit value) < S1 (input value) ..... Return value = MAX (upper limit value) When MIN (lower limit value) ≦ S1 (input value) ≦ MAX (upper limit value) ..... Return value = S1 (input value)

● Example of use

Argument Type	ST Program	Conversion Result			Used Instruction
REAL	Real4:= LIMIT(Real1,Real2,Real3);	LDE>= ANDE<= EMOV LDE< EMOV LDE> EMOV	Real2 Real1 Real2 Real3 Real2 Real4 Real2 Real1 Real1 Real4 Real2 Real3 Real3 Real4		LDE>=,ANDE<=,EMOV, LDE<,LDE>
INT	Int4:= LIMIT(Int1,Int2,Int3);	LD LIMIT	SM400 Int1 Int2	 Int3 Int4	LD,LIMIT
DINT	Dint4:= LIMIT(Dint1,Dint2,Dint3);	LD DLIMIT	SM400 Dint1 Dint2	 Dint3 Dint4	LD,DLIMIT
BOOL	bBit4:= LIMIT(bBit1,bBit2,bBit3);	LD OR AND OUT	bBit2 bBit1 bBit3 bBit4		LD,OR,AND,OUT
STRING	Str4:= LIMIT(Str1,Str2,Str3);	LD$>= AND$<= $MOV LD$< $MOV LD$> $MOV	Str2 Str1 Str2 Str3 Str2 Str4 Str2 Str1 Str1 Str4 Str2 Str3 Str3 Str4		LD$>=,AND$<=,MOV, LD<,LD$>

6 - 53

■ Function definition

BOOL LIMIT_E(BOOL EN, ANY_SIMPLE MIN, ANY_SIMPLE S1, ANY_SIMPLE MAX, ANY_SIMPLE D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
MIN	IN	Minimum output limit value
S1	IN	Input value
MAX	IN	Maximum output limit value
D1	OUT	Output value When MIN (lower limit value) > S1 (input value) ..... D1 = MIN (lower limit value) When MAX (upper limit value) < S1 (input value) ..... D1 = MAX (upper limit value) When MIN (lower limit value) ≦ S1 (input value) ≦ MAX (upper limit value) ..... D1 = S1 (input value)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the iData1 value is stored into Result  *)
(* if the iData2 data is less than the iData1 data or minimum value, the iData3   *)
(* value is stored if the iData2 data is greater than the iData3 data or maximum  *)
(* value, or the iData2 value is stored otherwise.                                *)
    M0 := LIMIT_E( X0, iData1, iData2, iData3, Result ) ;
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 54

6.8.5 Multiplexer MUX
MUX_E

One data is selected from among the specified data according to the specified selection condition.

■ Function definition

ANY MUX (INT n, ANY S1, ANY S2,….,ANY Sn);

● Argument

Argument Name	IN/OUT	Description
n	IN	Selection condition
S1 to Sn	IN	Selection target data

● Return value

Return Value	Description
ANY	Selection result When n = 1, return value = S1 When n = 2, return value = S2 : : When n = n, return value = Sn

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
REAL	Real4 := MUX(Int1, Real1,Real2, Real3);	LD= Int1 K1 EMOV Real1 Real4 LD= Int1 K2 EMOV Real2 Real4 LD= Int1 K3 EMOV Real3 Real4	LD=,EMOV
INT	Int4:= MUX(wCon1 , Int1 , Int2, Int3);	LD= wCon1 K1 MOV Int1 Int4 LD= wCon1 K2 MOV Int2 Int4 LD= wCon1 K3 MOV Int3 Int4	LD=,MOV
DINT	Dint4:= MUX(D0, Dint1,Dint2,Dint3);	LD= D0 K1 DMOV Dint1 Dint4 LD= D0 K2 DMOV Dint2 Dint4 LD= D0 K3 DMOV Dint3 Dint4	LD=,DMOV
BOOL	bBit4:= MUX(3,bBit1,bBit2,bBit3);	LD= K3 K1 MPS AND bBit1 SET bBit4 MPP ANI bBit1 RST bBit4 LD= K3 K2 MPS AND bBit2 SET bBit4 MPP ANI bBit2 RST bBit4 LD= K3 K3 MPS AND bBit3 SET bBit4 MPP ANI bBit3 RST bBit4	LD=,MPS,AND,SET, MPP,ANI,RST

■ Function definition

BOOL MUX_E(BOOL EN, INT n, ANY S1, ANY S2,….,ANY Sn, ANY D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
n	IN	Selection condition
S1 to Sn	IN	Selection target data
D1	OUT	Selection result When n = 1, D1 = S1 When n = 2, D1 = S2 : : When n = n, D1 = Sn

