The right choice for the ultimate yield!

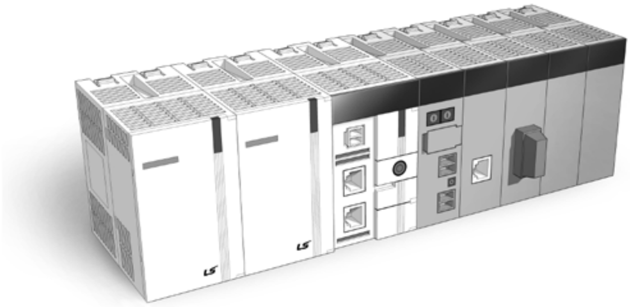
LS ELECTRIC strives to maximize your profits in gratitude for choosing us as your partner.

Programmable Logic Control

XGI/XGR/XEC/XMC Instructions and Programming

XGT Series

User's Maunal



Safety Instructions

- Read this manual carefully before installing, wiring, operating, servicing or inspecting this equipment.
- Keep this manual within easy reach for quick reference.

LS ELECTRIC

Before using the product …

For your safety and effective operation, please read the safety instructions thoroughly before using the product.

► Safety Instructions should always be observed in order to prevent accident or risk with the safe and proper use the product.

► Instructions are separated into “Warning” and “Caution”, and the meaning of the terms is as follows;

⚠ **Warning**	This symbol indicates the possibility of serious injury or death if some applicable instruction is violated
⚠ **Caution**	This symbol indicates the possibility of slight injury or damage to products if some applicable instruction is violated

► The marks displayed on the product and in the user’s manual have the following meanings.

⚠ Be careful! Danger may be expected.

⚠ Be careful! Electric shock may occur.

► The user’s manual even after read shall be kept available and accessible to any user of the product.

Safety Instructions when designing

⚠Warning

- **Please, install protection circuit on the exterior of PLC to protect the whole control system from any error in external power or PLC module.** Any abnormal output or operation may cause serious problem in safety of the whole system.
  - Install applicable protection unit on the exterior of PLC to protect the system from physical damage such as emergent stop switch, protection circuit, the upper/lowest limit switch, forward/reverse operation interlock circuit, etc.
  - If any system error (watch-dog timer error, module installation error, etc.) is detected during CPU operation in PLC, the whole output is designed to be turned off and stopped for system safety. However, in case CPU error if caused on output device itself such as relay or TR can not be detected, the output may be kept on, which may cause serious problems. Thus, you are recommended to install an addition circuit to monitor the output status.

- **Never connect the overload than rated to the output module nor allow the output circuit to have a short circuit**, which may cause a fire.

- **Never let the external power of the output circuit be designed to be On earlier than PLC power**, which may cause abnormal output or operation.

- **In case of data exchange between computer or other external equipment and PLC through communication or any operation of PLC (e.g. operation mode change), please install interlock in the sequence program to protect the system from any error**. If not, it may cause abnormal output or operation.

Safety Instructions when designing

⚠ Caution

- **I/O signal or communication line shall be wired at least 100mm away from a high-voltage cable or power line.** If not, it may cause abnormal output or operation.

Safety Instructions when designing

⚠ Caution

- **Use PLC only in the environment specified in PLC manual or general standard of data sheet.** If not, electric shock, fire, abnormal operation of the product or flames may be caused.

- **Before installing the module, be sure PLC power is off.** If not, electric shock or damage on the product may be caused.

- **Be sure that each module of PLC is correctly secured.** If the product is installed loosely or incorrectly, abnormal operation, error or dropping may be caused.

- **Be sure that I/O or extension connecter is correctly secured.** If not, electric shock, fire or abnormal operation may be caused.

- **If lots of vibration is expected in the installation environment, don’t let PLC directly vibrated.** Electric shock, fire or abnormal operation may be caused.

- **Don’t let any metallic foreign materials inside the product**, which may cause electric shock, fire or abnormal operation.

Safety Instructions when wiring

⚠Warning

- **Prior to wiring, be sure that power of PLC and external power is turned off.** If not, electric shock or damage on the product may be caused.

- **Before PLC system is powered on, be sure that all the covers of the terminal are securely closed.** If not, electric shock may be caused

⚠Caution

- **Let the wiring installed correctly after checking the voltage rated of each product and the arrangement of terminals.** If not, fire, electric shock or abnormal operation may be caused.

- **Secure the screws of terminals tightly with specified torque when wiring.** If the screws of terminals get loose, short circuit, fire or abnormal operation may be caused.

- **Surely use the ground wire of Class 3 for FG terminals, which is exclusively used for PLC**. If the terminals not grounded correctly, abnormal operation may be caused.

- **Don't let any foreign materials such as wiring waste inside the module while wiring,** which may cause fire, damage on the product or abnormal operation.

Safety Instructions for test-operation or repair

⚠ Warning

- **Don’t touch the terminal when powered**. Electric shock or abnormal operation may occur.
- **Prior to cleaning or tightening the terminal screws, let all the external power off including PLC power.** If not, electric shock or abnormal operation may occur.
- **Don’t let the battery recharged, disassembled, heated, short or soldered.** Heat, explosion or ignition may cause injuries or fire.

⚠ Caution

- **Don’t remove PCB from the module case nor remodel the module.** Fire, electric shock or abnormal operation may occur.
- **Prior to installing or disassembling the module, let all the external power off including PLC power.** If not, electric shock or abnormal operation may occur.
- **Keep any wireless installations or cell phone at least 30cm away from PLC**. If not, abnormal operation may be caused.

Safety Instructions for waste disposal

⚠ Caution

- **Product or battery waste shall be processed as industrial waste.** The waste may discharge toxic materials or explode itself.

Revision History

Version	Date	Remark	Chapter
V 1.0	’07. 3	First Edition	-
V 1.1	’07. 6	Process Control Library added	Ch13
V 1.2	’07. 12	ST (Structured Text) language added	Ch14
V 2.0	’08. 3	XGR CPU added	Entire
V 2.1	’09. 3	1. XEC added	Entire
		2. Function for XEC added	
		(1) APM_SSSB	11-31
		(2) PIDAT	13-4
		(3) PIDHBD	13-8
V 2.3	’10. 6	1. XPM dedicated instructions added	Ch6.4.11, Ch11.5
			Ch6.4.10~6.4.11
		2. 4 Positioning instructions (VRD, VWR) added	Ch11.4~11.5
		3. Description on ST language modified	Ch14
		4. Example of ST language added	Ch7~Ch11
V 2.4	’10. 9	1. Positioning instructions added or modified	Ch6.4.11, Ch11.5
V 2.5	’12.11	1. Positioning instructions added	Ch6.4.10~6.4.11
			Ch11.4~Ch11.5
V 2.6	’13.06	1. PUTE and GETE instructions added	Ch6.4.8
			Ch11.2
V 2.7	’14.04	1. UDATA instructions added	Ch11
		2. XPM_STC instruction added	Ch11
		3. CPT instruction information added	Ch8

Version	Date	Remark	Chapter
V 2.8	'14.09	1. SCALE instruction information modified - Information about handling max/ min input value 2. ARY_CMP_EQ, ARY_CMP_NE added - Compare elements with 2 array 3. EBWRITE, EBREAD, RSET information modified - Information about R block number	Ch13 Ch8 Ch8, Ch10
V 2.9	'15.10	1. FIFO instruction information modified 2. Safety Function Block added	Ch10 Ch15
V 3.0	'16.7	1. XPM_CRD instruction information modified 2. XPM_PASHING instruction added 3. XPM_SSSD instruction added 4. XPM_SSSPD instruction added 5. P2PRD_OFFSET instruction added 6. P2PWR_OFFSET instruction added	Ch11 Ch11 Ch11 Ch11 Ch11 Ch11
V 3.1	'17.3	1. Ch16. Motion Function Blocks added 2. App5. Flag List(XMC) added	Ch16 Appendix5
V 3.2	'18.2	1. GET_IP, SET_IP function added 2. IL(IEC) programming function added	11-16~11-19 CH17
V 3.3	'18.06	1. XPM_SETOVR, XPM_CAMA instruction added 2. LS_OnOffCam, LS_RotaryKnifeCamGen, LS_CrossSealCamGen instruction added	CH11 CH16
V3.4	'18.09	1. SPA instruction added	CH10
V3.5	'19.05	1. Motion instruction added (1) LS_OnOffCamEx instruction added (2) NC_RetraceMove and other 9 instructions are added (3) File_Open and other 4 instructions are added 2. Motion Flags are added	CH16 Appendix5

Version	Date	Remark	Chapter
V3.6	’20.05	1. LSIS to change its corporate name to LS ELECTRIC	Entire

Thank you for purchasing PLC of LS ELECTRIC Co., Ltd.
Before use, make sure to carefully read and understand the User's Manual about the functions, performances, installation and programming of the product you purchased in order for correct use and importantly, let the end user and maintenance administrator to be provided with the User's Manual.

The User's Manual describes the product. If necessary, you may refer to the following description and order accordingly. In addition, you may connect our website (http://www.lselectric.co.kr/) and download the information as a PDF file.

Relevant User's Manuals

Title	Description
XG5000 User's Manual (for XGK, XGB)	XG5000 software user manual describing online function such as programming, print, monitoring, debugging by using XGK, XGB CPU.
XG5000 User's Manual (for XGI, XGR)	XG5000 software user manual describing online function such as programming, print, monitoring, debugging by using XGI, XGR CPU.
XGK/XGB Instructions & Programming User's Manual	User's manual for programming to explain how to use instructions that are used PLC system with XGK, XGB CPU.
XGI/XGR/XEC Instructions & Programming User's Manual	User's manual for programming to explain how to use instructions that are used PLC system with XGI, XGR, XEC CPU.
XGK CPU User's Manual (XGK-CPUA/E/H/S/U)	XGK-CPUA/CPUE/CPUH/CPUS/CPUU user manual describing about XGK CPU module, power module, base, IO module, specification of extension cable and system configuration, EMC standard.
XGI CPU User's Manual (XGI-CPUU/CPUH/CPUS)	XGI-CPUU/CPUH/CPUS user manual describing about XGI CPU module, power module, base, IO module, specification of extension cable and system configuration, EMC standard.
XGR Redundant Series User's Manual	XGR- CPUH/F, CPUH/T user manual describing about XGR CPU module, power module, extension drive, base, IO module, specification of extension cable and system configuration, EMC standard.
XG-PM User's Manual	XG-PM software user manual describing online function such as motion programing, monitoring, debugging by using Motion Control Module.

Ch 1. Introduction 1-1

Ch 2. The Structure of Software .... 2-1~2-2

Ch 3. Common Elements .... 3-1~3-17

Ch 16. Motion Function Blocks .... 16-1~16-225

제 17 IL Edit ....17-1~17-33

Chapter 1. Introduction

1.1 Overview

1) Background

This user's guide describes the languages supported by XGI /XGR/XEC PLC. The XGI /XGR/XEC PLC is based on the standard language of International Electrotechnical Commission (IEC).

2) Features of IEC 61131-3 Language

The features of the IEC language supported by the PLC are as follows

▷ Supports several data types.

▷ Offers program elements such as functions, function blocks, and programs to enable bottom-up design and top-down design and structural creation of a PLC program.

▷ Program storage in a library system to enable future use in other environments. This enables the reuse of the software.

▷ Supports various languages so that the user can select the optimal language suitable for the environment.

3) Types of Language

The PLC language standardized by IEC consists of two illustrated languages, two character languages and SFC.

▷ Illustrated language

a) Ladder Diagram (LD): It is a graphical language based on the ladder logic.

b) Function Block Diagram (FBD): It is a graphical language for depicting signal and data flows through function blocks.

▷ Character language

a) Instruction List (IL): It is a low-level 'assembly like' language based on similar instruction list languages.

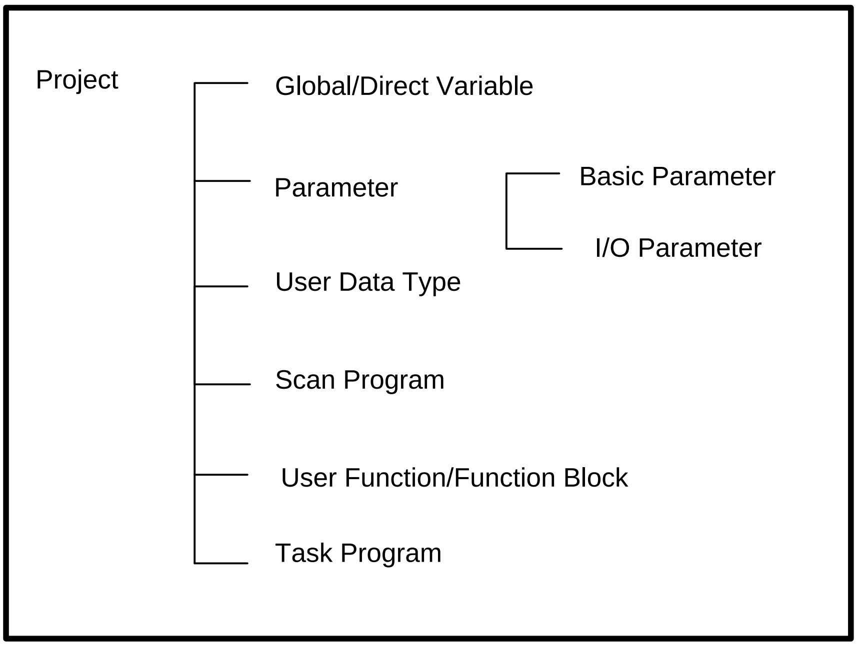
b) Structured Text (ST):It is a high-level PASCAL type language.

▷ Sequential Function Chart (SFC)

Chapter 2. Software Structure

2.1 Introduction

Before creating a PLC program, ensure that you have an overall PLC system defined in software terms. The overall PLC system is defined as one project in XGI /XGR/XEC PLC. In the project, you must define hierarchically all composition elements necessary for the PLC system.



2.2 Project

For a XGI/XGR/XEC PLC program, the first priority is given to project configuration. Creating a project comprises of configuring
and programming all elements necessary for a PLC system (scan programs, task definitions, basic parameters, I/O parameters,
and so on).

1) Global/Direct Variable

The project enables global variable setting, direct variable setting and flag, in which a user prepares or uses the necessary information.

2) Parameter

The user can alter the default CPU parameters and/or configure the IO Modules

▷ Basic Parameter: consists of four parts; setting such as basic operation set up, time and output control, retain area setting
,error operation setting and MODBUS data setting.

▷ I/O Parameter: Used to configure I/O modules.

3) User Data Type

Data type is a classification showing its unique characteristics. For instance, ANY_NUM contains all of LREAL, REAL, LINT, DINT, INT, SINT, ULINT, UDINT, UINT, and **USINT.** For additional information on User Data Type, refer to Common Elements

4) Scan Program

The scan program is a basic method of executing a program repeatedly on a PLC. It sequentially performs the same operations as per the program starting from the first step to the last step. For example, a scan program can read input data at the input module, run a program and display the results to the output module.

5) User Function/Function Block

▷ Function : Is an operation unit that immediately yields the operation results for an input such as four arithmetical operations and comparative operations

▷ Function block : Is an operation unit that memorizes the operation results within the commands such as timer and counter or results derived from several scans. Function blocks are the fundamental element for logic programs. Function blocks like timer and counter have input and output connections to indicate the flow.

6) Task Program

▷ Task program does not repeat scanning unlike a scan program and instead, executes only when its execution condition occurs. If several tasks are waiting, a higher priority task program is processed first. Among tasks of equal priority, the processing happens by the order of occurrence

▷ There are fixed cycle tasks and internal contact tasks.

Chapter 3. Common Elements

3.1 Overview

The elements of XGI/XGR/XEC PLC program (programs, functions, function blocks) can be programmed in other languages such as LD, SFC, and so on. All the language share common grammar elements.

3.2 Expression

3.1.1 Identifiers

▷ Identifiers must be mixed of alphabet, numeric and all letters starting with underlined letters.
▷ Identifiers are used as variable names.
▷ Blank (space) is not allowed in identifiers.
▷ In case of variable or instance name, identifiers may consist of Korean, Alphabet and Chinese characters.
▷ There's no difference between small letters and capitals in alphabet; all the letters of the alphabet are recognized as upper case.

Types	Examples
Capital alphabet and number	IW210, IW215Z, QX75, IDENT
Capital alphabets ,numbers and underline(_)	LIM_SW_2, LIMSW5, ABCD, AB_CD
Capital alphabet and number characters starting with an underline(_)	_MAIN, _12V7, _ABCD

3.1.2 Data Expression

The data in XGI/XGR/XEC PLC is; numeric data type, character string, time data type, and so on.

Types	Examples
Integer	-12, 0, 123_456, +986
Real number	-12.0, 0.0, 0.456, 3.14159_26
Real number with an exponent	-1.34E-12, 1.0E+6, 1.234E6
Binary number	2#1111_1111, 2#11100000
Octal number	8#377(decimal 255) 8#340(decimal 224)
Hexadecimal number	16#FF(decimal 255) 16#E0(decimal 224)
BOOL data	0, 1, TRUE, FALSE

1) Numeric data type

▷ There are integer and real numbers.
▷ Discontinuous underline (_) can be placed between numeric characters; and it doesn't have any meaning.
▷ Decimal complies with general decimal data type expression and if there is a decimal point, they are real numbers.
▷ In case of expressing exponent, you can use plus/minus signs can be used. The letter 'E' standing for the exponent does not distinguish capitals from small letters.

▷ When using real numbers with exponents, the followings are not allowed.
Ex) 12E-5 (×) 12.0E-5 (○)
▷ Integer includes binary, octal, hexadecimal numbers and decimal, which can be distinguished by placing # in front of each numerical character.
▷ 0 ~ 9 and A ~ F are used (including small letters a ~ f) in expressing hexadecimal.
▷ There is no need have plus/minus signs in expressing hexadecimal.
▷ Boolean data may be expressed as an integer 0 or 1.

2) Character String

▷ Character string covers all the letters with single quotation marks.
▷ In case of the character string constant and the initialization, the length is limited up to 31 letters.
Ex) 'CONVEYER'

3) Time data type

Time data types are classified as follow:
▷ Duration data: calculates and controls the elapsed time of a controlling event.
▷ Time of Day and Date data : displays the time of the starting/ending point of a controlling event.

(a) Duration

▷ Duration data starts with the reserved word, 'T#' or 't#'.
▷ Several data types such as date (d), hour (h), minute (m), second (s) and millisecond (ms) must be written in sequence. Duration data can start with any unit (d,h,m,s and ms). In case of millisecond , the minimum unit can be omitted but the medium unit between duration units must not be skipped.
▷ Cannot use the underline (_).
▷ Duration data can overflow at the maximum unit, if any, and the data with a decimal point is available except 'ms'. It does not exceed T#49d17h2m47s295ms (32bits by 'ms' unit)
▷ The data is limited to the third decimal place in the second unit (s).
▷ Decimal point is not available at 'ms' unit.
▷ Capital and small letters are both available.

Content	Examples
Duration (no underline)	T#14ms, T#14.7s, T#14.7m, T#14.7h t#14.7d, t#25h15m, t#5d14h12m18s356ms

(b) Time of day and date

▷ There are three types expressing 'Time of Day and Date' as follows: Date, Time of Day; Date and Time.

Content	Reserved word
Date prefix	D#
Time of Day prefix	TOD#
Date and time prefix	DT#

▷ The data of starting point is January 1, 1984.
▷ There's a limit on 'Time of Day' and 'Date and Time', which is up to the third decimal place in the 'ms' unit.
▷ The overflow is not allowed for all the units when expressing 'Time of Day' and 'Date and Time'.

Content	Examples
Date	D#1984-06-25 d#1984-06-25
Time of Day	TOD#15:36:55.36 tod#15:36:55.369
Date and Time	DT#1984-06-25-15:36:55.36 dt#1984-06-25-15:36:55.369

3.2 Data Type

Data has a data type showing its character.

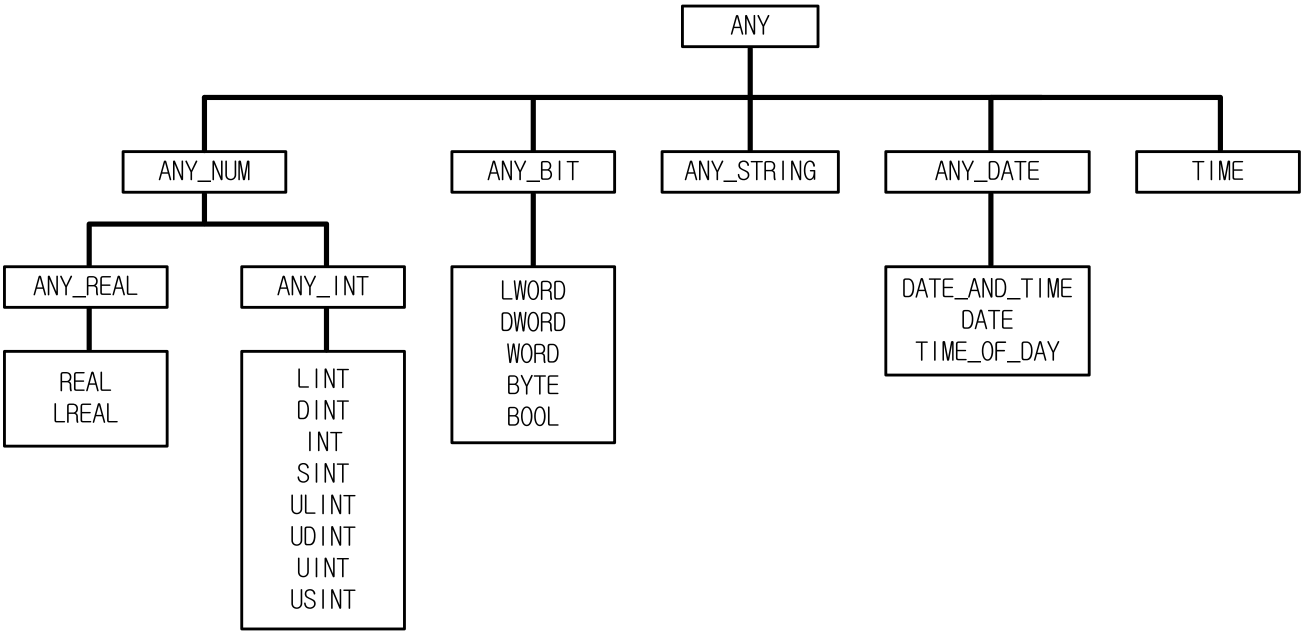
3.2.1 Basic Data Type

XGI/XGR/XEC PLC supports the following basic data types.

No.	Reserved Word	Data Type	Size (bits)	Range
1	SINT	Short Integer	8	-128 ~ 127
2	INT	Integer	16	-32,768 ~ 32,767
3	DINT	Double Integer	32	-2,147,483,648 ~ 2,147,483,647
4	LINT	Long Integer	64	-2^{63} ~ $2^{63}-1$
5	USINT	Unsigned Short Integer	8	0 ~ 255
6	UINT	Unsigned Integer	16	0 ~ 65,535
7	UDINT	Unsigned Double Integer	32	0 ~ 4,294,967,295
8	ULINT	Unsigned Long Integer	64	0 ~ $2^{64}-1$
9	REAL	Real Numbers	32	-3.402823466e+038 ~ -1.175494351e-038 or 0 or 1.175494351e-038 ~ 3.402823466e+038
10	LREAL	Long Real Numbers	64	-1.7976931348623157e+308 ~ -2.2250738585072014e-308 or 0 or 2.2250738585072014e-308 ~ 1.7976931348623157e+308
11	TIME	Duration	32	T#0S ~ T#49D17H2M47S295MS
12	DATE	Date	16	D#1984-01-01 ~ D#2163-6-6
13	TIME_OF_DAY	Time Of Day	32	TOD#00:00:00 ~ TOD#23:59:59.999
14	DATE_AND_TIME	Date and Time of Day	64	DT#1984-01-01-00:00:00 ~ DT#2163-06-06-23:59:59.999
15	STRING	Character String	32*8	-
16	BOOL	Boolean	1	0,1
17	BYTE	Bit String of Length 8	8	16#0 ~ 16#FF
18	WORD	Bit String of Length 16	16	16#0 ~ 16#FFFF
19	DWORD	Bit String of Length 32	32	16#0 ~ 16#FFFFFFFF
20	LWORD	Bit String of Length 64	64	16#0 ~ 16#FFFFFFFFFFFFFFFF

3.2.2 Data Type Hierarchy Chart

Data types used in XGI/XGR/XEC PLC are as follows:



▷ Data expressed as ANY_NUM includes LREAL, REAL, LINT, DINT, INT, SINT, ULINT, UDINT, UINT and USINT.
▷ For example, if a data type is expressed as ANY_BIT, it can use one of the following data types: LWORD, DWORD, WORD, BYTE and BOOL.

3.2.3 Initial Value

If an initial value of data is not assigned, it is automatically assigned as follows.

Data Type	Initial Value
SINT, INT, DINT, LINT	0
USINT, UINT, UDINT, ULINT	0
BOOL, BYTE, WORD, DWORD, LWORD	0
REAL, LREAL	0.0
TIME	T#0s
DATE	D#1984-01-01
TIME_OF_DAY	TOD#00:00:00
DATE_AND_TIME	DT#1984-01-01-00:00:00
STRING	' ' (empty string)

3.2.4 Data Type Structure

Bit String

BOOL

1 bit, range: 0, 1

BYTE (bits 7 … 0)

8 bit , range: 2#0000_0000 ~ 2#1111_1111, 16#00 ~ 16#FF

WORD (bits 15 … 8 7 … 0)

16 bit, range: 2#0000_0000 _0000_0000 ~ 2#1111_1111_1111_1111 16#0000 ~ 16#FFFF

DWORD (bits 31 … 16 15 … 0)

32 bit, range: 2#0000_...000 ~ 2#1111_...111 16#00000000 ~ 16#FFFFFFFF

LWORD (bits 63 … 32 31 … 0)

64 bit, range: 2#0000_...000 ~ 2#1111_...111, 16#0000000000000000 ~ 16#FFFFFFFFFFFFFFFF

Unsigned Integer

USINT (bits 7 … 0)

8 bit , range: 0 ~ 255

UINT (bits 15 … 8 7 … 0)

16 bit, range: 0 ~ 65,535

UDINT (bits 31 … 16 15 … 0)

32 bit, range: 0 ~ 4,294,967,295

ULINT (bits 63 … 32 31 … 0)

64 bit, range: 0 ~ 2^{64}-1

Integer (negative number is expressed as 2's complement)

SINT (bits 7 … 0)

8 bit, range: −128 ~ 127

INT (bits 15 … 8 7 … 0)

16 bit, range: −32,768 ~ 32,767

DINT (bits 31 … 16 15 … 0)

32 bit. range: −2,147,483,648 ~ 2,147,483,647

LINT (bits 63 … 32 31 … 0)

64 bit, range: -2^{63} ~ $2^{63}-1$

Real (based on the IEEE Standard 754-1984)

	31	30 – 23	22 – 0
REAL	S	Exponent	Fraction

32 bit, range: 1.401298E-45 ~ 3.402823E38

	63	62 – 52	51 – 0
LREAL	S	Exponent	Fraction

64 bit, range: 4.9406564E-324 ~ 1.7976931E308

- S: sign (0: positive number; 1: negative number)
- Exponent: exponent of 2(2^{e-127}: $e=b_{30}b_{29}...b_{23}$, $e=b_{62}b_{61}...b_{52}$)
- Fraction: a decimal fraction (Fraction: $f=b_{22}b_{21}...b_{0}$, $f=b_{51}b_{50}...b_{0}$)

Time

	31 – 0
TIME	

32 bit, range: 0 ~ 4,294,967,295ms ,T#49d17h2m47s295ms

Date

	63 – 48	47 – 32	31 – 0
DT	0000000000000000	TOD	DATE

64 bit, range: DT#1984-01-01-00:00:00 ~ DT#2163-12-31-23:59:59.999

	31 – 16	15 – 0
TOD		

32 bit, range: TOD#00:00:00 ~ TOD#23:59:59.999

	15 – 8	7 – 0
DATE		

16 bit, range: D#1984-01-01 ~ D#2163-6-6

#BCD

	7 – 4	3 – 0
BYTE	10^1	10^0

8 bit, range: 0 ~ 99

	15 – 8		7 – 0	
WORD	10^3	10^2	10^1	10^0

16 bit, range: 0 ~ 9999

	31 – 24		23 – 16		15 – 8		7 – 0	
DWORD	10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0

32 bit, range: 0 ~ 99,999,999

	63 – 48				47 – 32				31 – 16				15 – 0			
LWORD	10^{15}	10^{14}	10^{13}	10^{12}	10^{11}	10^{10}	10^9	10^8	10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0

64 bit, range: 0 ~ 9,999,999,999,999,999

3.3 Variable

A variable has its own value and refer to data used in a program. 'Variable' refer to something that can vary such as an input/output of PLC, memory, and so on.

3.3.1 Variable Expression

▷ Variables can be expressed in two ways: by giving a name to a data element using an identifier (Variable by Identifier) or by directly assigning a memory address or an input/output of PLC to a data element (Direct Variable).

▷ A variable by identifier must be unique within its 'effective scope' (program area where the variable was declared) in order to distinguish it from other variables.

▷ A direct variable is expressed as one, which starts with the percent sign (%) followed by the 'location prefix', a prefix of the data size, and more than one unsigned integer numbers divided by a period (.). The prefixes are shown as follows.

Location prefix

No.	Prefix	Meaning
1	I	Input Location
2	Q	Output Location
3	M	Memory Location (M)
4	R	Memory Location (R)
5	W	Memory Location (W)

Size prefix

No.	Prefix	Meaning
1	X	1 bit size
2	None	1 bit size
3	B	1 byte (8 bits) size
4	W	1 word (16 bits) size
5	D	1 double word (32 bits) size
6	L	1 long word (64 bits) size

Expression format

%[Location Prefix][Size Prefix] n1.n2.n3

Number	I, Q	M, R, W
n1	Base number (starting from "0")	The n1th data according to [size prefix] (starting from "0")
n2	Slot number (starting from "0")	The n2th data of the n1th data (starting from "0") : available to omit
n3	n3 data according to the [size prefix] (starting from "0")	Not used

Examples

%QX3.1.4 or %Q3.1.4	4th output of no.1 slot on no.3 base (1 bit)
%IW2.4.1	1st word input of no.4 slot on no.2 base (16bits)
%MD48	48th double word memory
%MW40.3	3rd bit of 40th word memory
	(internal memory does not have a base or a slot number)

▷ Small alphabets are not allowed as a prefix.
▷ A variable without a size prefix is treated as 1 bit.
▷ Direct variables are available to use without a variable declaration.

3.3.2 Variable Declaration

▷ Program elements (programs, functions, function blocks, and so on) have parts that can be declared to edit their variables.
▷ Variables must be declared before using them in the program elements.
▷ The contents of a variable declaration are as follows.

1) Variable types

The variable type defines how to declare variables.

Variable types	Description
VAR	General variable available to read/write
VAR_RETAIN	Retaining(data-keeping) variable
VAR_CONSTANT	Read only variable
VAR_EXTERNAL	Declaration to use the variable declared as VAR_GLOBAL

2) Data type
Data type sets a variable data type.
3) Memory allocation
Memory allocation assigns memory for a variable.
Auto ---- The compiler sets a variable location automatically (Automatic Allocation Variable).
Assign (AT) ---- A user sets a location of variable, using a direct variable (Direct Variable).

Reference

The location of Automatic Allocation Variable is not fixed. If variable VAL1, for example, was declared as BOOL, it is not fixed in the internal memory; the compiler and linker fix its location. If the program is compiled again after modification, the location may change.

The merit of Automatic Allocation Variable is that users do not have to care the location of the internal variables because its location is not overlapped as long as a variable name is different from others.

Use of Direct Variable is not recommended except %Ｉ and %Ｑ because the location of a variable is fixed and it could be overlapped in a wrong-used case.

▷ Initial Value Assignment: assigns an initial value. A variable is set with an initial value as shown in section '3.2.3. Initial Value' if not assigned.

Reference

The initial value is not assigned when it comes to VAR_EXTERNAL.

In case of 'Variable Declaration', you cannot assign an initial value to %Ｉ or %Q variables.

▷ You can declare variable VAR_RETAIN that keeps its data in case of power failure. Rules are:

1) 'Retention Variable' retains its data when the system is set as 'Warm Restart'.

2) In case of 'Cold Restart', variables are initialized as the initial values set by users or the basic initial values.

▷ Variables, which are not declared as VAR_RETAIN, must be initialized as the initial values set by a user or the basic initial values in case of 'Warm Restart' or 'Cold Restart'.

Reference

Variables, which are assigned as %I or %Q, must not to be declared as VAR_RETAIN or VAR_CONSTANT.

▷ Users can declare variables 'Array' with Elementary Data Type. When declaring the Array Variable, users are supposed to set Data Type and Array Size; 'STRING' type among Elementary Data Types is not allowed.

▷ Effective scope of variable declaration, the area which is available to use the variable, is limited to the program where variables are declared. And users can't use variables declared in other program in the above area. On the contrary, users can get an access to 'Global Variable' from other program elements by declaring it as 'VAR_EXTERNAL'.

Examples of Variable Declaration

Variable Name	Variable Kind	Data Type	Initial Value	Memory Allocation
I_VAL	VAR	INT	1234	Auto
BIPOLAR	VAR_RETAIN	REAL	-	Auto
LIMIT_SW	VAR	BOOL	-	%IX1.0.2
GLO_SW	VAR_EXTERNAL	DWORD	-	Auto
READ_BUF	VAR	ARRAY OF INT[10]	-	Auto

3.3.3 Reserved Variable

▷ 'Reserved Variable' refers to variables previously declared in the system. These variables are used for special purposes and users cannot declare variables with the name of the Reserved Variables.

▷ Users can use the reserved variables without variable declaration.

▷ For additional information, refer to Appendix 2 : Flag List(XGI) Summary of Special internal flag(F) and XGI-CPUU User's Manual.

3.3.4 Reserved Word

Reserved words are previously defined words to use in the system. And these reserved words cannot be used as an identifier.

Reserved words
ACTION ... END_ACTION
ARRAY ... OF
AT
CASE ... OF ... ELSE ... END_CASE CONFIGURATION ... END_CONFIGURATION
Name of data type
DATE#, D#DATE_AND_TIME#, DT#
EXIT
FOR ... TO ... BY ... DO ... END_FOR
FUNCTION ... END_FUNCTION
FUNCTION_BLOCK ... END_FUNCTION_BLOCK
Name of function block
IF ... THEN ... ELSIF ... ELSE ... END_IF
OK
Operator (IL language) Operator (ST language)
PROGRAM
PROGRAM ... END_PROGRAM
REPEAT ... UNTIL ... END_REPEAT
RESOURCE ... END_RESOURCE
RETAIN
RETURN
STEP ... END_STEP
STRUCTURE ... END_STRUCTURE
T#
TASK ... WITH
TIME_OF_DAY#, TOD#
TRANSITION ... FROM... TO ... END_TRANSITION

Reserved words
TYPE ... END_TYPE
VAR ... END_VAR VAR_INPUT ... END_VAR VAR_OUTPUT ... END_VAR VAR_IN_OUT ... END_VAR VAR_EXTERNAL ... END_VAR
VAR_ACCESS ... END_VAR
VAR_GLOBAL ... END_VAR
WHILE ... DO ... END_WHILE
WITH

3.4 Program Type

There are three types of program: function, function block and program. You cannot call its own program in the program (recursive call is prohibited)

3.4.1 Function

▷ A function has one output and does not have any data with status in it. That is, to be a function, consistent input must yield consistent output.
▷ An internal variable of a function cannot have an initial value.
▷ You cannot declare a function as VAR_EXTERNAL and use it.
▷ You cannot use direct variables inside the function.
▷ You can call a function program elements and use it.
▷ Data transfer from program composition elements which call the function, to the function, is executed through an input of a function.
▷ You cannot call a function block or a program from inside a function.
▷ A function has a variable whose name is the same as that of the function and whose data type is the same as the data type of the result of the function. This variable is automatically creates when you make a function and the result value of the function displays in the output.

3.4.2 Function Block

▷ A function block can have a several outputs.
▷ A function block has data inside. A function block must declare the instance as it declares variables before using them. Instance is a set of variables used in a function block. A function block must have its data memory to preserve the output value as well as variables used inside, which is called as “instance.” A program is a kind of a function block and also needs to declare “instance.” However, users cannot call a program inside a program or a function block for use, contrary to a function block.
▷ You can declare a direct variable inside a function block, and moreover, you can use a direct variable declared as Global Variable and allocated according to ‘Assign (AT)’ after declaring it as VAR_EXTERNAL.
▷ You can call a program inside the function block.

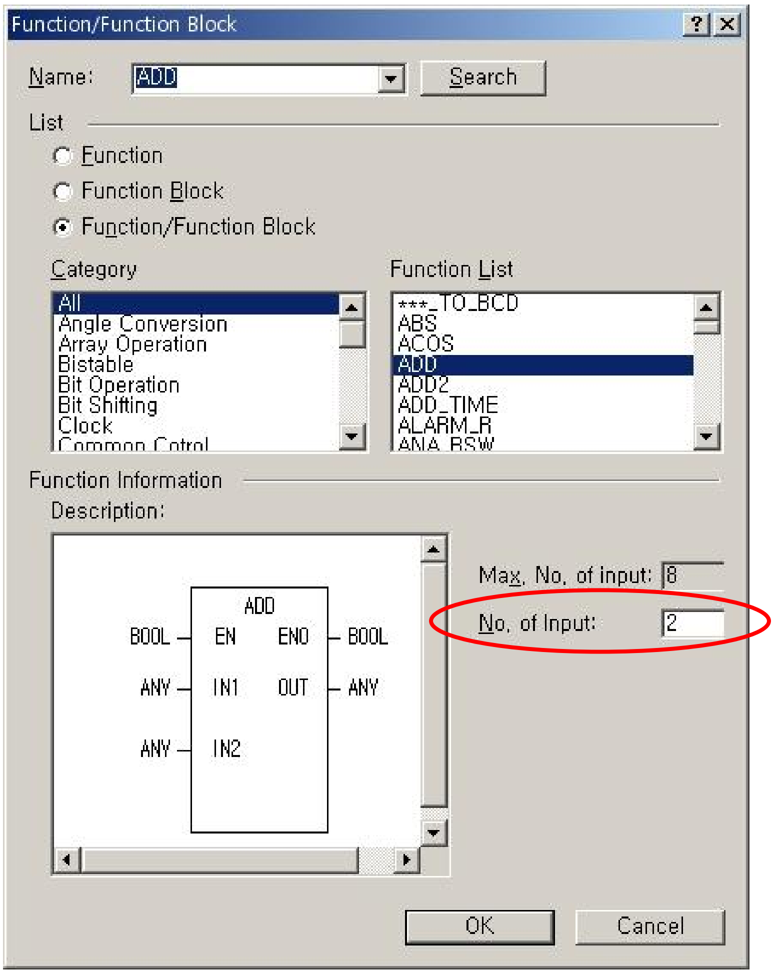
3.4.3 Program

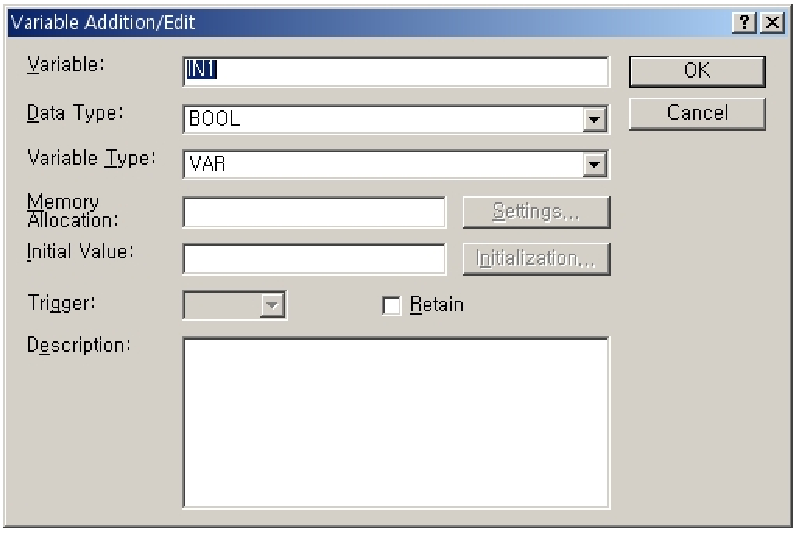
▷ Users can use a program after declaring an instance like a function block.
▷ User can use direct variables in the program.
▷ A program does not have input/output variables.
▷ A program can call functions or function blocks.

3.5 Function Selection

3.5.1 Internally Determined Function

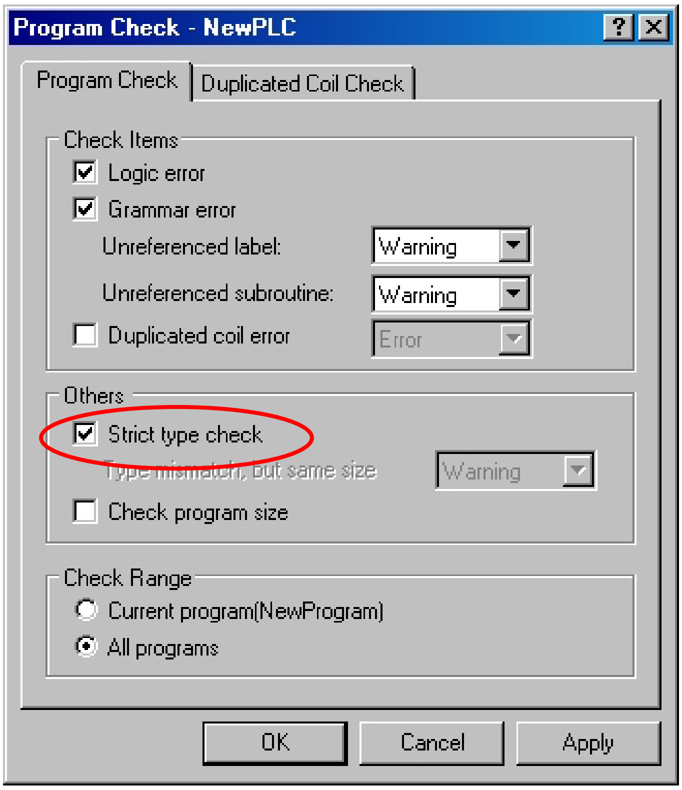
▷ Although a function has one name, a command in which a variety of variable types can be entered is divided into various commands, depending on available variables. For instance, ADD can be divided and processed in various kinds, depending on the number of input defined or I/O variable types. If you select in the following figure, the function shown in a ladder program is ADD but ADD2_SINT function executes internally.





▷ An internally used function automatically selects in XG5000, depending on a user-selected variable type. For instance, two inputs are selected among ADD function and I/O variables are selected as DINT, ADD2_DINT is selected as described above.

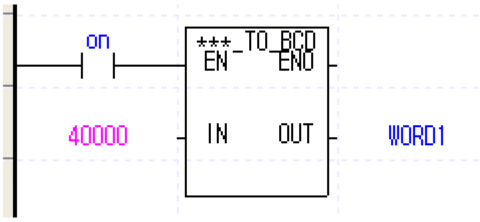
▷ Although IEC allows an operation between and among same types, XG5000 has a “Strict type check” (View→Program Check) option to allow an operation if its operand sizes (BYTE, WORD, DWORD, and LWORD) are same.



3.5.2 Function Selection Rules

▷ If an input variable is of multiple data type, then, an internally used function is used to determine the type of the output variable.

▷ If a constant is used as input in a function in which various input variable types and one output variable type are allowed, a function is determined by a constant.
For instance, ***_TO_BCD is used as below,



A function is determined depending on output variable type because input variable is constant; in this case, the following two functions which output is word are available (INT_TO_BCD_WORD/UINT_TO_BCD_WORD). UINT_TO_BCD_WORD is selected depending on constant type. Positive constant is determined as 'unsigned' while negative one is determined as 'signed'.

Chapter 4. SFC (Sequential Function Chart)

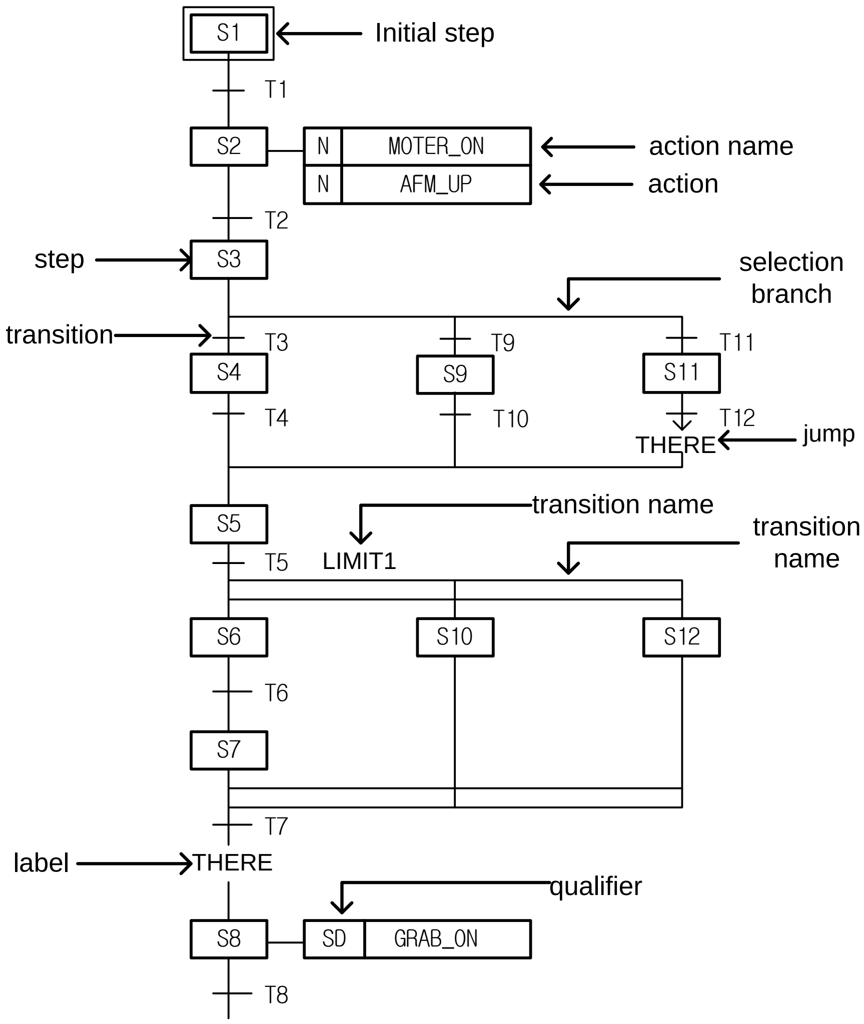
4.1 Introduction

▷ SFC is a structured language that extends an application program in the form of flow chart according to the processing sequence, using a PLC language.

▷ SFC splits an application program into step and transition, and provides how to connect them each other. Each step is related to action and each transition is related to transition condition.

▷ As SFC should contain the state information, only program and function block among program types are available to apply this SFC.

▷ Type



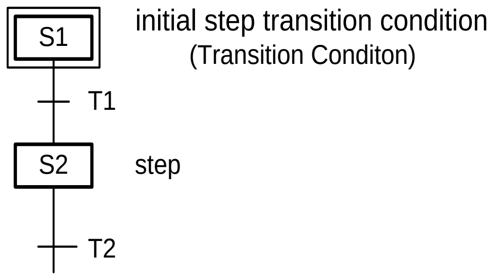
4.2 SFC Structure

4.2.1 Step

▷ Step indicates a sequence control unit by connecting the action.

▷ When step is in an active state, the attached content of action executes.

▷ You have to first activate the initial step.



▷ If a next transition condition of activated initial step (S1) is established, the currently activated step 1 (S1) is inactivated and Step 2 (S2) connected to S1 becomes activated.

4.2.2 Transition

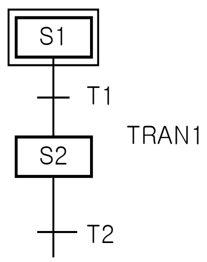
▷ Transition indicates the execution condition between steps.

▷ A transition condition must be described as a PLC language such as ST(Structured text) or LD.

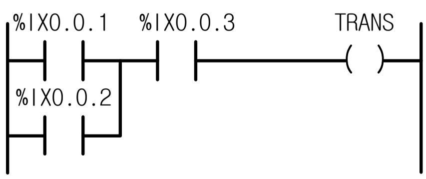
▷ The result of a transition condition must always be a BOOL type and the variable name must be TRANS for any transition.

▷ In case that the result of transition condition is 1, the current step is inactivated and the next step is activated.

▷ There must be a transition between steps.



The content of TRAN1



When TRANS is on, S1 is inactivated and S2 is activated.

TRANS is the internally declared variable.

A transition condition of all transition must be output in TRANS variable.

4.2.3 Action

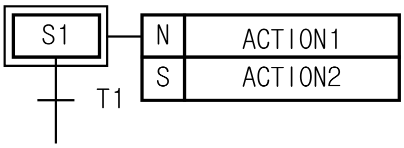
▷ Each step is able to connect up to two actions.

▷ The step without action is regarded as a waiting action and it is required to wait until the next transition condition is 1.

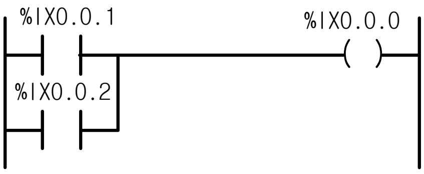
▷ Action is composed of PLC language such as LD/SFC/ST and the action execute while the step is activated.

▷ Action qualifier is used to control action.

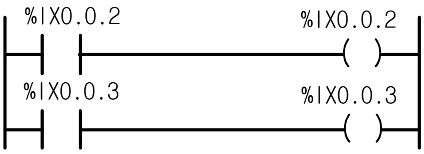
▷ When action becomes inactivated, the state after activating the contact output in action is 0.
However, S, R, function and function block output retain their state prior to inactivation.



The content of ACTION1



The content of ACTION2



- ACTION1 executes only when S1 is activated.
- ACTION2 executes until activated S1 meets R qualifier. It goes on executing even if S1 is inactivated.
- When action is deactivated, this action is Post Scanned and then passes to the next step.

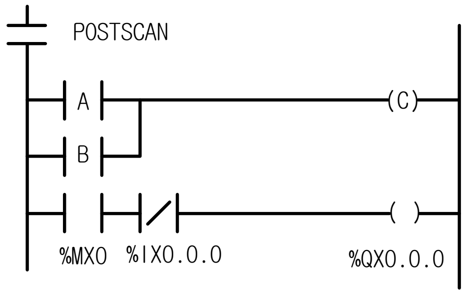
Reference

Post Scan

When action is inactivated, this action is scanned again.

As it is scanned as if there is a contact (contact with the value of 0) in the early part of an action program, the program output, which is composed of contacts, is 0.

Function, function block, S, R output and so on are not included.



In this figure, as the contact of post scan is 0, C and %Q0.0.0 is 0.

4.2.4 Action Qualifier

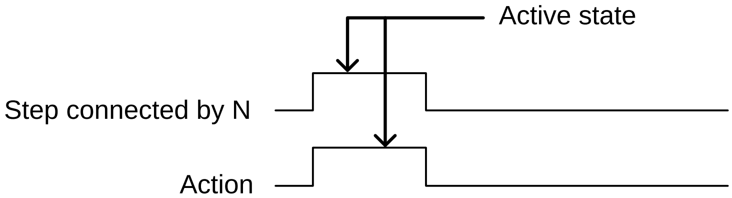
▷ Whenever action is used, action qualifier follows.

▷ The action of step defines an executing point and time according to the assigned qualifier.

▷ Types of action qualifier are as follows.

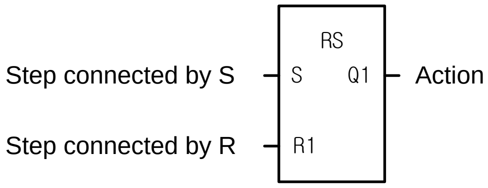
1) N (Non-Stored)

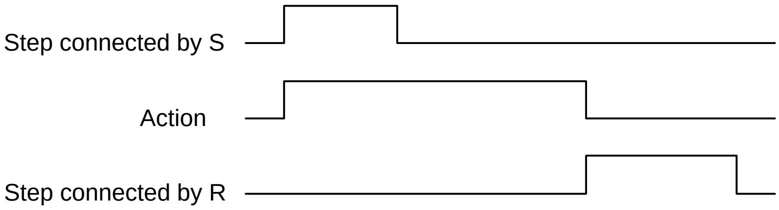
Action executes only when the step is activates.



2) S (Set)

It continues the action after the step is activates (until the action is reset by R qualifier).



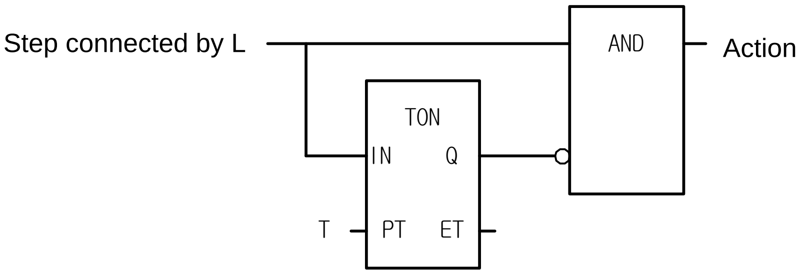


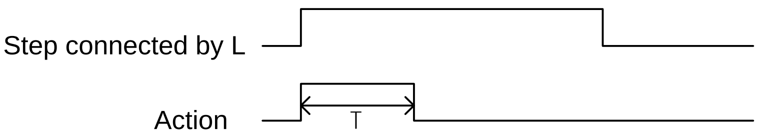
3) R (Overriding Reset)

It terminates the execution of an action previously started with the S, SD, SL or DS qualifier.

4) L (Time Limited)

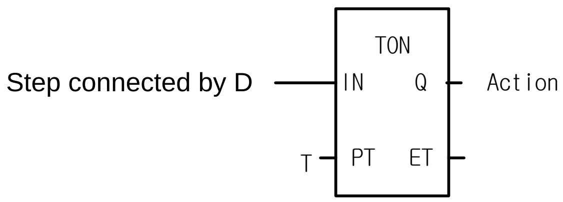
It starts the action when the step becomes active and continues until the step goes inactive or a set time elapses.

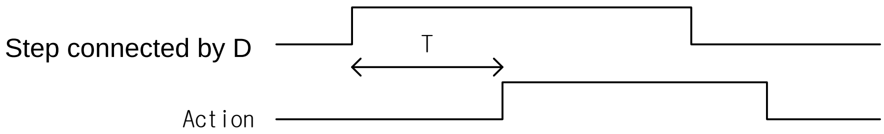




5) D (Time Delayed)

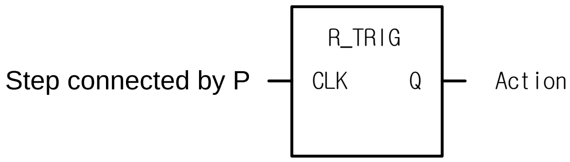
Start a delay timer when the step activates; after the time delay the action starts (if step is still active) and continues until inactivated.

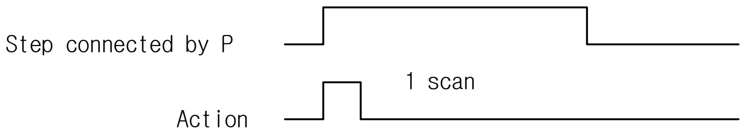




6) P (Pulse)

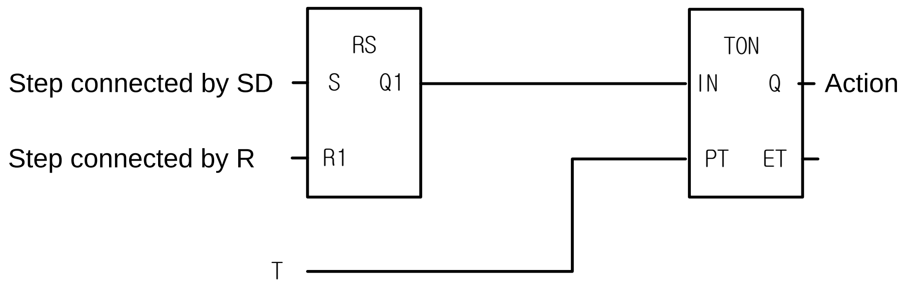
It starts the action when the step is active and executes the action only once.

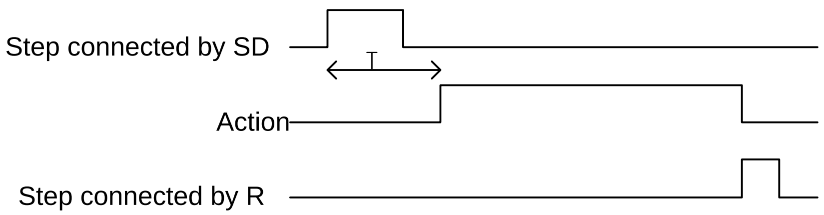




7) SD (Stored & Time Delayed)

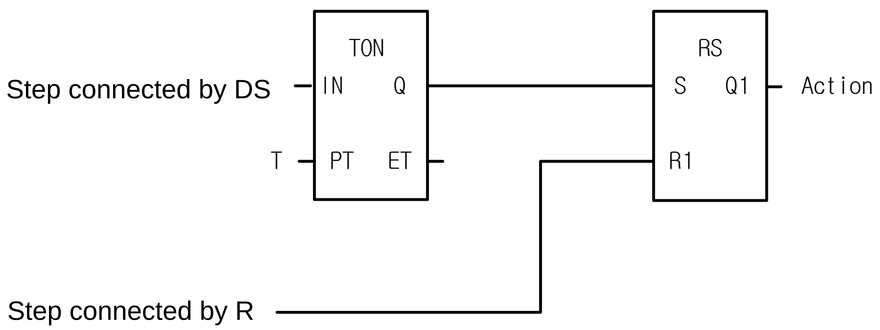
It starts a delay timer when the step activates; after the time delay, the action starts and continues until reset (regardless of step activation/inactivation). If the reset activates during the time delay, the action does not start.

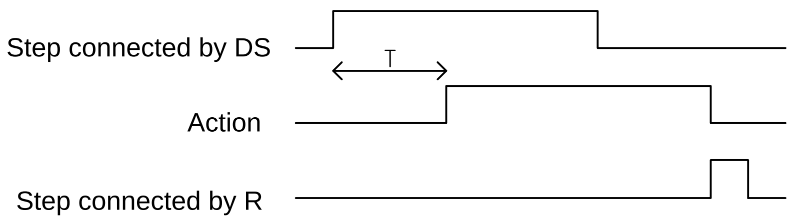




8) DS (Delayed & Stored)

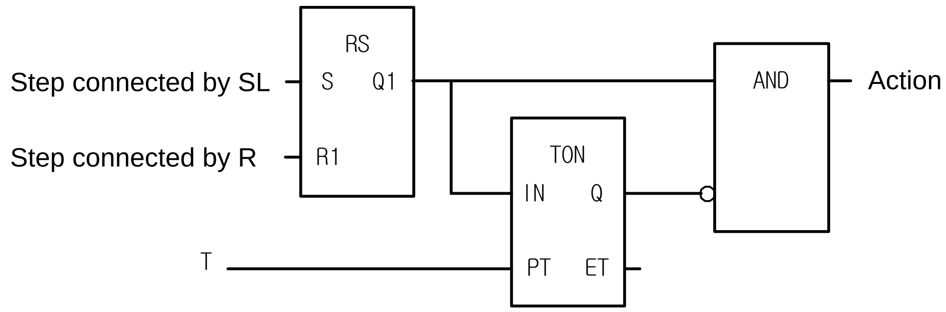
It starts a delay timer when the step activates; after the time delay the action starts (if step is still active) and continues until reset by R qualifier. If the step is inactivates or reset activates during the time delay, the action does not start.

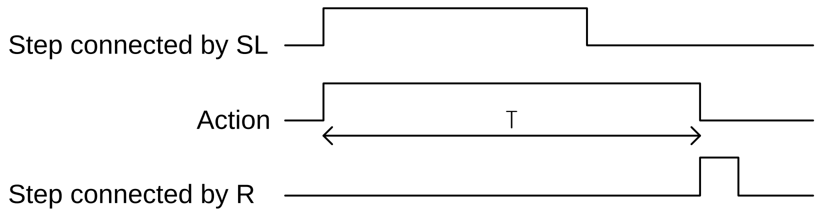




9) SL (Stored & Timed Limited)

It starts the action when the step activates and continues for a set time or until the action is reset (regardless of step activation/inactivation).



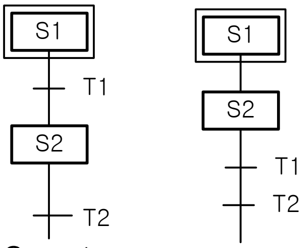


4.3 Extension regulation

4.3.1 Serial connection

▷ steps are always divided by transitions without direct connections.

▷ A Step always divides two transitions without direct connections.

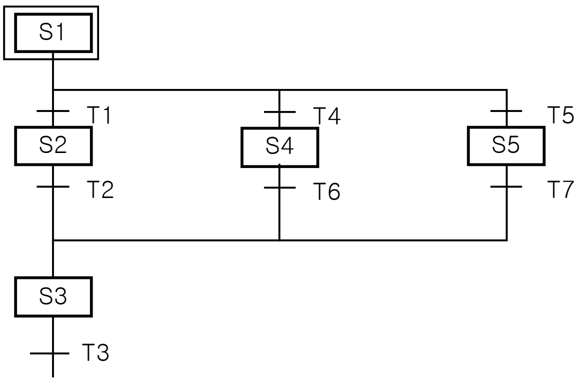


▷ For the transition between steps connected by serial, the lower step activates if the upper step is active and the transition condition connected to the next is 1.

4.3.2 Selection branch

▷ When a processor executes a selection branch, the processor finds the first path with a true transition in the sequence the program scan and executes the steps and transitions in that path. If more than one path in a selection branch becomes true at the same time, the processor chooses the left-most path. The following example shows a typical scan sequence.

Example



* If the transition condition of T1 is 1, the order of activation is S1 -> S2 -> S3.

* If the transition condition of T4 is 1, the order of activation is S1 -> S4 -> S3.

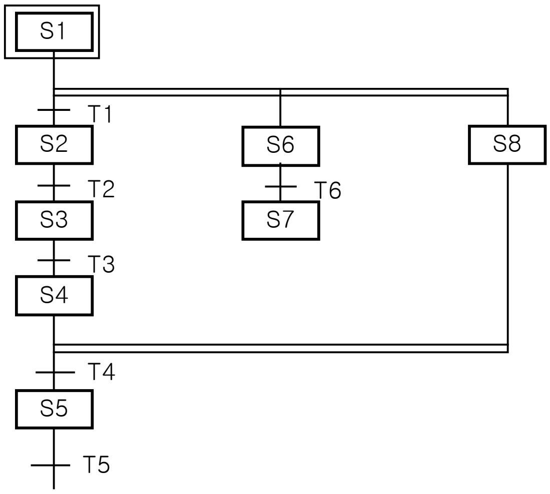
* If the transition condition of T5 is 1, the order of activation is S1 -> S5 -> S3.
  If the transition conditions are 1 at the same time, the processor chooses the left-most path.
* If the transition condition of T1 and T4 is 1 at the same time, the order of activation is S1 -> S2 -> S3..
* If the transition condition of T4 and T5 is 1 at the same time, the order of activation is S1 -> S4 -> S3.

4.3.3 Parallel branch (simultaneous branch)

▷ When connecting using a parallel branch, if the transition condition connected to the next is 1, all steps tied to this transition activates. The extension of each branch is the same as serial connection. The steps in the state of activation are as many as the number of branches.

▷ In case of combining in parallel branch, if the transition condition is 1, when the state of the last steps of each branch activates, then the step connected to the next step activates.

Example



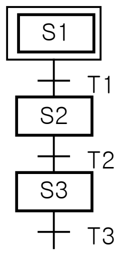
- If the transition condition of T1 is 1 when S1 is active, S2, S6 and S8 is activated and S1 is inactivated.
- If the transition condition of T4 is 1 when S4, S7 and S8 are activated, S5 is activated and S4, S7 and S8 are inactivated.

  * The order of activation
    S1-+->S2--->S3--->S4-+->S5
         +->S6--->S7-------+
         +->S8-------------+

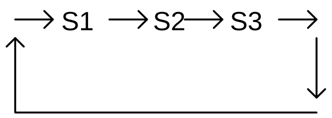
4.3.4 Jump

▷ If the transition condition connected to the next step is 1, after the last step of SFC activates, then the initial step of SFC activates.

Example



- The order of activation

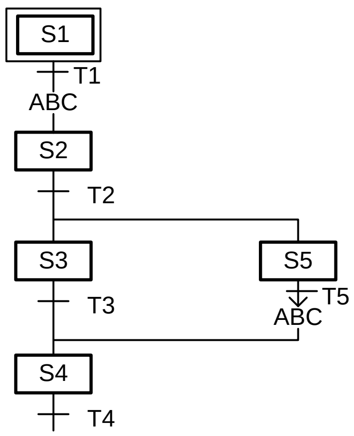


▷ It is possible to extend to the place using a jump.

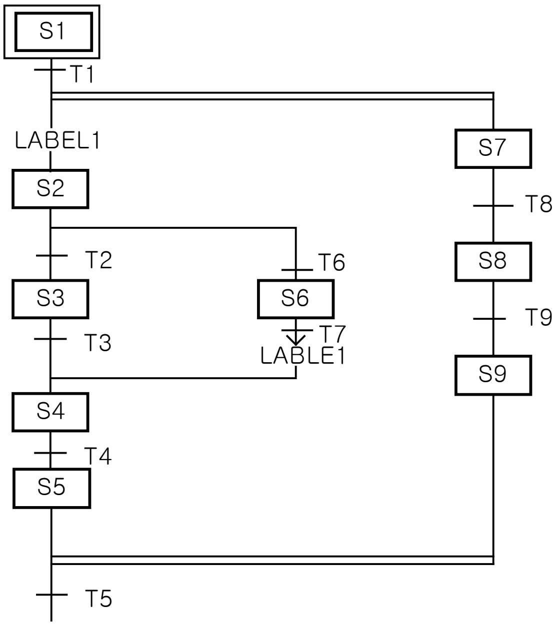
▷ Jump can only be placed at the end of SFC program or at the end of a selection branch.
A jump to the inside or outside of a parallel branch is not permissible; however the jump within a parallel branch is permissible.

Example

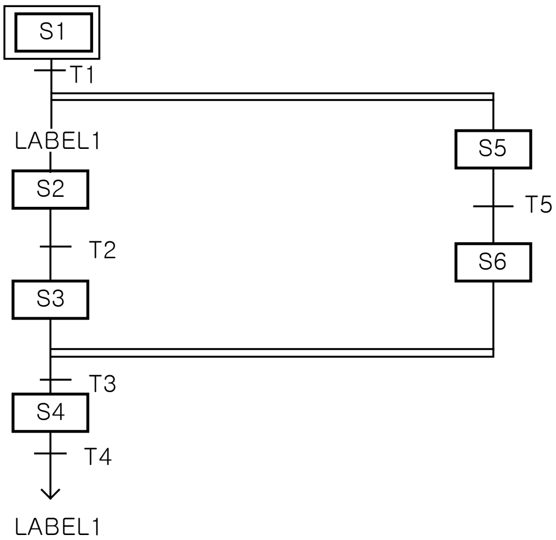
1) Jump at the end of selection branch S2 activates after S5.



2) Jump within a parallel branch



3) You can not jump inside a parallel branch.

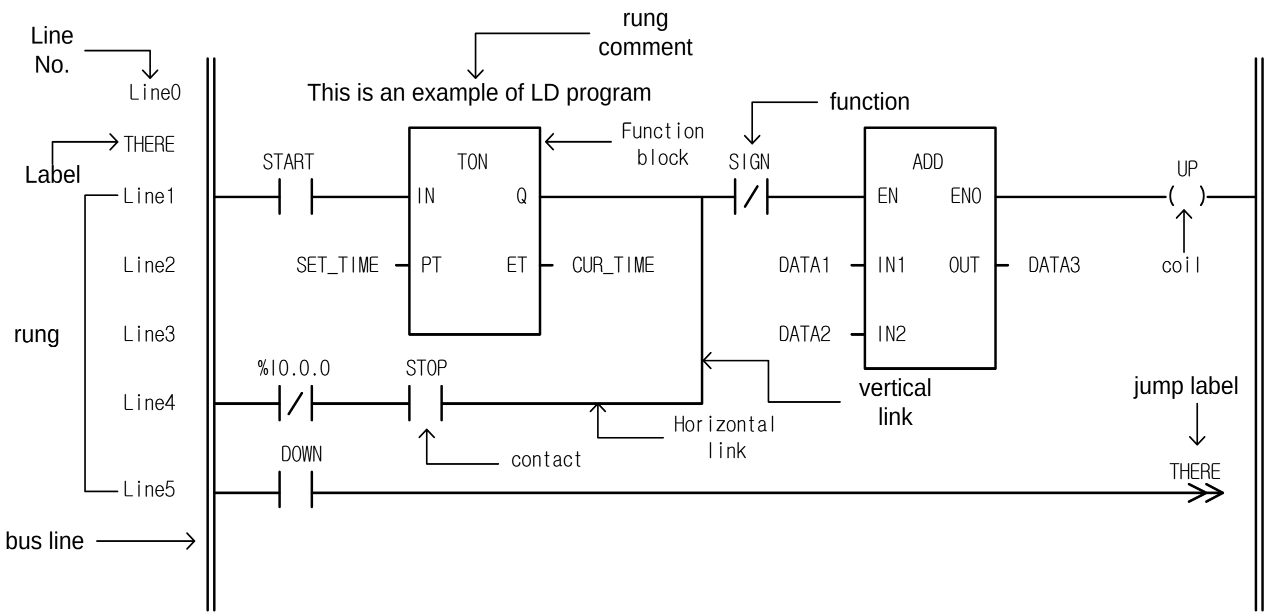


Chapter 5. LD (Ladder Diagram)

5.1 Introduction

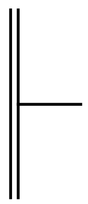
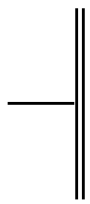
▷ LD program is the graphical representation of a PLC program using symbols such as a coil or contact used in relay logic diagram.

▷ Configuration



5.2 Bus

▷ Bus line as a power line is vertically placed on either sides of a LD graphic diagram.

No	Symbol	Name	Description
1		Left bus line	Its value is always 1 (BOOL).
2		Right bus line	The value is not fixed.

5.3 Link

▷ The value (BOOL 1) of left bus line transmits to the right side by the ladder diagram. The line that transmits value is called as 'power flow line' or 'connection line' which is connected to a contact or coil. Power flow line has always a BOOL value and there is only one power flow line in one rung that is connected by lines.

▷ There are two types of a connection line of LD: horizontal connection line and vertical connection line.

No.	Symbol	Name	Description
1	────	Horizontal connection line	It transmits the left side value to the right side
2	│	Vertical connection line	It is a logical OR of horizontal connection lines of its left side

5.4 Contact

▷ 'Contact' transmits a value to the right horizontal connection line, which is the result of logical AND operation of : the state of left horizontal connection line, Boolean input/output related to the current contact or memory variables. It does not change the value of variable related to the contact. Standard contact symbols are as follows.

Static contact			
No	Symbol	Name	Description
1	*** ─┤ ├─	Normally open contact	When the BOOL variable (marked with ***) is on, which transmits the state of the left connection line to the right connection line. Otherwise, the state of the right connection line is OFF.
2	*** ─┤/├─	Normally closed contact	When the BOOL variable (marked with ***) is off, which transmits the state of the left connection line to the right connection line. Otherwise, the state of the right connection line is off.

State transition-sensing contact			
No	Symbol	Name	Description
3	*** —|P|—	Positive Transition-Sensing Contact	When the BOOL variable (marked with ***), which was off in the previous scan is on, it maintains on state during one scan (current scan).
4	*** —|N|—	Negative Transition-Sensing Contact	When the BOOL variable (marked with ***), which was on in the previous scan is off, it maintains on state during just one scan (current scan).

5.5 Coil

▷ The coil stores the state of the left connection line or the processing result of state transition in the associated BOOL variable. Standard coil symbols are as follows.

▷ Coils are placed in the right extreme of LD, and its right is a right bus line.

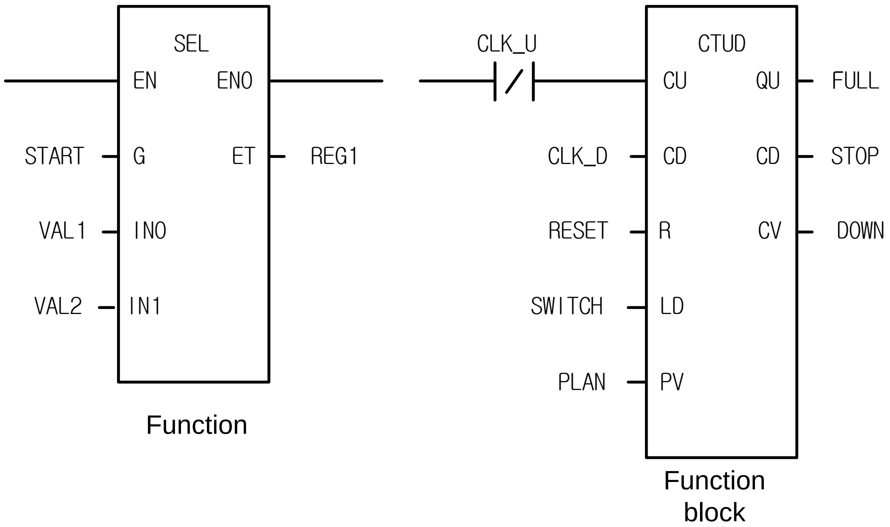
Momentary Coils			
No.	Symbol	Name	Description
1	*** —( )—	Coil	Put the state of left connection line into the associated BOOL variable (marked with ***).
2	*** —(/)—	Negated Coil	Put the negated value of the state of left connection line into the associated BOOL variable (marked with ***). That is, if the state of left connection line is off, the associated BOOL variable is on and if the state of left connection line is on, the associated BOOL variable is off.
Latched Coils			
No.	Symbol	Name	Description
3	*** —(S)—	Set (Latch) Coil	It sets the associated BOOL variable (marked with ***) to on when the left link is in the on state and remains set until reset by a Reset coil.
4	*** —(R)—	Reset (Unlatch) Coil	It sets the associated BOOL variable (marked with ***) to off when the left link is in the on state and remains reset until set by a Set coil.

State Transition-sensing Coils			
No.	Symbol	Name	Description
5	*** —(P)—	Positive Transition-Sensing Coil	If the state of its left connection that was off in the previous scan is on in the current scan, the associated BOOL variable (marked with ***) is on during the current scan.
6	*** —(N)—	Negative Transition-Sensing Coil	If the state of its left connection that was on in the previous scan is off in the current scan, the associated BOOL variable (marked with ***) is on during the current scan.

5.6 Calling of Function and Function Block

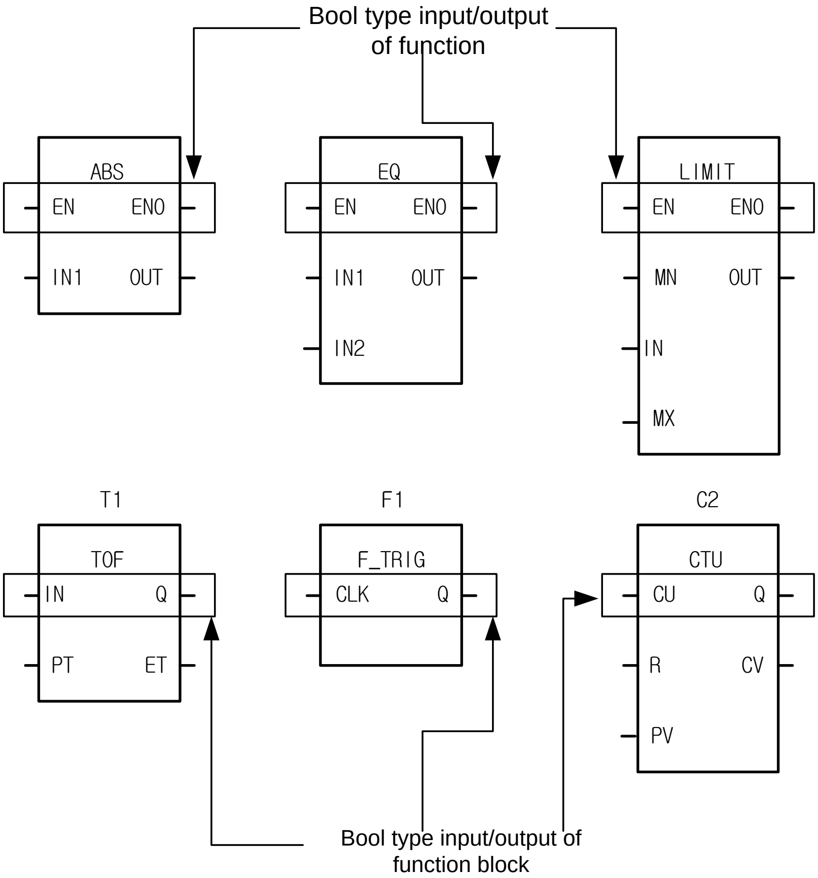
▷ The connection to a function or a function block is done by entering suitable data or variable to their input/output.

Example



▷ To enable power flow inside function or function block, it must contain at least one BOOL-type input and BOOL-type output. EN and ENO are BOOL-type input/output in a function while a data type of the first input and first output are BOOL-type in a function block.

Example



▷ Conventionally, the ladder logic connecting a Boolean input to a function is called EN and the corresponding output Boolean is called ENO, or enable out. If the value of EN is 1, then the function executes, otherwise it is do not execute. In all cases, the value of EN copies the output ENO.

▷ If an error occurs in the execution of a function, the function is responsible to set ENO to false (BOOL 0). EN is connected to the power flow line but ENO does not have to be connected to it. However, when connecting the power flow line to the function output instead of the ENO, the output data type must be a BOOL type.

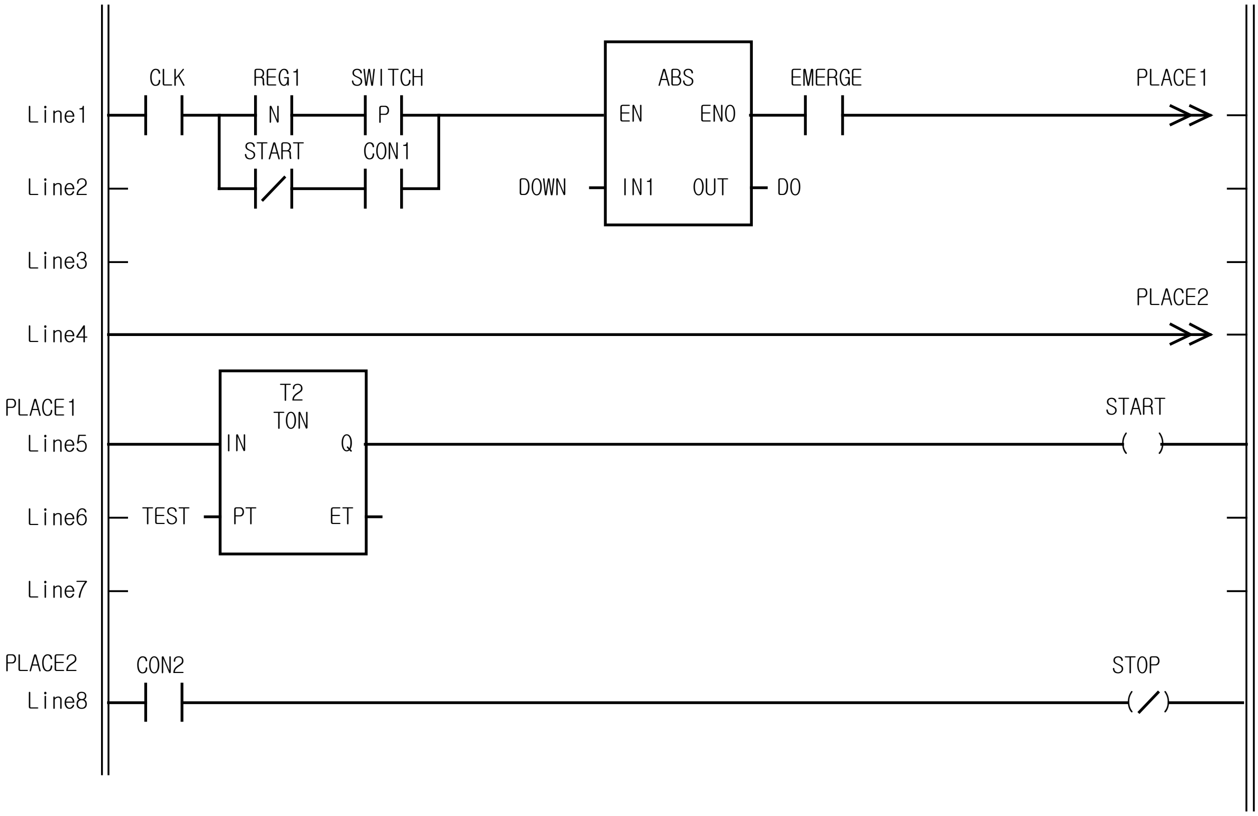
▷ When connecting the power flow line to the function output, do not connect anything to the ENO output. All the inputs of a function are assigned by entering its data at the left side of the function. The output of a function is stored at the output variable on its right side.

▷ Assignment of input of a function block in a LD is the same as that of a function. The name of function block is the 'instance' name, which can be user-defined and must be unique to LD in which the function block appears.

▷ You do not have to assign output variables because they are in the instance. If a function block is connected to the power flow line, it is always executes because there is neither EN nor ENO in it.

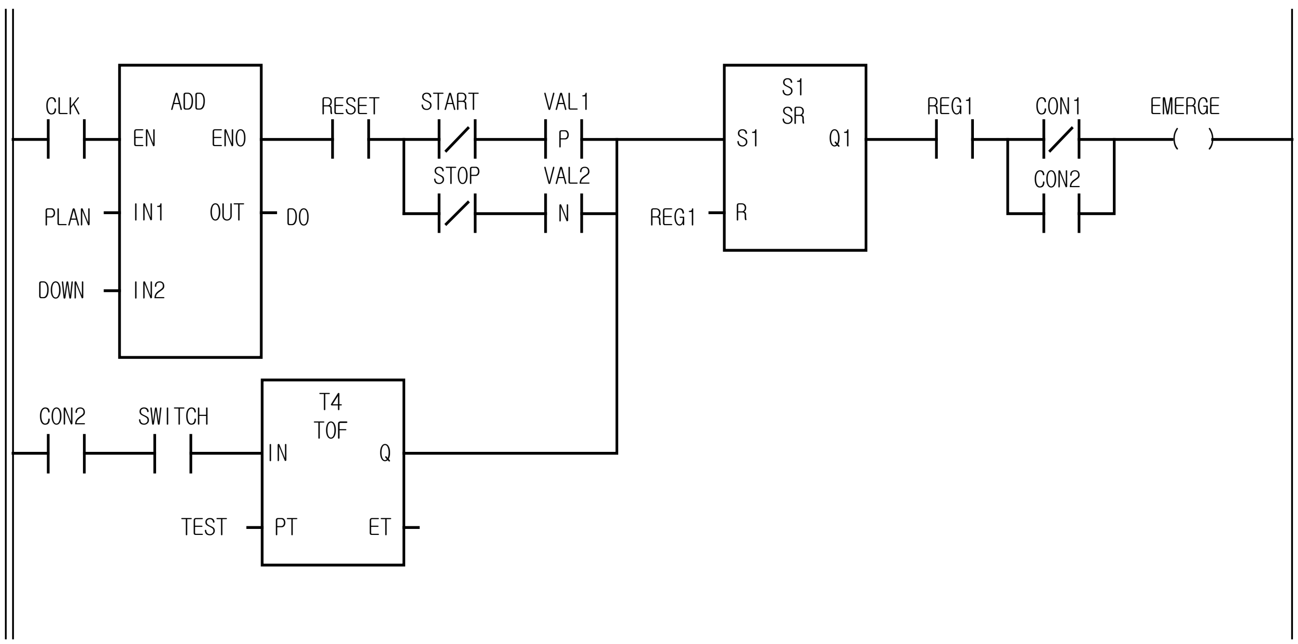
▷ Therefore, use Jump (-->>) to determine whether or not to execute a function block according to the logic result. When connecting the power flow line to the function block, connect it to the input/output whose data type is BOOL.

Example



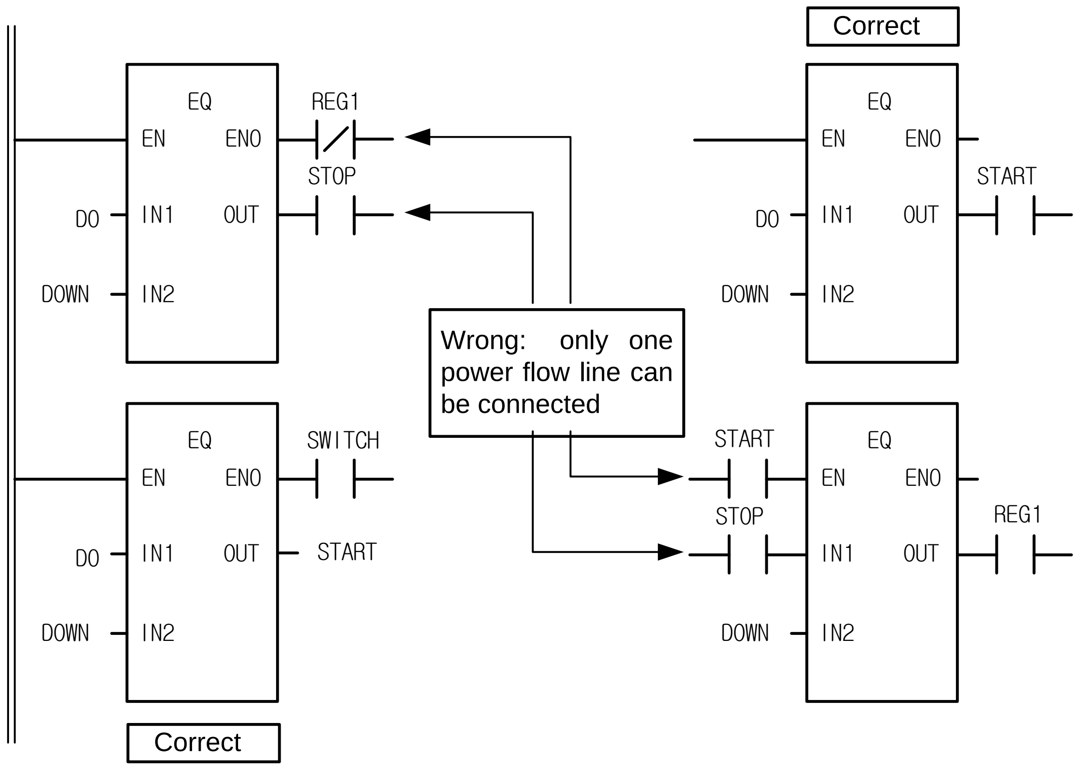
▷ You can place a function or a function block in any place of LD. You can create a program by connecting the power flow line to the output and then insert the contact to it.

Example



▷ Only one power line connects to a function or a function block.

Example



Chapter 6. Function and Function Block

It's a list of function and function block. For each function and function block, please refer to the next chapters (Ch .7/8 Basic/Application Functions and Ch 9/10 Basic/Application Function Blocks).

6.1 Functions

6.1.1 Type Conversion Function

It converts each input data type into an output data type.

Function Group	Function	Input data type	Output data type	Remarks
ARY_ASC_TO_***	ARY_ASC_TO_BYTE	WORD(ASCII)	BYTE	-
	ARY_ASC_TO_BCD	WORD(ASCII)	BYTE (BCD)	-
ARY_BYTE_TO_***	ARY_BYTE_TO_ASC	BYTE	WORD(ASCII)	-
ARY_BCD_TO_***	ARY_BCD_TO_ ASC	BYTE (BCD)	WORD(ASCII)	-
ASC_TO_***	ASC_TO_BCD	BYTE (BCD)	USINT	-
	ASC_TO_BYTE	WORD (BCD)	UINT	-
BCD_TO_***	BYTE_BCD_TO_SINT	BYTE (BCD)	SINT	-
	WORD_BCD_TO_INT	WORD (BCD)	INT	-
	DWORD_BCD_TO_DINT	DWORD (BCD)	DINT	-
	LWORD_BCD_TO_LINT	LWORD (BCD)	LINT	-
	BYTE_BCD_TO_USINT	BYTE (BCD)	USINT	-
	WORD_BCD_TO_UINT	WORD (BCD)	UINT	-
	DWORD_BCD_TO_UDINT	DWORD (BCD)	UDINT	-
	LWORD_BCD_TO_ULINT	LWORD (BCD)	ULINT	-
BCD_TO_ASC	BCD_TO_ASC	BYTE (BCD)	WORD	-
BYTE_TO_ASC	BYTE_TO_ASC	BYTE	ASC(BYTE)	-
TRUNC	TRUNC_REAL	REAL	DINT	-
	TRUNC_LREAL	LREAL	LINT	-
REAL_TO_***	REAL_TO_SINT	REAL	SINT	-
	REAL_TO_INT	REAL	INT	-
	REAL_TO_DINT	REAL	DINT	-
	REAL_TO_LINT	REAL	LINT	-
	REAL_TO_USINT	REAL	USINT	-
	REAL_TO_UINT	REAL	UINT	-
	REAL_TO_UDINT	REAL	UDINT	-
	REAL_TO_ULINT	REAL	ULINT	-
	REAL_TO_DWORD	REAL	DWORD	-
	REAL_TO_LREAL	REAL	LREAL	-
	REAL_TO_STRING	REAL	STRING	-

Function Group	Function	Input data type	Output data type	Remarks
LREAL_TO_***	LREAL_TO_SINT	LREAL	SINT	-
	LREAL_TO_INT	LREAL	INT	-
	LREAL_TO_DINT	LREAL	DINT	-
	LREAL_TO_LINT	LREAL	LINT	-
	LREAL_TO_USINT	LREAL	USINT	-
LREAL_TO_***	LREAL_TO_UINT	LREAL	UINT	-
	LREAL_TO_UDINT	LREAL	UDINT	-
	LREAL_TO_ULINT	LREAL	ULINT	-
	LREAL_TO_LWORD	LREAL	LWORD	-
	LREAL_TO_REAL	LREAL	REAL	-
	LREAL_TO_STRING	LREAL	STRING	-
SINT_TO_***	SINT_TO_INT	SINT	INT	-
	SINT_TO_DINT	SINT	DINT	-
	SINT_TO_LINT	SINT	LINT	-
	SINT_TO_USINT	SINT	USINT	-
	SINT_TO_UINT	SINT	UINT	-
	SINT_TO_UDINT	SINT	UDINT	-
	SINT_TO_ULINT	SINT	ULINT	-
	SINT_TO_BOOL	SINT	BOOL	-
	SINT_TO_BYTE	SINT	BYTE	-
	SINT_TO_WORD	SINT	WORD	-
	SINT_TO_DWORD	SINT	DWORD	-
	SINT_TO_LWORD	SINT	LWORD	-
	SINT_TO_REAL	SINT	REAL	-
	SINT_TO_LREAL	SINT	LREAL	-
	SINT_TO_STRING	SINT	STRING	-
INT_TO_***	INT_TO_SINT	INT	SINT	-
	INT_TO_DINT	INT	DINT	-
	INT_TO_LINT	INT	LINT	-
	INT_TO_USINT	INT	USINT	-
	INT_TO_UINT	INT	UINT	-
	INT_TO_UDINT	INT	UDINT	-
	INT_TO_ULINT	INT	ULINT	-
	INT_TO_BOOL	INT	BOOL	-
	INT_TO_BYTE	INT	BYTE	-
	INT_TO_WORD	INT	WORD	-
	INT_TO_DWORD	INT	DWORD	-
	INT_TO_LWORD	INT	LWORD	-
	INT_TO_REAL	INT	REAL	-
	INT_TO_LREAL	INT	LREAL	-
	INT_TO_STRING	INT	STRING	-
DINT_TO_***	DINT_TO_SINT	DINT	SINT	-
	DINT_TO_INT	DINT	INT	-
	DINT_TO_LINT	DINT	LINT	-

Function Group	Function	Input data type	Output data type	Remarks
	DINT_TO_USINT	DINT	USINT	-
	DINT_TO_UINT	DINT	UINT	-
	DINT_TO_UDINT	DINT	UDINT	-
	DINT_TO_ULINT	DINT	ULINT	-
	DINT_TO_BOOL	DINT	BOOL	-
	DINT_TO_BYTE	DINT	BYTE	-
	DINT_TO_WORD	DINT	WORD	-
DINT_TO_***	DINT_TO_DWORD	DINT	DWORD	-
	DINT_TO_LWORD	DINT	LWORD	-
	DINT_TO_REAL	DINT	REAL	-
	DINT_TO_LREAL	DINT	LREAL	-
	DINT_TO_STRING	DINT	STRING	-
LINT_TO_***	LINT_TO_SINT	LINT	SINT	-
	LINT_TO_INT	LINT	INT	-
	LINT_TO_DINT	LINT	DINT	-
	LINT_TO_USINT	LINT	USINT	-
	LINT_TO_UINT	LINT	UINT	-
	LINT_TO_UDINT	LINT	UDINT	-
	LINT_TO_ULINT	LINT	ULINT	-
	LINT_TO_BOOL	LINT	BOOL	-
	LINT_TO_BYTE	LINT	BYTE	-
	LINT_TO_WORD	LINT	WORD	-
	LINT_TO_DWORD	LINT	DWORD	-
	LINT_TO_LWORD	LINT	LWORD	-
	LINT_TO_REAL	LINT	REAL	-
	LINT_TO_LREAL	LINT	LREAL	-
	LINT_TO_STRING	LINT	STRING	-
USINT_TO_***	USINT_TO_SINT	USINT	SINT	-
	USINT_TO_INT	USINT	INT	-
	USINT_TO_DINT	USINT	DINT	-
	USINT_TO_LINT	USINT	LINT	-
	USINT_TO_UINT	USINT	UINT	-
	USINT_TO_UDINT	USINT	UDINT	-
	USINT_TO_ULINT	USINT	ULINT	-
	USINT_TO_BOOL	USINT	BOOL	-
	USINT_TO_BYTE	USINT	BYTE	-
	USINT_TO_WORD	USINT	WORD	-
	USINT_TO_DWORD	USINT	DWORD	-
	USINT_TO_LWORD	USINT	LWORD	-
	USINT_TO_REAL	USINT	REAL	-
	USINT_TO_LREAL	USINT	LREAL	-
	USINT_TO_STRING	USINT	STRING	-
UINT_TO_***	UINT_TO_SINT	UINT	SINT	-
	UINT_TO_INT	UINT	INT	-

Function Group	Function	Input data type	Output data type	Remarks
	UINT_TO_DINT	UINT	DINT	-
	UINT_TO_LINT	UINT	LINT	-
	UINT_TO_USINT	UINT	USINT	-
	UINT_TO_UDINT	UINT	UDINT	-
	UINT_TO_ULINT	UINT	ULINT	-
	UINT_TO_BOOL	UINT	BOOL	-
	UINT_TO_BYTE	UINT	BYTE	-
	UINT_TO_WORD	UINT	WORD	-
	UINT_TO_DWORD	UINT	DWORD	-
UINT_TO_***	UINT_TO_LWORD	UINT	LWORD	-
	UINT_TO_REAL	UINT	REAL	-
	UINT_TO_STRING	UINT	STRING	-
	UINT_TO_LREAL	UINT	LREAL	-
	UINT_TO_DATE	UINT	DATE	-
UDINT_TO_***	UDINT_TO_SINT	UDINT	SINT	-
	UDINT_TO_INT	UDINT	INT	-
	UDINT_TO_DINT	UDINT	DINT	-
	UDINT_TO_LINT	UDINT	LINT	-
	UDINT_TO_USINT	UDINT	USINT	-
	UDINT_TO_UINT	UDINT	UINT	-
	UDINT_TO_ULINT	UDINT	ULINT	-
	UDINT_TO_BOOL	UDINT	BOOL	-
	UDINT_TO_BYTE	UDINT	BYTE	-
	UDINT_TO_WORD	UDINT	WORD	-
	UDINT_TO_DWORD	UDINT	DWORD	-
	UDINT_TO_LWORD	UDINT	LWORD	-
	UDINT_TO_REAL	UDINT	REAL	-
	UDINT_TO_LREAL	UDINT	LREAL	-
	UDINT_TO_TOD	UDINT	TOD	-
	UDINT_TO_TIME	UDINT	TIME	-
	UDINT_TO_STRING	UDINT	STRING	-
ULINT_TO_***	ULINT_TO_SINT	ULINT	SINT	-
	ULINT_TO_INT	ULINT	INT	-
	ULINT_TO_DINT	ULINT	DINT	-
	ULINT_TO_LINT	ULINT	LINT	-
	ULINT_TO_USINT	ULINT	USINT	-
	ULINT_TO_UINT	ULINT	UINT	-
	ULINT_TO_UDINT	ULINT	UDINT	-
	ULINT_TO_BOOL	ULINT	BOOL	-
	ULINT_TO_BYTE	ULINT	BYTE	-
	ULINT_TO_WORD	ULINT	WORD	-
	ULINT_TO_DWORD	ULINT	DWORD	-
	ULINT_TO_LWORD	ULINT	LWORD	-
	ULINT_TO_REAL	ULINT	REAL	-

Function Group	Function	Input data type	Output data type	Remarks
	ULINT_TO_LREAL	ULINT	LREAL	-
	ULINT_TO_STRING	ULINT	STRING	-
BOOL_TO_***	BOOL_TO_SINT	BOOL	SINT	-
	BOOL_TO_INT	BOOL	INT	-
	BOOL_TO_DINT	BOOL	DINT	-
	BOOL_TO_LINT	BOOL	LINT	-
	BOOL_TO_USINT	BOOL	USINT	-
	BOOL_TO_UINT	BOOL	UINT	-
	BOOL_TO_UDINT	BOOL	UDINT	-
	BOOL_TO_ULINT	BOOL	ULINT	-
	BOOL_TO_BYTE	BOOL	BYTE	-
BOOL_TO_***	BOOL_TO_WORD	BOOL	WORD	-
	BOOL_TO_DWORD	BOOL	DWORD	-
	BOOL_TO_LWORD	BOOL	LWORD	-
	BOOL_TO_STRING	BOOL	STRING	-
BYTE_TO_***	BYTE_TO_SINT	BYTE	SINT	-
	BYTE_TO_INT	BYTE	INT	-
	BYTE_TO_DINT	BYTE	DINT	-
	BYTE_TO_LINT	BYTE	LINT	-
	BYTE_TO_USINT	BYTE	USINT	-
	BYTE_TO_UINT	BYTE	UINT	-
	BYTE_TO_UDINT	BYTE	UDINT	-
	BYTE_TO_ULINT	BYTE	ULINT	-
	BYTE_TO_BOOL	BYTE	BOOL	-
	BYTE_TO_WORD	BYTE	WORD	-
	BYTE_TO_DWORD	BYTE	DWORD	-
	BYTE_TO_LWORD	BYTE	LWORD	-
	BYTE_TO_STRING	BYTE	STRING	-
WORD_TO_***	WORD_TO_SINT	WORD	SINT	-
	WORD_TO_INT	WORD	INT	-
	WORD_TO_DINT	WORD	DINT	-
	WORD_TO_LINT	WORD	LINT	-
	WORD_TO_USINT	WORD	USINT	-
	WORD_TO_UINT	WORD	UINT	-
	WORD_TO_UDINT	WORD	UDINT	-
	WORD_TO_ULINT	WORD	ULINT	-
	WORD_TO_BOOL	WORD	BOOL	-
	WORD_TO_BYTE	WORD	BYTE	-
	WORD_TO_DWORD	WORD	DWORD	-
	WORD_TO_LWORD	WORD	LWORD	-
	WORD_TO_DATE	WORD	DATE	-
	WORD_TO_STRING	WORD	STRING	-
DWORD_TO_***	DWORD_TO_SINT	DWORD	SINT	-
	DWORD_TO_INT	DWORD	INT	-

Function Group	Function	Input data type	Output data type	Remarks
	DWORD_TO_DINT	DWORD	DINT	-
	DWORD_TO_LINT	DWORD	LINT	-
	DWORD_TO_USINT	DWORD	USINT	-
	DWORD_TO_UINT	DWORD	UINT	-
	DWORD_TO_UDINT	DWORD	UDINT	-
	DWORD_TO_ULINT	DWORD	ULINT	-
	DWORD_TO_BOOL	DWORD	BOOL	-
	DWORD_TO_BYTE	DWORD	BYTE	-
	DWORD_TO_WORD	DWORD	WORD	-
	DWORD_TO_LWORD	DWORD	LWORD	-
	DWORD_TO_REAL	DWORD	REAL	-
	DWORD_TO_TIME	DWORD	TIME	-
	DWORD_TO_TOD	DWORD	TOD	-
DWORD_TO_***	DWORD_TO_STRING	DWORD	STRING	-
LWORD_TO_***	LWORD_TO_SINT	LWORD	SINT	-
	LWORD_TO_INT	LWORD	INT	-
	LWORD_TO_DINT	LWORD	DINT	-
	LWORD_TO_LINT	LWORD	LINT	-
	LWORD_TO_USINT	LWORD	USINT	-
	LWORD_TO_UINT	LWORD	UINT	-
	LWORD_TO_UDINT	LWORD	UDINT	-
	LWORD_TO_ULINT	LWORD	ULINT	-
	LWORD_TO_BOOL	LWORD	BOOL	-
	LWORD_TO_BYTE	LWORD	BYTE	-
	LWORD_TO_WORD	LWORD	WORD	-
	LWORD_TO_DWORD	LWORD	DWORD	-
	LWORD_TO_LREAL	LWORD	LREAL	-
	LWORD_TO_DT	LWORD	DT	-
	LWORD_TO_STRING	LWORD	STRING	-
STRING_TO_***	STRING _TO_SINT	STRING	SINT	-
	STRING _TO_INT	STRING	INT	-
	STRING _TO_DINT	STRING	DINT	-
	STRING _TO_LINT	STRING	LINT	-
	STRING _TO_USINT	STRING	USINT	-
	STRING _TO_UINT	STRING	UINT	-
	STRING _TO_UDINT	STRING	UDINT	-
	STRING _TO_ULINT	STRING	ULINT	-
	STRING _TO_BOOL	STRING	BOOL	-
	STRING _TO_BYTE	STRING	BYTE	-
	STRING _TO_WORD	STRING	WORD	-
	STRING _TO_DWORD	STRING	DWORD	-
	STRING _TO_LWORD	STRING	LWORD	-
	STRING _TO_REAL	STRING	REAL	-
	STRING _TO_LREAL	STRING	LREAL	-

Function Group	Function	Input data type	Output data type	Remarks
	STRING _TO_DT	STRING	DT	-
	STRING _TO_DATE	STRING	DATE	-
	STRING _TO_TOD	STRING	TOD	-
	STRING _TO_TIME	STRING	TIME	-
TIME_TO_***	TIME_TO_UDINT	TIME	UDINT	-
	TIME_TO_DWORD	TIME	DWORD	-
	TIME_TO_STRING	TIME	STRING	-
DATE_TO_***	DATE_TO_UINT	DATE	UINT	-
	DATE_TO_WORD	DATE	WORD	-
	DATE_TO_STRING	DATE	STRING	-
TOD_TO_***	TOD_TO_UDINT	TOD	UDINT	-
	TOD_TO_DWORD	TOD	DWORD	-
	TOD_TO_STRING	TOD	STRING	-
DT_TO_***	DT_TO_LWORD	DT	LWORD	-
	DT_TO_DATE	DT	DATE	-
	DT_TO_TOD	DT	TOD	-
	DT_TO_STRING	DT	STRING	-
***_TO_BCD	SINT_TO_BCD_BYTE	SINT	BYTE (BCD)	-
	INT_TO_BCD_WORD	INT	WORD (BCD)	-
	DINT_TO_BCD_DWORD	DINT	DWORD (BCD)	-
	LINT_TO_BCD_LWORD	LINT	LWORD (BCD)	-
	USINT_TO_BCD_BYTE	USINT	BYTE (BCD)	-
	UINT_TO_BCD_WORD	UINT	WORD (BCD)	-
	UDINT_TO_BCD_DWORD	UDINT	DWORD (BCD)	-
	ULINT_TO_BCD_LWORD	ULINT	LWORD (BCD)	-

6.1.2 Numerical operation function

1) Numerical operation function with one Input

No.	Function	Function	Remarks
General Function			
1	ABS	Absolute value operation	-
2	SQRT	Square root operation	-
Logarithm			
3	LN	Natural logarithm operation	-
4	LOG	Common logarithm Base to 10 operation	-
5	EXP	Natural exponential operation	-

Trigonometric function			
6	SIN	Sine operation	-
7	COS	Cosine operation	-
8	TAN	Tangent operation	-
9	ASIN	Arc sine operation	-
10	ACOS	Arc Cosine operation	-
11	ATAN	Arc Tangent operation	-
Angle function			
12	RAD_REAL	Convert degree into radian	-
13	RAD_LREAL		
14	DEG_REAL	Convert radian into degree	-
15	DEG_LREAL		

2) Basic arithmetic function

No.	Function	Description	Remarks
Operation function whose input number (n) can be extended up to 8.			
1	ADD	Addition (OUT <= IN1 + IN2 + ... + INn)	-
2	MUL	Multiplication (OUT <= IN1 * IN2 * ... * INn)	-
Operation function of which input number is fixed.			
3	SUB	Subtraction (OUT <= IN1 - IN2)	-
4	DIV	Division (OUT <= IN1 / IN2)	-
5	MOD	Calculate remainder (OUT <= IN1 Modulo IN2)	-
6	EXPT	Exponential operation (OUT <= $IN1^{IN2}$)	-
7	MOVE	Copy data (OUT <= IN)	-
Input data exchange			
8	XCHG_***	Exchanges two input data	-

6.1.3 Bit array function

1) Bit-shift function

No.	Function	Description	Remarks
1	SHL	Shift input to the left of N bit(the right is filled with 0)	-
2	SHR	Shift input to the right of N bit (the left is filled with 0)	-
3	SHIFT_C_***	Shift input to the configured direction as much as N bit (carry)	-
4	ROL	Rotate input to the left of N bit	-
5	ROR	Rotate input to the right of N bit	-
6	ROTATE_C_***	Rotate input to the direction as much as N bit (carry)	-

2) Bit operation function

No.	Function	Description (n can be extended up to 8)	Remarks
1	AND	Logical AND (OUT <= IN1 AND IN2 AND ... AND INn)	-
2	OR	Logical OR (OUT <= IN1 OR IN2 OR ... OR INn)	-
3	XOR	Exclusive OR (OUT <= IN1 XOR IN2 XOR ... XOR INn)	-
4	NOT	Reverse logic (OUT <= NOT IN1)	-
5	XNR	Exclusive logic AND (OUT <= IN1 XNR IN2 XNR ... XNR INn)	-

6.1.4 Selection function

No.	Function	Description(n can be extended up to 8)	Remarks
1	SEL	Selects from two inputs (IN0 or IN1)	-
2	MAX	Produces the maximum value among input IN1,...INn	-
3	MIN	Produces the minimum value among input IN1,...INn	-
4	LIMIT	Limits upper and lower boundaries	-
5	MUX	Outputs the Kth input among input IN1,…INn	-

6.1.5 Data exchange function

No.	Function	Description	Remarks
1	SWAP_BYTE	Swaps upper NIBBLE for lower NIBBLE data of BYTE.	-
	SWAP_WORD	Swaps upper BYTE for lower BYTE data of WORD.	-
	SWAP_DWORD	Swaps upper WORD for lower WORD data DWORD.	-
	SWAP_LWORD	Swaps upper DWORD for lower DWORD data of LWORD.	-
2	ARY_SWAP_BYTE	Swaps upper/lower NIBBLE of BYTE elements in array.	-
	ARY_SWAP_WORD	Swaps upper/lower BYTE of WORD elements in array.	-
	ARY_SWAP_DWORD	Swaps upper/lower WORD of DWORD elements in array.	-
	ARY_SWAP_LWORD	Swaps upper/lower DWORD of LWORD elements in array.	-

6.1.6 Comparison function

No.	Function	Description (n can be extended up to 8)	Remarks
1	GT	‘Greater than’ comparison OUT <= (IN1>IN2) & (IN2>IN3) & ... & (INn-1 > INn)	-
2	GE	‘Greater than or equal to’ comparison OUT <= (IN1>=IN2) & (IN2>=IN3) & ... & (INn-1 >= INn)	-
3	EQ	‘Equal to’ comparison	-

		OUT <= (IN1=IN2) & (IN2=IN3) & ... & (INn-1 = INn)	
4	LE	'Less than or equal to' comparison OUT <= (IN1<=IN2) & (IN2<=IN3) & ... & (INn-1 <= INn)	-
5	LT	'Less than' comparison OUT <= (IN1<IN2) & (IN2<IN3) & ... & (INn-1 < INn)	-
6	NE	'Not equal to' comparison OUT <= (IN1<>IN2) & (IN2<>IN3) & ... & (INn-1 <> INn)	-

6.1.7 Character string function

No.	Function	Description	Remarks
1	LEN	Find a length of a character string	-
2	LEFT	Take a left side of a string (size of L) and output it	-
3	RIGHT	Take a right side of a string (size of L) and output it	-
4	MID	Take a middle side of a string (size of L from the Pth character)	-
5	CONCAT	Concatenate the input character string in order	-
6	INSERT	Insert the second string after the Pth character of the first string	-
7	DELETE	Delete a string (size of L from the Pth character)	-
8	REPLACE	Replace a size of L from the Pth character of the first string by the second string	-
9	FIND	Find a starting point of the first string which has a same pattern of the second string.	-

6.1.8 Date and time of day function

No.	Function	Description	Remarks
1	ADD_TIME	Add time (time/time of day/date and time addition)	-
2	SUB_TIME	Subtract time (time/time of day/date and time subtraction)	-
	SUB_DATE	Calculate time by subtracting date from date	-
	SUB_TOD	Calculate time by subtracting TOD from TOD	-
	SUB_DT	Calculate time by subtracting DT from DT	-
3	MUL_TIME	Multiply number to time	-
4	DIV_TIME	Divide time by number	-
5	CONCAT_TIME	Concatenate date to make TOD	-

6.1.9 System control function

No.	Function	Description	Remarks
1	DI	Invalidates interrupt (not to permit task program to start)	-
2	EI	Permits running for a task program	-
3	STOP	Stop running by a task program	-
4	ESTOP	Emergency running stop by a program	-
5	DIREC_IN	Update input data	-
6	DIREC_O	Updates output data	-
7	WDT_RST	Initialize a timer of watchdog	-
8	MCS	Master Control	-
9	MCSCLR	Master Control Clear	-
10	FALS	Self check(error display)	-
11	OUTOFF	Output off	-

6.1.10 File function

No.	Function	Description	Remarks
1	RSET	Setting file register block number	-
2	EBCMP	Block comparison	-
3	EMOV	Reading data from the preset flash area	-
4	EERRST	Flash memory related error flag clear	-

6.1.11 Data manipulation function

No.	Function	Description	Remarks
1	MEQ_***	Compare whether two inputs are equal after masking	-
2	DIS_***	Data distribution	-
3	UNI_***	Unite data	-
4	BIT_BYTE	Combine 8 bits into one BYTE	-
5	BYTE_BIT	Divide one BYTE into 8 bits	-
6	BYTE_WORD	Combine two bytes into one WORD	-
7	WORD_BYTE	Divide one WORD into two bytes	-
8	WORD_DWORD	Combine two WORD data into DWORD	-
9	DWORD_WORD	Divide DWORD into 2 WORD data	-

10	DWORD_LWORD	Combine two DWORD data into LWORD	-
11	LWORD_DWORD	Divide LWORD into two DWORD data	-
12	GET_CHAR	Get one character from a character string	-
13	PUT_CHAR	Puts a character in a string	-
14	STRING_BYTE	Convert a string into a byte array	-
15	BYTE_STRING	Convert a byte array into a string	-

6.1.12 Stack operation function

No.	Function	Description	Remarks
1	FIFO_***	First In First Out	-
2	LIFO_***	Last In First Out	-

6.2 MK (MASTER-K) function

No.	Function	Description(n can be extended up to 8)	Remarks
1	ENCO_B,W,D,L	Output a position of on bit by number	-
2	DECO_B,W,D,L	Turn a selected bit on	-
3	BSUM_B,W,D,L	Output a number of on bit	-
4	SEG_WORD	Convert BCD/HEX into 7-segment code	-
5	BMOV_B,W,D,L	Move part of a bit string	-
6	INC_B,W,D,L	Increase IN data	-
7	DEC_B,W,D,L	Decrease IN data	-

6.3 Array operation function

No.	Function	Description	Remarks
1	ARY_MOVE	Copy array-typed data (OUT <= IN)	-
2	ARY_CMP_***	Array comparison	-
3	ARY_SCH_***	Array search	-
4	ARY_FLL_***	Filling an array with data	-
5	ARY_AVE_***	Find an average of an array	-
6	ARY_SFT_C_***	Array bit shift left with carry	-
7	ARY_ROT_C_***	Bit rotation of array with carry	-
8	SHIFT_A_***	Shift array elements	-

9	ROTATE_A_***	Rotates array elements	-
10	ARY_CMP_EQ	Equivalent comparison of the two Array Elements	-
11	ARY_CMP_NE	Not equal comparison of the two Array Elements	-

6.4 Basic function block

6.4.1 Bistable function block

No.	Function Block	Description	Remarks
1	SR	Set preference bistable	-
2	RS	Reset preference bistable	-
3	SEMA	Semaphore	-

6.4.2 - detection function block

No.	Function Block	Description	Remarks
1	R_TRIG	Rising - detector	-
2	F_TRIG	Falling - detector	-
3	FF	Reverse output if input condition rises	-

6.4.3 Counter

No.	Function Block	Description	Remarks
1	CTU_***	Up Counter INT,DINT,LINT,UINT,UDINT,ULINT	-
2	CTD_***	Down Counter INT,DINT,LINT,UINT,UDINT,ULINT	-
3	CTUD_***	Up Down Counter INT,DINT,LINT,UINT,UDINT,ULINT	-
4	CTR	Ring Counter	-

6.4.4 Timer

No.	Function Block	Description	Remarks
1	TP	Pulse Timer	-
2	TON	On-Delay Timer	-

3	TOF	Off-Delay Timer	-
4	TMR	Integrating Timer	-
5	TP_RST	TP with reset	-
6	TRTG	Retriggerable Timer	-
7	TOF_RST	TOF with reset	-
8	TON_UINT	TON with integer setting	-
9	TOF_UINT	TOF with integer setting	-
10	TP_UINT	TP with integer setting	-
11	TMR_UINT	TMR with integer setting	-
12	TMR_FLK	Blink timer	-
13	TRTG_UINT	Integer setting retriggerable timer	-

6.4.5 File function block

No.	Function Block	Description	Remarks
1	EBREAD	Read R area data from flash area	-
1	EBWRITE	Write R area data to flash area	-

6.4.6 Other function block

No.	Function Block	Description	Remarks
1	SCON	Step Controller	-
2	DUTY	Scan setting on/off	-
3	RTC_SET	Write time data	-
4	SPA	Solar Position Algorithm	

6.4.7 Communication function block

No.	Function Block	Description	Remarks
1	P2PSN	Station no. setting	-
2	P2PRD	Read area setting	-
3	P2PWR	Write area setting	-
4	SEND_UDATA	User defined data send	-
5	RCV_UDATA	User defined data receive	-
6	SEND_DTR	Communication ready signal send	-
7	SEND_RTS	State signal of receive buffer send	-
8	GET_IP	Read local ethernet information	-
9	SET_IP	Local ethernet information setting	-

6.4.8 Special function block

No.	Function Block	Description	Remarks
1	GET	Read special module data	-
2	PUT	Write special module data	-
3	ARY_GET	Read special module data(array)	-
4	ARY_PUT	Write special module data(array)	-
5	GETE	Read special module data(Access upper word)	-
6	PUTE	Write special module data(Access upper word)	-
7	ARY_GETE	Read special module data(array, Access upper word)	-
8	ARY_PUTE	Write special module data(array, Access upper word)	-

6.4.9 Motion control function block

No.	Function Block	Description	Remarks
1	GETM	Read motion control module data	-
2	PUTM	Write motion control module data	-
3	ARY_GETM	Read motion control module data(array)	-
4	ARY_PUTM	Write motion control module data(array)	-

6.4.10 Positioning function block (APM)

No.	Function Block	Description	Remarks
1	APM_ORG	Return to original point	-
2	APM_FLT	Floating original point setting	-
3	APM_DST	Direct run	-
4	APM_IST	Indirect run	-
5	APM_LIN	Linear interpolation run	-
6	APM_CIN	Circular interpolation run	-
7	APM_SST	Simultaneous run	-
8	APM_VTP	Speed/position control conversion	-
9	APM_PTV	Position/speed control conversion	-
10	APM_STP	Decelerating stop	-
11	APM_SKP	Skip run	-
12	APM_SSP	Position synchronization	-
13	APM_SSS	Speed synchronization	-
14	APM_SSSP	Positioning speed synchronization	

No.	Function Block	Description	Remarks
15	APM_POR	Position override	-
16	APM_SOR	Speed override	-
17	APM_PSO	Positioning speed override	-
18	APM_NMV	Continuous run	-
19	APM_INC	Inching run	-
20	APM_RTP	Return run to the previous position of manual operation	-
21	APM_SNS	Run step no. change	-
22	APM_SRS	Repeat step no. change	-
23	APM_MOF	M code cancel	-
24	APM_PRS	Present position preset	-
25	APM_ZONE	Zone output allowed/prohibited	-
26	APM_EPRE	Encoder value preset	-
27	APM_TEA	Singular teaching(ROM, RAM)	-
28	APM_ATEA	Plural teaching(ROM, RAM)	-
29	APM_SBP	Basic parameter setting	-
30	APM_SEP	Extension parameter setting	-
31	APM_SHP	Original point return parameter setting	-
32	APM_SMP	Manual operation parameter setting	-
33	APM_SIP	Input signal parameter setting	-
34	APM_SCP	Common parameter setting	-
35	APM_SMD	Operation data setting	-
36	APM_EMG	Emergency stop	-
37	APM_RST	Error reset/output prohibition cancel	-
38	APM_PST	Point run	-
39	APM_WRT	Saving parameter/run data	-
40	APM_CRD	Reading run info	-
41	APM_SRD	Reading run info	-
42	APM_ENCRD	Reading encoder value	-
43	APM_JOG	Jog run	-
44	APM_MPG	Manual pulse generator(MPG) run	-
45	APM_RCP	Repeating current position section	
46	APM_VRD	Read Variable Data	-
47	APM_VWR	Write Variable Data	-
48	APM_VTPP	Positioning speed/position conversion control	-

6.4.11 Positioning function block (XPM)

No.	Function Block	Description	Remarks
1	XPM_ORG	Return to original point	-
2	XPM_FLT	Floating original point setting	-
3	XPM_DST	Direct run	-
4	XPM_IST	Indirect run	-
5	XPM_SST	Simultaneous run	-
6	XPM_VTP	Speed/position control conversion	-
7	XPM_VTPP	Position specified speed/position control conversion	
8	XPM_PTV	Position/speed control conversion	-
9	XPM_PTT	Position/torque control conversion	XGF-PN8A/B
10	XPM_STP	Decelerating stop	-
11	XPM_SKP	Skip run	-
12	XPM_SSP	Position synchronization	-
13	XPM_SSS	Speed synchronization	-
14	XPM_SSSP	Position specified speed synchronization	
15	XPM_POR	Position override	-
16	XPM_SOR	Speed override	-
B 17	XPM_PSO	Positioning speed override	-
18	XPM_NMV	Continuous run	-
19	XPM_INC	Inching run	-
20	XPM_RTP	Return run to the previous position of manual operation	-
21	XPM_SNS	Run step no. change	-
22	XPM_SRS	Repeat step no. change	-
23	XPM_MOF	M code cancel	-
24	XPM_PRS	Present position preset	-
25	XPM_EPRE	Encoder value preset	-
26	XPM_ATEA	Plural teaching(ROM, RAM)	-
27	XPM_SBP	Basic parameter setting	-
28	XPM_SEP	Extension parameter setting	-
29	XPM_SHP	Original point return parameter setting	XPM
30	XPM_SMP	Manual operation parameter setting	-

No.	Function Block	Description	Remarks
31	XPM_SIP	Input signal parameter setting	XPM
32	XPM_SCP	Common parameter setting	-
33	XPM_SMD	Operation data setting	-
34	XPM_EMG	Emergency stop	-
35	XPM_RST	Error reset/output prohibition cancel	-
36	XPM_HRST	Error history reset	
37	XPM_PST	Point run	-
38	XPM_WRT	Saving parameter/run data	-
39	XPM_CRD	Reading operation information	-
40	XPM_SRD	Reading operation state	-
41	XPM_ENCRD	Reading encoder value	-
42	XPM_SVERD	Reading servo error information	XGF-PN8A/B
43	XPM_JOG	Jog run	-
44	XPM_CAM	CAM run	-
45	XPM_CAMD	Main axis option de specified CAM run	-
46	XPM_ELIN	Ellipse interpolation	-
47	XPM_VRD	Read variable data	-
48	XPM_VWR	Write variable data	-
49	XPM_ECON	Connect servo communication	XGF-PN8A/B
50	XPM_DCON	Disconnect servo communication	XGF-PN8A/B
51	XPM_SVON	Servo on	XGF-PN8A/B
52	XPM_SVOFF	Servo off	XGF-PN8A/B
53	XPM_SRST	Reset servo error	XGF-PN8A/B
54	XPM_SHRST	Reset servo error history	XGF-PN8A/B
55	XPM_RSTR	Restart	-
56	XPM_POE	Setting position output allowed / prohibited	XPM
57	XPM_TRQ	Torque control	XGF-PN8A/B
58	XPM_SVIRD	Servo external input information read	XGF-PN8B
59	XPM_SVPRD	Servo parameter read	XGF-PN8B
60	XPM_SVPWR	Servo parameter write	XGF-PN8B
61	XPM_SVSAVE	Servo parameter save	XGF-PN8B
62	XPM_PTT	Position/torque switching control	XGF-PN8A/B

No.	Function Block	Description	Remarks
63	XPM_LRD	Latch position data read	XGF-PN8A/B
64	XPM_LCLR	Latch reset	XGF-PN8A/B
65	XPM_LSET	Latch set	XGF-PN8B
66	XPM_STC	Torque synchronization	XGF-PN8A/B
67	XPM_PHASING	Phase Compensation	XGF-PN8A/B
68	XPM_SSSD	32bit Speed Synchronization	XGF-PN8A/B
69	XPM_SSSPD	32bit Speed Synchronization with Position	XGF-PN8A/B
70	XPM_SETOVR	Velocity/Acceration/Decceleration Override	XGF-PN8A/B
71	XPM_CAMA	Absolute Position CAM Run	XGF-PN8A/B

6.5 Expanded function

No.	Function Block	Description	Remarks
1	FOR	Repeat a block of FOR ~ NEXT n times	-
2	NEXT		-
3	BREAK	Escape a block of FOR ~ NEXT	-
4	CALL	Call a SBRT routine	-
5	SBRT	Assign a routine to be called by the CALL function	-
6	RET	RETURN	-
7	JMP	Jump to a place of LABLE	-
8	INIT_DONE	Terminate an initial task	-
9	END	Terminate a program	-

6.6 Motion Function Block

NO.	Function Block	Description	Remarks
Single Axis Motion Command			
1	MC_Power	Servo On/Off	-
2	MC_Home	Perform the search home	-
3	MC_Stop	Stop immediately	-
4	MC_Halt	Stop	-
5	MC_MoveAbsolute	Absolute positioning operation	-
6	MC_MoveRelative	Relative positioning operation	-
7	MC_MoveAdditive	Additive positioning operation	-
8	MC_MoveVelocity	Specified velocity operation	-
9	MC_MoveContinuousAbsolute	Absolute position operation ending with specified velocity operation	-

NO.	Function Block	Description	Remarks
10	MC_MoveContinuousRelative	Relative position operation ending with specified velocity operation	-
11	MC_TorqueControl	Torque control	-
12	MC_SetPosition	Setting the current position	-
13	MC_SetOverride	Velocity/Acceleration override	-
14	MC_ReadParameter	Read Parameter	-
15	MC_WriteParameter	Write Parameter	-
16	MC_Reset	Reset axis error	-
17	MC_TouchProbe	Touch probe	-
18	MC_AbortTrigger	Abort trigger events	-
19	MC_MoveSuperImposed	SuperImposed operation	-
20	MC_HaltSuperImposed	SuperImposed operation halt	-
Multiple Axes Motion Command			
21	MC_CamIn	Camming run	-
22	MC_CamOut	Camming stop	-
23	MC_GearIn	Electrical gearing run	-
24	MC_GearOut	Electrical gearing disengage	-
25	MC_GearInPos	Electrical gearing by specifying the position	-
26	MC_Phasing	Phase compensation	-
Group Motion Command			
27	MC_AddAxisToGroup	Adds one axis to a group in a structure AxesGroup	-
28	MC_RemoveAxisFromGroup	Removes one axis to a group in a structure AxesGroup	-
29	MC_UngroupAllAxes	Removes all axes from the group AxesGroup	-
30	MC_GroupEnable	Changes the state for a group from GroupDisabled to GroupEnable	-
31	MC_GroupDisable	Changes the state for a group to GroupDisabled	-
32	MC_GroupHome	The AxesGroup to perform the search home sequence	-
33	MC_GroupSetPosition	Sets the Position of all axes in a group without moving	-
34	MC_GroupStop	Stop a Group immediately	-
35	MC_GroupHalt	Stop a Group	-
36	MC_GroupReset	Reset a group error	-
37	MC_MoveLinearAbsolute	Absolute positioning linear interpolation operation	-
38	MC_MoveLinearRelative	Relative positioning linear interpolation operation	-
39	MC_MoveCircularAbsolute	Absolute positioning circular interpolation operation	-
40	MC_MoveCircularRelative	Relative positioning circular interpolation operation	-
41	LS_Connect	Connect servo drives	-
42	LS_Disconnect	Disconnect servo drives	-
43	LS_ReadSDO	Read SDO	-
44	LS_WriteSDO	Write SDO	-
45	LS_SaveSDO	Save SDO	-
46	LS_EncoderPreset	Encoder preset	-
47	LS_Jog	JOG operation	-
48	LS_ReadCamData	Read CAM data	-
49	LS_WriteCamData	Write CAM data	-

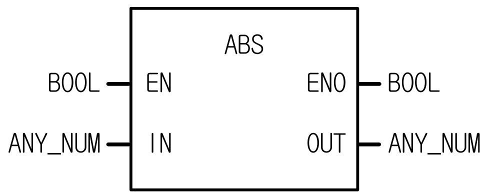
NO.	Function Block	Description	Remarks
50	LS_ReadEsc	Read ESC	-
51	LS_WriteEsc	Write ESC	-
52	LS_CamSkip	Skip CAM	-
53	LS_VarCamIn	Variable CAM operation	-
54	LS_VarGearIn	Variable gear operation	-
55	LS_VarGearInPos	Variable positioning gear operation	-
56	LS_ReadCAM tableSlavePos	Read the slave location of the CAM table	-
57	LS_InverterWriteVel	Write inverter speed	-
58	LS_InverterReadVel	Read inverter speed	-
59	LS_InverterControl	Write inverter control word	-
60	LS_InverterStatus1	Read inverter status 1	-
61	LS_InverterStatus2	Read inverter status 1	-
62	LS_SyncMoveVelocity	Speed control operation (csv mode)	-
63	LS_ReadCamTableMasterPos	Read the Master Location of the CAM table	-
64	LS_OnOffCam	Switch CAM table for on, off or skip operation	-
65	LS_RotaryKnifeCamGen	Generate rotary cutter CAM profile	-
66	LS_CrossSealCamGen	Generate cross sealer CAM profile	-
67	LS_OnOffCamEx	Extended Switch CAM table for on, off or skip operation	
Coordinate System Command			
68	MC_SetKinTransform	Machine information setting	-
69	MC_SetCartesianTransform	PCS setting	-
70	LS_SetWorkSpace	Work space setting	-
71	LS_MoveLinearTimeAbsolute	Time- linear interpolation operation for abolute position of coordinate system	-
72	LS_MoveLinearTimeRelative	Time- linear interpolation operation for relative position of coordinate system	-
73	MC_MoveCircularAbsolute2D	Circular interpolation operation for absolute position of coordinate system	-
74	MC_MoveCircularRelative2D	Circular interpolation operation for relative position of coordinate system	-
75	MC_TrackConveyorBelt	Synchronization setting of the conveyor belt	-
76	MC_TrackRotary table	Synchronization setting of the rotary table	-
77	LS_RobotJOG	JOG operation of the coordinate system	-
78	LS_SetMovePath	Set path operation data	-
79	LS_ResetMovePath	Delete path operation data	-
80	LS_GetMovePath	Read path operation data	-
81	LS_RunMovePath	Perform path operation	-
NC Control Commands			
82	NC_LoadProgram	Specify NC program	-
83	NC_BlockControl	Specify Block operation	-
84	NC_Reset	reset	-
85	NC_Emergency	Emergency stop	-
86	NC_CycleStart	Start automatic operation	-
87	NC_FeedHold	Feed Hold	-

NO.	Function Block	Description	Remarks
88	NC_Home	Homing	-
89	NC_RapidTraverseOverride	Rapid traverse override	-
90	NC_CuttingFeedOverride	Cutting feed override	-
91	NC_SpindleOverride	Spindle override	-
92	NC_M codeComplete	M Code operation completed	-
93	NC_ScodeComplete	S Code operation completed	-
94	NC_TcodeComplete	T Code operation completed	-
95	NC_ReadParameter	Read NC parameters	-
96	NC_WriteParameter	Write NC parameters	-
97	NC_RetraceMove	Reverse operation	
98	NC_BlockSkip	Block skip	
99	NC_DryRun	Dry run	
100	NC_ToolMode	Tool escape/return operation	
101	NC_ReadToolMode	Check tool operation mode	
102	NC_MirrorImage	Mirror image	
103	NC_SpindleControl	Spindle operation control	
104	NC_BlockOptionalSkip	Optional block skip	
105	NC_ManualToolComp	Adjust amount manually	
106	NC_ChgSpindleGear	Gear selection signal	
File Commands			
107	FILE_OPEN	Open file in SD memory card	
108	FILE_CLOSE	Close file in SD memory card	
109	FILE_WRITE	Write files to SD memory card	
110	FILE_READ	Reading files in SD memory card	
111	FILE_SEEK	Move SD memory card inside	
Others			
112	PID	PID Operation	-
113	LINAC	Linear Acceration Command 1	-
114	SLINAC	Linear Acceration Command 2	-

Chapter 7. Basic Functions

1. This chapter describes basic functions.
2. Before using basic functions it is recommended to understand 3.4.1 Function and to apply to function library on a program for easy writing a program.

ABS	Absolute value operation	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: input value of absolute value operation **Output** ENO: 1 OUT: absolute value IN, OUT should be the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN						○	○	○	○	○	○	○	○	○	○					
	OUT						○	○	○	○	○	○	○	○	○	○					

■ **Function**

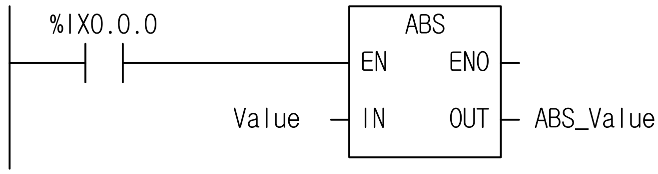
(1) Output the absolute value of IN as ‘OUT’.
OUT = │IN│
(2) X’s absolute value, │X│;
A. If X≥0, │X│= X,
B. If X < 0, │X│= -X.

■ **Flag**

Flag	Description
_ERR	If IN value is (-)min value, _ERR and _LER flags are set. ex) if data type is SINT and IN and value is -128, an error is activated.

■ **Program Example**

1. LD



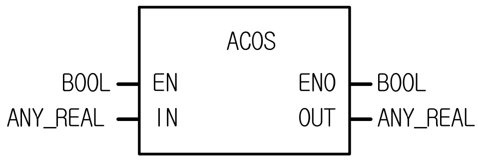
2. ST

ABS_Value := ABS(EN:=%IX0.0.0, IN:=Value);

(1) If the transition condition (%IX0.0.0) is on, ABS function executes.
(2) If VALUE = -7, ABS_VALUE = │-7│ = 7.
If VALUE = 200, ABS_VALUE = │200│ = 200.
(3) The negative number of INT type is represented as the 2's compliment form (refer to 3.2.4. Data type structure)

INPUT (IN) : VALUE (INT)= -7	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	(16#FFF9)
								↓									
OUTPUT (OUT) : ABS_VALUE (INT)= 7	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	(16#0007)

ACOS	**Arc Cosine operation**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: input value of Arc Cosine operation **Output** ENO: outputs EN value as it is OUT: Arc Cosine (radian) IN, OUT must be the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ **Function**

It converts input IN into its Arc Cosine value and produces output OUT. The output range is between 0 and π.

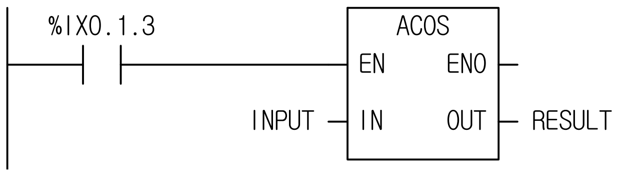
OUT = ACOS (IN)

■ **Flag**

Flag	Description
_ERR	Unless an IN value is between -1.0 and 1.0, _ERR, _LER flags are set.

■ **Program Example**

1) LD



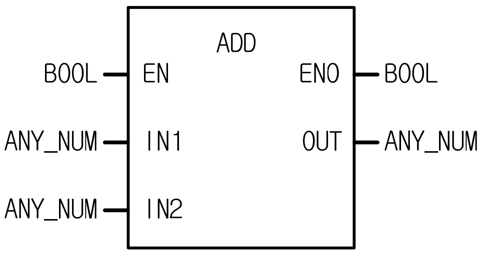
2) ST

RESULT := ACOS(EN:=%IX0.1.3, IN:=INPUT);

(1) If the transition condition (%IX0.1.3) is on, Arc Cosine operation function, ACOS executes

(2) If INPUT is 0.8660... ($\sqrt{3}$ / 2), RESULT will be 0.5235... (π/6 rad = 30°).

ADD	Addition	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: value to be add IN2: value to add Input variable number can be extended up to 8 **Output** ENO: without an error, it is 1 OUT: added value IN1, IN2, ..., OUT must be the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1						○	○	○	○	○	○	○	○	○	○					
	IN2						○	○	○	○	○	○	○	○	○	○					
	OUT						○	○	○	○	○	○	○	○	○	○					

■ **Function**

1. It adds input variables up (IN1, IN2, ..., and INn, n: number of inputs) and produces output ,OUT.

   OUT = IN1 + IN2 + ... + INn

■ **Flag**

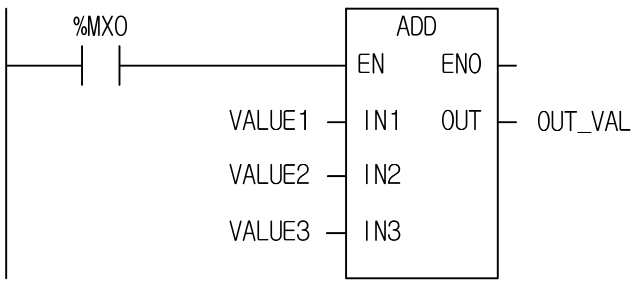
Flag	Description
_ERR	When the output value is out of its data type, _ERR, _LER flags are set.

☆ If REAL (or LREAL) type operation exceeds the max. or min. value of REAL (or LREAL) in the middle of operation because it performs operation sequentially from IN1 to IN8, _ERR, _LER flag are set and the result is unlimited or abnormal value.

(1.#INF000000000000e+000, 1.#SNAN00000000000e+000, 1.#QNAN00000000000e+000).

■ **Program Example**

1) LD



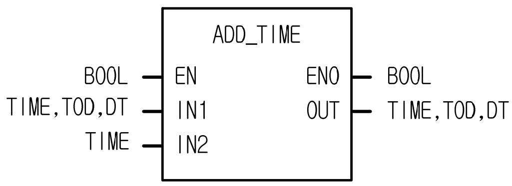
2) ST

OUT_VAL := ADD(EN:=%MX0, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);

(1) If the transition condition (%MX0) is on, ADD function executes
(2) If input variable VALUE1 = 300, VALUE2 = 200, and VALUE3 = 100, output variable OUT_VAL = 300 + 200 + 100 = 600

INPUT (IN1) : VALUE1 (INT) = 300(16#012C)	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0
	+(ADD)															
(IN2) : VALUE2(INT) = 200(16#00C8)	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
	+(ADD)															
(IN3) : VALUE3(INT) = 100(16#0064)	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
	↓															
(OUT): OUT_VAL(INT) = 600(16#0258)	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0

ADD_TIME	Time addition	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: reference time, time of date IN2: time to add **Output** ENO: without an error, it is 1 OUT: added result of TOD or time IN1, IN2, and OUT must be of the same data type: If IN1 type is TIME_OF_DAY, OUT type is also TIME_OF_DAY.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1																○		○	○	
	OUT																○		○	○	

■ **Function**

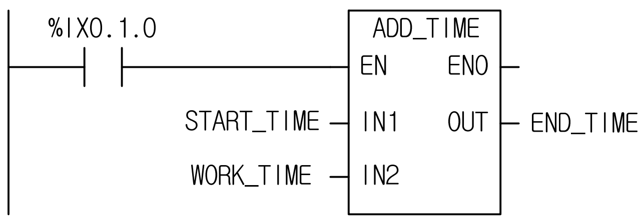
1) If IN1 is TIME, added TIME is an output.
2) IN1 is TIME_OF_DAY, it adds TIME to reference TIME_OF_DAY and produces output TIME_OF_DAY.
3) If IN1 is DATE_AND_TIME, the output data type is DT (Date and Time of Day) adding the time to the standard date and time of day.

■ **Flag**

Flag	Description
_ERR	If an output value is out of range of related data type, _ERR, _LER flag are set. An error occurs: 1) When the result of adding the time and the time is out of range of TIME data type : T#49D17H2M47S295MS 2) The result of adding TOD (Time of Day) and the time exceeds 24h; 3) The result of adding the date and DT (Date and the Time of Day) exceeds the year, 2163.

■ **Program Example**

1) LD



2) ST

```
END_TIME := ADD_TIME(EN:= %IX0.1.0, IN1:= START_TIME, IN2:= WORK_TIME);
```

(1) If the transition condition (%IX0.1.0) is on, ADD_TIME function is executes.
(2) If START_TIME is TOD#08:30:00 and WORK_TIME is T#2H10M20S500MS, END_TIME is TOD#10:40:20.5.

INPUT (IN1) : START_TIME (TOD) = TOD#08:30:00

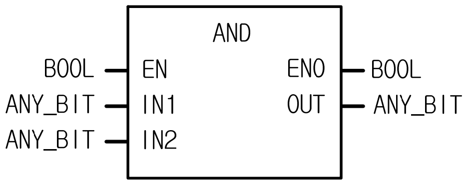
+ (ADD_TIME)

(IN2) : WORK_TIME(TIME) = T#2H10M20S500MS

↓

OUTPUT (OUT) : END_TIME (TOD) = TOD#10:40:20.5

AND	**Logical AND (Logical multiplication)**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN1: input 1 IN2: input 2 Input variables can be extended up to 8. **Output** ENO: outputs EN value as it is OUT: AND result IN1, IN2, and OUT must be of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○															
	IN2	○	○	○	○	○															
	OUT	○	○	○	○	○															

■ **Function**

It performs a logical AND operation on the input variables by bit and produces output ,OUT.

IN1 1111 0000

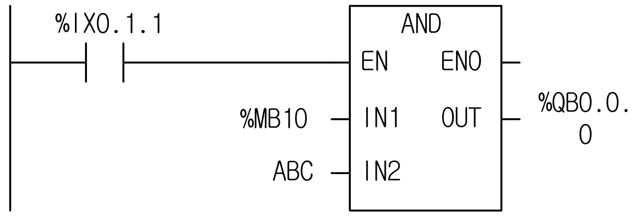
&

IN2 1010 1010

OUT 1010 0000

■ **Program Example**

1. LD



2. ST

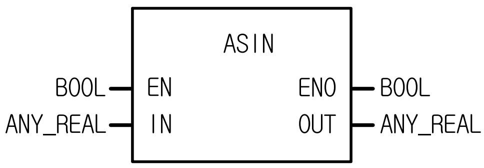
ST doesn't support AND.

In case of AND2_BYTE

%QB0.0.0 := AND2_BYTE(EN:=%IX0.1.1, IN1:= %MB10, IN2:= ABC);

(1) If the transition condition (%IX0.1.1) is on, the AND function executes.
(2) If INI = %MB10 and IN2 = ABC, the result of AND is shown in OUT (%QB0.0.0).

ASIN	Arc Sine operation	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: input value of Arc Sine operation **Output** ENO: outputs EN value as it is OUT: radian output value after Arc Sine operation IN and OUT must be of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ **Function**

It produces an output (Arc Sine value) of IN. The output value is between -π/2 and π/2.

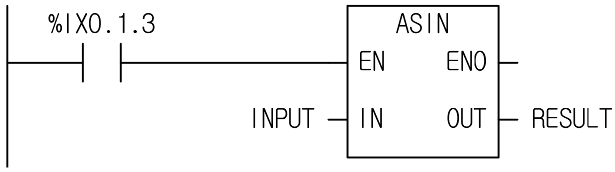
OUT = ASIN (IN)

■ **Error**

Flag	Description
_ERR	If an input value exceeds the range from -1.0 to 1.0, _ERR and _LER flags are set.

■ **Program Example**

1. LD



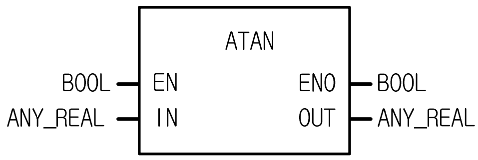
2. ST

RESULT := ASIN(EN:=%IX0.1.3, IN1:= INPUT);

(1) If the transition condition (%IX0.1.3) is on, ASIN function executes.

(2) If INPUT variable is 0.8660.... ($\sqrt{3}$ /2), the RESULT will be 1.0471.... (π/3 radian = 60°).

ATAN	Arc Tangent operation	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN: Input value of Arc Tangent operation **Output** ENO: outputs EN value as it is OUT: radian output value after Arc Tangent operation IN, OUT must be of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

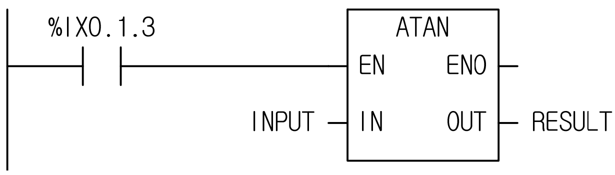
■ **Function**

It produces an output (Arc Tangent value) of IN value. The output value is between -π/2 and π/2.

OUT = ATAN (IN)

■ **Program Example**

1. LD



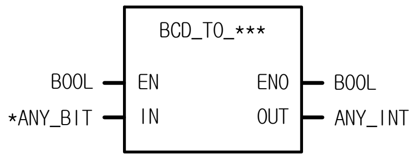
2. ST

RESULT := ATAN(EN:=%IX0.1.3, IN1:= INPUT);

(1) If the transition condition (%IX0.1.3) is on, ATAN function executes.

(2) If INPUT = 1.0, then output RESULT will be 0.7853... (π/4 rad = 45°).

BCD_TO_***	Converts BCD data into an integer number	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: ANY_BIT (BCD) **Output** ENO: outputs EN value as it is OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT						○	○	○	○	○	○	○	○							

*ANY_BIT : exclude BOOL from ANY_BIT type.

■ **Function**

It converts input IN type and produces output ,OUT.

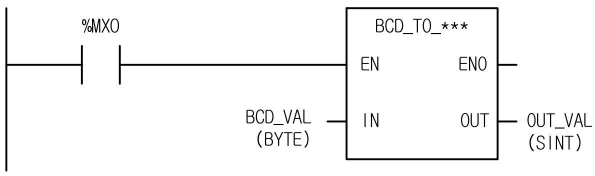
Function	Input type	Output type	Description
BYTE_BCD_TO_SINT	BYTE	SINT	It converts BCD data into an output data type. It coverts only when the input date type is a BCD value. If an input data type is WORD, only the part of its data (0 ~16#9999) is normally converted.
WORD_BCD_TO_INT	WORD	INT	
DWORD_BCD_TO_DINT	DWORD	DINT	
LWORD_BCD_TO_LINT	LWORD	LINT	
BYTE_BCD_TO_USINT	BYTE	USINT	
WORD_BCD_TO_UINT	WORD	UINT	
DWORD_BCD_TO_UDINT	DWORD	UDINT	
LWORD_BCD_TO_ULINT	LWORD	ULINT	

■ **Flag**

Flag	Description
_ERR	If IN is not a BCD data type, then the output will be 0 and _ERR, _LER flags are set.

■ **Program Example**

1. LD



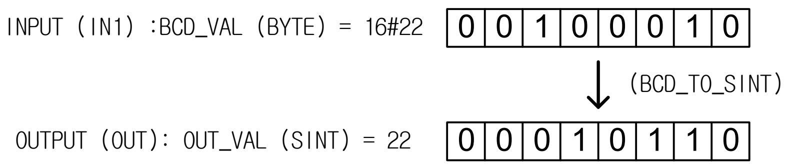
2. ST

ST language doesn't support BCD_TO_***

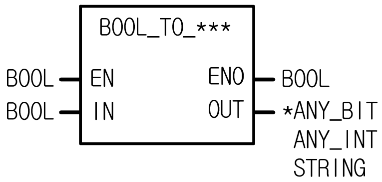
In case of BYTE_BCD_TO_SINT

OUT_VAL := BYTE_BCD_TO_SINT(EN:=%MX0, IN:= BCD_VAL);

(1) If the transition condition (%MX0) is on, BCD_TO_*** function executes.
(2) If BCD_VAL (BYTE) = 16#22 (2#0010_ 0010), then the output variable OUT_VAL (SINT) = 22 (2#0001_ 0110).



BOOL_TO_***	BOOL type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN: bit to convert (1 bit) **Output** ENO: outputs EN value as it is OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT		○	○	○	○	○	○	○	○	○	○	○	○							○

*ANY_BIT: exclude BOOL from ANY_BIT type.

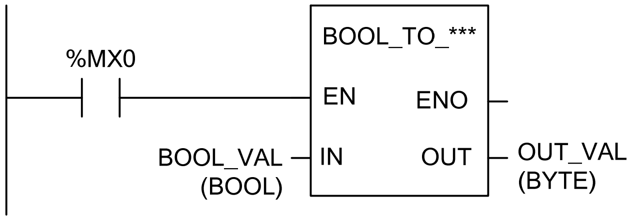
■ **Function**

It converts input IN type and produces output ,OUT.

Function	Output type	Description
BOOL_TO_SINT	SINT	If the input value (BOOL) is 2#0, it produces the integer number '0' and if it is 2#1, it produces the integer number '1' according to the output data type.
BOOL_TO_INT	INT	
BOOL_TO_DINT	DINT	
BOOL_TO_LINT	LINT	
BOOL_TO_USINT	USINT	
BOOL_TO_UINT	UINT	
BOOL_TO_UDINT	UDINT	
BOOL_TO_ULINT	ULINT	
BOOL_TO_BYTE	BYTE	It converts BOOL into the output data type whose upper bits are filled with 0.
BOOL_TO_WORD	WORD	
BOOL_TO_DWORD	DWORD	
BOOL_TO_LWORD	LWORD	
BOOL_TO_STRING	STRING	It converts BOOL into a STRING type, which is '0' or '1'.

■ **Program Example**

1. LD



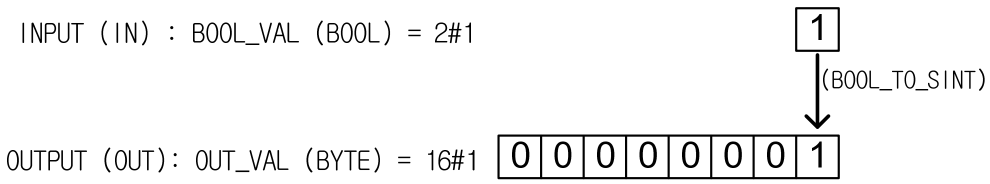
2. ST

ST language doesn't support BOOL_TO_***

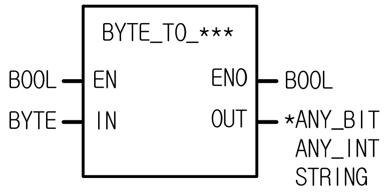
In case of BOOL_TO_BYTE

OUT_VAL := BOOL_TO_BYTE(EN:=%MX0, IN:= BOOL_VAL);

(1) If the transition condition (%MX0) is on, BOOL_TO_*** function executes.
(2) If input BOOL_VAL (BOOL) = 2#1, then output, OUT_VAL (BYTE) = 2#0000_ 0001.



BYTE_TO_***	BYTE type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN: bit String to convert (8 bits) **Output** ENO: outputs EN value as it is OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT		○	○	○	○	○	○	○	○	○	○	○	○							○

*ANY_BIT: exclude BOOL from ANY_BIT type.

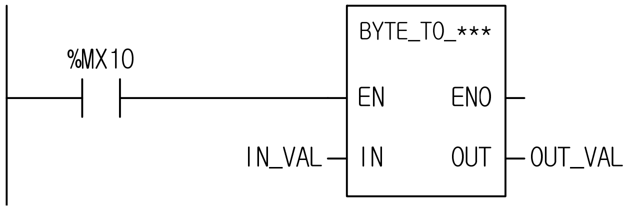
■ **Function**

It converts input IN type and produces output ,OUT.

Function	Output type	Description
BYTE _TO_SINT	SINT	Converts into SINT type without changing its internal bit array.
BYTE _TO_INT	INT	Converts into INT type filling the upper bits with 0.
BYTE _TO_DINT	DINT	Converts into DINT type filling the upper bits with 0.
BYTE _TO_LINT	LINT	Converts into LINT type filling the upper bits with 0.
BYTE _TO_USINT	USINT	Converts into USINT type without changing its internal bit array.
BYTE _TO_UINT	UINT	Converts into UINT type filling the upper bits with 0.
BYTE _TO_UDINT	UDINT	Converts into UDINT type filling the upper bits with 0.
BYTE _TO_ULINT	ULINT	Converts into ULINT type filling the upper bits with 0.
BYTE _TO_BOOL	BOOL	Takes the lower 1 bit and converts it into BOOL type.
BYTE _TO_WORD	WORD	Converts into WORD type filling the upper bits with 0.
BYTE _TO_DWORD	DWORD	Converts into DWORD type filling the upper bits with 0.
BYTE _TO_LWORD	LWORD	Converts into LWORD type filling the upper bits with 0.
BYTE _TO_STRING	STRING	Converts the input type value into STRING.

■ **Program Example**

1. LD



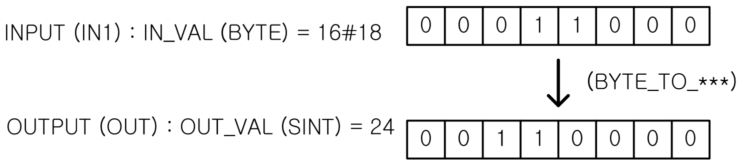
2. ST

ST language doesn't support BYTE_TO_***

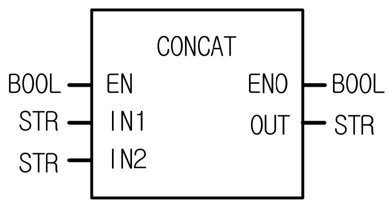
In case of BYTE_TO_SINT

OUT_VAL := BYTE_TO_SINT(EN:=%MX10, IN:= IN_VAL);

(1) If the transition condition (%MX10) is on, BYTE_TO_*** function executes.
(2) If IN_VAL (BYTE) = 2#0001_1000, OUT_VAL (SINT) = 24 (2#0011_0000).



CONCAT	Concatenates a String	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: input String IN2: input String Input variable number can be extended up to 8. **Output** ENO: without an error, it is 1. OUT: output String

■ **Function**

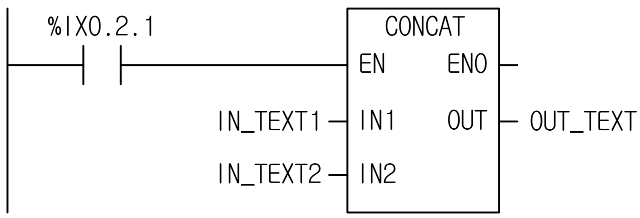
It concatenates the input String IN1, IN2, IN3, …, INn (n: number of inputs) in order and produces output String OUT.

■ **Flag**

Flag	Description
_ERR	If the sum of character number of each input String is greater than 31, then the output CONCAT is the concatenate String of each input String (up to 31 letters), and _ERR, _LER flags are set.

■ **Program Example**

1. LD



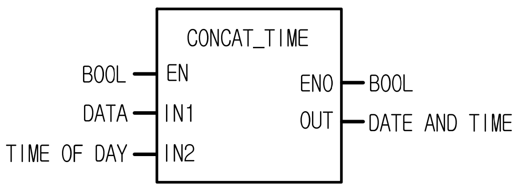
2. ST

OUT_TEXT := CONCAT(EN:=%IX0.2.1, IN1:= IN_TEXT1, IN2:= IN_TEXT2);

(1) If the transition condition (%IX0.2.1) is on, CONCAT function executes.
(2) If input variable IN_TEXT1 = 'ABCD' and IN_TEXT2 = 'DEF', then OUT_TEXT = 'ABCDDEF'.

```
  INPUT (IN1) : IN_TEXT1 (STRING) =    `ABCD`
                                      (CONCAT)
        (IN2) : IN_TEXT2(STRING) =     `DEF`
                                         ↓
OUTPUT (OUT) : OUT_TEXT (STRING) =   'ABCDDEF'
```

CONCAT_TIME	Concatenates date and time of day	
	Availability	XGI, XGR, XEC, XMC
	Flags	

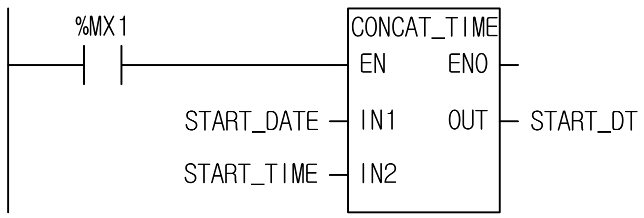
Function	Description
	**Input** EN: executes the function in case of 1 IN1: date data input IN2: Time of day data input **Output** ENO: outputs EN value as it is OUT: DT (Date and Time of Day) output

■ **Function**

It concatenates IN1 (date) and IN2 (time of day) and produces output, OUT (DT).

■ **Program Example**

1. LD



2. ST

START_DT := CONCAT_TIME(EN:=%MX1, IN1:= START_DATE, IN2:= START_TIME);

(1) If the transition condition (%MX1) is on, CONCAT_TIME function executes.
(2) If START_DATE = D#1995-12-06 and START_TIME = TOD#08:30:00, then, output START_DT = DT#1995-12-06-08:30:00.

INPUT (IN1) : START_DATE (DATE) = D#1995-12-06

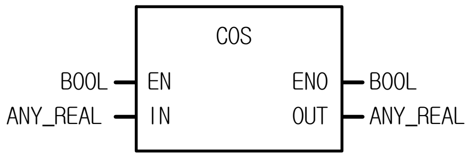
(CONCAT_TIME)

INPUT (IN2) : START_TIME (TOD) = TOD#08:30:00

↓

OUTPUT (OUT) : START_DT (DT) = DT#1995-12-06-08:30:00

COS	**Cosine operation**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN: radian input value of Cosine operation **Output** ENO: outputs EN value as it is OUT: result value of Cosine operation IN and OUT must be the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

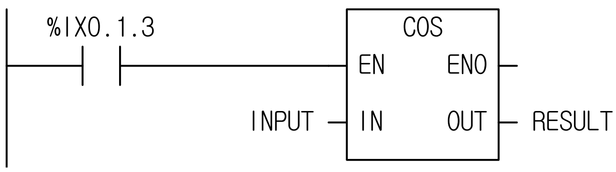
■ **Function**

It produces IN's Cosine operation value.

OUT = COS (IN)

■ **Program Example**

1. LD



2. ST

RESULT := COS(EN:=%IX0.1.3, IN:= INPUT);

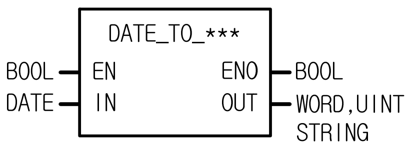
(1) If the transition condition (%IX0.1.3) is on, COS function executes.

(2) If input INPUT = 0.5235 (π/6 rad = 30°), output RESULT = 0.8660 ... ($\sqrt{3/2}$).

COS (π/6) = $\sqrt{3/2}$ = 0.866

```
    INPUT (IN) : INPUT (REAL)  =     0.5235
                                   ↓ (COS)
OUTPUT (OUT) : RESULT (REAL) = 8.66074800E-01
```

DATE_TO_***	Date type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN: date data to convert **Output** ENO: outputs EN value as it is OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN			○								○									○

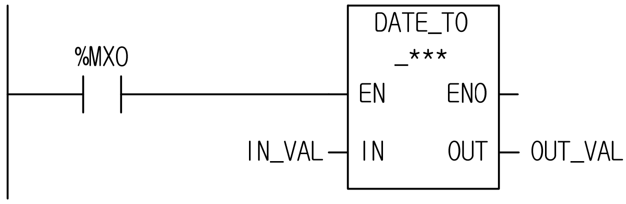
■ **Function**

It converts an input IN type and produces output, OUT.

Function	Output type	Description
DATE_TO_UINT	UINT	Converts DATE into UINT type.
DATE_TO_WORD	WORD	Converts DATE into WORD type.
DATE_TO_STRING	STRING	Converts DATE into STRING type.

■ **Program Example**

1. LD



2. ST

ST language doesn't support DATE_TO_****

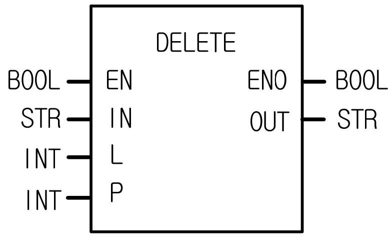
In case of DATE_TO_STRING

OUT_VAL := DATE_TO_STRING(EN:=%MX0, IN:= IN_VAL);

(1) If the transition condition (%MX0) is on, DATE_TO_*** function executes.
(2) If IN_VAL (DATE) = D#1995-12-01, OUT_VAL (STRING) = D#1995-12-01.

```
INPUT (IN) : IN_VAL (DATE)    =  D#1995-12-01
                                        ↓ (DATE_TO_STRING)
OUTPUT (OUT) : OUT_VAL (STRING) = 'D#1995-12-01'
```

DELETE	Delete a string	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: input String L: length of String to delete P: position of String to delete **Output** ENO: without an error, it is 1 OUT: output String

■ **Function**

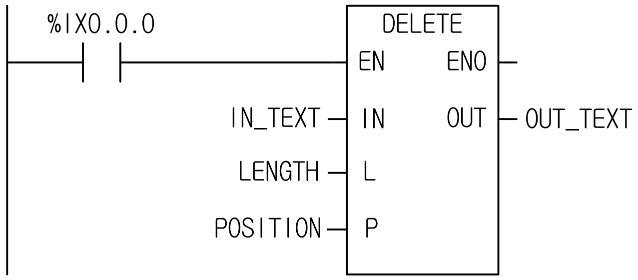
After deleting a String (L) from the P character of IN, produces output, OUT.

■ **Flag**

Flag	Description
_ERR	If P $\leq$ 0 or L < 0, or if P > character number of IN, _ERR and _LER flags are set.

■ **Program Example**

1. LD



2. ST

OUT_TEXT := DELETE(EN:= %IX0.0.0, IN:= IN_TEXT, L:= LENGTH, P:= POSITION);

(1) If the transition condition (%IX0.0.0) is on, DELETE function executes.
(2) If input variable IN_TEXT = 'ABCDEF', LENGTH = 3, and POSITION = 3, then OUT_TEXT (STRING) will be 'ABF'.

```
INPUT (IN) : IN_TEXT (STRING) =  `ABCDEF`

   (L)  : LENGTH(INT)         =    3

   (P)  : POSITION(INT)       =    3
                                 ↓ (DELETE)
OUTPUT (OUT): OUT_TEXT (STRING)  =  `ABF`
```

DINT_TO_***	DINT type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
DINT_TO_*** BOOL — EN ENO — BOOL DINT — IN OUT — *ANY	**Input** EN: executes the function in case of 1 IN: double integer value to convert **Output** ENO: without an error, it is 1. OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○	○		○	○	○	○	○	○	○			○	○	○

*ANY: exclude DINT, TIME and DATE from ANY type.

■ **Function**

It converts Input IN type and produces output, OUT.

Function	Output type	Description
DINT_TO_SINT	SINT	If input is -128 ~ 127, normal conversion. Except this, an error occurs.
DINT_TO_INT	INT	If input is -32,768 ~ 32,767, normal conversion. Except this, an error occurs.
DINT_TO_LINT	LINT	Converts normally into LINT type.
DINT_TO_USINT	USINT	If input is 0 ~ 255, normal conversion. Otherwise an error occurs.
DINT_TO_UINT	UINT	If input is 0 ~ 65,535, normal conversion. Otherwise an error occurs.
DINT_TO_UDINT	UDINT	If input is 0 ~ 2,147,483,647, normal conversion. Otherwise an error occurs.
DINT_TO_ULINT	ULINT	If input is 0 ~ 2,147,483,647, normal conversion. Otherwise an error occurs.
DINT_TO_BOOL	BOOL	Takes the low 1 bit and converts into BOOL type.
DINT_TO_BYTE	BYTE	Takes the low 8 bit and converts into BYTE type.

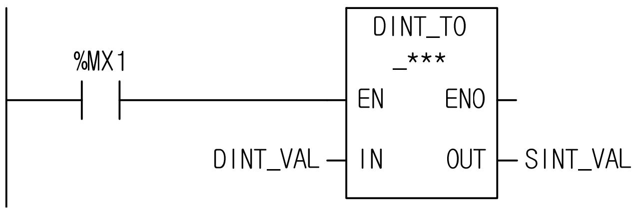
Function	Output type	Description
DINT_TO_WORD	WORD	Takes the low 16 bit and converts into WORD type.
DINT_TO_DWORD	DWORD	Converts into DWORD type without changing the internal bit array.
DINT_TO_LWORD	LWORD	Converts into LWORD type filling the upper bytes with 0.
DINT_TO_REAL	REAL	Converts DINT into REAL type. During conversion, an error caused by the precision may occur.
DINT_TO_LREAL	LREAL	Converts DINT into LREAL type. During conversion, an error caused by the precision may occur.
DINT_TO_STRING	STRING	Converts the input value into STRING type.

■ **Flag**

Flag	Description
_ERR	If a conversion error occurs, _ERR, _LER flags are set. When an error occurs, it takes as many lower bits as the bit number of the output type and produces an output without changing the internal bit array.

■ **Program Example**

1. LD



2. ST

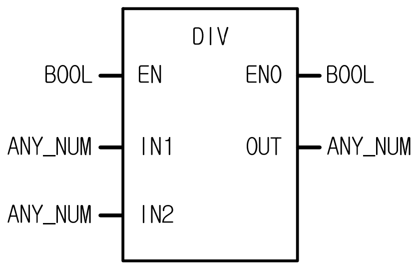
ST language doesn't support DINT_TO_***

In case of DINT_TO_SINT

```
SINT_VAL := DINT_TO_SINT(EN:= %MX1, IN:= DINT_VAL);
```

(1) If the transition condition (%MX1) is on, DINT_TO_*** function executes.
(2) If INI = DINT_VAL (DINT) = -77, SINT_VAL (SINT) = -77.

DIV	**Division**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: the value to be divided (dividend) IN2: the value to divide (divisor) **Output** ENO: without an error, it is 1. OUT: the divided result (quotient) The variable connected to IN1, IN2 and OUT must be of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1						○	○	○	○	○	○	○	○	○	○					
	IN2						○	○	○	○	○	○	○	○	○	○					
	OUT						○	○	○	○	○	○	○	○	○	○					

■ **Function**

It divides IN1by IN2 and produces an output omitting decimal fraction from the quotient.

OUT = IN1/IN2

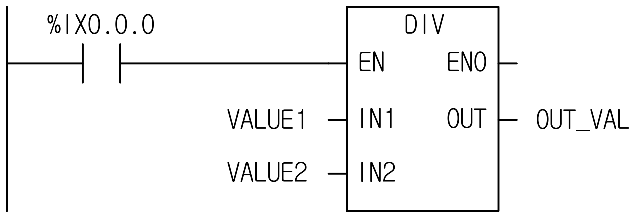
IN1	IN2	OUT	Remarks
7	2	3	Decimal fraction omitted
7	-2	-3	
-7	2	-3	
-7	-2	3	
7	0	×	Error

■ **Flag**

Flag	Description
_ERR	If the value to divide (divisor) is ‘0’, and the results exceeds the maximum value of each type, _ERR, _LER flags are set.

■ **Program Example**

1. LD



2. ST

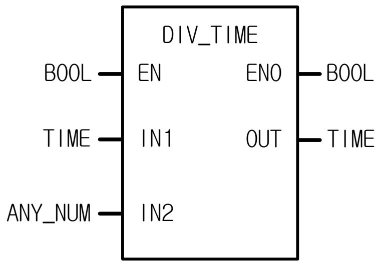
```
OUT_VAL := DIV(EN:= %IX0.0.0, IN1:= VALUE1, IN2:= VALUE2);
```

(1) If the transition condition (%IX0.0.0) is on, DIV function executes.

(2) If input VALUE1 = 300 and VALUE2 = 100, then output, OUT_VAL = 300/100 = 3.

INPUT (IN1) : VALUE1 (INT) = 300(16#012C)	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0
/(DIV)																
(IN2): VALUE2(INT) = 100(16#0064)	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
↓																
OUTPUT (OUT) : OUT_VAL (INT) = 3(16#3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

DIV_TIME	Time division	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	Input EN: executes the function in case of 1 IN1: Time to divide IN2: The value to divide Output ENO: without an error, it is 1. OUT: divided result time

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN2						○	○	○	○	○	○	○	○	○	○					

■ **Function**

1. It divides IN1 (time) by IN2 (number) and produces output OUT (divided time).

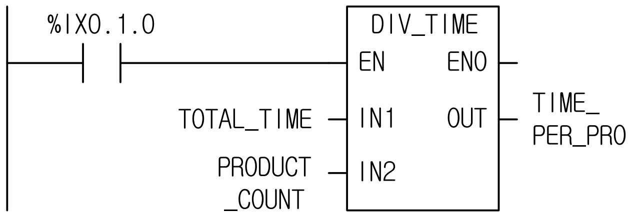
■ **Flag**

Flag	Description
_ERR	If a divisor (IN2) is 0 or less than 0, _ERR and _LER flags are set. If a negative number is entered into IN2, _ERR and _LER flags are on and the outputs is 0.

■ **Program Example**

This is the program that calculates the time required to produce one product in some product line if the working time of day is 12hr 24min 24sec and product quantity of a day is 12 in a product line.

1. LD



2. ST

TIME_PER_PRO := DIV_TIME(EN:= %IX0.1.0, IN1:= TOTAL_TIME, IN2:= PRODUCT_COUNT);

(1) If the transition condition (%IX0.1.0) is on, DIV_TIME function executes.

(2) If it divides TOTAL_TIME (T#12H24M24S) by PRODUCT_COUNT (12), the time required to produce one product TIME_PER_PRO (T#1H2M2S) is an output. That is, it takes 1hr: 2min :2sec to produce one product.

INPUT (IN1) : TOTAL_TIME (TIME) = T#12H24M24S

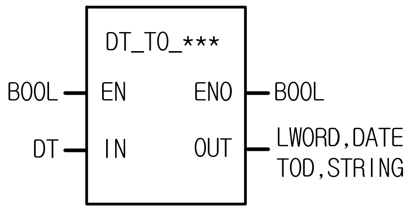
/(DIV_TIME)

(IN2) : PRODUCT_COUNT(INT) = 12

↓

OUTPUT (OUT) : TIME_PER_PRO (TIME) = T#1H2M2S

DT_TO_***	DT type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: date and time of day data to convert **Output** ENO: outputs EN value as it is OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT					○												○	○		○

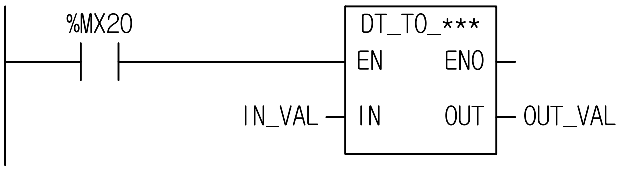
■ **Function**

It converts Input IN type and produces output, OUT.

Function	Output type	Description
DT_TO_LWORD	LWORD	Converts DT into LWORD type. (The inverse conversion is available as there is no internal data change).
DT_TO_DATE	DATE	Converts DT into DATE type.
DT_TO_TOD	TOD	Converts DT into TOD type.
DT_TO_STRING	STRING	Converts DT into STRING type.

■ **Program Example**

1. LD



2. ST

ST language doesn't support DT_TO_***

In case of DT_TO_DATE

OUT_VAL := DT_TO_DATE(EN:= %MX20, IN1:= IN_VAL);

(1) If the transition condition (%MX20) is on, DT_TO_*** function executes.

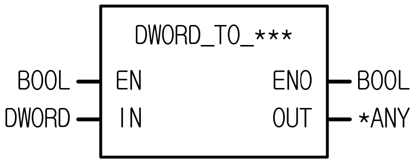
(2) If input IN_VAL (DT) = DT#1995-12-01-12:00:00, output ,OUT_VAL (DATE) = D#1995-12-01

INPUT (IN) : IN_VAL (DT) = DT#1995-12-01-12:00:00

↓ (DT_TO_DATE)

OUTPUT (OUT) : OUT_VAL (DATE) = D#1995-12-01

DWORD_TO_***	DWORD type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: bit String to convert (32bit) **Output** ENO: outputs EN value as it is OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○		○	○	○	○	○	○	○	○	○	○		○		○	○	○

*ANY: exclude DWORD, LREAL and DATE from ANY type.

■ **Function**

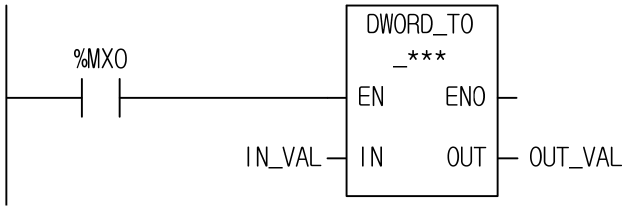
It converts Input IN type and produces output. OUT.

Function	Output type	Description
DWORD_TO_SINT	SINT	Takes the lower 8 bits and converts into SINT type.
DWORD_TO_INT	INT	Takes the lower 16 bits and converts into INT type.
DWORD_TO_DINT	DINT	Converts into DINT type without changing the internal bit array.
DWORD_TO_LINT	LINT	Converts into LINT type filling the upper bits with 0
DWORD_TO_USINT	USINT	Takes the lower 8 bits and converts into USINT type.
DWORD_TO_UINT	UINT	Takes the lower 16 bits and converts into UINT type.
DWORD_TO_UDINT	UDINT	Converts into UDINT type without changing the internal bit array.
DWORD_TO_ULINT	ULINT	Converts into ULINT type filling the upper bits with 0.
DWORD_TO_BOOL	BOOL	Takes the lower 1 bit and converts into BOOL type.
DWORD_TO_BYTE	BYTE	Takes the lower 8 bits and converts into BYTE type.
DWORD_TO_WORD	WORD	Takes the lower 16 bits and converts into WORD type.
DWORD_TO_LWORD	LWORD	Converts into LWORD type filling the upper bits with 0.
DWORD_TO_REAL	REAL	Converts into REAL type without changing the internal bit array.
DWORD_TO_TIME	TIME	Converts into TIME type without changing the internal bit array.

Function	Output type	Description
DWORD_TO_TOD	TOD	Converts into TOD type without changing the internal bit array. However, with a value out of TOD range (TOD#23:59:59.999), _ERR, _LER flags are set and it is alternately converted within the range of TOD.
DWORD_TO_STRING	STRING	Changes input value into decimal and converts into STRING type.

■ **Program Example**

1. LD



2. ST

ST language doesn't support DWORD_TO_***

In case of DWORD_TO_TOD

OUT_VAL := DWORD_TO_***(EN:= %MX0, IN1:= IN_VAL);

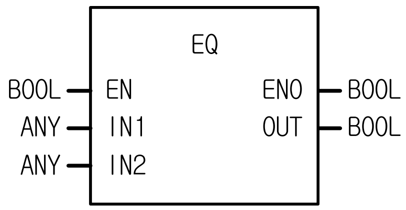
(1) If the transition condition (%MX0) is on, DWIRD_TO_TOD function executes.

(2) If output IN_VAL (DWORD) = 16#3E8 (1000), output , OUT_VAL (TOD) = TOD#1S.

(3) Calculates TIME, TOD by converting decimal into MS unit. That is, 1000 is 1000ms = 1s.

(Refer to 3.2.4. Data Type Structure)

EQ	**'Equal to' comparison**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN1: the value to be compared IN2: the value to compare Input variable number can be extended up to 8. IN1, IN2, ... must be the same type. **Output** ENO: outputs EN value as it is OUT: comparison result value

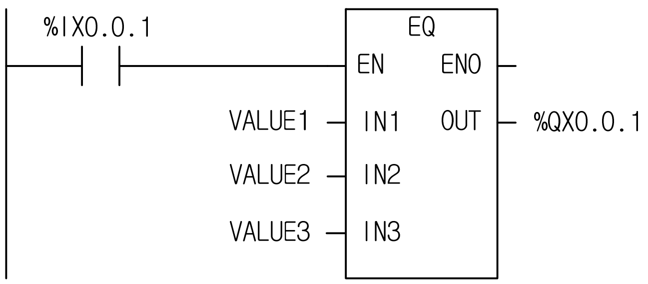
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ **Function**

1. If IN1 = IN2 = IN3 ... = INn (n : number of inputs), output , OUT is 1.
2. In other cases, OUT is 0.

■ **Program Example**

1. LD



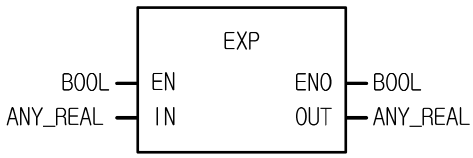
2. ST

%QX0.0.1 := EQ(EN:= %IX0.0.1, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);

(1) If the transition condition (%IX0.0.1) is on, EQ function executes.
(2) If VALUE1 = 300, VALUE2 = 300, VALUE3 = 300 (comparison result VALUE1 = VALUE2 = VALUE3), output %QX0.0.1 = 1.

INPUT (IN1) : VALUE1 (INT)= 300(16#012C)	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0
	=(EQ)															
(IN2) : VALUE2(INT)= 300(16#012C)	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0
	=(EQ)															
(IN3) : VALUE3(INT)= 300(16#012C)	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0
OUTPUT (OUT) : %QX0.0.1 (BOOL)= 1(16#1)																1

EXP	**EXP operation**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: input value of exponent operation **Output** ENO: outputs EN value as it is OUT: result value of exponent operation IN, OUT must be of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ **Function**

It calculates the natural exponent with exponent IN and produces output, OUT.

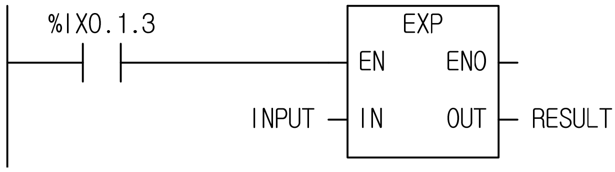
OUT = e^{IN}

■ **Error**

Flag	Description
_ERR	If output is out of the range of a type, _ERR and _LER flags are set.

■ **Program Example**

1. LD



2. ST

RESULT := EXP(EN:= %IX0.1.3, IN1:= INPUT);

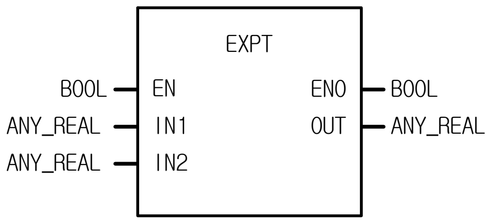
(1) If the transition condition (%IX0.1.3) is on, EXP function executes.

(2) If INPUT is 2.0, RESULT is 7.3890….

RESULT = e^{INPUT}

INPUT = 2.0, RESULT = 7.3890...

EXPT	**Exponential operation**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: real number IN2: exponent **Output** ENO: outputs EN value as it is OUT: result value IN1 and OUT must be of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1														○	○					
	IN2														○	○					
	OUT														○	○					

■ **Function**

It calculates IN1 with exponent IN2 and produces output, OUT.

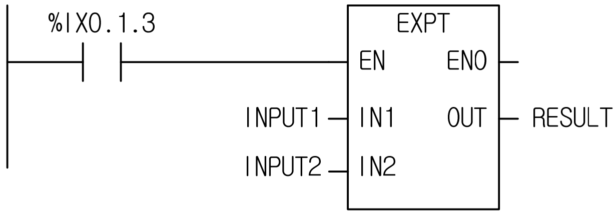
OUT = $IN1^{IN2}$

■ **Error**

Flag	Description
_ERR	If an output is out of range of related data type, _ERR and _LER flags are set.

■ **Program Example**

1. LD



2. ST

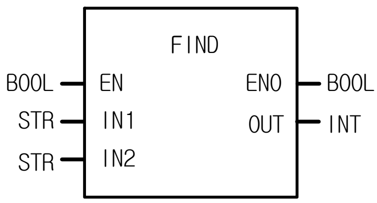
```
RESULT := EXPT(EN:= %IX0.1.2, IN1:= INPUT1, IN2:= INPUT2);
```

(1) If the transition condition (%IX0.1.3) is on, 'EXPT' exponential function executes.

(2) If input INPUT1= 1.5, INPUT2 = 3, output RESULT = 1.5^3 = 1.5 ×1.5 ×1.5 = 3.375.

$$3.375 = 1.5^3$$

FIND	Find a string	
	Availability	XGI, XGR, XEC, XMC
	Flags	

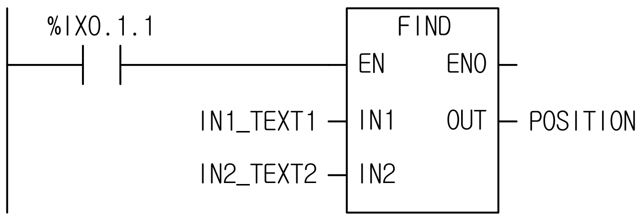
Function	Description
 	**Input** EN: executes the function in case of 1 IN1: input String IN2: String to find **Output** ENO: outputs EN value as it is OUT: location of String to be found

■ **Function**

It finds the location of String IN2 from input String IN1. If the location is found, it shows a position of a first character of String IN2 from String IN1. Otherwise, output is 0.

■ **Program Example**

1. LD



2. ST

POSITION := FIND(EN:= %IX0.1.2, IN1:= IN1_TEXT1, IN2:= IN2_TEXT2);

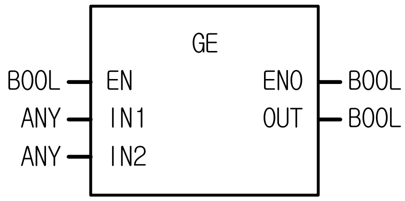
(1) If the transition condition (%IX0.1.1) is on, FIND function executes

(2) If input String IN_TEXT1=‘ABCEF’ and IN_TEXT2=‘BC’, then output variable POSITION = 2.

(3) The first location of IN_TEXT2 (‘BC’) from input String IN_TEXT1 (‘ABCEF’) is 2nd.

```
    INPUT (IN1) : IN_TEXT1 (STRING) = 'ABCEF'

          (IN2) : IN_TEXT2(STRING) =  'BC'
                                       ↓(FIND)
  OUTPUT (OUT)  : POSITION (INT)    =   2
```

GE	'Greater than or equal to' comparison	
	Availability	XGI, XGR, XEC, XMC
	Flags	

<table>
<tr><th>Function</th><th>Description</th></tr>
<tr><td>
</td>
<td>Input EN: executes the function in case of 1
IN1: the value to be compared
IN2: the value to compare
Input variable number can be extended up to 8.
IN1, IN2, ... must be of the same data type.

Output ENO: outputs EN value as it is
OUT: comparison result value</td></tr>
</table>

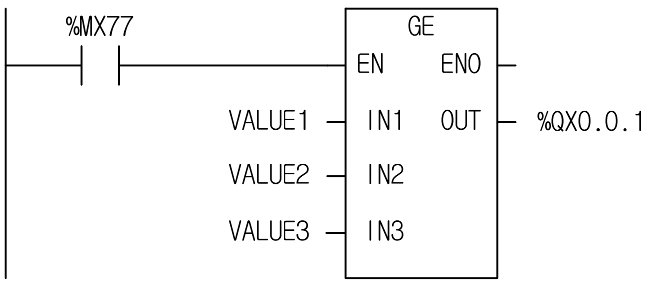
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ **Function**

If IN1 ≥ IN2 ≥ IN3... ≥ INn (n: number of inputs), an output is 1.
Otherwise it is 0.

■ **Program Example**

1. LD



2. ST

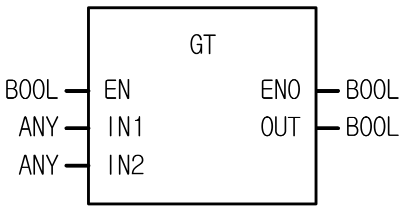
```
%QX0.0.1 := GE(EN:= %MX77, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);
```

(1) If the transition condition (%MX77) is on, GE function executes.

(2) If input variable VALUE1 = 300, VALUE3 = 200, comparison result is VALUE1 ≥ VALUE2 ≥ VALUE3. The output %QX0.01 = 1.

INPUT (IN1) : VALUE1 (INT) = 300(16#012C)	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0
									≥ (GE)							
(IN2) : VALUE2(INT) = 200(16#00C8)	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
									≥ (GE)							
(IN3) : VALUE3(INT) = 100(16#0064)	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
																↓
OUTPUT (OUT): %QX0.0.1 (BOOL) = 1(16#1)																1

GT	'Greater than' comparison	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN1: the value to be compared IN2: the value to compare Input variable number can be extended up to 8. IN1, IN2, ... must be of the same data type. **Output** ENO: outputs EN value as it is OUT: comparison result value

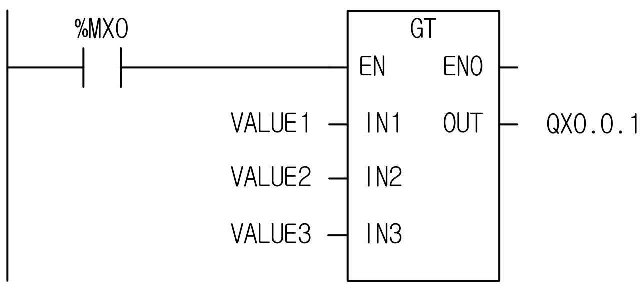
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ **Function**

1. If IN1 > IN2 > IN3... > INn (n: number of inputs), an output is 1.
2. Otherwise it is 0.

■ **Program Example**

1. LD



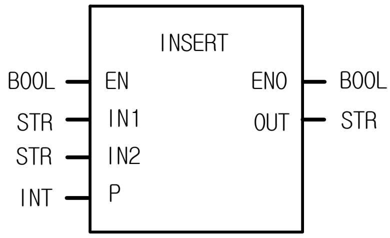
2. ST

%QX0.0.1 := GT(EN:= %MX0, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);

(1) If the transition condition (%MX0) is on, GT function executes.