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, one of the data stored in iData2,      *)
(* iData3, iData4 and iData5 is stored into Result, after judgment made from the *)
(* data in iData1.                                                               *)
    M0 := MUX_E( X0, iData1, iData2, iData3, iData4, iData5, Result );
```

For the usable data type, refer to "3.2.2 About ANY type".

6 - 56

6.9 Comparison Functions

6.9.1 Greater than right member (>) GT_E

In all the specified data, whether the relationship of > (greater than) is satisfied or not is acquired.

■ Function definition

BOOL GT_E(BOOL EN, ANY_SIMPLE S1, ANY_SIMPLE S2,…., ANY_SIMPLE Sn, BOOL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Comparison target data
D1	OUT	Comparison result

Remarks: D1 = (S1 > S2) & (S2 > S3) & ..... & (Sn -1 > Sn)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
REAL	GT_E(M0 , Real1, Real2, Real3, bBit1);	LDE> Real1 Real2 ANDE> Real2 Real3 OUT M8191 LD M0 AND M8191 SET bBit1 LD M0 ANI M8191 RST bBit1	LDE>, ANDE>,OUT, LD,AND,SET,ANI, RST
INT	GT_E(M0 , Int1, Int2, Int3, bBit1);	LD> Int1 Int2 AND> Int2 Int3 OUT M8191 LD M0 AND M8191 SET bBit1 LD M0 ANI M8191 RST bBit1	LD>,AND>,OUT,LD, AND,SET,ANI,RST
DINT	GT_E(M0 , Dint1, Dint2 , Dint3, bBit1);	LDD> Dint1 Dint2 ANDD> Dint2 Dint3 OUT M8191 LD M0 AND M8191 SET bBit1 LD M0 ANI M8191 RST bBit1	LDD>, ANDD>,OUT LD, AND,SET,ANI, RST

6 - 57

Argument Type	ST Program	Conversion Result	Used Instruction
BOOL	GT_E(M0 , M100, M101, M102, M103, bBit1);	LD M100 ANI M101 LD M101 ANI M102 ANB LD M102 ANI M103 ANB OUT M8191 LD M0 AND M8191 SET bBit1 LD M0 ANI M8191 RST bBit1	LD,ANI,ANB,OUT, AND,SET,RST
STRING	GT_E(M0 , Str1, Str2 , Str3, bBit1);	LD$> Str1 Str2 AND$> Str2 Str3 OUT M8191 LD M0 AND M8191 SET bBit1 LD M0 ANI M8191 RST bBit1	LD$>, AND$>, OUT LD, AND, SET, ANI RST

For the usable data type, refer to "3.2.2 About ANY type".

6 - 58

6.9.2 Greater than or equal to right member (>=) GE_E

In all the specified data, whether the relationship of ≧ (greater than or equal to) is satisfied or not is acquired.

■ Function definition

BOOL GE_E(BOOL EN, ANY_SIMPLE S1, ANY_SIMPLE S2,…., ANY_SIMPLE Sn, BOOL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Comparison target data
D1	OUT	Comparison result

Remarks: D1 = (S1 ≧ S2) & (S2 ≧ S3) & ..... & (Sn -1 ≧ Sn)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
REAL	GE_E(M0 , Real1, Real2, Real3, bBit1);	LDE>= Real1 Real2 ANDE>= Real2 Real3 OUT M8191 LD M0 AND M8191 SET bBit1 LD M0 ANI M8191 RST bBit1	LDE>=, ANDE>=, OUT,LD,AND,SET, ANI,RST
INT	GE_E(M0 , Int1, Int2, Int3, bBit1);	LD>= Int1 Int2 AND>= Int2 Int3 OUT M8191 LD M0 AND M8191 SET bBit1 LD M0 ANI M8191 RST bBit1	LD>=, AND>=,OUT LD,AND,SET,ANI, RST
DINT	GE_E(M0 , Dint1, Dint2 , Dint3, bBit1);	LDD>= Dint1 Dint2 ANDD>= Dint2 Dint3 OUT M8191 LD M0 AND M8191 SET bBit1 LD M0 ANI M8191 RST bBit1	LDD>=, ANDD>=, OUT,LD,AND,SET ANI,RST

6 - 59

Argument Type	ST Program	Conversion Result	Used Instruction
BOOL	GE_E(M0 , M100, M101, M102, M103, bBit1);	LD M100 ORI M101 LD M101 ORI M102 ANB LD M102 ORI M103 ANB OUT M8191 LD M0 AND M8191 SET bBit1 LD M0 ANI M8191 RST bBit1	LD,ORI,ANB,OUT AND,SET,ANI,RST
STRING	GE_E(M0 , Str1, Str2 , Str3, bBit1);	LD$>= Str1 Str2 AND$>= Str2 Str3 OUT M8191 LD M0 AND M8191 SET bBit1 LD M0 ANI M8191 RST bBit1	LD$>=, AND$>=, OUT,LD,AND,SET, LD,ANI,RST

For the usable data type, refer to "3.2.2 About ANY type".

6 - 60

6.9.3 Equal (=) EQ_E

In all the specified data, whether the relationship of = (equal) is satisfied or not is acquired.

■ Function definition

BOOL EQ_E(BOOL EN, ANY_SIMPLE S1, ANY_SIMPLE S2,…., ANY_SIMPLE Sn, BOOL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Comparison target data
D1	OUT	Comparison result

Remarks: D1 = (S1 = S2) & (S2 = S3) & ..... & (Sn -1 = Sn)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
REAL	b_result := EQ_E(b_select, r_data1, r_data2, r_data3, b_data1);	LDE= r_data1 r_data2 ANDE= r_data2 r_data3 OUT M8191 LD b_select AND M8191 SET b_data1 LD b_select ANI M8191 RST b_data1 LD b_select OUT b_result	LDE=, ANDE=,OUT LD,AND,SET,ANI, RST
INT	B100 := EQ_E(M20, D10, D20, D30, M200);	LD= D10 D20 AND= D20 D30 OUT M8191 LD M20 AND M8191 SET M200 LD M20 ANI M8191 RST M200 LD M20 OUT B100	LD=, AND=, OUT LD,AND,SET,ANI RST
DINT	b_result := EQ_E(b_select, di_data1, di_data2, di_data3, b_data1);	LDD= di_data1 di_data2 ANDD= di_data2 di_data3 OUT M8191 LD b_select AND M8191 SET b_data1 LD b_select ANI M8191 RST b_data1 LD b_select OUT b_result	LDD=, ANDD=,OUT, LD,AND,SET,ANI, RST

Argument Type	ST Program	Conversion Result	Used Instruction
BOOL	b_result := EQ_E(b_select, X10, X11, X12, M20);	LD X10 AND X11 LDI X10 ANI X11 ORB LD X11 AND X12 LDI X11 ANI X12 ORB ANB OUT M8191 LD b_select AND M8191 SET M20 LD b_select ANI M8191 RST M20 LD b_select OUT b_result	LD,AND,LDI,ANI, ORB,ANB,SET,RST
STRING	b_result := EQ_E(b_select, s_ary1, s_ary2, b_data1);	LD$= s_ary1 s_ary2 OUT M8191 LD b_select AND M8191 SET b_data1 LD b_select ANI M8191 RST b_data1 LD b_select OUT b_result	LD$=,OUT,LD,AND, SET,ANI,RST

For the usable data type, refer to "3.2.2 About ANY type".

6 - 62

6.9.4 Less than or equal to right member (<=) LE_E

In all the specified data, whether the relationship of ≦ (less than or equal to) is satisfied or not is acquired.

■ Function definition

BOOL LE_E(BOOL EN, ANY_SIMPLE S1, ANY_SIMPLE S2,…., ANY_SIMPLE Sn, BOOL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Comparison target data
D1	OUT	Comparison result

Remarks: D1 = (S1 ≦ S2) & (S2 ≦ S3) & ..... & (Sn -1 ≦ Sn)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
REAL	b_result := LE_E(b_select, r_data1, r_data2, r_data3, b_data1);	LDE<= r_data1 r_data2 ANDE<= r_data2 r_data3 OUT M8191 LD b_select AND M8191 SET b_data1 LD b_select ANI M8191 RST b_data1 LD b_select OUT b_result	LDE<=, ANDE<=, OUT,LD,AND,SET, ANI,RST
INT	B100 := LE_E(M20, D10, D20, D30, M200);	LD<= D10 D20 AND<= D20 D30 OUT M8191 LD M20 AND M8191 SET M200 LD M20 ANI M8191 RST M200 LD M20 OUT B100	LD<=, AND<=,OUT LD,AND,SET,ANI, RST

6 - 63

Argument Type	ST Program	Conversion Result	Used Instruction
DINT	b_result := LE_E(b_select, di_data1, di_data2, di_data3, b_data1);	LDD<= di_data1 di_data2 ANDD<= di_data2 di_data3 OUT M8191 LD b_select AND M8191 SET b_data1 LD b_select ANI M8191 RST b_data1 LD b_select OUT b_result	LDD<=, ANDD<=, OUT,LD,AND,SET, ANI,RST
BOOL	b_result := LE_E(b_select, X10, X11, X12, M20);	LDI X10 OR X11 LDI X11 OR X12 ANB OUT M8191 LD b_select AND M8191 SET M20 LD b_select ANI M8191 RST M20 LD b_select OUT b_result	LDI,OR,ANB,OUT, AND,SET,ANI,RST
STRING	b_result := LE_E(b_select, s_ary1, s_ary2, b_data1);	LD$<= s_ary1 s_ary2 OUT M8191 LD b_select AND M8191 SET b_data1 LD b_select ANI M8191 RST b_data1 LD b_select OUT b_result	LD$<=,OUT,LD, AND,SET,ANI,RST

For the usable data type, refer to "3.2.2 About ANY type".

6 - 64

6.9.5 Less than right member (<) LT_E

In all the specified data, whether the relationship of < (less than) is satisfied or not is acquired.

■ Function definition

BOOL LT_E(BOOL EN, ANY_SIMPLE S1, ANY_SIMPLE S2,…., ANY_SIMPLE Sn, BOOL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Comparison target data
D1	OUT	Comparison result

Remarks: D1 = (S1 < S2) & (S2 < S3) & ..... & (Sn -1 < Sn)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
REAL	b_result := LT_E(b_select, r_data1, r_data2, r_data3, b_data1);	LDE< r_data1 r_data2 ANDE< r_data2 r_data3 OUT M8191 LD b_select AND M8191 SET b_data1 LD b_select ANI M8191 RST b_data1 LD b_select OUT b_result	LDE<, ANDE<,OUT, LD,AND,SET,ANI, RST
INT	B100 := LT_E(M20, D10, D20, D30, M200);	LD< D10 D20 AND< D20 D30 OUT M8191 LD M20 AND M8191 SET M200 LD M20 ANI M8191 RST M200 LD M20 OUT B100	LD<, AND<,OUT, LD,SET,ANI,RST

6 - 65

Argument Type	ST Program	Conversion Result	Used Instruction
DINT	b_result := LT_E(b_select, di_data1, di_data2, di_data3, b_data1);	LDD< di_data1 di_data2 ANDD< di_data2 di_data3 OUT M8191 LD b_select AND M8191 SET b_data1 LD b_select ANI M8191 RST b_data1 LD b_select OUT b_result	LDD<, ANDD<,OUT, LD,AND,SET,ANI, RST
BOOL	b_result := LT_E(b_select, X10, X11, X12, M20);	LDI X10 AND X11 LDI X11 AND X12 ANB OUT M8191 LD b_select AND M8191 SET M20 LD b_select ANI M8191 RST M20 LD b_select OUT b_result	LDI,AND,ANB,OUT, LD,SET,ANI,RST
STRING	b_result := LT_E(b_select, s_ary1, s_ary2, b_data1);	LD$< s_ary1 s_ary2 OUT M8191 LD b_select AND M8191 SET b_data1 LD b_select ANI M8191 RST b_data1 LD b_select OUT b_result	LD$<,OUT,LD,AND, SET,ANI,RST

For the usable data type, refer to "3.2.2 About ANY type".

6 - 66

6.9.6 Unequal (<>) NE_E

In all the specified data, whether the relationship of ≠ (unequal) is satisfied or not is acquired.

■ Function definition

BOOL NE_E(BOOL EN, ANY_SIMPLE S1, ANY_SIMPLE S2, BOOL D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Comparison target data
S2	IN	Comparison target data
D1	OUT	Comparison result

Remarks: D1 = (S1 ≠ S2)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
REAL	b_result := NE_E(b_select, r_data1, r_data2, b_data1);	LDE<> r_data1 r_data2 OUT M8191 LD b_select AND M8191 SET b_data1 LD b_select ANI M8191 RST b_data1 LD b_select OUT b_result	LDE<>,OUT,LD, AND,SET,ANI,RST
INT	B100 := NE_E(M20, D10, D20, M200);	LD<> D10 D20 OUT M8191 LD M20 AND M8191 SET M200 LD M20 ANI M8191 RST M200 LD M20 OUT B100	LD<>,OUT,LD,AND, SET,ANI,RST
DINT	b_result := NE_E(b_select, di_data1, di_data2, b_data1);	LDD<> di_data1 di_data2 OUT M8191 LD b_select AND M8191 SET b_data1 LD b_select ANI M8191 RST b_data1 LD b_select OUT b_result	LDD<>,OUT,LD,AND, SET,ANI,RST

Argument Type	ST Program	Conversion Result	Used Instruction
BOOL	b_result := NE_E(b_select, X10, X11, M20);	LD X10 ANI X11 LDI X10 AND X11 ORB OUT M8191 LD b_select AND M8191 SET M20 LD b_select ANI M8191 RST M20 LD b_select OUT b_result	LD,ANI,LDI,AND, ORB,OUT,SET,RST
STRING	b_result := NE_E(b_select, s_ary1, s_ary2, b_data1);	LD$<> s_ary1 s_ary2 OUT M8191 LD b_select AND M8191 SET b_data1 LD b_select ANI M8191 RST b_data1 LD b_select OUT b_result	LD$<>,OUT,LD,AND, SET,ANI,RST

For the usable data type, refer to "3.2.2 About ANY type".

6 - 68

6.10 Character String Functions

6.10.1 Character string length acquisition LEN LEN_E

The character string length of the specified character string data is acquired.

■ Function definition

INT LEN (STRING S1);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data whose character string length will be acquired (character string data)

● Return value

Return Value	Description
INT	Character string length result (BIN 16-bit data)

Remarks: This function cannot be used with the Basic model QCPU.

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
STRING	i_data1 := LEN(s_ary1);	LD SM400 LEN s_ary1 i_data1	LD,LEN

■ Function definition

BOOL LEN_E(BOOL EN, STRING S1, INT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data whose character string length will be acquired (character string data)
D1	OUT	Character string length result (BIN 16-bit data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the length of the character string      *)
(* stored in sData1 is acquired and stored into Result.                          *)
    M0 := LEN_E( X0, sData, Result ) ;
```


6 - 69

6.10.2 Acquisition from start position of character string LEFT LEFT_E

The specified n characters of character string is acquired, starting at the left of the specified character string (head of the character string).

■ Function definition

STRING LEFT (STRING S1, INT n);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be acquired (character string data)
n	IN	Number of characters to be acquired (BIN 16-bit data)

● Return value

Return Value	Description
STRING	Acquisition result (character string data)

Remarks: This function cannot be used with the Basic model QCPU.
Secure the area of n+1 characters as the data area that will store the acquired character string data.

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
STRING	s_ary1 := LEFT (s_ary2, i_data1);	LD SM400 LEFT s_ary2 s_ary1 i_data1	LD,LEFT

■ Function definition

BOOL LEFT _E(BOOL EN, STRING S1, INT n, STRING D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be acquired (character string data)
n	IN	Number of characters to be acquired (BIN 16-bit data)
D1	OUT	Acquisition result (character string data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string of the number of  *)
(* characters specified in iData is acquired, starting at the left of the character  *)
(* string data stored in sData, and stored into Result.  *)
    M0 := LEFT_E( X0, sData, iData, Result ) ;
```


6 - 70

6.10.3 Acquisition from end of character string RIGHT RIGHT_E

The specified n characters of character string is acquired, starting at the right of the specified character string (end of the character string).

■ Function definition

STRING RIGHT (STRING S1, INT n);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be acquired (character string data)
n	IN	Number of characters to be acquired (BIN 16-bit data)

● Return value

Return Value	Description
STRING	Acquisition result (character string data)

Remarks: This function cannot be used with the Basic model QCPU.
Secure the area of n+1 characters as the data area that will store the acquired character string data.

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
STRING	s_ary1 := RIGHT(s_ary2, i_data1);	LD	SM400	LD,RIGHT
		RIGHT	s_ary2	
			s_ary1	
			i_data1	

■ Function definition

BOOL RIGHT _E(BOOL EN, STRING S1, INT n, STRING D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be acquired (character string data)
n	IN	Number of characters to be acquired (BIN 16-bit data)
D1	OUT	Acquisition result (character string data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string of the number of  *)
(* characters specified in iData is acquired, starting at the right of the character  *)
(* string stored in sData, and stored into Result.                                  *)
    M0 := RIGHT_E( X0, sData, iData, Result ) ;
```

6.10.4 Acquisition from specified position of character string MID MID_E

The specified n characters of character string data is acquired, starting at the specified position from the left of the specified character string (head of the character string).

■ Function definition

STRING MID(STRING S1, INT n, INT POS);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be acquired (character string data)
n	IN	Number of characters to be acquired (BIN 16-bit data)
POS	IN	Head position of data to be acquired (BIN 16-bit data)

● Return value

Return Value	Description
STRING	Acquisition result (character string data)

Remarks: This function cannot be used with the Basic model QCPU.
Secure the area of n+1 characters as the data area that will store the acquired character string data.

● Example of use

Argument Type	ST Program	Conversion Result			Used Instruction
STRING	s_ary1 := MID(s_ary2, i_data1, i_data2);	LD MOV MOV MIDR 	SM400 i_data1 D10239 i_data2 D10238 s_ary2 D10238	 s_ary1 	LD,MOV,MIDR

■ Function definition

BOOL MID_E(BOOL EN, STRING S1, INT n, INT POS, STRING D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be acquired (character string data)
n	IN	Number of characters to be acquired (BIN 16-bit data)
POS	IN	Head position of data to be acquired (BIN 16-bit data)
D1	OUT	Acquisition result (character string data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the specified number of characters   *)
(* stored in iData1 are acquired,  starting at the iData2 position from the head of  *)
(* the character string stored in sData, and stored into Result.               *)
    M0 := MID_E( X0, sData, iData1, iData2, Result ) ;
```

6.10.5 Concatenation of character strings CONCAT CONCAT_E

All the specified character strings are concatenated.

■ Function definition

STRING CONCAT(STRING S1, STRING S2,….,STRING Sn);

● Argument

Argument Name	IN/OUT	Description
S1 to Sn	IN	Data to be concatenated (character string data)

● Return value

Return Value	Description
STRING	Concatenation result (character string data)

Remarks: This function cannot be used with the Basic model QCPU.
Secure the area of the number of concatenated characters + 1 character as the data area that will store the concatenated character string data.

● Example of use

Argument Type	ST Program	Conversion Result		Used Instruction
STRING	s_result :=	LD	SM400	LD,$MOV,$+
	CONCAT(s_ary1, s_ary2,	$MOV	s_ary1	
	s_ary3);		s_result	
		$+	s_ary2	
			s_result	
		$+	s_ary3	
			s_result	

■ Function definition

BOOL_ CONCAT_E(BOOL EN, STRING S1, STRING S2,….,STRING Sn, STRING D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1 to Sn	IN	Data to be concatenated (character string data)
D1	OUT	Concatenation result (character string data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string data stored in  *)
(* sData1 and sData2 are concatenated, and the result is stored into Result.  *)
    M0 := CONCAT_E( X0, sData1, sData2, Result ) ;
```

6.10.6 Insertion of character string into specified position INSERT INSERT_E

The character string data is inserted into the specified position and later of the specified character string data.

■ Function definition

STRING INSERT(STRING S1, STRING S2, INT POS);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be inserted (character string data)
S2	IN	Data into which above data will be inserted (character string data)
POS	IN	Insertion position (BIN 16-bit data)

● Return value

Return Value	Description
STRING	Insertion result (character string data)

Remarks: This function cannot be used with the Basic model QCPU.
Secure the area of the number of characters after insertion + 1 character as the data area that will store the character string data after insertion.

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
STRING	w_Str3 := INSERT(w_Str1,w_Str2,w_Word1);	LD SM400 $+ w_Str2 w_Str1 w_Str3 AND<> w_Word1 K1 MOV K1 D10238 - w_Word1 K1 D10239 MIDW w_Str1 w_Str3 D10238 MOV w_Word1 D10238 LEN w_Str2 D10239 MIDW w_Str2 w_Str3 D10238	LD,$+,AND<>,MOV ,-,MIDW,LEN

■ Function definition

BOOL INSERT_E(BOOL EN, STRING S1, STRING S2, INT POS, STRING D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be inserted (character string data)
S2	IN	Data into which above data will be inserted (character string data)
POS	IN	Insertion position (BIN 16-bit data)
D1	OUT	Insertion result (character string data)

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string data in sData2 is  *)
(* inserted into the iData position from the head of the character string data in  *)
(* sData1, and the result is stored into Result.                                   *)
    M0 := INSERT_E( X0, sData1, sData2, iData, Result ) ;
```

6.10.7 Deletion of character string from specified position DELETE DELETE_E

n characters of character string is deleted from the specified position and later of the specified character string.

■ Function definition

STRING DELETE(STRING S1, INT n, INT POS);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be deleted (character string data)
n	IN	Number of characters to be deleted (BIN 16-bit data)
POS	IN	Deletion position (BIN 16-bit data)

● Return value

Return Value	Description
STRING	Deletion result (character string data)

Remarks: This function cannot be used with the Basic model QCPU.
Secure the area of the number of characters after deletion + 1 character as the data area that will store the character string data after deletion.
If the deletion position POS is 0, n characters of character string will be deleted, starting at the end (right) of the data to be deleted S1.

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
STRING	w_Str2 := DELETE(w_Str1,w_Word1, w_Word2);	LD SM400 LEN w_Str1 D10238 - w_Word1 D10238 RIGHT w_Str1 w_Str2 D10238 - w_Word2 K1 D10239 AND<> 10239 K0 MOV K1 D10238 MIDW w_Str1 w_Str2 D10238	LD,LEN,-,RIGHT, AND<>,MOV,MIDW

■ Function definition

BOOL DELETE_E(BOOL EN, STRING S1, INT n, INT POS, STRING D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be deleted (character string data)
n	IN	Number of characters to be deleted (BIN 16-bit data)
POS	IN	Deletion position (BIN 16-bit data)
D1	OUT	Deletion result (character string data)

Remarks: Secure the area of the number of characters after deletion + 1 character as the data area that will store the character string data after deletion.

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string of the number of  *)
(* characters specified in iData1 is deleted, starting at iData2 from the head of  *)
(* the character string data in sData, and the result is stored into Result.)  *)
    M0 := DELETE_E( X0, sData, iData1, iData2, Result ) ;
```

6.10.8 Replacement of character string from specified position REPLACE REPLACE_E

n characters of character string data starting at the specified position of the specified character string data is replaced by the specified character string.