(2) If input variable VALUE1 = 300, VALUE2 = 200, and VALUE3 = 100, comparison result is VALUE1 > VALUE2 > VALUE3. The output %QX0.0.1 = 1.

INSERT	**Inserts a String**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: String to be inserted IN2: String to insert P: position to insert a String **Output** ENO: without an error, it is 1. OUT: output String

■ **Function**

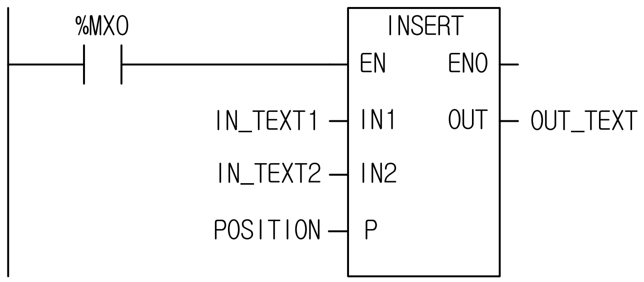
It inserts String IN2 after the P character of IN1 and produces output,OUT.

■ **Flag**

Flag	Description
_ERR	If P ≤ 0, 'character number of variable IN1' < P, or if the character number of result exceeds 31 (just 32 characters are produced), then _ERR, _LER flags are set.

■ **Program Example**

1. LD



2. ST

OUT_TEXT := INSERT(EN:= %MX0, IN1:= IN_TEXT1, IN2:= IN_TEXT2, P:= POSITION);

(1) If the transition condition (%M0) is on, INSERT function executes.
(2) If input variable IN_TEXT1 = 'ABCD', IN_TEXT2 = 'XY', and POSITON = 2, output variable OUT_TEXT = 'ABXYCD'.

```
INPUT (IN1) : IN_TEXT1 (STRING) = 'ABCD'
      (IN2) : IN_TEXT2(STRING)  =  'XY'
      (P)   : POSITION(INT)     =   2
                                    ↓ (FIND)
OUTPUT (OUT): OUT_TEXT          = `ABXYCD`
```

INT_TO_***	INT type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
<pre> INT_TO_*** BOOL ─ EN ENO ─ BOOL INT ─ IN OUT ─ *ANY</pre>	**Input** EN: executes the function in case of 1 IN: integer value to convert **Output** ENO: without an error, it is 1. OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○		○	○	○	○	○	○	○	○					○

*ANY: exclude INT, TIME, DATE, TOD and DT from ANY type.

■ **Function**

It converts input IN type and produces output, OUT.

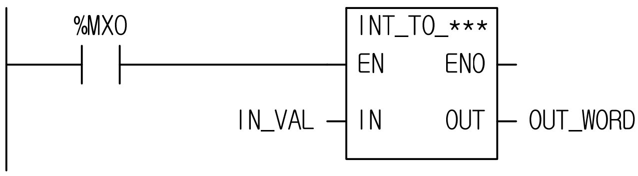
Function	Output Type	Description
INT_TO_SINT	SINT	If input is -128 ～ 127, normal conversion. Otherwise an error occurs.
INT_TO_DINT	DINT	Converts into DINT type normally.
INT_TO_LINT	LINT	Converts into LINT type normally.
INT_TO_USINT	USINT	If input is 0 ～ 255, normal conversion. Otherwise an error occurs.
INT_TO_UINT	UINT	If input is 0 ～ 32767, normal conversion. Otherwise an error occurs.
INT_TO_UDINT	UDINT	If input is 0 ～ 32767, normal conversion. Otherwise an error occurs.
h INT_TO_ULINT	ULINT	If input is 0 ～ 32767, normal conversion. Otherwise an error occurs.
INT_TO_BOOL	BOOL	Takes the lower 1 bit and converts into BOOL type.
INT_TO_BYTE	BYTE	Takes the lower 8 bits and converts into BYTE type.
INT_TO_WORD	WORD	Converts into WORD type without changing the internal bit array.
INT_TO_DWORD	DWORD	Converts into DWORD type filling the upper bits with 0.
INT_TO_LWORD	LWORD	Converts into LWORD type filling the upper bits with 0.
INT_TO_REAL	REAL	Converts INT into REAL type normally.
INT_TO_LREAL	LREAL	Converts INT into LREAL type normally.
INT_TO_STRING	STRING	Converts INT into STRING type normally.

■ **Flag**

Flag	Description
_ERR	If a conversion error occurs, _ERR _LER flags are set. If an error occurs, take as many lower bits as the bit number of the output type and produces an output without changing the internal bit array.

■ **Program Example**

1. LD



2. ST

ST language doesn't support INT_TO_***

In case of INT_TO_WORD

OUT_WORD := INT_TO_WORD(EN:= %MX0, IN1:= IN_VAL);

(1) If the input condition (%MX0) is on, INT_TO_*** function executes.

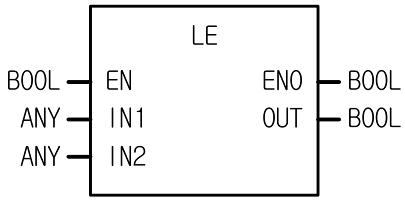
(2) If input variable IN_VAL (INT) = 512 (16#200), output variable OUT_WORD (WORD) = 16#200.

INPUT (IN1) : IN_VAL (INT) = 512(16#200)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

↓ (INT_TO_WORD)

OUTPUT (OUT) : OUT_WORD (WORD) = 16#200	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

LE	'Less than or equal to' comparison	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN1: the value to be compared IN2: the value to compare Input variable number can be extended up to 8. IN1, IN2, ...must be of the same data type. **Output** ENO: outputs EN value as it is OUT: comparison result value

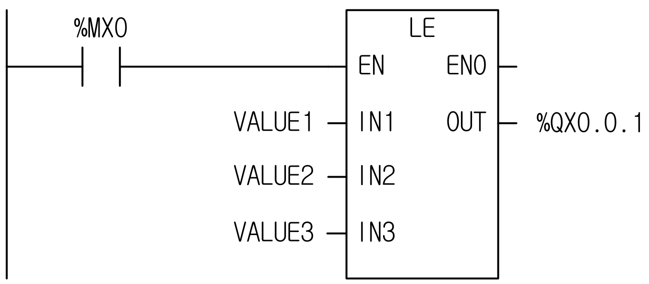
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ **Function**

1. If IN1 ≤ IN2 ≤ IN3... ≤ INn (n: number of inputs), output OUT is 1.
2. Otherwise it is 0.

■ **Program Example**

1. LD



2. ST

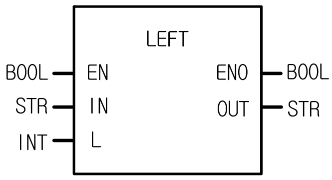
%QX0.0.1 := LE(EN:= %MX0, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);

(1) If the transition condition (%MX0) is on, LE function executes.

(2) If input variable VALUE1 = 100, VALUE2 = 200, and VALUE3 = 200, output %QX0.0.1 = 1 (VALUE1 ≤ VALUE2 ≤ VALUE3).

INPUT (IN1) : VALUE1 (INT) = 100(16#0064)	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
(IN2) : VALUE2(INT) = 200(16#00C8)	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
(IN3) : VALUE3(INT) = 300(16#012C)	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0
OUTPUT (OUT): %QX0.0.1 (BOOL) = 1(16#1)																1

LEFT	Takes the left side of a String	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: input String L: length of a String **Output** ENO: without an error, it is 1. OUT: output String

■ **Function**

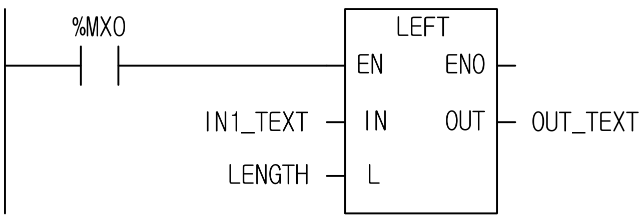
It takes a left String (L) of IN and produces output, OUT.

■ **Flag**

Flag	Description
_ERR	If L < 0, _ERR and _LER flags are set.

■ **Program Example**

1. LD



2. ST

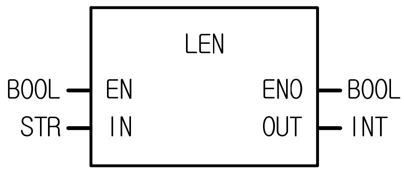
OUT_TEXT:= LEFT(EN:= %MX0, IN:= IN1_TEXT, L:= LENGTH);

(1) If the transition condition (%MX0) is on, function LEFT function executes.

(2) If input variable IN_TEXT = ‘ABCDEFG’ and LENGTH = 3, output String OUT_TEXT = ‘ABC’.

```
INPUT(IN1) : IN_TEXT(STRING) =  'ABCDEFG'
     (IN2) : LENGTH(INT)     =     3
                                   ↓(LEFT)
OUTPUT(OUT) : OUT_TEXT(STRING) =   `ABC`
```

LEN	**Finds a length of a String**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

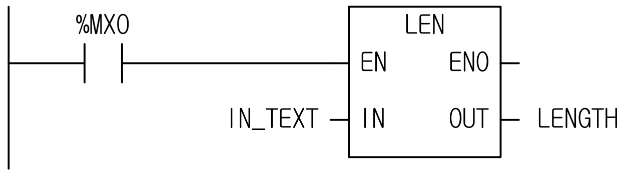
Function	Description
	**Input** EN: executes the function in case of 1 IN: input String **Output** ENO: outputs EN value as it is OUT: the length of a String

■ **Function**

It produces a length (character number) of the input String (IN).

■ **Program Example**

1. LD



2. ST

LENGTH := LEN(EN:= %MX0, IN1:= IN_TEXT);

(1) If the transition condition (%MX0) is on, LEN function executes.

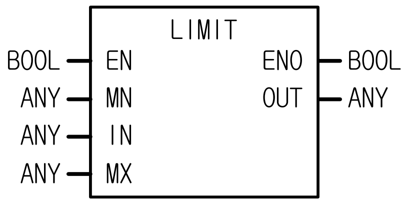
(2) If input variable IN_TEXT = 'ABCD', output variable LENGTH = 4.

INPUT (IN) : IN_TEXT(STRING) = 'ABCD'

↓

OUTPUT (OUT) : LENGTH(INT) = 4

LIMIT	**Limits upper and lower boundaries**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 MN: minimum value IN: the value to be limited MX: maximum value **Output** ENO: outputs EN value as it is OUT: value in the range MN, IN, MX, OUT must be of the same data type.

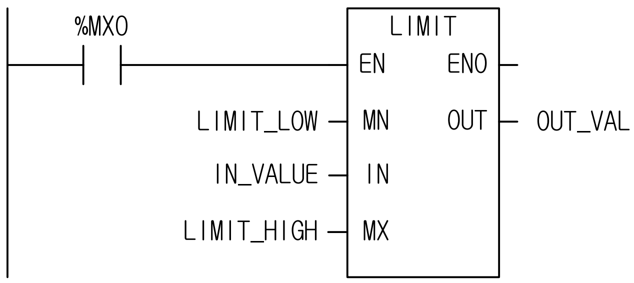
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	MN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	MX	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ **Function**

a) If input IN value is between MN and MX, the IN is an output. That is, if MN ≤ IN ≤ MX, OUT = IN.
b) If input IN value is less than MN, MN is an output. That is, if IN < MN, OUT = MN.
c) If input IN value is greater than MX, MX is an output. That is, if IN > MX, OUT = MX.

■ **Program Example**

1. LD



2. ST

```
OUT_VAL := LIMIT(EN:= %MX0, MX:= LIMIT_LOW, IN:= IN_VALUE, MX:= LIMIT_HIGH);
```

(1) If the transition condition (%MX0) is on, LIMIT function executes.

(2) Output variable OUT_VAL for lower limit input LIMIT_LOW, upper limit input (LIMIT_HIGH) and limited value input IN_VALUE is as follows.

LIMIT_LOW	IN_VALUE	LIMIT_HIGH	OUT_VAL
1000	2000	3000	2000
1000	500	3000	1000
1000	4000	3000	3000

```
INPUT (MN) : LIMIT_LOW (INT) =    1000

      (IN) : IN_VALUE (INT)  =    4000

      (MX) : LIMIT_HIGH(INT) =    3000
                                   ↓(LIMIT)
OUTPUT (OUT) : OUT_VAL (INT)  =   3000
```

LINT_TO_***	LINT type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
`` LINT_TO_*** BOOL ─ EN ENO ─ BOOL LINT ─ IN OUT ─ *ANY ```	**Input** EN: executes the function in case of 1 IN: long integer value to convert **Output** ENO: without an error, it is 1 OUT: type converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○	○	○		○	○	○	○	○	○					○

*ANY: exclude LINT, TIME, DATE, TOD, and DT from ANY type.

■ **Function**

It converts input IN type and produces output, OUT.

Function	Output type	Description
LINT_TO_SINT	SINT	If input is -128 ~ 127, normal conversion. Otherwise an error occurs.
LINT_TO_INT	INT	If input is –32,768 ~ 32,767, normal conversion. Otherwise an error occurs.
LINT_TO_DINT	DINT	If input is -2^{31} ~ 2^{31}-1, normal conversion. Otherwise an error occurs.
LINT_TO_USINT	USINT	If input is 0 ~ 255, normal conversion. Otherwise an error occurs.
LINT_TO_UINT	UINT	If input is 0 ~ 65,535, normal conversion. Otherwise an error occurs.
LINT_TO_UDINT	UDINT	If input is 0 ~ 2^{32}-1, normal conversion. Otherwise an error occurs.
LINT_TO_ULINT	ULINT	If input is 0 ~ 2^{63}-1, normal conversion. Otherwise an error occurs.
LINT_TO_BOOL	BOOL	Takes the lower 1 bit and converts into BOOL type.
LINT_TO_BYTE	BYTE	Takes the lower 8 bits and converts into BYTE type.
LINT_TO_WORD	WORD	Takes the lower 16 bits and converts into WORD type.
LINT_TO_DWORD	DWORD	Takes the lower 32 bits and converts into DWORD type.
LINT_TO_LWORD	LWORD	Converts into LWORD type without changing the internal bit array.
LINT_TO_REAL	REAL	Converts LINT into REAL type. During the conversion, an error caused by the precision may occur.

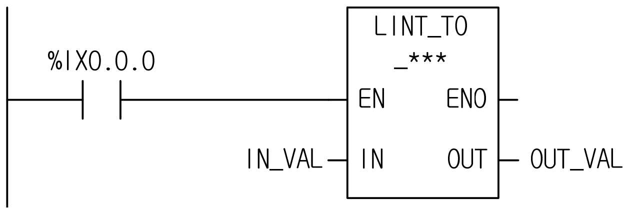
Function	Output type	Description
LINT_TO_LREAL	LREAL	Converts LINT into LREAL type. During the conversion, an error caused by the precision may occur.
LINT_TO_STRING	STRING	Converts the input value into STRING type.

■ **Flag**

Flag	Description
_ERR	If a conversion error occurs, _ERR and _LER flags are set. If an error occurs, lower bits equal to the bit number of the output type are taken to produces an output without changing the Internal bit array.

■ **Program Example**

1. LD



2. ST

ST language doesn't support LINT_TO_***

In case of LINT_TO_DINT

OUT_VAL := LINT_TO_DINT(EN:= %IX0.0.0, IN:= IN_VAL);

(1) If the input condition (%IX0.0.0) is on, LINT_TO_*** function executes.

(2) If input variable IN_VAL (LINT) = 123,456,789, output variable OUT_VAL (DINT) = 123,456,789.

INOUT (IN) : IN_VAL (LINT) = 123,456,789(16#00000000075BCD15)

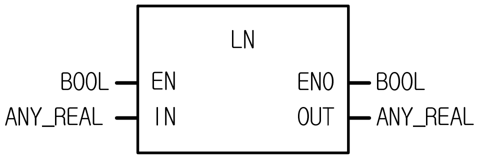
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	1	0	1	1	0	1	1
1	1	0	0	1	1	0	1	0	0	0	1	0	1	0	1

↓(LINT_TO_DINT)

OUTPUT (OUT) : OUT_VAL (DINT) = 123,456,789 (16#075BCD15)

0	0	0	0	0	1	1	1	0	1	0	1	1	0	1	1
1	1	0	0	1	1	0	1	0	0	0	1	0	1	0	1

LN	Natural logarithm operation	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: input value of natural logarithm operation **Output** ENO: outputs EN value as it is OUT: natural logarithm value IN, OUT must be of the same data type

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ **Function**

It finds a natural logarithm value of IN and produces output, OUT.

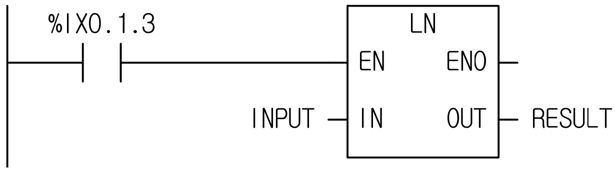
OUT = ln (IN)

■ **Error**

Flag	Description
_ERR	If an input is 0 or a negative number, _ERR and _LER flags are set.

■ **Program Example**

1. LD



2. ST

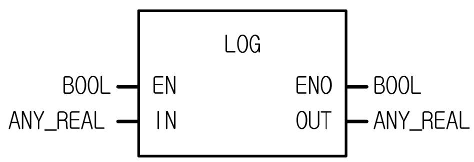
RESULT := LN(EN:= %UX0.1.3, IN1:= INPUT);

(1) If the transition condition (%IX0.1.3) is on, LN function executes.

(2) If input variable INPUT is 2.0, output variable RESULT is 0.6931 ....

ln(2.0) = 0.6931...

LOG	Base 10 Logarithm operation	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: input value of common logarithm operation **Output** END: outputs EN value as it is OUT: the value of common logarithm operation IN, OUT must be of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ **Function**

It finds the value of Base 10 Logarithm of IN and produces output, OUT.

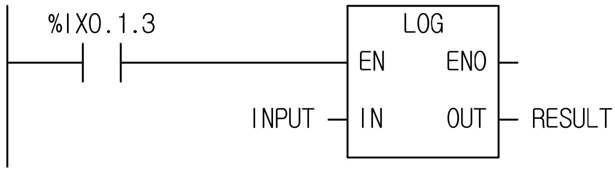
$OUT = \log_{10}(IN) = \log(IN)$

■ **Error**

Flag	Description
_ERR	If input value IN is 0 or a negative number, _ ERR and _LER flags are set.

■ **Program Example**

1. LD



2. ST

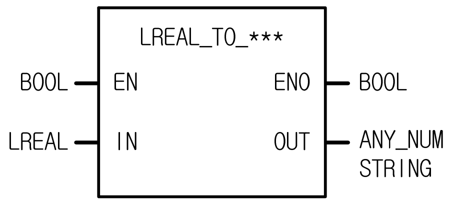
```
RESULT := LOG(EN:= %IX0.1.3, IN:= INPUT);
```

(1) If the transition condition (%IX0.1.3) is on, LOG function executes.

(2) If input variable INPUT is 2.0, output variable RESULT is 0.3010 ......

$Log_{10}(2.0) = 0.3010...$

LREAL_TO_***	LREAL type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: LREAL value to convert **Output** ENO: without an error, it is 1. OUT: type converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT					○	○	○	○	○	○	○	○	○	○						○

■ **Function**

It converts input IN type and produces output, OUT.

Function	Output type	Operation
LREAL_TO_SINT	SINT	If integer number of input is -128 ~ 127, normal conversion. Otherwise an error occurs (decimal round off).
LREAL_TO_INT	INT	If integer number of input is -32,768 ~ 32,767, normal conversion. Otherwise an error occurs (decimal round off).
LREAL_TO_DINT	DINT	If integer number of input is -2^{31} ~ 2^{31}-1, normal conversion. Otherwise an error occurs (decimal round off).
LREAL_TO_LINT	LINT	If integer number of input is -2^{63} ~ 2^{63}-1, normal conversion. Otherwise an error occurs (decimal round off).
LREAL_TO_USINT	USINT	If integer number of input is 0 ~ 255, normal conversion. Otherwise an error occurs (decimal round off).
LREAL_TO_UINT	UINT	If integer number of input is 0 ~ 65,535, normal conversion. Otherwise an error occurs (decimal round off).
LREAL_TO_UDINT	UDINT	If integer number of input is 0 ~ 2^{32}-1, normal conversion. Otherwise an error occurs (decimal round off).

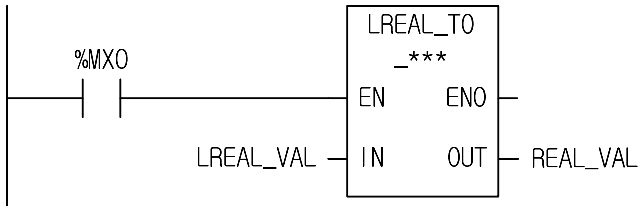
Function	Output type	Operation
LREAL_TO_ULINT	ULINT	If integer number of input is 0 ~ 2^{64}-1, normal conversion. Otherwise an error occurs (decimal round-off).
LREAL_TO_LWORD	LWORD	Converts into LWORD type without changing the internal bit array.
LREAL_TO_REAL	REAL	Converts LREAL into REAL type normally. During the conversion, an error caused by the precision may occur.
LREAL_TO_STRING	STRING	Converts LREAL into STRING type normally.

■ **Flag**

Flag	Description
_ERR	If an overflow occurs because an input value is greater than the value available for the output type, _ERR and _LER flags are set. If an error occurs, an output is 0.

■ **Program Example**

1. LD



2. ST

ST language doesn't support LREAL_TO_***

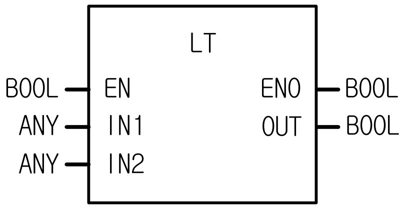
In case of LREAL_TO_REAL

REAL_VAL := LREAL_TO_REAL(EN:= %MX0, IN:= LREAL_VAL);

(1) If the input condition (%MX0) is on, LREAL_TO_*** function executes.
(2) If input variable LREAL_VAL (LREAL) = -1.34E-12, output variable REAL_VAL (REAL) = -1.34E-12.

```
 INPUT (IN) : LREAL_VAL (LREAL) =   -1.34E-12
                                        ↓ (LREAL_TO_REAL)
OUTPUT (OUT) : REAL_VAL (REAL)   =   -1.34E-12
```

LT	‘Less than’ comparison	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN1: the value to be compared IN2: the value to compare Input variable number can be extended up to 8 IN1, IN2, ...must be of the same data type **Output** ENO: outputs EN value as it is OUT: comparison result value

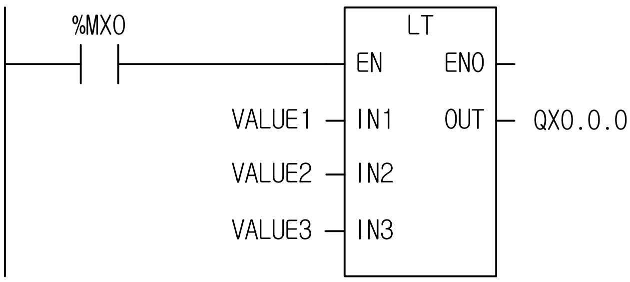
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ **Function**

1. If IN1 < IN2 < IN3... < INn (n: number of inputs), output value OUT is 1.
2. Otherwise output, OUT is 0.

■ **Program Example**

1. LD



2. ST

%QX0.0.0 := LT(EN:= %MX0, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);

(1) If the transition condition (%MX0) is on, LT function executes.

(2) If input variable VALUE1 = 100, VALUE2 = 200, and VALUE3 = 300, output %Q0.0.0 = 1 because of VALUE1 < VALUE 2 < VALUE 3 as a result of the comparison.

INPUT (IN1) : VALUE1 (INT) = 100(16#0064)	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
<(LT)																
(IN2) : VALUE2(INT) = 200(16#00C8)	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
<(LT)																
(IN3) : VALUE3(INT) = 300(16#012C)	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0
OUTPUT (OUT) : %QX0.0.0 (BOOL) = 1(16#0001)																1

LWORD_TO_* **	**LWORD type conversion**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
LWORD_TO_*** BOOL — EN ENO — BOOL LWORD — IN OUT — *ANY	**Input** EN: executes the function in case of 1 IN: bit String to convert (64bit) **Output** ENO: outputs EN value as it is OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○		○	○	○	○	○	○	○	○		○				○	○

*ANY: exclude LWORD, REAL, TIME, DATE and TOD from ANY type.

■ **Function**

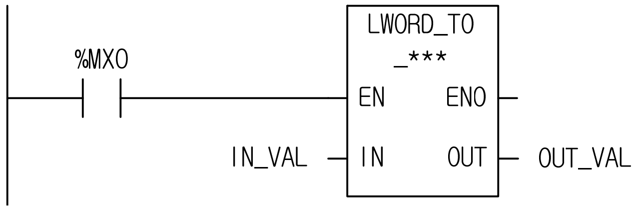
It converts input IN type and produces output, OUT.

Function	Output type	Description
LWORD_TO_SINT	SINT	Takes the lower 8 bits and converts into SINT type.
LWORD_TO_INT	INT	Takes the lower 16bits and converts into INT type.
LWORD_TO_DINT	DINT	Takes the lower 32bits and converts into DINT type.
LWORD_TO_LINT	LINT	Converts into LINT type without changing the internal bit array.
LWORD_TO_USINT	USINT	Takes the lower 8 bits and converts into USINT type.
LWORD_TO_UINT	UINT	Takes the lower 16 bits and converts into UINT type.
LWORD_TO_UDINT	UDINT	Takes the lower 32bits and converts into UDINT type.
LWORD_TO_ULINT	ULINT	Converts into ULINT type without changing the internal bit array.
LWORD_TO_BOOL	BOOL	Takes the lower 1 bit and converts into BOOL type.
LWORD_TO_BYTE	BYTE	Takes the lower 8 bits and converts into BYTE type.
LWORD_TO_WORD	WORD	Takes the lower 16 bits and converts into WORD type.
LWORD_TO_DWORD	DWORD	Takes the lower 32 bits and converts into DWORD type.
LWORD_TO_LREAL	LREAL	Converts LWORD into LREAL type.
LWORD_TO_DT	DT	Converts into DT type without changing the internal bit array. However,

Function	Output type	Description
		with a value out of DT range (DT#2163-12-31-23:59:59:999), _ERR, _LER flags are set and it is alternately converted within the range of DT.
LWORD_TO_STRING	STRING	Converts input value into STRING type.

■ **Program Example**

1. LD



2. ST

ST language doesn't support LWORD_TO_***

In case of LWORD_TO_LINT

OUT_VAL := LWROD_TO_LINT(EN:= %MX0, IN:= IN_VAL);

(1) If the input condition (%MX0) is on, LWORD_TO_*** function executes.

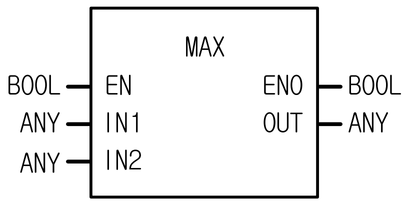
(2) If input variable IN_VAL (LWORD) = 16#FFFF_FFFF_FFFF_FFFF, output variable OUT_VAL (LINT) is -1 (16#FFFF_FFFF_FFFF_FFFF).

INPUT (IN) : IN_VAL (LWORD) = 16#FFFFFFFFFFFFFFFF

↓ (LWORD_TO_LINT)

OUTPUT (OUT) : OUT_VAL (LINT) = -1

MAX	Maximum value	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN1: the value to be compared IN2: the value to compare Input variable number can be extended up to 8. **Output** ENO: outputs EN value as it is OUT: maximum value among input IN1, IN2,…, OUT must be of the same data type

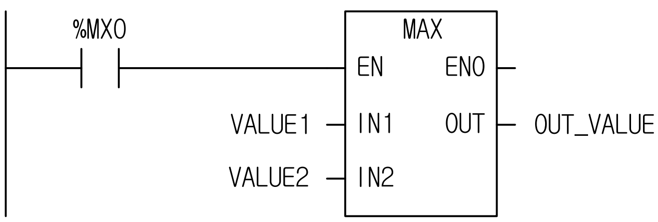
ANY type Variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ **Function**

It produces the maximum value among input IN1, IN2,..., INn (n: number of inputs).

■ **Program Example**

1. LD



2. ST

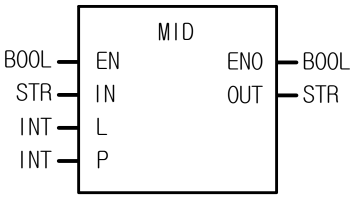
OUT_VALUE := MAX(EN:= %MX0, IN1:= VALUE1, IN2:= VALUE2);

(1) If the transition condition (%MX0) is on, MAX function executes.

(2) As the result of comparing input variable (VALUE1 = 100 and VALUE2 = 200), maximum value is 200.
Output OUT_VAL is 200.

INPUT (IN1) : VALUE1 (INT) = 100(16#0064)	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
(MAX)																
(IN2) : VALUE2(INT) = 200(16#00C8)	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
↓																
OUTPUT (OUT): OUT_VALUE (INT) = 200(16#00C8)	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0

MID	Takes the middle part of a String	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
 	**Input** EN: executes the function in case of 1 IN: input String L: the length of String to output P: starting location of String to output **Output** ENO: without an error, it is 1. OUT: output String

■ **Function**

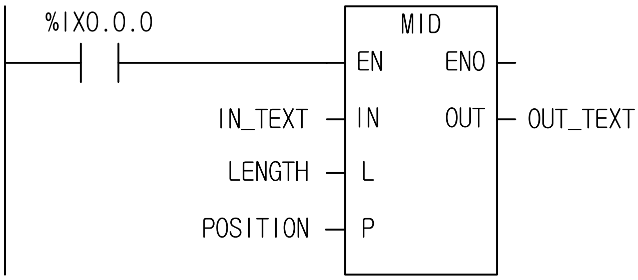
It produces a String (L) of IN from the P character.

■ **Flag**

Flag	Description
_ERR	If (character number of variable IN) < P, P <= 0 or L < 0, then _ERR and _LER flags are set.

■ **Program Example**

1. LD



2. ST

OUT_TEXT := MID(EN:= %IX0.0.0, IN:= IN_TEXT, L:= LENGTH, P:= POSITION);

(1) If the transition condition (%IX0.0.0) is on, MID function executes.

(2) If input String IN_TEXT = 'ABCDEFG', the length of String LENGTH = 3, and starting location of character starting POSITION = 2, output variable OUT_TEXT = 'BCD'.

INPUT (IN) : IN_TEXT(STRING) = 'ABCDEFG'

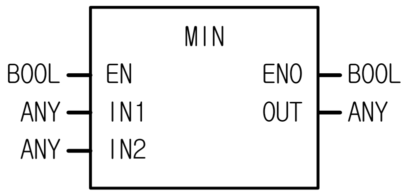
(L) : LENGTH(INT) = 3

(P) : POSITION(INT) = 2

↓ (MID)

OUTPUT (OUT) : OUT_TEXT = `BCD`

MIN	Minimum value	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN1: value to be compared IN2: value to compare Input variable number can be extended up to 8 **Output** ENO: outputs EN value as it is OUT: minimum value among input values IN1, IN2, ..., OUT must be of all the same data type.

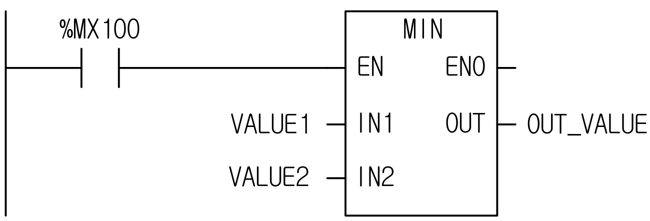
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ **Function**

Produces the minimum value among input IN1, IN2, ... , INn (n: number of inputs).

■ **Program Example**

1. LD



2. ST

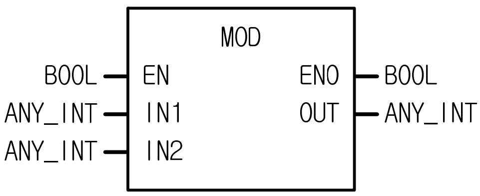
OUT_VALUE := MIN(EN:= %MX100, IN1:= VALUE1, IN2:= VALUE2);

(1) If the transition condition (%MX100) is on, MIN function executes.

(2) The output is OUT_VALUE = 100 because its minimum value is 100 as the result of comparing VALUE1 = 100 to VALUE2 = 200.

INPUT (IN1) : VALUE1 (INT) = 100(16#0064)	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
(MIN)																
(IN2) : VALUE2(INT) = 200(16#00C8)	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
↓																
OUTPUT(OUT): OUT_VALUE (INT) = 100(16#0064)	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

MOD	Dividing result (remainder)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN1: dividend IN2: divisor **Output** ENO: outputs EN value as it is OUT: dividing result (remainder) IN1, IN2, ..., OUT must be of all the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1						○	○	○	○	○	○	○	○							
	IN2						○	○	○	○	○	○	○	○							
	OUT						○	○	○	○	○	○	○	○							

■ **Function**

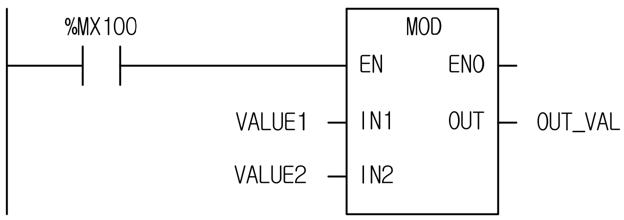
1. Divides IN1 by IN2 and outputs its remainder as OUT.

   OUT = IN1 - (IN1/IN2) × IN2 (If IN2 = 0, OUT = 0)

IN1	IN2	OUT
7	2	1
7	-2	1
-7	2	-1
-7	-2	-1
7	0	0

■ **Program Example**

1. LD



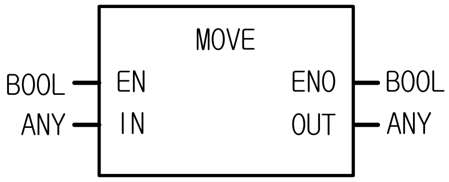
2. ST

```
OUT_VAL := MOD(EN:= %MX100, IN1:= VALUE1, IN2:= VALUE2);
```

(1) If the transition condition (%MX100) is on, MOD function executes.
(2) If the dividend VALUE1 = 37 and the divisor VALUE2 = 10, the remainder value OUT_VAL is 7 as a result of dividing 37 by 10.

INPUT (IN1) : VALUE1 (INT) = 37(16#0025)	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1
	(MOD)															
(IN2) : VALUE2(INT) = 10(16#000A)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
	↓															
OUTPUT (OUT) : OUT_VAL (INT) = 7(16#0007)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

MOVE	Data movement (Copy data)	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: value to be moved **Output** ENO: outputs EN value as it is OUT: moved value Variables connected to IN and OUT are of the same type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ **Function**

Moves an IN value to OUT.

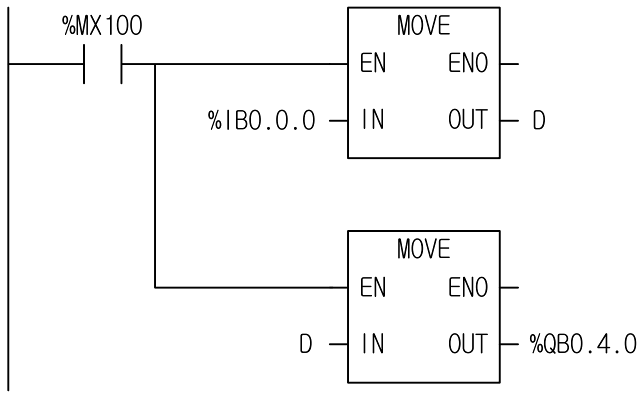
■ **Flag**

Flag	Description
_ERR	If IN and OUT array data type’s size are different each other, data move is not operated and ENO value is 0, _ERR and _LER flags are set.

■ **Program Example**

This is a program that transfers the 8-contact inputs %I0.0.0～%I0.0.7 to the variable D and then moves them to output %Q0.4.0～%Q0.4.7.

1. LD



2. ST

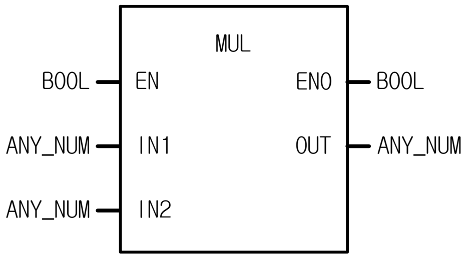
```
D := MOVE(EN:= %MX100, IN:= %IB0.0.0);
%QB0.4.0 := MOVE(EN:= %MX100, IN:= D);
```

(1) If the transition condition (%MX100) is on, MOVE function executes.

(2) It moves 8-contact input module data to the variable D by the first MOVE function and moves them to %Q0.4.0～%Q0.4.7 by the second one.

INPUT (IN) : %IB0.0.0 (BYTE) = 16#18	0	0	0	1	1	0	0	0
↓ (MOVE)								
D(BYTE) = 16#18	0	0	0	1	1	0	0	0
↓ (MOVE)								
OUTPUT (OUT) : %QB0.4.0 (BYTE) = 16#18	0	0	0	1	1	0	0	0

MUL	Multiplication	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: multiplicand IN2: multiplier Input is available to extend up to 8. **Output** ENO: without an error, it is 1 OUT: multiplied value Variables connected to IN1, IN2, ..., OUT are all of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1						○	○	○	○	○	○	○	○	○	○					
	IN2						○	○	○	○	○	○	○	○	○	○					
	OUT						○	○	○	○	○	○	○	○	○	○					

■ **Function**

Multiplies an IN1, IN2,..., INn (n: number of inputs) and outputs the result as OUT.

OUT = IN1 × IN2 × ... × INn

■ **Flag**

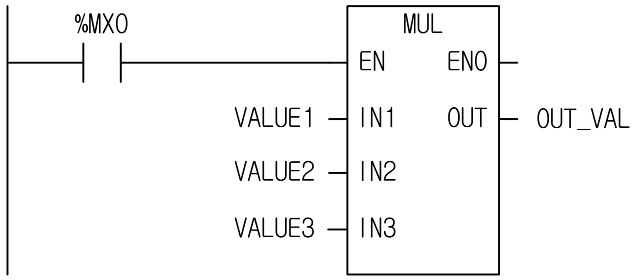
Flag	Description
_ERR	If an output value is beyond the range of its data-type, _ERR and _LER flags are set.

☆ If REAL, LREAL type operation exceeds the maximum or minimum value in the middle of the operation because it performs the operation sequentially from IN1 to IN8, _ERR, _LER flag are set and the result is an unlimited or abnormal value.

(1.#INF000000000000e+000, 1.#SNAN00000000000e+000, 1.#QNAN00000000000e+000).

■ **Program Example**

1. LD



2. ST

OUT_VAL := MUL(EN:= %MX0, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);

(1) If the transition condition (%MX0) is on, MUL function executes.

(2) If input variables of MUL function, VALUE1 = 30, VALUE2 = 20, VALUE3 = 10, then the output variable OUT_VAL = 30 ×20 ×10 = 6000.

INPUT (IN1) : VALUE1 (INT) = 30(16#001E)	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0

+(MUL)

(IN2) : VALUE2(INT) = 20(16#0014)	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

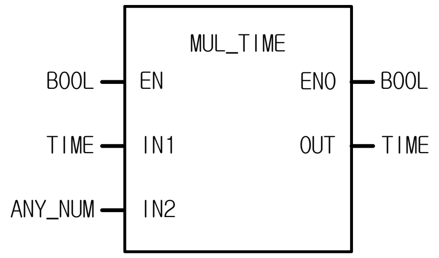
+(MUL)

(IN3) : VALUE3(INT) = 10(16#000A)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

↓

OUTPUT (OUT) : OUT_VAL (INT) = 6,000(16#1770)	0	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0

MUL_TIME	Time multiplication	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: time to be multiplied IN2: multiplying value **Output** ENO: without an error, it is 1 OUT: multiplied result

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN2						○	○	○	○	○	○	○	○	○	○					

■ **Function**

Multiplies the IN1 (time) by IN2 (number) and outputs the result time as OUT.

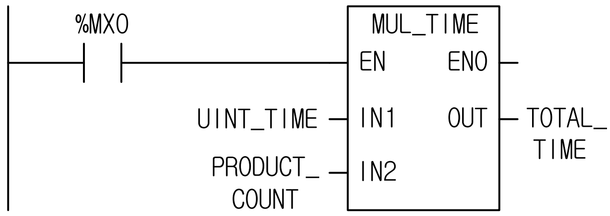
■ **Flag**

Flag	Description
_ERR	If an output value is out of its TIME-data range, _ERR and _LER flags are set. If a negative value is entered to IN2, _ERR and _LER flags are on and IN2 is converted to hexadecimal, producing the multiplication result.

■ **Program Example**

This is the program that sets the required working time: the average estimated time per unit product is 20min 2sec and the number of product to produce a day is 20 in one product line.

1. LD



2. ST

TOTAL_TIME := MUL_TIME(EN:= %MX0, IN1:= UINT_TIME, IN2:= PRODUCT_COUNT);

(1) Write input variable (IN1: the estimated time per unit product) UNIT_TIME: T#20M2S.

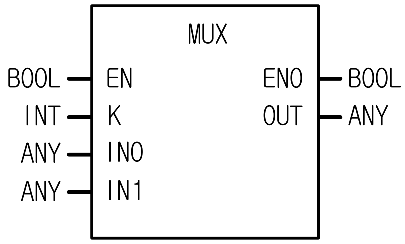
(2) Write input variable (IN2: quantity of production) PRODUCT_COUNT: 20.

(3) Write TOTAL_TIME to the output variable (OUT: total required working time).

(4) If the transition condition (%MX0) is on, T#6H40M40S is produced in output TOTAL_TIME.

INPUT (IN1): UNIT_TIME (TIME) = T#20MS2S
(MUL_TIME)
(IN2): PRODUCT_COUNT(INT) = 16#18
↓
OUTPUT (OUT): TOTAL_TIME (TIME) = T#6H40M40S

MUX	**Selection from multiple inputs**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 K: selection IN0: the value to be selected IN1: the value to be selected Input variable number can be extended up to 7(IN0, IN1, …, IN6) **Output** ENO: without an error, it is 1. OUT: the selected value IN0, IN1, ..., OUT must be of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN0	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ **Function**

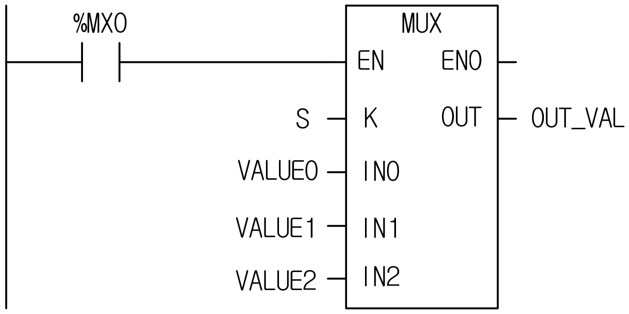
1. Selects one among several inputs (IN0, IN1, …, INn) with K value and produces it.
2. If K = 0, IN0 is an output; if K = 1, IN1 is an output; if K = n, INn is an output.

■ **Flag**

Flag	Description
_ERR	If K is greater than or equal to 'n' which is the number of input variable INn, then IN0 is an output and _ERR, _LER flags are set. If K is negative, _ERR and _LER flags are set

■ **Program Example**

1. LD



2. ST

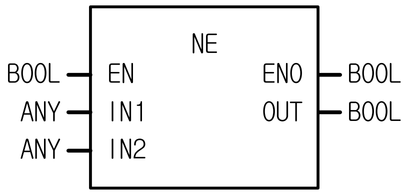
```
OUT_VAL := MUX(EN:= %MX0, K:= S, IN0:= VALUE0, IN1:= VALUE1, IN2:= VALUE2);
```

(1) If the transition condition (%MX0) is on, MUX function executes.

(2) Input variable is selected by selection variable S and is moved to OUT.

INPUT (K) : S (INT) = 2
(IN0) : VALUE0(WORD) = 16#0011
(IN1) : VALUE1(WORD) = 16#0022
(IN2) : VALUE2(WORD) = 16#0033
↓(MUX)
OUTPUT (OUT) : OUT_VAL (WORD) = 16#0033

NE	'Not equal to' comparison	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN1: The value to be compared IN2: The value to be compared IN1, IN2 must be of the same data type. **Output** ENO: outputs EN value as it is OUT: the compared result value

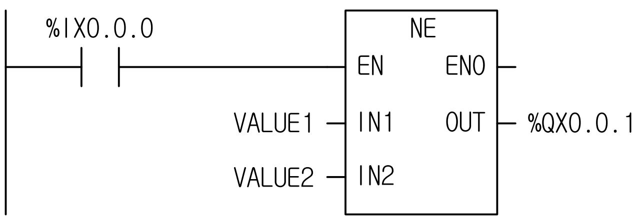
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ **Function**

1. If IN1 is not equal to IN2, output, OUT is 1.
2. If IN1 is equal to IN2, output, OUT is 0.

■ **Program Example**

1. LD



2. ST

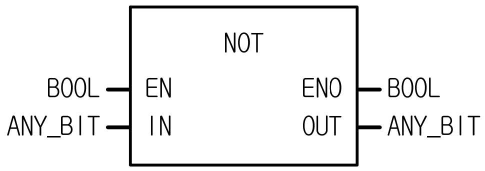
%QX0.0.1 := NE(EN:= %IX0.0.0, IN1:= VALUE1, IN2:= VALUE2);

(1) If the transition condition (%IX0.0.0) is on, NE function executes.

(2) If input variable VALUE1 = 300, VALUE2 = 200 (the compared result VALUE1 and VALUE2 are different), output result value is %QX0.0.1 = 1.

INPUT (IN1) : VALUE1 (INT) = 300(16#012C)	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0
(IN2) : VALUE2(INT) = 200(16#0C8)	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
OUTPUT (OUT) : %QX0.0.1 (BOOL) = 1(16#1)																1

NOT	Reverse Logic (Logic inversion)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN: the value to be logically inverted **Output** ENO: outputs EN value as it is OUT: the inversed (NOT) value IN, OUT must be of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○															
	OUT	○	○	○	○	○															

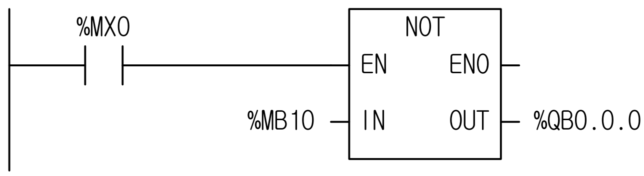
■ **Function**

It inverts the IN (by bit) and produces output, OUT.

IN	1100 1010
OUT	0011 0101

■ **Program Example**

1. LD



2. ST

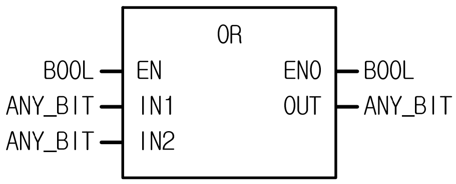
```
%QB0.0.0 := NOT_BYTE(EN:= %MX0, IN1:=MB10);
```

(1) If the transition condition (%MX0) is on, NOT function executes.

(2) If NOT function executes, input data value of %MB10 is inversed and is written in %QB0.0.0.

INPUT (IN) : %MB10 (BYTE) = 16#CC	1	1	0	0	1	1	0	0
↓ (NOT)								
OUTPUT (OUT) : %QB0.0.0 (BYTE) = 16#33	0	0	1	1	0	0	1	1

OR	**Logic Sum**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN1: input 1 IN2: input 2 Input variables extend up to 8. **Output** ENO: outputs EN value as it is OUT: OR result IN1, IN2, OUT must be of all the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○															
	OUT	○	○	○	○	○															

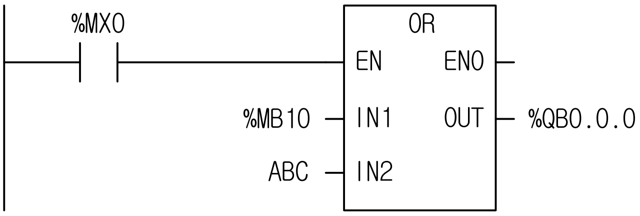
■ **Function**

It performs a logical OR on the input variables by bit and produces output, OUT.

IN1	1111 0000
OR	
IN2	1010 1010
OUT	1111 1010

■ **Program Example**

1. LD



2. ST

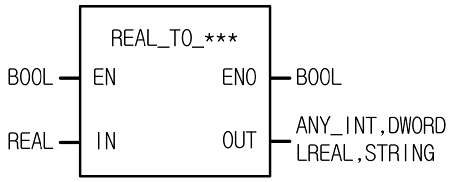
%QB0.0.0 := OR2_BYTE(EN:=%MX0, IN1:=%MB10, IN2:=ABC);

(1) If the transition condition (%MX0) is on, function OR executes.

(2) The result of a logic sum (OR) for %MB10 = 2#1100_1100 and ABC = 2#1111_0000 is produced in %QB0.0.0 = 2#1111_1100

INPUT (IN1) : %IB0.0.0 (BYTE) = 16#CC	1	1	0	0	1	1	0	0
& (OR)								
(IN2) : ABC(BYTE) = 16#F0	1	1	1	1	0	0	0	0
↓								
OUTPUT (OUT) : %QB0.0.0 (BYTE) = 16#FC	1	1	1	1	1	1	0	0

REAL_TO_***	REAL type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: the REAL value to be converted **Output** ENO: without an error, it is 1. OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT				○		○	○	○	○	○	○	○	○		○					○

■ **Function**

It converts the IN type and outputs it as OUT.

Function	Output Type	Description
REAL_TO_SINT	SINT	If integer part of input is -128 ~ 127, normal conversion. Otherwise an error occurs. (Decimals round-off)
REAL_TO_INT	INT	If integer part of input is -32,768 ~ 32,767, normal conversion. Otherwise an error occurs. (Decimals round-off)
REAL_TO_DINT	DINT	If integer part of input is -2^{31} ~ 2^{31}-1, normal conversion. Otherwise an error occurs. (Decimals round-off)
REAL_TO_LINT	LINT	If integer part of input is -2^{63} ~ 2^{63}-1, normal conversion. Otherwise an error occurs. (Decimals round-off)
REAL_TO_USINT	USINT	If integer part of input is 0 ~ 255, normal conversion. Otherwise an error occurs. (Decimals round-off)
REAL_TO_UINT	UINT	If integer part of input is 0 ~ 65,535, normal conversion. Otherwise an error occurs. (Decimals round-off)
REAL_TO_UDINT	UDINT	If integer part of input is 0 ~ 2^{32}-1, normal conversion. Otherwise an error occurs. (Decimals round-off)
REAL_TO_ULINT	ULINT	If integer part of input is 0 ~ 2^{64}-1, normal conversion. Otherwise an error occurs. (Decimals round-off)
REAL_TO_DWORD	DWORD	Converts into DWORD type without changing the internal bit array.

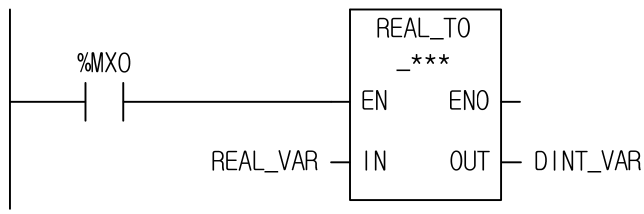
Function	Output Type	Description
REAL_TO_LREAL	LREAL	Converts REAL into LREAL type normally.
REAL_TO_STRING	STRING	Converts REAL into STRING type normally.

■ **Flag**

Flag	Description
_ERR	If overflow occurs (input value is greater than the value to be stored in output type), _ERR, _LER flags are set. If an error occurs, the output is 0.

■ **Program Example**

1. LD



2. ST

ST language doesn't support REAL_TO_***

In case of REAL_TO_DINT

DINT_VAR := REAL_TO_DINT(EN:=%MX0, IN:=REAL_VAR);

(1) If the transition condition (%MX0) is on, function REAL_TO_*** executes.

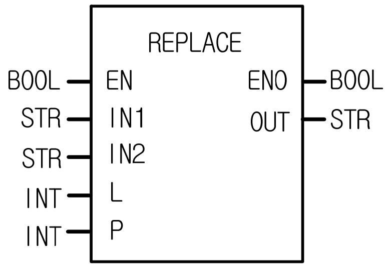
(2) If REAL_VAL (REAL type) = 1.234E4, DINT_VAL (DINT) = 12,340.

INPUT (IN) : REAL_VAL (REAL) = 1.234E4

↓(REAL_TO_DINT)

OUTPUT (OUT) : DINT_VAL (DINT) = 12,340

REPLACE	String replacement	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: character string to be replaced IN2: character string to replace L: the length of character string to be replaced P: position of character string to be replaced **Output** ENO: without an error, it is 1 OUT: output character string

■ **Function**

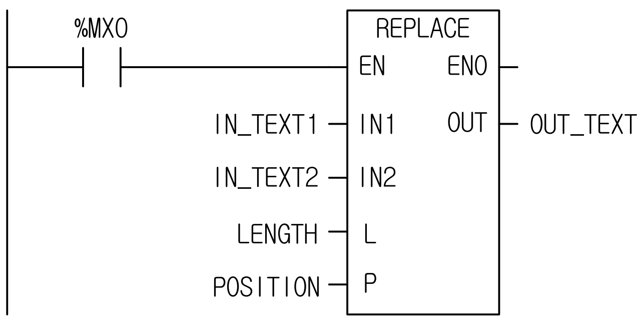
1. Its function is to remove the L-length charter from IN1 (starting from P) and put IN2 in the removed position as output OUT.

■ **Flag**

Flag	Description
_ERR	_ERR, _LER flags are set if P ≤ 0 or L < 0, P > (input character number of IN1) or character number of result > 30

■ **Program Example**

1. LD



2. ST

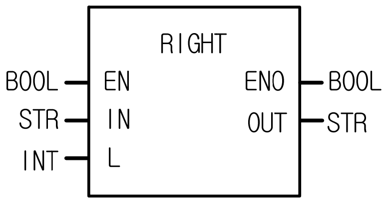
OUT_TEXT := REPLACE(EN:=%MX0, IN1:=IN_TEXT1, IN2:= IN_TEXT2, L:=LENGTH, P:=POSITION);

(1) If the transition condition (%MX0) is on, function REPLACE (character string replacement) executes.

(2) If input variable of character string to be replaced IN_TEXT1 = \`ABCDEF\`, input variable of character string to replace is IN_TEXT2 = \`X\`, input variable of character string length to be replaced LENGTH = 3 and input variable of character string position designation to be replaced is POSITION = 2, then 'BCD' of IN_TEXT1 is replaced with 'X' of IN_TEXT2 and output variable OUT_TEXT is 'AXEF'.

```
INPUT (IN1) : IN_TEXT1 (STRING) = `ABCDEF`
      (IN2) : IN_TEXT2(STRING) =  ` X`
        (L) : LENGTH(INT) =        3
        (P) : POSITION(INT) =      2
                                   ↓
OUTPUT (OUT)  : OUT_TEXT (STRING) =  `AXEF`
```

RIGHT	**To take the right of character string**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: If EN is 1, function executes IN: input character string L: length of character string **Output** ENO: without an error, it is 1 OUT: output character string

■ **Function**

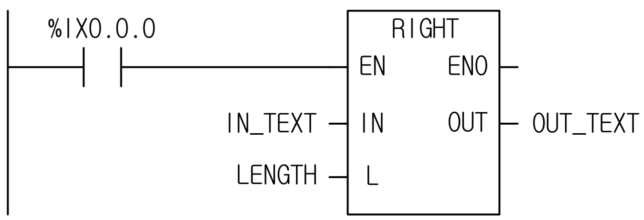
It takes a right L-length character string of IN and produces output, OUT.

■ **Flag**

Flag	Description
_ERR	If L < 0, _ERR and _LER flags are set.

■ **Program Example**

1. LD



2. ST

```
OUT_TEXT := RIGHT(EN:=%IX0.0.0, IN:=IN_TEXT, L:=LENGTH);
```

(1) If the transition condition (%IX0.0.0) is on, function RIGHT (to take the right of character string) executes.

(2) If character string declared as input variable IN_TEXT = `ABCDEFG` and the length of character string to output is LENGTH = 3, output character string variable is OUT_TEXT = `EFG`.

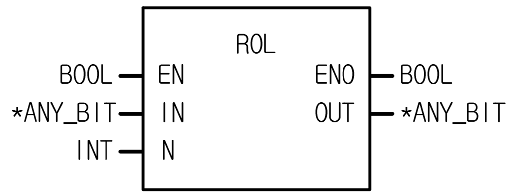
INPUT (IN) : IN_TEXT (STRING) = `ABCDEFG`

(L) : LENGTH(INT) = 3

↓ (RIGHT)

OUTPUT (OUT) : OUT_TEXT (STRING) = `EFG`

ROL	Rotate to Left	
	Availability	XGI, XGR, XEC, XMC
	Flags	

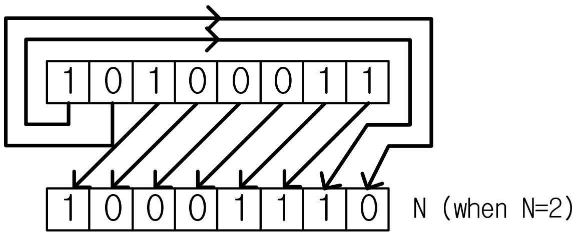
Function	Description
	**Input** EN: executes the function in case of 1 IN: the value to be rotated N: bit number to rotate **Output** ENO: outputs EN value as it is OUT: the rotated value

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT.

■ **Function**

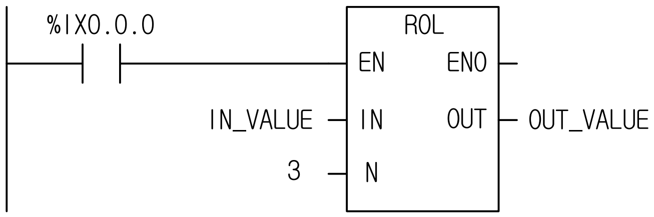
It rotates input IN to the left as many as N bit number.



■ **Program Example**

This is the program that rotates the value of input data (2#1100_1100_1100_1100:16#CCCC) to the left by 3 bits if input %IX0.0.0 is on.

1. LD



2. ST

```
OUT_VALUE := ROL(EN:=%IX0.0.0, IN:=IN_VALUE, N:=3);
```

(1) Set input variable IN_VALUE to rotate.

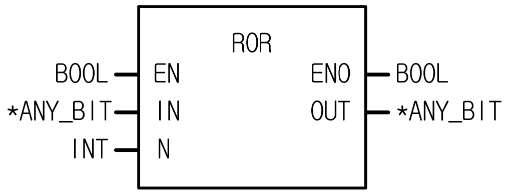
(2) Set the value to be rotated.

(3) Set output variable to output the rotated data value as OUT_VALUE.

(4) If the transition condition (%IX0.0.0) is on, function ROL executes and a data bit set as input variable is rotated to the left by 3 bits and produces output, OUT_VALUE..

INPUT (IN) : IN_VALUE (WORD) = 16#CCCC	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
(N) : 3	↓ (ROL)															
OUTPUT (OUT) :OUT_VALUE (WORD) =16#6666	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

ROR	Rotate to right	
	Availability	XGI, XGR, XEC, XMC
	Flags	

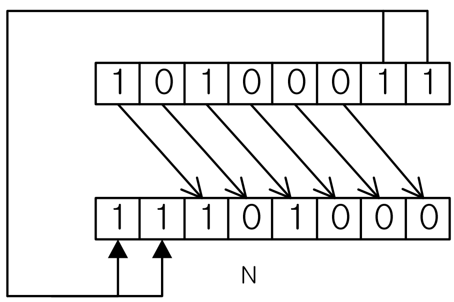
Function	Description
	**Input** EN: executes the function in case of 1 IN: the value to be rotated N: bit number to rotate **Output** ENO: outputs EN value as it is OUT: the rotated value

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY type.

■ **Function**

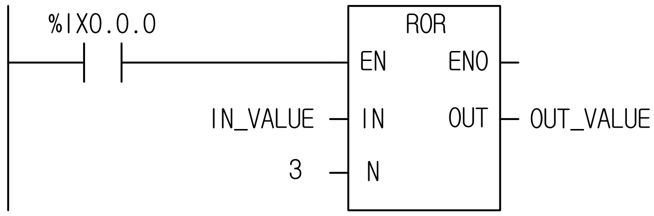
It rotates input IN to the right as many as N bit number.



■ **Program Example**

This is the program that rotates input data value (2#1110_0011_0011_0001: 16#E331) to the right by 3 bits if input %I0.0.0 is on.

1. LD



2. ST

OUT_VALUE := ROR(EN:=%IX0.0.0, IN:=IN_VALUE, N:=3);

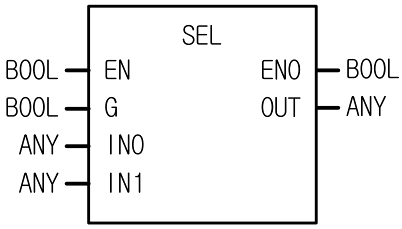
(1) Set input variable of a data value to rotate as IN_VALUE.

(2) Insert bit number 3 into bit number input N.

(3) If the transition condition (%IX0.0.0) is on, function ROR (rotate Right) executes and data bit set as input variable is rotated to the right by 3 bits and produces output ,OUT_VALUE.

INPUT (IN): IN_VALUE (WORD) = 16#E331	1	1	1	0	0	0	1	1	0	0	1	1	0	0	0	1
(N) : 3	↓ (ROR)															
OUTPUT (OUT): OUT_VALUE (WORD) = 16#3C66	0	0	1	1	1	1	0	0	0	1	1	0	0	1	1	0

SEL	**Selection from two inputs**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 G: selection IN0: the value to be selected IN1: the value to be selected **Output** ENO: outputs EN value as it is OUT: the selected value IN1, IN2, OUT must be of all the same type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN0	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

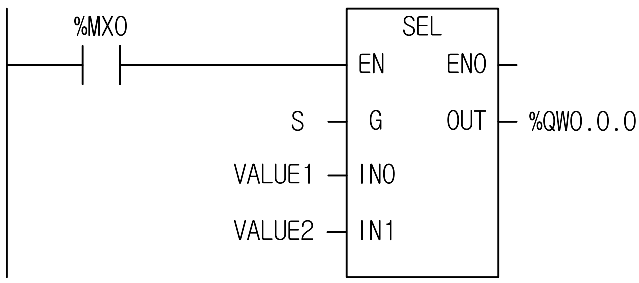
■ **Function**

If G is 0, IN0 is an output and if G is 1, IN1 is an output.

■ **Program Example**

If the input (%MX0) is on, this program selects an input between the two (VALUE1, VALUE2) and outputs the value as described in S.

1. LD



2. ST

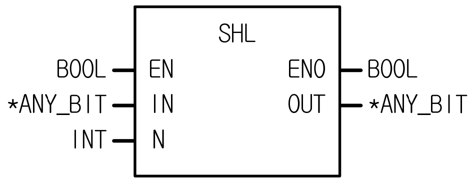
```
%QW0.0.0 := SEL(EN:=%MX0, G:=S, IN0:=VALUE1, IN1:=VALUE2);
```

(1) If the transition condition (%MX0) is on, function SEL executes.

(2) If S = 1 and VALUE1 = 16#1110, VALUE2 = 16#FF00, then output variable %QW0.0.0 = 16#FF00.

INPUT (G) : S = 1
(IN0) : VALUE1(WORD) = 16#1110
(IN1) : VALUE2(WORD) = 16#FF00
↓ (SEL)
OUTPUT (OUT) : %QW0.0.0 (WORD) = 16#FF00

SHL	Shift Left	
	Availability	XGI, XGR, XEC, XMC
	Flags	

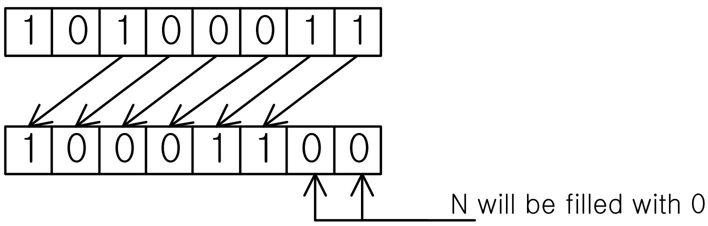
Function	Description
	**Input** EN: If EN is 1, function is executes. IN: bit string to be shifted N: bit number to be shifted **Output** ENO: outputs EN value as it is OUT: the shifted value

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT.

■ **Function**

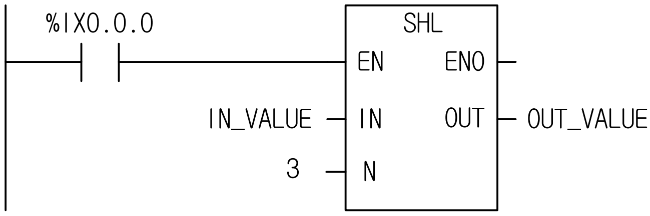
1. It shifts input IN to the left as many as N bit number.
2. N number bit on the rightmost of input IN is filled with 0.



■ **Program Example**

This is the program that shifts input data value (2#1100_1100_1100_1100:16#CCCC) to the left by 3 bits if input %IX0.0.0 is on

1. LD



2. ST

OUT_VALLUE := SHL(EN:=%IX0.0.0, IN:=IN_VALUE, N:=3);

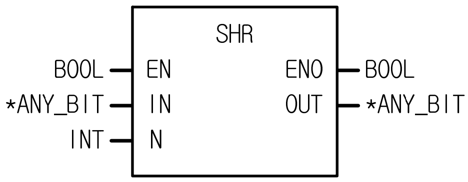
(1) Set the input variable IN_VALUE (2#1100_1100_1100_1100: 16#CCCC).

(2) Insert bit number 3 into N.

(3) If the transition condition (%IX0.0.0) is on, function SHL (shift Left) executes and data bit set as input variable shifts to the left by 3 bits and produces output, OUT_VALUE.

INPUT (IN) : IN_VALUE (WORD) = 16#CCCC	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
(N) : 3	↓ (ROL)															
OUTPUT (OUT) : OUT_VALUE (WORD) = 16#6660	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0

SHR	Shift Right	
	Availability	XGI, XGR, XEC, XMC
	Flags	

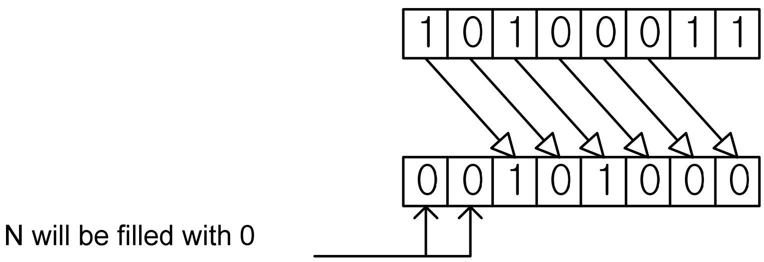
Function	Description
	**Input** EN: executes the function in case of 1 IN: bit string to be shifted N: bit number to be shifted **Output** ENO: outputs EN value as it is OUT: the shifted value

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT.

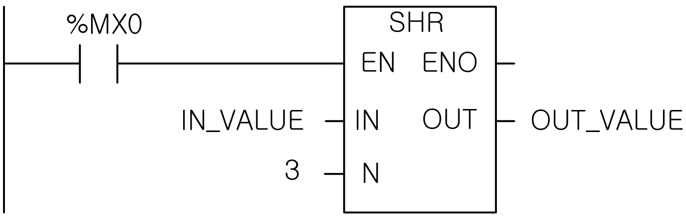
■ **Function**

1. It shifts input IN to the right as many as N bit number.
2. N number bit on the leftmost of input IN is filled with 0.



■ **Program Example**

1. LD



2. ST

OUT_VALUE := SHR(EN:=%MX0, IN:=IN_VALUE, N:=3);

(1) If the transition condition (%MX0) is on, function SHL (Shift Left) executes.

(2) Data bit set as input variable shift to the right by 3 bits and produces outputs, OUT_VALUE.

INPUT (IN) : IN_VALUE (WORD) = 16#E331

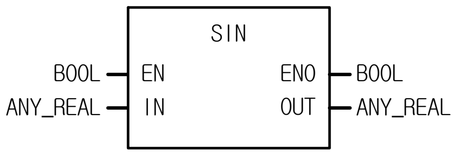
1	1	1	0	0	0	1	1	0	0	1	1	0	0	0	1

(N) : 3 ↓ (ROR)

OUTPUT (OUT) : OUT_VALUE (WORD) = 16#1C66

0	0	0	1	1	1	0	0	0	1	1	0	0	1	1	0

SIN	**Sine operation**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN: input value of Sine operation (radian) **Output** ENO: outputs EN value as it is OUT: Sine operation result value IN, OUT must be of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

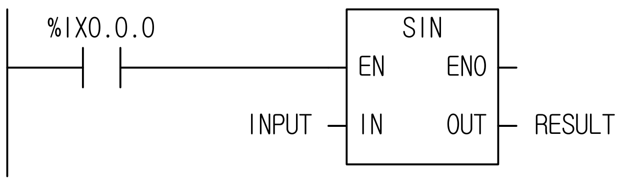
■ **Function**

Finds the Sine operation value of IN and produces output, OUT.

OUT = SIN (IN)

■ **Program Example**

1. LD



2. ST

```
RESULT := SIN(EN:=IX0.0.0, IN:=INPUT);
```

(1) If the transition condition (%IX0.0.0) is on, function SIN (Sine operation) executes.

(2) If the value of input variable INPUT is 1.0471 .. . (π/3 rad = 60°), RESULT declared as output variable is 0.8660 ....

($\sqrt{3}$ /2). SIN(π/3) = $\sqrt{3}$ /2 = 0.8660

INPUT (IN) : INPUT (REAL) = 1.0471

↓ (SIN)

OUTPUT (OUT) : RESULT (REAL) = 8.65976572E-01

SINT_TO_***	SINT type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
`SINT_TO_***` `BOOL — EN    ENO — BOOL` `SINT — IN    OUT — *ANY`	**Input** EN: executes the function in case of 1 IN: short Integer value **Output** ENO: without an error, it is 1. OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○		○	○	○	○	○	○	○	○	○					○

*ANY: exclude SINT, TIME, DATE, TOD and DT from ANY type.

■ **Function**

It converts the IN type and outputs it as OUT.

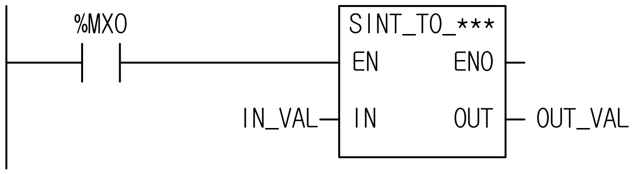
Function	Output type	Description
SINT_TO_INT	INT	Converts into INT type normally.
SINT_TO_DINT	DINT	Converts into DINT type normally.
SINT_TO_LINT	LINT	Converts into LINT type normally.
SINT_TO_USINT	USINT	If input is 0 ~ 127, normal conversion. Otherwise an error occurs.
SINT_TO_UINT	UINT	If input is 0 ~ 127, normal conversion. Otherwise an error occurs.
SINT_TO_UDINT	UDINT	If input is 0 ~ 127, normal conversion. Otherwise an error occurs.
SINT_TO_ULINT	ULINT	If input is 0 ~ 127, normal conversion. Otherwise an error occurs.
SINT_TO_BOOL	BOOL	Takes the lower 1 bit and converts into BOOL type.
SINT_TO_BYTE	BYTE	Converts into BYTE type without changing the internal bit array.
SINT_TO_WORD	WORD	Converts into WORD type filling the upper bits with 0.
SINT_TO_DWORD	DWORD	Converts into DWORD type filling the upper bits with 0.
SINT_TO_LWORD	LWORD	Converts into LWORD type filling the upper bits with 0.
SINT_TO_REAL	REAL	Converts SINT into REAL type normally.
SINT_TO_LREAL	LREAL	Converts SINT into LREAL type normally.
SINT_TO_STRING	STRING	Converts SINT into STRING type normally.

■ **Flag**

Flag	Description
_ERR	If a conversion error occurs, _ERR and _LER flags are set. If an error occurs, take the lower bits as many as bit number of output type and output it without changing the internal bit array.

■ **Program Example**

1. LD



2. ST

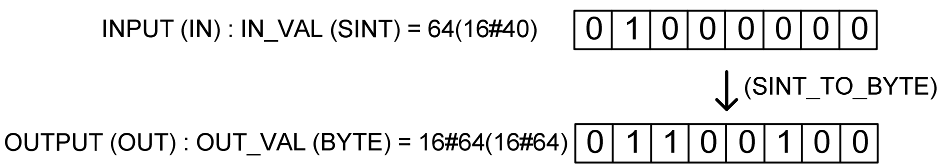
ST language doesn't support SINT_TO_***

In case of SINT_TO_BYTE

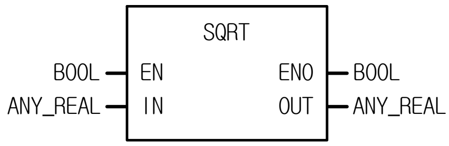
OUT_VAL := SINT_TO_BYTE(EN:=%MX0, IN:=IN_VAL);

(1) If the input condition (%MX0) is on, function SINT_TO_*** executes.

(2) If input variable IN_VAL (SINT type) = 64 (2#0100_0000), output variable OUT_VAL (BYTE type) = 16#40 (2#0100_0000).



SQRT	Square root operation	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: input value of square root operation **Output** ENO: without an error, it is 1. OUT: square root value IN, OUT must be of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ **Function**

It finds the square root value of IN and output it as OUT.

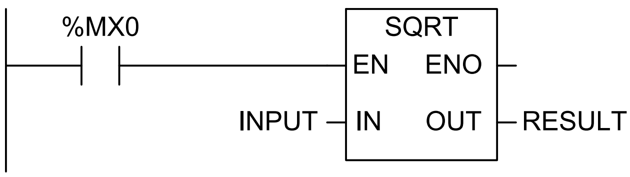
$$OUT = \sqrt{IN}$$

■ **Flag**

Flag	Description
_ERR	If the value of IN is a negative number, _ERR and _LER flag are set.

■ **Program Example**

1. LD



2. ST

```
RESULT := SQRT(EN:=%MX0, IN:=INPUT);
```

(1) If the transition condition (%MX0) is on, function SQRT (square root operation) executes.

(2) If the value of input variable declared as INPUT is 9.0, RESULT declared as output variable is 3.0.

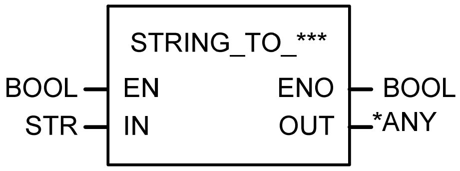
$\sqrt{9.0} = 3.0$

INPUT (IN) : INPUT (REAL) = 9.0

↓ (SQRT)

OUTPUT (OUT) : RESULT (REAL) = 3.0

STRING_TO_***	STRING type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: If EN is 1, function converts. IN: character string **Output** ENO: without an error, it is 1. OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: exclude STRING from ANY type.

■ **Function**

1. Converts the IN type and outputs it as OUT.

Function	Output type	Description
STRING_TO_SINT	SINT	Converts STRING into SINT type.
STRING_TO_INT	INT	Converts STRING into INT type.
STRING_TO_DINT	DINT	Converts STRING into DINT type.
STRING_TO_LINT	LINT	Converts STRING into LINT type.
STRING_TO_USINT	USINT	Converts STRING into USINT type.
STRING_TO_UINT	UINT	Converts STRING into UINT type.
STRING_TO_UDINT	UDINT	Converts STRING into UDINT type.
STRING_TO_ULINT	ULINT	Converts STRING into ULINT type.
STRING_TO_BOOL	BOOL	Converts STRING into BOOL type.
STRING_TO_BYTE	BYTE	Converts STRING into BYTE type.
STRING_TO_WORD	WORD	Converts STRING into WORD type.
STRING_TO_DWORD	DWORD	Converts STRING into DWORD type.
STRING_TO_LWORD	LWORD	Converts STRING into LWORD type.
STRING_TO_REAL	REAL	Converts STRING into REAL type.
STRING_TO_LREAL	LREAL	Converts STRING into LREAL type.
STRING_TO_DT	DT	Converts STRING into DT type.
STRING_TO_DATE	DATE	Converts STRING into DATE type.

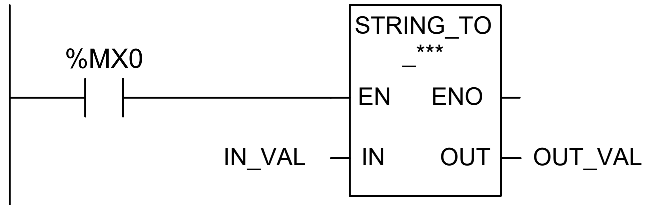
Function	Output type	Description
STRING_TO_TOD	TOD	Converts STRING into TOD type.
STRING_TO_TIME	TIME	Converts STRING into TIME type.

■ **Flag**

Flag	Description
_ERR	If input character type does not match with output data type, _ERR and _LER flags are set.

■ **Program Example**

1. LD



2. ST

ST language doesn't support STRING_TO_***

In case of STRING_TO_REAL

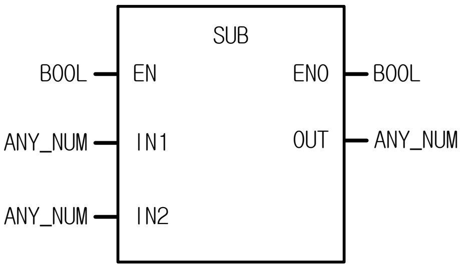
OUT_VAL := STRING_TO_REAL(EN:=%MX0, IN:=IN_VAL);

(1) If the input condition (%MX0) is on, function STRING_TO_*** executes.

(2) If input variable IN_VAL (STRING) = '-1.34E12', output variable OUT_VAL (REAL) = -1.34E12.

```
INPUT (IN) : IN_VAL (STRING)  =  '-1.34E12'
                                  ↓ (STRING_TO_REAL)
OUTPUT (OUT) : OUT_VAL (REAL)  =   -1.34E12
```

SUB	**Subtraction**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: the value to be subtracted IN2: the value to subtract **Output** ENO: without an error, it is 1. OUT: the subtracted result value The variables connected to IN1, IN2 and OUT must be of all the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1						○	○	○	○	○	○	○	○	○	○					
	IN2						○	○	○	○	○	○	○	○	○	○					
	OUT						○	○	○	○	○	○	○	○	○	○					

■ **Function**

It subtracts IN2 from IN1 and outputs it as OUT.

OUT = IN1 — IN2

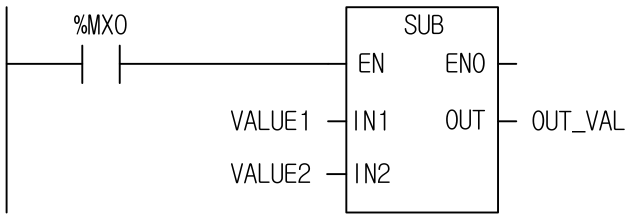
■ **Flag**

Flag	Description
_ERR	If output value is out of range of related data type, _ERR and _LER flags are set.

☆ If LREAL type operation exceeds the maximum or minimum value in the middle of operation because it performs operation serially from IN1 to IN8, _ERR,_LER flag is set and the result is an unlimited or abnormal value.
(1.#INF000000000000e+000, 1.#SNAN00000000000e+000, 1.#QNAN00000000000e+000)

■ **Program Example**

1. LD



2. ST

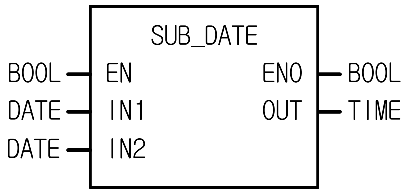
```
OUT_VAL := SUB(EN:=%MX0, IN1:=VALUE1, IN2:=VALUE2);
```

(1) If the transition condition (%MX0) is on, function SUB executes.

(2) If input variables VALUE1 = 300, VALUE2 = 200, OUT_VAL is 100 after the operation.

INPUT (IN1) : VALUE1 (INT) = 300(16#012C)	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0
									-(SUB)							
(IN2) : VALUE2(INT) = 200(16#00C8)	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
									↓							
OUTPUT (OUT) : OUT_VAL (INT) = 100(16#0064)	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

SUB_DATE	Date subtraction	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: standard date IN2: the date to subtract **Output** ENO: without an error, it is 1. OUT: produces the difference between two dates as time data.

■ **Function**

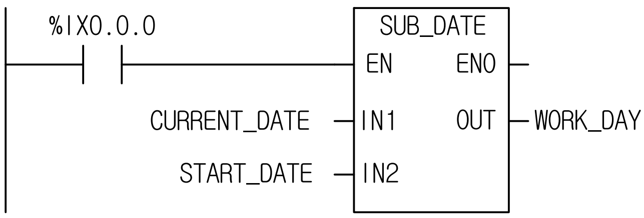
It subtracts IN2 (specific date) from IN1 (standard date) and outputs the difference between two dates as OUT.

■ **Flag**

Flag	Description
_ERR	If output value is out of range (TIME data type), _ERR and _LER flags are set. An error occurs: 1) when date difference exceeds the range of TIME data type (T#49D17H2M47S295MS); 2) the result of date operation is a negative number.

■ **Program Example**

1. LD



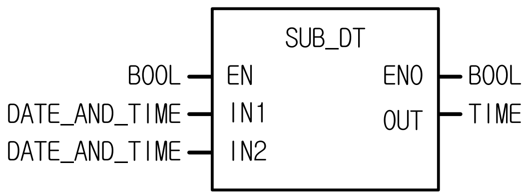
2. ST

WORK_DAY := SUB_DATE(EN:=%IX0.0.0, IN1:=CURRENT_DATE, IN2:=START_DATE);

(1) If the transition condition (%IX0.0.0) is on, function SUB_DATE executes.
(2) If input variable CURRENT_DATE is D#1995-12-15 and START_DATE is D#1995-11-1, the working days declared as output variable WORK_DAY is T#44D.

INPUT (IN1) : CURRENT_DATE (DATE) = D#1995-12-15
(SUB_DATE)
(IN2) : START_DATE(DATE) = D#1995-11-1
↓
OUTPUT (OUT) : WORK_DAY (TIME) = T#44D

SUB_DT	Date and Time subtraction	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: standard date and time of day IN2: date and time of day to subtract **Output** ENO: without an error, it is 1. OUT: the subtracted result time

■ **Function**

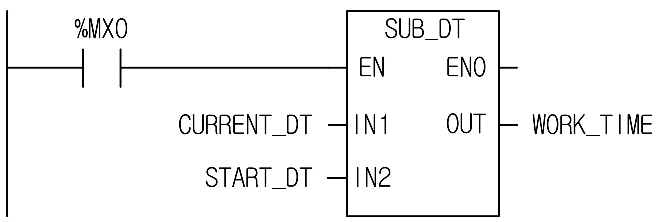
It subtracts IN2 (specific date and time of day) from IN1 (standard date and time of day) and outputs the time difference as OUT.

■ **Flag**

Flag	Description
_ERR	If output value is out of range of TIME data type, _ERR and _LER flags are set. If the result of date and time of day subtraction operation is a negative number, an error occurs.

■ **Program Example**

1. LD



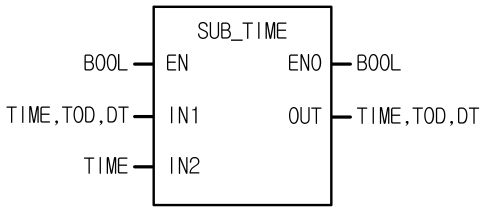
2. ST

```
WORK_TIME := SUB_DT(EN:=%MX0, IN1:=CURRNET_DT, IN2:=START_DT);
```

(1) If the transition condition (%MX0) is on, function SUB_DT (Time and Date subtraction) executes.
(2) If the current date and time of day CURRENT_DT is DT#1995-12-15-14:30:00 and the starting date and the time of day to work START_DT is DT#1995-12-13-12:00:00, the continuous working time declared as output variable WORK_TIME is T#2D2H30M.

```
INPUT (IN1) : CURRENT_DT (DT) =  DT#1995-12-15-14:30:00
                                            (SUB_DATE)
      (IN2) : START_DT(DT)  =    DT#1995-12-13-12:00:00
                                          ↓
  OUTPUT (OUT) : WORK_TIME (TIME) =       T#2D2H30M
```

SUB_TIME	Time subtraction	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: standard time of day IN2: the time to subtract **Output** ENO: without an error, it is 1. OUT: the subtracted result time or time of day OUT data type is the same as the input IN1 type. That is, if IN1 type is TIME, OUT type must be TIME.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1																○		○	○	
	OUT																○		○	○	

■ **Function**

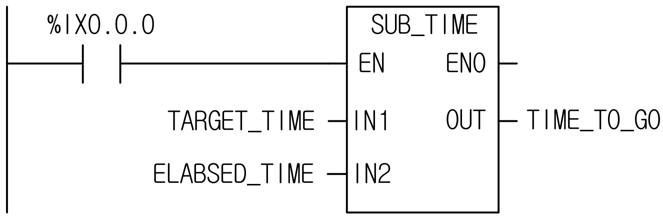
1. If IN1 is TIME, it subtracts the time from the standard time and produces OUT (time difference).
2. If IN1 is TIME_OF_DAY, it subtracts the time from the standard time of day and outputs the time of a day as OUT.
3. If IN1 is DATE_AND_TIME, it subtracts the time from the standard date and the time of day and produces the date and the time of day as OUT.

■ **Flag**

Flag	Description
_ERR	If the output value is out of range of related data type, _ERR and _LER flags are set. If the result subtracting the time from the standard time is a negative number or the result subtracting the time from the time of day is a negative number, an error occurs.

■ **Program Example**

1. LD



2. ST

TIME_TO_GO := SUB_TIME(EN:=%IX0.0.0, IN1:=TARGET_TIME, IN2:=ELABSED_TIME);

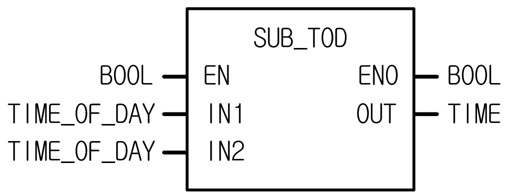
(1) If the transition condition (%IX0.0.0) is on, function SUB_TIME (time subtraction) executes.

(2) If total working time declared as input variable TARGET_TIME is T#2H30M, the elapsed time ELAPSED_TIME is T#1H10M30S300MS, the remaining working time declared as output variable TIME_TO_GO is T#1H19M29S700MS.

INPUT (IN1) : TARGET_TIME (TIME) = T#2H30M
(SUB_DATE)
(IN2) : ELAPSED_TIME(TIME) = T#1H10M30S300MS

OUTPUT (OUT) : TIME_TO_GO (TIME) = T#1H19M29S700MS

SUB_TOD	TOD Subtraction	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: standard time of day IN2: the time of day to subtract **Output** ENO: without an error, it is 1 OUT: the subtracted result time

■ **Function**

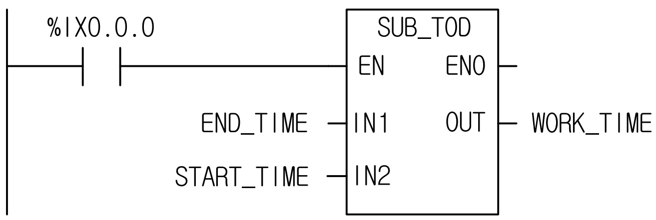
It subtracts the IN2 (specific time of day) from IN1 (standard time of day) and outputs the time difference as OUT.

■ **Flag**

Flag	Description
_ERR	If the result subtracting the time of day from the time of day is a negative number, an error occurs.

■ **Program Example**

1. LD



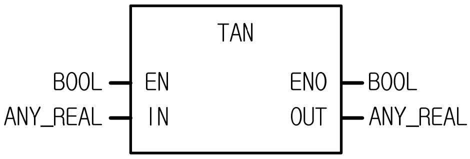
2. ST

WORK_TIME := SUB_TOD(EN:=%IX0.0.0, IN1:=END_TIME, IN2:=START_TIME);

(1) If the transition condition (%IX0.0.0) is on, function SUB_TOD (time of day subtraction) executes.
(2) If END_TIME declared as input variable is TOD#14:20:30.500 and the starting time to work, START_TIME is TOD#12:00:00, the required time to work, WORK_TIME declared as output variable is T#2H20M30S500MS.

INPUT (IN1) : END_TIME (TOD) = TOD#14:20:30.500
(SUB_TOD)
(IN2) : START_TIME(TOD) = TOD#12:00:00
↓
OUTPUT (OUT) : WORK_TIME (TIME) = T#2H20M30S500MS

TAN	Tangent Operation	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN: tangent input value (radian) **Output** ENO: outputs EN value as it is OUT: the result value of Tangent operation IN, OUT must be of the same data type

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

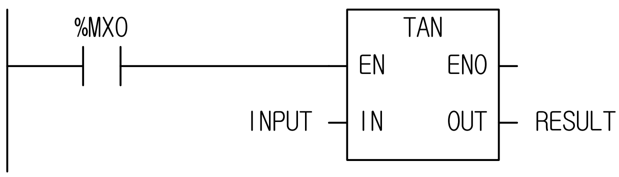
■ **Function**

It performs Tangent operation of IN and produces output, OUT.

OUT = TAN(IN)

■ **Program Example**

1. LD



2. ST

```
RESULT := TAN(EN:=%MX0, IN:=INPUT);
```

(1) If the transition condition (%MX0) is on, function TAN (Tangent operation) executes.
(2) If the value of input variable declared as INPUT is 0.7853... (π/4 rad = 45°), RESULT declared as output variable is 1.0000.

TAN(π/4) = 1

INPUT (IN) : INPUT (REAL) = 0.7853

$\downarrow$ (TAN)

OUTPUT (OUT) : RESULT (REAL) = 9.99803722E-01

TIME_TO_***	TIME type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
<pre> TIME_TO_*** BOOL ─ EN ENO ─ BOOL TIME ─ IN OUT ─ DWORD,UDINT STRING</pre>	**Input** EN: executes the function in case of 1 IN: time data to be converted **Output** ENO: outputs EN value as it is OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT				○								○								○

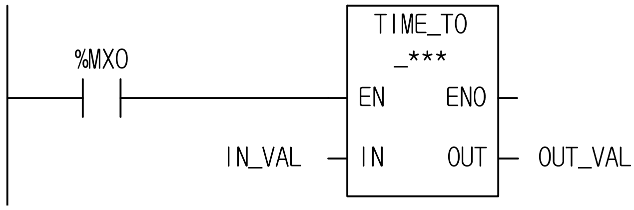
■ **Function**

It converts the IN type and produces OUT.

Function	Output type	Description
TIME_TO_UDINT	UDINT	Converts TIME into UDINT type. It converts only data type without changing the data (internal bit array state).
TIME_TO_DWORD	DWORD	Converts TIME into DWORD type. It converts only data type without changing the data (internal bit array state).
TIME_TO_STRING	STRING	Converts TIME into STRING type.

■ **Program Example**

1. LD



2. ST

ST language doesn't support TIME_TO_***

In case of TIME_TO_UDINT

OUT_VAL := TIME_TO_UDINT(EN:=%MX0, IN:=IN_VAL);

(1) If the transition condition (%MX0) is on, function TIME_TO_*** executes.

(2) If input variable IN_VAL (TIME) = T#120MS, output variable OUT_VAL (UDINT) = 120.

INPUT (IN) : IN_VAL (TIME) = T#120MS(16#78)	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0

↓ (TIME_TO_UDINT)

OUTPUT (OUT) : OUT_VAL (UDINT) = 120(16#78)	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0

TOD_TO_***	TOD type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
<pre> TOD_TO_*** BOOL ——— EN ENO ——— BOOL TIME OF DAY ——— IN OUT ——— DWORD,UDINT STRING</pre>	**Input** EN: executes the function in case of 1 IN: time of a day data to be converted **Output** ENO: outputs EN value as it is OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT				○								○								○

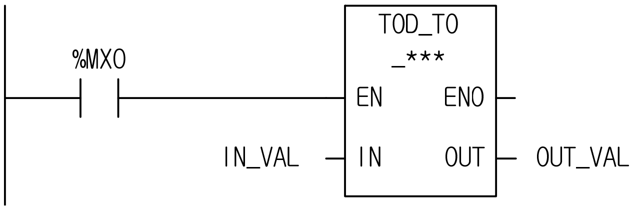
■ **Function**

It converts the IN type and outputs it as OUT.

Function	Output type	Description
TOD_TO_UDINT	UDINT	Converts TOD into UDINT type. Converts only data type without changing a data (internal bit array state).
TOD_TO_DWORD	DWORD	Converts TOD into DWORD type. Converts only data type without changing a data (internal bit array state).
TOD_TO_STRING	STRING	Converts TOD into STRING type.

■ **Program Example**

1. LD



2. ST

ST language doesn’t support TIME_TO_***

In case of TIME_TO_UDINT

OUT_VAL := TOD_TO_STRING(EN:=%MX0, IN:=IN_VAL);

(1) If the transition condition (%MX0) is on, function TOD_TO_*** executes.

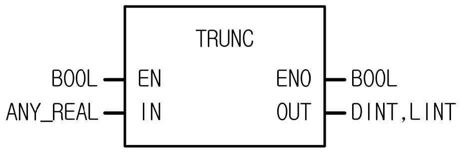
(2) If input variable IN_VAL (TOD) = TOD#12:00:00, output variable OUT_VAL (STRING) = ‘TOD#12:00:00’.

INPUT (IN) : IN_VAL (TOD) = TOD#12:00:00

↓ (TOD_TO_STRING)

OUTPUT (OUT) : OUT_VAL (STRING) = 'TOD#12:00:00'

TRUNC	**Round off the decimal fraction of IN and converts into integer number**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: REAL value to be converted **Output** ENO: without an error, it is 1. OUT: the Integer converted value

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT								○	○											

■ **Function**

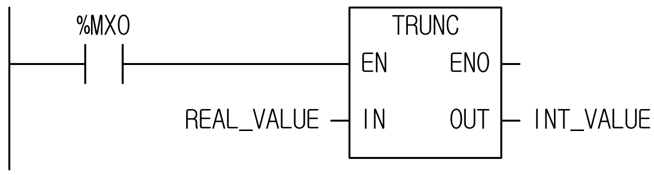
Function	Input type	Output type	Description
TRUNC	REAL	DINT	Round off the decimal fraction of input IN and outputs the Integer value as OUT.
	LREAL	LINT	

■ **Flag**

Flag	Description
_ERR	_ERR, _LER flags is set: 1) if the converted value is greater than maximum value of data type connected to OUT; 2) if the variable connected to OUT is an Unsigned Integer and the converted output value is a negative number, the output is 0.

■ **Program Example**

1. LD



2. ST

INT_VALUE:=TRUNC(EN:=%MX0, IN:=REAL_VALUE);

(1) If the transition condition (%MX0) is on, function TRUNC executes.

(2) If input variable REAL_VALUE (REAL) = 1.6, output variable INT_VALUE (INT) = 1. If REAL_VALUE(REAL) = -1.6, INT_VALUE(INT) = -1.

INPUT (IN) : REAL_VALUE (REAL) = 1.6

↓ (TRUNC)

OUTPUT (OUT) : INT_VALUE (INT) = 1

UDINT_TO_***	UDINT type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
(function block below)	**Input** EN: executes the function in case of 1 IN: Unsigned Double Integer value to be converted **Output** ENO: outputs EN value as it is OUT: type-converted data

```
           UDINT_TO_***
BOOL  ─ EN          ENO ─ BOOL
UDINT ─ IN          OUT ─ *ANY
```

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○	○	○	○	○	○		○	○	○	○		○		○

*ANY: exclude UDINT, DATE and DT from ANY type.

■ **Function**

It converts the IN type and outputs it as OUT.

Function	Output type	Description
UDINT_TO_SINT	SINT	If input is 0~127, normal conversion. Otherwise an error occurs.
UDINT_TO_INT	INT	If input is 0~32,767, normal conversion. Otherwise an error occurs.
UDINT_TO_DINT	DINT	If input is 0~2,147,483,647, normal conversion. Otherwise an error occurs.
UDINT_TO_LINT	LINT	Converts UDINT into LINT type normally.
UDINT_TO_USINT	USINT	If input is 0~255, normal conversion. Otherwise an error occurs.
UDINT_TO_UINT	UINT	If input is 0~65,535, normal conversion. Otherwise an error occurs.
UDINT_TO_ULINT	ULINT	Converts UDINT into ULINT type normally.
UDINT_TO_BOOL	BOOL	Takes the lower 1 bit and converts into BOOL type.
UDINT_TO_BYTE	BYTE	Takes the lower 8 bits and converts into BYTE type.
UDINT_TO_WORD	WORD	Takes the lower 16 bits and converts into WORD type.
UDINT_TO_DWORD	DWORD	Converts into DWORD type without changing the internal bit array.
UDINT_TO_LWORD	LWORD	Converts into LWORD type filling the upper bits with 0.
UDINT_TO_REAL	REAL	Converts UDINT into REAL type. During the conversion, an error caused by the precision may occur.

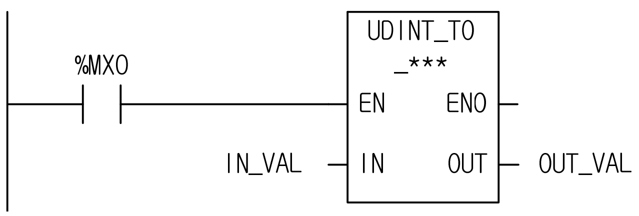
Function	Output type	Description
UDINT_TO_LREAL	LREAL	Converts UDINT into LREAL type. During the conversion, an error caused by the precision may occur.
UDINT_TO_TOD	TOD	Converts into TOD type without changing the internal bit array. However, with a value out of TOD range (TOD#23:59:59.999), _ERR, _LER flags are set and it is alternately converted within the range of TOD.
UDINT_TO_TIME	TIME	Converts into TIME type without changing the internal bit array.
UDINT_TO_STRING	STRING	Converts UDINT into STRING type.

■ **Flag**

Flag	Description
_ERR	If a conversion error occurs, _ERR and _LER flags are set. If an error occurs, take the lower bits as many as a bit number of an output data type and produces the output without changing the internal bit array.

■ **Program Example**

1. LD



2. ST

ST language doesn't support UDINT_TO_***

In case of UDINT_TO_TIME

OUT_VAL := UDINT_TO_TIME(EN:=%MX0, IN:=IN_VAL);

(1) If the input condition (%MX0) is on, function UDINT_TO_*** will be executed.

(2) If input variable IN_VAL (UDINT) = 123, output variable OUT_VAL (TIME) = T#123MS.

INPUT (IN) : IN_VAL (UDINT) = 123

↓

OUTPUT (OUT) : OUT_VAL (TIME) = T#123MS

UINT_TO_***	UINT type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
`` UINT_TO_*** BOOL ─ EN ENO ─ BOOL UINT ─ IN OUT ─ *ANY ```	**Input** EN: executes the function in case of 1 IN: Unsigned Integer value to be converted **Output** ENO: outputs EN value as it is OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○	○	○	○	○		○	○	○	○		○			○

*ANY: exclude UINT, TIME, TOD and DT from ANY type.

■ **Function**

It converts the IN type and outputs it as OUT.

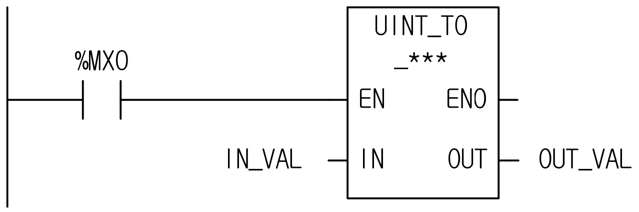
Function	Output type	Description
UINT_TO_SINT	SINT	If input is 0~127, normal conversion. Otherwise an error occurs.
UINT_TO_INT	INT	If input is 0~32,767, normal conversion. Otherwise an error occurs.
UINT_TO_DINT	DINT	Converts UINT into UDINT type normally.
UINT_TO_LINT	LINT	Converts UINT into ULINT type normally.
UINT_TO_USINT	USINT	If input is 0~255, normal conversion. Otherwise an error occurs.
UINT_TO_UDINT	UDINT	Converts UINT into UDINT type normally.
UINT_TO_ULINT	ULINT	Converts UINT into ULINT type.
UINT_TO_BOOL	BOOL	Takes the lower 1 bit and converts into BOOL type.
UINT_TO_BYTE	BYTE	Takes the lower 8 bits and converts into BYTE type.
UINT_TO_WORD	WORD	Converts into WORD type without changing the internal bit array.
UINT_TO_DWORD	DWORD	Converts into DWORD type filling the upper bits with 0.
UINT_TO_LWORD	LWORD	Converts into LWORD type filling the upper bits with 0.
UINT_TO_REAL	REAL	Converts UINT into REAL type.
UINT_TO_LREAL	LREAL	Converts UINT into LREAL type.
UINT_TO_DATE	DATE	Converts into DATE type without changing the internal bit array.
UINT_TO_STRING	STRING	Converts UINT into STRING type.

■ **Flag**

Flag	Description
_ERR	If a conversion error occurs, _ERR and _LER flags are set. If error occurs, it takes as many lower bits as a bit number of output type and produces an output without changing its internal bit array.

■ **Program Example**

1. LD



2. ST

ST language doesn't support UINT_TO_***

In case of UINT_TO_WORD

```
OUT_VAL := UINT_TO_WORD(EN:=%MX0, IN:=IN_VAL);
```

(1) If the input condition (%MX0) is on, function UINT_TO_*** executes.

(2) If input variable IN_VAL (UINT) = 255 (2#0000_0000_1111_1111), output variable OUT_VAL (WORD) = 2#0000_0000_1111_1111.

INPUT (IN) : IN_VAL (UINT) = 255	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
↓ (UINT_TO_WORD)																
OUTPUT (OUT) : OUT_VAL (WORD) = 16#FF	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

ULINT_TO_***	ULINT type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
ULINT_TO_*** block: BOOL — EN, ENO — BOOL; ULINT — IN, OUT — *ANY	**Input** EN: executes the function in case of 1 IN: Unsigned Long Integer value to be converted **Output** ENO: outputs EN value as it is OUT: type-converted data

```
              ULINT_TO_***
BOOL  ─ EN               ENO ─ BOOL
ULINT ─ IN               OUT ─ *ANY
```

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○	○	○	○	○		○	○	○	○		○			○

*ANY: exclude UINT, TIME, TOD and DT from ANY type.

■ **Function**

It converts the IN type and outputs it as OUT.

Function	Output type	Description
ULINT_TO_SINT	SINT	If input is 0~127, normal conversion. Otherwise an error occurs.
ULINT_TO_INT	INT	If input is 0~32,767, normal conversion. Otherwise an error occurs.
ULINT_TO_DINT	DINT	If input is 0~2^{31}-1, normal conversion. Otherwise an error occurs.
ULINT_TO_LINT	LINT	If input is 0~2^{63}-1, normal conversion. Otherwise an error occurs.
ULINT_TO_USINT	USINT	If input is 0~255, normal conversion. Otherwise an error occurs.
ULINT_TO_UINT	UINT	If input is 0~65,535, normal conversion. Otherwise an error occurs.
ULINT_TO_UDINT	UDINT	If input is 0~2^{32}-1, normal conversion. Otherwise an error occurs.
ULINT_TO_BOOL	BOOL	Takes the lower 1 bit and converts into BOOL type.
ULINT_TO_BYTE	BYTE	Takes the lower 8 bits and converts into BYTE type.
ULINT_TO_WORD	WORD	Takes the lower 16 bits and converts into WORD type.
ULINT_TO_DWORD	DWORD	Takes the lower 32 bits and converts into DWORD type.
ULINT_TO_LWORD	LWORD	Converts into LWORD type without changing the internal bit array.
ULINT_TO_REAL	REAL	Converts ULINT into REAL type. During the conversion, an error caused by the precision may occur.
ULINT_TO_LREAL	LREAL	Converts ULINT into LREAL type. During the conversion, an error caused by the precision may occur.

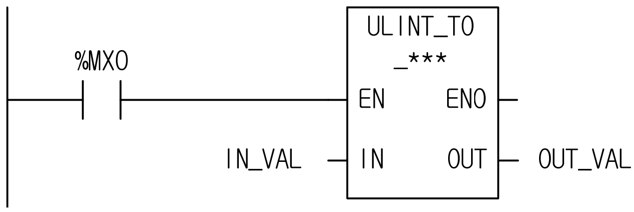
Function	Output type	Description
ULINT_TO_STRING	STRING	Converts ULINT into STRING type.

■ **Flag**

Flag	Description
_ERR	If a conversion error occurs, _ERR and _LER flags are set. If error occurs, it takes as many lower bits as a bit number of output type and produces an output without changing its internal bit array

■ **Program Example**

1. LD



2. ST

ST language doesn't support ULINT_TO_***
In case of ULINT_TO_LINT

OUT_VAL := ULINT_TO_LINT(EN:=%MX0, IN:=IN_VAL);

(1) If the input condition (%MX0) is on, function ULINT_TO_*** executes.
(2) If input variable IN_VAL (ULINT) = 123,567,899, then output variable OUT_VAL (LINT) = 123,567,899.

INPUT (IN) : IN_VAL (ULINT) = 123,567,899

↓ (ULINT_TO_LINT)

OUTPUT (OUT) : OUT_VAL (LINT) = 123,567,899

USINT_TO_***	USINT type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
USINT_TO_*** BOOL — EN ENO — BOOL USINT — IN OUT — *ANY	**Input** EN: executes the function in case of 1 IN: To convert Unsigned Short Integer value. **Output** ENO: outputs EN value as it is OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○	○	○	○		○	○	○	○	○					○

*ANY: exclude USINT, TIME, DATE, TOD and DT from ANY type.

■ **Function**

It converts the IN type and outputs it as OUT.

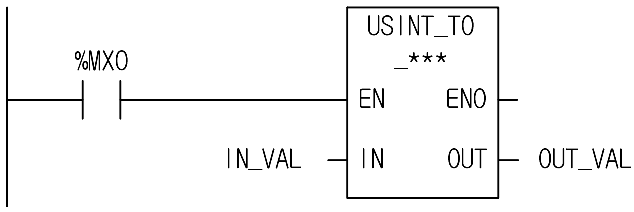
Function	Output type	Description
USINT_TO_SINT	SINT	If input is 0~127, normal conversion. Otherwise an error occurs.
USINT_TO_INT	INT	Converts USINT into INT type normally.
USINT_TO_DINT	DINT	Converts USINT into DINT type normally.
USINT_TO_LINT	LINT	Converts USINT into LINT type normally.
USINT_TO_UINT	UINT	Converts USINT into UINT type normally.
USINT_TO_UDINT	UDINT	Converts USINT into UDINT type normally.
USINT_TO_ULINT	ULINT	Converts USINT into ULINT type normally.
USINT_TO_BOOL	BOOL	Takes the lower 1 bit and converts into BOOL type.
USINT_TO_BYTE	BYTE	Converts into BYTE type without changing the internal bit array.
USINT_TO_WORD	WORD	Converts into WORD type filling the upper bits with 0.
USINT_TO_DWORD	DWORD	Converts into DWORD type filling the upper bits with 0.
USINT_TO_LWORD	LWORD	Converts into LWORD type filling the upper bits with 0.
USINT_TO_REAL	REAL	Converts USINT into REAL type.
USINT_TO_LREAL	LREAL	Converts USINT into LREAL type.
USINT_TO_STRING	STRING	Converts USINT into STRING type.

■ **Flag**

Flag	Description
_ERR	If a conversion error occurs, _ERR and _LER flags are set. If error occurs, it takes as many lower bits as a bit number of output type and produces an output without changing its internal bit array.

■ **Program Example**

1. LD



2. ST

ST language doesn't support USINT_TO_***

In case of USINT_TO_SINT

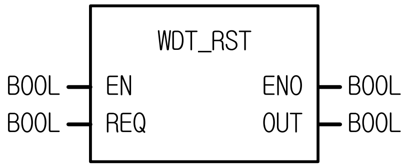
OUT_VAL := USINT_TO_SINT(EN:=%MX0, IN:=IN_VAL);

(1) If the input condition (%MX0) is on, function ULINT_TO_*** executes.

(2) If input variable IN_VAL (USINT) = 123, output variable OUT_VAL (SINT) = 123.

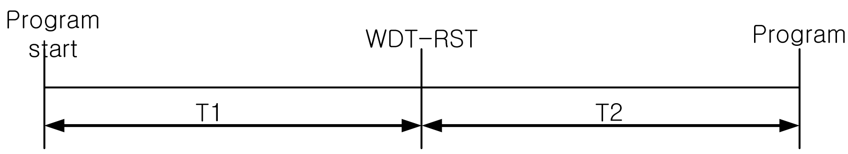
INPUT (IN) : IN_VAL (USINT) = 123(16#7B)	0	1	1	1	1	0	1	1
(UINT_TO_SINT)								
OUTPUT (OUT) : OUT_VAL (SINT) = 123(16#7B)	0	1	1	1	1	0	1	1

WDT_RST	**Initialize Watch_Dog timer**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 REQ: requires to initialize watchdog timer **Output** ENO: outputs EN value as it is OUT: After Watch_Dog timer initialization, output is 1

■ **Function**

1. It resets Watch-Dog Timer among the programs.
2. Available to use in case that scan time exceeds Watch-Dog Time set by the condition in the program.
3. If scan time exceeds the scan Watch_Dog Time, change the scan time with the setting value of scan Watch_Dog Timer.
4. Care must be taken so that either the time from 0 line of program to WDT_RST function T1 or the time from WDT_RST function to the time by the end of program T2 does not exceed the setting value of scan Watch_Dog Timer.

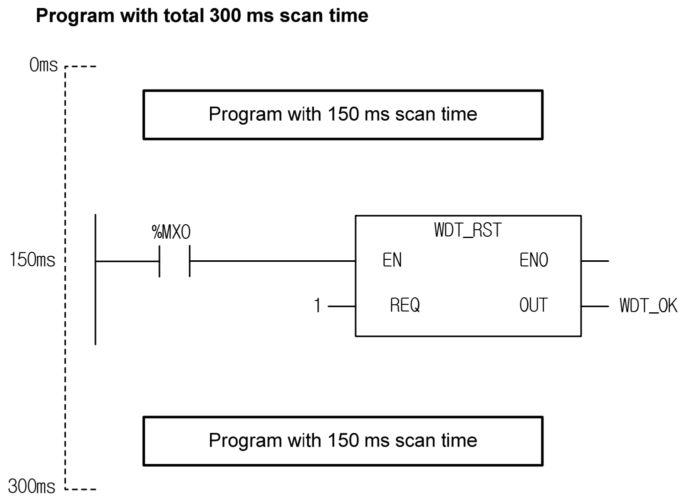


5. WDT_RST function is available to use several times during 1 scan.

■ **Program Example**

This is the program that the time to execute the program becomes 300ms according to the transition condition in the program of which scan Watch_Dog timer is set as 200ms.

1. LD



2. ST

WDT_OK := WDT_RST(EN:=%MX0, REQ:=%MX0);

(1) If the transition condition (%MX0) is on, function WDT-RST executes.

(2) If WDT-RST function executes, it is available to set the program that extends the scan time to 300ms according to the transition condition of program within the scan Watch_Dog Time (200ms).

WORD_TO_***	WORD type conversion	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
<pre> WORD_TO_*** BOOL ─ EN ENO ─ BOOL WORD ─ IN OUT ─ *ANY</pre>	**Input** EN: executes the function in case of 1 IN: Bit string to be converted (16 bit) **Output** ENO: outputs EN value as it is OUT: type-converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○		○	○	○	○	○	○	○	○	○	○				○			○

*ANY: exclude WORD, REAL, LREAL, TIME, TOD and DT from ANY type.

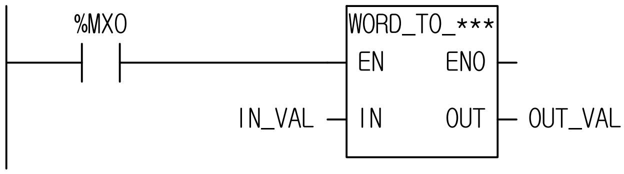
■ **Function**

It converts the IN type and outputs it as OUT.

Function	Output type	Description
WORD _TO_SINT	SINT	Takes the lower 8 bits and converts into SINT type.
WORD _TO_INT	INT	Converts into INT type without changing the internal bit array.
WORD _TO_DINT	DINT	Converts into DINT type filling the upper bits with 0.
WORD _TO_LINT	LINT	Converts into LINT type filling the upper bits with 0.
WORD _TO_USINT	USINT	Takes the lower 8 bits and converts into SINT type.
WORD _TO_UINT	UINT	Converts into INT type without changing the internal bit array.
WORD _TO_UDINT	UDINT	Converts into DINT type filling the upper bits with 0.
WORD _TO_ULINT	ULINT	Converts into LINT type filling the upper bits with 0.
WORD _TO_BOOL	BOOL	Takes the lower 1 bit and converts into BOOL type.
WORD _TO_BYTE	BYTE	Takes the lower 8 bits and converts into SINT type.
WORD _TO_DWORD	DWORD	Converts into DWORD type filling the upper bits with 0.
WORD _TO_LWORD	LWORD	Converts into LWORD type filling the upper bits with 0.
WORD _TO_DATE	DATE	Converts into DATE type without changing the internal bit array.
WORD _TO_STRING	STRING	Converts WORD into STRING type.

■ **Program Example**

1. LD



2. ST

ST language doesn't support WORD_TO_***

In case of WORD_TO_INT

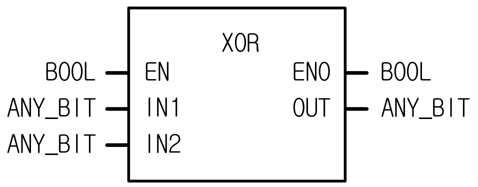
```
OUT_VAL := WORD_TO_INT(EN:=%MX0, IN:=IN_VAL);
```

(1) If the input condition (%MX0) is on, function WORD-TO-*** executes.

(2) If input variable IN_VAL (WORD) = 2#0001_0001_0001_0001, output variable OUT_VAL (INT) = 4,096 + 256 + 16 + 1 = 4,369

INPUT (IN) : IN_VAL (WORD) = 16#1111	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
↓ (WORD_TO_INT)																
OUTPUT (OUT) : OUT_VAL (INT) = 4,369 (16#1111)	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

XOR	Exclusive OR	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN1: the value to be XOR IN2: the value to be XOR Input variable number can be extended up to 8. **Output** ENO: outputs EN value as it is OUT: the result of XOR operation IN1, IN2, OUT must be of all the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○															
	OUT	○	○	○	○	○															

■ **Function**

1. Do XOR operation for IN1 and IN2 per bit and to produces OUT.

   IN1 1111 0000

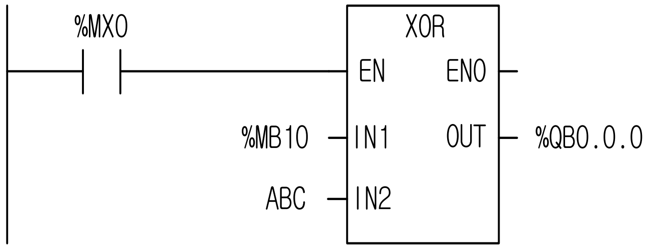
   XOR

   IN2 1010 1010

   OUT 0101 1010

■ **Program Example**

1. LD

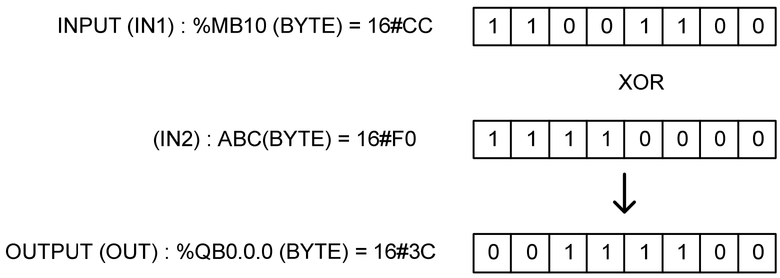


2. ST

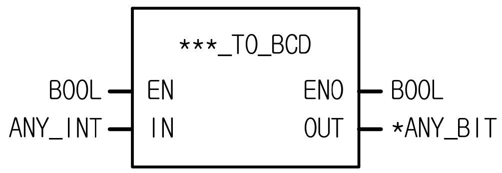
ST language doesn't support XOR
In case of XOR2_BYTE

%QB0.0.0 := XOR2_BYTE(EN:=%MX0, IN1:=%MB10, IN2:=ABC);

(1) If the transition condition (%MX0) is on, function XOR executes.
(2) If input variable %MB10 = 1100_1100, ABC = 1111_0000, the result of XOR operation for two inputs is %QB0.0.0 = 0011_1100.



***_TO_BCD	Converting ANY Type to BCD type	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: execute the function in case of 1 IN: enter ANY_BIT with BCD type data **Output** ENO: outputs EN value as it is OUT: type converted data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN						○	○	○	○	○	○	○	○							
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL type from ANY_BIT.

■ **Function**

It converts the IN type and outputs it as OUT

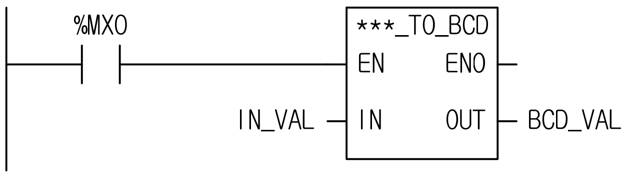
Function	Input type	Output type	Description
SINT_TO_BCD_BYTE	SINT	BYTE	Converting ANY type to BCD type. Normally converted as long as it is BCD value. (if input data type is WORD, the values, 0～16#9999 are normally converted)
INT_TO_BCD_WORD	INT	WORD	
DINT_TO_BCD_DWORD	DINT	DWORD	
LINT_TO_BCD_LWORD	LINT	LWORD	
USINT_TO_BCD_BYTE	USINT	BYTE	
UINT_TO_BCD_WORD	UINT	WORD	
UDINT_TO_BCD_DWORD	UDINT	DWORD	
ULINT_TO_BCD_LWORD	ULINT	LWORD	

■ **Flag**

Flag	Description
_ERR	If IN is not the data within BCD range, output is 0; _ERR and _LER flags are set.

■ **Program Example**

1. LD



2. ST

ST language doesn't support ***_TO_BCD

In case of SINT_TO_BCD_BYTE

BCD_VAL := SINT_TO_BCD_BYTE(EN:=%MX0, IN:=IN_VAL);

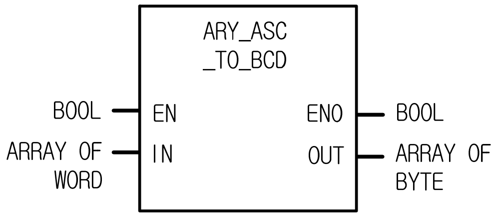
(1) If the execution condition (%MX0) is on, SINT_TO_BCD function executes.
(2) If IN_VAL (SINT type) = 16#22(2#0001_0110), BCD_VAL (BYTE type) = 16#22 (2#0010_0010) declared as a function's output variable is produced.

INPUT (IN) :IN_VAL (SINT) = 22	0	0	0	1	0	1	1	0
↓SINT_TO_BCD_BYTE								
OUTPUT (OUT): BCD_VAL (BYTE) = 16#22	0	0	1	0	0	0	1	0

Chapter 8. Application Functions

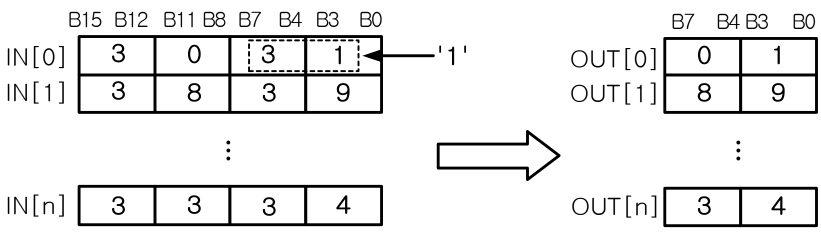
This chapter describes application functions unlike the basic functions described in the previous chapter.

ARY_ASC_TO_BCD	**Input : ASCII Array, Output: BCD Array**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: ASCII Array input **Output** ENO: without an error, it is 1 OUT: BCD Array output

■ **Function**

It converts a word array input (ASCII data) to a byte array output (BCD data).



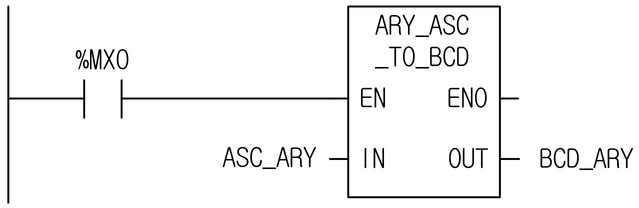
■ **Flag**

Flag	Description
_ERR	If the number of each input/output array is different, there's no change in OUT data, and _ERR and _LER flags are set. If the elements of IN array are not between 0 and 9 (hexadecimal), its responding elements of OUT array are 16#00 (while other elements of IN1 are normally converted), and _ERR and _LER flags are set.

☆ If the number of each input/output array is different, _ERR and _LER flags occur; if output array variable is omitted, the number of array is regarded as '0' and _ERR and _LER flags occur.

■ **Program Example**

1. LD



2. ST

CD_ARY := ARY_ASC_TO_BCD(EN:=%MX0, IN:=ASC_ARY);

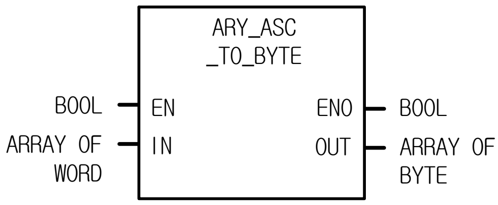
(1) If the transition condition (%MX0) is on, ARY_ASC_TO_BCD function executes.
(2) If the input ASC_ARY data is

ASC_ARY[0]	16#3031
ASC_ARY[1]	16#3839
ASC_ARY[2]	16#3334

Output BCD_ARY data is as follows.

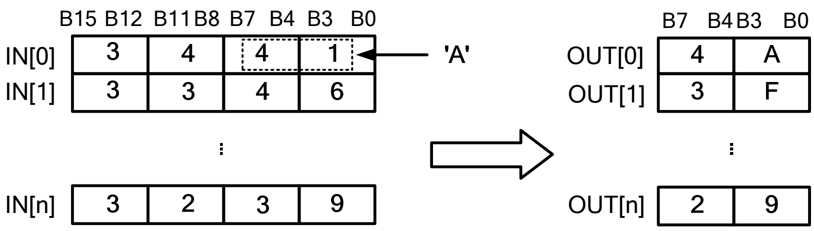
BYTE_ARY[0]	01
BYTE_ARY[1]	89
BYTE_ARY[2]	34

ARY_ASC_TO_BYTE	Input: ASCII Array, Output: BYTE Array	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: ASCII Array input **Output** ENO: without an error, it is 1 OUT: BYTE Array output

■ **Function**

It converts a word array input (ASCII data) to a byte array output (hexadecimal).



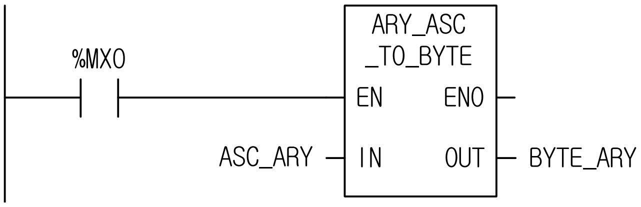
■ **Flag**

Flag	Description
_ERR	If the number of each input/output array is different, there's no change in OUT data, and _ERR and _LER flags are set. If the elements of IN array are not between 0 and F (hexadecimal), its responding elements of OUT array are 0 (while other elements of IN1 are normally converted), and _ERR and _LER flags are set.

☆ If the number of each input/output array is different, _ERR and _LER flags occur; if output array variable is omitted, the number of array is regarded as '0' and _ERR and _LER flags occur.

■ **Program Example**

1. LD



2. ST

YTE_ARY := ARY_ASC_TO_BYTE(EN:=%MX0, IN:=ASC_ARY);

(1) If the transition condition is (%MX0) is on, ARY_ASC_TO_BYTE function executes.

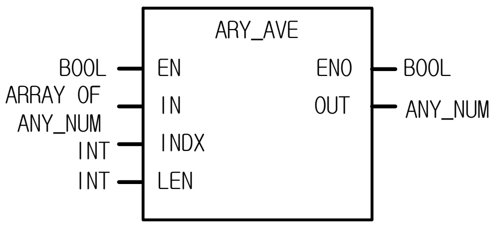
(2) If Input ASC_ARY is as below;

ASC_ARY[0]	16#3441
ASC_ARY[1]	16#3346
ASC_ARY[2]	16#3239

Output BYTE_ARY data is as follows.

BYTE_ARY[0]	4A
BYTE_ARY[1]	3F
BYTE_ARY[2]	29

ARY_AVE	Finds an average of an array	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: data array for average INDX: starting point to average in an array LEN: number of array elements for average **Output** ENO: without an error, it will be 1 OUT: average of an array

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN						○	○	○	○	○	○	○	○	○	○					
	OUT						○	○	○	○	○	○	○	○	○	○					

■ **Function**

1. ARY_AVE function finds an average for a specified length of an array.
2. Input and output array is the same type.
3. If LEN is a negative number, it finds an average between INDX (Array index) and 'INDX – |LEN|'. Its output is rounded off.

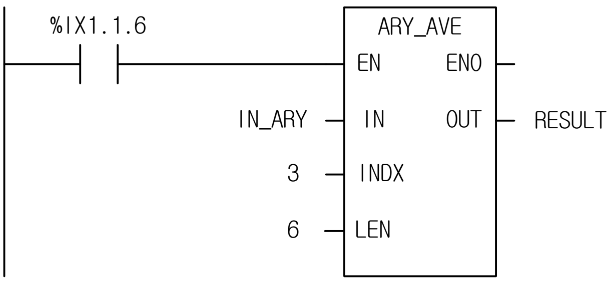
Function	Output type	Description
ARY_AVE	SINT	Finds an average for SINT value (decimal is rounded off)
ARY_AVE	INT	Finds an average for INT value (decimal is rounded off)
ARY_AVE	DINT	Finds an average for DINT value (decimal is rounded off)
ARY_AVE	LINT	Finds an average for LINT value (decimal is rounded off)
ARY_AVE	USINT	Finds an average for USINT value (decimal is rounded off)
ARY_AVE	UINT	Finds an average for UINT value (decimal is rounded off)
ARY_AVE	UDINT	Finds an average for UDINT value (decimal is rounded off)
ARY_AVE	ULINT	Finds an average for ULINT value (decimal is rounded off)
ARY_AVE	REAL	Finds an average for REAL value.
ARY_AVE	LREAL	Finds an average for LREAL value.

■ **Flag**

Flag	Description
_ERR	If it is designated beyond the array range, _ERR and _LER flags are set. If an error occurs, the output is 0. ※ An error occurs when: INDX < 0 or INDX > max. number of IN INDX + LEN > max. number of IN

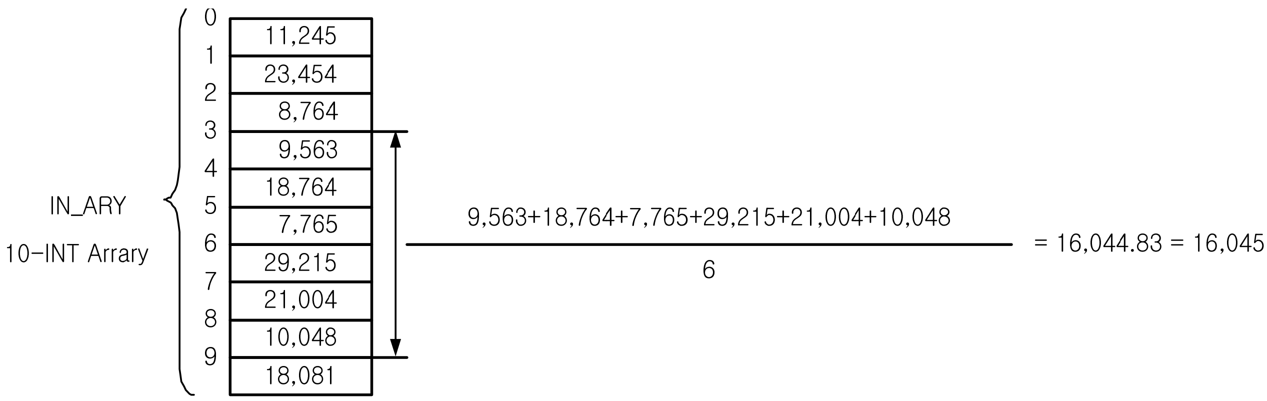
■ **Program Example**

1. LD



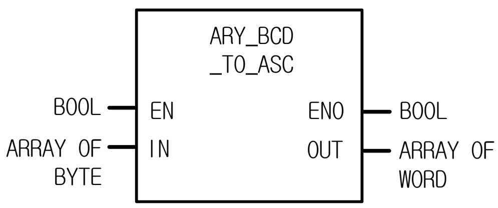
2. ST

RESULT := ARY_AVE(EN:=%IX1.1.6, IN:=IN_ARY, INDX:=3, LEN:=6);



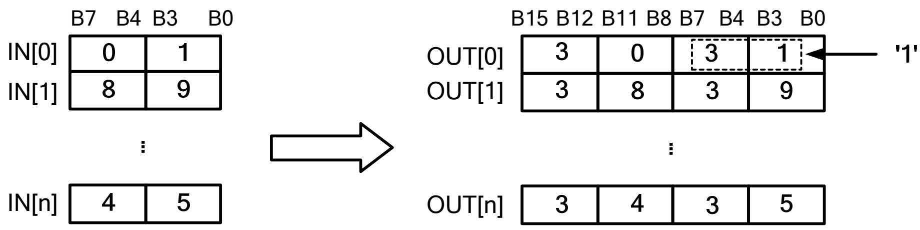
(1) If input transition condition (%IX1.1.6) is On, ARY_AVE_INT function executes.
(2) If the value within ARRAY is as same as the above-presented picture, it calculates the average value of 6 from the 3rd of Array Index.
(3) Since the mean value is 16,044.8 but its output type is INT, it rounds off and outputs 16,045.

ARY_BCD_TO_ASC	Input: BCD Array, Output: ASCII Array	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: BCD array input **Output** ENO: without an error, it is 1 OUT: ASCII array output

■ **Function**

It converts a byte array input (BCD) to a word array (ASCII).



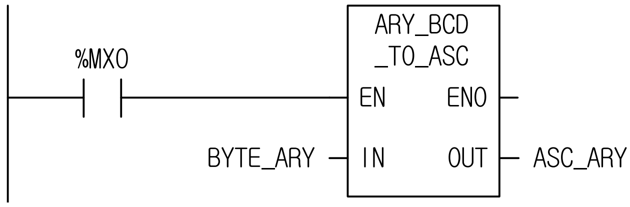
■ **Flag**

Flag	Description
_ERR	If the number of each input/output array is different, there's no change in OUT data, and _ERR and _LER flags are set. If the elements of IN array are not between 0 and 9 (hexadecimal), its responding elements of OUT array are 0 (while other elements of IN1 are normally converted), and _ERR and _LER flags are set.

☆ If the number of each input/output array is different, _ERR and _LER flags occur; if output array variable is omitted, the number of array is regarded as '0' and _ERR and _LER flags occur.

■ **Program Example**

1. LD



2. ST

ASC_ARY := ARY_BCD_TO_ASC(EN:=%MX0, IN:=BCD_ARY);

(1) If the transition condition (%MX0) is on, ARY_BCD_TO_ASC function executes.

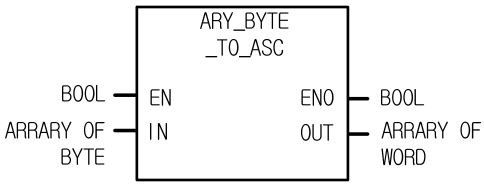
(2) If the input BCD_ARY is as below:

BYTE_ARY[0]	01
BYTE_ARY[1]	89
BYTE_ARY[2]	45

Output ASC_ARY is as follows:

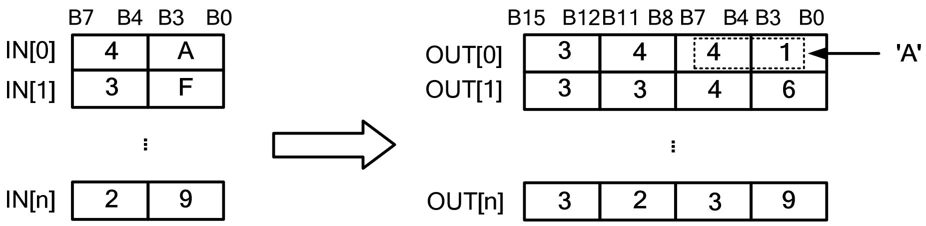
ASC_ARY[0]	3031
ASC_ARY[1]	3839
ASC_ARY[2]	3435

ARY_BYTE_TO_ASC	Input: BYTE Array, Output: ASCII Array	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: BYTE array input **Output** ENO: without an error, it is 1 OUT: ASCII Array output

■ **Function**

It converts a byte array input (HEX) to a word array (ASCII).



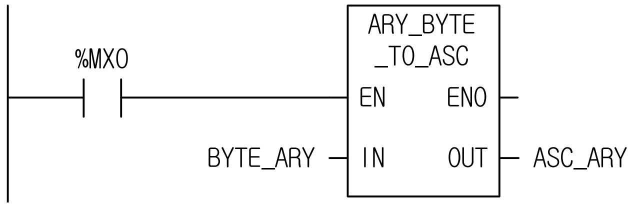
■ **Flag**

Flag	Description
_ERR	If the number of each input/output array is different, there's no change in OUT data, and _ERR and _LER flags are set.

☆ If the number of each input/output array is different, _ERR and _LER flags occur; if output array variable is omitted, the number of array is regarded as '0' and _ERR and _LER flags occur.

■ **Program Example**

1. LD



2. ST

ASC_ARY := ARY_BYTE_TO_ASC(EN:=%MX0, IN:=BYTE_ARY);

(1) If the transition condition (%MX0) is on, ARY_BYTE_TO_ASC function executes.

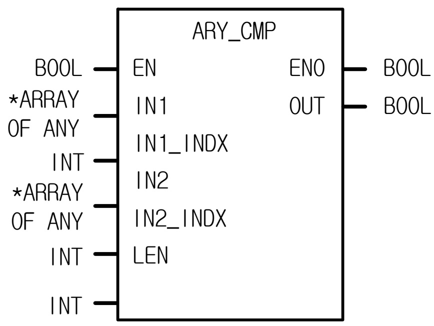
(2) If the input BYTE_ARY is as below:

BYTE_ARY[0]	4A
BYTE_ARY[1]	3F
BYTE_ARY[2]	29

The output ASC_ARY is as follows:

ASC_ARY[0]	3441
ASC_ARY[1]	3346
ASC_ARY[2]	3239

ARY_CMP	**Array comparison**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: first array to compare IN1_INDX : starting point in 1st array for comparison IN2: second array to compare IN2_INDX : starting point in 2nd array for comparison LEN: number of elements to compare **Output** ENO: without an error, it is 1 OUT: if two arrays are equal, it is 1

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ARRAY OF ANY: exclude STRING from ANY type.

■ **Function**

1. It compares two arrays whether they have the same value.
2. If LEN is a negative number, it compares two arrays between IN*_INDX (Array INDX) and “Array INDX – |LEN|.”

Function	Input array type	Description
ARY_CMP	BOOL	Compares two BOOL Arrays.
ARY_CMP	BYTE	Compares two BYTE Arrays.
ARY_CMP	WORD	Compares two WORD Arrays.
ARY_CMP	DWORD	Compares two DWORD Arrays.
ARY_CMP	LWORD	Compares two LWORD Arrays.
ARY_CMP	SINT	Compares two SINT Arrays.
ARY_CMP	INT	Compares two INT Arrays.
ARY_CMP	DINT	Compares two DINT Arrays.
ARY_CMP	LINT	Compares two LINT Arrays.
ARY_CMP	USINT	Compares two USINT Arrays.

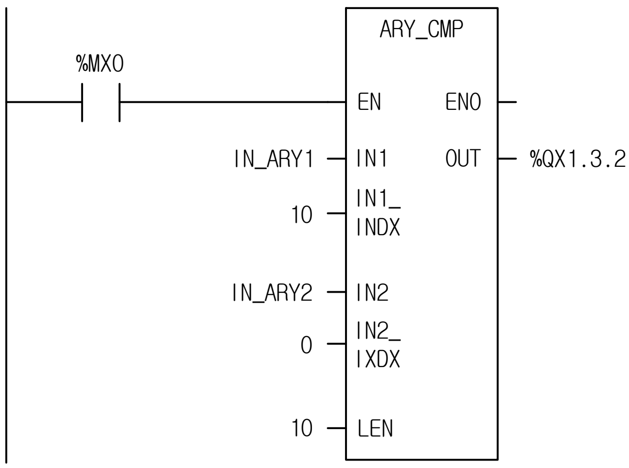
Function	Input array type	Description
ARY_CMP	UINT	Compares two UINT Arrays.
ARY_CMP	UDINT	Compares two UDINT Arrays.
ARY_CMP	ULINT	Compares two ULINT Arrays.
ARY_CMP	REAL	Compares two REAL Arrays.
ARY_CMP	LREAL	Compares two LREAL Arrays.
ARY_CMP	TIME	Compares two TIME Arrays.
ARY_CMP	DATE	Compares two DATE Arrays.
ARY_CMP	TOD	Compares two TOD Arrays.
ARY_CMP	DT	Compares two DT Arrays.

■ **Flag**

Flag	Description
_ERR	If it is designated beyond the array range, _ERR and _LER flags are set. ※ An error occurs when: IN1_INDX < 0 or IN1_INDX > max. number of IN1 IN2_INDX < 0 or IN2_INDX > max. number of IN2 IN1_INDX + LEN ≥ max. number of IN1 IN2_INDX + LEN ≥ max. number of IN2

■ **Program Example**

1. LD



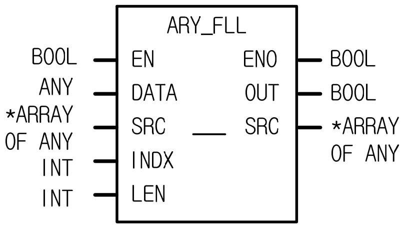
2. ST

%QX1.3.2 := ARY_CMP(EN:=%MX0, IN1:=IN_ARY1, IN1_INDX:=10, IN2:=IN_ARY2, IN2_INDX:=0, LEN:=10);

(1) If the input transition condition (%MX0) is on, ARY_CMP function executes.

(2) When IN_ARY1 is a time array with 100 elements and IN_ARY2 is a time array with 10 elements, if the elements from 11th to 20th of IN_ARY1 and the elements of IN_ARY 2 are equal, the output %Q1.3.2 is on.

ARY_FLL	Filling an array with data	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 DATA: the data to fill an array INDX: starting point of an array to be filled LEN: number of array elements to be filled **Output** ENO: without an error, it is 1 OUT: without an error, it is 1 **In/Out** SRC: an array to be filled

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	DATA	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	SRC	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ARRAY OF ANY: exclude STRING from ANY type.

■ **Function**

1. It fills an array with the input data.
2. If LEN is minus, it fills an array from INDX to "INDX – |LEN|."

Function	In/out array type	Description
ARY_FLL	BOOL	Fills a BOOL Array with the input data.
ARY_FLL	BYTE	Fills a BYTE Array with the input data.
ARY_FLL	WORD	Fills a WORD Array with the input data.
ARY_FLL	DWORD	Fills a DWORD Array with the input data.
ARY_FLL	LWORD	Fills a LWORD Array with the input data.
ARY_FLL	SINT	Fills a SINT Array with the input data.
ARY_FLL	INT	Fills a INT Array with the input data.
ARY_FLL	DINT	Fills a DINT Array with the input data.
ARY_FLL	LINT	Fills a LINT Array with the input data.
ARY_FLL	USINT	Fills a USINT Array with the input data.
ARY_FLL	UINT	Fills a UINT Array with the input data.
ARY_FLL	UDINT	Fills a UDINT Array with the input data.

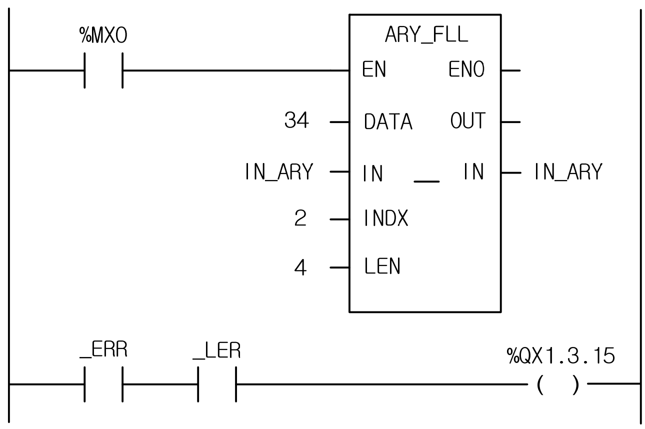
Function	In/out array type	Description
ARY_FLL	ULINT	Fills a ULINT Array with the input data.
ARY_FLL	REAL	Fills a REAL Array with the input data.
ARY_FLL	LREAL	Fills a LREAL Array with the input data.
ARY_FLL	TIME	Fills a TIME Array with the input data.
ARY_FLL	DATE	Fills a DATE Array with the input data.
ARY_FLL	TOD	Fills a TOD Array with the input data.
ARY_FLL	DT	Fills a DT Array with the input data.

■ **Flag**

Flag	Description
_ERR	If it is designated beyond the array range, _ERR and _LER flags are set. If an error occurs, there's no change in arrays and OUT is Off. ※ An error occurs when: INDX < 0 or INDX > max. element number of IN INDX + LEN ≥ max. element number of IN

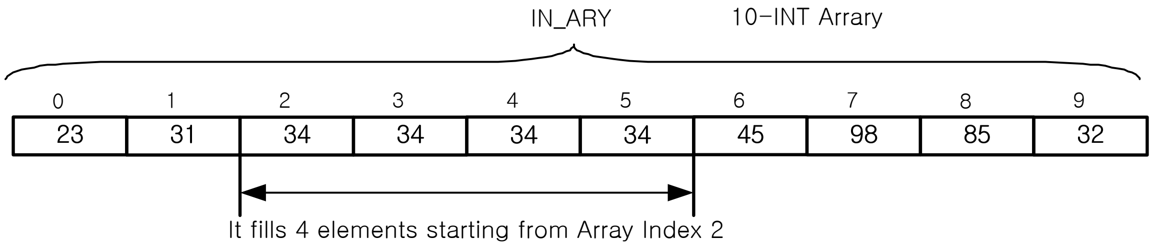
■ **Program Example**

1. LD



2. ST

```
OUT :=ARY_FLL(EN:=%MX0, DATA:=34, SRC:=IN_ARY, INDX:=2, LEN:=4);
IF _ERR = 1 AND _LER = 1 THAN %QX1.3.15 := 1;
END_IF;
```

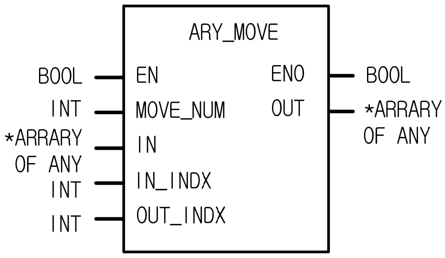


(1) If input condition (%MX0) is on, ARY_FLL function executes.

(2) It fills 4 elements of IN_ARY starting from INDX with 34.

(3) If LEN is 9, it is beyond the array range and an error occurs; _ERR and _LER flags are on and the output (%QX1.13.15) is on.

ARY_MOVE	Array move	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN : executes the function in case of 1 MOVE_NUM: array number to move IN: array variable to move (STRING type, unavailable) IN_INDX: starting pointer of array to move OUT_INDX: starting pointer of array to be moved **Output** ENO: without an error, it is 1 OUT: array variable to be moved (STRING type, unavailable)

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ARRAY OF ANY: exclude STRING from ANY type.

■ **Function**

1. If EN is 1, it moves IN data to OUT.
2. It copies MOVE_NUM elements of IN (from IN_INDX) and pastes it in OUT (from OUT_INDX).
3. IN and OUT are the same data type (the number of each array can be different).
4. The data size is as follows:

Data size	Variable type
1 Bit	BOOL
8 Bit	BYTE/ SINT/ USINT
16 Bit	WORD / INT / UINT / DATE
32 Bit	DWORD / DINT / UDINT / TIME / TOD
64 Bit	DT

■ **Flag**

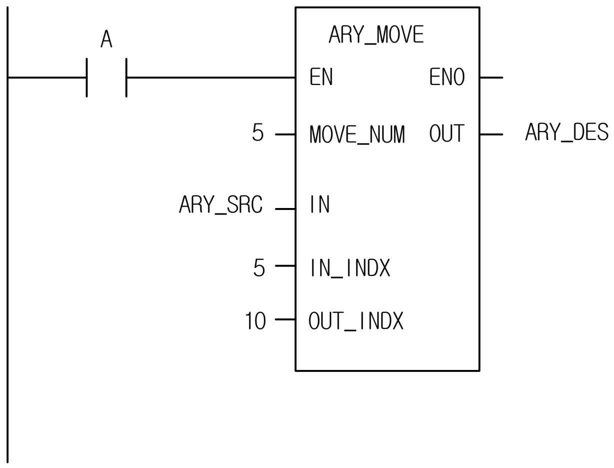
Flag	Description
_ERR	An error occurs when IN and OUT array data sizes are different. An error occurs when 1) the array number of IN Array < (IN_INDX + MOVE_NUM) and 2) the array number of OUT Array < (OUT_INDX + MOVE_NUM). Then ARY_MOVE function is not executed, and OUT is 0. EN0 is Off and _ERR and _LER flags are set.

☆ If the number of each input/output array is different, _ERR and _LER flags occur; if output array variable is omitted, the number of array is regarded as '0' and _ERR and _LER flags occur.

■ **Program Example**

Variable name	Variable type	Array number
ARY_SRC	INT	10
ARY_DES	WORD	15

1. LD



2. ST

ARY_DES := ARY_MOVE(EN:=A, MOVE_NUM:=5, IN:=ARY_SRC, IN_INDX:=5, OUT_INDX:=10);

(1) If the transition condition (A) is on, ARY_MOVE function executes.

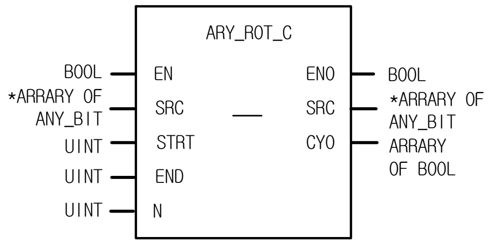
(2) It moves 5 elements from ARY_SRC[5] to ARY_DES[10].

Now the data type of ARY_DES is WORD, it's a hexadecimal.

Before				After			
ARY_SRC[0]	0	ARY_DES[0]	16#0	ARY_SRC[0]	0	ARY_DES[0]	16#0
ARY_SRC[1]	11	ARY_DES[1]	16#1	ARY_SRC[1]	11	ARY_DES[1]	16#1
ARY_SRC[2]	22	ARY_DES[2]	16#2	ARY_SRC[2]	22	ARY_DES[2]	16#2
ARY_SRC[3]	33	ARY_DES[3]	16#3	ARY_SRC[3]	33	ARY_DES[3]	16#3

Before				After			
ARY_SRC[4]	44	ARY_DES[4]	16#4	ARY_SRC[4]	44	ARY_DES[4]	16#4
ARY_SRC[5]	55	ARY_DES[5]	16#5	ARY_SRC[5]	55	ARY_DES[5]	16#5
ARY_SRC[6]	66	ARY_DES[6]	16#6	ARY_SRC[6]	66	ARY_DES[6]	16#6
ARY_SRC[7]	77	ARY_DES[7]	16#7	ARY_SRC[7]	77	ARY_DES[7]	16#7
ARY_SRC[8]	88	ARY_DES[8]	16#8	ARY_SRC[8]	88	ARY_DES[8]	16#8
ARY_SRC[9]	99	ARY_DES[9]	16#9	ARY_SRC[9]	99	ARY_DES[9]	16#9
-	-	ARY_DES[10]	16#A	-	-	ARY_DES[10]	16#37
-	-	ARY_DES[11]	16#B	-	-	ARY_DES[11]	16#42
-	-	ARY_DES[12]	16#C	-	-	ARY_DES[12]	16#4D
-	-	ARY_DES[13]	16#D	-	-	ARY_DES[13]	16#58
-	-	ARY_DES[14]	16#E	-	-	ARY_DES[14]	16#63

ARY_ROT_C	Array Bit Rotate with Carry	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 STRT: starting bit to rotate END: ending bit to rotate N: number to rotate **Output** ENO: without an error, it is 1 CYO: Output Carry bit Array after rotate **Input/Output** SRC: Source Array to rotate

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	SRC		○	○	○	○															

*ARRAY OF ANY_BIT: exclude BOOL from ANY_BIT type.

■ **Function**

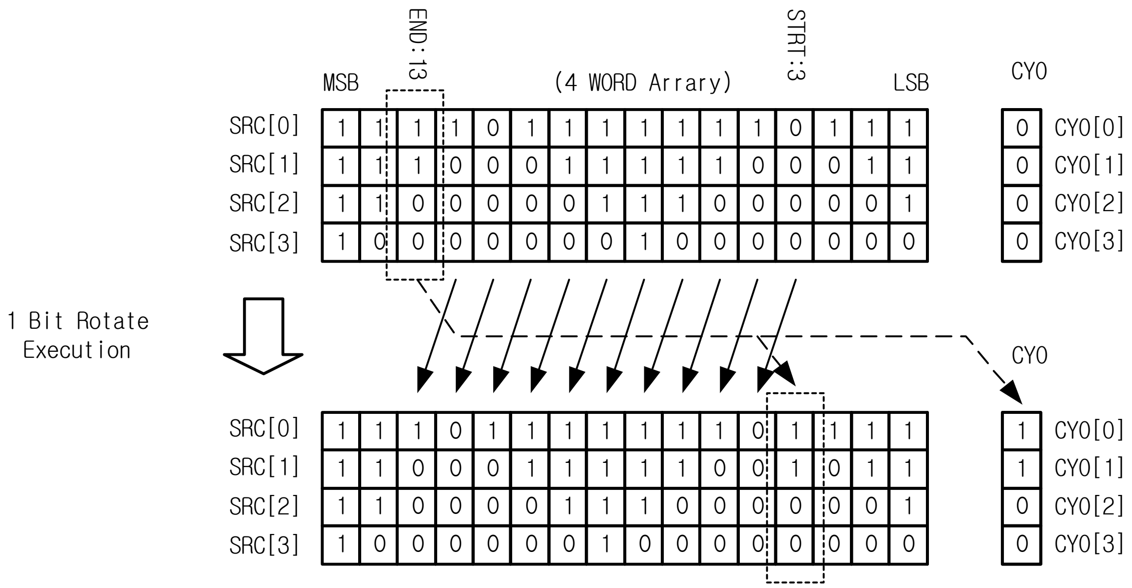
1. It rotates as many bits of array elements as they're specified.

2. Setting

   - Scope: it sets a rotation scope with STRT and END.

   - Rotation direction and time: it rotates N times from STRT to END.

   - Output: the result is stored in configured array in SRC and a bit array data from END to STRT is written at CYO.



Function	In/Out Array Type	Description
ARY_ROT_C	BYTE	It rotates elements of an array as many bits as specified.
ARY_ROT_C	WORD	
ARY_ROT_C	DWORD	
ARY_ROT_C	LWORD	

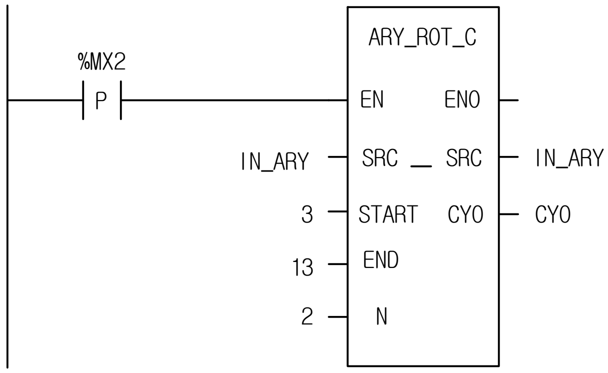
■ **Flag**

Flag	Description
_ERR	If the number of SRC and CYO Arrays are different, _ERR and _LER flags are set. If STRT and END are out of bit range of SRC, an error occurs. When an error occurs, there's no change in SRC and CYO.

☆ If the number of each input/output array is different, _ERR and _LER flags occur; if output array variable is omitted, the number of array is regarded as '0' and _ERR and _LER flags occur.

■ **Program Example**

1. LD



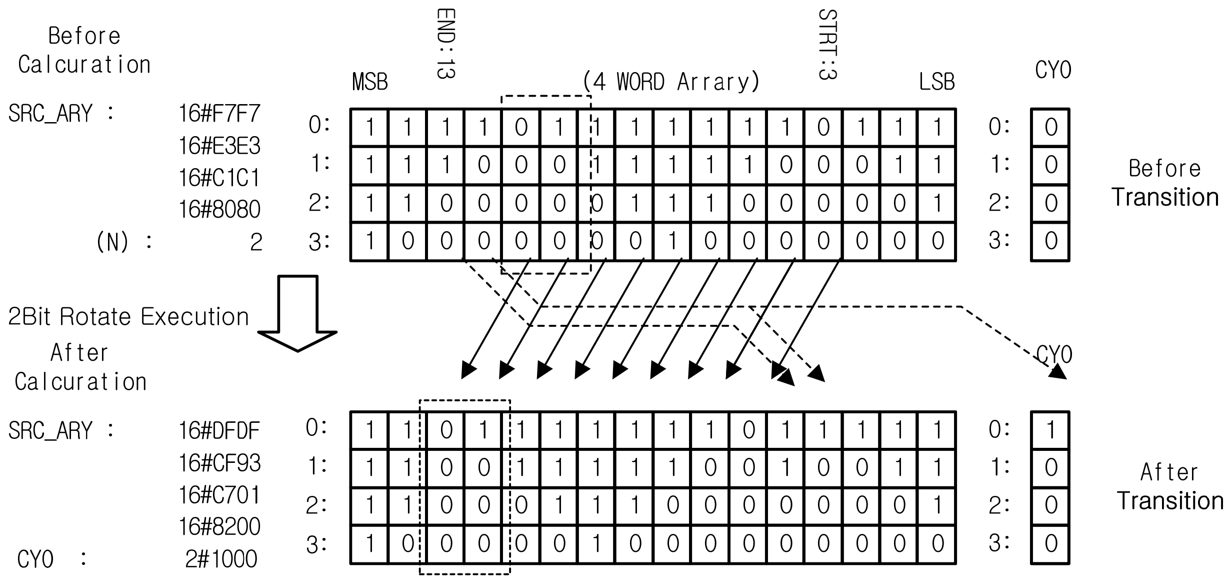
2. ST

ARY_ROT_C(EN:=%MX2, SRC:=IN_ARY, STRT:=3, END:=13, N:=2, CYO=>CYO);

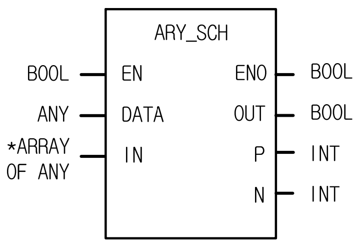
(1) If the input condition (%MX2) is on, ARY_ROT_C function executes.

(2) It rotates 2 times the bit (from 3 to 13 bit) arrays of IN_ARY from STRT to END.

(3) The result is stored at IN_ARY and the carry bit arrays are written in CYO Array.



ARY_SCH	**Array search**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 DATA: data to search IN: array to search **Output** ENO: outputs EN value as it is OUT: if it finds, it is 1 P: first position of an object array N: total number of array elements equal to an object

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	DATA	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ARRAY OF ANY: exclude STRING from ANY type.

■ **Function**

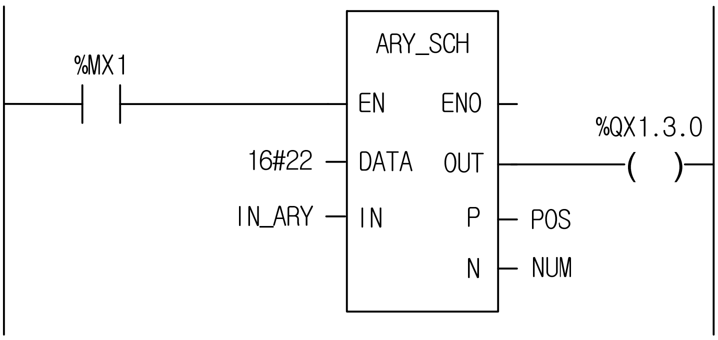
It finds an equal value of input in arrays and produces its first position and total number. When it finds at least one which is equal to an object in arrays, OUT is 1.

Function	Input Array type	Description
ARY_SCH	BOOL	Search in BOOL Array.
ARY_SCH	BYTE	Search in BYTE Array.
ARY_SCH	WORD	Search in WORD Array.
ARY_SCH	DWORD	Search in DWORD Array.
ARY_SCH	LWORD	Search in LWORD Array.
ARY_SCH	SINT	Search in SINT Array.
ARY_SCH	INT	Search in INT Array.
ARY_SCH	DINT	Search in DINT Array.
ARY_SCH	LINT	Search in LINT Array.
ARY_SCH	USINT	Search in USINT Array.
ARY_SCH	UINT	Search in UINT Array.

Function	Input Array type	Description
ARY_SCH	UDINT	Search in UDINT Array.
ARY_SCH	ULINT	Search in ULINT Array.
ARY_SCH	REAL	Search in REAL Array.
ARY_SCH	LREAL	Search in LREAL Array.
ARY_SCH	TIME	Search in TIME Array.
ARY_SCH	DATE	Search in DATE Array.
ARY_SCH	TOD	Search in TOD Array.
ARY_SCH	DT	Search in DT Array.

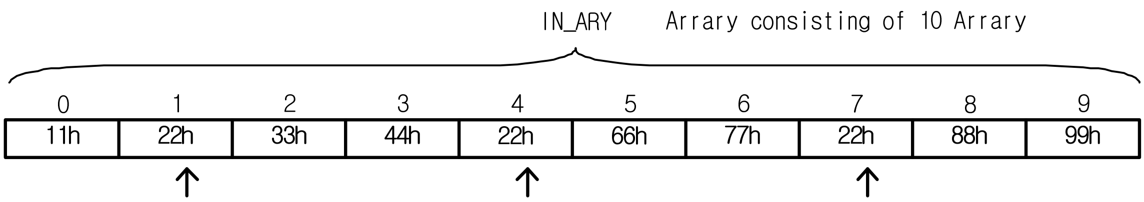
■ **Program Example**

1. LD



2. ST

%QX1.3.0 := ARY_SCH(EN:=%MX1, DATA:=16#22, IN:=IN_ARY, P=>POS, N=>NUM);

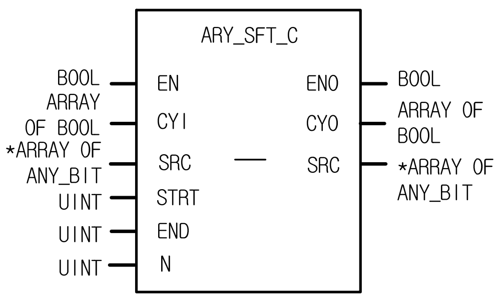


(1) If the input condition (%MX1) is on, ARY_SCH function executes.

(2) When IN_ARY is a 10-byte array, if you search for “22h” in this array, three bytes are found as the above.

(3) The result is: 1) 1, the first position of an array, is stored at POS; 2) 3, the total number, is stored at NUM. The total number is 3, so the output %Q1.3.0 is on.

<table>
<tr><td rowspan="3">ARY_SFT_C</td><td colspan="2">Array of Bit Shift Left with Carry</td></tr>
<tr><td>Availability</td><td>XGI, XGR, XEC, XMC</td></tr>
<tr><td>Flags</td><td>_ERR, _LER</td></tr>
</table>

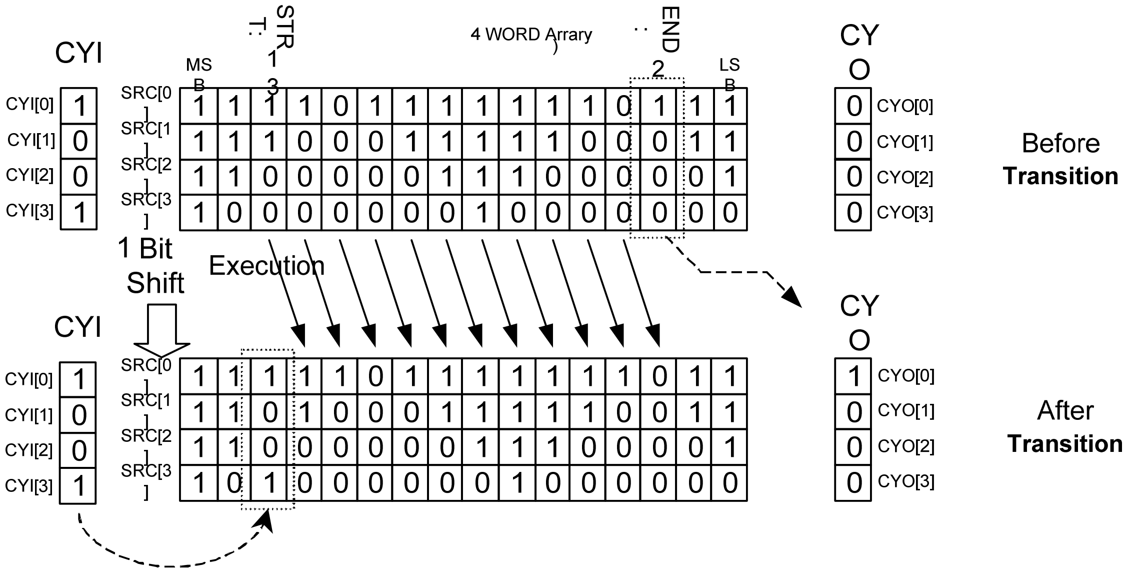
Function	Description
	**Input** EN: executes the function in case of 1 CYI: Input Carry bit Array STRT: starting bit to shift END: ending bit to shift N: bit number to shift **Output** ENO: without an error, it is 1 CYO: Output Carry bit Array after shift **In/Out** SRC: Source Array to shift

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	SRC		○	○	○	○															

*ARRAY OF ANY_BIT: exclude BOOL from ANY_BIT type.

■ **Function**

1. It shifts as many bits of array elements as specified.
2. Setting
   - Scope: it sets a shifting scope with STRT and END.
   - Shifting direction and time: it shifts N times from STRT to END.
   - Input data: it fills the empty bits with input data (CYI).
   - Output: the result is stored in ANY_BIT_ARY and an overflowing bit array data from END is written at CYO.



Function	In/Out array type	Description
ARY_SFT_C	BYTE	It shifts as many bits of array elements as specified.
ARY_SFT_C	WORD	
ARY_SFT_C	DWORD	
ARY_SFT_C	LWORD	

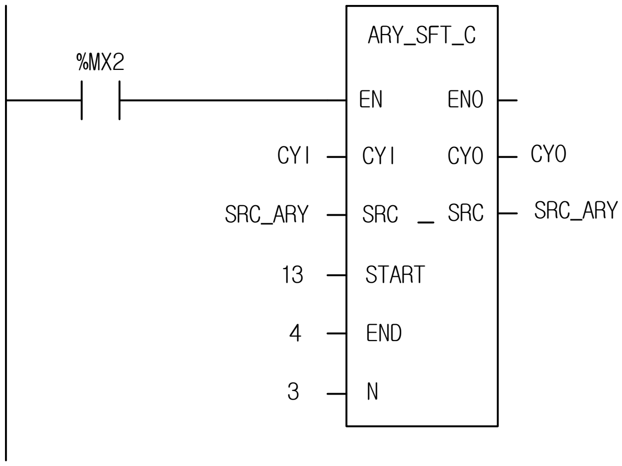
■ **Flag**

Flag	Description
_ERR	If the number of CYI, SRC and CYO Array are different, _ERR and _LER flags are set. An error occurs if STRT and END are out of SRC range. When an error occurs, there's no change in SRC and CYO.

☆ If the number of each input/output array is different, _ERR and _LER flags occur; if output array variable is omitted, the number of array is regarded as '0' and _ERR and _LER flags occur.

■ **Program Example**

1. LD



2. ST

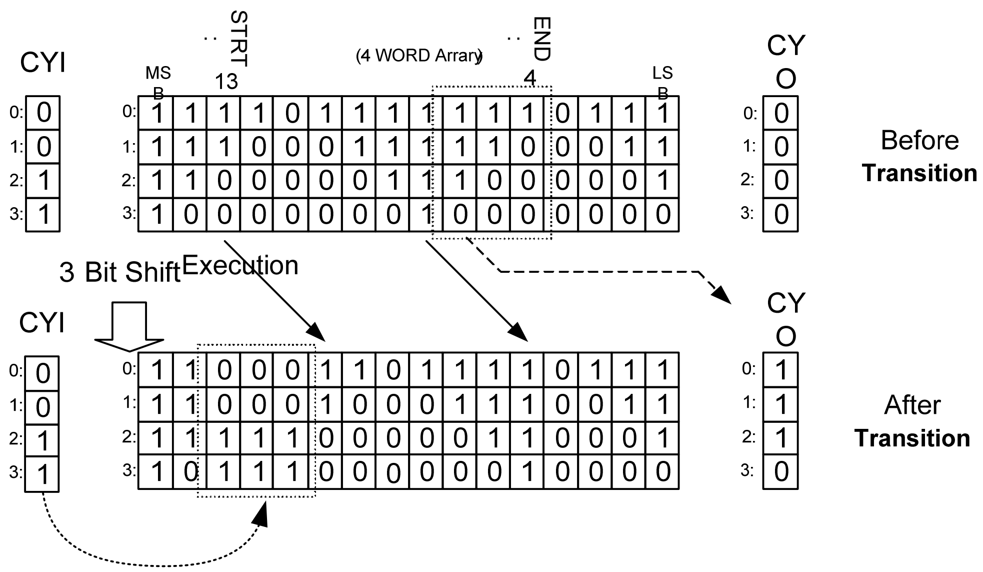
ARY_SFT_C(EN:=%MX2, CYI:=CYO, SRC:=SRC_ARY, STRT:=13, END:=4, N:=2, CYO=>CYO);

(1) If input condition (%MX2) is on, ARY_SFT_C function executes.

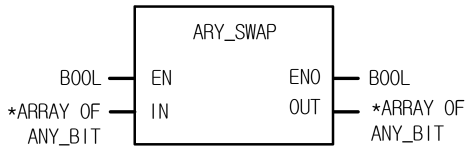
(2) It shifts a bit array (from 4 to 13 bit) of SRC 3 times from STRT to END.

(3) The bit array after shifting is filled with CYI (2#0011).

(4) It produces its shifting result at SRC_ARY and a carry bit array is written at CYO.



ARY_SWAP	Upper/Lower elements swapping of an array	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: array input **Output** ENO: without an error, it is 1 OUT: array output after swapping

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ARRAY OF ANY_BIT: exclude BOOL from ANY_BIT type.

■ **Function**

It swaps upper/lower elements after dividing an array.

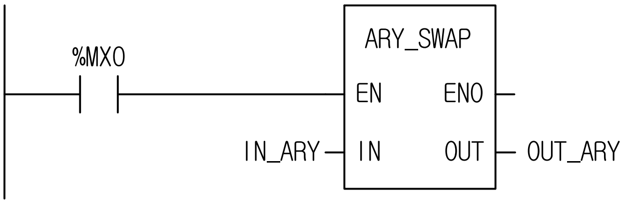
■ **Flag**

Flag	Description
_ERR	_ERR and _LER flags are set if two arrays are different; there's no change in an OUT array.

☆ If the number of each input/output array is different, _ERR and _LER flags occur; if output array variable is omitted, the number of array is regarded as '0' and _ERR and _LER flags occur.

■ **Program Example**

1. LD



2. ST

OUT_ARY := ARY_SWAP(EN:=%MX0, IN:=IN_ARY);

(1) If the transition condition (%MX0) is on, ARY_SWAP function with WORD type executes.

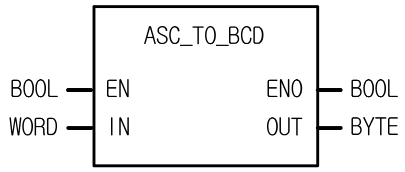
(2) If IN_ARY data is as below:

IN_ARY[0]	12AB
IN_ARY[1]	23BC
IN_ARY[2]	34CD

OUT_ARY data is as follows:

OUT_ARY[0]	AB12
OUT_ARY[1]	BC23
OUT_ARY[2]	CD34

ASC_TO_BCD	Converts ASCII to BCD	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1. IN: ASCII input **Output** ENO: without an error, it is 1 OUT: BCD output

■ **Function**

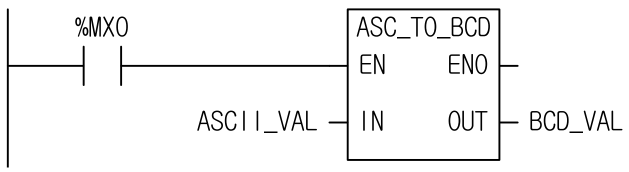
It converts two ASCII data into two-digit BCD (Binary Coded Decimal) data.

■ **Flag**

Flag	Description
_ERR	If IN is not a hexadecimal number between 0 ~ 9, the output is 0 and _ERR and _LER flags are set.

■ **Program Example**

1. LD



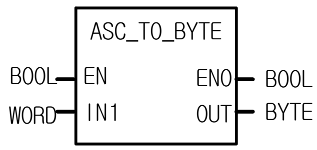
2. ST

```
BCD_VAL := ASC_TO_BCD(EN:=%MX0, IN:=ASCII_VAL);
```

(1) If the transition condition (%MX0) is on, ASC_TO_BCD function executes.

(2) If input variable ASCII_VAL (WORD) = 16#3732 = “72”, output variable BCD_VAL (BYTE) = 16#72.

# ASC_TO_BYTE	**Converts ASCII to BYTE data**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1. IN: ASCII input **Output** ENO: without an error, it is 1 OUT: BYTE Output

■ **Function**

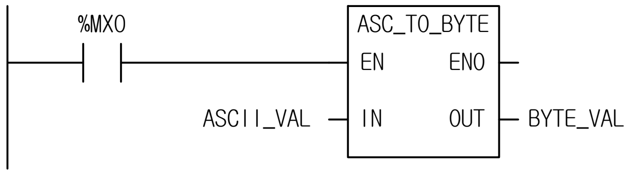
It converts two ASCII data to 2-digit hexadecimal (HEX).

■ **Flag**

Flag	Description
_ERR	If IN is not between '0' and 'F', its output is 0 and _ERR and _LER flags are set.

■ **Program Example**

1. LD



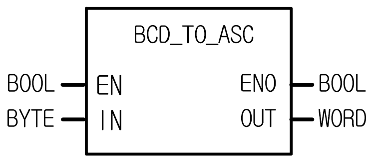
2. ST

```
BYTE_VAL := ASC_TO_BYTE(EN:=%MX0, IN:=ASCII_VAL);
```

(1) If the transition condition (%MX0) is on, ASC_TO_BYTE function executes.

(2) If input ASCII_VAL (WORD) = 16#4339, output BYTE_VAL (BYTE) = 16#C9.

BCD_TO_ASC	Converts BCD to ASCII data	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1. IN: BCD input **Output** ENO: without an error, it is 1 OUT: ASCII Output

■ **Function**

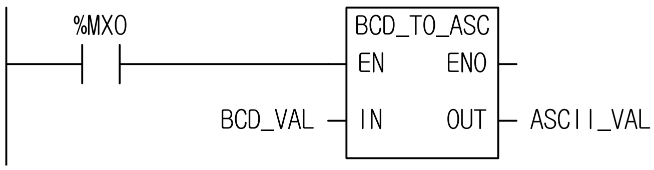
It converts 2-digit BCD data to two ASCII data.

■ **Flag**

Flag	Description
_ERR	If IN is not a hexadecimal number between 0 and 9, its output is 16#3030 (“00”) and _ERR/_LER flags are set.

■ **Program Example**

1. LD

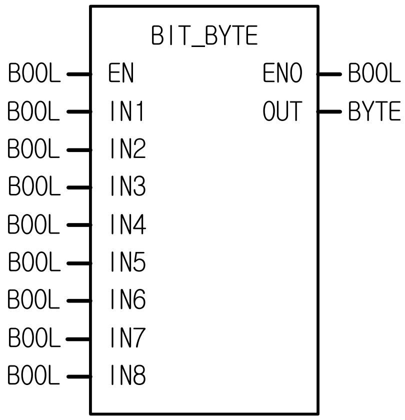


2. ST

```
ASCII_VAL := BCD_TO_ASC(EN:=%MX0, IN:=BCD_VAL);
```

(1) If the transition condition (%MX0) is on, BCD_TO_ASC function executes.
(2) If input BCD_VAL (BYTE) = 16#85, output ASCII_VAL (WORD) = 16#3835 = “85.”

BIT_BYTE	Combines 8 bits into BYTE	
	Availability	XGI, XGR, XEC, XMC
	Flags	

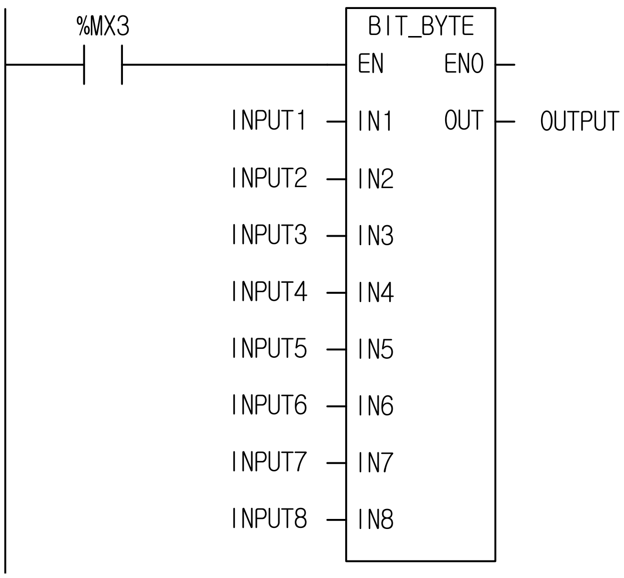
Function	Description
	**Input** EN: executes the function in case of 1. IN1 ~ IN8: Bit input **Output** ENO: without an error, it is 1 OUT: Byte output

■ **Function**

It combines 8 bits into one byte.
IN8: MSB (Most Significant Bit), IN1: LSB (Least Significant Bit).

■ **Program Example**

1. LD

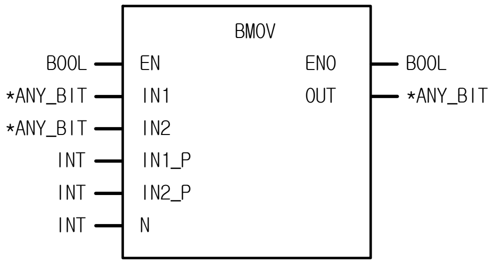


2. ST

OUTPUT := BIT_BYTE(EN:=%MX3, IN1:=INPUT1, IN2:=INPUT2, IN3:=INPUT3, IN4:=INPUT4, IN5:=INPUT5, IN6:=INPUT6, IN7:=INPUT7, IN8:=INPUT8);

(1) If the transition condition (%MX3) is on, BIT_BYTE function executes.
(2) If 8 input are (from INPUT1 to INPUT 8) {0,1,1,0,1,1,0,0}, OUTPUT (BYTE) = 2#0110_1100.

BMOV	Moves part of a bit string	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1. IN1: String data having bit data to be combined IN2: String data having bit data to be combined IN1_P: Start bit position on IN1 set data IN2_P: Start bit position on IN2 set data N: Bit number to be combined **Output** ENO: without an error, it is 1 OUT: Combined bit string data output

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1		○	○	○	○															
	IN2		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT type.

■ **Function**

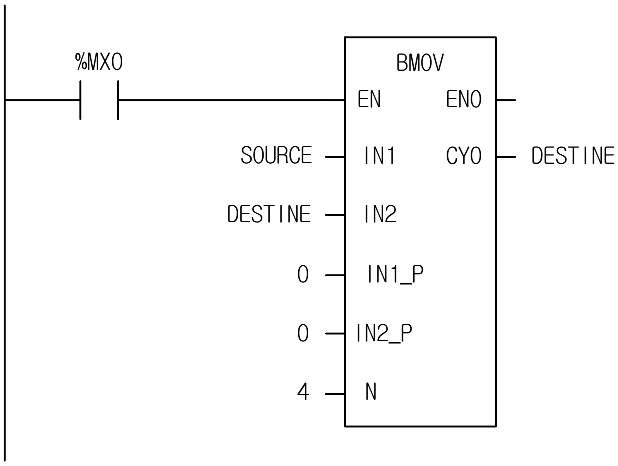
1. If EN is 1, it takes N bits of IN1 starting from the IN1_P bit and moves it to IN2 starting from IN2_P bit.
2. If N1 = 1111_0000_1111_0000, IN2 = 0000_1010_1010_1111, IN1_P = 4, IN2_P = 8, N = 4, then output data is 0000_1111_1010_1111. Input data types are B (BYTE), W (WORD), D (DWORD), L (LWORD).

■ **Flag**

Flag	Description
_ERR	If IN1_P and IN2_P exceed the data range or N is negative or N bit of IN1_P and IN2_P exceeds the data range, _ERR and _LER flags are set.

■ **Program Example**

1. LD



2. ST

DESTINE := BMOV(EN:=%MX0, IN1:=SOURCE, IN2:=DESTINE, IN1_P:=0, IN2_P:=0, N:=4);

(1) If the transition condition (%MX0) is on, BMOV function executes.

(2) Since SOURCE = 2#0101_1111_0000_1010, DESTINE = 2#0000_0000_0000_0000 as declared as input variable and IN1_ P = 0, IN2_P = 8, N = 4, the operations yields 2#0000_1010_0000_0000, and it is changed to DESTINE = 2#0000_1010_0000_0000 because output is designated as DESTINE.

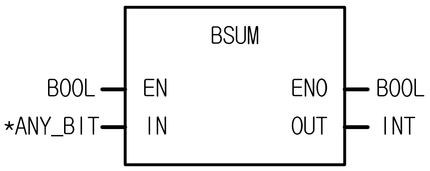
```
INPUT (IN1) : SOURCE (WORD) = 16#5F0A
      (IN2) : DESTINE(WORD) = 16#0000
      (IN1_ P) = 0
      (IN2_P) = 8
      (N) =    4

OUTPUT (OUT) : DESTINE(WORD) = 16#0A00
```

0	1	0	1	1	1	1	1	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

↓ (BMOV)

0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0

BSUM	**Counts on-bit number of input**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1. IN: input data to detect on bit **Output** ENO: outputs EN value as it is OUT: Result data (sum of on-bit number)

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT type.

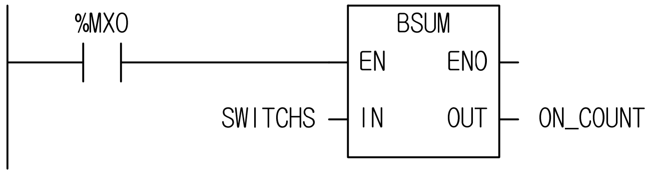
■ **Function**

1. If EN is 1, it counts bit number of 1 among IN bit string and produces output, OUT.
2. Input data types are BYTE, WORD, DWORD and LWORD.

Function	IN type	Description
BSUM	BYTE	You can select one of these functions according to input data.
BSUM	WORD	
BSUM	DWORD	
BSUM	LWORD	

■ **Program Example**

1. LD



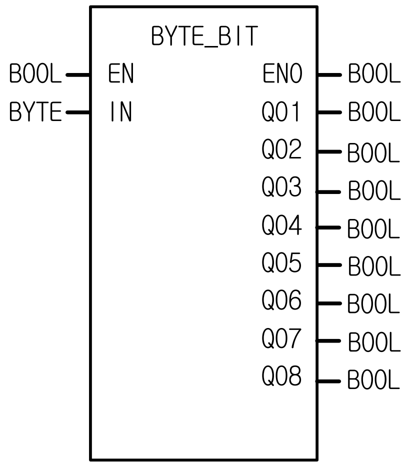
2. ST

ON_COUNT := BSUM(EN:=%MX0, IN:=SWITCHS);

(1) If the transition condition (%MX0) is on, BSUM function executes.

(2) If input SWITCHS (WORD) = 2#0000_0100_0010_1000, then it counts on-bit number, 3. So the output ON_COUNT (INT) = 3.

BYTE_BIT	Divides byte into 8 bits	
	Availability	XGI, XGR, XEC, XMC
	Flags	

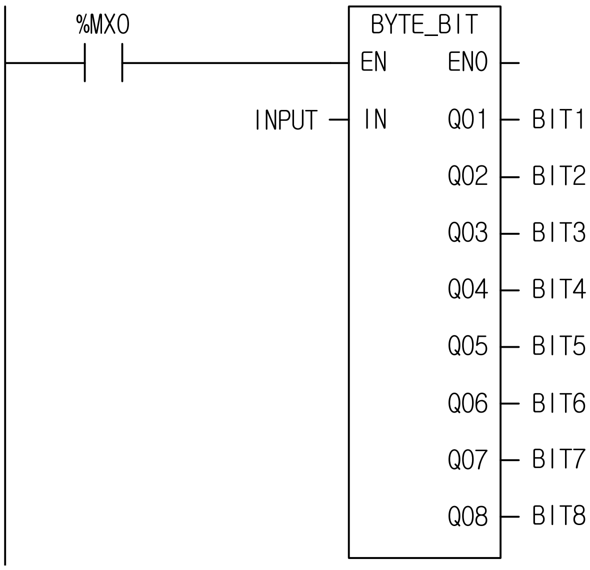
Function	Description
	**Input** EN: executes the function in case of 1. IN: BYTE input **Output** ENO: outputs EN value as it is QO1~8: bit output

■ **Function**

1. It divides one byte into 8 bits (QO1~QO2).
2. QO8: MSB (Most Significant Bit), QO1: LSB (Least Significant Bit)

■ **Program Example**

1. LD

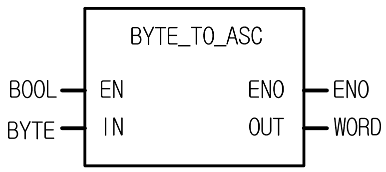


2. ST

BYTE_BIT(EN:=%MX0, IN:= INPUT, Q01=> BIT1, Q02=> BIT2, Q03=> BIT3, Q04=> BIT4, Q05=> BIT5, Q06=> BIT6, Q07=> BIT7, Q08=> BIT8);

(1) If the execution condition (%MX0) is on, BYTE_BIT function executes.
(2) If INPUT = 16#AC = 2#1010_1100, it distributes INPUT from Q01 to Q08 in order. The order is 2#{0, 0, 1, 1, 0, 1, 0, 1}.

BYTE_TO_ASC	Converts BYTE to ASCII data	
	Availability	XGI, XGR, XEC, XMC
	Flags	

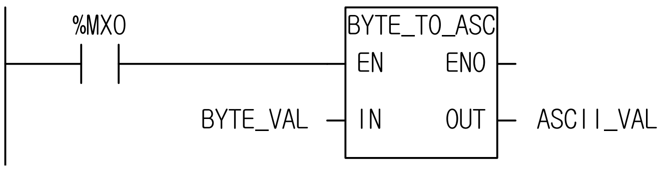
Function	Description
	**Input** EN: executes the function in case of 1. IN: BYTE input **Output** ENO: outputs EN value as it is OUT: ASCII output

■ **Function**

1. It converts 2-digit hexadecimal into two ASCII data.
   Ex) 16#12 -> 3132
2. In case of 16#A~F, it produces ASCII data for character.