■ Function definition STRING REPLACE(STRING S1, STRING S2, INT n, INT POS);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Data to be replaced (character string data)
S2	IN	Data that will replace (character string data)
n	IN	Number of characters to be replaced (BIN 16-bit data)
POS	IN	Replacement start position (BIN 16-bit data)

● Return value

Return Value	Description
STRING	Replacement result (character string data)

Remarks: This function cannot be used with the Basic model QCPU.
Secure the area of the number of characters after replacement + 1 character as the data area that will store the character string data after replacement.

● Example of use

Argument Type	ST Program	Conversion Result			Used Instruction
STRING	w_Str3 := REPLACE(w_Str1,w_Str2, w_Word1, w_Word2);	LD $MOV MOV MOV MIDW	SM400 w_Str1 w_Word1 D10239 w_Word2 D10238 w_Str2 D10238	 w_Str3 w_Str3	LD,$MOV,MOV, MIDW

■ Function definition BOOL REPLACE_E(BOOL EN, STRING S1, STRING S2, INT n, INT POS, STRING D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Data to be replaced (character string data)
S2	IN	Data that will replace (character string data)
n	IN	Number of characters to be replaced (BIN 16-bit data)
POS	IN	Replacement start position (BIN 16-bit data)
D1	OUT	Replacement result (character string data)

Remarks: Secure the area of the number of characters after replacement + 1 character as the data area that will store the character string data after replacement.

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(* When execution condition X0 turns ON, the character string data of the      *)
(* number of characters specified in iData1, starting at iData2 from the head of  *)
(* the character string data in sData1, is replaced by the character string data in  *)
(* sData2, and the result is stored into Result.                                *)
    M0 := REPLACE_E( X0, sData1, sData2, iData1, iData2, Result ) ;
```

6.10.9 Search for character string from specified position FIND FIND_E

The specified character string is searched for the specified character string.

■ Function definition

INT FIND(STRING S1, STRING S2);

● Argument

Argument Name	IN/OUT	Description
S1	IN	Character string to be searched (character string data)
S2	IN	Character string to be searched for (character string data)

● Return value

Return Value	Description
INT	Position where character string is found first (BIN 16-bit data)

Remarks: This function cannot be used with the Basic model QCPU.
If the character string is not found, the return value turns to 0.

● Example of use

Argument Type	ST Program	Conversion Result	Used Instruction
STRING	w_Word1:= FIND(w_Str1,w_Str2);	LD SM400 INSTR w_Str2 w_Str1 w_Word1 K1	LD,INSTR

■ Function definition

BOOL FIND_E(BOOL EN, STRING S1, STRING S2, INT D1);

● Argument

Argument Name	IN/OUT	Description
EN	IN	Execution condition (Function is executed only when the result is TRUE)
S1	IN	Character string to be searched (character string data)
S2	IN	Character string to be searched for (character string data)
D1	OUT	Position where character string is found first (BIN 16-bit data)

Remarks: If the character string is not found, the return value turns to 0.

● Return value

Return Value	Description
BOOL	Execution condition

● Example of use