■ **Program Example**

1. LD

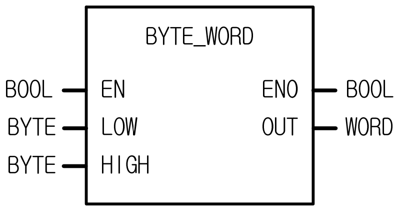


2. ST

```
ASCII_VAL := BYTE_TO_ASC(EN:=%MX0, IN:=BYTE_VAL);
```

(1) If the transition condition (%MX0) is on, BYTE_TO_ASC function executes.
(2) If input BYTE_VAL (BYTE) = 16#3A, output ASCII_VAL (WORD) = 16#3341 = ‘3’, ‘A’.

BYTE_WORD	Combines 2 bytes into WORD	
	Availability	XGI, XGR, XEC, XMC
	Flags	

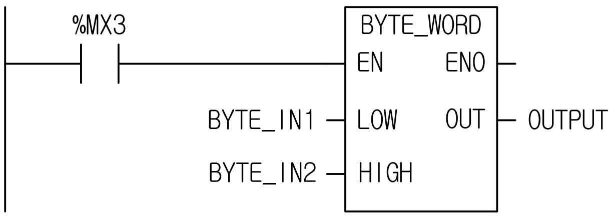
Function	Description
	**Input** EN: executes the function in case of 1. LOW: lower BYTE input HIGH: upper BYTE input **Output** ENO: outputs EN value as it is OUT: WORD output

■ **Function**

It combines two bytes into one word.
LOW: lower BYTE input, HIGH: upper BYTE input

■ **Program Example**

1. LD

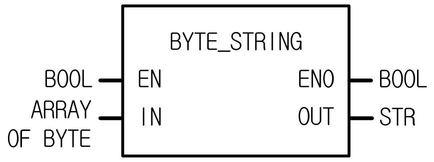


2. ST

```
OUTPUT := BYTE_WORD(EN:=%MX3, LOW:=BYTE_IN1, HIGH:=BYTE_IN2);
```

(1) If the transition condition (%MX3) is on, BYTE_WORD function executes.
(2) If input BYTE_IN1 = 16#56 and BYTE_IN2 = 16#AD, output variable OUTPUT = 16#AD56.

BYTE_STRING	Converting Byte Array to String	
	Availability	XGI, XGR, XEC, XMC
	Flags	

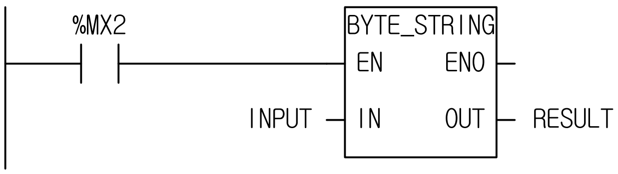
Function	Description
	**Input** EN : executes the function in case of 1 IN : input Byte Array **Output** ENO : outputs EN value as it is OUT : outputs converted string

■ **Function**

Converts Byte Array to a string.

■ **Program Example**

1. LD

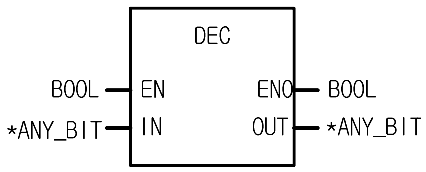


2. ST

```
RESULT := BYTE_STRING(EN:=%MX2, IN:=INPUT);
```

(1) If the execution condition(%MX2) is on, BYTE_STRING function executes.
(2) If setting INPUT array variable as 3 and if entering INPUT[0] = 16#41, INPUT[1] = 16#31, INPUT[2] = 16#35, Output RESULT = ‘A15’.

DEC	Decrease IN data by 1 bit	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1. IN: input data to decrease **Output** ENO: outputs EN value as it is OUT: result data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT type.

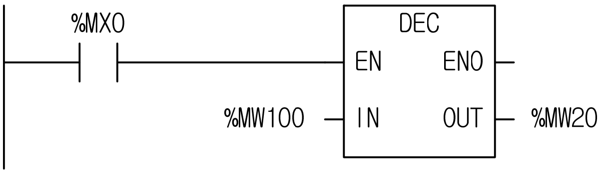
■ **Function**

1. If EN is 1, it produces an output after decreasing bit-string data of IN by 1.
2. Even though the underflow occurs, an error won't occur and if the result is 16#0000, then the output result data is 16#FFFF.
3. Input data types are BYTE, WORD, DWORD and LWORD.

FUNCTION	IN/OUT type	Description
DEC	BYTE	You can select one of these functions according to in/out data type.
DEC	WORD	
DEC	DWORD	
DEC	LWORD	

■ **Program Example**

1. LD



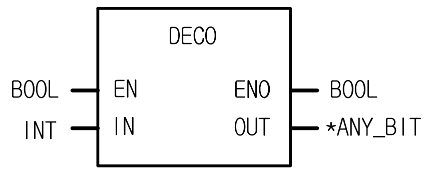
2. ST

%MW20 := DEC(EN:=%MX0, IN:=%MW100);

(1) If the transition condition (%MX0) is on, DEC function executes.

(2) If input variable %MW100 = 16#0007 (2#0000_0000_0000_0111), output variable %MW20 = 16#0006 (2#0000_0000_0000_0110).

DECO	Decodes the designated bit position	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1. IN: input data for Decoding **Output** ENO: without an error, it is 1 OUT: Decoding result data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT type.

■ **Function**

1. If EN is 1, it turns on 'the designated position bit of output bit-string data' according to the value of IN, and produces an output.
2. Output data types are BYTE, WORD, DWORD and LWORD.

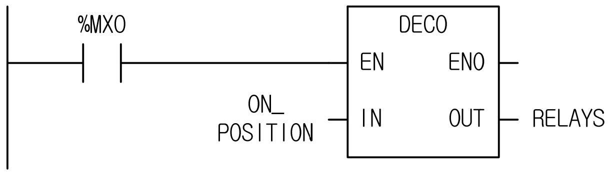
FUNCTION	OUT type	Description
DECO	BYTE	You can select one of these functions according to output data type.
DECO	WORD	
DECO	DWORD	
DECO	LWORD	

■ **Flag**

Flag	Description
_ERR	If input data is a negative number or bit position data is out of output-type range, (in case of DECO_WORD, it's more than 16), then OUT is 0 and _ERR/_LER flags are set.

■ **Program Example**

1. LD



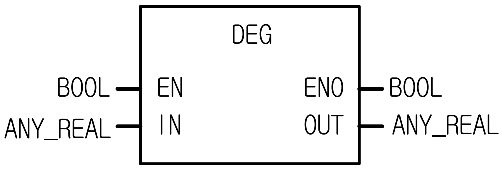
2. ST

RELAYS := DECO_DWORD(EN:=%MX0, IN:=ON_POSITION);

(1) If the transition condition (%MX0) is on, DECO function executes.

(2) Since the only 5th bit of output is on if ON_POSITON(INT) = 5 as declared as input variable, RELAYS(WORD type) = 2#0000_0000_0010_0000.

DEG	Converts radian into degree	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1. IN: radian input **Output** ENO: outputs EN value as it is OUT: degree output

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

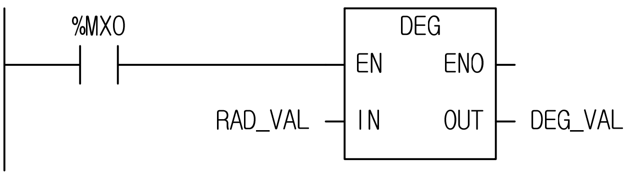
■ **Function**

It converts radian input into degree output.

Function	Input type	Output type	Description
DEG	REAL	REAL	It converts input (radian) into output (degree).
DEG	LREAL	LREAL	

■ **Program Example**

1. LD

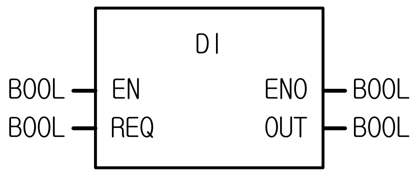


2. ST

```
DEG_VAL := DEG(EN:=%MX0, IN:=RAD_VAL);
```

(1) If the transition condition (%M0) is on, DEG function executes.
(2) If input variable RAD_VAL = 1.0, then output variable DEG_VAL = 5.7295779513078550e+001.

<table>
<tr><td rowspan="3">DI</td><td colspan="2">Disable start of task program</td></tr>
<tr><td>Availability</td><td>XGI, XGR, XEC, XMC</td></tr>
<tr><td>Flags</td><td></td></tr>
</table>

<table>
<tr><th>Function</th><th>Description</th></tr>
<tr><td>
</td><td>Input EN: executes the function in case of 1
REQ: requires to invalidate when task program starts

Output ENO: outputs EN value as it is
OUT: if DI executes, it is 1</td></tr>
</table>

■ **Function**

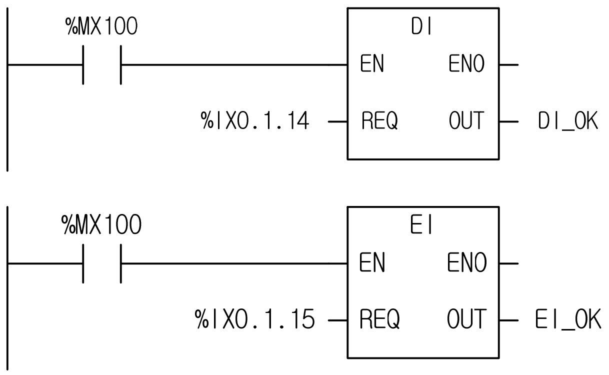
1. If EN = 1 and REQ = 1, it stops a task program (single, interval, interrupt).
2. Once DI function executes, a task program does not start even if REQ input is 0.
3. In order to start a task program normally, use 'EI' function.
   If you want to partially stop the task program for the troubled part, (otherwise, the continuity of operation process due to the execution of other task program), you can to use this function.
4. The task programs created while its execution is not invalidated is executed according to task program types as follows:.
   - Single task: It executes after 'EI' function or current-running task program executes. In this case, it repeats a task program as many as the state of single variable changes.
   - Interval task, interrupt: the task occurred when it is not permitted to execute and executes after 'EI' function or the current-running task program executes. But, if it occurs more than 2 times, TASK_ERR is on and TC_CNT (the number of task collision) is counted.

■ **Program Example**

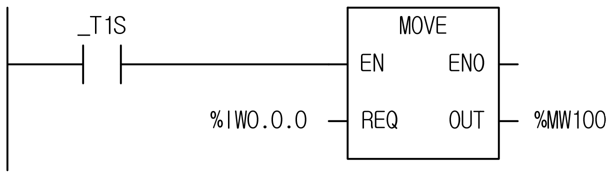
This is the program that controls the task program, increasing the value per second by using DI (Invalidates task program) and EI (permits running for task program).

1. LD

Scan program (TASK program control)



Task program increasing every second



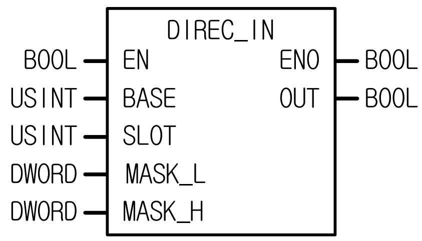
2. ST

Scan program (TASK program control)
%IX0.1.14 := DI(EN:=%MX100, REQ:=DI_OK);
%IX0.1.15 := EI(EN:=%MX100, REQ:=EI_OK);

Task program increasing every second
%MW100 := MOVE(EN:=_T1S, IN:=%IW0.0.0);

(1) If REQ (assigned as direct variable %IX0.1.14) of DI is on, DI function executes and output DI_OK is 1.
(2) If DI function executes, the task program to be executed per second stops.
(3) If REQ (assigned as direct variable %IX0.1.15) of EI is on, EI function executes and output EI_OK is 1.
(4) If EI function executes, the task program stops and the function DI restarts.

DIREC_IN	Update input data immediately	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 BASE: base number of an input module installed SLOT: slot number of an input module installed MASK_L: designates bits not to be updated among lower 32-bit data of input MASK_H: designates bits not to be updated among upper 32-bit data of input **Output** ENO: without an error, it is 1. OUT: if update is completed, output is 1.

■ **Function**

1. If EN is 1 during the scan, DIREC_IN function reads 64-bit data of an input module from the designated position of a BASE and a SLOT, and updates them.
2. Only the actual contacts of an input module updates in the image scope.
3. DIREC_IN function is available to use when you want to change the On/Off state of input (%I) during the scan.
4. Generally, it's impossible to update input data during 1 scan (executing a scan program) because a scan-synchronized batch processing mode executes the batch processing to read input data and produce output data after a scan program.
5. If you use DIREC_IN function during program execution, related input data updates.

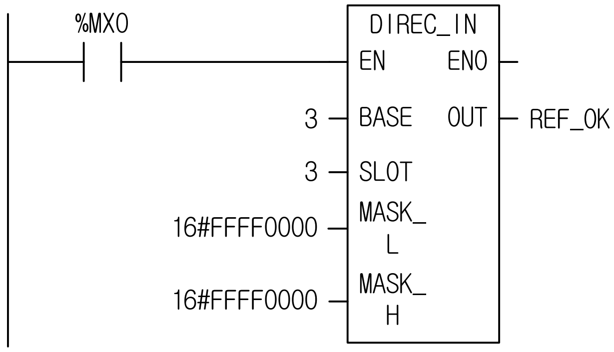
■ **Flag**

Flag	Description
_ERR	If BASE, SLOT input range is exceeded, or if an error is occurred while input/output data refresh, the output is 0 and _ERR and _LER flags are set.

■ **Program Example**

1. This program updates a 16-contact module installed in the slot no.3 of the 3rd extension base for which input data are 2# 1010_1010_1110_1011.

1. LD



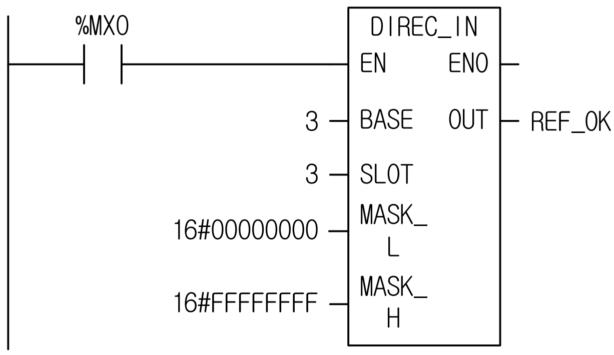
2. ST

REF_OK := DIREC_IN(EN:=%MX0, BASE:=3, SLOT:=3, MASK_L:=16#FFFF0000, MASK_H:=16#FFFF0000);

(1) If the input condition (%MX0) is on, DIREC_IN function executes.
(2) The image scope to update is %IW3.3.0 because a 16-contact module installs. %IW3.3.0 is updated with 2#1010_1010_1110_1011 during the scan because a lower 16-bit data of MASK_L (lower 32-bit input) which is not going to be changed is updatable.
(3) It does not matter what data are set in MASK_H (upper 32-bit input) because a 16-contact module is installed on the slot and base.

2. This program updates the lower 32-bit data of the 32-contact module installed in the slot no.3 of the 3rd extension base for which input data are 2#0000_0000_1111_1111_1100_1100_0011_0011.

1. LD



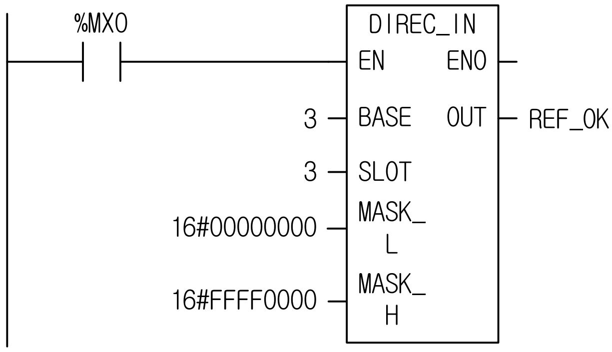
2. ST

REF_OK := DIREC_IN(EN:=%MX0, BASE:=3, SLOT:=3, MASK_L:=16#00000000, MASK_H:=16#FFFFFFFF);

(1) If input condition (%MX0) is on, function DIREC_IN executes.
(2) The image scope to update is %ID3.3.0 because a 32-contact module installs. %ID3.3.0 is updated with 2#0000_0000_1111_1111_1100_1100_0011_0011 during the scan because a lower 32-bit data of MASK_L (lower 32-bit input) which is not going to be changed is updatable.

3. This program updates the lower 48-bit data of the 64-contact module installed in the slot no.3 of the 3rd extension base for which input data are 16#0000_FFFF_AAAA_7777
(2#0000_0000_0000_0000_1111_1111_1111_1111_1010 _1010_1010_1010_0111_0111_0111_0111).

1. LD



2. ST

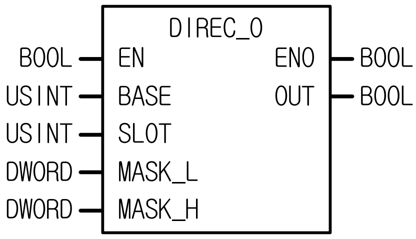
REF_OK := DIREC_IN(EN:=%MX0, BASE:=3, SLOT:=3, MASK_L:=16#00000000, MASK_H:=16#0000FFFF);

(1) If the input condition (%MX0) is on, function DIREC_IN function executes.
(2) The installed module is a 64-contact module and the image scope to update is %IL3.3.0 (%ID3.3.0 and ID3.3.1).
(3) %ID3.3.0 updated because the lower 32-bit data (MASK_L) update is allowed.
(4) %IW3.3.2 of %ID3.3.1 is updated because only the lower 16-bit data update among upper 32 bits (MASK_H) is allowed.
(5) Accordingly, the data update of the image scope is as follows..

%IL3.3.0
- %ID3.3.0
  - %IW3.3.0: 2#0111_0111_0111_0111
  - %IW3.3.1: 2#1010_1010_1010_1010
- %ID3.3.1
  - %IW3.3.2: 2#1111_1111_1111_1111
  - %IW3.3.3: maintains the previous value

(6) If the input update is completed, output REF_OK is 1.

DIREC_O	Update output module data immediately	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 BASE: base number of an input module installed SLOT: slot number of an input module installed MASK_L: designates bits not to be updated among lower 32-bit data of output MASK_H: designates a bit not to update among upper 32-bit data of output **Output** ENO: without an error, it is 1. OUT: if update is completed, output is 1.

■ **Function**

1. If EN is 1 during the scan, DIREC_O function reads 64-bit data of an output module from the configured position of BASE and SLOT and updates the unmasked (MASK (1)) data.
2. DIREC_O is available to use when you want to change the on/off state of output (%Q) during the scan.
3. Generally, it is impossible to update input data during 1 scan (executing a scan program) because a scan-synchronized batch processing mode executes the batch processing to read input data and produce output data after a scan program.
4. It is available to update related output data, if you use DIREC_O function during program execution.
5. If the base/slot number is wrong or it is not available to write data normally in an output module, ENO and OUT are '0' (without an error, it is 1).

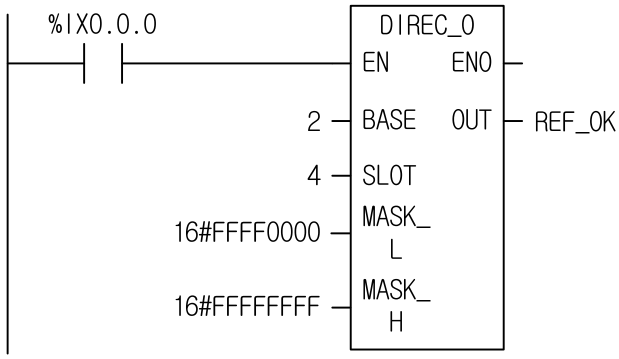
■ **Flag**

Flag	Description
_ERR	If BASE, SLOT input range is exceeded, or if an error is occurred while input/output data refresh, the output is 0 and _ERR and _LER flags are set.

■ **Program Example**

1. This is the program that produces output data 2#0111_0111_0111_0111 in a 32-contact relay output module installed in the slot no.4 of the 2nd extension base.

1. LD



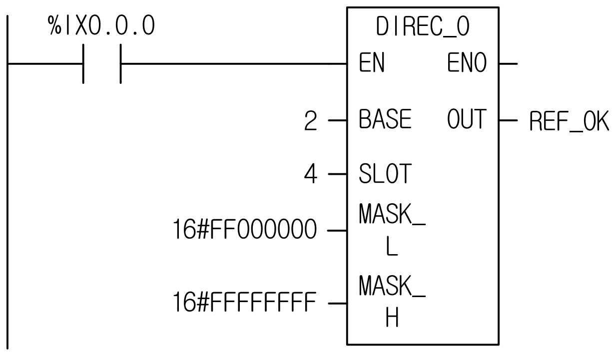
2. ST

REF_OK := DIREC_O(EN:=%IX0.0.0, BASE:=2, SLOT:=4, MASK_L:=16#FFFF0000, MASK_H:=16#FFFFFFFF);

(1) Input the base number 2 and slot number 4 in which an output module is installed.
(2) Set MASK_L as 16#FFFF0000 because the output data to produce are the lower 16 bits among the output contacts.
(3) If the transition condition (%IX0.0.0) is on, DIREC_O executes and the data of the output module is updated as 2#0111_0111_0111_0111 during the scan.

2. This is the program that updates the lower 24 bits of the 32-contact transistor output module, installed in the slot no.4 of the 2nd extension base, with 2#1111_0000_1111_0000_1111_0000 during the scan.

1. LD



2. ST

REF_OK := DIREC_O(EN:=%IX0.0.0, BASE:=2, SLOT:=4, MASK_L:=16#00000000, MASK_H:=16#FFFFFFFF);

(1) Input the base number 2 and slot number 4 in which an output module is installed.

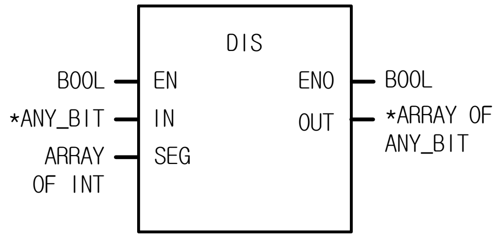
(2) Set MASK_L as 16#FF000000 because the output data to produce are the lower 24 bits among the output contacts.

(3) If the transition condition (%IX0.0.0) is off, function DIREC_O executes and the data of the output module is updated.

2#□□□□_□□□□_1111_0000_1111_0000_1111_0000 during the scan.

Maintains the previous value.

DIS	Data distribution	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1. IN: input data SEG: configured bit array for data distribution **Output** ENO: without an error, it is 1 OUT: distributed array output

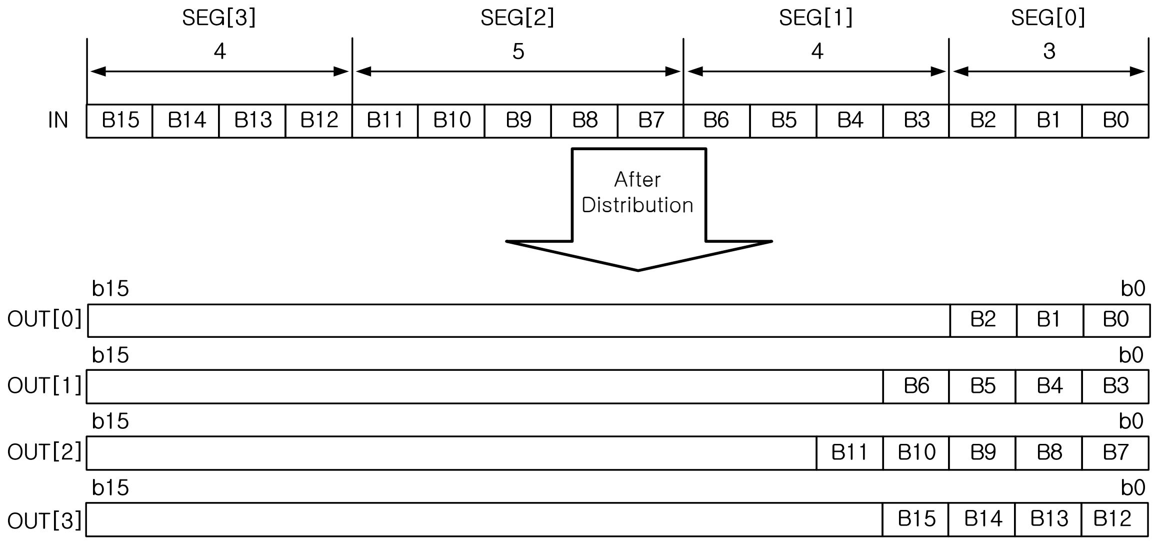
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT type.

■ **Function**

It distributes input data over OUT after segmenting input data by bit number set by SEG.

Function	Input	Description
DIS	BYTE	It distributes IN input by bit number set with SEG array and outputs, OUT array which is the same type as IN.
DIS	WORD	
DIS	DWORD	
DIS	LWORD	



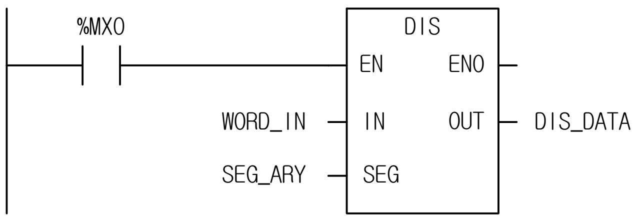
■ **Flag**

Flag	Description
_ERR _LER	If the sum of configured number of SEG exceeds input variable bit number, _ERR and _LEF flags are set.

☆ If output array is omitted, it assumes the number of array as 0, producing _ERR and _LER flags.

■ **Program Example**

1. LD



2. ST

DIS_DATA := DIS(EN:=%MX0, IN:= WORD_IN, SEG:=SEG_ARY);

(1) If the transition condition (%MX0) is o, DIS function executes.

(2) If input variable WORD_IN = 16#3456, SEG_ARY = {3, 4, 5, 4}, then, output variable DIS_DATA is:

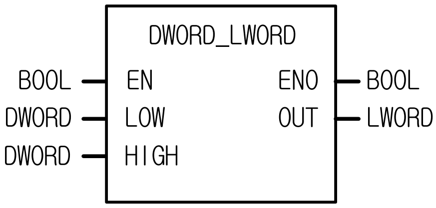
DIS_DATA[0]=16#0006

DIS_DATA[1]=16#000A

DIS_DATA[2]=16#0008

DIS_DATA[3]=16#0003

DWORD_LWORD	Combines two DWORD data into LWORD	
	Availability	XGI, XGR, XEC, XMC
	Flags	

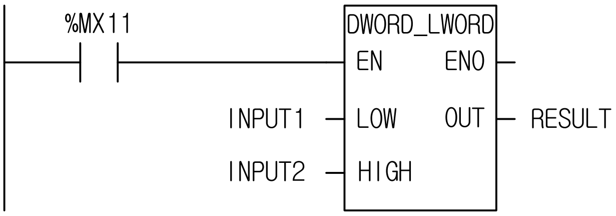
Function	Description
	**Input** EN: executes the function in case of 1. LOW: lower DWORD Input HIGH: upper DWORD Input **Output** ENO: outputs EN value as it is OUT: LWORD Output

■ **Function**

It combines 2 DWORD data into one LWORD data.
LOW: lower DWORD Input, HIGH: upper DWORD Input

■ **Program Example**

1. LD

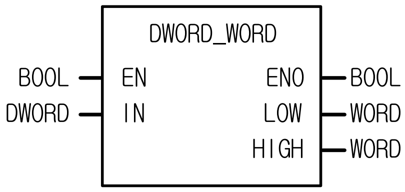


2. ST

RESULT := DWORD_LWORD(EN:=%MX11, LOW:=INPUT1, HIGH:=INPUT2);

(1) If the transition condition (%MX11) is on, DWORD_LWORD function executes.
(2) If input variable INPUT1 = 16#1A2A_3A4A and INPUT2 = 16#8C7C_6C5C, then, output variable RESULT = 16#8C7C_6C5C_1A2A_3A4A.

DWORD_WORD	Divides DWORD into 2 WORD data	
	Availability	XGI, XGR, XEC, XMC
	Flags	

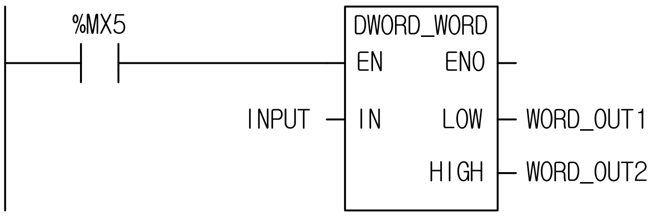
Function	Description
	**Input** EN: executes the function in case of 1 IN: DWORD Input **Output** ENO: outputs EN value as it is LOW: lower WORD Output HIGH: upper WORD Output

■ **Function**

It divides one DWORD into two WORD data.
LOW: lower WORD Output, HIGH: upper WORD Output

■ **Program Example**

1. LD

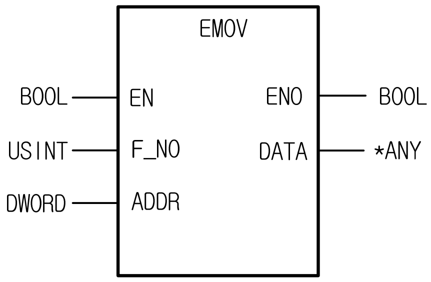


2. ST

DWORD_WORD(EN:=%MX5, IN:=INPUT, LOW=>WORD_OUT1, HIGH=>WORD_OUT2);

(1) If the transition condition (%MX5) is on, DWORD_WORD function executes.
(2) If input variable INPUT = 16#1122_AABB, then, WORD_OUT1 = 16#AABB and WORD_OUT2= 16#1122.

EMOV	Reading data from the preset flash area	
	Availability	XGI, XGR, XEC
	Flags	_ERR, _LER

Function	Description
	**Input** REQ : executes the function in case of 1 F_NO: Block NO(0~31) with the data to move ADDR: Byte address of a block set with B_NO. **Output** EN0: produces 1 if executing without error DATA: data saving area (all variables except BOOL and STRING available)

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	DATA		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: exclude BOOL and STRING from ANY type.

■ **Function**

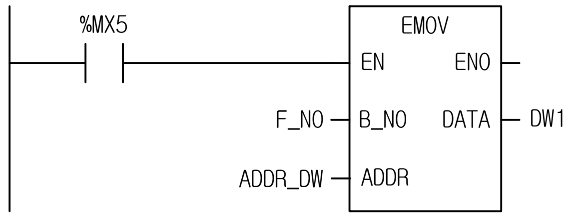
1. The command moves one data among 32 block data in flash memory.
2. It moves the data in ADDR of the F_NO (flash number) block according to the type set in DATA. then the moved data is entered to DATA variable.
3. If the variable type declared as DATA and the ADDR variable type are not identical, it does not produce any error but any undesirable data may be moved; set ADDR value according to DATA type. For instance, if declaring 4BYTE type variables (DWORD, UDINT, DINT, REAL ...) to DATA, ADDR variable must also use 4BYTE type variable.
4. If F_NO is 31 and greater or ADDR value exceeds 65,535, _ERR and _LER are set.

■ **Flag**

Flag	Description
_ERR	If F_NO value is 31 and over or ADDR value exceeds 65,535

■ **Program Example**

1. LD

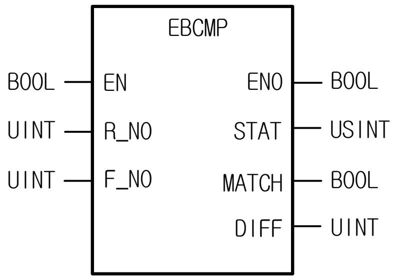


2. ST

EMOV(EN:=%MX5, F_NO:= F_NO, ADDR:= ADDR_DW, DATA=> DW1);

(1) If the execution condition (%MX5) is on, EMOV function executes.
(2) If setting F_N0 = 1, ADDR_DW(DWORD type) = 4, move DWORD DATA in 4BYTE OFFSET of No.1 Flash Block to DW1(DWORD).

EBCMP	Check the consistency after comparing content	
	Availability	XGI, XGR, XEC, XMC
	Flags	

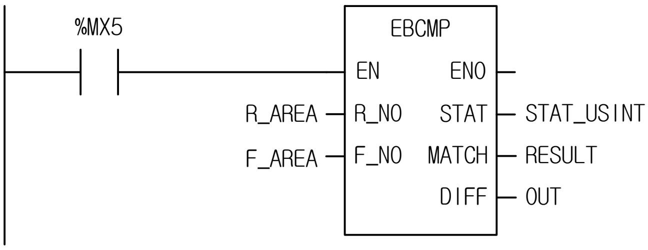
Function	Description
	**Input** EN: executes the function in case of 1 R_N0: R device block no. F_N0: Flash memory block no. **Output** ENO: On if comparison is complete STAT: Error status MATCH: On if comparison results are consistent DIFF: No. of inconsistency (DWORD)

■ **Function**

1. The command to check the consistency by comparing a block of R device and another block of flash memory while input contact is on; it compares data in DWORD.
2. STAT shows error status; if it is greater than 1 in R_NO input, STAT = 1; if it is greater than 31 in F_NO input, STAT = 2. Even though there is only one error after the entire comparison, it shows an error; STAT = 3.
3. In case of inconsistency, it saves the number in DIFF.

■ **Program Example**

1. LD

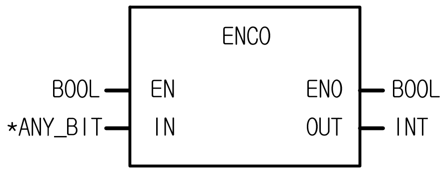


2. ST

EBCMP(EN:=%MX5, R_NO:=R_AREA, F_NO:=F_AREA, STAT=>STAT_USINT, MATCH=>RESULT, DIFF=>OUT);

(1) If the execution condition (%MX5) is on, EBCMP function executes.
(2) If setting R_AREA = 0, F_AREA = 1 and if R device block no.0 and flash block no.1 are consistent, RESULT(BOOL) is on and shows OUT(no. of inconsistency) = 0.

ENCO	Produces On bit position as number	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: input data to encode **Output** ENO: without an error, it is 1 OUT: Encoding result data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT							○													

*ANY_BIT: exclude BOOL from ANY_BIT type.

■ **Function**

1. If EN is 1, it produces the most priority bit position among bits of 1 to OUT.
2. Input data types are B(BYTE), W(WORD), D(DWORD) and L(LWORD).

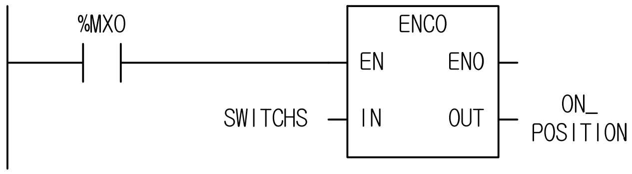
FUNCTION	IN type	Description
ENCO	BYTE	Uses a desirable ENCO function type depending on input variable type.
ENCO	WORD	
ENCO	DWORD	
ENCO	LWORD	

■ **Flag**

Flag	Description
_ERR	OUT is -1 if no bit among input data is 1; _ERR and _LER flags are set.

■ **Program Example**

1. LD



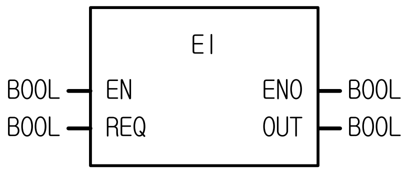
2. ST

ON_POSITION := ENCO(EN:=%MX0, IN:=SWITCHS);

(1) If the execution condition (%MX0) is on, ENCO function executes.

(2) If SWITCHS (WORD type) = 2#0000_1000_0000_0010, it produces the positions of 2 bits with on, that is, ‘11’ out of ‘11’ and ‘1’, so that ‘11’ is saved into ON_POSITION(INT Type).

EI	**Permits running for task program (Cancel of DI)**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

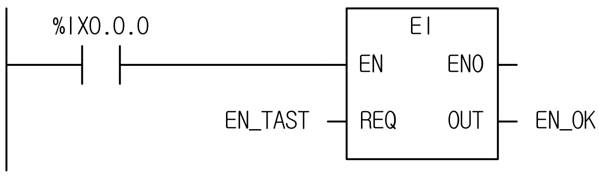
Function	Description
	**Input** EN: executes the function in case of 1 REQ: requires to permit running for task program **Output** ENO: outputs EN value as it is OUT: If EI is executed, an output is 1

■ **Function**

1. If EN is 1 and REQ input is 1, task program blocked by 'DI' function starts normally.
2. Once 'EI' command executes, task program starts normally even if REQ input is 0.
3. Task programs created when they are not permitted to operate executes after 'EI' function or the current-running task program execution.

■ **Program Example**

1. LD

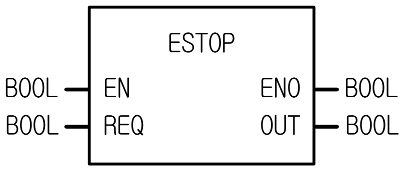


2. ST

```
EN_OK := EI(EN:=%IX0.0.0, REQ:=EN_TAST);
```

(1) If EN_TASK is 1, a task program starts normally.
(2) If EI function permits running for a task program, output EN_OK is 1.

ESTOP	Emergency running stop by program	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 REQ: requires the emergency running stop **Output** ENO: outputs EN value as it is. Refer to function 1 OUT: if ESTOP executes, an output is 1

■ **Function**

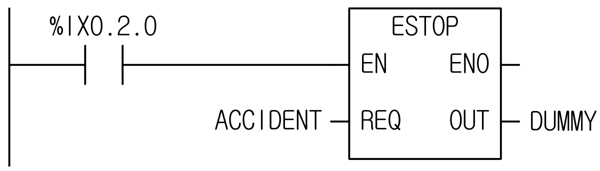
1. If transition condition EN is 1 and the signal to require the emergency running stop by program REQ is 1, program operation stops immediately and returns to STOP mode.
2. In case that a program stops by 'ESTOP' function, it does not start despite of power re-supply.
3. If operation mode moves from STOP to RUN, it restarts.
4. If 'ESTOP' function executes, it stops the running program during operation; if it is not a cold restart mode, an error may occur when restarts.

■ **Flag**

Flag	Description
_ESTOP_ON	It turns On if the program is stopped by ESTOP command. It is off when the program enters into RUN in the status.

■ **Program Example**

1. LD



2. ST

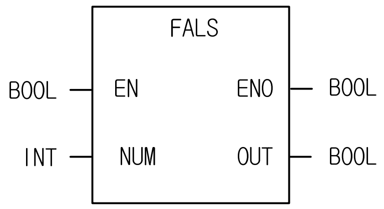
DUMMY := ESTOP(EN:=%IX0.2.0, REQ:=ACCIDENT);

(1) If the transition condition (%IX0.2.0) is on, ESTOP function executes.

(2) If ACCIDENT = 1, the running program stops immediately and returns to STOP mode.

※ In case of emergency, it is available to use it as a double safety device with mechanical interrupt.

FALS	Saving a user-defined constant(N) to the designated address in F(_FALS_NUM)	
	Availability	XGI, XGR
	Flags	

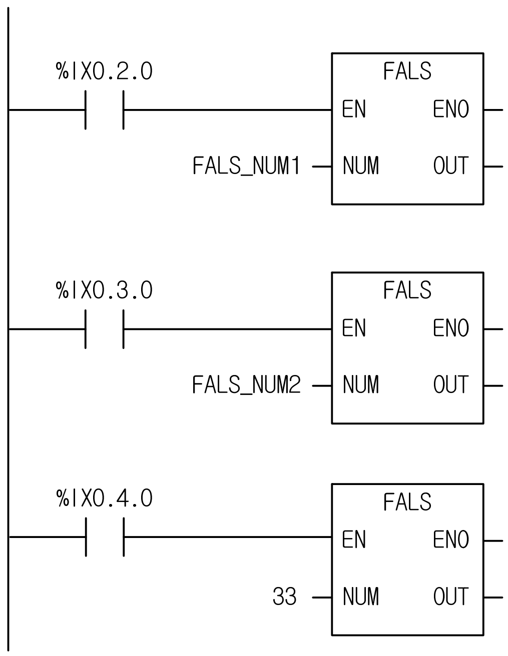
Function	Description
	**Input** EN: executes the function in case of 1 NUM: number to be saved in F **Output** ENO: outputs EN value as it is OUT: produces on if it normally works

■ **Function**

1. The command saves a user-defined constant (N) to the designated address in F (_FALS_NUM).
2. NUM can be designated between 16#0000 ~ 16#FFFF and the first generated number is saved until it is cancelled.
3. To cancel FALS, FALS 0000 executes.

■ **Program Example**

1. LD



2. ST

```
OUT1 := FALS(EN:=%IX0.2.0, NUM:=FALS_NUM1);
OUT2 := FALS(EN:=%IX0.3.0, NUM:=FALS_NUM2);
OUT3 := FALS(EN:=%IX0.4.0, NUM:=33);
```

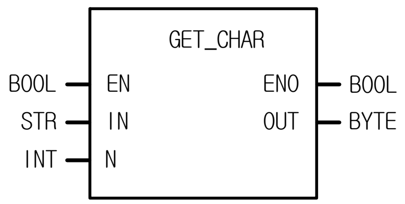
(1) If the execution condition is on, each FALS function executes (ex: FALS_NUM1=31, FALS_NUM2=32).

(2) The value is saved in _FALS_NUM Flag according to the execution condition (%IX0.2.0, %IX0.3.0, %IX0.4.0), the value is saved into the first _FALS_NUM_Flag, and the next value is not saved until FALS is canceled.

(3) To cancel FALS, 0000 must be set in NUM.

(4) It is convenient to view the status if executing the program by setting a value of special condition and checking _FALS_NUM Flag.

GET_CHAR	Gets one character from a String	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: STRING input N: position in a String **Output** ENO: outputs EN value as it is OUT: Byte Output

■ **Function**

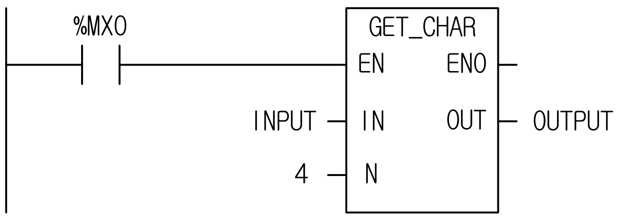
1. It extracts one byte from a String starting from N.

■ **Flag**

Flag	Description
_ERR	_ERR/_LER flags are set if N exceeds the number of byte in STRING. If an error occurs, the output is 16#00.

■ **Program Example**

1. LD



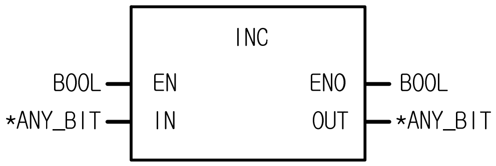
2. ST

OUTPUT := GET_CHAR(EN:=%MX0, IN:=INPUT, N:=4);

(1) If the transition condition (%MX0) is on, GET_CHAT function executes.

(2) When input INPUT (STRING) = “LS XGI PLC,” if you extract 4th character from this string, output variable OUTPUT is 16#58 (“X”).

INC	Increase IN data by 1	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN: Input data to increase **Output** ENO: outputs EN value as it is OUT: result data after increase

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT type.

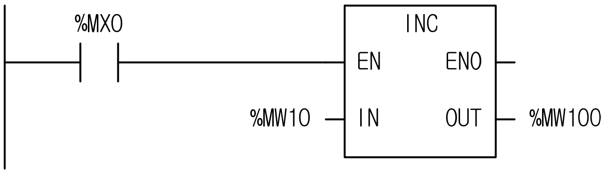
■ **Function**

1. If EN is 1, it increases IN bit string data by 1 and produces an output.
2. An error does not occur when there's an overflow; the result is 16#0000 in case of 16#FFFF.
3. Input data types are BYTE, WORD, DWORD and LWORD.

FUNCTION	IN/OUT type	Description
INC	BYTE	You can select one of these functions according to the in/out data type.
INC	WORD	
INC	DWORD	
INC	LWORD	

■ **Program Example**

1. LD



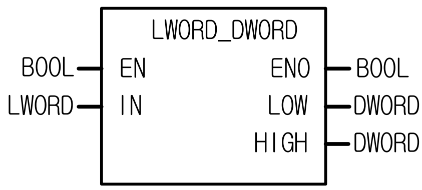
2. ST

```
%MW100 := INC(EN:=%MX0, IN:=%MW10);
```

(1) If the transition condition (%MX0) is on, INC function executes.

(2) If input variable %MW10 = 16#0007 (2#0000_0000_0000_0111), then output variable %MW100 =16#0008 (2#0000_0000_0000_1000).

LWORD_DWORD	Divides LWORD into two DWORD data	
	Availability	XGI, XGR, XEC, XMC
	Flags	

<table>
<tr><th>Function</th><th>Description</th></tr>
<tr><td>
</td><td>Input EN: executes the function in case of 1
IN: LWORD Input

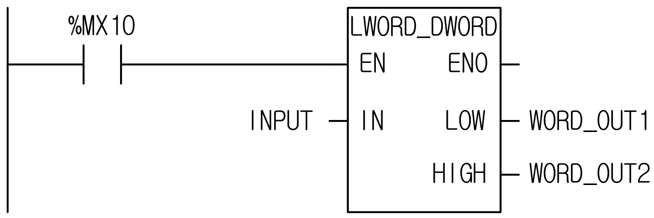
Output ENO: outputs EN value as it is
LOW: lower DWORD Output
HIGH: upper DWORD Output</td></tr>
</table>

■ **Function**

1. It divides one LWORD into two DWORD data.
   LOW: lower DWORD Output, HIGH: upper DWORD Output

■ **Program Example**

1. LD



2. ST

LWORD_DWORD(EN:=%MX10, IN:=INPUT, LOW=>DWORD_OUT1, HIGH=>DWORD_OUT2);

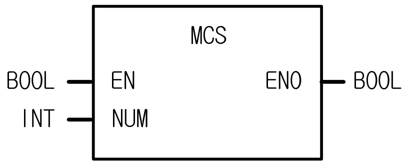
(1) If the transition condition (%MX10) is on, LWORD_DWORD function executes.

(2) If the input variable INPUT = 16#AAAA_BBBB_CCCC_DDDD, then

DWORD_OUT1 = 16#CCCC_DDDD

DWORD_OUT2 = 16#AAAA_BBBB.

MCS	Master Control	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1. NUM: Nesting (0~15) **Output** ENO: If MCS is executed, it is 1

■ **Function**

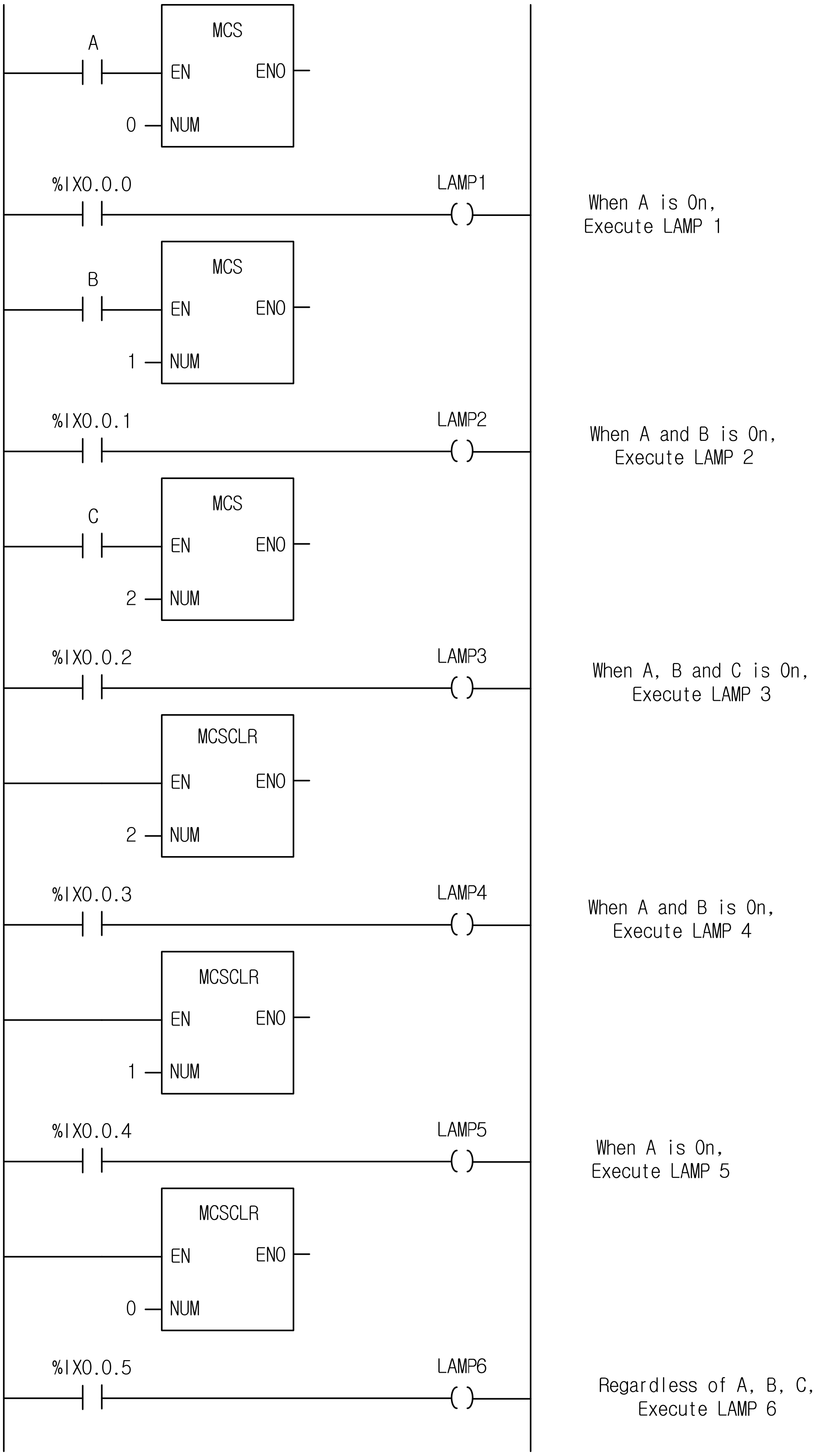
1. If EN is on, MCS function executes and the program between MCS and MCSCLR function is normally executes.
2. If EN is off, the program between MCS and MCSCLR function executes as follows:

Instruction	Description
Timer	Current value (CV) becomes 0 and the output (Q) becomes off.
Counter	Output (Q) becomes off and CV retains its present state.
Coil	All becomes off.
Negated coil	All becomes off.
Set coil, reset coil	All retains its current value.
Function, function block	All retains its current value.

3. Even when EN is off, scan time is not shortened because the instructions between MCS and MCSCLR function are executed as the above.
4. Nesting is available in MCS. That is to say, Master Control is divided by Nesting (NUM). You can set up Nesting (NUM) from 0 to 15 and if you set it more than 16, MCS is not executed normally.
   * Note: if you use MCS without 'MCSCLR', MCS function executes till the end of the program.

■ **Program Example**

1. LD

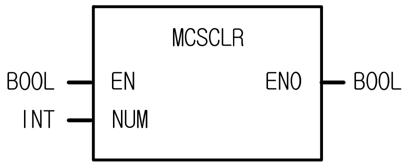


2. ST

```
MCS(EN:=A, NUM:=0);
LAMP1 := %IX0.0.0;          // When A is on, execute LAMP1

MCS(EN:=B, NUM:=1);
LAMP2 := %IX0.0.1;          // When A and B are on, execute LAMP2

MCS(EN:=C, NUM:=2);
LAMP3 := %IX0.0.2;          // When A, B and C are on, execute LAMP3

MCSCLR(NUM:=2);
LAMP4 := %IX0.0.3;          // When A and B are on, execute LAMP4

MCSCLR(NUM:=1);
LAMP5 := %IX0.0.4;          // When A is on, execute LAMP5

MCSCLR(NUM:=0);
LAMP6 := %IX0.0.5;          //Regardless of A, B, C, execute LAMP6
```

(1) The value corresponding to NUM of each MCS function sets an area with its counterpart, MCSCLR of the number. NESTING (NUM) can be set between 0~15 and the higher number is not allowed. Unless MCS and MCSCLR are combined as a pair, MCS function executes to the end of the program.

MCSCLR	Master Control Clear	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 NUM: Nesting (0~15) **Output** ENO: if MCSCLR is executed, it will be 1

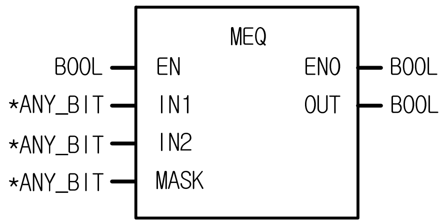
■ **Function**

1. It clears a Master Control instruction. And it indicates the end of the Master Control.
2. If MCSCLR function executes, it clears all the MCS instructions which are less than or equal to Nesting (NUM).
3. There’s no contact before MCSCLR function.

■ **Program Example**

Refer to the MCS function example.

MEQ	Masked Equal	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1. IN1: Input1 IN2: Input2 MASK: input data to mask **Output** ENO: outputs EN value as it is OUT: when equal, it is 1

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1		○	○	○	○															
	IN2		○	○	○	○															
	MASK		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT type.

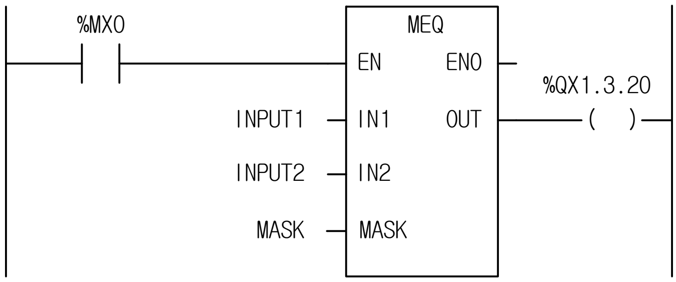
■ **Function**

1. It compares whether two input variables are equal after masking. If it masks an 8-bit variable with 2#11111100, then, lower 2 bits are excluded when it compares input values.
2. It's available to see whether or not specific bits are on in a variable. For example, in case of comparing 8-bit variables, IN1 is an input variable, IN2 is 16#FF, and MASK for masking is a bit array 2#00101100. If IN1 and IN2 after masking are equal, then output OUT is 1.

Function	Input type	Description
MEQ	BYTE	It compares whether two variables are equal after making.
MEQ	WORD	
MEQ	DWORD	
MEQ	LWORD	

■ **Program Example**

1. LD



2. ST

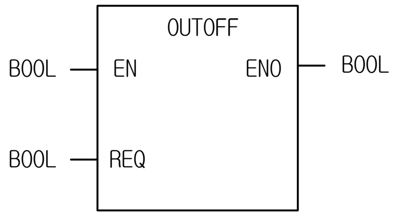
%QX1.3.20 := MEQ(EN:=%MX0, IN1:=INPUT1, IN2:=INPUT2, MASK:=MASK);

(1) If the transition condition (%MX0) is on, MEQ function executes.
(2) Input variable

INPUT1 (BYTE) = 2#01011100
INPUT2 (BYTE) = 2#01110101
MASK (BYTE) = 2#11010110
Then, the compared bits of input variables after masking are as follows:
INPUT1 (BYTE) = 2#01010100
INPUT2 (BYTE) = 2#01010100

INPUT1 and INPUT2 are equal; therefore, output contact %QX1.3.20 is on.

OUTOFF	Every Output Off if input condition is On	
	Availability	XGI, XGR, XEC, XMC
	Flags	

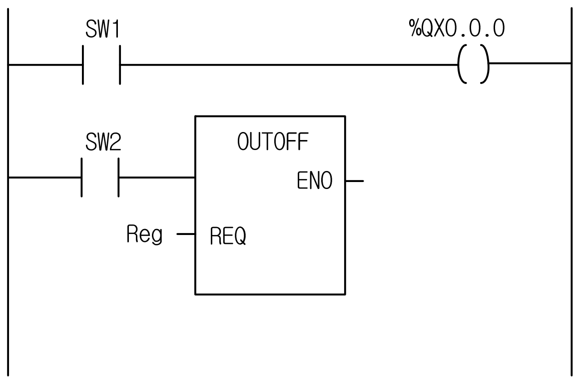
Function	Description
	**Input** EN : executes the function in case of 1 REQ: stop every output by program **Output** EN0: check the operation

■ **Function**

1. Every output is off if EN = 1 and REQ = 1.
2. Clear all the output off when EN = 1, REQ = 0.
3. Above and beyond these cases, it keeps the previous state.

■ **Program Example**

1. LD



2. ST

```
%QX0.0.0 := SW1;
  OUTOFF(EN:=SW2, REQ:= Reg);
```

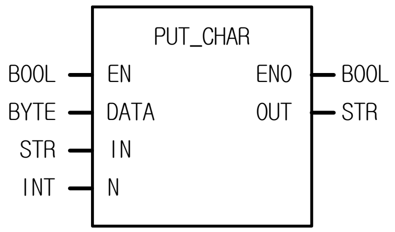
(1) It sets a program as the above example after output module establishes.

(2) if SW1 is on, the output (%QX0.0.0) is set.

(3) If operating with Reg = 1 after setting SW2 On, OUTOFF function is executed and every output module is off.

The actual output module is off although it seems to be set on the program monitor.

PUT_CHAR	Puts a character in a string	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 DATA: BYTE input to insert a STRING IN: STRING input N: setting position in a STRING **Output** ENO: outputs EN value as it is OUT: STRING output

■ **Function**

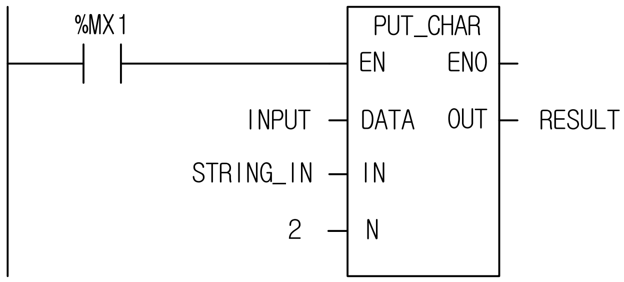
1. It overwrites one BYTE input on a specific position (N) string.

■ **Flag**

Flag	Description
_ERR	If N value exceeds a byte number of a string, _ERR and _LER flags are set. If an error occurs, the output is 16#00.

■ **Program Example**

1. LD



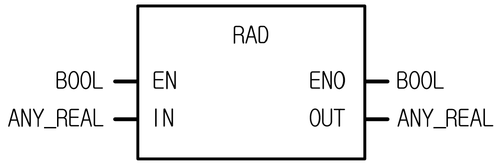
2. ST

RESULT := PUT_CHAR(EN:=%MX1, DATA:= INPUT, IN:= STRING_IN, N:=2);

(1) If the transition condition (%MX1) is on, PUT_CHAR function executes.

(2) If input variable INPUT = 16#41 (“A”) and STRING_IN = “TOKEN”, and N = 2, then, output RESULT is “TAKEN”.

RAD	Converts degree into radian	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1. IN: degree Input **Output** ENO: outputs EN value as it is OUT: radian output

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

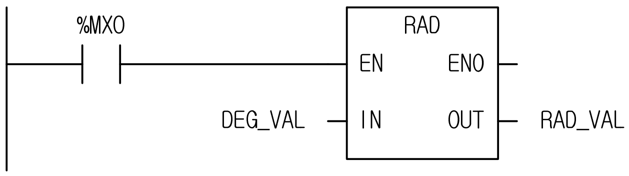
■ **Function**

1. It converts a degree value (°) into a radian value.
2. If the degree is over 360°, it converts normally.
   For example, if input is 370°, output is radian value corresponding to 370° - 360° = 10°.

Function	Input type	Output type	Description
RAD	REAL	REAL	It converts a degree value (°) into a radian value.
RAD	LREAL	LREAL	

■ **Program Example**

1. LD

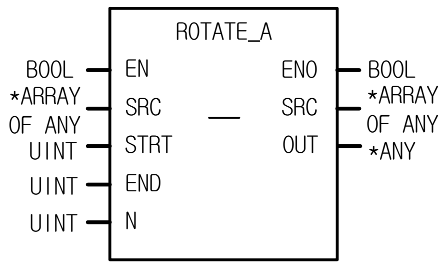


2. ST

RAD_VAL := RAD(EN:=%MX0, IN:= DEG_VAL);

(1) If the transition condition (%MX0) is on, RAD_REAL function executes.
(2) If input variable DEG_VAL = 127(°), its output RAD_VAL = 2.21656823.

ROTATE_A	Rotates designated array elements	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

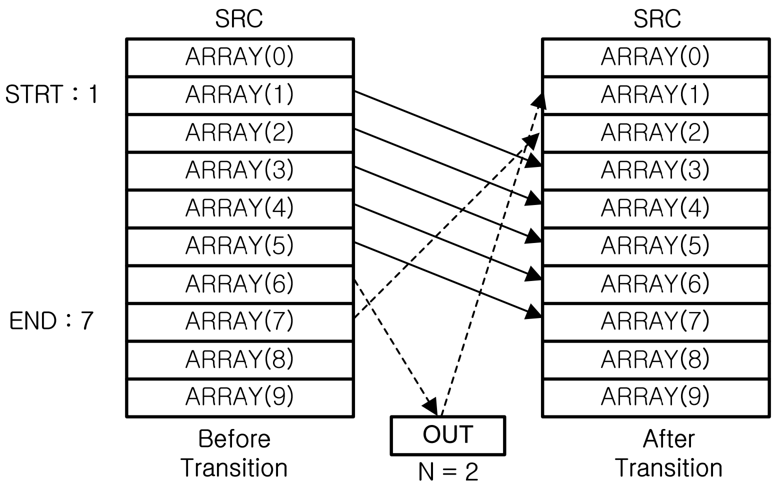
Function	Description
	**Input** EN: executes the function in case of 1 N: element number to rotate STRT: starting position to rotate in an array block END: ending position to rotate in an array block **Output** ENO: without an error, it is 1 OUT: overflowing data **In/Out** SRC: array block to rotate

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	SRC	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: exclude STRING from ANY type.

■ **Function**

1. It rotates designated elements of an array block in the chosen direction.
2. Setting:
   A. Scope: STRT and END set a data array to rotate.
   B. Rotation direction and time: rotates N times in the chosen direction set by STRT and END (STRT → END)
   C. Input data setting: fills an empty element with data pushed from END after rotation with Input data (IN)
   D. Output: the result is written at the ARRAY configured by SRC, and the data to rotate from END to STRT is written at OUT.



Function	In/Out array type	Description
ROTATE_A	BOOL	It rotates configured elements of an array block in the chosen direction.
ROTATE_A	BYTE	
ROTATE_A	WORD	
ROTATE_A	DWORD	
ROTATE_A	LWORD	
ROTATE_A	SINT	
ROTATE_A	INT	
ROTATE_A	DINT	
ROTATE_A	LINT	
ROTATE_A	USINT	
ROTATE_A	UINT	
ROTATE_A	UDINT	
ROTATE_A	ULINT	
ROTATE_A	REAL	
ROTATE_A	LREAL	
ROTATE_A	TIME	
ROTATE_A	DATE	
ROTATE_A	TOD	
ROTATE_A	DT	

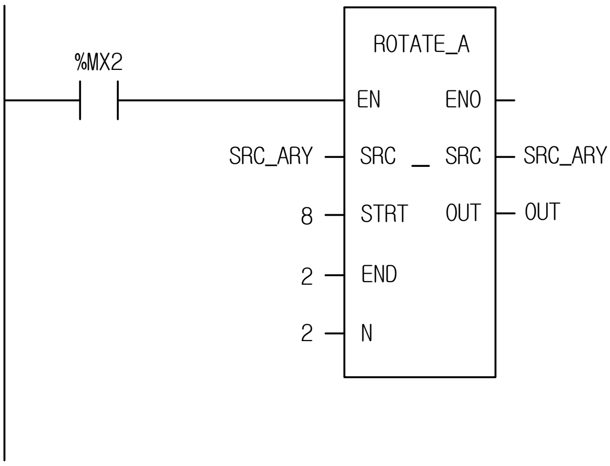
■ **Flag**

Flag	Description
_ERR	If STRT or END exceed the range of SRC array element, _ERR and _LER flags are set. If an error occurs, there's no change in SRC and output OUT is the initial value of each variable type(i.e. INT=0, TIME=T#0S).

☆ If output array variable is omitted, it assumes the output array number as 0, producing _ERR and _LER flags.

■ **Program Example**

1. LD



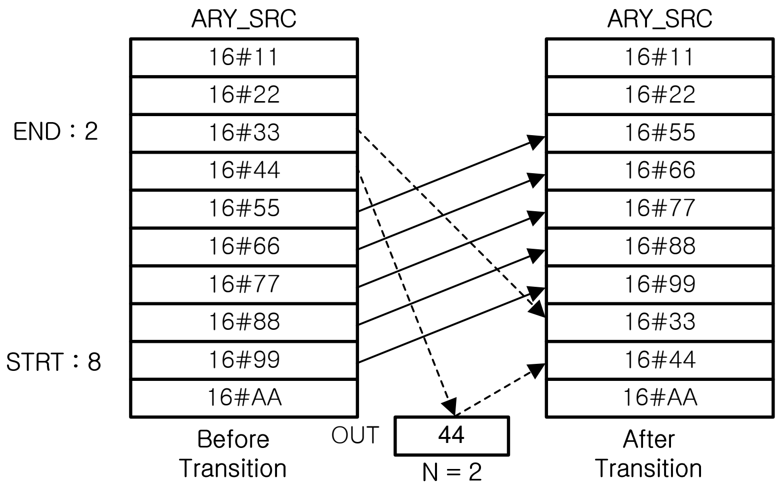
2. ST

OUT := ROTATE_A(EN:=%MX2, SRC:=SRC_ARY, STRT:=8, END:=2, N:=2);

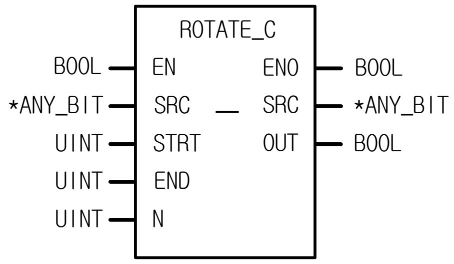
(1) If input condition (%MX2) is on, ROTATE_A function executes.

(2) It rotates designated elements (from 2nd to 8th elements) of SRC_ARY in the chosen direction set by STRT and END (from index 8 to index 2).

(3) The overflowing data (16#44) is written at OUT.



ROTATE_C	Rotate with Carry	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

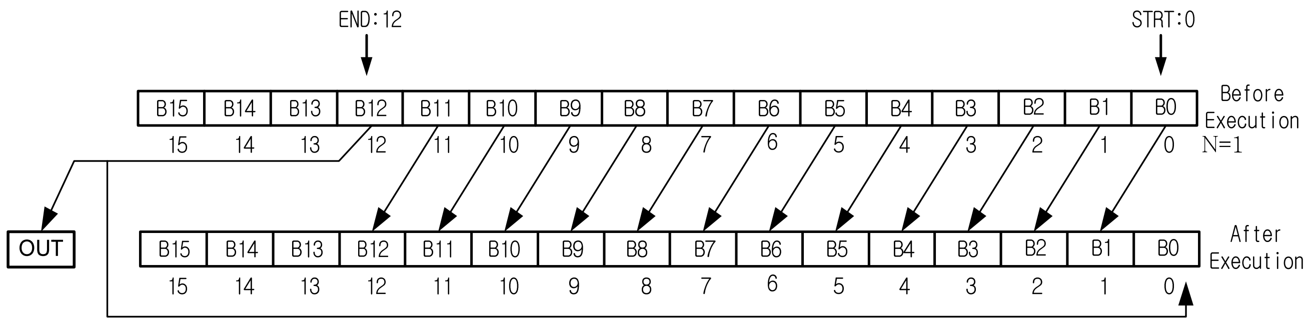
Function	Description
	**Input** EN: executes the function in case of 1. STRT: starting bit position of SRC bit array to rotate END: ending bit position of SRC bit array to rotate N: bit number to shift **Output** ENO: without an error, it is 1 OUT: carry output **In/Out** SRC: variable for rotation

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	SRC		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT type.

■ **Function**

1. It rotates a configured bit array of SRC bit arrays in the chosen direction.
2. Setting:
   A. Scope: STRT and END set a bit data to rotate.
   B. Rotation direction and time: rotates N times in the chosen direction set by STRT and END (STRT → END)
   C. Output: the result is written at ANY_BIT configured by SRC, and the data to rotate from END to STRT is written at OUT.



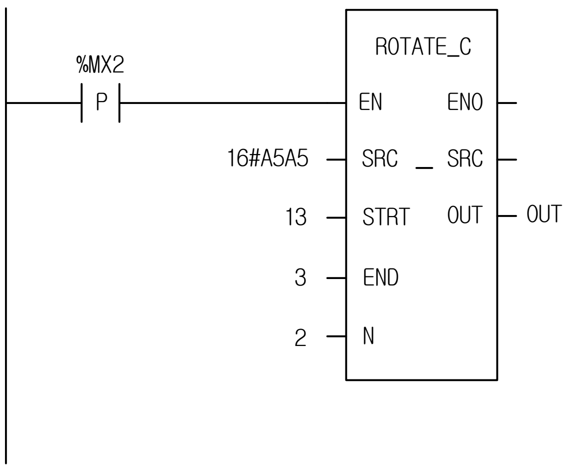
Function	SRC type	Description
ROTATE_C	BYTE	It rotates a designated bit array of SRC bit arrays N times in the chosen direction.
ROTATE_C	WORD	
ROTATE_C	DWORD	
ROTATE_C	LWORD	

■ **Flag**

Flag	Description
_ERR	If STRT or END exceed the bit number of SRC variable type, there's no change in SRC and _ERR and _LER flags are set

■ **Program Example**

1. LD



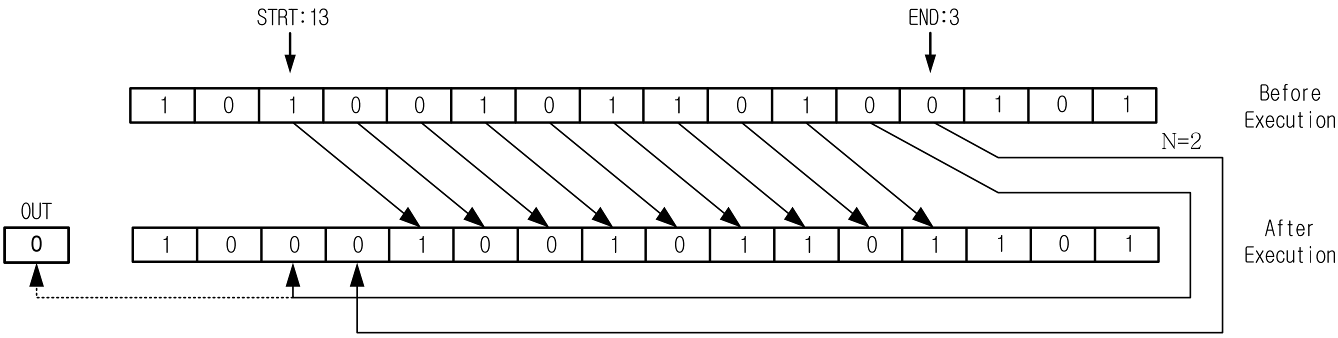
2. ST

OUT := ROTATE_C(EN:=%MX2, SRC:=16#A5A5, STRT:=13, END:=3, N:=2);

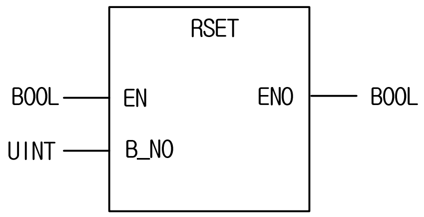
(1) If the transition condition (%MX2) is on, ROTATE_C function executes.

(2) It rotates the designated bit array, from STRT (13) to END (3), of SRC (16#A5A5) 2 times in the chosen direction set by STRT and END (from STRT to END): refer to the diagram below.

(3) The result data after rotation is written at SRC (16#896D), and the overflowing bit (0) is written at OUT.


STRT:13
END:3
1 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1
Before Execution
N=2
OUT
0
1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 1
After Execution

RSET	Converting the set block number to the designated block number	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 B_NO: block NO(0~1) to convert (XGI-CPUU/D, CPUUN : 0~15) **Output** EN0: without an error, it is 1

■ **Function**

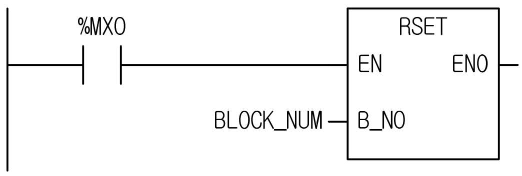
1. Convert the set block number (_RBANK_NUM) to the designated block number.
2. Block number is initialized to 0 if converting stop to run.
3. If S is over the max block number, error flag (_ERR) is set.

■ **Flag**

Flag	Description
_ERR	If B_N0 value is 2 and over (XGI-CPUU/D, CPUUN : 16 and over), _ERR and _LER Flags are set.

■ **Program Example**

1. LD



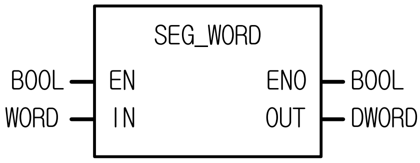
2. ST

RSET(EN:=%MX0, B_NO:=BLOCK_NUM);

(1) If the execution condition (%MX0) is on, RSET function executes.

(2) BLOCK_NUM (UINT type) can be 0 or 1 and convert it to the designated R block.

SEG_WORD	Converts BCD or HEX into 7 segment display code	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
 	**Input** EN: executes the function in case of 1. IN: Input data to covert into 7 segment code **Output** ENO: outputs EN value as it is OUT: result data converted into 7 segment data

■ **Function**

1. If EN is 1, it converts BCD or HEX (hexadecimal) of IN into 7 segment display code as follow and produces output, OUT.
2. If an input is BCD type, it is available to display a number between 0000 and 9999. And in case of HEX input, it's available to display a number between 0000 and FFFF on 4-digit 7 segment display.

Display example

1) 4-digit BCD -> 4-digit 7 segment code: use SEG function.

2) 4-digit HEX -> 4-digit 7 segment code: use SEG function.

3) INT -> 4-digit BCD-type 7 segment code: use INT_TO_BCD function first and SEG function.

4) INT -> 4-digit HEX-type 7 segment code: use INT_TO_WORD function first and SEG function.

5) When 7 segment display digits are more than 4.

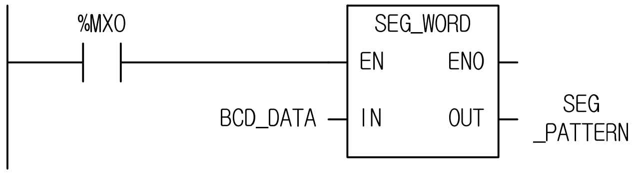
A) In case of BCD, HEX type, use SEG function, after dividing them into 4 digits.

B) INT -> 8-digit BCD-type 7 segment code:

Divide INT by 10,000 and convert ‘quotient’ and ‘remainder’ into upper/lower 4-digit 7 segment code using INT_TO_BCD and SEG function.

■ **Program Example**

1. LD



2. ST

SEG_PATTERN := SEG_WORD(EN:=%MX0, IN:=BCD_DATA);

(1) If the transition condition (%MX0) is on, SEG_WORD function executes.

(2) If input variable BCD_DATA (WORD) = 16#1234, the output is '2#00000110_01011011_01001111_01100110' which is displayed as a 7 segment code (1234) and written at SEG_PATTERN (DWORD).

INPUT(IN) : BCD_DATA(WORD) = 16#1234

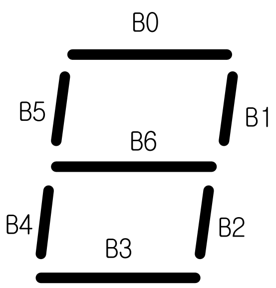
0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0

↓ (SEG)

OUTPUT(OUT) : SEG_PATTERN(DWORD) = 16#065B4F66

0	0	0	0	0	1	1	0	0	1	0	1	1	0	1	1	Upper
0	1	0	0	1	1	1	1	0	1	1	0	0	1	1	0	Lower

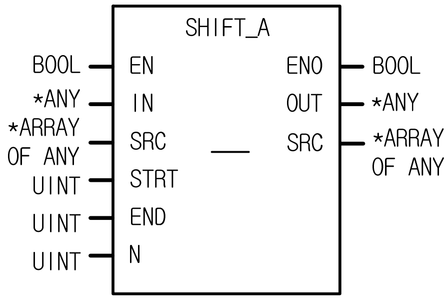
■ **7 Segment Configuration**



■ **Conversion table for 7 segment code**

Input (BCD)	Input (Hex)	INT	Output B7	B6	B5	B4	B3	B2	B1	B0	Display Data
0	0	0	0	0	1	1	1	1	1	1	**0**
1	1	1	0	0	0	0	0	1	1	0	**1**
2	2	2	0	1	0	1	1	0	1	1	**2**
3	3	3	0	1	0	0	1	1	1	1	**3**
4	4	4	0	1	1	0	0	1	1	0	**4**
5	5	5	0	1	1	0	1	1	0	1	**5**
6	6	6	0	1	1	1	1	1	0	1	**6**
7	7	7	0	0	1	0	0	1	1	1	**7**
8	8	8	0	1	1	1	1	1	1	1	**8**
9	9	9	0	1	1	0	1	1	1	1	**9**
-	A	10	0	1	1	1	0	1	1	1	**A**
-	B	11	0	1	1	1	1	1	0	0	**B**
-	C	12	0	0	1	1	1	0	0	1	**C**
-	D	13	0	1	0	1	1	1	1	0	**D**
-	E	14	0	1	1	1	1	0	0	1	**E**
-	F	15	0	1	1	1	0	0	0	1	**F**

SHIFT_A	Shifts designated array elements	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1. IN: Input data to empty element after shifting N: number to shift STRT: starting position to shift in an array block END: ending position to shift in an array block **Output** ENO: without an error, it is 1 OUT: overflowing data **In/Out** SRC: array block to shift

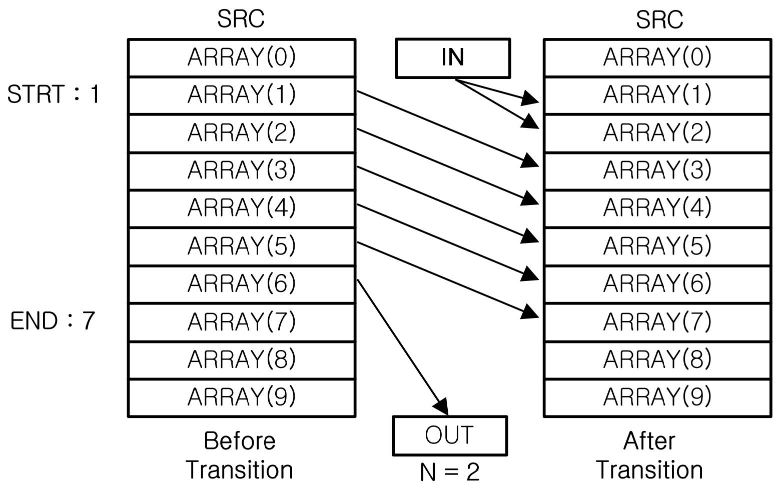
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: exclude STRING from ANY type.

■ **Function**

1. It shifts designated elements of an array block in the chosen direction.
2. Setting:

- Scope: STRT and END set a data array to rotate.
- Shifting direction and time: rotates N times in the chosen direction set by STRT and END (STRT → END).
- Input data setting: fills an empty element after shifting with input data (IN).
- Output: the result is written at ARRAY configured by SRC, and the overflowing data by shifting from END to STRT is written at OUT.



Function	In/Out Array Type	Description
SHIFT_A	BOOL	It shifts configured elements of an array block in the chosen direction.
SHIFT_A	BYTE	
SHIFT_A	WORD	
SHIFT_A	DWORD	
SHIFT_A	LWORD	
SHIFT_A	SINT	
SHIFT_A	INT	
SHIFT_A	DINT	
SHIFT_A	LINT	
SHIFT_A	USINT	
SHIFT_A	UINT	
SHIFT_A	UDINT	
SHIFT_A	ULINT	
SHIFT_A	REAL	
SHIFT_A	LREAL	
SHIFT_A	TIME	
SHIFT_A	DATE	
SHIFT_A	TOD	
SHIFT_A	DT	

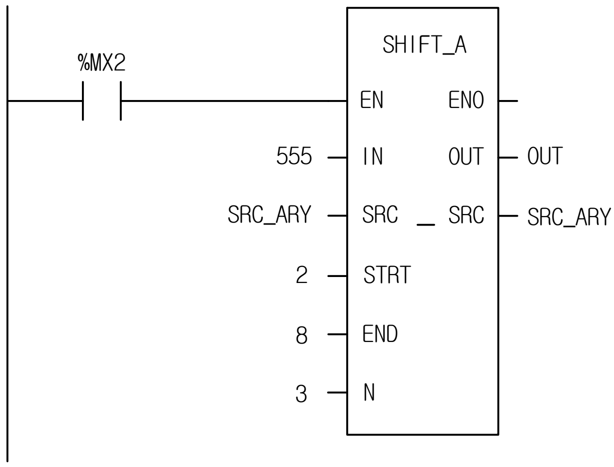
■ **Flag**

Flag	Description
_ERR	If STRT or END exceed the range of SRC array element, _ERR and _LER flags are set. If an error occurs, there's no change in SRC and output, OUT is the initial value of each variable type(i.e. INT=0, TIME=T#0S).

☆ If output array is omitted, it assumes the number of array as 0, producing _ERR and LER flags.

■ **Program Example**

1. LD



2. ST

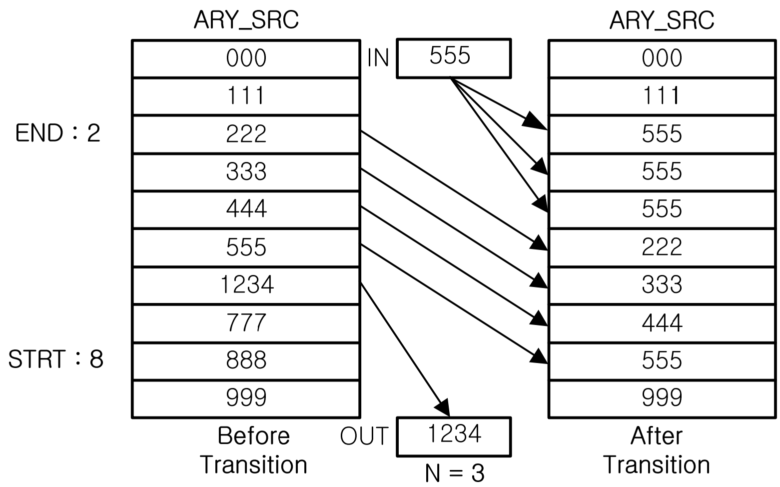
(1) If the input condition (%MX2) is on, SHIFT_A function executes.

(2) It shifts designated elements (from 2nd to 8th elements) of SRC_ARY.

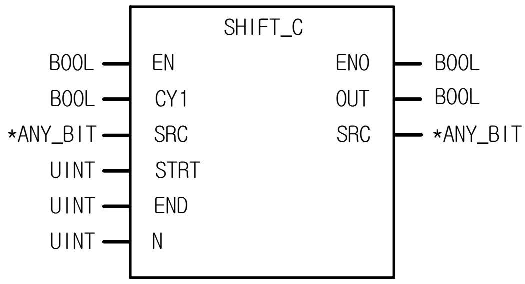
(3) It shifts three times the configured elements.

(4) The empty elements after shifting, from array index 2 to array index 3, are filled with input '555'.

(5) The overflowing data (1234), carry output, is written at OUT.



SHIFT_C	Shift with Carry	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 CYI: Carry Input STRT: starting bit position of SRC bit array to shift END: ending bit position of SRC bit array to shift N: bit number to shift **Output** ENO: without an error, it is 1 OUT: carry output **In/Out** SRC: variable to shift

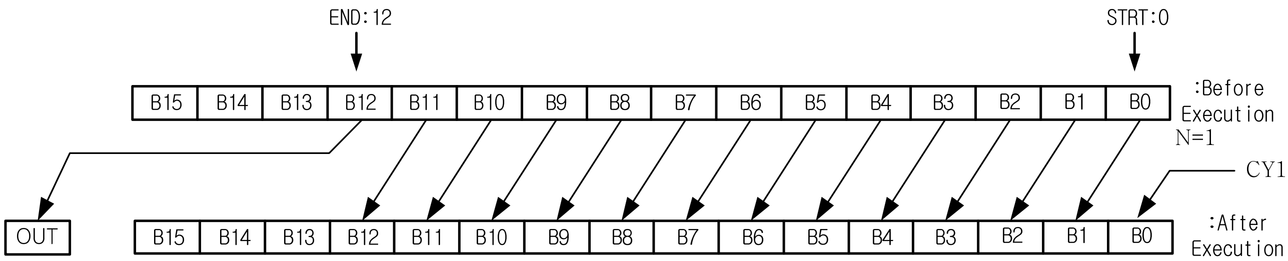
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT type.

■ **Function**

1. It shifts a configured bit array of SRC bit arrays N times in the chosen direction.
2. Setting:
   - Scope: STRT and END set a bit data to shift.
   - Shifting direction and time: shifts N times from STRT to END.
   - Input data setting: fills empty bit after shifting with input data (CYI).
   - Output: the result is written at ANY_BIT configured by SRC, and the overflowing bit data by shifting from END to STRT is written at OUT.

.



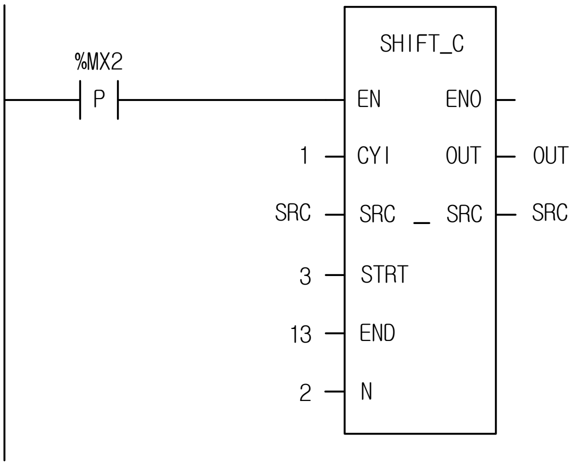
Function	SRC type	Description
SHIFT_C	BYTE	It shifts a configured bit array of SRC bit arrays N times.
SHIFT_C	WORD	
SHIFT_C	DWORD	
SHIFT_C	LWORD	

■ **Flag**

Flag	Description
_ERR	If STRT or END exceed the bit number of SRC variable type, there's no change in SRC and _ERR and _LER flags are set.

■ **Program Example**

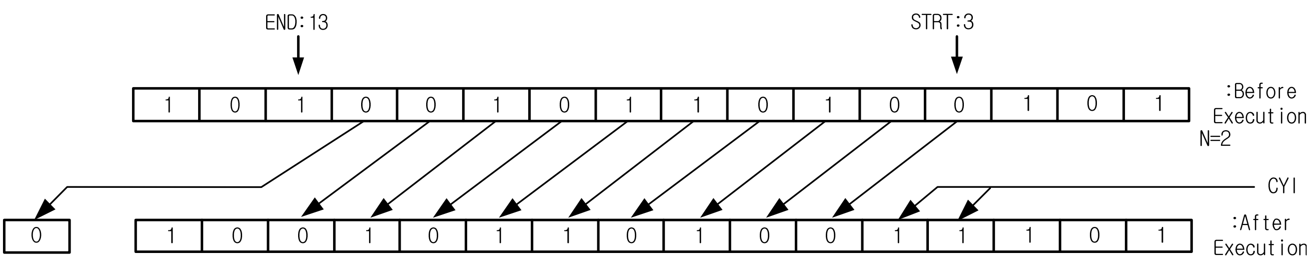
1. LD



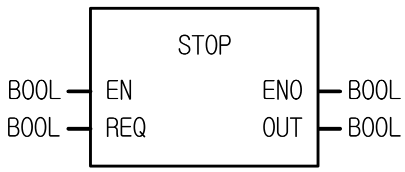
2. ST

OUT := SHIFT_C(EN:=%MX2, CYI:=1, SRC:=SRC, STRT:=3, END:=13, N:=2);

(1) If the transition condition (%MX2) is on, SHIFT_C function executes.
(2) 16#A5A5 is shifted from STRT to END by 2 bits and the empty bits after shifting are filled with 1 (CYI).
(3) SRC after shifting is 16#969D and the overflowing bit data (0) is written at OUT after 2-bit shifting.



STOP	Stop running by program	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 RE: requires the operation stop by program **Output** ENO: outputs EN value as it is OUT: If STOP function executes, it is 1.

■ **Function**

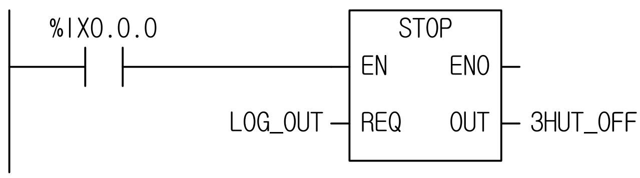
1. If EN and REQ are 1, stop running and return to STOP mode.
2. If function 'STOP' executes, the program stops after completing scan program in executing.
3. Program restarts in case of power re-supply or the change of operation mode from STOP to RUN.

■ **Flag**

Flag	Description
_USTOP_ON	On if stopped by STOP instruction. It is off if entering into RUN.

■ **Program Example**

1. LD



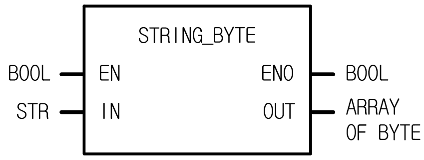
2. ST

3HUT_OFF := STOP(EN:=%IX0.0.0, REQ:=LOG_OUT);

(1) If the transition condition (%IX0.0.0) and LOG_OUT is 1, it enters to STOP mode after completing the scan program in executing.

(2) It is recommended to turn off the power of PLC in the stable state after executing 'STOP' function declared as input variable.

STRING_BYTE	Convert a string into a byte array	
	Availability	XGI, XGR, XEC, XMC
	Flags	

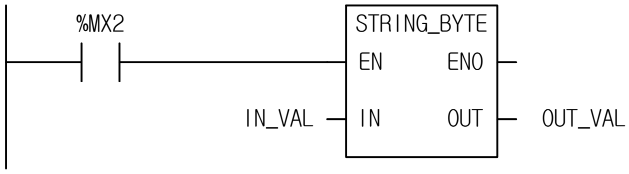
Function	Description
	**Input** EN: if EN is 1, function converts. IN: string input **Output** ENO: outputs EN value as it is OUT: outputs converted Byte Array

■ **Function**

It converts a string into 31 byte arrays.

■ **Program Example**

1. LD

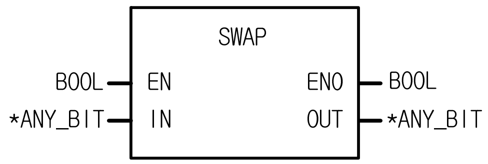


2. ST

OUT_VAL := STRING_BYTE(EN:=%MX2, IN:=IN_VAL);

(1) If the transition condition (%MX2) is on, STRING_BYTE function executes.
(2) If IN_VAL = ‘ABC’, OUT_VAL[0] = 16#41, OUT_VAL[1] = 16#42, OUT_VAL[2] = 16#43.

SWAP	Swaps upper data for lower data	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1. IN: Input **Output** ENO: outputs EN value as it is OUT: swapped data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT type.

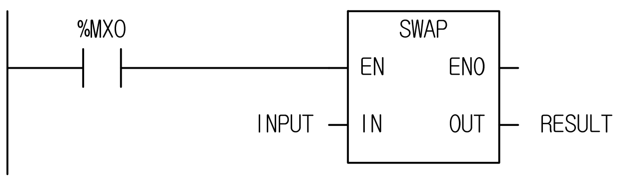
■ **Function**

It swaps upper data for lower data.

Function	Input type	Description
SWAP	BYTE	Swaps upper nibble for lower nibble data.
SWAP	WORD	Swaps upper byte for lower byte data.
SWAP	DWORD	Swaps upper word for lower word data.
SWAP	LWORD	Swaps upper double word for lower double word data.

■ **Program Example**

1. LD



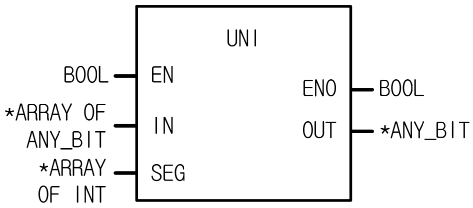
2. ST

```
RESULT := SWAP(EN:=%MX0, IN:=INPUT);
```

(1) If the transition condition (%MX0) is on, SWAP function executes.

(2) If INPUT (BYTE) = 16#5F, RESULT (BYTE) = 16#F5.

UNI	Unites data	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN: input data array SEG: bit-number-designate array to united data **Output** ENO: without an error, it is 1 OUT: united data output

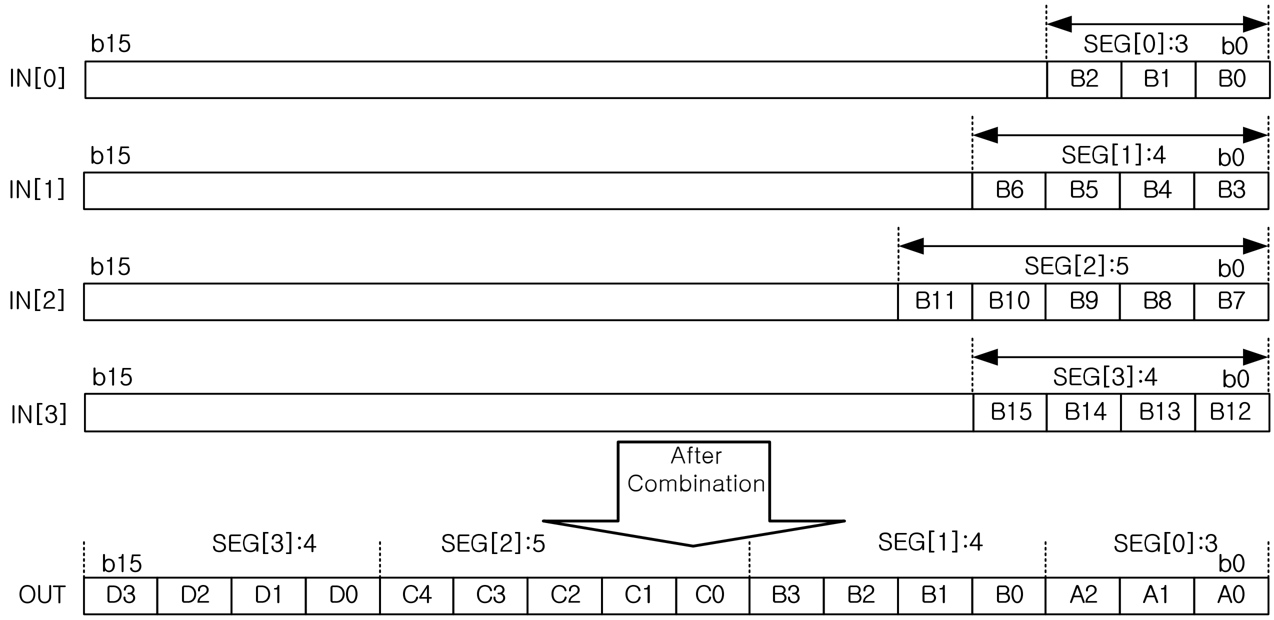
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: exclude BOOL from ANY_BIT type.

■ **Function**

1. It unites an input data array from the lower bit to a configured bit set by SEG and produces an output.

Function	Input type	Output type	Description
UNI	BYTE	BYTE	It cuts an input array into bit data set by SET and produces an output (united data) with the same array type of input.
UNI	WORD	WORD	
UNI	DWORD	DWORD	
UNI	LWORD	LWORD	



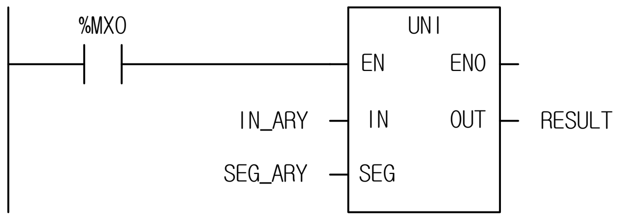
If the sum of value set by SEG exceeds the bit number of input data type, _ERR and _LER flags are set.

■ **Flag**

Flag	Description
_ERR	If the sum of value set by SEG exceeds the bit number of input data type, _ERR and _LER flags are set. If the number of arrays of IN and SEG is different, output OUT is 0 and _ERR and _LER flags are set.

■ **Program Example**

1. LD



2. ST

RESULT := UNI(EN:=%MX0, IN:=IN_ARY, SEG:=SEG_ARY);

(1) If the transition condition (%MX0) is on, UNI function executes.

(2) If input IN_ARY and SEG_ARY are as below

IN_ARY[0]	A3B5	SEG_ARY[0]	3
IN_ARY[1]	B4C6	SEG_ARY[1]	4
IN_ARY[2]	C5D7	SEG_ARY[2]	7
IN_ARY[3]	D6E8	SEG_ARY[3]	2

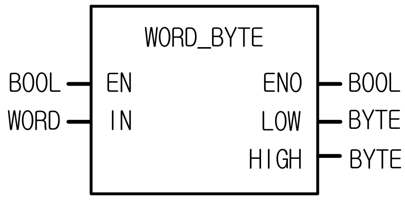
output RESULT = 2#0010_1011_1011_0101 = 16#2BB5.

IN_ARY[0]	2#1010 0011 1011 0101	SEG_ARY[0]	3
IN_ARY[1]	2#1011 0100 1100 0110	SEG_ARY[1]	4
IN_ARY[2]	2#1100 0101 1101 0111	SEG_ARY[2]	7
IN_ARY[3]	2#1101 0110 1110 1000	SEG_ARY[3]	2

↓

RESULT : 2#00 1010111 0110 101

WORD_BYTE	Divides WORD into two bytes	
	Availability	XGI, XGR, XEC, XMC
	Flags	

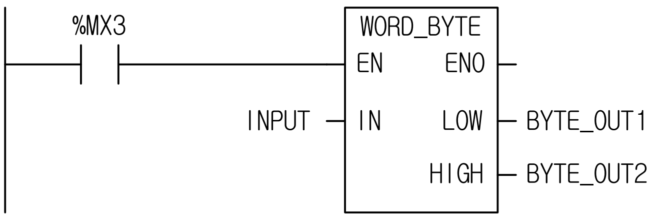
Function	Description
	**Input** EN: executes the function in case of 1 IN: WORD Input **Output** ENO: outputs EN value as it is LOW: lower BYTE output HIGH: upper BYTE output

■ **Function**

1. It divides one word data into two byte data.
   LOW: lower byte output, HIGH: upper byte output

■ **Program Example**

1. LD

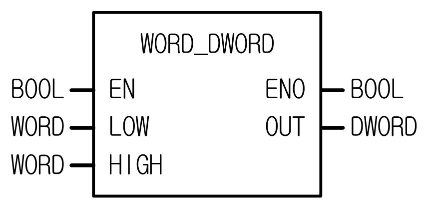


2. ST

```
WORD_BYTE(EN:=%MX3, IN:=INPUT, LOW=>BYTE_OUT1, HIGH=>BYTE_OUT2);
```

(1) If the transition condition (%MX3) is on, WORD_BYTE function executes.
(2) If input variable INPUT is 16#ABCD, then BYTE_OUT1 = 16#CD and BYTE_OUT2 = 16#AB.

WORD_DWORD	Combines two WORD data into DWORD	
	Availability	XGI, XGR, XEC, XMC
	Flags	

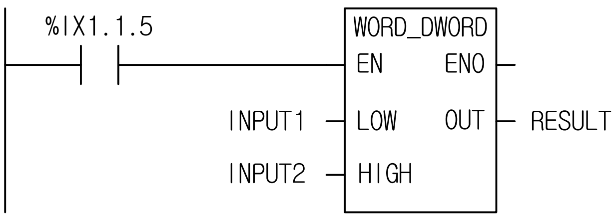
Function	Description
	**Input** EN: executes the function in case of 1. LOW: lower WORD input HIGH: upper WORD input **Output** ENO: outputs EN value as it is OUT: DWORD output

■ **Function**

It combines two WORD data into one DWORD.
LOW: lower WORD input, HIGH: upper WORD input.

■ **Program Example**

1. LD

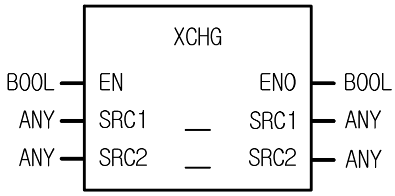


2. ST

RESULT := WORD_DWORD(EN:=%IX1.1.5, LOW:=INPUT1, HIGH:=INPUT2);

(1) If the transition condition (%IX1.1.5) is on, WORD_DWORD function executes.
(2) If input variable INPUT1 = 16#1020 and INPUT2 = 16#A0B0, output variable RESULT=16#A0B0_1020.

XCHG	Exchanges two input data	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 **Output** ENO: outputs EN value as it is **In/Out** SRC1: In/Output 1 SRC2: In/Output 2

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	SRC1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	SRC2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ **Function**

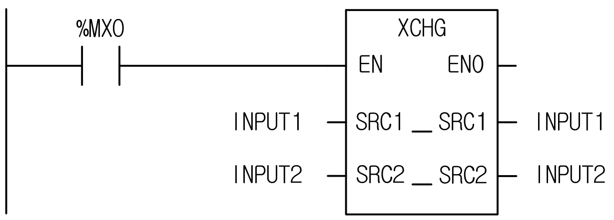
1. Exchanges input1 data with input2 data.

Function	In/Out type	Description
XCHG	BOOL	Exchanges two BOOL input data.
XCHG	BYTE	Exchanges two BYTE input data.
XCHG	WORD	Exchanges two WORD input data.
XCHG	DWORD	Exchanges two DWORD input data.
XCHG	LWORD	Exchanges two LWORD input data.
XCHG	SINT	Exchanges two SINT input data.
XCHG	INT	Exchanges two INT input
XCHG	DINT	Exchanges two DINT input data.
XCHG	LINT	Exchanges two LINT input data.
XCHG	USINT	Exchanges two USINT input data.
XCHG	UINT	Exchanges two UINT input data.
XCHG	UDINT	Exchanges two UDINT input data.
XCHG	ULINT	Exchanges two ULINT input data.
XCHG	REAL	Exchanges two REAL input data.
XCHG	LREAL	Exchanges two LREAL input data.

Function	In/Out type	Description
XCHG	TIME	Exchanges two TIME input data.
XCHG	DATE	Exchanges two DATE input data.
XCHG	TOD	Exchanges two TOD input data.
XCHG	DT	Exchanges two DT input data.
XCHG	STRING	Exchanges two STRING input data.

■ **Program Example**

1. LD

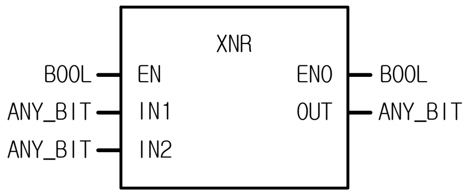


2. ST

XCHG(EN:=%MX0, SRC1:=INPUT1, SRC2:=INPUT2);

(1) If the transition condition (%MX0) is on, XCHG function executes.
(2) If INPUT1 = 0 and INPUT2 = 1, it will exchange two input data. After the function execution, INPUT1 = 1 and INPUT2 = 0.

XNR	Exclusive Logical AND	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1 IN1: XNR-to-be value IN2: XNR-to-be value Input variables can be extended up to 8. **Output** ENO: outputs EN value as it is OUT: XNR result IN1, IN2, and OUT must be of the same data type.

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○															
	IN2	○	○	○	○	○															
	OUT	○	○	○	○	○															

■ **Function**

1. It performs XNR operation on the input variables by bit and produces output, OUT.

IN1 1111 0000

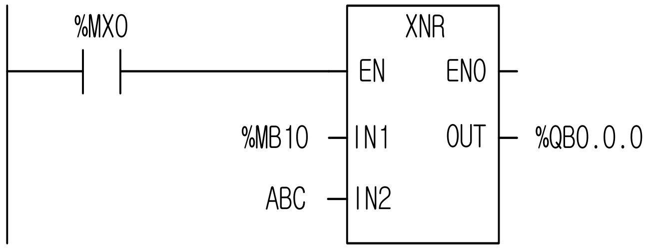
XNR

IN2 1010 1010

OUT 1010 0101

■ **Program Example**

1. LD



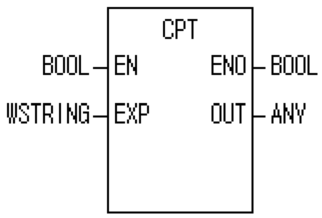
2. ST

%QB0.0.0 := XNR(EN:=%MX0, IN1:=%MB10, IN2:=ABC);

(1) If the transition condition (%MX0) is on, XNR function executes.

(2) If %MB10 = 16#F0 = 2#1111_0000 and ABC(BYTE type) = 16#AA = 2#1010_1010, the result of XNR is shown in OUT (%QB0.0.0 = 16#A5 = 2#1010_0101).

CPT	ST expression computation	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function	Description
	**Input** EN: executes the function in case of 1. EXP: ST expression **Output** ENO: outputs EN value as it is OUT: result data

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	WSTRING
	IN																				○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○					

■ **Function**

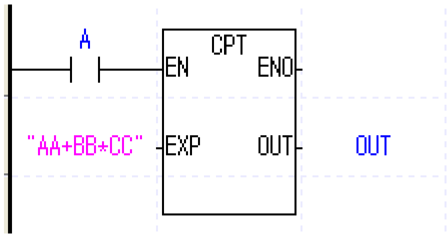
1. If EN is 1, it produces an output after computation of EXP input ST expression.
2. Maximum size of input expression is 100 Byte. (English : 100 character)
3. Available functions to expression are only comparison, numerical operation, degree conversion and type conversion.
   (1) Comparison: EQ, GE, GT, LE, LT, NE
   (2) Numerical operation: ABS, ACOS, ADD, ASIN, ATAN, COS, DIV, EXP, EXPT, LN, LOG, MOVE, MUL, SIN, SQRT, SUB, TAN, TRUNC (but MOD is not available, operated as a keyword)
   (3) Degree conversion: DEG, RAD
   (4) Type conversion: Type conversion functions without special symbol (***)
4. Refer to ST instruction manual for the information of ST expression

FUNCTION	IN/OUT type	Description
CPT	BOOL	Output value must be BOOL type.
CPT	BYTE	Output value must be BYTE type.
CPT	WORD	Output value must be WORD type.
CPT	DWORD	Output value must be DWORD type.
CPT	LWORD	Output value must be LWORD type.
CPT	SINT	Output value must be SINT type.
CPT	INT	Output value must be INT type.

CPT	DINT	Output value must be DINT type.
CPT	LINT	Output value must be LINT type.
CPT	USINT	Output value must be USINT type.
CPT	UINT	Output value must be UINT type.
CPT	UDINT	Output value must be UDINT type.
CPT	ULINT	Output value must be ULINT type.
CPT	REAL	Output value must be REAL type.
CPT	LREAL	Output value must be LREAL type.

■ **Program Example**

1. LD



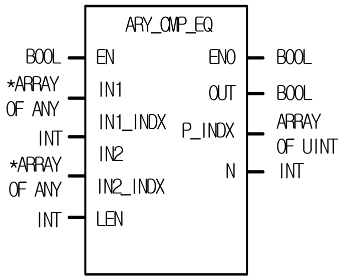
2. ST

```
-CPT function is not available. But ST expression is available directly.

IF A THEN
  OUT := AA+BB *CC ;
END_IF;
```

(1) If the transition condition (A) is on, CPT function executes.

(2) If input variable AA = 10, BB = 10, CC = 2, output variable OUT = 30

ARY_CMP_EQ	Equivalent comparison of the two Array Elements	
	Availability	XGI, XGR, XEC, XMC(U)
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: first array to compare IN1_INDX : starting point in 1st array for comparison IN2: second array to compare IN2_INDX : starting point in 2nd array for comparison LEN: number of elements to compare **Output** ENO: without an error, it is 1 OUT: if there is a same element, it is 1 P_INDX : index position that same array in the IN1 N : The number of same array elements

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ARRAY OF ANY: exclude STRING from ANY type.

■ **Function**

1. It Compare that with the same value as the other two receiving Array.
2. If LEN is a negative number, it compares two arrays between IN*_INDX (Array INDX) and "Array INDX – |LEN|."
3. If the size of P_INDX Array is less than LEN, the location information that beyond the size of P_INDX Array can be lost.

Function	Input array type	Description
ARY_CMP_EQ	BOOL	Compare that to the element with a value equal to each other in two BOOL Array.
ARY_CMP_EQ	BYTE	Compare that to the element with a value equal to each other in two BYTE Array.
ARY_CMP_EQ	WORD	Compare that to the element with a value equal to each other in two WORD Array.
ARY_CMP_EQ	DWORD	Compare that to the element with a value equal to each other in two DWORD Array.
ARY_CMP_EQ	LWORD	Compare that to the element with a value equal to each other in two

Function	Input array type	Description
		LWORD Array.
ARY_CMP_EQ	SINT	Compare that to the element with a value equal to each other in two SINT Array.
ARY_CMP_EQ	INT	Compare that to the element with a value equal to each other in two INT Array.
ARY_CMP_EQ	DINT	Compare that to the element with a value equal to each other in two DINT Array.
ARY_CMP_EQ	LINT	Compare that to the element with a value equal to each other in two LINT Array.
ARY_CMP_EQ	USINT	Compare that to the element with a value equal to each other in two USINT Array.
ARY_CMP_EQ	UINT	Compare that to the element with a value equal to each other in two UINT Array.
ARY_CMP_EQ	UDINT	Compare that to the element with a value equal to each other in two UDINT Array.
ARY_CMP_EQ	ULINT	Compare that to the element with a value equal to each other in two ULINT Array.
ARY_CMP_EQ	REAL	Compare that to the element with a value equal to each other in two REAL Array.
ARY_CMP_EQ	LREAL	Compare that to the element with a value equal to each other in two LREAL Array.
ARY_CMP_EQ	TIME	Compare that to the element with a value equal to each other in two TIME Array.
ARY_CMP_EQ	DATE	Compare that to the element with a value equal to each other in two DATE Array.
ARY_CMP_EQ	TOD	Compare that to the element with a value equal to each other in two TOD Array.
ARY_CMP_EQ	DT	Compare that to the element with a value equal to each other in two DT Array.

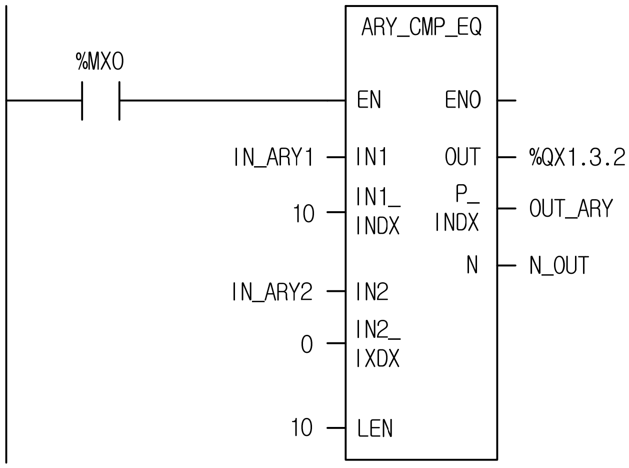
■ **Flag**

Flag	Description
_ERR	If it is designated beyond the array range, _ERR and _LER flags are set. ※ An error occurs when: IN1_INDX < 0 or IN1_INDX > max. number of IN1

	IN2_INDX < 0 or IN2_INDX > max. number of IN2 IN1_INDX + LEN ≥ max. number of IN1 IN2_INDX + LEN ≥ max. number of IN2

■ **Program Example**

1. LD



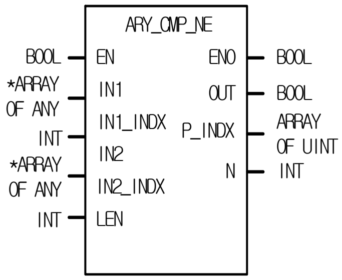
2. ST

%QX1.3.2 := ARY_CMP_EQ(EN:=%MX0, IN1:=IN_ARY1, IN1_INDX:=10, IN2:=IN_ARY2, IN2_INDX:=0, LEN:=10, P_INDX=>OUT_ARY, N=>N_OUT);

(1) If the input transition condition (%MX0) is on, ARY_CMP_EQ function executes.

(2) When IN_ARY1 is a WORD array with 1000 elements and IN_ARY2 is a WORD array with 100 elements, if there are same value as compared to each of 10 elements between the elements from 11th (IN_ARY1[10]) to 20th (IN_ARY1[19]) of IN_ARY1 and the elements from 1st (IN_ARY2[0]) to 10th (IN_ARY2[9]) of IN_ARY1, the output %Q1.3.2 is on and index value of IN_ARY1 is written in order, count of array elements that have same value output to N_OUT

ARY_CMP_NE	Not equal comparison of the two Array Elements	
	Availability	XGI, XGR, XEC, XMC(U)
	Flags	_ERR, _LER

Function	Description
	**Input** EN: executes the function in case of 1 IN1: first array to compare IN1_INDX : starting point in 1st array for comparison IN2: second array to compare IN2_INDX : starting point in 2nd array for comparison LEN: number of elements to compare **Output** ENO: without an error, it is 1 OUT: if there is a different element, it is 1 P_INDX : index position that not equal in the IN1 Array N : The number of array elements that not equal

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ARRAY OF ANY: exclude STRING from ANY type.

■ **Function**

4. It Compare that with the not equal value as the other two receiving Array.
5. If LEN is a negative number, it compares two arrays between IN*_INDX (Array INDX) and “Array INDX – |LEN|.”
6. If the size of P_INDX Array is less than LEN, the location information that beyond the size of P_INDX Array can be lost.

Function	Input array type	Description
ARY_CMP_NE	BOOL	Compare that to the element with a value equal to each other in two BOOL Array.
ARY_CMP_NE	BYTE	Compare that to the element with a value equal to each other in two BYTE Array.
ARY_CMP_NE	WORD	Compare that to the element with a value equal to each other in two WORD Array.
ARY_CMP_NE	DWORD	Compare that to the element with a value equal to each other in two DWORD Array.

Function	Input array type	Description
ARY_CMP_NE	LWORD	Compare that to the element with a value equal to each other in two LWORD Array.
ARY_CMP_NE	SINT	Compare that to the element with a value equal to each other in two SINT Array.
ARY_CMP_NE	INT	Compare that to the element with a value equal to each other in two INT Array.
ARY_CMP_NE	DINT	Compare that to the element with a value equal to each other in two DINT Array.
ARY_CMP_NE	LINT	Compare that to the element with a value equal to each other in two LINT Array.
ARY_CMP_NE	USINT	Compare that to the element with a value equal to each other in two USINT Array.
ARY_CMP_NE	UINT	Compare that to the element with a value equal to each other in two UINT Array.
ARY_CMP_NE	UDINT	Compare that to the element with a value equal to each other in two UDINT Array.
ARY_CMP_NE	ULINT	Compare that to the element with a value equal to each other in two ULINT Array.
ARY_CMP_NE	REAL	Compare that to the element with a value equal to each other in two REAL Array.
ARY_CMP_NE	LREAL	Compare that to the element with a value equal to each other in two LREAL Array.
ARY_CMP_NE	TIME	Compare that to the element with a value equal to each other in two TIME Array.
ARY_CMP_NE	DATE	Compare that to the element with a value equal to each other in two DATE Array.
ARY_CMP_NE	TOD	Compare that to the element with a value equal to each other in two TOD Array.
ARY_CMP_NE	DT	Compare that to the element with a value equal to each other in two DT Array.

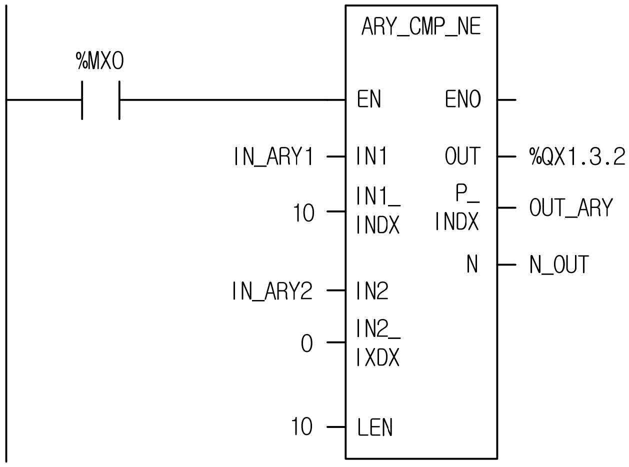
■ **Flag**

Flag	Description
_ERR	If it is designated beyond the array range, _ERR and _LER flags are set. ※ An error occurs when:

	IN1_INDX < 0 or IN1_INDX > max. number of IN1 IN2_INDX < 0 or IN2_INDX > max. number of IN2 IN1_INDX + LEN ≥ max. number of IN1 IN2_INDX + LEN ≥ max. number of IN2

■ **Program Example**

1. LD



2. ST

%QX1.3.2 := ARY_CMP_NE(EN:=%MX0, IN1:=IN_ARY1, IN1_INDX:=10, IN2:=IN_ARY2, IN2_INDX:=0, LEN:=10, P_INDX=>OUT_ARY, N=>N_OUT);

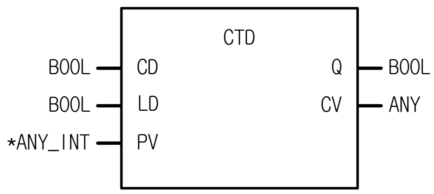
(1) If the input transition condition (%MX0) is on, ARY_CMP_NE function executes.

(2) When IN_ARY1 is a WORD array with 1000 elements and IN_ARY2 is a WORD array with 100 elements, if there are not equal value as compared to each of 10 elements between the elements from 11th (IN_ARY1[10]) to 20th (IN_ARY1[19]) of IN_ARY1 and the elements from 1st (IN_ARY2[0]) to 10th (IN_ARY2[9]) of IN_ARY1, the output %Q1.3.2 is on and index value of IN_ARY1 is written in order, count of array elements that have not equal value output to N_OUT

Chapter 9. Basic Function Blocks

1. This chapter describes basic function block library.
2. Before using basic function block, it is recommended to understand 3.4.2 Function Block and apply function block library to a program, it is facilitative to write a program.

CTD	Down Counter (function block)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** CD: down counter pulse input LD: loads a preset value PV: preset value **Output** Q: down counter output CV: current value

Any type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	PV							○	○	○		○	○	○							
	CV							○	○	○		○	○	○							

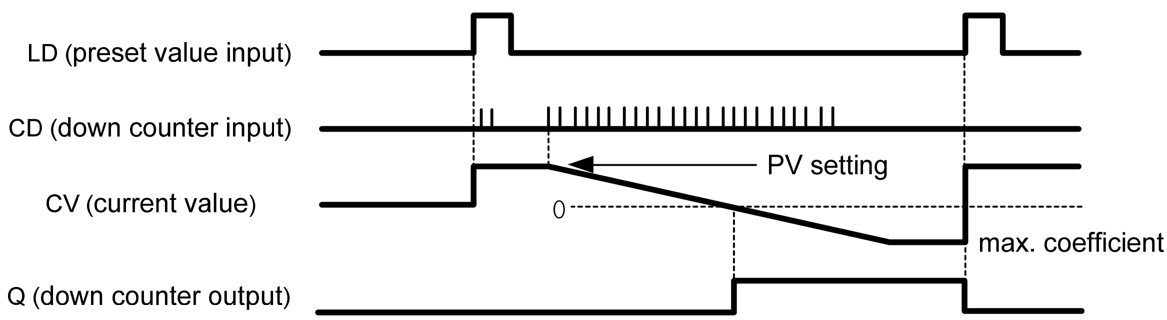
*ANY_INT: exclude SINT and USINT from ANY_INT type.

■ **Function**

1. Down counter function block CTD decreases the current value (CV) by 1 with every rising pulse input.
2. CV decreases only when CV is more than the minimum value of INT (-32768); after reaching it, CV does not change its value.
3. When LD is 1, PV is loaded into CV (CV=PV).
4. Output Q is 1 when CV is 0 or a negative number.

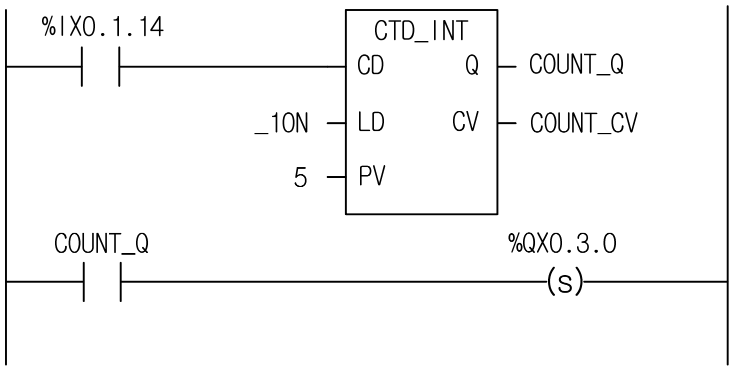
Function Block	PV	Description
CTD_INT	INT	Decrease as much as the min INT(-32,768).
CTD_DINT	DINT	Decrease as much as the min DINT(-2,147,483,648).
CTD_LINT	LINT	Decrease as much as the min LINT(-9,223,372,036,854,775,808).
CTD_UINT	UINT	Decrease as much as the min UINT(0).
CTD_UDINT	UDINT	Decrease as much as the min UDINT(0).
CTD_ULINT	ULINT	Decrease as much as the min ULINT(0).

■ **Time Chart**



■ **Program Example**

1. LD



2. ST

INST_CTD_INT(CD:=%IX0.1.14, LD:=_10N, PV:=5, Q=>COUNT_Q, CV=>COUNT_CV);

%QX0.3.0 := COUNT_Q

This is the program that sets the output contact (%QX0.3.0) when the down counter pulse input enters the input contact (%IX0.1.14) five times.

(1) Register the name of CTD function block (COUNT_D).

(2) Make the input contact (%IX0.1.14) attached to CD.

(3) Make the flag _10N (1 scan On contact) that loads PV into CV.

(4) Set the PV value as 5 in range of INT ((-32,768~32,767).

(5) Set the CV value as the random output variable (COUNT_CV).

(6) Set the Q value as the random output variable (COUNT_Q).

(7) Compile and write your program to the PLC after completing the program.

(8) After writing, change the PLC mode (Stop -> Run).

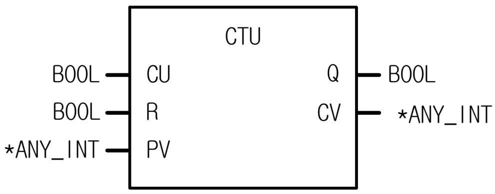
(9) If program runs, PV 5 will be loaded into CV (Count_CV).

(10) The current value CV (COUNT_CV) decreases by 1 when the pulse input enters the input contact (%IX0.1.14).

(11) When the down counter pulse input enters the input contact (%IX0.1.14) five times, CV (COUNT_CV) will be 0 and Q (COUNT_CV) will be 1.

(12) If Q (COUNT_Q) is 1, the output contact (%Q0.3.0) will be set.

CTU	Up Counter (function block)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** CU: up counter pulse input R: reset input PV: loads a preset value **Output** Q: increase counter output CV: current value

Any type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	PV							○	○	○		○	○	○							
	CV							○	○	○		○	○	○							

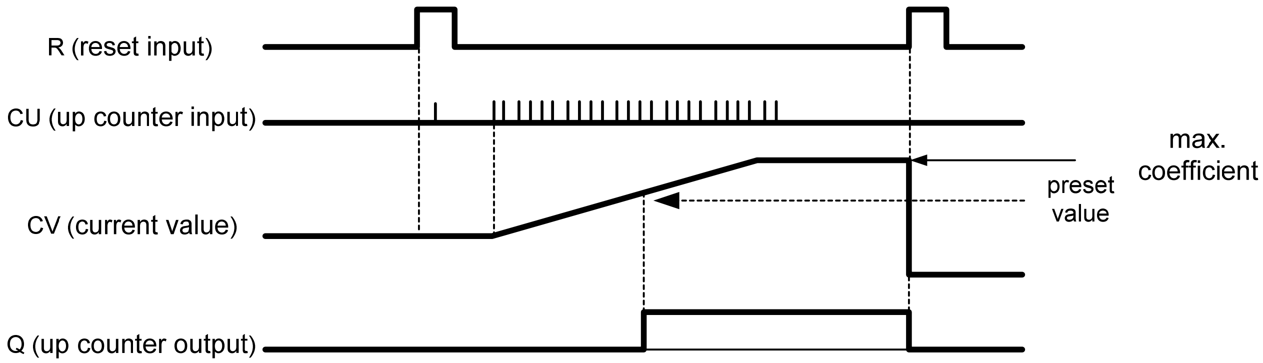
*ANY_INT: exclude SINT and USINT from ANY_INT type.

■ **Function**

1. Up counter function block CTU increases the current value (CV) by 1 with every rising pulse input.
2. CV increases only when CV is less than the maximum value of INT (32767); after reaching it, CV does not change its value.
3. When the reset input (R) is 1, CV is cleared (0).
4. Output Q is 1 when CV is equal to or more than PV.
5. PV value reloads the preset value and operate it when CTU function block executes.

Function Block	PV	Description
CTU_INT	INT	Increase as much as the max INT (32767).
CTU_DINT	DINT	Increase as much as the max DINT (2147483647).
CTU_LINT	LINT	Increase as much as the max LINT (9223372036854775807).
CTU_UINT	UINT	Increase as much as the max UINT (0).
CTU_UDINT	UDINT	Increase as much as the max UDINT (0).
CTU_ULINT	ULINT	Increase as much as the max ULINT (0).

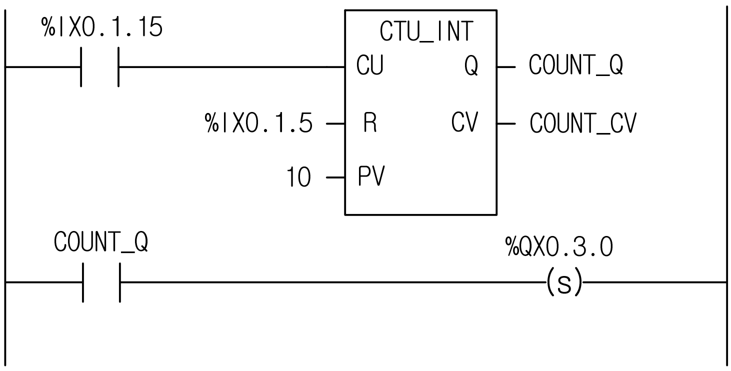
■ **Time Chart**



■ **Program Example**

1. This is the program that sets the output contact (%QX0.3.0) when the increase counter pulse input enters the input contact (%IX0.1.15) ten times

1. LD



2. ST

INST_CTU_INT(CU:=%IX0.1.15, R:=%IX0.1.5, PV:=10, Q=>COUNT_Q, CV=>COUNT_CV);

%QX0.3.0 := COUNT_Q;

(1) Register the name of CTU function block (COUNT_U).

(2) Make the input contact %I0.1.15 attach to CU.

(3) Set the PV value as 10.

(4) Assign input contact %IX0.1.5 to the reset input R.

(5) Set the CV value as the random output variable (COUNT_CV).

(6) Set the Q value as the random output variable (COUNT_Q).

(7) Compile and write your program to the PLC after completing the program.

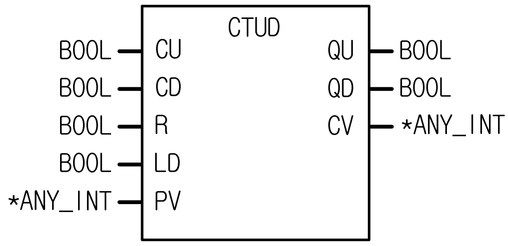
(8) After writing, change the PLC mode (Stop → Run).

(9) The current value CV (COUNT_CV) increases by 1 when the pulse input enters the input contact (%IX0.1.15).

(10) When the up counter pulse input enters the input contact (%IX0.1.15) ten times, CV (COUNT_CV) is 10 and Q (COUNT_Q) is 1.

(11) If Q (COUNT_Q) is 1, the output contact (%QX0.3.0) is set.

CTUD	Up/Down Counter (function block)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** CU: up counter pulse input CD: down counter pulse input R: reset LD: loads a preset value PV: preset value **Output** QU: up counter output QD: down counter output CV: current value

Any type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	PV							○	○	○		○	○	○							
	CV							○	○	○		○	○	○							

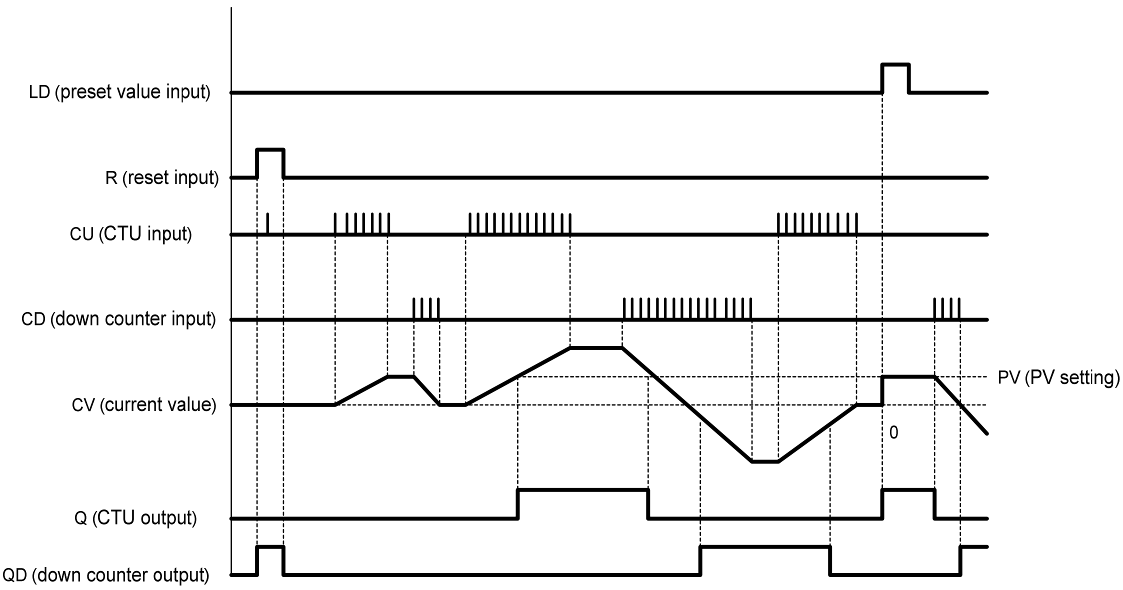
*ANY_INT: excluding SINT and USINT from ANY_INT types

■ **Function**

1. Up/Down counter function block CTUD increases the current value (CV) by 1 with every rising up-counter pulse input (CU) and decreases CV by 1 with every rising down-counter pulse input (CD).
2. Note that CV is between -32768 and 32767 (INT).
3. When LD is 1, PV is loaded into CV (CV=PV).
4. When the reset input R is 1, CV is cleared (0).
5. When CV reaches PV, the output QU is 1; when CV is 0 or a negative integer, the output QD is 1.
6. The operation for each input signal executes in order of R > LD > CU > CD. Note that if the input signals are fed to the input (CU, CD, R, and LD) of CTUD at the same time, the operation of CTU follows the above priority.

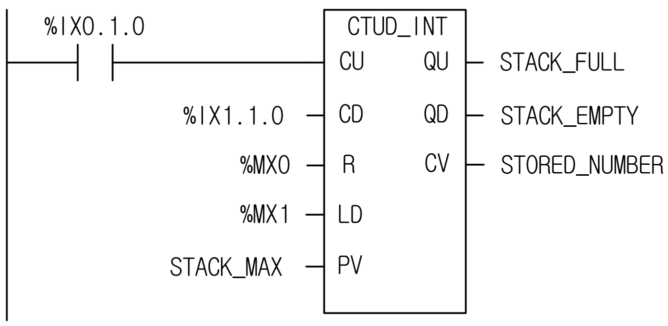
Function Block	PV	Description
CTUD_INT	INT	Increase/decrease as much as INT(-32768 ~ 32767)
CTUD_DINT	DINT	Increase/decrease as much as DINT(0 ~ 2^{31}-1)
CTUD_LINT	LINT	Increase/decrease as much as LINT(0 ~ 2^{63}-1)
CTUD_UINT	UINT	Increase/decrease as much as UINT(0 ~ 65535)
CTUD_UDINT	UDINT	Increase/decrease as much as UDINT(0 ～ 2^{32}-1)
CTUD_ULINT	ULINT	Increase/decrease as much as ULINT(0 ～ 2^{63}-1)

■ **Time Chart**



■ **Program Example**

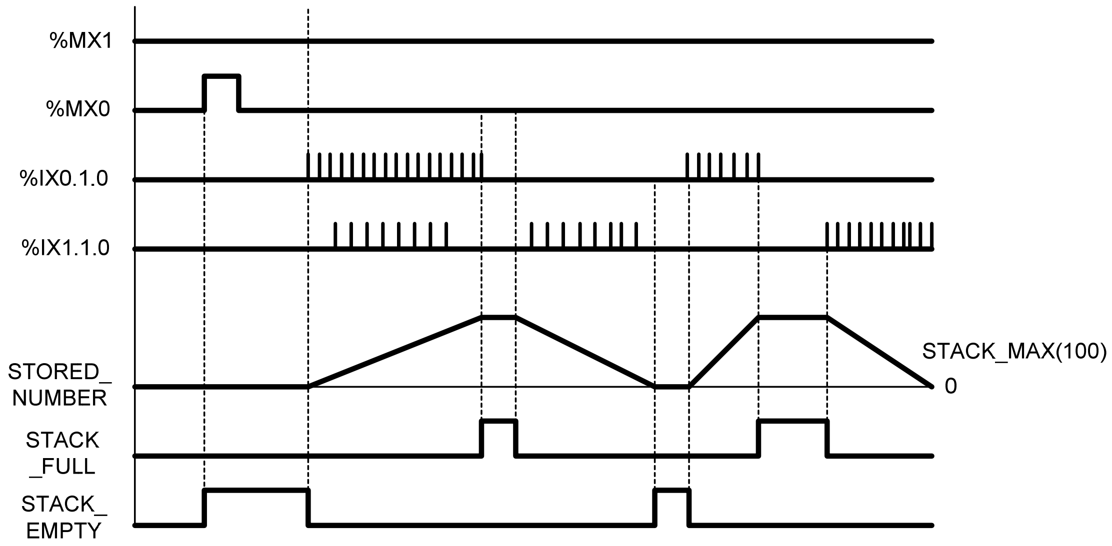
1. LD



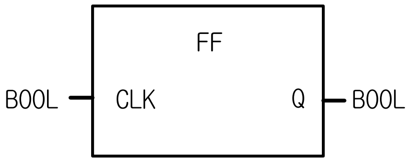
2. ST

INST_CTUD_INT(CU:=%IX0.1.0, CD:=%IX1.1.0, R:=%MX0, LD:=%MX1, PV:=STACK_MAX, QU=>STACK_FULL, QD=>STACK_EMPTY, CV=>STORED_NUMBER);

Conditions are: the temporary loading part STACK_MAX is 100; IN is 1 with every material-input signal while OUT is 1 with every material-output signal. If the material input process is faster than the material-output one and every material is loaded so that the STACK_MAX is equal to or more than 100, then QU is 1 (STACK_FULL = 1); if there's no material left in the loading part, QD is 1 (STACK_EMPTY = 1). At the STORED_NUMBER, the number of remaining material in the loading part is shown.


%MX1
%MX0
%IX0.1.0
%IX1.1.0
STORED_
NUMBER
STACK_MAX(100)
0
STACK
_FULL
STACK_
EMPTY

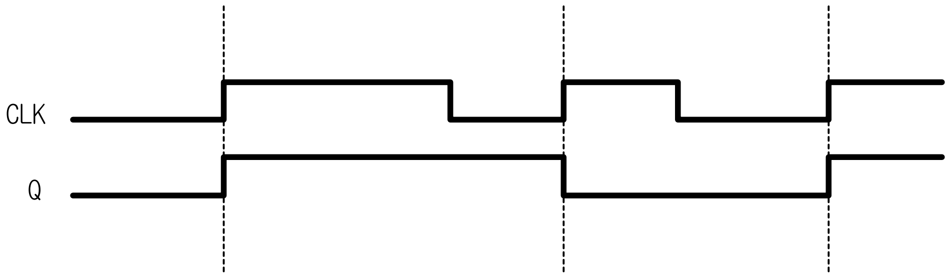
FF	Reverse output bit	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** CLK : input signal **Output** Q : reverse output by instruction

■ **Function**

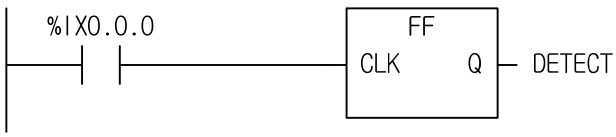
FF reverses output Q as the input status connected to CLK is changed from 0 to 1.

■ **Time Chart**



■ **Program Example**

1. LD

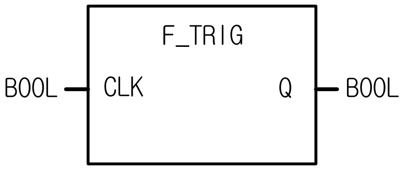


2. ST

INST_FF(CLK:=%IX0.0.0, Q=>DETECT);

(1) By watching the status of input variable, %IX0.0.0, when the input is changed from 0 to 1, the DETECT is reversed.

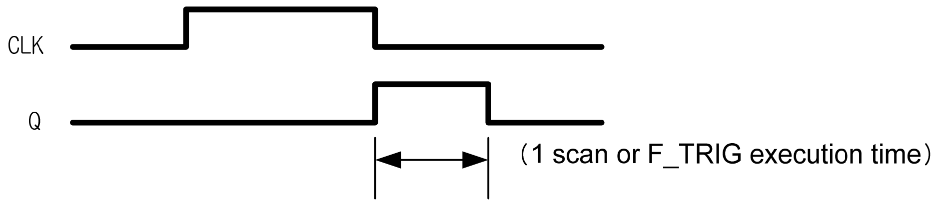
F_TRIG	Falling Edge Detection (function block)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** CLK: input signal **Output** Q: falling edge detection result

■ **Function**

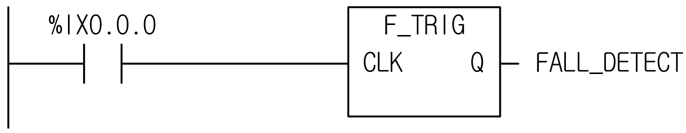
1. The output Q of function block F_TRIG is 1 with the falling pulse input to CLK. And 1 scan later, without further falling pulse input, the output Q is 0 ever after.

■ **Time Chart**



■ **Program Example**

1. LD

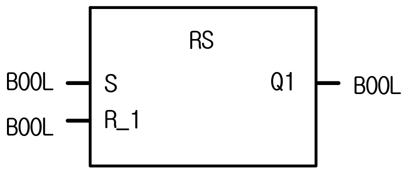


2. ST

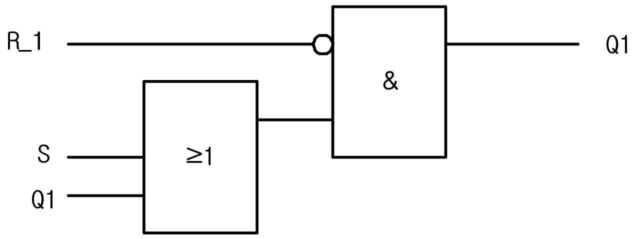
```
INST_F_TRIG(CLK:=%IX0.0.0, Q=>FALL_DETECT);
```

(1) If the input variable (%IX0.0.0) changes from 1 to 0, while detecting its state, the output variable FALL_DETECT is 1. And 1 scan later, the output variable FALL_DETECT is 0.

RS	Reset Priority Bistable (function block)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

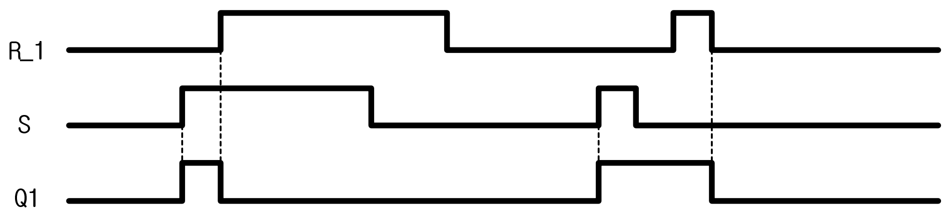
Function Block	Description
	**Input** R_1: Reset condition S: Set condition **Output** Q1: operation result

■ **Function**



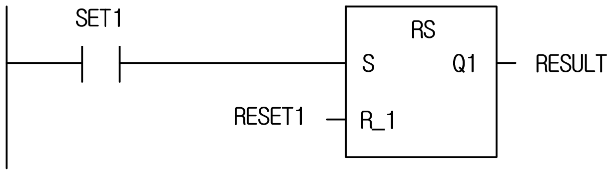
If R1 is 1, output Q1 is 0 regardless of the state of S. The output variable Q1 is 1 when it maintains the previous state, R1 is 0, and S is 1, it is 1. The initial state of Q1 is 0.

■ **Time Chart**



■ **Program Example**

1. LD



2. ST

INST_RS(S:=SET1, R_1:=RESET1, Q=>RESULT);

It outputs the operation results with RESET1 as Reset condition and SET1 as Set condition to RESULT.

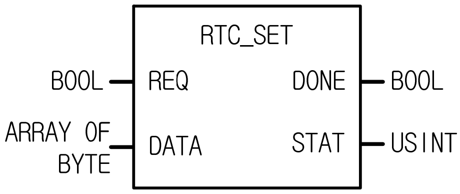
Replace the operation conditions; as the above time chart, R_1 to RESET1, S to SET1 and Q1 to RESULT.

(1) If SET1 declared as input variable is on, output variable RESULT is 1.

(2) If RESET1 declared as output variable is on, output variable declared as RESULT is 0.

(3) If SET1 and RESET1 declared as input variables are on, the output variable RESULT is 0.

RTC_SET	Writes Time data	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR, _LER

Function Block	Description
	**Input** REQ: executes the function with rising pulse input DATA: TIME data to input **Output** DONE: without an error, it is 1 STAT: If an error occurs, an error code is written

■ **Function**

1. It writes RTC data to Clock Device with a rising pulse input.

Variable	Content	Example	Variable	Content	Example
DATA[0]	Year	16#01	DATA[4]	Minute	16#30
DATA[1]	Month	16#03	DATA[5]	Second	16#45
DATA[2]	Dates	16#15	DATA[6]	No check	-
DATA[3]	Hours	16#18	DATA[7]	Year	16#20

* The above example is “2001-03-15 18:30:45, Thursday”.

* Day of the week data is not separately entered. The day of the week will be automatically set.

2. The above DATA variables are declared as array Byte variables and set as BCD data.

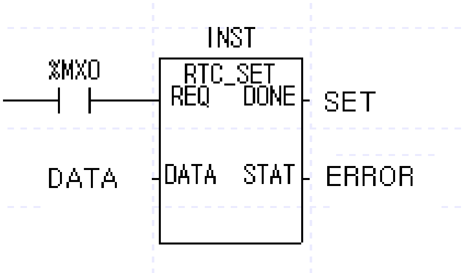
■ **Flag**

Flag	Description
_ERR	If CPU does not support RTC function or RTC data is out of range, the output is 0 and the error code is written at STAT.

Error code	Description
00	No error
02	Wrong RTC data. Example: 14 (Months) 32 (Dates) 25 (Hours) * Modify RTC data.

■ **Program Example**

1. LD



2. ST

INST_RTC_SET(REQ:=%MX0, DATA:=DATA, DONE=>SET, STAT=>ERROR);

Its RTC data is Dec 5, 2006. 10:39:45, Tuesday.

(1) When SET_SW is on, RTC_SET function block renews or modifies the SET_data (RTC data).

(2) Variable setting is shown as below.

Variable	Content	Example	Variable	Content	Example
DATA[0]	Year	16#06	DATA[4]	Minute	16#39
DATA[1]	Month	16#12	DATA[5]	Second	16#45
DATA[2]	Date	16#05	DATA[6]	No check	-
DATA[3]	Hour	16#10	DATA[7]	Year	16#20

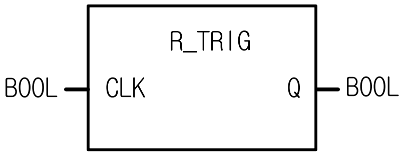
(3) In addition to the method set by allowing initial value to DATA variable, it may be set by saving each preset value to DATA[] variable, using function MOVE.

(4) Use the following flags to read RTC data.

e.g. 1998-12-22 19:37:46, Tuesday

Flag	Type	Content	Description	Data
_RTC_TOD	TOD	Current time	Current time of RTC	TOD#19:37:46
_RTC_WEEK	UINT	Current day	Current day of RTC *(0: Sun, 1: Mon, 2: Tue, 3: Wed, 4: Thu, 5: Fri, 6: Sat)	2
_RTC_DATE	DATE	Current date	Current date of RTC (1984-01-01 ~ 2063-06-06)	D#1998-12-22
_HUND_WK	WORD	Hundred year/day	Discriminated by BYTE	16#1902
_TIME_DAY	WORD	Time/date		16#1922
_MON_YEAR	WORD	Month/year		16#1298
_SEC_MIN	WORD	Second/minute		16#4637

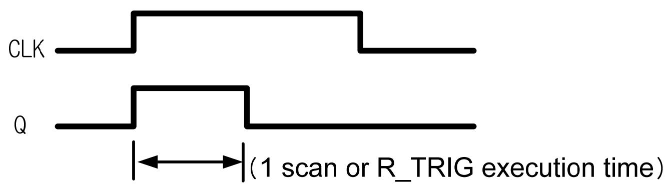
R_TRIG	Rising Edge Detection (function block)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** CLK: input signal **Output** Q: rising edge detection result

■ **Function**

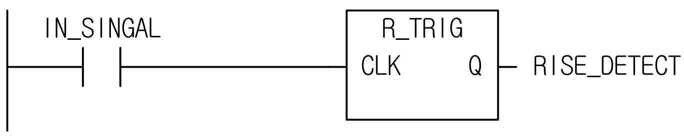
The output Q of function block R_TRIG is 1 with the rising pulse input to CLK. And 1 scan later, without further rising pulse input, the output Q is 0.

■ **Time Chart**



■ **Program Example**

1. LD



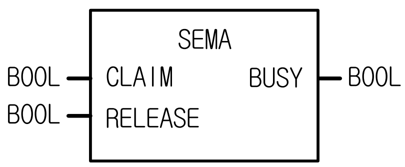
2. ST

INST_R_TRIG(CLK:=IN_SIGNAL, Q=>RISE_DETECT);

If the input variable IN_SIGNAL changes from 0 to 1, while detecting its state, the output variable RISE_DETECT is 1.

And 1 scan later, the output variable RISE_DETECT is 0.

SEMA	Semaphore (System resource allocation)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** CLAIM: signal to claim a resource monopoly RELEASE: release signal **Output** BUSY: waiting signal not to obtain the claimed resource

■ **Function**

This function block is used to get an exclusive control right for system resources.

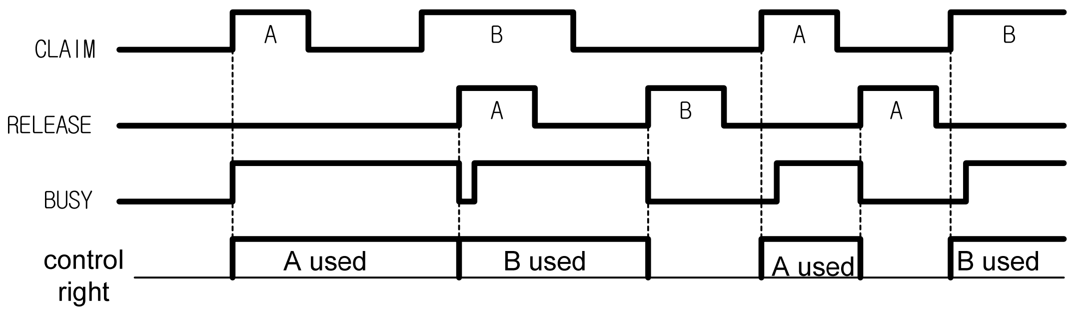
BUSY that is using the resource in other program is 1 when SEMA function executes (CLAIM = 1 or 0, RELEASE = 0). If you want to obtain the resource control right, wait until BUSY is 0 after executing SEMA function block (CLAIM = 1, RELEASE = 0). When BUSY is 0, it controls the associate resource and after completing the control, it transfers the control right executing SEMA function block once again with CLAIM = 0 and RELEASE = 1. (At this time, only the program that has the control right can execute SEMA function block with CLAIM = 0 and RELEASE = 1)

1. The instance of SEMA must be declared as "GLOBAL" so that its access is available in the programs requiring the resource.
2. Each program to claim the same resource must be designated as the same priority.
3. Internal execution structure of SEMA function block.

```
VAR    X : BOOL : = 0 ;    END_VAR
BUSY : = X ;
IF    CLAIM    THEN    X : = 1 ;
ELSIF    RELEASE    THEN    BUSY : = 0; X : = 0 ;
END_IF
```

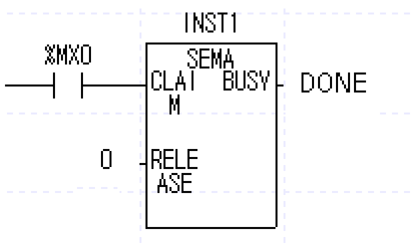
■ **Time Chart**

The access right to control the same resource is transferred between the program block A and the program block B.



■ **Program Example**

1. LD

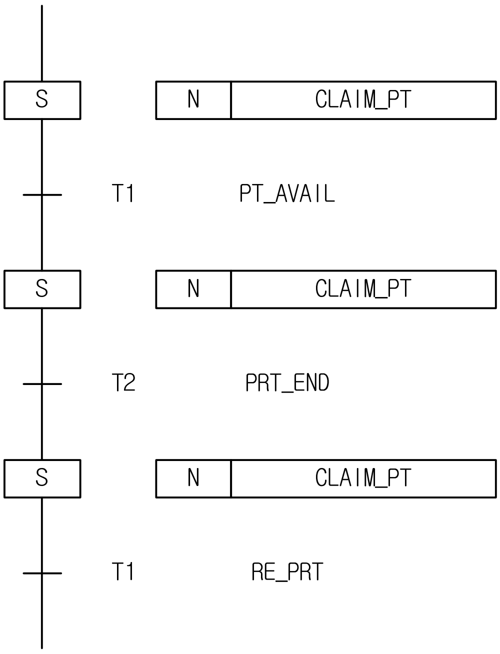


2. ST

INST_SEMA(CLAIM:=%MX0, RELEASE:=0, BUSY=>DONE);

When you want to produce a printer output in different program blocks with the printer attached to the PLC system, you can easily control it by declaring the instance 'PRINTER' as a 'GLOBAL' and using SEMA function block named as 'PRINTER' in each program. If you execute SEMA function block (PRINTER), when START is 1 and END is 0, and claim the right to control the printer, while the printer is used in other program block, BUSY is 1 then outputs 1 to OT_AVAIL. If the printer is not used in other program block, BUSY is 0, which means you can start the program to produce the printer output with it. After completing the print control, execute SEMA with START = 0 and END = 1 so that other program can get the right to control it.

```
VAR_EXTERNAL
PRINTER : SEMA
END_VAR
```



S	CLAIM_PT;Claim the printer control right

```
CAL  SEMA      PRINTER
      CLAIM:=    1
      RELEASE:=  0
```

T	PT_AVAIL;Print control right check

```
LDN    PRINTER.BUSY
  ST     TRANS
```

S	PRINTING;Printer output

Printer control program
If print is completed, PRINT_CODE:=1

T	PRT_END;Print completion check

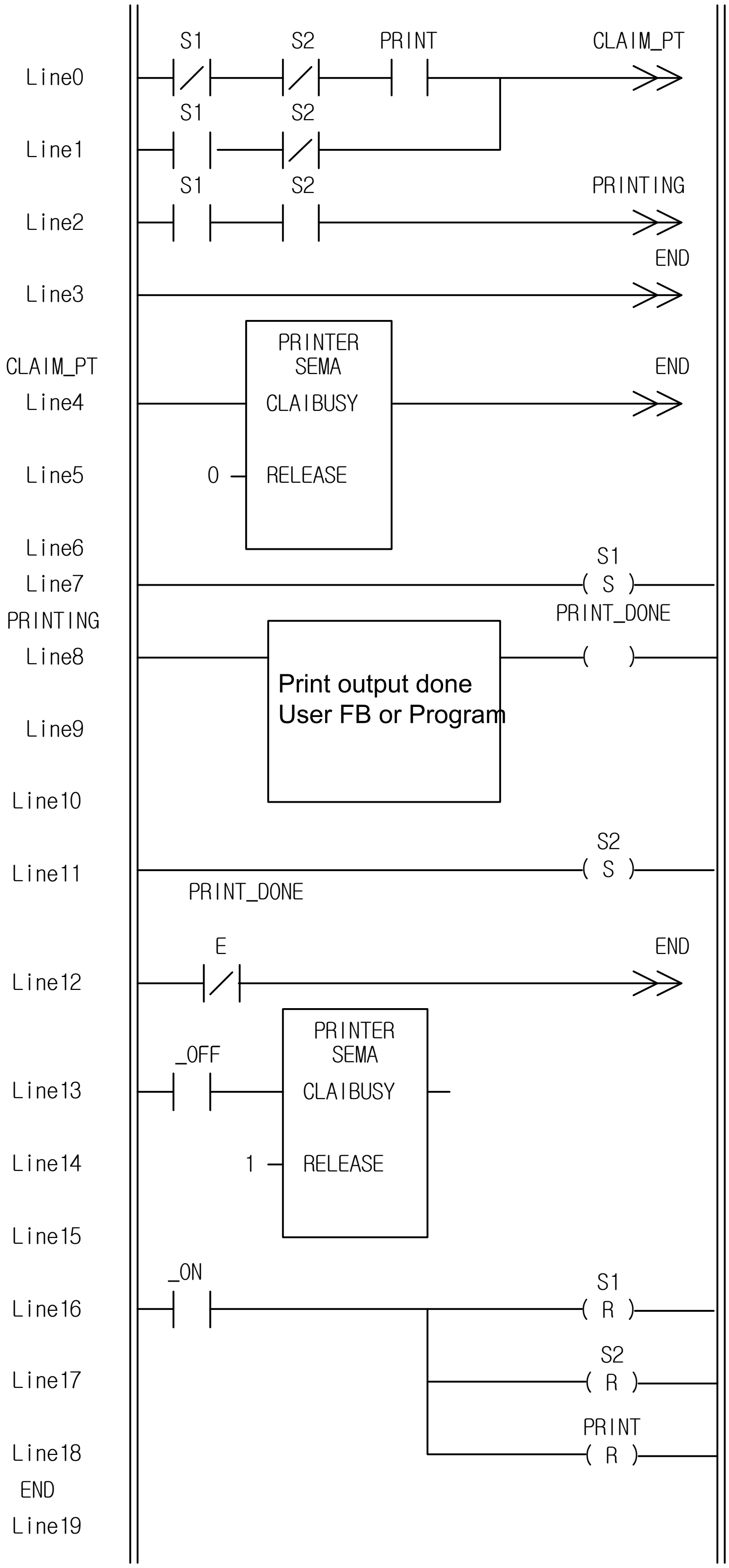
```
LD     PRINTER_DONE
  ST     TRANS
```

S	REL_PRT;Transfer printer control

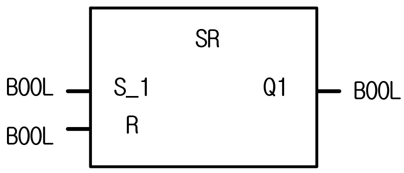
```
CAL    SEMA      PRINTER
       CLAIM:=    0
       RELEASE:=  1
```

T	RE_PRT;Printer request again

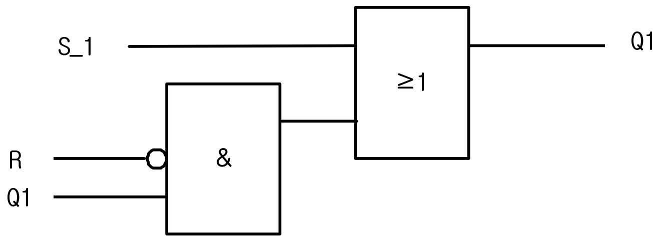
```
LD     PRT_REQ
ST     TRANS
```


Line0
S1
S2
PRINT
CLAIM_PT
Line1
S1
S2
Line2
S1
S2
PRINTING
Line3
END
CLAIM_PT
Line4
PRINTER SEMA
CLAIBUSY
END
Line5
0
RELEASE
Line6
Line7
S1
S
PRINTING
Line8
PRINT_DONE
Print output done
User FB or Program
Line9
Line10
Line11
S2
S
PRINT_DONE
Line12
E
END
Line13
_OFF
PRINTER SEMA
CLAIBUSY
Line14
1
RELEASE
Line15
Line16
_ON
S1
R
Line17
S2
R
Line18
PRINT
R
END
Line19

SR	Set Priority Bistable (function block)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

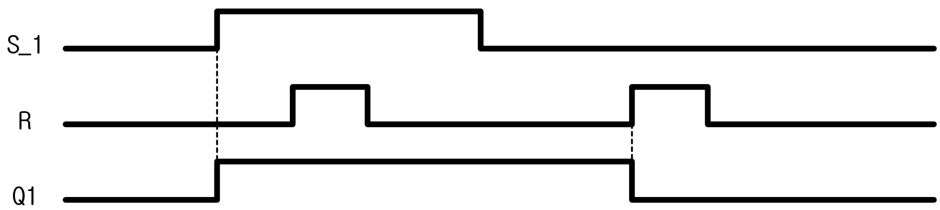
Function Block	Description
	**Input** S1: set condition R: reset condition **Output** Q1: operation result

■ **Function**



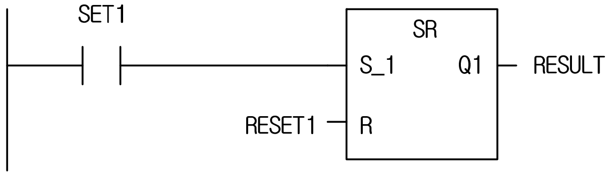
1. If S1 is 1, output Q1 is 1 regardless of the state of R.
2. The output variable Q1 is 0 and it maintains the previous state when S1 is 0, and R is 1.
3. The initial state of Q1 is 0.

■ **Time Chart**



■ **Program Example**

1. LD



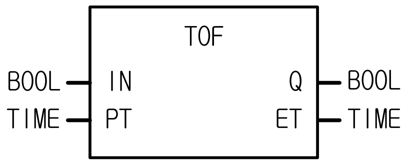
2. ST

```
INST_SR(S_1:=SET1, R:=RESET1, Q=>RESULT);
```

(1) If input variable SET1 becomes on, output variable RESULT is 1.

(2) The output variable RESULT becomes 0 when input variable SET1 becomes off and RESET on.

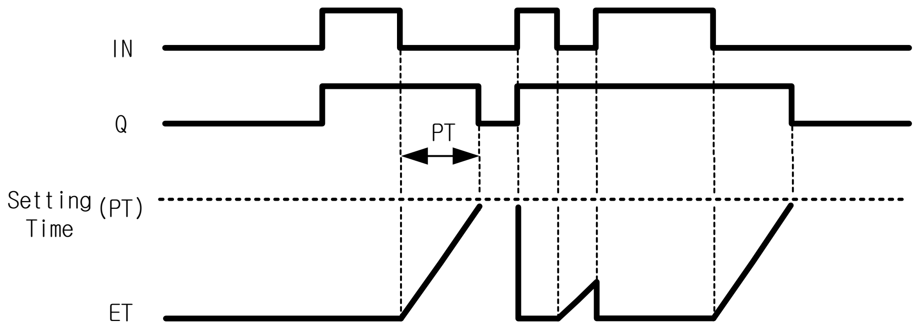
TOF	Off Delay Timer (function block)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** IN: timer operation condition PT: preset time **Output** Q: timer output ET: elapsed time

■ **Function**

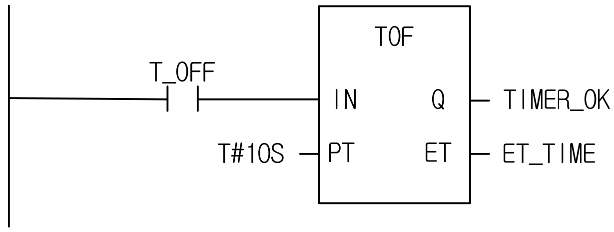
1. If IN is 1, Q is 1. And after IN becomes 0 and the preset time (PT) of TOF passes, Q becomes 0.
2. After IN becomes 0, the elapsed time (ET) is shown.
3. If IN becomes 1 before ET reaches the preset time, ET is 0 again.

■ **Time Chart**



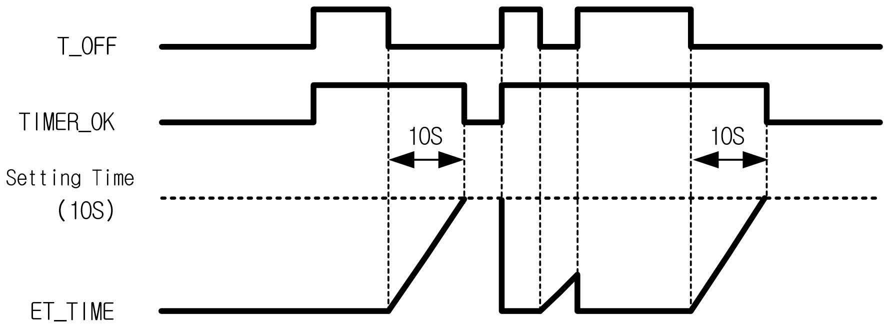
■ **Program Example**

1. LD



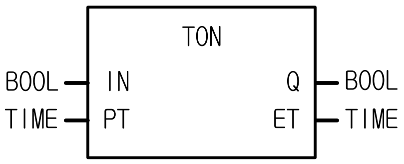
2. ST

INST_TOF(IN:=T_OFF, PT:=T#10S, Q=>TIMER_OK, ET=>ET_TIME);



(1) Output variable TIMER_OK is 1 when input variable T_OFF becomes 1. TIMER_OK is 0 only if 10 seconds passes after T_OFF becomes 0.
(2) If T_OFF becomes 1 again in 10 seconds after it turned off, TOF is initialized (TIMER_OK is 1).
(3) After T_OFF becomes 0, the elapsed time (ET_TIME) is measured and shown.

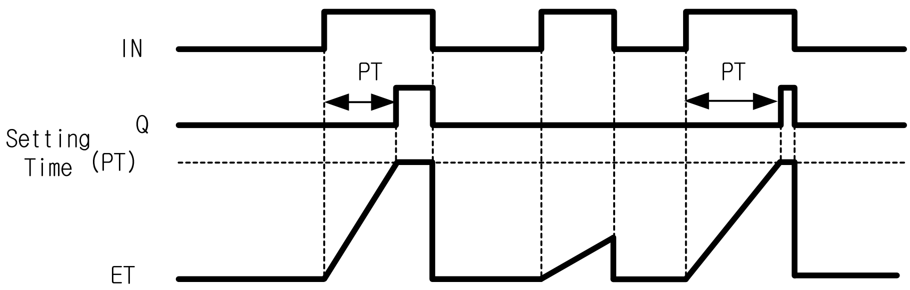
TON	On Delay Timer (function block)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** IN: timer operation condition PT: preset time **Output** Q: timer output ET: elapsed Time

■ **Function**

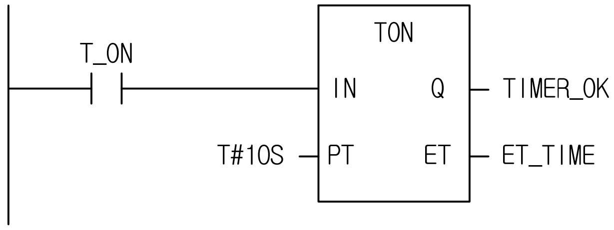
1. Elapsed time (ET) is measured and shown after IN becomes 1.
2. When IN becomes 0 before ET reaches the preset time, ET is 0.
3. If IN becomes 0 after Q is 1, Q is 0.

■ **Time Chart**



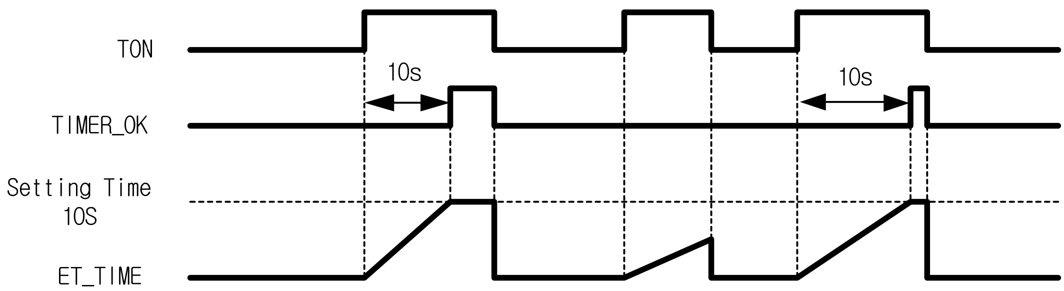
■ **Program Example**

1. LD



2. ST

INST_TON(IN:=T_ON, PT:=T#10S, Q=>TIMER_OK, ET=>ET_TIME);



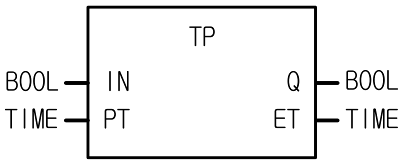
(1) The output TIMER_OK = 1 ten seconds later after the input T_ON is asserted (T_ON = 1).

(2) After input variable T_ON is 1, the elapsed time is output to output variable, ET_TIME.

(3) When T_ON = 0 before ET_TIME reaches the preset time (10s), ET_TIME is 0.

(4) If T_ON = 0 after TIMER_OK = 1, then TIMER_OK = 0 and ET_TIME = 0.

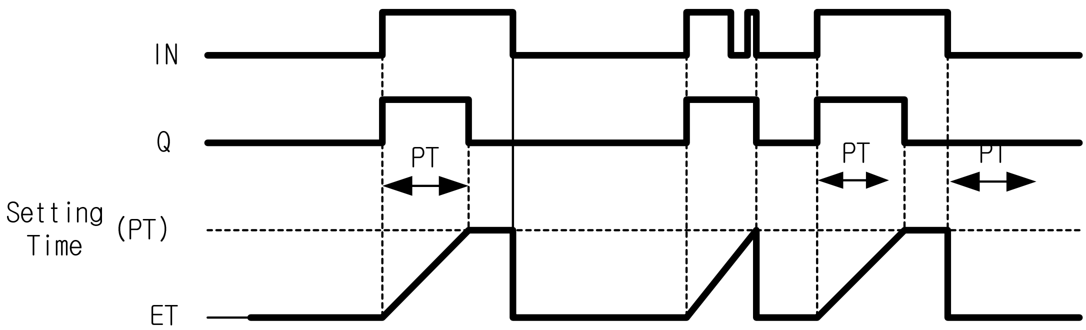
TP	**Pulse timer (function block)**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** IN: timer operation condition PT: preset time **Output** Q: timer output ET: elapsed Time

■ **Function**

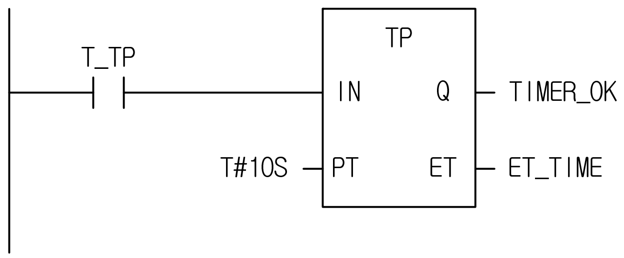
1. If IN = 1, Q is 1 only during the preset time PT; if ET reaches PT, Q is 0.
2. If IN = 1, elapsed time ET starts to be measured and maintains its value after when it reaches PT; if IN = 0 after ET reaches PT, ET = 0.
3. The state of IN doesn't matter while ET is measured (increased).

■ **Time Chart**



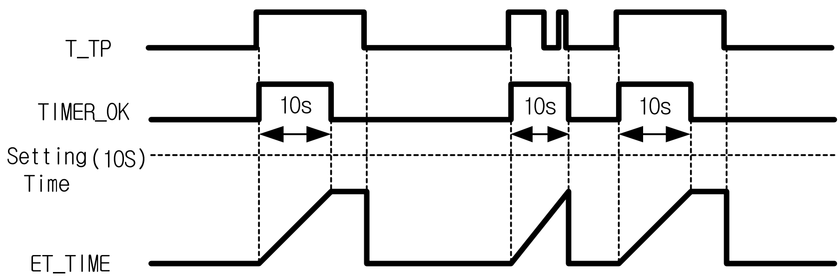
■ **Program Example**

1. LD



2. ST

INST_TP(IN:=T_TP, PT:=T#10S, Q=>TIMER_OK, ET=>ET_TIME);



(1) TIMER_OK is 1 during 10 seconds after input T_TP was asserted (T_TP = 1). While ET_TIME increases during 10 seconds, the state of input T_TP doesn't affect TIMER_OK.

(2) ET_TIME increases when it reaches T#10S and then it becomes 0 when T_TP = 0.

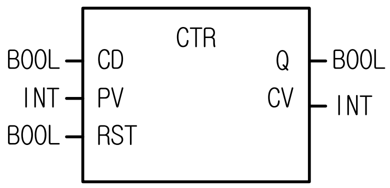
☆ **Note**

TP function block keeps operating until its operation is complete even if the contact is changed from on to off. In case of a variable using array index, array index error occurs only when the contact is on. Therefore, TP function block does not produce any array index error as long as the contact is off although function block is operating.

Chapter 10. Application Function Blocks

This chapter describes the basic function block library mentioned in the previous chapter and other application function block library.

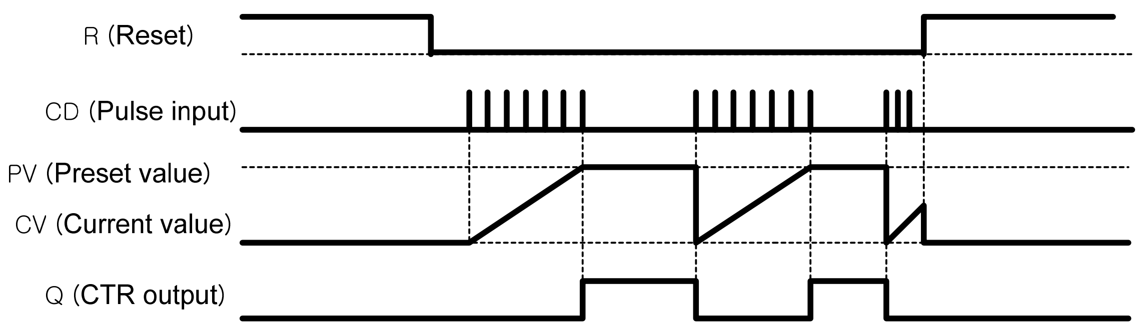
CTR	Ring Counter	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** CD: pulse input of Ring Counter PV: preset value RST: reset **Output** Q: Ring Counter output CV: current value

■ **Function**

1. CTR function block (Ring Counter) functions: current value (CV) increases with the rising pulse input (CD) and if, after CV reaches PV, CD becomes 1, then CV is 1.
2. When CV reaches PV, output Q is 1.
3. If CV is less than PV or reset input (RST) is 1, output Q is 0.

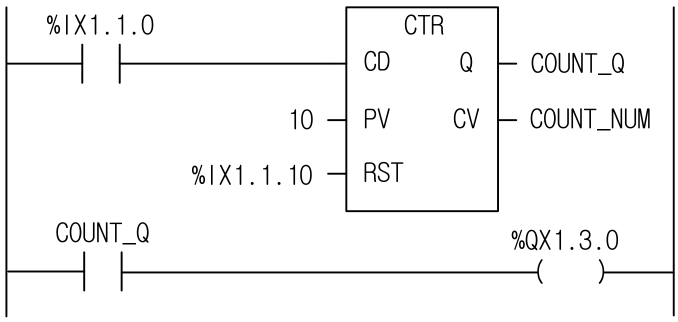
■ **Time Chart**



■ **Program Example**

Output %QX1.3.1 is on with 10-time rising pulse input of %IX1.1.0 is depicted as follows:

1. LD

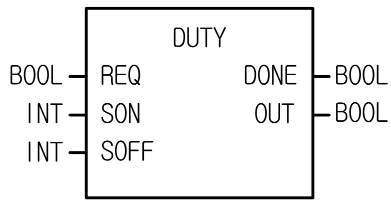


2. ST

INST_CTR(CD:=%IX1.1.0, PV:=10, RST:=%IX1.1.10, Q=>COUNT_Q, CV=>COUNT_NUM);
%QX1.3.0 := COUNT_Q;

(1) Define CTR function block as INS_CTR.
(2) Set %IX1.1.0 to the input contact of CD referring to the above.
(3) Set 10 to PV.
(4) Set %IX1.1.10 to RST resetting CV.
(5) Set random variable COUNT_NUM to CV
(6) Set random output variable COUNT_Q to Q.
(7) After a program is complete, compile and write it to PLC.
(8) When 'Write' is complete, do 'Mode Change' (Stop → Run).
(9) CV (COUNT_NUM) increases by 1 in number with the rising input pulse of %IX1.1.0.
(10) With 10-time rising input pulse of input contact, CV is 10 which is the same as PV and output variable COUNT_Q is 1.
(11) If Q (COUNT_Q) is 1, output contact %QX1.3.0 is on
(12) If the rising input pulse is loaded into input contact %IX1.1.0, then Q (COUNT_Q) is 0 and output contact %QX1.3.0 is off.

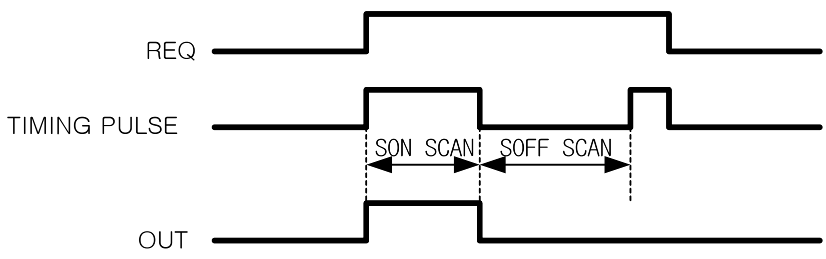
DUTY	Scan setting On/Off	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block SON: scan number to turn on SOFF: scan number to turn off **Output** DONE: it is 1 when REQ is on and both input variables are not less than 0. OUT: output is 1 during on scan time

■ **Function**

1. DUTY function block produces a pulse which is on during the SON scan time and off during the SOFF scan time while REQ is on.
2. If SON = 0, OUT is always off.
3. If SON > 0 and SOFF = 0, OUT is always on.
4. If REQ is off, OUT is off.
5. If SON < 0 or SOFF < 0, then DONE is off and OUT is 0.

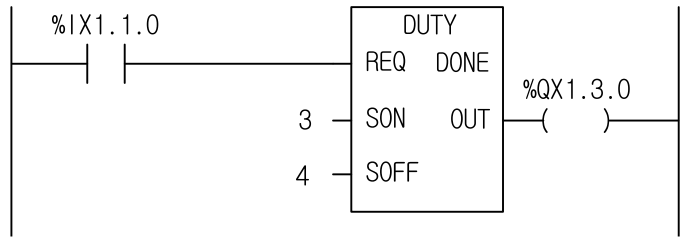
■ **Time Chart**



■ **Program Example**

If input contact %IX1.1.0 is set, output contact %QX1.3.0 is on during 3 scan times and off during 4 scan times.

1. LD



2. ST

INST_DUTY(REQ:=%IX1.1.0, SON:=3, SOFF:=4, OUT=>%QX1.3.0);

(1) Define DUTY function block as DUTY_C.

(2) Set %IX1.1.0 to REQ (the input contact) of DUTY.

(3) Set 3 to SON.

(4) Set 4 to SOFF.

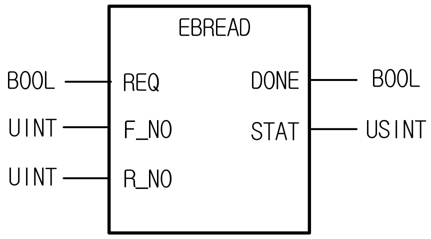
(5) Set %QX1.3.0 to output, OUT.

(6) After a program is complete, compile and write it to PLC.

(7) When 'Write' is complete, do 'Mode Change' (Stop → Run).

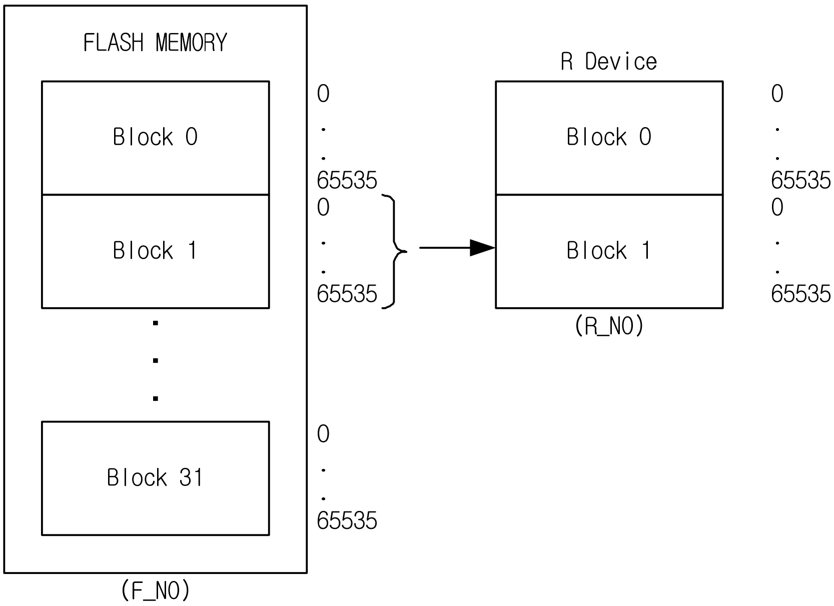
(8) If input contact %IX1.1.0 is on, output contact %QX1.3.0 is on during 3 scan times and off during 4 scan times.

EBREAD	**Write R area data to Flash area**	
	Availability	XGI, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute Function Block F_NO: Flash block no. when reading data - 0~1 (XGI-CPUU/D, CPUUN : 0~15) R_NO: R area block number **Output** DONE: maintains 1 after normally working STAT: displaying error info

■ **Function**

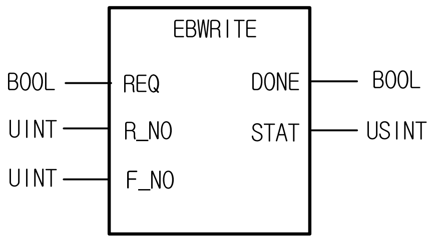
(1) Transfer 1 block (64Kbyte) of a designated R device to a block of flash area to save. DONE is 1 if it is normally completed.



(2) If R_NO is 2 and over (XGI-CPUU/D, CPUUN : 16 and over), STAT = 1 and if F_NO is 32 and over, STAT = 2, while _ERR and _LER is on. In addition, if reading data from flash, DONE = 0 and STAT = 5. DONE = 0 and STAT = 10 if Read/Write operation on a flash area is in progress during the operation is running.

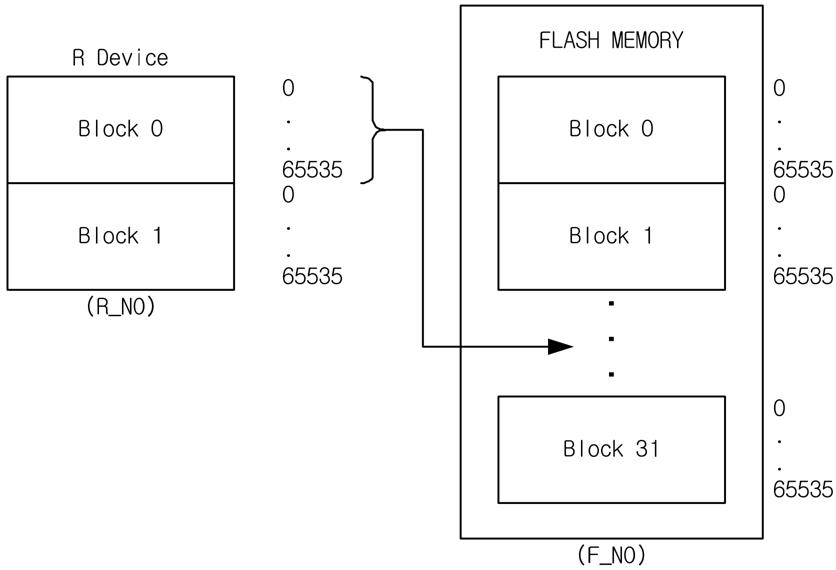
(3) While processing an instruction, the bit corresponds to F_NO of _RBLOCK_RD_FLAG is on.

EBWRITE	**Write R area data to Flash area**	
	Availability	XGI, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute Function Block R_NO: block number of R device(internal RAM) - 0~1 (XGI-CPUU/D, CPUUN : 0~15) E_NO: block number of flash area to save **Output** DONE: maintains 1 after normally working STAT: ERR info

■ **Function**

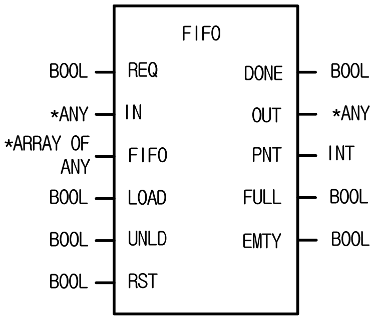
(1) Transfer 1 block (64Kbyte) of a configured R device to a block of flash area to save. DONE is 1 if normally completed.



(2) If R_NO is 2 and over (XGI-CPUU/D, CPUUN : 16 and over), STAT = 1 and if F_NO is 32 and over, STAT = 2, while _ERR and _LER is on. In addition, if writing to flash, DONE = 0 and STAT = 5. DONE = 0 and STAT = 10 if Read/Write operation on a flash area is in progress during the operation is running.

(3) While processing an instruction, the bit corresponding to F_NO of _RBLOCK_WR_FLAG is on.