```
(*  When execution condition X0 turns ON, the character string data in sData1 is  *)
(*  searched for the character string data in sData2, and the position where the    *)
(*  character string is found first is stored into Result.                          *)
    M0 := FIND_E( X0, sData1, sData2, Result ) ;
```

7 ERROR LIST

This chapter explains the errors that may occur during conversion of a created ST program.
For the execution errors that may occur when the ST program is written to the CPU module, refer to the "MELSEC-Q/L Programming Manual (Common Instructions)", "QCPU User's Manual (Hardware Design, Maintenance and Inspection)" or "MELSEC-L CPU Module User's Manual (Hardware Design, Maintenance and Inspection)".

● When conversion error occurs
The error dialog corresponding to the error in the program is displayed.
The maximum number of errors that may occur in a single program is 1000. Errors in excess of 1000 errors are not displayed in the error list.

● About conversion error indication
More than one error may be displayed for a single program statement, or more than one message may be displayed for a single error.

● Conversion error list (error message, cause, corrective action)

No.	Error Message	Cause	Corrective Action
1	An unanalyzable character exists. (C1009)	The character string that cannot be analyzed exists. As character strings that cannot be analyzed, there are the following examples. Example 1: 2## The format is wrong. Example 2: STRING type: STRV1 is defined. STRV1 := $"abc"; The $ symbol is used. Example 3: D0 := !10; The ! symbol is used. Example 4: J25|\K4X0 := 5; The | symbol is used.	Correct the character string.
2	An unanalyzable operator exists. (C1010)	The operator that cannot be analyzed exists. Example 1: Y0 : = M0 => M1;	Correct the operator.
3	The real number constant is wrong. (C1013)	The description of the real number constant is illegal. As illegal descriptions, there are the following examples. Example 1: REAL type : RealV1 is defined. RealV1 := 1.; The format of the real number constant is wrong. Example 2: RealV1 := 0.1E; The format of the real number constant is wrong.	Correct the description of the real number constant.

7 - 1

No.	Error Message	Cause	Corrective Action
4	Description of a device is wrong. (C1014)	The description of the device is illegal. As illegal descriptions, there are the following examples. Example 1: D0.10 := TRUE; The bit No. specification of the word device is wrong. Example 2: D0@ := 0;	Correct the description of the device.
5	Description of a device is wrong. (C1017)	The description of the device is illegal. As an illegal description, there is the following example. Example 1: D0 := %MMW0.10; The device was described in an unusable format.	Correct the description of the device.
6	Description of a comment is wrong. (C1018)	The description of the comment is illegal. It is not written in the "(*" "*)" format. As illegal descriptions, there are the following examples. Example 1: (* * A parenthesis is insufficient. Example 2: (*(* The parenthesis and * format is wrong. Example 3: (* *) There is a space between "*" and ")". Example 4: (*aaaaa) * is insufficient.	Correct the description of the comment.
7	Description of a character constant is wrong. (C1019)	The description of the character string constant is illegal. As illegal descriptions, there are the following examples. Example 1: STRING type : STRV1 is defined. STRV1 := """"; Example 2: STRV1 := ''; '' is insufficient. Example 3: STRV1 := ''' character '; There is ' within the character string ''. Example 4: STRV1 := '$'; The method of using the escape sequence is wrong.	Correct the description of the character string constant.

7 - 2

No.	Error Message	Cause	Corrective Action
8	Description of a constant is wrong. (C1020 to C1023)	The unsupported data type was used, or the constant was described wrongly. As illegal descriptions, there are the following examples. Example 1: W_TMP := TIME#1100_0101; Example 2: W_TMP := T#0; Example 3: W_TMP := 2#0; Example 4: W_TMP := DT#1900-01-01, 00:00:00; Example 5: W_TMP := D#1994-06; Example 6: W_TMP := TOD#09:30:61;	The used data type is not supported. Correct the data type. Correct the description of the constant.
9	Variable '*1' undefined. (C1028) (Variable name enters *1.)	An undefined variable was used. There are the following examples of using undefined variables. Example 1: I_TEST := 1 ; The label is used without label setting being made. Example 2: D0 := HAAH; Characters other than A to F are used in hexadecimal. Example 3: D0 := 1234 ;	Define the used variable.
10	An error is in element specification of array. (C1033)	The method of specifying the array element is wrong. Example 1: Word type array label: W_ARY W_ARY[0,1] := 1; The array was described in the format different from the defined one.	Correct the description of the array.
11	Function '*1' is undefined. (C1049) (Function name enters *1.)	An undefined function was used. There is the following example of using an undefined function. Example 1: Real number type label: RE_1 M0 := OS_E_MD(TRUE,E1.0,RE_1);	Correct the description of the function name.
12	A variable name or a device name is too long. (C1077)	The variable name has more than 16 characters, or the device name is too long. There are the following program examples that will result in an error. Example 1: abcde678901234567 := D10; Example 2: D0 := D000000 • • • 00000000000001; The device name is too long.	Use the defined variable name. Correct the description of the device.

No.	Error Message	Cause	Corrective Action
13	It is used except a constant for the %d argument. (C2021) (Argument error location enters *1.)	Other than a constant was used in the argument for which a constant should be specified. There are the following program examples that will result in an error. Example 1: M1 := ROL(M0,X0); Other than a constant was used in argument No. 2. Example 2: D100 := SHL(D0,D1); Other than a constant was used in argument No. 2.	Use a constant in the specified argument.
14	Syntax error. (C2054)	Wrong grammar was described. There are the following examples where grammar will be illegal. Example 1: D0 : 0; "=" is not described in the assignment statement. Example 2: FOR ARY[0] := 0 TO D10 BY D20 DO D100 := D100+1; END_FOR; The array element was specified in the repeat variable. Example 3: FOR STR.W_TMP := 0 TO D10 BY D20 DO D100 := D100+1; END_FOR; The structure element was specified in the repeat variable. Example 4: D0 := 1++++++++2; The method of using the + operator is wrong. Example 5: Word type array : IntAry1 D0 := IntAry1[[0; The method of describing the array is wrong.	Correct the grammar.

No.	Error Message	Cause	Corrective Action

15	'*1' missing. (C8006)
	(END FOR
	;
	END WHILE
	END FOR
	END_REPEAT
	END_CASE
	or
	END_IF
	enters *1.)
		The statement is not ended by ";".	Describe ";" at the end of the statement.

		"END FOR" is not described in the FOR syntax.	Describe "END FOR" in the FOR syntax.
		"END WHILE" is not described in the WHILE syntax.	Describe "END WHILE" in the WHILE syntax.
		"END_REPEAT" is not described in the REPEAT syntax.	Describe "END_REPEAT" in the REPEAT syntax.
		"END_CASE" is not described in the CASE conditional statement.	Describe "END_CASE" in the CASE conditional statement.
		"END_IF" is not described in the IF conditional statement.	Describe "END_IF" in the IF conditional statement.
16	EXIT outside a loop statement. (C8009)	The EXIT syntax is described outside the loop syntax.	Describe the EXIT syntax in the loop syntax.
17	Description of a constant is wrong. (C8010)	The unsupported data type was used. Example 1: Timer label: wTime wTime := T#1111111111111111 1s;	The used data type is not supported. Correct the data type.
18	Undefined FB was called. (C8011)	The undefined FB was called. There are the following examples of using undefined FBs. Example 1: FB_1(); The undefined FB is called. Example 2: Word type label: W_TMP W_TMP(); The variable other than FB is described.	Define the used FB.
19	The value is not specified to IN/IN_OUT variable '*1'. (C8012) (Input or I/O variable name enters *1.)	No value has been specified in the input or I/O variable of the FB. There are the following examples that will result in the above error. Example 1: I/O variable: IO_TEST1 Diverted FB name: FB1 FB1(); Example 2: I/O variable: IO_TEST1 Diverted FB name: FB1 FB1(IO_TEST); No value is assigned to the input variable.	Specify a value in the input or I/O variable of the FB.

No.	Error Message	Cause	Corrective Action
20	Type mismatch at parameter '*1'. (C8013) (Argument name enters *1.)	The FB call argument does not match in type with the specified value or variable. There are the following examples that will result in the above error. Example 1: Input variable (word type): IN1 Diverted FB name: FB1 FB1(IN1 := TRUE); A bit type variable is specified in the word type input variable. Example 2: Input variable (word type): IN1 Output variable (word type): OUT1 Diverted FB name: FB1 Double word type: DIN1 FB1(IN1 := DIN1); A double word type variable is specified in the word type input variable.	Match the type with that of the FB call argument.
21	The variable which cannot substitute a value for IN_OUT/OUTPUT variable cannot be specified. ('*1') (C8014) (I/O or output variable name enters *1.)	The variable to which a value cannot be assigned has been specified as the I/O variable or output variable of the called FB. Example 1: I/O variable: IO_TEST1 IO_TEST1 := TRUE; Diverted FB name: FB1 A constant is passed to the I/O variable. Example 2: Input variable: IN1 Output variable: OUT1 Diverted FB name: FB1 Word type constant label: wCon FB1(IN1 :=1,OUT1 := wCon); A constant label is passed to the word type output variable.	Specify a variable to which a value can be assigned as the I/O variable or output variable of the called FB.