FIFO	Load/Unload data to FIFO stack (First In First Out)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

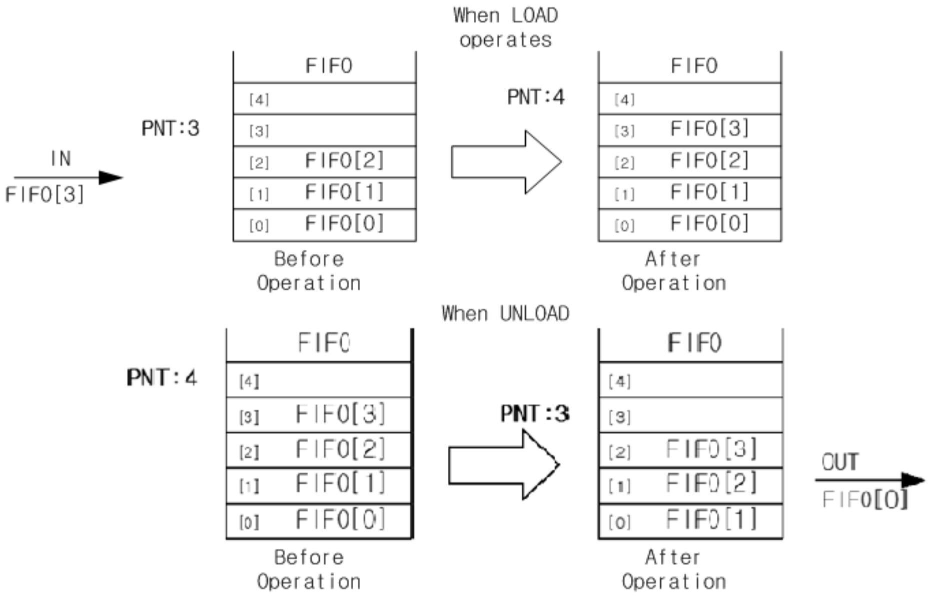
Function Block	Description
	**Input** REQ: requires to execute the function block IN: input data to be stored at FIFO stack LOAD: FB is on the input mode, if it's on. UNLD: FB is On the output mode, if it's on, RST: pointer value reset FIFO : Array used as FIFO stack **Output** DONE: it's 1 after first execution OUT: on output mode, it's the data from FIFO stack PNT: pointer for input data of FIFO stack FULL: if FIFO stack is full, it is 1 EMTY: if FIFO stack is empty, It is 1

ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	FIFO	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: exclude STRING from ANY types; *ARRAY OF ANY: excluding STRING from ARRAY_ANY type.

■ **Function**

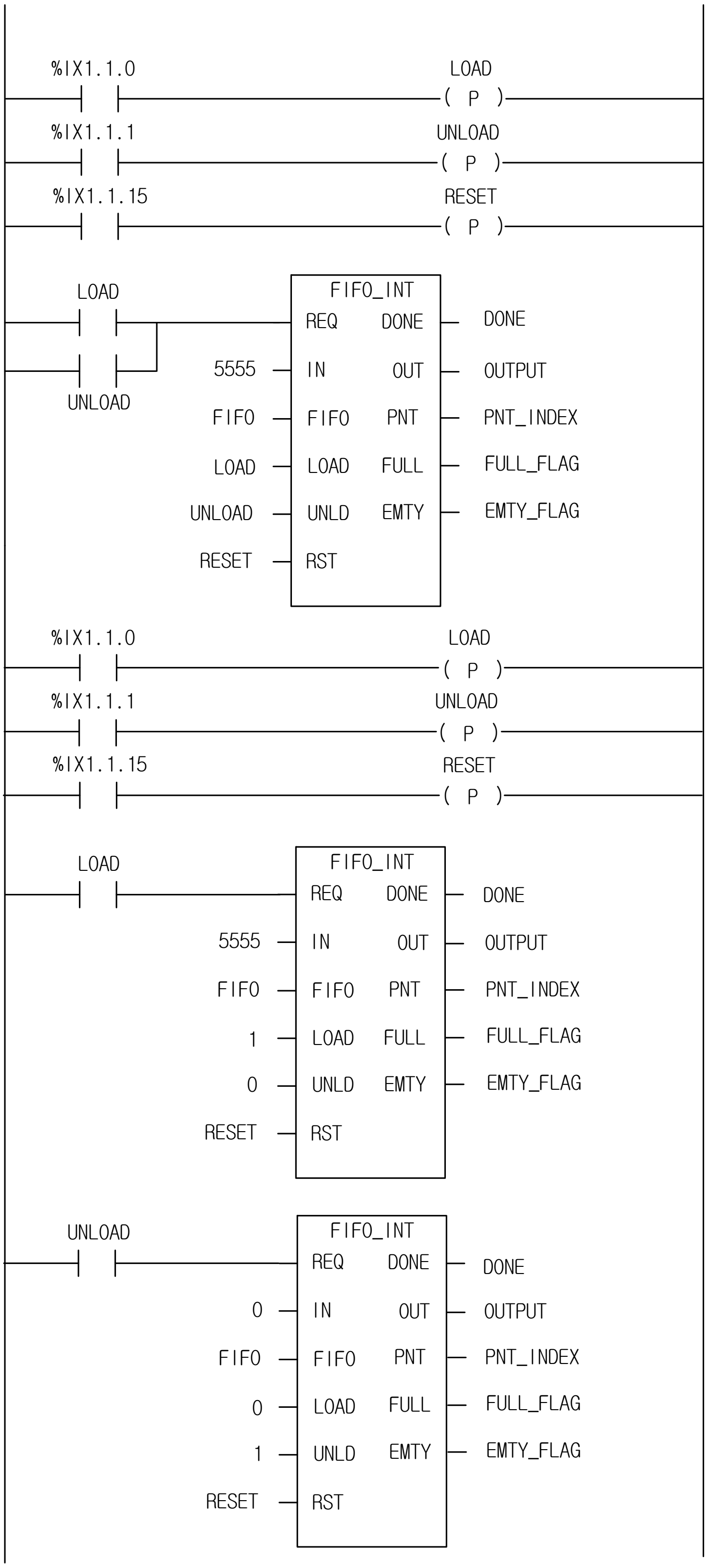
(1) It loads IN to FIFO or unloads data from FIFO.
(2) If Input and Output mode are set on at the same time, it executes In/Output simultaneously.
(3) If data is unloaded from FIFO, then the output is the lowest element of stack, the rest elements are shifts, PNT value is decreased by 1, and the element position of PNT is cleared (0).
(4) If RST is loaded to FIFO, PNT is initialized as 0, EMTY is on and all the data of FIFO stack are cleared as 0.
(5) The stack number is the input array number set by In/Output variable FIFO.
(6) If you want to keep the data of FIFO array variables and FIFO function block instance in case that power is off or power failure occurs, set them as 'RETAIN'.
(7) Reset functions are able to operate without REQ input.
(8) PNT shows the position of IN to be loaded next time, or the number of pointers to be loaded.
(9) If it's on the input mode, OUT is 0. But OUT at the output mode is retained in the converted input mode after output mode operation.



Function Block	FIFO variable type	Description
FIFO_BOOL	BOOL	It functions as FIFO for BOOL-type data
FIFO_BYTE	BYTE	It functions as FIFO for BYTE-type data
FIFO_WORD	WORD	It functions as FIFO for WORD-type data
FIFO_DWORD	DWORD	It functions as FIFO for DWORD-type data
FIFO_LWORD	LWORD	It functions as FIFO for LWORD-type data
FIFO_SINT	SINT	It functions as FIFO for SINT-type data
FIFO_INT	INT	It functions as FIFO for INT-type data
FIFO_DINT	DINT	It functions as FIFO for DINT-type data
FIFO_LINT	LINT	It functions as FIFO for LINT-type data
FIFO_USINT	USINT	It functions as FIFO for USINT-type data
FIFO_UINT	UINT	It functions as FIFO for UINT-type data
FIFO_UDINT	UDINT	It functions as FIFO for UDINT-type data
FIFO_ULINT	ULINT	It functions as FIFO for ULINT-type data
FIFO_REAL	REAL	It functions as FIFO for REAL-type data
FIFO_LREAL	LREAL	It functions as FIFO for LREAL-type data
FIFO_TIME	TIME	It functions as FIFO for TIME-type data
FIFO_DATE	DATE	It functions as FIFO for DATE-type data
FIFO_TOD	TOD	It functions as FIFO for TOD-type data
FIFO_DT	DT	It functions as FIFO for DT-type data

■ **Program Example**

1. LD



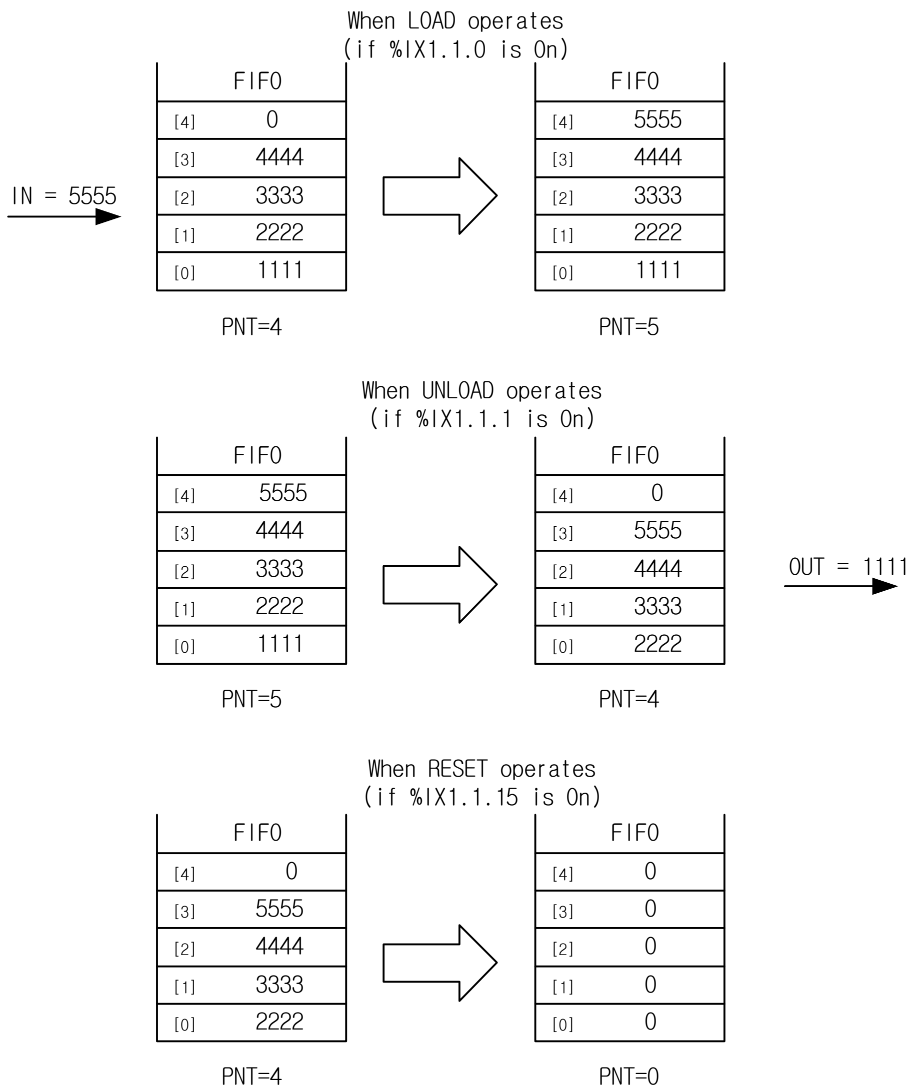
FIFO_INT function block is used as the above. The two examples of the above execute the same operation. The above figure illustrate a program which executes input and output functions at the same time using only one function block and following figure illustrates a program which executes input and output functions independently, using input function and output function, respectively. Note that both instance names must be the same.

(1) If the input conditions (%IX1.1.0, %IX1.1.1, %IX1.1.15) are on, FIFO_INT executes.

(2) If input contact %IX1.1.0 is on, load function is executed. 5555 is loaded to FIFO stack and PNT_INDEX increased by 1.

(3) If input contact %IX1.1.1 is on, unload function executes. 1111 is unloaded from FIFO stack and PNT_INDEX is decreased by 1.

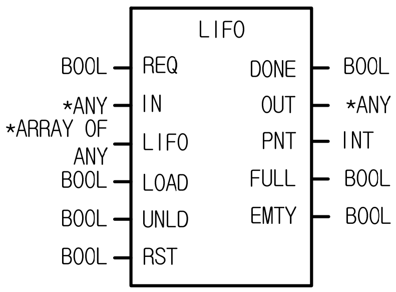
(4) If input contact %IX1.1.15 is on, reset function executes. All the stack of FIFO is cleared as 0, PNT_INDEX is initialized as 0 and EMTY_FLAG is on.



2. ST

```
INST_FIFO_INT(REQ:=LOAD OR UNLOAD, IN:=5555, FIFO:=FIFO, LOAD:=LOAD, UNLD:=UNLOAD,
RST:=RESET, DONE=>DONE, OUT=>OUTPUT, PNT=>PNT_INDEX, FULL=>FULL_FLAG, EMTY=>EMTY_FLAG);
```

LIFO	Load/Unload data to LIFO stack (Last In First Out)	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** REQ: to execute the function block IN: input data to be stored at LIFO stack LOAD: FB is on, the input mode, if it is on UNLD: FB is on the output mode, if it is on RST: pointer value reset LIFO : Array used as LIFO stack. **Output** DONE: it is 1 after first execution OUT: on output mode, it is the data from LIFO stack PNT: pointer for input data of LIFO stack FULL: if LIFO stack is full, it is 1 EMTY: if LIFO stack is empty, it is 1

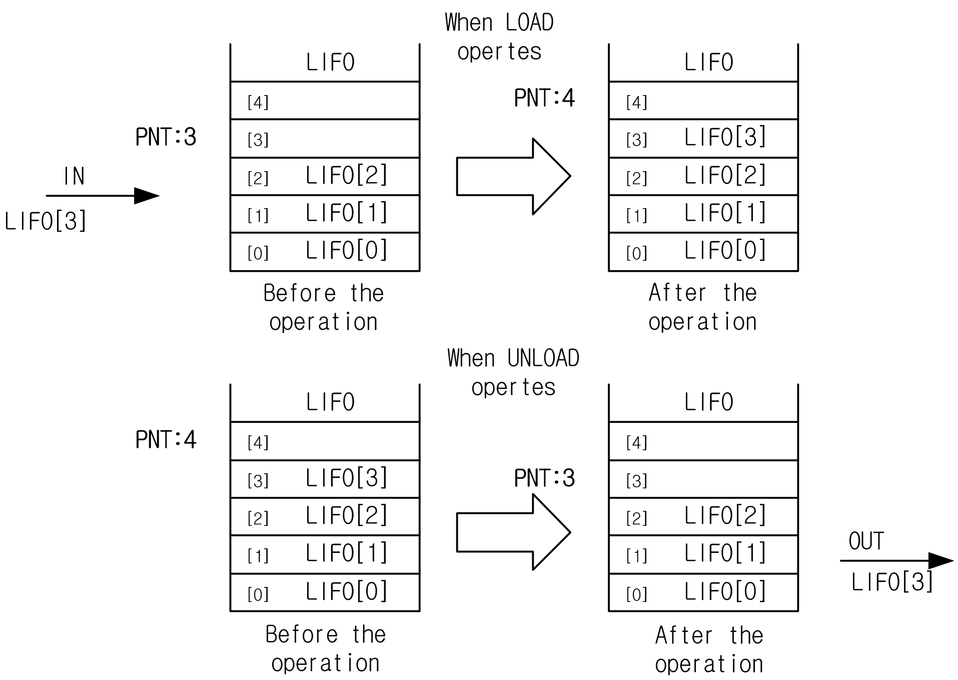
ANY type variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	LIFO	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: exclude STRING from ANY type, *ARRAY OF ANY: exclude STRING from ARRAY OF ANY type.

■ **Function**

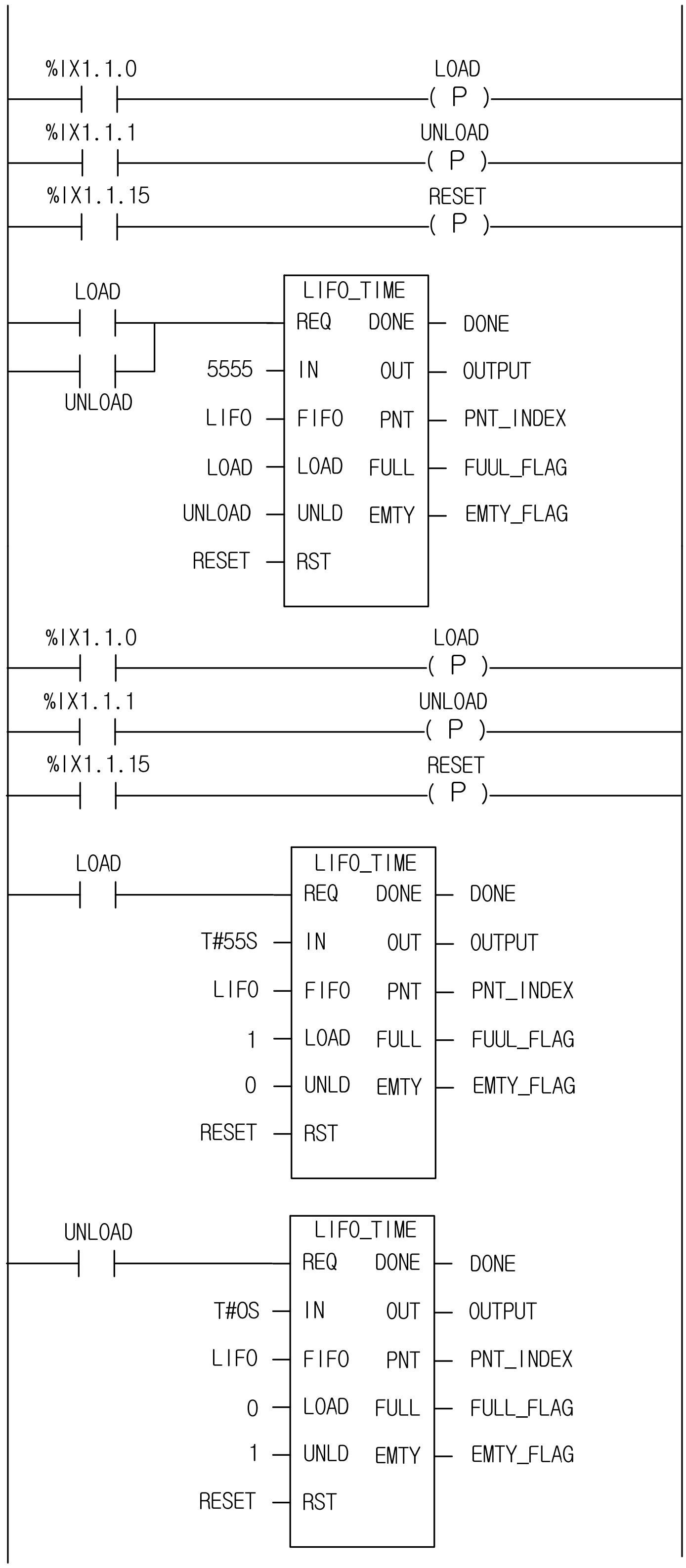
(1) It loads IN to LIFO or unloads data from LIFO.
(2) If LOAD and UNLD are on at the same time, input IN is produced as output ,OUT.
(3) If data is unloaded from LIFO by unload function of LIFO_***, unloaded data is deleted in stack and initialized as 0.
(4) If RST is loaded to LIFO, PNT is initialized as 0, EMTY is on and all the data of LIFO stack are cleared as 0.
(5) The stack number is the array number set by In/Output variable LIFO.
(6) If you want to keep the data of LIFO array variables and LIFO function block instance, in case that power is off or power failure occurs, set them as 'RETAIN'.
(7) Reset functions are able to operate without REQ input.
(8) PNT shows the position of IN to be loaded next time, or the number of pointers to be loaded.
(9) If it is on the input mode, output ,OUT is 0.
(10) If load and unload signals are entered simultaneously, IN is produced to OUT.
(11) In case of input mode, OUT is 0. However, if the input mode converted after output mode operation, OUT value of output mode is maintained

Function Block	FIFO variable type	Description
LIFO_BOOL	BOOL	It functions as LIFO for BOOL-type data
LIFO_BYTE	BYTE	It functions as LIFO for BYTE-type data
LIFO_WORD	WORD	It functions as LIFO for WORD-type data
LIFO_DWORD	DWORD	It functions as LIFO for DWORD-type data
LIFO_LWORD	LWORD	It functions as LIFO for LWORD-type data
LIFO_SINT	SINT	It functions as LIFO for SINT-type data
LIFO_INT	INT	It functions as LIFO for INT-type data
LIFO_DINT	DINT	It functions as LIFO for DINT-type data
LIFO_LINT	LINT	It functions as LIFO for LINT-type data
LIFO_USINT	USINT	It functions as LIFO for USINT-type data
LIFO_UINT	UINT	It functions as LIFO for UINT-type data
LIFO_UDINT	UDINT	It functions as LIFO for UDINT-type data
LIFO_ULINT	ULINT	It functions as LIFO for ULINT-type data
LIFO_REAL	REAL	It functions as LIFO for REAL-type data
LIFO_LREAL	LREAL	It functions as LIFO for LREAL-type data
LIFO_TIME	TIME	It functions as LIFO for TIME-type data
LIFO_DATE	DATE	It functions as LIFO for DATE-type data
LIFO_TOD	TOD	It functions as LIFO for TOD-type data
LIFO_DT	DT	It functions as LIFO for DT-type data



■ **Program Example**

1. LD



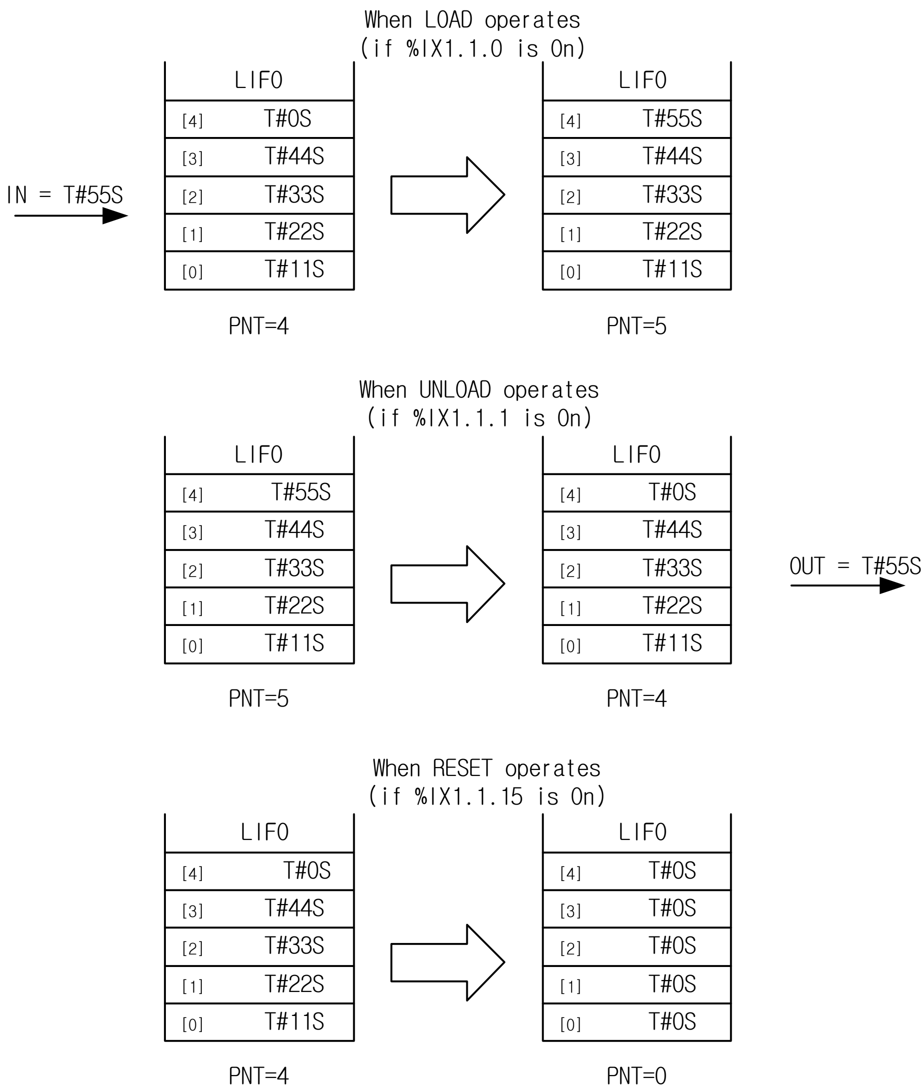
LIFO_TIME function block is used as the above. The two examples of the above execute the same operation. The above figure illustrate a program which executes input and output functions at the same time using only one function block and the below figure illustrates a program which executes input and output functions independently, using input function and output function, respectively. Note that both instance names must be the same.

(1) If the input conditions (%IX1.1.0, %IX1.1.1, %IX1.1.15) are on, LIFO_TM executes.

(2) If input contact %IX1.1.0 is on, load function executes. T#55S is loaded to LIFO stack and PNT_INDEX is increased by 1.

(3) If input contact %IX1.1.1 is on, unload function executes. T#55S is unloaded from LIFO stack and PNT_INDEX is decreased by 1.

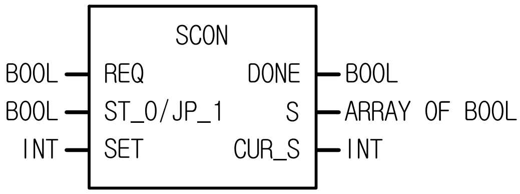
(4) If input contact %IX1.1.15 is on, reset function executes. All the stack of LIFO is cleared as T#0S, PNT_INDEX is initialized as 0 and EMTY_FLAG is on.



2. ST

```
INST_LIFO_TIME(REQ:=LOAD OR UNLOAD, IN:=T#55S, LIFO:=LIFO, LOAD:=LOAD, UNLD:=UNLOAD, RST:=RST,
DONE=>DONE, OUT=>OUTPUT, PNT=>PNT_INDEX, FULL=>FULL_FLAG, EMTY=>EMTY_FLAG);
```

SCON	**Step Controller (Step in order and jump of step)**	
	Availability	XGI, XGR, XEC, XMC
	Flags	_ERR , _LER

Function Block	Description
	**Input** REQ: if it is 1, the function block executes S/O: if 0, SET function is enabled; if 1, OUT function is enabled. SET: step number (0 ~ 99) **Output** DONE: without an error, it is on. If error is occurred or there is no request of execution, it is off S: produces an set bit array CUR_S: produces a current step number

■ **Function**

(1) Setting of step controller group
The instance name of function block is the name of step controlling group.
(Examples of FB declaration: S00, G01, Manu1, Examples of step contacts: S00.S[1], G01.S[1], Manu1.S[1])

2. In case of SET function (ST_0/JP_1 = 0)
In the same step controller group, the present step number can be on when the previous step number is on.
If the present step number is on, it keeps its state even when the input is off.
Only one step number is on even when several input conditions are on at the same time.
If Sxx.S[0] is on, all the SET output is cleared.

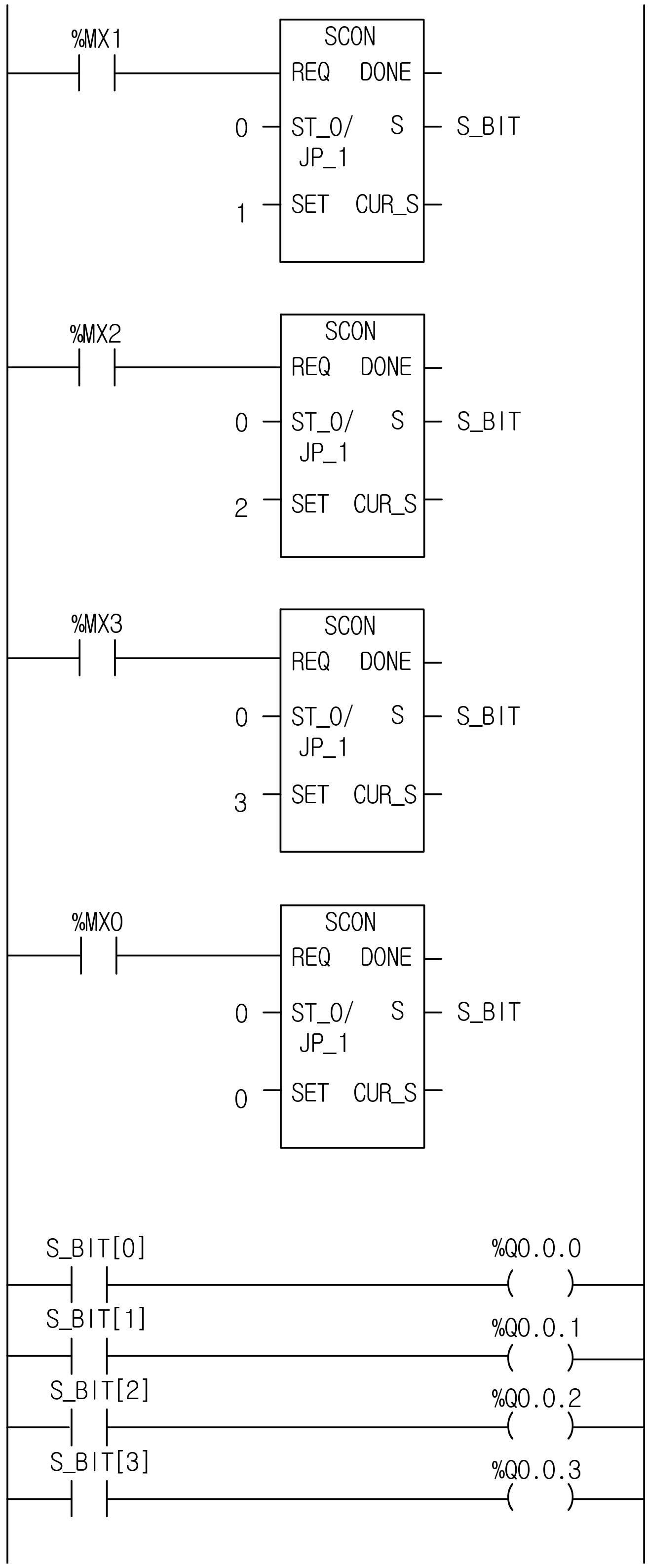
3. In case of JUMP function (ST_0/JP_1 = 1)
In the same step controller group, only one step number is on, even when several input conditions are on.
If input conditions are on at the same time, last programmed one is produced.
If the present step number is on, it keeps its state even when the input is off..
If Sxx.S[0] is on, it returns to its first step.

■ **Flag**

Flag	Description
_ERR	An error occurs when step setting (SET) is out of its range (0 ~ 99). If an error occurs, DONE is off and step output maintains its previous step.

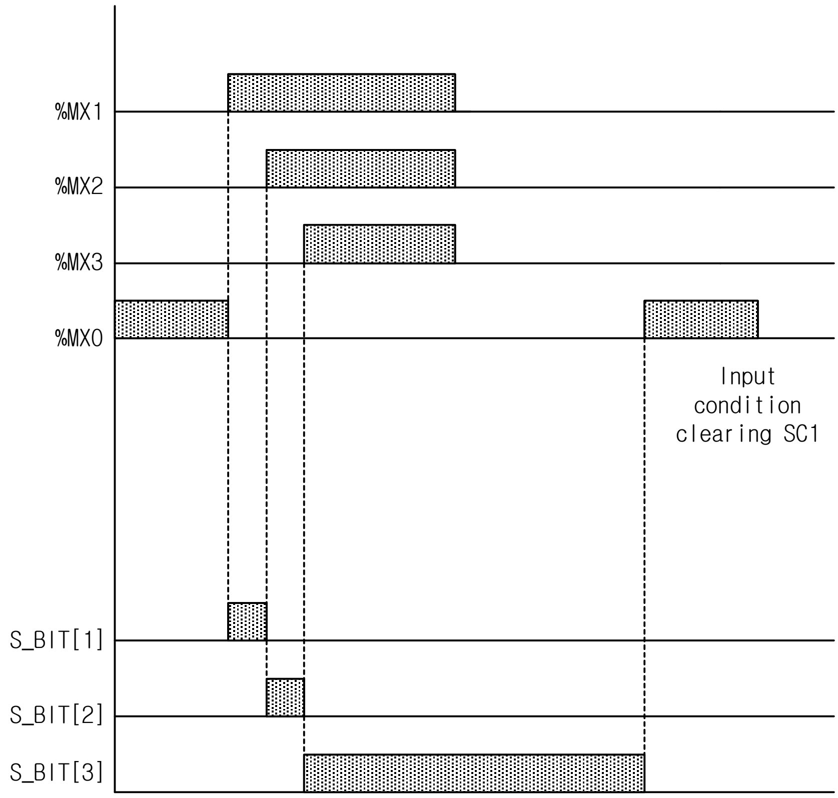
■ In case of SET function (ST_0/JP_1 = 0), using SC1 group

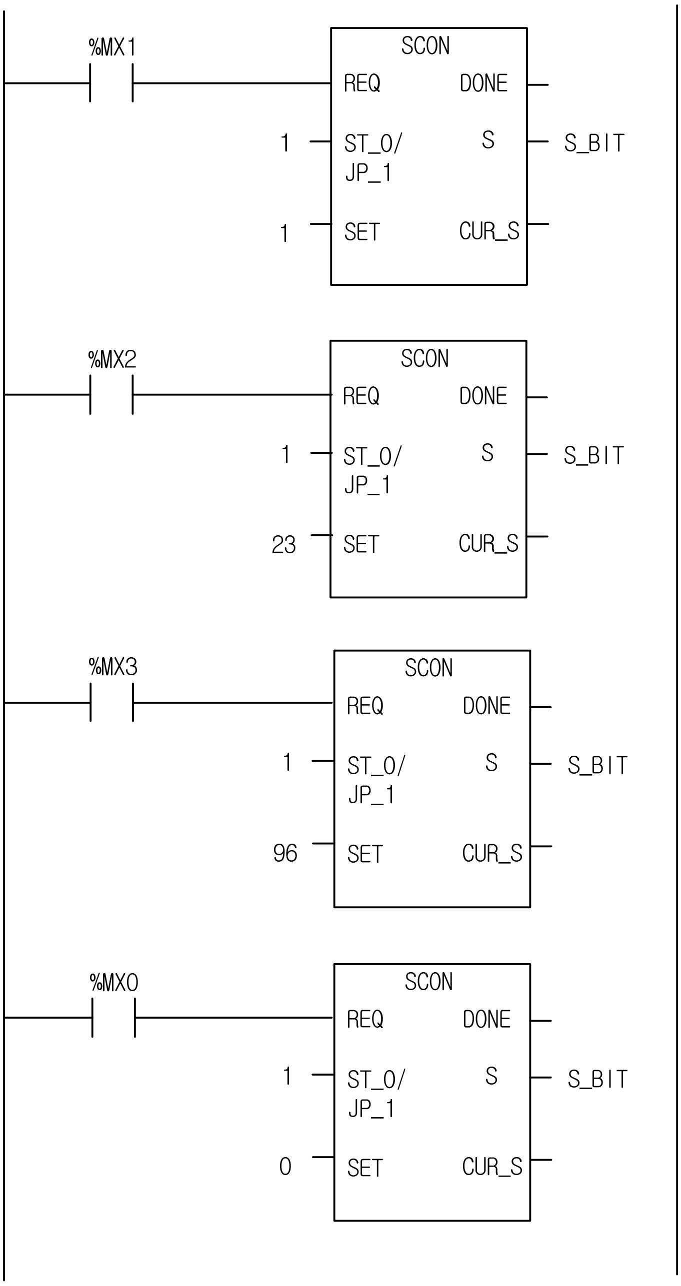
1. LD



2. ST

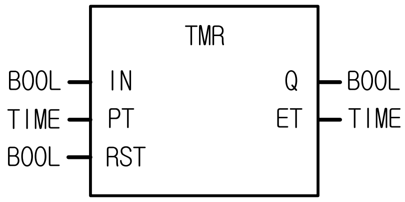
```
INST_SCON(REQ:=%MX1, ST0_JP1:=0, SET:=1, S=>S_BIT);
INST_SCON1(REQ:=%MX2, ST0_JP1:=0, SET:=2, S=>S_BIT);
INST_SCON2(REQ:=%MX3, ST0_JP1:=0, SET:=3, S=>S_BIT);
INST_SCON3(REQ:=%MX4, ST0_JP1:=0, SET:=0, S=>S_BIT);

%QX0.0.0 := S_BIT[0];
%QX0.0.1 := S_BIT[1];
%QX0.0.2 := S_BIT[2];
%QX0.0.3 := S_BIT[3];
```





NO	%MX 1	%MX 2	%MX 3	%MX 4	S_O [1]	S_O [23]	S_O [98]	S_O [0]
1	On	Off	Off	Off	○			
2	On	On	Off	Off		○		
3	On	On	On	Off			○	
4	On	On	On	On				○

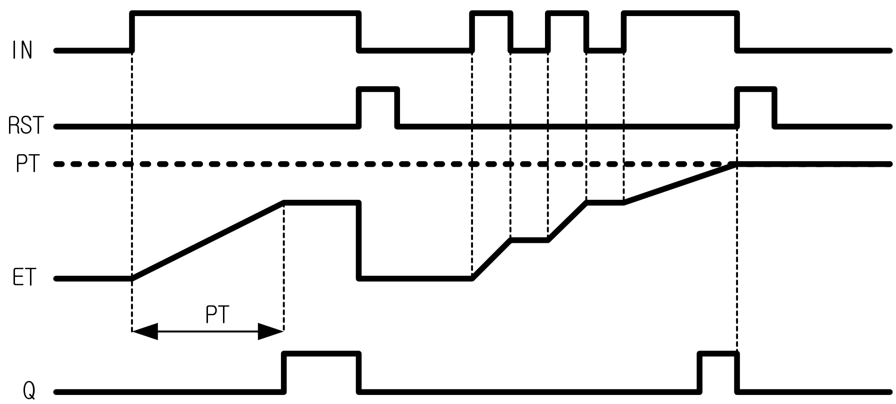
TMR	Integration Timer	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** IN: operation condition for Timer PT: preset time RST: reset **Output** Q: timer output ET: elapsed time

■ **Function**

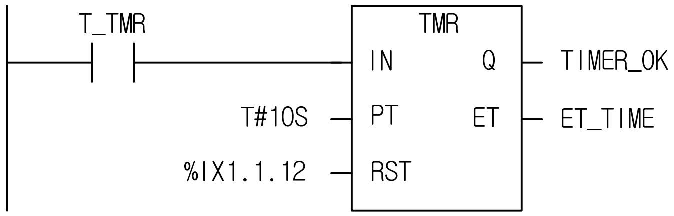
1. When IN is 1, elapsed time is produced at ET.
2. Even if IN is 0 before ET reaches PT, ET keeps its value. If IN is 1 again, elapsed time is produced at ET integrating its previous value.
3. If ET reaches PT, Q is 1.
4. If RST is 1, Q and ET are 0.

■ **Time Chart**



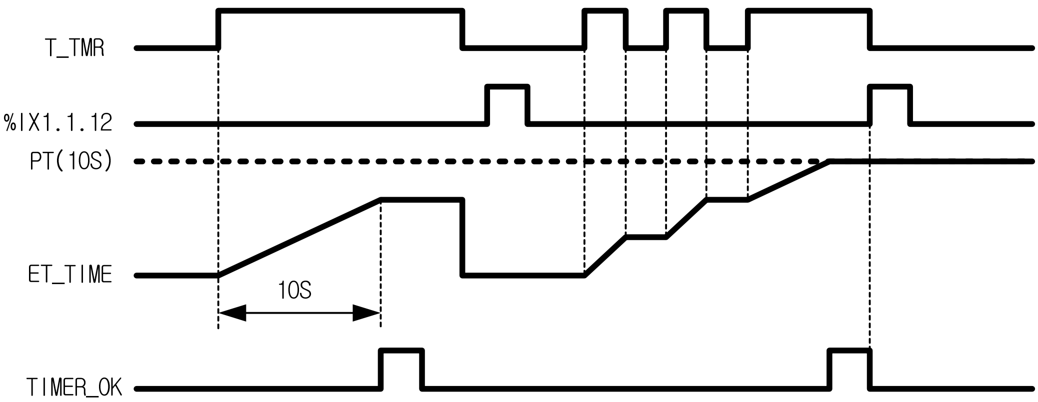
■ **Program Example**

1. LD



2. ST

INST_TMR(IN:=T_TMR, PT:=T#10S, RST:=%IX1.1.12, Q=>TIMER_OK, ET=>ET_TIME);



(1) If 10 seconds passes after input variable T_TMR is 1, output variable TIMER_OK is 1.

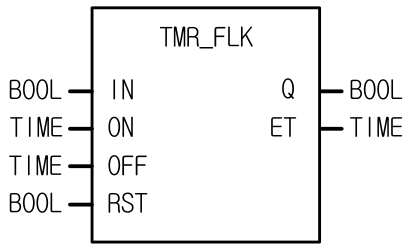
(2) Elapsed time is produced at ET_TIME after T_TMR is 1.

(3) ET_TIME keeps its value even if T_TMR is 0 before ET_TIME reaches its preset time 10 seconds.

(4) If T_TMR is 1, elapsed time is produced at ET_TIME integrating its previous value.

(5) If input contact %IX1.1.12 is 1, elapsed time ET_TIME and output variable TIMER_OK are all cleared.

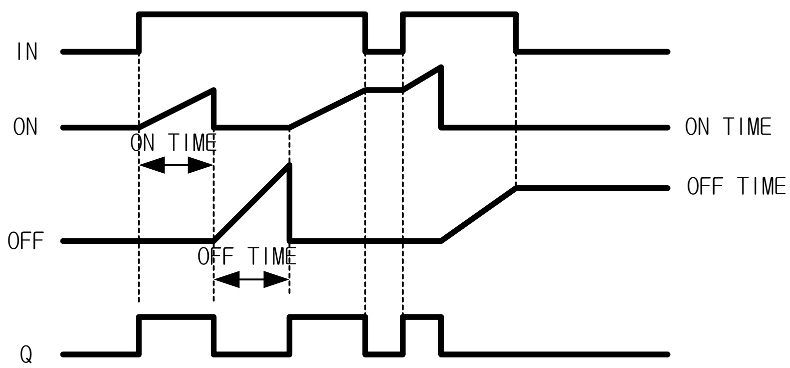
TMR_FLK	**TMR with Flicker**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** IN: operation condition for Timer ON: on setting time of timer OFF: off setting time of timer RST: reset **Output** Q: Timer output ET: elapsed time

■ **Function**

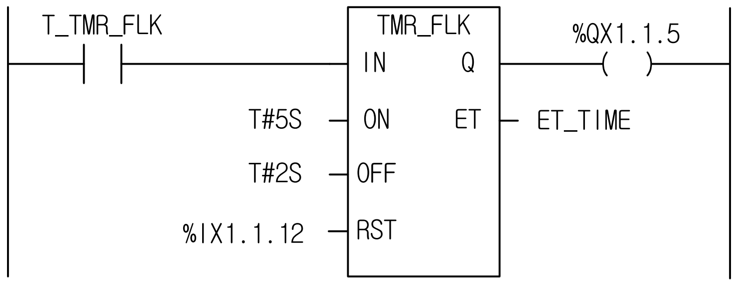
(1) As soon as IN gets 1, Q becomes 1 and Q maintains its value during on setting time.
(2) After setting time which is set by on, Q is 0 during the time which is set by off.
(3) If IN is 0, it stops its function of either on or off operation and keeps its time. If IN is 1 again, it executes with its previous data.
4. Output Q is 0 while IN is 0.
5. If ON is 0, output Q is always 0.

■ **Time Chart**



■ **Program Example**

1. LD



2. ST

```
INST_TMR_FLK(IN:=T_TMR_FLK, ON:=T#5S, OFF:=T#2S, RST:=%IX1.1.12, Q=>%QX1.1.5, ET=>ET_TIME);
```

(1) If input variable T_TMR_FLK is 1, TMR_FLK function block executes.

(2) Output contact %QX1.1.5 is 1 during 5 seconds set by on after input variable T_TMR_FLK is 1.

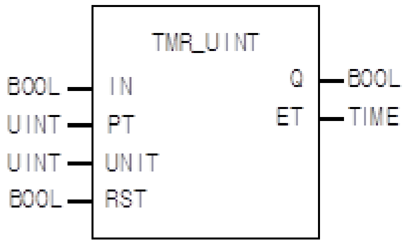
(3) Output contact %QX1.1.5 is 0 during 2 seconds set by off after 5 seconds set by on.

(4) TON time (On) when Q is 1 and TOF time (Off) when Q is 0 are produced at ET_TIME by turns while T_TMR_FLK is 1.

(5) If input variable T_TMR_FLK is 0, then it keeps its time and output contact %QX1.1.5 is 0. If T_TMR_FLK is 1, it executes again.

(6) If input %IX1.1.12 is 1, elapsed time ET_TIME and output contact %QX1.1.5 are all cleared.

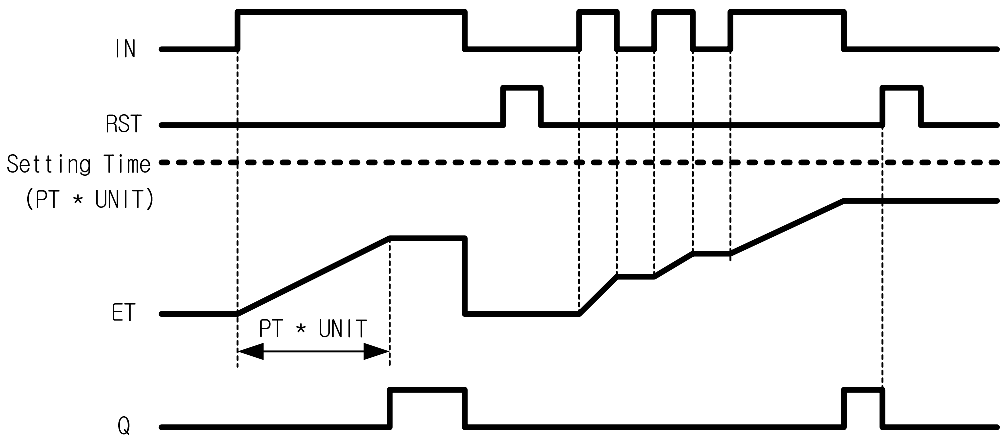
TMR_UINT	TMR with Integer setting	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** IN: operation condition for Timer PT: preset time UNIT: time unit of setting time RST: reset input **Output** Q: timer output ET: elapsed time

■ **Function**

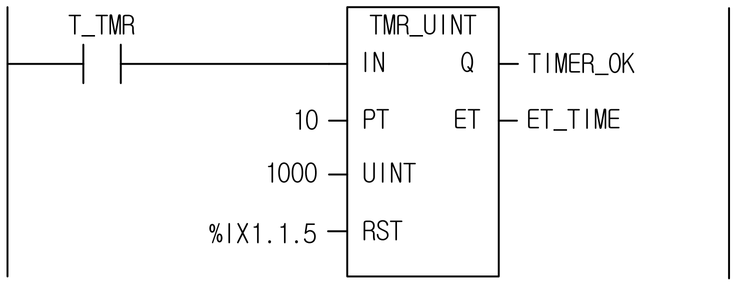
(1) Elapsed time is produced at ET after IN is 1.
(2) Even if IN is 0 before ET reaches PT, ET keeps its value. If IN is 1 again, elapsed time is increased.
(3) Q is 1 when elapsed time reaches preset time.
(4) If RST is 1, Q and ET are 0.
(5) Setting time is PT x UNIT (ms).

■ **Time Chart**



■ **Program Example**

1. LD



2. ST

INST_TMR_UINT(IN:=T_TMR, PT:=10, UNIT:=1000, RST:=%IX1.1.5, Q=>TIMER_OK, ET=>ET_TIME);

(1) Setting time is PT x UNIT[ms] = 10 x 1000[ms] = 10[s].

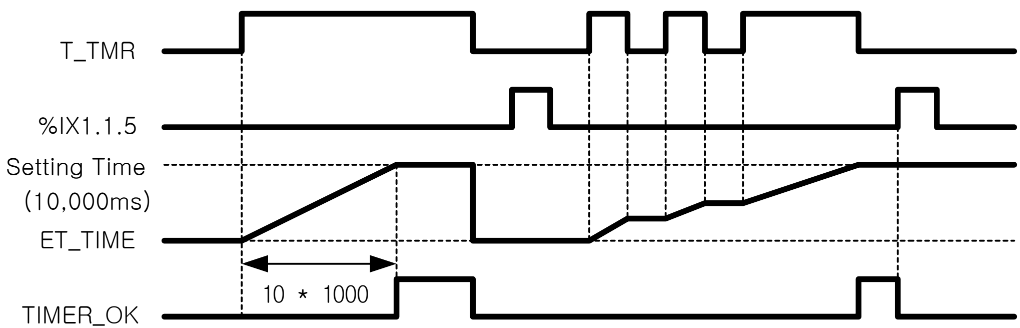
(2) Output variable TIMER_OK is 1, if 10 seconds passes after input variable T_TMR is 1.

(3) Elapsed time is produced at ET_TIME after input variable T_TMR is 1.

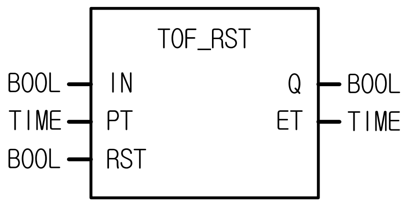
(4) Even if T_TMR is 0 before ET_TIME reaches preset time ,10 seconds, ET_TIME keeps its value.

(5) If input variable T_TMR is 1 again, elapsed time is produced at ET integrating its previous value.

(6) If input contact %IX1.1.5 is 1, elapsed time ET_TIME and output contact TIMER_OK are all cleared.



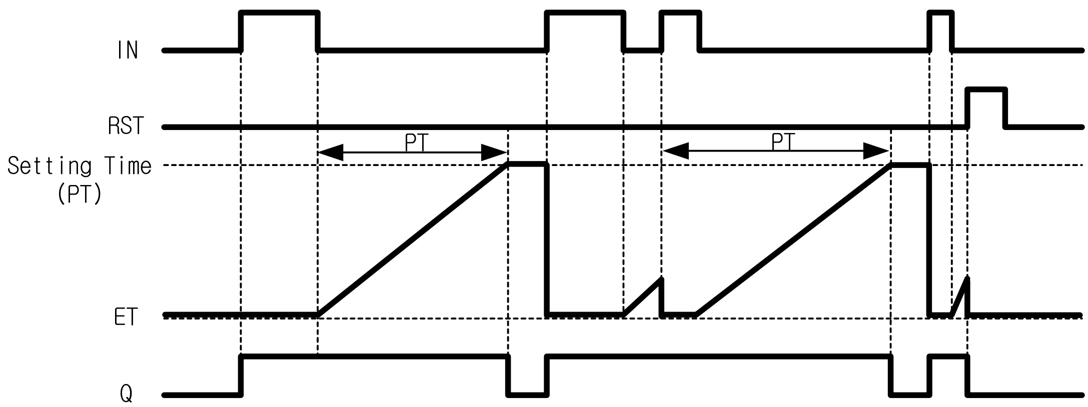
TOF_RST	Delay Timer is able to output Off in operation	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** IN: operation condition for Timer PT: preset time RST: reset **Output** Q: Timer output ET: elapsed time

■ **Function**

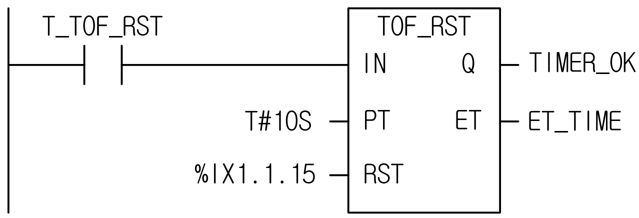
(1) Q is 1 when IN is 1 and Q is 0 when preset time (PT) elapses after IN became 0.
(2) Elapsed time is produced at ET after IN is 0.
(3) Elapsed time is 0 if IN is 1 before ET reaches PT.
(4) If RST is 1, Q and ET are 0.

■ **Time Chart**



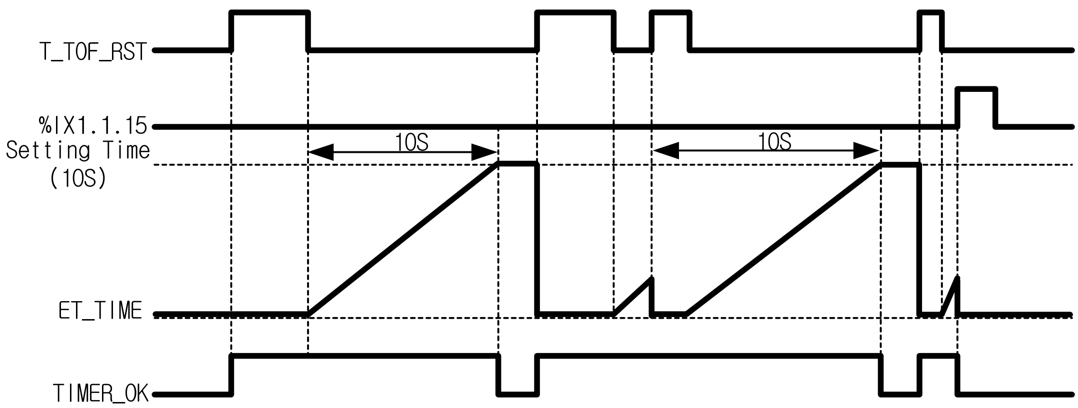
■ **Program Example**

1. LD



2. ST

INST_TOF_RST(IN:=T_TOF_RST, PT:=T#10S, RST:=%IX1.1.15, Q=>TIMER_OK, ET=>ET_TIME);



(1) If input variable T_TOF_RST is 1, output variable TIMER_OK is 1. And TIMER_OK is 0 when 10 seconds elapse after T_TOF_RST became 0.

(2) If T_TOF_RST is 1 within 10 seconds after it turns off, TOF_RST is initialized.

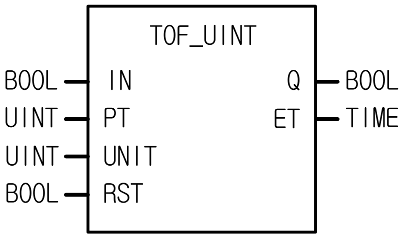
(3) Elapsed time is produced at ET_TIME.

(4) If input contact %IX1.1.15 is 1, elapsed time ET_TIME and output contact TIMER_OK are all cleared.

☆ **Note**

TOF_RST Function Block keeps operating after the contact is on until its operation is complete. In case of a variable using array index, array index error occurs only when the contact is on. Therefore, TOF_RST Function Block does not produce any array index error as long as the contact is off ,although function block is operating.

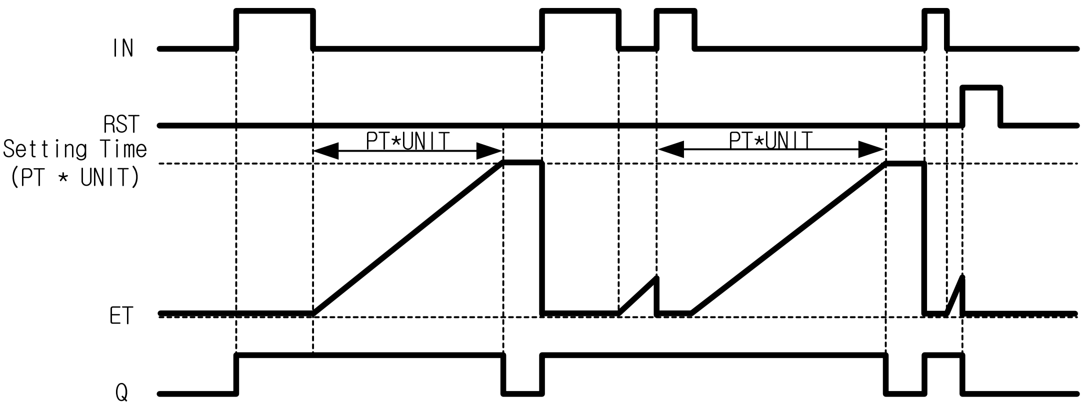
TOF_UINT	**Off Timer of Integer setting**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** IN: operation condition for Timer PT: preset time UNIT: time unit of setting time RST: reset **Output** Q: Timer output ET: elapsed time

■ **Function**

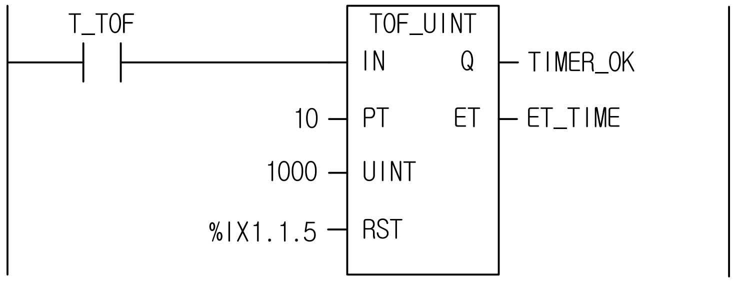
(1) Q is 1 when IN is 1. And Q is 0, if setting time (PT) passes after IN is 0.
(2) Elapsed time is produced at ET after IN is 0.
(3) If IN is 1 before ET reaches PT, ET becomes 0 again.
(4) If RST is 1, Q and ET are 0.
(5) Setting time is PT x UNIT (ms).

■ **Time Chart**



■ **Program Example**

1. LD



2. ST

INST_TOF_UINT(IN:=T_TOF, PT:=10, UNIT:=1000, RST:=%IX1.1.5, Q=>TIMER_OK, ET=>ET_TIME);

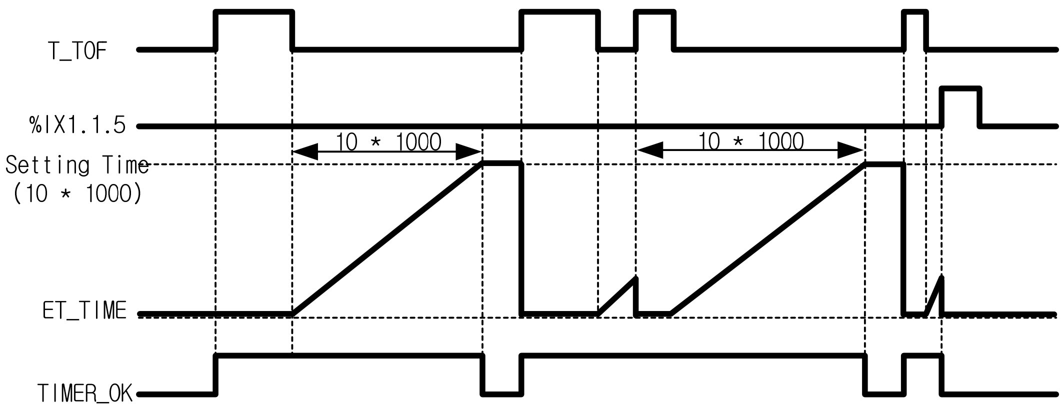
(1) Preset time PT x UNIT[ms] = 10 x 1000[ms] = 10[s].

(2) If input variable T_TOF is 1, output variable TIMER_OK is 1. TIMER_OK is 0, if 10 seconds passes after T_TOF is 0.

(3) If T_TOF becomes 1 again within 10 seconds, TOF_UINT initializes.

(4) Elapsed time is produced at ET_TIME.

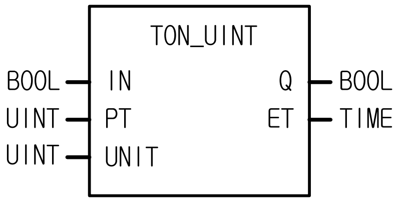
(5) If input contact %IX1.1.5 is 1, TIMER_OK and ET_TIME are all cleared



☆ **Note**

TOF_UINT Function Block keeps operating after the contact is on until its operation is complete. In case of a variable using array index, array index error occurs only when the contact is on. Therefore, TOF_UINT Function Block does not produce any array index error as long as the contact is off although function block is operating.

<table>
<tr><td rowspan="3">TON_UINT</td><td colspan="2">On Timer of Integer setting</td></tr>
<tr><td>Availability</td><td>XGI, XGR, XEC, XMC</td></tr>
<tr><td>Flags</td><td></td></tr>
</table>

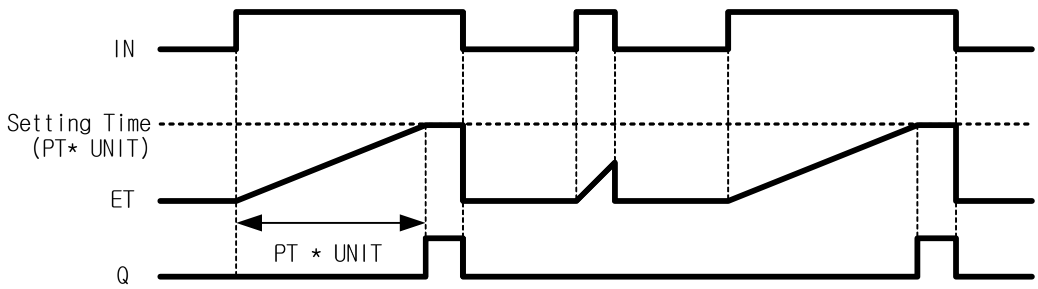
<table>
<tr><th>Function Block</th><th>Description</th></tr>
<tr><td>
</td><td>Input IN: operation condition for Timer
PT: preset time
UNIT: time unit of setting time

Output Q: timer output
ET: elapsed time</td></tr>
</table>

■ **Function**

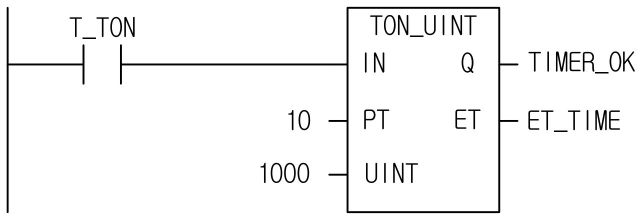
(1) Elapsed time is produced at ET after IN is 1.
(2) Elapsed time ET is 0, if IN is 0 before ET reaches PT.
(3) Q is 0, if IN is 0 after Q is 1.
(4) Preset time is PT x UNIT[ms].

■ **Time Chart**



■ **Program Example**

1. LD



2. ST

INST_TON_UINT(IN:=T_TON, PT:=10, UNIT:=1000, Q=>TIMER_OK, ET=>ET_TIME);

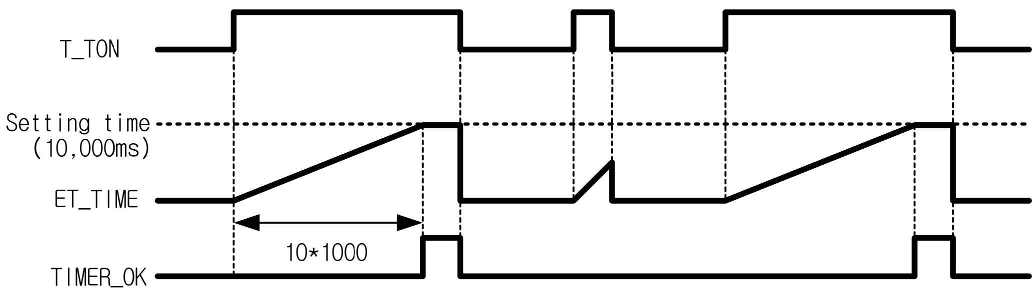
(1) Preset time is PT x UNIT[ms] = 10 x 1000[ms] = 10[s].

(2) If 10 seconds passes after input variable T_TON is on, output variable TIMER_OK is 1.

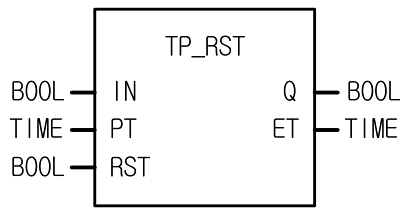
(3) Elapsed time is produced at ET_TIME after input variable T_TON is on.

(4) If T_TON is 0 before elapsed time ET_TIME reaches 10 seconds, ET_TIME is 0.

(5) If T_TON is 0 after TIMER_OK is 1, TIMER_OK and ET_TIME are 0.



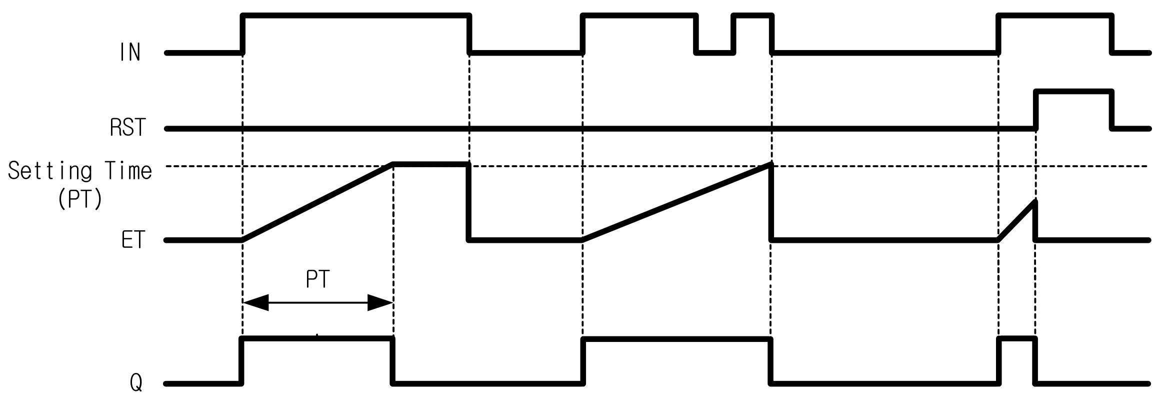
TP_RST	Pulse timer is able to Off output of contact.	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** IN: operation condition for Timer PT: preset time RST: reset **Output** Q: timer output ET: elapsed time

■ **Function**

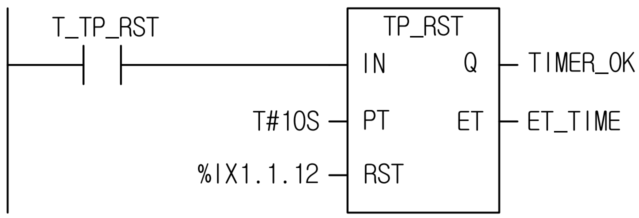
(1) If IN is 1, Q is 1. And if elapsed time reaches preset time, timer output Q is 0.
(2) ET increases its value from when IN is 1, keeps its value at PT and is cleared when IN is 0.
(3) It doesn't matter whether IN changes its state or not while timer output Q is 1 (during a pulse output).
(4) If RST is 1, output Q and ET are 0.

■ **Time Chart**



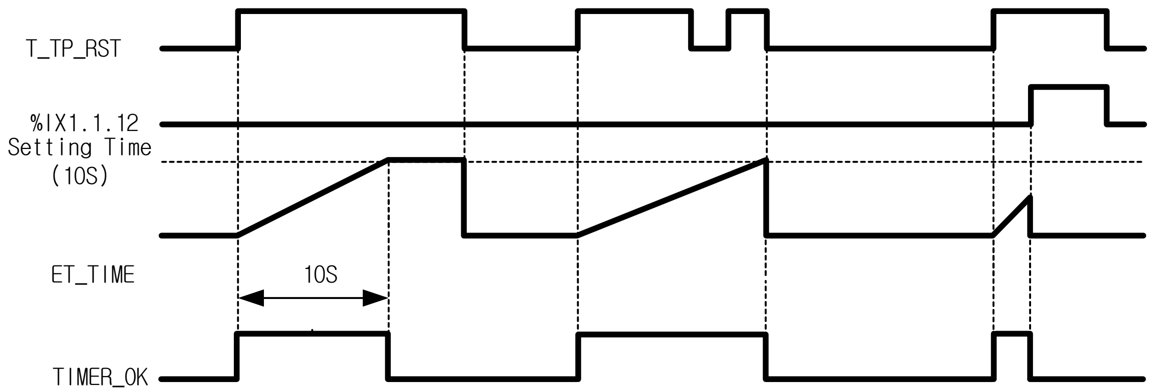
■ **Program Example**

1. LD



2. ST

INST_TP_RST(IN:=T_TP_RST, PT:=T#10S, RST:=%IX1.1.12, Q=>TIMER_OK, ET=>ET_TIME);



(1) If input variable T_TP_RST is 1, output variable TIMER_OK is 1. And 10 seconds later, TIMER_OK is 0. Once TP_RST timer executes, input T_TP_RST doesn't matter during 10 seconds.

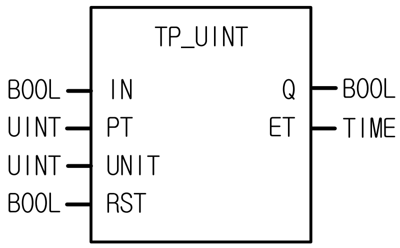
(2) ET_TIME value increases and stops at 10S. And if T_TP_RST is 0, ET_TIME becomes 0.

(3) If input contact %IX1.1.12 is 1, TIIMER_OK and ET_TIME are all cleared.

☆ **Note**

TP_RST Function Block keeps operating after the contact is on until its operation is complete. In case of a variable using array index, array index error occurs only when the contact is on. Therefore, TP_RST Function Block does not produce any array index error as long as the contact is off although function block is operating.

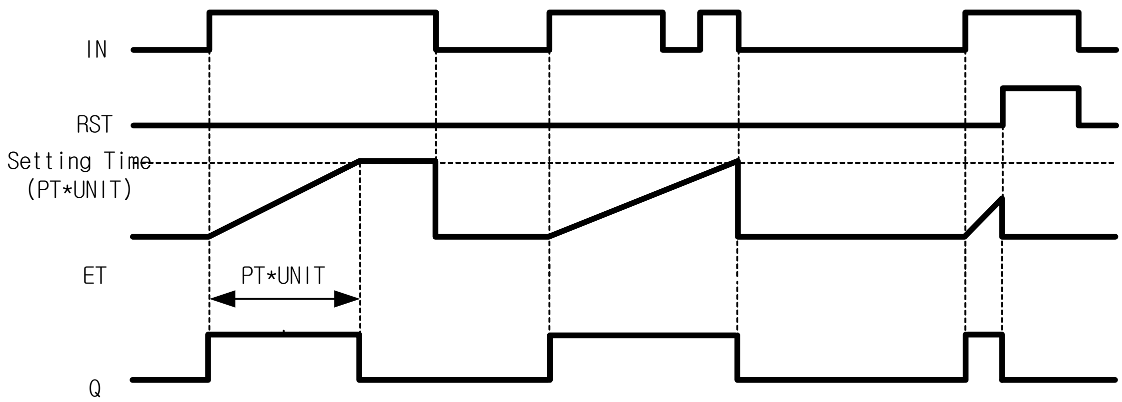
TP_UINT	Pulse Timer with Integer setting	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** IN: operation condition for Timer PT: preset time UNIT: time unit of setting time RST: reset **Output** Q: timer output ET: elapsed time

■ **Function**

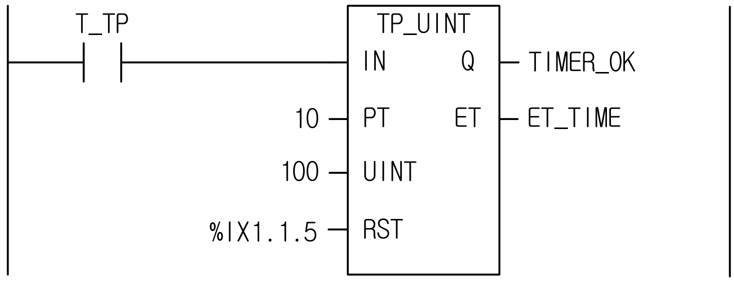
(1) If IN is 1, Q is 1. And if elapsed time reaches preset time, timer output Q is 0.
(2) ET increases its value from when IN is 1, keeps its value at PT and is cleared when IN is 0.
(3) It does not matter whether IN changes its state or not while timer output Q is 1 (during a pulse output).
(4) If RST is 1, output Q and ET are 0.
(5) Preset time is PT x UNIT[ms].

■ **Time Chart**



■ **Program Example**

1. LD



2. ST

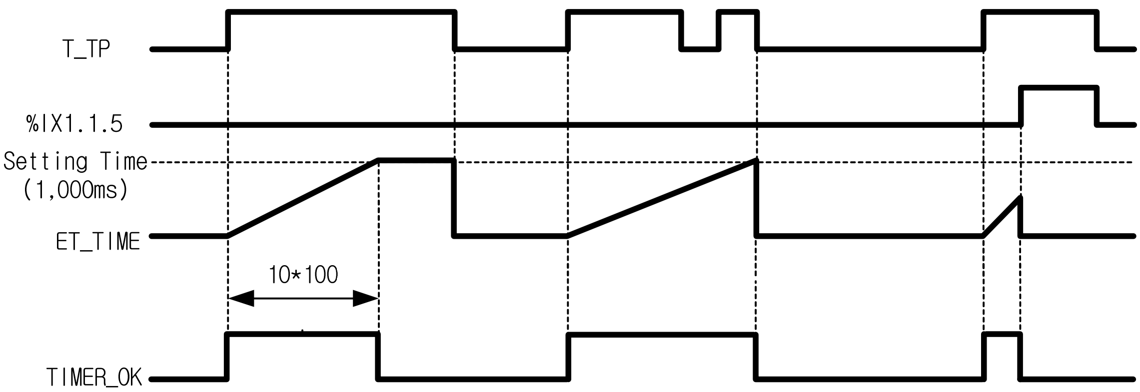
INST_TP_UINT(IN:=T_TP, PT:=10, UNIT:=100, RST:=%IX1.1.5, Q=>TIMER_OK, ET=>ET_TIME);

(1) Preset time is PT x UNIT[s] = 10 x 100[ms] = 1[s].

(2) If input variable T_TP is 1, output variable TIMER_OK is 1. And 10 seconds later, TIMER_OK is 0. Once TP_UINT timer executes, input T_TP does not matter.

(3) ET_TIME value increases and stops at 1,000. And if T_TP is 0, it is 0.

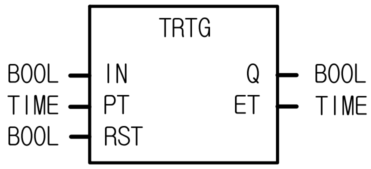
(4) If input contact %IX1.1.5 is 1, TIMER_OK and ET_TIME are all cleared.



☆ **Note**

TP_UINT Function Block keeps operating after the contact is on until its operation is complete. In case of a variable using array index, array index error occurs only when the contact is on. Therefore, TP_UINT Function Block does not produce any array index error as long as the contact is off although function block is operating.

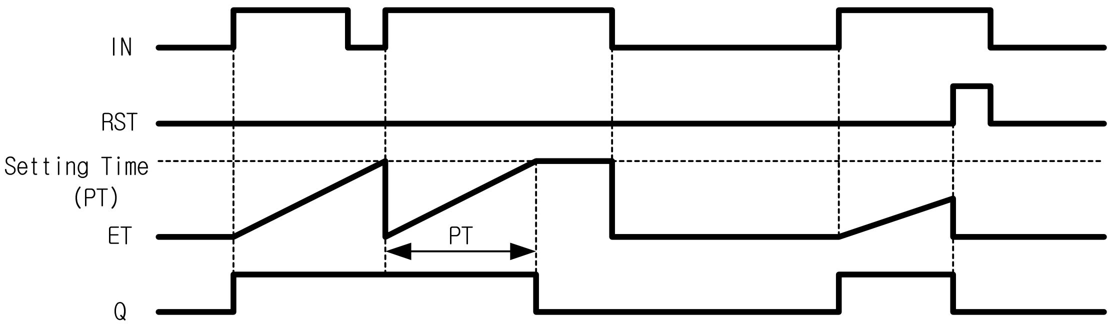
TRTG	**Retriggerable Timer**	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** IN: operation condition for Timer PT: preset time RST: reset **Output** Q: timer output ET: elapsed time

■ **Function**

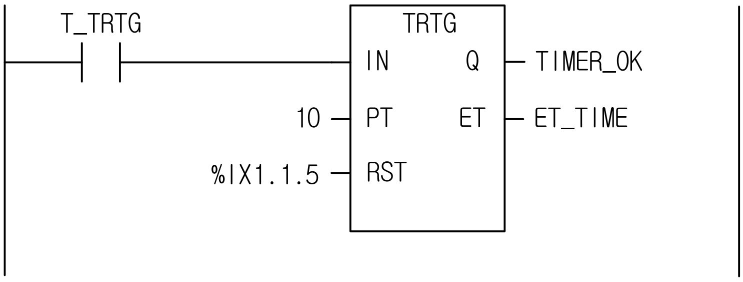
(1) Q is 1 as soon as IN becomes 1. And if elapsed time reaches preset time, timer output Q is 0.
(2) If IN turns on again before elapsed time reaches preset time, then elapsed time is set as 0 and increased again. And if it reaches PT, Q is 0.
(3) If RST is 1, timer output Q and elapsed time ET are 0.

■ **Time Chart**



■ **Program Example**

1. LD



2. ST

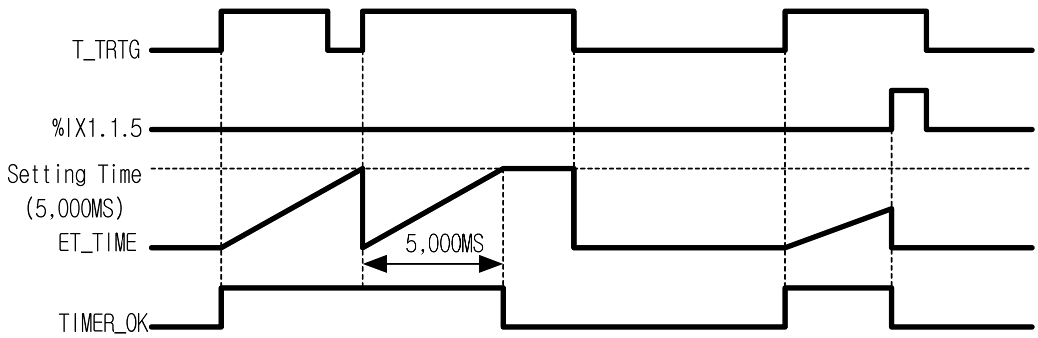
INST_TRTG(IN:=T_TRTG, PT:=10, RST:=%IX1.1.5, Q=>TIMER_OK, ET=>ET_TIME);

(1) TIMER_OK is 1 during 10 seconds after input variable T_TRTG becomes 1 from 0. If T_TRTG becomes 1 from 0 after timer executes, ET_TIME is set as 0 and increased again.

(2) TIMER_OK is 1 during 10 seconds even when T_TRTG becomes 0 from 1.

(3) ET_TIME value increases and stops at T#10S. And it is 0 when T_TRTG is 0.

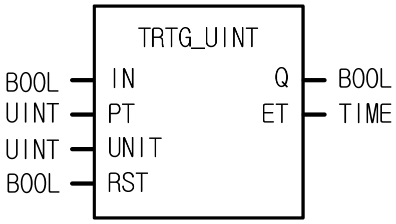
(4) If input contact %IX1.1.15 is 1, TIMER_OK and ET_TIME are all cleared.



☆ **Note**

TRTG Function Block keeps operating after the contact is on until its operation is complete. In case of a variable using array index, array index error occurs only when the contact is on. Therefore, TRTG Function Block does not produce any array index error as long as the contact is off although function block is operating.

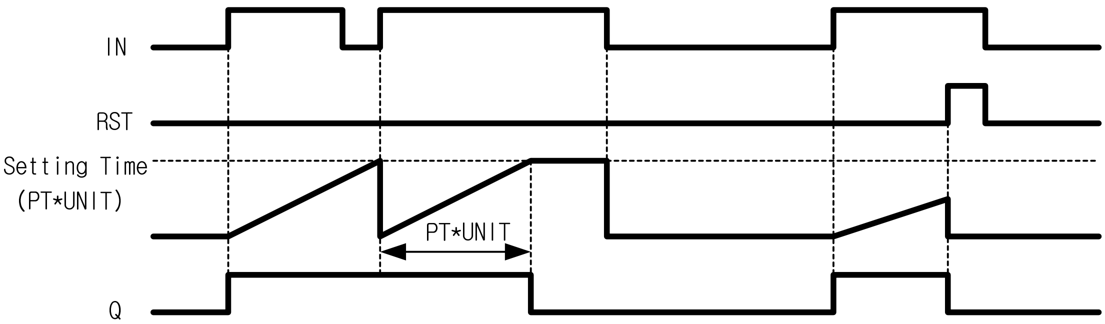
TRTG_UINT	Retriggerable Timer with Integer setting	
	Availability	XGI, XGR, XEC, XMC
	Flags	

Function Block	Description
	**Input** IN: operation condition for Timer PT: preset time UNIT: time unit of setting time RST: reset **Output** Q: timer output ET: elapsed time

■ **Function**

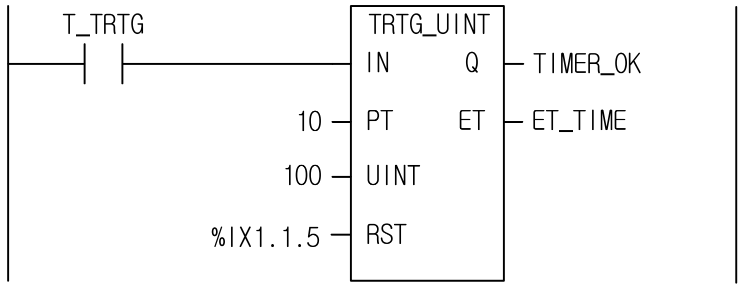
(1) Q is 1 as soon as IN becomes 1. And if elapsed time reaches preset time, timer output Q is 0.
(2) If IN turns on again before elapsed time reaches preset time, then elapsed time is set as 0 and increased again. And if it reaches PT, Q is 0.
(3) If RST is 1, timer output Q and elapsed time ET are 0.
(4) Preset time is PT x UNIT[ms].

■ **Time Chart**



■ **Program Example**

1. LD



2. ST

INST_TRTG_UINT(IN:=T_TRTG, PT:=10, UNIT:=100, RST:=%IX1.1.5, Q=>TIMER_OK, ET=>ET_TIME);

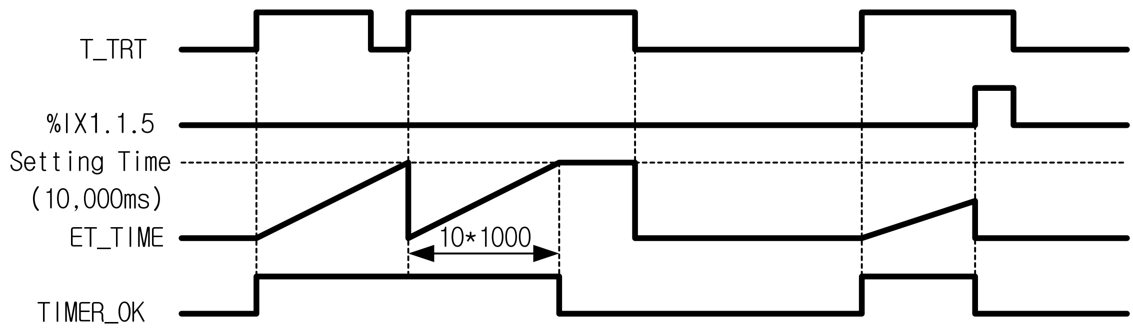
(1) Preset time is PT x UNIT[ms] = 10 x 1000[ms] = 10[s].

(2) TIMER_OK is 1 during 10 seconds after input variable T_TRTG becomes 1 from 0. If T_TRTG becomes 1 from 0 after timer executes, ET_TIME is set as 0 and increased again.

(3) TIMER_OK is 1 during 10 seconds even when T_TRTG becomes 0 from 1.

(4) ET_TIME value increases and stops at 10000. And it is 0 when T_TRTG is 0.

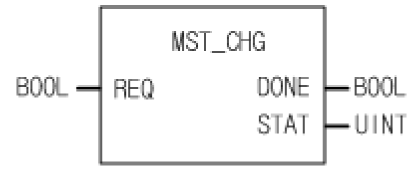
(5) If input contact %IX1.1.5 is 1, TIMER_OK and ET_TIME are all cleared.



☆ **Note**

TRTG_UINT Function Block keeps operating after the contact is on until its operation is complete. In case of a variable using array index, array index error occurs only when the contact is on. Therefore, TRTG_UINT Function Block does not produce any array index error as long as the contact is off, although function block is operating.

MST_CHG	Converting master by program	
	Availability	XGR
	Flags	_MASTER_CHG

Function Block	Description
	**Input** REQ : requests converting master by program **Output** DONE : keeps on after conversion STAT : indicates result. 0 means no error.

■ **Function**

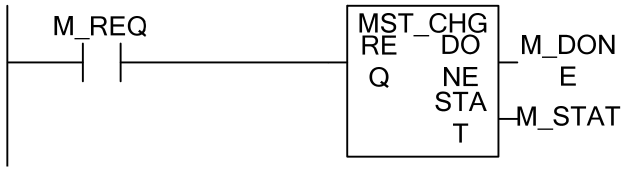
(1) If REQ (requests converting master by program) becomes 0 → 1, master is converted after finishing currently executed scan.
(2) DONE keeps on from when master is converted until REQ becomes off.
(3) STAT yields the following information after finishing execution of FB
- 0 : normal
- 1 : stand - by CPU power is off
- 2 : stand - by CPU power is stop
- 3 : stand - by CPU power is error
- 4 : Online Editing status

■ **Flag**

Flag	Description
_MASTER_CHG	Write-able bit flag In case of On, master is converted and flag becomes off.

■ **Program example**

1. LD



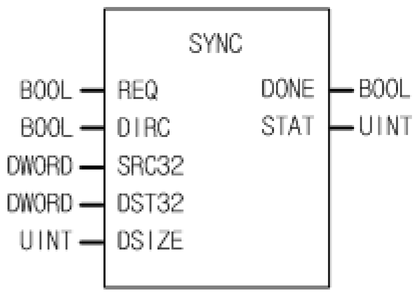
2. ST

INST_MST_CHG(REQ:=M_REQ, DONE=>M_DONE, STAT=>M_STAT);

(1) M_REQ becomes 0→1, master is converted.

(2) After conversion, M_DONE becomes on. If error occurs, error code is displayed in M_STAT.

SYNC	Synchronizing data between master CPU and stand-by CPU	
	Availability	XGR
	Flags	_MASTER_CHG

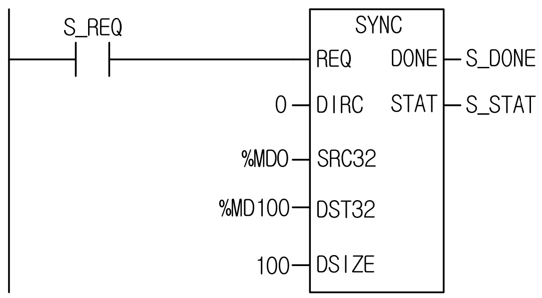
Function Block	Description
	**Input** REQ : requests execution of FB DIRC : 0: synchronizes data of master CPU to stand-by CPU 1: synchronizes data of stand-by CPU to master CPU SRC32 : direct variable to send data. DWORD type DST32 : direct variable to receive data. DWORD type DSIZE : number of DWORD data to synchronize **Output** DONE : in case of normal execution, on STAT : indicates result of execution. 0 means no error

■ **Function**

(1) It is used to synchronize device area between master CPU and stand-by CPU.
(2) If DIRC variable is off, DWORD data as many as number set in DSIZE are moved promptly from master CPU's device set in SRC32 to stand-by CPU's device set in DST32
(3) If DIRC variable is on, DWORD data as many as number set in DSIZE are moved promptly from stand-by CPU's device set in SRC32 to master CPU's device set in DST32
(4) Only direct variable can be declared in the location of SRC32 and DST32.
(5) Synchronization is done tough stand-by CPU is STOP, ERROR status.
(6) STAT yields the following information after finishing execution of FB
  - 0 : normal
  - 1 : device area of destination is exceeded when moving DWORD data
  - 2 : There is no stand-by CPU or SYNC FB can not be executed.

■ **Program example**

1. LD



2. ST

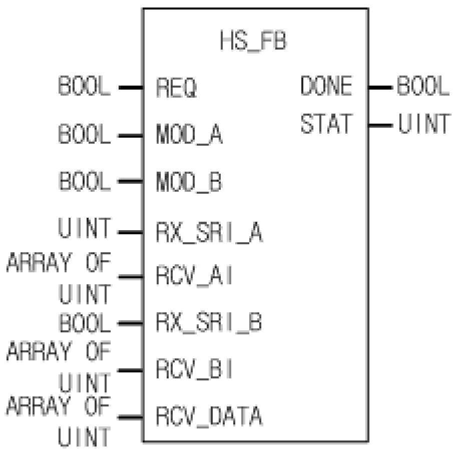
INST_SYNC(REQ:=S_REQ, DIRC:=0, SRC32:=%MD0, DST32:=%MD100, DSIZE:=100, DONE=>S_DONE, STAT=>S_STAT);

(1) If S_REQ becomes 0→1, data synchronization executes between master CPU and stand-by CPU

(2) 200 DWORD data is copied from %MD0 of master CPU to %MD100 of stand-by CPU.

(3) After synchronization, S_DONE becomes on. If error occurs, error code is displayed in S_STAT.

HS_FB	Synchronizing data between master CPU and stand-by CPU	
	Availability	XGI, XGR
	Flags	_HSn_STATEm [n:1~12, m:0~127]

Function Block	Description
	**Input** REQ : requests execution of FB MOD_A: HS link STATE flag of A side MOD_B: HS link STATE flag of B side RX_SRI_A: SEQ no. of A side RCV_AI: array variable to save A side data RX_SRI_B: B side SEQ no. RCV_BI: array variable to save B side data RCV_DATA: array variable to save input data **Output** DONE : in case of normal execution, on STAT : indicates result of execution. 0 means no error

■ **Function**

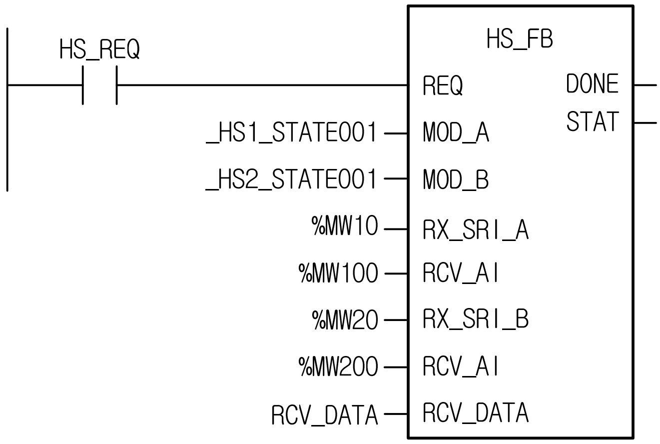
(1) If REQ of FB for executing redundant HS link service becomes 0 → 1, instruction is executed.
(2) DONE is kept on until REQ is off.
(3) Input HS link flag (_HSn_STATEm: total status display flag) into MOD_A, MOD_B according to block index and parameter no. of HS link set in XG-PD.
(4) Set SEQ number increased by one every scan at transmission side
(5) Input SEQ no. storage area set in XG-PD into RX_SRI_A, RX_SRI_B (SEQ no. uses 1 WORD).
(6) Input DATA storage area set in XG-PD into RCV_AI, RCV_BI.
(7) Input data storage area according to array type and number set in RCV_AI, RCV_BI.
(8) STAT provides the following information during execution.
 (1) 0 : Normal
 (2) 1 : The number of array of input side is different (RCV_AI, RCV_BI, RCV_DATA)
 (3) 2 : HS links of A/B side are in error

■ **Related flag**

Flag	Desciprition
_HSn_STATEm [n:1~12, m:0~127]	Total status display of HS link Nth Mth block

■ **Program example**

1. LD



2. ST

INST_HS_FB(REQ:=HS_REQ, MOD_A:=_HS1_STATE001, MOD_B:=_HS2_STATE001, RX_SRI_A:=%MW10, RCV_AI:=%MW100, RX_SRI_B:=%MW20, RCV_BI:=%MW200, RCV_DATA:=RCV_DATA);

(1) If HS_REQ becomes 0→1, HS_FB executes.

(2) SEQ no. of A side is received into %MW10 and SEQ no. of B side is received into %MW20. (Set in XG-PD)

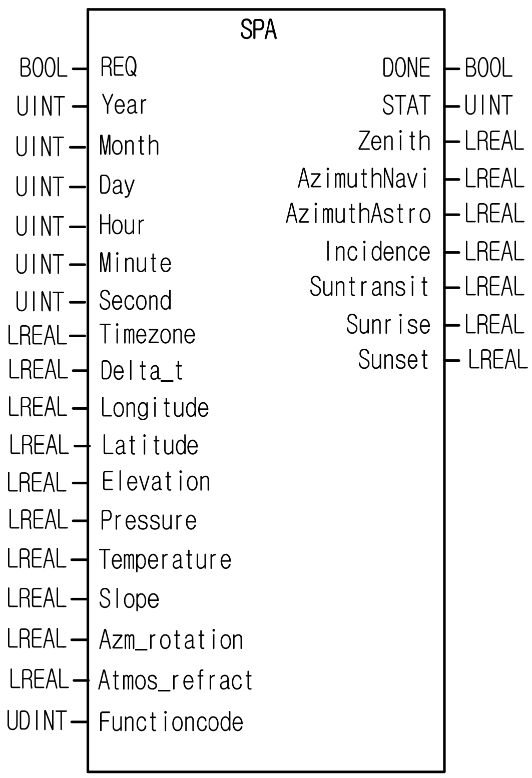
(3) Data of A side is received into %MW100 and data of B side is received into %MW200. (Set in XG-PD)

(4) In case communication module error of A side occurs, B side data is saved in RCV_DATA.

(5) In case communication module error of B side occurs, A side data is saved in RCV_DATA.

SPA	Applied model	Occurrence flag
Solar tracking algorithm	**XEC (SU, H, U, XEMH2, XEMHP)**	**-**

Function block	Explanation

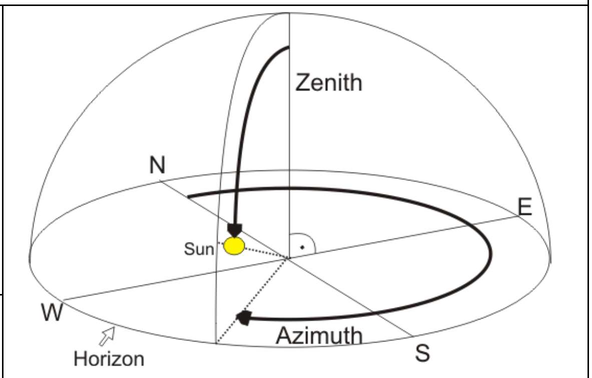


input REQ: Execution of Function Block at Rising Edge
- Year: year
- Month: month
- Day: days
- Hour: hour
- Minute: minute
- Second: second
- Timezone: Local time zone
- Delta_t: TT-UT
- Longitude: Local longitude
- Latitude: Local latitude
- Elevation: Local altitude
- Pressure: Annual average pressure
- Temperature: Average annual temperature
- Slope: Surface slope based on horizontal plane
- Azm_rotation: Rotational azimuth
- Atmos_refract: Atmospheric refraction angle
- Functioncode: select function

output DONE: Outputs 1 if SPA command is normally executed
- STAT: Error code in case of error
- Zenith: Zenith angle
- AzinuthNavi: azimuth
- AzinuthAstro: azimuth
- Incidence: angle of incidence
- Suntransit: Culmination of the Sun
- Sunrise: Sunrise time
- Sunset: Sunset time

■ **Detailed input / output**

<table>
<tr><th>division</th><th>Contents</th><th colspan="2">Detailed description</th></tr>
<tr><td rowspan="19">input</td><td>Year</td><td colspan="2">Year (> 6000)</td></tr>
<tr><td>Month</td><td colspan="2">Month (1 to 12)</td></tr>
<tr><td>Day</td><td colspan="2">Days (1 to 31)</td></tr>
<tr><td>Hour</td><td colspan="2">Hour (0-24)</td></tr>
<tr><td>Minute</td><td colspan="2">Minute (0 ~ 59)</td></tr>
<tr><td>Second</td><td colspan="2">Seconds (0 to 59)</td></tr>
<tr><td>Timezone</td><td colspan="2">Local time zone (difference from Greenwich (London) Standard Time)</td></tr>
<tr><td>Delta_t</td><td colspan="2">Difference between Earth Rotation Time and Ground Time
Delta_t = Terrestrial Time (TT) - Universal Time (UT) difference
[unit: Seconds]</td></tr>
<tr><td>Longitude</td><td>Local longitude
[unit: Degrees]</td><td rowspan="2">Yes)<table><tr><th></th><th>Longitude</th><th>Latitude</th></tr><tr><td>Sydney, Australia</td><td>151.2 [deg.]</td><td>-33.9 DEG</td></tr><tr><td>New York, USA</td><td>-74.0 [deg.]</td><td>40.7 [deg.]</td></tr><tr><td>London, England</td><td>-0.1 °</td><td>51.5 DEG</td></tr><tr><td>Seoul, South Korea</td><td>127 °</td><td>37.6 [deg.]</td></tr></table></td></tr>
<tr><td>Latitude</td><td>Local latitude
[unit: Degrees]</td></tr>
<tr><td>Elevation</td><td colspan="2">Area altitude[Unit: Meters]</td></tr>
<tr><td>Pressure</td><td colspan="2">Average annual pressure [Unit: Millibars]</td></tr>
<tr><td>Temperature</td><td colspan="2">Average annual temperature [Unit: Degrees Celsius]</td></tr>
<tr><td>Slope</td><td colspan="2">Surface slope based on horizontal plane [Unit: Degrees]</td></tr>
<tr><td>Azm_rotation</td><td colspan="2">Rotating azimuth [Unit: Degrees]</td></tr>
<tr><td>Atmos_refreact</td><td colspan="2">Atmospheric Refraction [Unit: (Degrees)
- Standard value: 0.5667 °</td></tr>
<tr><td>Functioncode</td><td colspan="2">Select function
1. Solar zenith angle / azimuth calculation
2. Solar zenith angle / Azimuth calculation + Incident angle calculation
3. Solar zenith angle / Azimuth calculation + Sun sunrise / Sunset / Moon hour calculation
4. Full function execution (1 to 3)</td></tr>
</table>

division	Contents	Detailed description
Print	Zenith	The zenith of the sun: [unit: Degrees] Definition of the angle between the connecting line of the sun and the station 
	Azinuthnavi	Azimuth of the sun [unit: Degrees] (North = 0 °, east = 90 °, south = 180 °, west = 270 °)
	AzinuthAstro	Azimuth of the Sun (Azimuth-180 ° = AzinuthAstro) [unit: Degrees]
	Incidence	Surface and incident angle of the sun [unit: Degrees]

■ **Error**

If the input parameter is out of the allowable range, the following error may occur.

STAT	Contents	Detailed description
0	Normal performance	Command execution complete
1	Year setting error	Occurs when a value other than Year (0 ~ 6000) is set.
2	Month setting error	Occurs when a value other than Month (1 to 12) is set.
3	Setting error	Occurs when a value other than Day (1 ~ 31) is set.
4	Time setting error	Occurs when a value other than Hour (0 to 24) is set.
5	Minute setting error	Occurs when a value other than Minute (0 ~ 59) is set.
6	Second setting error	Occurs when a value other than Second (0 ~ 59) is set.
7	Delta_t setting error	Occurs when a value other than Delta_t (-8000 ~ 8000) is set.
8	Timezone setting error	Occurs when a value other than Timezone (-18 ~ 18) is set.
9	Longitude setting error	Occurs when a value other than Longitude (-180 ~ 180) is set.
10	Latitude setting error	Occurs when a value other than Latitude (-90 ~ 90) is set.
11	Elevation setting error	Occurs when setting the Elevation value (less than -6500000)
12	Pressure setting error	Occurs when a value other than Pressure (0 ~ 5000) is set.
13	Temperature setting error	Occurs when setting a value other than Temperature (-273 ~ 6000)
14	Slope setting error	Occurs when setting a value other than Slope (-360 ~ 360)
15	Azm_rotation setting error	Occurs when a value other than Azm_rotation (-360 ~ 360) is set.

STAT	Contents	Detailed description
16	Atomos_refract setting error	Occurs when a value other than Atomos_refract (-5 to 5) is set.
17	Functioncode setting error	Occurs when setting a value other than Functioncode (0 ~ 3)

■ **Features**

1. You can estimate the solar zenith angle, azimuth, angle of incidence, and solar time in the local area with the SPA command.

2. SPA commands are available only for XECSU, XECH, XECU, XEMH2, and XEMHP among the XEC models.

3. This algorithm is based on the technical report (NREL / TP-560-34302) of the National Renewable Energy Laboratory (NREL) of the United States. The solar angle error is +/- 0.0003 °.

4. You can set the command time input value through the PLC clock information flag area. (See Program Example 1)

   (XECU, XEMH2, XEMHP: RTC built-in, XECSU: Optional board mounting required.)
5. When external clock data is used, it is necessary to convert it to the command input data type.

6. Through the type conversion instruction, Suntransit, Sunrise, and Sunset output values can be converted to clock data types. (See Program Example 2)

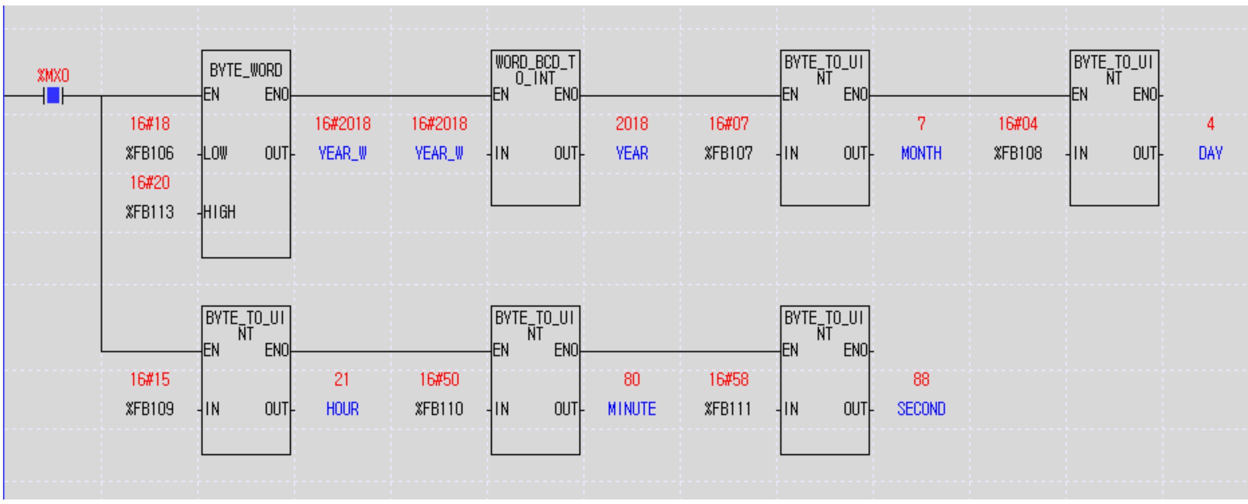
7. DONE is set to 1 when command execution is completed without error, and output value is updated according to Functioncode setting value. (1Scan)

8. If an error occurs, the previous output value is maintained, but DONE is set to 0 and STAT is output to the error number.

■ **Program Example**

(1) Time data setting using PLC clock flag value

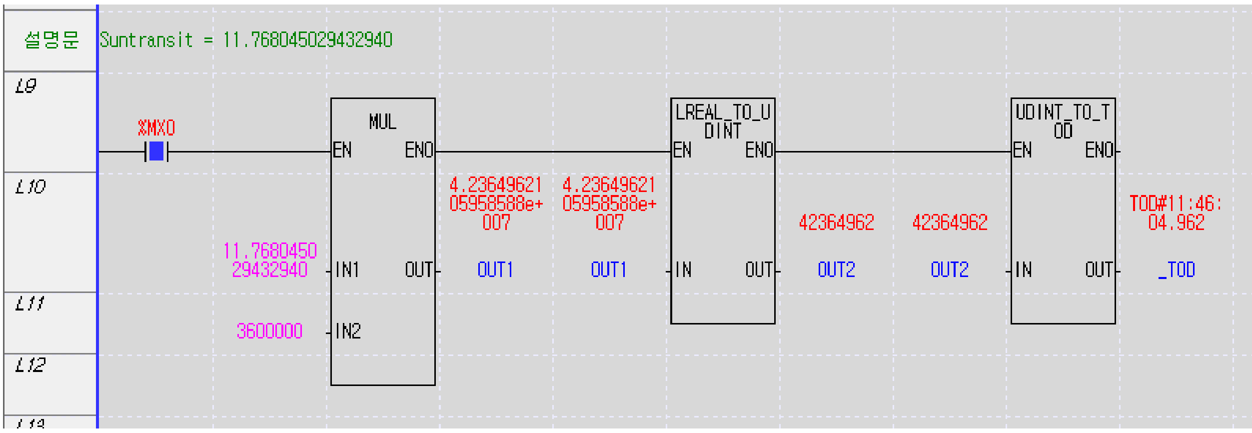
- When input condition% MX0 is On, type conversion instruction is executed.
- Converts the PLC clock flags (% FW53 to% FW56) to YEAR, MONTH, DAY, HOUR, MINUTE and SECOND respectively according to the SPA input data type.



(2) Solar time conversion through type conversion instruction

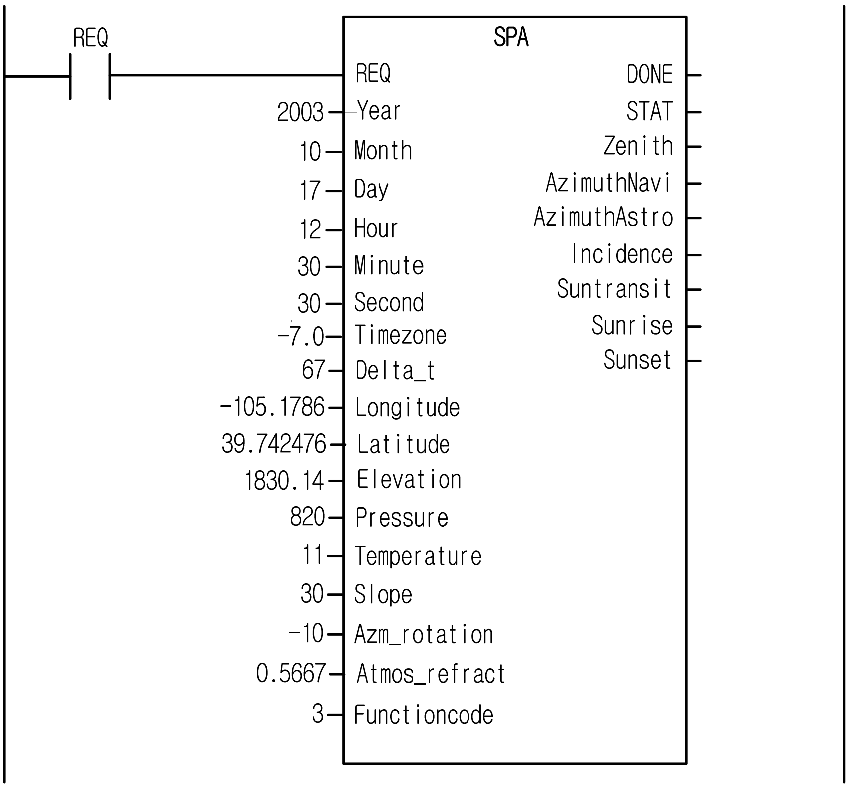
- When input condition% MX0 is On, the type conversion instruction is executed.
- You can multiply 3600000 by the output time value (LREAL data type) and execute the conversion instruction to check

the value by clock data type. (Final conversion value: 11:49:04)



(3) Executing a command

- REQ is Off → If it is On, SPA function block is executed. DONE is set to 1 after completion of command execution and output value is updated.



Chapter 11. Communication and Special Function Blocks

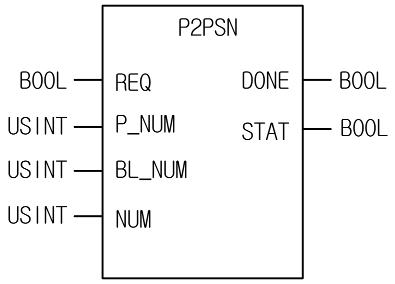
This chapter describes communication function blocks, special function blocks, motion control function blocks and positioning function blocks.

For the details of communication function blocks, refer to User's Manual about each communication block. For the directions of special function blocks, motion control function blocks and positioning function blocks, refer to User's Manual of each special module, motion control module and positioning module.

11.1 Communication Function Blocks

It describes each communication function block.

P2PSN	**Station No. setting**	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: to execute the function block P_NUM: P2P number BL_NUM: block number NUM: station number **Output** DONE: maintains 1 after the first operation STAT: completion and ERR info

■ **Function**

(1) You can change the station number of P2P destination while running by using P2PSN instruction.
(
(2) Change the block station number of P2P BL_NUM block of P_NUM to NUM.
Communication modules: FDEnet, Cnet.

■ **Error**

1. If an error occurs, it displays the error number in STAT.

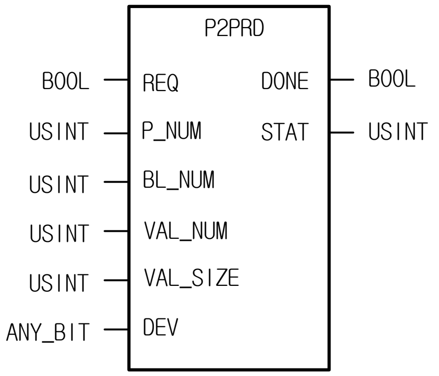
STAT_NUM	Message	Description
1	P2P no. setting	If a value except P_NUM(1~8) is set
2	Block No. setting	If a value except BL_NUM(0~63) is set < In case of Cnet, 0~31 >
4	No slot	-
5	Module inconsistency	Not a communication module
6	Module inconsistency	communication module not available in the instruction
7	Error of station No. setting	It is occurred, when it is set out of value NUM(0~63) < In case of Cnet, 0~31 >

■ **Program example**

1. ST

```
INST_P2PSN(REQ:=REQ_BOOL, P_NUM:=P_NUM_USINT, BL_NUM:=BL_NUM_USINT, NUM:=NUM_USINT,
DONE=>DONE_BOOL, STAT=>STAT_USINT);
```

P2PRD	Read area setting	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block P_NUM: P2P number BL_NUM: block number VAL_NUM: variable number VAL_SIZE: variable size DEV: device(input only for a direct variable) **Output** DONE: maintains 1 after the first operation STAT: completion and ERR info

ANY Type Variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	DEV	○	○	○	○	○															

■ **Function**

(1) P2PRD instruction changes the variable size and READ device area of P2P parameter block.
(both individual/continuous reads are changeable)
(2) After designating P2P parameter, block and variable by using P_NUM, BL_NUM, VAL_NUM, it changes the variable size and device to VAL_SIZE(if continuous, VAL_SIZE means variable size and if individual, it means the size of variable type), where DEV can be input only for a direct variable(ex, %MW100).
Communication modules: FEnet, FDEnet, Cnet.

■ **Error**

If it is out of the allowable scope of P2P parameter set in PD, the error number occurs as follows.

STAT	Message	Description
1	P2P number setting error	If a value except P_NUM(1~8) is set
2	Block number setting error	If a value except BL_NUM(0~63) is set < In case of Cnet, 0~31 >
3	Variable number setting error	If a variable number not allowed in P2P parameter set in PD is input
4	No slot	-
5	Module inconsistency	No communication module

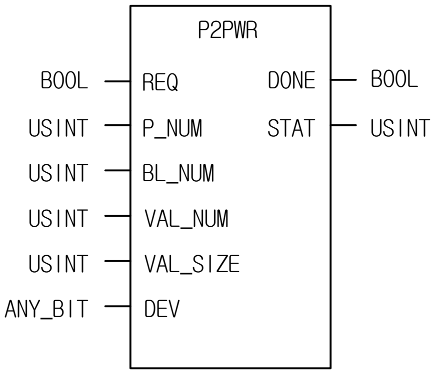
STAT	Message	Description
6	Module inconsistency	Communication module not available in the instruction
10	MODBUS setting error	MODBUS offset can not be input(ex, h10000). Because DEV can be input only for a direct variable
11	Variable size setting error	If a variable size not allowed in P2P parameter set in PD is input
12	Data type setting error	If a variable type not allowed in P2P parameter set in PD is input

■ **Program example**

ST

```
INST_P2PRD_BOOL(REQ:=REQ_BOOL, P_NUM:=P_USINT, BL_NUM:=BL_USINT, VAL:=VAL_USINT,
VAL_SIZE:=SIZE_UINT, DEV_NUM:=DEV_BOOL, DONE=>DONE_BOOL, STAT=>STAT_USINT);
```

P2PWR	Write area setting	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block P_NUM: P2P number BL_NUM: block number VAL_NUM: variable number VAL_SIZE: variable size DEV: device(input only for a direct variable) **Output** DONE: maintains 1 after the first operation STAT: completion and ERR info

ANY Type Variable	Variable	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	DEV	○	○	○	○	○															

■ **Function**

(1) P2PRD instruction changes the variable size and WRITE device area of P2P parameter block.
(both individual/continuous reads are changeable)
(2) After designating P2P parameter, block and variable by using P_NUM, BL_NUM, VAL_NUM, it changes the variable size and device to VAL_SIZE(if continuous, VAL_SIZE means variable size and if individual, it means the size of variable type), where DEV can be input only for a direct variable(ex, %MW100).
Communication modules: FEnet, FDEnet, Cnet.

■ **Error**

If it is out of the allowable scope of P2P parameter set in PD, the error number occurs as follows.

STAT	Message	Description
1	P2P number setting error	If a value except P_NUM(1~8) is set
2	Block number setting error	If a value except BL_NUM(0~63) is set <In case of Cnet, 0~31>
3	Variable number setting error	If a variable number not allowed in P2P parameter set in PD is input
4	No slot	-

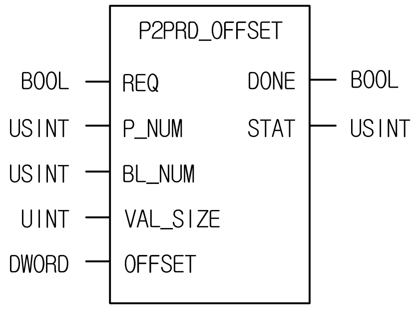
STAT	Message	Description
5	Module inconsistency	No communication module
6	Module inconsistency	Communication module not available in the instruction
10	MODBUS setting error	MODBUS offset can not be input(ex, h10000). Because DEV can be input only for a direct variable
11	Variable size setting error	If a variable size not allowed in P2P parameter set in PD is input
12	Data type setting error	If a variable type not allowed in P2P parameter set in PD is input

■ **Program example**

ST

```
INST_P2PWR_BOOL(REQ:=REQ_BOOL, P_NUM:=P_USINT, BL_NUM:=BL_USINT, VAL:=VAL_USINT,
VAL_SIZE:=SIZE_UINT, DEV_NUM:=DEV_BOOL, DONE=>DONE_BOOL, STAT=>STAT_USINT);
```

P2PRD_OFFSET	Read area offset setting	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block P_NUM: P2P number BL_NUM: block number VAL_SIZE: variable size OFFSET: offset value **Output** DONE: maintains 1 after the first operation STAT: completion and ERR info

■ **Function**

(1) P2PRD_OFFSET instruction changes the read area's offset value and READ data size of P2P parameter block. (both individual/continuous reads are changeable)
(2) After designating P2P parameter, block and variable by using P_NUM, BL_NUM, it changes read area's offset value to read data size(VAL_SIZE) and read area offset(OFFSET). (when it is set as individual read, set VAL_SIZE=1) Communication modules: FEnet, Cnet.

(3) Range of read area's offset value

Data type	P2P mode	Maximum data size		OFFSET range	remark
		Modbus ASCII	Modbus TCP/RTU		
BOOL	READ	976	2000	0x00000 ~ 0x1FFFF	-
	WRITE	944	1968	0x00000 ~ 0x0FFFF	P2PWR_OFFSET use
WORD	READ	61	125	0x30000 ~ 0x4FFFF	-
	WRITE	59	123	0x40000 ~ 0x4FFFF	P2PWR_OFFSET use

* In case of read mode, bit read area(0x1XXXX), it can access to P2P server's bit write area(0x0XXXX), word read area(0x3XXXX), word write area(0x4XXXX)

■ **Error**

If it is out of the allowable scope of P2P parameter set, the error number occurs as follows.

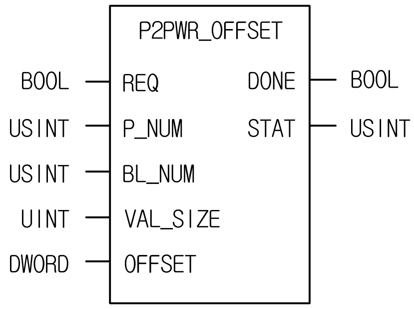
STAT	Message	Description
1	P2P number setting error	If a value except P_NUM(1~8) is set
2	Block number setting error	If a value except BL_NUM(0~63) is set
3	Variable number setting error	If a variable number not allowed in P2P parameter set is input
4	No slot	-
5	Module inconsistency	No communication module
6	Module inconsistency	Communication module not available in the instruction
10	MODBUS setting error	MODBUS offset can not be input(ex, h10000). Because DEV can be input only for a direct variable
11	Variable size setting error	If a variable size not allowed in P2P parameter set is input
12	Data type setting error	If a variable type not allowed in P2P parameter set is input
13	Offset setting error	If read area's offset value is exceed the range

■ **Program example**

ST

```
INST_P2PRD_OFFSET(REQ:=REQ_BOOL, P_NUM:=P_USINT, BL_NUM:=BL_USINT, VAL_SIZE:=SIZE_UINT,
OFFSET:=OFFSET_DWORD, DONE=>DONE_BOOL, STAT=>STAT_USINT);
```

P2PWR_OFFSET	Read area offset setting	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block P_NUM: P2P number BL_NUM: block number VAL_SIZE: variable size OFFSET: offset value **Output** DONE: maintains 1 after the first operation STAT: completion and ERR info

■ **Function**

(1) P2PWR_OFFSET instruction changes the write area's offset value and write data size of P2P parameter block. (both individual/continuous writes are changeable)
(2) After designating P2P parameter, block and variable by using P_NUM, BL_NUM, it changes write area's offset value to write data size(VAL_SIZE) and write area offset(OFFSET). (when it is set as individual write, set VAL_SIZE=1) Communication modules: FEnet, Cnet.

(3) Range of write area's offset value

Data type	P2P mode	Maximum data size		OFFSET range	remark
		Modbus ASCII	Modbus TCP/RTU		
BOOL	READ	976	2000	0x00000 ~ 0x1FFFF	-
	WRITE	944	1968	0x00000 ~ 0x0FFFF	P2PWR_OFFSET use
WORD	READ	61	125	0x30000 ~ 0x4FFFF	-
	WRITE	59	123	0x40000 ~ 0x4FFFF	P2PWR_OFFSET use

* In case of read mode, bit read area(0x1XXXX), it can access to P2P server's bit write area(0x0XXXX), word read area(0x3XXXX), word write area(0x4XXXX)

■ **Error**

If it is out of the allowable scope of P2P parameter set, the error number occurs as follows.

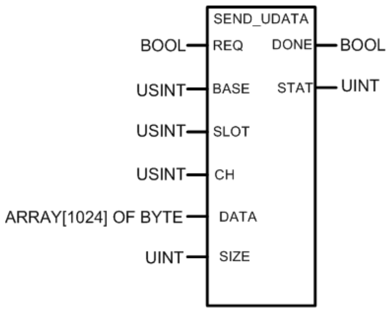
STAT	Message	Description
1	P2P number setting error	If a value except P_NUM(1~8) is set
2	Block number setting error	If a value except BL_NUM(0~63) is set
3	Variable number setting error	If a variable number not allowed in P2P parameter set is input
4	No slot	-
5	Module inconsistency	No communication module
6	Module inconsistency	Communication module not available in the instruction
10	MODBUS setting error	MODBUS offset can not be input(ex, h10000). Because DEV can be input only for a direct variable
11	Variable size setting error	If a variable size not allowed in P2P parameter set is input
12	Data type setting error	If a variable type not allowed in P2P parameter set is input
13	Offset setting error	If write area's offset value is exceed the range

■ **Program example**

ST

INST_P2PRD_OFFSET(REQ:=REQ_BOOL, P_NUM:=P_USINT, BL_NUM:=BL_USINT, VAL_SIZE:=SIZE_UINT, OFFSET:=OFFSET_DWORD, DONE=>DONE_BOOL, STAT=>STAT_USINT);

SEND_UDATA	User defined data send	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE : base number SLOT: slot number CH: channel(1 or 2) DATA: data area to send SIZE: data size to send **Output** DONE: maintains 1 after operation STAT: completion and ERR info

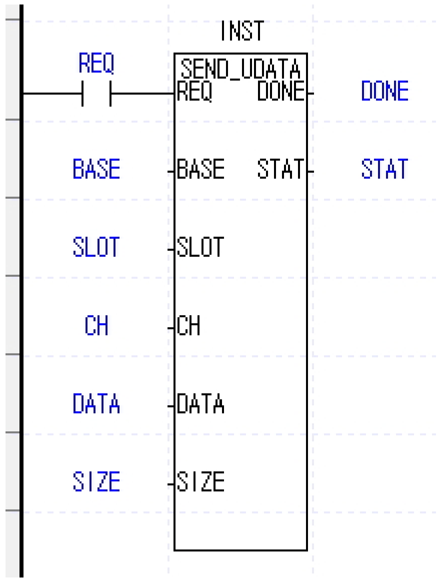
■ **Function**

(1) SEND_UDATA instruction sends user defined data(UDATA).
(2) DATA must be declared only ARRAY OF BYTE type.
(3) Array size is 1 ~ 1024 byte.
(4) Save to transmit buffer as number as SIZE from DATA[0]. (Limit of data size is 1024 at once)

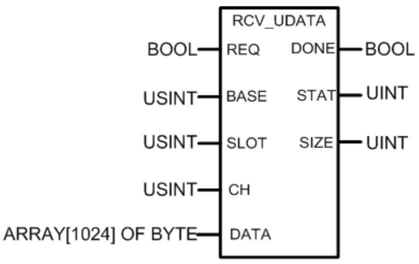
■ **Error**

STAT	Message	Description
0	Initial state	Initial state before instruction operation
1	No error	normal operation
2	Module setting error	Module is not installed or CNET module trouble
3	Channel setting error	Input range(1, 2) is exceeded
4	Array size error	Transmit data size exceed 1024
5	Parameter setting error	CNET module's parameter is not set as User defined or link enable is not set
6	Instruction timeout error	No response from module or maximum scan time is exceeded(10 scan)
7	Version mismatch error	XGI CPU version is under V3.9, XGR CPU version is under V2.6 or CNET module version is under V3.2

■ **Program example**



RCV_UDATA	User defined data receive	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE : base number SLOT: slot number CH: channel(1 or 2) DATA: data area to save **Output** DONE: maintains 1 after operation STAT: completion and ERR info SIZE: received data size

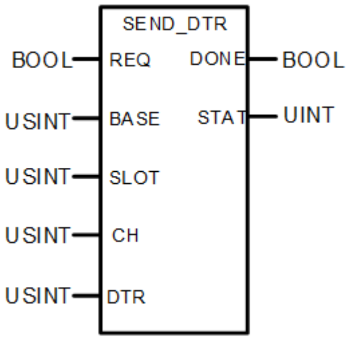
■ **Function**

(1) RCV_UDATA instruction saves received user defined data(UDATA) from CNET module.
(2) DATA must be declared only ARRAY OF BYTE type.
(3) Array size is 1 ~ 1024 byte.

■ **Error**

STAT	Message	Description
0	Initial state	Initial state before instruction operation
1	No error	normal operation
2	Module setting error	Module is not installed or CNET module trouble
3	Channel setting error	Input range(1, 2) is exceeded
4	Array size error	Transmit data size exceed 1024
5	Parameter setting error	CNET module’s parameter is not set as User defined or link enable is not set
6	Instruction timeout error	No response from module or maximum scan time is exceeded(10 scan)
7	Version mismatch error	XGI CPU version is under V3.9, XGR CPU version is under V2.6 or CNET module version is under V3.2

SEND_DTR	User defined data send	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE : base number SLOT: slot number CH: channel(1 or 2) DTR: 0 or 1 **Output** DONE: maintains 1 after operation STAT: completion and ERR info

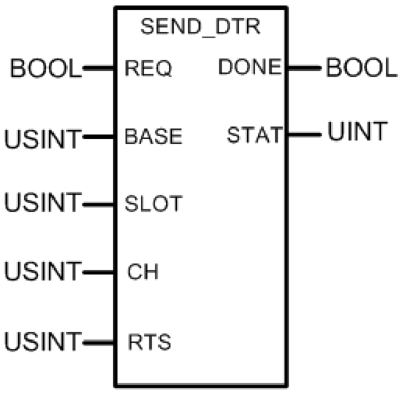
■ **Function**

(1)SEND_DTR instruction send DTR(Data Terminal Ready) signal that means communication ready complete.

■ **Error**

STAT	Message	Description
0	Initial state	Initial state before instruction operation
1	No error	normal operation
2	Module setting error	Module is not installed or CNET module trouble
3	Channel setting error	Input range(1, 2) is exceeded
4	DTR setting error	Input range(0, 1) is exceeded
5	Parameter setting error	CNET module's parameter is not set as User defined or link enable is not set
6	Instruction timeout error	No response from module or maximum scan time is exceeded(10 scan)
7	Version mismatch error	XGI CPU version is under V3.9, XGR CPU version is under V2.6 or CNET module version is under V3.2

SEND_RTS	User defined data send	
	Availability	XGI, XGR
	Flags	

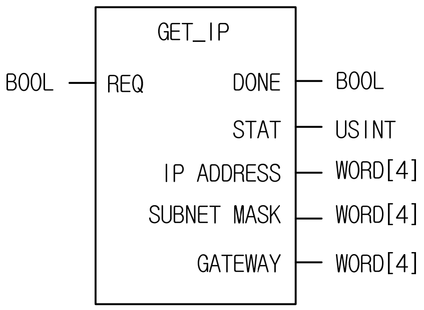
Function Block	Description
	**Input** REQ: requires to execute the function block BASE : base number SLOT: slot number CH: channel(1 or 2) RTS: 0 or 1 **Output** DONE: maintains 1 after operation STAT: completion and ERR info

■ **Function**

(1)SEND_RTS instruction send RTS(Request To Send) signal that means state of receive buffer.

■ **Error**

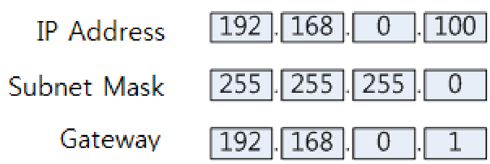
STAT	Message	Description
0	Initial state	Initial state before instruction operation
1	No error	normal operation
2	Module setting error	Module is not installed or CNET module trouble
3	Channel setting error	Input range(1, 2) is exceeded
4	RTS setting error	Input range(0, 1) is exceeded
5	Parameter setting error	CNET module's parameter is not set as User defined or link enable is not set
6	Instruction timeout error	No response from module or maximum scan time is exceeded(10 scan)
7	Version mismatch error	XGI CPU version is under V3.9, XGR CPU version is under V2.6 or CNET module version is under V3.2

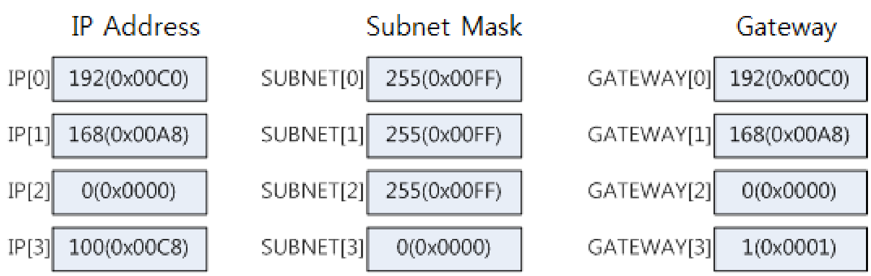
<table>
<tr><th><h1>GET_IP</h1></th><th colspan="2">Applied model</th><th>Occurrence flag</th></tr>
<tr><td>Read local Ethernet IP, SUBNET MASK, GATEWAY</td><td colspan="2">XGI-CPUUN</td><td>-</td></tr>
<tr><th>Function block</th><th colspan="3">Explanation</th></tr>
<tr><td>
</td><td colspan="3">Input REQ: Function block execution request

Output DONE: Maintain 1 after initial operation
STAT: Complete and ERR information
IP: Local Ethernet IP address
SUBNET MASK: Local Ethernet subnet mask
GATEWAY: Local Ethernet gateway</td></tr>
</table>

■ **Features**

1. The GET_IP command allows you to read the IP address, subnet mask, and gateway information of the local Ethernet.
2. Only available with XGI-CPUUN with local Ethernet.
3. After executing the command, the IP address of the local Ethernet is displayed as follows.





■ **Error**

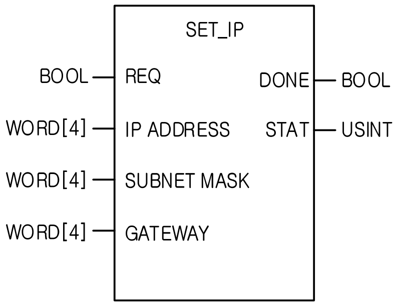
If the local Ethernet parameter is abnormal or the command is duplicated, the following error may occur.

STAT	Contents	Detailed description
0	Normal performance	Command execution complete
11	Above user setting value	User set IP / SUBNET / GATEWAY setting value is not valid
12	Above the default setting	Above existing local Ethernet parameter setting (Local Ethernet parameters have never been downloaded or parameter errors are present)
13	Duplicate request error	If the instruction is already being executed (The instruction can not be duplicated)
14	Timeout	Timeout processed because command execution is not completed

■ **Program Example**

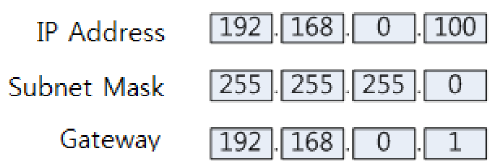
1. ST

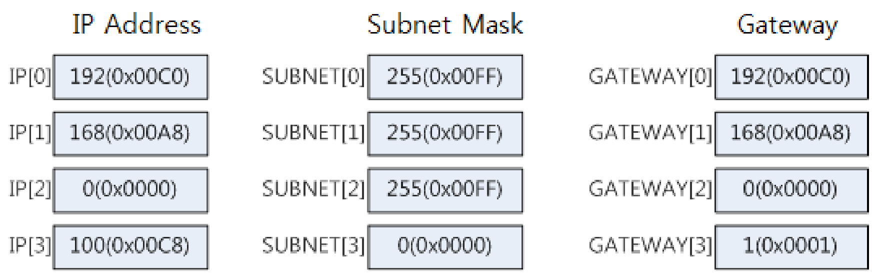
INST_GET_IP (REQ: REQ_BOOL, DONE => DONE_BOOL, STAT => STAT_USINT, IP => ARY_IP, SUBNET => ARY_SUBNET, GATEWAY => ARY_GATEWAY)

SET_IP	Applied model	Occurrence flag
Local Ethernet IP, SUBNET MASK, GATEWAY settings	**XGI-CPUUN**	-
Function block	Explanation	
	**Input** REQ: Function block execution request IP ADDRESS: Local ethernet IP address to set SUBNET MASK: Local ethernet subnet mask to set GATEWAY: Local ethernet gateway to set **Output** DONE: Maintain 1 after initial operation STAT: Complete and ERR information	

■ **Features**

1. The SET_IP command allows you to set the IP address, subnet mask, and gateway of the local Ethernet.
2. Only available with XGI-CPUUN with local Ethernet.
3. When setting the IP address, subnet mask, and gateway, you need to set the IP address, subnet mask, and gateway as shown below.





■ **Error**

If the local Ethernet parameter is abnormal or the command is duplicated, the following error may occur.

STAT	Contents	Detailed description
0	Normal performance	Command execution complete
11	Above user setting value	User set IP / SUBNET / GATEWAY setting value is not valid
12	Above the default setting	Above existing local Ethernet parameter setting (Local Ethernet parameters have never been downloaded or parameter errors are present)
13	Duplicate request error	If the instruction is already being executed (The instruction can not be duplicated)
14	Timeout	Timeout processed because command execution is not completed

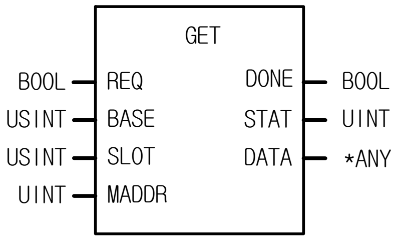
■ **Program Example**

1. ST

INST_SET_IP (REQ: = REQ_BOOL, IP: = ARY_IP, SUBNET: = ARY_SUBNET, GATEWAY: = ARY_GATEWAY, DONE => DONE_BOOL, STAT => STAT_USINT)

11.2 Special Function Block

GET	Read special module data	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: executes the function in case of 1 BASE: Base position setting SLOT: Slot position setting MADDR: Module address 512(h200) ~ 1023(h3FF) **Output** DONE: 1 output in case of normal execution STAT: Error information DATA: Data read from a module

*ANY: Among ANY types, WORD, DWORD, INT, UINT, DINT and UDINT types are available

■ **Function**

Read data from a configured special module.

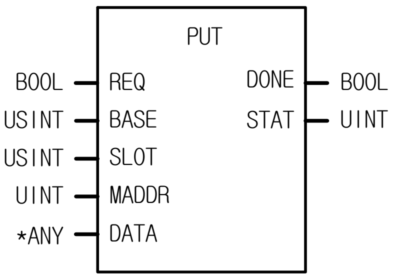
Function Block	Output(ANY) type	Description
GET_WORD	WORD	Read data as much as WORD from the configured module address (MADDR).
GET_DWORD	DWORD	Read data as much as DWORD from the configured module address (MADDR).
GET_INT	INT	Read data as much as INT from the configured Module address (MADDR).
GET_UINT	UINT	Read data as much as UNIT from the configured module address (MADDR).
GET_DINT	DINT	Read data as much as DINT from the configured module address (MADDR).
GET_UDINT	UDINT	Read data as much as UDINT from the configured module address (MADDR).

■ **Program example**

ST

```
INST_GET_WORD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT, DATA=>DATA_WORD);
```

PUT	Write data to a special module	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: execute the function in case of 1 BASE: Base position setting SLOT: Slot position setting MADDR: Module address DATA: data to save into a module **Output** DONE: 1 output in case of normal execution STAT: Error information

*ANY: Among ANY types, WORD, DWORD, INT, USINT, DINT and UDINT types are available

■ **Function**

Read data from the designated special module.

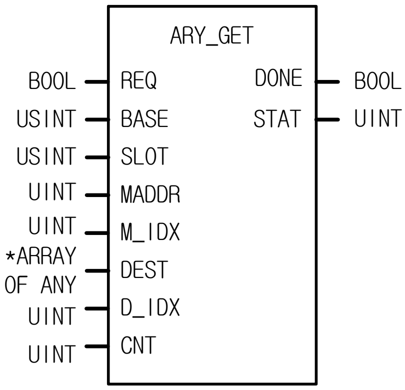
Function Block	Input(ANY) type	Description
PUT_WORD	WORD	Save WORD data into the configured module address (MADDR).
PUT_DWORD	DWORD	Save DWORD data into the configured module address (MADDR).
PUT_INT	INT	Save INT data into the configured module address (MADDR).
PUT_UINT	UINT	Save UNIT data into the configured module address (MADDR).
PUT_DINT	DINT	Save DINT data into the configured module address (MADDR).
PUT_UDINT	UDINT	Save UDINT data into the configured module address (MADDR).

■ **Program example**

ST

INST_PUT_WORD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT, DATA:=DATA_WORD, DONE=>DONE_BOOL, STAT=>STAT_UINT);

ARY_GET	Read special module data(Array)	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: execute the function in case of 1 BASE: Base position setting SLOT: Slot position setting MADDR: Module address M_IDX: distance away from MADDR DEST: array variable to save read data D_IDX: Start index of DEST variable CNT: Number of data to read **Output** DONE: 1 output in case of normal execution STAT: Error information

*ARRAY OF ANY: among ANY types, WORD, DWORD, INT, UINT, DINT and UDINT types are available

■ **Function**

Read data from the designated special module.

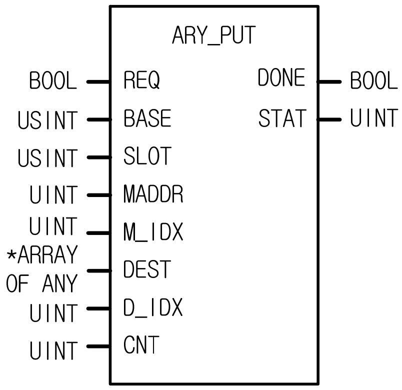
Function Block	Output(DEST) Type	Description
ARY_GET_WORD	WORD	Read data as much as CNT in WORD from the configured module address (MADDR)
ARY_GET_DWORD	DWORD	Read data as much as CNT in DWORD from the configured module address (MADDR)
ARY_GET_INT	INT	Read data as much as CNT in INT from the configured module address (MADDR).
ARY_GET_UINT	UINT	Read data as much as CNT in UINT from the configured module address (MADDR).
ARY_GET_DINT	DINT	Read data as much as CNT in DINT from the configured module address (MADDR).
ARY_GET_UDINT	UDINT	Read data as much as CNT in UDINT from the configured module address (MADDR).

■ **Program example**

ST

```
INST_ARY_GET_WORD (REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT,
MADDR:=MADDR_UINT, M_IDX:=M_UINT, DEST:=ARY_DEST, D_IDX:=D_UINT, CNT:=CNT_UINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

ARY_PUT	Write special module data(Array)	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: execute the function in case of 1 BASE: Base position setting SLOT: Slot position setting MADDR: Module address M_IDX: distance away from MADDR DEST: Data array variable to save D_IDX: Start index of DEST variable CNT: Number of data to read **Output** DONE: 1 output in case of normal execution STAT: Error information

*ARRAY OF ANY: among ANY types, WORD, DWORD, INT, UINT, DINT and UDINT types are available

■ **Function**

Read data from the designated special module.

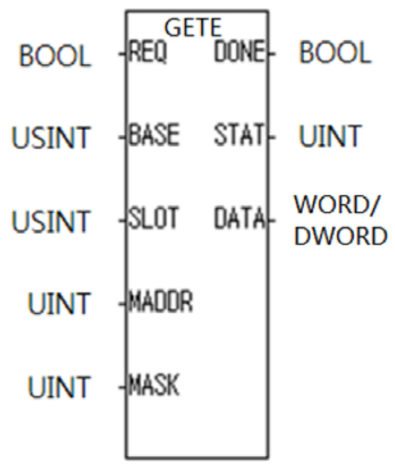
Function Block	Input(DEST) type	Description
ARY_PUT_WORD	WORD	Save data as much as CNT in WORD into the configured module address (MADDR)
ARY_PUT_DWORD	DWORD	Save data as much as CNT in DWORD into the configured module address (MADDR)
ARY_PUT_INT	INT	Save data as much as CNT in INT into the configured module address (MADDR).
ARY_PUT_UINT	UINT	Save data as much as CNT in UINT into the configured module address (MADDR)
ARY_PUT_DINT	DINT	Save data as much as CNT in DINT into the configured module address (MADDR)
ARY_PUT_UDINT	UDINT	Save data as much as CNT in LDINT into the configured module address (MADDR)

■ **Program example**

ST

INST_ARY_PUT_WORD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT, M_IDX:=M_UINT, DEST:=ARY_DEST, D_IDX:=D_UINT, CNT:=CNT_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);

GETE	Read special module data(Access upper word)	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: executes the function in case of 1 BASE: Base position setting SLOT: Slot position setting MADDR: Module address 0~1023 MASK: Word position setting 0(Lower word), 1(Upper word) **Output** DONE: 1 output in case of normal execution STAT: Error information DATA: Data read from a module(WORD/DWORD)

■ **Function**

1) Read data from a configured special module.
2) Select WORD / DWORD type according to data type.
3) Position of data selected according to MASK setting.
   0 -> Lower word of module address at MADDR
   1 -> Upper world of module address at MADDR

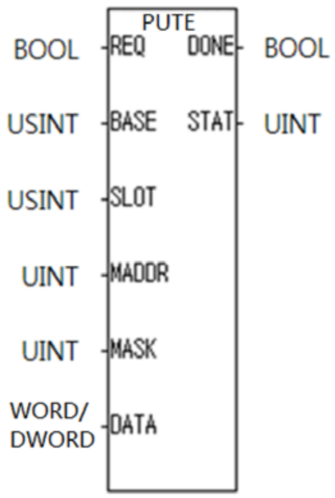
Function Block	Output type	Description
GETE_WORD	WORD	Read WORD data from the configured module address (MADDR).
GETE_DWORD	DWORD	Read DWORD data from the configured module address (MADDR).

■ **Program example**

ST

```
INST_GETE_WORD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT,
MASK:=MASK_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT, DATA=>DATA_WORD);
```

PUTE	Write data to a special module(Access upper word)	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: execute the function in case of 1 BASE: Base position setting SLOT: Slot position setting MADDR: Module address MASK: Word position setting 0(Lower word), 1(Upper word) DATA: data to save into a module(WORD/DWORD) **Output** DONE: 1 output in case of normal execution STAT: Error information

■ **Function**

1) Write data to the designated special module.
2) Select WORD or DWORD type according to data type.
3) Position of data selected according to MASK setting.
   0 -> Lower word of module address at MADDR
   1 -> Upper world of module address at MADDR

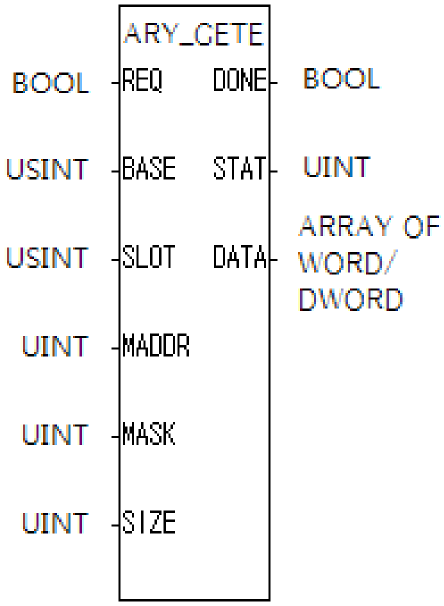
Function Block	Input type	Operation description
PUTE_WORD	WORD	Write WORD data at the designated module address (MADDR)
PUTE_DWORD	DWORD	Write DWORD data at the designated module address (MADDR)

■ **Program example**

ST

INST_PUTE_WORD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT, MASK:=MASK_UINT, DATA:=DATA_WORD, DONE=>DONE_BOOL, STAT=>STAT_UINT);

ARY_GETE	Read special module data(Array, Access upper word)	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: executes the function in case of 1 BASE: Base position setting SLOT: Slot position setting MADDR: Module address 0~1023 MASK: Word position setting 0(Lower word), 1(Upper word) SIZE: Quantity of data (1~64[WORD], 1~32[DWORD]) **Output** DONE: 1 output in case of normal execution STAT: Error information DATA: Data(Array) read from a module (WORD/DWORD)

■ **Function**

1) Read data as quantity user set from a configured special module.
2) Select WORD / DWORD type according to data type(Array).
3) Position of data selected according to MASK setting.
   0 -> Lower word of module address at MADDR
   1 -> Upper world of module address at MADDR

Function Block	Output Type	Description
ARY_GETE_WORD	WORD	Read data as much as SIZE in WORD from the configured module address (MADDR)
ARY_GETE_DWORD	DWORD	Read data as much as SIZE in DWORD from the configured module address (MADDR)

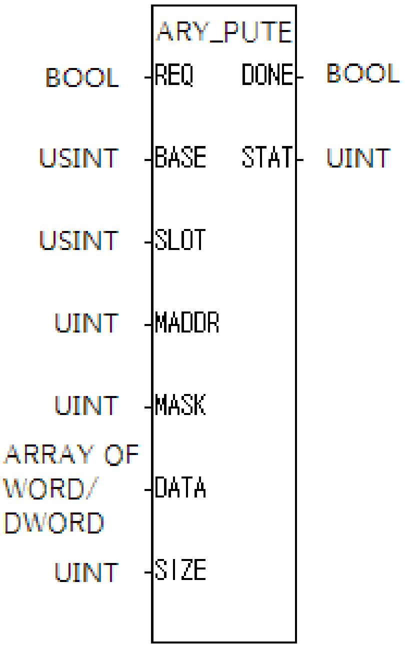
■ **Program example**

ST

```
INST_ARY_GETE_WORD (REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT,
MASK:=MASK_UINT, SIZE:=SIZE_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT, DATA:=ARY_DATA);
```

ARY_PUTE	Write special module data(Array, Access upper word)	
	Availability	XGI, XGR
	Flags	

Function Block	Description



Input REQ: execute the function in case of 1
BASE: Base position setting
SLOT: Slot position setting
MADDR: Module address
0~1023
MASK: Word position setting
0(Lower word), 1(Upper word)
DATA: Data(Array) to save into a module
(WORD/DWORD)
SIZE: Quantity of data
(1~64[WORD], 1~32[DWORD])

Output DONE: 1 output in case of normal execution
STAT: Error information

■ **Function**

1) Write data as quantity user set to the designated special module.
2) Select WORD / DWORD type according to data type(Array).
3) Position of data selected according to MASK setting.
 0 -> Lower word of module address at MADDR
 1 -> Upper world of module address at MADDR

.

Function Block	Input type	Description
ARY_PUTE_WORD	WORD	Save data as much as SIZE in WORD into the configured module address (MADDR)
ARY_PUTE_DWORD	DWORD	Save data as much as SIZE in DWORD into the configured module address (MADDR)

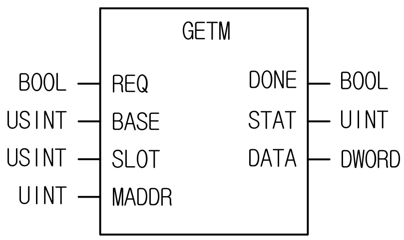
■ **Program example**

ST

INST_ARY_PUTE_WORD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT, MASK:=MASK_UINT, DATA:=ARY_DATA, DONE=>DONE_BOOL, STAT=>STAT_UINT);

11.3 Motion Control Function Block

GETM	Read motion control module data	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: execute the function in case of 1 BASE: Base position setting SLOT: Slot position setting MADDR: Module address 512(0x200) ~ 1023(0x3FF) **Output** DONE: 1 output in case of normal execution STAT: Error information DATA: Data read from a module

■ **Function**

Read data from the shared read memory address MADDR of the configured motion control module.

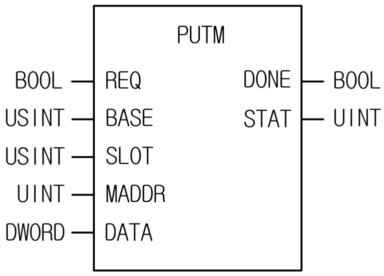
Function Block	Output(DATA) type	Description
GETM	DWORD	Read data as much as DWORD from the configured module address (MADDR).

■ **Program example**

ST

```
INST_GETM(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT, DATA=>DATA_DWORD);
```

PUTM	Write data into a special module(motion module)	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: execute the function in case of 1 BASE: Base position setting SLOT: Slot position setting MADDR: Module address 0(0x00) ~ 511(0x1FF) DATA: data to save into a module **Output** DONE: 1 output in case of normal execution STAT: Error information

■ **Function**

Save data into the shared write memory MADDR of the configured motion control module.

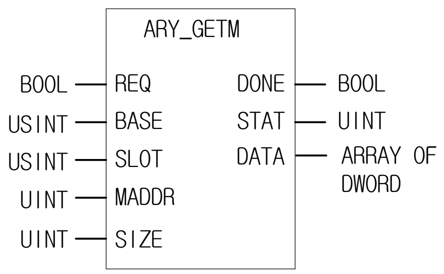
Function Block	DATA type	Description
PUTM	DWORD	Save DWORD data into the configured module address (MADDR).

■ **Program example**

ST

INST_PUTM(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT, DATA:=DATA_DWORD, DONE=>DONE_BOOL, STAT=>STAT_UINT);

ARY_GETM	**Read motion control module data (Array)**	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: execute the function in case of 1 BASE: Base position setting SLOT: Slot position setting MADDR: Address to start reading 512(0x200) ~ 1023(0x3FF) SIZE: Number of data to read (1 ~ 512) **Output** DONE: 1 output in case of normal execution STAT: Error information DATA: Array variable to save read data (ARRAY of DWORD)

■ **Function**

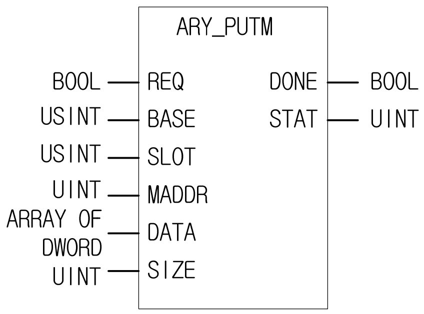
Read data as much as the size from the shared read memory MADDR of the configured motion control module.

■ **Program example**

ST

```
INST_ARY_GETM(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT,
SIZE:=SIZE_UINT, DONE=>DNOE_BOOL, STAT=>STAT_UINT, DATA=>ARY_DATA);
```

ARY_PUTM	Write motion control module data(Array)	
	Availability	XGI
	Flags	

Function Block	Description
	**Input** REQ: execute the function in case of 1 BASE: Base position setting SLOT: Slot position setting MADDR: Address to start writing; 0(h0) ~ 511(h1FF) DATA: Array variable to save data (ARRAY OF DWORD) SIZE: No. of data to write (1 ~ 512) **Output** DONE: 1 output in case of normal execution STAT: Error information

■ **Function**

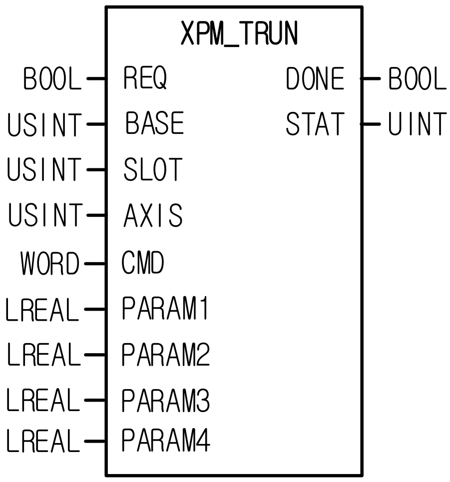
Save data as much as the size to the shared write memory addresses MADDR of the configured motion control module.

■ **Program example**

ST

```
INST_ARY_PUTM(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT,
DATA:=ARY_DATA, SIZE:=SIZE_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_TRUN	Motion controller module test run	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ: execute the function in case of 1 BASE: Base position setting SLOT: Slot position setting AXIS: Specify the axis to issue the command 1 to 32 (1 to 32 axes), 37 to 40 (37 to 40 axes), 255 (total axes) CMD : Command Code (1 to 10) PARAM1: Command auxiliary data 1 PARAM2: Command auxiliary data 2 PARAM3: Command auxiliary data 3 PARAM4: Command auxiliary data 4 **Output** DONE: Keep 1 after initial operation STAT: Error information

■ **Function**

(1) This command is a test operation command that can execute simple motion control operations such as EtherCAT Slave connection / disconnection, servo on / off, and position control to the motion control module.
(2) The module can be viewed by executing a simple module operation with the test run command in the STOP state.
(3) Gives CMD command to the axis designated as AXIS of the motion control module designated by BASE (base number of motion module) and SLOT (slot number of motion module).
(4) In AXIS, specify the axis to issue CMD and set the following values. If you set a value other than the set value, "Error 6" occurs.
1 to 32 (1 to 32 axes), 37 to 40 (37 to 40 axes), 255 (total axes)
(5) If the value set in CMD is 0, "Error 11" occurs in STAT.
(6) If the motion control module executes a test operation command in the RUN state, a 0x002A error occurs in the motion control module and 0x002A is output to the STAT of the function block.

(7) Command code and command auxiliary data setting values are as follows.

Function	Command code	Auxiliary data 1	Auxiliary data 2	Auxiliary data 3	Auxiliary data 4
EtherCAT connection	1	-	-	-	-
Disconnect EtherCAT	2	-	-	-	-
Servo on	3	-	-	-	-
Servo off	4	-	-	-	-
Error reset	5	Error Type 0: Axis error 1: Common error	-	-	-
Homing	6	-	-	-	-
Position control (absolute)	7	Position	Velocity	Acceleration	Deceleration
Position control (relative)	8	Position	Velocity	Acceleration	Deceleration
Velocity control	9	Velocity	Acceleration	Deceleration	-
Stop	10	Deceleration		-	-

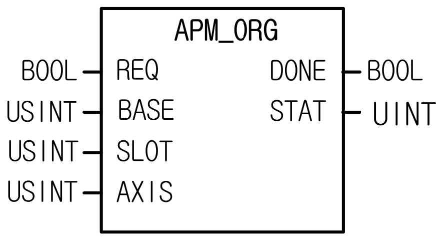
■ **Program Example**

ST

INST_XPM_TRUN (REQ: = (* BOOL *), BASE: = (* USINT *), SLOT: = (* USINT *), AXIS: = (* USINT *), CMD: = (* WORD *), PARAM1: = (* LREAL *), PARAM2: = (* LREAL *), PARAM3: = (* LREAL *), PARAM4: = (* LREAL *), DONE => (* BOOL *), STAT => (* UINT *))

11.4 Positioning Function Block (APM)

APM_ORG	Homing Start	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis **Output** DONE: maintains 1 after the first operation STAT: output error number that occurs while executing the function block

■ **Function**

(1) The instruction commands origin return run to the positioning module.
(2) Run instruction to find origin by means of the direction, compensation, speed (high speed/low speed) and dwell time set in origin return parameter of each axis.
(3) Instruct origin return instruction to the designated AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."

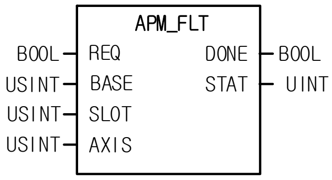
0: X axis, 1: Y axis, 2: Z axis (In case of XEC, Z axis is not supported)

■ **Program example**

ST

```
INST_APM_ORG(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
DONE=>DNOE_BOOL, STAT=>STAT_UINT);
```

APM_FLT	Floating origin setting	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands executing floating origin setting to the positioning module.

(2) As the command used to set the current position as origin, instead of executing return of a machine, the address configured in origin return address is set as the current position.

(3) It commands floating origin command to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).

(4) It can set an axis to instruct and value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis

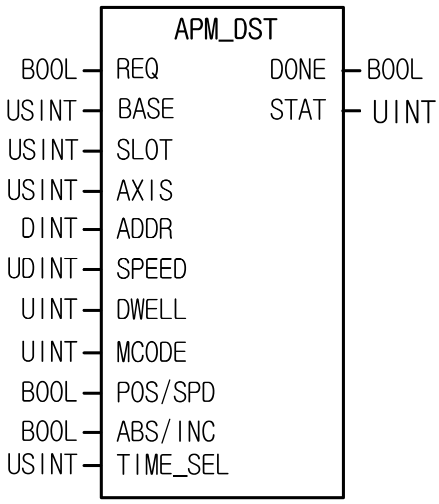
■ **Program example**

ST

```
INST_APM_FLT(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_DST	**Direct Start**	
	Availability	XGI, XGR, XEC
	Flags	

Function Block



Description

Input
REQ: requires to execute the function block
BASE: Setting the base number with a module
SLOT: Setting the slot number with a module
AXIS: Setting an axis to instruct
0:X axis, 1:Y axis, 2:Z axis
ADDR: Setting target position address
-2,147,483,648 ~ +2,147,483,647
SPEED: Setting target speed
Open Collector : 1 ~ 200,000[pps]
Line Driver : 1 ~ 1,000,000[pps]
DWELL: dwell time
0 ~ 50000[ms]
MCODE: Setting M Code
POS/SPD: Setting position control/speed control
0 : position control, 1 : speed control
ABS/INC: Setting absolute/relative coordinates
0 : absolute, 1 : relative
TIME_SEL: setting acc./dec. time number
0 : acc./dec. time 1
1 : acc./dec. time 2
2 : acc./dec. time 3
3 : acc./dec. time 4

Output
DONE: maintains 1 after the first operation
STAT: Output the error number that occurs while executing the function block

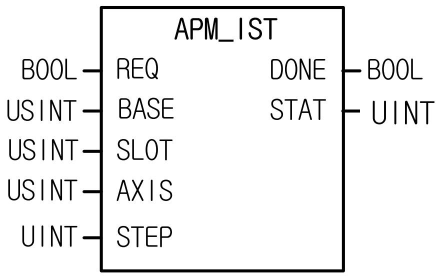
■ **Function**

(1) The instruction commands direct run to the positioning module.
(2) It used when running by designating the run step number of the axis configured as run data.
(3) It command direct run instruction to the configured axis of the positioning module where it is configured at BASE (base number of positioning module) and SLOT(slot number of positioning module).
It can set an axis to instruct and the value is as follows. If other value is set, it produces 'Error6'.
If can value set in SPEED, DWELL, and TIME_SEL is out of the range, it generates 'Error11' to STAT.

■ **Program example**
ST

```
INST_APM_DST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
ADDR:=ADDR_DINT, SPEED:=SPEED_UDINT, DWELL:=DWELL_UINT, MCODE:=MCODE_UINT,
POS_SPD:=POS_BOOL, ABS_INC:=ABS_BOOL, TIME_SEL:=TIME_USINT, DONE=>DNOE_BOOL,
STAT=>STAT_UINT);
```

APM_IST	Indirect Start	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis STEP: Step number to run 0 ~ 400 **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

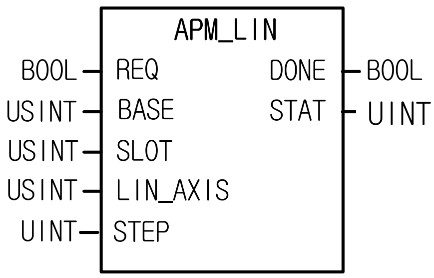
1. The instruction commands direct run to the positioning module.
2. It used when running by designating the run step number of the axis configured as run data.
3. It commands indirect run to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
4. It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis (in case of XEC, Z axis is not supported)
5. If the value set in STEP is out of the range (0 ~ 400 (in case of XEC, 0 ~ 80)), it generates "Error11" to STAT.
6. If 0 is set in STEP, it operates the current step.

■ **Program example**

1. ST

```
INST_APM_IST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
STEP:=STEP_UINT, DONE=>DNOE_BOOL, STAT=>STAT_UINT);
```

APM_LIN	Linear interpolation run	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module LIN_AXIS: Setting interpolation run axis 3 : X/Y axis 5 : X/Z axis 6 : Y/Z axis 7 : X/Y/Z axis STEP: Step number to run 0 ~ 400 **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands linear interpolation run instruction to the positioning module.
(2) It commands for linear interpolation run in the 2 or 3 axes positioning module.
(3) It commands linear interpolation run instruction to the designated AXIS of the positioning module where it is designated at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) If other value is set in LIN_AXIS, it produces "Error6." It can be set by setting each bit as follows.

15 ~ 4	2	1	0
-	Z axis (in case of XEC, Z axis is not supported)	Y axis	X axis

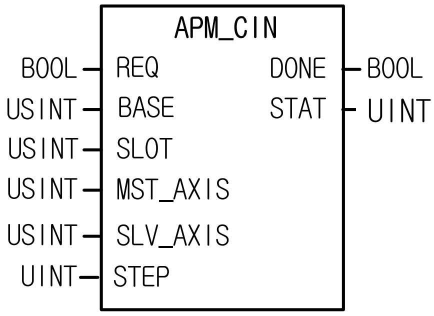
(5) If the value is out of the range, set in STEP (0 ~ 400 (In case of XEC, 0~80)), it generates "Error11" to STAT.
(6) If 0 is set in STEP, it operates the current step.

■ **Program example**

1. ST

```
INST_APM_LIN(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, LIN_AXIS:=LIN_USINT,
STEP:=STEP_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_CIN	Circular interpolation run	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module MST_AXIS: Setting circular interpolation main axis 0:X axis, 1:Y axis, 2:Z axis SLV_AXIS: Setting linear interpolation sub axes 0:X axis, 1:Y axis, 2:Z axis STEP: Step number to run 0 ~ 400 **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

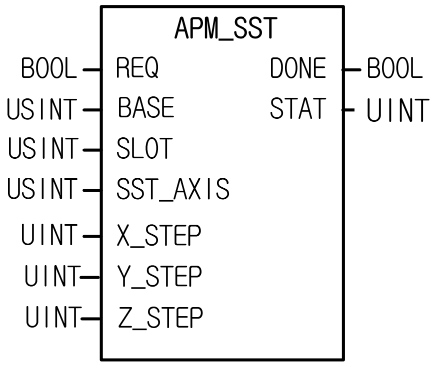
(1) The instruction commands circular interpolation run instruction to the positioning module.
(2) It commands for circular interpolation run in 2 or 3 axes positioning module.
(3) It commands circular interpolation run instruction to the designated AXIS of the positioning module where it is designated at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) MST_AXIS sets the main axis of circular interpolation run and the following values can be set.
0: X axis, 1: Y axis, 2: Z axis
(5) SLV_AXIS sets the sub axis of circular interpolation run and the following values can be set.
0: X axis, 1: Y axis, 2: Z axis
- If the values of MST_AXIS and SLV_AXIS are set out of the range, it generates “Error6.”
- If other value set in STEP (0 ~ 400), it generates “Error11” to STAT.
- If 0 is set in STEP, it operates the current step.

■ **Program example**

1. ST

INST_APM_CIN(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MST_AXIS:=MST_USINT, SLV_AXIS:=SLY_USINT, STEP:=STEP_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);

APM_SST	Simultaneous Start	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module SST_AXIS : Setting simultaneous run axes 3 : X/Y axis 5 : X/Z axis 6 : Y/Z axis 7 : X/Y/Z axis X_STEP: Setting the simultaneous run step number of X axis(0 ~ 400) Y_STEP: Setting the simultaneous run step number of Y axis(0 ~ 400) Z_STEP: Setting the simultaneous run step number of Z axis(0 ~ 400) **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands simultaneous run instruction to the positioning module.
(2) It is executed when simultaneously running 2 or 3 axes
(3) It commands the simultaneous run instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) If the value is set out of the range to SST_AXIS, it generates "Error6." It can be set as follows by setting each bit.

15 ~ 4	2	1	0
-	Z axis (in case of XEC, Z axis is not supported)	Y axis	X axis

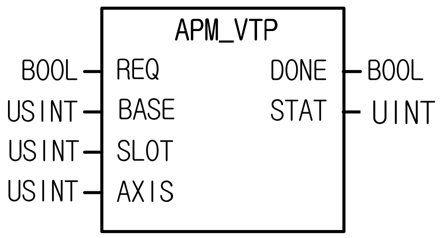
(5) Set the step number run by X axis, Y axis and Z axis simultaneously to X_STEP, Y_STEP and Z_STEP .
(6) If the value set in X_STEP, Y_STEP and Z_STEP is out of the range (0 ~ 400(in case of XEC, 0~80)), it generates "Error11" to STAT.
(7) If 0 is set in X_STEP, Y_STEP and Z_STEP, it operates the current step.

■ **Program example**

1. ST

```
INST_APM_SST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, SST_AXIS:=SST_USINT,
X_STEP:=X_UINT, Y_STEP:=Y_UINT, Z_STEP:=Z_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_VTP	Speed/Position switching	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

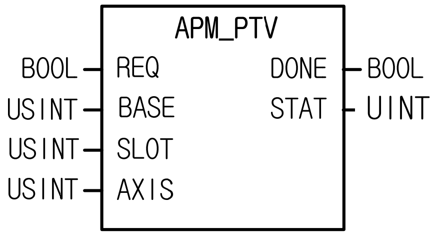
(1) The instruction commands speed/position control conversion instruction to the positioning module.
(2) A configured axis converts speed control to position control if receiving speed/position control instruction while being run by speed control run.
(3) As soon as the instruction is executed, the origin is determined and it moves to the target position by the previous speed control, completing positioning.
(4) It commands speed/position control instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(5) It can set an axis to instruct and the following value. If other value set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis (in case of XEC, Z axis is not supported)

■ **Program example**

1. ST

INST_APM_VTP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT)

APM_PTV	Position/Speed switching	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

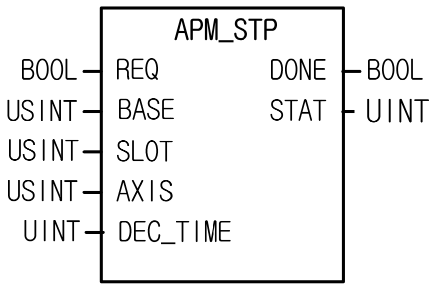
(1) The instruction commands position/speed control conversion instruction to the positioning module.
(2) A configured axis converts speed control to position control if receiving position/speed control instruction while being run by speed control run.
(3) As soon as the instruction is executed, the origin is not determined and it moves the target position by the previous speed control and completes positioning.
(4) It commands speed/position control instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(5) It can set an axis to instruct and the value is as follows. If other value is set out of range, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis (In case of XEC, Z axis is not supported)

■ **Program example**

1. ST

```
INST_APM_PTV(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_STP	**Decelerating stop**	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
 	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis DEC_TIME: Decelerating stop time 0: Acc./dec. time applied when it starts running 1 ~ 65,535 : 1 ~ 65,535ms **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

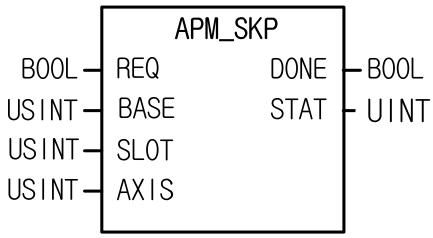
(1) Instruction executing decelerating stop to the positioning module.
(2) It decelerates and stops when it receives the stop command while running by run data and resumes running by run command.
(3) It is used to exit each speed/position synchronization in speed synchronization or position synchronization.
(4) It command decelerating stop to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(5) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis (In case of XEC, Z axis is not supported)

■ **Program example**

1. ST

```
INST_APM_STP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
DEC_TIME:=DEC_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SKP	Skip run	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

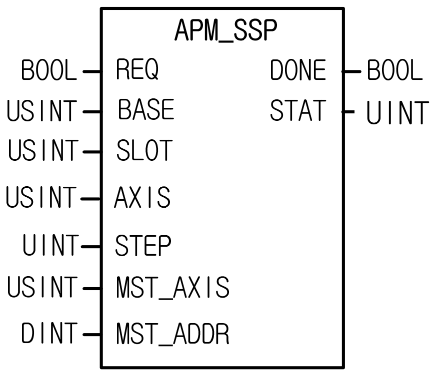
(1) The instruction commands skip run instruction to the positioning module.
(2) It executes when moving to the next step without run step.
(3) Every time the instruction executes, it skips the current run step and starts the next run step.
(4) It commands skip run instruction to the configured AXIS of the positioning module where it is configured at BASE
(base number of positioning module) and SLOT (slot number of positioning module).
(5) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis

■ **Program example**

1. ST

INST_APM_SKP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);

APM_SSP	Position synchronization	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis STEP: Step number to run 0 ~ 400 MST_AXIS: Setting position synchronization main axis 0:X axis, 1:Y axis, 2:Z axis MST_ADDR: Setting main axis to execute position synchronization -2,147,483,648 ~ 2,147,483,647 **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

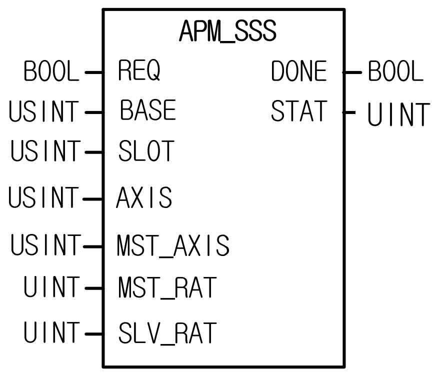
(1) The instruction commands position synchronization instruction to the positioning module
(2) If an axis with the instruction is set as sub axis and the axis set as main axis reaches to the set synchronization position, it starts run step set in instruction axis.
(3) It commands positioning instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the following value. If other value is set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis (In case of XEC, Z axis is not supported)
(5) It sets the position synchronization main axis to MST_AXIS and the following values can be set. If other value is set, it generates "Error6."
0: X axis, 1: Y axis, 2: Z axis (In case of XEC, Z axis is not supported)

■ **Program example**

1. ST

```
INST_APM_SSP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
STEP:=STEP_UINT, MST_AXIS:=AXIS_USINT, MST_ADDR:=ADDR_DINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

<table>
<tr><td rowspan="3">APM_SSS</td><td colspan="2">Speed synchronization</td></tr>
<tr><td>Availability</td><td>XGI, XGR</td></tr>
<tr><td>Flags</td><td></td></tr>
</table>

<table>
<tr><th>Function Block</th><th>Description</th></tr>
<tr><td>
</td><td>Input REQ: requires to execute the function block
BASE: Setting the base number with a module
SLOT: Setting the slot number with a module
AXIS: Setting an axis to instruct
0:X axis, 1:Y axis, 2:Z axis
MST_AXIS: Setting main axis of speed synchronization
0:X axis, 1:Y axis, 2:Z axis, 3:Encoder
MST_RAT: Setting speed rate of main axis
1 ~ 65,535
SLV_RAT: Setting speed rate of sub axis
1 ~ 65,535

Output DONE : maintains 1 after the first operation
STAT : Output the error number that occurs while executing the function block.</td></tr>
</table>

■ **Function**

(1) The instruction commands speed synchronization instruction to the positioning module.

(2) It is executes when controlling at the rate of run speed between both axes.

(3) It must be set to be "speed rate of sub axis/speed rate of main axis ≤ 1" if using speed synchronization run.

(4) It commands speed synchronization instruction to the assigned AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).

(5) It can set an axis to instruct and the following value. If other value is set, it produces "Error6."

0: X axis, 1: Y axis, 2: Z axis

(6) It can set an main axis in MST_AXIS and the following value. If other value is set, it produces "Error6."

0: X axis, 1: Y axis, 2: Z axis, 3: Encoder

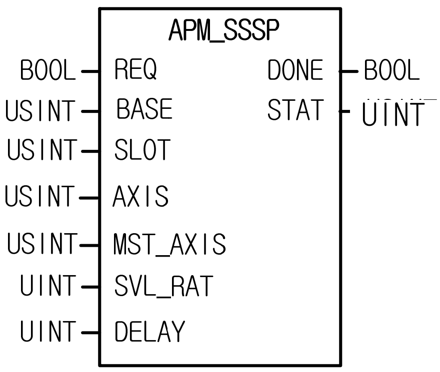
■ **Program example**

1. ST

```
INST_APM_SSS(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
MST_AXIS:=AXIS_USINT, MST_RAT:=MST_UINT, SLV_RAT:=SLV_UINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

APM_SSSP	Positioning speed synchronization	
	Availability	XGI, XGR
	Flags	

Function Block	Description



Input REQ : requires to execute the function block
BASE : Setting the base number with a module
SLOT : Setting the slot number with a module
AXIS : Setting an axis to instruct
0:X axis, 1:Y axis
MST_AXIS : Setting main axis of speed synchronization

Setting value	Main axis setting	Setting value	Main axis setting
0	X axis	5	High Speed Counter Ch3
1	Y axis	6	High Speed Counter Ch4
2	High Speed Counter Ch0	7	High Speed Counter Ch5
3	High Speed Counter Ch1	8	High Speed Counter Ch6
4	High Speed Counter Ch2	9	High Speed Counter Ch7

SLV_RAT : Setting speed rate of main axis
1 ~ 65,535
DELAY : Setting speed rate of sub axis
1 ~ 65,535

Output DONE : maintains 1 after the first operation
STAT : Output the error number that occurs while executing the function block.

■ Function

(1) The instruction commands speed synchronization instruction to the positioning module
(2) At the rising edge of input condition, axis set in AXIS is set as subsidiary axis and axis set in MST_AXIS is set as main axis and speed synchronization instruction is executed.
(3) If instruction executes, subsidiary axis doesn't yield pulse. (At this time, operation status flag (X axis: %KX6720, Y axis: %KX6880) is on). At this time, if axis set in MST_AXIS starts, subsidiary axis starts with speed synchronization rate set in AXIS.
(4) Synchronization rate can be set in SLV_RAT is 0.01% ~ 100.00% (setting value 1 ~ 10,000). If synchronization speed rate exceeds this range, error code 356 occurs.
(5) Delay time of DEALY means how long it takes for speed of subsidiary axis to get equal with current main axis speed. In XGB built-in positioning, when speed synchronization control, it detects the current speed of main axis every 500 ㎲ and adjust speed of subsidiary axis. At this time, if speed of subsidiary axis changes rapidly by speed synchronization, rapid change of subsidiary axis may cause damage of motor and noise.

For example, we assume that synchronization speed rate is 100.00% and delay time is 5(ms). In case speed of main axis is 10,000[pps], after 5ms, XGB adjusts speed of subsidiary axis to be 10,000[pps] every 500[㎲] according to current speed of main axis.

The more delay time is large, the more stability increases. When you want high stability of motor, increase the delay time.

(6) The range of delay time can be set in DELAY n2 is 1 ~ 10[ms]. If it exceeds the range, error code 357 occurs.

(7) The range of MST_AXIS is 0~9. If it exceeds the range, error code 355 occurs.

(8) You can specify axis for command at AXIS, The following setting is available. If you input invalid value, error code 6 occurs.

0: X axis, 1: Y axis

(9) You can specify main axis of speed synchronization at MST_AXIS. If you input invalid value, error code 6 occurs.

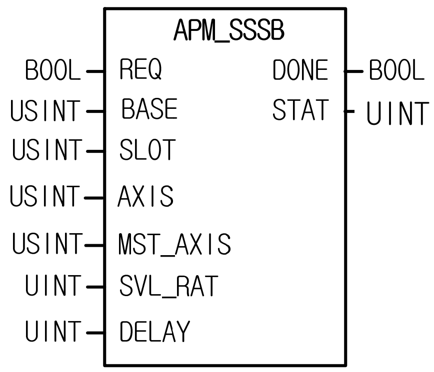
■ **Program example**

1. ST

```
INST_APM_SSSP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
MST_AXIS:=AXIS_USINT, MST_RAT:=MST_UINT, SLV_RAT:=SLV_UINT, POS:=POS_DINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

APM_SSSB	Positioning speed synchronization	
	Availability	XEC
	Flags	-

Function Block	Description



Input REQ : requires to execute the function block
BASE : Setting the base number with a module
SLOT : Setting the slot number with a module
AXIS : Setting an axis to instruct
0:X axis, 1:Y axis
MST_AXIS : Setting main axis of speed synchronization

Setting value	Main axis setting	Setting value	Main axis setting
0	X axis	5	High Speed Counter Ch3
1	Y axis	6	High Speed Counter Ch4
2	High Speed Counter Ch0	7	High Speed Counter Ch5
3	High Speed Counter Ch1	8	High Speed Counter Ch6
4	High Speed Counter Ch2	9	High Speed Counter Ch7

SLV_RAT : Setting speed rate of sub axis
1 ~ 10,000(0.01 ~ 100.00%)
DELAY : Delay time of sub axis
1 ~ 10(1 ~ 10ms)

Output DONE : maintains 1 after the first operation
STAT : Output the error number that occurs while executing the function block.

■ Function

(1) The instruction commands speed synchronization instruction to the positioning module
(2) At the rising edge of input condition, axis set in AXIS is set as subsidiary axis and axis set in MST_AXIS is set as main axis and speed synchronization instruction is executed.
(3) If instruction executes, subsidiary axis doesn't yield pulse. (At this time, operation status flag (X axis: %KX6720, Y axis: %KX6880) is on). At this time, if axis set in MST_AXIS starts, subsidiary axis starts with speed synchronization rate set in AXIS.
(4) Synchronization rate can be set in SLV_RAT is 0.01% ~ 100.00% (setting value 1 ~ 10,000). If synchronization speed rate exceeds this range, error code 356 occurs.
(5) Delay time of DEALY means how long it takes for speed of subsidiary axis to get equal with current main axis speed. In XGB built-in positioning, when speed synchronization control, it detects the current speed of main axis every 500 ㎲ and adjust speed of subsidiary axis. At this time, if speed of subsidiary axis changes rapidly by speed synchronization, rapid change of subsidiary axis may cause damage of motor and noise.

For example, we assume that synchronization speed rate is 100.00% and delay time is 5(ms). In case speed of main axis is 10,000[pps], after 5ms, XGB adjusts speed of subsidiary axis to be 10,000[pps] every 500[㎲] according to current speed of main axis.
The more delay time is large, the more stability increases. When you want high stability of motor, increase the delay time.

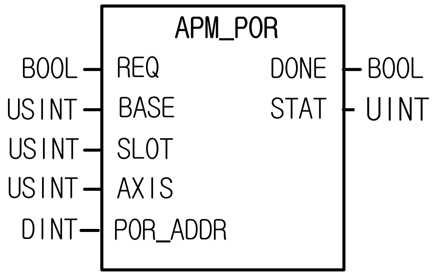
(6) The range of delay time can be set in DELAY n2 is 1 ~ 10[ms]. If it exceeds the range, error code 357 occurs.
(7) The range of MST_AXIS is 0~9. If it exceeds the range, error code 355 occurs.
(8) You can specify axis for command at AXIS, The following setting is available. If you input invalid value, error code 6 occurs.
    0: X axis, 1: Y axis
(9) You can specify main axis of speed synchronization at MST_AXIS. If you input invalid value, error code 6 occurs.

■ **Program example**

2. ST

INST_APM_SSSB(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, MST_AXIS:=AXIS_USINT,
MST_RAT:=MST_UINT, SLV_RAT:=SLV_UINT, POS:=POS_DINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);

APM_POR	Position override	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis POR_ADDR : Setting new target position -2,147,483,648 ~ 2,147,483,647 **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

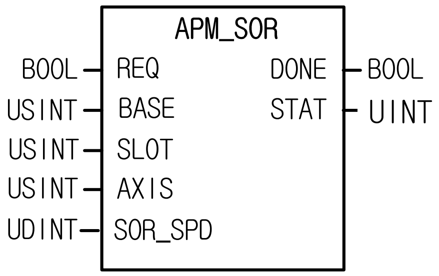
(1) The instruction commands position override instruction to the positioning module.
(2) It used when changing target position while instruction axis is running.
(3) It commands position override instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis (in case of XEC, Z axis is not supported)
(5) Set the target position to change in POR_ADDR.

■ **Program example**

1. ST

```
INST_APM_POR(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
POR_ADDR:=POR_DINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SOR	Speed override	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis SOR_SPD: Setting new run speed value Open Collector: 0 ～ 200,000[pps] Line Driver: 0 ～ 1,000,000[pps] **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands speed override instruction to the positioning module.
(2) It used when changing run speed while instruction axis is running.
(3) It commands speed override instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis (in case of XEC, Z axis is not supported)
(5) Set the target speed to change in SOR_SPD. If the value is set out of the range, it generates “Error11.”
Open Collector: 0 ～ 200,000[pps] (in case of XEC, Z axis is not supported)
Line Driver: 0 ～ 1,000,000[pps]

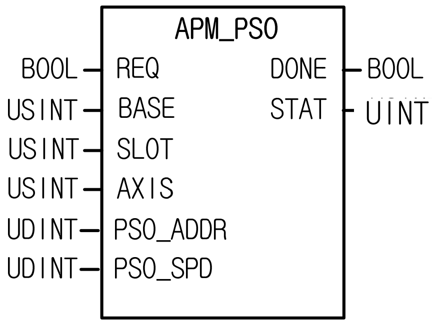
■ **Program example**

1. ST

```
INST_APM_SOR(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
SOR_SPD:=SOR_UDINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_PSO	Positioning speed override	
	Availability	XGI, XGR, XEC
	Flags	

Function Block



Description

Input
REQ: requires to execute the function block
BASE: Setting the base number with a module
SLOT: Setting the slot number with a module
AXIS: Setting an axis to instruct
0:X axis, 1:Y axis, 2:Z axis
PSO_ADDR: Position to change speed
-2,147,483,648 ~ 2,147,483,647
PSO_SPD: Setting new run speed value
Open Collector: 0 ~ 200,000[pps]
Line Driver: 0 ~ 1,000,000[pps]

Output
DONE: maintains 1 after the first operation
STAT: Output the error number that occurs while executing the function block.

■ **Function**

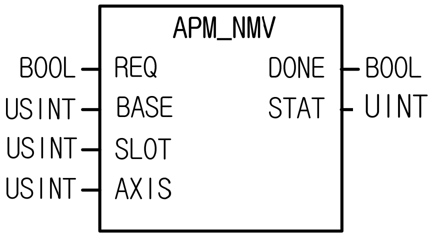
(1) The instruction commands positioning speed override instruction to the positioning module.
(2) It executes when changing run speed after the axis reaches to a certain position while it is running.
(3) It commands speed override instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis (in case of XEC, Z axis is not supported)
(6) Set the target speed to change in PSO_SPD. The value is as follows. If the value is set out of the range, it generates “Error11.”
Open Collector: 0 ~ 200,000[pps] (in case of XEC, Z axis is not supported)
Line Driver: 0 ~ 1,000,000[pps]

■ **Program example**

1. ST

```
INST_APM_PSO(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
PSO_ADDR:=ADDR_UDINT, PSO_SPD:=SPD_UDINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_NMV	Continuous run	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

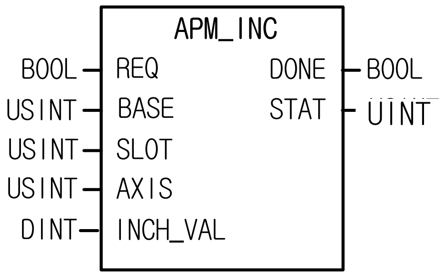
(1) The instruction commands continuous run instruction to the positioning module.
(2) It executes to change the current step to the next step without stop.
(3) It commands continuous run instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis

■ **Program example**

1. ST

```
INST_APM_NMV(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_INC	Inching run	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis INCH_VAL: Setting the movement to move to inching run -2,147,483,648 ~ 2,147,483,647 **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

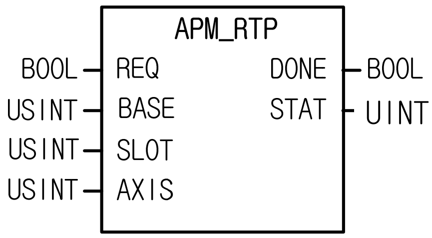
(1) The instruction commands inching run instruction to the positioning module.
(2) Inching run is a type of manual run, used to process minute movement as quantitative run.
(3) The inching run speed is set in manual run parameter.
(4) It commands inching run floating origin instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(5) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis (In case of XEC, Z axis is not supported)

■ **Program example**

1. ST

```
INST_APM_INC(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
INCH_VAL:=INCH_DINT, DONE=>DNOE_BOOL, STAT=>STAT_UINT);
```

APM_RTP	Return to the position before manual run	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

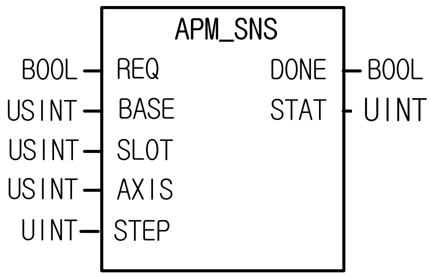
(1) The instruction commands return to the position before manual run to the positioning module.
(2) It executes to return to the position before manual run when the position is changed by manual run after positioning.
(3) It commands Return to the position before manual run instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis

■ **Program example**

1. ST

```
INST_APM_RTP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SNS	Run step number change	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis STEP: Setting run step number to run 1 ~ 400 **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

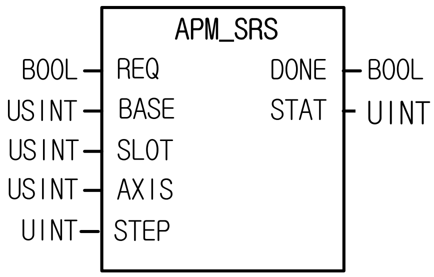
(1) The instruction commands run step number change instruction to the positioning module.
(2) It executes to change run step of the axis
(3) It commands run step number change instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis (In case of XEC, Z axis is not supported)
(5) Set the step number to run in STEP between 1 ~ 400; if other value is set , it generates "Error11."

■ **Program example**

1. ST

```
INST_APM_SNS(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
STEP:=STEP_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SRS	Repeat step number change	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis STEP: Setting repeat step number to change 1 ~ 400 **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

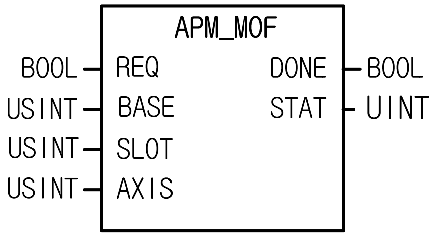
(1) The instruction commands repeat step number change instruction to the positioning module.
(2) It executes to start run in a certain run step by configuring start step number of repeat run in case of repeat run in which it returns to repeat run step if it meets repeat run while running by run data.
(3) It commands repeat step change instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis
(5) Set the step number to start repeat run in STEP between 1 ~ 400; if other value is set , it generates “Error11.”

■ **Program example**

1. ST

INST_APM_SRS(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, STEP:=STEP_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);

APM_MOF	M code cancellation	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands M code cancellation instruction to the positioning module.
(2) If M code is set in the parameter of each axis to With or After mode, it executes to turn off the signal when the M code signal of the axis is on.
(3) It commands M code cancellation instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

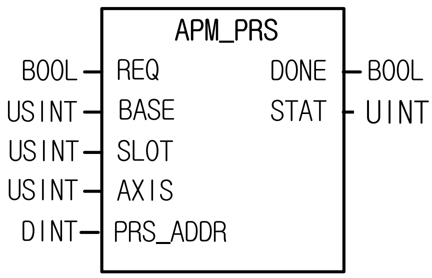
0: X axis, 1: Y axis, 2: Z axis (in case of XEC, Z axis is not supported)

■ **Program example**

1. ST

```
INST_APM_MOF(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_PRS	Current position preset	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis PRS_ADDR: Setting the current position value to change -2,147,483,648 ~ 2,147,483,647 **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands current position preset instruction to the positioning module.
(2) As the command used to change the current position to a temporary position, the origin is determined if executing the command.
(3) It commands current position preset instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."

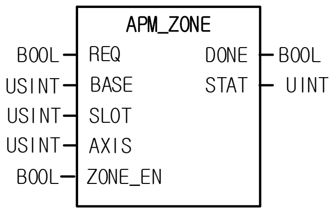
0: X axis, 1: Y axis, 2: Z axis (in case of XEC, Z axis is not supported)

■ **Program example**

1. ST

```
INST_APM_PRS(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
PRS_ADDR:=ADDR_DINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_ZONE	Zone Output allowed/prohibited	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis ZONE_EN: Zone Output allowed/prohibited 0: prohibited, 1: allowed **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

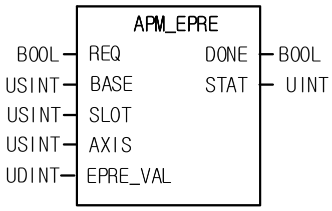
(1) The instruction commands Zone Output allowed/prohibited instruction to the positioning module.
(2) It commands to allow or prohibit Zone Output by using the position data of zone set in common parameter and the position data value set in Zone1, Zone2 and Zone3.
(3) It commands Zone Output allowed/prohibition instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis

■ **Program example**

1. ST

```
INST_APM_ZONE(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
ZONE_EN:=ZONE_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_EPRE	**Encoder value preset**	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis EPRE_VAL: Setting encoder preset value 0 ~ 4,294,967,295 **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

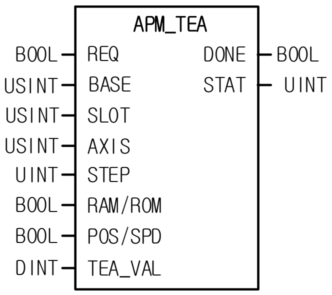
(1) The instruction commands encoder value preset instruction to the positioning module.
(2) It commands to preset the current encoder value set in EPRE_VAL.
(3) It commands encoder value preset instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis

■ **Program example**

1. ST

```
INST_APM_EPRE(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
EPRE_VAL:=EPRE_UDINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_TEA	Singular teaching	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis STEP: Setting step number for teaching 0 ~ 400 RAM/ROM: Selecting RAM teaching/ROM teaching type 0 : RAM teaching, 1 : ROM teaching POS/SPD: Selecting position teaching/speed teaching type 0 : position teaching, 1 : speed teaching TEA_VAL: Setting teaching value Position teaching: -2,147,483,648 ~ 2,147,483,647 Speed teaching: Open Collector 0 ~ 200,000[pps] Line Driver 0 ~ 1,000,000[pps] **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands singular teaching instruction to the positioning module.

(2) Speed teaching can be used when using a temporary speed for run data of a certain step while position teaching is used to set a temporary position for run data of a certain run step.

(3) It commands singular teaching instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).

(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis

(5) It can set the step number of run data for teaching in STEP between 0 ~ 400. If other value is set, it generates “Error11.”

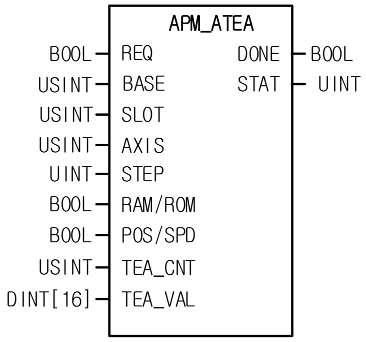
(6) In case of position teaching, a position value for teaching is set in TEA_VAL while speed value for teaching is set; the setting ranges are as follows. If other value is set, it generates “Error11.”
- Position teaching range: -2,147,483,648 ~ 2,147,483,647
- Speed teaching range: Open Collector Output -> 0 ~ 200,000 [pps]
  Line Driver Output -> 0 ~ 1,000,000 [pps]

■ **Program example**

1. ST

```
INST_APM_TEA(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
STEP:=STEP_UINT, RAM_ROM:=RAM_BOOL, POS_SPD:=SPD_BOOL);
```

APM_ATEA	Singular teaching	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0 : X axis, 1:Y axis, 2:Z axis STEP: Setting step number for teaching, 0 ~ 400 RAM/ROM: Selecting RAM teaching/ROM teaching 0 : RAM teaching, 1 : ROM teaching POS/SPD: Selecting position teaching/speed teaching type 0 : position teaching, 1 : speed teaching TEA_CNT : Setting the no. of data for teaching, 1 ~ 16 TEA_VAL : Setting teaching value Position teaching: -2,147,483,648 ～ 2,147,483,647 Speed teaching : Open Collector 0 ～ 200,000[pps] Line Driver 0 ～ 1,000,000[pps] **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands plural teaching instruction to the positioning module.
(2) Speed teaching can be used when using a temporary speed for run data of a certain step while position teaching is used to set a temporary position for run data of a certain run step.
(3) Using the teaching plural function block, up to 16 target positions and speed values can be changed.
(4) It commands plural teaching instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(5) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis
(6) It can set the step number of run data for teaching in STEP between 0 ~ 400. If other value is set, it generates “Error11.”
(7) The number of data is set in TEA_CNT up to 16. If other value is set out of the range, it generates “Error11.”
(8) In case of position teaching, a position value for teaching is set in TEA_VAL while speed value for teaching is set; the setting ranges are as follows.
● Position teaching range: -2,147,483,648 ～ 2,147,483,647
● Speed teaching range: Open Collector Output -> 0 ～ 200,000 [pps]
Line Driver Output -> 0 ～ 1,000,000 [pps]

■ **Program example**

1. ST

```
INST_APM_ATEA1(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
STEP:=STEP_UINT, RAM_ROM:=RAM_BOOL, POS_SPD:=SPD_BOOL, TEA_CNT:=CNT_USINT,
ATEA_VAL:=ARY_ATEA, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SBP	Basic parameter teaching	
	Availability	XGI, XGR
	Flags	

Function Block	Description
APM_SBP BOOL – REQ DONE – BOOL USINT – BASE STAT – UINT USINT – SLOT USINT – AXIS UDINT – BP_VAL USINT – BP_NO	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis BP_VAL: basic parameter value to change BP_NO: basic parameter item number to change **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

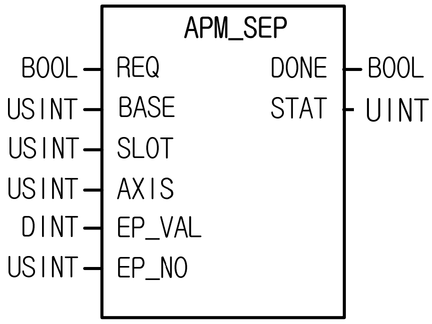
(1) The instruction commands basic parameter teaching instruction to the positioning module.
(2) The parameter modified by basic parameter setting instruction is valid only when the power is on. To save the parameter modified by basic parameter setting instruction, it is necessary to save the parameter value modified by save parameter/run data save instruction (WRT) to ROM after setting basic parameter.
(3) It commands basic parameter setting instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis
(5) The following values can be set in the basic parameter item number.
1: speed limit
2: bias speed
3: acc./dec. time 1
4: acc./dec. time 2
5: acc./dec. time 3
6: acc./dec. time 4
7: no. of pulse per rotation
8: conveyance distance per rotation
9: pulse output mode
10: unit
11: unit allocation

■ **Program example**

1. ST

```
INST_APM_SBP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
BP_NO:=EP_USINT*), BP_VAL:=BP_UDINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SEP	Extension parameter teaching	
	Availability	XGI, XGR
	Flags	

Function Block	Description
 	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis EP_VAL: Extension parameter value to change EP_NO: Extension parameter number to change **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands extension parameter teaching instruction to the positioning module.
(2) The parameter modified by extension parameter setting instruction is valid only when the power is on. To save the parameter modified by extension parameter setting instruction, it is necessary to save the parameter value modified by save parameter/run data save instruction (WRT) to ROM after setting extension parameter.
(3) It commands extension parameter setting instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis
(5) The following values can be set in the extension parameter item number.
1: Software upper limit
2: Software lower limit
3: Backlash compensation
4: Position completion output time
5: S-Curve rate
6: External instruction selection
7: Pulse output direction
8: Acc./dec. pattern
9: M code number
10: Position display during uniform run
11: Upper/lower limit display during uniform run
12: External speed/position control conversion allowed
13: External instruction allowed
14: External stop allowed
15: External simultaneous run allowed

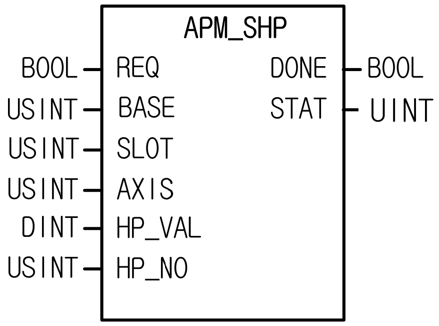
16: Positioning completion condition
17: Driver ready/in-position

■ **Program example**

1. ST

```
INST_APM_SEP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
EP_NO:=NO_USINT, EP_VAL:=EP_DINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SHP	Origin return parameter setting	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis HP_VAL: origin return parameter value to change HP_NO: origin return parameter item number to change **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

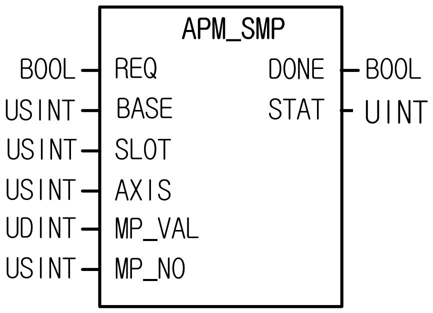
(1) The instruction commands an origin return parameter teaching instruction to the positioning module.
(2) The parameter modified by origin return parameter setting instruction is valid only when the power is on. To save the parameter modified by origin return parameter setting instruction, it is necessary to save the parameter value modified by save parameter/run data save instruction (WRT) to ROM after setting origin return parameter.
(3) It commands origin return parameter teaching instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
  0: X axis, 1: Y axis, 2: Z axis
(5) The values to set to origin return parameter items are as follows.
  1: Origin address
  2: Origin return high speed
  3: Origin return low speed
  4: Acc./dec. time of origin return
  5: Dwell time of origin return
  6: Origin compensation
  7: Re-run time of origin return
  8: Origin return mode
  9: Origin return direction

■ **Program example**

1. ST

```
INST_APM_SHP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
HP_NO:=NO_USINT, HP_VAL:=HP_DINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SMP	Manual run parameter teaching	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis MP_VAL: Manual run parameter value to change MP_NO: Manual run parameter item number to change **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

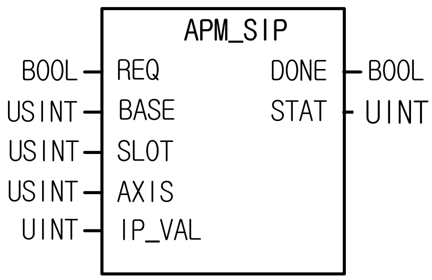
(1) The instruction commands manual run parameter teaching instruction to the positioning module.
(2) The parameter modified by manual run parameter teaching instruction is valid only when the power is on. To save the parameter modified by manual run parameter teaching instruction, it is necessary to save the parameter value modified by parameter/run data save instruction (WRT) to ROM after setting manual run parameter teaching.
(3) It commands manual run parameter teaching instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis
(5) The values to set in manual run parameter item number are as follows.
1: Jog high speed
2: Jog low speed
3: Jog acc./dec. time
4: Inching speed

■ **Program example**

1. ST

INST_APM_SMP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, MP_NO:=NO_USINT, MP_VAL:=MP_UDINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);

APM_SIP	Input signal parameter teaching	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis IP_VAL: External signal parameter value to change / setting the signal allocated by each bit. **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands input signal parameter teaching to the positioning module.
(2) The parameter modified by input signal parameter teaching instruction is valid only when the power is on. To save the parameter modified by input signal parameter setting instruction, it is necessary to save the parameter value modified by save parameter/run data save instruction (WRT) to ROM after setting external signal parameter.
(3) It commands input signal parameter teaching to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis

5. The setting of each input signal setting area has the following meaning.
0: contact A, 1: contact B
The signals allocated to each bit of input signal parameter value to change are as follows.

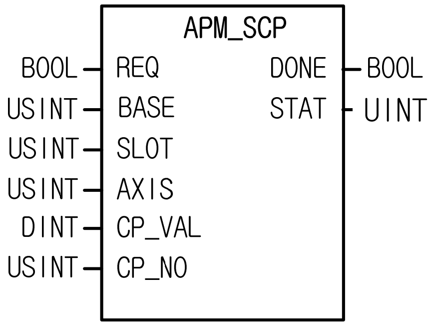
Bit	Input signal	Bit	Q signal
0	Upper limit signal	6	Instruction signal
1	Lower limit signal	7	Sub instruction signal
2	Approx. origin signal	8	Speed/position conversin signal
3	Origin signal	9	Driver ready/in-position signal
4	Emergency stop signal	10	External simultaneous run signal
5	Dec. stop signal	15 ~ 11	-

■ **Program example**

1. ST

```
INST_APM_SIP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
IP_VAL:=IP_WORD, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SCP	**Common parameter teaching**	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis CP_VAL: Common parameter value to change CP_NO: Common parameter item number to change **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands common parameter teaching instruction to the positioning module.
(2) The parameter modified by common parameter setting instruction is valid only when the power is on. To save the parameter modified by common parameter setting instruction, it is necessary to save the parameter value modified by using save parameter/run data instruction (WRT) to ROM after common parameter teaching.
(3) It commands common parameter teaching instruction to the axis configured as the positioning axis configured as BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis

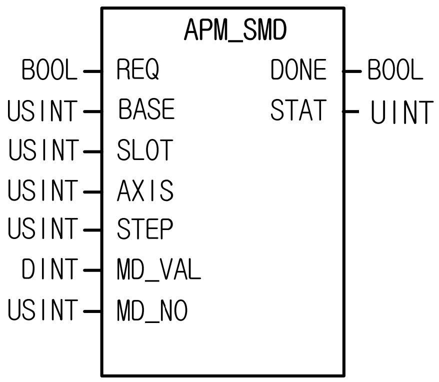
5. The values to set in common parameter item number are as follows.
1: Pulse Output level
2: Circular interpolation method
3: Encoder Input mode
4: Encoder Auto Reload value
5: ZONE Output mode
6: ZONE1 axis setting
7: ZONE2 axis setting
8: ZONE3 axis setting
9: ZONE1 On area
10: ZONE1 Off area
11: ZONE2 On area
12: ZONE2 Off area
13: ZONE3 On area
14: ZONE3 Off area

■ **Program example**

1. ST

```
INST_APM_SCP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
CP_NO:=NO_USINT, CP_VAL:=CP_DINT, ENC_LD:=ENC_UDINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SMD	**Run data teaching**	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis STEP: Run step number to change 0 ~ 400 MD_VAL: Run data value to change MD_NO: Run data item number to change **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands run data teaching instruction to the positioning module.
(2) The parameter modified by run data teaching instruction is valid only when the power is on. To save the parameter modified by run data setting instruction, it is necessary to save the parameter value modified by using save parameter/run data instruction to ROM.
(3) It commands run data teaching instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis

5. The following values can be set into the run data item number.

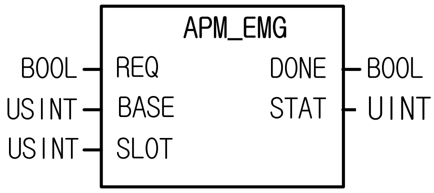
1: target position
2: circular interpolation sub point
3: target speed
4: dwell time
5: M code
6: control method
7: run mode
8: run pattern
9: coordinate
10: acc./dec. number
11: circular interpolation direction

■ **Program example**

1. ST

```
INST_APM_SMD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
STEP:=STEP_UINT, MD_NO:=NO_USINT, MD_VAL:=MD_DINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_EMG	Emergency stop	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

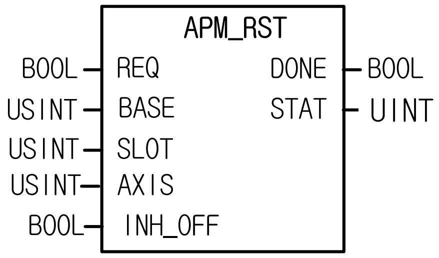
(1) The instruction commands emergency stop instruction to the positioning module.
(2) It commands Emergency stop instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(3) It is executed when stopping running due to emergency situation and every axis receiving the instruction would stop.
(4) Since it is converted to output prohibition and origin not determined, to resume running, it needs to cancel output prohibition and determine the origin again.

■ **Program example**

1. ST

INST_APM_EMG(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);

APM_RST	Error reset/Output prohibition cancel	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis INH_OFF: Output prohibition cancellation 0: Error reset 1: Error reset/Output prohibition cancellation **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

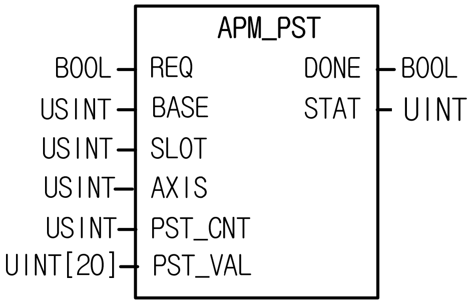
(1) The instruction commands error reset/output prohibition cancellation to the positioning module.
(2) It commands error reset/output prohibition cancel instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis (in case of XEC, Z axis is not supported)
(4) It is executed when canceling the status of pulse output prohibited by external emergency stop or upper/lower limit detection or resetting an error that occurs when parameter is out of the range or while running.

■ **Program example**

1. ST

INST_APM_RST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, INH_OFF:=INH_BOOL, DONE=>DOONE_BOOL, STAT=>STAT_UINT);

APM_PST	Point run	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis PST_CMT: Setting the number of point run step 0 ~ 19 PST_VAL: Setting the point run step number 0 ~ 400 **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands point run instruction to the positioning module.
(2) It commands point run instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

0: X axis, 1: Y axis, 2: Z axis

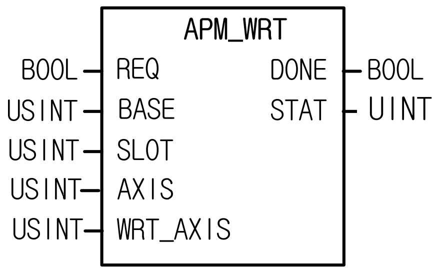
(4) It executes when continuously running without stop by one instruction by setting max. 20 run steps in case of PTP (point to point) run.
(5) If other value is set in PST_CNT or PST_VAL, it generates “Error6.”

■ **Program example**

1. ST

```
INST_APM_PST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
PST_CNT:=CNT_USINT, PST_VAL:=ARY_PST, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_WRT	Save parameter/run data	
	Availability	XGI, XGR, XEC
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis WRT_AXIS: Setting save axis(by setting each bit) 0bit:X axis, 1bit:Y axis, 2bit:Z axis **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

(1) The instruction commands save parameter/run data instruction to the positioning module.
(2) It commands save parameter/run data instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."

0: X axis, 1: Y axis, 2: Z axis

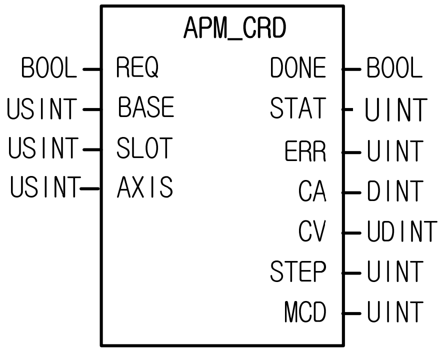
(4) It commands the instruction to save the current run parameter and run data of the axis set in WRT_AXIS to Flash ROM.

■ **Program example**

1. ST

INST_APM_WRT(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, WRT_AXIS:=WRT_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);

APM_CRD	**Read run info**	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block. ERR: display error during operation CA: display current position address CV: display current run speed STEP: display current run data step number MCD: display current MCode value

■ **Function**

(1) The instruction commands read run info instruction to the positioning module.
(2) It commands Read current run info instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis
(4) It can monitor by reading the current position address, run speed, run data number and M code number of the preset axis or be used in a user program.

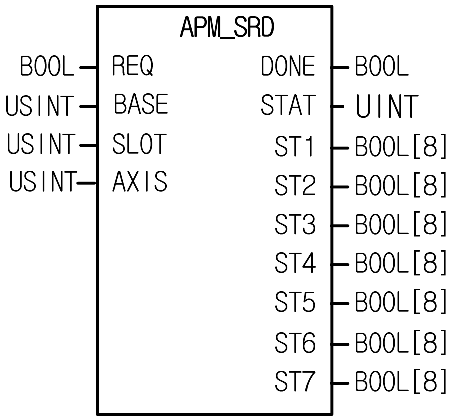
■ **Program example**

1. ST

```
NST_APM_CRD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT, ERR=>ERR_UINT, CA=>CA_DINT, CV=>CV_UDINT,
STEP=>STEP_UINT, MCD=>MCD_UINT);
```

APM_SRD	**Read run state**	
	Availability	XGI, XGR
	Flags	

Function Block



Description

Input
REQ: requires to execute the function block
BASE: Setting the base number with a module
SLOT: Setting the slot number with a module
AXIS: Setting an axis to instruct
0:X axis, 1:Y axis, 2:Z axis

Output
DONE: maintains 1 after the first operation
STAT: Output the error number that occurs while executing the function block.
ST1: state 1
ST2: state 2
ST3: state 3
ST4: state 4
ST5: state 5
ST6: state 6
ST7: state 7

■ **Function**

(1) The instruction commands read run state run instruction to the positioning module.
(2) It commands Read run state instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
0: X axis, 1: Y axis, 2: Z axis
(4) The content of ST1 ~ ST7, the output variables of current run state bit read function block is important information that should be applied in the program.
(5) Each bit of ST1 ~ ST4 has the following meaning.

	Bit	Description	Bit	Description
ST1	[0]	Running(0: stop, 1: BUSY)	[4]	Origin determined (0: not determined, 1: completed)
	[1]	Error state	[5]	Pulse Output prohibited (0: allowed, 1: prohibited)
	[2]	Positioning complete	[6]	Stop
	[3]	M Code On signal (0: Off, 1: On)	[7]	-

	Bit	Description	Bit	Description
ST2	[0]	Upper limit detected	[4]	Accelerating
	[1]	Lower limit detected	[5]	Constant speed
	[2]	Emergency stop state	[6]	Decelerating
	[3]	Direction (0: forward, 1: reverse)	[7]	Dwelling
ST3	[0]	1 axis position control	[4]	2 axes circular interpolating
	[1]	1 axis speed control	[5]	Origin return running
	[2]	2 axes linear interpolation	[6]	Position synchronization running
	[3]	3 axes linear interpolation	[7]	Speed synchronization running
ST4	[0]	Jog low speed running	[4]	Returning to the position before manual run
	[1]	Jog high speed running	[5]	-
	[2]	Inching running	[6]	-
	[3]	MPG running	[7]	-

(6) Each bit of ST5 ~ ST7 has the following meaning, respectively.

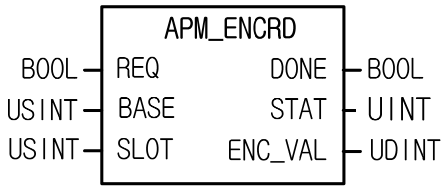
	Bit	Description	Bit	Description
ST5	[0]	Axis state(0: sub, 1: main)	[4]	Main axis info[Encoder]
	[1]	Main axis info(X axis)	[5]	-
	[2]	Main axis info(Y axis)	[6]	-
	[3]	Main axis info(Z axis)	[7]	-
ST6	[0]	Emergency stop signal	[4]	Upper limit signal
	[1]	External stop signal	[5]	Lower limit signal
	[2]	External command signal	[6]	Origin signal
	[3]	Jog high speed reverse signal	[7]	Approx. origin signal
ST7	[0]	Speed/position control conversion signal	[4]	-
	[1]	Driver ready/in-position signal	[5]	-
	[2]	External simultaneous run signal	[6]	-
	[3]	-	[7]	-

■ **Program example**

1. ST

INST_APM_SRD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT, ST1=>ARY_ST1, ST2=>ARY_ST2, ST3=> ARY_ST3, ST4=> ARY_ST4, ST5=> ARY_ST5, ST6=> ARY_ST6, ST7=> ARY_ST7);

APM_ENCRD	Read encoder value	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block. ENC_VAL: current encoder value

■ **Function**

(1) The instruction commands read encoder value instruction to the positioning module.

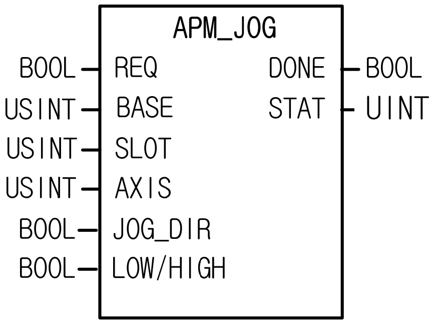
(2) It commands Read encoder value instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).

■ **Program example**

ST

```
INST_APM_ENCRD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT, ENC_VAL=>ENC_UDINT);
```

APM_JOG	Jog run	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis JOG_DIR: Setting rotation direction of jog run 0: forward, 1: reverse LOW/HIGH: Setting jog run speed 0: low speed jog run 1: high speed jog run **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block.

■ **Function**

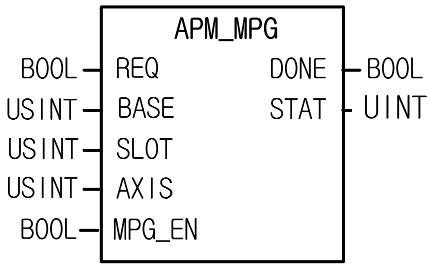
(1) The instruction commands jog run instruction to the positioning module.
(2) The manual run function for test is used to verify the address for system operation, wiring state and teaching.
(3) If connection condition of input variable REQ is on, pulse is output by the value; it stops in case of off.
(4) It commands jog run instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(5) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis

■ **Program example**

1. ST

```
INST_APM_JOG(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
JOG_DIR:=JOG_BOOL, LOW_HIGH:=LOW_HIGH_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_MPG	Manual pulse generator(MPG) run	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis MPG_EN: MPG run allowed/prohibited setting 0: prohibited, 1: allowed **Output** DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block

■ **Function**

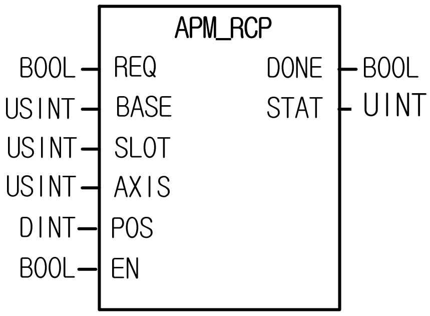
(1) It commands to instruct positioning module to execute MPG run.
(2) Instruct positioning module to be ready for running when it is necessary to run by using externally installed MPG.
(3) It commands MPG run instruction to the configured AXIS of the positioning module where it is configured at BASE (base number of positioning module) and SLOT (slot number of positioning module).
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
0: X axis, 1: Y axis, 2: Z axis

■ **Program example**

ST

INST_APM_MPG(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, MPG_EN:=MPG_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);

APM_RCP	Current position section repetition	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	Input REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis POS: Setting repetition position(address): -2,147,483,648 ~ 2,147,483,647 EN : Enable current position section repetition 0: Prohibit current position section repetition 1: Enable current position section repetition Output DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block

■ **Function**

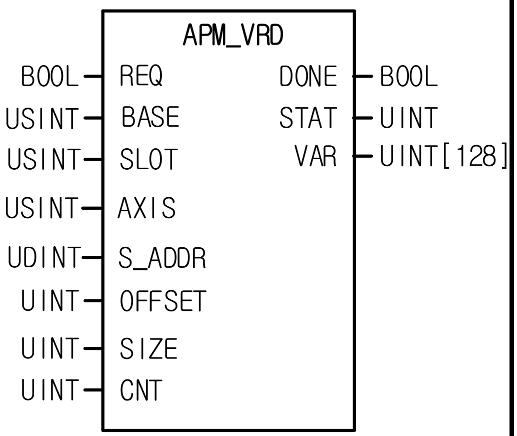
(1) It commands to instruct positioning module to set or prohibit current position section repetition.
(2) It only operates at direct start.
(3) It commands RCP run instruction to the configured AXIS of the positioning module where it is configured at BASE
(base number of positioning module) and SLOT (slot number of positioning module).
(4) For “AXIS”, you can configure the axis to give an instruction. If other value is set, it produces “Error6.”

■ **Program example**

ST

INST_APM_RCP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), POS:=(*DINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

APM_VRD	**Read Variable Data**	
	Availability	XGI, XGR
	Flags	

Function Block	Description
 	Input: REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis S_ADDR: Head address of data in module internal memory to read (0 ~ 12147) OFFSET: Offset between Read data blocks 0 ~ 53329 SIZE : Size of Read data block : 1 ~ 128 CNT : No. of Read data block : 1 ~ 128 Output: DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block VAR : PLC device where Read data is saved

■ **Function**

(1) It commands to instruct positioning module to read parameter, operation data directly

(2) You can read data you want by designating the module internal memory address of parameter and operation data

(3) It reads the positioning module internal memory from the position set by "S_ADDR" by WORD unit and save them in the device set by "VAR". The number of data to read is the number set by "Size". In case "CNT" is larger than 2, it reads multiple data blocks and save them in the device set by "VAR" in order. At this time, head address of next block is "Offset" apart from head address of current block.

(4) Max. data size one instruction can read (SIZE x CNT) is 128 WORD

(5) "VRD" instruction can be executed during operation

(6) For "AXIS", you can configure the axis to give an instruction. If other value is set, it produces "Error6."

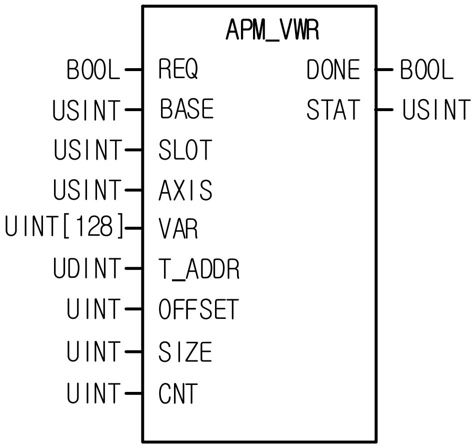
(7) If Read data size (SIZE x CNT) is o or larger than 128 WORD, error "11" occurs at STAT.

■ **Program example**

1. ST

```
INST_APM_VRD(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*),
S_ADDR:=(*UINT*),OFFSET:=(*UINT*), SIZE:=(*UINT*), CNT:=(*UINT*), DONE=>(*BOOL*), STAT=>(*UINT*),
R=>(*ARRAY[0..127]_OF_UINT*))
```

APM_VWR	Write Variable Data	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	Input REQ: requires to execute the function block BASE: Setting the base number with a module SLOT: Setting the slot number with a module AXIS: Setting an axis to instruct 0:X axis, 1:Y axis, 2:Z axis VAR : PLC device where data to write is saved T_ADDR: module internal memory head address to write data 0 ~ 12147 OFFSET : Offset between Write data blocks 0 ~ 53329 SIZE : Size of Write data block : 1 ~ 128 CNT : No. of Write data block : 1 ~ 128 Output DONE: maintains 1 after the first operation STAT: Output the error number that occurs while executing the function block

■ **Function**

(1) It commands to instruct positioning module to write parameter, operation data directly

(2) You can read data you want by configure the module internal memory address of parameter and operation data

(3) It writes the WORD data in "VAR" to module internal memory. The data are saved from internal memory position set by "T_ADDR" and the number of data is the number set by "Size". In case the number of block "CNT" is larger than 2, multiple blocks are made. At this time, head address of next block is "Offset" apart from head address of current block.

(4) Max. data size one instruction can read (SIZE x CNT) is 128 WORD

(5) "VWR" instruction can executes during operation

(6) For "AXIS", you can designate the axis to give an instruction. If other value is set, it produces "Error6."

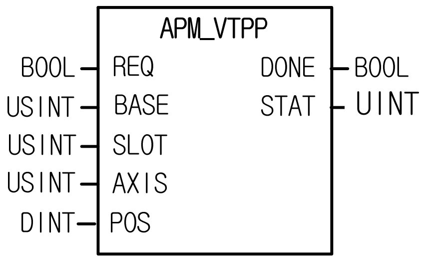
(7) If Write data size (SIZE x CNT) is o or larger than 128 WORD, error "11" occurs at STAT.

■ **Program example**

1. ST

```
INST_APM_VWR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*),
VAR:=(*ARRAY[0..127]_OF_UINT*), T_ADDR:=(*UINT*), OFFSET:=(*UINT*), SIZE:=(*UINT*), CNT:=(*UINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

<table>
<tr><td rowspan="3">APM_VTPP</td><td colspan="2">Position specified Speed/Position Switching Control</td></tr>
<tr><td>Availability</td><td>XGI, XGR</td></tr>
<tr><td>Flags</td><td></td></tr>
</table>

<table>
<tr><th>Function Block</th><th>Description</th></tr>
<tr><td>
</td><td>Input
REQ : Request for execution of function block
BASE : Set the base no. with module
SLOT : Set the slot no. with module
AXIS : Axis to command
0:X axis, 1:Y axis, 2:Z axis
POS: transfer amount
1 ~2,147,483,647
Output
DONE : Maintain 1 after first operating
STAT : Output the error no. in operation</td></tr>
</table>

■ **Function**

(1) Give "Position specified Speed/Position Switching Control" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) When the configured axis receives speed/position control switching command in speed control operation, speed control changes to position control and move by transfer amount configured by POS.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."

0:X axis, 1:Y axis, 2:Z axis

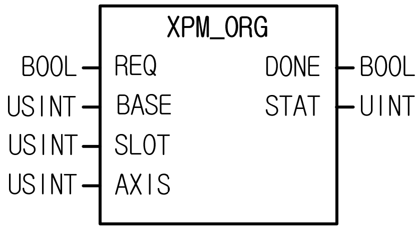
■ **Program example**

1. ST

INST_APM_VTPP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), POS:=(*DINT*), DONE=>(*BOOL*), STAT=>(*UINT*));

11.5 Positioning Function Block (XPM)

XPM_ORG	Homing Start	
	Availability	XGI, XGR
	Flags	

Function Block	Description
 	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) **Output** DONE : Maintain 1 after first operation STAT : Output the error no in operation

■ **Function**

(1) This is the command that give homing command to XPM module.

(2) This is the command to find the origin of machine by Direction, Correction, Speed, Address and Dwell set on parameter of each axis for homing according to the homing access.

(3) Give “Homing” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

XPM: 1 ~ 4 (1-axis ~ 4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

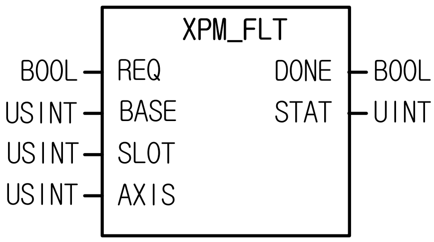
(5) If homing command executes normally, it starts homing according to “homing method” of “homing parameter”.

■ **Program example**

1. ST

```
INST_XPM_ORG(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_FLT	Floating Origin Setting	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Floating Origin” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for setting the current position as the origin by compulsion. The address value saved on homing address will be the current position.

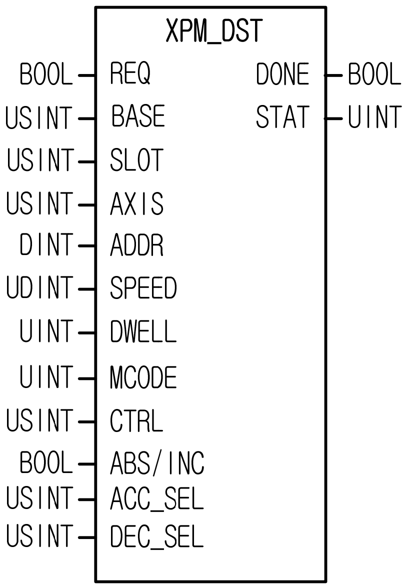
(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

```
INST_XPM_FLT(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_DST	Direct Start	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) ADDR : Destination position address setting -2147483648 ~ +2147483647 SPEED : Destination speed setting DWELL : Dwell time setting 0 ~ 65535[ms] M code : M code value setting CTRL : Control method setting 0: Position, 1: Speed, 2: Feed ABS/INC: Absolute/Relative coordibates setting 0: Absolute, 1: Relative ACC_SEL: Acc.time no. setting 0: Acc. Time 1, 1: Acc. Time 2 2: Acc. Time 3, 3: Acc. Time 4 DCC_SEL: Dec.time no. setting 0: Dec. time 1, 1: Dec. time 2 2: Dec. time 3, 3: Dec. time 4 **Output** DONE : Maintain 1 after first operation STAT : Output the error no in operation

■ **Function**

(1) Give “Direct Start” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This is for operating by setting destination position address, operation speed, dwell time, M code, control method, coordinates setting and no. of Acc./Dec time, not by operation data.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

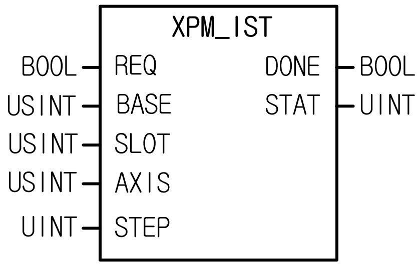
(4) If the value set on SPEED, CTRL, TIME_SEL is out of setting range, “Error11” occur on STAT.

■ **Program example**

1. ST

```
INST_XPM_DST(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), ADDR:=(*DINT*),
SPEED:=(*UDINT*), DWELL:=(*UINT*), MCODE:=(*UINT*), CTRL:=(*USINT*), ABS_INC:=(*BOOL*),
ACC_SEL:=(*USINT*), DEC_SEL:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_IST	Direct Start	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) STEP : Set the step no. to do teaching 0 ~ 400 **Output** DONE : Maintain 1 after first operation STAT : Output the error no in operation

■ **Function**

(1) Give “Indirect Start” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This is for operating by setting operation step no. of axis which set as an operation data.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(4) If the value set on STEP is out of the setting range (0~400), “Error11” arises on STAT.

(5) If the value set on STEP is 0, it operates the current step.

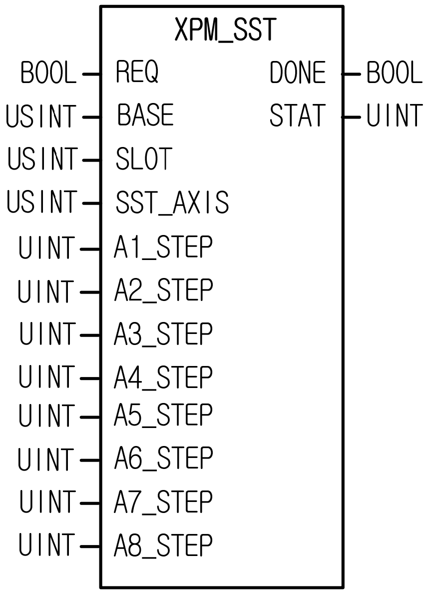
(6) Linear interpolation, circular interpolation and helical interpolation execute in indirect start by setting the control method.

■ **Program example**

1. ST

INST_APM_IST(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), STEP:=(*UINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_SST	Simultaneous Start	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module SST_AXIS : Simultaneous axis setting XPM: 0bit ~ 3bit: (1-axis ~4-axis) XGF-PN8A/B: 0bit~7bit (1-axis~8-axis) Set bit of each axis to select A1_STEP : step no. of axis1 to start A2_STEP : step no. of axis2 to start A3_STEP : step no. of axis3 to start A4_STEP : step no. of axis4 to start A5_STEP : Not use A6_STEP : Not use A7_STEP : Not use A8_STEP : Not use **Output** DONE : Maintain 1 after first operation STAT : Output the error no in operation

■ **Function**

(1) Give "Simultaneous Start" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This is for starting 2~4 axes for XPM, 2~8 axes for XGF-PN8A at once..

(3) If you set a value out of setting range, "Error6" arises. Set with each bit as follows.

7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
8-axis	7-axis-	6-axis	5-axis	4-axis	3-axis	2-axis	1-axis

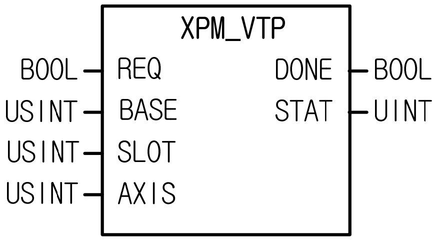
(4) Set the step no. of each axis to execute simultaneous start on A1_STEP ~ A4_STEP.

■ **Program example**

1. ST

```
INST_XPM_SST1(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), SST_AXIS:=(*USINT*),
A1_STEP:=(*UINT*), A2_STEP:=(*UINT*), A3_STEP:=(*UINT*), A4_STEP:=(*UINT*), A5_STEP:=(*UINT*),
A6_STEP:=(*UINT*), A7_STEP:=(*UINT*), A8_STEP:=(*UINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_VTP	Speed/Position Switching Control	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Speed/Position Switching Control” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) When the configured axis receives speed/position control switching command in speed control operation, speed control changes to position control and keep operating by the position value at the beginning.

(3) If this command executes, origin would be decided at the same time and it finishes the positioning after arrive at the destination position.

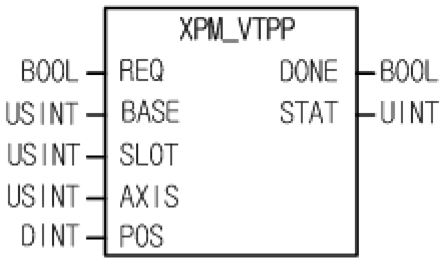
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

INST_XPM_VTP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_VTPP	Position specified Speed/Position Switching Control	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) POS: transfer amount -2,147,483,648~2,147,483,647 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give "Position specified Speed/Position Switching Control" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) When the configured axis receives speed/position control switching command in speed control operation, speed control changes to position control and move by transfer amount configured by POS.

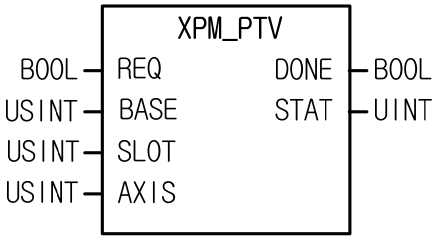
(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

INST_XPM_VTPP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), POS:=(*DINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_PTV	Position/Speed Switching Control	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Position/Speed Switching Control” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) When the configured axis is in positioning control operation, if it receives position/speed control switching command, positioning control operation changes into speed control operation and continue to operate until stop command.

(3) Once the command executes, origin would not be assigned and then operate in speed control.

(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

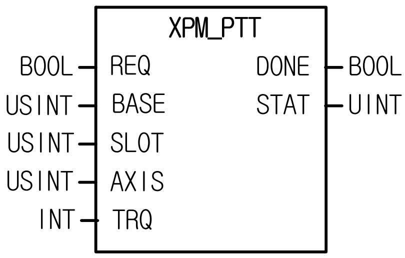
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

```
INST_XPM_PTV(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_PTT	Position/Torque Switching Control	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1 ~ 8 (1-axis ~ 8-axis) TRQ: Torque value -300~300 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Position/Speed Switching Control” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) When the configured axis is in positioning control operation, if it receives the position/torque control switching command, the positioning control operation changes into the torque control operation with the torque value in TRQ and continues to operate until stop command.

(3) The range of torque value is -300~300 and unit is [%]

(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

1 ~ 8 (1-axis ~ 8-axis)

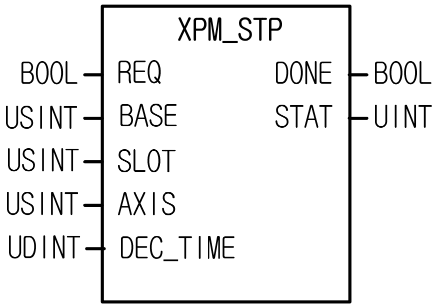
(5) This instruction is only for XGF-PN8A/B.

■ **Program example**

1. ST

INST_XPM_PTT(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), TQR:=(*INT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_STP	Deceleration Stop	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) DEC_TIME : Decelerating stop time 0: Acc./Dec. time applied when start operating 1 ~ 2147483647: 1 ~ 2147483647ms **Output** DONE : Maintain 1 after first operation STAT : Output the error no in operation

■ **Function**

(1) Give "Decelerating Stop" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) If receive the stop command by operation data, it will stop operating and continue to operate by start command.

(3) If "Decelerating Stop" executes in speed/position synchronization or CAM operation, speed/position synchronization or CAM operation stops depending on the state of the current operation control.

(4) "Decelerating Stop" executes in not only acc./dec. area but also steady speed area.

(5) Deceleration time means the time between the point of start decelerating and the point of stop and may be set to 0 ~ 2,147,483,647ms. But, if it is set to "0", it stops by the time set at the starting of operation.

(6) Decelerating time means the time between the speed limit of basic parameter and stop.

(7) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."

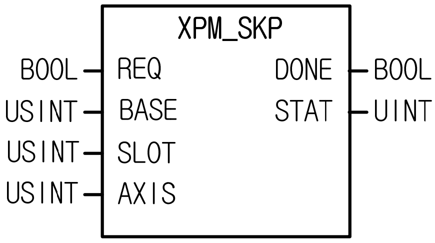
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

INST_XPM_STP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DEC_TIME:=(*UDINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_SKP	Skip Operation	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Skip Operation” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for operating the next step. That is, stop operating of the current step and then start operating the next step.

(3) Skip a step at once.

(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

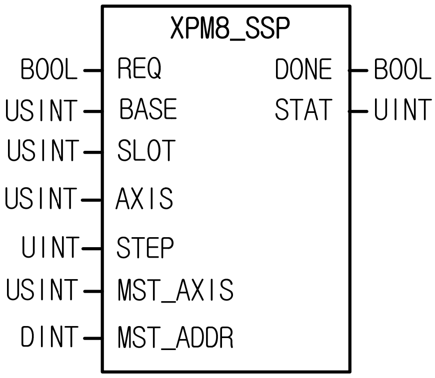
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

```
INST_XPM_SKP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_SSP	Position Synchronization	
	Availability	XGI, XGR
	Flags	

Function Block	Description
 	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) STEP : Step no. to operate 0 ~ 400 MST_AXIS : Set the main axis XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) 9: Encoder MST_ADDR : Set the position of main axis -2,147,483,648 ～ 2,147,483,647 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give "Synchronization Start" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) Operate operation step set by command axis after main axis comes to the position of synchronization.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."

XPM: 1 ~ 4 (1-axis ~4-axis),XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(4) You may set the main axis on MST_AXIS with following values. If other value is set, it produces "Error6."

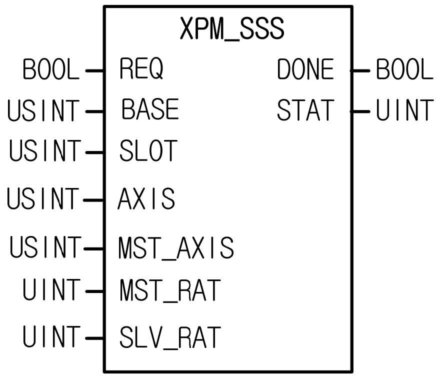
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis), 9: Encoder

■ **Program example**

1. ST

```
INST_XPM_SSP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), STEP:=(*UINT*),
MST_AXIS:=(*USINT*), MST_ADDR:=(*DINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_SSS	Speed Synchronization	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) MST_AXIS : Set main axis XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis), 9: Encoder MST_RAT : Set speed rate of main axis -32768 ~ 32767 SLV_RAT : Set speed rate of sub axis -32768 ~ 32767 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Speed Synchronization” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for operating at the operation speed ratio between main axis and subordinate axis.

(3) There is no rule about size of the speed ratio between main/sub axis. If the speed ratio of main axis is bigger than sub’s, the main axis moves faster than sub axis. If the speed ratio of sub axis is bigger than main’s, the sub axis moves faster than main.

(4) Set an axis to command. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(5) You may set the main axis on MST_AXIS with following values. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis), 9: Encoder

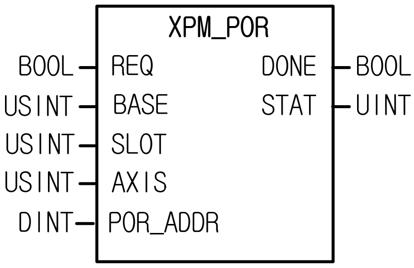
(6) The operating direction of subordinate depends on speed synchronization ratio $(\frac{Sub}{Main})$. If it is positive, operate in direction of main axis. If it is negative, operate in reverse direction of main axis.

■ **Program example**

1. ST

```
INST_XPM_SSS(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), MST_AXIS:=(*USINT*),
MST_RAT:=(*INT*), SLV_RAT:=(*INT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_POR	Position Override	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) POR_ADDR : Set a new goal position -2,147,483,648 ～ 2,147,483,647 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give "Position Override" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for changing the goal position in operation.

(3) after passing override destination position, if position override command executes position module stops and turn back to the position set on POR_ADDR.

(4) Set the destination position to modify on POR_ADDR.'

(5) Override position set on position override is absolute coordinates.

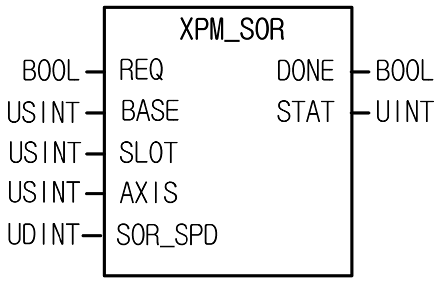
(6) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

INST_XPM_POR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), POR_ADDR:=(*DINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_SOR	Speed Override	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) SOR_SPD : Set a new operation speed value **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Speed Override” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for changing the operating speed in operation.

(3) It may be set to “%” or “Speed value (unit/time)” according to “Speed Override” value of common parameter.

(4) If unit of Speed override is %, setting range is from 1 to 65,535. It means 0.01% ~ 655.35%.

(5) If unit of speed override is speed value, the setting range is from 1 to speed limit. The speed limit is the value set on “Speed Limit” item of basic parameter and the unit of speed override is the same as unit of axis.

(6) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

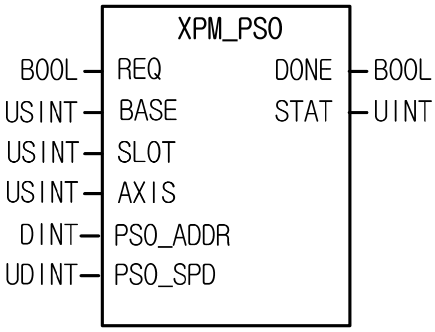
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

```
INST_XPM_SOR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), SOR_SPD:=(*UDINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_PSO	Position Assigned Speed Override	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) PSO_ADDR : The position to change speed -2,147,483,648 ~ 2,147,483,647 PSO_SPD : Set new speed value **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give "Position Assigned Speed Override" command to the axis designated as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for changing operating speed in operation after command axis arrive at definite position.

(3) The speed value set on PSO_SPD will be "% Designation" or "Speed value Designation" depending on the value set on "Speed Override" of common parameter.

(4) If unit of speed value is %, the setting range is from 1 ~ 65,535 and it means 0.01% ~ 655.35%.

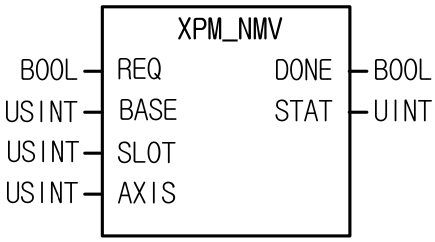
(5) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

```
INST_XPM_PSO(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), PSO_ADDR:=(*DINT*),
PSO_SPD:=(*UDINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_NMW	Continuous Operation	
	Availability	XGI, XGR
	Flags	

Function Block	Description
 	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Continuous Operation” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for command axis to continue to operate the next step without stop.

(3) If this command executes, the current step no. would be changed to the next step no. and continue to execute positioning operation at the next step speed to the goal position.

(4) Continuous Operation command only changes the current operation pattern, not changes operation data.

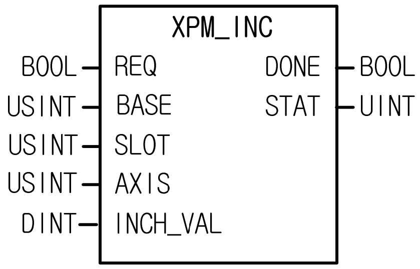
(5) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

INST_XPM_NMV(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_INC	Inching Operation	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) INCH_VAL: Amount of movement by Inching Operation -2,147,483,648 ~ 2,147,483,647 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Inching Operation” command to the axis designated as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is a kind of manual operation for process a minute movement as an operation of fixed amount.

(3) Speed of inching operation is set on manual operation parameter.

(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

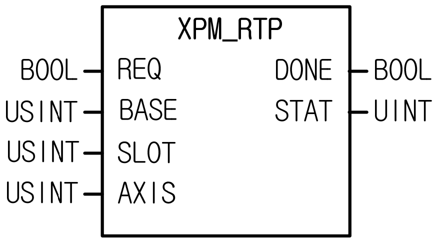
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

INST_XPM_INC(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), INCH_VAL:=(*DINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_RTP	Returning to Previous Manual Operation Position	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give "Returning to previous manual operation" command to the axis designated as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) When the position is changed by manual operation, this command may move the axis to previous manual operation position.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."

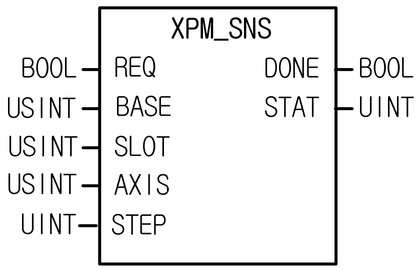
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

```
INST_XPM_RTP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

<table>
<tr><td rowspan="3">XPM_SNS</td><td colspan="2">Start Step Number Change</td></tr>
<tr><td>Availability</td><td>XGI, XGR</td></tr>
<tr><td>Flags</td><td></td></tr>
</table>

<table>
<tr><th>Function Block</th><th>Description</th></tr>
<tr><td>
</td><td>Input
REQ : Request for execution of function block
BASE : Set the base no. with module
SLOT : Set the slot no. with module
AXIS : Axis to command
XPM: 1 ~ 4 (1-axis ~4-axis)
XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)
STEP : Set the operation step no. to operate
1 ~ 400
Output
DONE : Maintain 1 after first operating
STAT : Output the error no. in operation</td></tr>
</table>

■ **Function**

(1) Give "Start Step no. Change" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for changing the operation step of command axis.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

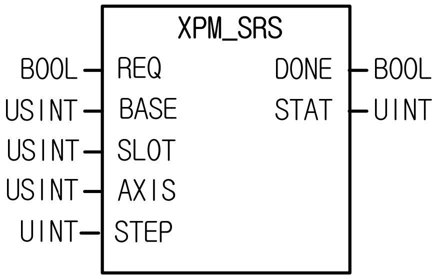
(4) Set the step no. on STEP. The setting range is 1 ~ 400, If other value is set, it produces "Error11."

■ **Program example**

1. ST

INST_XPM_SNS(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), STEP:=(*UINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_SRS	Repeat Step No. Change	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) STEP : Set the repeat step no. to change 1 ~ 400 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give "Repeat Step no. Change" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for configuring the starting step no. of repeat operation and operating from the configured operation step.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

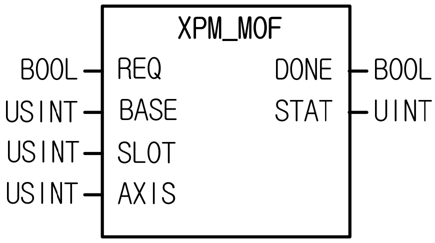
(4) Set the step no. to operate repeatedly on STEP. The setting range is 1 ~ 400, If other value is set, it produces "Error11".

■ **Program example**

1. ST

```
INST_XPM_SRS(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), STEP:=(*UINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_MOF	M code Release	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “M code Release” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) In the case that M code of parameter of each axis is set as “With” of “After”, you may turn the M code off with this command. That is, M code signal is off, M code no. is 0.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

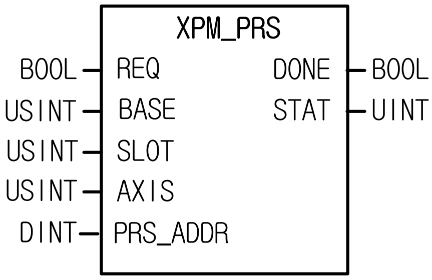
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

```
INST_XPM_MOF(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_PRS	Current Position Change	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) PRS_ADDR : Set the current position value to change. -2,147,483,648 ~ 2,147,483,647 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Basic Parameter Setting” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for changing the current position to random position. If it executes in the state of non-origin, the origin signal would be on and the current position would be set as setting value (PRS_ADDR).

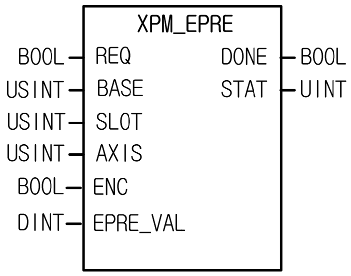
(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

```
INST_XPM_PRS(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), PRS_ADDR:=(*DINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_EPRE	Encoder Value Preset	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) ENC : Encoder no. (Always 0) 0: Encoder EPRE_VAL : Set the value of encoder preset -2147483648 ~ 2147483647 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ Function

(1) Give “Encoder Preset” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for changing the current value of encoder to the value set on EPRE_VAL

(3) Set the encoder to preset on ENC and it has to be 0 in APM module of XPM.

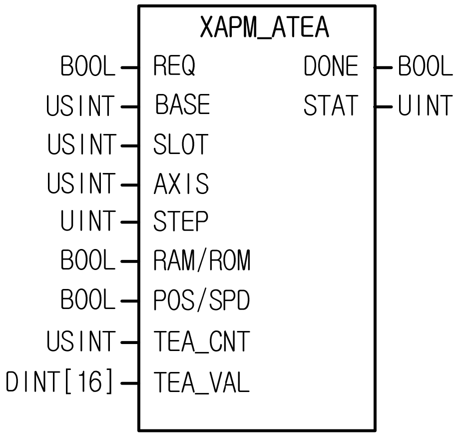
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ Program example

1. ST

INST_XPM_EPRE(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), ENC:=(*BOOL*), EPRE_VAL:=(*DINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_ATEA	Teaching Array	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) STEP : Set the step no. to do teaching 0 ~ 400 RAM/ROM : Selection of RAM/ROM teaching 0 : RAM teaching, 1 : ROM teaching POS/SPD : Selection of position/speed teaching 0 : Position, 1 : Speed TEA_CNT : Set the no. of data to do teaching 1 ~ 16 TEA_VAL : Set the teaching value **Output** DONE : Maintain 1 after first operation STAT : Output the error no in operation

■ **Function**

(1) Give “Teaching Array” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) Speed teaching is for user to use random speed value in a operation data of specified step and position teaching is for user to use random position value in a operation data of specified operation step.

(3) This command is for modifying maximum 16 destination positions/speed value at once with teaching array function block.

(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(5) You may set step no.(0~400) of operation data on STEP. If other value is set, it produces “Error11.”

(6) You may set the no. of data to do teaching on TEA_CNT and do teaching max. 16. If other value is set, it produces “Error11.

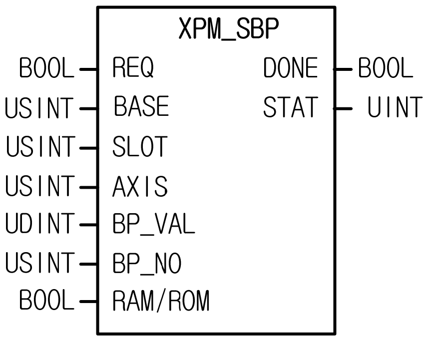
(7) Parameter value modified by teaching command and setting RAM/ROM as “0” is valid within power connection. If you want to keep the parameter without power connection, execute teaching command with setting “1” on RAM/ROM or save the modified parameter value on FRAM with XPM_WRT (Parameter/Operation Data Saving command) after teaching.

■ **Program example**

1. ST

```
INST_XPM_ATEA(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), STEP:=(*UINT*),
RAM_ROM:=(*BOOL*), POS_SPD:=(*BOOL*), TEA_CNT:=(*USINT*), TEA_VAL:=(*ARRAY[0..15]_OF_DINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_SBP	Basic Parameter Teaching	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) BP_VAL : Basic parameter to change BP_NO : Item no. of basic parameter to change RAM/ROM : Method of parameter save 0: save on RAM 1: save on ROM **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Basic Parameter Teaching” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) Parameter value modified by basic parameter teaching command and setting RAM/ROM to “0” is valid within power connection. If you want to keep the parameter without power connection, execute basic parameter teaching command with setting RAM/ROM as “1” or save the modified parameter value on FRAM with XPM_WRT (Parameter/Operation Data Saving command) after basic parameter teaching.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(4) The value that needs to be set in basic parameter is as follows.

Value	Items	Setting Range
1	Speed Limit	㎜ : 1 ～ 2,147,483,647 [X10^{-2}㎜/min] Inch : 1 ～ 2,147,483,647 [X10^{-3}Inch/min] degree : 1 ～ 2,147,483,647 [X10^{-3} degree/min] pulse : 1 ～ 2,147,483,647 [pulse/sec]
2	Acc. Time 1	1 ~ 2,147,483,647 [ms]
3	Acc. Time 2	
4	Acc. Time 3	
5	Acc. Time 4	
6	Dec. Time 1	1 ~ 2,147,483,647 [ms]
7	Dec. Time 2	
8	Dec. Time 3	
9	Dec. Time 4	
10	Urgent stop Dec. Time	1 ~ 2,147,483,647 [ms]
11	Demultiply ouput pulse/rotation	1 ~ 200,000,000
12	Transfering Distance/rotation	
13	Unit	0:Pulse, 1:mm, 2:Inch, 3:Degree
14	Unit assignment	0: x 1, 1: x 10, 2: x 100, 3: x 1000
15	Unit for speed command	0: unit/time, 1: rpm
16	Bias speed	1 ~ speed limit
17	Pulse output mode	0: CW/CCW, 1: PLS/DIR, 2: PHASE

■ **Program example**

1. ST

```
INST_XPM_SBP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
BP_VAL:=BP_UDINT, BP_NO:=BP_USINT, RAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

XPM_SEP	Extended Parameter Teaching	
	Availability	XGI, XGR
	Flags	

Function Block	Description
XPM_SEP BOOL — REQ DONE — BOOL USINT — BASE STAT — UINT USINT — SLOT USINT — AXIS DINT — EP_VAL USINT — EP_NO BOOL — RAM/ROM	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) EP_VAL : Parameter value to modify EP_NO : Item no. of parameter to modify RAM/ROM : Method for saving parameter 0: Save on RAM 1: Save on ROM **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give "Extended Parameter Teaching" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) Parameter value modified by extended parameter teaching command and setting RAM/ROM to "0" is valid within power connection. If you want to keep the parameter without power connection, execute extended parameter teaching command with setting RAM/ROM as "1" or save the modified parameter value on FRAM with XPM_WRT (Parameter/Operation Data Saving command) after extended parameter teaching.

(3 It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(4) The extended parameter items and setting values are as follows.

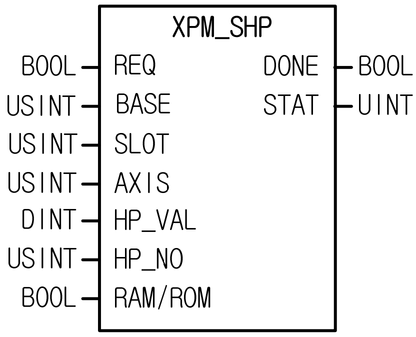
Value	Item	Setting Range
1	Software high limit	㎜ :-2147483648 ~ 2147483647[X10^{-4}㎜] Inch:-2147483648 ~ 2147483647[X10^{-5}Inch] degree:-2147483648 ~ 2147483647[X10^{-5}degree] pulse:-2147483648 ~ 2147483647[pulse]
2	Software low limit	
3	Backlash compensation amount	mm: 0 ~ 65,535[X10^{-4}㎜] inch: 0 ~ 65,535[X10^{-5}Inch] degree: 0 ~ 65,535[X10^{-5}degree] pulse: 0 ~ 65,535[pulse]
4	Positioning end output time	0 ~ 65,535[ms]
5	S-Curve ratio	1 ~ 100
6	Position to interpolate circular arc of 2axis linear interpolation	mm: 0 ~ 2147483647[X10^{-4}㎜] Inch: 0 ~ 2147483647[X10^{-5}Inch] degree: 0 ~ 2147483647[X10^{-5}degree] pulse: 0 ~ 2147483647[pulse]
7	Acc./dec. pattern	0: Trapezoid operating, 1: S-curve operating
8	M code mode	0: None, 1: With, 2: After
9	Detection of High/Low limit in speed control	0: Not detect, 1: Detect
10	Condition for positioning completion	0: Dwell time 1: In-position 2: Dwell time AND In-position 3: Dwell time OR In-position
11	Positioning method of interpolation continuous operation	0: passage of goal position, 1: passage of near position
12	2axis linear interpolation continuous operation circular arc interpolating	0: No circular interpolating, 1: Circular interpolating continuous operation
13	External speed/position control switching	0: Not permit, 1: Permit
14	Selection of external emergent stop/dec stop	0: Emergent stop, 1: Dec. Stop
15	Coordinates of positioning speed override	0: Absolute, 1: Relative
16	Pulse output direction	0: Forward, 1: Reverse

■ **Program example**

1. ST

```
INST_XPM_SEP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
EP_VAL:=EP_DINT, EP_NO:=NO_USINT, RAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

XPM_SHP	Homing Parameter Teaching	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) HP_VAL : Homing parameter value to modify HP_NO : Item no. of homing parameter to modify RAM/ROM : Method for saving parameter 0: Save on RAM 1: Save on ROM **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give "Homing Parameter Setting" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) Parameter value modified by homing parameter teaching command and setting RAM/ROM to "0" is valid within power connection. If you want to keep the parameter without power connection, execute homing parameter teaching command with setting RAM/ROM as "1" or save the modified parameter value on FRAM with XPM_WRT (Parameter/Operation Data Saving command) after homing parameter teaching.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(4) The homing parameter items and setting ranges are as follows.

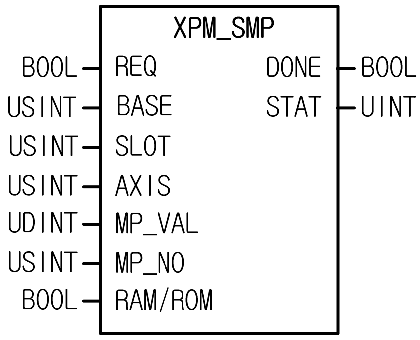
Setting value	Items	Setting Range
1	Homing position	㎜ : -2147483648 ~ 2147483647 [X10^{-4}㎜] Inch : -2147483648 ~ 2147483647 [X10^{-5}Inch] degree : -2147483648 ~ 2147483647 [X10^{-5}degree] pulse : -2147483648 ~ 2147483647 [pulse]
2	High speed for homing	㎜ : 1 ~ 2,147,483,647 [X10^{-2}㎜/min] Inch : 1 ~ 2,147,483,647 [X10^{-3}Inch/min] degree : 1 ~ 2,147,483,647 [X10^{-3} degree/min] pulse : 1 ~ 2,147,483,647 [pulse/sec]
3	Low speed for homing	
4	Homing Acc. Time	0 ~ 2,147,483,647 [ms]
5	Homing Dec. Time	
6	Homing Dwell Time	0 ~ 65,535[ms]
7	Revision amount of origin	㎜ : -2147483648 ~ 2147483647 [X10^{-3}㎜] Inch : -2147483648 ~ 2147483647 [X10^{-5}Inch] degree : -2147483648 ~ 2147483647 [X10^{-5}degree] pulse : -2147483648 ~ 2147483647 [pulse]
8	Restart time for homing	0 ~ 65,535[ms]
9	Homing mode	0:Near origin/Origin(Off), 1:Near origin/Origin(On), 2:High&Low limit/Origin, 3:Near origin, 4:High speed origin, 5:High/Low limit, 6:Origin
10	Homing direction	0:Forward, 1:Reverse

■ **Program example**

1. ST

```
INST_XPM_SHP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
HP_VAL:=HP_DINT, HP_NO:=NO_USINT, ㄱㄱ RAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

XPM_SMP	Manual Operation Parameter Teaching	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) MP_VAL : Manual operation parameter value to modify MP_NO : Item no. of manual operation parameter to modify RAM/ROM : Method for saving parameter 0: Save on RAM 1: Save on ROM **Output** DONE : Maintain 1 after first operation STAT : Output the error no in operation

■ **Function**

(1) Give "Manual Operation Parameter Setting" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) Parameter value modified by manual operation parameter teaching command and setting RAM/ROM to "0" is valid within power connection. If you want to keep the parameter without power connection, execute manual operation parameter teaching command with setting RAM/ROM as "1" or save the modified parameter value on FRAM with XPM_WRT (Parameter/Operation Data Saving command) after manual operation parameter teaching.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(4) The manual operation parameter items and setting ranges are as follows.

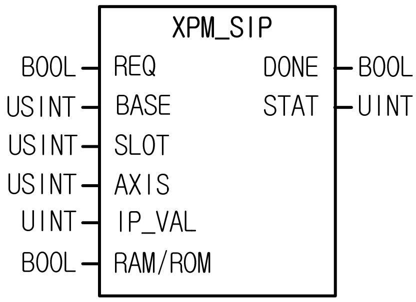
Setting Value	Items	Setting Range
1	JOG high speed	㎜ : 1 ~ 2,147,483,647 [X10^{-2}㎜/min] Inch : 1 ~ 2,147,483,647 [X10^{-3}Inch/min] degree : 1 ~ 2,147,483,647 [X10^{-3} degree/min] pulse : 1 ~ 2,147,483,647 [pulse/sec]
2	JOG low speed	
3	JOG acc. time	0 ~ 2,147,483,647 [ms]
4	JOG dec, time	
5	Inching speed	㎜ : 1 ~ 65,535[X10^{-2}㎜/min] Inch : 1 ~ 65,535[X10^{-3}Inch/min] degree : 1 ~ 65,535[X10^{-3} degree/min] pulse : 1 ~ 65,535[pulse/sec]

■ **Program example**

1. ST

```
INST_XPM_SMP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
MP_VAL:=MP_UDINT, MP_NO:=NO_USINT, RAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

XPM_SIP	I/O Signal Parameter Teaching	
	Availability	XGI, XGR
	Flags	

Function Block	Description
 	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) IP_VAL : External signal parameter value to modify Set the corresponding signal for each Bit RAM/ROM : Method for saving parameter 0: Save on RAM 1: Save on ROM **Output** DONE : Maintain 1 after first operation STAT : Output the error no in operation

■ **Function**

(1) Give “Input Signal Parameter Setting” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) Parameter value modified by input signal parameter teaching command and setting RAM/ROM to “0” is valid within power connection. If you want to keep the parameter without power connection, execute input signal parameter teaching command with setting RAM/ROM as “1” or save the modified parameter value on FRAM with XPM_WRT (Parameter/Operation Data Saving command) after input signal parameter teaching.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(4) The setting value of each setting area of external signal has the meaning as below.

0 : A contact, 1 : B contact

(5) The manual operation parameter items and setting values are as follows.

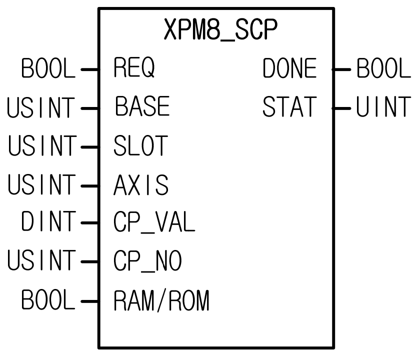
Bit	Signal
0	High limit signal
1	Low limit signal
2	Near origin signal
3	Origin signal
4	Emergent stop/Dec. stop signal
5	Speed/Position control switching siganl
6	Drive ready signal
7	In-position signal
8	Deviation counter clear output signal
9 ~ 15	Not Use

■ **Program example**

1. ST

```
INST_XPM_SIP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
IP_VAL:=IP_WORD, RAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_SCP	Common Parameter Teaching	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) CP_VAL : Common parameter value to modify CP_NO : Item no. of common parameter to modify RAM/ROM : Method for saving parameter 0: Save on RAM 1: Save on ROM **Output** DONE : Maintain 1 after first operation STAT : Output the error no in operation

■ **Function**

(1) Give “Common Parameter Setting” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) Parameter value modified by common parameter teaching command and setting RAM/ROM to “0” is valid within power connection. If you want to keep the parameter without power connection, execute common parameter teaching command with setting RAM/ROM as “1” or save the modified parameter value on FRAM with XPM_WRT (Parameter/Operation Data Saving command) after common parameter teaching.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(4) The common parameter items and setting values are as follows.

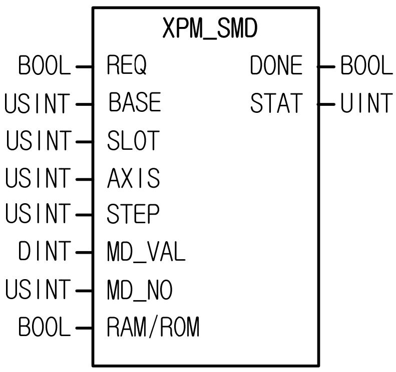
Setting Value	Items	Setting values
1	Speed override	0 : % designation, 1 : speed designation
2	Mode for encoder pulse input	0:CW/CCW 1multiply, 1:CW/CCW 2 multiply 2:PULSE/DIR 1 multiply, 3:PULSE/DIR 2 multiply 4:PHASE A/B 1 multiply, 5:PHASE A/B 2 multiply 6:PHASE A/B 4 multiply
3	Maximum value of encoder	-2147483648 ~ 2147283647
4	Minimum value of encoder	
5	Pulse output level	0 : Low Active, 1 : High Active

■ Program example

1. ST

```
INST_XPM_SCP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
CP_VAL:=CP_DINT, CP_NO:=NO_USINT, RAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

<table>
<tr><td rowspan="3">XPM_SMD</td><td colspan="2">Operation Data Teaching</td></tr>
<tr><td>Availability</td><td>XGI, XGR</td></tr>
<tr><td>Flags</td><td></td></tr>
</table>

<table>
<tr><th>Function Block</th><th>Description</th></tr>
<tr><td>
</td><td>Input
REQ : Request for execution of function block
BASE : Set the base no. with module
SLOT : Set the slot no. with module
AXIS : Axis to command
XPM: 1 ~ 4 (1-axis ~4-axis)
XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)
STEP : Step no. to modify
0 ~ 400
MD_VAL : Operation data value to modify
MD_NO : Item no. of operation data to modify
RAM/ROM : Method for saving parameter
0: Save on RAM
1: Save on ROM
Output
DONE : Maintain 1 after first operation
STAT : Output the error no in operation</td></tr>
</table>

■ **Function**

(1) Give “Operation Data Teaching” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) Parameter value modified by operation data teaching command and setting RAM/ROM to “0” is valid within power connection. If you want to keep the parameter without power connection, execute operation data teaching command with setting RAM/ROM as “1” or save the modified parameter value on FRAM with XPM_WRT (Parameter/Operation Data Saving command) after operation data teaching.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(4) The operation data items and setting range are as follows.

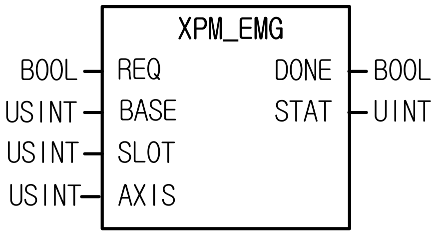
<table>
<tr><th>Setting value</th><th>Items</th><th>Setting Range</th></tr>
<tr><td>1</td><td>Goal position</td><td>㎜ : -2147483648 ~ 2147483647 [X10^{-4}㎜]
Inch : -2147483648 ~ 2147483647 [X10^{-5}Inch]
degree : -2147483648 ~ 2147483647 [X10^{-5}degree]
pulse : -2147483648 ~ 2147483647 [pulse]</td></tr>
<tr><td>2</td><td>Auxiliary position for circular interpolation</td><td>-2147483648 ~ 2147483647</td></tr>
<tr><td>3</td><td>Operating speed</td><td>㎜ : 1 ~ 2,147,483,647 [X10^{-2}㎜/min]
Inch : 1 ~ 2,147,483,647 [X10^{-3}Inch/min]
degree : 1 ~ 2,147,483,647 [X10^{-3} degree/min]
pulse : 1 ~ 2,147,483,647 [pulse/sec]</td></tr>
<tr><td>4</td><td>Dwell time</td><td>0 ~ 65,535[ms]</td></tr>
<tr><td>5</td><td>M code no.</td><td>0 ~ 65,535</td></tr>
<tr><td>6</td><td>Sub axis setting</td><td>Bit unit setting<table><tr><td>Bit 7</td><td>Bit 6</td><td>Bit 5</td><td>Bit 4</td><td>Bit 3</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr><tr><td>axis8</td><td>axis7</td><td>axis6</td><td>axis5</td><td>axis4</td><td>axis3</td><td>axis2</td><td>axis1</td></tr></table></td></tr>
<tr><td>7</td><td>Helical interpolation axis</td><td>0, axis1 ~ axis4 (0: General circular interpolation)</td></tr>
<tr><td>8</td><td>The no. of turn for circular interpolation</td><td>0~65,535</td></tr>
<tr><td>9</td><td>Coordinates</td><td>0:absolute, 1:relative</td></tr>
<tr><td>10</td><td>Control method</td><td>0:Abbreviation position control, 1:Abbreviation speed control,
2:Abbreviation Feed control, 3:linear interpolation, 4:circular interpolation</td></tr>
<tr><td>11</td><td>Operating method</td><td>0:single, 1:repeat</td></tr>
<tr><td>12</td><td>Operating pattern</td><td>0:end, 1:go on, 2:continue</td></tr>
<tr><td>13</td><td>Size of circular arc</td><td>0:circular arc<180 1:circular arc>=180</td></tr>
<tr><td>14</td><td>Acc. No.</td><td>0 ~ 3</td></tr>
<tr><td>15</td><td>Dec. No.</td><td>0 ~ 3</td></tr>
<tr><td>16</td><td>Method of circular interpolation</td><td>0:middle point, 1:center point, 2:radius</td></tr>
<tr><td>17</td><td>Direction of circular interpolation</td><td>0:CW, 1:CCW</td></tr>
</table>

■ **Program example**

1. ST

```
INST_APM_SMD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
STEP:=STEP_UINT, MD_VAL:=MD_DINT, MD_NO:=NO_USINT, RAM_ROM:=RAM_ROM_BOOL,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_EMG	Emergency Stop	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Emergency Stop” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for immediate stop. The axis to execute this command will stop.

(3) Dec. time of emergent stop is the time set on “Dec. time of Emergent stop” of basic parameter.

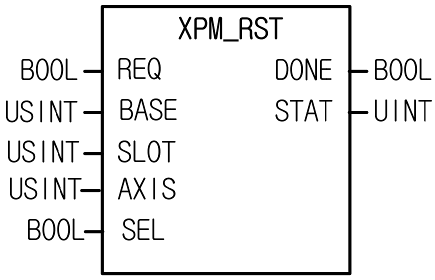
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

```
INST_XPM_EMG(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_RST	Error Reset	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) SEL : Select axis error/common error 0:axis error (Always 0) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Error Reset” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(3) This is for resetting the errors.

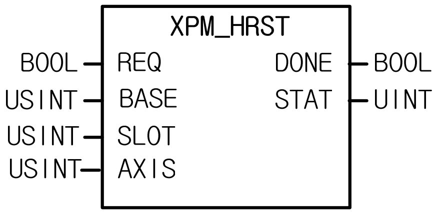
(4) Select the kind of error to reset on SEL. If it is set to 0, reset the errors of each axis. XGF series has to be set 0.

■ **Program example**

1. ST

INST_XPM_RST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, SEL:=SEL_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);

XPM_HRST	Error History Reset	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Error History Reset” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

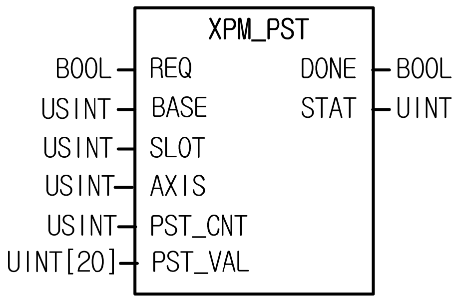
(3) If errors arise, Max.10 errors are saved on module. This command is for resetting error history.

■ **Program example**

1. ST

INST_XPM_HRST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);

XPM_PST	Point Start	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) PST_CMT : Set the no. of step for point operation 1 ~ 20 PST_VAL : Set the step no. for point operation 0 ~ 400 **Output** DONE : Maintain 1 after first operation STAT : Output the error no in operation

■ **Function**

(1) Give “Point start” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(3) This is for when operating PTP(Point to Point), operate continuously by setting max. 20 operation steps.

(4) Point operation may be executed with max. 20 point steps. Therefore, you may use the parameter which has 20 elements and like UNIT arrangement.

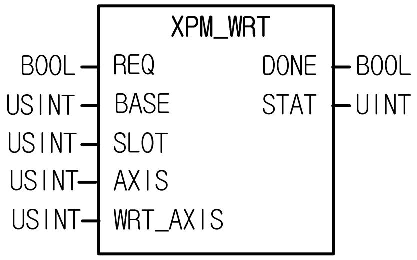
(5) If other value is set , it produces “Error6.

■ **Program example**

1. ST

```
INST_XPM_PST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
PST_CNT:=CNT_USINT, PST_VAL:=ARY_PST, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

<table>
<tr><td rowspan="3">XPM_WRT</td><td colspan="2">Saving Parameter/Operation Data</td></tr>
<tr><td>Availability</td><td>XGI, XGR</td></tr>
<tr><td>Flags</td><td></td></tr>
</table>

<table>
<tr><th>Function Block</th><th>Description</th></tr>
<tr><td>
</td><td>Input
REQ : Request for execution of function block
BASE : Set the base no. with module
SLOT : Set the slot no. with module
AXIS : Axis to command
XPM: 1 ~ 4 (1-axis ~4-axis)
XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)
XPM_WRT_AXIS : Saving axis setting
(by setting bit)
XPM: 0bit ~ 3bit: 1-axis ~ 4-axis
XGF-PN8A: 0bit ~ 7bit (1-axis ~ 8-axis)
Output
DONE : Maintain 1 after first operation
STAT : Output the error no in operation</td></tr>
</table>

■ **Function**

(1) Give “Basic Parameter Setting” command to the axis designated as the axis of positioning module with BASE (Base no. of positioning module) and SLOT (Slot no. of positioning module).

(2) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(3) If function block executes normally, the current operation parameter and data which saved on WRT_AXIS are saved on FRAM and maintain the data without the power connection.

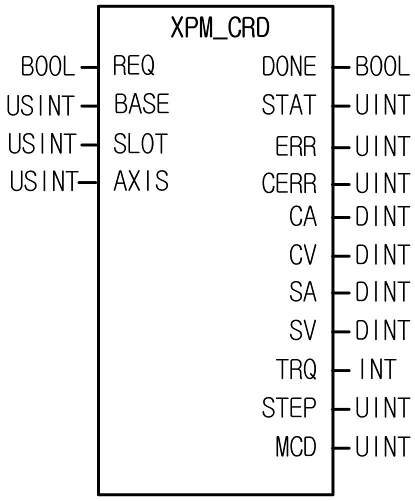
(4) In case of modifying the CAM data with XPM_VWR instruction, when you execute XPM_WRT, the modified data saves in FLASH.

■ **Program example**

1. ST

INST_XPM_WRT(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, WRT_AXIS:=WRT_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);

XPM_CRD	Operation Information Read	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation ERR : Display axis error CERR : Display common error CA : Display the command position CV : Display the command speed SA : Display the current position SV : Display the current speed TRQ: Display the current torque STEP : Display step no. of the current operation data MCD : Display the current M code value

■ **Function**

(1) Read the axis state of current operation configured in the axis of configured positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) The operation information is saved in parameter set on output of function block.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”.
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(4) You can monitor command position, command speed, current position, current speed, torque, operation data no. and M code value of axis already set through reading them or use them as a condition in user’s program.

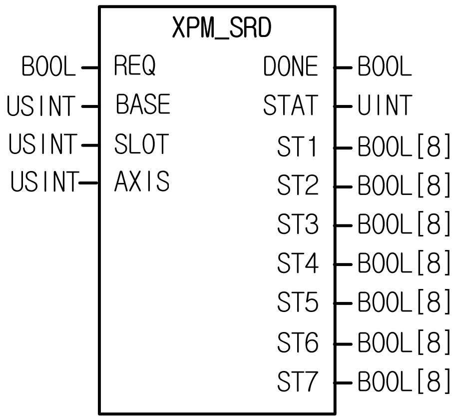
(5) “-” speed displayed as command speed(CV) or current speed(SV) means reverse direction.

■ **Program example**

1. ST

```
INST_XPM_CRD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT, ERR=>ERR_UINT, CERR=>CERR_UINT, CA=>CA_DINT,
CV=>CV_UDINT, SA=>SA_DINT, SV=>SV_DINT, TRQ=>TRQ_INT, STEP=>STEP_UINT, MCD=>MCD_UINT);
```

XPM_SRD	Operation State Read	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation ST1 : State 1 ST2 : State 2 ST3 : State 3 ST4 : State 4 ST5 : State 5 ST6 : State 6 ST7 : State 7

■ **Function**

(1) Give “Bit Information of Current operation reading” command to the axis designated as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) The bit information about the state of current operation is saved in parameter set on ST1 ~ ST7.

(3) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(4) The contents of output parameters, ST1 ~ ST7 are important information necessarily applied in the program.

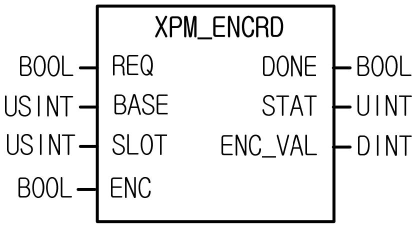
	Bit	Description	Bit	Description
ST1	[0]	Operating(0:STOP, 1:BUSY)	[4]	Origin fix state (0:Uncompletion, 1:Completion)
	[1]	Error state	[5]	-
	[2]	Positioning completion	[6]	Stop
	[3]	Mcode On signal(0:Off, 1:On)	[7]	-
ST2	[0]	High limit detection	[4]	In acceleration
	[1]	Low limit detection	[5]	In stable speed
	[2]	Emergent Stop	[6]	In deceleration
	[3]	Direction(0:Forward, 1:Reverse)	[7]	In dwell
ST3	[0]	Axis1 in positioning control	[4]	In circular interpolation operation
	[1]	Axis1 in speed control	[5]	In homing operation
	[2]	In linear interpolation	[6]	In position synchronous start operation
	[3]	-	[7]	In speed synchronous start operation
ST4	[0]	In jog operation	[4]	In previous position of manual operation returning operation
	[1]	-	[5]	In CAM control operation
	[2]	In inching operation	[6]	In Feed control operation
	[3]	-	[7]	In ellipse interpolation operation
ST5	[0]	Main axis information 1 ~ 4: axis1 ~ axis4 9: Encoder	[4]	Axis state(0:Main axis, 1: sub axis)
	[1]		[5]	-
	[2]		[6]	-
	[3]		[7]	-
ST6	[0]	Emergent stop/Dec. stop signal	[4]	High limit signal
	[1]	-	[5]	Low limit signal
	[2]	-	[6]	Origin signal
	[3]	-	[7]	Near origin signal
ST7	[0]	Switching signal of Speed/Position control	[4]	In-position signal
	[1]	-	[5]	Declination counter clear output signal
	[2]	-	[6]	-
	[3]	Drive ready signal	[7]	-

■ **Program example**

1. ST

INST_XPM_SRD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT, ST1=>ARY_ST1, ST2=>ARY_ST2, ST3=> ARY_ST3, ST4=> ARY_ST4, ST5=> ARY_ST5, ST6=> ARY_ST6, ST7=> ARY_ST7);

XPM_ENCRD	Encoder Value Read	
	Availability	XGI, XGR
	Flags	

Function Block	Description
 	**Input** REQ : Resquest for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module ENC : Encoder no. (Always 0) 0: Encoder **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation ENC_VAL : Current value of encoder

■ **Function**

(1) Give "Encoder Reading" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) The current encoder value is displayed on ENC_VAL

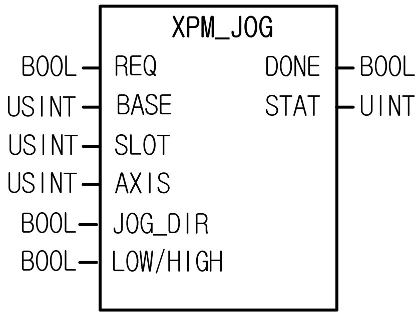
(3) Set the encoder want to read on ENC, it has to be always 0 in XPM positioning module.

■ **Program example**

1. ST

```
INST_XPM_ENCRD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, ENC:=ENC_BOOL,
DONE=>DONE_BOOL, STAT=>STAT_UINT, ENC_VAL=>ENC_UDINT);
```

XPM_JOG	JOG Operation	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) JOG_DIR : Set the direction of JOG operation 0:Forward, 1:Reverse LOW/HIGH : Set the speed of JOG operation 0:Low speed, 1:High speed **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “JOG Operation” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for checking operation of system, wiring and address for teaching. It may be used in High/Low speed.

(3) The operating condition of JOG operation function block is Level type. That is, when the condition of input parameter (REQ) is ON, pulse is outputted by setting value.

(4) If the value of LOW/HIGH is changed, the speed changes without stop and if the value of JOG_DIR is changed, it changes the direction after decelerating stop.

(5) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

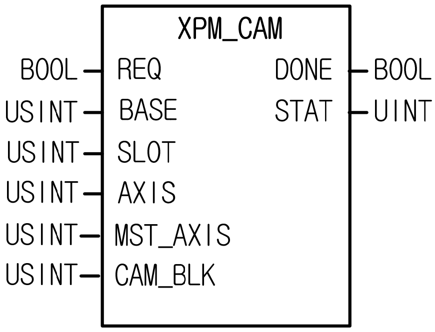
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

```
INST_XPM_JOG(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, JOG_DIR:=JOG_BOOL,
LOW_HIGH:=LOW_HIGH_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_CAM	CAM Operation	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) MST_AXIS : Set main axis XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) 9: Encoder CAM_BLK : Set CAM block 1 ~ 8: Block1 ~ Block8 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

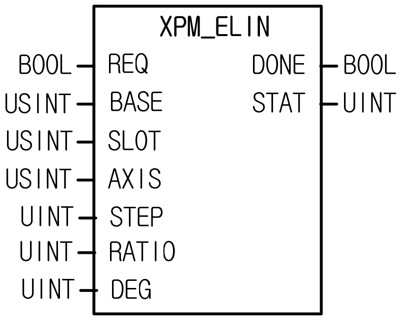
(1) Give “CAM Operation” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).
(2) Execute CAM operation with CAM main axis and CAM data block.
(3) When executing CAM operation, sub axis indicates that it is in operation but it does not work actually. When main axis starts, the motor starts working according to the data value of CAM data block which already set on CAM block (CAM_BLK)
(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)
(5) Set main axis of CAM operation at MST_AXIS. If other value is set, it produces “Error11.”.
(6) Set CAM block number in CAM_BLK and available value is as follows. If other value is set, it produces “Error11.”
1 ~ 8 : block1 ~ block8
(7) CAM data sets on positioning package and you sets max. 8 blocks.

■ **Program example**

1. ST

```
INST_XPM_CAM(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
MST_AXIS:=MST_AXIS_USINT, CAM_BLK:=CAM_BLK_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_ELIN	Ellipse Interpolation	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) STEP : Step no. to operate RATIO : Ellipse ratio(%) DEG : Operating angle **Output** DONE : Maintain 1 after first operation STAT : Output the error no in operation

■ **Function**

(1) Give “Ellipse Interpolation” command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This is the command that execute ellipse interpolation to the configured step as much as the angle set on DEG in the ratio of it which set on RATIO.

(3) Ellipse interpolation is that distort operation data of the step already set at the rate already set on RATIO to execute ellipse interpolation. Therefore, the step of operation data set on STEP has to be set in accordance with circular interpolation control.

(4) Ellipse rate range from 1 to 65535, it has [X10^{-2}%] as its unit. If you set 65535, the rates is 655.35%.

(5) Operation angle range from 1 to 65535, it has [X10^{-1} degree] as its unit. If you set 3650, the angle is 365.0

(6) It can set an axis to instruct and the value is as follows. If other value is set, it produces “Error6.”

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

■ **Program example**

1. ST

```
INST_XPM_ELIN(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
STEP:=STEP_UINT, RATIO:=RATIO_UINT, DEG:=DEG_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_SSSP	Position Assigned Speed Synchronization	
	Availability	XGI, XGR
	Flags	

Function Block

```
              XPM_SSSP
BOOL  — REQ              DONE — BOOL
USINT — BASE             STAT — UINT
USINT — SLOT
USINT — AXIS
USINT — MST_AXIS
UINT  — MST_RAT
UINT  — SLV_RAT
DINT  — POS
```

Description

Input

REQ : Request for execution of function block
BASE : Set the base no. with module
SLOT : Set the slot no. with module
AXIS : Axis to command
XPM: 1 ~ 4 (1-axis ~4-axis)
XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)
MST_AXIS : Set main axis
XPM: 1 ~ 4 (1-axis ~4-axis)
XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)
9: Encoder
MST_RAT : Set speed rate of main axis
-32768 ~ 32767
SLV_RAT : Set speed rate of sub axis
-32768 ~ 32767
POS : Destination position
-2,147,483,648 ~ 2,147,483,647

Output

DONE : Maintain 1 after first operating
STAT : Output the error no. in operation

■ **Function**

(1) Give "Position Assigned Speed Synchronization" command to the axis configured as the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is for operating at the operation speed ratio between main axis and subordinate axis. It stops operating when the position of sub axis come to the position set on POS.

(3) There is no rule about size of the speed ratio between main/sub axis. If the speed ratio of main axis is bigger than sub's, the main axis moves faster than sub. If the speed ratio of sub axis is bigger than main's, the sub axis moves faster than main.

(4) It can set an axis to instruct and the value is as follows. If other value is set, it produces "Error6."
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

(5) You may set the main axis on MST_AXIS with following values. If other value is set, it produces "Error6
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis), 9: Encoder

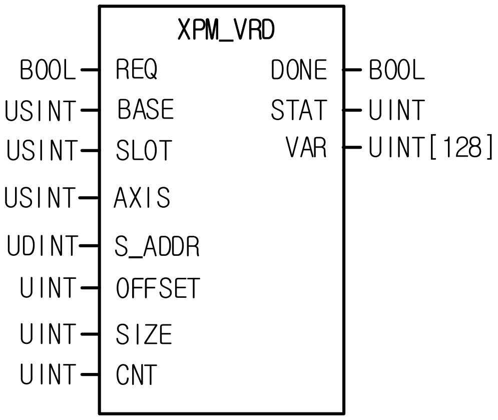
(6) The operating direction of subordinate depends on speed synchronization ratio $(\frac{Sub}{Main})$. If it is positive, operate in direction of main axis. If it is negative, operate in reverse direction of main axis.

■ **Program example**

1. ST

```
INST_XPM_SSSP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
MST_AXIS:=AXIS_USINT, MST_RAT:=MST_INT, SLV_RAT:=SLV_INT, POS:=POS_DINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

XPM_VRD	Position Assigned Speed Synchronization	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) S_ADDR : Module internal memory head address of Read Data 0 ~ 53329 OFFSET : Offset between Read Data blocks 0 ~ 54217 * XGF-PNxB: 0 ~ 65535 SIZE : Block size of Read data 1 ~ 128 CNT : No. of Read Data block 1 ~ 128 **Output** DONE : Maintain 1 after first operation STAT : Output the error no. in operation VAR : PLC device where Read Data is saved

■ **Function**

(1) Gives “Read parameter, operation data, CAM data directly” command to positioning module.

(2) You read data you want by configuring module internal memory address of parameter, operation data, CAM data directly.

(3) It reads the positioning module internal memory from the position set by “S_ADDR” by WORD unit and save them in the device set by “VAR”. The number of data to read is the number set by “Size”. In case “CNT” is larger than 2, it reads multiple data blocks and save them in the device set by “VAR” in order. At this time, head address of next block is “Offset” apart from head address of current block.

(4) Max. data size (SIZE x CNT) you can read with one command is 128 word.

(5) “Read Variable Data” command can execute in operation.

(6) You can set an axis to command in “AXIS” and the following value is available. If other value is set, it produces “Error6.”appears.

XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)

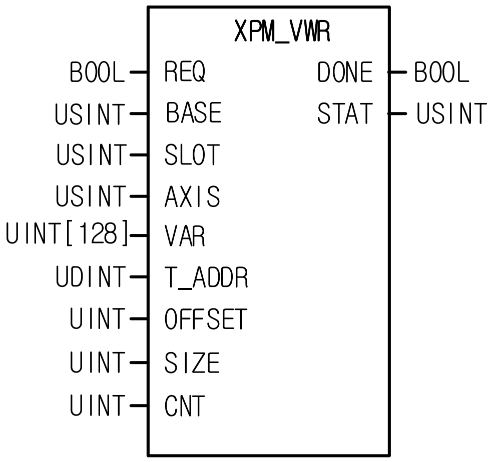
(7) In case Read Data size (SIZE x CNT) is 0 or higher than 128 word, error code “11” appears in STAT.

■ **Program example**

1. ST

INST_XPM_VRD(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), S_ADDR:=(*UDINT*), OFFSET:=(*UINT*), SIZE:=(*UINT*), CNT:=(*UINT*), DONE=>(*BOOL*), STAT=>(*UINT*), VAR=>(*ARRAY[0..127]_OF_UINT*))

XPM_VWR	Write Variable Data	
	Availability	XGI, XGR
	Flags	

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command XPM: 1 ~ 4 (1-axis ~4-axis) XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis) VAR : PLC device where Write Data is saved T_ADDR : Module internal memory head address where data is written 0 ~ 53329 OFFSET : Offset between Write data blocks 0 ~ 54217 * XGF-PNxB: 0 ~ 65535 SIZE : Size of block to write 1 ~ 128 CNT : No. of Write data block 1 ~ 128 **Output** DONE : Maintain 1 after first operation STAT : Output the error no. in operation

■ **Function**

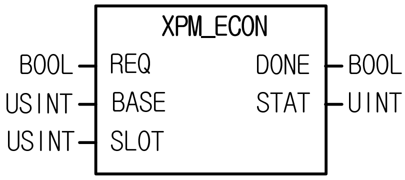
(1) Gives "Write parameter, operation data, CAM data directly" command to positioning module.
(2) You can write data you want by configuring module internal memory address of parameter, operation data, CAM data directly.
(3) It writes the WORD data in "VAR" to module internal memory. The data are saved from internal memory position set by "T_ADDR" and the number of data is the number set by "Size". In case the number of block "CNT" is larger than 2, multiple blocks are made. At this time, head address of next block is "Offset" apart from head address of current block.
(4) Max. data size (SIZE x CNT) you can write with one command is 128 word.
(5) "Write Variable Data" command can't execute in operation.
(6) You can set an axis to command in "AXIS" and the following value is available. If other value is set, it produces "Error6."
XPM: 1 ~ 4 (1-axis ~4-axis), XGF-PN8A/B: 1 ~ 8 (1-axis ~ 8-axis)
(7) In case Read Data size (SIZE x CNT) is 0 or higher than 128 WORD, error code "11" appears in STAT
(8) In case no. of block (CNT) is higher than 2, and block offset is smaller than block size, error code "11" appears in STAT because module internal memory block to write is overlapped each other.

■ **Program example**

1. ST

```
INST_XPM_VWR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*),
VAR:=(*ARRAY[0..127]_OF_UINT*), T_ADDR:=(*UDINT*), OFFSET:=(*UINT*), SIZE:=(*UINT*), CNT:=(*UINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_ECON	Connect Servo Communication	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module **Output** DONE : Maintain 1 after first operation STAT : Output the error no. in operation

■ **Function**

(1) Gives “EtherCAT Communication Connection” command to positioning module.
(2) Instruct the positioning module configured by BASE (base number of positioning module) and SLOT (slot number of positioning module) to connect communication with Servo
(3) If Servo driver is connected normally, the bit corresponding to the connected axis is set.

	Global variable	Contents
1-axis	_xxyy_A1_RDY	1-axis operation ready
2-axis	_xxyy_A2_RDY	2-axis operation ready
3-axis	_xxyy_A3_RDY	3-axis operation ready
4-axis	_xxyy_A4_RDY	4-axis operation ready
5-axis	_xxyy_A5_RDY	5-axis operation ready
6-axis	_xxyy_A6_RDY	6-axis operation ready
7-axis	_xxyy_A7_RDY	7-axis operation ready
8-axis	_xxyy_A8_RDY	8-axis operation ready

(For xxyy, “xx” means base number and “yy” means slot number where module is installed

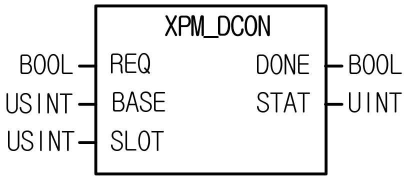
(4) This instruction is only for XGF-PN8A/B.

■ **Program example**

1. ST

INST_XPM_ECON(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_DCON	Disconnect Servo Communication	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module **Output** DONE : Maintain 1 after first operation STAT : Output the error no. in operation

■ **Function**

(1) Gives "EtherCAT Communication Disconnection" command to positioning module.

(2) Instruct the positioning module configured by BASE (base number of positioning module) and SLOT (slot number of positioning module) to disconnect communication with Servo

(3) If Servo driver is connected normally, the bit corresponding to the disconnected axis is cleared.

	Global variable	Contents
1-axis	_xxyy_A1_RDY	1-axis operation ready
2-axis	_xxyy_A2_RDY	2-axis operation ready
3-axis	_xxyy_A3_RDY	3-axis operation ready
4-axis	_xxyy_A4_RDY	4-axis operation ready
5-axis	_xxyy_A5_RDY	5-axis operation ready
6-axis	_xxyy_A6_RDY	6-axis operation ready
7-axis	_xxyy_A7_RDY	7-axis operation ready
8-axis	_xxyy_A8_RDY	8-axis operation ready

(For xxyy, "xx" means base number and "yy" means slot number where module is installed

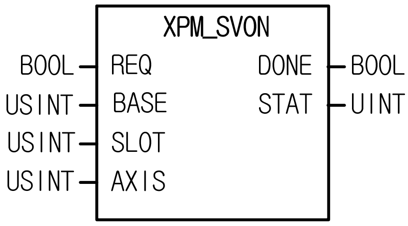
(4) This instruction is only for XGF-PN8A/B.

■ **Program example**

1. ST

```
INST_XPM_DCON(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_SVON	Servo On	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1~8: 1-axis ~ 8-axis **Output** DONE : Maintain 1 after first operation STAT : Output the error no. in operation

■ **Function**

(1) Give "Servo On" command to positioning module.

(2) Instruct the positioning module configured by BASE (base number of positioning module) and SLOT (slot number of positioning module) to disconnect communication with Servo

(3) In order to start a motor, Servo On signal should be on.

(4) You can set an axis to command in "AXIS" and the following value is available. If other value is set, it produces "Error6."
1 ~ 8 (1-axis ~ 8-axis)

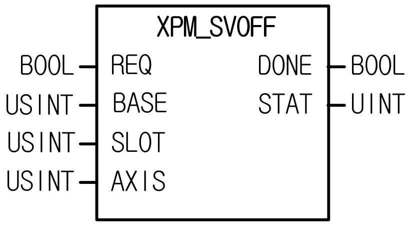
(5) This instruction is only for XGF-PN8A/B.

■ **Program example**

1. ST

```
INST_XPM_SVON(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_SVOFF	Servo Off	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
 	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1~8: 1-axis ~ 8-axis **Output** DONE : Maintain 1 after first operation STAT : Output the error no. in operation

■ **Function**

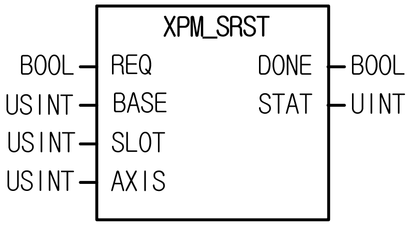
(1) Gives "Servo Off" command to positioning module.
(2) Instruct the positioning module configured by BASE (base number of positioning module) and SLOT (slot number of positioning module) to disconnect communication with Servo
(3) You can set an axis to command in "AXIS" and the following value is available. If other value is set, it produces "Error6."
1 ~ 8 (1-axis ~ 8-axis)
(4) This instruction is only for XGF-PN8A/B.

■ **Program example**

1. ST

INST_XPM_SVOFF(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_SRST	Servo Error Reset	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1~8: 1-axis ~ 8-axis **Output** DONE : Maintain 1 after first operation STAT : Output the error no. in operation

■ **Function**

(1) Gives “Servo Error Reset” command to positioning module.

(2) Instruct the positioning module configured by BASE (base number of positioning module) and SLOT (slot number of positioning module) to disconnect communication with Servo

(3) If you give a “Servo Error Reset” command without removing the reason of server drive alarm, servo driver alarm may not ne cleared. So remove the reason of servo driver alarm and then execute a “Servo Error Reset” command.

(4) You can set an axis to command in “AXIS” and the following value is available. If other value is set, it produces “Error6.”

1 ~ 8 (1-axis ~ 8-axis)

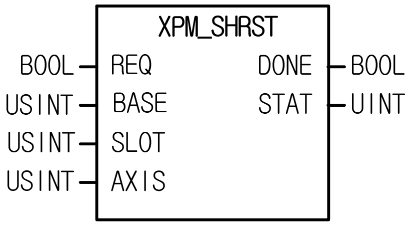
(5) This instruction is only for XGF-PN8A/B.

■ **Program example**

1. ST

```
INST_XPM_SRST(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_SHRST	Servo Error History Reset	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1~8: 1-axis ~ 8-axis **Output** DONE : Maintain 1 after first operation STAT : Output the error no. in operation

■ **Function**

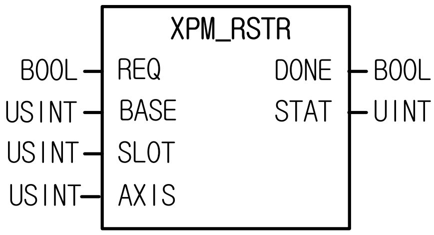
(1) Gives “Servo Error History Reset” command to positioning module.
(2) Instruct the positioning module configured by BASE (base number of positioning module) and SLOT (slot number of positioning module) to disconnect communication with Servo
(3) Instruct the servo corresponding to the selected axis among the servos connected to the module to reset alarm histories
(4) Servo drive can save up to 10 server alarm histories
(5)You can set an axis to command in “AXIS” and the following value is available. If other value is set, it produces “Error6.”
1 ~ 8 (1-axis ~ 8-axis)
(6) This instruction is only for XGF-PN8A/B.

■ **Program example**

1. ST

INST_XPM_SHRST(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))

XPM_RSTR	Restart	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1 ~ 8: aixs1 ~ axis8 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) Give “Restart” command to the axis of positioning module designated by BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(2) This command is used when restarting the axis which stops by EMG stop command. If this command is executed, the axis operates again with previous operating information.

(3) If you start the axis with commands other than “Restart” after it stops with DEC. stop, “Restart” will not be executed

(4) Set an axis to command from 1 ~ 8. If you set wrongly, “Error6” arises.

1 ~ 8: axis1 ~ axis8

(5) For detailed information on “Restart”, refer to “9.2.20. Restart”.

■ **Program example**

1. ST

```
INST_XPM_RSTR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*));
```

<table>
<tr><td rowspan="3">XPM_POE</td><td colspan="2">Setting Position Output Enable/Disable</td></tr>
<tr><td>Availability</td><td>XGI, XGR</td></tr>
<tr><td>Flags</td><td>-</td></tr>
</table>

<table>
<tr><th>Function Block</th><th>Description</th></tr>
<tr><td><pre>
              XPM_POE
BOOL - REQ   DONE - BOOL
USINT - BASE   STAT - UINT
USINT - SLOT
USINT - AXIS
USINT - DATA_NUM
UINT - TIME
BOOL - ENABLE
</pre></td><td>Input
REQ : Request for execution of function block
BASE : Set the base no. with module
SLOT : Set the slot no. with module
AXIS : Axis to command
1 ~ 4: aixs1 ~ axis4
DATA_NUM : The number of setting position output (0~50)
TIME : Keeping time of setting position output (0~65,535ms)
ENABLE : Setting position output enable/disable
0: Disable , 1: Enable
Output
DONE : Maintain 1 after first operating
STAT : Output the error no. in operation</td></tr>
</table>

■ **Function**

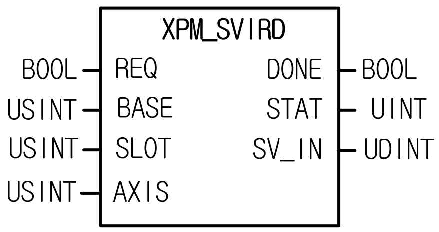
(1) Give “Setting Position Output Enable/Disable” command to the axis of positioning module designated by BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).
(2) When Setting position output enable and current position come to setting position output ,the position module outputs signal to deviation count clear pin or setting position output pin.
(3) Setting the number of data on DATA_NUM. The number of data can set between 0 to 50, If other value is set, it produces “Error11” and if the number of data on DATA_NUM is zero, the function block operates disable.
(4) During setting time on Time of F/B, Setting Position Output signal is on.
(5) If disables the F/B, Current output signal changes off immediately.
(6) Set an axis to command from 1 ~ 4. If you set wrongly, ‘‘Error6’’ arises.
1 ~ 4: axis1 ~ axis4
(7) This instruction is only for XPM Module.

■ **Program example**

1. ST

```
INST_XPM_POE(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DATA_NUM:=(*USINT*), TIME:=(*UINT*),
ENABLE:=(*BOOL*), DONE=>(*BOOL*), STAT=>(*UINT*));
```

XPM_SVIRD	Servo External Input Information Read	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1 ~ 8: aixs1 ~ axis8 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation SV_IN: Servo input signal information

■ **Function**

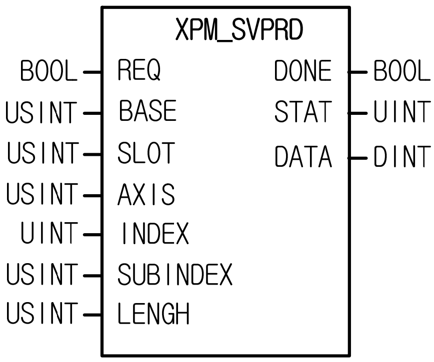
(1) Give "Servo External Input Information Read" command to the axis of positioning module designated with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).
(2) This is command reading input signal state of the servo driver corresponding to the selected axis among servos connected to the module
(3) Input signal state is outputted at SV_IN.
(4) Set an axis to command from 1 ~ 8. If you set wrongly, "Error6" arises.
1 ~ 8 : axis1 ~ axis8

■ **Program example**

1. ST

```
INST_XPM_SVIRD(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*),
SV_IN=>(*UDINT*));
```

XPM_SVPRD	Servo Parameter Read	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1 ~ 8: aixs1 ~ axis8 INDEX: SUBINDEX: LENGTH: **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation DATA: Read servo parameter data

■ **Function**

(1) Only for XGF-PN8B, this is the command that reads parameters (CoE object) of the servo driver connected to positioning module.

(2) Give “Servo Parameter Read” command to the axis of positioning module designated with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(3) Save in DATA to read value of LENGTH size at the servo parameter object designated with INDEX, SUBINDEX, at the axis designated with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(4) Set an axis to command from 1 ~ 8. If you set wrongly, “Error6” arises.

1 ~ 8 : axis1 ~ axis8

(5) INDEX can be set as follows. If you set wrongly, “Error11” arises at STATE.

Set value	Description
0x1000 ~ 0x1FFF	Communication Profile Area
0x2000 ~ 0x5FFF	Manufacturer Specific Profile Area
0x6000 ~ 0x9FFF	Standardized Device Profile Area

(6) SUBINDEX can be set as follows. If you set wrongly, “Error11” arises at STATE.

Set value	Description
0x0 ~ 0xFF	Object Subindex of servo parameter

(7) LENGTH can be set as follows. If you set wrongly, “Error11” arises at STATE.

Set value	Description
1 ~ 4	Object Byte Length of servo parameter

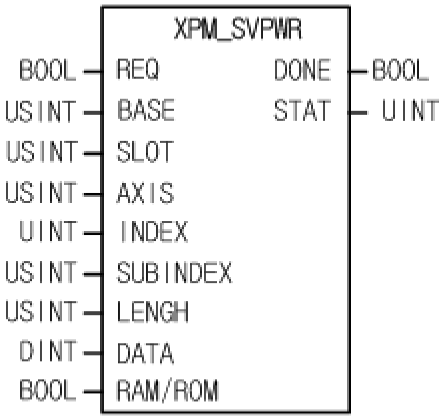
(8) This instruction is only for XGF-PN8B.

■ **Program example**

1. ST

```
INST_XPM_SVPRD(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), INDEX:=(*UINT*),
SUBINDEX:=(*USINT*), LENGH:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*), DATA=>(*DINT*));
```

XPM_SVPWR	Servo Parameter Write	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1 ~ 8: aixs1 ~ axis8 INDEX : Servo parameter object Index SUBINDEX : Servo paramter object subindex LENGTH : Servo parameter object size DATA: Servo parameter value RAM/ROM : how to save parameter 0: save at RAM, 1: save at ROM **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) This is the function block only for XGF-PN8B and that changes parameters (CoE object) of the servo driver connected to positioning module
(2) Give “Servo Parameter Write” command to the axis of positioning module designated with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).
(3) If you want to save at the internal ROM of the servo driver with “Servo parameter write” command, set up 1 at RAM/ROM and execute the command, or set up 0 at RAM/ROM and execute the command and later save them at servo driver EEPROM with XPM_SVSAVE command.
(4) Save DATA of LENGTH size at the servo parameter object designated with INDEX, SUBINDEX, at the axis designated with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).
(5) Set an axis to command from 1 ~ 8. If you set wrongly, “Error6” arises.
1 ~ 8 : axis1 ~ axis8
(6) You can set INDEX as follows. If you set wrongly, “Error11” arises

Setting value	Description
0x2000 ~ 0x5FFF	Manufacturer Specific Profile Area
0x6000 ~ 0x9FFF	Standardized Device Profile Area

(7) You can set SUBINDEX as follows. If you set wrongly, “Error11” arises

Setting value	Description
0x0~0xFF	Servo parameter Object Subindex

(8) You can set SUBINDEX as follows. If you set wrongly, "Error11" arises

Setting value	Description
1~4	Servo parameter Object Byte Length

(9) You can set SUBINDEX as follows.

Setting value	Teaching method
0	RAM teaching
1	ROM teaching

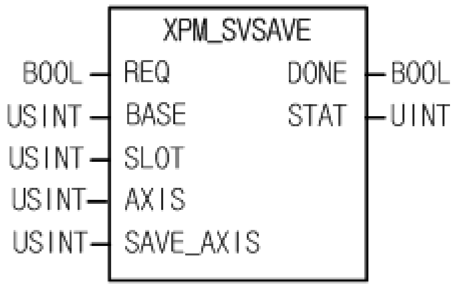
(10) This instruction is only for XGF-PN8B.

■ **Program example**

1. ST

```
INST_XPM_SVPWR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), INDEX:=(*UINT*),
SUBINDEX:=(*USINT*), LENGTH:=(*USINT*), DATA:=(*DINT*), RAM_ROM:=(*BOOL*), DONE=>(*BOOL*), STAT=>(*UINT*));
```

XPM_SVSAVE	Servo Parameter Save	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1 ~ 8: aixs1 ~ axis8 SAVE_AXIS: Set the axis to save by setting each bit (bit 0~7: 1-axis~8-axis) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

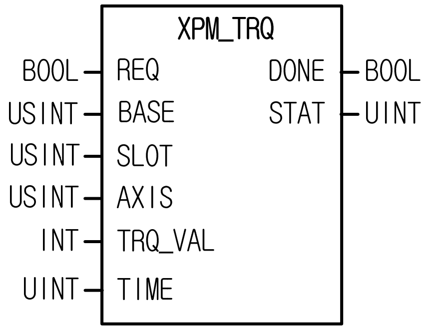
(1) This is the function block only for XGF-PN8B and that saves parameters of the servo driver connected to positioning module at the EEPROM of the servo driver.
(2) Give "Servo Parameter Save" command to the axis of positioning module designated with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).
(3) Set up the axis to give a command at AXIS and you can set as follows. If you set wrongly, "Error6" arises. Command axis is different with the axis for saving servo parameter. If you want to save servo parameter of the command axis, set the corresponding bit at SAVE_AXIS.
1 ~ 8: 1-axis ~ 8-axis
(4) Set up the servo driver axis at SAVE_AXIS. If you set wrongly, "Erro11" arises
Bit 0 ~ 7 : 1-axis ~ 8-axis
(5) This instruction is only for XGF-PN8B.

■ **Program example**

1. ST

```
INST_XPM_SVSAVE(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), SAVE_AXIS:=(*USINT*),
DONE=>(*BOOL*), STAT=>(*UINT*));
```

XPM_TRQ	Torque Control	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1 ~ 8: aixs1 ~ axis8 TRQ_VAL: Torque value (unit: %, -32768 ~ 32767) TIME: Torque gradient (unit: ㎳, 0 ~ 65535 ㎳) **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

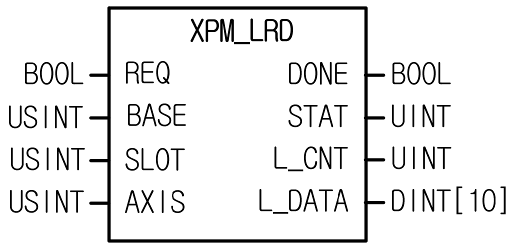
(1) Give "Torque Control" command to the axis of positioning module designated by BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).
(2) Torque control executes if torque value and torque gradient are set and a command is issued.
(3) Set torque value (%) to TRQ_VAL. Torque values work in % rated torque. (1 = 1% of rated torque)
For example, set 200 if the user wants to control torque in 200% of torque.
※ The allowable range of torque value may vary according to the connected servo drive. In general, target torque value is limited to the maximum torque setting.
(4) Set time to take in reaching the target torque to TIME. If a command is executed, torque increases in this gradient until it reaches the set torque value.
(5) Any command cannot be executed, the relevant axis is being operated for functions other than torque control.
(6) Set an axis to command from 1 ~ 8. If you set wrongly, "Error6" arises.
1 ~ 8: axis1 ~ axis8
(7) For detailed information on "Torque Control", refer to "9.2.21. Torque Control".
(8) This instruction is only for XGF-PN8B.

■ **Program example**

1. ST

```
INST_XPM_TRQ(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), TRQ_VAL:=(*INT*), TIME:=(*UINT*),
DONE=>(*BOOL*), STAT=>(*UINT*));
```

XPM_LRD	Servo External Input Information Read	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1 ~ 8: axis1 ~ axis8 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation L_CNT: Number of latch position data L_DATA: Latch position data 1 ~ 10

■ **Function**

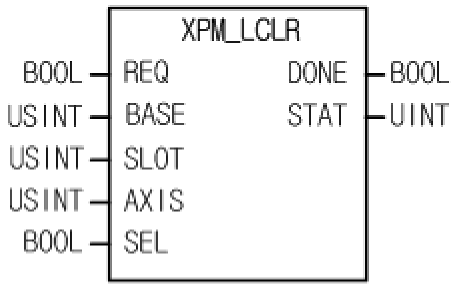
(1) This command is used to read data count and latch position data saved and latched by the positioning module's external latch command.
(2) Save the position data count read and latched the latch data of the axis designated as the positioning module's AXIS(Command axis) designated as BASE(Base number of the positioning module) and SLOT(Slot number of the positioning module) to L_CNT and save the latch position data to L_DATA.
(3) Set an axis to which Command is issued to Axis and one among 1 through 8 can be set. If any other value except the setting value is set, "Error 6" arises.
(4) This instruction is only for XGF-PN8A/B.

■ **Program example**

1. ST

```
INST_XPM_LRD(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*),
L_CNT=>(*UINT*), L_DATA=>(*ARRAY[0..9]_OF_UDINT*));
```

XPM_LCLR	Latch Reset	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1 ~ 8: aixs1 ~ axis8 SEL: Latch reset item selection **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

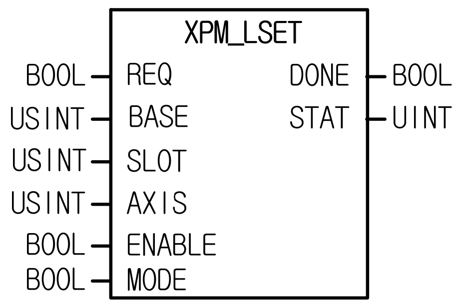
(1) This command is used to initialize the data count and latch position data saved and latched on the positioning module or the state when latch is completed.
(2) Give “Latch Reset” command to the positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).
(3) The following items are reset according to the Reset Latch items designated to SEL.
0: Reset the state when latch is completed
1: Reset latch position data and the state when latch is completed
(Values high than “1” are processed equally with “1”)
(4) If latch position data are read through the “Read Latch Position Data (XPM_LRD)” command after 1 is set to SEL and the “Reset Latch” command is executed, all of data become 0.
(5) Set an axis to command from 1 ~ 8. If you set wrongly, “Error6” arises.
1 ~ 8 : axis1 ~ axis8
(6) This instruction is only for XGF-PN8A/B.

■ **Program example**

1. ST

```
INST_XPM_LCLR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), SEL:=(*BOOL*), DONE=>(*BOOL*),
 STAT=>(*UINT*));
```

XPM_LSET	Servo External Input Information Read	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1 ~ 8: aixs1 ~ axis8 ENABLE: Latch enable/disable MODE: Latch mode **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

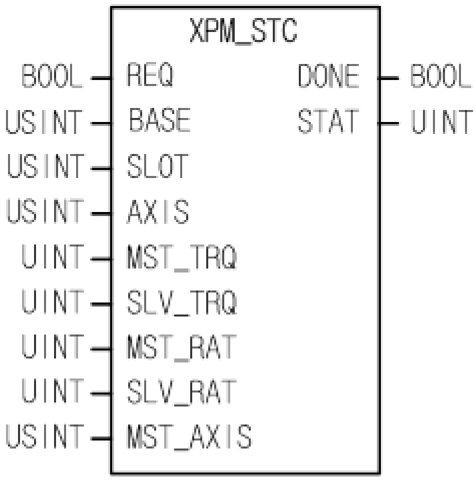
(1) This command is used to initialize the data count and latch position data saved and latched on the positioning module or the state when latch is completed.
(2) Give “Latch Set” command to the positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).
(3) Actions according to the Enable/Disable Latch item designated to ENABLE are as following.
0: latch prohibition 1: latch permission
(Values high than “1” are processed equally with “1”)
(4) Actions according to the latch mode item designated to MODE are as following.
0: Single trigger (The current position latch is available only the touch probe 1 signal inputted at first after latch is enabled)
1: Continuous trigger (The current position latch is available at every touch probe 1 signal after latch is enabled)
(Values high than “1” are processed equally with “1”)
(5) Set an axis to command from 1 ~ 8. If you set wrongly, “Error6” arises.
1 ~ 8 : axis1 ~ axis8
(6) “Latch Set” command is applied to only XGF-PN8B.
(7) This instruction is only for XGF-PN8B.

■ **Program example**

1. ST

```
INST_XPM_LSET(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), ENABLE:=(*BOOL*), MODE:=(*BOOL*),
DONE=>(*BOOL*), STAT=>(*UINT*));
```

XPM_STC	Torque Synchronization	
	Availability	XGI, XGR
	Flags	-

Function Block	Description
	**Input** REQ : Request for execution of function block BASE : Set the base no. with module SLOT : Set the slot no. with module AXIS : Axis to command 1 ~ 8: aixs1 ~ axis8 MST_TRQ : Torque rate of main axis 0 ~ 65535 SLV_TRQ : Torque rate of sub axis 0 ~ 65535 MST_RAT : Speed rate of main axis 0 ~ 65535 SLV_RAT : Speed rate of sub axis 0 ~ 65535 MST_AXIS : Torque synchronization main axis 1 ~ 8: aixs1 ~ axis8 **Output** DONE : Maintain 1 after first operating STAT : Output the error no. in operation

■ **Function**

(1) This command is used to order torque synchronization to axis of servo drive that is connected to positioning module.

(2) Give "Torque synchronization" command to the axis of positioning module with BASE (Base no. of Positioning module) and SLOT (Slot no. of Positioning module).

(3) The axis to performing a command operates torque synchronization with main axis set as MST_AXIS.

(4) The axis to performing a command operates torque synchronization with torque rate set as MST_TRQ, SLV_TRQ and speed rate set as MST_RAT, SLV_RAT.

Torque of sub axis = (SLV_TRQ/MST_TRQ) * torque of main axis

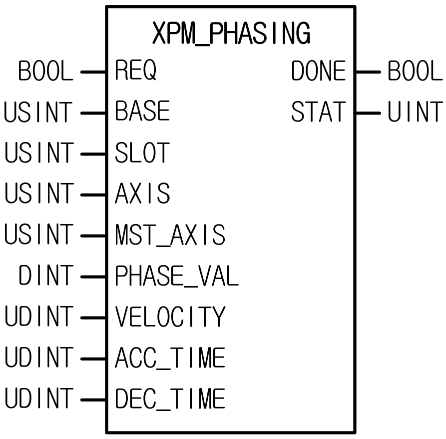
Torque synchronization speed of sub axis = (SLV_RAT/MST_RAT) * speed of main axis

(5) Set an axis to AXIS from 1 ~ 8. If you set wrongly, "Error 6" arises.

1 ~ 8 : axis1 ~ axis8

(6) Set an main axis of torque synchronization to MST_AXIS from 1 ~ 8. If you set wrongly, "Error 11" arises.

1 ~ 8 : axis1 ~ axis8

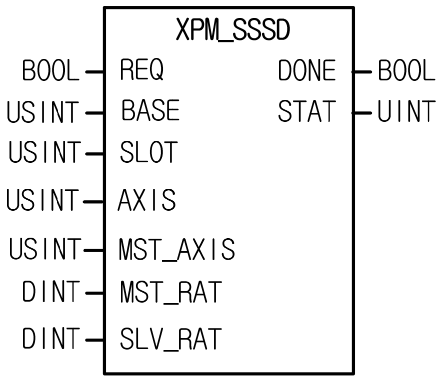
XPM_PHASING	Applied model	Occurrence flag
Phase correction control	**XGI, XGR**	**-**
Function block	Explanation	
	**input** REQ: Function block execution request BASE: Set the number of the base on which the module is mounted SLOT: Set the number of the slot where the module is mounted AXIS: Assign axis to command XGF-PN4B: 1 to 4 (1 to 4 axes) XGF-PN8A / XGF-PN8B: 1 to 8 (1 to 8 axes) MST_AXIS: Phase correction main axis setting XGF-PN4B: 1 to 4 (1 to 4 axes) XGF-PN8A / XGF-PN8B: 1 to 8 (1 to 8 axes) 9: Encoders 1 and 10: Encoder 2 PHASE_VAL: Phase correction value VELOCITY: Phase correction speed (relative speed to main shaft speed) ACC_TIME: Acceleration time (0 ~ 2,147,483,647 ms) DEC_TIME: Deceleration time (0 ~ 2,147,483,647 ms) **Print** DONE: Maintain 1 after initial operation STAT: Output error number generated during function block execution	

■ **Features**

1. It is a function block that executes phase correction with respect to the position of the main axis referenced by AXIS of the positioning module and enables synchronous operation to the position of main axis whose subordinate axis is corrected.
2. ACC_TIME, DEC_TIME by the amount of phase correction set in PHAS_VAL for the main axis set in MST_AXIS on the axis specified by AXIS of the positioning module specified by BASE (base number of positioning module) and SLOT (slot number of positioning module) Perform phase correction with.
3. AXIS sets the axis on which to issue the command. You can set the following values. If a value other than the set value is set, "Error 6" occurs.
   1) XBF-PN08B
      1 to 8: 1 to 8 axes
   2) XBF-PN04B
      1 to 4: 1 to 4 axes
4. MST_AXIS sets the main axis of the phase compensation command and the following values can be set. If a value other than the set value is set, "Error 11" occurs.

1) XBF-PN08B
   1 to 8: 1 to 8 axes, 9: Encoders 1 and 10: Encoder 2
2) XBF-PN04B
   1 to 4: 1 to 4 axes, 9: Encoders 1 and 10: Encoder 2

XPM_SSSD	**Applied model**	**Occurrence flag**
32-bit speed sync	**XGI, XGR**	**-**

Function block	Explanation
 	**input** REQ: Function block execution request BASE: Set the number of the base on which the module is mounted SLOT: Set the number of the slot where the module is mounted AXIS: Assign axis to command XGF-PN4B: 1 to 4 (1 to 4 axes) XGF-PN8A / XGF-PN8B: 1 to 8 (1 to 8 axes) MST_AXIS: Speed synchronous spindle setting XGF-PN4B: 1 to 4 (1 to 4 axes) XGF-PN8A / XGF-PN8B: 1 to 8 (1 to 8 axes) 9: Encoders 1 and 10: Encoder 2 MST_RAT: Speed ratio of main shaft -2,147,483,648-2,147,483,647 SLV_RAT: Speed ratio of subordinate axis -2,147,483,648-2,147,483,647 **Print** DONE: Maintain 1 after initial operation

■ **Features**

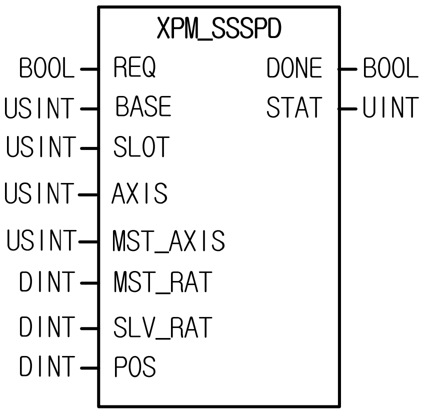
One. It outputs a speed synchronous command to the axis specified by AXIS of the positioning module specified by BASE (base number of positioning module) and SLOT (slot number of positioning module).

2. It is used to control the ratio of the operation speed between two axes. You can set the spindle and ordinate ratios to a 32-bit integer range.
3. There is no rule for size between the spindle speed ratio and the subordinate axis speed ratio. That is, if the speed ratio of the main axis is higher than the speed ratio of the vertical axis, the main axis moves faster than the vertical axis. If the speed ratio of the sub axis is larger than the speed ratio of the main axis, the sub axis moves faster than the main axis.
4. AXIS sets the axis on which to issue the command. You can set the following values. If a value other than the set value is set, "Error 6" occurs.

   XGF-PN4B: 1 to 4 (1 to 4 axes), XGF-PN8A / XGF-PN8B: 1 to 8 (1 to 8 axes)
5. MST_AXIS sets the main axis of speed synchronization and the following values can be set. If a value other than the set value is set, "Error 11" occurs.

   XGF-PN4B: 1 to 4 (1 to 4 axes), XGF-PN8A / XGF-PN8B: 1 to 8 (1 to 8 axes), 9: Encoders 1 and 10: Encoder 2

6. The driving direction of the vertical axis is $\text{Speed synchronization ratio}\left(\frac{\text{main axis ratio}}{\text{Longitudinal axis ratio}}\right)$ If positive, it operates in the direction of main spindle. If negative, it operates in the opposite direction of main spindle.

<table>
<tr><th>XPM_SSSPD</th><th>Applied model</th><th>Occurrence flag</th></tr>
<tr><td>Positioning Speed Synchronization</td><td>XGI, XGR</td><td>-</td></tr>
<tr><td>Function block</td><td colspan="2">Explanation</td></tr>
<tr><td>
</td><td colspan="2">input
REQ: Function block execution request
BASE: Set the number of the base on which the module is mounted
SLOT: Set the number of the slot where the module is mounted
AXIS: Assign axis to command
XGF-PN4B: 1 to 4 (1 to 4 axes)
XGF-PN8A / XGF-PN8B: 1 to 8 (1 to 8 axes)
MST_AXIS: Speed synchronous spindle setting
XGF-PN4B: 1 to 4 (1 to 4 axes)
XGF-PN8A / XGF-PN8B: 1 to 8 (1 to 8 axes)
9: Encoders 1 and 10: Encoder 2
MST_RAT: Speed ratio of main shaft
-2,147,483,648-2,147,483,647
SLV_RAT: Speed ratio of subordinate axis
-2,147,483,648-2,147,483,647
POS: Goal location
-2,147,483,648-2,147,483,647
Print
DONE: Maintain 1 after initial operation
STAT: Error number occurred during execution of function block</td></tr>
</table>

■ **Features**

One. The positioning speed synchronous command is issued to the axis specified by AXIS of the positioning module specified by BASE (base number of positioning module) and SLOT (slot number of positioning module).

2. It is used to control the ratio of the operation speed between two axes. You can set the spindle and ordinate ratios to a 32-bit integer range. After XPM_SSSPD is executed, when the position where the subordinate axis moved is the position designated by POS, it ends the speed synchronization and stops.

3. There is no rule for size between the spindle speed ratio and the subordinate axis speed ratio. That is, if the speed ratio of the main axis is higher than the speed ratio of the vertical axis, the main axis moves faster than the vertical axis. If the speed ratio of the sub axis is larger than the speed ratio of the main axis, the sub axis moves faster than the main axis.

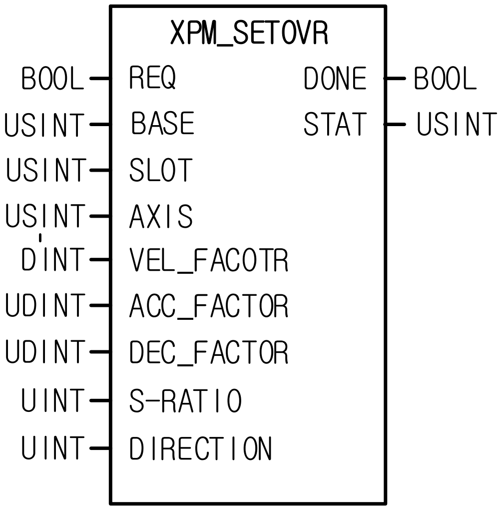
4. AXIS sets the axis on which to issue the command. You can set the following values. If a value other than the set value is set, "Error 6" occurs.

XGF-PN4B: 1 to 4 (1 to 4 axes), XGF-PN8A / XGF-PN8B: 1 to 8 (1 to 8 axes)

5. MST_AXIS sets the main axis of speed synchronization and the following values can be set. If a value other than the set value is set, "Error 11" occurs.
   XGF-PN4B: 1 to 4 (1 to 4 axes), XGF-PN8A / XGF-PN8B: 1 to 8 (1 to 8 axes), 9: Encoders 1 and 10: Encoder 2

6. The driving direction of the vertical axis is $\text{Speed synchronization ratio}\left(\frac{\text{main axis ratio}}{\text{Longitudinal axis ratio}}\right)$ If positive, it operates in the direction of main spindle. If negative, it operates in the opposite direction of main spindle.

XPM_SETOVR	Applied model	Occurrence flag
Speed / acceleration / deceleration override	**XGI, XGR**	**-**

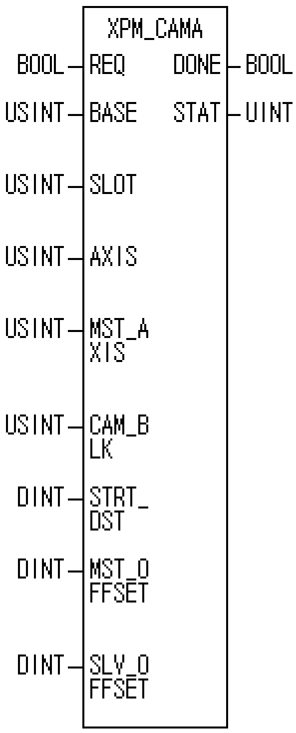
Function block form	Contents
	input REQ: Function block execution request BASE: Set the number of the base on which the module is mounted SLOT: Set the number of the slot where the module is mounted AXIS: Assign axis to command XGF-PN4B: 1 to 4 (1 to 4 axes) XGF-PN8B: 1 to 8 (1 to 8 axes) VEL_FACTOR: Speed Override Ratio (Or command speed) ACC_FACTOR: Acceleration Override Ratio (Or command acceleration time) DEC_FACTOR: Deceleration Override Ratio (Or command deceleration time) S_RATIO: unused (S-curve ratio (0 = trapezoid, 1 to 100: S-curve ratio)) DIRECTION: Driving direction (1 ~ 3: 1-forward direction, 2-reverse direction, 3-current direction) Print DONE: Maintain 1 after initial operation STAT: Output error number generated during function block execution

(1) The speed / acceleration / deceleration override command is given to the axis specified by AXIS of the positioning module specified by BASE (base number of positioning module) and SLOT (slot number of positioning module).
(2) It is used to change the operation speed, acceleration, deceleration, and direction while command axis is in operation.
(3) VEL_FACTOR, ACC_FACTOR and DEC_FACTOR can be set to "%" or "speed value (unit / hour)" according to the value set in "Speed override" of the common parameter.

(4) If the unit of speed override value is%, the setting range is -65,535 ~ 65,535, which means -655.35 ~ 655.35%.
(5) If the unit of speed override value is the speed value, the setting range is - speed limit value ~ speed limit value. In this case, speed limit value is the value set in "speed limit value" item of basic parameter. The units of the speed override value follow the axis unit.

(6) When the unit of acceleration override and deceleration override value is%, the setting range is 0 ~ 65,535, which means 0% ~ 655.35%.
(7) When the acceleration override and deceleration override value units are speed values, the setting range is 0 to 4,294,967,295.
(8) Operation direction value can be input 1 ~ 3, 1 means forward, 2 means reverse, and 3 means current direction.
(9) AXIS sets the axis to be commanded and the following values can be set. If a value other than the set value is set, "Error 6" occurs.

XGF-PN4B: 1 to 4 (1 to 4 axes), XGF-PN8B: 1 to 8 (1 to 8 axes)

<table>
<tr><th>XPM_CAMA</th><th>Applied model</th><th>Occurrence flag</th></tr>
<tr><td>Absolute position cam drive</td><td>XGI, XGR</td><td>-</td></tr>
<tr><th>Function block form</th><th colspan="2">Contents</th></tr>
<tr><td>
</td><td colspan="2">input
REQ: Function block execution request
BASE: Set the number of the base on which the module is mounted
SLOT: Set the number of the slot where the module is mounted
AXIS: Assign axis to command
XGF-PN4B: 1 to 4 (1 to 4 axes)
XGF-PN8B: 1 to 8 (1 to 8 axes)
MST_AXIS: Main axis setting
XGF-PN4B: 1 to 4 (1 to 4 axes)
XGF-PN8B: 1 to 8 (1 to 8 axes)
9: Encoder 1
CAM_BLK: Cam block setting
1 to 9: 1 block 1 to 9 blocks
STRT_DST: Cam operation start movement setting
-2147483648 ~ 2147483647
MST_OFFSET: Spindle offset position movement amount setting
-2147483648 ~ 2147483647
SLV_OFFSET: Subordinate axis offset position movement setting
-2147483648 ~ 2147483647
Print
DONE: Maintain 1 after initial operation
STAT: Output error number generated during function block execution</td></tr>
</table>

(1) Absolute position cam operation command is issued to the axis specified by AXIS of the positioning module specified by BASE (base number of positioning module) and SLOT (slot number of positioning module).

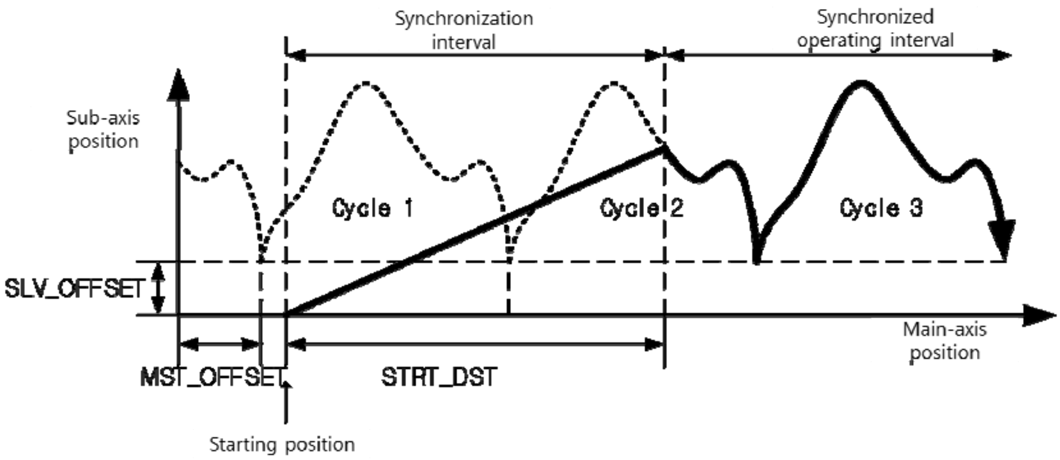
(2) Cam is driven by using the cam main axis, cam data block, cam operation start position, spindle offset, and vertical axis

offset.

(3) Execute absolute position cam operation command and start to move to the synchronous position until the axis set as main axis starts to move by the distance set in STRT_DST.

The synchronized position can be moved according to the setting of the MST_OFFSET and SLV_OFFSET values to the position on the subordinate axis according to the cam data value set in the cam block (CAM_BLK) when the main axis is in STRT_DST. When the main axis reaches the distance set in STRT_DST, the motor starts to move to the subordinate axis position corresponding to the main axis position according to the data value of the cam data block set in the cam block (CAM_BLK).

(4) AXIS sets the axis to be commanded and the following values can be set. If a value other than the set value is set,



"Error 6" occurs.

XGF-PN4B: 1 to 4 (1 to 4 axes), XGF-PN8B: 1 to 8 (1 to 8 axes)

(5) In MST_AXIS, main axis of cam operation is set and the following values can be set. If a value other than the set value is set, "Error 11" occurs.

XGF-PN4B: 1 to 4 (1 to 4 axes), XGF-PN8B: 1 to 8 (1 to 8 axes), 9: Encoder 1

(6) CAM_BLK sets the cam block number to be executed and the following values can be set. If a value other than the set value is set, "Error 11" occurs.

1 to 9: Block 1 to Block 9

(7) Cam data can be created in the positioning package, and up to 8 blocks (block 1 to block 8) can be set.

(8) In order to use the user cam (CAM) operation, the cam block number must be set to 9.

(9) Refer to "9.4.4 User Cam (CAM) Operation" for details of user cam (CAM) operation.

Chapter 12. Expanded Functions

This chapter describes each expanded function.. It is used for a specific processing (ex. FOR ~ NEXT, CALL, etc.) of a part of program during user program run.

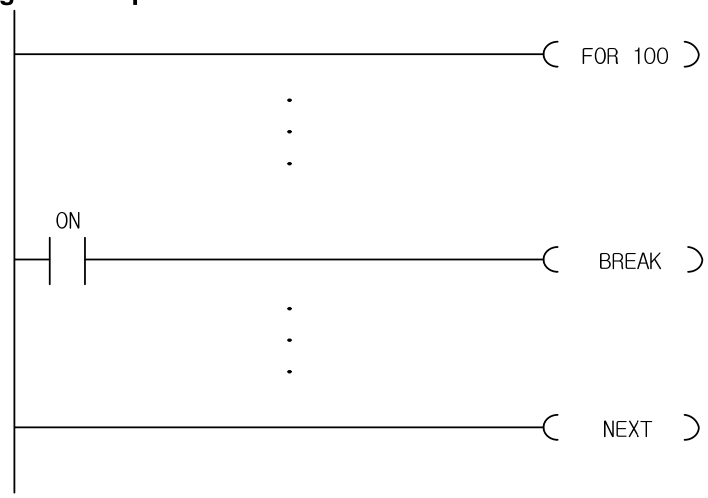
FOR/NEXT/BREAK	LOOP command	
	Availability	XGI, XGR, XEC
	Flags	_ERR, _LER

Function	Description
(FOR | N)	Repeat a block of FOR ~ NEXT n times
(NEXT)	
(BREAK)	Escape a block of FOR ~ NEXT

■ **Function**

(1) PLC repeats FOR ~ NEXT command n times and then processes the next step of NEXT command.
(2) n is available 1 ~ 65,535.
(3) FOR ~ NEXT command is able to use 16 NESTINGs.
(4) REAK command is the instruction to escape FOR ~ NEXT loop.
(5) Keep the range of WDT value to avoid delaying the scan time.

■ **Program Example**



(1) It operates FOR ~ NEXT loop 100 times repeatedly.
(2) To escape the loop during a repetition, turn the switch on and run the BREAK command.

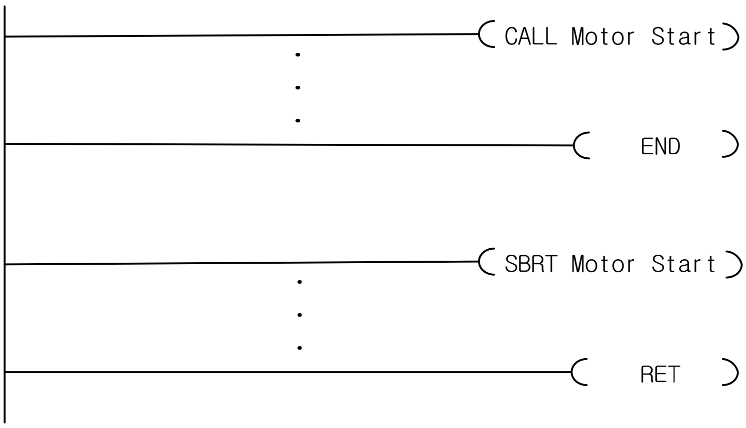
CALL/SBRT/RET	Command of function call	
	Availability	XGI, XGR, XEC
	Flags	

Function	Description
(CALL NAME)	Call a SBRT routine
(SBRT NAME)	Assign a routine to be called by the CALL function
(RET)	RETURN

■ **Function**

(1) With an input condition and the CALL n command, it operates a program among the SBRT n ~ RET command.
(2) Nested CALL n command is usable, and the program among SBRT n ~ RET must be placed after END command.
(3) A program which is in SBRT can call another SBRT. In this case, END command is impossible to use in the SBRT.
(4) A program can escape the FOR ~ NEXT loop with a BREAK command.

■ **Program Example**



(1) It calls a SBRT (Motor Start) if the program operates CALL command.
(2) SBRT command must be placed after the END command.
(3) When SBRT (Motor Start) is called, a program is run in the SBRT until RET command. It goes to the position again where CALL command is called.

JMP	**JUMP command**	
	Availability	XGI, XGR, XEC
	Flags	

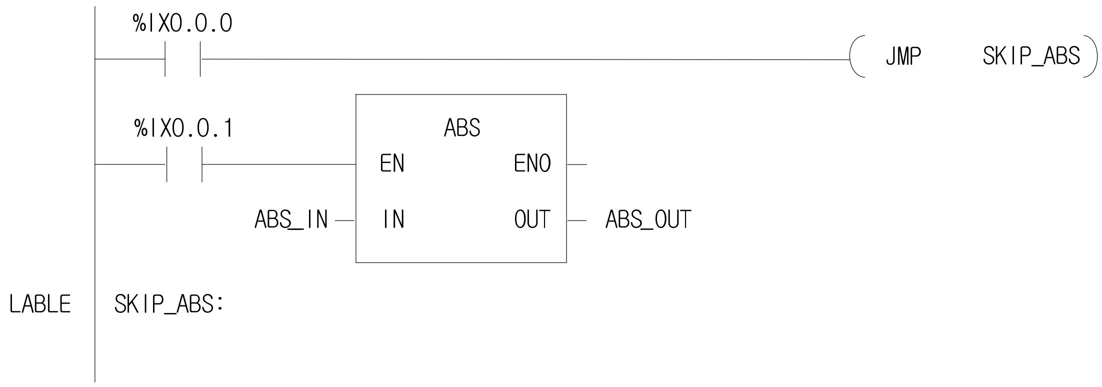
Function	Description
——(JMP LABLE)	Jump to a place of LABLE

■ **Function**

(1) If a switch of JMP (LABLE) command is on, it jumps to the next of the assigned LABLE. All the commands between JMP and LABLE are not processed.
(2) LABLE must not be duplicated, but JMP can be repeated.
(3) It is recommended that the program which must not be run in a state of emergency is placed between JMP and LABLE.

■ **Program Example**

(1) When %IX0.0.0 is on, it does not operate ABS function.



INIT_DONE	Command to terminate an initial task	
	Availability	XGI, XGR, XEC
	Flags	

Function	Description
——(INIT_DONE)	Terminate an initial task

■ **Function**

(1) It terminates an initial task.
(2) You have to terminate an initial task program using this command when you program an initial task program. Otherwise, you neither terminate the initial task program nor enter a scan program.

■ **Program Example**

(1) When %IX0.0.0 is on, it terminates an initial task.

END	END command	
	Availability	XGI, XGR, XEC
	Flags	

Function	Description
——(END)	Terminate a program

■ **Function**

(1) It indicates the end of a program.

(2) After the processing of the END command, the program goes to the beginning of itself and process again.

Chapter 13. Process Control Library

This chapter describes the process control library relating to process control, data process, arithmetic instruction, data measurement and data creation.

13.1 Process Control Library

1) STAT
Some process control library functions and function blocks have STAT, which is used to notify of any error of instruction. If STAT has any other value, other than 0, it means that the instruction has an error; the content of STAT code is as follows.

STAT	Name	Operation on occurrence	Description
1	T_s error	Scan cycle operation	In case it may not work as previously set because T_s setting is earlier than the current scan time, it operates with the earliest time as possible and displays it.
2	X_min, X_max inversion	Operation stop Output reset	Input is designed to be X; it displays if X_min is larger than X_max while it is limited to max./min.
4	Y_min, Y_max inversion	Operation stop Output reset	Output is designed to be X; it displays if Y_min is larger than Y_max while it is limited to max./min.
8	Other setting error	Operation stop Output reset	It means any other erroneous state of setting except the above statements

If two and more are detected in the above, the sum of two STATs is output. That is, if 2 should be the output to STAT as X_min and X_max are inversed while 4 should be output to STAT as Y_min and Y_max are inversed, the sum of 2 and 4, 6 should be output.
Errors except T_s error in which STAT is 1 stop function or function block, outputs 0 and make, if any, DONE and ENO off.

2) T_s
T_s existing in some instructions represents operation cycle of instruction and if setting T_s, the instruction operates every T_s time. As being structured to execute an operation if passing T_s time after comparing the previous operation time and the present time as it approaches to the instruction, it has temporal error E (T_s) and the error is not accumulated ordinarily because it reflects the error in the next operation cycle.

$$0 \leq E(T\_s) < T_{scan}$$

In the case, T_s error is accumulated, and the instruction executes operation every time it scans to solve accumulated error and it outputs 1 to STAT value. Therefore, if setting T_s as 0, it processes the instruction every time it scans.

3) Setting same max. limit and min. limit
Process library keeps several min./max. limits of X or Y. In general, if max./min. values are limited, a bit displaying on the bottom of output that such limits are valid exists (i.e.: X_max_AL) and especially, if max. limit and min. limit are set alike, both alarms are turned on, which is the way displaying that it is limited both to max. and min. limits.

4) Abnormal input
It may not work properly if an instruction to have real numbers had abnormal input such as 1.#inf00000 E+000, -1.#inf00000 E+000 or 1.#QNAN0000 E+000.

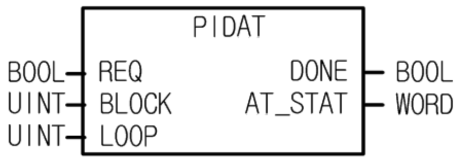
Notes

Blinking STAT 1 (T_S error)

Since every scan of PLC may have different data volume, the execution speed may not be same per scan. In case, it may work with STAT 1 indicated or STAT 1 blinks unless T_s setting does not have tolerance properly. For instance, if a user sets T_s as 3ms and its scan cycle fluctuates between 2 ~ 4ms, it may work properly if its scan cycle is 2ms or 3ms but the instruction may not work normally if it reaches to 4ms, so it should indicate 1 in STAT and the scan operates with 4ms. If scan is shortened to 2ms or 3ms, STAT 1 is turned off and blinks.

13.2 Process Control Function and Function Block

PIDAT	PID Auto tuning	
	Availability	XEC
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request BLOCK : Block number (0) LOOP : Loop number (0~15) **Output** DONE : On if done without error PID_STAT : PID state alarm

■ Functions

(1) It executes PID operation of the related block and loop.
(2) Totally 16 PID loops are available independently because BLOCK is fixed as 0 and LOOP can take input as 0~15
(3) Output AT_STAT is hexadecimal and each PID loop shows the state as presented in <Table 13.1>.

<Table 13.1>

Class	Display	Flag	Description
STATE	16#0001	PID_STAT	A loop is being operated.
	16#0080	AT_DONE	AT (Auto-tuning) ends.
	16#0100	MV_MIN_MAX_ERR	Max. MV is smaller than Min. MV
	16#0300	PWM_PERIOD_ERR	Output period of PWM output is smaller than 100 (10ms).
	16#0400	SV_RANGE_ERR	In case of forward operation, Set value at the start of auto-tuning is smaller than present value. In case of reverse operation, Set value at the start of auto-tuning is larger than present value
	16#0500	PWM_ADDRESS_ERR	The value other than %QX0.0.0~0.0.31 is set as PWM output
	16#0A00	TUNE_DIR_CHG	Operation direction is changed while auto-tuning
	16#0B00	AT_PERIOD_ERR;	Operation period of auto-tuning is smaller than 100(10ms)
	16#0E00	LOOP_EXCEED	Auto-tuning LOOP number is larger than 15

(4) Each state may be presented simultaneously.
.

PIDRUN	PID Operator	
	Availability	XGI, XGR, XEC
	Flags	-

Function block	Description
<pre> PIDRUN BOOL- REQ DONE - BOOL UINT- BLOCK PID_STAT - WORD UINT- LOOP</pre>	**Input** REQ : Function block execution request BLOCK : Block number (0~7) LOOP : Loop number (0~31) **Output** DONE : On if done without error PID_STAT : PID state alarm

■ **Functions**

(1) It executes PID operation of the related block and loop.
(2) Totally 256 PID loops are available independently because block may be 0 ~ 7 (In case of XEC), and loop of each block may be 0 ~ 31. (In case of XEC 0~15)
(3) Output PID_STAT is hexadecimal and each PID loop shows the state as presented in the following table.

<Table 13.2>
In case of XGI, XGR

Class	Display	Flag	Description
ALARM	16#0001	T_s ERR	It may not execute every T_s because T_s setting is too small.
	16#0002	K_p ERR	Note that K_p is 0.
	16#0004	dPV_AL	PV is limited by dPV_max setting.
	16#0008	dMV_AL	MV is limited by dMV_max setting.
	16#0010	MVmax_AL	MV is limited by MV_max setting.
	16#0020	MVmin_AL	MV is limited by MV_min setting.
	16#0040	AT_fail	AT (Auto-tuning) is abnormally ended.
	16#0080	Unused	Unused
STATE	16#0100	PID_STAT	A loop is being operated.
	16#0200	AT_STAT	AT (Auto-tuning) in progress
	16#0400	AT_DONE	AT (Auto-tuning) ends.
	16#0800	EX_RUN	Started by external run signal.
	16#1000	MAN_OUT	Manual output in progress
	16#2000	CAS_STAT	CAS (Cascade) in progress
	16#4000	CAS_MST	CAS (Cascade) operates as master.
	16#8000	AW_STAT	AW1(Anti wind-up) or AW2 is operating.

In case of XEC

Class	Display	Flag	Description
ALARM	16#0001	PV_MIN_MAX_ALM	Present value exceeds the range
	16#0002	PID_SCANTIME_ALM	Operation period is too small
	16#0003	PID_dPV_WARN	Delta present value of this PID period exceeds Delta PV limit
	16#0004	PID_dMV_WARN	Delta manipulated value of this PID period exceeds Delta MV limit
	16#0005	PID_MV_MAX_WARN	MV of this PID period exceeds Max. MV
	16#0006	PID_MV_MIN_WARN	MV of this PID period exceeds Min. MV
ERROR	16#0100	MV_MIN_MAX_ERR	Max. MV is smaller than Min. MV
	16#0200	PV_MIN_MAX_ERR	Max. PV is smaller than Min. MV
	16#0300	PWM_PERIOD_ERR	PWM output period is smaller than 100 (10ms)
	16#0400	SV_RANGE_ERR	In case of forward operation, Set value at the start of auto-tuning is smaller than present value. In case of reverse operation, Set value at the start of auto-tuning is larger than present value
	16#0500	PWM_ADDRESS_ERR	The value other than %QX0.0.0~0.0.31 is set as PWM output
	16#0600	P_GAIN_SET_ERR	Proportional Gain is smaller than 0
	16#0700	I_TIME_SET_ERR	Integral Time is smaller than 0
	16#0800	D_TIME_SET_ERR	Derivative Time is smaller than 0
	16#0900	CONTROL_MODE_ERR	Control mode is other than P, PI and PID.
	16#0B00	PID_PERIOD_ERR;	PID operation period is smaller than 100(10ms)
	16#0C00	HBD_WRONG_DIR	In case of combined operation, direction parameter of forward operation loop is set as reverse or direction parameter of reverse operation loop is set as forward
	16#0D00	HBD_SV_NOT_MATCH	In case of combined operation, Set values of two loops are different.
	16#0E00	LOOP_EXCEED	PID LOOP number is larger than 15

(4) Each state may be presented simultaneously.

PIDCAS	Cascade PID Operator	
	Availability	XGI, XGR, XEC
	Flags	-

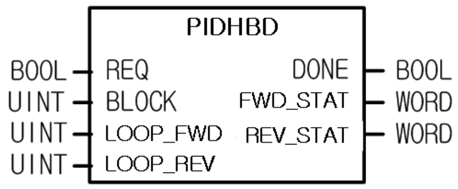
<table>
<tr><th>Function block</th><th>Description</th></tr>
<tr><td><pre>
            PIDCAS
BOOL ─ REQ   DONE ─ BOOL
UINT ─ BLOCK   MST_STAT ─ WORD
UINT ─ LOOP_MST SLV_STAT ─ WORD
UINT ─ LOOP_SLV
</pre></td><td>Input REQ :Function block execution request
BLOCK :Block number
LOOP_MST :Master loop number
LOOP_SLV :Slave loop number

Output
DONE : On if done without error
MST_STAT : Master loop state alarm
SLV_STAT : Slave loop state alarm</td></tr>
</table>

■ **Functions**

(1) Executes Cascade PID operation with a combination of two loops for a block.
(2) Block may be 0 ~ 7 (In case of XEC, 0), and master loop and slave loop should be between 0 ~ 31 (in case of XEC, 0~15) in a same block and differently.
(3) MST_STAT and SLV_STAT for output are hexadecimal and represent the states of master and slave respectively as presented in the above table.
(4) Each state may be presented simultaneously.

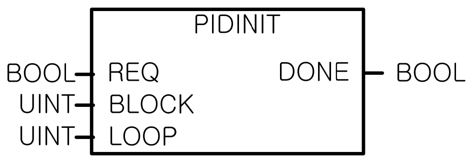
PIDHBD	Forward-reverse combined output PID operator	
	Availability	XEC
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request BLOCK : Block number LOOP_FWD : Forward direction loop number LOOP_REV : reverse direction loop number **Output** DONE : On if done without error FWD_STAT : Forward direction loop state alarm REV_STAT : Reverse direction loop state alarm

■ Function

(1) Combines two related loops of related block and executes Forward/reverse combined output PID operation.
(2) Block is 0 and master loop and slave loop should use different number of 0~ 15 in the same block.
(3) Output FWD_STAT, REV_STAT are hexadecimal and each represents the status like <Table 13.2> of forward direction and reverse direction.
(4) Each state may be presented simultaneously

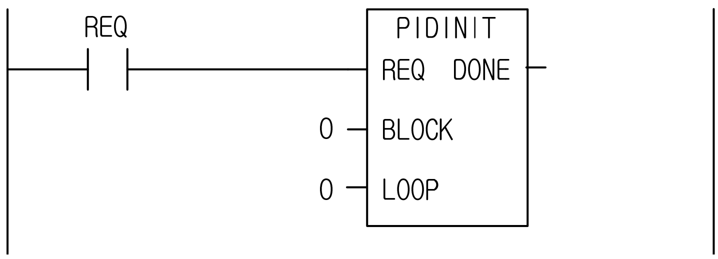
PIDINIT	PID Initialize	
	Availability	XGI, XGR
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request BLOCK : Block number LOOP : Loop number **Output** DONE : On if done without error

■ **Function**

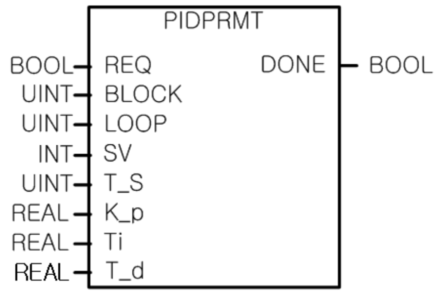
(1) Initializes all loop PID settings of a block to 0.

■ **Program Example**



Once input contact REQ is set, it initializes every setting of PID block 0 and loop 0 to 0.

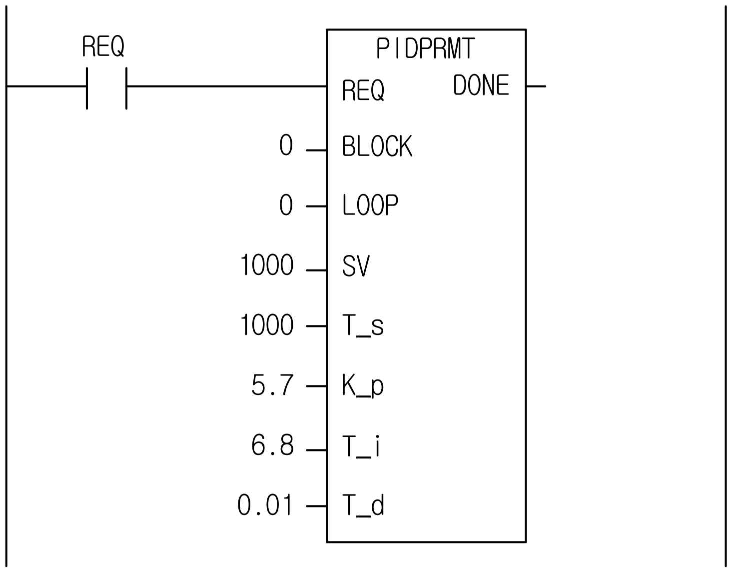
PIDPRMT	PID Parameter Change	
	Availability	XGI, XGR
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request BLOCK : Block number LOOP : Loop number SV : Set value T_s : Operation cycle K_p : Proportional constant Ti : Integral constant T_d : Differential constant **Output** DONE : On if done without error

■ **Functions**

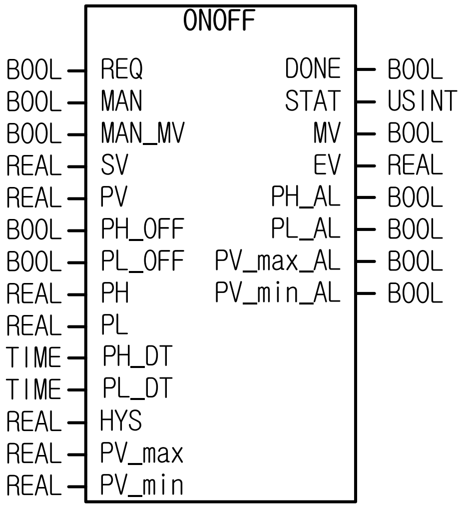
(1) It changes PID settings of loop and block to input value.
(2) The setting items to be changed are SV, T_s, K_p, T_i and T_d as expressed in input.
(3) Since applying PIDPPMT instruction may change coefficient according to the conditions of a PID loop, pattern control may be executed in accordance with system response.

■ **Program Example**



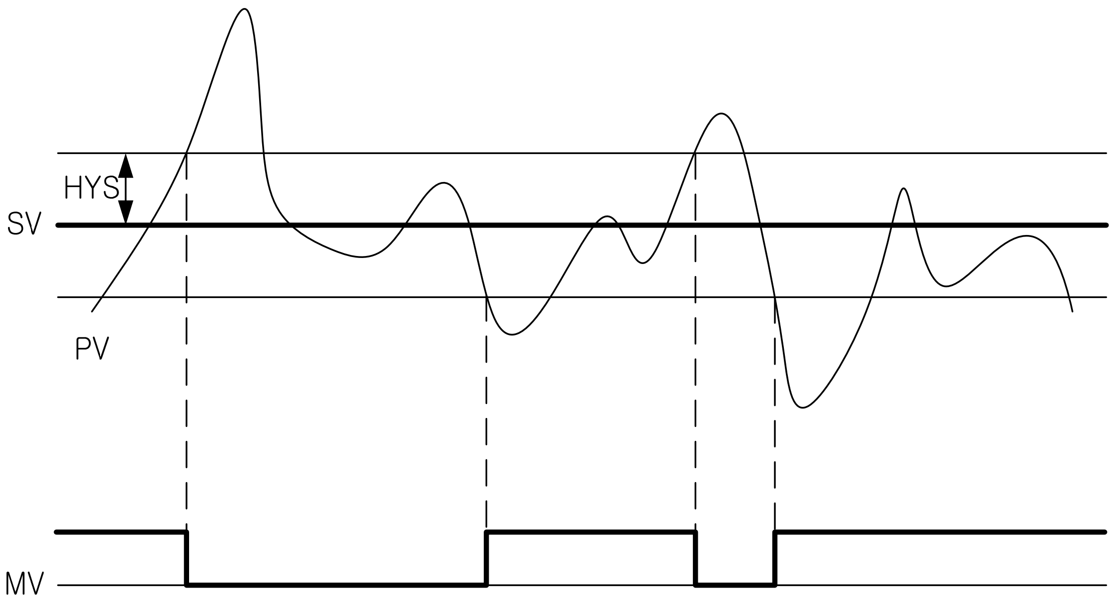
Main setting of PID block 0 and loop 0 is changed with the input values as seen in the above figure.

ONOFF	ON / OFF Control	
	Availability	XGI, XGR, XEC(U)
	Flags	-

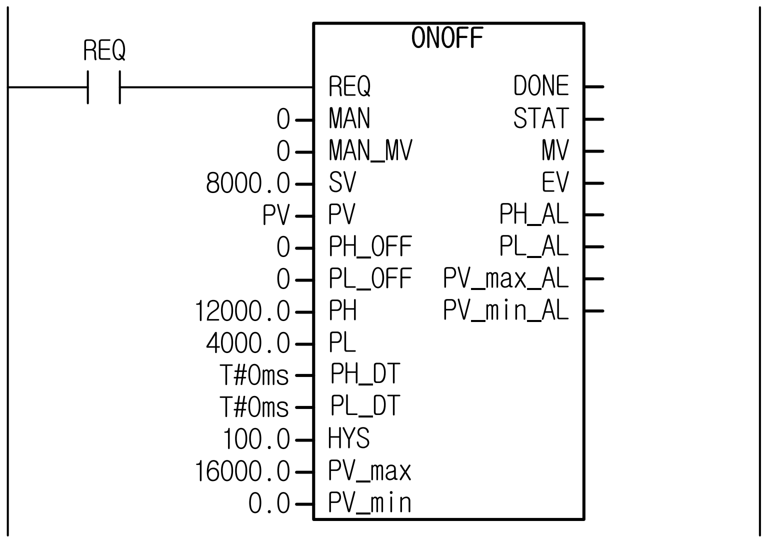
Function block	Description
	**Input** REQ : Function block execution request MAN : Manual mode conversion bit MAN_MV : Manual mode conversion value SV : Set value PV : Present value PH_OFF : PV High section cancel bit PL_OFF : PV Low section cancel bit PH : PV High section set value PL : PV Low section set value PH_DT : PV High section set delay time PL_DT : PV Low section set delay time HYS : Hysterisis radius setting PV_max : PV max. limit PV_min : PV min. limit **Output** DONE : On if done without error STAT : State alarm MV : Output value EV : Error value PH_AL : PV High alarm PL_AL : PV Low alarm PV_max_AL : PV max. high alarm PV_min_AL : PV min. low alarm

■ **Functions**

(1) ON/OFF control creating Booltype output MV
(2) If PV is received from AD, it is necessary to convert the data type to REAL prior to use.
(3) Once setting MAN, it is converted to manual mode and MAN_MV value is output to MV, irrespective of the operation results.
(4) In case of (SV – HYS) > PV, MV = On
(5) In case of (SV + HYS) < PV, MV = Off
(6) In case of (SV – HYS) ≤ PV ≤ (SV + HYS), MV = MV(previous)
(7) It represents 'Error value EV = SV – PV'.
(8) If setting each up/down section of PV to PH/PL, it displays the corresponding PH_AL/PL_AL alarm when it is beyond the sections.
(9) However, if PH_OFF/PL_OFF bit is on, it does not execute each PH_AL/PL_AL operation.
(10) In PH_DT/PL_DT, the output delay time of PH_AL/PL_AL may be set.
(11) PV input may be limited by setting the max./min. value of each PV in PV_max/PV_min. When it reaches the limits, PV_max_AL/PV_min_AL alarms are on.
(12) If EV is out of the real number data range, the output displays with '1.#inf00000 E+000' or '-1.#inf00000 E+000'but the output except EV is normally operates.

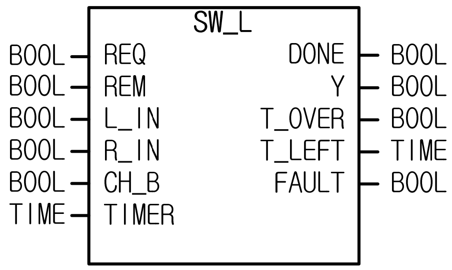


■ **Program Example**



- If PV is over 8100 (8000+100), MV is off while if PV is less than 7900 (8000-100), MV is on.
- If PV is not less than 16000, it is regarded as 16000 and PV_max_AL is set; if it is not more than 0, it is regarded as 0 and PV_min_AL is set.
- If PV is not less than 12000, PH_AL is set; in case of not more than 4000, PL_AL is set.

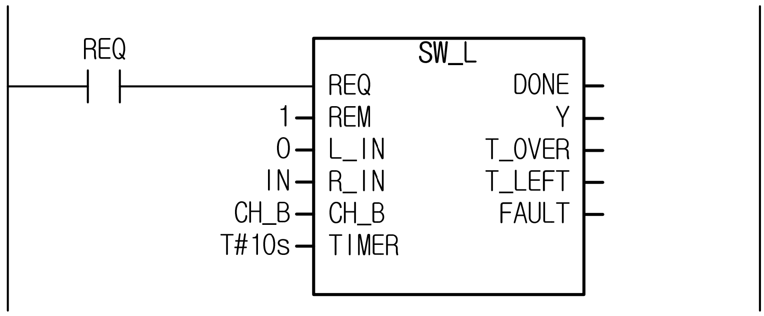
SW_L	1 Input latch	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request REM : Remote input setting L_IN : Local input R_IN : Remote input CH_B : Check back input TIMER : Check back queue time **Output** DONE : On if done without error Y : Output value T_OVER : Time over alarm T_LEFT : Left time display FAULT : Check back failure alarm

■ **Functions**

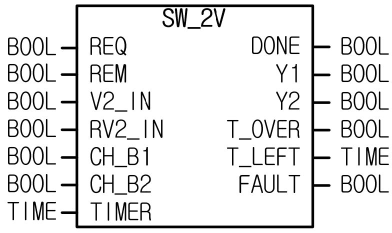
(1) If using pump control, it may not work due to a fault/trouble or it may cause an accident due to any other reasons, as it outputs continuous operation instruction unless it is checked whether a pump actually works with a check back signal after receiving pump operation instruction. Against it, it is designed that it determines a trouble and outputs fault without any operation instruction unless CHECK_BACK signal (RUN signal of a pump) is input after an operation instruction Y is output.
(2) If REM is off, it receives L_IN as its input; in case of on, it receives R_IN as its input.
(3) Once the first input is on, output Y is on and it waits for CH_B (check back) signal for a time set in TIMER.
(4) At the moment, T_LEFT shows the left time and T_OVER is on after the left time passes.
(5) If CH_B and input are on after a time set in TIMER, Y continues to be on; if CH_B is off even for a while, it regards it as system fault, outputs off to Y and turns FAULT on. Then it outputs off to Y even though CH_B is on again.
(6) If input is off, it operates from the first step.

■ **Program Example**



If IN is on with REQ set, Y is on and timer works for 10s, during while T_LEFT shows the left time. In 10 s, T_OVER is on; if CH_B is on, Y is maintained as on while if CH_B is off, Y is off and Fault is on.

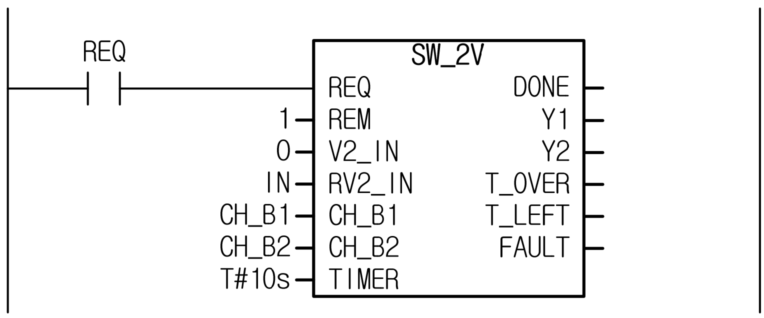
SW_2V	2-Way Valve Control	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request REM : Remote input setting V2_IN : Select local valve2/1 RV2_IN : Select remote valve2/1 CH_B1 : Input valve 1 check back CH_B2 : Input valve 2 check back TIMER : Check back queue time **Output** DONE : On if done without error Y1 : Output 1 Y2 : Output 2 T_OVER : Time over T_LEFT : Left time display FAULT : Check back failure alarm

■ **Functions**

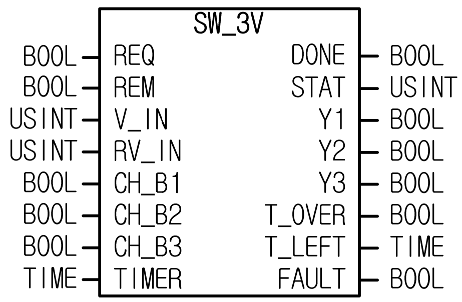
(1) In case of 2-way valve, the only selected side should be open and the other side should be closed. In addition, if check back signal is inputted, a valve may work properly unless it generates any output. If check back signal is not input in a check back input delay time after open instruction, fault is output.
(2) If REM is off, it receives V2_IN as its input ;if REM is on, it receives RV2_IN as its input.
(3) If input is changed from/to off -> on, output Y2 is on and it waits for CH_B2 signal for a time set in timer.
(4) If input is reversely changed from/to on -> off, output Y1 is on and it waits for CH_B1 signal for a time set in timer.
(5) At the moment, T_LEFT shows the left time and T_OVER is on once the queue time passes.
(6) if a time set in timer, the output is off; if CH_B is on, fault is off. If CH_B is off, fault is on.
(7) It works from the first with input changed, and the output may be secured as long as timer setting is set more than twice of scan cycle.

■ **Program Example**



If IN is on with REQ set, Y2 is on and the timer works for 10s, during which T_LEFT shows the left time. In 10s, T_OVER is on, and the output, Y1 and Y2 are off. if CH_B2 is on, fault is off; if CH_B2 is off, the fault is on.

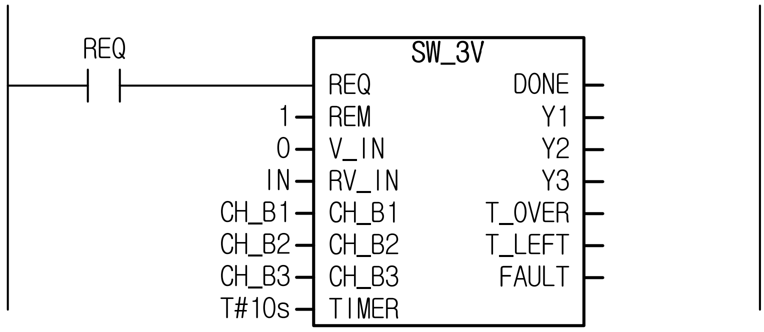
SW_3V	3-Way Valve Control	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request REM : Remote input setting V_IN : Local input selection (1~3) RV_IN : Remote input selection (1~3) CH_B1 : Input valve1 check back CH_B2 : Input valve2 check back CH_B2 : Input valve3 check back TIMER : Check back queue time **Output** DONE : On if done without error Y1 : Output 1 Y2 : Output 2 Y3 : Output 3 T_OVER : Time over T_LEFT : Left time display FAULT : Check back failure alarm

■ **Functions**

(1) In case of 3-way valve, the only selected side should be open and the other side should be closed. In addition, if check back signal is input, a valve may work properly unless it generates any output. If check back signal is not input in a check back input delay time after open instruction, fault is output.
(2) If REM is off, it receives V_IN as its input ;if REM is on, it receives RV_IN as its input.
(3) If input is changed from/to Vm -> Vn, output Yn is on and it waits for CH_Bn signal for a time set in timer.
(4) T_LEFT shows the left time and T_OVER is on once the queue time passes.
(5) If a time set in timer, the output is off; if CH_Bn is on, fault is off. If CH_Bn is off, fault is on.
(6) It works from the first with input changed, and the output may be secured as long as timer setting is set more than twice of scan cycle.
(7) Input should have a value between 1 ~ 3, and if it is not in the range, it outputs 8 to STAT.

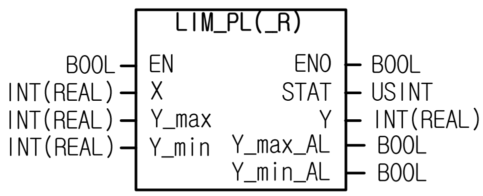
■ **Program Example**



If IN is changed to 4 with REQ set, Y3 is on and timer works for 10s.
During the time, T_LEFT shows the left time.
In 10s, T_OVER is on and the output, Y1, Y2 and Y3 are off.
If CH_B3 is on, fault is off; if CH_B3 is off, fault is on.

13.3 Data Process Function, Function Block

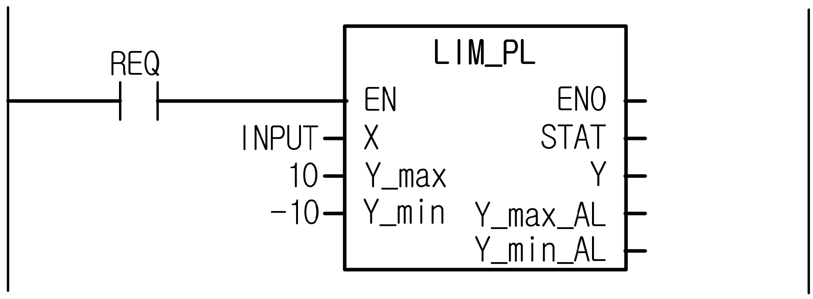
LIM_PL(_R)	Max./Min. value limit	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function	Description
	**Input** EN : Function execution request X : Input Y_max : Max. output limit Y_min : Min. output limit **Output** ENO : On if done without error STAT : State alarm Y : Output Y_max_AL : Over max. output alarm Y_min_AL : Less min. output alarm

■ **Functions**

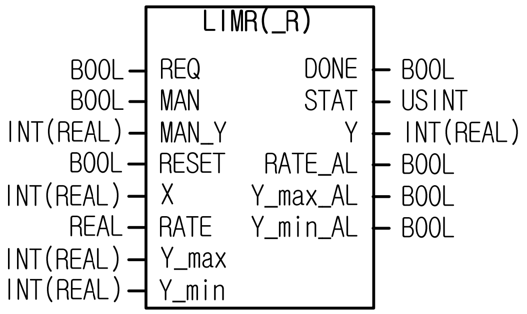
(1) It generates output Y by limiting input X within the max./min. values.
(2) A value between Y_max and Y_min passes without restriction.
(3) If max. limit is not less than Y_max, Y_max_AL is on and it outputs Y_max to Y.
(4) If min. limit is not more than Y_min, Y_min_AL is on and it outputs Y_min to Y.
(5) If Y_max is not more than Y_min, STAT indicates 4 and it outputs 0.

■ **Program Example**



(1) If INPUT is 20 : it outputs 10 (Y_max) to Y and Y_max_AL is on.
(2) If INPUT is 3 : it outputs 3 to Y without restriction.
(3) If INPUT is -12 : it outputs -10 (Y_min) to Y and Y_min_AL is on.

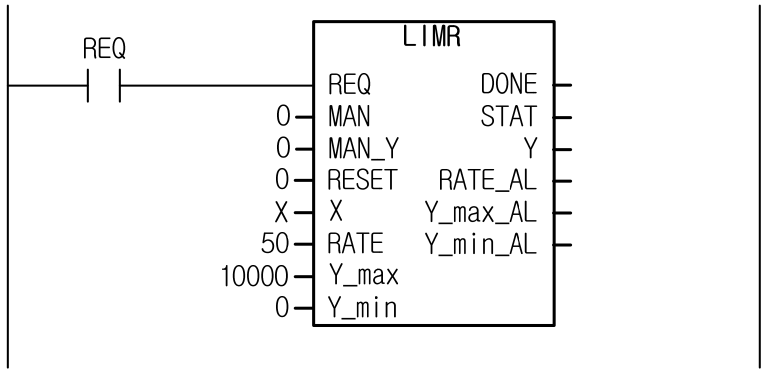
LIMR(_R)	Max./Min. value, max. variance limit	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
 	**Input** REQ : Function block execution request MAN : Manual mode setting MAN_Y : Manual output RESET : Block operation reset X : Input RATE : Max. variance rate limit Y_max : Max. output limit Y_min : Min. output limit **Output** DONE : On if done without error STAT : State alarm Y : Output value RATE_AL : Max. variance rate limit state alarm Y_max_AL : Over max. output alarm Y_min_AL : Less min. output alarm

■ **Functions**

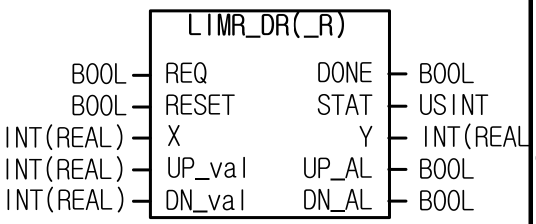
(1) It limits the max. variance rate of input X and outputs by limiting the max./min. value.
(2) The function block saves the internal state even though REQ is off and it resumes the previous operation if REQ is on again.
(3) Variance limit equation : $Y_{old} - \frac{RATE(Y_{max} - Y_{min})}{100} \le Y \le Y_{old} + \frac{RATE(Y_{max} - Y_{min})}{100}$
(4) If variation is limited, it indicates RATE_AL; if max./min. values are limited, it indicates Y_max_AL or Y_min_AL.
(5) If MAN is on, it outputs the value of MAN_Y to Y; if MAN is off again, the variance is limited from the state.
(6) If RESET is on, it initializes the output Y to 0.
(7) It may work at a desirable cycle if using the volume conversion detection contact of clock (i.e. _T1s) or other volume conversion detection contact (that is, P contact) to REQ.