22	Variable '*1' which cannot be used as an argument of FB is used. (C8015) (Variable name enters *1.)	The value is passed to the variable other than the input, output or I/O variable of the called FB. There is the following example that will result in the above error. Example 1: Input variable IN1 Output variable OUT1 Variable TEST1 Diverted FB name: FB1 FB1(TEST1 := X10);	For an FB call, do not use the variable other than the input, output or I/O variable of the FB.

No.	Error Message	Cause	Corrective Action
23	Input '*1' was multiply assigned. (C8016) (Argument name enters *1.)	The same argument is used two or more times for an FB call. Example 1: I/O variable (bit type) : INOUT1 Diverted FB name: FB1 FB1(INOUT1 := TRUE,INOUT1 := FALSE);	Do not use the same argument more than once for an FB call.
24	Input '*1' unknown. (C8017) (Argument name enters *1.)	The argument of the FB to be called is not defined. Example 1: I/O variable (bit type) : INOUT1 Diverted FB name: FB1 FB1(TMP_INOUT1 := TRUE);	Define the argument of the FB to be called.
25	Invalid integer literal '*1'. (C8018) (Integer value enters *1.)	The integer value is illegal. Example 1: D1 := 9999999999 ; The integer value is greater than the allowed range.	Correct the integer value to within the allowed range.
26	Constant '*1' is wrong. (C8019) (Constant enters *1.)	The Boolean constant is illegal. Example 1: D1 := 2##0011_0101; The wrong Boolean constant is described. Example 2: M0 :=2 #F; The wrong Boolean constant is described.	Change the description of the Boolean constant into the usable one.
27	It is used except the INT type for the element number of an array variable. (C8021)	Other than the word type is used for element specification. Example 1: Bit type array: BoolAry1 Real number type label: RealVal BoolAry1[RealVal] := x0; Example 2: Bit type array: BoolAry1 BoolAry1[D0<D1] The wrong element specification is described.	Change the data type of the element into the word type.
28	Array subscript is out of bounds. (C8022)	The specified element number exceeds the element range of the array definition. Example 1: Word type array label (number of elements 2): Kosu Unit_No[5] := D0;	Change the element number to the one within the element range of the array definition.
29	The variable which is non-array variable is used as array. (C8023)	The array format syntax was described in the variable that is not an array variable. Example 1: Word type label: W_TMP1 W_TMP1[2] := 100; Described in the array format in the variable that is not an array. Example 2: aaa[1] := D0; The undefined label is described in the array format.	Correct the description of the variable.

No.	Error Message	Cause	Corrective Action
30	Member '*1' of '*2' is undefined. (C8024) (Structure element name or FB variable name enters *1, and structure name or FB name enters *2.)	The element name of the structure is wrong or the variable name of the FB is wrong. Example 1: Structure element name: mem1 Diverted structure name: InsSDT1 InsSDT1.mem2 := 100; The wrong structure element name is described. Example 2: Input variable: IN1 Diverted FB name: FB1 FB1(IN1 := 10); d0 := FB1.aaa; An undefined FB output variable is described.	Correct the element name of the structure, or correct the variable name of the FB.
31	Member '*1' of '*2' which cannot be used as a FB output is used. (C8025) (FB variable name enters *1, and FB name enters *2.)	The FB variable that cannot be used as FB output was used. Example 1: Internal variable (word type): TEMP1 Diverted FB name: FB1 D100 := FB1.TEMP1; The internal variable is used as FB output.	Use the correct FB variable and describe it as FB output.
32	Variable '*1' (FB: *2) cannot be used other than an argument. (C8026) (FB variable name enters *1, and FB name enters *2.)	The FB variable using method is wrong. Example 1: [FB definition] Input variable: IN1 Output variable: OUT1 Diverted FB name: FB1 X1 := FB1.IN1; The input variable is used as FB output.	Use the correct FB variable in the argument of the FB.
33	It is a undefined structure. (C8027)	The structure name is illegal. Example 1: Structure: None Word type label: W_TMP2 W_TMP2.mem1 := 100; The wrong structure is described.	Correct the structure name.

No.	Error Message	Cause	Corrective Action
34	The variable which cannot substitute a value for the *1 cannot be specified. '*2'(C8028) (Error location enters *1, and function name, ":=" enters *2.)	The variable to which a constant, input variable or other value cannot be assigned is specified in the location where the value is assigned. Example 1: Label (constant type): cnt cnt := D10; Assigned to the label constant. Example 2: ABS_E(TRUE, d0, K10); The constant is described in the output variable of the function. Example 3: FB input variable (word type): IN1 BPLUS_3_M(M0, K1, D0, FB1.IN1); The input variable is specified in the argument where the value is output.	Change the value into the variable to which a value can be assigned.
35	Type mismatch at variable *1 of '*2'. (C8029) (Argument error location enters *1, and function name enters *2.)	The type of the variable does not match. Example 1: Word type array: IntAry1[0..1] M1 := BACOS_MD(TRUE, IntAry1, D1);	Correct the type in the specified error location of the function argument, or correct the variable type.
36	Type mismatch for '*1'. (C8030) (Operator, such as ":=" or "*" enters *1.)	The left member of the variable/device differs in data type from the right member. Example 1: D0 : =TRUE; The bit type is assigned to the word device. Example 2: D1 := D2*M1; The word type and bit type are operated. Example 3: M0 := d1 > M1; The word type and bit type are compared.	Specify the same data type in the left and right members of the variable/device.
37	No overload of '*1' takes *2 parameters. (C8031) (Function name enters *1, and the number of arguments that do not match the definition enters *2.)	The number of arguments for a function call does not match the definition. Example 1: ABS(); The number of arguments described is less than the number of arguments defined. Example 2: d0 := ABS(10, 10); The number of arguments described is greater than the number of arguments defined.	Correct the number of function arguments.

No.	Error Message	Cause	Corrective Action

38	The type of a formula is illegal. (C8032)	The format type does not match in the control syntax.
		Example 1: Double word type: DwLBL
		FOR DwLBL := W1 TO W2 BY W3 DO
		W5 := W6;
		END_FOR;
		The data types of the repeat variable and last value expression/incremental expression do not match.
		Example 2: CASE W1 OF
		1: D0 := 1;
		2147483648 : D0 := 2;
		ELSE
		D0 := 10;
		END_CASE;
		The data types of the integer expression and selection value do not match.
		Example 3: IF W1 THEN
		D100 := 1;
		END_IF;
		The word type is specified for the Boolean expression.
			Correct the format type.

39	Substitution is impossible for a constant variable (inside of FOR syntax). (C8033)	An attempt was made to write to the constant variable. There is the following program example for the above error. Example 1: Constant label : tei FOR tei := W10 TO W20 BY W30 DO R10 := R20; END_FOR;	Write to the constant variable (in FOR syntax) cannot be performed.
40	By FOR syntax, variables other than INT/DINT type are used. (C8034)	The variable of other than the word/double word type is used in the FOR syntax. (For example, when the character string, array or structure variable name is specified for the repeat variable) Example 1: Character string label: Str1, Str2, Str3, Str4 FOR Str1 := Str2 TO Str3 BY Str4 DO D0 := D100; END_FOR; The character string variable name was specified for the repeat variable.	Use the correct type in the FOR syntax.

No.	Error Message	Cause	Corrective Action
41	The keyword '*1' is missing. (C8039) (DO UNTIL OF THEN or DO enters *1.)	"DO" is not described in the FOR syntax. Example 1: FOR D1 := D2 TO D3 BY D4	Describe "DO" in the FOR syntax.
		"UNTIL" is not described in the REPEAT syntax.	Describe "UNTIL" in the REPEAT syntax.
		"OF" is not described in the CASE conditional statement.	Describe "OF" in the CASE conditional statement.
		"THEN" is not described in the IF conditional statement.	Describe "THEN" in the IF conditional statement.
		"THEN" is not described in the ELSIF conditional statement.	Describe "THEN" in the ELSIF conditional statement.
		"DO" is not described in the WHILE syntax.	Describe "DO" in the WHILE syntax.
42	Illegal parameter for call of *1. (C8040) (FB name enters *1.)	The description of the input, output or I/O variable of the called FB is illegal. Example 1: Diverted FB name: FB1 FB1(X10); Example 2: Input variable: IN1 Diverted FB name: FB1 FB1(FB1.IN1);	Correct the call description of the FB.
43	The variable which stores the return value of a function is not specified. (C8041)	The return value or the variable that will store the return value does not exist in the function that has no EN/EN0. There is the following example where the variable that will store the return value. Example 1: INT_TO_DINT(D0);	Describe the return value of the function.

No.	Error Message	Cause	Corrective Action

44	There are many nesting and the conditions of control syntax, or between control syntax is too long. (C9017)	There are too many nesting levels or conditions in the control syntax, or the program of the control syntax is too long.
		Example 1: IF D0 = 0 THEN
		IF D1 = 0 THEN
		• • •
		END_IF;
		END_IF;
		Nesting was performed to 598 or more levels in the IF statement.
		Example 2: FOR D0 := 0 TO 100 BY 1 DO
		FOR D1 := 0 TO 100 BY 1 DO
		• • •
		END_FOR;
		END_FOR;
		Nesting was performed to 299 or more levels in the FOR statement.
		Example 3: WHILE D0 < 10 DO
		WHILE D1 < 10 DO
		• • •
		END_WHILE;
		END_WHILE;
		The WHILE statement was nested to 598 or more levels.
		Example 4: CASE W0 OF
		0: D0 := 0;
		1: D0 := 1;
		• • •
		1491: D0 := 1491;
		END_CASE;
		1492 or more integer selection values were used in the CASE statement.