■ **Program Example**



(1) X is changed from/to 0 → 3000 : the max. variance is allowed up to $\frac{RATE(Y_{max} - Y_{min})}{100}$ = 5000, so it passes the variance limit and max./min. value limits and outputs Y = 3000.
(2) X is changed from/to 0 → 10000 : the max. variance is allowed up to 5000, so it is restricted to the variance limit for 2 scans. Then, it increases by 5000, outputs Y = 10000 and Y_max_AL is on.
(3) X is changed from/to 0 → 30000 : the max. variance is allowed up to 5000, so it is restricted to the variance limit for 6 scans. Then, it increases by 5000, outputs Y = 10000 due to max. value limit and Y_max_AL is on.

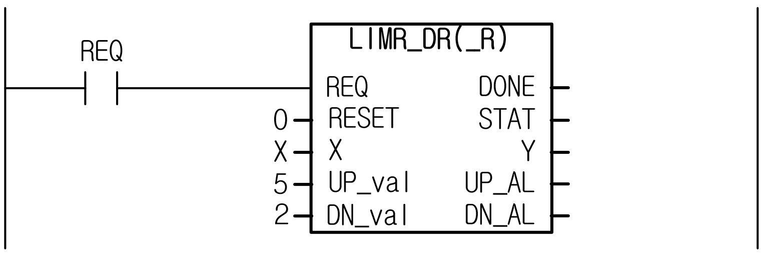
LIMR_DR(_R)	Directional max. variance limit	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request RESET : Block operation reset X : Input UP_val : Up limit DN_val : Down limit **Output** DONE : On if done without error STAT : State alarm Y : Output value UP_AL : Up limit alarm DN_AL : Down limit alarm

■ **Functions**

(1) It outputs by limiting the max. up/down variation of input X, respectively.
(2) The function block saves the internal state even though REQ is off and it resumes the previous operation if REQ is on again.
(3) For the variation of X, Y may be increased or decreased as much as UP_val or DN_val.
(4) In case the Up/Dn limits are applied, it displays with UP_AL or DN_AL bit.
(5) In case of RESET, the input X is directly reflected to Output Y.
(6) If UP_val or DN_val is negative, it outputs 8 to STAT.
(7) It may work at a desirable cycle if using the volume conversion detection contact of clock (i.e. _T1s) or other volume conversion detection contact (i.e. P contact) to REQ.

■ **Program Example**



(1) X is changed from/to 0 → 3000 : since the max. up variation is 5, Y increases by 5 for 600 scans, during which UP_AL is on ; if it outputs Y = 3000, UP_AL is off.
(2) X is changed from/to 1000 → 0 : since the max. down variation is 2, Y decreases by 2 for 500 scans, during which DN_AL is on; if it outputs Y = 0, DN_AL is off.

<table>
<tr><td rowspan="3">RATIO(_R)</td><td colspan="2">Ratio converter</td></tr>
<tr><td>Availability</td><td>XGI, XGR, XEC(U)</td></tr>
<tr><td>Flags</td><td>-</td></tr>
</table>

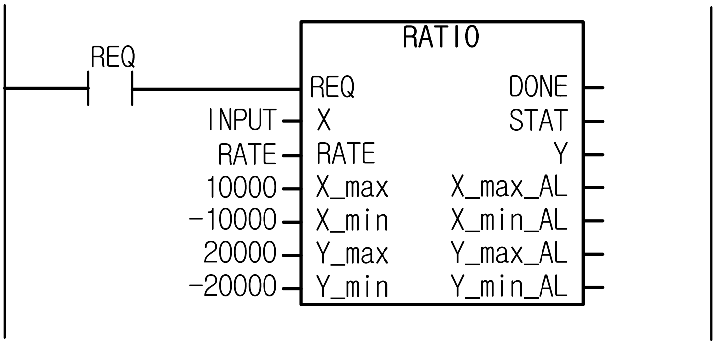
<table>
<tr><th>Function block</th><th>Description</th></tr>
<tr><td>
<pre>
                 RATIO(_R)
          ┌──────────────────────┐
     BOOL ┤ REQ   DONE ├ BOOL
INT(REAL) ┤ X   STAT ├ USINT
     REAL ┤ RATE   Y ├ INT(REAL)
INT(REAL) ┤ X_max   X_max_AL ├ BOOL
INT(REAL) ┤ X_min   X_min_AL ├ BOOL
INT(REAL) ┤ Y_max   Y_max_AL ├ BOOL
INT(REAL) ┤ Y_min   Y_min_AL ├ BOOL
          └──────────────────────┘
</pre>
</td><td>
Input REQ : Function block execution request
X : Input
RATE : Rate
X_max : Max. input limit
X_min : Min. input limit
Y_max : Max. output limit
Y_min : Min. output limit

Output DONE : On if done without error
STAT : State alarm
Y : Output value
X_max_AL : Input high alarm
X_min_AL : Input low alarm
Y_max_AL : Output high alarm
Y_min_AL : Output low alarm
</td></tr>
</table>

■ **Functions**

(1) It outputs a certain ratio of input X to Y.
(2) Note that the reference point is not 0 but X_min.
(3) Output Y is calculated from the equation, $Y = (X - X\_min) \times \frac{RATE}{100} + X\_min$.
(4) X_max and X_min limit the max./min. values of X; it operates with X_max, instead of X if X is not less than X_max, and vice versa.
(5) Y_max and Y_min limit the max./min. values of Y; it operates with Y_max if Y is not less than Y_max, and vice versa.
(6) In case of not less than the max. value or not more than the min. value set in I/O, it displays X_max_AL, X_min_AL, Y_max_AL or Y_min_AL alarm.

■ **Program Example**



1. **In case of X = 20000 & RATE = 50 :** If X is not less than X_max, X_max, 10000 is input,

$$Y = (10000 - (-10000)) \times \frac{50}{100} + (-10000)\ , \quad X\_max\_AL = on$$

Y = 0

2. **In case of X=1000 & RATE=20 :** X is input with 1000,

$$Y = (1000 - (-10000)) \times \frac{20}{100} + (-10000)$$

Y = -7800

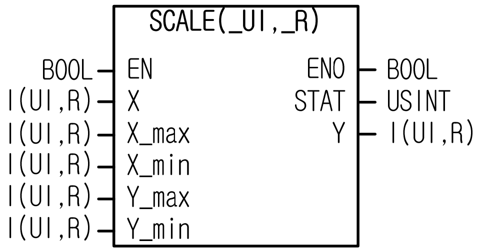
3. **In case of X = 20000, RATE = -250** : since X is not less than X_max, it is operated with X_max, 10000,

$$-60000 = (10000 - (-10000)) \times \frac{-250}{100} + (-10000)\ ,X\_max\_AL = on,\ Y\_min\_AL = on$$

Since Y is not more than Y_min, it is output with Y_min,

Y = -20000

SCALE(_UI, _R)	Scale converter	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function	Description
	**Input** EN : Function execution request X : Input X_max : Max. input limit X_min : Min. input limit Y_max : Max. output scale Y_min : Min. output scale **Output** ENO : On if done without error STAT : State alarm Y : Output value

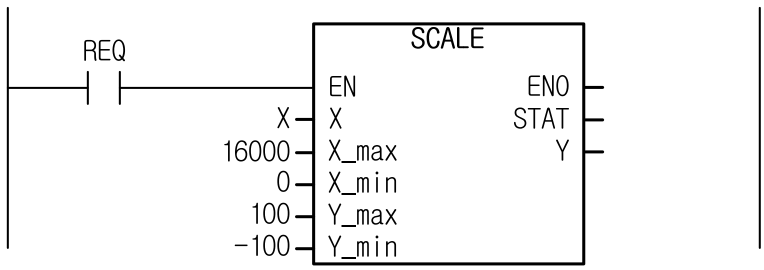
■ **Functions**

(1) It changes input X to the scale set after limiting the max./min. values.
(2) It sets the range of input X to X_max, X_min and that of Y to Y_max, Y_min.
(3) The output equation is as follows.

$$Y = (X - X_{min})\,\frac{Y_{max} - Y_{min}}{X_{max} - X_{min}} + Y_{min}$$

(4) If X_max and X_min are same, it outputs 8 to STAT because the denominator of the equation is 0.
(5) If X input value exceeds X_min ~ X_max, it outputs each X_max, X_min.

■ **Program Example**



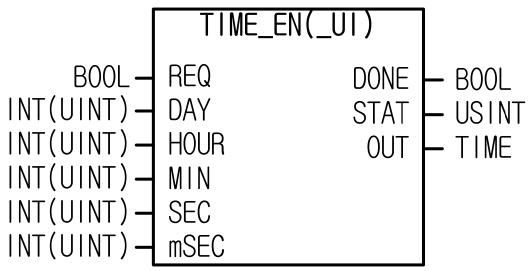
It scales the value between 0 ~ 16000 to a value between -100 ~ 100.

(1) If X is 4000: $Y = (4000 - 0)\,\frac{100 + 100}{16000 - 0} - 100 = -50$

(2) If X is 20000: it limits X to 16000, $Y = (20000 - 0)\,\frac{100 + 100}{16000 - 0} - 100 = 150$

Despite of Y = 150, it outputs Y = 100 because of Y_max = 100.

TIME_EN(_UI)	Converting day, hour, minute, second and 1/1000 sec to TIME type data	
	Availability	XGI, XGR, XEC(U)
	Flags	-

<table>
<tr><th>Function block</th><th>Description</th></tr>
<tr><td>


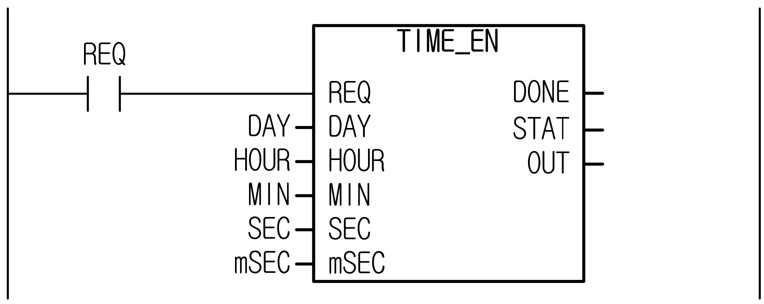
</td><td>
Input REQ : Function block execution request
DAY : day
HOUR : hour
MIN : minute
SEC : second
mSEC : 1/1000 second

Output DONE : On if done without error
STAT : State alarm
OUT : Time output value
</td></tr>
</table>

■ **Functions**

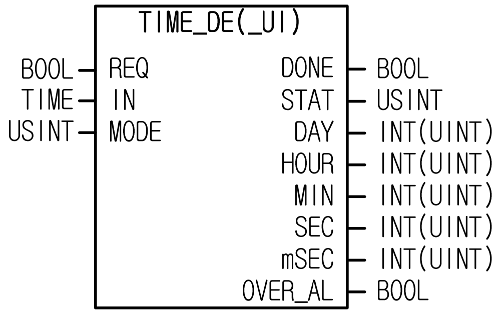
(1) It converts day, hour, minute, second and 1/1000 second data to TIME type parameter.
(2) If input is negative or if output result is output of the data expression range (0~49d17h2m47s295ms) of TIME type data, it generates STAT 8 and does not execute any operation.

■ **Program Example**



(1) In case of DAY=1, HOUR=1, MIN= 1, SEC=1, mSEC=1, it is OUT = T#1d1h1m1s1ms
(2) In case of DAY=0, HOUR=0, MIN=30000, SEC=0, mSEC=0, it is OUT = T#0d20h20m0s0ms

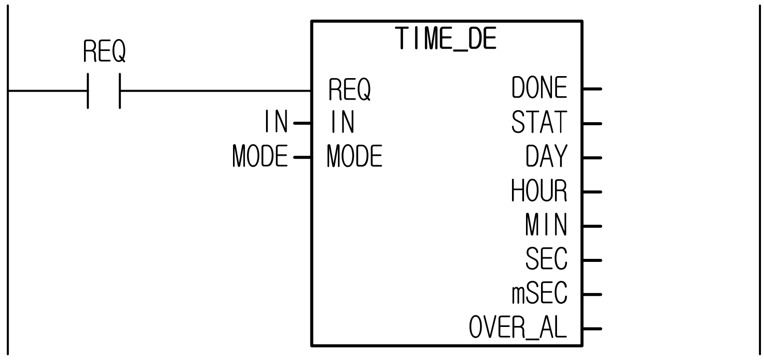
TIME_DE(_UI)	Separating TIME type data to day, hour, minute, second and 1/1000 second	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request IN : Time input MODE : Output mode(0~4) **Output** DONE : On if done without error STAT : State alarm DAY : Day HOUR : Hour MIN : Minute SEC : Second mSEC : 1/1000 second OVER_AL : Overflow alarm

■ **Functions**

(1) It outputs TIME type input separately by day, hour, minute, second and 1/1000 second.
(2) It outputs as follows, depending on mode.
   A. MODE 0 : display all day/hour/minute/second/ms
   B. MODE 1 : display hour/minute/second/ms
   C. MODE 2 : display minute/second/ms
   D. MODE 3 : display second/ms
   E. MODE 4 : display ms only
(3) If it is out of the range of output data, it outputs the max. value, (65535 in case of TIME_DE_UI) and sets OVER_AL.
(4) If MODE is more than 5, it indicates STAT 8 and does not work.

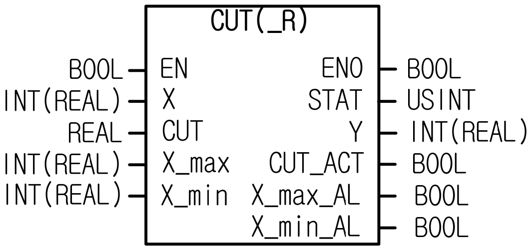
■ **Program Example**



(1) In case of IN =T#1d1h1m1s1ms, MODE = 0; DAY =1, HOUR= 1, MIN= 1, SEC= 1, mSEC= 1, OVER_AL=off
(2) In case of IN =T#1d1h1m1s1ms, MODE = 1; DAY =0, HOUR=25, MIN= 1, SEC= 1, mSEC= 1, OVER_AL=off
(3) IN case of IN =T#1d1h1m1s1ms, MODE = 2; DAY =0, HOUR= 0, MIN=1501, SEC= 1, mSEC= 1, OVER_AL=off
(4) In case of IN =T#1d1h1m1s1ms, MODE = 3; DAY =0, HOUR= 0, MIN= 0, SEC=32767, mSEC= 1, OVER_AL=on
(5) In case of IN =T#1d1h1m1s1ms, MODE = 4; DAY =0, HOUR= 0, MIN= 0, SEC= 0, mSEC=32767, OVER_AL=on
(6) In case of IN =T#90061001ms, MODE = 0; input is modified and displayed as T#1d1h1m1s1ms.

The results are DAY=1, HOUR=1, MIN=1, SEC=1, mSEC=1, OVER_AL=off.

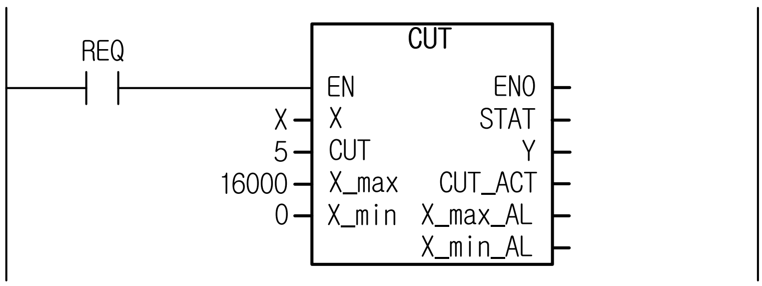
CUT(_R)	Small signal cut filter	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function	Description
	**Input** EN : Function execution request X : Input CUT : Small signal cut range (%) X_max : Max. input limit X_min : Min. input limit **Output** ENO : On if done without error STAT : State alarm Y : Output value CUT_ACT : CUT operation in progress. X_max_AL : Input max. limit alarm X_min_AL : Input min. limit alarm

■ **Functions**

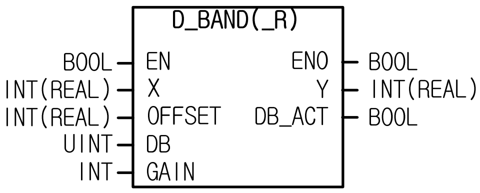
(1) If input is a value between [X_min] and [CUT% of X_min ~ X_max], it is ignored and the system outputs X_min.
(2) Note that the reference point is not 0 but [X_min].
(3) For input, the max./min. values are limited by X_max/X_min, which is notified by alarm: X_max_AL and X_min_AL.
(4) If the input of max./min. limit is $X \le X\_min + CUT \dfrac{X\_max - X\_min}{100}$,
it outputs Y = X_min and CUT_ACT is on.
(5) If X_min is larger than X_max, STAT indicates 2 and outputs 0.

■ **Program Example**



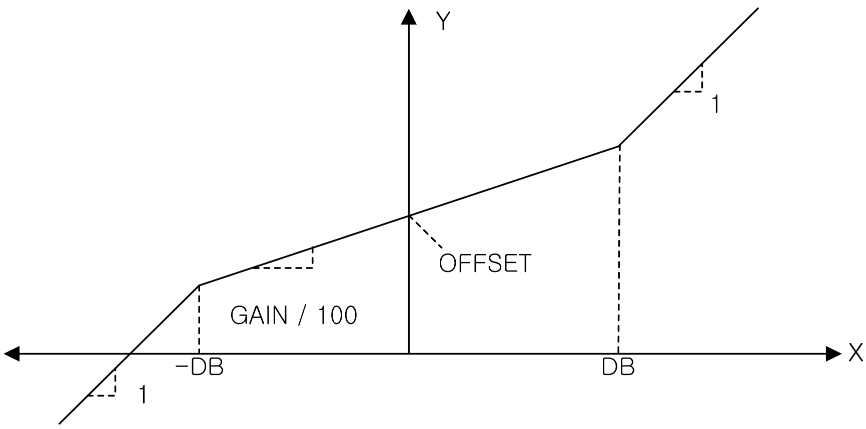
(1) If X is 4000 : since it is not in 5% (CUT) of 16000 (Xmax - Xmin), 4000 is output with no change.
(2) If X is 18000 : since it is limited to 16000, the value of 16000 is output and X_max_AL is on.
(3) If X is 100 : since it is not more than 800, 5% of 16000, it outputs 0(X_min) and CUT_ACT is on.

D_BAND(_R)	Deadband Application Output	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function	Description
	**Input** EN : Function execution request X : Input OFFSET : Output offset DB : Deadband half width GAIN : GAIN(%) of Deadband section **Output** ENO : On if done without error Y : Output value DB_ACT : Alarm if input is within DB

■ **Functions**

(1) Output Y is calculated by applying deadband to input X.
(2) Since DB represents scale, it should be used through absolute value operation like |DB|.
(3) Deadband is set with a range of –|DB| ~ |DB|.
(4) DB_ACT bit is on if input X is within deadband.
(5) Both ends of deadband affect the output outside the deadband.
(6) If operation result is out of the data expression range of integer(INT), the output is limited to INT (-32768 ~ 32767).
(7) If operation result is out of the data expression range of real number (REAL), output is indicated '1.#inf00000 E+000' or '-1.#inf00000 E+000'and in the case, ENO bit is off.
(8) The I/O equation of deadband is as follows.



A. UNDER THE BAND (X is not more than -|DB|) :

$$Y = X - (\frac{GAIN}{100} \times DB) + DB + OFFSET$$

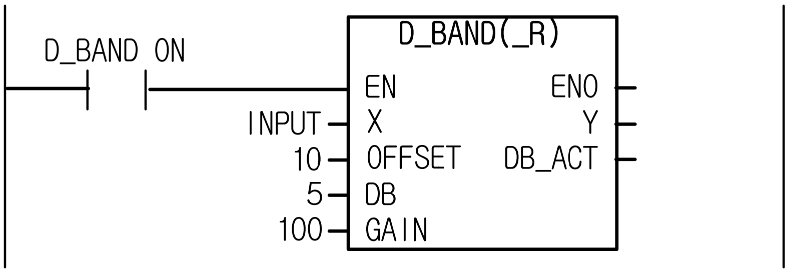
B. IN THE BAND (X is within -|DB| ~ |DB|) :

$$Y = (\frac{GAIN}{100} \times X) + OFFSET$$

C. OVER THE BAND (X is larger than |DB|) :

$$Y = X + (\frac{GAIN}{100} \times DB) - DB + OFFSET$$

■ **Program Example**



1. If INPUT is -8 :

$$-8_{(X)} - (\frac{100_{(GAIN)}}{100} \times 5_{(DB)}) + 5_{(DB)} + 10_{(OFFSET)} = 2_{(Y)}$$

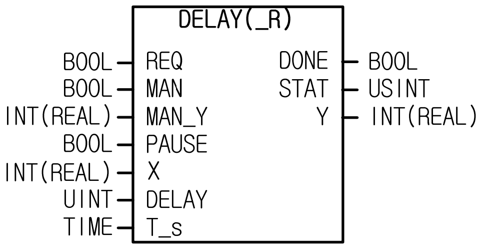
2. If INPUT is 3 : X is within DB = 5, DB_ACT is on

$$(\frac{100_{(GAIN)}}{100} \times 3_{(X)}) + 10_{(OFFSET)} = 13_{(Y)}$$

3. If INPUT is 16 :

$$16_{(X)} + (\frac{100_{(GAIN)}}{100} \times 5_{(DB)}) - 5_{(DB)} + 10_{(OFFSET)} = 26_{(Y)}$$

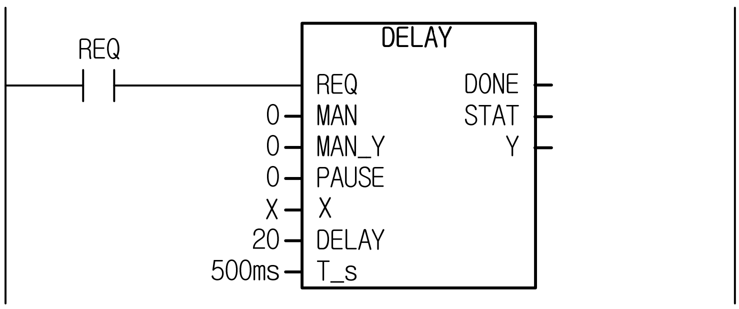
DELAY(_R)	Delay Output	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request MAN : Manual mode MAN_Y : Manual mode output PAUSE : Pause X : Input DELAY : No. of Delay sample T_s : Operation cycle **Output** DONE : On if done without error STAT : State alarm Y : Output value

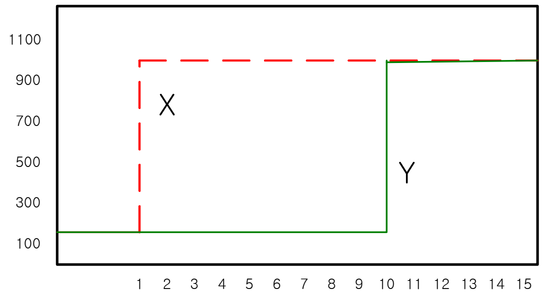
■ **Functions**

(1) It generates output X of which input X is delayed as much as T_s * DELAY (T_s unit : [sec]).
(2) It saves the current input every scan cycle and outputs the previous input at the same time.
(3) If the first operation is permitted, it outputs 0 as much as T_s * DELAY because there is no previous value.
(4) It is possible to input DELAY scan up to 100 scans; if more value is input, it outputs 8 to the STAT and does not work.
(5) If PAUSE is on, output pauses and the current data are saved.
(6) If MAN is on, it outputs MAN_Y in manual mode and it does not save the current data, so it outputs 0 as much as T_s * DELAY when it returns to auto mode.

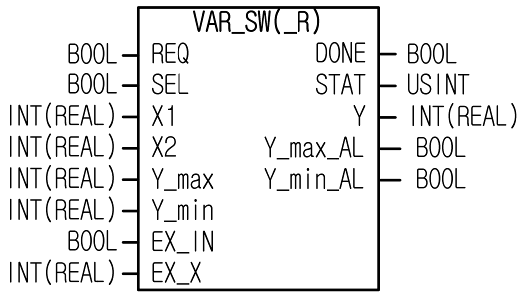
■ **Program Example**



(1) Since DELAY is 20 and T_s is 500ms, Y outputs X value 10s before.


1100
900
700
500
300
100
X
Y
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

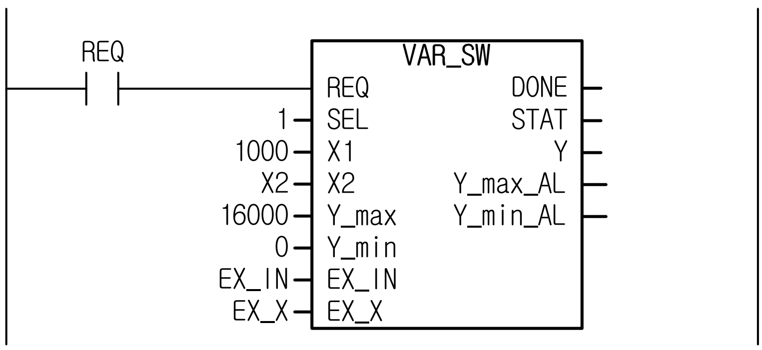
VAR_SW(_R)	Constant selection switch	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request SEL : Select Input 1/2 X1 : Input 1 X2 : Input 2 Y_max : Max. output limit Y_min : Min. output limit EX_IN : Select external input EX_X : External input **Output** DONE : On if done without error STAT : State alarm Y : Output value Y_max_AL : Over max. output alarm Y_min_AL : Less min. output alarm

■ **Functions**

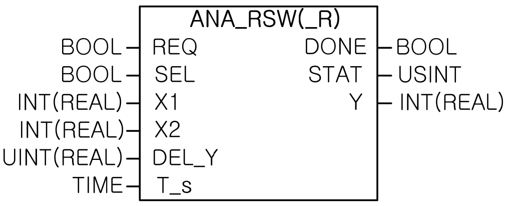
(1) It outputs X1 or X2 depending on SEL bit setting.
(2) The max./min value of output may be limited by setting Y_max and Y_min.
(3) It is possible to output EX_IN by connecting external devices (MMI and etc) to EX_X.
(4) EX_X is also limited by the max./min. values.
(5) If Y_min is larger than Y_max, STAT outputs 4.

■ **Program Example**



Since SEL is 1, it outputs X2 if EX_IN is off.
(1) If X2 is 10000 and EX_IN is off: X2 is applied and it outputs 10000.
(2) If X2 is 20000 and EX_IN is off: X2 is applied and after being limited by the max. value, it outputs 16000 and Y_max_AL is on.
(3) If X2 is 1000 and in case of EX_IN=on, EX_X=-1000: EX_IN is applied and after being limited by the min. value, it outputs 0 and Y_min_AL is on.

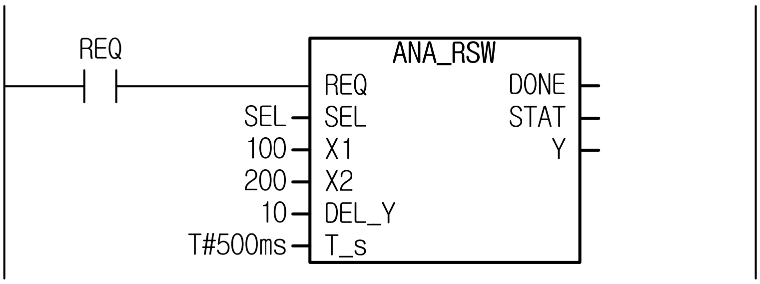
ANA_RSW(_R)	Analog increment limit switch	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request SEL : Select input X1 : Input 1 X2 : Input 2 DEL_Y : Output increment limit T_s : Operation cycle **Output** DONE : On if done without error STAT : State alarm Y : Output value

■ **Functions**

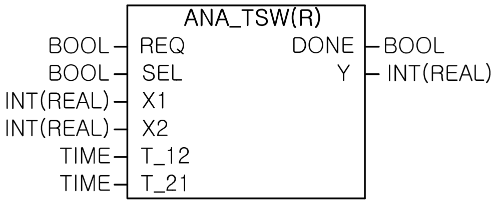
(1) It selectively outputs X1 or X2 depending on SEL bit setting.
(2) RESET works as soon as REQ is on. Therefore, it outputs the input selected by SEL as its initial value.
(3) If SEL bit is changed, it reaches to the value (X1 / X2) selected as Y increases or decreases as much as DEL_Y every T_s.
(4) Even though SEL bit is not changed, it reaches to the value selected as Y increases or decreases as much as DEL_Y every T_s if the value selected by SEL (X1 / X2) is changed.

■ **Program Example**



(1) If it is changed from SEL=off to SEL=on, Y increases by 10 every 500ms and it reaches to Y=200.
(2) If X1 is changed to 300 with SEL=off, Y increases by 10 every 500ms and it reaches to Y=300 in 10s.

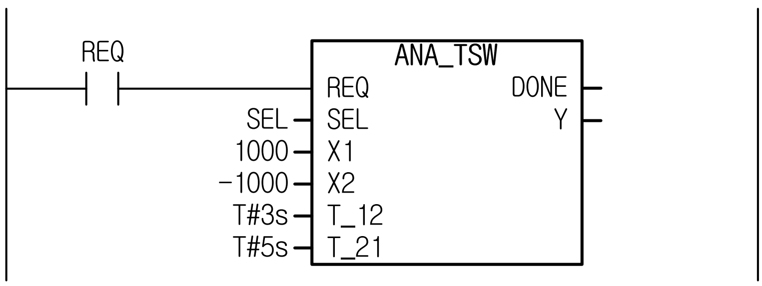
ANA_TSW(_R)	Analog time limit switch	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request SEL : Select input X1 : Input 1 X2 : Input 2 T_12 : Input 1->2 conversion time T_21 : Input 2->1 conversion time **Output** DONE : On if done without error Y : Output value

■ **Functions**

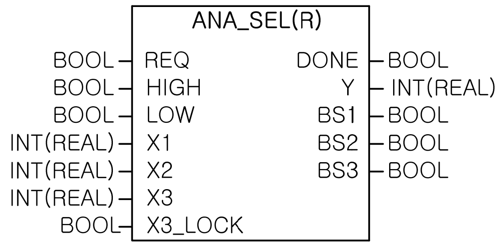
(1) It selectively outputs X1 or X2 depending on SEL bit setting.
(2) RESET works as soon as REQ is on. Therefore, it outputs the input selected by SEL as its initial value.
(3) It changes the data before SEL change to the data after SEL change gradually (RAMP), based on the pre-determined time.
(4) If it is changed from X1 to X2, depending on SEL selection, it follows T_12 time; if it is conversely changed from X2 to X1, it follows T_21 time.
(5) An integer type instruction, ANA_TSW is subject to round-off during the conversion, so it has an error up to 0.5. therefore, it may reach to the target input earlier than the pre-determined time.
(6) If the operation result is out of the data expression range of integer (INT), the output is limited to INT (-32768 ~ 32767).
(7) If the operation result is out of the data expression range of real number (REAL), the output displays as '1.#inf00000 E+000'or '-1.#inf00000 E+000' and in the case, DONE bit is off.

■ **Program Example**



(1) In case of SEL=off → on : it decreases toward Y=1000 → -1000 for 3s.
(2) In case of SEL=on → off: it increases toward Y=-1000 → 1000 for 5s.

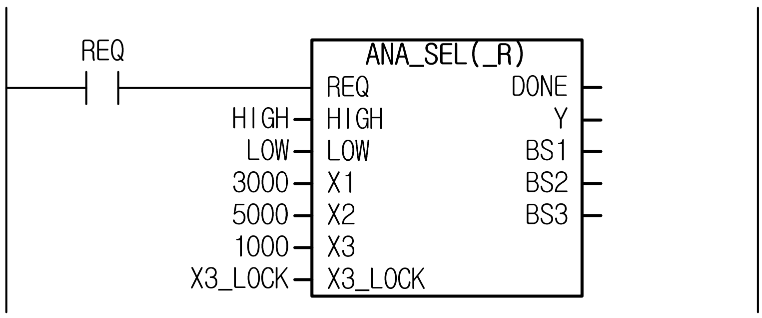
ANA_SEL(_R)	Analog scale comparative switch	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request HIGH : Select scale-based input LOW : Select scale-based input X1 : Input 1 X2 : Input 2 X3 : Input 3 X3_LOCK : Input 3 effective bit **Output** DONE : On if done without error Y : Output value BS1 : Block select1 BS2 : Block select2 BS3 : Block select3

■ **Functions**

(1) In case of HIGH = on, LOW = off, it outputs the highest one among X1 ~ X3 and the corresponding BS is on.
(2) In case of HIGH = off, LOW = on, it outputs the lowest one among X1 ~ X3 and the corresponding BS is on.
(3) If HIGH = low (both on or off) is set, it selects a middle one. It outputs a middle value among X1 ~ X3 and the corresponding BS is on.
(4) After selecting a middle value as above, if two inputs are same, it outputs the two values to output Y and the corresponding two BS are on.
(5) After selecting a middle value, if three inputs are same, it outputs these three values to output Y and every BS is on.
(6) In case of X3_LOCK = on, X3 among the inputs is disregarded. In the case, it is equal to 2 input, so the middle value is defined as a larger one between them.

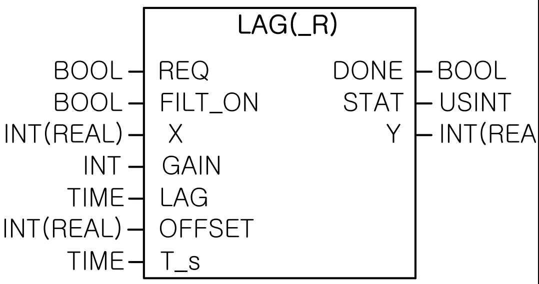
■ **Program Example**



(1) In case of HIGH = on, LOW = off, X3_LOCK = off, it outputs Y = 5000 and BS2 is on.
(2) In case of HIGH = on, LOW = on, X3_LOCK = off, it outputs Y = 3000 and BS1 is on.
(3) In case of HIGH = off, LOW = off, X3_LOCK = off, it outputs Y = 3000 and BS1 is on.
(4) In case of HIGH = off, LOW = on, X3_LOCK = off, it outputs Y = 1000 and BS3 is on.

(5) In case of HIGH = off, LOW = on, X3_LOCK = on, it outputs Y = 3000 and BS1 is on.
(6) In case of HIGH = on, LOW = on, X3_LOCK = on, it outputs Y = 5000 and BS2 is on.

LAG(_R)	HF limit filter	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request FILT_ON : Filter ON X : Input GAIN : Filter gain (%) LAG : LAG filter coefficient OFFSET : Output offset T_s : Operation cycle **Output** DONE : On if done without error STAT : State alarm Y : Output value

■ **Functions**

(1) It processes with filter limiting HF components.
(2) Input X is outputted to output Y via LAG filter.
(3) The input-output procedure may have an error lower than 0.001%.
(4) If FILT_ON bit is off, LAG filter does not filtrate input and the output equation is as follows.

$$Y' = \frac{GAIN}{100} \times X$$

(5) If FILT_ON bit is on, LAG filter operates and the output equation is as follows.

$$Y' = Y'_{old} + \frac{T\_s}{LAG + T\_s} \times \left(\frac{GAIN}{100} \times \frac{X + X_{old}}{2} - Y'_{old} \right)$$

T_s : [sec]

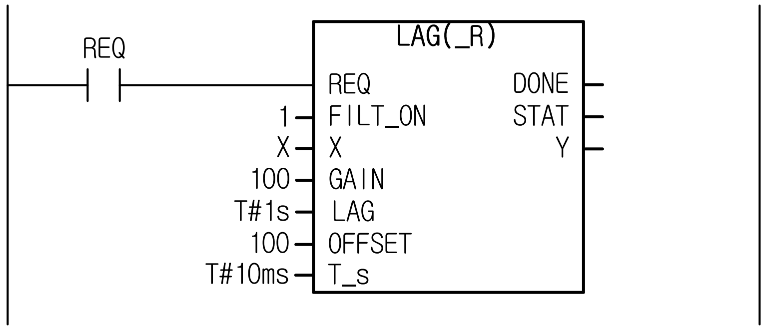
(6) After the filter operation, OFFSET is added to the internal output value and the offset does not pass the filter.

$$Y = Y' + OFFSET$$

Note) in the above equation, Y represents actual output while Y' represents internal output.

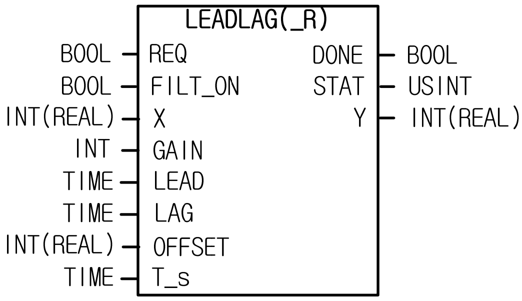
(7) If in the LAG_R operation, the data are out of the expression range of real number parameter(REAL), it indicates STAT 8 and outputs 0.

■ **Program Example**



If input X is changed with REQ and FILT_ON turned on, it filtrates HF component and outputs.
It is operated by I/O equation every 10ms (T_s), it generates output.

LEADLAG(_R)	HF/LF limit filter	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request FILT_ON : Filter ON X : Input GAIN : Filter gain (%) LEAD : LEAD filter coefficient LAG : LAG filter coefficient OFFSET : Output offset T_s : Operation cycle **Output** DONE : On if done without error STAT : State alarm Y : Output value

■ **Functions**

(1) It processes with filter limiting HF/LF components
(2) Output is generated through LEAD filter and LAG filter.
(3) The input-output procedure may have an error lower than 0.001%.
(4) If FILT_ON bit is off, LEADLAG filter does not filtrate input and the output equation is as follows.

$$Y' = \frac{GAIN}{100} \times X$$

(5) If FILT_ON bit is on, LEADLAG filter operates and the output equation is as follows.

$$Y' = \frac{LAG \times Y'_{old} + GAIN((LEAD + T\_s)X - LEAD \times X_{old})}{LAG + T\_s}$$

T_s : [sec]

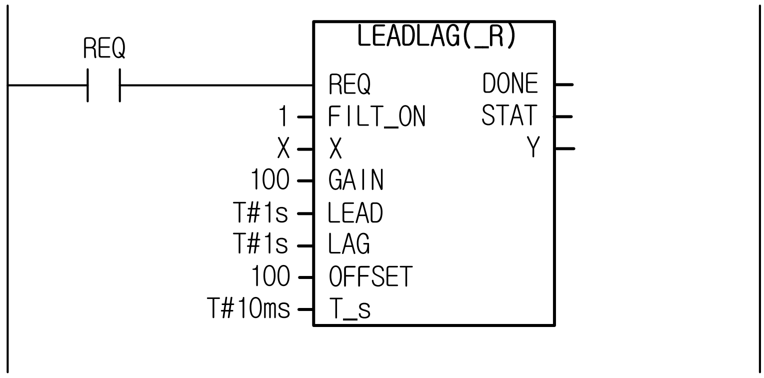
(6) After the filter operation, OFFSET is added to the internal output value and the offset does not pass the filter.

$$Y = Y' + OFFSET$$

Note) in the above equation, Y represents actual output while Y' represents internal output.

(7) If in the LEADLAG_R operation, the data are out of the expression range of real number parameter (REAL), it indicates STAT 8 and outputs 0.

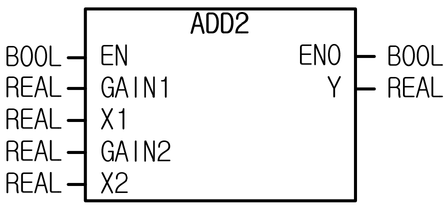
■ **Program Example**



If input X is changed with REQ and FILT_ON turned on, it filers HF/LF component and outputs.
It is operated by I/O equation every 10ms (T_s), it generates output.

13.4 Arithmetic Operation Function, Function Block

ADD2	Y = G1X1 + G2X2	
	Availability	XGI, XGR, XEC(U)
	Flags	-

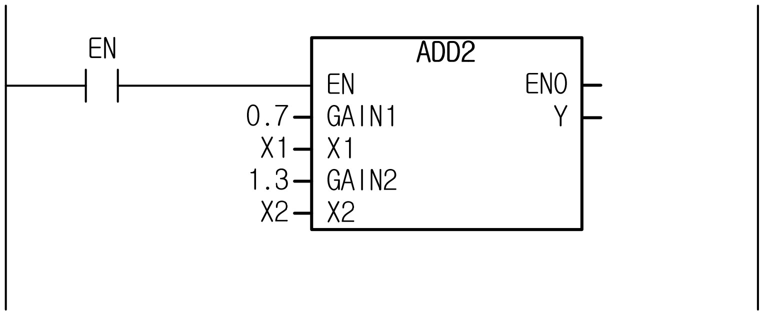
Function	Description
	**Input** EN : Function execution request GAIN1 : Operation gain 1 X1 : Input 1 GAIN2 : Operation gain 2 X2 : Input 2 **Output** ENO : On if done without error Y : Output value

■ **Function**

(1) It executes the pre-determined arithmetic operations.
(2) If the operation result is out of the data expression range of Y (REAL), ENO is off and it is displayed as 1.#inf00000 E+000', '-1.#inf00000 E+000', '1.#QNAN0000e+000'and in the case, DONE bit is off.

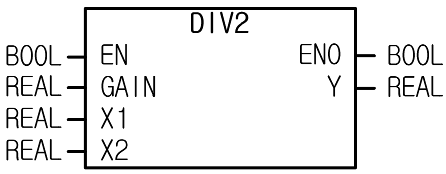
$$Y = GAIN1 * X1 + GAIN2 * X2$$

■ **Program Example**



In case of X1 = 10.0, X2 = 20.0, it results in 'Y = 0.7 (10.0) + 1.3 (20.0) = 7.0 + 26.0 = 33.0.

DIV2	Y = Gain (X1 / X2)	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function	Description
	**Input** EN : Function execution request GAIN : Operation gain X1 : Input 1 X2 : Input 2 **Output** ENO : On if done without error Y : Output value

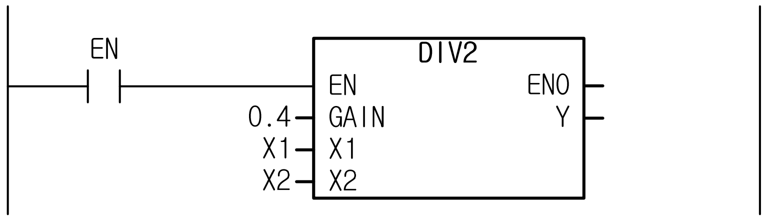
■ **Functions**

(1) It executes the pre-determined arithmetic operations.

$$Y = GAIN\ (X1\ /\ X2)$$

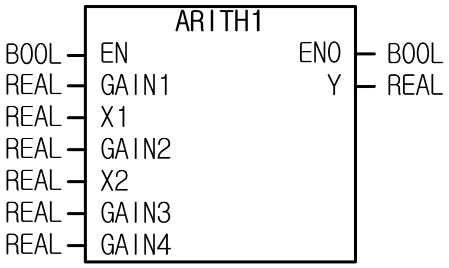
(2) If X2 value is 0, it outputs '1.#QNAN0000 E+000'because its denominator is 0.
(3) If the operation result is out of the data expression range of Y(REAL), ENO is off and it is displayed as 1.#inf00000 E+000'or '-1.#inf00000 E+000'and in the case, DONE bit is off.

■ **Program Example**



In case of X1 = 10.0, X2 = 20.0, it results in 'Y = 0.4 (10.0 / 20.0) = 0.2.

ARITH1	Y = (G1X1+G2X2)G3 + G4	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function	Description
	**Input** EN : Function execution request GAIN1 : Operation gain 1 X1 : Input 1 GAIN2 : Operation gain 2 X2 : Input 2 GAIN3 : Operation gain 3 GAIN4 : Operation gain 4 **Output** ENO : On if done without error Y : Output value

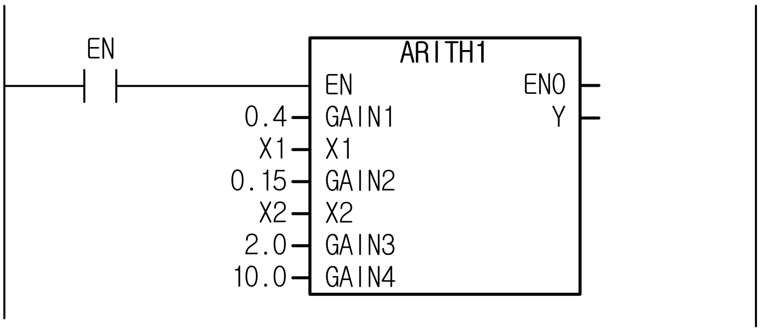
■ **Functions**

(1) It executes the pre-determined arithmetic operations.

$$Y = (GAIN1 \times X1 + GAIN2 \times X2)GAIN3 + GAIN4$$

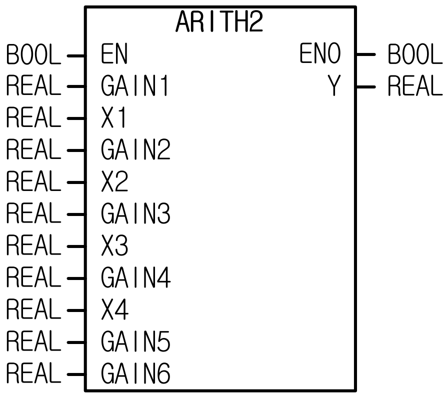
(2) If the operation result is out of the data expression range of Y(REAL), ENO is off and it is displayed as 1.#inf00000 E+000', '-1.#inf00000 E+000', '1.#QNAN0000e+000'and in the case, DONE bit is off.

■ **Program Example**



In case of X1 = 10.0, X2 = 20.0, it results in 'Y = (0.4(10.0)+0.15(20.0))2.0+10.0 = (4.0+3.0)2.0+10.0 = 24.0.

ARITH2	Y = (G1X1+G2X2+G3X3+G4X4)G5 + G6	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function	Description
 	**Input** EN : Function execution request GAIN1 : Operation gain 1 X1 : Input 1 GAIN2 : Operation gain 2 X2 : Input 2 GAIN3 : Operation gain 3 X3 : Input 3 GAIN4 : Operation gain 4 X4 : Input 4 GAIN5 : Operation gain 5 GAIN6 : Operation gain 6 **Output** ENO : On if done without error Y : Output value

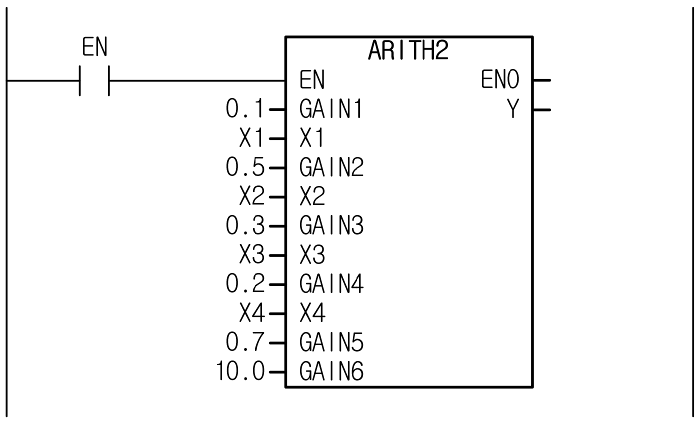
■ **Functions**

(1) It executes the pre-determined arithmetic operations.

$$Y = (GAIN1 \times X1 + GAIN2 \times X2 + GAIN3 \times X3 + GAIN4 \times X4)GAIN5 + GAIN6$$

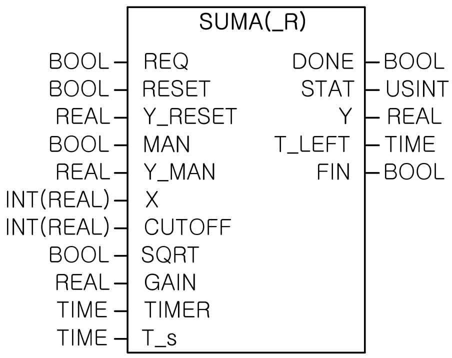
(2) If the operation result is out of the data expression range of Y (REAL), ENO is off and it is displayed as 1.#inf00000 E+000', '-1.#inf00000 E+000', '1.#QNAN0000e+000'and in the case, DONE bit is off.

■ **Program Example**



In case of X1 = 10.0, X2 = 20.0, X3 = 10.0, x4 = 30.0, it results in 'Y = (0.1(10.0)+0.5(20.0)+0.3(10.0)+0.2(30.0))0.7+10.0 = (1+10+3+6)0.7+10.0 = 24.0.

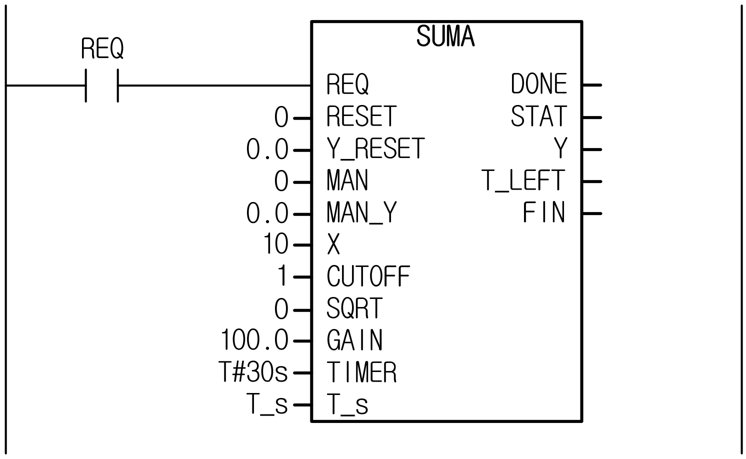
SUMA(_R)	Analog Summer	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request RESET : Block operation reset Y_RESET : reset value MAN : manual mode Y_MAN : Manual output value X : Input CUTOFF : Small signal cut width SQRT : Square root setting GAIN : Input gain (%) TIMER : Timer setting T_s : Operation cycle **Output** DONE : On if done without error STAT : State alarm Y : Output value T_LEFT : Timer left time FIN : Timer finish display

■ **Functions**

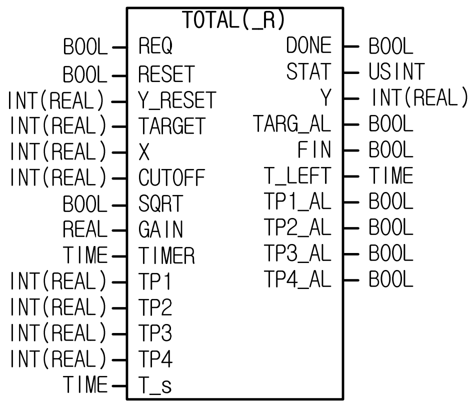
(1) It sums up analog data inputted to X at the preset interval and outputs the result to Y.
(2) SUMA (INT type) instruction supports real number type output to prevent too fast saturation that may occur when output rapidly increases if it is summed up to a direction, whether negative or positive.
(3) If RESET bit is on, it outputs Y_RESET value; if RESET bit is off, it resumes the operation from Y_RESET value.
(4) If MAN bit is on, MAN_Y value is output but if the bit is off, it operates from the first as much as from Y_RESET to TIMER time.
(5) If |X| is equal to or not more than |CUTOFF|, it processes it as X = 0.
(6) If SQRT bit is on, it operates with square-rooted X.
(7) If program scan time is longer than 1m, it may have a skipping section of operation. Therefore, it may have an error less than T_s set time when the timer is finished.
(8) If the operation results is out of the data expression range of Y(REAL), it is indicated with '1.#inf00000 E+000' or '-1.#inf00000 E+000' and in the case, DONE bit is off but the internal state(T_LEFT, FIN and etc) will be normally processed.

■ **Program Example**



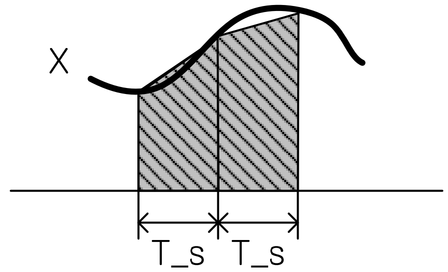
(1) In case of X=10, T_s= T#1s : If REQ is on, Y increases by 10 every second and it outputs Y = 300. Then, it results in 'FIN = on'.
(2) In case of X=10, T_s= T#2s : If REQ is on, Y increases by 10 every 2 seconds and it outputs Y = 150. Then, it results in 'FIN = on'.

TOTAL(_R)	Analog totalizer	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request RESET : Block operation reset Y_RESET : Reset value TARGET : Set value X : Input value CUTOFF : Small signal cut width SQRT : Square root setting GAIN : Input gain (%) TIMER : Operation time TP1 : Trip point 1 TP2 : Trip point 2 TP3 : Trip point 3 TP4 : Trip point 4 T_s : Operation cycle **Output** DONE : On if done without error STAT : State alarm Y : Output value TARG_AL : Set value alarm FIN : Operation finish alarm T_LEFT : Operation time end alarm TP1_AL : Trip point 1 alarm TP2_AL : Trip point 2 alarm TP3_AL : Trip point 3 alarm TP4_AL : Trip point 4 alarm

■ **Functions**

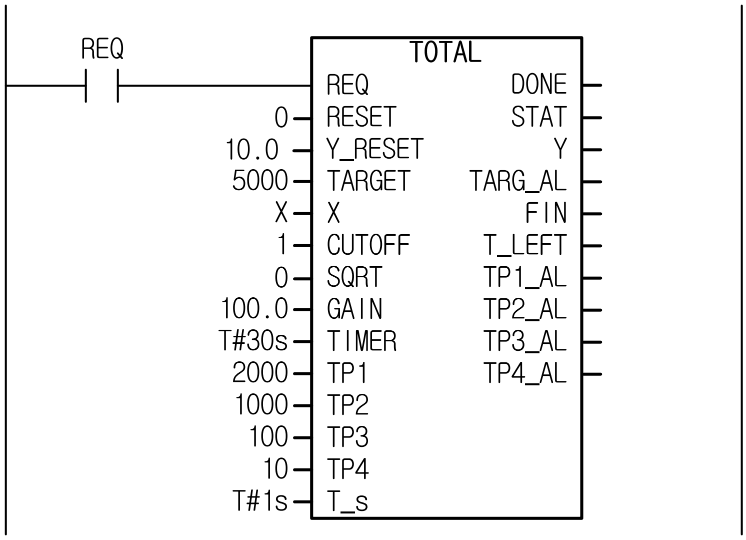
(1) It totals analog data input to X.
(2) Totaling is executed from Y_RESET.
(3) As in the below figure, it totals by means of the operation of trapezoid addition, in which the shaded area is added every T_s of operation cycle, and it applies the delivery rate through gain.



$$Y = Y_{old} + GAIN/100\ (X_{old} + X)T\_s / 2 \qquad T\_s : [sec]$$

(4) If RESET bit is on, it becomes reset and outputs Y_RESET.
(5) If RESET is canceled as RESET bit is off, it restarts the operation from Y_RESET value.
(6) After the set value is set, it notifies a user that output value is more than the set value by means of TARG_AL.
(7) If output value is within TARGET-TP[n] ≤ Y ≤ TARGET+TP[n], it turns on TP[n]_AL and shows how close it approaches to the set value.
(8) Output Y increases or decreases with no influence of target.
(9) If |X| is not more than |CUTOFF|, it processes it as X = 0.
(10) If SQRT bit is on, it operates with square-rooted X.
(11) If program scan time is not less than 1m, it may have a skipping section of operation, so it may have an error less than T_s time.
(12) Input-output may have an error less than 0.001%.
(13) If |GAIN * X| has a huge range over 1.0e+38, it may result in incorrect operation procedure.
(14) If operation result is out of the data expression range of integer(INT), the output is limited to INT (-32768 ~ 32767).
(15) If operation result is out of the data expression range of real number (REAL), output is displayed as 1.#inf00000 E+000' or '-1.#inf00000 E+000'. In the case, DONE bit is off but the internal state (T_LEFT, FIN and etc) is normally processed.

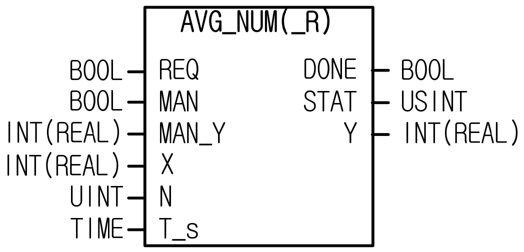
■ **Program Example**



(1) In case of X=200, T_s=T#1s: output Y increases from 10 (Y_RESET) by 100 for the first cycle (trapezoid addition). Then, it increases by 200 per second from the next cycle and it outputs 5910 in 30s.
TARG_AL is on in case of Y ≥ 5000
TP1_AL is on in case of 5000 – TP1 ≤ Y ≤ 5000 + TP1
TP2_AL is on in case of 5000 – TP2 ≤ Y ≤ 5000 + TP2
TP3_AL is on in case of 5000 – TP3 ≤ Y ≤ 5000 + TP3
TP4_AL is on in case of 5000 – TP4 ≤ Y ≤ 5000 + TP4

(2) In case of X=200, T_s=T#5s: output Y increases from 10 (Y_RESET) by 500 for the first cycle (trapezoid addition). Then, it increases by 1000 per 5 seconds from the next cycle and it outputs 5510 in 30s.
TARG_AL is on in case of Y ≥ 5000
TP1_AL is on in case of 5000 – TP1 ≤ Y ≤ 5000 + TP1
TP2_AL is on in case of 5000 – TP2 ≤ Y ≤ 5000 + TP2
TP3_AL is on in case of 5000 – TP3 ≤ Y ≤ 5000 + TP3
TP4_AL is on in case of 5000 – TP4 ≤ Y ≤ 5000 + TP4

AVG_NUM(_R)	Average number output	
	Availability	XGI, XGR, XEC(U)
	Flags	_LER

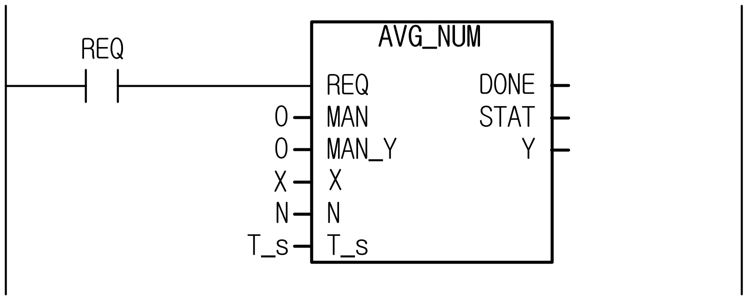
<table>
<tr><th>Function block</th><th>Description</th></tr>
<tr><td>
</td><td>Input REQ : Function block execution request
MAN : Manual mode setting
MAN_Y : Manual output
X : Input
N : Average number
T_s : Operation cycle

Output DONE : On if done without error
STAT : State alarm
Y : Output value</td></tr>
</table>

■ **Functions**

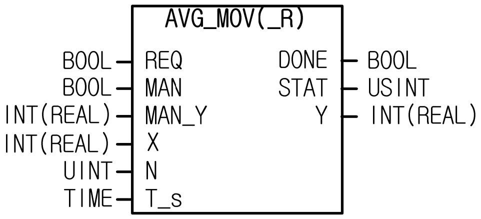
(1) It receives input X every T_s and outputs N average value.
(2) Output Y is updated with a new average every N * T_s.
(3) If MAN bit is on, T_s is disregarded; output Y has MAN_Y.
(4) If N is 0 or not less than 30001, it outputs 8 to STAT.
(5) If operation result is out of the data expression of integer(INT), the output is limited to INT (-32768 ~ 32767).
(6) If in the operation procedure, X * N is out of the data expression range of real number (REAL), the output is indicated as '1.#inf00000 E+000' or '-1.#inf00000 E+000' and _LER flag is set. In the case, DONE bit is off.

■ **Program Example**



(1) X increases by 1 per second from 0, T_s= T#1s, N=3 : Y increases by 3 per 3s
(2) X increases by 1 per second from 0, T_s= T#2s, N=3 : Y increases by 6 per 6s
(3) X increases by 1 per second from 0, T_s= T#1s, N=6 : Y increases by 6 per 6s

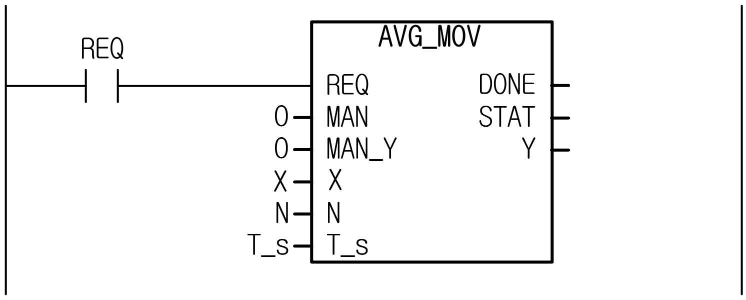
AVG_MOV(_R)	Moving average output	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
 	**Input** REQ : Function block execution request MAN : Manual mode setting MAN_Y : Manual output X : Input N : Average number T_s : Operation cycle **Output** DONE : On if done without error STAT : State alarm Y : Output value

■ **Functions**

(1) It receives input X every T_s and outputs the values before the present time and N average value.
(2) Output Y is updated with a new average every T_s.
(3) If MAN bit is on, T_s is disregarded; output Y has MAN_Y.
(4) If N is 0 or not less than 101, it outputs 8 to STAT.
(5) If operation result is out of the data expression of integer (INT), the output is limited to INT (-32768 ~ 32767).
(6) If in the operation procedure, X * N is out of the data expression range of real number (REAL), the output is indicated as '1.#inf00000 E+000' or '-1.#inf00000 E+000' and in the case, DONE bit is off.

■ **Program Example**



(1) X increases by 1 from 0, T_s= T#1s, N=3 : Y increases by 1 per second
(2) X increases by 1 from 0, T_s= T#2s, N=3 : Y increases by 2 per 2 seconds
(3) X increases by 1 from 0, T_s= T#1s, N=6 : Y increases by 1 per second

13.5 Data Measuring Function, Function Block

ALARM_R	Alarm indicator	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
(see diagram below)	(see description below)

```
              ALARM_R
BOOL ─ REQ              DONE ─ BOOL
 INT ─ X                   Y ─ REAL
BOOL ─ YH1_OFF          STAT ─ USINT
BOOL ─ YH2_OFF        YH1_AL ─ BOOL
BOOL ─ YL1_OFF        YH2_AL ─ BOOL
BOOL ─ YL2_OFF        YL1_AL ─ BOOL
REAL ─ YH1            YL2_AL ─ BOOL
REAL ─ YH2          X_max_AL ─ BOOL
REAL ─ YL1          X_min_AL ─ BOOL
REAL ─ YL2
TIME ─ YH1_DT
TIME ─ YH2_DT
TIME ─ YL1_DT
TIME ─ YL2_DT
 INT ─ X_MAX
 INT ─ X_MIN
REAL ─ Y_sMAX
REAL ─ Y_sMIN
```

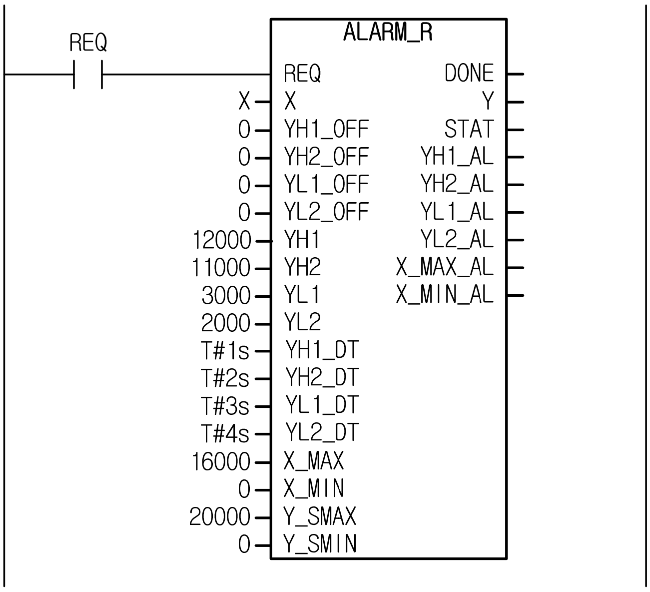
Input REQ : Function block execution request
X : Input
YH1_OFF : Output value high 1 section off bit
YH2_OFF : Output value high 2 section off bit
YL1_OFF : Output value low 1 section off bit
YL2_OFF : Output value low 2 section off bit
YH1 : Output high 1 section value
YH2 : Output high 2 section value
YL1 : Output low 1 section value
YL2 : Output low 2 section value
YH1_DT : Output high 1 section waiting time (sec)
YH2_DT : Output high 2 section waiting time (sec)
YL1_DT : Output low 1 section waiting time (sec)
YL2_DT : Output low 2 section waiting time (sec)
X_MAX : Max. input limit
X_MIN : Min. input limit
Y_sMAX : Max. output scale
Y_sMIN : Min. output scale

Output DONE : On if done without error
Y : Output value
STAT : State alarm
YH1_AL : Output high 1 section alarm
YH2_AL : Output high 2 section alarm
YL1_AL : Output low 1 section alarm
YL2_AL : Output low 2 section alarm
X_max_AL: Input high alarm
X_max_AL: Input high alarm

■ **Functions**

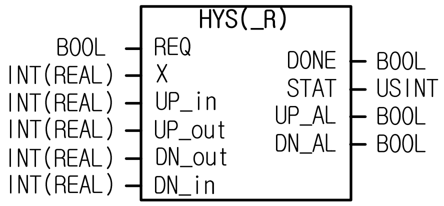
(1) It changes and outputs integer input X to real number; it can execute the operations of 2 upper limits, 2 lower limits and scale.
(2) Since input is integer type, it receives input from special module or external device and uses it as its input with no conversion.
(3) It executes scale operation from the value between X_MIN ~ X_MAX to the value between Y_sMIN ~ Y_sMAX.
(4) YH1 and YH2 may set high limits and notify an operator of any fault; with it, an operator may set whether to use the function (YH_OFF) and the delay time (YH_DT).
(5) YL1 and YL2 may set low limits and notify an operator of when it is not more than it; with it, an operator may set whether to use the function (YL_OFF) and the delay time (YL_DT).
(6) In case of X_max = X_min, it does not work because the denominator is 0 and STAT outputs 8.

■ **Program Example**



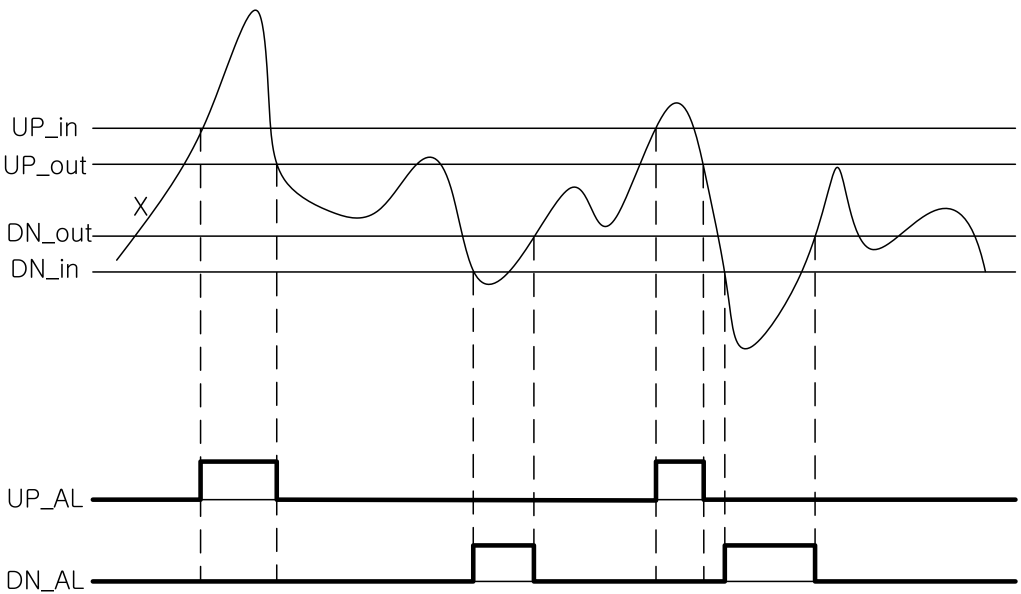
(1) In case of X = 8900: Y = 11125, YH2_AL on in 2s
(2) In case of X = 11000: Y = 13750, YH1_AL on in a second, YH2_AL on in 2s
(3) In case of X = 2100: Y = 2625, YL1_AL on in 3s
(4) In case of X = 1200: Y = 1500, YL1_AL on in 3s, YL2_AL on in 4s.

HYS(_R)	Directional deadband	
	Availability	XGI, XGR, XEC(U)
	Flags	-

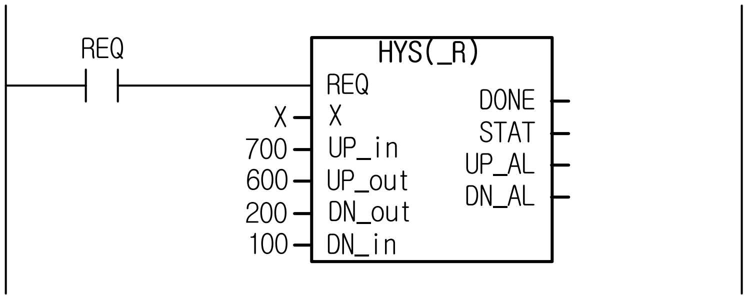
Function block	Description
	**Input** REQ : Function block execution request X : Input UP_in : Up set trigger UP_out : Up reset trigger DN_out : Down reset trigger DN_in : Down set trigger **Output** DONE : On if done without error STAT : State alarm UP_AL : Max. value high alarm DN_AL : Min. value high alarm

■ **Functions**

(1) It receives input X, applies directional deadband (hysterisis) to it and notifies an operator of UP/DOWN state.
(2) In case of UP_in < X, UP_AL is on.
(3) In case of UP_out ≤ X ≤ UP_in, it maintains the previous UP_AL state.
(4) In case of X < UP_out, UP_AL is off.
(5) In case of X < DN_in, DN_AL is on.
(6) In case of DN_in ≤ X ≤ DN_out, it maintains the previous DN_AL state.
(7) In case of DN_out < X, DN_AL is off.
(8) In case UP_in value is not more than UP_out value, it outputs 8 to STAT.
(9) In case DN_out value is not more than DN_in value, it outputs 8 to STAT.

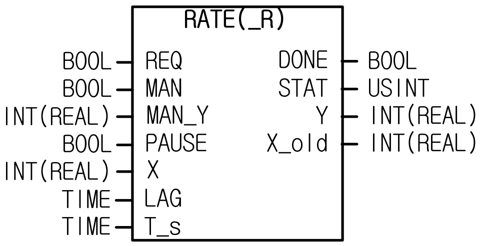


■ **Program Example**



(1) If X is changed from 0 to 800: UP_AL on, DN_AL off
(2) If X is changed from 800 to 650: UP_AL on, DN_AL off
(3) If X is changed from 650 to 300: UP_AL off, DN_AL off
(4) If X is changed from 300 to 50: UP_AL off, DN_AL on
(5) If X is changed from 50 to 150: UP_AL off, DN_AL on

RATE(_R)	Measuring Variation Per Section	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request MAN : Converting to Manual mode MAN_Y : Manual output value PAUSE : Pause X : Input LAG : LAG filter coefficient T_s : Operation cycle **Output** DONE : On if done without error STAT : State alarm Y : Output value X_old : Previous X

■ **Functions**

(1) RATE function is the instruction indicating the variation per second of input X.
(2) If MAN bit is on, it outputs MAN_Y.
(3) If PAUSE bit is on, the block pauses.
(4) If setting time constant in LAG, it processes it with low pass filter of input.
(5) The I/O equation of RATE instruction including LAG is as follows.

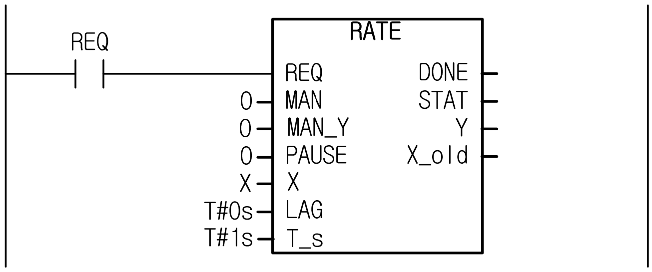
$$Y = Y_{old} + \frac{T\_s}{LAG + T\_s} \times \left(\frac{X + X_{old}}{Ts} - Y_{old}\right) \qquad [T\_s : sec]$$

(6) The above equation may be summarized as follows if LAG is 0.

$$Y = \frac{X - X_{old}}{Ts} \qquad [T\_s : sec]$$

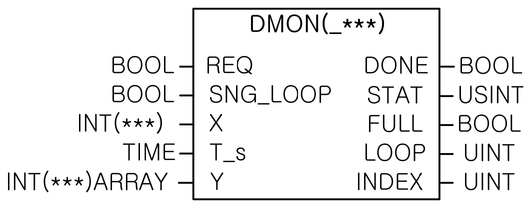
(7) If the operation result is out of the data expression range of integer (INT), the output is limited to INT (-32768 ~ 32767).
(8) If the operation result is out of the data expression range of real number (REAL), the output displays as '1.#inf00000 E+000' or '-1.#inf00000 E+000'. In the case, DONE bit is off but the internal state (i.e. X_old) is normally processed.

■ **Program Example**



(1) If X increases by 1 per second, Y outputs 1
(2) If X increases from 10 by 1 per second, Y outputs 1
(3) If X decreases from 10 by 30 per second, Y outputs -30

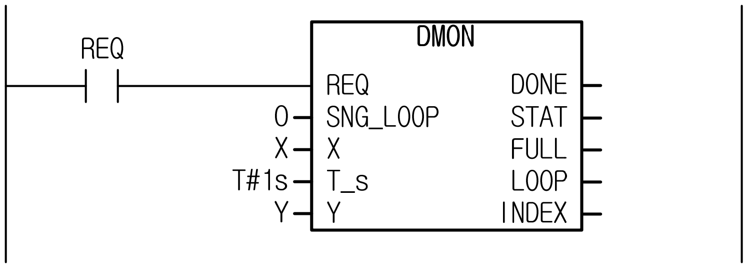
DMON(_***)	Saving input array as much as output array	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request SNG/LOOP : Single/Loop operation X : Input T_s : Operation cycle Y : Output value **Output** DONE : On if done without error STAT : State alarm FULL : Output array full LOOP : No. of full output array INDEX : Array No. of location to save

■ **Functions**

(1) It is used to save the data that are changing temporally.
(2) It saves input X to Y (Array) every operation cycle (T_s).
(3) DMON function block is INT type instruction; the data type started with DMON such as _DI (DINT), _R (REAL), _UI (UINT), _UDI (UDINT), _W (WORD) and _DW (DWORD) may be used selectively, depending on I/O data.
(4) If SNG_LOOP is off, it is engaged in single operation, saves as much as no. of array and stops with FULL on.
(5) If SNG_LOOP is on, it is engaged in loop operation, saves as much as no. of array and continues to rewrite the original values from the first.
(6) If SNG_LOOP is converted to single/loop, it is necessary to allow REQ again and initialize it prior to use.
(7) During loop operation, LOOP increases ever time array is full. If LOOP value is over 65535, it is reset to 0.

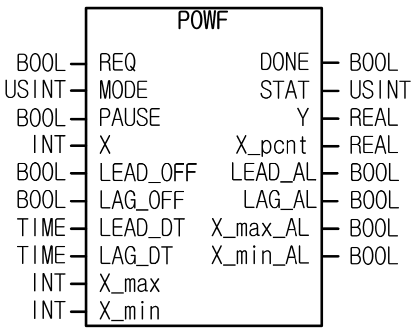
■ **Program Example**



Y is set to ARRAY [0..10] of INT type.
(1) X increases from 0 by 1 per second Y[0]=0 … a value is saved in good order of Y[10]=10 and it results in FULL=on from 12s.
(2) X increases from 10 by 1 per second : a value is saved per second in good order of Y[0]=10 … Y[10]=20 and it results in FULL=on from 12s.
(3) X decreases from 10 by 3 per seconds : a value is saved per second in good order of Y[0]=10 … Y[10]=-20 and it results in FULL=on from 12s.

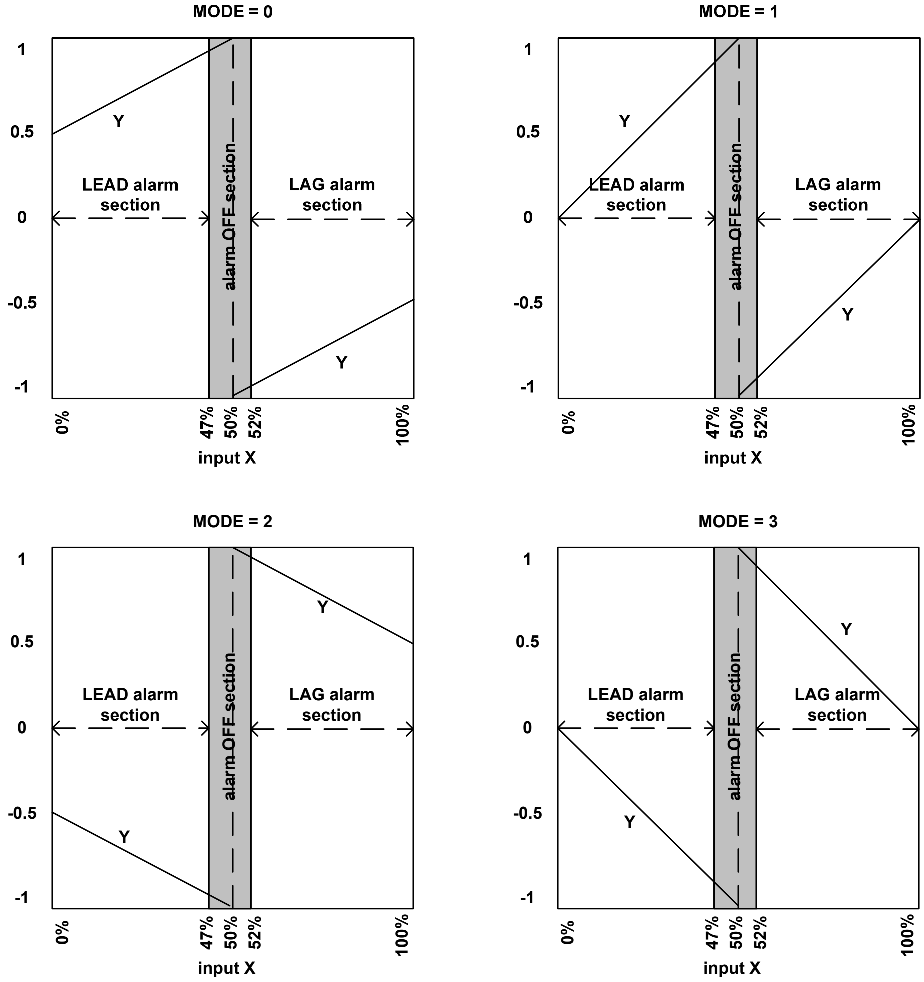
13.6 Data Function Block, Function Block

POWF	PF Instrument	
	Availability	XGI, XGR, XEC(U)
	Flags	-

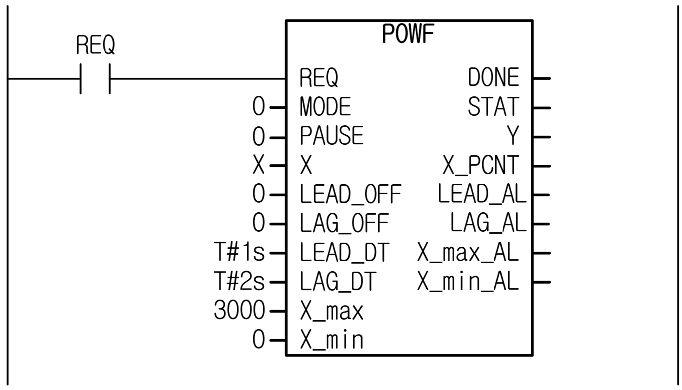
Function block	Description
	**Input** REQ : Function block execution request MODE : Mode conversion PAUSE : Pause X : Input LEAD_OFF : Lead alarm off LAG_OFF : Lag alarm off LEAD_DT : Lead alarm ON delay time LAG_DT : Lag alarm ON delay time X_max : Max. input limit X_min : Min. input limit **Output** DONE : On if done without error STAT : State alarm Y : Output value X_pcnt : Percent output LEAD_AL : Lead alarm LAG_AL : Lag alarm X_max_AL : Max. value high alarm X_min_AL : Min. value low alarm

■ **Functions**

(1) By referring to the input X receiving from PF sensor, it generates output Y along the PF profile.
(2) The max./min. value of input X is limited by setting X_max and X_min.
(3) Input X is converted to the unit of % by setting X_max and X_min, indicated in X_PCNT and executes operation with %.
(4) Profile type is selected depending on mode (0 ~ 3 selectable). The outputs by modes are as presented in the figure below.
  a) MODE 0 : inclination 0.5, lead offset 1 and lag offset -1.
  b) MODE 1 : inclination 1, lead offset 1 and lag offset -1.
  c) MODE 2 : inclination -0.5, lead offset -1 and lag offset 1.
  d) MODE 3 : inclination -1, lead offset -1 and lag offset 1.
(5) At a point where X is 50%(center of the graph), output Y is defined as 0.
(6) If PAUSE is on, operation stops and it does not indicate alarm bit until operation resumes.
(7) It indicates lead and lag in LEAD_AL and LAG_AL and it is possible to set indication (_OFF) and delay time (_DT).
(8) It is possible to set the max./min. value of input X in X_max and X_min.
(9) When MODE is more than 3, it outputs 8 to STAT.
(10) In case of X_max = X_min, it does not operate because the denominator is 0 and STAT indicates 8.
(11) Input-output may have an error less than 0.001%.

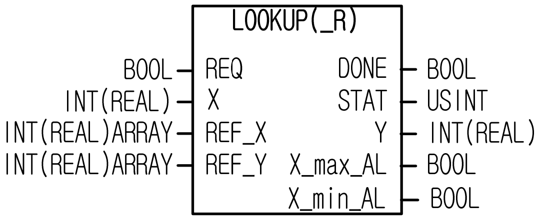


■ **Program Example**



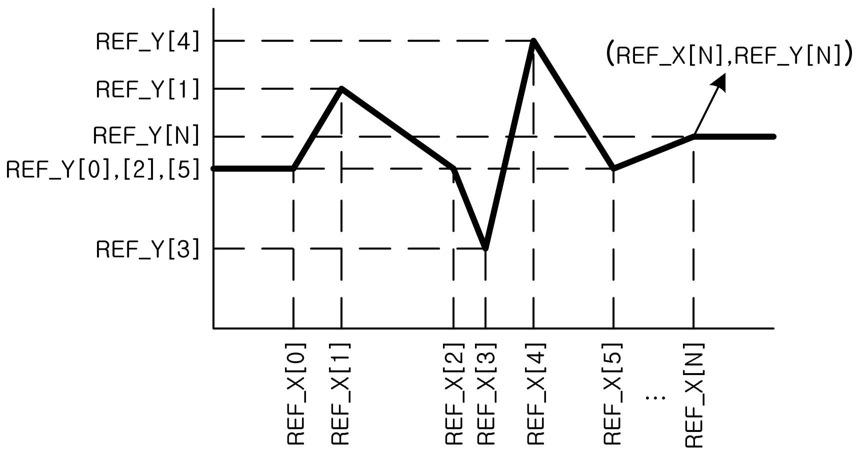
(1) If X is 0 : X_PCNT = 0 and Y = 0.5, in 1 second, LEAD_AL = on, LAG_AL = off
(2) If X is 1500 : X_PCNT = 50 and Y = 0, LEAD_AL = off, LAG_AL = off
(3) If X is 2000 : X_PCNT = 66 and Y = -0.84, LEAD_AL = off, in 2 seconds LAG_AL = on

LOOKUP(_R)	LOOK-UP Table output	
	Availability	XGI, XGR, XEC(U)
	Flags	-

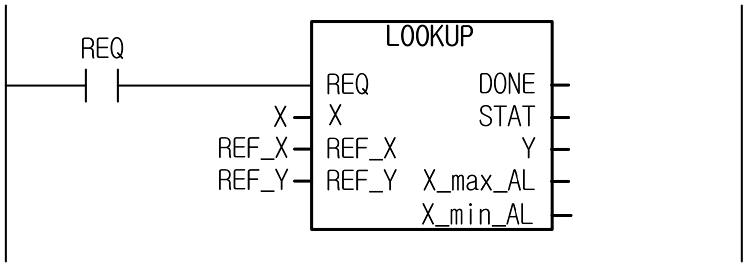
Function block	Description
 	**Input** REQ : Function block execution request X : Input REF_X : X coordinate array of LOOK-UP table REF_Y : Y coordinate array of LOOK-UP table **Output** DONE : On if done without error STAT : State alarm Y : Output value X_max_AL : REF_X high alarm X_min_AL : REF_X low alarm

■ **Functions**

(1) By using input array (REF_X) and output array (REF_Y), it creates LOOK-UP table by sections and gets output by applying input X.
(2) Input array REF_X should be arranged in ascending order, and if the elements of array are same, it generates alarm.
(3) If the value inputted through input X is same or out of the range of input array (REF_X), it indicates X_max_AL and X_min_AL.
(4) If the elements of REF_X are not arranged in ascending order, STAT outputs 8.
(5) If the no. of REF_X and REF_Y arrays are different, STAT outputs 8.
(6) If operation result is out of the data expression range of integer (INT), the output is limited to INT (-32768 ~ 32767).
(7) If operation result is out of the data expression range of real number (REAL), it is indicated as '1.#inf00000 E+000' or '-1.#inf00000 E+000', and in the case, DONE bit is off but the internal state (i.e. X_max_AL, X_min_AL) is normally processed.



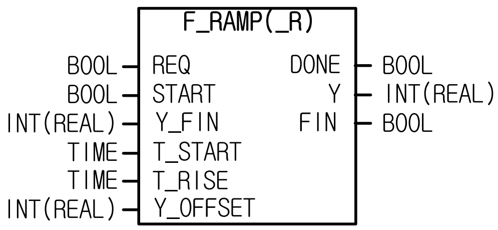
■ **Program Example**



It sets REF_X as ARRAY [0..4] of INT and also sets the element of array as [10, 20, 30, 40, 50].
It sets REF_Y as ARRAY [0..4] of INT and also sets the elements of array as [10, 20, 10, 50, 20].

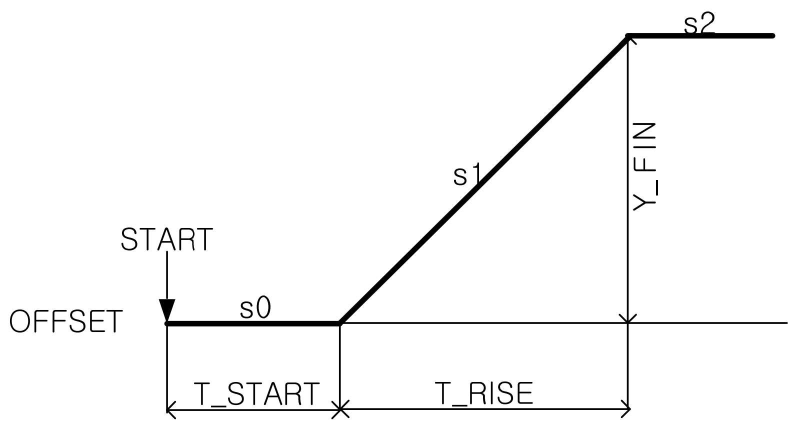
(1) If X is 5: Y = 10, X_min_AL = on, X_max_AL = off
(2) If X is 15 : Y = 15, X_min_AL = off, X_max_AL = off
(3) If X is 45 : Y = 35, X_min_AL = off, X_max_AL = off
(4) If X is 100 : Y = 20, X_min_AL = off, X_max_AL = on

F_RAMP(_R)	Singular RAMP Function output	
	Availability	XGI, XGR, XEC(U)
	Flags	-

Function block	Description
	**Input** REQ : Function block execution request START : Operation start Y_FIN : RAMP function target value T_START : Operation waiting time T_RISE : Total rise section Y_OFFSET : Output offset **Output** DONE : On if done without error Y : Output FIN : Normal state alarm

■ **Functions**

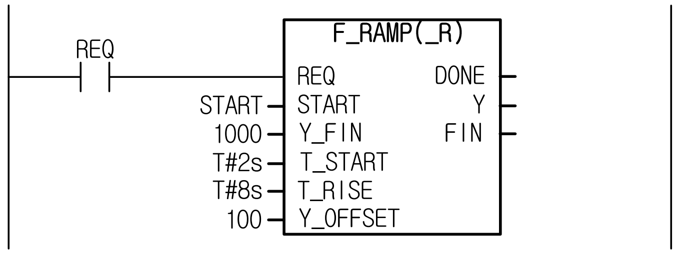
(1) It outputs RAMP function.
(2) In case of START on, it starts waveform output.
(3) If REQ is off, it maintains the value of last state in an operation.
(4) If START is off with REQ on, it initializes with its initial value and waits for operation start (START on).
(5) it sets RAMP function target value in Y_FIN, waiting time after start in T_START, waveform rise time in T_RISE and offset in Y_OFFSET.
(6) If waveform rise is finished, FIN is on.
(7) F_RAMP: if Y_FIN + Y_OFFSET is out of the data expression range of Y (INT), it is limited to $-32768 \le Y \le 32767$.
(8) F_RAMP_R: if Y_FIN + Y_OFFSET is out of the data expression range of Y (REAL), the result is indicated as '1.#inf00000 E+000' or '-1.#inf00000 E+000' during operation and in the case, DONE bit is off but the internal state(that is, FIN) is normally processed.



The equation of each section is as follows.

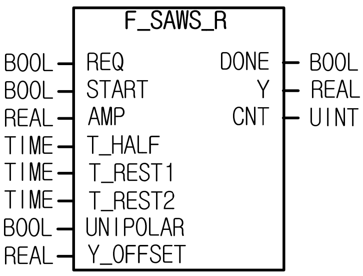
s0 : Y = Y_OFFSET
s1 : Y = Y_FIN * (t – T_START) / T_RISE + Y_OFFSET
s2 : Y = Y_FIN + Y_OFFSET
(where, t is the time passed after START)

■ **Program Example**



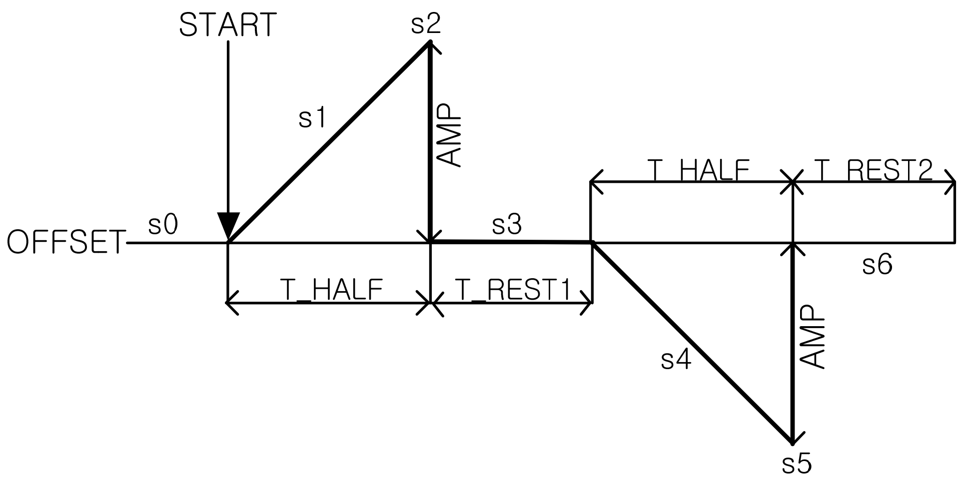
If setting START on with the above setting, it is possible to get a waveform increasing from 100 to 1000 in 2s.

F_SAWS_R	SAW Tooth Wave Output	
	Availability	XGI, XGR, XEC(U)
	Flags	-

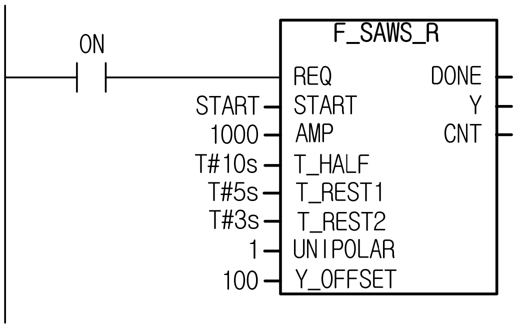
Function block	Description
	**Input** REQ : Function block execution request START : Operation start AMP : SAWS function target value T_HALF : Function half cycle T_REST1 : Waveform waiting time 1 T_REST2 : Waveform waiting time 2 UNIPOLAR : Unipolar function output Y_OFFSET : Output offset **Output** DONE : On if done without error Y : Output value CNT : Output repeat frequency

■ **Functions**

(1) It outputs saw tooth wave.
(2) In case of START on, it starts waveform output.
(3) If REQ is off, it maintains the value of last state in an operation.
(4) If START is off with REQ on, it initializes with its initial value and waits for operation start (START on).
(5) It sets amplitude of SAWS function in AMP, rise time of saw tooth wave in T_HALF and offset in Y_OFFSET.
(6) If UNIPOLAR is on, it outputs unipolar function; in case of off, it outputs bipolar function.
(7) Function's output count CNT increases once a cycle output ends; if it is over 65535, the range of UINT, it increases from 0 again.
(8) In case it skips a scan (if scan is longer than 1msec), scan may have an error at S2 and S5, the max./min. values; an error is larger because as smaller H_HALF value as larger the inclination of a graph.
(9) F_SAWS: if Y_FIN + Y_OFFSET is out of the data expression range of Y (INT), it is limited to -32768 ≤ Y ≤ 32767.
(10) F_SAWS_R: if Y_FIN + Y_OFFSET is out of the data expression range of Y (REAL), it is indicates as '1.#inf00000 E+000' or '-1.#inf00000 E+000'during operation and in the case, DONE bit is off but the internal state (that is, CNT) is normally processed.

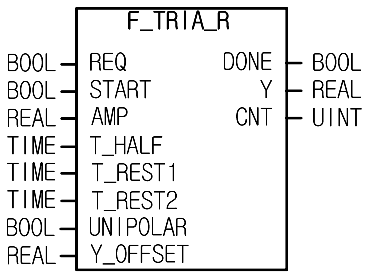

START
s2
s1
AMP
T_HALF
T_REST2
OFFSET
s0
s3
s6
T_HALF
T_REST1
s4
AMP
s5

■ **Program Example**



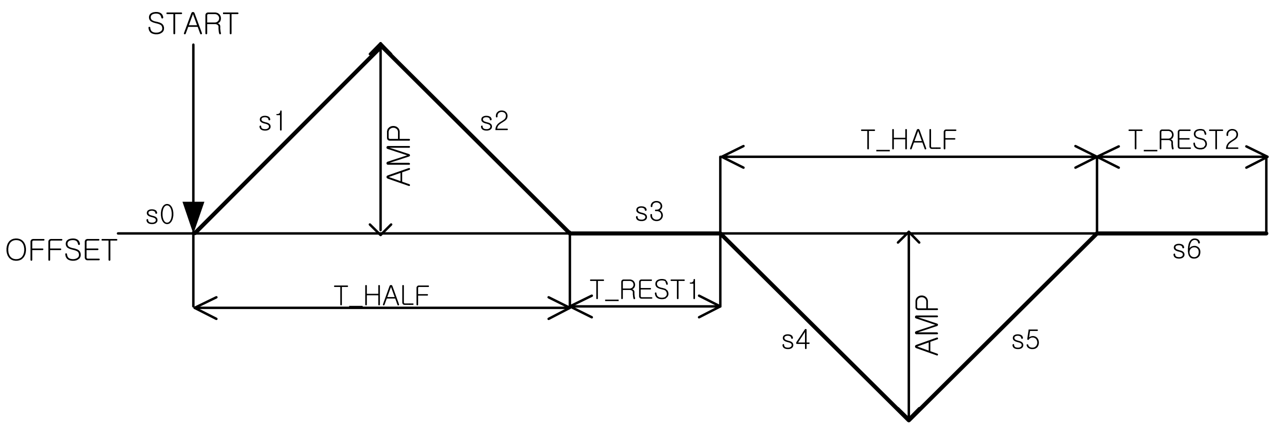
In case of START on in the above setting, it outputs the waveform.

F_TRIA_R	Triangular wave output	
	Availability	XGI, XGR, XEC(U)
	Flags	-

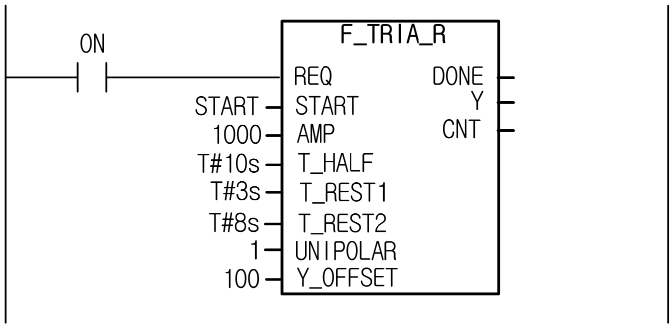
Function block	Description
	**Input** REQ : Function block execution request START : Operation start AMP : TRIA function target value T_HALF : Function half cycle T_REST1 : Waveform waiting time 1 T_REST2 : Waveform waiting time 2 UNIPOLAR : Unipolar function output Y_OFFSET : Output offset **Output** DONE : On if done without error Y : Output value CNT : Output repeat frequency

■ **Functions**

(1) It outputs triangular wave.
(2) In case of START on, it starts waveform output.
(3) If REQ is off, it maintains the value of last state in an operation.
(4) If START is off with REQ on, it initializes with its initial value and waits for operation start (START on).
(5) It sets amplitude of TRIA function in AMP, triangular rise time in T_HALF and offset in Y_OFFSET.
(6) If UNIPOLAR is on, it outputs unipolar function; in case of off, it outputs bipolar function.
(7) Function's output count CNT increases once a cycle output ends; if it is over 65535, the range of UINT, it increases from 0 again.
(8) In case it skips a scan (if scan is longer than 1m), scan may have an error at S2 and S5, the max./min. values; an error is larger because as smaller H_HALF value as larger the inclination of a graph.
(9) F_TRIA: if Y_FIN + Y_OFFSET is out of the data expression range of Y (INT), it is limited to -32768 ≤ Y ≤ 32767.
(10) F_TRIA_R: if Y_FIN + Y_OFFSET is out of the data expression range of Y (REAL), it is indicates as '1.#inf00000 E+000' or '-1.#inf00000 E+000'during operation and in the case, DONE bit is off but the internal state (that is, CNT) is normally processed.

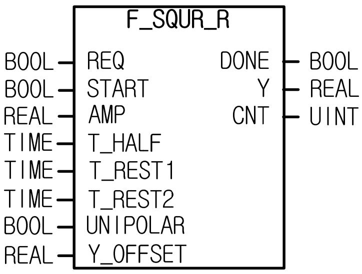


■ **Program Example**



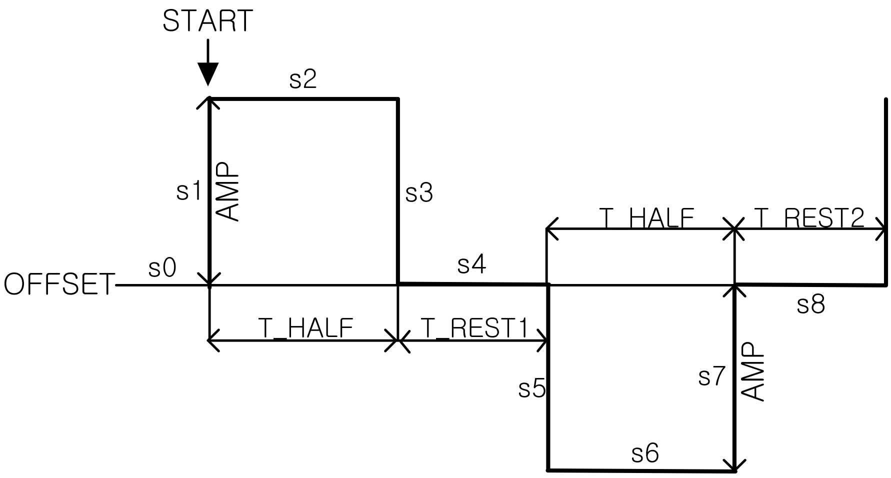
In case of START on in the above setting, it outputs the waveform.

F_SQUR_R	Square wave output	
	Availability	XGI, XGR, XEC(U)
	Flags	-

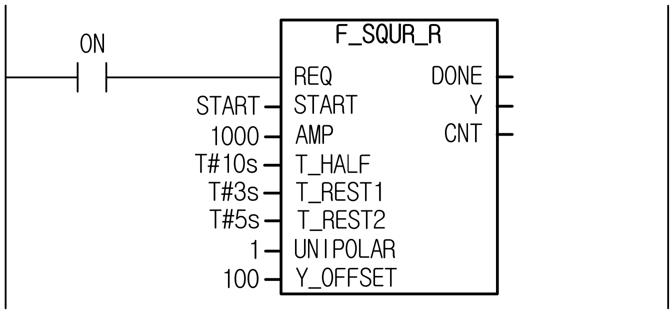
Function block	Description
	**Input** REQ : Function block execution request START : Operation start AMP : SQUR function target value T_HALF : Function half cycle T_REST1 : Waveform waiting time 1 T_REST1 : Waveform waiting time 2 UNIPOLAR : Unipolar function output Y_OFFSET : Output offset **Output** DONE : On if done without error Y : Output value CNT : Output repeat frequency

■ **Functions**

(1) It outputs square waveform.
(2) In case of START on, it starts waveform output.
(3) If REQ is off, it maintains the value of last state in an operation.
(4) If START is off with REQ on, it initializes with its initial value and waits for operation start (START on).
(5) It sets amplitude of SQUR function in AMP, rise half cycle of square wave in T_HALF and offset in Y_OFFSET.
(6) If UNIPOLAR is on, it outputs unipolar function; in case of off, it outputs bipolar function.
(7) Function's output count CNT increases once a cycle output ends; if it is over 65535, the range of UINT, it increases from 0 again.
(8) F_SQUR: if Y_FIN + Y_OFFSET is out of the data expression range of Y (INT), it is limited to -32768 ≤ Y ≤ 32767.
(9) F_SQUR_R: if Y_FIN + Y_OFFSET is out of the data expression range of Y (REAL), it is indicates as '1.#inf00000 E+000' or '-1.#inf00000 E+000'during operation and in the case, DONE bit is off but the internal state (that is, CNT) is normally processed.

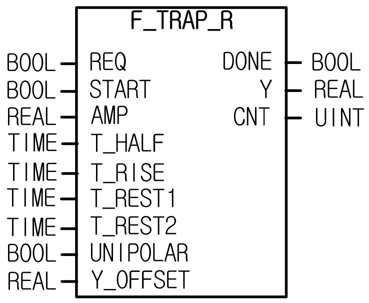


■ **Program Example**



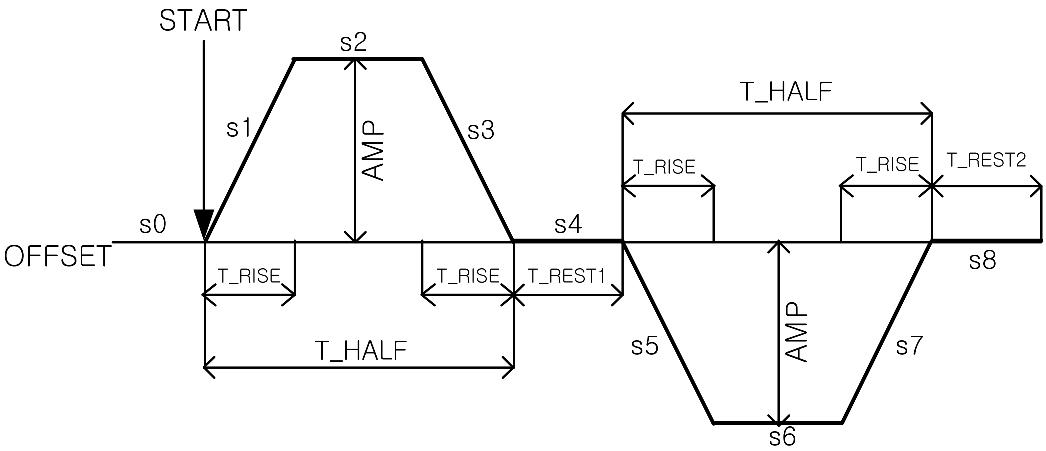
In case of START on in the above setting, it outputs the waveform.

F_TRAP_R	Trapezoid wave output	
	Availability	XGI, XGR, XEC(U)
	Flags	-

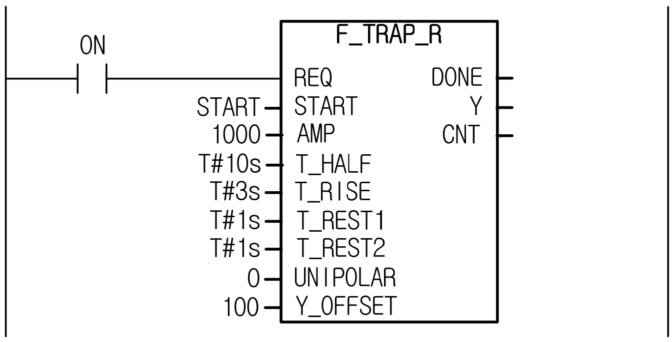
Function block	Description
	**Input** REQ : Function block execution request START : Operation start AMP : TRAP function target value T_HALF : Function half cycle T_RISE : Trapezoid output time T_REST1 : Waveform waiting time 1 T_REST1 : Waveform waiting time 2 UNIPOLAR : Unipolar function output Y_OFFSET : Output offset **Output** DONE : On if done without error Y : Output value CNT : Output repeat frequency

■ **Functions**

(1) It outputs trapezoid wave.
(2) In case of START on, it starts waveform output.
(3) If REQ is off, it maintains the value of last state in an operation.
(4) If START is off with REQ on, it initializes with its initial value and waits for operation start (START on).
(5) It sets amplitude of TRAP function in AMP, trapezoid output time in T_RISE, half cycle of waveform in T_HALF and offset in Y_OFFSET.
(6) If UNIPOLAR is on, it outputs unipolar function; in case of off, it outputs bipolar function.
(7) Function's output count CNT increases once a cycle output ends; if it is over 65535, the range of UINT, it increases from 0 again.
(8) If T_RISE is more than half of T_HALF, it outputs triangular wave and the output of AMP scale is not secured.
(9) F_TRAP: if Y_FIN + Y_OFFSET is out of the data expression range of Y (INT), it is limited to -32768 ≤ Y ≤ 32767.
(10) F_TRAP_R: if Y_FIN + Y_OFFSET is out of the data expression range of Y (REAL), it is indicates as '1.#inf00000 E+000' or '-1.#inf00000 E+000'during operation and in the case, DONE bit is off but the internal state (that is, CNT) is normally processed.

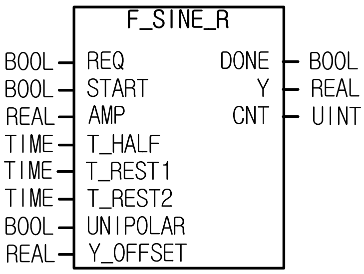


■ **Program Example**



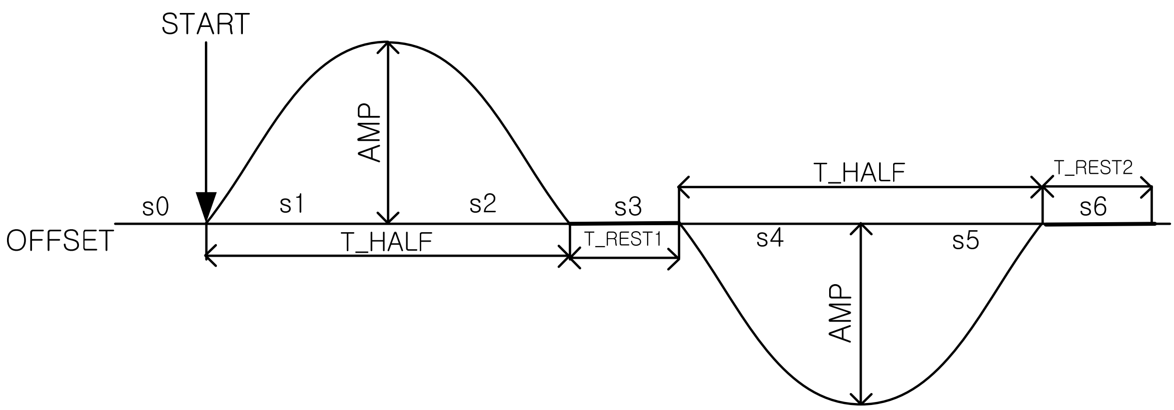
In case of START on in the above setting, it outputs the waveform.

F_SINE_R	Sine wave output	
	Availability	XGI, XGR, XEC(U)
	Flags	_LER

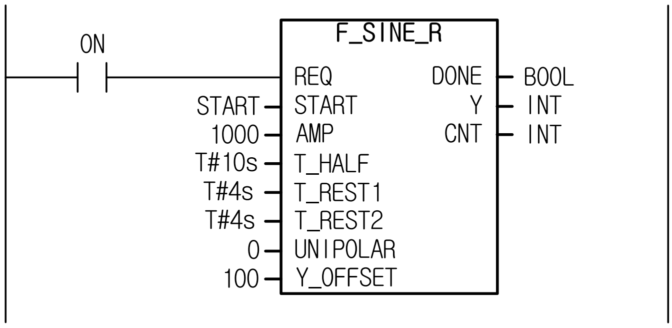
Function block	Description
	**Input** REQ : Function block execution request START : Operation start AMP : SINE function target value T_HALF : Function half cycle T_REST1 : Waveform waiting time 1 T_REST2 : Waveform waiting time 2 UNIPOLAR: Unipolar function output Y_OFFSET: Output offset **Output** DONE : On if done without error Y : Output value CNT : Output repeat frequency

■ **Functions**

(1) It outputs sine wave.
(2) In case of START on, it starts waveform output.
(3) If REQ is off, it maintains the value of last state in an operation.
(4) If START is off with REQ on, it initializes with its initial value and waits for operation start (START on).
(5) It sets amplitude of SINE function in AMP, half cycle of sine wave in T_HALF and offset in Y_OFFSET.
(6) If UNIPOLAR is on, it outputs unipolar function; in case of off, it outputs bipolar function.
(7) Function's output count CNT increases once a cycle output ends; if it is over 65535, the range of UINT, it increases from 0 again.
(8) In case it skips a scan (if scan is longer than 1m), scan may have an error at S2 and S5, the max./min. values; an error is larger because as smaller H_HALF value as larger the inclination of a graph.
(9) F_SINE: if Y_FIN + Y_OFFSET is out of the data expression range of Y (INT), it is limited to -32768 ≤ Y ≤ 32767.
(10) F_SINE_R: if Y_FIN + Y_OFFSET is out of the data expression range of Y (REAL), it is indicates as '1.#inf00000 E+000' or '-1.#inf00000 E+000'during operation and _LER flag is set. In the case, DONE bit is off but the internal state (that is, CNT) is normally processed.

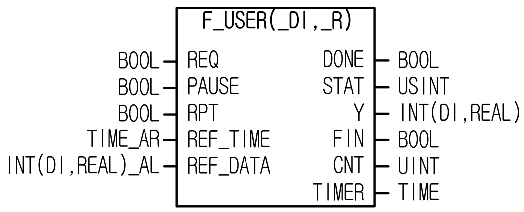


■ **Program Example**



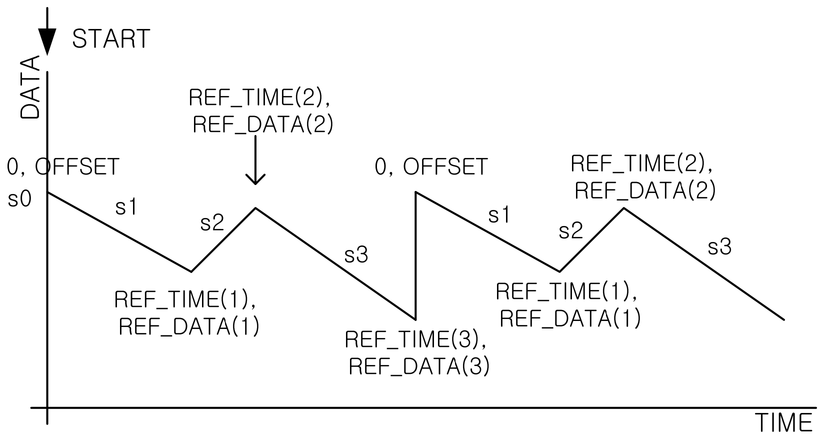
In case of START on in the above setting, it outputs the waveform.

F_USER(_DI, _R)	User-defined wave output	
	Availability	XGI, XGR, XEC(U)
	Flags	-

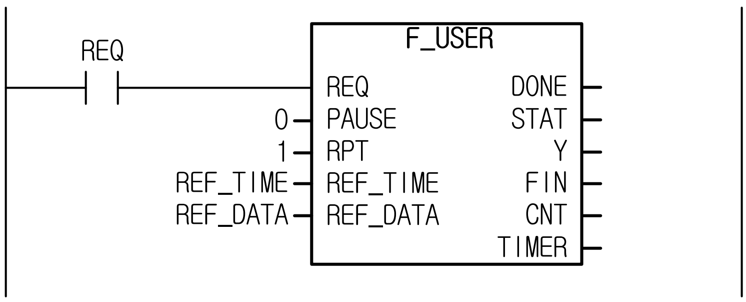
Function block	Description
	**Input** REQ : Function block execution request PAUSE : Pause RPT : Repeat REF_TIME : Time array REF_DATA : Data array **Output** DONE : On if done without error STAT : State alarm Y : Output value FIN : Output complete (if not repetitive) CNT : Repeat frequency TIMER : Timer value within FB

■ **Functions**

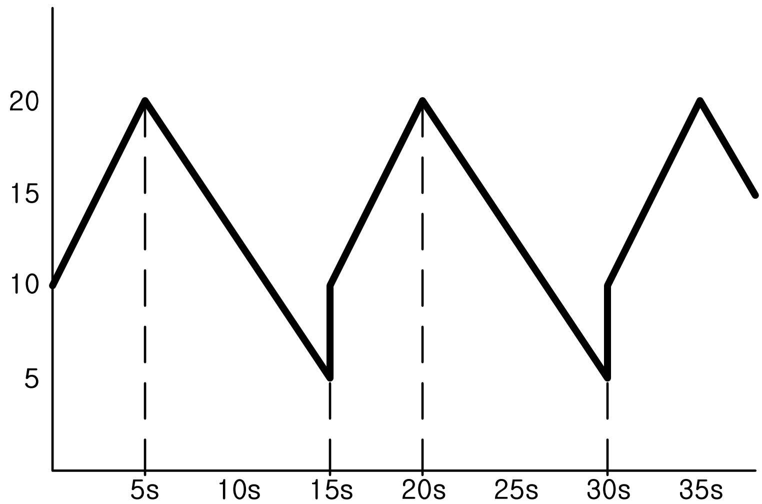
(1) It outputs user-defined waveform.
(2) If REQ is off, it maintains the value of last state in an operation.
(3) If the data of initial state (0 second) is not defined, it is regarded as the first value of REF_DATA. That is, if it is defined as the first data (2 seconds, 3000), it outputs 3000 for 2 seconds just after wave start.
(4) Output pauses if PAUSE bit is on. However, the initialize output with REQ on is not limited by PAUSE.
(5) If RPT bit is on, the wave is repetitively output.
(6) A user defines the wave by using REF_TIME and REF_DATA.
(7) In case of singular (RPT = off), FIN is on after output is complete and TIMER indicates the progress time.
(8) In case of repetitive (RPT = on), it outputs repetitively from the first after output is complete. CNT indicates function output count while timer displays the progress time of the cycle.
(9) The output count CNT of repetitive function increases if a cycle of output ends; if it is over 65535, the range of UINT, it increases from 0 again.
(10) As soon as a waveform ends, RPT is checked; if RPT is on, it is regarded as repetitive function, and in case of off, it is regarded as singular function. Even in case of repetitive waveform, it is regarded as singular function if RPT is off when the waveform ends.
(11) If waveform output ends in singular function, FIN is on and waveform output does not resume even though RPT is changed. It may be initialized after REQ is off.
(12) In case the elements of REF_TIME are not arranged in ascending order, STAT outputs 8.
(13) In case the number of REF_TIME and REF_DATA are different, STAT outputs 8.



■ **Program Example**



It sets REF_TIME as ARRAY [0..2] of INT and also sets the element of array as [T#0s, T#5s, T#15s].
It sets REF_DATA as ARRAY [0..2] of INT and also sets the element of array as [10, 20, 5].
If you executes the above, the following waveform is outputted when REQ is allowed in the following block.



Chapter 14. ST (Structured Text)

14.1 General

- ▷ ST program can use all of text editor and has high portability.
- ▷ It can express complicated expression and algorithm well
- ▷ A person skilled at computer language can use easily.

```
//FUNCTION
 CMD_TMR(IN:=%IX5.0.0, PT:=T#300ms) ;
 bb := CMD_TMR.Q ;

// IF
A := 1.0;
B := 1.000e+3;
C := 2.0;
D := B*B - 4*A*C ;
IF D < 0.0 THEN NROOTS := 0 ;
ELSIF D = 0.0 THEN
    NROOTS := 1 ;
    X1 := - B/(2.0*A) ;
ELSE
    NROOTS := 2 ;
    X1 := (- B + SQRT(D))/(2.0*A) ;
    X2 := (- B - SQRT(D))/(2.0*A) ;
END_IF ;
```

14.2 Comments

There are two types in comments, one line comment and block comment.

- One line comment uses “//”, that line is used as comment line.
- Block comment considers text between “*” and “*”.

For example)

```
//one line comment
(*Block
comment
*)
```

14.3 Expression

1) Expression always has result value.
2) Expression consists of operator and operand. Operand may be constant, character, character string, time character, defined variable (named variable, direct variable), defined function (function, function block). Operator of ST is described in the follow table. And also expression is calculated according to order of operator of ST language table.
3) Among same operations which have same order, operation in left of expression has higher order.
   For example: A+B-C: first, adds A to B and subtracts C from result of A+B.
   If operator has two operands, left operand executes first.
   For example, SIN(A)*COS(B): SIN(A) executes first and COS(B) executes last.
4) When executing operation, the following condition is dealt with error.
   - Division by 0
   - Operand is not applicable data type for operation.
     For example, ADD(1,2,3): unable to determine the data type of number so compile error occurs
   - Result of arithmetic operation exceeds range of data type.
     For example, B*C: When B, C are UINT type, result is higher than 65,535, operation error occurs.

Number	Operation	Symbol	Order
1	Parenthesis	(Expression)	High
2	Function	Function name (Parameter list) Ex.) ADD(X, Y)	
3	not Complement	- NOT	
4	Exponent	**	
5	Multiplication Division Remain	* / MOD	
6	Add Subtract	+ -	
7	Compare	<, >, <=, >=	
8	same Not same	= <>	
9	Bool logical AND	& AND	
10	Bool logical Exclusive OR	XOR	
11	Bool logical OR	OR	Low

<Table 1> Operator of ST language

5) Bool type expression is calculated until determining the result value.

6) Function is recalled as an expression factor which has function name and parenthesis including parameter. When function is used in the expression, operand and conversion of result follows as in the following table.

Method	Characteristic			OUT := LIMIT(MIN, IN, MX);
	Variable Assignment	Variable Order	No. of Variable	
Fixed type	Available	Changeable	Changeable	Function Ex. A:= LIMIT(IN:= B, MX:= 5, MIN:= 1); Function block Ex. INST_TOF (BOOL_IN,TIME_PT, BOOL_Q,TIME_ET)
Non-fixed type	Unavailable	Fixed	Fixed	Function Ex A:= LIMIT(1, B, 5); Function block Ex. INST_TOF (BOOL_IN,TIME_PT, BOOL_Q,TIME_ET)

- EN, ENO parameter cannot be used.
- VAR_IN_OUT can be used one time.
- Function block uses instant name. Ex: **INST_TON1**(IN := TRUE, PT := T#100MS, Q =>Q_OUT, ET => ET_OUT).
- In fixed type, in case, inner parameter is VAR_INPUT, VAR_IN_OUT, ‘:=’ is used.
- In fixed type, in case, inner parameter is VAR_OUTPUT, ‘=>’ is used.

14.3.1 + Operator

1) + Operator is used to add two operands.
2) Expression

result* := *expression1* + *expression2

Items	Description
Result	Named variable or direct variable
expression1	ANY_NUM type
expression2	ANY_NUM type

Example	Description
Val1 := 20; Val2 := 4; Result := Val1 + Val2;	Adds Val1(20) to Val2(4) and inputs result Value of Result becomes 24. Constant and variable can be used as operands (Val1, Val2).

Note

ANY_NUM includes ANY_REAL type and ANY_INT. For more detail, refer to data type layer of ch.3.2.2

14.3.2 - Operator

1) Subtracts right value from left value.
2) Expression

result* := *expression1* - *expression2

Items	Description
result	Named variable or direct variable
expression1	ANY_NUM
expression2	ANY_NUM

Example	Description
Val1 := 20; Val2 := 4; Result := Val1 - Val2;	Subtracts right value(Val2) from left value(Val1) and inputs result. Value of result becomes 16 Constant and variable can be used as operands (Val1, Val2).

14.3.3 * Operator

1) Multiplies two operands
2) Expression

result* := *expression1* * *expression2

Items	Description
result	Named variable or direct variable
expression1	ANY_NUM type
expression2	ANY_NUM type

Example	Description
In1 := 2 ; Result := 20 * In1 ;	Multiplies 20 by In1(2) and inputs result. Value of result becomes 40. Constant and variable can be used as operands (Val1, Val2).

14.3.4 / Operator

1) Divides left value by right value.
2) Data type of result is different according to data type of operand. If operand is REAL type, result is also REAL type. If operand is integer, result is also integer. If 5 (int) is divided by 3 (int), result is real but number less than decimal point is removed.

```
Result := 20 / INT_TYPE ;

Result1 := 20 / REAL_TYPE ;
```

```
Result = 6, INT_TYPE = 3

Result1 = 6.666666508e+000, REAL_TYPE = 3.000000000e+000
```

3) Expression

result* := *expression1* / *expression2

Item	Description
result	Named variable or direct variable
expression1	ANY_NUM type
expression2	ANY_NUM type

Example	Description
In1 := 2 ; Result := 20 / In1 ;	Divides 20 by 2(In1) and inputs result. Result becomes 10. Constant and variable can be used as operands.

Notes

If some value is divided by 0, operation error flag (_ERR) is on and CPU is in RUN mode.

14.3.5 MOD operation

1) Finds remain when dividing left value by right value
2) Expression

result* := *expression1 MOD expression2

Item	Description
result	Named variable or direct variable
expression1	ANY_NUM type
expression2	ANY_NUM type

Example	Description
In1 := 10 ; Result := 12 MOD In1 ;	Divides 12 by 10(In1) and inputs remain into result Constant and variable can be used as operands.

Notes

If some value is divided by 0, operation error flag (_ERR) is on and CPU is in RUN mode.

14.3.6 ** Operator

1) Exponential operator is used to multiply left number as many as right number times
2) Expression

result* := *expression1* ** *expression2

Items	Description
result	Named variable or direct variable
expression1	ANY_REAL type
expression2	ANY_REAL type

Example	Description
In1 := 3 ; Result := 10 ** In1 ;	Multiplies 10 as many as 3 times and inputs it to result. Result becomes 1000. Constant and variable can be used as operands.

14.3.7 AND or & Operator

1) Executes logical bit AND operation.
2) Expression

result* := *expression1 AND expression2 **or** ***result* := *expression1 & expression2***

Item	Description
result	Named variable or direct variable
expression1	ANY_BIT type
expression2	ANY_BIT type

Result of logical bit AND operation is as follows.

expression1	*expression2*	*result*
0	0	0
0	1	0
1	0	0
1	1	1

Example	Description
Result := 2#10010011 AND 2#00111101 ;	Since first bit and 5th bit of two operands are both 1, result is 2#00010001. Constant and variable can be used as operands.

14.3.8 OR operator

1) Executes logical bit OR operation.
2) Expression

result* := *expression1 OR expression2

Items	Description
result	Named variable or direct variable
expression1	ANY_BIT type
expression2	ANY_BIT type

Result of logical bit OR operation is as follows.

expression1	*expression2*	*result*
0	0	0
0	1	1
1	0	1
1	1	1

Example	Description
Result := 2#1**0**010011 OR 2#0**0**111101 ;	Since there are 1 except 7th bit in two operands, result is 2#10111111.

14.3.9 XOR operator

1) If bits of two operands are different, result bit is 1.
2) Expression

result* := *expression1 XOR expression2

Item	Description
result	Named variable or direct variable
expression1	ANY_BIT type
expression2	ANY_BIT type

Result of logical bit XOR operation is as follows.

expression1	*expression2*	*result*
0	0	0
0	1	1
1	0	1
1	1	0

Example	Description
Result := 2#10010011 XOR 2#00111101;	Since first bits of two operands are 1, first bit of result is 0. Result is 2#10101110.

14.3.10 Operator

1) Compares two operands if they are same.
2) Expression

result* := *expression1* = *expression2

Item	Description
result	Named variable or direct variable
expression1	ANY type
expression2	ANY type

Result of logical bit = operation is as follows.

expression1	*expression2*	*result*
0	0	1
0	1	0
1	0	0
1	1	1

Example	Description
Val1 := 20; Val2 := 20 ; Result := Val1 = Val2 ;	Compares Val1 and Val2 and output result. Result is 1.

14.3.11 <> operator

1) Compares two operands if they are not same.
2) Expression

result* := *expression1* <> *expression2

Item	Description
result	Named variable or direct variable
expression1	ANY type
expression2	ANY type

Result of logical bit <> operation is as follows.

expression1	*expression2*	*result*
0	0	0
0	1	1
1	0	1
1	1	0

Example	Description
Val1 := 20; Val2 := 20 ; Result := Val1 <> Val2 ;	Compares Val1 and Val2 and output result. Result is 0.

14.3.12 > operator

1) Compares two operands if left one is larger than right one.
2) Expression

result* := *expression1* > *expression2

Item	Description
result	Named variable or direct variable
expression1	ANY type
expression2	ANY type

Result of logical bit > operation is as follows.

expression1	***expression2***	***result***
0	0	0
0	1	0
1	0	1
1	1	0

Example	Description
Val1 := 20; Val2 := 10 ; Result := Val1 > Val2 ;	Compares two operands if left one is larger than right one. Result is 1.

14.3.13 < operator

1) Compares two operands if left one is smaller than right one.
2) Expression

result := expression1 < expression2

Item	Description
result	Named variable or direct variable
expression1	ANY type
expression2	ANY type

Result of logical bit < operation is as follows.

expression1	***expression2***	***result***
0	0	0
0	1	1
1	0	0
1	1	0

Example	Description
Val1 := 20; Val2 := 10 ; Result := Val1 < Val2 ;	Compares two operands if left one is smaller than right one. Result is 0.

14.3.14 >= operator

1) Compares two operands if left one is larger than right one or same.
2) Expression

result* := *expression1* >= *expression2

Item	Description
result	Named variable or direct variable
expression1	ANY type
expression2	ANY type

Result of logical bit >= operation is as follows.

expression1	***expression2***	***result***
0	0	1
0	1	0
1	0	1
1	1	1

Example	Description
Val1 := 20; Val2 := 20 ; Result := Val1 >= Val2 ;	Compares two operands if left one is larger than right one or same. Result is 1.

14.3.15 <= operator

1) Compares two operands if left one is smaller than right one or same.
2) Expression

result* := *expression1* <= *expression2

Item	Description
result	Named variable or direct variable
expression1	ANY type
expression2	ANY type

Result of logical bit <= operation is as follows.

expression1	***expression2***	***result***
0	0	1
0	1	0
1	0	1
1	1	1

Example	Description
Val1 := 2; Val2 := 20 ; Result := Val1 <= Val2 ;	Compares two operands if left one is smaller than right one or same. Result is1.

14.3.16 NOT operator

1) Changes bit value from 1 to 0 or from 0 to 1.
2) Expression

result* := *NOT expression

Item	Description
result	Named variable or direct variable
expression	ANY_BIT type

Example	Description
Val1 = 2#1100; Result:= NOT Val1 ;	Changes Val1 and output Result. Result is 2#0011.

14.3.17 - operator

1) Adds negative sign into value.
2) Expression

result* := *- expression

item	Description
result	Named variable or direct variable
expression	ANY_NUM type

Example	Description
Val1 = 10; Result:= - Val1 ;	Adds negative sign into value and output is result. Result is -10.

14.4 Statements

Statement is ended by semi colon(;).

14.4.1 Assignment statements

1) Assignment statement consists of Variable, operator(:=) and expression.
   Ex.) A := B + C ;
2) It is available to assign return value of function.

14.4.2 Selection statements

1) There are two types: IF and CASE.
2) According to specific condition, selection statement executes one statement or one group of statements among diverse statements.
   - IF
     (1) If condition of Bool expression is 1, it executes a group of statements.
     (2) If condition is not 1, it does not execute group of statements. But there is ELSE, it executes a group of statements following ELSE. If condition of ELSEIF is 1, a group of statements following ELEIF executes.
   - CASE
     (1) It consists of list of groups of statements and expression that calculates variable of INT type.
     (2) Each group can be set as integer and range of integer.
     (3) A group of statements in range of Selector executes and if any value is not in range of Selector, a group of statements following ELSE executes. If there is no ELSE, group of statements is not executed.

14.4.3 Iteration statements

1) There are three types, FOR, WHILE and REPEAT.
2) Some group executes repeatedly by iteration statement.
   - FOR
     (1) It is used when number of repetition is already determined.
     (2) In FOR statement, a group of statements executes repeatedly until END_FOR and status of repetition is saved in control variable of FOR loop.
     (3) Control variable, initial value and final value is expressed as integer type (SINT, INT, DINT) and does not change by repeated statement. Checking the condition for the end executes at the start of each repetition. If initial value exceeds the final value, a group of statements is not executed any more.
   - WHILE and REPEAT
     (1) WHILE statement (ended by END_WHILE) executes repeatedly until Bool expression is 0.
     (2) REPEAT statement (ended by UNTIL) executes repeatedly until Bool expression is 1.

(A group of statements executes at least one time)

(3) WHILE and REPEAT is not used for synchronizing process like “wait loop” which has the end condition determined exteriorly.

(4) EXIT statement is used to end iteration statements before meeting the end condition.

(5) EXIT statement is used to stop repetition before meeting the condition. When EXIT statement is used in overlapped repetition statements, relevant EXIT is applied to the loop in which EXIT exists. So, statements after first loop terminator (END_FOR, END_WHILE, END_REPEAT) are executed.

(6) IF WHILE and REPEAT are executed in unlimited loop, error occurs.

Number	Command	Example
1	Assignment	A:=B; CV:= CV+1; C:=SIN(X);
2	Recall of FB Using output of FB	CMD_TMR(IN:=%IX5, PT:= T#300ms); A:=CMD_TMR.Q;
3	RETURN	**RETURN**;
4	IF	D:=B*B -4*A*C; **IF** D<1.0 **THEN** NROOTS :=0; **ELSIF** D= 0.0 **THEN** NROOTS := 1; X1:= -B/(2.0*A); **ELSE** X1:= (-B+SQRT(D))/(2.0*A); X2:= (-B-SQRT(D))/(2.0*A); **END_IF**;
5	CASE	TW := BCD_TO_INT(THUMBWHEEL); TW_ERROR := 0; **CASE** TW **OF** 1, 5: DISPLAY := OVEN_TEMP; 2: DISPLAY := MOTOR_SPEED; 3: DISPLAY := GROSS – TARE; 4,6..10: DISPLAY := STATUS(TW-4); **ELSE** DISPLAY := 0; TW_ERROR := 1; **END_CASE**; QW100 := INT_TO_BCD(DISPLAY);
6	FOR	J := 101; **FOR** I := 1 **TO** 100 **BY** 2 **DO**

Number	Command	Example
		IF WORDS[I] = 'KEY' THEN J := I; **EXIT**; END_IF; **END_FOR** ;
7	WHILE	J := 1; **WHILE** J <= 100 & WORDS[J] <> 'KEY' **DO** J := J+2; **END_WHILE**;
8	REPEAT	J := -1; **REPEAT** J := J+2; **UNTIL** J = 101 OR WORDS[J] = 'KEY' **END_REPEAT ;**
9	EXIT	**EXIT**;
10	Null/Space command text	;
EXIT is used for all repetition texts (FOR, WHILE, REPEAT).		

<Table 3> Command for ST

14.4.4 IF

1) It is used for program to select more than one
2) Expression

IF *condition* **THEN** *statements* [**ELSE** *elsestatements*] **END_IF**

Or

```
IF condition THEN
   statements
[ELSIF condition-n THEN
   elseifstatements] . . .
[ELSE
   elsestatements]
END_IF
```

Item	Description
condition	If ***condition*** is TRUE, a ***statement*** following THEN is executed. In case of FLASE, ELSIF or ELSE executes.
statements	If ***condition*** is TRUE, a statement more than one executes.

Item	Description
condition-n	N ***conditions*** can be used.
elseifstatements	If ***condition-n*** is TRUE, a statement more than one executes.
elsestatements	If ***condition*** or ***condition-n*** is false, a statement more than one executes.

Example	Description
IF Val1 <= 10 **THEN** Result := 10; **END_IF**;	If condition (Val1 <= 10) is TRUE, 10 is assigned into result.
IF Val1 <= 10 **THEN** Result := 10; **ELSE** Result := 20; **END_IF**;	If condition (Val1 <= 10) is TRUE, 10 is assigned into result. If condition is FALSE, 20 is assigned into result.
IF Val1 <= 10 **THEN** Result := 10; **ELSIF** Val1 <= 20 **THEN** Result := 20; **ELSE** Result := 30; **END_IF**;	If condition (Val1 <= 10) is TRUE, 10 is assigned into result. If condition is FALSE, ELSEIF executes. If second condition (Val <= 20) is TRUE, 20 is assigned into result. If second is FALSE, a statement under ELSE executes. Namely, 30 is assigned into result.

14.4.5 CASE

1) Diverges according to value of expression following CASE. Expression should be integer. When value of expression is not included in case list, a statement after ELSE executes. If there is no ELSE, no statement list executes.
2) Expression

CASE *expression* **OF**
case_list **:** *statement_list*
{ *case_list* **:** *statement_list*}
[**ELSE**
statement_list]
END_CASE

Item	Description
expression	Only INT type is available.
case_list	*case_list_element* {',' *case_list_element*} There are diverse statement like above.

Item	Description
case_list_element	*Subrange* or *signed_integer are available*
subrange	*signed_integer* **..** *signed_integer* type
statement_list	Executes more than one statements

Example	Description
CASE Val1 **OF** 1 **:** Result := 10 ; 2..5 **:** Result := 20 ; 7, 10 **:** Result := 30 ; **ELSE** Result := 40 ; **END_CASE ;**	If value of Val1 is 1, 10 is assigned into result. If value of Val1 is 2~5, 20 is assigned into result. If value of Val1 is 7 or 10, 30 is assigned into result. In case of other values, 40 is assigned into result.

14.4.6 FOR

1) It is used to deal with repetition and uses three control statements. First, statement for initialization is necessary. If To expression is TRUE (present counter value is less than end value), loop executes one time. Then counter values increases as many as BY value and condition is checked again. In FOR statement, condition is checked first and loop executes later. So no loop may be executed.
2) Expression

FOR *counter* **:=** *start* **TO** *end* [**BY** *step*] **DO**
statements
END_FOR

Item	Description
counter	Integer (SINT, INT, DINT) s start, end, step should be the same type.
start	Initial value of *counter*
end	Last value of *counter*
step	Indicates increment of *count* variable whenever loop executes. If this is not used, increment is 1.
statements	It executes according to three control texts.

Example	Description
SUM := 0; **FOR** counter := 0 **TO** 10 **DO**	*Counter* variable increases from 0 to 10 as many as 1. 1 is added into SUM variable repeatedly. Final value of SUM is 11.

SUM := SUM + 1; **END_FOR** ;	
SUM := 0; **FOR** counter = 0 **TO** 10 **BY** 2 **DO** SUM := SUM + 1; **END_FOR** ;	*Counter* variable increases from 0 to 10 as many as 2. 1 is added into SUM variable repeatedly. Final value of SUM is 6.

> **Note**
>
> 1) Because of long scan time, watch - dog may be on.
> 2) BY part can be skipped. In case of skip, it increases as many as 1.
> 3) If *start* is larger than *end*, FOR text is not executed.

14.4.7 WHILE

1) It executes repeatedly until condition is 0. In WHILE statement, condition is checked first and loop is executed later. So no loop executes.

2) Expression

WHILE *condition* **DO**
statements
END_ WHILE

Item	Description
condition	If *condition* is TRUE, statements after DO executes. In case of FLASE, it goes out from loop.
statements	If *condition* is TRUE, statements more than one executes.

Example	Description
Counter := 0 **WHILE** Counter < 20 **DO** Counter := Counter + 1; **END_WHILE** ;	If condition that counter is less than 20 is TRUE, a statement executes. If condition is FALSE, it goes out from loop.

> **Note**
>
> In WHILE statement, in case, condition does not become 0, it cannot go out from loop. In this case, due to long scan time, watch-dog is on. So be careful so that condition is not always TRUE.

14.4.8 REPEAT

1) Statement executes repeatedly until condition is TRUE. In REPEAT statement, loop executes first and condition is

checked later. So loop executes at least one time.

2) Expression

REPEAT

statements

UNTIL *condition*

END_REPEAT

Item	Description
condition	If *condition* is FALSE, it executes repeatedly and if *TRUE*, goes out from loop.
statements	Loop executes repeatedly until condition is TRUE.

Example	Description
Counter := 0; **REPEAT** DO Counter := Counter + 1; **UNTIL** Counter > 20 **END_REPEAT** ;	First, Counter variable is set to 1. If the condition that Counter variable is larger than 2 is met, it goes out from loop or it executes loop. If Counter variable is 21, condition is TRUE and it goes out from loop.

Note

In REPEAT statement, in case condition doesn’t become 1, it cannot go out from loop. In this case, due to long scan time, watch-dog is on. So be careful so that condition is not always FALSE.

14.4.9 EXIT

1) It is used to go out from iteration statements (WHILE, FOR, REPEAT).
2) If it is used outside iteration statements, error occurs.
3) Expression

   EXIT

Example	Description
SUM := 0; **FOR** Counter := 0 TO 10 DO SUM := SUM + 1; **EXIT**; **END_FOR** ;	Counter variable increases from 0 to 10 as many as1. But because of EXT, loop ends. Counter variable becomes 0 and SUM becomes 1.
Counter := 0; **WHILE** Counter < 20 DO Counter := Counter + 1 ; **IF** Counter = 10 **THEN** **EXIT**; **END_IF**; **END_WHILE** ;	Text executes repeatedly when Counter is less than 20 and if Counter is larger than 20, loop ends. But because of IF statement and EXIT statement, loop ends when Counter is 10.
Counter := 0; **REPEAT DO** Counter := Counter + 1 ; **IF** Counter = 10 **THEN** **EXIT**; **END_IF**; **UNTIL** Counter > 20 **END_REPEAT** ;	Counter variable increase as many as 1. If Counter is larger than 20, loop ends otherwise loop executes repeatedly. But because of IF statement and EXIT statement, loop ends when Counter is 10.

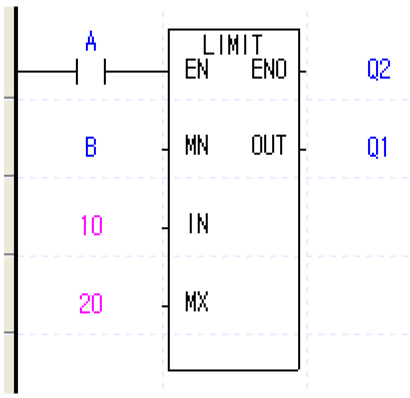
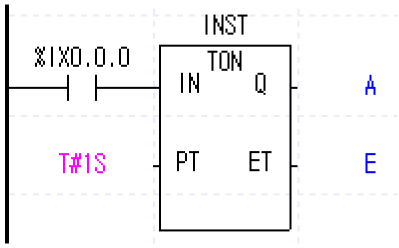
14.5 Function and Function Block

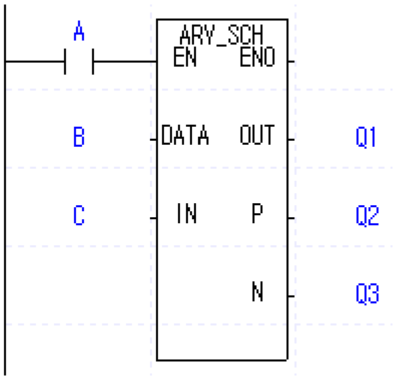
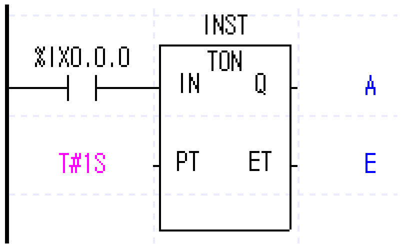
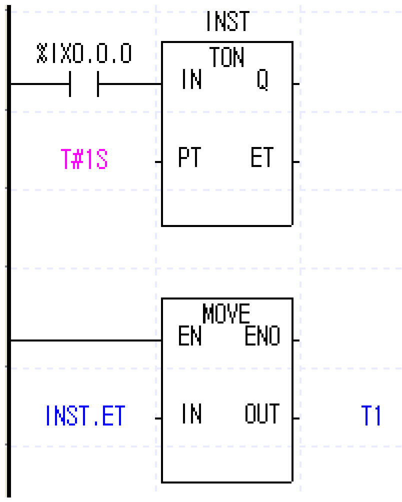
14.5.1 How to use

1) There are two types (Standard type, nonstandard type) for use of function and function block. Both are available according to environment.

(1) Standard type:

It writes the input, output parameter name of function and function block

Parameter	Function	Function Block
Common	Order of parameter does not matter. Q1 := LIMIT(MN := B, MX := 20, IN := 10) ; Q1 := LIMIT(MX := 20, MN := B, IN := 10) ; EN, ENO can be omitted. Q1 := LIMIT(**EN := A,** MN := B, MX := 20, IN := 10, ENO => Q2) ; 	Order of parameter does not matter. INST(IN := %IX0.0.0, PT := T#1s, Q => A, ET => E) ; INST(PT := T#1s, IN := %IX0.0.0, Q => A, ET => E) ; 
Input	Use “:=” for input parameter allocation. C := LIMIT(**MN := B, MX := 20**, IN := 10) ;	Use “:=” for input parameter allocation.. INST(**IN := %IX0.0.0, PT := T#1s**, Q => A, ET => B) ;
Output	If output parameter name is OUT or Y (For user defined function, function name), allocate as the return value. For other output parameters, use “=>”. **Q1 :=** ARY_SCH(DATA := B, IN := C, **P => Q2, N => Q3**) ;	Use “=>” for out parameter allocation Output parameter allocation can be omitted. INST(IN := %IXunction 0.0.0, PT := T#1s, **Q => A, ET => E**) ;

Parameter	Function	Function Block
	 Not used output parameter can be omitted as follows. (Q2, Q3 have been omitted) Q1 **:=** ARY_SCH(DATA := B, IN := C) ;	 INST(IN := %IX0.0.0, PT := T#1s) ; T1 := INST.ET; 

Note

To use the function block, write instance name of function block. Declare the function block as how to declare the variable and write this variable name (instance name)

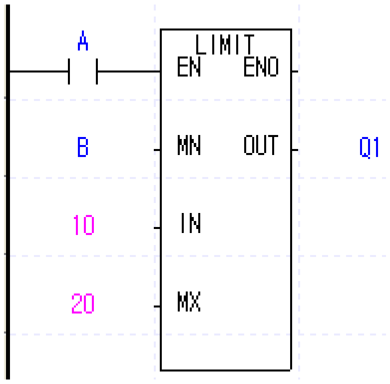
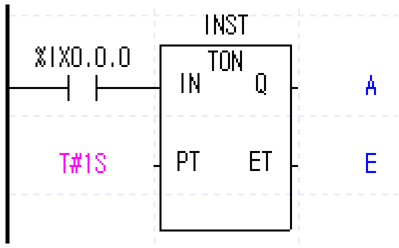
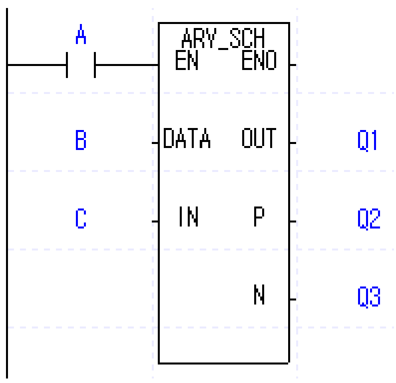
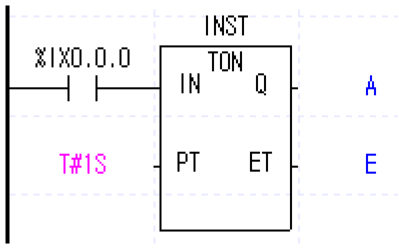
Ex.) Use of timer

	Variable Kind	Variable	Type
1	VAR	INST_TON1	TON

INST_TON1(IN := TRUE, PT := T#100MS, Q => Q_OUT, ET => ET_OUT);

(2) Nonstandard type

In this type, I/O parameter name of function and function block is omitted

Parameter	Function	Function Block
Common	You cannot change the order of all parameters. You cannot omit any parameter Q1 := LIMIT(B, 20, 10) ;  You cannot use EN, ENO	You cannot change the order of all parameters. You cannot omit any parameter INST(%IX0.0.0, T#1s, A, E) ; 
Input	You cannot change the order of input parameter. C := LIMIT(**B, 20**, IN := 10) ;	You cannot change the order of input parameter. INST(**%IX0.0.0, T#1s**, A, E) ;
Output	If output parameter name is OUT or Y (For user defined function, function name), allocate as the return value. For other output parameters, input in order of position **Q1 :=** ARY_SCH(B, C, **Q2, Q3**) ; 	For all output parameters, input in order of position INST(%IX0.0.0, T#1s, **A, E**) ; 

Note

For function whose parameter type is variable, input parameter type should be determined.

Example	Description
INT1 := ADD(1, 2, 3);	Error occurs while determining function type

For normal operation, choose one among below three examples

Example	Description
INT1 := ADD(**INT#1**, 2, 3);	Sets the type of constant
INT1 := ADD(**B**, 2, 3);	Uses variable (B)
INT1 := **ADD_INT**(1, 2, 3);	Uses the type-defined function

Note

- Input parameter EN is condition to execute the function. If you use the EN as follows, LIMIT function executes

when A is 1.

OUT := LIMIT(EN := A, MX := 20, MN := B, IN := 10) ;

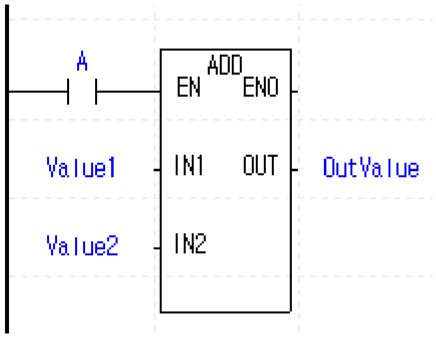
ENO parameter becomes 1 when function executes without error. It cannot be used in ST and available in LD

Note

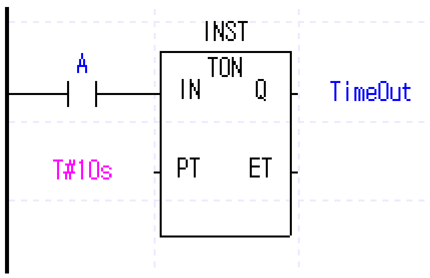
1. ST does not support the extension functions(BREAK, CALL, END, FOR, INIT_DONE, JMP, NEXT, RET, SBRT)
2. You cannot use the function whose name is same as operator name. (OR, XOR, AND, MOD, NOT)

14.5.2 Example

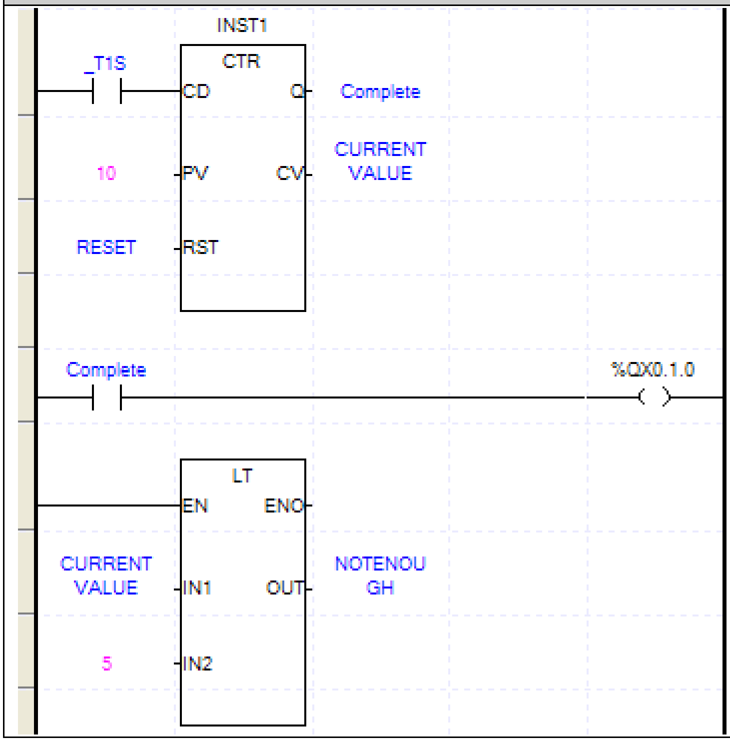
1) Function

Use of LD	Use of ST
	1) Standard type EN used OutValue := ADD(EN := A, IN1 := Value1, IN2 := Value2); EN not used OutValue := ADD(IN1 := Value1, IN2 := Value2); 2) Nonstandard type OutValue := ADD(Value1, Value2); EN, ENO cannot be used

2) Function Block

Use of LD	Use of ST
	1) Standard type INST(IN := A, PT := T#10S, Q => TimeOut); 2) Nonstandard type INST(A, T#10S, TimeOut, TimeValue); Output variable cannot be omitted. So you have to allocate the applicable variable to output parameter ET. (TimeValue)

3) Application

<table>
<tr><th>Use of LD</th><th>Use of ST</th></tr>
<tr><td>


</td><td>INST1(CD := _T1S, PV := 10, RST := RESET, Q => COMPLETE,
CV => CURRENTVALUE);

%QX0.1.0 := Complete;

NOTENOUGH := LT(IN1 := CURRENTVALUE, IN2 := 5);</td></tr>
</table>

Chapter 15. Safety Function Blocks

15.1. Safety Function Blocks List

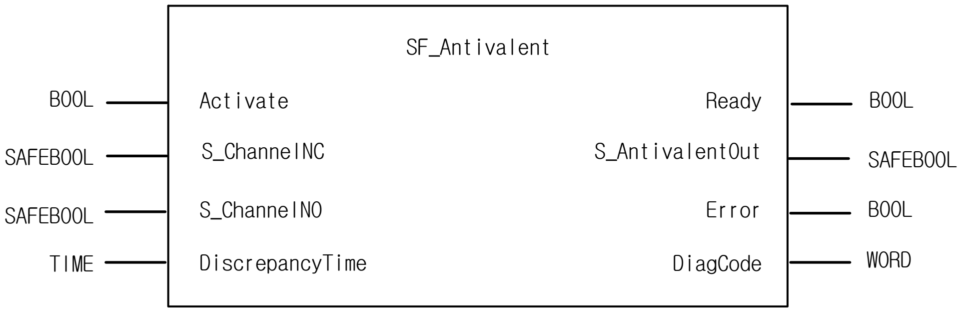
No	Function Block
1	SF_ANTIVALENT
2	SF_EDM
3	SF_ENABLESWITCH
4	SF_EQUIVALENT
5	SF_ESPE
6	SF_ESTOP
7	SF_GUARDLOCKING
8	SF_MODESEL
9	SF_MUTINGPAR
10	SF_MUTINGPAR_2SENSOR
11	SF_MUTINGSEQ
12	SF_OUTCONTROL
13	SF_SAFEGUARD
14	SF_SAFETYREQUEST
15	SF_TESTABLESAFETYSENSOR
16	SF_TWOHANDCTRLII
17	SF_TWOHANDCTRLIII

15.2. Safety Function Blocks

15.2.1 SF_ANTIVALENT

1) Overview

This function block converts two antivalent SAFEBOOL inputs (NO/NC pair) to one SAFEBOOL output with discrepancy time monitoring. This FB should not be used stand-alone since it has no restart interlock. It is required to connect the output to other safety related functionalities.



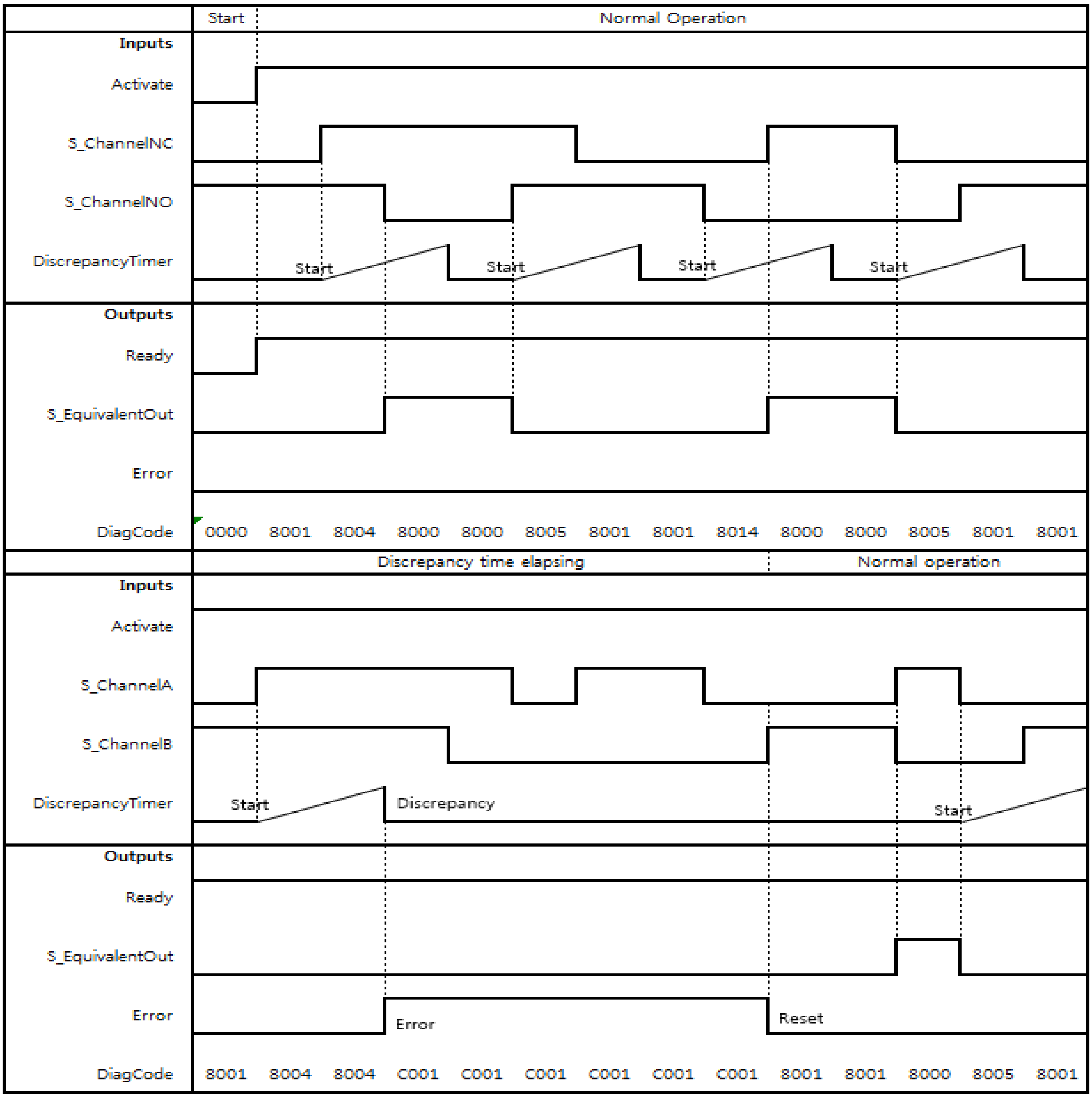
2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_ChannelNC	SAFEBOOL	0	Variable. NC stands for Normally Closed. Input for NC connection. FALSE: NC contact open. TRUE: NC contact closed.
	S_ChannelNO	SAFEBOOL	1	Variable. NO stands for Normally Open. Input for NO connection. FALSE: NO contact open TRUE: NO contact closed
	DiscrepancyTime	TIME	T#0ms	Constant. Maximum monitoring time for discrepancy status of both inputs.
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_AntivalentOut	SAFEBOOL	0	Safety related output FALSE: Minimum of one input signal "not active" or status change outside of monitoring time. TRUE: Both inputs signals "active" and status change within monitoring time.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more then 16 codes.

3) Functional Description

This function block converts two equivalent SAFEBOOL inputs to one SAFEBOOL output with discrepancy time monitoring. Both input Channels A and B are interdependent. The function block output shows the result of the evaluation of both channels. If one channel signal changes from TRUE to FALSE the output immediately switches off (FALSE) for safety reasons.
Discrepancy time monitoring: The discrepancy time is the maximum period during which both inputs may have different states without the function block detecting an error. Discrepancy time monitoring starts when the status of an input changes. The function block detects an error when both inputs do not have the same status once the discrepancy time has elapsed.
The inputs must be switched symmetrically. This means that monitoring is performed for both the switching on process as well as the switching off process.

4) Typical Timing Diagrams



5) Error Detection

The function block monitors the discrepancy time between Channel NO and Channel NC.

6) Error Behavior

The output SF_AntivalentOut is set to FALSE. Error is set to TRUE. DiagCode indicates the Error states.
There is no Reset defined as an input coupled with the reset of an error. If an error occurs in the inputs, one new set of inputs with the correct value must be able to reset the error flag. (Example: if a switch is faulty and replaced, using the switch again results in a correct output)

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Error 1	Discrepancy time elapsed in state 8004. Ready = TRUE S_AntivalentOut = FALSE Error = TRUE
C002	Error 2	Discrepancy time elapsed in state 8014. Ready = TRUE S_AntivalentOut = FALSE Error = TRUE
C003	Error 3	Discrepancy time elapsed in state 8005. Ready = TRUE S_AntivalentOut = FALSE Error = TRUE

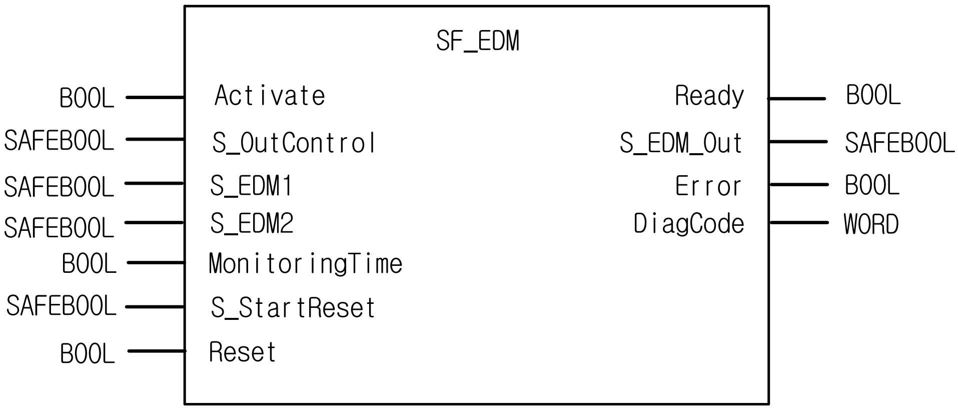
8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_AntivalentOut = FALSE Error = FALSE
8001	Init	An activation has been detected by the FB and the FB is now activated. Ready = TRUE S_AntivalentOut = FALSE Error = FALSE
8000	Safety Output Enabled	The inputs switched to the Active state in antivalent mode. Ready = TRUE S_AntivalentOut = TRUE Error = FALSE
8004	Wait for NO	ChannelNC has been switched to TRUE - waiting for ChannelNO to be switched to FALSE; discrepancy timer started. Ready = TRUE S_AntivalentOut = FALSE Error = FALSE
8014	Wait for NC	ChannelNO has been switched to FALSE - waiting for ChannelNC to be switched to TRUE; discrepancy timer started. Ready = TRUE S_AntivalentOut = FALSE Error = FALSE
8005	From Active Wait	One channel has been switched to inactive; waiting for the second channel to be switched to inactive too. Ready = TRUE S_AntivalentOut = FALSE Error = FALSE

15.2.2 SF_EDM

1) Overview

External device monitoring – The FB controls a safety output and monitors controlled actuators, e.g. subsequent contactors



2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_OutControl	SAFEBOOL	0	Control signal of the preceeding safety FB's. Typical function block signals from the library (e.g.,SF_OutControl, SF_TwoHandControlTypeII, and/or others). FALSE: Disable safety output (S_EDM_Out). TRUE: Enable safety output (S_EDM_Out).
	S_EDM1	SAFEBOOL	0	Feedback signal of the first connected actuator. FALSE: Switching state of the first connected actuator. TRUE: Initial state of the first connected actuator.
	S_EDM2	SAFEBOOL	0	Feedback signal of the second connected actuator. If using only one signal in the application, the user must use a graphic connection to jumper the S_EDM1 and S_EDM2 parameters. S_EDM1 and S_EDM2 are then controlled by the same signal. FALSE: Switching state of the second connected actuator. TRUE: Initial state of the second connected actuator.
	MonitoringTime	TIME	#0ms	Max. response time of the connected and monitored actuators.
	S_StartReset			FALSE (= initial value): Manual reset when PES is started (warm or cold). TRUE: Automatic reset when PES is started (warm or cold).
	Reset	BOOL	0	Reset

Type	Name	Data Type	Initial Value	Description
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_EDM_Out	SAFEBOOL	0	Controls the actuator. The result is monitored by the feedback signal S_EDMx. FALSE: Disable connected actuators. TRUE: Enable connected actuators.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more then 16 codes.

3) Functional Description

General:

The SF_EDM FB controls a safety output and monitors controlled actuators.

This function block monitors the initial state of the actuators via the feedback signals (S_EDM1 and S_EDM2) before the actuators are enabled by the FB.

The function block monitors the switching state of the actuators (MonitoringTime) after the actuators have been enabled by the FB.

Two single feedback signals must be used for an exact diagnosis of the connected actuators. A common feedback signal from the two connected actuators must be used for a restricted yet simple diagnostic function of the connected actuators. When doing so, the user must connect this common signal to both parameter S_EDM1 and parameter S_EDM2. S_EDM1 and S_EDM2 are then controlled by the same signal.

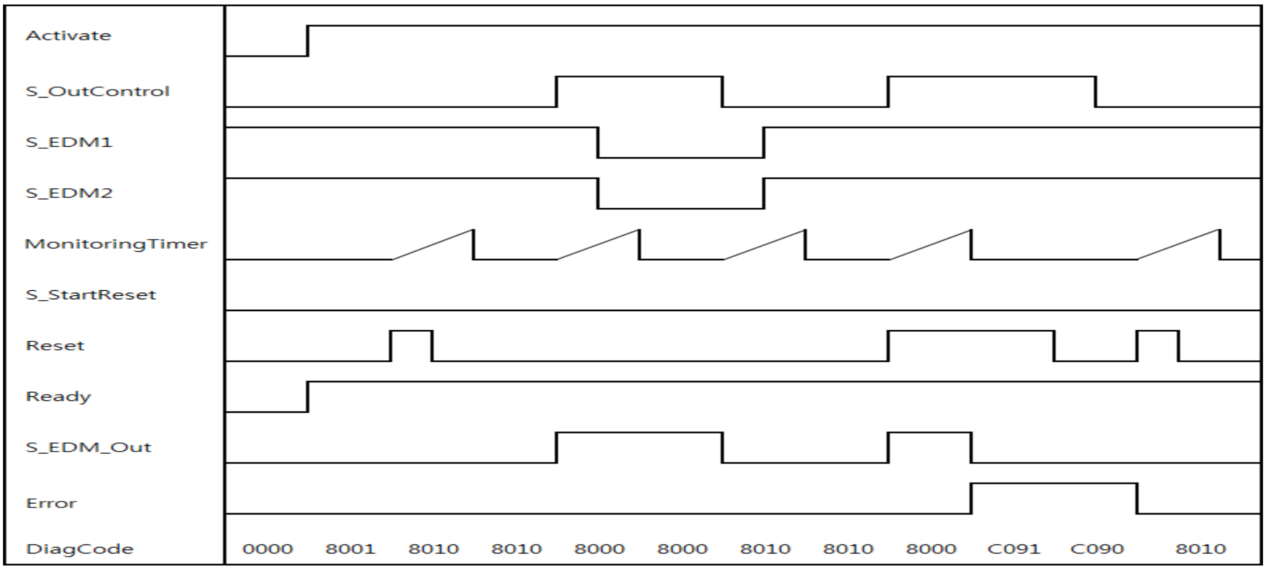
The switching devices used in the safety function should be selected from the category specified in the risk analysis (EN 954-1).

Optional startup inhibits:

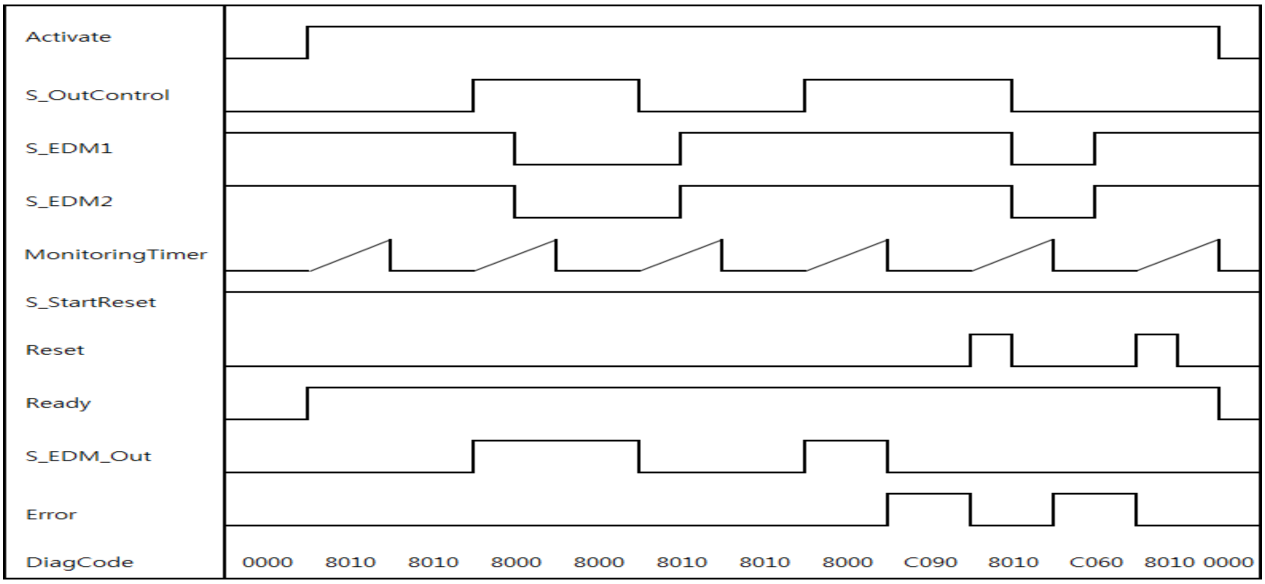
• Startup inhibit in the event of block activation.

The S_StartReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

4) Typical Timing Diagrams



< S_StartReset=Off >



< S_StartReset=On >

5) Error Detection

The following conditions force a transition to the Error state:

• Invalid static Reset signal in the process.

• Invalid EDM signal in the process.

• S_OutControl and Reset are incorrectly interconnected due to programming error.

6) Error Behavior

In error states, the outputs are as follows:

• In the event of an error, the S_EDM_Out is set to FALSE and remains in this safe state.

• An EDM error message must always be reset by a rising trigger at Reset.

• A Reset error message can be reset by setting Reset to FALSE.

After block activation, the optional startup inhibit can be reset by a rising edge at the Reset input.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Reset Error 1	Static Reset signal in state 8001. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C011	Reset Error 21	Static Reset signal or same signals at EDM1 and Reset (rising trigger at Reset and EDM1 at the same time) in state C010. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C021	Reset Error 22	Static Reset signal or same signals at EDM2 and Reset (rising trigger at Reset and EDM2 at the same time) in state C020. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C031	Reset Error 23	Static Reset signal or same signals at EDM1, EDM2, and Reset (rising trigger at Reset, EDM1, and EDM2 at the same time) in state C030. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C041	Reset Error 31	Static Reset signal or same signals at EDM1 and Reset (rising trigger at Reset and EDM1 at the same time) in state C040. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C051	Reset Error 32	Static Reset signal or same signals at EDM2 and Reset (rising trigger at Reset and EDM2 at the same time) in state C050. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C061	Reset Error 33	Static Reset signal or same signals at EDM1, EDM2, and Reset (rising trigger at Reset, EDM1, and EDM2 at the same time) in state C060. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C071	Reset Error 41	Static Reset signal in state C070. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C081	Reset Error 42	Static Reset signal in state C080. Ready = TRUE S_EDM_Out = FALSE Error = TRUE

DiagCode	State Name	State Description and Output Setting
C091	Reset Error 43	Static Reset signal in state C090. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C010	EDM Error 11	The signal at EDM1 is not valid in the initial actuator state. In state 8010 the EDM1 signal is FALSE when enabling O_OutControl. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C020	EDM Error 12	The signal at EDM2 is not valid in the initial actuator state. In state 8010 the EDM2 signal is FALSE when enabling O_OutControl. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C030	EDM Error 13	The signals at EDM1 and EDM2 are not valid in the initial actuator states. In state 8010 the EDM1 and EDM2 signals are FALSE when enabling O_OutControl. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C040	EDM Error 21	The signal at EDM1 is not valid in the initial actuator state. In state 8010 the EDM1 signal is FALSE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C050	EDM Error 22	The signal at EDM2 is not valid in the initial actuator state. In state 8010 the EDM2 signal is FALSE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C060	EDM Error 23	The signals at EDM1 and EDM2 are not valid in the initial actuator states. In state 8010 the EDM1 and EDM2 signals are FALSE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C070	EDM Error 31	The signal at EDM1 is not valid in the actuator switching state. In state 8000 the EDM1 signal is TRUE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE

DiagCode	State Name	State Description and Output Setting
C080	EDM Error 32	The signal at EDM2 is not valid in the actuator switching state. In state 8000 the EDM2 signal is TRUE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C090	EDM Error 33	The signals at EDM1 and EDM2 are not valid in the actuator switching state. In state 8000 the EDM1 and EDM2 signals are TRUE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C111	Init Error	S Similar signals at S_OutControl and Reset (R_TRIG at same cycle) detected (may be a programming error) Ready = TRUE S_EDM_Out = FALSE Error = TRUE

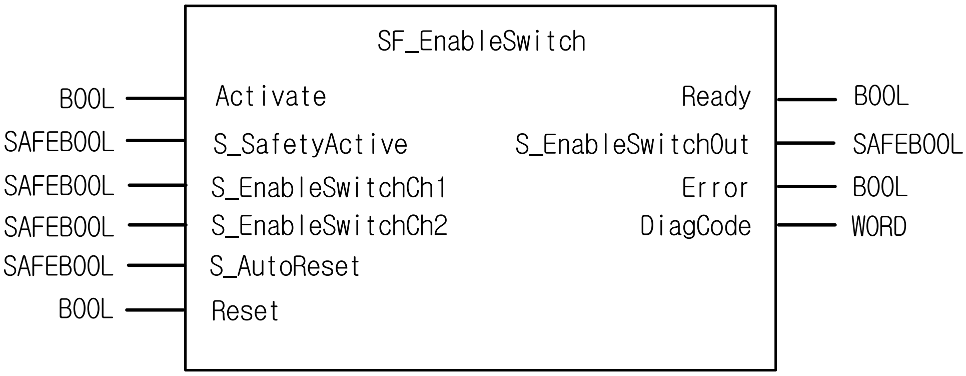
8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_EDM_Out = FALSE Error = FALSE
8001	Init	Block activation startup inhibit is active. Reset required. Ready = TRUE S_EDM_Out = FALSE Error = FALSE
8010	Output Disable	EDM control is not active. Timer starts when state is entered Ready = TRUE S_EDM_Out = FALSE Error = FALSE
8000	Output Enable	EDM control is active. Timer starts when state is entered Ready = TRUE S_EDM_Out = TRUE Error = FALSE

15.2.3 SF_ENABLESWITCH

1) Overview

The SF_EnableSwitch FB evaluates the signals of an enable switch with three positions.



2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_SafetyActive	SAFEBOOL	0	Confirmation of the safe mode (limitation of the speed or the power of motion, limitation of the range of motion). FALSE: Safe mode is not active. TRUE: Safe mode is active.
	S_EnableSwitchCh1	SAFEBOOL	0	Signal of contacts E1 and E2 of the connected enable switch. FALSE: Connected switches are open. TRUE: Connected switches are closed.
	S_EnableSwitchCh2	SAFEBOOL	0	Signal of contacts E3 and E4 of the connected enable switch. FALSE: Connected switches are open. TRUE: Connected switches are closed.
	S_AutoReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when emergency stop button is released. TRUE: Automatic reset when emergency stop button is released. This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
	Reset	BOOL	0	Reset
Type	**Name**	**Data Type**	**Initial Value**	**Description**

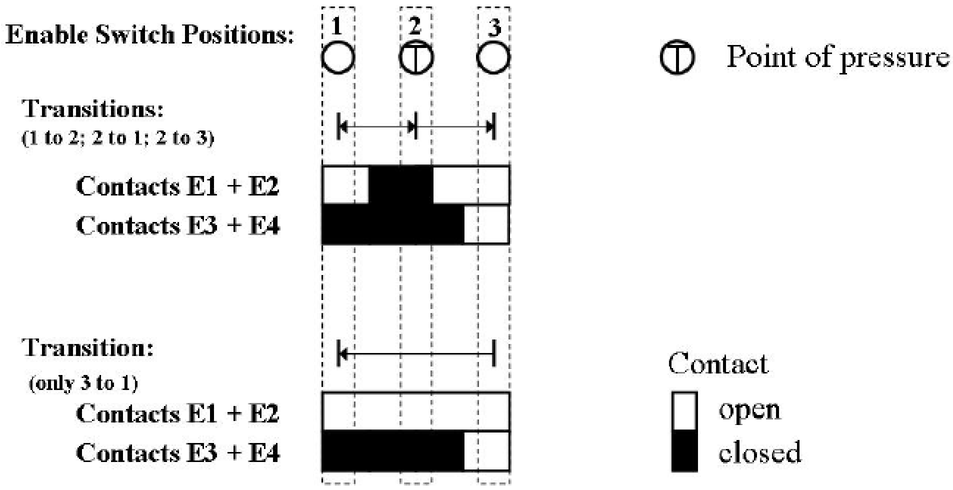
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_EnableSwitchOut	SAFEBOOL	0	Safety related output: Indicates suspension of guard. FALSE: Disable suspension of safeguarding. TRUE: Enable suspension of safeguarding.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more then 16 codes.

3) Functional Description

The SF_EnableSwitch FB supports the suspension of safeguarding using enable switches, if the relevant operating mode is selected and active. The relevant operating mode (limitation of the speed or the power of motion, limitation of the range of motion) must be selected outside the SF_EnableSwitch FB.

The SF_EnableSwitch FB evaluates the signals of an enable switch with three positions

The S_EnableSwitchCh1 and S_EnableSwitchCh2 input parameters process the following signal levels of contacts E1 to E4:



The signal from E1+E2 must be connected to the S_EnableSwitchCh1 parameter. The signal from E3+E4 must be connected to the S_EnableSwitchCh2 parameter. The position of the enable switch is detected in the FB using this signal sequence. The transition from position 2 to 3 can be different from shown here.

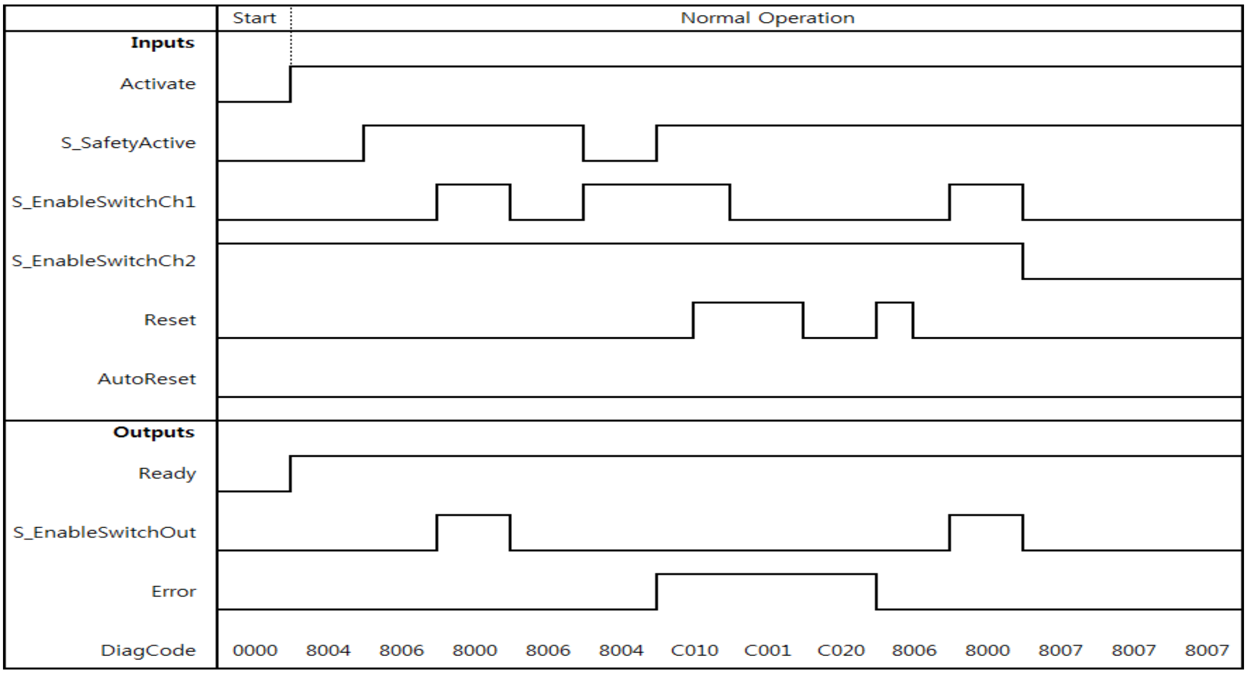
The switching direction (position 1 => position 2/position 3 => position 2) can be detected in the FB using the defined signal sequence of the enable switch contacts. The suspension of safeguarding can only be enabled by the FB after a move from position 1 to position 2. Other switching directions or positions may not be used to enable the suspension of safeguarding.

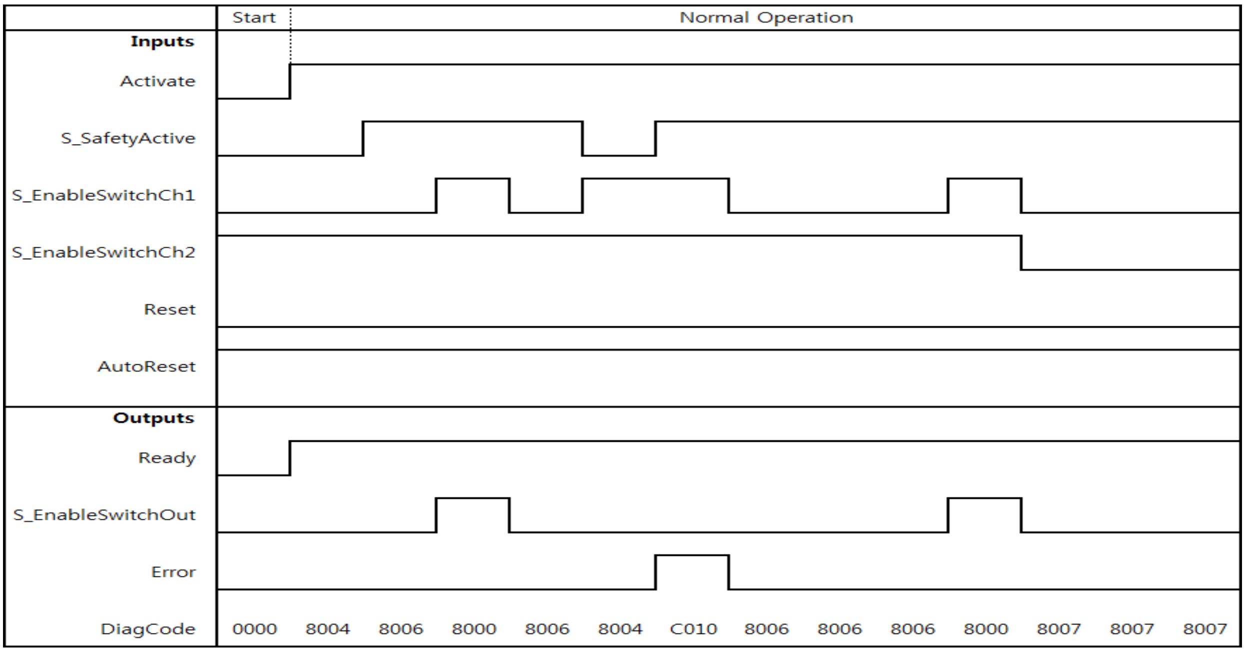
In order to meet the requirements of DIN EN 60204 Section 9.2.4, the user shall use a suitable switching device. In addition, the user must ensure that the relevant operating mode is selected in the application (automatic operation must be disabled in this operating mode using appropriate measures).

The operating mode is usually specified using an operating mode selection switch in conjunction with the SF_ModeSelector FB and the SF_SafeRequest or SF_SafelyLimitedSpeed FB.

The SF_EnableSwitch FB processes the confirmation of the "safe mode" state via the "S_SafetyActive" parameter. On implementation in an application of the safe mode without confirmation, a static TRUE signal is connected to the "S_SafetyActive" parameter. The S_AutoReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

4) Typical Timing Diagrams





5) Error Detection

The following conditions force a transition to the Error state:

- Invalid static Reset signal in the process.
- Invalid switch positions.

6) Error Behavior

In the event of an error, the S_EnableSwitchOut safe output is set to FALSE and remains in this Safe state. Different from other FBs, a Reset Error state can be left by the condition Reset = FALSE or, additionally, when the signal S_SafetyActive is FALSE. Once the error has been removed, the enable switch must be in the initial position specified in the process before the S_EnableSwitchOut output can be set to TRUE using the enable switch. If S_AutoReset = FALSE, a rising trigger is required at Reset.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Reset Error 1	Static Reset signal detected in state C020. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C002	Reset Error 2	Static Reset signal detected in state C040. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C010	Operation Error 1	Enable switch not in position 1 during activation of S_SafetyActive. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C020	Operation Error 2	Enable switch in position 1 after C010. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C030	Operation Error 3	Enable switch in position 2 after position 3. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C040	Operation Error 4	Enable switch not in position 2 after C030. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE

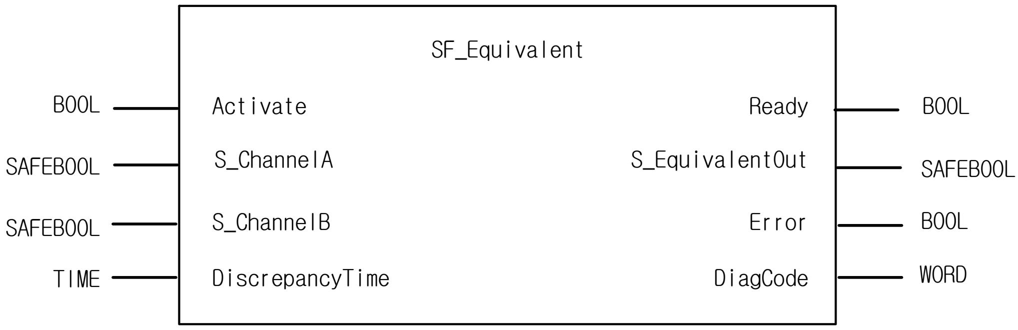
8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_EnableSwitchOut = FALSE Error = FALSE
8004	Basic Operation Mode	Safe operation mode is not active. Ready = TRUE S_EnableSwitchOut = FALSE Error = FALSE
8005	Safe Operation Mode	Safe operation mode is active. Ready = TRUE S_EnableSwitchOut = FALSE Error = FALSE
8006	Position 1	Safe operation mode is active and the enable switch is in position 1. Ready = TRUE S_EnableSwitchOut = FALSE Error = FALSE
8007	Position 3	Safe operation mode is active and the enable switch is in position 3. Ready = TRUE S_EnableSwitchOut = FALSE Error = FALSE
8000	Position 2	Safe operation mode is active and the enable switch is in position 2. Ready = TRUE S_EnableSwitchOut = TRUE Error = FALSE

15.2.4 SF_EQUIVALENT

1) Overview

This function block converts two equivalent SAFEBOOL inputs (both NO or NC) to one SAFEBOOL output, including discrepancy time monitoring. This FB should not be used stand-alone since it has no restart interlock. It is required to connect the output to other safety related functionalities.



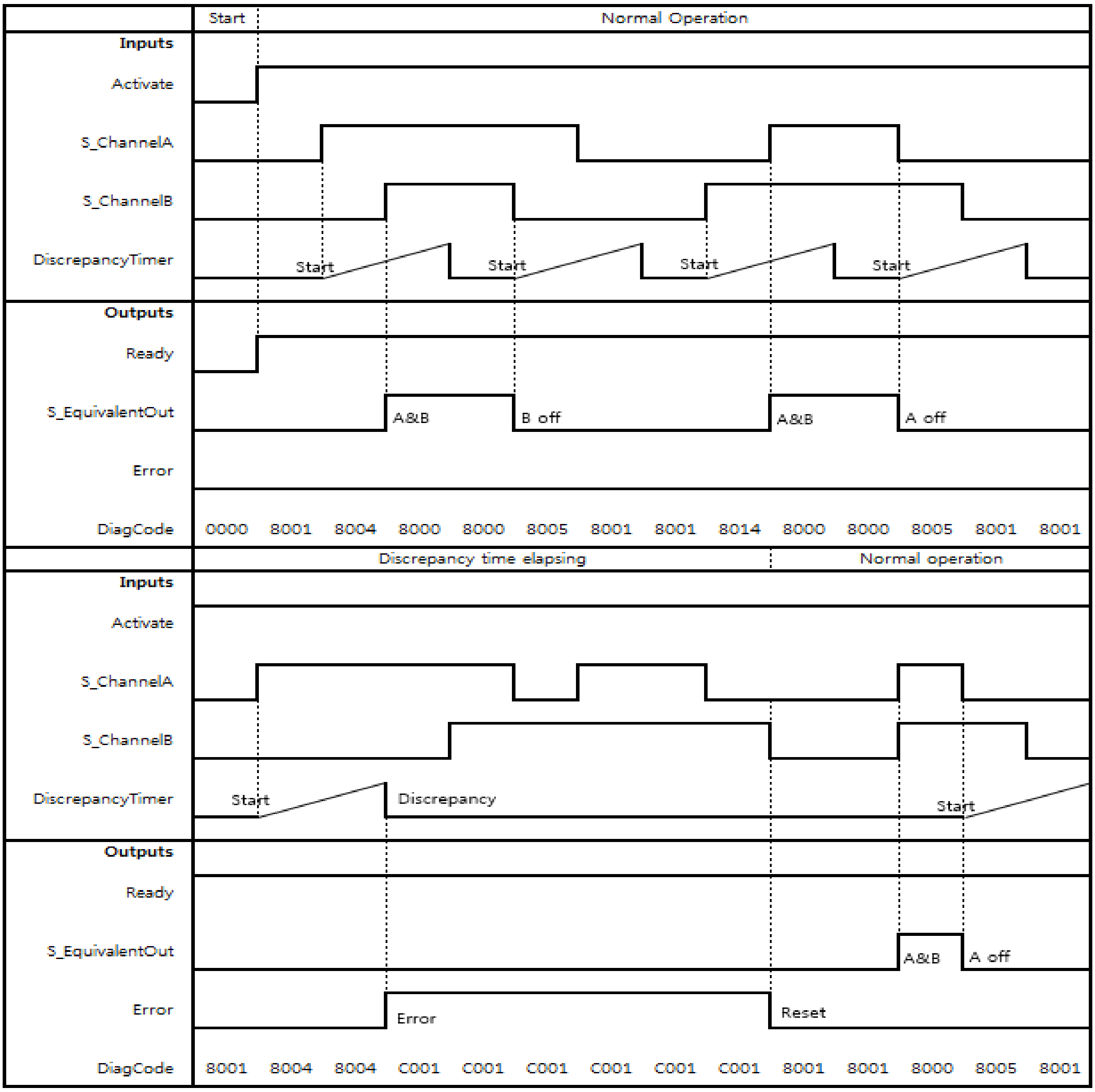
2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_ChannelA	SAFEBOOL	0	Input A for logical connection. FALSE: Contact A open TRUE: Contact A closed.
	S_ChannelB	SAFEBOOL	0	Input B for logical connection. FALSE: Contact B open TRUE: Contact B closed.
	DiscrepancyTime	TIME	T#0ms	2 개 Input 의 Discrepancy time 설정 0 ~ 65535ms
Output	Ready	BOOL	0	Maximum monitoring time for discrepancy status of both inputs.
	S_EquivalentOut	SAFEBOOL	0	Safety related output FALSE: Minimum of one input signal = "FALSE" or status change outside of monitoring time. TRUE: Both input signals "active" and status change within monitoring time
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more then 16 codes.

3) Functional Description

This function block converts two equivalent SAFEBOOL inputs to one SAFEBOOL output with discrepancy time monitoring. Both input Channels A and B are interdependent. The function block output shows the result of the evaluation of both channels. If one channel signal changes from TRUE to FALSE the output immediately switches off for safety reasons. Discrepancy time monitoring: The discrepancy time is the maximum period during which both inputs may have different states without the function block detecting an error. Discrepancy time monitoring starts when the status of an input changes. The function block detects an error when both inputs do not have the same status once the discrepancy time has elapsed.
The inputs must be switched symmetrically. This means that monitoring is performed for both the switching on process as well as the switching off process.

4) Typical Timing Diagrams



5) Error Detection

The function block monitors the discrepancy time between Channel A and B, when switching to TRUE and also when switching to FALSE.

6) Error Behavior

S_EquivalentOut is set to FALSE. Error is set to TRUE. DiagCode indicates the Error states. There is no Reset defined as an input coupled with the reset of an error. If an error occurs in the inputs, a new set of inputs with correct S_EquivalentOut must be able to reset the error flag. (Example: if a switch is faulty and replaced, using the switch again results in a correct output)

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Error 1	Discrepancy time elapsed in state 8004. Ready = TRUE S_EquivalentOut = FALSE Error = TRUE
C002	Error 2	Discrepancy time elapsed in state 8014. Ready = TRUE S_EquivalentOut = FALSE Error = TRUE
C003	Error 3	Discrepancy time elapsed in state 8005. Ready = TRUE S_EquivalentOut = FALSE Error = TRUE

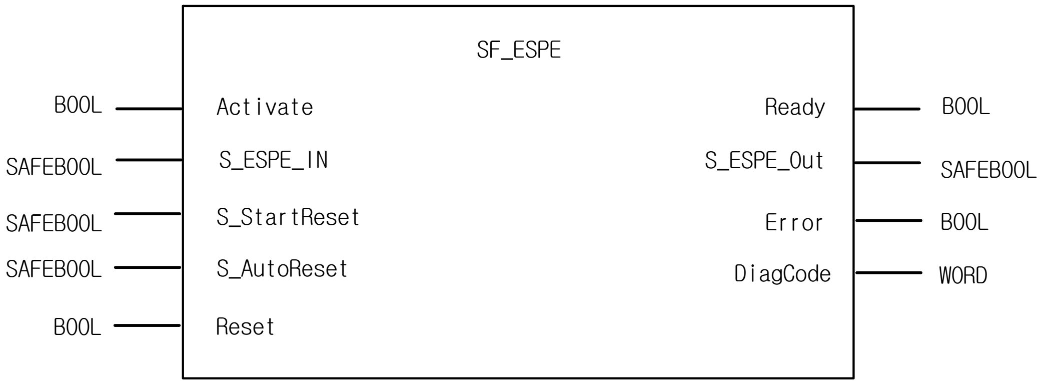
8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_EquivalentOut = FALSE Error = FALSE
8001	Init	An activation has been detected by the FB and the FB is now activated. Ready = TRUE S_EquivalentOut = FALSE Error = FALSE
8000	Safety Output Enabled	The inputs switched to TRUE in equivalent mode. Ready = TRUE S_EquivalentOut = TRUE Error = FALSE
8004	Wait for Channel B	Channel A has been switched to TRUE - waiting for Channel B; discrepancy timer started. Ready = TRUE S_EquivalentOut = FALSE Error = FALSE
8014	Wait for Channel A	Channel B has been switched to TRUE - waiting for Channel A; discrepancy timer started. Ready = TRUE S_EquivalentOut = FALSE Error = FALSE
8005	From Active Wait	One channel has been switched to FALSE; waiting for the second channel to be switched to FALSE, discrepancy timer started. Ready = TRUE S_EquivalentOut = FALSE Error = FALSE

15.2.5 SF_ESPE

1) Overview

This function block is a safety-related function block for monitoring electro-sensitive protective equipment (ESPE).



2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_ESPE_In	SAFEBOOL	0	Safety demand input. FALSE: ESPE actuated, demand for safety-related response. TRUE: ESPE not actuated, no demand for safety-related response. Safety control system must be able to detect a very short interruption of the sensor (which is specified in 61496-1: minimum 80 ms), when the ESPE is used in applications as a trip device
	S_StartReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when PES is started (warm or cold). TRUE: Automatic reset when PES is started (warm or cold). This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.

Type	Name	Data Type	Initial Value	Description
Input	S_AutoReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when emergency stop button is released. TRUE: Automatic reset when emergency stop button is released. This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
	Reset	BOOL	0	Reset
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_ESPE_OUT	SAFEBOOL	0	Output for the safety-related response. FALSE: Safety output disabled. Demand for safety-related response (e.g., reset required or internal errors active). TRUE: Safety output enabled. No demand for safety-related response.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more then 16 codes.

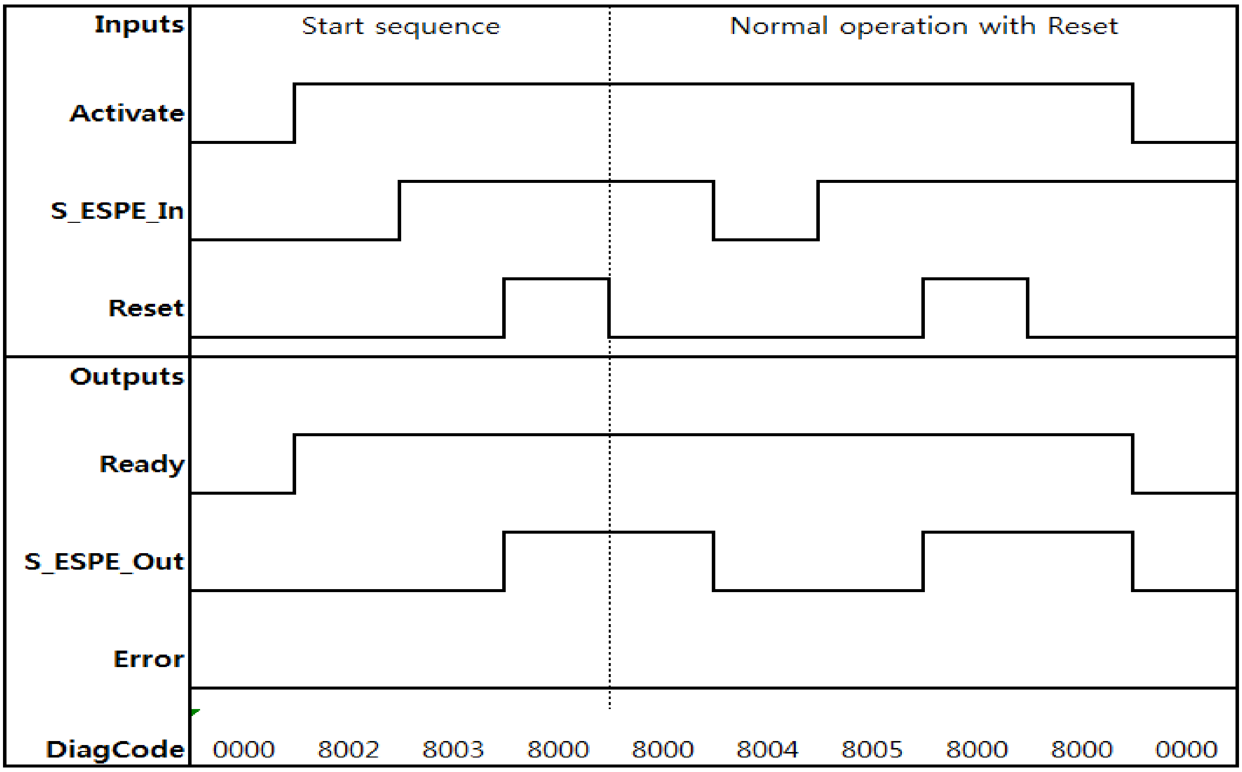
3) Functional Description

This function block is a safety-related function block for monitoring electro-sensitive protective equipment (ESPE). The function is identical to SF_EmergencyStop. The S_ESPE_Out output signal is set to FALSE as soon as the S_ESPE_In input is set to FALSE. The S_ESPE_Out output signal is set to TRUE only if the S_ESPE_In input is set to TRUE and a reset occurs. The enable reset depends on the defined S_StartReset, S_AutoReset, and Reset inputs.

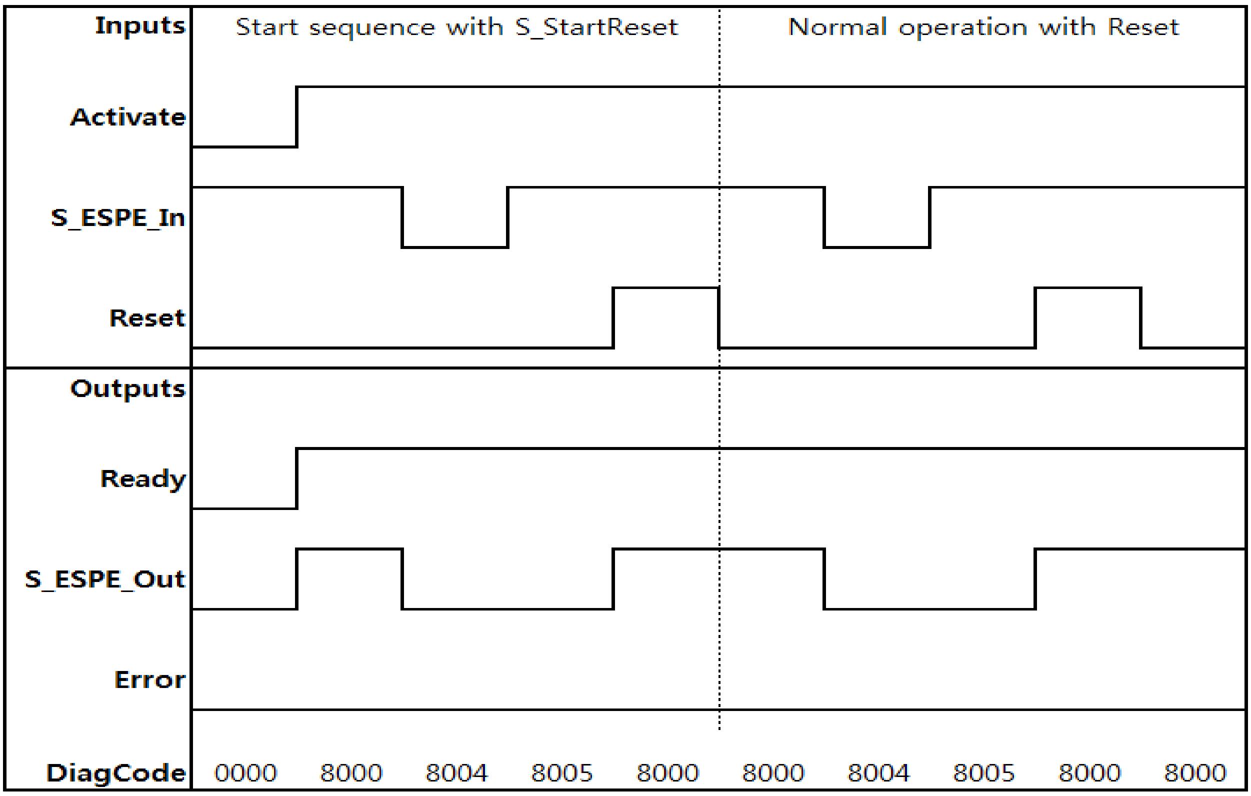
If S_AutoReset = TRUE, acknowledgment is automatic.
If S_AutoReset = FALSE, a rising trigger at the Reset input must be used to acknowledge the enable.
If S_StartReset = TRUE, acknowledgment is automatic the PES is started the first time.
If S_StartReset = FALSE, a rising trigger at the Reset input must be used to acknowledge the enable.

The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured, that no hazardous situation can occur when the PES is started.

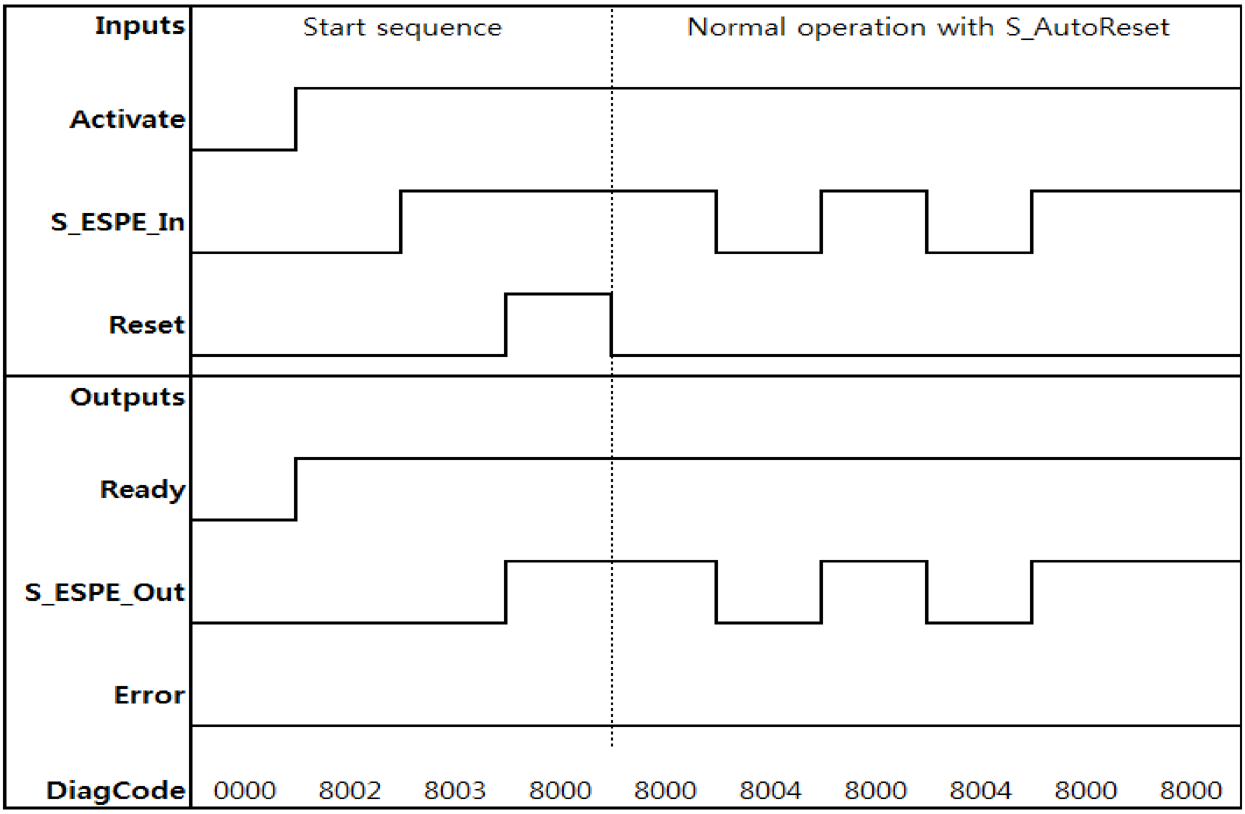
4) Typical Timing Diagrams



< S_StartReset=Off, S_AutoReset=Off >



< S_StartReset=On, S_AutoReset=Off >



< S_StartReset=Off, S_AutoReset=On >

5) Error Detection

The function block detects a static TRUE signal at Reset input.

6) Error Behavior

S_ESPE_Out is set to FALSE. In case of a static TRUE signal at the Reset input, the DiagCode output indicates the relevant error code and the Error output is set to TRUE.

To leave the error states, the the Reset must be set to FALSE.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Reset Error 1	Reset is TRUE while waiting for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE Error = TRUE
C002	Reset Error 2	Reset is TRUE while waiting for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE Error = TRUE

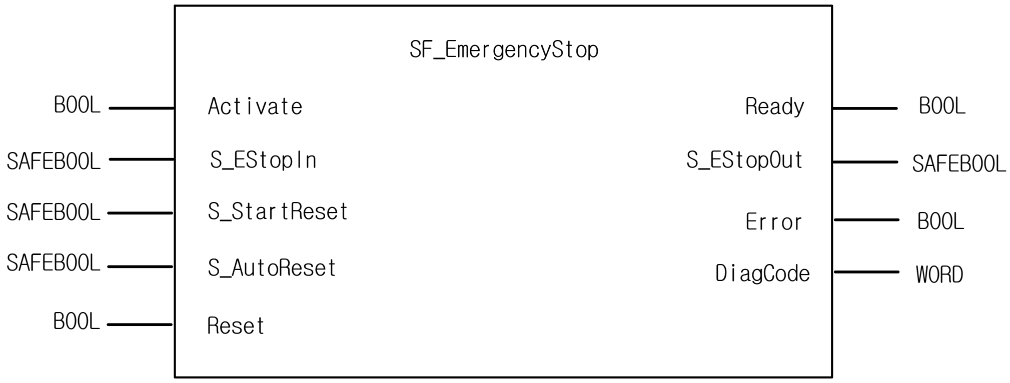
8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_ESPE_Out = FALSE Error = FALSE
8001	Init	Activation is TRUE. The function block was enabled. Check if S_StartReset is required. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8002	Wait for S_ESPE_In 1	Activation is TRUE. Check if Reset is FALSE and wait for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8003	Wait for Reset 1	Activation is TRUE. S_ESPE_In = TRUE. Wait for rising trigger of Reset. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8004	Wait for S_ESPE_In 2	Activation is TRUE. Safety demand detected. Check if Reset is FALSE and wait for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8005	Wait for Reset 2	Activation is TRUE. S_ESPE_In = TRUE. Check for S_AutoReset or wait for rising trigger of Reset. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8000	Safety Output Enabled	Activation is TRUE. S_ESPE_In = TRUE. Functional mode with S_ESPE_Out = TRUE. Ready = TRUE S_ESPE_Out = TRUE Error = FALSE

15.2.6 SF_ESTOP

1) Overview

This function block is a safety-related function block for monitoring an emergency stop button. This FB can be used for emergency switch off functionality (stop category 0), or - with additional peripheral support - as emergency stop.



2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_EStopIn	SAFEBOOL	0	Safety demand input. FALSE: Demand for safety-related response (e.g., emergency stop button is engaged). TRUE: No demand for safety-related response (e.g., emergency stop button not engaged).
	S_StartReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when PES is started (warm or cold). TRUE: Automatic reset when PES is started (warm or cold). This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.

Type	Name	Data Type	Initial Value	Description
Input	S_AutoReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when emergency stop button is released. TRUE: Automatic reset when emergency stop button is released. This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
	Reset	BOOL	0	Reset
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_EStopOut	SAFEBOOL	0	Output for the safety-related response. FALSE: Safety output disabled. Demand for safety-related response (e.g., emergency stop button engaged, reset required or internal errors active) TRUE: Safety output enabled. No demand for safety-related response (e.g., emergency stop button not engaged, no internal errors active).
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more then 16 codes.

3) Functional Description

The S_EStopOut enable signal is reset to FALSE as soon as the S_EStopIn input is set to FALSE. The S_EStopOut enable signal is reset to TRUE only if the S_EStopIn input is set to TRUE and a reset occurs. The enable reset depends on the defined S_StartReset, S_AutoReset, and Reset inputs.

If S_AutoReset = TRUE, acknowledgment is automatic.

If S_AutoReset = FALSE, a rising trigger at the Reset input must be used to acknowledge the enable.

If S_StartReset = TRUE, acknowledgment is automatic the fist time the PES is started.

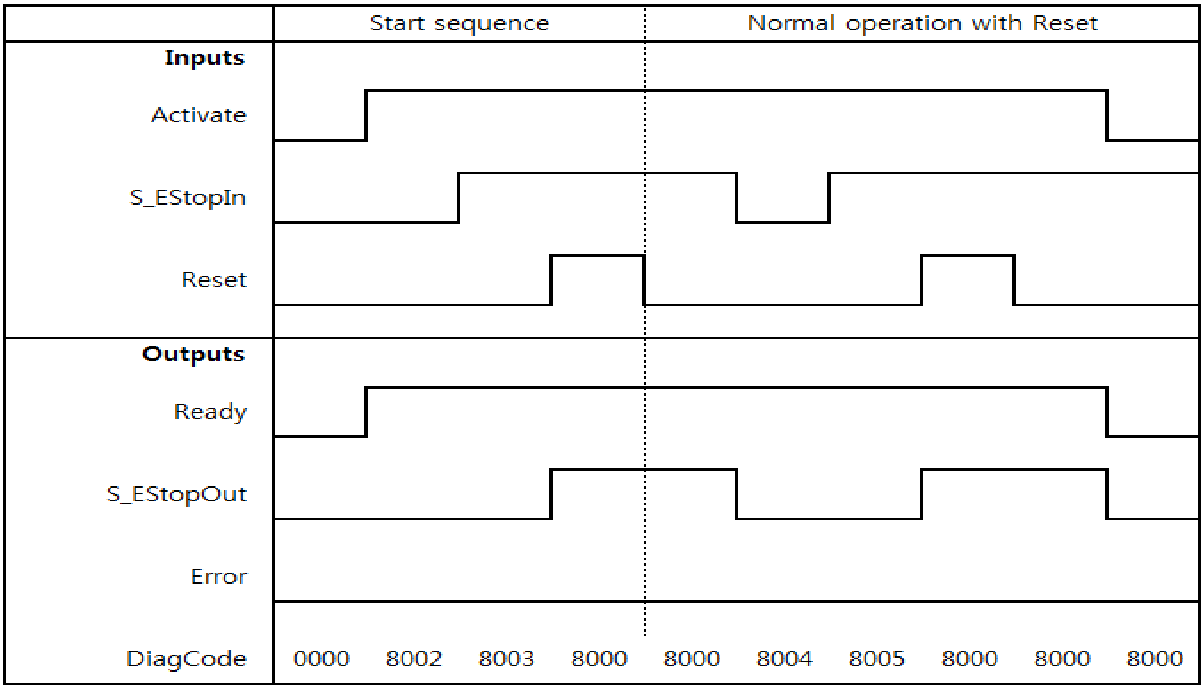
If S_StartReset = FALSE, a rising trigger at the Reset input must be used to acknowledge the enable.

The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

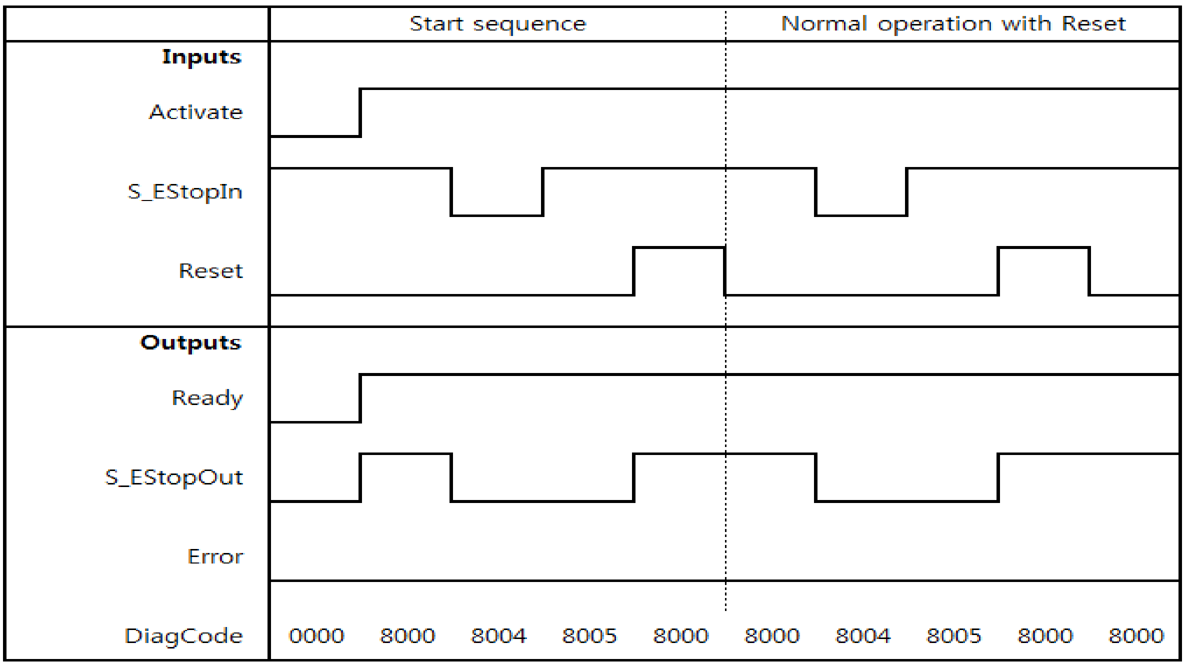
SF_EmergencyStop can be used to monitor both single and two-channel emergency stop buttons. For example, for twochannel applications, the additional function blocks SF_Equivalent can be used to detect whether the contact synchronization has been exceeded. The category classification in accordance with EN 954-1 will depend on the final elements that are used.

The SF_EmergencyStop automatically detects a static TRUE on Reset. Further error detection, e.g., wire break, short circuit depends on the dedicated hardware that is used.

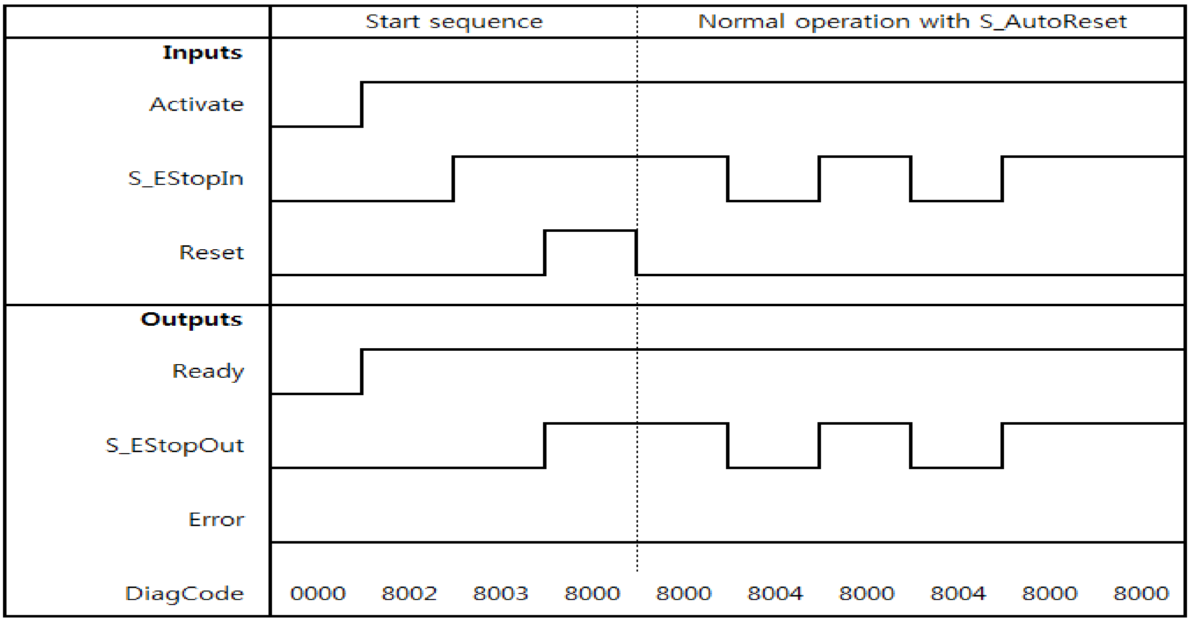
4) Typical Timing Diagrams



< S_StartReset=Off, S_AutoReset=Off >



< S_StartReset=On, S_AutoReset=Off >



< S_StartReset=Off, S_AutoReset=On >

5) Error Detection

The function block detects a static TRUE signal at Reset input.

6) Error Behavior

S_EStopOut is set to FALSE. In case of a static TRUE signal at the Reset input, the DiagCode output indicates the relevant error code and the Error output is set to TRUE.

To leave the error states, the Reset must be set to FALSE.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Reset Error 1	Reset is TRUE while waiting for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE Error = TRUE
C002	Reset Error 2	Reset is TRUE while waiting for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE Error = TRUE

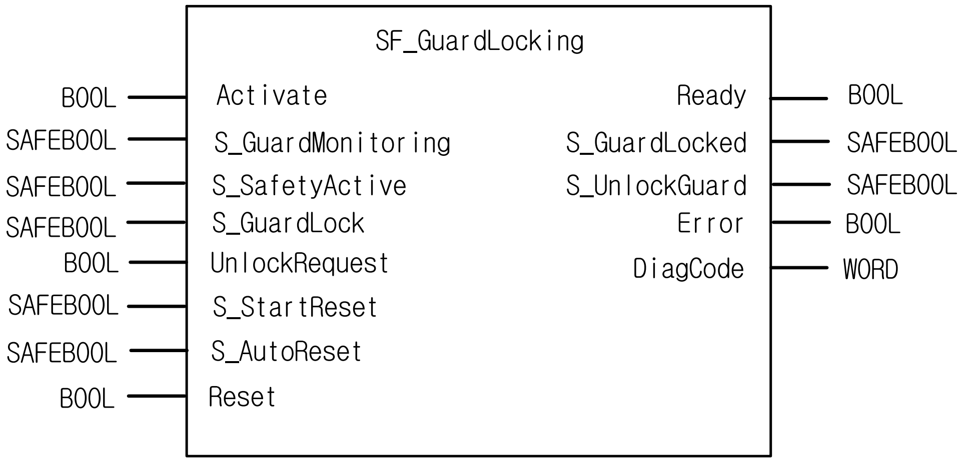
8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_EStopOut = FALSE Error = FALSE
8001	Init	Activation is TRUE. The function block was enabled. Check if S_StartReset is required. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8002	Wait for S_EstopIn 1	Activation is TRUE. Check if Reset is FALSE and wait for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8003	Wait for Reset 1	Activation is TRUE. S_EStopIn = TRUE. Wait for rising trigger of Reset. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8004	Wait for S_EstopIn 2	Activation is TRUE. Safety demand detected. Check if Reset is FALSE and wait for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8005	Wait for Reset 2	Activation is TRUE. S_EStopIn = TRUE. Check for S_AutoReset or wait for rising trigger of Reset. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8000	Safety Output Enabled	Activation is TRUE. S_EStopIn = TRUE. Functional mode with S_EStopOut = TRUE. Ready = TRUE S_EStopOut = TRUE Error = FALSE

15.2.7 SF_GUARDLOCKING

1) Overview

This FB controls an entrance to a hazardous area via an interlocking guard with guard locking (“four state interlocking”)



2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_GuardMonitoring	SAFEBOOL	0	Variable. Monitors the guard interlocking. FALSE: Guard open. TRUE: Guard closed.
	S_SafetyActive	SAFEBOOL	0	Status of the hazardous area (EDM), e.g., based on speed monitoring or safe time off delay. FALSE: Machine in "non-safe" state. TRUE: Machine in safe state.
	S_GuardLock	SAFEBOOL	0	Status of the mechanical guard locking. FALSE: Guard is not locked. TRUE: Guard is locked.
	UnlockRequest	BOOL	0	Operator intervention – request to unlock the guard. FALSE: No request. TRUE: Request made.

Type	Name	Data Type	Initial Value	Description
Input	S_StartReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when PES is started (warm or cold). TRUE: Automatic reset when PES is started (warm or cold). This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
	S_AutoReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when emergency stop button is released. TRUE: Automatic reset when emergency stop button is released. This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
	Reset	BOOL	0	Reset
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_GuardLocked	SAFEBOOL	0	Interface to hazardous area which must be stopped. FALSE: No safe state. TRUE: Safe state.
	S_UnlockGuard	SAFEBOOL	0	Signal to unlock the guard. FALSE: Close guard. TRUE: Unlock guard.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more then 16 codes.

3) Functional Description

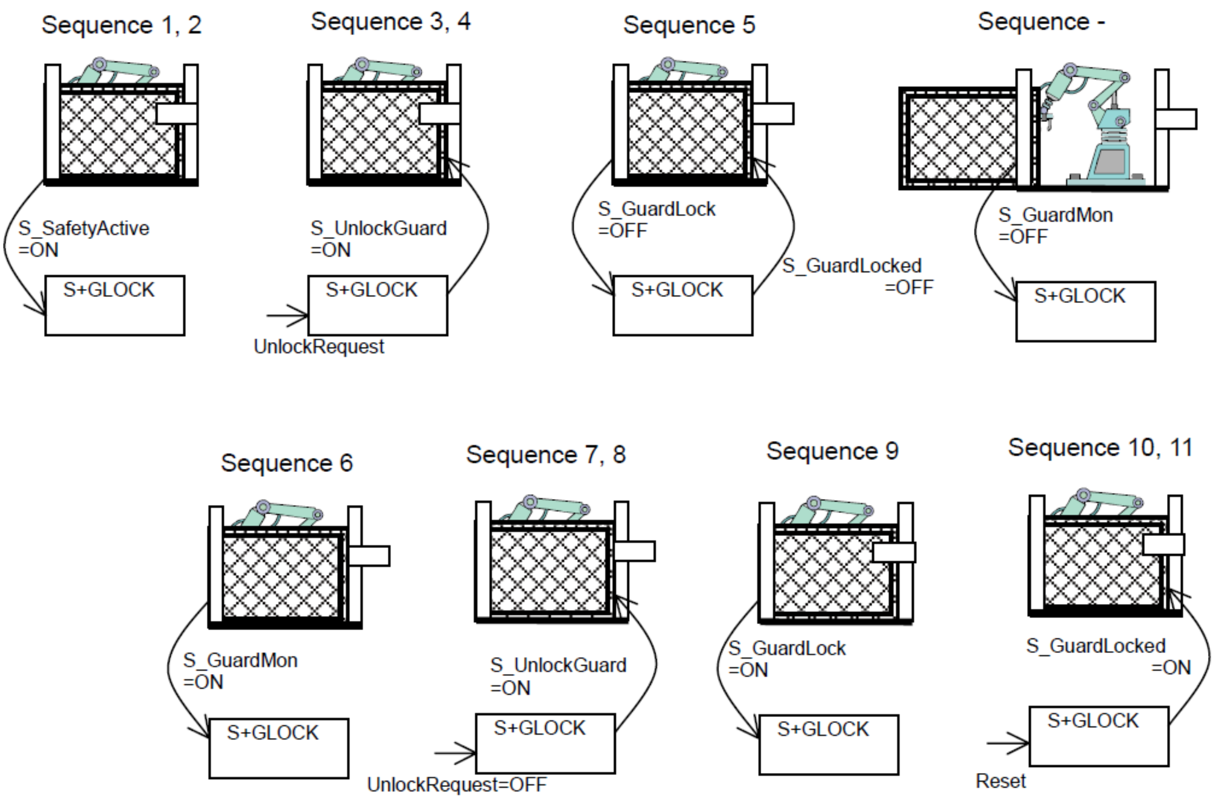
The function controls the guard lock and monitors the position of the guard and the lock. This function block can be used with a mechanical locked switch.

The operator requests to get access to the hazardous area. The guard can only be unlocked when the hazardous area is in a safe state.The guard can be locked if the guard is closed. The machine can be started when the guard is closed and the guard is locked. An open guard or unlocked guard will be detected in the event of a safety-critical situation.

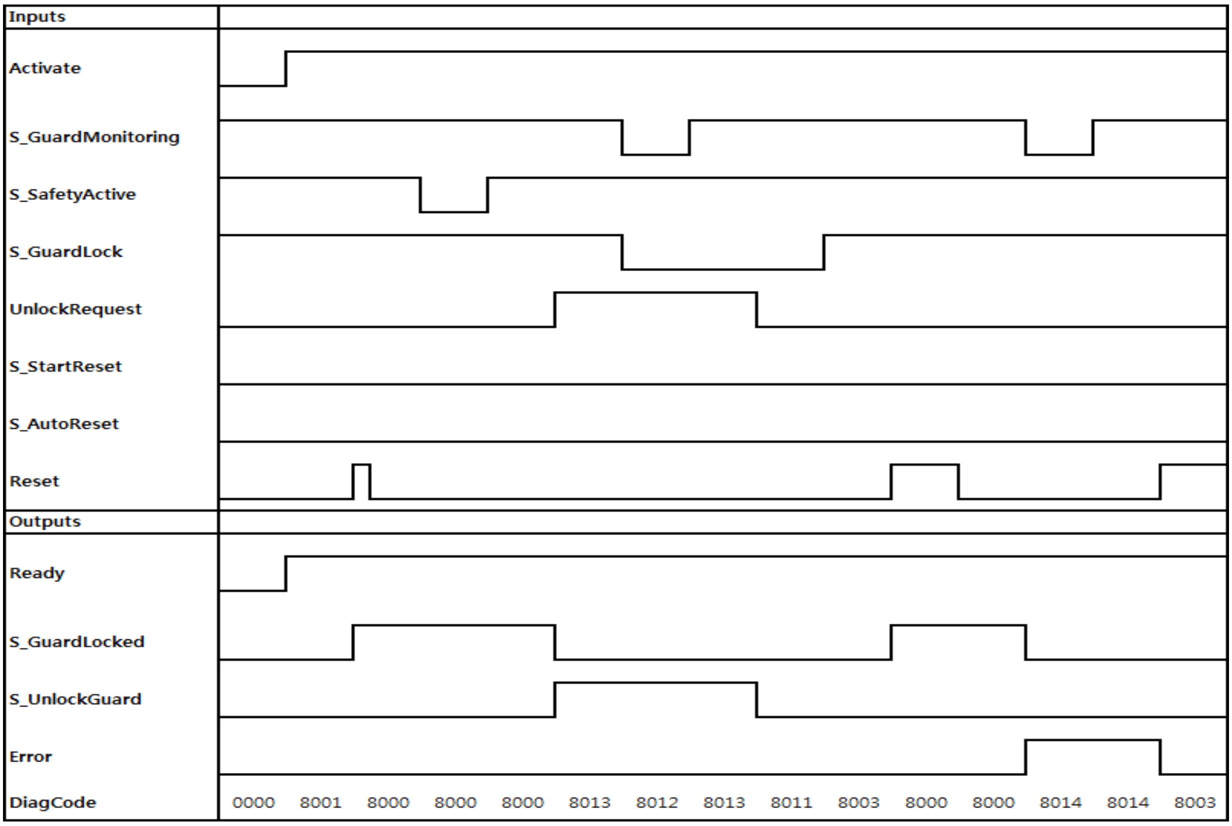
The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Operation Sequence

NO	Position	Operation
1	External	Request to get the hazardous area to a safe state - not part of this FB
2	In	Feedback from applicable hazardous area that it is in a safe state (via S_SafetyActive)
3	In	Operator request to unlock the guard (via UnlockRequest)
4	Out	Enable guard to be opened (via S_UnlockGuard)
5	In	Guard unlocked (via S_GuardLock). Guard can be opened now. (S_GuardLocked = FALSE)
-	-	Operator opens the guard
6	In	Monitoring of status guard via S_GuardMonitoring – signals when guard is closed again
7	In	Feedback from operator to restart the hazardous area (Reset)
8	Out	Lock guard guard (S_UnlockGuard)
9	In	Check if guard is locked (S_GuardLock)
10	Out	Hazardous area can operate again (S_GuardLocked = TRUE)
11	Extern	Restart the operation in the hazardous area



4) Typical Timing Diagrams



5) Error Detection

Static signals are detected at Reset. Errors are detected at the Guard switches.

6) Error Behavior

In the event of an error the S_GuardLocked and S_UnlockGuard outputs are set to FALSE, the DiagCode output indicates the relevant error code, and the Error output is set to TRUE.
An error must be acknowledged by a rising trigger at the Reset input.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Reset Error1	Static Reset detected in state 8001. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = TRUE
C002	Reset Error2	Static Reset detected in state C004. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = TRUE
C003	Reset Error3	Static Reset detected in state 8011. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = TRUE
C004	Safety Lost	Safety lost, guard opened or guard unlocked. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = TRUE

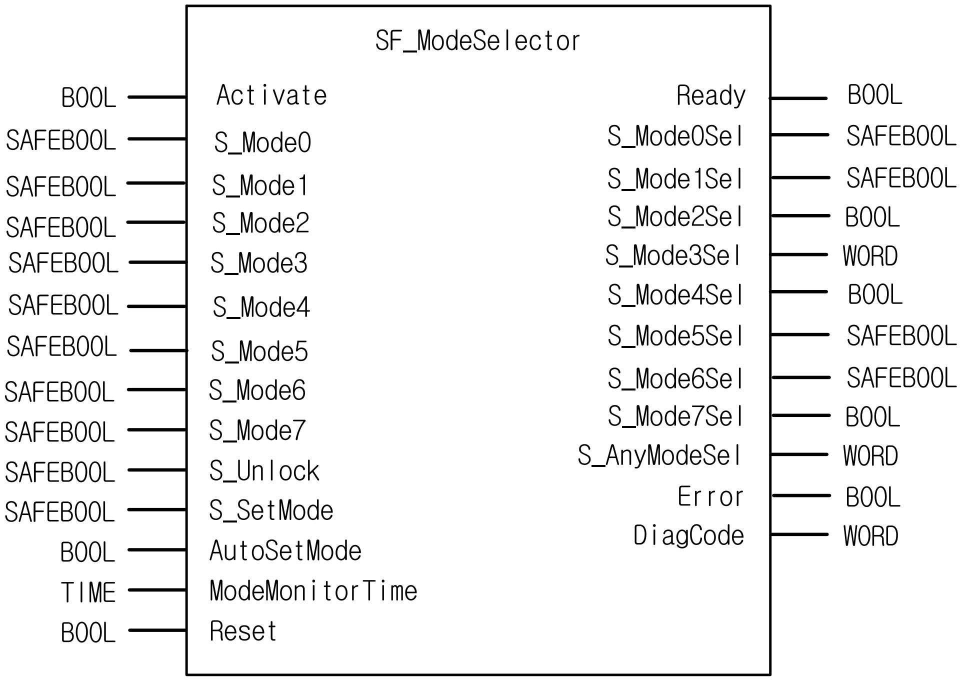
8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE
8000	Guard Closed and Locked	Guard is locked. Ready = TRUE S_GuardLocked = TRUE S_UnlockGuard = FALSE Error = FALSE
8001	Init	Function block was activated and initiated. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE
8003	Wait for Reset	Door is closed and locked, now waiting for operator reset Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE
8011	Wait for Operator	Waiting for operator to either unlock request or reset. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE
8012	Guard Open and Unlocked	Lock is released and guard is open. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE Error = FALSE
8013	Guard Closed but Unlocked	Lock is released but guard is closed. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE Error = FALSE
8014	Safety Return	Return of S_SafetyActive signal, now waiting for operator acknowledge. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE

15.2.8 SF_MODESEL

1) Overview

This function block selects the system operation mode, such as manual, automatic, semi-automatic, etc.



2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_ModeX (X = 0~7)	SAFEBOOL	0	Input X from mode selector switch FALSE: Mode X is not requested by operator. TRUE: Mode X is requested by operator.
	S_Unlock	SAFEBOOL	0	Locks the selected mode FALSE: The actual S_ModeXSel output is locked therefore a change of any S_ModeX input does not lead to a change in the S_ModeXSel output even in the event of a rising edge of Set-Mode. TRUE: The selected S_ModeXSel is not locked; a mode selection change is possible.
	S_SetMode	SAFEBOOL	0	Sets the selected mode Operator acknowledges the setting of a mode. Any change to new S_ModeX = TRUE leads to S_AnyModeSel/S_ModeXSel = FALSE, only a rising SetMode trigger then leads to new S_ModeXSel = TRUE.

Type	Name	Data Type	Initial Value	Description
Input	AutoSetMode	BOOL	0	Parameterizes the acknowledgement modeFALSE: A change in mode must be acknowledged by the operator via SetMode. TRUE: A valid change of the S_ModeX input to another S_ModeX automatically leads to a change in S_ModeXSel without operator acknowledgment via SetMode (as long as this is not locked by S_Unlock).
	ModeMonitorTime	TIME	T#0	Maximum permissible time for changing the selection input.
	Reset	BOOL	0	Reset
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_ModeXSel (X = 0~7)	SAFEBOOL	0	Indicates that mode X is selected and acknowledged. FALSE: Mode X is not selected or not active. TRUE: Mode X is selected and active.
	S_AnyModeSel	SAFEBOOL	0	Indicates that any of the 8 modes is selected and acknowledged. FALSE: No S_ModeX is selected. TRUE: One of the 8 S_ModeX is selected and active
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more then 16 codes.

3) Functional Description

This function block selects the system operation mode, such as manual, automatic, semi-automatic, etc. On controller startup, it should be assumed that the machine is in safe mode. On machine startup, the transition to the mode set by the mode selector switch must be initiated by a function block input (e.g., machine START button).

The default state following activation of the FB is the ModeChanged state. This is also the safe state of the FB, where all S_ModeXSel and S_AnyModeSel are FALSE.

If the FB is in the ModeChanged state:

- The new S_ModeX input must be acknowledged by a rising S_SetMode trigger (if AutoSetMode = FALSE), which leads to a new S_ModeXSel output.
- The new S_ModeX input automatically leads to a new S_ModeXSel output (if AutoSetMode = TRUE).
- Such a transition from state 8005 to 8000 is only valid, if one S_ModeX input is TRUE. As long as all S_ModeX are FALSE, the FB remains in state 8005, even if the S_SetMode triggers.

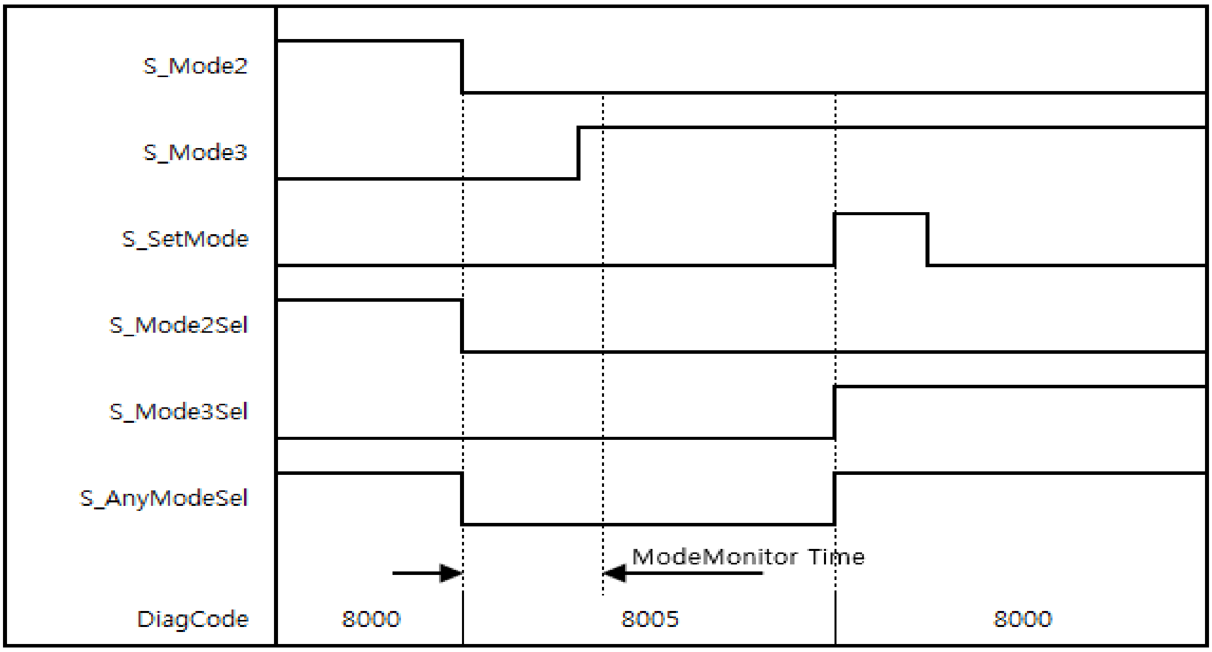
The transition from the ModeChanged to ModeSelected state, i.e., S_SetMode set by the operator, is not monitored by a timer.

If the FB is in the ModeSelected state, the simultaneous occurrence of a new S_ModeX input (higher priority) and the NOT S_Unlock signal (lower priority) leads to the ModeChanged state.

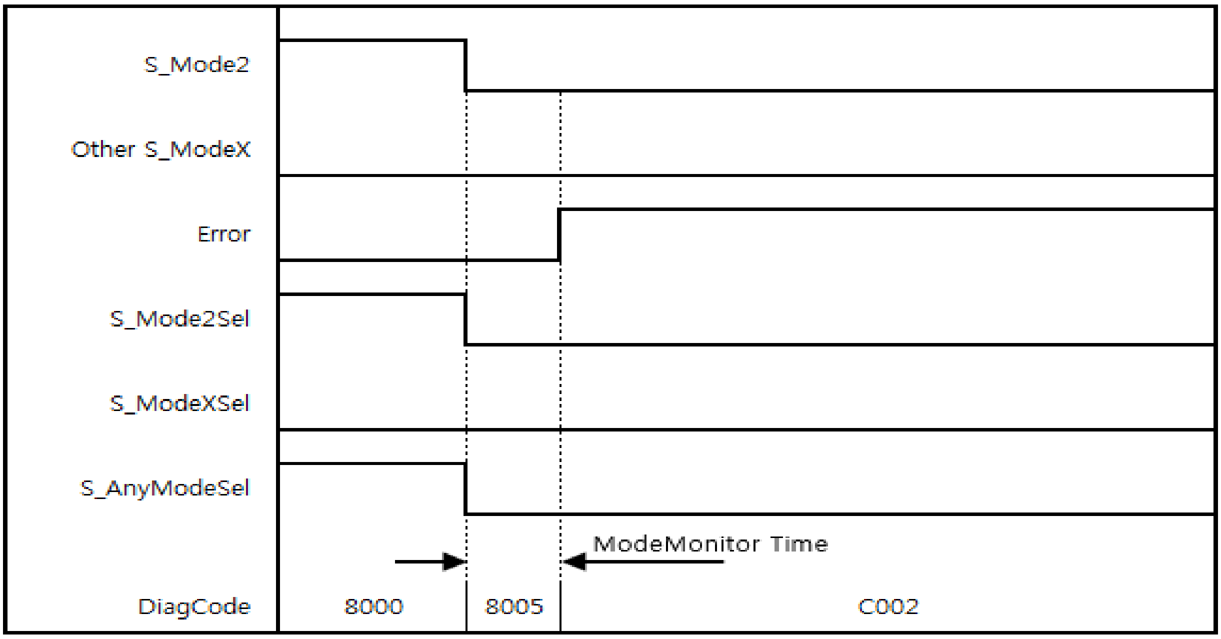
The S_ModeX input parameters, which are not used for mode selection, should be called with the default value FALSE to

simplify program verification.
The AutoSetMode input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

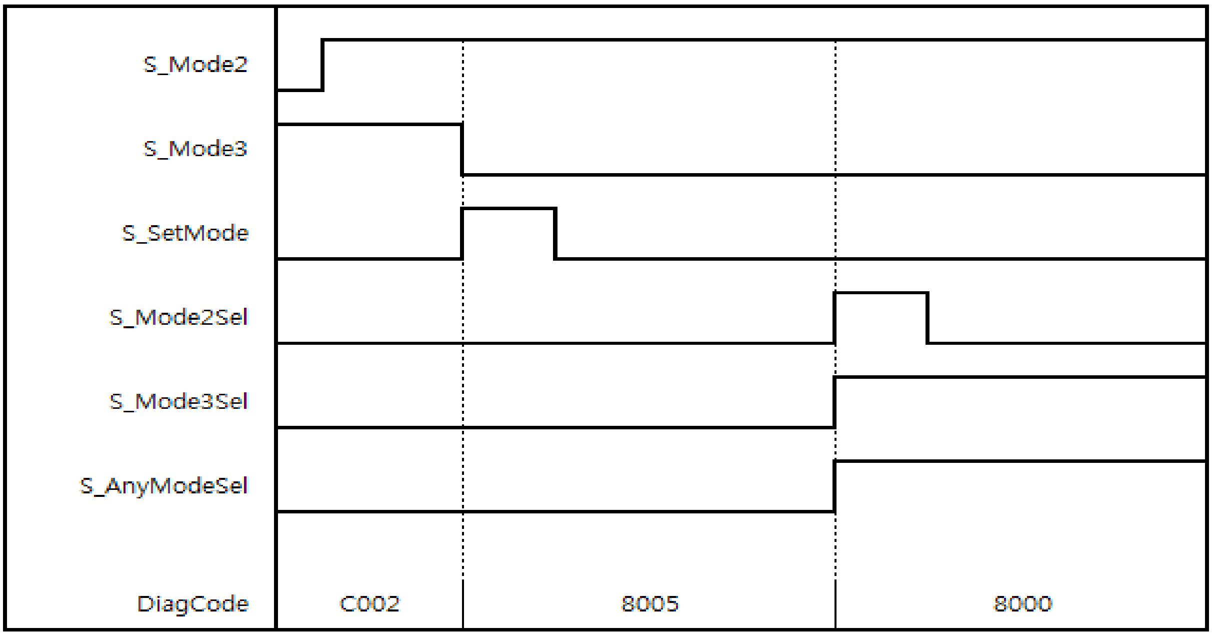
4) Typical Timing Diagrams



< Timing diagram for SF_ModeSelector, valid change in Mode input with acknowledgment>



< Timing diagram for SF_ModeSelector, error condition 2 at Mode inputs >



< Timing diagram for SF_ModeSelector, reset of error condition >

5) Error Detection

The FB detects whether none of the mode inputs is selected. This invalid condition is detected after ModeMonitorTime has elapsed:

- Which restarts with each falling trigger of an S_ModeX switched mode input
- Which is then in the ModeChanged state following activation of the FB

In contrast, the FB directly detects whether more than one S_ModeX mode input is selected at the same time.

A static reset condition is detected when the FB is either in Error state C001 or C002.

6) Error Behavior

In the event of an error, the S_ModeXSel and S_AnyModeSel outputs are set to safe state = FALSE. The DiagCode output indicates the relevant error code and the Error output is set to TRUE.

An error must be acknowledged with the rising trigger of the Reset BOOL input. The FB changes from an error state to the ModeChanged state.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Error Short-circuit	The FB detected that two or more S_ModeX are TRUE, e.g., short-circuit of cables. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
C002	Error Open-circuit	The FB detected that all S_ModeX are FALSE: The period following a falling S_ModeX trigger exceeds ModeMonitorTime, e.g., open-circuit of cables. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
C003	Reset Error 1	Static Reset signal detected in state C001. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
C004	Reset Error 2	Static Reset signal detected in state C002. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE

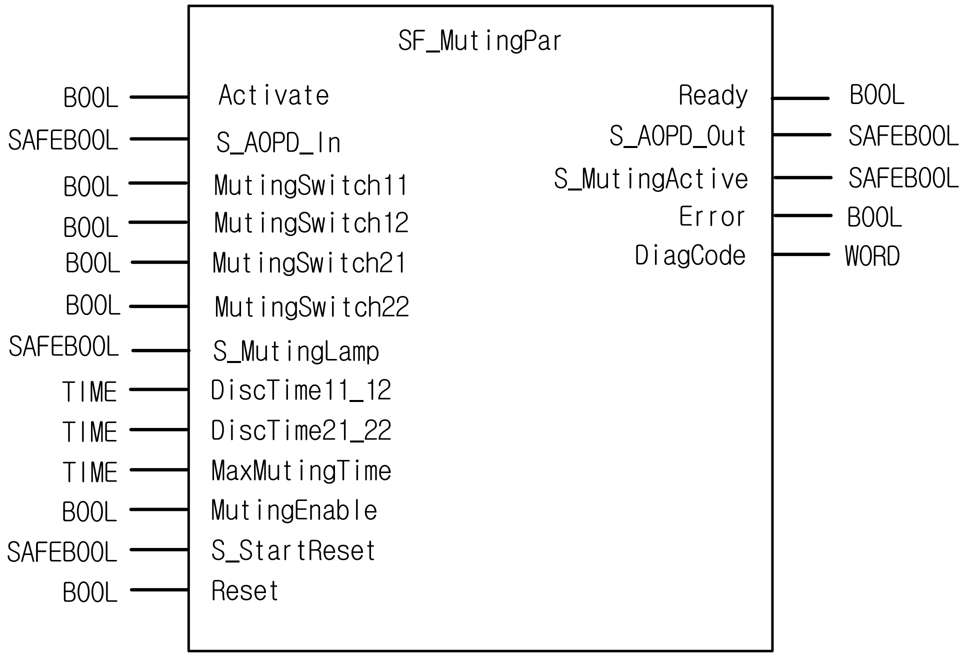
8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE Error = FALSE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
8005	ModeChanged	State after activation or when S_ModeX has changed (unless locked) or after Reset of an error state. Ready = TRUE Error = FALSE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
8000	ModeSelected	Valid mode selection, but not yet locked. Ready = TRUE Error = FALSE S_AnyModeSel = TRUE S_ModeXSel = Selected X is TRUE, others are FALSE.
8004	ModeLocked	Valid mode selection is locked. Ready = TRUE Error = FALSE S_AnyModeSel = TRUE S_ModeXSel = Selected X is TRUE, others are FALSE

15.2.9 SF_MUTINGPAR

1) Overview

Muting is the intended suppression of the safety function. In this FB, parallel muting with four muting sensors is specified.



2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_AOPD_In	SAFEBOOL	0	OSSD signal from AOPD. FALSE: Protection field interrupted. TRUE: Protection field not interrupted.
	MutingSwitch11	BOOL	0	Status of Muting sensor 11. FALSE: Muting sensor 11 not actuated. TRUE: Workpiece actuates muting sensor 11. It shall be noted in the FB manual that a SAFEBOOL must be connected instead of a BOOL depending on the safety requirements.
	MutingSwitch12	BOOL	0	Status of Muting sensor 12. FALSE: Muting sensor 12 not actuated. TRUE: Workpiece actuates muting sensor 12. It shall be noted in the FB manual that a SAFEBOOL must be connected instead of a BOOL depending on the safety requirements.

Type	Name	Data Type	Initial Value	Description
Input	MutingSwitch21	BOOL	0	Status of Muting sensor 21. FALSE: Muting sensor 21 not actuated. TRUE: Workpiece actuates muting sensor 21. It shall be noted in the FB manual that a SAFEBOOL must be connected instead of a BOOL depending on the safety requirements.
	MutingSwitch22	BOOL	0	Status of Muting sensor 22. FALSE: Muting sensor 22 not actuated. TRUE: Workpiece actuates muting sensor 22. It shall be noted in the FB manual that a SAFEBOOL must be connected instead of a BOOL depending on the safety requirements.
	S_MutingLamp	SAFEBOOL	0	Indicates operation of the muting lamp. FALSE: Muting lamp failure. TRUE: Muting lamp no failure.
	DiscTime11_12	TIME	T#0s	Constant 0..4 s; Maximum discrepancy time for MutingSwitch11 and MutingSwitch12.
	DiscTime21_22	TIME	T#0s	Constant 0..4 s; Maximum discrepancy time for MutingSwitch21 and MutingSwitch22
	MaxMutingTime	TIME	T#0s	Constant 0..10 min; Maximum time for complete muting sequence, timer started when first muting sensor is actuated.
	MutingEnable	BOOL	0	Command by the control system that enables the start of the muting function when needed by the machine cycle. After the start of the muting function, this signal can be switched off. FALSE: Muting not enabled TRUE: Start of Muting function enabled
	S_StartReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when PES is started (warm or cold). TRUE: Automatic reset when PES is started (warm or cold). This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
	Reset	BOOL	0	Reset

Type	Name	Data Type	Initial Value	Description
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_AOPD_Out	SAFEBOOL	0	Safety related output, indicates status of the muted guard. FALSE: AOPD protection field interrupted and muting not active. TRUE: AOPD protection field not interrupted or muting active.
	S_MutingActive	SAFEBOOL	0	Indicates status of Muting process. FALSE: Muting not active. TRUE: Muting active.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more then 16 codes.

3) Functional Description

Muting is the intended suppression of the safety function. This is required, e.g., when transporting the material into the danger zone without causing the machine to stop. Muting is triggered by muting sensors. The use of two or four muting sensors and correct integration into the production sequence must ensure that no persons enter the danger zone while the light curtain is muted. Muting sensors can be proximity switches, photoelectric barriers, limit switches, etc. which do not have to be failsafe. Active muting mode must be indicated by indicator lights.

There are sequential and parallel muting procedures. In this FB, parallel muting with four muting sensors was used; an explanation is provided below. The FB can be used in both directions, forward and backward. The muting should be enabled with the MutingEnable signal by the process control to avoid manipulation.

The FB input parameters include the signals of the four muting sensors (MutingSwitch11 ... MutingSwitch22), the OSSD signal from the "active opto-electronic protective device", S_AOPD_In, as well as three parameterizable times (DiscTime11_12, DiscTime21_22, and MaxMutingTime).

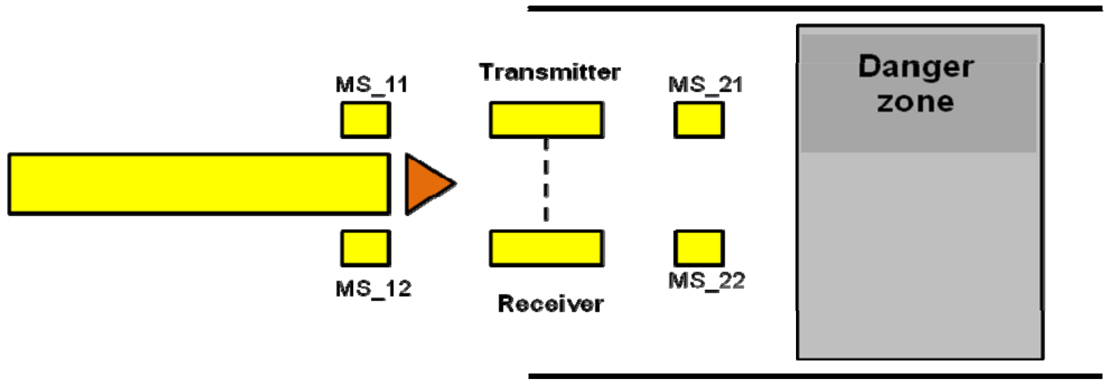
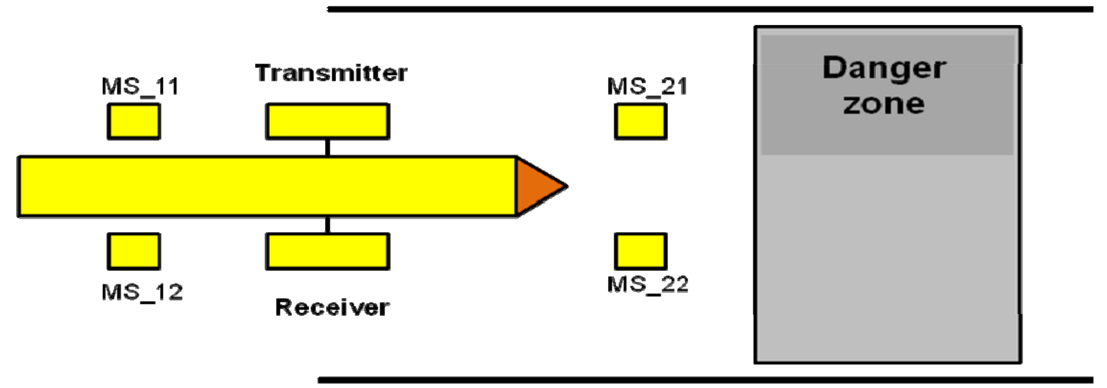
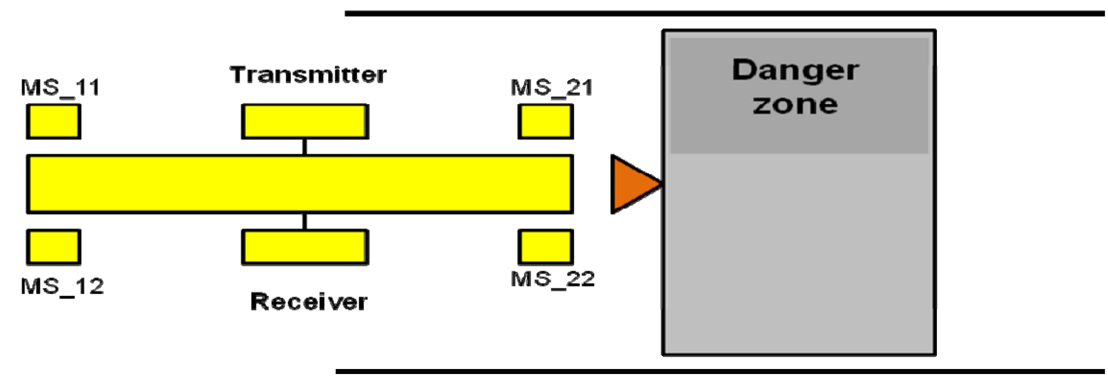
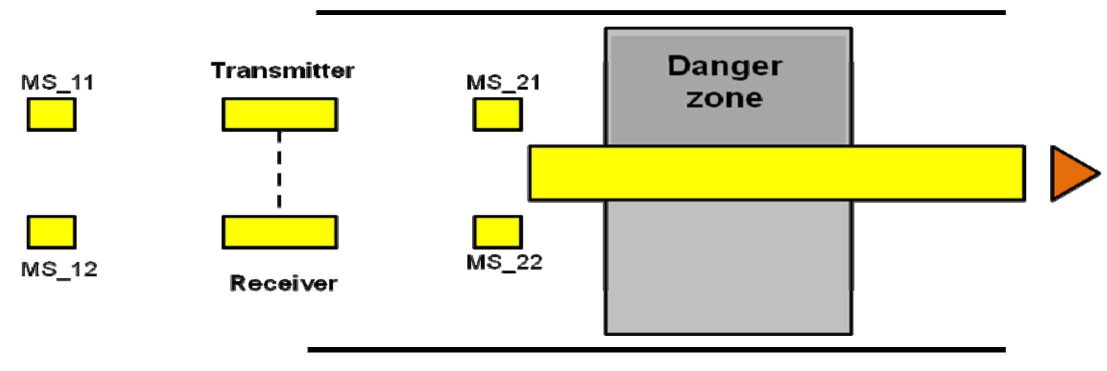
The S_StartReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Step 1: If the muting sensors MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are activated by the product within the time DiscTime11_12, muting mode is activated (S_MutingActive = TRUE).

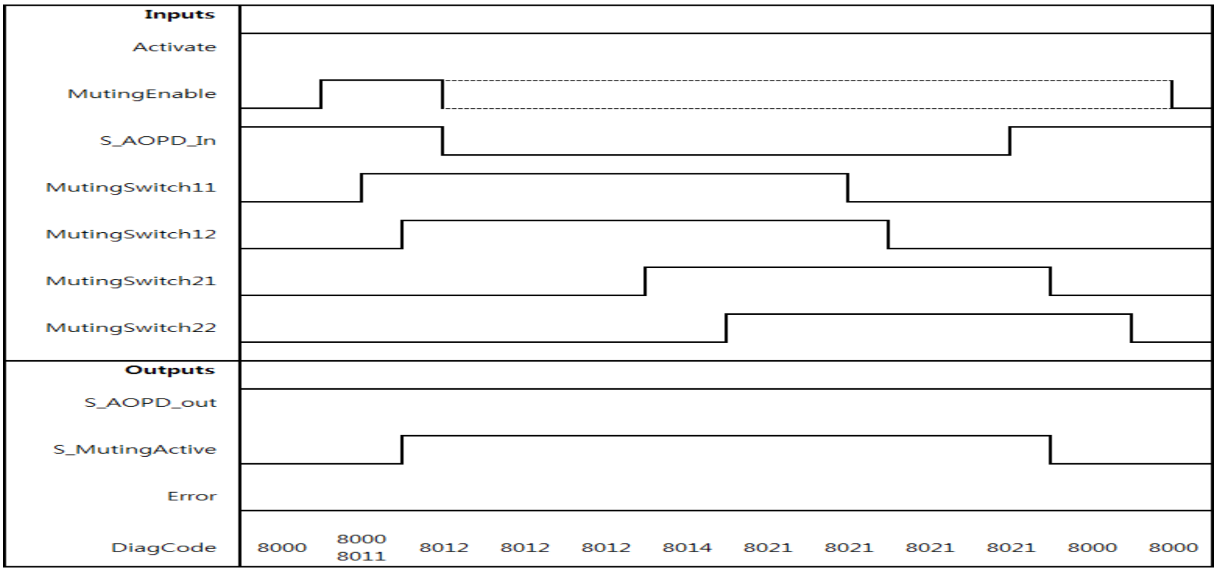
Step 2: Muting mode remains active as long as MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are activated by the product. The product may pass through the light curtain without causing a machine stop.

Step 3: Before muting sensors MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are disabled, muting sensors MutingSwitch21 (MS_21) and MutingSwitch22 (MS_22) must be activated. This ensures that muting mode remains active. The time discrepancy between switching of MutingSwitch21 and MutingSwitch22 is monitored by the time DiscTime21_22.

Step 4: Muting mode is terminated if either muting sensor MutingSwitch21 (MS_21) or MutingSwitch22 (MS_22) is disabled by the product. The maximum time for muting mode to be active is the Max-MutingTime.

No.	Figure
1	
2	
3	
4	

4) Typical Timing Diagrams



5) Error Detection

The FB detects the following error conditions:

- DiscTime11_12 and DiscTime21_22 have been set to values less than T#0s or greater than T#4s.
- MaxMutingTime has been set to a value less than T#0s or greater than T#10min.
- The discrepancy time for the MutingSwitch11/MutingSwitch12 or MutingSwitch21/MutingSwitch22 sensor pairs has been exceeded.
- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.
- Muting sensors MutingSwitch11, MutingSwitch12, MutingSwitch21, and MutingSwitch22 are activated in the wrong order.
- Muting sequence starts without being enabled by MutingEnable
- A faulty muting lamp is indicated by S_MutingLamp = FALSE.
- A static Reset condition is detected in state 8001 and 8003.

6) Error Behavior

In the event of an error, the S_AOPD_Out and S_MutingActive outputs are set to FALSE. The DiagCode output indicates the relevant error code and the Error output is set to TRUE.

A restart is inhibited until the error conditions are cleared and the Safe state is acknowledged with Reset by the operator.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Reset Error 1	Static Reset condition detected after FB activation in state 8001. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C002	Reset Error 2	Static Reset condition detected in state 8003. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE

DiagCode	State Name	State Description and Output Setting
C003	Error Muting Lamp	Error detected in muting lamp. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
CYx4	Error Muting sequence	Error detected in muting sequence state 8000, 8011, 8311, 8012, 8021, 8014, 8314, 8122, 8422, 8121, 8112, 8114 or 8414. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE Y = Status in the sequence (6 states for forward and 6 states for backward direction). C0x4 = Error occurred in state 8000 C1x4 = Error occurred in state Forward 8011 C2x4 = Error occurred in state Forward 8311 C3x4 = Error occurred in state Forward 8012 C4x4 = Error occurred in state Forward 8014 C5x4 = Error occurred in state Forward 8314 C6x4 = Error occurred in state Forward 8021 C7x4 = Error occurred in state Backward 8122 C8x4 = Error occurred in state Backward 8422 C9x4 = Error occurred in state Backward 8121 CAx4 = Error occurred in state Backward 8114 CBx4 = Error occurred in state Backward 8414 CCx4 = Error occurred in state Backward 8112 CFx4 = Muting Enable missing x = Status of the sensors when error occurred (4 bits: LSB = MS_11; MS_12; MS_21; MSB = MS_22)
C005	Parameter Error	DiscTime11_12, DiscTime21_22 or MaxMutingTime value out of range. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C006	Error Timer MaxMuting	Timing error: Active muting time (when S_MutingActive = TRUE) exceeds MaxMutingTime. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE

DiagCode	State Name	State Description and Output Setting
C007	Error Timer MS11_12	Timing error: Discrepancy time for switching MutingSwitch11 and MutingSwitch12 > DiscTime11_12. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C008	Error Timer MS21_22	Timing error: Discrepancy time for switching MutingSwitch21 and MutingSwitch22 > DiscTime21_22. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE

8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8000	AOPD Free	Muting not active and no safety demand from AOPD. If timers from subsequent muting are still running, they are stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8001	Init	Function block has been activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8002	Safety Demand AOPD	Safety demand detected by AOPD, muting not active. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8003	Wait for Reset	Safety demand or errors have been detected and are now cleared. Operator acknowledgment by Reset required. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8005	Safe	Safety function activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8011	Muting Forward Start 1	Muting forward sequence is in starting phase after rising trigger of MutingSwitch 11. Monitoring of DiscTime11_12 is activated. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8311	Muting Forward Start 2	Muting forward sequence is in starting phase after rising trigger of MutingSwitch 12. Monitoring of DiscTime11_12 is activated. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE

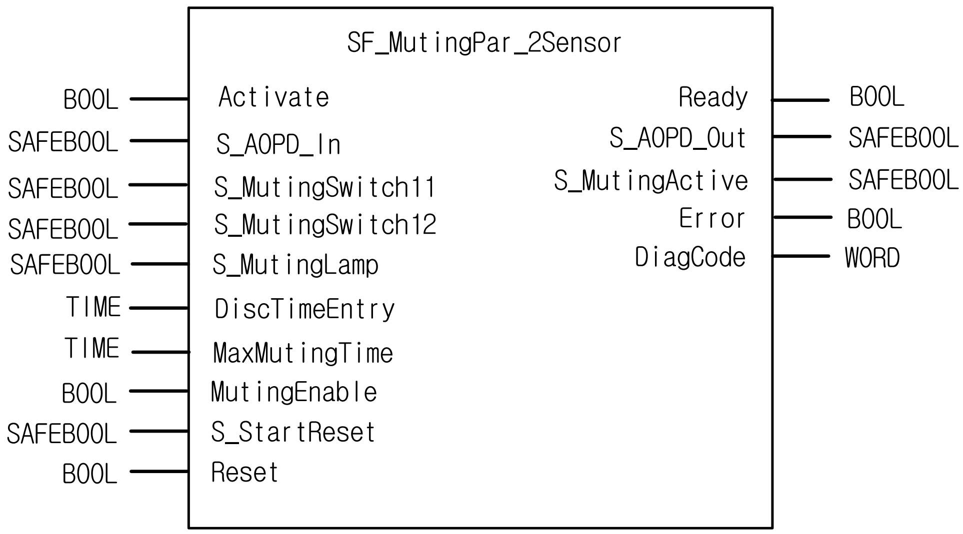
DiagCode	State Name	State Description and Output Setting
8012	Muting Forward Active 1	Muting forward sequence is active either: - After rising trigger of the second entry MutingSwitch 12 or 11 has been detected. - When both MutingSwitch 11 and 12 have been actuated in the same cycle. Monitoring of DiscTime11_12 is stopped. Monitoring of MaxMuting-Time is activated, when transition came directly from state 8000. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8014	Muting Forward Step 1	Muting forward sequence is active. MutingSwitch21 is the first exit switch actuated. Monitoring of DiscTime21_22 is started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8314	Muting Forward Step 2	Muting forward sequence is active. MutingSwitch22 is the first exit switch actuated. Monitoring of DiscTime21_22 is started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8021	Muting Forward Active 2	Muting forward sequence is still active. Both MutingSwitch21 and 22 are actuated, the monitoring of DiscTime21_22 is stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8122	Muting Backward Start 1	Muting backward sequence is in starting phase after rising trigger of MutingSwitch21. Monitoring of DiscTime21_22 is activated. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8422	Muting Backward Start 2	Muting backward sequence is in starting phase after rising trigger of MutingSwitch22. Monitoring of DiscTime21_22 is activated. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE

DiagCode	State Name	State Description and Output Setting
8121	Muting Backward Active 1	Muting backward sequence is active either: - After rising trigger of the second MutingSwitch 21 or 22 has been detected. - When both MutingSwitch 21 and 22 have been actuated in the same cycle. Monitoring of DiscTime21_22 is stopped. Monitoring of MaxMuting-Time is activated, when transition came directly from state 8000. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8114	Muting Backward Step 1	Muting backward sequence is active. MutingSwitch11 is the first exit switch actuated. Monitoring of DiscTime11_12 is started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8414	Muting Backward Step 2	Muting backward sequence is active. MutingSwitch12 is the first exit switch actuated. Monitoring of DiscTime11_12 is started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8112	Muting Backward Active 2	Muting backward sequence is still active. Both exit switches MutingSwitch11 and 12 are actuated, the monitoring of DiscTime11_12 is stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE

15.2.10 SF_MUTINGPAR_2SENSOR

1) Overview

Muting is the intended suppression of the safety function. In this FB, parallel muting with two muting sensors is specified..



2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_AOPD_In	SAFEBOOL	0	OSSD signal from AOPD. FALSE: Protection field interrupted. TRUE: Protection field not interrupted.
	MutingSwitch11	BOOL	0	Status of Muting sensor 11. FALSE: Muting sensor 11 not actuated. TRUE: Workpiece actuates muting sensor 11.
	MutingSwitch12	BOOL	0	Status of Muting sensor 12. FALSE: Muting sensor 12 not actuated. TRUE: Workpiece actuates muting sensor 12
	S_MutingLamp	SAFEBOOL	0	Indicates operation of the muting lamp. FALSE: Muting lamp failure. TRUE: Muting lamp no failure.
	DiscTimeEntry	TIME	T#0s	Constant 0..4 s; Max. discrepancy time for S_MutingSwitch11 and S_MutingSwitch12 entering muting gate
	MaxMutingTime	TIME	T#0s	Constant 0..10 min; Maximum time for complete muting sequence, timer started when first muting sensor is actuated.

Type	Name	Data Type	Initial Value	Description
Input	MutingEnable	BOOL	0	Command by the control system that enables the start of the muting function when needed by the machine cycle. After the start of the muting function, this signal can be switched off. FALSE: Muting not enabled TRUE: Start of Muting function enabled
	S_StartReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when PES is started (warm or cold). TRUE: Automatic reset when PES is started (warm or cold). This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
	Reset	BOOL	0	Reset
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_AOPD_Out	SAFEBOOL	0	Safety related output, indicates status of the muted guard. FALSE: AOPD protection field interrupted and muting not active. TRUE: AOPD protection field not interrupted or muting active.
	S_MutingActive	SAFEBOOL	0	Indicates status of Muting process. FALSE: Muting not active. TRUE: Muting active.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more than 16 codes.

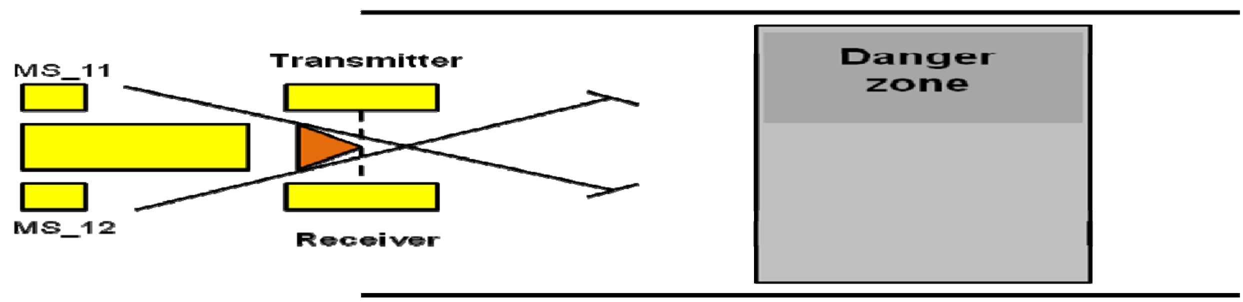
3) Functional Description

Muting is the intended suppression of the safety function. This is required, e.g., when transporting the material into the danger zone without causing the machine to stop. Muting is triggered by muting sensors. The use of two muting sensors and correct integration into the production sequence must ensure that no persons enter the danger zone while the light curtain is muted.
Muting sensors can be push buttons, proximity switches, photoelectric barriers, limit switches, etc. which do not have to be failsafe. Active muting mode must be indicated by indicator lights.

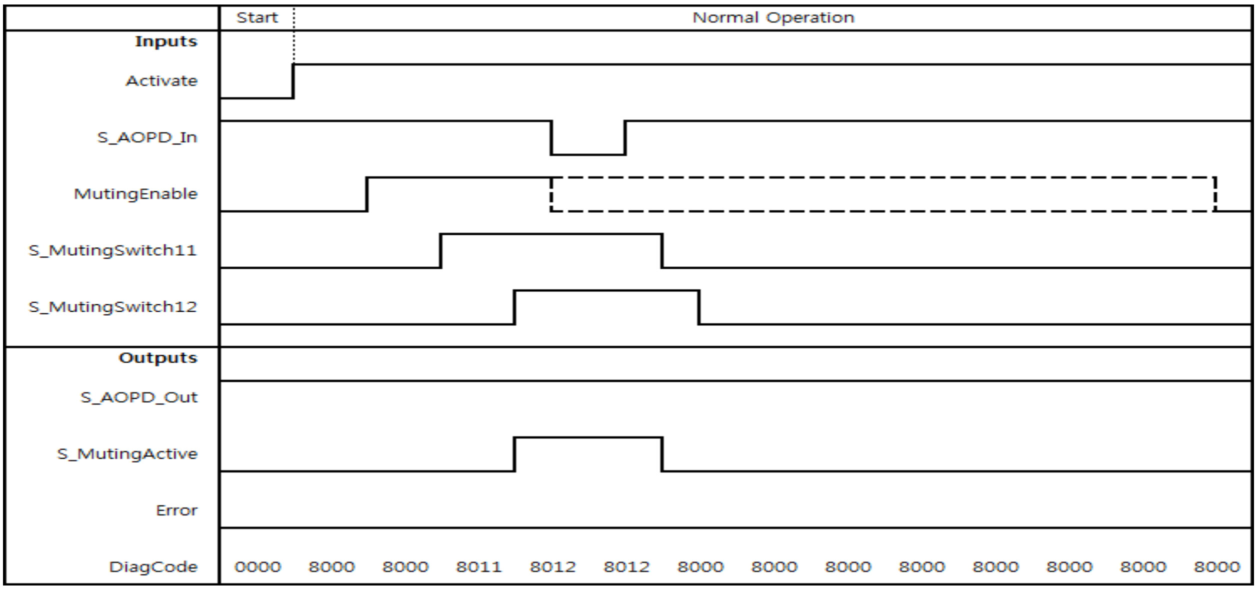
There are sequential and parallel muting procedures. In this FB, parallel muting with two muting sensors was used; an explanation is provided below. The positioning of the sensors should be as described in Annex F.7 of IEC 62046, CD 2005, as shown in Figure 48. The FB can be used in both directions, forward and backward. However, the actual direction cannot be identified. The muting should be enabled with the MutingEnable signal by the process control to avoid manipulation.

The FB input parameters include the signals of the two muting sensors (S_MutingSwitch11 and S_MutingSwitch12), the OSSD signal from the "active opto-electronic protective device", S_AOPD_In, as well as two parameterizable times (Disc-TimeEntry and MaxMutingTime).

The S_StartReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started
Step 1: If reflection light barriers are used as muting sensors, they are generally arranged diagonally. In general, this arrangement of reflection light barriers as muting sensors requires only two light barriers, and only S_MutingSwitch11 (MS_11) and S_MutingSwitch12 (MS_12) are allocated.

NO.	Figure
1	

4) Typical Timing Diagrams



5) Error Detection

The FB detects the following error conditions:

- DiscTimeEntry has been set to value less than T#0s or greater than T#4s.
- MaxMutingTime has been set to a value less than T#0s or greater than T#10min.
- The discrepancy time for the S_MutingSwitch11/S_MutingSwitch12 sensor pair has been exceeded.
- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.
- Muting sensors S_MutingSwitch11,S_MutingSwitch12 are activated in the wrong order.
- Muting sequence starts without being enabled by MutingEnable
- Static muting sensor signals.
- A faulty muting lamp is indicated by S_MutingLamp = FALSE.
- A static Reset condition is detected in state 8001 and 8003.

6) Error Behavior

In the event of an error, the S_AOPD_Out and S_MutingActive outputs are set to FALSE. The DiagCode output indicates the relevant error code and the Error output is set to TRUE.

A restart is inhibited until the error conditions are cleared and the Safe state is acknowledged with Reset by the operator.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Reset Error 1	Static Reset condition detected after FB activation in state 8001. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C002	Reset Error 2	Static Reset condition detected in state 8003. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C003	Error Muting Lamp	Error detected in muting lamp. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
CYx4	Error Muting sequence	Error detected in muting sequence state 8000, 8011, 8311. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE Y = Status in the sequence C0x4 = Error occurred in state 8000 C1x4 = Error occurred in state 8011 C2x4 = Error occurred in state 8311 CFx4 = Muting Enable missing x = Status of the sensors when error occurred (4 bits: LSB = MS_11; next to LSB = MS_12).
C005	Parameter Error	DiscTimeEntry or MaxMutingTime value out of range. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C006	Error timer MaxMuting	Timing error: Active muting time (when S_MutingActive = TRUE) exceeds MaxMutingTime. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C007	Error timer Entry	Timing error: Discrepancy time for switching S_MutingSwitch11 and S_MutingSwitch12 from FALSE to TRUE > DiscTimeEntry. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE

8) Status codes

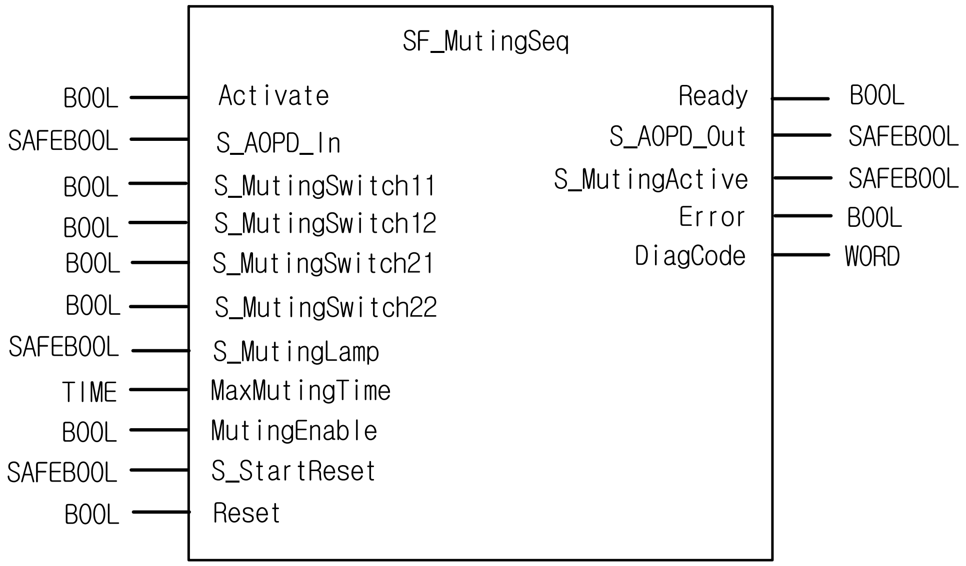
DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8000	AOPD Free	Muting not active and no safety demand from AOPD. If timers from subsequent muting are still running, they are stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8001	Init	Function block was activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8002	Safety Demand AOPD	Safety demand detected by AOPD, muting not active. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8003	Wait for Reset	Safety demand or errors have been detected and are now cleared. Operator acknowledgment by Reset required. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8005	Safe	Safety function activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8011	Muting Start 1	Muting sequence is in starting phase after rising trigger of S_MutingSwitch11. Monitoring of DiscTimeEntry is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8311	Muting Start 2	Muting sequence is in starting phase after rising trigger of S_MutingSwitch12. Monitoring of DiscTimeEntry is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE

DiagCode	State Name	State Description and Output Setting
8012	Muting Active	Muting sequence is active either: - After rising trigger of the second S_MutingSwitch 12 or 11 has been detected. - When both S_MutingSwitch 11 and 12 have been actuated in the same cycle. Monitoring of DiscTimeEntry is stopped. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE

15.2.11 SF_MUTINGSEQ

1) Overview

Muting is the intended suppression of the safety function (e.g., light barriers). In this FB, sequential muting with four muting sensors is specified.



2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_AOPD_In	SAFEBOOL	0	OSSD signal from AOPD. FALSE: Protection field interrupted. TRUE: Protection field not interrupted.
	MutingSwitch11	BOOL	0	Status of Muting sensor 11. FALSE: Muting sensor 11 not actuated. TRUE: Workpiece actuates muting sensor 11. It shall be noted in the FB manual that a SAFEBOOL must be connected instead of a BOOL depending on the safety requirements.
	MutingSwitch12	BOOL	0	Status of Muting sensor 12. FALSE: Muting sensor 12 not actuated. TRUE: Workpiece actuates muting sensor 12. It shall be noted in the FB manual that a SAFEBOOL must be connected instead of a BOOL depending on the safety requirements.

Type	Name	Data Type	Initial Value	Description
Input	MutingSwitch21	BOOL	0	Status of Muting sensor 21. FALSE: Muting sensor 21 not actuated. TRUE: Workpiece actuates muting sensor 21. It shall be noted in the FB manual that a SAFEBOOL must be connected instead of a BOOL depending on the safety requirements.
	MutingSwitch22	BOOL	0	Status of Muting sensor 22. FALSE: Muting sensor 22 not actuated. TRUE: Workpiece actuates muting sensor 22. It shall be noted in the FB manual that a SAFEBOOL must be connected instead of a BOOL depending on the safety requirements.
	S_MutingLamp	SAFEBOOL	0	Indicates operation of the muting lamp. FALSE: Muting lamp failure. TRUE: Muting lamp no failure
	MaxMutingTime	TIME	T#0s	Constant 0 .. 10 min; Maximum time for complete muting sequence, timer started when first muting sensor is actuated.
	MutingEnable	BOOL	0	Command by the control system that enables the start of the muting function when needed by the machine cycle. After the start of the muting function, this signal can be switched off. FALSE: Muting not enabled TRUE: Start of Muting function enabled
	S_StartReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when PES is started (warm or cold). TRUE: Automatic reset when PES is started (warm or cold). This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
	Reset	BOOL	0	Reset

Type	Name	Data Type	Initial Value	Description
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_AOPD_Out	SAFEBOOL	0	Safety related output, indicates status of the muted guard. FALSE: AOPD protection field interrupted and muting not active. TRUE: AOPD protection field not interrupted or muting active.
	S_MutingActive	SAFEBOOL	0	Indicates status of Muting process. FALSE: Muting not active. TRUE: Muting active.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more than 16 codes.

3) Functional Description

Muting is the intended suppression of the safety function. This is required, e.g., when transporting the material into the danger zone without causing the machine to stop. Muting is triggered by muting sensors. The use of two or four muting sensors and correct integration into the production sequence must ensure that no persons enter the danger zone while the light curtain is muted. Muting sensors can be proximity switches, photoelectric barriers, limit switches, etc. which do not have to be failsafe. Active muting mode must be indicated by indicator lights.

There are sequential and parallel muting procedures. In this FB, sequential muting with four muting sensors was used; an explanation for the forward direction of transportation is provided below. The FB can be used in both directions, forward and backward. The muting should be enabled with the MutingEnable signal by the process control to avoid manipulation. When the MutingEnable signal is not available, this input must be set to TRUE.

The FB input parameters include the signals of the four muting sensors (MutingSwitch11 ... MutingSwitch22) as well as the OSSD signal from the "active opto-electronic protective device", S_AOPD_In.

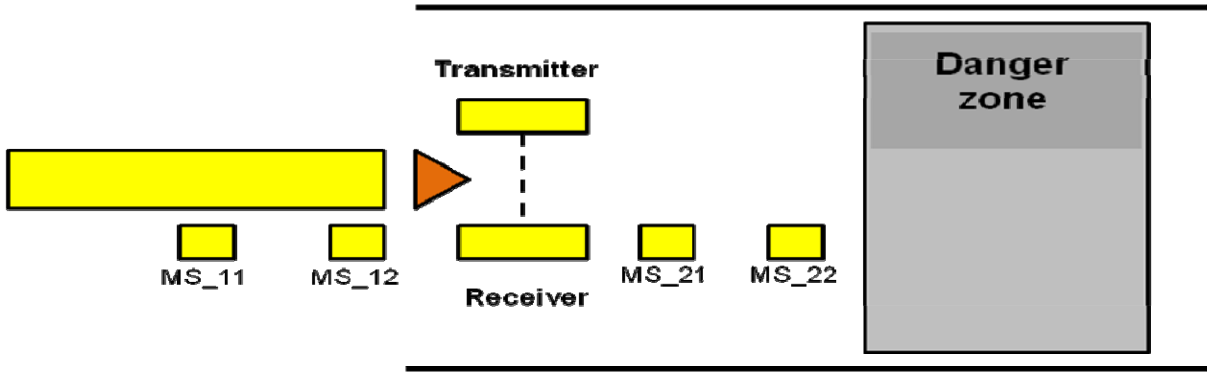
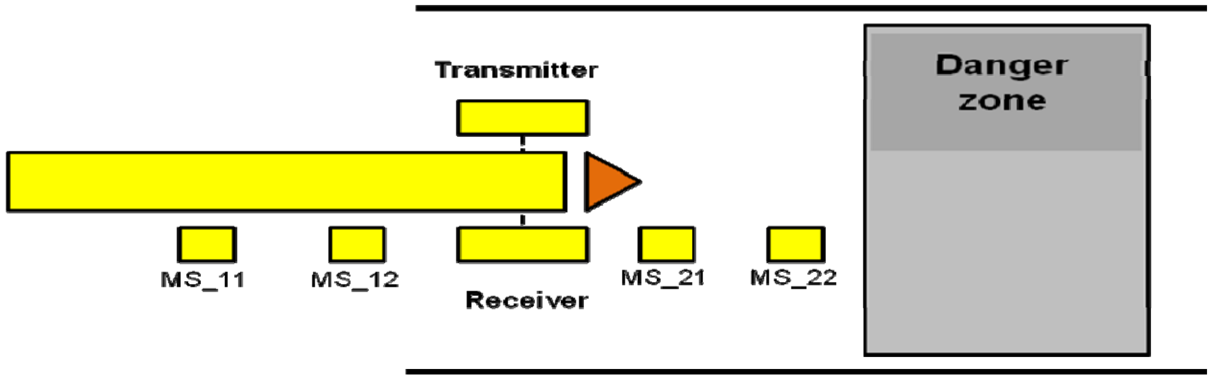
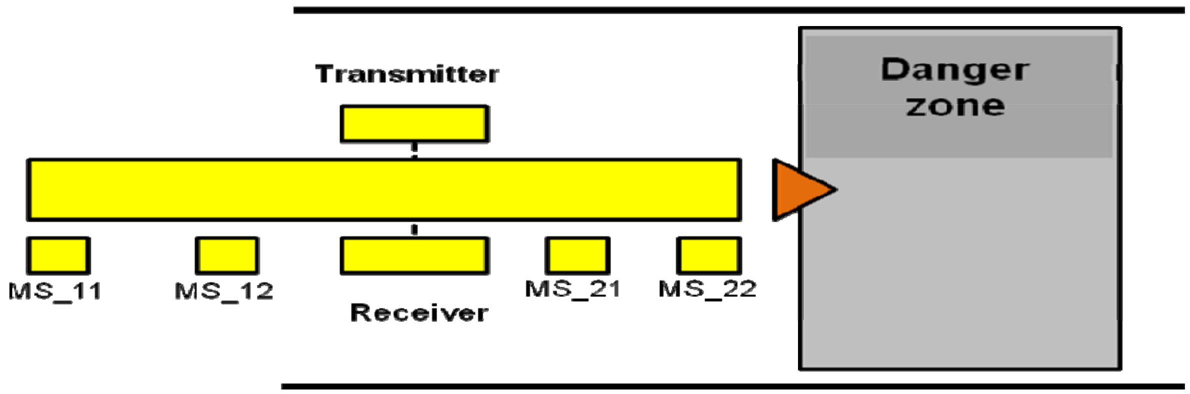
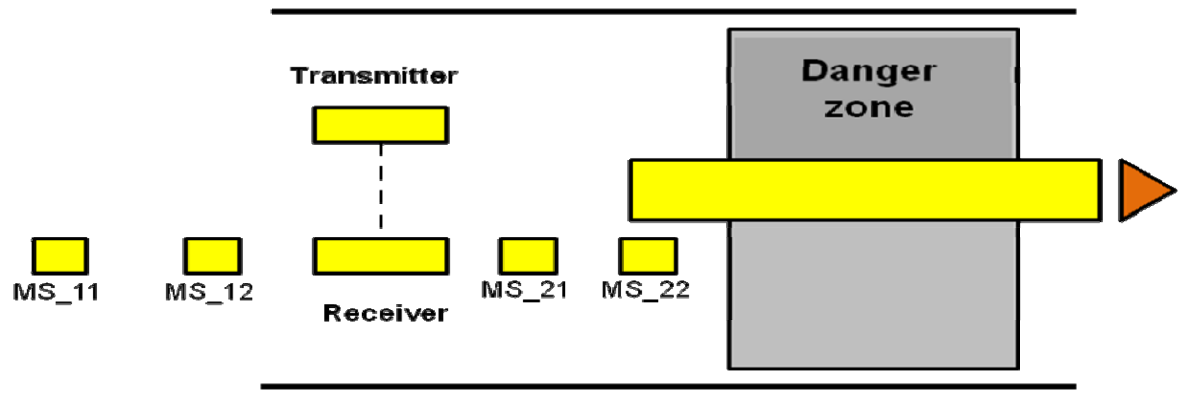
The S_StartReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Step 1 : If muting sensor MutingSwitch12 (MS_12) is activated by the product after MutingSwitch11 (MS_11), the muting mode is activated.

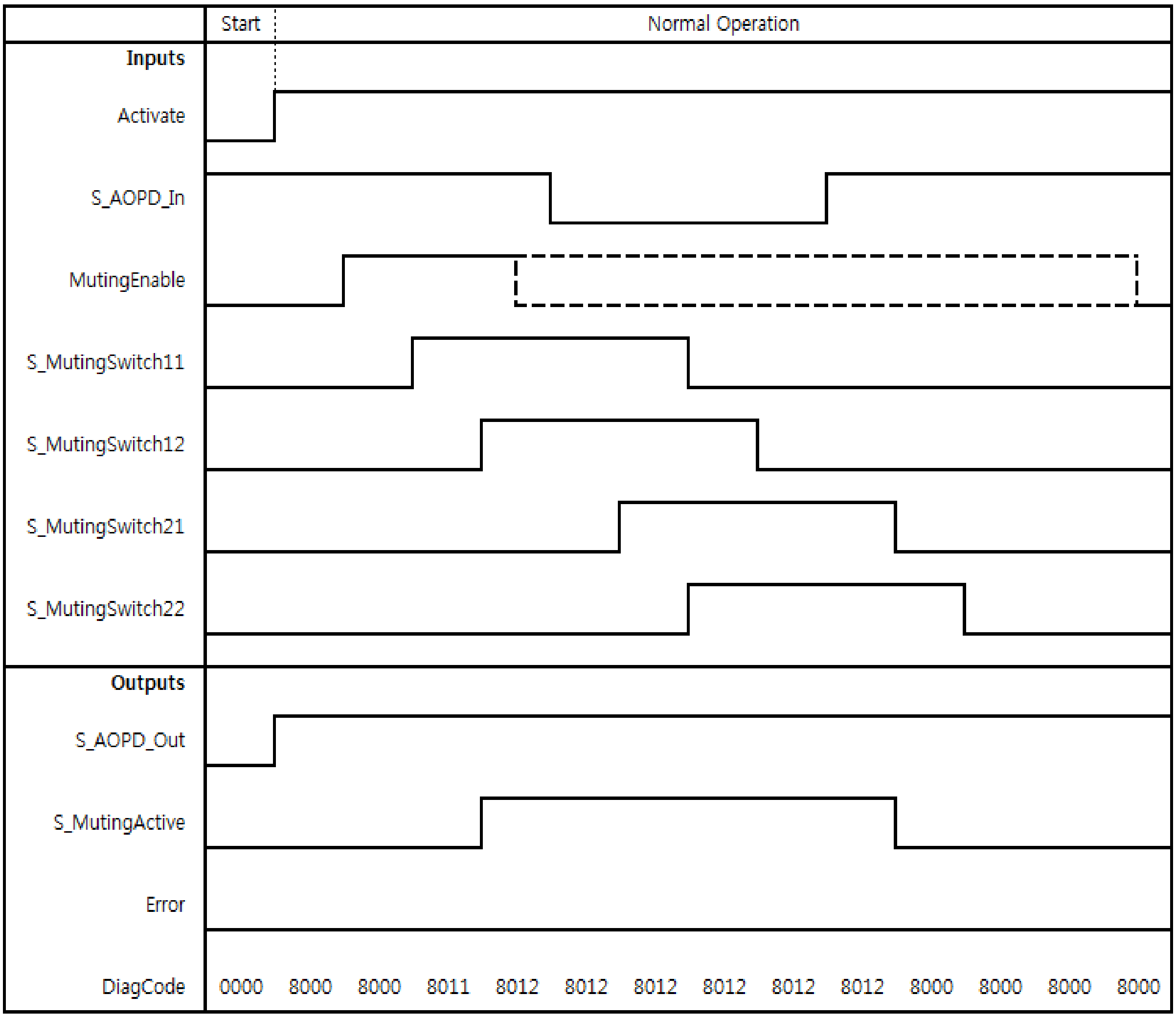
Step 2 : Muting mode remains active as long as MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are activated by the product. The product may pass through the light curtain without causing a machine stop.

Step 3 : Before muting sensors MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are disabled, muting sensors MutingSwitch21 (MS_21) and MutingSwitch22 (MS_22) must be activated. This ensures that muting mode remains active.

Step 4 : Muting mode is terminated if only muting sensor MutingSwitch22 (MS_22) is activated by the product.

NO.	Figure
1	
2	
3	
4	

4) Typical Timing Diagrams



5) Error Detection

The FB detects the following error conditions:

- Muting sensors MutingSwitch11, MutingSwitch12, MutingSwitch21, and MutingSwitch22 are activated in the wrong order.
- Muting sequence starts without being enabled by MutingEnable
- A faulty muting lamp is indicated by S_MutingLamp = FALSE.
- A static Reset condition.
- MaxMutingTime has been set to a value less than T#0s or greater than T#10min.
- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.

6) Error Behavior

In the event of an error, the S_AOPD_Out and S_MutingActive outputs are set to FALSE. The DiagCode output indicates the relevant error code and the Error output is set to TRUE.

A restart is inhibited until the error conditions are cleared and the Safe state is acknowledged with Reset by the operator.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Reset Error 1	Static Reset condition detected after FB activation. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C002	Reset Error 2	Static Reset condition detected in state 8003. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C003	Error Muting lamp	Error detected in muting lamp. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
CYx4	Error Muting sequence	Error detected in muting sequence in states 8000, 8011, 8012, 8112 or 8122. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE Y = Status in the sequence (2 states for forward and 2 states for backward direction). C0x4 = Error occurred in state 8000 C1x4 = Error occurred in state Forward 8011 C2x4 = Error occurred in state Forward 8012 C3x4 = Error occurred in state Backward 8122 C4x4 = Error occurred in state Backward 8112 CFx4 = Muting Enable missing x = Status of the sensors when error occurred (4 bits: LSB = MS_11; MS_12; MS_21; MSB = MS_22).
C005	Parameter Error	MaxMutingTime value out of range. Ready = TRUE, Error = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE
C006	Error Timer MaxMuting	Timing error: Active muting time (when S_MutingActive = TRUE) exceeds MaxMutingTime. Ready = TRUE, Error = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE

8) Status codes

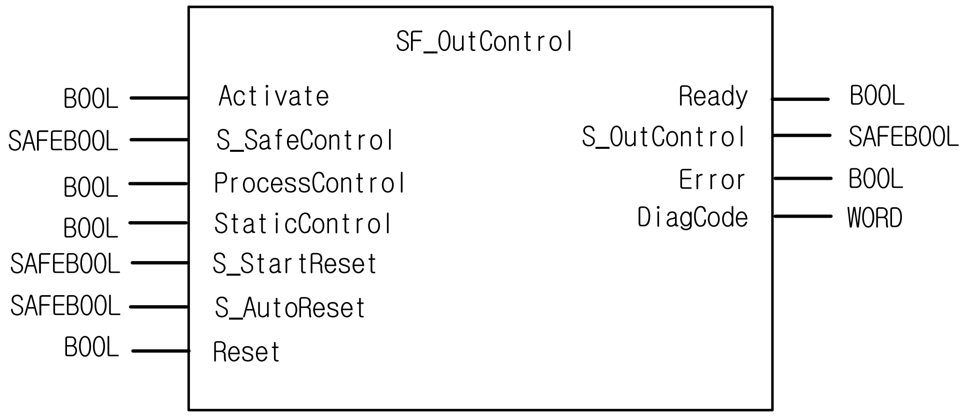
DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8000	AOPD Free	Muting not active and no safety demand from AOPD. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8001	Init	Function block has been activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8002	Safety Demand AOPD	Safety demand detected by AOPD, muting not active. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8003	Wait for Reset	Safety demand or errors have been detected and are now cleared. Operator acknowledgment by Reset required. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8005	Safe	Safety function activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8011	Muting Forward Start	Muting forward, sequence is in starting phase and no safety demand. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8012	Muting Forward Active	Muting forward, sequence is active. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE

DiagCode	State Name	State Description and Output Setting
8112	Muting Backward Active	Muting backward, sequence is active. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8122	Muting Backward Start	Muting backward, sequence is in starting phase and no safety demand. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE

15.2.12 SF_OUTCONTROL

1) Overview

Control of a safety output with a signal from the functional application and a safety signal with optional startup inhibits.



2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_SafetyControl	SAFEBOOL	0	Control signal of the preceding safety FB. Typical function block signals from the library (e.g., SF_EStop, SF_GuardMonitoring, SF_TwoHandControlTypeII, and/or others). FALSE: The preceding safety FB's are in safe state. TRUE: The preceding safety FB's enable safety control.
	ProcessControl	BOOL	0	Control signal from the functional application. FALSE: Request to set S_OutControl to FALSE. TRUE: Request to set S_OutControl to TRUE.
	StaticControl	BOOL	0	Optional conditions for process control. FALSE: Dynamic change at ProcessControl (FALSE => TRUE) required after block activation or triggered safety function. Additional function start required. TRUE: No dynamic change at ProcessControl (FALSE => TRUE) required after block activation or triggered safety function.

Type	Name	Data Type	Initial Value	Description
Input	S_StartReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when PES is started (warm or cold). TRUE: Automatic reset when PES is started (warm or cold). This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
	S_AutoReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when emergency stop button is released. TRUE: Automatic reset when emergency stop button is released. This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
	Reset	BOOL	0	Reset
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_OutControl	SAFEBOOL	0	Controls connected actuators. FALSE: Disable connected actuators. TRUE: Enable connected actuators.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more than 16 codes.

3) Functional Description

General: The SF_OutControl FB is an output driver for a safety output.

The safety output is controlled via S_OutControl using a signal from the functional application (ProcessControl/BOOL to control the process) and a signal from the safety application (S_SafeControl/SAFEBOOL to control the safety function).

Optional conditions for process control (ProcessControl):

- An additional function start (ProcessControl FALSE => TRUE) is required following block activation or feedback of the safe signal (S_SafeControl). A static TRUE signal at ProcessControl does not set S_OutControl to TRUE.
- An additional function start (ProcessControl FALSE => TRUE) is not required following block activation or feedback of the safe

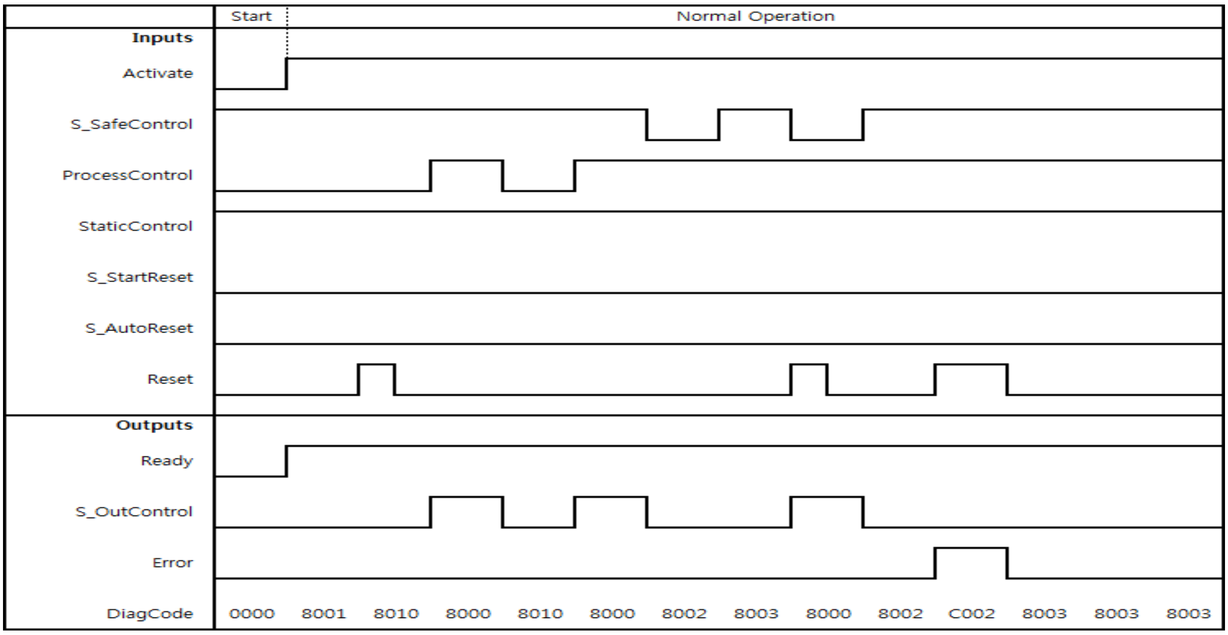
signal (S_SafeControl). A static TRUE signal at ProcessControl sets S_OutControl to TRUE if the other conditions have been met.

Optional startup inhibits:

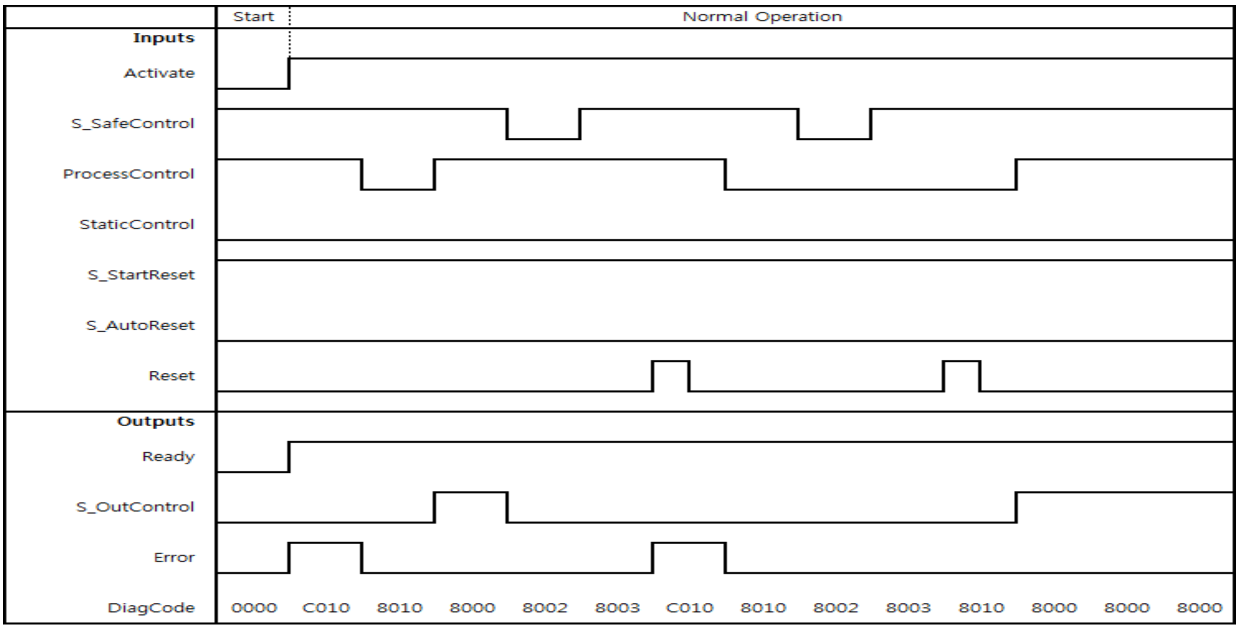
- Startup inhibit after function block activation.
- Startup inhibit after interruption of the protective device.

The StaticControl, S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

4) Typical Timing Diagrams



< S_StartReset=Off >



< S_StartReset=On >

5) Error Detection

The following conditions force a transition to the Error state:

- Invalid static Reset signal in the process.
- Invalid static ProcessControl signal.
- ProcessControl and Reset are incorrectly interconnected due to programming error.

6) Error Behavior

In the event of an error, the S_OutControl output is set to FALSE and remains in this safe state.

To leave the Reset, Init or Lock error states, the Reset input must be set to FALSE. To leave the Control error state, the ProcessControl input must be set to FALSE.

After transition of S_SafeControl to TRUE, the optional startup inhibit can be reset by a rising edge at the Reset input.

After block activation, the optional startup inhibit can be reset by a rising edge at the Reset input.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Reset Error 1	Static Reset signal in state 8001. Ready = TRUE S_OutControl = FALSE Error = TRUE
C002	Reset Error 2	Static Reset signal in state 8003. Ready = TRUE S_OutControl = FALSE Error = TRUE
C010	Control Error	Static signal at ProcessControl in state 8010. Ready = TRUE S_OutControl = FALSE Error = TRUE
C111	Init Error	Simultaneous rising trigger at Reset and ProcessControl in state 8001. Ready = TRUE S_OutControl = FALSE Error = TRUE
C211	Lock Error	Simultaneous rising trigger at Reset and ProcessControl in state 8003. Ready = TRUE S_OutControl = FALSE Error = TRUE

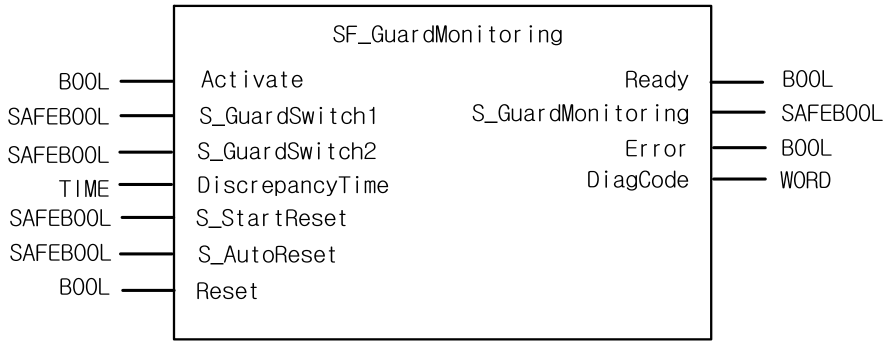
8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_OutControl = FALSE Error = FALSE
8001	Init	Block activation startup inhibit is active. Reset required. Ready = TRUE S_OutControl = FALSE Error = FALSE
8002	Safe	Triggered safety function. Ready = TRUE S_OutControl = FALSE Error = FALSE
8003	Lock	Safety function startup inhibit is active. Reset required. Ready = TRUE S_OutControl = FALSE Error = FALSE
8010	Output Disable	Process control is not active. Ready = TRUE S_OutControl = FALSE Error = FALSE
8000	Output Enable	Process control is active and safety is enabled. Ready = TRUE S_OutControl = TRUE Error = FALSE

15.2.13 SF_SAFEGUARD

1) Overview

This function block monitors the relevant safety guard. There are two independent input parameters for two switches at the safety guard coupled with a time difference (MonitoringTime) for closing the guard.



2) Input / Output Variables

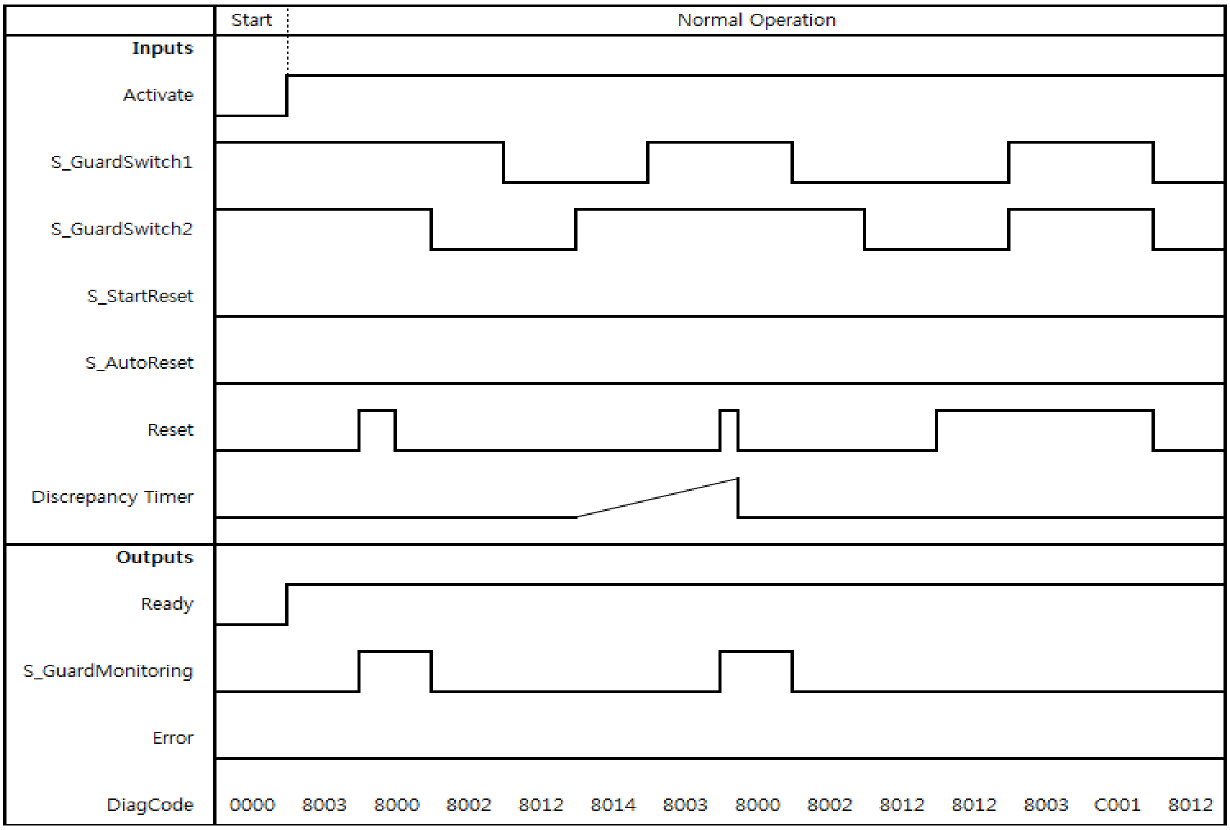
Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_GuardSwitch1	SAFEBOOL	0	Guard switch 1 input. FALSE: Guard is open. TRUE: Guard is closed.
	S_GuardSwitch2	SAFEBOOL	0	Guard switch 2 input. FALSE: Guard is open. TRUE: Guard is closed.
	DiscrepancyTime	Time	T#0ms	Configures the monitored synchronous time between S_GuardSwitch1 and S_GuardSwitch2.
	S_StartReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when PES is started (warm or cold). TRUE: Automatic reset when PES is started (warm or cold). This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.

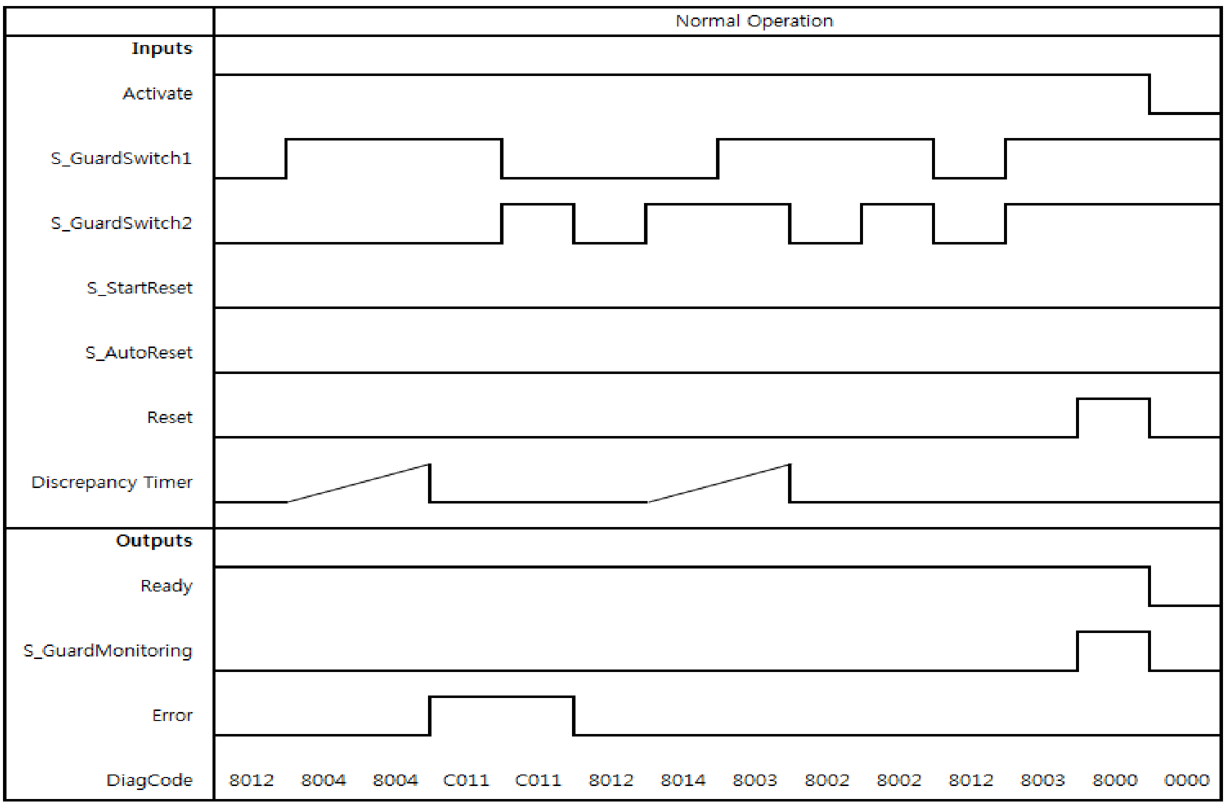
Type	Name	Data Type	Initial Value	Description
	S_AutoReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when emergency stop button is released. TRUE: Automatic reset when emergency stop button is released. This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
	Reset	BOOL	0	Reset
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_GuardMonitoring	SAFEBOOL	0	Output indicating the status of the guard. FALSE: Guard is not active. TRUE: both S_GuardSwitches are TRUE, no error and acknowledgment. Guard is active.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more than 16 codes.

3) Functional Description

The function block requires two inputs indicating the guard position for safety guards with two switches, a DiscrepancyTime input and Reset input. If the safety guard only has one switch, the S_GuardSwitch1 and S_GuardSwitch2 inputs can be bridged. The monitoring time is the maximum time required for both switches to respond when closing the safety guard. The Reset, S_StartReset, and S_AutoReset inputs determine how the function block is reset after the safety guard has been opened.
When opening the safety guard, both S_GuardSwitch1 and S_GuardSwitch2 inputs should switch to FALSE. The S_GuardMonitoring output switches to FALSE as soon as one of the switches is set to FALSE. When closing the safety guard, both S_GuardSwitch1 and S_GuardSwitch2 inputs should switch to TRUE.
This FB monitors the symmetry of the switching behavior of both switches. The S_GuardMonitoring output remains FALSE if only one of the contacts has completed an open/close process.
The behavior of the S_GuardMonitoring output depends on the time difference between the switching inputs. The discrepancy time is monitored as soon as the value of both S_GuardSwitch1/S_GuardSwitch2 inputs differs. If the DiscrepancyTime has elapsed, but the inputs still differ, the S_GuardMonitoring output remains FALSE. If the second corresponding S_GuardSwitch1/S_GuardSwitch2 input switches to TRUE within the value specified for the DiscrepancyTime input, the S_GuardMonitoring output is set to TRUE following acknowledgment.
The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

4) Typical Timing Diagrams





5) Error Detection

External signals: SAFEBOOL inputs provide inherent error detection. Mechanical setup combines that of an opening and closing switch according to EN 954 (safety guard with two switches). Discrepancy time monitoring for time lag between both mechanical switches reaction, according to EN 954 (to be considered as "application error" detection, i.e., generated by the application).

An error is detected if the time lag between the first S_GuardSwitch1/S_GuardSwitch2 input and the second is greater than the value for the DiscrepancyTime input. The Error output is set to TRUE.

The function block detects a static TRUE signal at the RESET input.

6) Error Behavior

The S_GuardMonitoring output is set to FALSE. If the two S_GuardSwitch1 and S_Guardswitch2 inputs are bridged, no error is detected. To leave the Reset error state, the Reset input must be set to FALSE. To leave the discrepancy time errors, the inputs S_GuardSwitch1 and 2 must both be set to FALSE.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Reset Error	Static reset detected in state 8003. Ready = TRUE S_GuardMonitoring = FALSE Error = TRUE
C011	Discrepancytime Error 1	DiscrepancyTime elapsed in state 8004. Ready = TRUE S_GuardMonitoring = FALSE Error = TRUE
C012	Discrepancytime Error 2	DiscrepancyTime elapsed in state 8014. Ready = TRUE S_GuardMonitoring = FALSE Error = TRUE

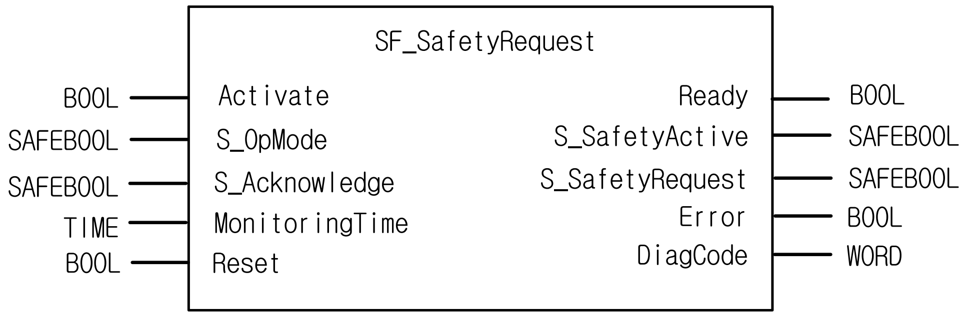
8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_GuardMonitoring = FALSE Error = FALSE
8000	Normal	Safety guard closed and Safe state acknowledged. Ready = TRUE S_GuardMonitoring = TRUE Error = FALSE
8001	Init	Function block has been activated. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8002	Open Guard Request	Complete switching sequence required. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8003	Wait for Reset	Waiting for rising trigger at Reset. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8012	Guard Opened	Guard completely opened. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8004	Wait for GuardSwitch2	S_GuardSwitch1 has been switched to TRUE - waiting for S_GuardSwitch2; discrepancy timer started. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8014	Wait for GuardSwitch1	S_GuardSwitch2 has been switched to TRUE - waiting for S_GuardSwitch1; discrepancy timer started. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8005	Guard Closed	Guard closed. Waiting for Reset, if S_AutoReset = FALSE. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE

15.2.14 SF_SAFETYREQUEST

1) Overview

This function block provides the interface to a generic actuator, e.g. a safety drive or safety valve, to place the actuator in a safe state.



2) Input / Output Variables

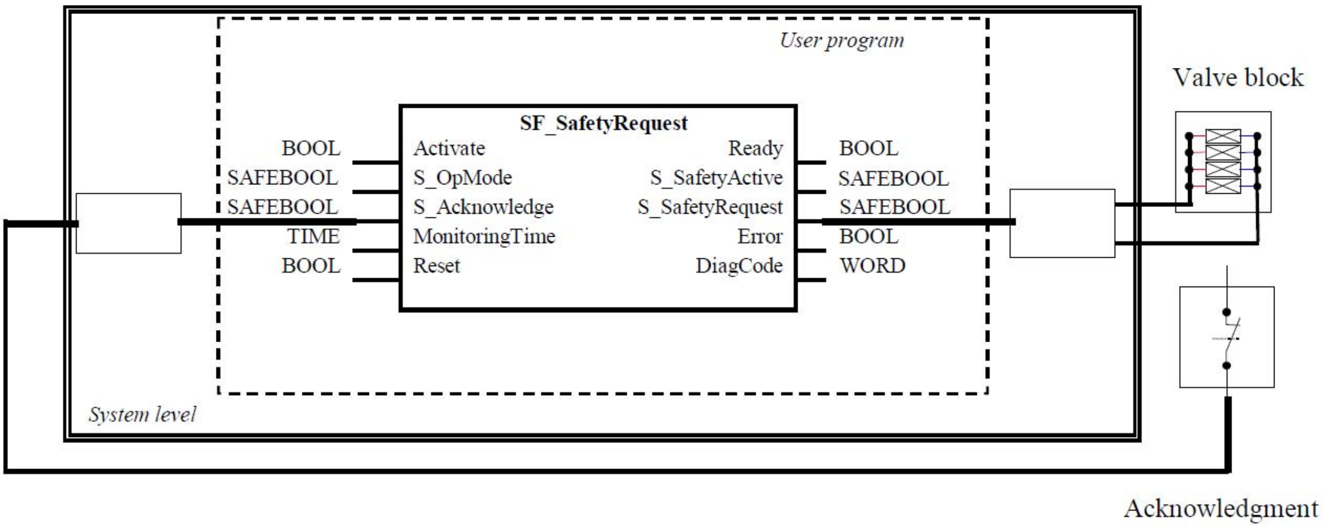
Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_OpMode	SAFEBOOL	0	Requested mode of a generic safe actuator. FALSE: Safe mode is requested. TRUE: Operation mode is requested.
	S_Acknowledge	SAFEBOOL	0	Confirmation of the generic actuator, if actuator is in the Safe state. FALSE: Operation mode (non-safe). TRUE: Safe mode.
	MonitoringTime	TIME	T#0s	Monitoring of the response time between the safety function request (S_OpMode set to FALSE) and the actuator acknowledgment (S_Acknowledge switches to TRUE).
	Reset	BOOL	0	Reset
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_SafetyActive	SAFEBOOL	0	Confirmation of the Safe state. FALSE: Non-safe state. TRUE: Safe state.
	S_SafetyRequest	SAFEBOOL	0	Request to place the actuator in a safe state. FALSE: Safe state is requested. TRUE: Non-safe state.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more than 16 codes.

3) Functional Description

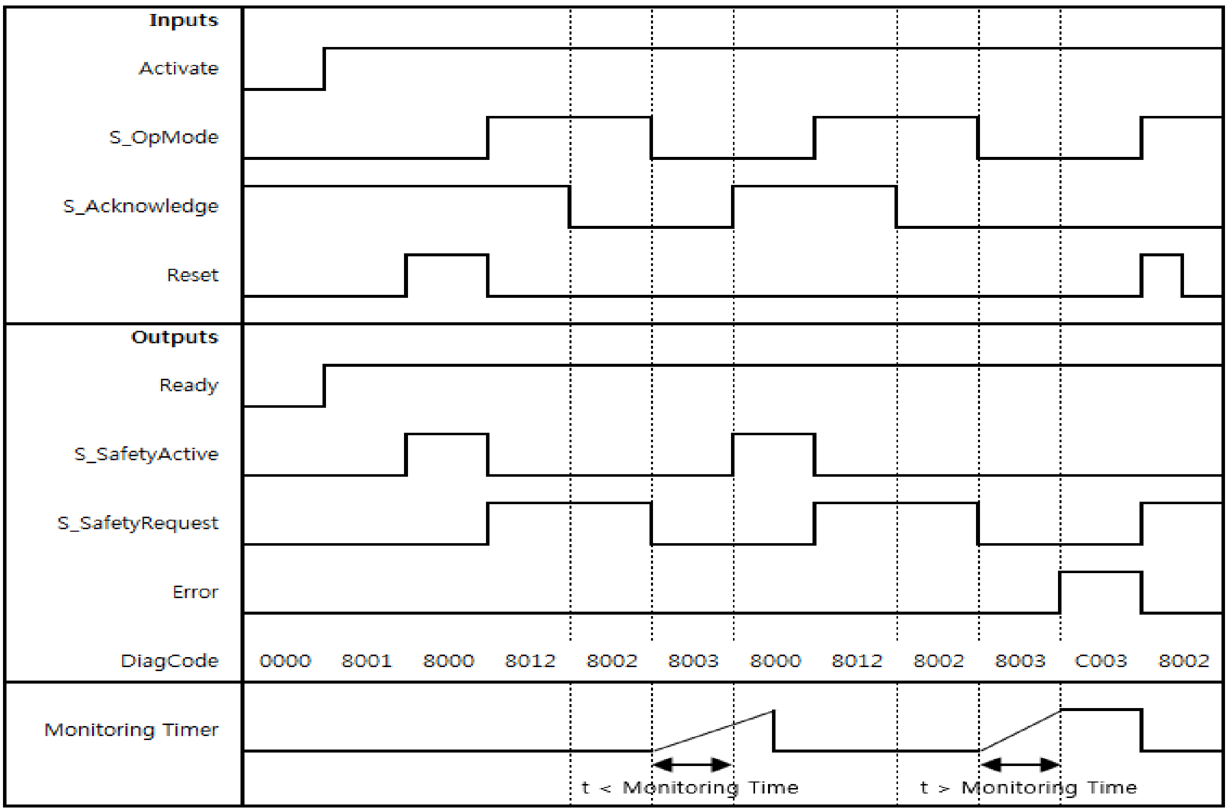
This FB provides the interface between the safety-related system and a generic actuator. This means that the safety-related functions of the actuator are available within the application program. However, there are only two binary signals to control the Safe state of the generic actuator, i.e., one for requesting and one for receiving the confirmation.

The safety function will be provided by the actuator itself. Therefore the FB only initiates the request, monitors it, and sets the output when the actuator acknowledges the Safe state. This will be indicated with the "S_SafetyActive" output.

This FB does not define any generic actuator-specific parameters. They should have been specified in the generic actuator itself. It switches the generic actuator from the operation mode to a safe state.



4) Typical Timing Diagrams



5) Error Detection

The FB detects whether the actuator does not enter the Safe state within the monitoring time.
The FB detects whether the acknowledge signal is lost while the request is still active.
The FB detects a static Reset signal.

External FB errors:
There are no external errors, since there is no error bits/information provided by the generic actuator.

6) Error Behavior

In the event of an error, the S_SafetyActive output is set to FALSE.
An error must be acknowledged by a rising trigger at the Reset input. To continue the function block after this reset, the S_OpMode request must be set to TRUE.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C002	Acknowledge Lost	Acknowledgment lost while in the Safe state. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = TRUE
C003	MonitoringTime Elapsed	S_OpMode request could not be completed within the monitoring time. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = TRUE
C004	Reset Error 2	Static Reset detected in state C002 (Acknowledge Lost). Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = TRUE
C005	Reset Error 3	Static Reset detected in state C003 (MonitoringTime Elapsed). Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = TRUE

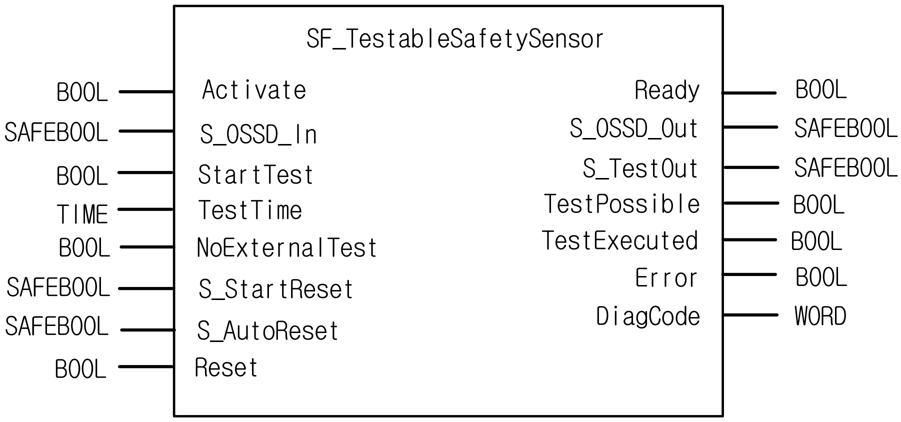
8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = FALSE
8000	Safe Mode	Actuator is in a safe mode. Ready = TRUE S_SafetyActive = TRUE S_SafetyRequest = FALSE Error = FALSE
8001	Init	State after Activate is set to TRUE or after a rising trigger at Reset. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = FALSE
8002	Operation Mode	Operation mode without Acknowledge of safe mode Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = TRUE Error = FALSE
8012	Wait for Confirmation OpMode	Operation mode with Acknowledge of safe mode Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = TRUE Error = FALSE
8003	Wait for Confirmation	Waiting for confirmation from the drive (system interface). Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = FALSE
8005	Wait for OpMode	Error was cleared. However S_OpMode must be set to TRUE before the FB can be initialized. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = FALSE

15.2.15 SF_TESTABLESAFETYSENSOR

1) Overview

This function block detects, for example, the loss of the sensing unit detection capability, the response time exceeding that specified, and static ON signal in single-channel sensor systems. It can be used for external testable safety sensors (ESPE: Electro-sensitive protective equipment, such as a light beam).



2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_OSSD_In	SAFEBOOL	0	Status of sensor output, e.g., light curtain. FALSE: Safety sensor in test state or demand for safety-related response. TRUE: Sensor in the state for normal operating conditions.
	StartTest	BOOL	0	Input to start sensor test. Sets "S_TestOut" and starts the internal time monitoring function in the FB. FALSE: No test requested. TRUE: Test requested.
	TestTime	Time	T#10ms	Constant. Range: 0 … 150ms. Test time of safety sensor.
	NoExternalTest	BOOL	0	Indicates if external manual sensor test is supported. FALSE: The external manual sensor test is supported. Only after a complete manual sensor switching sequence, a automatic test is possible again after a faulty automatic sensor test. TRUE: The external manual sensor test is not supported. An automatic test is possible again without a manual sensor switching sequence after faulty automatic sensor test.

Type	Name	Data Type	Initial Value	Description
	S_AutoReset	SAFEBOOL	0	FALSE (= initial value): Manual reset when emergency stop button is released. TRUE: Automatic reset when emergency stop button is released. This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) startup does not occur.
	Reset	BOOL	0	Reset
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_OSSD_Out	SAFEBOOL	0	Safety related output indicating the status of the ESPE. FALSE: The sensor has a safety-related action request or test error. TRUE: The sensor has no safety-related action request AND no test error.
	S_TestOut	SAFEBOOL	1	Coupled with the test input of the sensor. Although specified as SAFEBOOL, in practice this signal will often be connected to a BOOL output. FALSE: Test request issued. TRUE: No test request.
	TestPossible	BOOL	0	Feedback signal to the process. FALSE: An automatic sensor test is not possible. TRUE: An automatic sensor test is possible.
	TestExecuted	BOOL	0	A positive signal edge indicates the successful execution of the automatic sensor test. FALSE: - An automatic sensor test was not executed yet. - An automatic sensor test is active. - An automatic sensor test was faulty. TRUE: A sensor test was executed successfully.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more than 16 codes.

3) Functional Description

Type 2 ESPE shall have a means of periodic testing to detect a hazardous fault (e.g., loss of sensing unit detection capability, response time exceeding that specified). The test signal shall simulate the actuation of the sensing device and the duration of the periodic test shall not exceed 150 ms. The test shall verify that each light beam operates in the manner specified by the supplier. If the periodic test is intended to be initiated by an external safety-related control system (e.g., a machine), the ESPE shall be provided with suitable input facilities (e.g., terminals).The ESPE must be selected in respect of the product standards EN IEC 61496-1, -2 and -3 and the required categories according EN 954-1. It must be monitored by separate functionality, that the test is initiated within appropriate intervals. The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

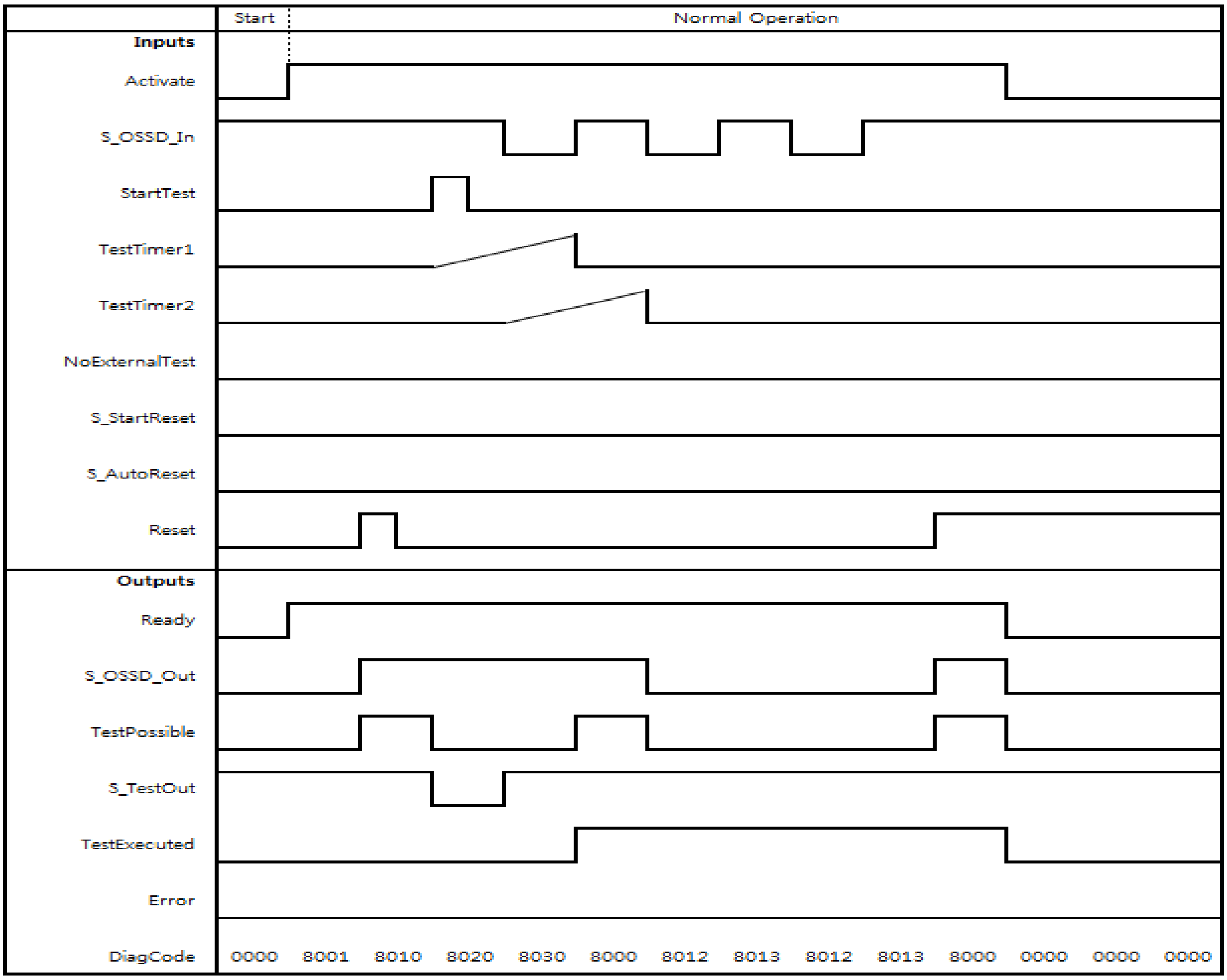
Test mode:

1. StartTest = TRUE: S_TestOut = FALSE. Start monitoring time
2. S_TestOut signal stops transmitter (Monitoring of TestTime started first time)
3. S_OSSD_In changes from TRUE to FALSE (Monitoring of TestTime started second time)
4. S_TestOut changes from FALSE to TRUE
5. Start transmitter
6. Sensor S_OSSD_In changes from FALSE to TRUE
7. Stop monitoring time
8. S_OSSD_Out is set to TRUE during testing

Optional startup inhibits:

- Startup inhibit after function block activation.
- Startup inhibit after interruption of the protective device.

4) Typical Timing Diagrams



5) Error Detection

The following conditions force a transition to the Error state:

- Test time overrun without delayed sensor feedback.
- Test without sensor signal feedback.
- Invalid static reset signal in the process.
- Plausibility check of the monitoring time setting.

6) Error Behavior

In the event of an error, the S_OSSD_Out output is set to FALSE and remains in this safe state.
Once the error has been removed and the sensor is on (S_OSSD_In = TRUE) – a reset removes the error state and sets the S_OSSD_Out output to TRUE.
If S_AutoReset = FALSE, a rising trigger is required at Reset.
After transition of S_OSSD_In to TRUE, the optional startup inhibit can be reset by a rising edge at the Reset input.
After block activation, the optional startup inhibit can be reset by a rising edge at the Reset input.

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C000	Parameter Error	Invalid value at the TestTime parameter. Values between 0 ms and 150 ms are possible. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C001	Reset Error 1	Static Reset condition detected after FB activation. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
C002	Reset Error 2	Static Reset condition detected in state 8003. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C003	Reset Error 3	Static Reset condition detected in state C010. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C004	Reset Error 4	Static Reset condition detected in state C020. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C005	Reset Error 5	Static Reset condition detected in state 8006. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE

DiagCode	State Name	State Description and Output Setting
C006	Reset Error 6	Static Reset condition detected in state C000. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C007	Reset Error 7	Static Reset condition detected in state 8015. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = TRUE Error = TRUE
C010	Test Error 1	Test time elapsed in state 8020. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C020	Test Error 2	Test time elapsed in state 8030. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE

8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8001	Init	An activation has been detected by the FB. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8002	ESPE Interrupted 1	The FB has detected a safety demand. The switch has not been automatically tested yet. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8003	Wait for Reset 1	Wait for rising trigger of Reset after state 8002. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8004	External Function Test	The automatic sensor test was faulty. An external manual sensor test is necessary. The support for the necessary external manual sensor test has been activated at the FB (NoExternalTest = FALSE). A negative signal edge at the sensor is required. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE

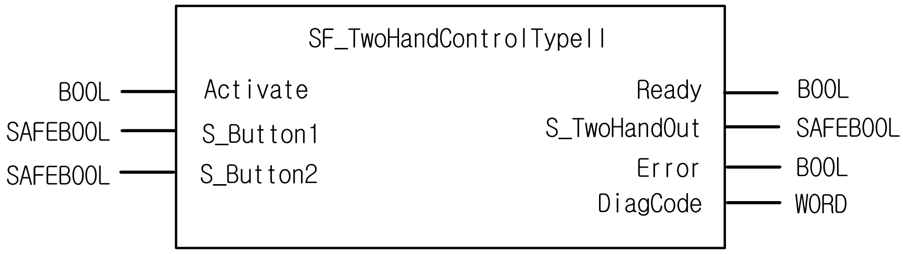
DiagCode	State Name	State Description and Output Setting
8005	ESPE Interrupted External Test	The automatic sensor test was faulty. An external manual sensor test is necessary. The support for the necessary external manual sensor test has been activated at the FB (NoExternalTest = FALSE). A TRUE signal at the sensor is required. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8006	End External Test	The automatic sensor test was faulty. An external manual sensor test is necessary. The support for the necessary external manual sensor test has been activated at the FB (NoExternalTest = FALSE). The external manual test is complete. The FB detected a complete sensor switching cycle (external controlled). Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8010	ESPE Free No Test	The FB has not detected a safety demand. The sensor has not been tested automatically. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = TRUE TestPossible = TRUE TestExecuted = FALSE Error = FALSE
8020	Test Request	The automatic sensor test is active. Test Timer is started first time. The transmitter signal of the sensor is switched off by the FB. The signal of the receiver must follow the signal of the transmitter. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = FALSE TestPossible = FALSE TestExecuted = FALSE Error = FALSE

DiagCode	State Name	State Description and Output Setting
8030	Test Active	The automatic sensor test is active. Test Timer is started second time. The transmitter signal of the sensor is switched on by the FB. The signal of the receiver must follow the signal of the transmitter. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8000	ESPE Free Test ok	The FB has not detected a safety demand. The sensor was automatically tested. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = TRUE TestPossible = TRUE TestExecuted = TRUE Error = FALSE
8012	ESPE Interrupted 2	The FB has detected a safety demand. The switch was automatically tested. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = TRUE Error = FALSE
8013	Wait for Reset 2	Wait for rising trigger of Reset after state 8012. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = TRUE Error = FALSE

15.2.16 SF_TWOHANDCTRLII

1) Overview

This function block provides the two-hand control functionality.



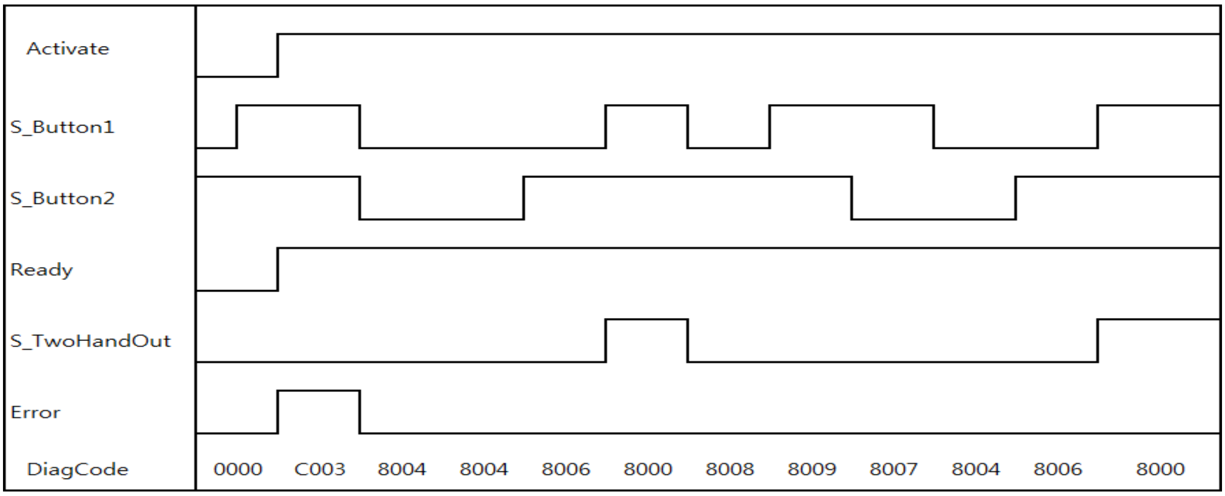
2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_Button1	SAFEBOOL	0	FALSE: Button 1 released. TRUE: Button 1 actuated.
	S_Button2	SAFEBOOL	0	FALSE: Button 2 released. TRUE: Button 2 actuated.
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_TwoHandOut	SAFEBOOL	0	Safety related output signal. FALSE: No correct two hand operation. TRUE: S_Button1 and S_Button2 inputs are TRUE and no error occurred. Correct two hand operation.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more than 16 codes.

3) Functional Description

This function block provides the two-hand control functionality according to EN 574, Section 4 Type II. If S_Button1 and S_Button2 are set to TRUE in correct sequence, then the S_TwoHandOut output will also be set to TRUE. The FB also controls the release of both buttons before setting the output S_TwoHandOut again to TRUE.

4) Typical Timing Diagrams



5) Error Detection

After activation of the FB, any button set to TRUE is detected as an invalid input setting leading to an error.

6) Error Behavior

In the event of an error, the S_TwoHandOut output is set to FALSE and remains in this safe state.
The Error state is exited when both buttons are released (set to FALSE).

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Error B1	S_Button1 was TRUE on FB activation. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C002	Error B2	S_Button2 was TRUE on FB activation. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C003	Error B1&B2	The signals at S_Button1 and S_Button2 were TRUE on FB activation. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE

8) Status codes

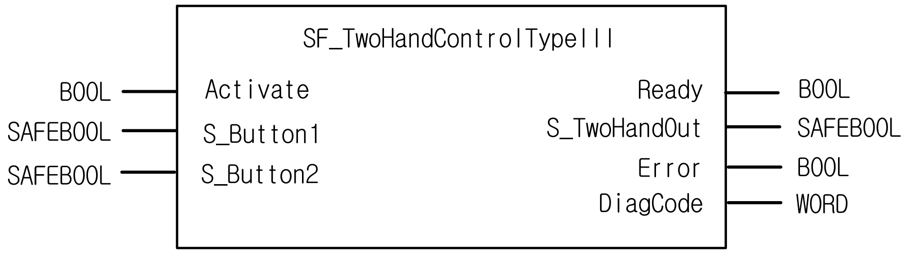
DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE Error = FALSE S_TwoHandOut = FALSE
8000	Buttons Actuated	Both buttons actuated correctly. The safety related output is enabled. Ready = TRUE Error = FALSE S_TwoHandOut = TRUE
8001	Init	Function block is active, but in the Init state. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8004	Buttons Released	No button is actuated. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8005	Button 1 Actuated	Only Button 1 is actuated. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8006	Button 2 Actuated	Only Button 2 is actuated. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8007	Button 2 Released	The safety related output was enabled and is disabled again. FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output. In this state, S_Button1 is TRUE and S_Button2 is FALSE after disabling the safety related output. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8008	Button 1 Released	The safety related output was enabled and is disabled again. FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output. In this state, S_Button1 is FALSE and S_Button2 is TRUE after disabling the safety related output. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE

DiagCode	State Name	State Description and Output Setting
8009	Locked Off	The safety related output was enabled and is disabled again. FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output. In this state, S_Button1 is TRUE and S_Button2 is TRUE after disabling the safety related output. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8019	Locked On	Incorrect actuation of the buttons. Waiting for release of both buttons. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE

- ## 15.2.17 SF_TWOHANDCTRLIII

1) Overview

This function block provides the two-hand control functionality.



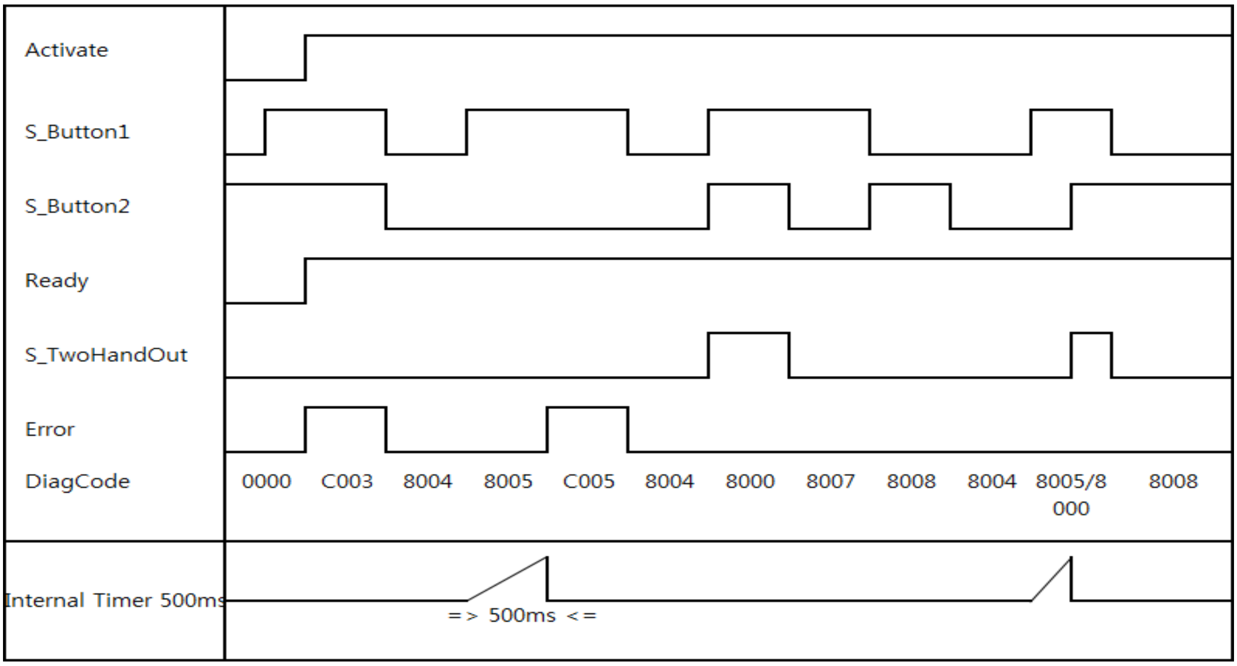
2) Input / Output Variables

Type	Name	Data Type	Initial Value	Description
Input	Activate	BOOL	0	Activation of the FB
	S_Button1	SAFEBOOL	0	Input of button 1 FALSE: Button 1 released. TRUE: Button 1 actuated.
	S_Button2	SAFEBOOL	0	Input of button 2 FALSE: Button 2 released. TRUE: Button 2 actuated.
Output	Ready	BOOL	0	If TRUE, indicates that the FB is activated and the output results are valid.
	S_TwoHandOut	SAFEBOOL	0	Safety related output signal. FALSE: No correct two hand operation. TRUE: S_Button1 and S_Button2 inputs changed from FALSE to TRUE within 500 ms and no error occurred. The two hand operation has been performed correctly.
	Error	BOOL	0	Error flag
	DiagCode	WORD	16#0000	Diagnostic register. All states of the FB are represented by this register. This information is encoded in hexadecimal format in order to represent more than 16 codes.

3) Functional Description

This function block provides the two-hand control functionality according to EN 574, Section 4 Type III. If S_Button1 and S_Button2 are set to TRUE within 500 ms and in correct sequence, then the S_TwoHandOut output is also set to TRUE. The FB also controls the release of both buttons before setting the output S_TwoHandOut again to TRUE.

4) Typical Timing Diagrams



5) Error Detection

After activation of the FB, any button set to TRUE is detected as an invalid input setting leading to an error. The FB detects when the divergence of the input signals exceeds 500 ms.

6) Error Behavior

In the event of an error, the S_TwoHandOut output is set to FALSE and remains in this safe state.

The Error state is exited when both buttons are released (set to FALSE).

7) Error Codes

DiagCode	State Name	State Description and Output Setting
C001	Error 1 B1	S_Button1 was TRUE on FB activation. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C002	Error 1 B2	S_Button2 was TRUE on FB activation. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C003	Error 1 B1&B2	The signals at S_Button1 and S_Button2 were TRUE on FB activation. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C004	Error 2 B1	S_Button1 was FALSE and S_Button 2 was TRUE after 500 ms in state 8005. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C005	Error 2 B2	S_Button1 was TRUE and S_Button 2 was FALSE after 500 ms in state 8005. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C006	Error 2 B1&B2	S_Button1 was TRUE and S_Button 2 was TRUE after 500 ms in state 8005 or 8006. This state is only possible when the states of the inputs (S_Button1 and S_Button2) change from divergent to convergent (both TRUE) simultaneously when the timer elapses (500 ms) at the same cycle. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE

8) Status codes

DiagCode	State Name	State Description and Output Setting
0000	Idle	The function block is not active (initial state). Ready = FALSE Error = FALSE S_TwoHandOut = FALSE
8000	Buttons Actuated	Both buttons actuated correctly. The safety related output is enabled. Ready = TRUE Error = FALSE S_TwoHandOut = TRUE
8001	Init	Function block is active, but in the Init state. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8004	Buttons Released	No Button is actuated. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8005	Button 1 Actuated	Only Button 1 is actuated. Start monitoring timer. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8006	Button 2 Actuated	Only Button 2 is actuated. Start monitoring timer. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8007	Button 2 Released	The safety related output was enabled and is disabled again. FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output. In this state, S_Button1 is TRUE and S_Button2 is FALSE after disabling the safety related output. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8008	Button 1 Released	The safety related output was enabled and is disabled again. FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output. In this state, S_Button1 is FALSE and S_Button2 is TRUE after disabling the safety related output. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE

DiagCode	State Name	State Description and Output Setting
8009	Locked Off	The safety related output was enabled and is disabled again. FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output. In this state, S_Button1 is TRUE and S_Button2 is TRUE after disabling the safety related output. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8019	Locked On	Incorrect actuation of the buttons. Waiting for release of both buttons. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE

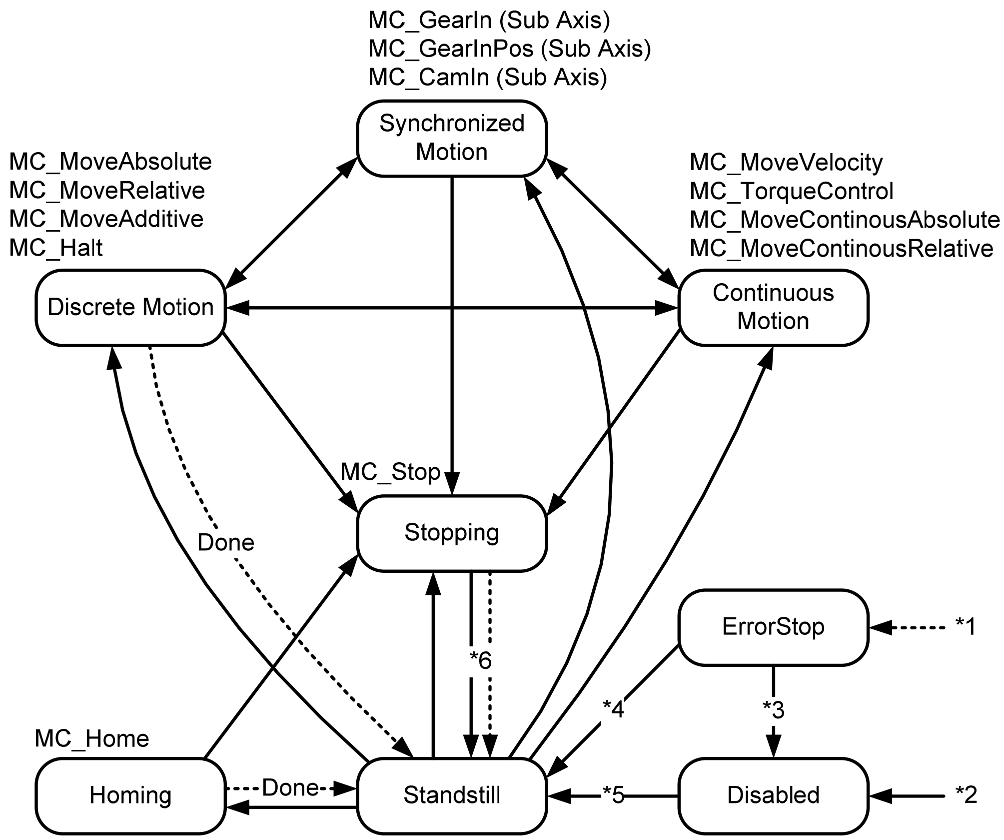
Chapter 16 Motion Function Blocks

This chapter describes the basic function block library mentioned in the previous chapter and other application function block library.

16.1 Common Elements of Motion Function Blocks

16.1.1 The State of axis

Each axis in the motion control module is changed to the relevant state depending on the situation and command. The changing structure of each situation is shown in the figure below.



*1 ErrorStop: in case axis error occurs regardless of the current state of axis

*2 Disabled: in case MC_Power.Enable input is Off when axis error does not occur

*3 ErrorStop → Disabled: in case MC_Reset command has issued when MC_Power.Status output is Off

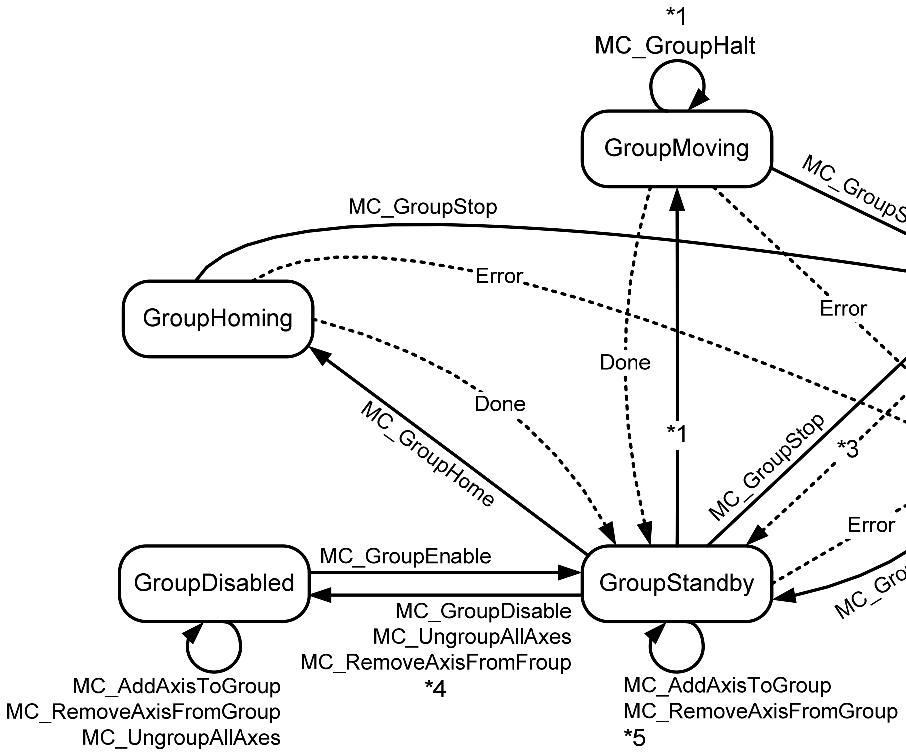
*4 ErrorStop → Standstill: in case MC_Reset command has issued when MC_Power.Status output is on and MC_Power.Enable input is On

*5 Disabled → Standstill: in case of turning On MC_Power.Enable input when MC_Power.Status output is On
*6 Stopping → Standstill: in case of turning Off MC_Stop.Execute input when MC_Stop.Done output is On

The state of axis	Description
Disabled	Disabled state indicates the state in which no command is given to a single axis, and no error occurs. In case there is no motion control module at the time of first operation, each axis begins in the disabled state. Afterwards, axis status is changed to standstill state in case servo-on status emerges when Enable input of servo On/Off (MC_Power) motion function block is On. The axis becomes disabled state when Enable input of serve On/Off (MC_Power) motion function block is Off in case of not being in ErrorStop state. In case there is motion function block which is currently being performed, the command is interrupted.(The CommandAborted output of the motion block function is On)
ErrorStop	No matter which state the current axis is in, it is changed to ErrorStop state when axis error occurs, and the axis decelerates to stop. In the state where error occurs, ErrorStop state is maintained even though servo On/Off (MC_Power) motion function block is executed. The motion axis which is in ErrorStop state maintains stationary state, and any command except for error reset is not executed.
StandStill	When the power of axis is activated, there is no error in the axis and any command is not made, the axis state indicates StandStill state.
Homing	Homing state indicates the axis is in homing operation.
Stopping	In case emergency stop (MC_Stop) function block is executed, the axis state is changed to stopping state. When the axis is in stopping state, other motion commands cannot be given to the axis until the Stop is completed (until Done output is activated). If Done output is On, and Execute input is On, the state is switched to Standstill status.
Continuous Motion	It indicates state where operation continues until the current axis becomes operation stop status.
Discrete Motion	It indicates reduced operating status with target position.
Synchronized Motion	Synchronized motion indicates axis is in synchronized operation.

16.1.2 The State of Group

Each group in motion control module is changed to the relevant state depending on the situation and command. The changing structure of each state is shown in the figure below.



*1 GroupMoving: in case of performing the motion function block of general group operation

*2 GroupStopping, GroupErrorStop: The relevant motion function block is not performed when different motion function block is performed in GroupStopping or GroupErrorStop state, and when MC_GroupReset function block is performed in GroupErrorStop state, the state of the relevant group is changed to GroupStandby.

*3 GroupStopping -> GroupStandby: when MC_GroupStop.DONE output is On and MC_SroupStop.EXECUTE input is Off

*4 GroupStandby -> GroupDisabled: in case there is no axis belonging to the group when performing the axis remove command (MC_RemoveAxisFromGroup, MC_UnGroupAllAxes)

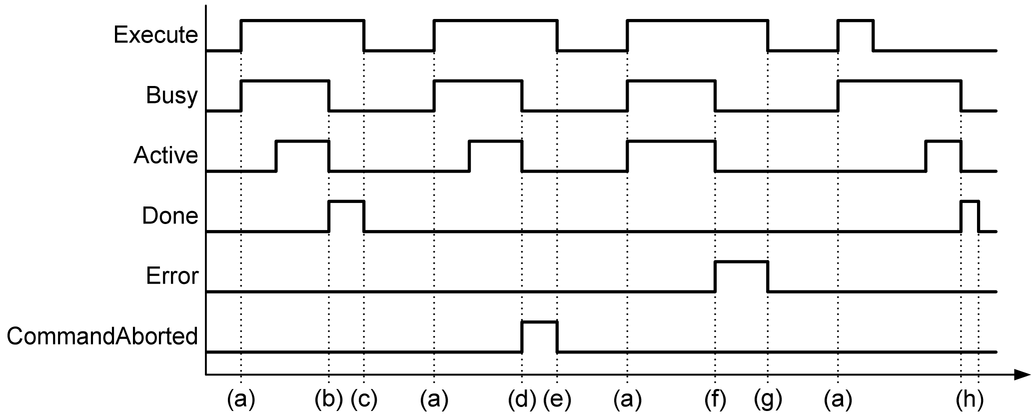
*5 GroupStandby: in case more than one axis belongs to the group when performing the axis add or remove command in group (MC_AddAxisToGroup, MC_RemoveAxisFromGroup)

*6 GroupDisabled: When performing MC_GroupDisable or MC_UnGroupAllDisable function block, the relevant group is changed to GroupDisabled state regardless of its current state.

16.1.3 Basic I/O Variable

Edge operation motion function block

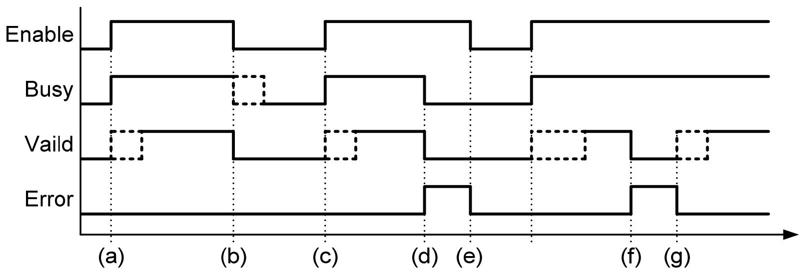
Relationships of the basic I/O parameter in the Edge operation motion function block are as below.



Variable	Description
Execute	This is an input to run the relevant function block in Edge operation function block. Function block is executed in the rising Edge. (Figure a state)
Busy	This is an output to indicate the relevant motion function block is currently running (= not completed), and this indicates the output of motion function block can be changed. Busy output is On in the rising Edge of Execute input (Figure a state), and it is Off when Done output is On (Figure b state), CommandAborted output is On (Figure d state), or Error output is On (Figure f state).
Active	This indicates the relevant motion function block is actually controlling axis. When running many motion function block to one axis (in case only one motion function block is controlling and other notion function blocks are Buffered), Active output is On in only one motion function block which is controlling, and in motion function blocks which are Buffered, Busy output is On.
Done	This is an output to indicate operation of the relevant motion function block has been successfully completed. If Done output is On, Busy and Active output is Off. (Figure d state) Done output is Off when Execute input is Off (Figure e state), if Execute output was Off when Done output became On, it remains On only during 1 scan (Figure h state).
Error	This is an output to indicate an error occurs while running motion function block. Error output is Off when Execute input is Off (Figure f state). If Execute output was Off when Error output became On, it remains On only during 1 scan (Figure h state).
ErrorID	This outputs error code regarding the relevant error when an error occurs while running motion function block. ErrorID output and elimination time are same with Error output.
CommandAborted	This indicates the relevant motion function block is interrupted by the other motion function

	block. CommandAborted output is Off when Execute input is Off (Figure g state). If Execute output was Off when Done output became On, it remains On only during one scan.
※ When Execute input is On in Edge operation(Execute input) motion function block, depending on the state of axis, one output in Busy, Done, Error, and CommandAborted output is On. Busy, Done, Error, and CommandAborted output are available to be On one at a time, and if one output in four is On, other three outputs become Off.	

▪ Motion function block for level motion



Variable	Description
Enable	This is an input to run function block for level operation motion. This runs motion function block in the rising Edge (Figure a state), and stops it in the falling Edge(Figure b state).
Busy	This is an output to indicate the relevant motion function block is currently running ((= not completed), and it indicates the output of motion function block can be changed. Busy output is On in the rising Edge of Enable input (Figure b state), and it remains on while motion function is in operation.
Valid	This is an output to indicate the relevant motion function block is successfully performed and output & motion are valid. Valid output is Off when Enable input is Off (Figure b state).
Error	This is an output to indicate an error occurs while running motion function block. If an error which cannot be automatically restored occurs while motion function block is in operation, Error output is On, Busy & Valid output is Off (Figure d state), and motion function block stops operating. Error output is Off when Enable input is Off (Figure e state). If an error which can be automatically restored occurs while function block is in operation, Error output is On and Valid input is Off (Figure f state). When the error in the relevant motion function block is restored, Error output is Off, and operation is resumed (Figure g state).
※ Valid and Error outputs are not On at the same time.	

▪ Axis input Note 1)

Each motion function block can be specified by Axis input to the axis which is subject to the relevant command. Motion control module can control 1-32 actual axes and 33~36 virtual axes, and 41-41 encoders can be used as main axis depending on motion function block. Therefore, values of 1~32, 33~36, and 1001~1002 can be input in

Axis input depending on motion function block. When it is out of the range which is available to set in each motion function block, "error 0x0006"occurs.

Note 1) The setting range of Axis input variable is explained based on XMC-E32A

16.1.4 BufferMode Input

This is an input which can specify whether to wait until the existing command is completed or to cancel the existing motion function block and execute the command in case the axis is already running other motion function block when running motion function block in a certain axis. The number between 0-5 can be specified, and if it is out of the range, "error 0x101A" occurs in the axis command and "error 0x201A" occurs in the axis group command. The values which are available to be set in BufferMode are as below.

Number	Buffer Mode	Explanation
0	mcAborting	Execute the command immediately. The existing command in operation is interrupted.
1	mcBuffered	Execute the command after the existing command in operation is completed.
2	mcBlendingLow	Do combined operation to combine the speeds of the existing command and command issuing to the low speed by comparing.
3	mcBlendingPrevious	Do combined operation to combine the speeds of the existing command.
4	mcBlendingNext	Do combined operation to combine the speeds of the command issuing.
5	mcBlendingHigh	Do combined operation to combine the speeds of the existing command and command giving to the high speed by comparing.

Note

In axis control, the maximum number that can be queued to the buffer is 100. An error (error code: 0x1022) occurs when executing a command in buffer mode more than this.

16.1.5 Changes in Parameters during Execution of Motion Function Block

The parameter of the relevant command can be changed at the time motion function block is running, and the detailed operations are as below.

- When executing Edge operation motion function block in the Off state of ContinuousUpdate input (turn On the Execute input), the relevant motion function block is operated by application of the parameter at the time when Execute input was On (rising Edge). In this case, the change of the parameter input value in the middle of execution of motion function block does not affect operation.
- When wanting to change the parameter while the relevant motion function block is in operation, change the parameter and turn On Execute input again.
- When executing Edge operation motion function block in the On state of ContinuousUpdate input (turn On the Execute input), the parameter of the time when Execute input was On (rising Edge) is applied at first.
- When changing the parameter while ContinuousUpdate input is On, the relevant motion function block operates reflecting the every change in parameter.

  But, if you change the parameter at the completion or after the stop of the operation of the relevant motion function block (Busy output is Off), the change is not reflected any more. (Parameter changing operation using ContinuousUpdate does not rerun the motion function block which is completed or interrupted, In other words, ContinuousUpdate operation is applied only to the motion function block which is currently running.)
- As for level operation motion function block, it is operated by the application of the parameter at the time when Enable input was On (rising Edge), and continuous change of parameter is available while Enable input is On.

16.1.6 Group Operation Route Change Settings

When the axis group of the current motion control module is executing a command, other command can be issued to the relevant axis group. At this point, the path, which the next command will achieve, can specify how the existing command will be connected to the existing path. The parameter of connection track is specified in TransitionParameter input.

Number	TRANSITION Mode	Explanation
0	TMNone	Do not generate a connection track.
3	TMCornerDistance	Generate a connection track which specifies the corner distance of a connection track and draws circular arcs at the specified corner distance.

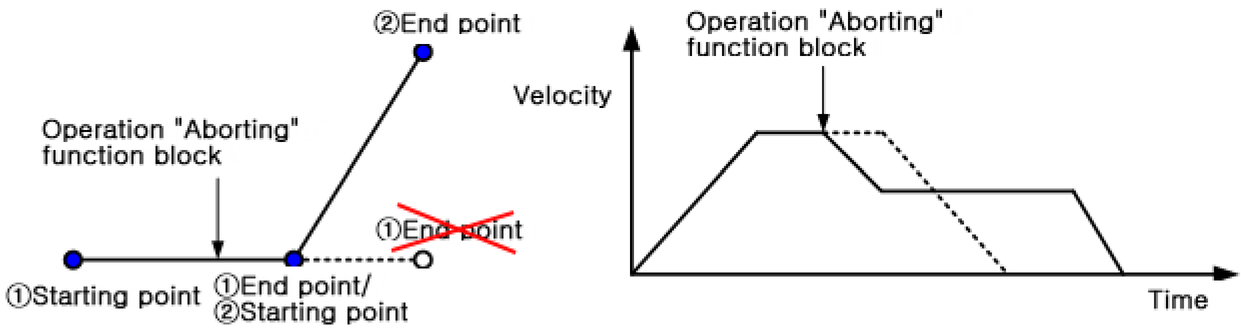
Note

In axis control, the maximum number that can be queued to the buffer is 100. An error (error code: 0x1022) occurs when executing a command in buffer mode more than this.