			The program of the control syntax is too long. Shorten the control syntax program, e.g. reduce the number of nesting level or reduce the number of conditions.

45	The value of the execution conditions EN of function '*1' is not right. (C9019) (Function name enters *1.)	In a specific function, TRUE must always be entered into execution condition EN but FALSE has been entered. Example 1: EI_M(FALSE); Example 2: DI_M(0); Example 3: COM_M (FALSE);	Specify the correct value in execution condition EN.
46	Failed to read a system file. (C9020)	The system file is corrupted.	Reinstall.
47	Since it is used by the system, Z0 and Z1 cannot be used. (C9035)	Z0 or Z1 is used. Example 1: INC_M(M10, D0Z1); Example 2: Z0 := 10;	Make correction so that Z0 or Z1 is not used.
48	Constant %d is outside the range of an element number (%d .. %d). (C9039) (Element number enters *1, and the numbers of elements enter *2 and *3.)	The element number of the array is illegal. Example 1: Word type array label: IntAry1[255] IntAry1[K255] := 0;	Correct the specified constant to within the element number range.

No.	Error Message	Cause	Corrective Action
49	Division by zero. (C9065)	0 is used as the divisor. Example 1: D0 := 10/0; Example 2: D1 := W1/K0;	Correct the portion where 0 is used as the divisor.
50	The return value of function '*1' cannot carry out direct reference. (C9066) (Function name enters *1.)	When operation could not be performed by directly referring to the return value of the character string function (indicates the ***_STR(), LEFT(), RIGHT() function). Example 1: M0 := INT_TO_STR(D0) < "AAA";	Change the character string function that resulted in error into the other program, and correct the program to use the return value of that character string function.
51	Failed to read a system file. (C9072)	The system file is corrupted.	The system file is corrupted. Reinstall.
52	The error occurred at the conversion of function '*1'. (C9076) (Function name enters *1.)	The conversion result has an error. Example 1: TIMER_H_M(X0, TC0, -1); A negative value is used in the third argument.	In the argument of the function, use the specifiable data type or the data within the specifiable range.
53	The formula is used for the input variable. (C9118)	An operation expression or function was specified in the input variable that specifies the head device of the MELSEC function. The bit type array element whose element number is variable was specified in the specified input variable of the MELSEC function. The array element of other than bit type whose element number is variable was specified in the specified input variable of the MELSEC function. Example 1: BMOV_M(X0, MAX (D0, D1, D2), D100,D200); The function was used in the input variable. Example 2: TO_M(X0, D0+1, D1, D2, D3); The operation expression was used in the input variable. Example 3: DTO_M(X0, Dint1+K8X0, D1, D2, D3); The operation expression was used in the input variable. Example 4: BKRST_M(X0, ARY[D0], D1); The bit type array element whose element number is variable was specified in the input variable S1 that specifies the head device. Example 5: BKPLUS_M(M0, ARY[D1], ARY[D2], ARY[D3], ARY[D4]); The array elements whose element number is variable were specified in the input variables S1, S2 that specify the head device.	• When the operation expression or function was specified An operation expression or function cannot be specified in the input variable that specifies the head device. Specify a label name or device. • When the bit type array element whose element number is variable was specified The bit type array element whose element number is variable cannot be specified in the argument that specifies the head of the device. Change the element number into a constant, or specify the label name or bit device. • When the array element other than bit type whose element number is variable was specified If the array element whose element number is variable is specified, there is a limit on the index registers used in the compiler. Therefore, make correction, e.g. change the element number into a constant, specify the label name or bit device, or reduce the number of specified array elements used in a single function.

No.	Error Message	Cause	Corrective Action
54	An error is in a conversion result. (F0028)"*1" (Illegal conversion result is displayed in *1.)	The program is correct in ST grammar but an error occurs due to device specifications, etc. Example 1: TS0 := TRUE;	Check the contents of the list displayed in the error message, and correct the program.
55	The number of the maximum which can use a character is to 32 characters. (F0102)	The number of characters used is greater than the preset maximum value. There is the following error example that uses characters greater than the maximum value. Example 1: Character string label: Str1 Str1 := "123456789012345 678901234567890 123"; When the number of character string characters is 33	Change the character string to within 32 characters.
56	The illegal device or value beyond the range is used. (F0137) "*1" (Illegal conversion result is displayed in *1.)	An illegal device or a numeric value outside the range is used. Example 1: M0 := COUNTER_M(TRUE, CC2, -1);	Correct the device or the numeric value to within the range.
57	Devices other than a timer are used for the argument of TIMER_M. (F0177)	The device other than the timer is used in the argument of TIMER_M. Example 1: TIMER_M(X0, CC0, 2);	Use the timer device in the argument of function TIMER_M.
58	Devices other than a counter are used for the argument of COUNTER_M. (F0178)	The device other than the counter is used in the argument of COUNTER_M. Example 1: COUNTER_M(X0, TC0, 2);	Use the counter device in the argument of function COUNTER_M.

No.	Error Message	Cause	Corrective Action
59	With the CONCAT(_E) function, the argument and the same variable as a return value are used. (F0196)	The same variable is used in the argument and return value of the CONCAT(_E) function. Example 1: Character string label: Str1 • Str2 Str1 := CONCAT(Str2, Str1);	Use different variables in the argument and return value of the CONCAT(_E) function.
60	With the INSERT(_E) function, the argument and the same variable as a return value are used. (F0206)	The same variable is used in the argument and return value of the INSERT(_E) function. Example 1: Character string label: Str1 Str1 := INSERT (Str1, Str2, D0); The same variable is used in the argument and return value.	Use different variables in the argument and return value of the INSERT(_E) function.
61	An illegal device type is used. (C10000)	The illegal device type (timer, retentive timer, counter, pointer) is used. Example 1: Timer1 := 0; The device type timer was used.	The illegal device type (timer, retentive timer, counter, pointer) cannot be used. Change it into the applicable device type.
62	The device and numerical value which were specified can be over it, or cannot use the range. (C10001)	The device is greater than the applied range, the unusable device is specified, or the numeric value is greater than the applied range. Example 1: M0 := X2000; The device number greater than 1FFF was specified as the device number of X. Example 2: D0 := A0; The accumulator was used with the QCPU/LCPU Example 3: Double word type label: DW1 DW1 := K2147483648;	Correct the device number to within the applied range. Alternatively, change the device into the usable one, or correct the numeric value to within the applied range.

No.	Error Message	Cause	Corrective Action
63	There is too much function or number of a operator. (C10002)	A total of 1025 or more functions or operators are used in a single statement. Example 1: D0 := 1+1+1+1+ • • • +1+1; 1025 or more operators "+" were used in a single statement. Example 2: d0 := ABS(ABS(ABS(ABS(wlabel) • • • •)); 1025 or more functions ABS were used in a single statement.	When functions or operators are used in a single statement, make correction to use less than 1025 functions or operators.
64	There is too much nesting of array element specification. (C10003)	When nesting was performed to 17 or more levels in the array element specification. Example 1: Array1[Array1[Array1[Array1 [Array1[Array1[. . . . .]]]]]; Nesting was performed to 17 or more levels in the array element specification.	Correct the nesting of the array element specification to up to five levels. Six or more levels are not supported. 17 or more levels will result in an error.
65	Specified FB name is already used. (C10004)	An FB call is made two or more times under the same FB name in the program. Example 1: Diverted FB name: FB1 I/O variable name: INOUT1 FB1(INOUT1 := D100); FB1(INOUT1 := D101);	Make an FB call under the same FB name only once.
66	An output variable is used before the call of FB. (C10005)	FB output is provided before an FB call. Example 1: Diverted FB name: FB1 I/O variable name: INOUT1 Output variable: OUT1 D0 := FB1.OUT1; FB1(INOUT1 := D100);	Put the FB output after the FB call.
67	The illegal type is used by '*1'. (C10006) (Operator enters *1.)	The value greater than the data type range was used in the double word or real number type assignment statement or operation. Example 1: Double word type label: w_Dword w_Dword := -2147483649;	Specify the correct range.

7 - 16

No.	Error Message	Cause	Corrective Action
68	The illegal type is used at the function '*1'. (C10007) (Function name enters *1.)	The illegal data type was used for the argument of the MELSEC function. Example 1: RST_M(M0, ddev1); The double word type was specified in the second argument of function RST_M. Example 2: DECO_M(M0, Real1, K8, Real2); The real number type was specified in the second/fourth argument of function DECO_M. Example 3: COMRD_S_MD(M0, ddev1, Str32); The double word type was specified in the second argument of function COMRD_SD_MD.	Use the variable of correct data type in the argument.

7 - 17

APPENDICES

Appendix 1 Character Strings that cannot be Used as Labels and FB Names

This section indicates the character strings that cannot be used as label and FB names during ST programming.
The character strings used in the device names, instruction names or function names cannot be used as labels and FB names.
If any of the character strings indicated in the following table is used, an error will occur at the execution of entry or compile.

	Character strings that cannot be used as labels and FB names
A	A, ACALL, ACJ, ACTION, ANB, ANY, ANY_BIT, ANY_DATE, ANY_DERIVED, ANY_ELEMENTARY, ANY_INT, ANY_MAGNITUDE, ANY_NUM, ANY_REAL, ANY_SIMPLE, ANY_STRING, ARRAY, AT
B	B, BEND, BL, BLOCK, BOOL, BOOL_TO_BYTE (DINT, DWORD, INT, REAL, SINT, UDINT, UINT, USINT, WORD), BY, BYTE, BYTE_TO_BOOL (DINT, DWORD, INT, REAL, SINT, STRING, UDINT, UINT, USINT, WORD), BWORKR, BWORKRP, BWORKW, BWORKWP, B_BCD_TO_DINT (INT, SINT)
C	C, CASE, CAL, CALC, CALCN, CONFIGURATION, CONSTANT, CTD, CTU, CTUD
D	D, DATE, DATE_AND_TIME, DINT, DINT_TO_BCD (BOOL, BYTE, DWORD, INT, REAL, SINT, STRING, TIME, UDINT, UINT, USINT, WORD), DO, DT, DWORD, DWORD_TO_BOOL (BYTE, DINT, INT, REAL, SINT, STRING, UDINT, UINT, USINT, WORD), DX, DY, D_BCD_TO_DINT (INT, SINT)
E	E, ELSE, ELSIF, EN, END, END_ACTION, END_CASE, END_FOR, END_FUNCTION, END_PROGRAM, END_IF, END_REPEAT, END_RESOURCE, END_STEP, END_STRUCT, END_TRANSITION, END_TYPE, END_VAR, END_WHILE, ENQ, EQ, EQ_STRING,EXIT
F	F, FALSE, FD, FOR, FROM, FUNCTION, FUNCTION_BLOCK, FX, FY, F_EDGE, F_TRIG
G	G, GE, GE_STRING, GT, GT_STRING
H	H
I	I, IF, INITIAL_STEP, INT, INT_TO_BOOL (BYTE, DINT, DWORD, REAL, SINT, STRING, UDINT, UINT, USINT, WORD)
J	J, JMPC, JMPCN
K	K
L	L, LDN, LE, LE_STRING, LIMIT_STRING, LINT, LREAL, LT, LT_STRING, LWORD
M	M, MAX_STRING, MIN_STRING, MOD, MPP, MPS, MRD
N	N, NE, NE_STRING, NOP, NOT
O	OF, ON, ORB, ORN
P	P, PROGRAM
Q	Q
R	R, R1, RCALL, RCJ, READ, READ_ONLY, READ_WRITE, REAL, REAL_TO_BOOL (BYTE, DINT, DWORD, INT, SINT, STRING, UDINT, UINT, USINT, WORD), RECV, REPEAT, RESOURCE, RETAIN, RETC, RETCN, RETURN, REQ, RS, R_EDGE, R_TRIG
S	S, SB, SD, SEND, SEL_STRING, SFCP, SFCPEND, SG, SINT, SINT_TO_BOOL (BYTE, DINT, DWORD, INT, REAL, STRING, UDINT, UINT, USINT, WORD), SM, SR, SREAD, ST, STEP, STEPC, STEPD, STEPG, STEPI, STEPID, STEPIR, S TEPISC, STEPISE, STEPIST, STEPN, STEPR, STEPSC, STEPSE, STEPST, STN, STRING, STRING_TO_BYTE (DINT, DWORD, INT, REAL, SINT, TIME, UDINT, UINT, USINT, WORD), STRUCT, SW, SWRITE, SZ

App - 1

	Character strings that cannot be used as labels and FB names
T	T, TASK, THEN, TIME, TIME_OF_DAY, TIME_TO_STRING, TO, TOD, TOF, TON, TP, TR, TRAN, TRANA, TRANC, TRANCA, TRANCO, TRANCOC, TRANCOCJ, TRUNC_DINT (INT, SINT), TRANJ, TRANL, TRANO, TRANOA, TRANOC, TRANOCA, TRANOCJ, TRANOJ, TRANSITION, TRUE, TYPE
U	U, UDINT, UDINT_TO_BOOL (BYTE, DINT, DWORD, INT, REAL, SINT, STRING, UINT, UNTIL, USINT, WORD), UINT, UINT_TO_BOOL (BYTE, DINT, DWORD, INT, REAL, SINT, STRING, UDINT, USINT, WORD), ULINT, UNTIL, USINT, USINT_TO_BOOL (BYTE, DINT, DWORD, INT, REAL, SINT, STRING, UDINT, UINT, WORD)
V	V, VAR, VAR_CONSTANT, VAR_EXT, VAR_EXTERNAL, VAR_EXTERNAL_FB, VAR_EXTERNAL_PG, VAR_GLOBAL, VAR_GLOBAL_FB, VAR_GLOBAL_PG, VAR_IN_OUT, VAR_INPUT, VAR_OUTPUT, VAR_TEMP, VD, VOID
W	W, WHILE, WITH, WORD, WORD_TO_BOOL (BYTE, DINT, DWORD, INT, REAL, SINT, STRING, UDINT, UINT, USINT), WORKR, WORKRP, WORKW, WORKWP, WRITE, WSTRING, W_BCD_TO_DINT (INT, SINT)
X	X, XOR, XORN
Y	Y
Z	Z, ZNRF, ZR

Precaution on label name

1. A space cannot be used.
2. A numeral cannot be used as the first character.
3. The following characters cannot be used.
   (,), *, /, +, -, <, >, =, &,
   !, ”, #, $, %, ’, ~, ^, |, @, ` , [,], {, }, ;, :, , , ., ?, \, _
   An error occurs if an underscore exists at the end of a character string or two or more underscores are used consecutively.
4. Device names cannot be used.
   An error occurs if any of 0 to F is appended after a device name.
   Examples: XFFF, M100
5. Do not use "EnDm" as a label name (Example: E001D9).
   (n and m are any values.)
   It may be recognized as a real number value and unavailable as a label name.
6. Instruction names (sequence instructions, basic instructions, application instructions) and function names (MELSEC functions, IEC functions) cannot be used.

App - 2

Appendix 2 ST instruction table for GX Developer and GX Works2

Instructions that can be used in ST programs of GX Developer may not be able to be used in GX Works2. As a result, an error may occur when a project that includes an ST program and is saved in GX Works2 format is read and compiled with GX Developer. In such case, correct the ST program in accordance with the following table.

GX Works2	GX Developer
BACOS	BACOS_MD
BAND	BAND_MD
BASIN	BASIN_MD
BATAN	BATAN_MD
BCD	BCD_M
BCOS	BCOS_MD
BDSQR	BDSQR_MD
BIN	BIN_M
BKAND	BKAND_M
BKBCD	BKBCD_M
BKBIN	BKBIN_M
BKOR	BKOR_M
BKRST	BKRST_M
BKXNR	BKXNR_M
BKXOR	BKXOR_M
BMOV	BMOV_M
BRST	BRST_M
BSET	BSET_M
BSFL	BSFL_M
BSFR	BSFR_M
BSIN	BSIN_MD
BSQR	BSQR_MD
BTAN	BTAN_MD
BTOW	BTOW_MD
BXCH	BXCH_M
CML	CML_M
COM	COM_M
DATERD	DATERD_MD
DATEWR	DATEWR_MD
DBAND	DBAND_MD
DBCD	DBCD_M
DBIN	DBIN_M
DBL	DBL_M
DCML	DCML_M
DDEC	DDEC_M

GX Works2	GX Developer
DEC	DEC_M
DECO	DECO_M
DELTA	DELTA_M
DFLT	DFLT_M
DFRO	DFRO_M
DGRY	DGRY_M
DI	DI_M
DINC	DINC_M
DIS	DIS_M
DLIMIT	DLIMIT_MD
DMAX	DMAX_M
DMIN	DMIN_M
DNEG	DNEG_M
DOR	DOR_M
DRCL	DRCL_M
DRCR	DRCR_M
DROL	DROL_M
DROR	DROR_M
DSER	DSER_M
DSFL	DSFL_M
DSFR	DSFR_M
DSORT	DSORT_M
DSUM	DSUM_M
DTEST	DTEST_MD
DTO	DTO_M
DWSUM	DWSUM_M
DXCH	DXCH_M
DXNR	DXNR_M
DXOR	DXOR_M
DZONE	DZONE_MD
EI	EI_M
EMOD	EMOD_M
ENCO	ENCO_M
ENEG	ENEG_M
EREXP	EREXP_M

GX Works2	GX Developer
ESTR	ESTR_M
EVAL	EVAL_M
FLT	FLT_M
FMOV	FMOV_M
FROM	FROM_M
GBIN	GBIN_M
GRY	GRY_M
HOUR	HOUR_M
INC	INC_M
MIDR	MIDR_M
NDIS	NDIS_M
NEG	NEG_M
NUNI	NUNI_M
OUT	OUT_M
PLOW	PLOW_M
POFF	POFF_M
PSCAN	PSCAN_M
PSTOP	PSTOP_M
QCDSET	QCDSET_M
QDRSET	QDRSET_M
RCL	RCL_M
RCR	RCR_M
RFS	RFS_M
RND	RND_M
RSET	RSET_MD
RST	RST_M
SECOND	SECOND_M
SEG	SEG_M
SER	SER_M
SET	SET_M
SFL	SFL_M
SFR	SFR_M
SFT	SFT_M
SORT	SORT_M
SRND	SRND_M

(Next page)

App - 3

GX Works2	GX Developer
STOP	STOP_M
SUM	SUM_M
SWAP	SWAP_MD
TEST	TEST_MD
UNI	UNI_M

GX Works2	GX Developer
WAND	WAND_M
WDT	WDT_M
WOR	WOR_M
WSUM	WSUM_M
WTOB	WTOB_MD

GX Works2	GX Developer
WXNR	WXNR_M
WXOR	WXOR_M
XCH	XCH_M
ZONE	ZONE_MD

App - 4

INDEX

Ind

Index - 1

Index - 2

Index - 3

Index - 4

Index - 5

Index - 6

Index - 7

Index - 8

Index - 9

Index - 10

WARRANTY

Please confirm the following product warranty details before using this product.

1. Gratis Warranty Term and Gratis Warranty Range

If any faults or defects (hereinafter "Failure") found to be the responsibility of Mitsubishi occurs during use of the product within the gratis warranty term, the product shall be repaired at no cost via the sales representative or Mitsubishi Service Company.
However, if repairs are required onsite at domestic or overseas location, expenses to send an engineer will be solely at the customer's discretion. Mitsubishi shall not be held responsible for any re-commissioning, maintenance, or testing on-site that involves replacement of the failed module.

[Gratis Warranty Term]

The gratis warranty term of the product shall be for one year after the date of purchase or delivery to a designated place.
Note that after manufacture and shipment from Mitsubishi, the maximum distribution period shall be six (6) months, and the longest gratis warranty term after manufacturing shall be eighteen (18) months. The gratis warranty term of repair parts shall not exceed the gratis warranty term before repairs.

[Gratis Warranty Range]

(1) The range shall be limited to normal use within the usage state, usage methods and usage environment, etc., which follow the conditions and precautions, etc., given in the instruction manual, user's manual and caution labels on the product.
(2) Even within the gratis warranty term, repairs shall be charged for in the following cases.
1. Failure occurring from inappropriate storage or handling, carelessness or negligence by the user. Failure caused by the user's hardware or software design.
2. Failure caused by unapproved modifications, etc., to the product by the user.
3. When the Mitsubishi product is assembled into a user's device, Failure that could have been avoided if functions or structures, judged as necessary in the legal safety measures the user's device is subject to or as necessary by industry standards, had been provided.
4. Failure that could have been avoided if consumable parts (battery, backlight, fuse, etc.) designated in the instruction manual had been correctly serviced or replaced.
5. Failure caused by external irresistible forces such as fires or abnormal voltages, and Failure caused by force majeure such as earthquakes, lightning, wind and water damage.
6. Failure caused by reasons unpredictable by scientific technology standards at time of shipment from Mitsubishi.
7. Any other failure found not to be the responsibility of Mitsubishi or that admitted not to be so by the user.

2. Onerous repair term after discontinuation of production

(1) Mitsubishi shall accept onerous product repairs for seven (7) years after production of the product is discontinued. Discontinuation of production shall be notified with Mitsubishi Technical Bulletins, etc.
(2) Product supply (including repair parts) is not available after production is discontinued.

3. Overseas service

Overseas, repairs shall be accepted by Mitsubishi's local overseas FA Center. Note that the repair conditions at each FA Center may differ.

4. Exclusion of loss in opportunity and secondary loss from warranty liability

Regardless of the gratis warranty term, Mitsubishi shall not be liable for compensation of 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 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.

MELSEC-Q/L
Programming Manual
Structured Text

MODEL	QCPU-P-ST-E
MODEL CODE	13JF68
SH(NA)-080366E-H(1001)MEE	

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