■ **TransitionMode "TMNone"**

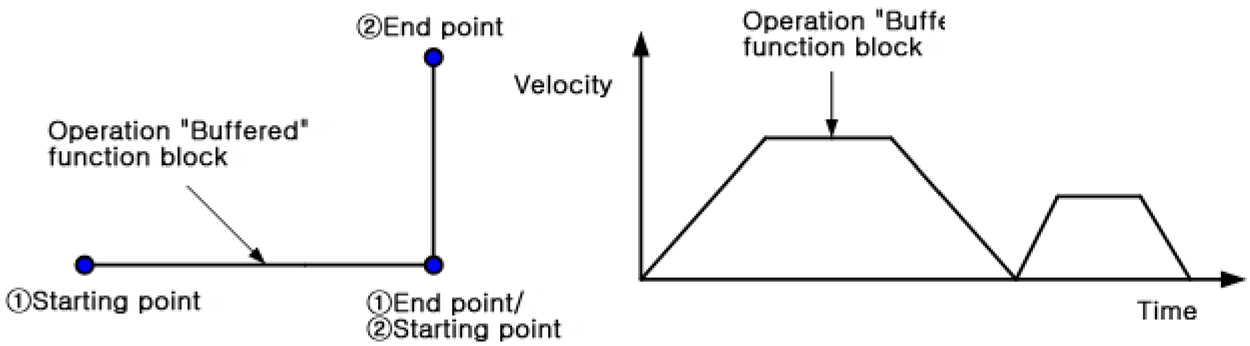
Connection track is not generated. TransitionMode input is available only to "TMNone" in case BufferMode input of motion function block is "Aborting" or "Buffered".

The Figure below shows the case when running BufferMode of motion function block in the setting of 'Aborting'. The Figure in the left shows that motion function block ② is executed in the setting of 'Aborting' while motion function block ① is running. Motion function block ① is forced to be terminated at 'end point ① / starting point ②' without reaching 'end point ①'. The Figure in the right shows that deceleration pause is performed at the moment of the execution of 'Aborting' function block, and the next motion function block is executed.



<In case BufferMode is specified as "Aborting">

The Figure below shows that the case when running BufferMode of motion function block in the setting of 'Buffered'. The Figure in the left shows that motion function block ② is executed in the setting of 'Buffered' while motion function block ① is running. Motion function block ② is executed after motion function block ① has reached target position. The Figure in the right shows that when 'Buffered' function block is executed, the next motion function block is executed after it reaches original target position.



<In case BufferMode is specified as "Buffered">

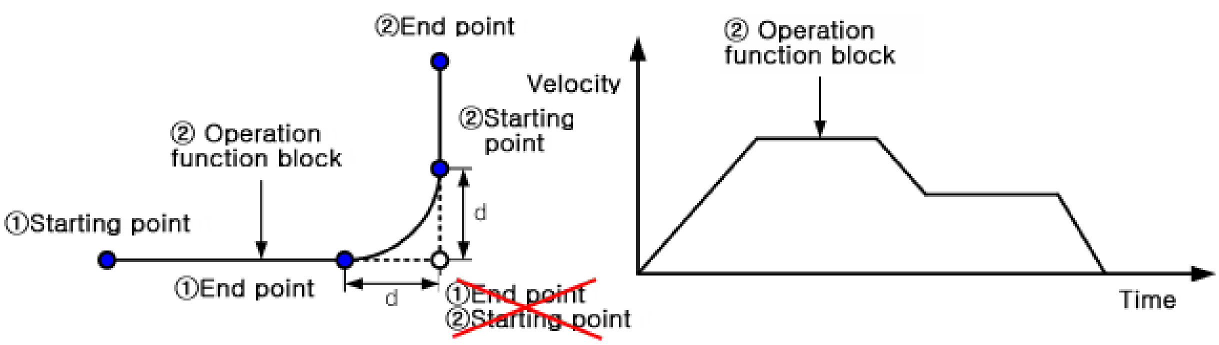
■ **TransitionMode "TMCornerDistance"**

The radius of a connection track is specified and the connection track which draws a circle having specified radius is output. This mode is operated only when BufferMode is "BlendingXXXX", and it is operated in "TMNone" when BufferMode is "Aborting" or "Buffered".

When drawing a connection track, the maximum speed of the path complies with the specified speed in BufferMode, and the length of radius complies with the value specified in TransitionParameter.

The Figure below shows the generation of a connection track which draws radius circle in two linear interpolation commands. The Figure in the left shows that motion function block ② is executed in the setting of "TMcornerDistance" while motion function block ① is running. The original target position of motion function block ① was end point ① / starting point ②, but straight-line motion is stopped and circular motion is started at the point ahead as far as radius 'd' (end point ①). Circular operation starts at end point ① and finishes at starting point ②, and executes motion function block ②.

The Figure in the right shows that the speed does not stop in the middle of two function blocks and continues.



<In case BufferMode is specified as "BlendingLow" and TransitionMode is specified as "TMCornerDistance">

16.1.7 Motion Function Block Errors

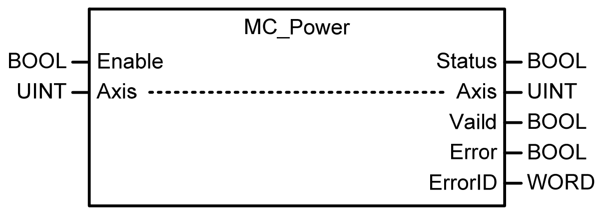
Errors occurring in ErrorID variable of motion function block are as follows.

STAT	Content	Detailed Description
0x0000	Normal	In case motion function block is normally executed, “O” is displayed on ErrorID.
0x0005	The current motion module does not support the motion function block.	The motion function block is not executed in the version of current module. Check the version in which the motion function block can be executed.
0x0006	Axis number of motion function block (Axisinput) exceeded allowable range.	Set axis and encoder number as product range.
0x0007	Axis group number of motion function block (AxisGroup input) exceeded allowable range.	Set axis group number to a value between 1 and 16.
0x0012	Internal execution error of motion function block occurred during the execution of the motion function block.	Check the version of XG5000 and XMC.-E32A
0x0013	Motion response error occurred during the execution of motion function block.	Check the version of XG5000 and XMC.-E32A
0x0020 : 0x0FFF	It indicates a common error of the motion control module. For more details, refer to ‘error information and measures in APPENDIX 1’.	
0x1000 : 0x1FFF	It indicates error that occurs in relation to axis control of motion control module. For more details, refer to ‘error information and measures in APPENDIX 1’.	
0x2000 : 0x2FFF	It indicates error that occurs in relation to axis control of motion control module. For more details, refer to ‘error information and measures in APPENDIX ‘.	

16.2 Motion Function Blocks

MC_Power	**Availability**
Servo On/OFF	**XMC**

Motion Function Block



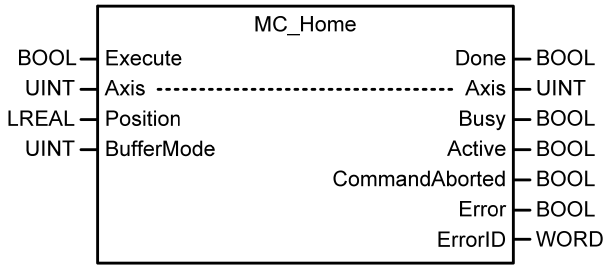
Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Enable	Servo motor of the relevant axis is servo On while input is activated.
Output		
BOOL	Status	Indicate the power permission status of the relevant axis.
BOOL	Valid	Indicate the validity of motion function block output. (same with Status output here)
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give servo On/Off command to the relevant axis.

(2) When Enable input is On, Servo On command is given to the relevant axis, and when it is Off, servo Off command is given.

(3) If servo On command is executed when the axis is in 'Disable' state, the axis state is 'StandStill', and failure in servo On brings 'ErrorStop' state.

MC_Home		Availability
Perform the search home		**XMC**
Motion Function Block		



Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Start the homing operation in rising Edge.
LREAL	Position	Specify the absolute position of axis when reference signal is detected.
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
Output		
BOOL	Done	Indicate the completion state of motion function block.
BOOL	Busy	Indicate that execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted by other command.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give a homing command to the relevant axis.

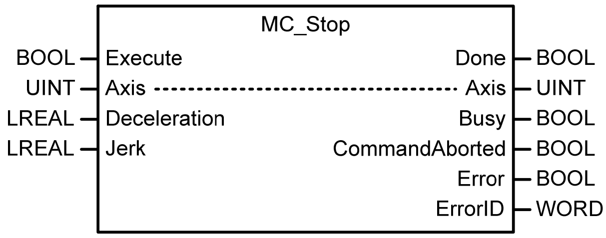
(2) Homing method is operated as specified in the operation parameter of the relevant axis in advance.

(3) As for Position input, absolute position of axis is specified when Reference Signal is detected or homing is completed.

(4) While this motion function block is running, the axis is 'Homing' state, and when the command is completed, it is switched to 'Standstill'.

MC_Stop	Availability
Stop immediately	**XMC**

Motion Function Block



Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give immediate stop command to the relevant axis in the rising Edge.
LREAL	Deceleration	Specify deceleration in time of stop. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
Output		
BOOL	Done	Indicate that the speed of the relevant axis reaches 0.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give an emergency stop command to the relevant axis.

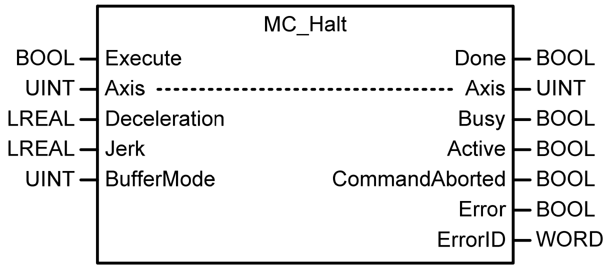
(2) When executing immediate stop (MC_Stop) motion function block, the existing motion function block being executed in the relevant axis is stopped, and the axis state changed to 'Stopping'. When the relevant axis is in 'Stopping' state, other motion function block cannot be executed in the relevant axis until the stopping is completed (until the Done output is activated).

(3) CommandAborted output indicates that the current motion function block is interrupted while it is running. Other motion function block cannot interrupt immediate stop (MC_Stop) motion function block while immediate stop (MC_Stop) motion function block is running, therefore, CommandAborted output is On in general when the power of servo is blocked or servo Off command is executed.

(4) If Execute input is On or the speed of axis is not 0, the axis is in 'Stopping' state, and when Done output is On and Execute input is Off, it is switched to 'Standstill' state.

MC_Halt	**Availability**
Halt	**XMC**

Motion Function Block



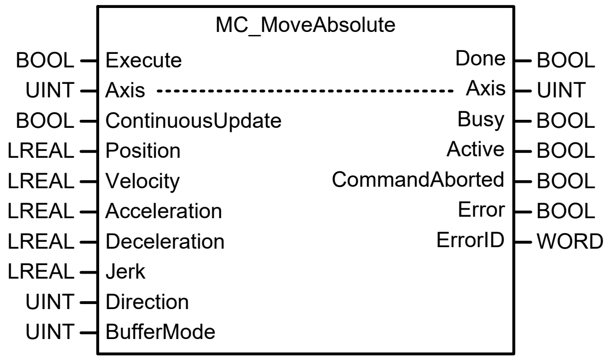
Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give stop command to the relevant axis in the rising Edge.
LREAL	Deceleration	Specify deceleration in time of stop. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
Output		
BOOL	Done	Indicate that the speed of the relevant axis reaches 0.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give a stop command to the relevant axis.

(2) The axis is 'DiscreteMotion' state while this motion function block is running, and when the speed of the relevant axis is 0, ‘Done’ output is On and changed to 'Standstill' state.

MC_MoveAbsolute	Availability
Absolute positioning operation	**XMC**

Motion Function Block



Input-Output

UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)

Input

BOOL	Execute	Give an absolute position operation command to the relevant axis in the rising Edge.
BOOL	ContinuousUpdate	Specify the update setting of input value. (Refer to 16.1.5.Changes in Parameters during Execution of Motion Function Block)
LREAL	Position	Specify the target position.
LREAL	Velocity	Specify the maximum speed. [u/s]
LREAL	Acceleration	Specify the acceleration. [u/s^2]
LREAL	Deceleration	Specify the deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	Direction	Specify the operation direction. (0~4: 0-Not specified, 1-Forward direction, 2-Shortest distance, 3-Reverse direction, 4-Current direction)
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)

Output

BOOL	Done	Indicate whether to reach the specified distance.
BOOL	Busy	Indicate that the execution of motion function block is not completed.

BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give the relevant absolute position operation commands.

(2) Operation direction of the axis in Infinite length repetition operation is set in Direction input, and if Infinite length repetition operation is set to Prohibited, Direction input is ignored. When Direction input is the shortest distance(=2), the relevant axis doing Infinite length repetition operation automatically selects the direction which allows the shortest distance. The available range is 0-4 (0-Not specified, 1-Forward direction, 2-Shortest distance, 3-Reverse direction, 4-Current direction), and "error 0x1017" occurs in case of excess of the range.

(3) On condition that there is no motion function block is on standby after the current motion function block, If the speed is 0 after reaching the target point, operation is completed and Done output is On.

(4) The axis is in 'DiscreteMotion' state while this motion function block is running, and it is switched to 'Standstill' state when operation is completed.

MC_MoveRelative	Availability
Relative positioning operation	**XMC**

Motion Function Block

	MC_MoveRelative		
BOOL	Execute	Done	BOOL
UINT	Axis ----------------------------------	Axis	UINT
BOOL	ContinuousUpdate	Busy	BOOL
LREAL	Distance	Active	BOOL
LREAL	Velocity	CommandAborted	BOOL
LREAL	Acceleration	Error	BOOL
LREAL	Deceleration	ErrorID	WORD
LREAL	Jerk		
UINT	BufferMode		

Input-Output

UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)

Input

BOOL	Execute	Give an absolute position operation command to the relevant axis in the rising Edge.
BOOL	ContinuousUpdate	Specify the update setting of input value. (Refer to 16.1.5.Changes in Parameters during Execution of Motion Function Block)
LREAL	Distance	Specify the target distance.
LREAL	Velocity	Specify the maximum speed. [u/s]
LREAL	Acceleration	Specify the acceleration. [u/s^2]
LREAL	Deceleration	Specify the deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)

Output

BOOL	Done	Indicate whether to reach the specified distance.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give relative position operation command to the relevant axis.

(2) Relative position motion (MC_MoveRelative) is the motion function block which moves as far as the target distance specified in Distance input from the current position.

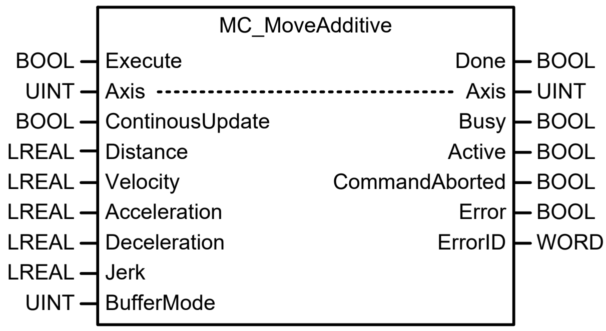
(3) Moving direction is decided depending on the sign of the target distance specified in Distance input, and positive (+ or No sign) moving direction leads to the forward direction, and negative (-) moving direction leads to the reverse direction.

(4) If there is no motion function block is on standby after the current motion function block and the speed is 0 after moving to the target distance, operation is completed and Done output is On.

(5) The axis is in "DiscreteMotion" state when this motion function block is running, and it is switched to "StandStill" state when operation is completed.

MC_MoveAdditive	**Availability**
Additive positioning operation	**XMC**

Motion Function Block



Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give an absolute position operation command to the relevant axis in the rising Edge.
BOOL	ContinuousUpdate	Specify the update setting of input value. (Refer to 16.1.5.Changes in Parameters during Execution of Motion Function Block)
LREAL	Distance	Specify the target distance.
LREAL	Velocity	Specify the maximum speed. [u/s]
LREAL	Acceleration	Specify the acceleration. [u/s^2]
LREAL	Deceleration	Specify the deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
Output		
BOOL	Done	Indicate whether to reach the specified distance.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give the relevant additive position operation commands.
(2) Additive position motion (MC_MoveAdditive) is the motion function block which additionally moves as far as the position specified in Distance input from the final target position of the currently running motion function block or the latest motion function block executed in 'DiscreteMotion' state. If the current axis is executing motion function block 'ContinuousMotion' state, it executes operation based on the position where additive position motion (MC_MoveAdditve) is executing.
(3) Moving direction is decided depending on the sign of the specified target distance in Distance input, and positive (+ or No sign) moving direction leads to forward direction, and negative (-) moving direction leads to reverse direction.
(4) When reaching the target position without motion function block on standby after the current motion function block, 'Done' output is On.
(5) The axis is in 'DiscreteMotion' state while this motion function block is running, and it is switched to 'Standstill' state when operation is completed.

MC_MoveVelocity	Availability
Specified velocity operation	**XMC**

Motion Function Block

```
                   MC_MoveVelocity
BOOL  ─ Execute                        InVelocity ─ BOOL
UINT  ─ Axis ---------------------------------- Axis ─ UINT
BOOL  ─ ContinuousUpdate                     Busy ─ BOOL
LREAL ─ Velocity                           Active ─ BOOL
LREAL ─ Acceleration             CommandAborted ─ BOOL
LREAL ─ Deceleration                        Error ─ BOOL
LREAL ─ Jerk                              ErrorID ─ WORD
UINT  ─ Direction
UINT  ─ BufferMode
```

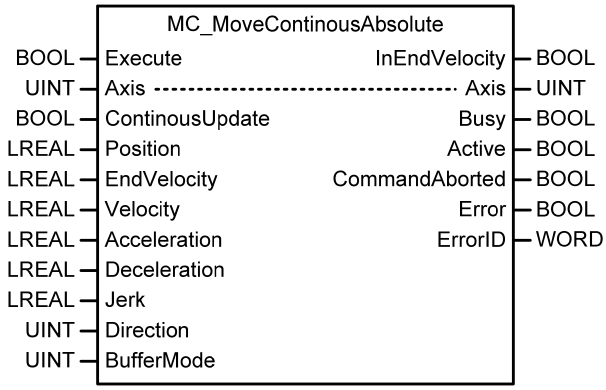
Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give an absolute position operation command to the relevant axis in the rising Edge.
BOOL	ContinuousUpdate	Specify the update setting of input value. (Refer to 16.1.5.Changes in Parameters during Execution of Motion Function Block)
LREAL	Velocity	Specify the maximum speed. [u/s]
LREAL	Acceleration	Specify the acceleration. [u/s^2]
LREAL	Deceleration	Specify the deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	Direction	Specify the operation speed. (1 ~ 3 : 1-Forward direction, 2-Reverse direction, 3-Current direction)
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
Output		
BOOL	InVelocity	Indicate whether to reach the specified speed.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.

BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give specified velocity operation command to the relevant axis.
(2) Giving a stop command or execution of other motion function block allow to interrupt specified velocity motion.
(3) Specify the operation speed in Velocity input. Positive sign (+ or No sign) of the operation speed value leads to forward direction, and negative (-) sign leads to reverse direction.
(4) Specify the operation direction in Direction input. But, the operation direction is affected by the sign of the specified speed value by Velocity input. For example, if you specify the negative number for the Velocity value and reverse direction for Direction input, the relevant axis lastly does forward direction operation.
(5) Output InVelocity is On when the relevant axis reaches the specified speed, and it is Off when the specified speed operation is interrupted.
(6) The axis is in 'ContinuousMotion' state when this motion function block is running.

MC_MoveContinuousAbsolute	Availability
Absolute position operation ending with specified velocity operation	**XMC**

Motion Function Block



Input-Output

UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)

Input

BOOL	Execute	Give an absolute position operation command to the relevant axis in the rising Edge.
BOOL	ContinuousUpdate	Specify the update setting of input value. (Refer to 16.1.5.Changes in Parameters during Execution of Motion Function Block)
LREAL	EndVelocity	Specify the operation speed after reaching the target position. [u/s]
LREAL	Velocity	Specify the maximum speed to reach the target position. [u/s]
LREAL	Acceleration	Specify the acceleration. [u/s^2]
LREAL	Deceleration	Specify the deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	Direction	Specify the operation direction. (0~4: 0-Not specified, 1-Forward direction, 2-Shortest distance, 3-Reverse direction, 4-Current direction)
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)

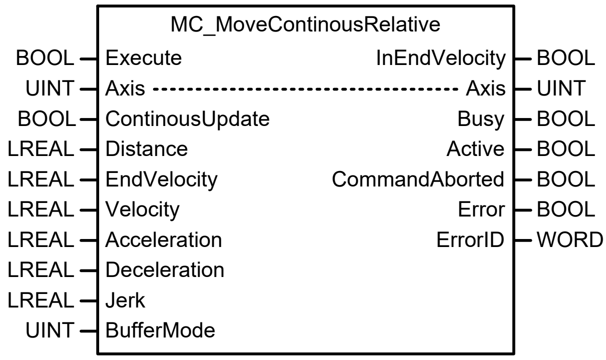
Output

BOOL	InEndVelocity	Indicate the operation at the specified speed after reaching the target position.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give Specified velocity operation after relative position operation command to the relevant axis.
(2) When executing MC_MoveContinuousAbsolute, the relevant axis moves to the position specified in Position and operates at the specified speed in EndVelocity if there is no motion function block is on standby.
(3) Giving a stop command or execution of other motion function block allow to interrupt speed operation.
(4) Set the operation direction of the axis in infinite length repetition operation in Direction input, and if infinite length repetition operation is set to Prohibited, Direction input is ignored. When Direction input is the shortest distance (=2), the relevant axis selects the direction which allows the shortest distance and operates if it does infinite length repetition operation. The range can be set to 0~4(0-No specified, 1-Forward direction, 2-Shortest distance, 3-Reverse direction, 4-Current direction), if the value outside the range is set and motion function block is executed, Error is On and “0x1017” occurs in ErrorID.
(5) Output InEndVelocity is on when the relevant axis starts speed operation after reaching the specified position, and when the specified operation is interrupted, it is Off.
(6) The axis is in 'ContinuousMotion' state while this command is executing.

MC_MoveContinuousRelative	**Availability**
Relative position operation ending with specified velocity operation	**XMC**

Motion Function Block



Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give an absolute position motion command to the relevant axis in the rising Edge.
BOOL	ContinuousUpdate	Specify the update setting of input value. (Refer to 16.1.5.Changes in Parameters during Execution of Motion Function Block)
LREAL	Distance	Specify the target distance.
LREAL	EndVelocity	Specify the operation speed after reaching the target position. [u/s]
LREAL	Velocity	Specify the maximum speed to reach the target position. [u/s]
LREAL	Acceleration	Specify the acceleration. [u/s^2]
LREAL	Deceleration	Specify the deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
Output		
BOOL	InEndVelocity	Indicate the operation at the specified speed after reaching the target position.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.

BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block gives MC_MoveContinuousRelative command to the relevant axis.
(2) When executing MC_MoveContinuousRelative, the relevant axis operates at the speed specified in EndVelocity after moving the distance specified in Distance if there is no motion function block is on standby.
(3) Giving a stop command or operation of other motion function block allow to interrupt specified velocity motion.
(4) Output InEndVelocity is On when the relevant axis starts speed operation and reaches the specified speed after moving the specified distance, and when specified velocity motion is interrupted, it is Off.
(5) The axis is in 'ContinuousMotion' state while this motion function block is running.

# MC_TorqueControl	Availability
Torque control	**XMC**

Motion Function Block

MC_TorqueControl			
BOOL	Execute	InTorque	BOOL
UINT	Axis	Axis	UINT
BOOL	ContinousUpdate	Busy	BOOL
LREAL	Torque	Active	BOOL
LREAL	TorqueRamp	CommandAborted	BOOL
LREAL	Velocity	Error	BOOL
LREAL	Acceleration	ErrorID	WORD
LREAL	Deceleration		
LREAL	Jerk		
UINT	Direction		
UINT	BufferMode		

Input-Output

UINT	Axis	Specify the axis to be commanded (1~32: real axis)

Input

BOOL	Execute	Give an absolute position operation command to the relevant axis in the rising Edge.
BOOL	ContinuousUpdate	Specify the update setting of input value. (Refer to 16.1.5.Changes in Parameters during Execution of Motion Function Block)
LREAL	Torque	Specify the target torque. [u]
LREAL	TorqueRamp	Specify the ascending slope of torque. [u/s]
LREAL	Velocity	Unused
LREAL	Acceleration	Unused
LREAL	Deceleration	Unused
LREAL	Jerk	Unused
UINT	Direction	Specify the operation direction. (1~2 : 1-Forward direction, 2-Reverse direction)
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)

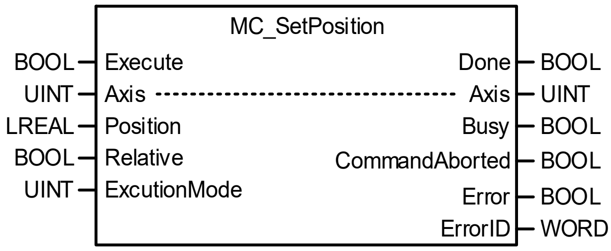
Output

BOOL	InTorque	Indicate that the input torque value and currently operating torque value are same.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give torque control command to the relevant axis.
(2) When executing torque control (MC_Torque), the relevant axis performs the control to keep the torque value specified in Torque input.
(3) Giving a stop command or operation of other motion function block allow to interrupt specified velocity motion.
(4) Specify the gradient to reach the target torque value in TorqueRamp input.
(5) Specify the maximum speed in torque control operation in Speed input, and the value in negative number is not allowed. Rotation direction is decided depending on the size of load in torque and the relative axis.
(6) Specify the operation direction in Direction input. When setting the value outside the range and executing motion function block, Error is On and “0x1017” occurs in ErrorID.
(7) Output InTorque is On when the relevant axis reaches the specified torque, and when torque control operation is interrupted, it is Off.
(8) The axis is in 'ContinuousMotion' state when this motion function block is running.

MC_SetPosition	Availability
Setting the current position	**XMC**

Motion Function Block



Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)

Input		
BOOL	Execute	Specify the current position of the relevant axis in the rising Edge.
LREAL	Postion	Specify the position.
BOOL	Relative	0: Position value=Absolute position, 1: Position value=Relative position
UINT	ExecutionMode	0: Immediately applied the position value, 1: Applied at the same point with ‘Buffered’ of Buffermode

Output		
BOOL	Enabled	Indicate that override rate is successfully applied.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to set the current position of the relevant axis.

(2) Specify the position in Position input. When executing motion function block, if Relative input is Off, the position of the relevant axis is replaced by the value of Position input, and if Relative input is On, the value of Position input is added to the current position of the relevant axis.

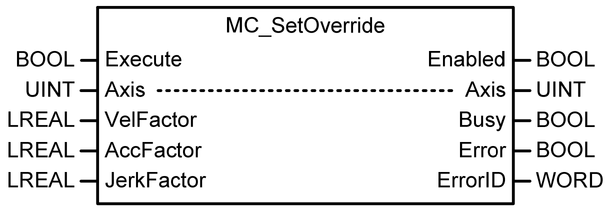
(3) ExcutionMode input specifies the setting point. 0 means to be set immediately after motion function block, and 1 means to be set at the same point with ‘Buffered’ in sequential operation setting. The value unable to be set causes "error0x101B”.

0 (mcImmediately): Change the parameter value immediately after executing function block (rising Edge in Execute input). If the relevant axis is in running, operation can be affected.

1 (mcQueued): Changed at the same point with ‘Buffered’ in Buffermode. (**Error! Reference Source Not Found**. Refer to input)

MC_SetOverride	Availability
Velocity/Acceleration override	**XMC**

Motion Function Block

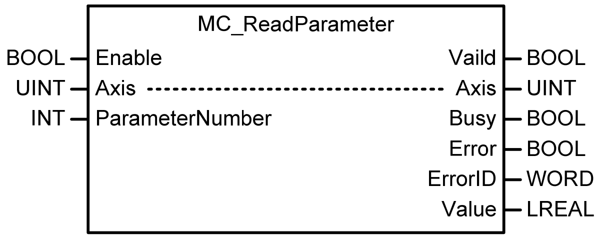


Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Enable	Execute override operation in the relevant axis while input is activated.
LREAL	VelFactor	Specify the override rate of speed.
LREAL	AccFactor	Specify the override rate of acceleration/deceleration.
LREAL	JerkFactor	Specify the override rate of the change rate of acceleration.
Output		
BOOL	Enabled	Indicate that override rate is successfully applied.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to override the speed of the relevant axis, acceleration, and the change rate of acceleration.
(2) Override rate which is applied to the relevant axis can be specified and changed while Enable input is On. If Enable input is Off, override rate right before the Off is maintained.
(3) Speed override rate is specified in VelFactor input. If the specified value is 0.0, the relevant axis stops but it is not changed to 'StandStill' state.
(4) Specify acceleration/deceleration and override rate of jerk (change rate of acceleration) in AccFactor and JerkFactor input respectively.
(5) Negative number cannot be input in each Facotr, and if it is input, "error 0x10C1” occurs.
(6) Default of each override rate is 1.0, and it means 100% of the command speed of function block currently running.
(7) Override operation does not affect the serve axis of the relevant axis.

MC_ReadParameter	Availability
Read Parameter	**XMC**

Motion Function Block



Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Enable	Execute override operation in the relevant axis while input is activated.
INT	ParameterNumber	Specify the number of parameter to read. (0 ~ 25)
Output		
BOOL	Vaild	Indicate whether the output of the current motion function block is valid.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
LREAL	Value	Output the value of parameter.

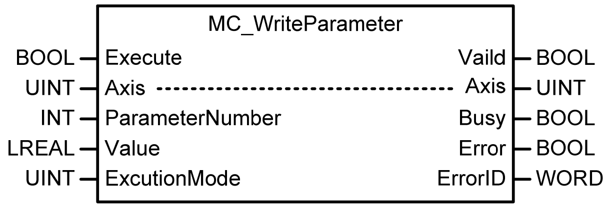
(1) This command is a motion function block which outputs parameter of the relevant axis.
(2) The value of the relevant parameter is continuously output in Value while Enable input is On.
(3) Specify the number of parameter to read in ParameterNumber input.
(4) The numbers of parameter are as below.

No.	Parameter	Item	Description
0	Basic Parameter	Unit	0:pulse,1:mm,2:inch,3:degree
1		Purses per rotation	1 ~ 4,294,967,295 [pulse]
2		Travel per rotation	0.000000001 ~ 4,294,967,295 [Unit]
3		Speed command unit	0:Unit/Time, 1:rpm
4		Speed limit	LREAL Positive number [Unit/s, rpm] (Change according to Unit, Pulses per rotation, Travel per rotation, Speed command unit)
5		Emergency stop deceleration	0 or LREAL Positive number [Unit/s^2]
6		Encoder select	0:Incremental Encoder,1:Absolute Encoder
7		Gear ratio(Motor)	1 ~ 65,535
8		Gear ratio(Machine)	1 ~ 65,535
9		Operating mode of the reverse rotation	0:E.Stop, 1:Stop
10	Extented Parameter	S/W upper limit	LREAL [Unit]
11		S/W lower limit	LREAL [Unit]
12		Infinite running repeat position	LREAL Positive number [Unit]
13		Infinite running repeat	0:Disable, 1:Enable
14		Command Inposition range	0 or LREAL Positive number [Unit]
15		Tracking error over-range value	0 or LREAL Positive number [Unit]
16		Current position compensation amount	0 or LREAL Positive number [Unit]
17		Current speed filter time constant	0 ~ 100
18		Error reset monitoring time	1 ~ 1000 [ms]
19		S/W limit during speed control	0:Don't detect, 1:Detect
20		Tracking error level	0:Warning, 1:Alarm
21		JOG high Speed	LREAL Positive number [Unit] (Jog low speed ~speed limit) [Unit/s]
22		JOG low Speed	LREAL Positive number [Unit] (< Jog high speed) [Unit/s]
23		JOG acceleration	0 or LREAL Positive number [Unit/ s^2]
24		JOG deceleration	0 or LREAL Positive number [Unit/ s^2]
25		JOG jerk	0 or LREAL Positive number [Unit/ s^2]
26		Override mode	0: Specified by ratio, 1: Specified by unit

No.	Parameter	Item	Description
100	Encoder Parameter	Encorder1 unit	0: pulse, 1: mm, 2: inch, 3:degree
101		Encorder1 pulse per rotation	1 ~ 4294967295
102		Encorder1 travel per rotation	0.000000001 ~ 4294967295
103		Encorder1 pulse input	0:CW/CCW 1 multiplier, 1:PULSE/DIR 1 multiplier 2:PULSE/DIR 2 multiplier, 3:PHASE A/B 1 multiplier 4:PHASE A/B 2 multiplier, 5: PHASE A/B 4multiplier
104		Encorder1 max. value	(Enc1 min. value+1) ~ 2147483647
105		Encorder1 min. value	-2147483648~(Enc1 max. vlaue-1)
106		Encoder1 Input filter value	0: not used, 1: 500kPPS 2: 200kPPS, 3: 100kPPS 4: 10kPPS, 5: 1kPPS 6: 0.1kPPS
200		Encorder2 unit	0: pulse, 1: mm, 2: inch, 3:degree
201		Encorder2 pulse per rotation	1 ~ 4294967295
202		Encorder2 travel per rotation	0.000000001 ~ 4294967295
203		Encorder2 pulse input	0:CW/CCW 1 multiplier, 1:PULSE/DIR 1 multiplier 2:PULSE/DIR 2 multiplier, 3:PHASE A/B 1 multiplier 4:PHASE A/B 2 multiplier, 5: PHASE A/B 4multiplier
204		Encorder1 max. value	(Enc2 min. value+1) ~ 2147483647
205		Encorder1 min. value	-2147483648~(Enc2 max. value-1)
206		Encoder 2 Input filter value	0: not used, 1: 500kPPS 2: 200kPPS, 3: 100kPPS 4: 10kPPS, 5: 1kPPS 6: 0.1kPPS

MC_WriteParameter	Availability
Write Parameter	**XMC**

Motion Function Block



Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Rising Edge corresponding parameters of input is written. .
INT	ParameterNumber	Specify the number of parameter to write. (0 ~ 25)
LREAL	Value	Specify the value of parameter to write.
UINT	ExecutionMode	Specify the time when parameter is written.
Output		
BOOL	Vaild	Indicate whether parameter is successfully written.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to write the value specified in parameter of the relevant axis.
(2) Parameter is written in the rising Edge of Execute input.
(3) Specify the number of parameter to write in ParameterNumber input. The value unable to be set causes "error 0x10F0".
(4) Specify the value to write in parameter for Value input.
(5) In ExecutionMode, correct the time when parameter is written and the values below can be set. The value unable to be set causes "error 0x101B".

0 (mcImmediately): Change the parameter value immediately after executing function block (rising Edge in Execute input). If the relevant axis is in running, operation can be affected.

1 (mcQueued): Changed at the same point with 'Buffered' in Buffermode. (**Error! Reference Source Not Found**. Refer to input)

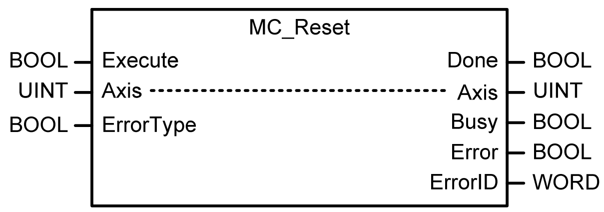
(6) The numbers of parameter are as below.

No.	Parameter	Item	Description
0	Basic Parameter	Unit	0:pulse,1:mm,2:inch,3:degree
1		Purses per rotation	1 ~ 4,294,967,295 [pulse]
2		Travel per rotation	0.000000001 ~ 4,294,967,295 [Unit]
3		Speed command unit	0:Unit/Time, 1:rpm
4		Speed limit	LREAL Positive number [Unit/s, rpm] (Change according to Unit, Pulses per rotation, Travel per rotation, Speed command unit)
5		Emergency stop deceleration	0 or LREAL Positive number [Unit/s^2]
6		Encoder select	0:Incremental Encoder,1:Absolute Encoder
7		Gear ratio(Motor)	1 ~ 65,535
8		Gear ratio(Machine)	1 ~ 65,535
9		Operating mode of the reverse rotation	0:E.Stop, 1:Stop
10	Extented Parameter	S/W upper limit	LREAL [Unit]
11		S/W lower limit	LREAL [Unit]
12		Infinite running repeat position	LREAL Positive number [Unit]
13		Infinite running repeat	0:Disable, 1:Enable
14		Command Inposition range	0 or LREAL Positive number [Unit]
15		Tracking error over-range value	0 or LREAL Positive number [Unit]
16		Current position compensation amount	0 or LREAL Positive number [Unit]
17		Current speed filter time constant	0 ~ 100
18		Error reset monitoring time	1 ~ 1000 [ms]
19		S/W limit during speed control	0:Don't detect, 1:Detect
20		Tracking error level	0:Warning, 1:Alarm
21		JOG high Speed	LREAL Positive number [Unit] (Jog low speed ~speed limit) [Unit/s]
22		JOG low Speed	LREAL Positive number [Unit] (< Jog high speed) [Unit/s]
23		JOG acceleration	0 or LREAL Positive number [Unit/ s^2]
24		JOG deceleration	0 or LREAL Positive number [Unit/ s^2]
25		JOG jerk	0 or LREAL Positive number [Unit/ s^2]
26		Override mode	0: Specified by ratio, 1: Specified by unit

No.	Parameter	Item	Description
100	Encoder Parameter	Encorder1 unit	0: pulse, 1: mm, 2: inch, 3:degree
101		Encorder1 pulse per rotation	1 ~ 4294967295
102		Encorder1 travel per rotation	0.000000001 ~ 4294967295
103		Encorder1 pulse input	0:CW/CCW 1 multiplier, 1:PULSE/DIR 1 multiplier 2:PULSE/DIR 2 multiplier, 3:PHASE A/B 1 multiplier 4:PHASE A/B 2 multiplier, 5: PHASE A/B 4multiplier
104		Encorder1 max. value	(Enc1 min. value+1) ~ 2147483647
105		Encorder1 min. value	-2147483648~(Enc1 max. vlaue-1)
106		Encoder1 Input filter value	0: not used, 1: 500kPPS 2: 200kPPS, 3: 100kPPS 4: 10kPPS, 5: 1kPPS 6: 0.1kPPS
200		Encorder2 unit	0: pulse, 1: mm, 2: inch, 3:degree
201		Encorder2 pulse per rotation	1 ~ 4294967295
202		Encorder2 travel per rotation	0.000000001 ~ 4294967295
203		Encorder2 pulse input	0:CW/CCW 1 multiplier, 1:PULSE/DIR 1 multiplier 2:PULSE/DIR 2 multiplier, 3:PHASE A/B 1 multiplier 4:PHASE A/B 2 multiplier, 5: PHASE A/B 4multiplier
204		Encorder1 max. value	(Enc2 min. value+1) ~ 2147483647
205		Encorder1 min. value	-2147483648~(Enc2 max. value-1)
206		Encoder 2 Input filter value	0: not used, 1: 500kPPS 2: 200kPPS, 3: 100kPPS 4: 10kPPS, 5: 1kPPS 6: 0.1kPPS

MC_Reset	Availability
Reset axis error	**XMC**

Motion Function Block



Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Reset the axis error in the rising Edge of input.
BOOL	ErrorType	The types of error to be reset (0: Axis error, 1: Common error)
Output		
BOOL	Done	Indicate whether the axis error is successfully reset.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

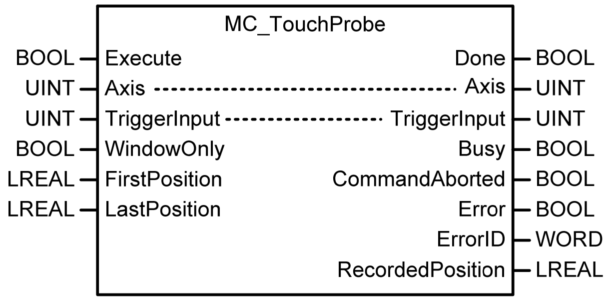
(1) This motion function block is to reset the error of the relevant axis. When setting ErrorType to '0' and executing motion function block in case the relevant axis is in ' ErrorStop' state, every axis error is reset and the axis state is switched to 'StandStill' or 'Disabled' state.

(2) If ErrorType is set to ‘1’ and motion function block is executed, common error occurred in the relevant module is reset.

(3) Motion function block is executed in the rising Edge of Execute input.

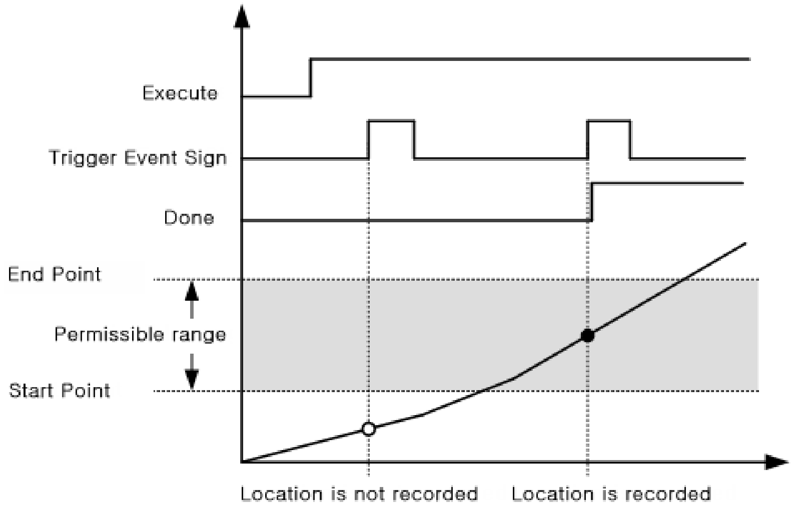
MC_TouchProbe	Availability
Touch probe	**XMC**

Motion Function Block



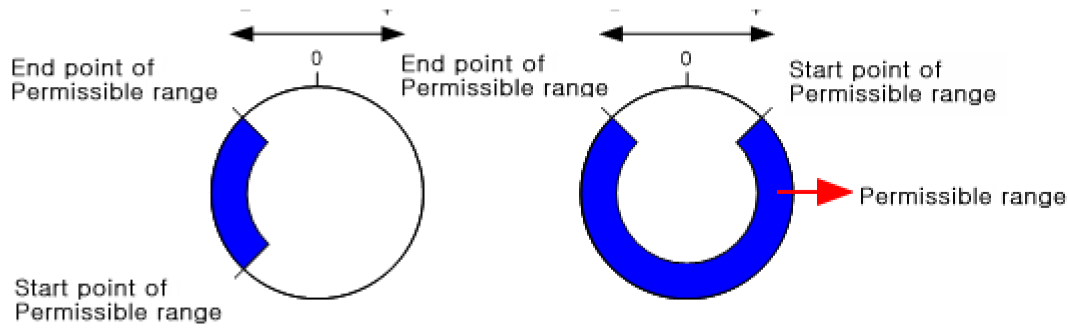
Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis)
UINT	TriggerInput	Specify the signal to be used as a trigger. (0: TouchProbe 1, 1: TouchProbe 2)
Input		
BOOL	Execute	TouchProbe function starts at the rising Edge of input.
BOOL	WindowOnly	Activate the window mode.
LREAL	FirstPosition	Specify the starting position of allowable area in the window mode.
LREAL	LastPosition	Specify the end position of allowable area in the window mode.
Output		
BOOL	Done	Indicate that the trigger signal is successfully recorded.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted by other command.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
REAL	RecordedPosition	Output the axis position where the trigger occurs.

(1) This motion function block is to execute 'TouchProbe' function which records the axis position at the time when the trigger event occurs.
(2) TouchProbe function starts at the rising Edge of Execute input.
(3) Specify the signal to be used as a trigger in TriggerInput. The value unable to be set causes "error 0x10E1".
(4) When activating the window mode, allowable area where accepts the trigger signal of axis can be set. Operation timing of each signal when the window mode is activated is as below.

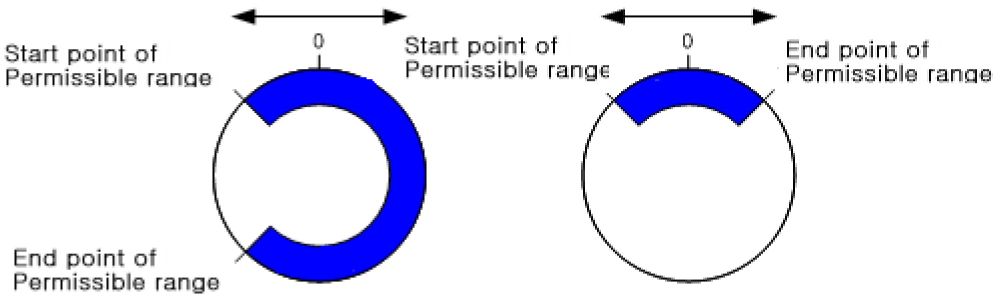


< In case TouchProbe function is the window mode, Operation timing >

● In case of Permissible range start point < Permissible range end point

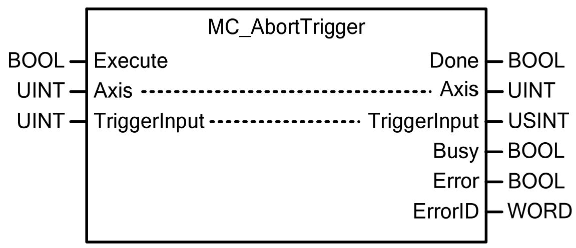


● In case of Permissible range start point > Permissible range end point



MC_AbortTrigger	**Availability**
Abort trigger events	**XMC**

Motion Function Block

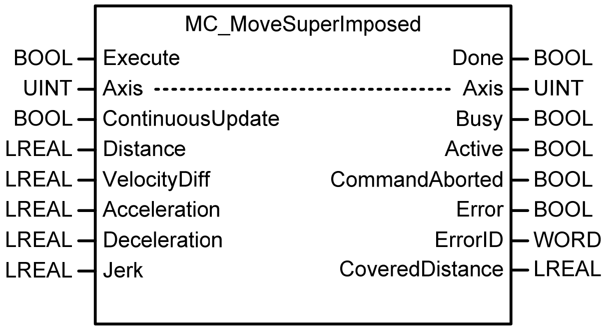


Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis)
UINT	TriggerInput	Specify the trigger signal to be disengaged. (0: TouchProbe 1, 1: TouchProbe 2)
Input		
BOOL	Execute	The trigger on standby in the relevant axis in the rising Edge is disengaged.
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to disengage the trigger which is on standby in the relevant axis.
(2) Specify the trigger signal to be disengaged in TriggerInput. The value unable to be set causes "error 0x10E1”.

MC_MoveSuperImposed	Availability
SuperImposed Operation	**XMC**

Motion Function Block



Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give a SuperImposed operation command to the relevant axis in the rising Edge.
BOOL	ContinuousUpdate	Specify the update setting of input value. (Refer to 6.1.5.Changes in Parameters during Execution of Motion Function Block)
LREAL	Distance	Specify the target distance. [u]
LREAL	VelocityDiff	Specify the added velocity. [u/s]
LREAL	Acceleration	Specify the added acceleration. [u/s^2]
LREAL	Deceleration	Specify the added deceleration. [u/s^2]
LREAL	Jerk	Specify the added change rate of acceleration/deceleration. [u/s^3]
Output		
BOOL	Done	Indicate whether to reach the specified distance.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted by other command
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
LREAL	CoveredDistance	Indicate the distance moved with SuperImposed operation after SuperImposed command.

(1) This motion function block is a command issuing aSuperImposed operation order to the relevant axis.

(2) SuperImposed is a command ordering to move from the current position at the time of the command to the target

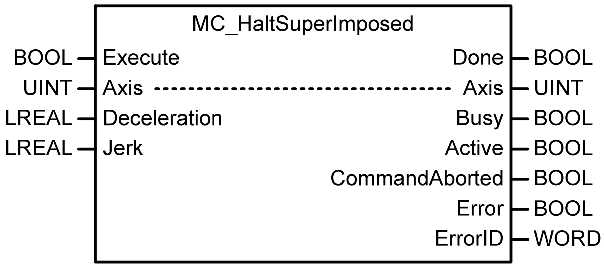
distance set by Distance input.

(3) The direction of the movement is determined by the positivity/negativity of the set distance. Positive distance (+ or no sign) means forward movement, and negative distance (-) means reverse movement.

(4) After moving the target distance, when the velocity reaches 0, the command is completed and Doneoutput is on.

MC_HaltSuperImposed	Availability
SuperImposed Operation Halt	**XMC**

모션 펑션 블록 형태



입력-출력		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis
입력		
BOOL	Execute	Give a SuperImposed operation halt command to the relevant axis in the rising Edge.
LREAL	Deceleration	Specify deceleration in time of stop. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
출력		
BOOL	Done	Indicate that the speed of the relevant axis reaches 0.
BOOL	Busy	Indicate that the execution of function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Output the number of error occurred while motion function block is running.
WORD	ErrorID	Indicate the distance moved with SuperImposed operation after SuperImposed command.

(1) This motion function block is a command issuing an order to halt SuperImposed operation to the relevant axis.
(2) Halt command for SuperImposed operation is a command ordering to decelerate and halt at a given acceleration and jerk at the time of performing the command.
(3) After moving the target distance, when the velocity reaches 0, the command is completed and Done output is on.

MC_CamIn	Availability
Camming run	**XMC**

Motion Function Block

```
                         MC_CamIn
BOOL  ─ Execute                            InSync ─ BOOL
UINT  ─ Master ---------------------------- Master ─ UINT
UINT  ─ Slave  -----------------------------  Slave ─ UINT
LREAL ─ ContinousUpdate                      Busy ─ BOOL
LREAL ─ MasterOffset                       Active ─ BOOL
LREAL ─ SlaveOffset               CommandAborted ─ BOOL
LREAL ─ MasterScaling                       Error ─ BOOL
LREAL ─ SlaveScaling                      ErrorID ─ WORD
LREAL ─ MasterStartDistance          EndOfProfile ─ BOOL
LREAL ─ MasterSyncPosition
UINT  ─ StartMode
UINT  ─ MasterValueSource
UINT  ─ CamTableID
UINT  ─ BufferMode
```

Input-Output

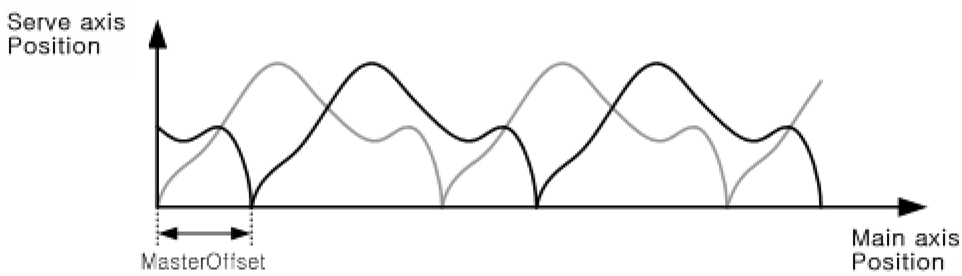
UINT	Master	Set the main axis. (1~32: Actual axes, 33~36: Virtual axes, 1001~1002: Encoders)
UINT	Slave	Set the the serve axis. (1~32: Actual axes, 33~36: Virtual axes)

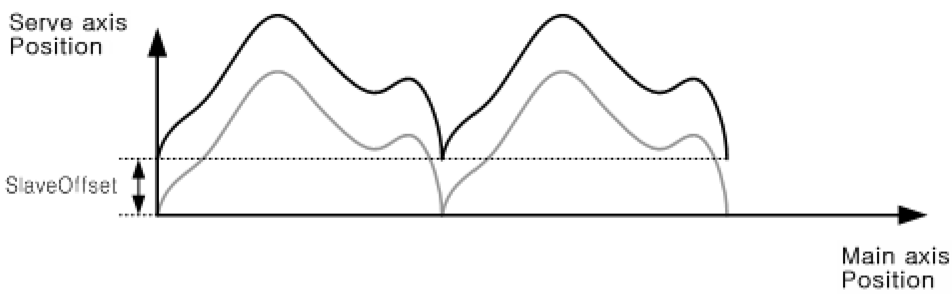
Input

BOOL	Execute	Give cam operation command to the relevant axis in the rising Edge.
BOOL	ContinuousUpdate	Specify the update setting of input value. (Refer to 16.1.5.Changes in Parameters during Execution of Motion Function Block)
LREAL	MasterOffset	Set the offset value of the main axis.
LREAL	SlaveOffset	Set the offset value of the the serve axis cam table.
LREAL	MasterScaling	Specify the magnification of the main axis.
LREAL	SlaveScaling	Specify the magnification of the serve axis cam table.
LREAL	MasterStartDistance	Specify the position of the main axis where cam operation of the slave.
LREAL	MasterSyncPosition	Specify the starting point at cam table when cam operation starts.
UINT	StartMode	Set the cam operation mode. 0 : Cam table is applied as an absolute value (mcAbsolute) 1: Cam table is applied as a relative value based on the command starting point (mcRelative)

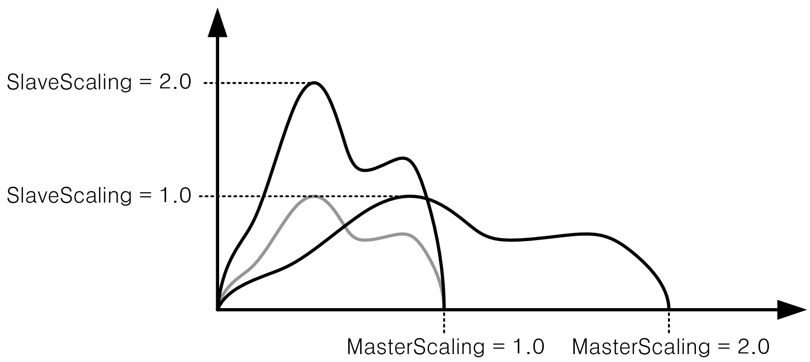
UINT	MasterValueSource	Select the source of the main axis for cam operation. 0 : Synchronized in the target value of the main axis. 1 : Synchronized in the current value of the serve axis.
UINT	CamTableID	Specify the cam table to operate.
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
Output		
BOOL	InSync	Indicate that cam operation is normally being fulfilled. (Indicate that the serve axis is following the cam table.)
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to operate the serve axis cam depending on the main axis.
(2) Cam operation command can be given to the serve axis even if the main axis is in stop state.
(3) You must give cam operation abort (MC_CamOut) command to the serve axis or operate other motion function block to stop cam operation.
(4) The axis is in 'Synchronized Motion' while this motion function block is running.
(5) Set the offset of cam table to be applied in MasterOffset and SlaveOffset. MasterOffset sets the offset with the starting point of the main axis, and SlaveOffset sets the offset with the starting point of the serve axis. Refer to the Figure below.

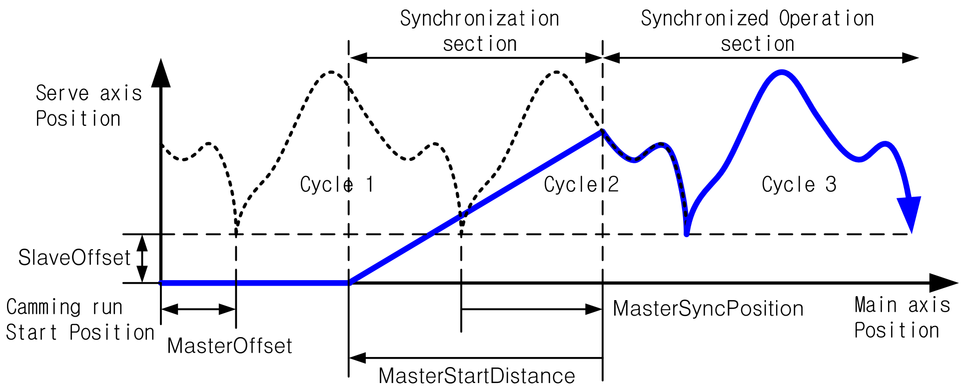




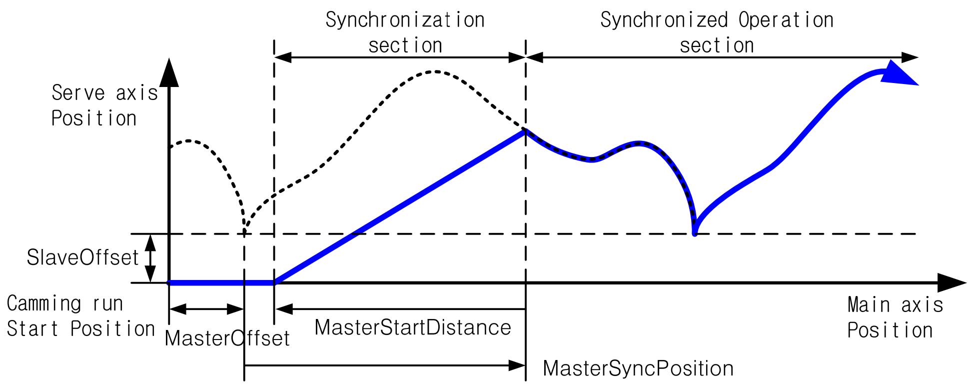
(6) Set the magnification of cam data to be applied in MasterScaling and SlaveScaling. Set the magnification of the main axis data in MasterScaling, and set the magnification of the the serve axis data. Refer to the Figure below.



(7) MasterSyncPosition input specifies the position of the main axis within the table where the synchronization of actual cam operation is completed, and MasterStartDistane input specifies the relative position of the main axis where the synchronization starts.



In case MasterScaling is 1.0



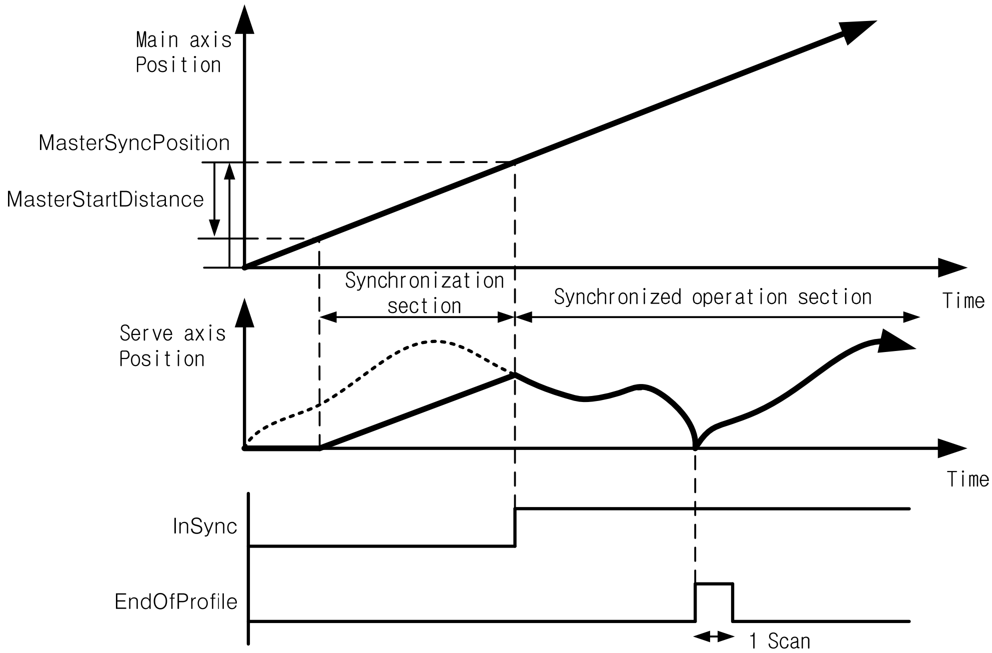
In case MasterScaling is 2.0

MasterSyncPosition position is based on the position within the cam table, and actual synchronization position is decided by considering MasterOffset and MasterScale parameters.

The serve axis starts moving to the synchronization position from the distance of the input value away based on the position where MasterSyncPosition is actually applied. If it is before starting moving, the serve axiss waits at the relevant position in stop state, and if the serve axis is already in the section to move to the synchronization position at the beginning of the command, take back the position of the synchronization starting point by the length of a table until it escapes the MasterStartDistance range.

Actual synchronization position can vary depending on MasterScaling and SlaveScaling because MasterSyncPosition is a value based on the inside of cam table, but MasterOffset and MasterStartDistance value remain unaffected.

(8) Once cam operation starts normally, InSync output is On, and EndOfProfile output is 1 scan On every time one cam table operation is completed.



(9) Cam operation mode is set in StartMode. Setting range is 0 or 1, and the input value outside the setting range causes an error.

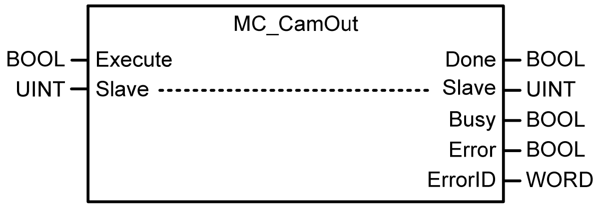
(10) MasterValueSource selects the source of the main axis to be synchronized. If it is set to 0, the serve axis performs cam operation based on the command position of the main axis which is calculated in motion control module, and if it is set to 1, the serve axis performs cam operation based on the current position which is received by communication in servo drive of main axis.

(11) CamTableID sets the number of cam table to be applied to cam operation. Setting range is 1~32 , and the input value outside the setting range causes error "0x1115" in motion function block.

(12) The relevant axis is in "SynchronizedMotion" state while this motion function block is running.

MC_CamOut	Availability
Camming stop	**XMC**

Motion Function Block



Input-Output		
UINT	Slave	Set the the serve axis. (1~32: Actual axes, 33~36: Virtual axes)
Input		
BOOL	Execute	Give cam operation stop command to the relevant axis in the rising Edge.
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block immediately disengages cam operation running in the serve axis.

(2) If motion function block of which BufferMode is Aborting in the serve axis where cam operation is running, cam operation is automatically disengaged and the relevant motion function block is executed. To execute cam operation abort (MC_CamOut) motion function block, the relevant axis do operation which keeps the speed at the time when cam operation is disengaged. If you want to completely stop the serve axis, use stop (MC_Halt) or immediate stop (MC_Stop) motion function block.

MC_GearIn	Availability
Electrical gearing run	**XMC**

Motion Function Block

```
                         MC_GearIn
  BOOL ─ Execute                               InGear ─ BOOL
  UINT ─ Master ------------------------------ Master ─ UINT
  UINT ─ Slave  ------------------------------  Slave ─ UINT
  BOOL ─ ContinousUpdate                         Busy ─ BOOL
   INT ─ RatioNumerator                        Active ─ BOOL
  UINT ─ RatioDenominator            CommandAborted ─ BOOL
  UINT ─ MasterValueSource                      Error ─ BOOL
 LREAL ─ Acceleration                         ErrorID ─ WORD
 LREAL ─ Deceleration
 LREAL ─ Jerk
  UINT ─ BufferMode
```

Input-Output

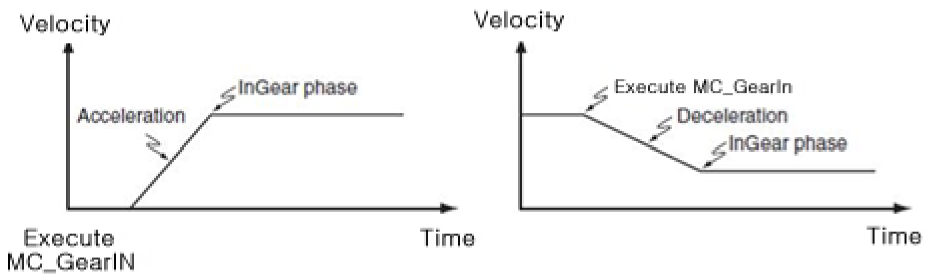
UINT	Master	Set the main axis. (1~32: Actual axes, 33~36: Virtual axes, 1001~1002: Encoders)
UINT	Slave	Set the the serve axis. (1~32: Actual axes, 33~36: Virtual axes)

Input

BOOL	Execute	Give gear operation command to the relevant axis in the rising Edge.
BOOL	ContinuousUpdate	Specify the update setting of input value. (Refer to 16.1.5.Changes in Parameters during Execution of Motion Function Block)
INT	RatioNumerator	Specify the numerator of gear ratio. (-32768 ~ 32767)
UINT	RatioDenominator	Specify the denominator of gear ratio. (0 ~ 65535)
UINT	MasterValueSource	Select data of the main axis to be synchronized. 0: Synchronize in the command position of the main axis. 1: Synchronize in the current position of the main axis.
LREAL	Acceleration	Specify the acceleration at the beginning of gear operation synchronization. [u/s^2]
LREAL	Deceleration	Specify the deceleration at the beginning of gear operation synchronization. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)

Output		
BOOL	InGear	Indicate that gear operation is running by applying gear ration.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is an operation to synchronize the speed of the main axis and the serve axis depending on gear ratio which is set.
(2) Giving gear operation abort (MC_GearOut) commands to the relevant axis or execution of other motion function block allow to disengage gear operation.
(3) RatioNumerator and RatioDenominator set the numerator and denominator to be applied to the serve axis respectively. If the numerator is set to negative number, the rotation direction of the serve axis is the opposite of the main axis.
(4) MasterValueSource select the data of the main axis which is a standard of synchronization. If it is set to 0, synchronization operation is based on the command position of the main axis of motion control module, and if it is set to 1, synchronization operation is based on the current position. Other values set besides these two make Error of motion function block On and cause “0x1114” in ErrorID.
(5) When this motion function block is executed, the serve axis is synchronized with the main axis through acceleration/deceleration at the speed in synch with the relevant gear ratio.
(6) The serve axis is in 'SynchronizedMotion' while this motion function block is running.



# MC_GearOut		Availability
Electrical gearing disengage		**XMC**
Motion Function Block		

```
                 MC_GearOut
BOOL ─ Execute                          Done ─ BOOL
UINT ─ Slave -------------------------- Slave ─ UINT
                                        Busy ─ BOOL
                                       Error ─ BOOL
                                     ErrorID ─ WORD
```

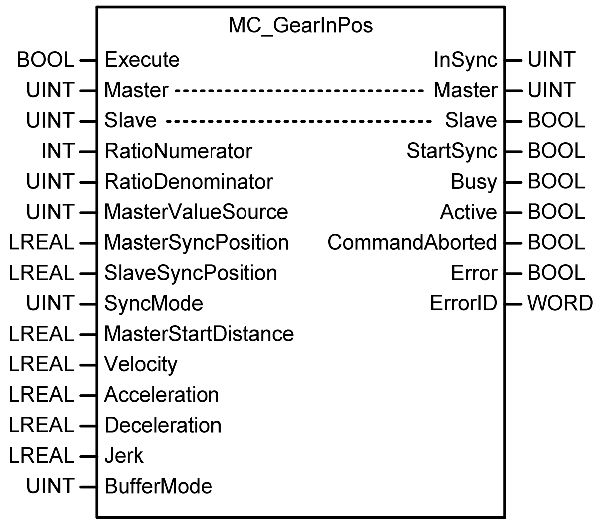
Input-Output		
UINT	Slave	Set the the serve axis. (1~32: Actual axes, 33~36: Virtual axes)
Input		
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block immediately disengages gear operation running in the spindle.

(2) If motion function block of which BufferMode is Aborting in the spindle where cam operation is running, gear operation is automatically disengaged and the relevant motion function block is executed. If gear operation abort (MC_GearOut) motion function block is only to be executed, the relevant axis performs operation to maintain the speed at the time when gear operation is disengaged. To completely stop the spindle, use stop (MC_Halt) or immediate stop (MC_Stop) motion function block.

MC_GearInPos	Availability
Electrical gearing by specifying the position	**XMC**

Motion Function Block



Input-Output

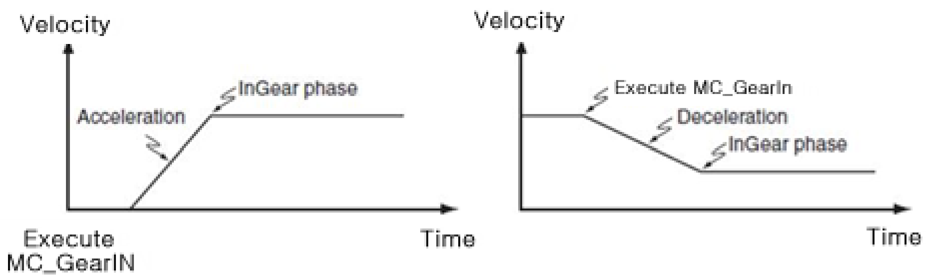
UINT	Master	Set the main axis. (1~32: Actual axes, 33~36: Virtual axes, 1001~1002: Encoders)
UINT	Slave	Set the the serve axis. (1~32: Actual axes, 33~36: Virtual axes)

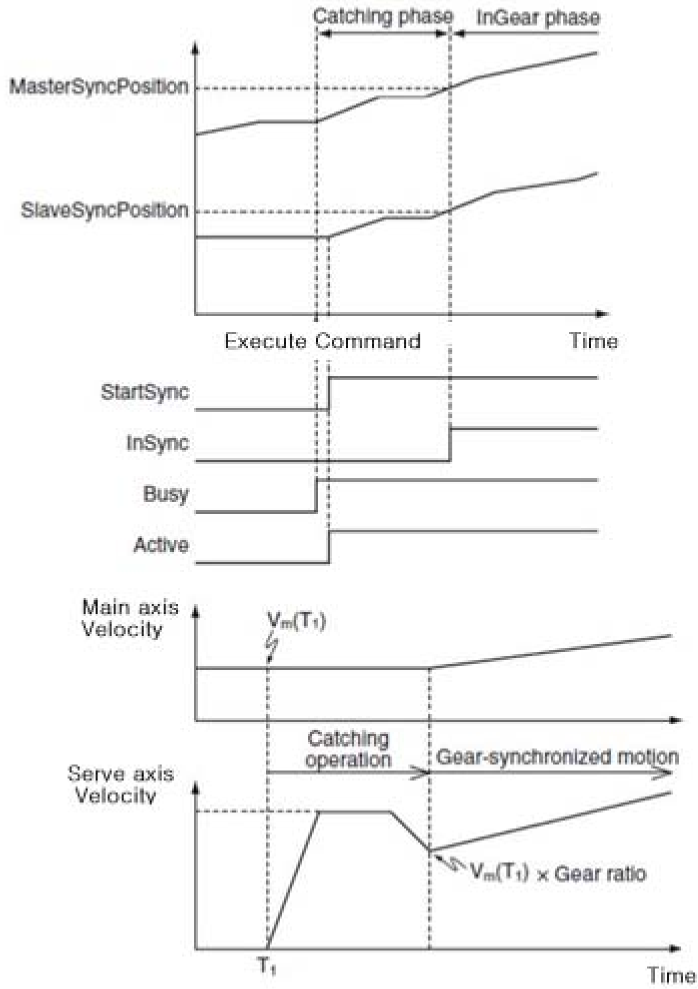
Input

BOOL	Execute	Give a gear operation command to the relevant axis in the rising Edge.
INT	RatioNumerator	Specify the numerator of gear ratio. (-32768~32767)
UINT	RatioDenominator	Specify the denominator of gear ratio. (0~65535)
UINT	MasterValueSource	Select the standard of the main axis value to be synchronized. 0(mcSetValue): Synchronize in the target position of the main axis. 1(mcActualValue): Synchronize in the current position of the main axis.
LREAL	MasterSyncPosition	Specify the position of the main axis where gear operation starts.
LREAL	SlaveSyncPosition	Specify the position of the spindle where gear operation starts.
LREAL	MasterStartDistance	Specify the distance of the main axis where synchronization starts.
LREAL	Velocity	Specify the maximum speed of the spindle at the beginning of synchronization. [u/s]

LREAL	Acceleration	Specify the maximum acceleration of the spindle at the beginning of synchronization. [u/s^2]
LREAL	Deceleration	Specify the maximum deceleration of the spindle at the beginning of synchronization. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
Output		
BOOL	InSync	Indicate that gear operation is normally being fulfilled as the specified gear ratio is applied.
BOOL	StartSync	Indicate synchronization is starting.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

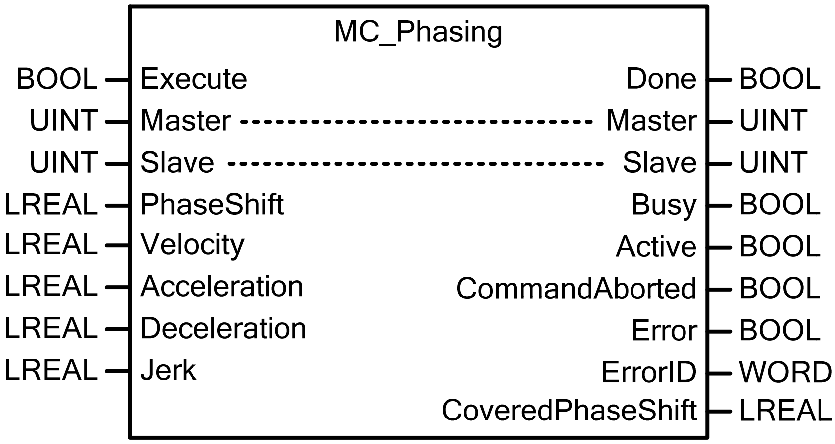
(1) This motion function block is an operation to synchronize the speed of the main axis and the spindle in the set position depending on gear ratio which is set in the specific position.
(2) Giving gear operation abort (MC_GearOut) commands to the spindle or operation of other motion function block allow to stop gear operation.
(3) RatioNumerator and RatioDenominator set the numerator and denominator of gear ratio to be applied to the spindle respectively. If the numerator is set to negative number, the rotation direction of the spindle goes into reverse of the main axis.
(4) MasterValueSource selects the source of the main axis to be synchronized. If it is set to 0 (mcSetValue), synchronization is performed by putting the target position of the main axis in the current motion control period as a source, and if it is set to 1(mcActualValue), synchronization is performed by putting the current position of the main axis got feedback from the current motion control period as a source. Other values set besides these two cause "error 0x10D1".
(5) Input the positions of the main axis and the spindle where gear operation is completed synchronization in MasterSyncPosition input and SlaveSyncPosition input respectively. Input the distance where the spindle starts synchronization in MasterStartDistance input, and the spindle starts synchronization at the position away the distance set in MasterStartDistance input from the position set in MasterSyncPosition input.
(6) Once synchronization starts, StartSync output is On. When synchronization is completed and gear operation starts, StartSync output is Off and InSync output is On.
(7) The spindle is in 'SynchronizedMotion' while this motion function block is running.




Catching phase
InGear phase
MasterSyncPosition
SlaveSyncPosition
Execute Command
Time
StartSync
InSync
Busy
Active
Main axis Velocity
Vm(T1)
Catching operation
Gear-synchronized motion
Serve axis Velocity
Vm(T1) × Gear ratio
T1
Time

MC_Phasing	Availability
Phase Compensation	**XMC**

Motion Function Block



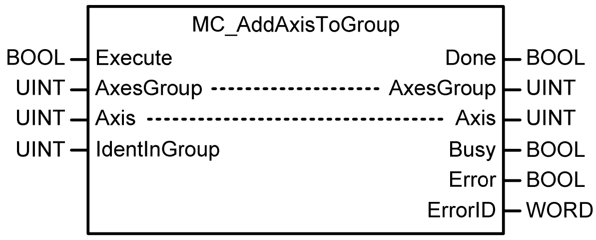
Input-Output		
UINT	Master	Set the main axis. (1~32: real/virtual axis, 33~36: virtual axis, 1001~1002: encoder)
UINT	Slave	Set theserve axis. (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give a phase compensation command to the relevant axis in the rising Edge
LREAL	PhaseShift	Specify the main axis compensation amount.
LREAL	Velocity	Specify the phase compensation velocity. [u/s]
LREAL	Acceleration	Specify the acceleration. [u/s^2]
LREAL	Deceleration	Specify the deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
Output		
BOOL	Done	Indicate whether to reach the specified phase compensation distance.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
LREAL	CoveredPhaseShift	Continuously output the compensation amount reflected while the phase

		compensation is running

(1) This motion function block performs phase correction of axis during synchronous control operation. Phase correction is performed on the main-axis position referred to by sub-axis in synchronous control operation, to perform synchronous control operation of the sub-axis to the corrected main-axis position.
(2) Once phase correction command is executed, the current position of the main-axis is phase-corrected using the phase shift setting at PhaseShift- Velocity / Acceleration /Deceleration / Jerk.
(3) Phase correction does not change the actual command position or current position of the main-axis. Phase correction is performed on the main-axis position referred to by sub-axis in synchronous control operation. In other words, the main-axis does not know that phase correction is executed by the sub-axis.
(4) Phase correction of the same amount can be performed again from the current position by re-executing the function block (Execute input is on) before the command is completed. In other words, phase shift is a relative value from the execution point.
(5) After executing phase correction command, when the phase shift is reached, Done output is on.

MC_AddAxisToGroup	Availability
Adds one axis to a group in a structure AxesGroup	XMC

Motion Function Block

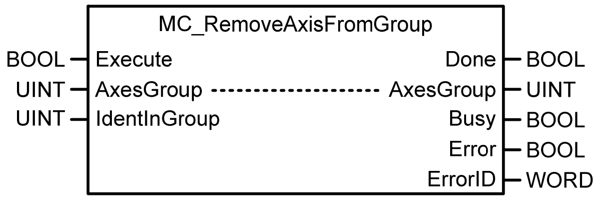


Input-Output		
UINT	AxesGroup	Set the group where the relevant axis is added. (1 ~ 16 : Group 1 ~ Group 16)
UINT	Axis	Set the axis to be added to the relevant group. (1~32: Actual axes, 33~36: Virtual axes)
Input		
BOOL	Execute	Give group axis addition command to the relevant axis in the rising Edge.
UINT	IdentInGroup	Set the ID of the relevant axis to be used in the relevant group. (1 ~ 4)
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block adds Axis specified axis to the axis group specified in AxesGroup input.

(2) ID in the axis group specified to IdentInGroup must have unique value for each axis. (ID of each axis must be different.) Maximum 4 axes can be included in each axis group, axis ID can be specified in the range of 1-4. If the specified axis number is outside the range, “error 0x0006” occurs, and if numbers in the axis group overlap, “error 0x2051” occurs.

MC_RemoveAxisFromGroup		**Availability**
Removes one axis to a group in a structure AxesGroup		**XMC**
Motion Function Block		



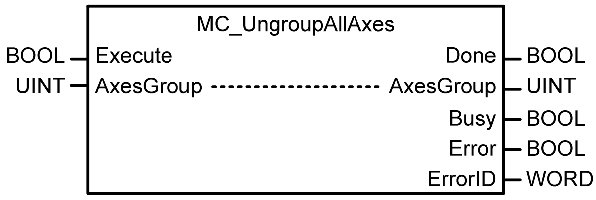
Input-Output		
UINT	AxesGroup	Set the group where the relevant axis is removed. (1 ~ 16 : Group1 ~ Group 16)
Input		
BOOL	Execute	Give group axis exclusion command to the relevant group in the rising Edge.
UINT	IdentInGroup	Set the axis number in the relevant group to be removed from the relevant group.
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block removes the axis which is specified to IdentInGroup in the axis group specified in AxesGroup input.

(2) If the execution of group axis exclusion is tried when the axis group is not in GroupDisabled, GroupStandBy, and GroupErrorStop state, "error 0x2003 or 0x2004 or 0x2005" occurs and the axis is not removed. In other words, the axis cannot be removed when the axis group does not completely stop.

# MC_UngroupAllAxes	Availability
Removes all axes from the group AxesGroup	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group where every axis is to be removed. (1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give MC_UngroupAllAxes command to the relevant group in the rising Edge.
UINT	IdentInGroup	Set the axis number in the relevant group to be removed from the relevant group.
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

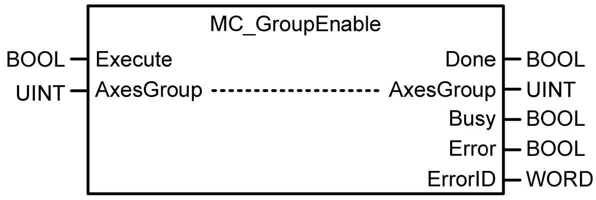
(1) This motion function block removes every axis which belongs to the axis group specified in AxesGroup input.

(2) If this motion function block is executed when the axis group is not in GroupDisabled, GroupStandBy, and GroupErrorStop state, "error 0x2003 or 0x2004 or 0x2005" occurs and the axis is not removed. In other words, the axis cannot be removed when the axis group does not completely stop.

(3) When the axis which belongs to the group is successfully removed, the relevant group is switched to GroupDisabled state.

MC_GroupEnable	**Availability**
Changes the state for a group from GroupDisabled to GroupEnable	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group to be activated. (1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give group activation command to the relevant group in the rising Edge.
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

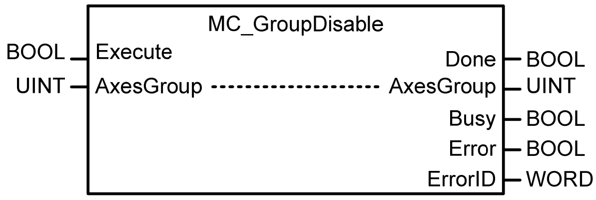
(1) This motion function block is to activate the axis group specified in AxesGroup input.

(2) When giving this command to the axis group in GroupDisable state, the relevant axis group is switched to GroupStandby state.

(3) This motion function block does not affect the power state of each axis in the relevant group.

MC_GroupDisable	Availability
Changes the state for a group to GroupDisabled	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group to be deactivated. (1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give group disablement command to the relevant group in the rising Edge.
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

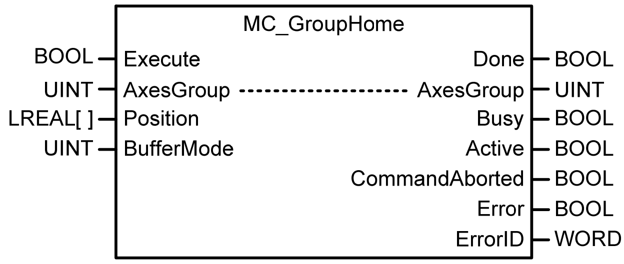
(1) This motion function block is to deactivate the axis group specified in AxesGroup input.

(2) The axis group which executes this motion function block is switched to GroupDisabled.

(3) This motion function block does not affect the power state of each axis in the relevant group.

MC_GroupHome	**Availability**
The AxesGroup to perform the search home sequence	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group returning to home. (1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give group homing command to the relevant group in the rising Edge.
LREAL[ ]	Position	Specify the absolute position of each axis when reference signal is detected.
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give homing command to the axis group specified in AxesGroup input.

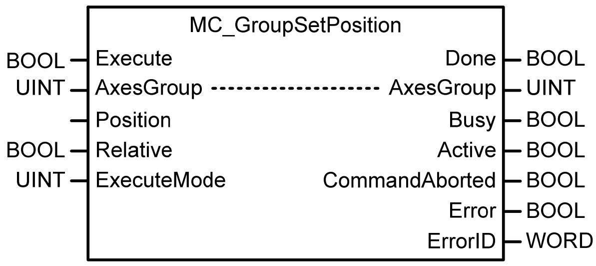
(2) Homing method is operated as specified in servo parameter of the relevant axis in advance.

(3) In Position input, specify the absolute position to the array to be set when homing is completed or Reference Signal is detected. Values in the array and the axis in the group correspond in the order of [1, 2, 3, 4]. (1~4 are the axis ID in the axis group)

(4) The axis group is in 'GroupHoming' state while this motion function block is running, and it is switched to 'GroupStandby' state when motion function block is completed.

MC_GroupSetPosition	Availability
Sets the Position of all axes in a group without moving	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Select the group to set the current position. (1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give group current position setting command to the relevant group in the rising Edge.
LREAL[ ]	Position	Specify the position.
BOOL	Relative	0: Position value=Absolute position, 1: Position value=Relative position
UINT	ExecuteMode	0: Immediately applied the position value, 1: Applied at the same point with ‘Buffered’ of Buffermode
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block sets the current position of the relevant axis group.

(2) Specify the position of each axis in the group to the array. When executing this motion function block, if Relative input is Off, the position of the relevant axis is replaced by the Position input value, and if Relative input is On, the Position input value is added to the current position of the relevant axis. Values in the array and the axis in the group correspond in the order of [1, 2, 3, 4]. (1~4 are the axis ID in the axis group)

(3) ExcutionMode input specifies the setting point. If it is 0, it is set immediately after the execution of a command, If it is 1, it is set at the same point with ‘Buffered’ of sequential operation setting. The value unable to be set causes

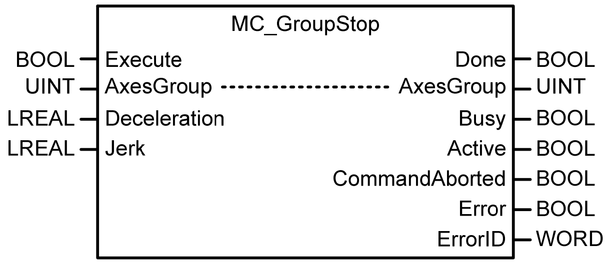
"error 0x201B".

0 (mcImmediately): Change the value of parameter immediately after the execution of motion function block (rising Edge in Execute input). If the relevant axis is running, the operation can be affected.

1 (mcQueued): Changed at the same point of 'Buffered' of Buffermode (Refer to 16.1.4 BufferMode).

MC_GroupStop	Availability
Stop a Group immediately	XMC

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group to stop immediately. (1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give group immediate stop command to the relevant group in the rising Edge.
LREAL	Deceleration	Specify the deceleration in time of stop. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give an emergency stop command to the relevant axis group.

(2) The relevant axis group moves on the route which it was following until it completely stops.

(3) When executing group immediate stop (MC_GroupStop) motion function block, motion function block which the relevant axis group is performing is interrupted, and the axis is changed to 'GroupStopping'. When the relevant axis group is in 'GroupStopping' state, other motion function block cannot be given to the relevant axis until the stop is completed (until Done output is On).

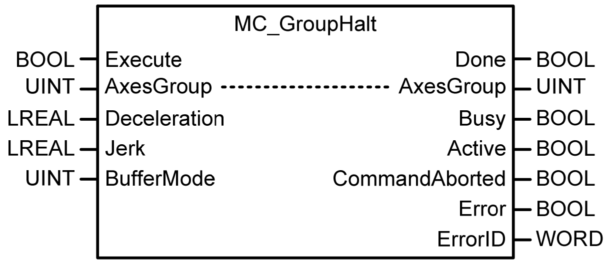
(4) CommandAborted output indicates that the current motion function block is interrupted while it was executed. Because other motion function block cannot interrupt group immediate stop (MC_GroupStop) command while

group immediate stop (MC_GroupStop) command is being executed, CommandAborted output is On when the power of servo is cut, servo Off command is executed, or servo connection is disconnected.

(5) If Execute input is On or the speed of the axis is not 0, the axis is in ' GroupStopping' state, and if Done output is On and Execute input is Off, the axis is switched to ' GroupStandBy' state.

# MC_GroupHalt	**Availability**
Stop a Group	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group to stop. (1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give group stop command to the relevant group in the rising Edge.
LREAL	Deceleration	Specify the deceleration in the time of stop. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

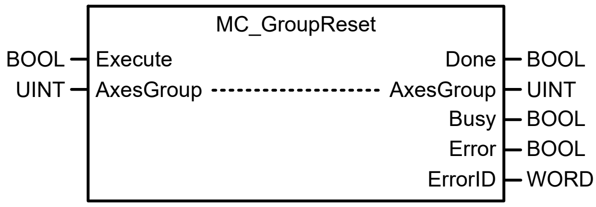
(1) This motion function block is to give a stop command to the relevant axis.

(2) The relevant axis group moves on the route which it was following until it completely stops.

(3) The axis is in 'GroupMoving' state while this motion function block is running, and if the axis group completely stops, 'Done' output is On and the group state is changed to 'GroupStandBy' state.

MC_GroupReset	Availability
Reset a group error	**XMC**

Motion Function Block



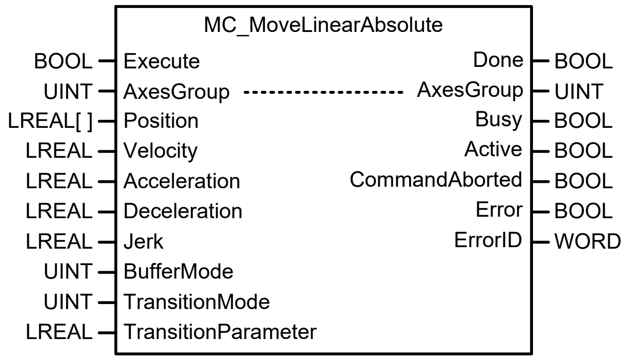
Input-Output		
UINT	AxesGroup	Set the group to do error reset. (1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give group error reset command to the relevant group in the rising Edge.
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to reset the error of the relevant axis group. When the relevant axis is in 'GroupErrorStop', the execution of motion function block resets the error occurred in the current relevant axis and switches the axis group to 'GroupStandBy' state.

(2) When executing this motion function block, every error occurred in each axis in the group is reset. (This has the same effect with when executing the axis error reset (MC_Reset) command in each axis.)

MC_MoveLinearAbsolute	Availability
Absolute positioning linear interpolation operation	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group to perform absolute position linear interpolation operation. (1 ~ 16: Group 1 ~ Group 16)
Input		
BOOL	Execute	Give absolute position linear interpolation operation command to the relevant group in the rising Edge.
LREAL[ ]	Position	Specify the target position of each axis.
LREAL	Velocity	Specify the maximum speed of the route. [u/s]
LREAL	Acceleration	Specify the maximum acceleration. [u/s^2]
LREAL	Deceleration	Specify the maximum deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
UINT	TransitionMode	Specify the route change mode of group operation. (Refer to 10.1.6.TransitionMode)
LREAL	TransitionParameter	Specify the parameter of the route change setting of group operation.. (Refer to 10.1.6.TransitionMode)
Output		
BOOL	Done	Indicate whether to reach the specified position.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.

BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give an absolute position linear interpolation command to the axis group specified in AxesGroup input.

(2) When this motion function block is executed, interpolation control is performed in a linear path from the current position to the target position of each axis, and the moving direction is decided by the starting point and the target point of each axis.

Beginning position < Target position: Forward direction operation

Beginning position > Target position: Reverse direction operation

(3) In Position input, specify the target position of each axis in the group as matrix. The values in the array and the axis in the group correspond in the order of [1, 2, 3, 4]. (1~4 are axis ID in the axis group)

(4) Specify the speed, acceleration, deceleration, and the change rate of acceleration/deceleration of interpolation route in Velocity, Acceleration, Deceleration, and Jerk inputs respectively.

(5) Velocity is to set the interpolation speed of the axis group, and it indicates the integrated speed of each axis. Operation speeds of each configuration axis are calculated as follows.

$$\text{Interpolation speed (F)} = \text{Target speed specified in the Velocity}$$

$$\text{Interpolation movement amount (S)} = \sqrt{S_1{}^2 + S_2{}^2 + S_3{}^2 + S_4{}^2}$$

$$\text{Configuration axis 1 speed } (V_1) = \text{Interpolation speed (F)} \times \frac{\text{Configuration axis 1 movement amount } (S_1)}{\text{Interpolation movement amount (S)}}$$

$$\text{Configuration axis 2 speed } (V_2) = \text{Interpolation speed (F)} \times \frac{\text{Configuration axis 2 movement amount } (S_2)}{\text{Interpolation movement amount (S)}}$$

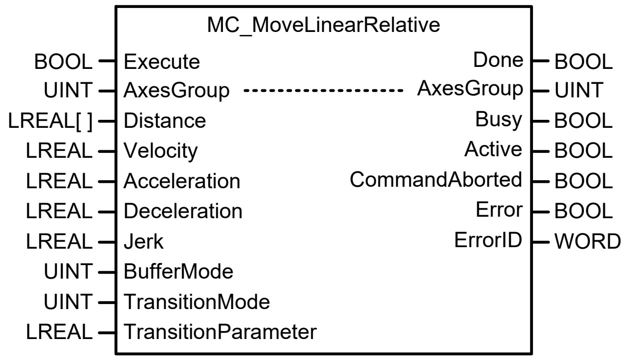
$$\text{Configuration axis 3 speed } (V_3) = \text{Interpolation speed (F)} \times \frac{\text{Configuration axis 3 movement amount } (S_3)}{\text{Interpolation movement amount (S)}}$$

$$\text{Configuration axis 4 speed } (V_4) = \text{Interpolation speed (F)} \times \frac{\text{Configuration axis 4 movement amount } (S_4)}{\text{Interpolation movement amount (S)}}$$

(6) Refer to linear interpolation control part in motion control module’s manual for more details.

MC_MoveLinearRelative	Availability
Relative positioning linear interpolation operation	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group to do relative position linear interpolation operation. (1 ~ 16: Group 1 ~ Group 16)
Input		
BOOL	Execute	Give relative position linear interpolation operation command to the relevant group in the rising Edge.
LREAL[ ]	Distance	Set the target distance of each axis.
LREAL	Velocity	Specify the maximum speed of the route. [u/s]
LREAL	Acceleration	Specify the maximum acceleration. [u/s^2]
LREAL	Deceleration	Specify the maximum deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
UINT	TransitionMode	Specify the route change mode of group operation. (Refer to 10.1.6.TransitionMode)
LREAL	TransitionParameter	Specify the parameter of the route change setting of group operation.. (Refer to 10.1.6.TransitionMode)
Output		
BOOL	Done	Indicate whether to reach the specified position.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.

BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give a relative position linear interpolation command to the axis group specified in AxesGroup input.

(2) When this motion function block is executed, interpolation control performed in a linear path from the current position to the target position of each axis, and the moving direction is decided by the sign of the target distance of each axis.

Target distance > 0: Forward direction operation

Target distance < 0: Reverse direction operation

(3) In Distance input, specify the target distance of each axis in the group as array. The specified array and the axis in the group correspond in the order of specified axis ID [ID1 target distance, ID2 target distance, …].

(4) Set the speed, acceleration, deceleration, and the change rate of acceleration/deceleration of interpolation route in Velocity, Acceleration, Deceleration, and Jerk inputs respectively.

(5) Velocity is to set the interpolation speed of the axis group, and it indicates the integrated speed of each axis. Operation speeds of each configuration axis are calculated as follows.

$$\text{Interpolation speed (F)} = \text{Target speed specified in the Velocity}$$

$$\text{Interpolation movement amount (S)} = \sqrt{S_1^{\,2} + S_2^{\,2} + S_3^{\,2} + S_4^{\,2}}$$

$$\text{Configuration axis 1 speed } (V_1) = \text{Interpolation speed (F)} \times \frac{\text{Configuration axis 1 movement amount } (S_1)}{\text{Interpolation movement amount (S)}}$$

$$\text{Configuration axis 2 speed } (V_2) = \text{Interpolation speed (F)} \times \frac{\text{Configuration axis 2 movement amount } (S_2)}{\text{Interpolation movement amount (S)}}$$

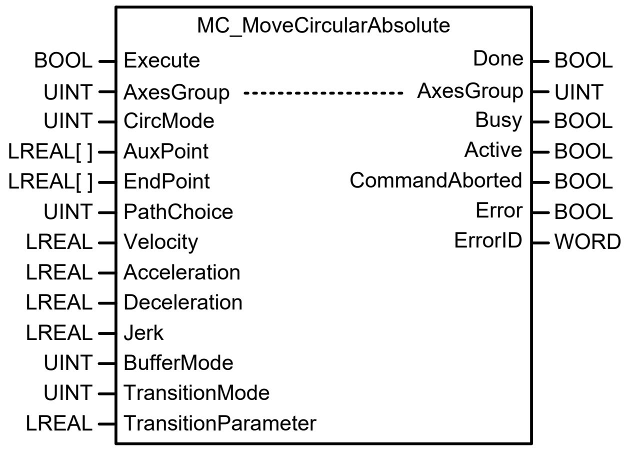
$$\text{Configuration axis 3 speed } (V_3) = \text{Interpolation speed (F)} \times \frac{\text{Configuration axis 3 movement amount } (S_3)}{\text{Interpolation movement amount (S)}}$$

$$\text{Configuration axis 4 speed } (V_4) = \text{Interpolation speed (F)} \times \frac{\text{Configuration axis 4 movement amount } (S_4)}{\text{Interpolation movement amount (S)}}$$

(6) Refer to linear interpolation control part in motion control module’s manual for more details.

MC_MoveCircularAbsolute	Availability
Absolute positioning circular interpolation operation	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group to do absolute position circular interpolation operation. (1 ~ 16: Group 1 ~ Group 16)

Input		
BOOL	Execute	Give absolute position circular interpolation operation command to the relevant group in the rising Edge.
UINT	CirMode	Circular interpolation method setting [0: Midpoint, 1: Central point, 2: Radius]
LREAL[ ]	AuxPoint	Specify the position of auxiliary point depending on the circular interpolation method in an absolute coordinate.
LREAL[ ]	EndPoint	Specify the end point of circular arc in an absolute coordinate.
BOOL	PathChoice	Circular route selection 0: Clockwise, 1: Counterclockwise
LREAL	Velocity	Specify the maximum speed of the route. [u/s]
LREAL	Acceleration	Specify the maximum acceleration. [u/s^2]
LREAL	Deceleration	Specify the maximum deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4.BufferMode)
UINT	TransitionMode	Unused

LREAL	TransitionParameter	Unused
Output		
BOOL	Done	Indicate whether to reach the specified position.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give an absolute position circular interpolation command to the axis group specified in AxesGroup input.

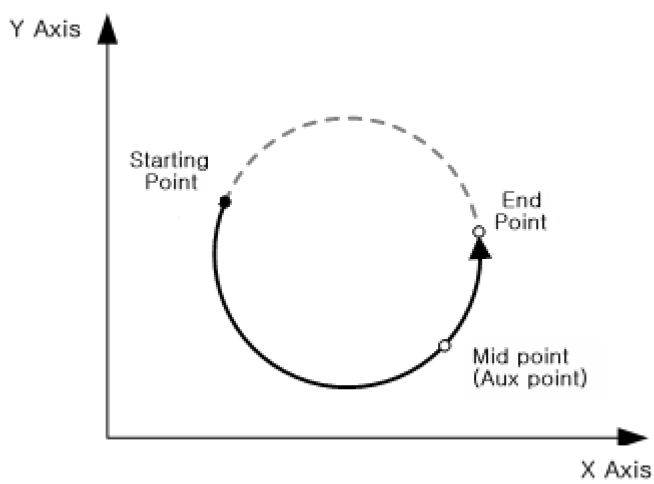
(2) When this motion function block starts, each axis performs circular path interpolation control which refers to the set auxiliary point, and the movement direction is decided by PathChoice input. When setting PathChoice input to 0, circular interpolation operation is done clockwise, and when setting it to 1, circular interpolation operation is done counterclockwise.

(3) Specify the absolute position of the auxiliary point to refer when doing circular interpolation of each axis in AuxPoint and EndPoint inputs as array. The entered array and the axis in the group correspond in the order of the specified axis ID [ID1, ID2, ID3, ···]. (The 3 LEAL type sized array should be entered in Position input as there are 3 axes which comprise the group to give a circular interpolation operation command.)

(4) Specify the speed, acceleration, deceleration, and the change rate of acceleration of interpolation route in Velocity, Acceleration, Deceleration, and Jerk inputs respectively.

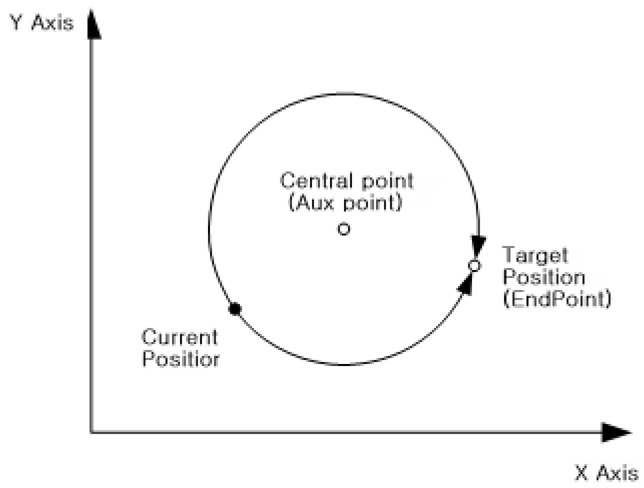
(5) Set the circular interpolation method in CircMode input. The circular interpolation methods which are different from the value specified in CircMode are as below.

- Circular interpolation of midpoint specifying method (BORDER, CircMode = 0)
  In this method, operation starts at the starting point and it does circular interpolation through the specified position of the central point to the target position. The Figure below shows that the coordinate of the axis group at the beginning of a command corresponds to the starting point, the coordinate entered in AuxPoint corresponds to the central point, and the coordinate entered in EndPoint corresponds to the target position in an absolute value.

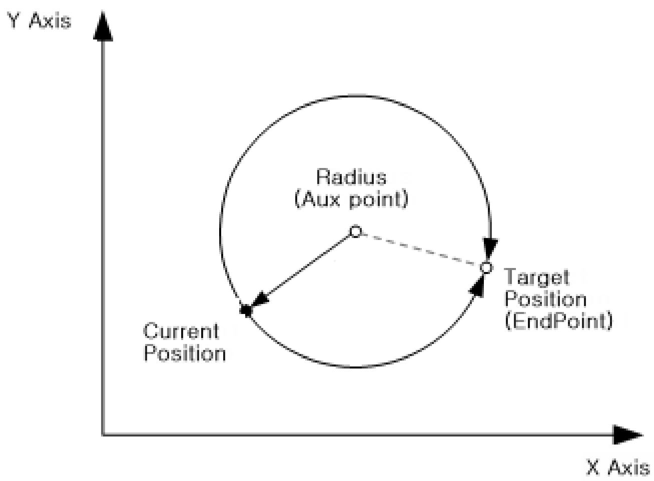


- Circular interpolation of central point specifying method
  In this method, operation starts at the current position, and it does circular interpolation to the target position along the circular path, which has a radius of the distance to the specified central position. The Figure below shows that the coordinate of the axis group at the beginning of a command corresponds to the current

position, the coordinate entered in AuxPoint corresponds to the central point, and the coordinate entered in EndPoint corresponds to the target point as an absolute value.



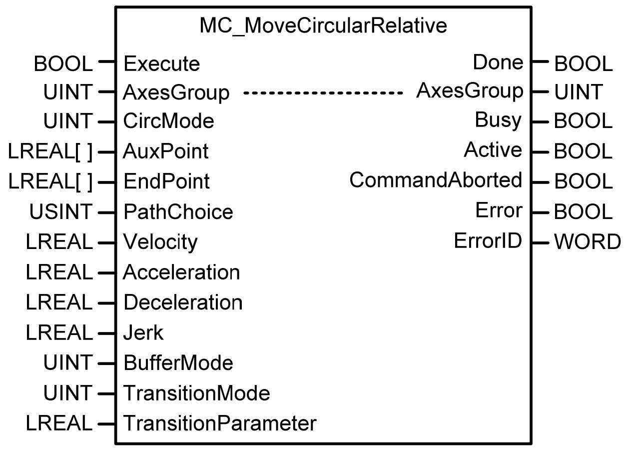
- Circular interpolation using the radius specifying method
  In this method, operation starts at the current position, and it does circular interpolation to the target position along the circular path which has a radius of the value specified in the radius. The Figure below shows that the coordinate of the axis group at the beginning of a command corresponds to the current position, the value entered in X-axis of AuxPoint corresponds to the radius, and the coordinate entered in EndPoint corresponds to the target point in an absolute value.



(6) Refer to linear interpolation control part in motion control module's manual for more details.

MC_MoveCircularRelative	Availability
Relative positioning circular interpolation operation	**XMC**

Motion Function Block



Input-Output

UINT	AxesGroup	Set the group to do absolute position circular interpolation operation. (1 ~ 16: Group 1 ~ Group 16)

Input

BOOL	Execute	Give relative position circular interpolation operation command to the relevant group in the rising Edge.
UINT	CirMode	Circular interpolation method setting [0: Midpoint, 1: Central point, 2: Radius]
LREAL[ ]	AuxPoint	Specify the position of auxiliary point depending on the circular interpolation method as the relative coordinate based on the starting point.
LREAL[ ]	EndPoint	Specify the end point of circular arc as the relative coordinate based on the starting point.
BOOL	PathChoice	Circular route selection 0: Clockwise, 1: Counterclockwise
LREAL	Velocity	Specify the maximum speed of the route. [u/s]
LREAL	Acceleration	Specify the maximum acceleration. [u/s^2]
LREAL	Deceleration	Specify the maximum deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block.

		(Refer to 16.1.4.BufferMode)
UINT	TransitionMode	Unused
LREAL	TransitionParameter	Unused
Output		
BOOL	Done	Indicate whether to reach the specified position.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to give a relative position circular interpolation command to the axis group specified in AxesGroup input.

(2) When this motion function block starts, each axis performs circular path interpolation control which refers to the set auxiliary point, and the movement direction is decided by PathChoice input. When setting PathChoice input to 0, circular interpolation operation is done clockwise, and when setting it to 1, circular interpolation operation is done counterclockwise.

(3) Specify the relative position of the auxiliary point to refer when doing circular interpolation of each axis in AuxPoint and EndPoint inputs as array. The entered array and the axis in the group correspond in the order of the specified axis ID [ID1, ID2, ID3, ···]. (The 3 LEAL type sized array should be entered in Position input as there are 3 axes which comprise the group to give a circular interpolation operation command.)

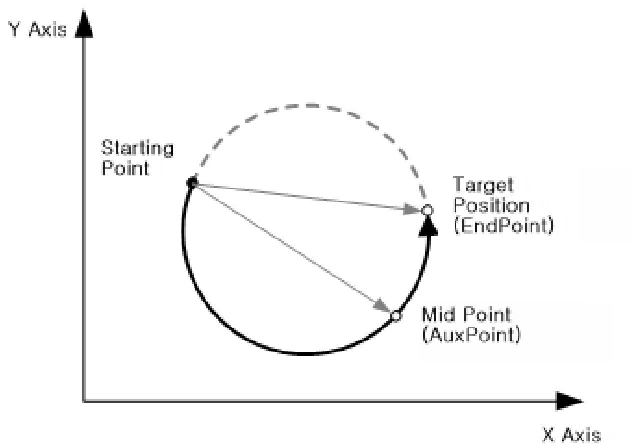
(4) Specify the speed, acceleration, deceleration, and the change rate of acceleration of interpolation route in Velocity, Acceleration, Deceleration, and Jerk inputs respectively.

(5) Set the circular interpolation method in CircMode input. The circular interpolation methods which are different from the value specified in CircMode are as below.

- Circular interpolation of midpoint specifying method (BORDER, CircMode = 0)

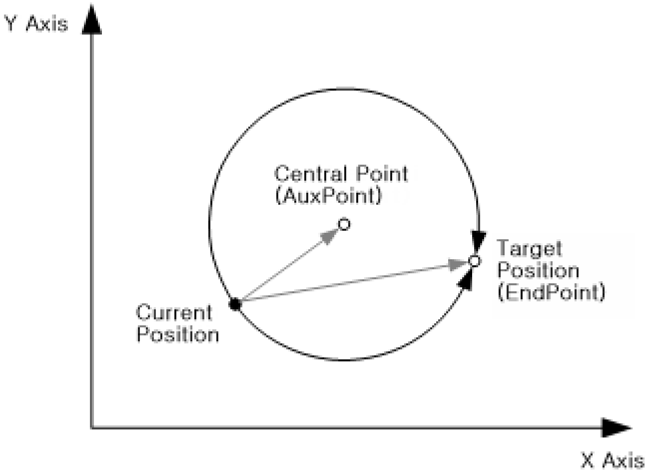
  In this method, operation starts at the current position and it does circular interpolation through the specified position of the central point to the target position.

  The Figure below shows that the coordinate of the axis group at the beginning of a command corresponds to the current position, the coordinate entered in AuxPoint corresponds to the central point, and the coordinate entered in EndPoint corresponds to the target position in a relative value.



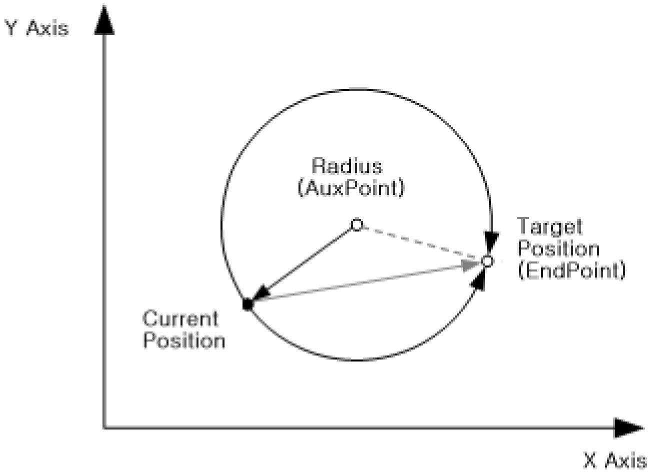
- Circular interpolation of central point specifying method

  In this method, operation starts at the current position, and it does circular interpolation to the target position along the circular path, which has a radius of the distance to the specified central position. The Figure below shows that the coordinate of the axis group at the beginning of a command corresponds to the current position, the coordinate entered in AuxPoint corresponds to the central point, and the coordinate entered in EndPoint corresponds to the target point as a relative value.



- Circular interpolation using the radius specifying method

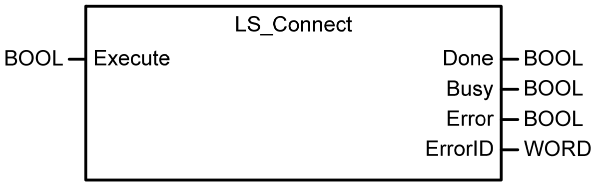
  In this method, operation starts at the current position, and it does circular interpolation to the target position along the circular path which has a radius of the value specified in the radius. The Figure below shows that the coordinate of the axis group at the beginning of a command corresponds to the current position, the value entered in X-axis of AuxPoint corresponds to the radius, and the coordinate entered in EndPoint corresponds to the target point in a relative value.



(6) Refer to linear interpolation control part in motion control module's manual for more details.

# LS_Connect	Availability
Connect servo drives	**XMC**

Motion Function Block



Input		
BOOL	Execute	Give communication connection command to the relevant module in the rising Edge.
Output		
BOOL	Done	Indicate whether to complete communication connection.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

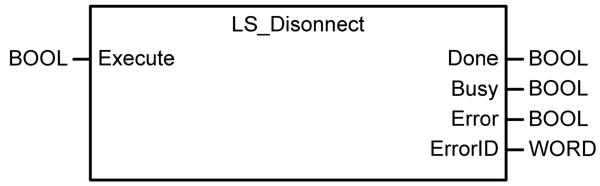
(1) This motion function block is to give a command to connect communication with servo drive or external input/output apparatus to the module.

(2) When slave devicees are normally connected, Done is On and Busy is Off.

(3) If an error occurs during the communication connection, Error is On and error number is output in ErrorID according to the cause.

LS_Disconnect	Availability
Disconnect servo drives	**XMC**

Motion Function Block



Input		
BOOL	Execute	Give communication disconnection command to the relevant module in the rising Edge.

Output		
BOOL	Done	Indicate whether to complete communication disconnection.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

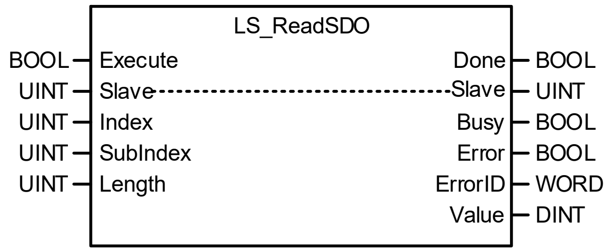
(1) This motion function block gives a command which orders the module to disconnect the communication with servo drive or external input/output apparatuses.

(2) If communication slave is disconnected, Done is On and Busy is off.

(3) If an error occurs during the execution of communication disconnection, Error is On and error number is output in ErrorID according to the error situation.

# LS_ReadSDO	Availability
Read SDO	**XMC**

Motion Function Block



Input-Output		
UINT	Slave	Set the slave to be given a command. (1~64: Slave)
Input		
BOOL	Execute	Give SDO reading command to the relevant slave in the rising Edge.
UINT	Index	Set the Index of slaver Object to be read. (0x0000~0x9FFF)
UINT	SubIndex	Set the SubIndex of slave Object to be read. (0 ~ 255)
UINT	Length	Set the distance of slave Object to be read by Byte. (1 ~ 4)
Output		
BOOL	Done	Indicate that SDO is successfully read.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
LREAL	Value	Output the value of SDO.

(1) This motion function block is to read the SDO (CoE Object) value of servo drive in the relevant axis, and reads the SDO value of the position specified in Index and SubIndex of the axis specified by Axis input as much as the size of Length and indicates it on Value output.

(2) Value output is eliminated to 0 when motion function block is running, and it is output as the read value when the running is completed (Done output is On).

(3) Index input can be set as below. If the value is set outside the range, "error 0x1F12" occurs.

Variable	Description

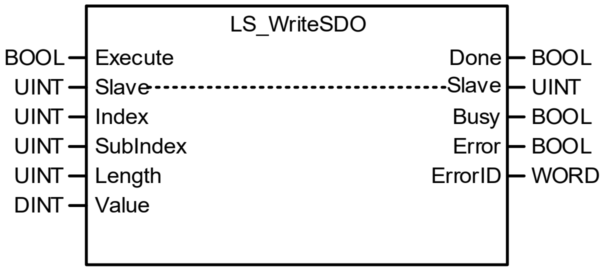
16#0000 ~ 16#0FFF	Data Type Description
16#1000 ~ 16#1FFF	Communication objects
16#2000 ~ 16#5FFF	Manufacturer Specific Profile Area
16#6000 ~ 16#9FFF	Standardized Device Profile Area

(4) The value between 0~255 can be entered in SubIndex, and if the value is set outside the range, "error 0x1F12" occurs.

(5) The value between 1~4 can be set in Length, which means 1~4 Byte. If the value is set outside the range, "error 0x1F12" occurs.

LS_WriteSDO	Availability
Write SDO	**XMC**

Motion Function Block



Input-Output		
UINT	Slave	Set the Slave to be given a command. (1~64: Slave)
Input		
BOOL	Execute	Give SDO writing command to the relevant slave in the rising Edge.
UINT	Index	Set the Index of slave Object to be written. (0x0000~0x9FFF)
UINT	SubIndex	Set the SubIndex of slave Object to be written. (0 ~ 255)
UINT	Length	Set the distance of slave Object to be written by Byte. (1 ~ 4)
DINT	Value	Set the value to be written in SDO.
Output		
BOOL	Done	Indicate that SDO is successfully read.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to write the SDO (CoE Object) value of the relevant slave, and it writes the value entered in Value as the size of the Length in SDO of the position specified as Index and SubIndex of the slave specified in slave input.

(2) Index input can be set as below. When it is set to the value besides the set value, “error 0x1F12” occurs.

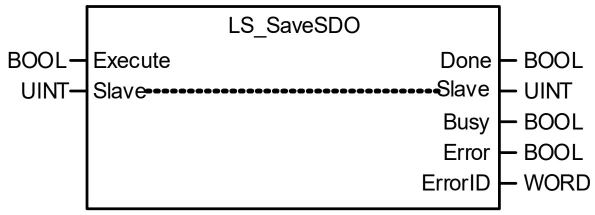
Value	Description
16#0000 ~ 16#0FFF	Data Type Description
16#1000 ~ 16#1FFF	Communication objects
16#2000 ~ 16#5FFF	Manufacturer Specific Profile Area
16#6000 ~ 16#9FFF	Standardized Device Profile Area

(3) The value between the range of 0~255 can be entered in SubIndex, and if the value outside the range is set, “error 0x1F12” occurs.

(4) The value between the range of 1~4 can be entered in Length, which means 1~4 Byte. If the value outside the range is set, "error 0x1F12” occurs.

LS_SaveSDO	Availability
Save SDO	**XMC**

Motion Function Block

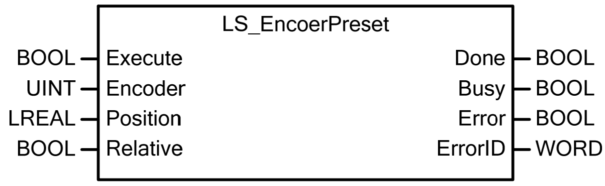


Input-Output		
UINT	Slave	Set the slave to be given a command. (1~64: slave 1~slave 64)
Input		
BOOL	Execute	Give SDO saving command to the relevant slave in the rising Edge.
Output		
BOOL	Done	Indicate that SDO is successfully save.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is a command to save SDO of the designated slave to the memory of the slave.

LS_EncoderPreset	Availability
Encoder preset	**XMC**

Motion Function Block

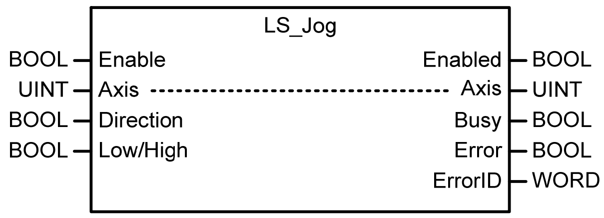


Input		
BOOL	Execute	Specify the position of the relevant encoder in the rising Edge.
UINT	Encoder	Set the encoder to set the position. (1~2: Encoder 1~Encoder 2)
LREAL	Position	Specify the position to set. [u]
BOOL	Relative	0: Absolute coordinate position 1: Relative coordinate position

Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

LS_Jog	Availability
JOG operation	**XMC**

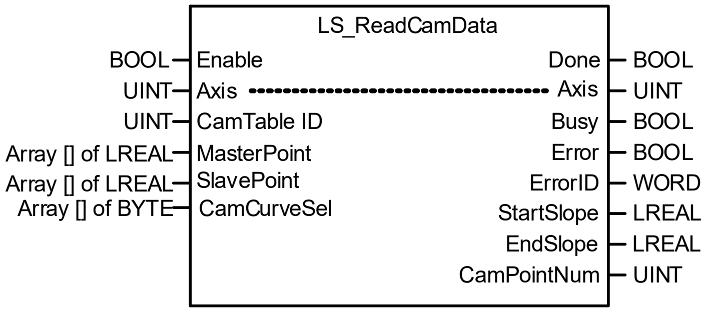
Motion Function Block



Input-Output		
UINT	Axis	Set the axis to be given a command. (1~32: Actual axes)
Input		
BOOL	Enable	Give jog command to the relevant axis while input is On.
BOOL	Direction	Set the rotation direction in jog (0: Forward direction, 1: Reverse direction)
BOOL	Low/High	Set the jog speed in jog. (0: Jog low speed operation, 1: Jog high speed operation)
Output		
BOOL	Enabled	Indicate that the relevant axis is in jog.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to make the relevant axis perform jog operation.
(2) Jog is a manual operation function for test and is used to confirm the position address for system operation, wiring condition check, and teaching. Jog can be used by dividing the speed into high speed and low speed.
(3) When Enable input is On (in jog), if the value set in Low/High is changed, speed change occurs without stop in jog, and if the value set in JOG_DIR is changed, Jog is continued by changing the direction after the deceleration pause.

LS_ReadCamData		**Availability**
Read Cam Data		**XMC**
Motion Function Block		

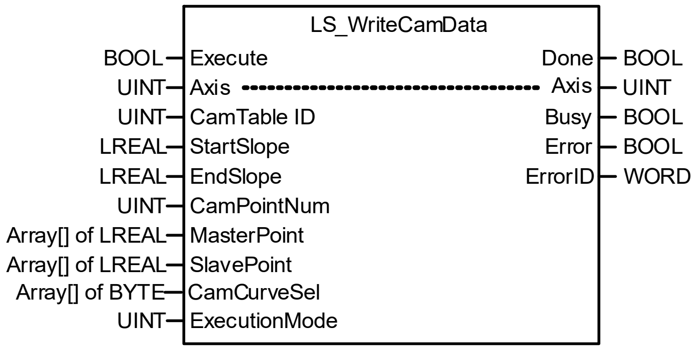


입력-출력		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
입력		
BOOL	Enable	Read the relevant cam data while input is On.
UINT	CamTableID	Specify the cam table to read. (1~32)
LREAL	MasterPoint	MasterPoint values of the cam table are displayed on the areas of which front address is the set device.
LREAL	SlavePoint	SlavePoint values of the cam table are displayed on the areas of which front address is the set device.
BYTE[]	CamCurveSel	Cam Curve form of the cam table are displayed on the areas of which front address is the set device. (0: Linear, 1: Cubic)
출력		
BOOL	Vaild	Indicate the validity of motion function block output.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
LREAL	StartSlope	Output the StartSlope value of the relevant cam table.
LREAL	EndSlope	Output the EndtSlope value of the relevant cam table.
UINT	CamPointNum	Output the cam data point number of the relevant cam table.

(1) This function block displays the data of the cam table.
(2) While Enable input is activated, the data values of the cam table are displayed in succession.
(3) The first address of the variables to store "Main-axis Position" and "Sub-axis Position" read from the camp profile is set at the MasterPoint and the SlavePoint.

LS_WriteCamData	Availability
Write Cam Data	**XMC**

Motion Function Block



Input-Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give the cam data writing command in the rising Edge of the input.
UINT	CamTableID	Specify the ID of the cam table to write. (1~32)
LREAL	StartSlope	Specify the StartSlope value of the cam table to write.
LREAL	EndSlope	Specify the StartSlope value of the cam table to write.
UINT	CamPointNum	Specify the cam data point number of the cam table to write.
LREAL	MasterPoint	Of the cam data to write, set the leading address of the device where Master Point value is stored.
LREAL	SlavePoint	Of the cam data to write, set the leading address of the device where Slave Point value is stored.
BYTE[]	CamCurveSel	Specify the cam curve type (0: Linear, 1: Cubic)
UINT	ExecutionMode	Set the timing to write the cam data. 0 - Immediately applied, 1: Applied at the same point with 'Buffered' of Buffermode
Output		
BOOL	Done	This represents successful cam data writing.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is a command to write the data value of the cam table. Of the cam table data set by CamTableID input, use the value of the device set at MasterPoint and Slave Point at the value set at StartSlope

and EndSlope and the set number at CamPointNum as the MasterPoint and SlavePoint values.

(2) CamTableID input can be set to between 1 and 32. Setting a value outside the above range will cause "Error 16#000B"

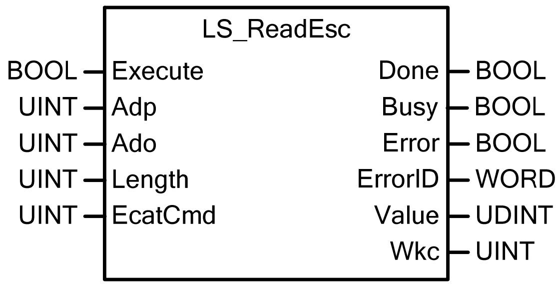
(3) ExecutionMode input sets the setting timing. When the input is 0, setting is performed upon executing the command. When the input is 1, setting is performed at the same time as "Buffered" at the sequential operation. Setting an incorrect value will cause "Error 16#000B".

0(mcImmediately) : Itchanges the (Upward Edge of Execute input) parameter value upon executing the function block. If the axis is in operation, the motion may be affected.

1(mcQueued) : It is changed at the same point of time as in "Buffered" of Buffermode.

LS_ReadEsc	Availability
Read ESC	**XMC**

Motion Function Block



Input

BOOL	Execute	Give the ESC reading command to the slave controller in the rising Edge.
UINT	Adp	Set the slave controller address according to the EcatCmd.
UINT	Ado	Set the slave controller ESC address.
UINT	Length	Set the data length to read. (1 ~ 4 Byte)
UINT	EcatCmd	Set the EtherCAT command. (1: APRD, 4: FPRD, 7: BRD)

Output

BOOL	Done	This represents successful ESC reading to complete normally.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
UDINT	Value	Output the ESC reading value of the slave controller
UINT	Wkc(Working Counter)	After the execution of the command, Working Counter value is displayed.

(1) This motion function block is a function block to read the data of the address in Ado set from the ESC (EtherCAT Slave Controller) of the designated slave device.

(2) Value and Wkc(Working Counter) is displayed as 0 when the motion function block is executed. When the execution is completed (Done output is on), the read data value is displayed at Value, and the Working Counter value is displayed at Wkc.

(3) Adp(Address position) is designating the address of the EtherCAT slave device. The following values can be set depending on the EcatCmd setting. If EcatCmd setting is 7(BRD), Adp input value is ignored. If a value outside the range is set for Adp input, “Error 0x0F60” occurs.

EcatCmd	Adp range
1 (APRD)	0x0000: The first slave connected 0xFFFF: The second slave connected 0xFFFE: The third slave connected : 0xFFC1: 64th slave connected
4 (FPRD)	1 ~ 64: slave 1~slave 64
7 (BRD)	-

(4) Length can be set to between 1 and 4, which means 1-4 bytes. Setting a value outside the above range will cause "Error 0x0F61. "

(5) At EcatCmd, set the type of command to use when reading ESC (EtherCAT Slave Controller). One of the following commands can be used: Setting a value outside the above range at EcatCmd will cause "Error 0x0F62."

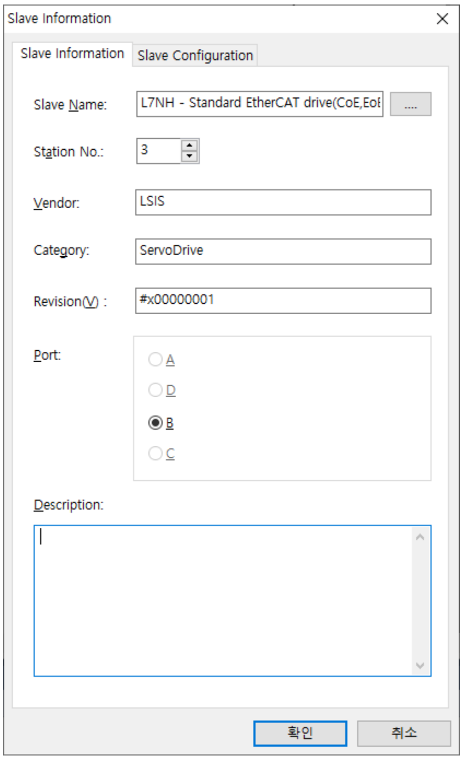
1) 1 - APRD (Auto Increment Physical Read)

This command is used when reading the slave device data following the order of physical connection before normal communication connection by the master. A slave device receiving Adp with 0 value will read data of the size designated by Length. Adp of each slave device increases when EtherCAT frame is received. . For example, if EcatCmd is 1, and Adp is set to 0xFFFF, when executing ESC read function block, read motion is not performed because the Adp at the time of receiving EtherCAT frame from the first slave device is not 1, only increasing Adp by 1. When the second slve device receives EtherCAT frame, read motion is performed because the Adp value of the first slave value increased by 1 to 0. The Adp setting values depending on the slave device connection order are as follows.

Slave controller	Setting value
The first slave connected	0
The second slave connected	0xFFFF
:	:
64th slave connected	0xFFC1

2) 4 - FPRD (Configured Address Physical Read)

This order is used to read the data by designating the station address of the slave device after normal communication connection by the master. If the Station Address of the slave device set by EtherCAT master matches the transmitted Adp, the slave device reads data of the size designated by Length in the Ado area.



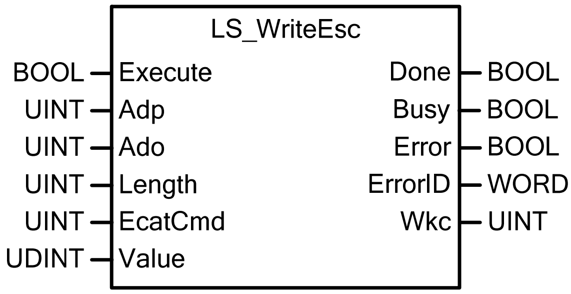
3) 7 – BRD (Broadcast Read)
All connected slave devices read data of the size set by Length in the Ado area, and saves the result after Bitwise-OR (OR operation of each bit). The designated address value at Adp is ignored, and Wkc increase by 1due to all slaves that performed normal read operation

(6) Wkc stands for Working Counter. If data is successfully read at the designated slave device, it increases by 1. If EcatCmd is 7(BRD), it increases by 1 due to all slaves that performed normal read operation.

(7) After the execution of ESC read command, if normal data read operation is executed from the designated slave device, Doneoutput is on.

LS_WriteEsc	**Availability**
Write ESC	**XMC**

Motion Function Block



Input		
BOOL	Execute	Give the ESC writing command to the slave controller in the rising Edge.
UINT	Adp	Set the slave controller address according to the EcatCmd.
UINT	Ado	Set the slave controller ESC address.
UINT	Length	Set the data length to write. (1 ~ 4 Byte)
UINT	EcatCmd	Set the EtherCAT command. (2: APWR, 5: FPWR, 8: BWR)
UDINT	Value	Output the ESC writing value of the slave controller

Output		
BOOL	Done	This represents successful ESC writing to complete normally.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
UINT	Wkc	After the execution of the command, Working Counter value is displayed.

(1) This motion function block writes data using the address set by Ado to ESC (EtherCAT Slave Controller) of the slave device set by Adp.

(2) Wkc value is displayed as 0 when the motion function block is executed, and the Working Counter value is displayed when execution is completed (Done output is on). Wkc increases by 1 through each slave device designated by EcatCmd and Adp.

(3) Adp input designates the EtherCAT slave device address. The following values can be set depending on EcatCmd setting. If EcatCmd setting is 8(BWR), Adp input value is ignored. If a value outside the range is set for Adp input, “Error 0x0F70” occurs.

EcatCmd	Adp range
2 (APWR)	0x0000: The first slave connected 0xFFFF: The second slave connected 0xFFFE: The third slave connected : 0xFFC1: 64th slave connected
5 (FPWR)	1~64: slave 1~slave 64
8 (BWR)	-

(4) Length can be set to between 1 and 4, which means 1-4 bytes. Setting a value outside the above range will cause "Error 0x0F71".

(5) At EcatCmd, set the type of command to use when reading ESC (EtherCAT Slave Controller). The following write commands can be used. Setting a value outside the range at EcatCmd will cause "Error 0x0F72".

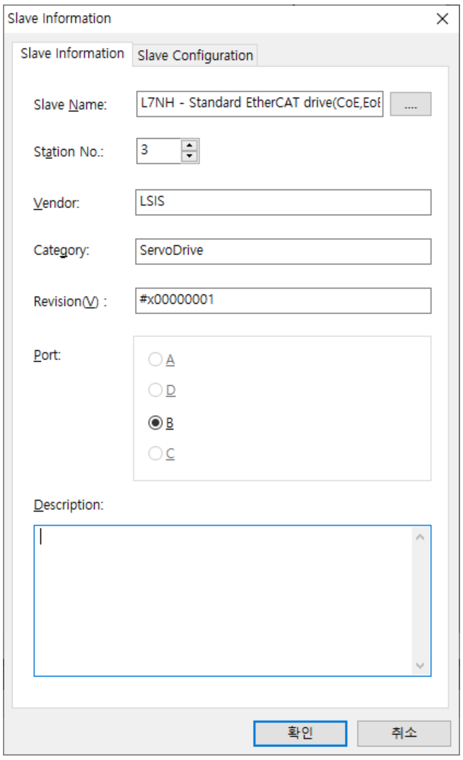
1) 2 - APW (Auto Increment Physical Write)

This command is used when reading the slave device data following the order of physical connection before normal communication connection by the master. A slave device receiving Adp with 0 value will read data of the size designated by Length. Adp of each slave device increases when EtherCAT frame is received. . For example, if EcatCmd is 2, and Adp is set to 0xFFFF, when executing ESC read function block, reading is not performed because the Adp at the time of receiving EtherCAT frame from the first slave device is not 0, only increasing Adp by 1. When the second slave device receives EtherCAT frame, writing is performed because the Adp value of the first slave value increased by 1 to 0. The Adp values depending on the slave device connection order are as follows.

Slave controller	Setting value
The first slave connected	0
The second slave connected	0xFFFF
:	:
64th slave connected	0xFFC1

2) 5 - FPWR (Configured Address Physical Write)

This order is used to write the data by designating the station address of the slave device after normal communication connection by the master. If the Station Address of the slave device set by EtherCAT master matches the transmitted Adp, the slave device writes data of the size designated by Length in the Ado area.



3) 8 –BWR, Broadcast Write

All connected slave devices write data of the size set by Length in the Ado area, and saves the result after Bitwise-OR (OR operation of each bit). The designated address value at Adp is ignored, and Wkc increase by 1 due to all slaves that performed normal write operation.

(6) Wkc stands for Working Counter. If data is successfully written at the designated slave device, it increases by 1. If EcatCmd is 8(BWR), it increases by 1 due to all slaves that performed normal write operation.

After the execution of ESC write command, if normal data write operation is executed in the designated slave device, Doneoutput is on.

LS_CamSkip	Availability
Skip Cam	**XMC**

Motion Function Block

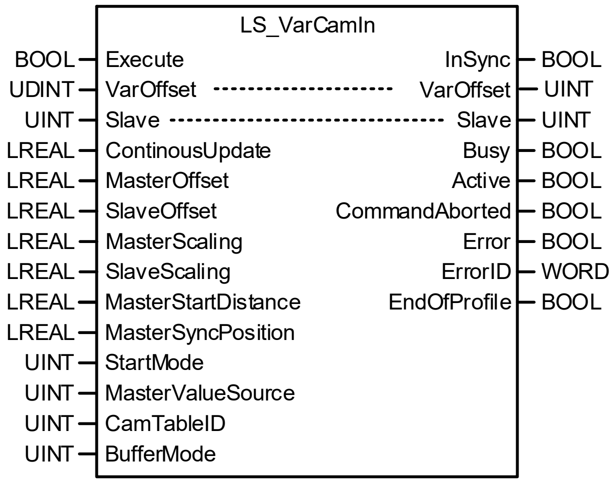
```
                 LS_CamSkip
BOOL ─ Execute                            Done ─ BOOL
UINT ─ Slave  ----------------------     Slave ─ UINT
UINT ─ SkipCount                          Busy ─ BOOL
                                        Active ─ BOOL
                              CommandAborted ─ BOOL
                                         Error ─ BOOL
                                       ErrorID ─ WORD
                             CoveredSkipCount ─ UINT
```

Input - Output		
UINT	Slave	Set the serve axis. (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give cam skip command on the axis in the rising Edge.
UINT	SkipCount	Set the number of cam cycles to skip.
Output		
BOOL	Done	Indicate the completion state of cam skip operation.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current axis is controlling the cam skip.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
LREAL	CoveredSkipcount	Output the number of cam cycle skipped.

(1) This motion function block commands Cap Skip command which skip cam operation cycles as designated for the cam currently in operation.
(2) SkipCount determines the number of cam cycles to skip. If 0 is entered, SkipCount Error (Error 0x111E) is displayed.
(3) When Cam Skip command is issued on a sub-axis during cam operation, the skip motion starts when the current cam cycle is completed. During cam skip, the sub-axis is in stand-by at the end of the cam table.
(4) CoveredSkipCount displays the number of cam cycles skipped. The count increases with each skpped cycle, and becomes 0 when Done output is off after the function block motion is completed
(5) Done output is on when the set number of cycles are skipped after executing Cam Skip command.

LS_VarCamIn	Availability
Variable Cam Operation	**XMC**

Motion Function Block



Input - Output

UDINT	VarOffset	Set the offset value of the M device where the variable to be used as the main axis is located.
UINT	Slave	Set the serve axis. (1~32: real/virtual axis, 33~36: virtual axis)

Input

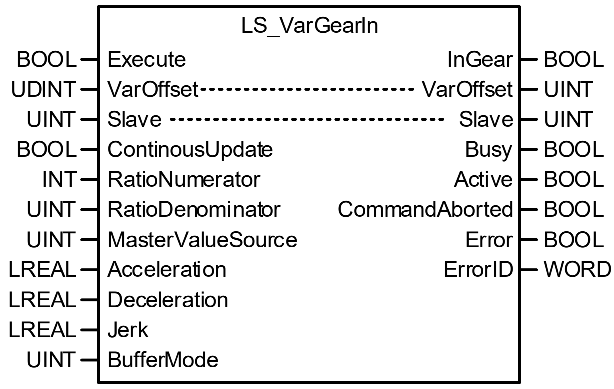
BOOL	Execute	Give cam operation command on the axis in the rising Edge.
BOOL	ContinuousUpdate	Specify the update setting of input value. (Refer to 6.1.5.Changes in Parameters during Execution of Motion Function Block)
LREAL	MasterOffset	Set the offset value of the main axis.
LREAL	SlaveOffset	Set the offset value of the serve axis cam table.
LREAL	MasterScaling	Specify the magnification of the main axis.
LREAL	SlaveScaling	Specify the magnification of the serve axis cam table.
LREAL	MasterStartDistance	Specify the position of the main axis where cam operation of the slave.
LREAL	MasterSyncPosition	Specify the starting point at cam table when cam operation starts.
UINT	StartMode	Set the cam operation mode. 0 : Cam table is applied as an absolute value (mcAbsolute) 1: Cam table is applied as a relative value based on the command starting point (mcRelative)

UINT	MasterValueSource	Select the source of the main axis for cam operation. 0 : Synchronized in the target value of the main axis. 1 : Synchronized in the current value of the serve axis.
UINT	CamTableID	Specify the cam table to operate.
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 6.1.4.BufferMode)
Output		
BOOL	InSync	Indicate that cam operation is normally being fulfilled. (Indicate that the serve axis is following the cam table.)
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is the function block that operates the sub-axis CAM along the main axis by setting the variable value designated by offset as the main axis.
(2) The variable value specified as the main axis should be the LREL type. Example) When specifying the variable to be allocated to the memory by %ML100 as the main axis value, %ML100 should be LREAL type, and the offset value specifying a variable is UDINT type and you should input 100 to the VarOffset.
(3) Remaining settings and functions are the same as the MC_CamIn function block.

LS_VarGearIn	Availability
Variable Gear Operation	**XMC**

Motion Function Block



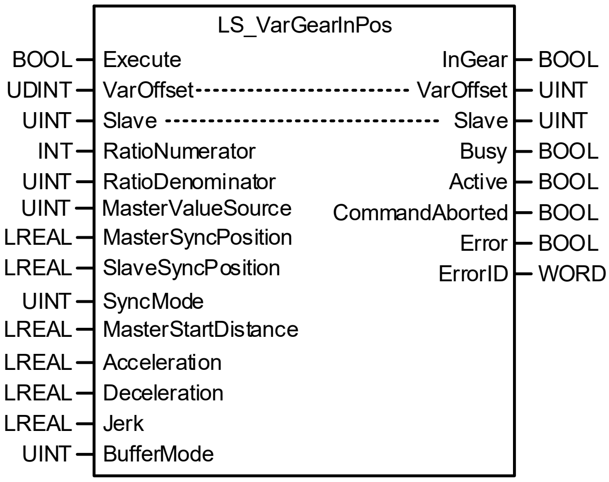
Input - Output		
UDINT	VarOffset	Set the offset value of the M device where the variable to be used as the main axis is located.
UINT	Slave	Set the serve axis. (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give gear operation command to the relevant axis in the rising Edge.
BOOL	ContinuousUpdate	Specify the update setting of input value. (Refer to 6.1.5.Changes in Parameters during Execution of Motion Function Block)
LREAL	RatioNumerator	Specify the numerator of gear ratio. (-32768 ~ 32767)
LREAL	RatioDenominator	Specify the denominator of gear ratio. (0 ~ 65535)
LREAL	MasterValueSource	Select data of the main axis to be synchronized. 0: Synchronize in the command position of the main axis. 1: Synchronize in the current position of the main axis.
LREAL	Acceleration	Specify the acceleration at the beginning of gear operation synchronization. [u/s^2]
LREAL	Deceleration	Specify the deceleration at the beginning of gear operation synchronization. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 6.1.4.BufferMode)

Output		
BOOL	InGear	Indicate that gear operation is running by applying gear ration.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is the function block that drives the main axis and the sub axis in gear operation (speed synchronization) by setting the variable value designated by offset as the main axis.
(2) The variable value specified as the main axis should be the LREL type. Example) When specifying the variable to be allocated to the memory by %ML100 as the main axis value, %ML100 should be LREAL type, and the offset value specifying a variable is UDINT type and you should input 100 to the VarOffset.
(3) Remaining settings and functions are the same as the MC_GearIn function block.

LS_VarGearInPos	Availability
Variable Positioning Gear Operation	**XMC**

Motion Function Block



Input - Output		
UDINT	Master	Set the main axis. (1~32: real/virtual axis, 33~36: virtual axis)
UINT	Slave	Set the serve axis. (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give gear operation command to the relevant axis in the rising Edge.
INT	RatioNumerator	Specify the numerator of gear ratio. (-32768~32767)
UINT	RatioDenominator	Specify the denominator of gear ratio. (0~65535)
UINT	MasterValueSource	Select the standard of the main axis value to be synchronized. 0(mcSetValue): Synchronize in the target position of the main axis. 1(mcActualValue): Synchronize in the current position of the main axis.
LREAL	MasterSyncPosition	Specify the position of the main axis where gear operation starts.
LREAL	SlaveSyncPosition	Specify the position of the spindle where gear operation starts.
UINT	SyncMode	Unused
LREAL	MasterStartDistance	Specify the distance of the main axis where synchronization starts.
LREAL	Velocity	Specify the maximum speed of the spindle at the beginning of synchronization. [u/s]
LREAL	Acceleration	Specify the maximum acceleration of the spindle at the beginning of

		synchronization. [u/s^2]
LREAL	Deceleration	Specify the maximum deceleration of the spindle at the beginning of synchronization. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 6.1.4.BufferMode)
Output		
BOOL	InSync	Indicate that gear operation is normally being fulfilled as the specified gear ratio is applied.
BOOL	StartSync	Indicate synchronization is starting.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is the function block that synchronizes the main axis and the servo axis according to the gear ratio set at the specific position by setting the variable value designated by the offset as the main axis

(2) The variable value specified as the main axis should be the LREL type. Example) When specifying the variable to be allocated to the memory by %ML100 as the main axis value, %ML100 should be LREAL type, and the offset value specifying a variable is UDINT type and you should input 100 to the VarOffset.

(3) Remaining settings and functions are the same as the MC_GearInPos function block.

LS_ReadCamTableSlavePos	Availability
Read the slave position of the CAM table	**XMC**

Motion Function Block

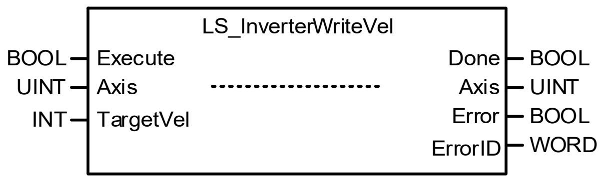
```
                 LS_ReadCamTableSlavePos
BOOL  — Execute                          Done       — BOOL
UINT  — Axis      -----------------      Axis       — UINT
UINT  — CamTableID                       Busy       — BOOL
LREAL — MasterPos                        SlavePos   — LREAL
                                         SlaveVel   — LREAL
                                         SlaveAccel — LREAL
                                         Error      — BOOL
                                         ErrorID    — WORD
```

Input - Output		
UINT	Axis	Specify the axis to be commanded (1~32: real/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give ReadCamTableSlavePos operation command to the relevant axis in the rising Edge.
UINT	CAM tableID	Specify the number of the CAM table to read (1~32)
LREAL	MasterPos	Specify the position of the main axis on the CAM table.
Output		
BOOL	Done	Indicate the completion state of ReadCamTabelSlavePos operation.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
LREAL	SlavePos	It outputs the position of the slave.
LREAL	SlaveVel	It outputs the speed of the slave. [u/s]
LREAL	SlaveAccel	It outputs the acceleration of the slave. [u/s^2]
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block outputs the position of the serve axis according to the position of the main axis in the specified CAM table.
(2) Set the position value of the main axis to be read in the CAM table as the MasterPos value. Offset / gear ratio / phase correction operation, etc. applied to the command axis are not reflected in the SlavePos output.
(3) When reading the slave position on the CAM table is completed, the 'Done Output' will be turned on.

LS_InverterWriteVel	Availability
Write inverter velocity	**XMC**

Motion Function Block



Input - Output		
UINT	Axis	Specify the axis to be commanded (1~32: real axis)
Input		
BOOL	Execute	Give InverterWriteVel operation command to the relevant axis in the rising Edge.
INT	TargetVel	The inverter speed to be set (-30000 ~ 30000, unit: rpm)
Output		
BOOL	Done	Indicate the completion state of InverterWriteVel operation.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

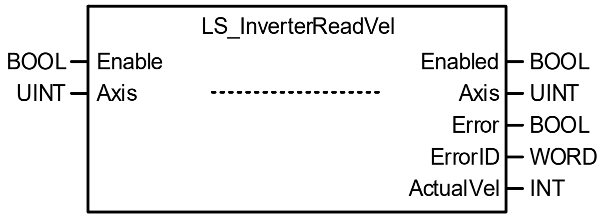
(1) This motion function block is the function block that sets the speed of the inverter to operate when controlling the inverter by the axis

(2) If you set the speed in TargetVel and execute the function block, the inverter connected to the axis will operate at the corresponding speed.

(3) The speed value set in TargetVel is in units of rpm, and can be set to the value from -30000 to 30000.

LS_InverterReadVel	Availability
Read inverter velocity	**XMC**

Motion Function Block

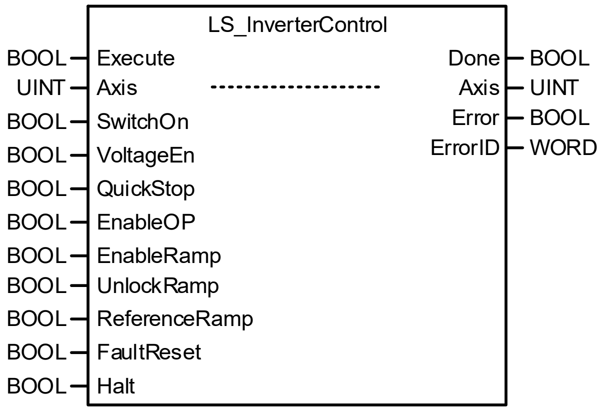


Input - Output		
UINT	Axis	Specify the axis to be commanded (1~32: real axis)
Input		
BOOL	Enable	While the condition is ON, the speed of the inverter connected to the axis is read.
Output		
BOOL	Enabled	It indicates whether reading the inverter speed is being executed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
INT	ActualVel	Speed value of the read inverter

(1) This motion function block is the function block that reads the speed of the connected inverter when controlling the inverter by the axis.
(2) When the function block is executed, the current speed of the inverter connected to the axis is read and displayed in ActualVel.
(3) The speed value set in ActualVel is in units of rpm, and can be displayed as the value from -30000 to 30000.

LS_InverterControl	Availability
Write inverter control word	XMC

Motion Function Block



Input - Output		
UINT	Axis	Specify the axis to be commanded (1~32: real axis)
Input		
BOOL	Execute	Set the inverter control word in the rising Edge.
BOOL	SwitchOn	Switch On
BOOL	VoltageEn	Voltage Enable
BOOL	QuickStop	Quick Stop
BOOL	EnableOP	Enable operation
BOOL	EnableRamp	Enable ramp
BOOL	UnlockRamp	Unlock ramp
BOOL	ReferenceRamp	Reference ramp
BOOL	FaultReset	Fault Reset
BOOL	Halt	Halt
Output		
BOOL	Done	It indicates whether or not the inverter control word setting is done normally.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

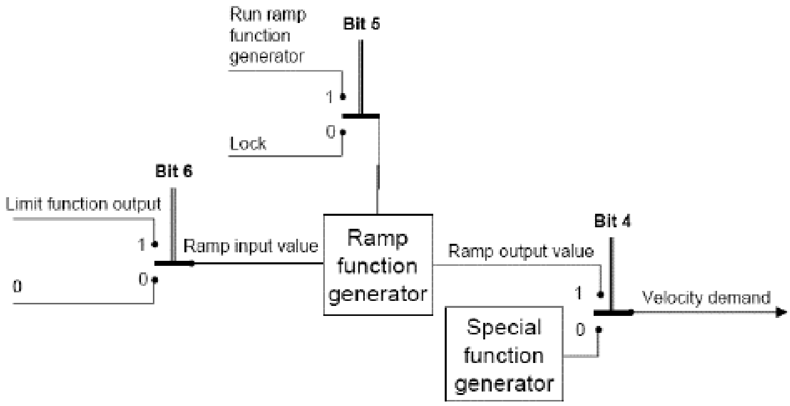
(1) This motion function block is the function block that sets the controlword of the connected inverter when controlling the inverter by the axis.
(2) In order to operate the inverter, the controlword must be set to enable operation.

(3) Please refer to the following.

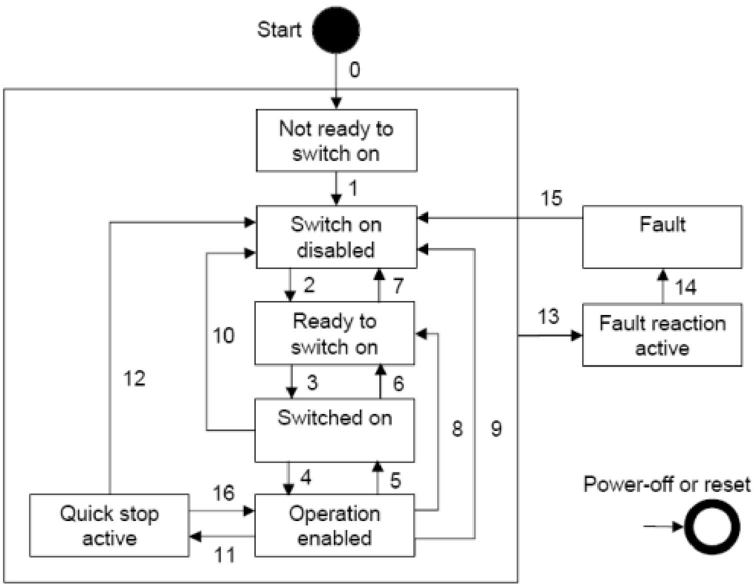
Command bit used in Enable Operation

Bit	Value	설명
4 (Enable Ramp)	0	이전 운전 상태를 유지
	1	명령 비트에 의해 인버터 운전
5 (Unlock Ramp)	0	출력 주파수 Hold
	1	목표 주파수까지 구동
6 (Reference Ramp)	0	목표주파수가 Zero가 입력
	1	목표주파수가 설정한 값으로 입력
8 (Halt)	X	사용 안함

Inverter status according to the bit setting of the control word



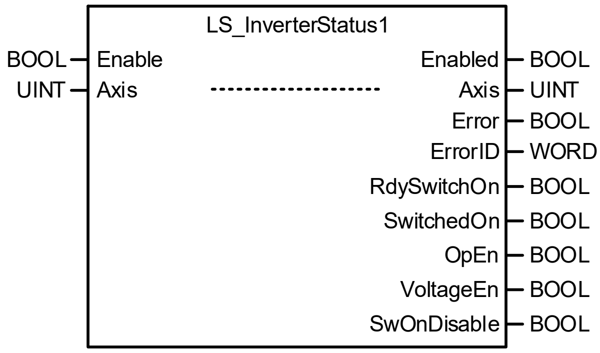
Change the inverter status according to the bit setting of the control word



Command	Bits of the *controlword*					Transitions
	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0	
Shutdown	0	X	1	1	0	2,6,8
Switch on	0	0	1	1	1	3
Switch on + enable operation	0	1	1	1	1	3 + 4 (NOTE)
Disable voltage	0	X	X	0	X	7,9,10,12
Quick stop	0	X	0	1	X	7,10,11
Disable operation	0	0	1	1	1	5
Enable operation	0	1	1	1	1	4,16
Fault reset	↑	X	X	X	X	15
NOTE Automatic transition to Enable operation state after executing SWITCHED ON state functionality.						

LS_InverterStatus1	**Availability**
Read inverter Status1	**XMC**

Motion Function Block



Input - Output		
UINT	Axis	Specify the axis to be commanded (1~32: real axis)
Input		
BOOL	Enable	Read the “Status 1” of the inverter while the input is enabled.
Output		
BOOL	Enabled	It indicates the state of reading the “Status 1” of the inverter.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
BOOL	RdySwitchOn	Ready to Switch On
BOOL	SwitchedOn	Switched On
BOOL	OpEn	Operation Enabled
BOOL	VoltageEn	Voltage Enabled
BOOL	SwOnDisable	Switch On Disable

(1) This motion function block is the function block that reads and displays the "Status 1" of the connected inverter when controlling the inverter by the axis.

(2) RdySwitchOn, SwitchedOn, OpEn, VoltageEn, SwOnDisable are respectively the lower bit values of the Status Word among the inverter PDO Data.

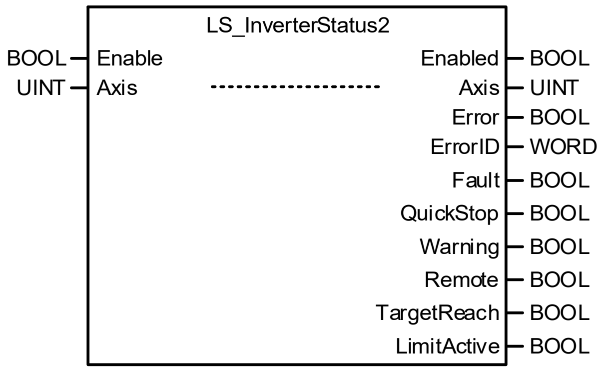
RdySwitchOn	Bit 0
SwitchedOn	Bit 1
OpEn	Bit 2
VoltageEn	Bit 4

SwOnDisable	Bit 6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nu	nu	nu	Nu	lla	tr	rm	nu	w	sod	qs	Ve	f	oe	so	rtso

LS_InverterStatus2	Availability
Read inverter Status2	**XMC**

Motion Function Block



Input - Output		
UINT	Axis	Specify the axis to be commanded (1~32: real axis)
Input		
BOOL	Enable	Read the “Status 2” of the inverter while the input is enabled.
Output		
BOOL	Enabled	It indicates the state of reading the “Status 2” of the inverter.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
BOOL	Fault	Fault(trip)
BOOL	QuickStop	Quick stop
BOOL	Warning	Warning
BOOL	Remote	Remote
BOOL	TargetReach	Target Reached
BOOL	LimitActive	Internal Limit active

(1) This motion function block is the function block that reads and displays the "Status 2" of the connected inverter when controlling the inverter by the axis.

(2) Fault, QuickStop, Warning, Remote, TagetReach, LimiActive are respectively the lower bit values of the Status Word among the inverter PDO Data.

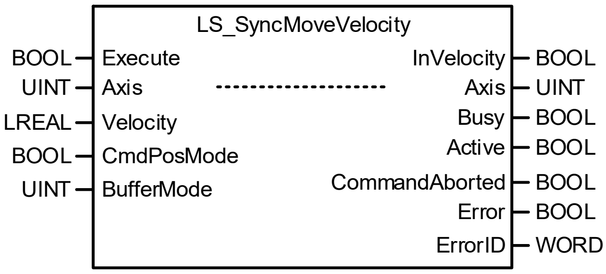
Fault	Bit 3
QuickStop	Bit 5
Warning	Bit 7

Remote	Bit 6
TargetReach	Bit 10
LimitActive	Bit11

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nu	nu	nu	Nu	lla	tr	rm	nu	w	sod	qs	Ve	f	oe	so	rtso

LS_SyncMoveVelocity	Availability
CSV(Cyclic Synchronous Velocity mode) control operation	**XMC**

Motion Function Block

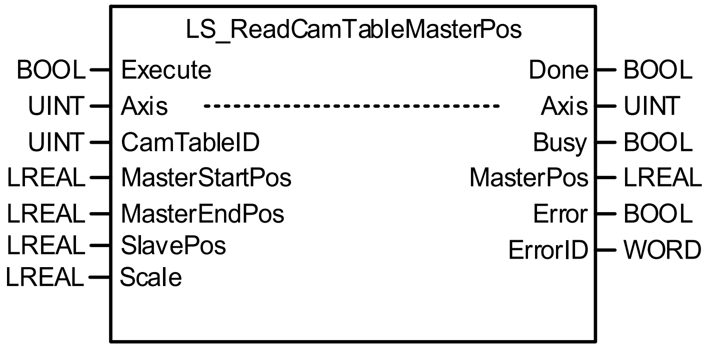


Input - Output		
UINT	Axis	Specify the axis to be commanded (1~32: real axis)
Input		
BOOL	Execute	In the rising Edge, it performs speed control operation through the CSV mode.
BOOL	CmdPosMode	0: Apply the current position to the command position.
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 6.1.4.BufferMode)
Output		
BOOL	Done	Indicate whether to reach the specified distance.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is the function block that allows speed control using the CSV (Cyclic Synchronous Velocity) mode of CiA402 profile on the set axis.
(2) In order to stop the specified speed operation, you can make a stop command or execute another motion function block.
(3) Velocity input specifies the speed to operate. When the sign of the operation speed value is positive (+ or no sign), it moves in the forward direction and when it is negative (-), it moves in the reverse direction.
(4) CmdPosMode is used to set the update methods of the current position at the time of command. Only the initial value of 0 is available and the current position of the command is updated using the feedback current position.
(5) The output InVelocity is turned on when the axis reaches the specified speed, and it is turned off when the specified speed operation is stopped.
(6) When this Motion Function Block is running, the axis status is 'Continuous Motion'.

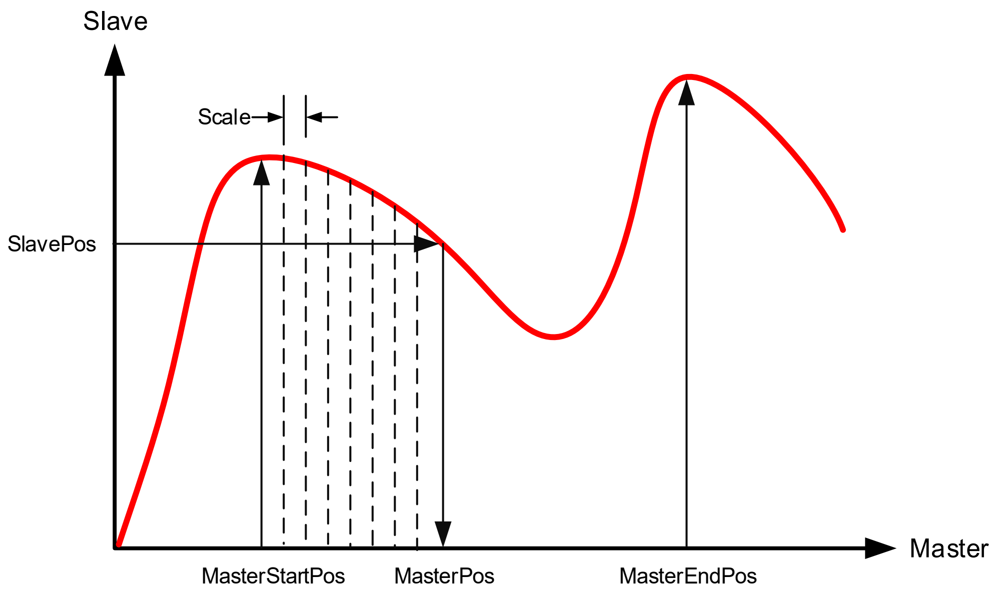
LS_ReadCamTableMasterPos	Availability
Read CAM table master position	**XMC**

Motion Function Block



Input-Output		
UINT	Axis	Set the command axis. (1~32: real axis/virtual axis, 33~36: virtual axis)
Input		
BOOL	Execute	Give cam table master position reading command to the relevant axis in the rising Edge.
UINT	CamTableID	Set the number of cam table to read (1~32)
LREAL	MasterStartPos	Start position to read position of cam main axis
LREAL	MasterEndPos	End position to read position of cam main axis
LREAL	SlavePos	Position of cam serve axis
LREAL	Scale	Accuracy of main axis position reading
Output		
BOOL	Done	Indicate that the cam table main axis reading is successfully completed.
BOOL	Busy	Indicated that the execution of motion function is not completed.
LREAL	MasterPos	Output the position of the slave
BOOL	Error	Indicate whether an error occurs or not
WORD	ErrorID	Output the number of error occurred while motion function block is running

(1) This motion function block outputs the position of the main axis corresponding to the position of the serve axis set in SlavePos, among the values between MasterStartPos and MasterEndPos in the specified cam table.

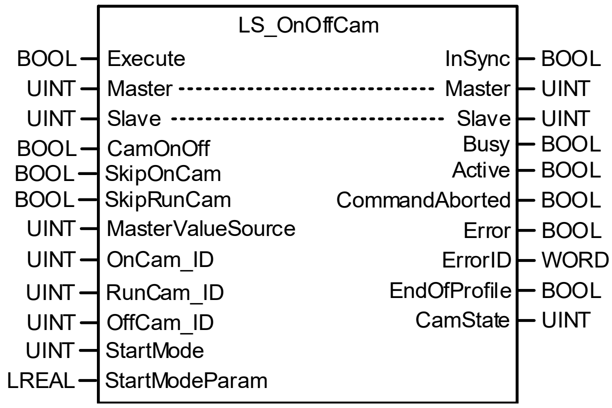


(2) Set the position of serve axis to read in the cam table as SlavePos value. Offset/Gear ratio/Phase correction operation applied to the command axis is not reflected in the MasterPos output.
(3) When the cam table master position reading operation is completed, the Done output turns on.
(4) The 'Scale', which is the accuracy value of the cam table master position reading, can't input 0. If the 'Scale' is 0, an error (error number: 0x0B) occurs. If the 'Scale' value is large, an error may occur between the magnified MasterPos value and the actual spindle position. Also, if the 'Scale' value is small, the execution time of the function block may become long.
(5) If the position of the main axis corresponding to the position of the serve axis set in SlavePos does not exist among the values between MasterStartPos and MasterEndPos, Error is On and "0x1124" occurs in ErrorID.
(6) The value of MasterEndPos must be greater than the value of MasterStartPos. If the MasterEndPos value is less than or equal to MasterStartPos, Error is On and "0x0B" occurs in ErrorID.

Category / Product	Module O/S	XG5000
XMC-E32A	V1.10	V4.23

# LS_OnOffCam	Availability
OnOff CAM Operation	**XMC**

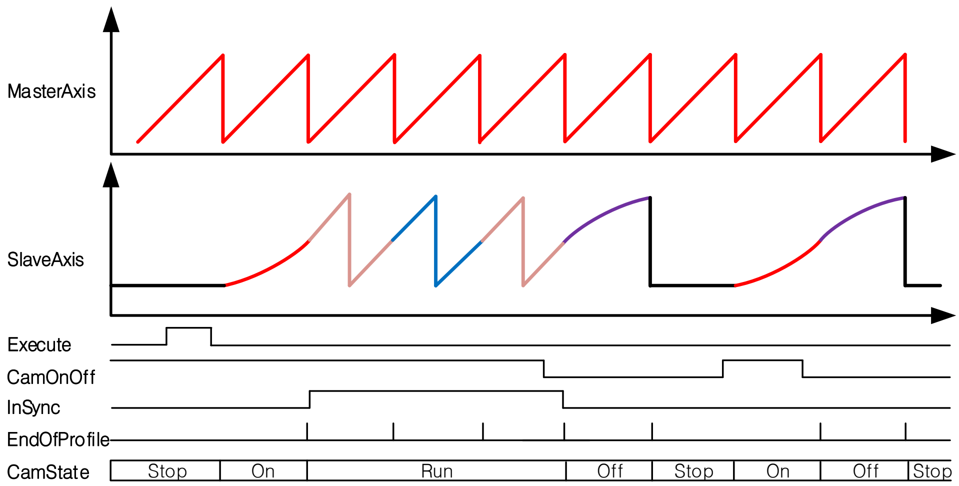
Motion Function Block



Input-Output		
UINT	Master	Set the main axis. (1-32: real/virtual axis, 33-36: virtual axis, 1001-1002: Encoder)
UINT	Slave	Set the serve axis. (1-32: real/virtual axis, 33-36: virtual axis)
Input		
BOOL	Execute	Give the OnOff cam operation command to the relevant axis on the rising Edge.
BOOL	CamOnOff	Set the on/off state of the cam operation. 1: Complete OnCam and switch to RunCam. 0: Complete OffCam in RunCam and switch the cam to the stop status
BOOL	SkipOnCam	Exclude OnCam from OnOff cam operation and carry out RunCam->OffCam in order.
BOOL	SkipRunCam	Exclude RunCam from OnOff cam operation and carry out OnCam->OffCam in order.
UINT	MasterValueSource	Select the source of the main axis for cam operation. 0: Synchronizes to the command position of the main axis. 1: Synchronizes to the current position of the main axis.
UINT	OnCam_ID	Specify the cam table to operate in the OnCam state.
UINT	RunCam_ID	Specify the cam table to operate in the RunCam state.
UINT	OffCam_ID	Specify the cam table to operate in the OffCam state.
UINT	StartMode	Specify the method for starting the cam operation. 0: Start when CamOnOff is set to 1. 1: Start when CamOnOff is set to 1 and the main axis reaches the position set in StartModeParam. 2: Start when CamOnOff is set to 1 and the main axis moves the distance set in StartModeParam. 3: Use the profile generated with LS_CrossSealCamGen.
LREAL	StartModeParam	Set the parameter according to the method for starting the cam operation.
Output		
BOOL	InSync	Indicates that cam operation has entered the RunCam state.

BOOL	Busy	Indicates that the execution of the motion function block is not completed.
BOOL	Active	Indicates that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicates that the current motion function block is interrupted by another command.
BOOL	Error	Indicates whether an error occurs or not.
WORD	ErrorID	Outputs the error ID that occurred while the motion function block is running.
BOOL	EndOfProfile	Indicates the end of the current cam operation.
UINT	CamState	0: Stop state 1: Executing OnCam 2: Executing RunCam 3: Executing OffCam

(1) This motion function block uses three cam tables to carry out the cam operation that is switched to a Stop state->OnCam->RunCam or a RunCam->OffCam->Stop state depending on the CamOnOff input.



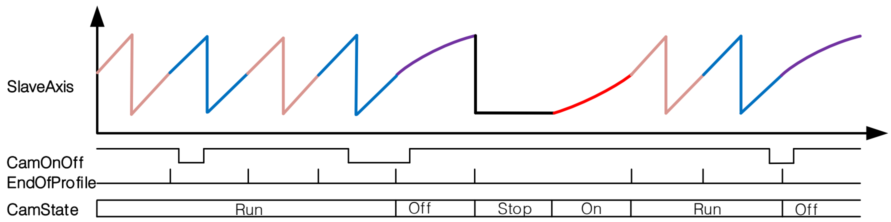
(2) The cam operation runs under a state where Execute is the rising Edge. The cam operation does not stop even if Execute is changed to Off during the operation. To stop the OnOffCam operation, you must give the MC_CamOut command or run another motion function block.

(3) If StartMode is set to 0, OnCam runs as soon as 1 is input in CamOnOff. If StartMode is set to 1, OnCam does not run as soon as 1 is input in CamOnOff, but when the position of the main axis passes by the position set in StartModeParam. If StartMode is set to 2, OnCam runs when 1 is input in CamOnOff and the main axis then moves in the distance set in StartModeParam.

(4) If you are using a cam generated with the LS_CrossSealCamGen function block, set StartMode to 3. If StartMode is set to 3 and the length of OnCam_ID is 270, the same operation is conducted as if StartMode is set to 1 and StartModeParam is 270. If OnCam_ID is 180, the same operation is conducted as if StartMode is set to 1 and StartModeParam is set to 0.

(5) EndOfProfile outputs On when passing the end of a profile during the operation of each OnCam/OffCam/RunCam cam profile.

(6) If the CamOnOff signal is Off, the operation to switch to RunCam->OffCam->Stop state is performed. If the CamOnOff signal is switched from Off to On in the RunCam state, the RunCam state is maintained if OffCam is not yet executed. In a state where OffCam is executed, the state switches to the OnCam->RunCam state again after switching to the OffCam->Stop state. (When turning off CamOnOff in RunCam, the operation must be maintained until an EndOfProfile signal is generated.)

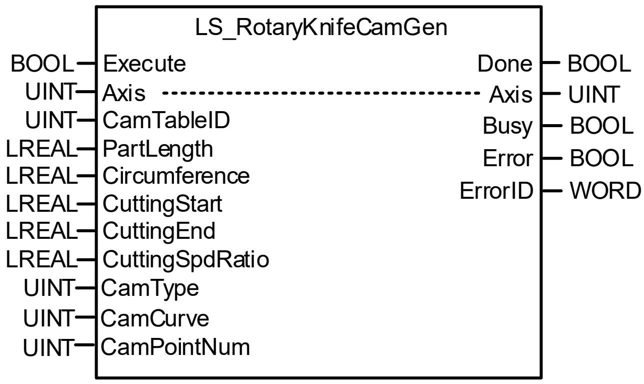


(7) If the SkipOnCam signal is On, RunCam is executed instantly without OnCam. If CamOnOff turns off after executing RunCam, perform the operation to switch to RunCam->OffCam->Stop state. In an operation where the SkipOnCam signal is On, the operation is executed from the middle of RunCam.
(8) If the SkipRunnCam signal is On, OffCam is executed without executing RunCam after executing OnCam. If CamOnOff is On at this time, the operation repeats in the order of OnCam->OffCam->Stop->OnCam->OffCam->Stop.
(9) To stop the OnOffCam operation completely, use the halt (MC_Halt) or immediate stop (MC_Stop) motion function block.
(10) The CamState value is output as Stop(0) / OnCam(1) / RunCam(2) / OffCam(3) depending on the state of cam operation.
(11) Once the cam operation set in RunCam_ID is executed, InSync outputs On.
(12) MasterValueSource selects the source of the main axis for synchronization. If set to 0, the serve axis performs cam operations based on the command position of the main axis calculated in the motion controller, and if set to 1, the serve axis performs cam operations based on the current position received via communication from the servo drive of the main axis.
(13) RunCam_ID sets the cam profile to execute during the operation of OnOffCam. Before executing RunCam in a Stop state, set the cam profile to run as OnCam_ID. OffCam_ID sets the cam profile to execute before RunCam reaches the Stop state. The setting range for each ID is 1-32, and an input value outside of the range causes a "0x1115" error in the motion function block.
(14) Any changes made to the MasterValueSource/OnCam_ID/RunCam_ID/OffCam_ID value during operation are not reflected.
(15) The corresponding axis is in a "SynchronizedMotion" state when this motion function block is running.
(16) For more information, see Chapter 8.6 RotaryKnife Operation under Chapter 8 Motion Control Function.
(17) This motion function block is supported in the following versions:

Category / Product	Module O/S	XG5000
XMC-E32A	V1.20	V4.25

LS_RotaryKnifeCamGen	**Availability**
RotaryKnife cam profile generation	**XMC**

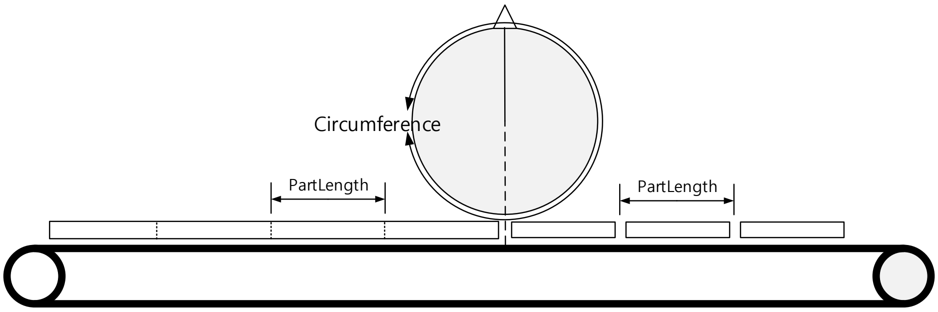
Motion Function Block



Input-Output		
UINT	Axis	Specify the axis to give the command. (1-32: real/virtual axis, 33-36: virtual axis)
Input		
BOOL	Execute	Performs cam profile generation in the rising Edge.
UINT	CamTableID	Set the cam table ID where the profile is stored.
LREAL	PartLength	Set the length of the object to cut by the RotaryKnife.
LREAL	Circumference	Set the circumference of the RotaryKnife.
LREAL	CuttingStart	Set the position for the RotaryKnife to start cutting.
LREAL	CutingEnd	Set the position for the RotaryKnife to end cutting.
LREAL	CuttingSpdRatio	Adjust the synchronization speed by a percentage while the RotaryKnife is cutting. (If 100 is entered, the cutting speed is synchronized 1:1 with the main axis.)
UINT	CamType	Set the type of the cam profile to generate. (0:ALL 1:RampIn 2:Running 3:RampOut) (4:sALL 5:sRampIn 6:Running 7:sRampOut)
UINT	CamCurve	Set the cam curve type used in cam profile generation. (0:Linear 1:Cubic)
UINT	CamPointNum	Set the number of cam points used for the cam profile.
Output		
BOOL	Done	Indicates that the cam profile is generated successfully.
BOOL	Busy	Indicates that the execution of the motion function block is not completed.
BOOL	Error	Indicates whether an error occurs or not.
WORD	ErrorID	Outputs the error ID occurred while the motion function block is running.

(1) This motion function block generates the cam profile which performs the RotaryKnife action.
(2) Use the cam profile generated through LS_RotaryKnifeCamGen in the LS_OnOffCam function block.
(3) On the PartLength input, enter the length of the object to perform cutting using the RotaryKnife.

(4) On the Circumference input, enter the circumference of the RotaryKnife.



(5) On the CuttingStart input, enter the starting position for the RotaryKnife to start cutting. On the CuttingStart input, enter the ending position for the RotaryKnife to end cutting. The speed of the conveyor and the RotaryKnife are synchronized between CuttingStart and CuttingEnd. (If you want a cutting region of 10 when the Circumference is 360, set CuttingStart to 175 and CuttingEnd to 185.)

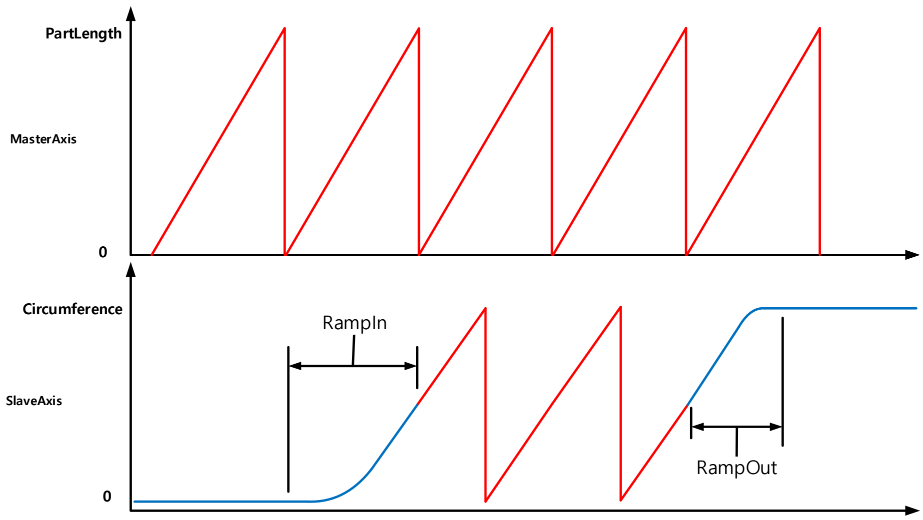
(6) On the generated cam profile, the movement amount of the main axis is 360Degree in ratio to PartLength. This means that you must set the gear ratio of the motor and the machine in the parameter so that 1 rotation of the main axis equals PartLength.

(7) On the generated cam profile, the movement amount of the serve axis is 360Degree in ratio to the Circumference. This means that you must set the gear ratio of the motor and the machine in the parameter so that 1 rotation of the serve axis equals the Circumference.

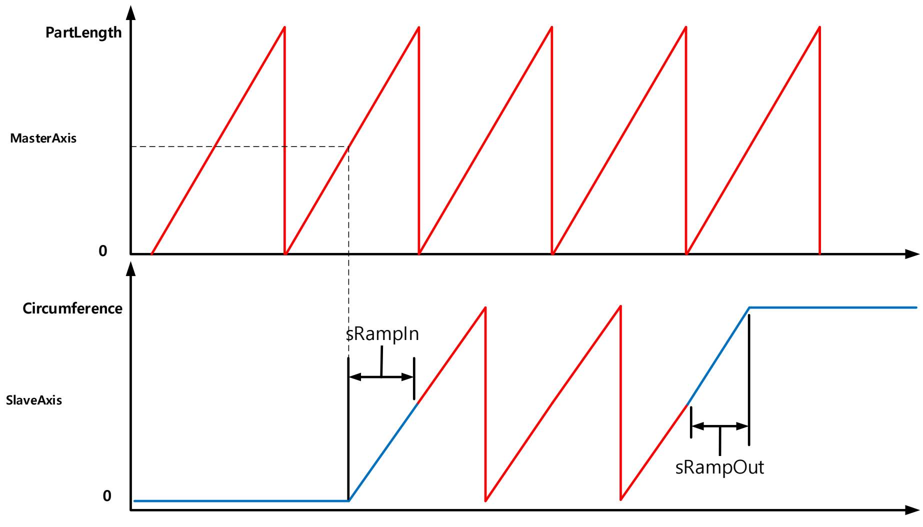
(8) For CuttingStart, you cannot enter a value that is less than 1/8 of the Circumference or greater than CuttingEnd. A "0x1172" error occurs if there is an error in the CuttingStart value.

(9) For CuttingEnd, you cannot enter a value that is greater than 7/8 of the Circumference or smaller than CuttingEnd. A "0x1172" error occurs if there is an error in the CuttingEnd value. To set the cutting region to the minimum, set CuttingEnd and CuttingStart as equal values.

(10) On the CamType, enter the type of cam profile to generate. Available values are 1:RampIn 2:Running 3:RampOut 5:sRampIn 6:Running 7:sRampOut. If you enter 0, RampIn/Running/RampOut will be generated at once. The Running type generates a cam profile which performs repeated cutting actions. The RampIn type generates a profile that includes the stop state to the action of the Running cam profile performing the cutting action. The RampOut type generates a profile to switch RotaryKnife from a running state to a stop state. A "0x1176" error occurs if the CamType value is outside of the range.



(11) The sRampIn and sRampOut types generate a shortened cam profile of RampIn and RampOut respectively. When operating using sRampIn and sRampOut and you want to main axis to reach the 1/2Circumference position of the serve axis, the main axis must start at the 1/2 position of PartLength.



(12) On the CuttingSpdRatio input, set the speed ratio for the cutting region. If CuttingSpdRatio is set to 100, a cam profile is generated which operates by synchronizing 1:1 with the speed of the main axis in the cutting region. As the CuttingSpdRatio value is higher, the faster the synchronization speed on the cutting region. The setting range of CuttingSpdRatio is 50-200 and a "0x1174" error occurs if there is an error in the CuttingSpdRatio value.

(13) On the CamCurve, enter the curve of the cam profile to generate. If you enter 0:Linear, a cam profile is generated using linear interpolation. Once you select linear interpolation, you must specify the number of cam profile points to generate by setting CamPointNum. Take care when setting the number of points as too little can lead to a shock due to the acceleration or deceleration of cam operation and too many can lead to an overload in the program due to the amount of computing resources for saving cam profiles. If you enter 1:Cubic, a cam profile is generated that uses cubic interpolation. A "0x1176" error occurs if the CamCurve value is outside of the range.

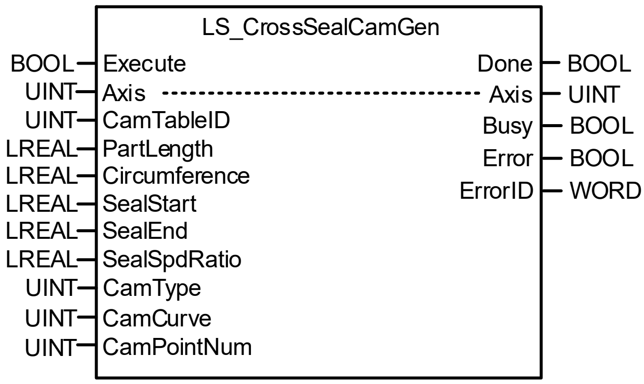
(14) The minimum number of cam points required for CamPointNum is 10 and a "0x1177" error occurs if there is an error in the CamPointNum value.

(15) This motion function block is supported in the following versions:

Category / Product	Module O/S	XG5000
XMC-E32A	V1.20	V4.25

LS_CrossSealCamGen	Availability
Cross sealer cam profile generation	XMC

Motion Function Block



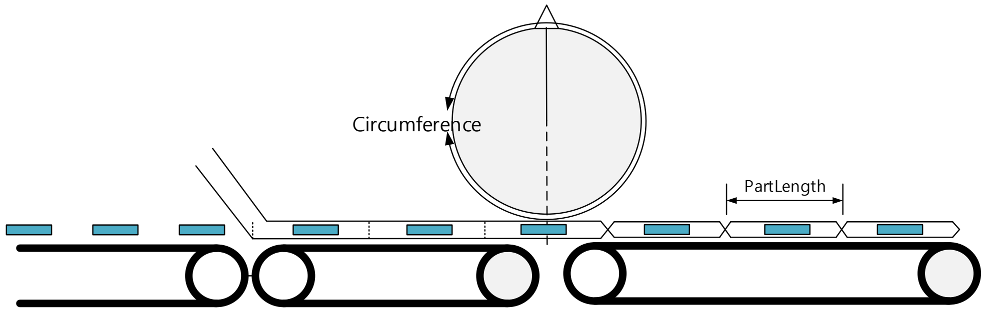
Input-Output		
UINT	Axis	Specify the axis to give the command. (1-32: real/virtual axis, 33-36: virtual axis)
Input		
BOOL	Execute	Performs cam profile generation in the rising Edge.
UINT	CamTableID	Set the cam table ID to store the cam profile.
LREAL	PartLength	Set length of the object sealed by the cross sealer.
LREAL	Circumference	Set circumference of the cross sealer.
LREAL	SealStart	Set the position for the cross sealer to start sealing.
LREAL	SealEnd	Set the position for the cross sealer to end sealing.
LREAL	SealSpdRatio	Adjust the synchronization speed in percentage while the cross sealer is sealing. (If 100 is entered, the sealing speed is synchronized 1:1 with the main axis.)
UINT	CamType	Set the type of the cam profile to generate. (0:ALL 1:RampIn 2:Running 3:RampOut) (4:sALL 5:sRampIn 6:Running 7:sRampOut)
UINT	CamCurve	Set the cam curve type used in cam profile generation. (0:Linear 1:Cubic)
UINT	CamPointNum	Set the number of cam points used for the cam profile.
Output		
BOOL	Done	Indicates that the cam profile is generated successfully.
BOOL	Busy	Indicates that the execution of the motion function block is not completed.
BOOL	Error	Indicates whether an error occurs or not.
WORD	ErrorID	Outputs the error ID occurred while the motion function block is running.

(1) This motion function block generates the cam profile which performs the cross sealer action. Use the cam profile generated through LS_CrossSealCamGen in the LS_OnOffCam function block.

(2) On the PartLength input, enter the length of the object to perform sealing using the cross sealer.

(3) On the Circumference input, enter the circumference of cross sealer.

(4) Both the main and serve axes of the generated cam profile is output within the 0-360 range. For the PartLength and Circumference values, you must enter the distance moved by the main axis when the main and serve axes move in 360 value.



(5) On the SealStart input, enter the starting position for the cross sealer to start sealing. On the SealStart input, enter the starting position for the cross sealer to end sealing. The speed of conveyor and the cross sealer are synchronized between SealStart and SealEnd. (If you want a sealing region of 10 when the Circumference is 360, set SealStart to 175 and SealEnd to 185.)

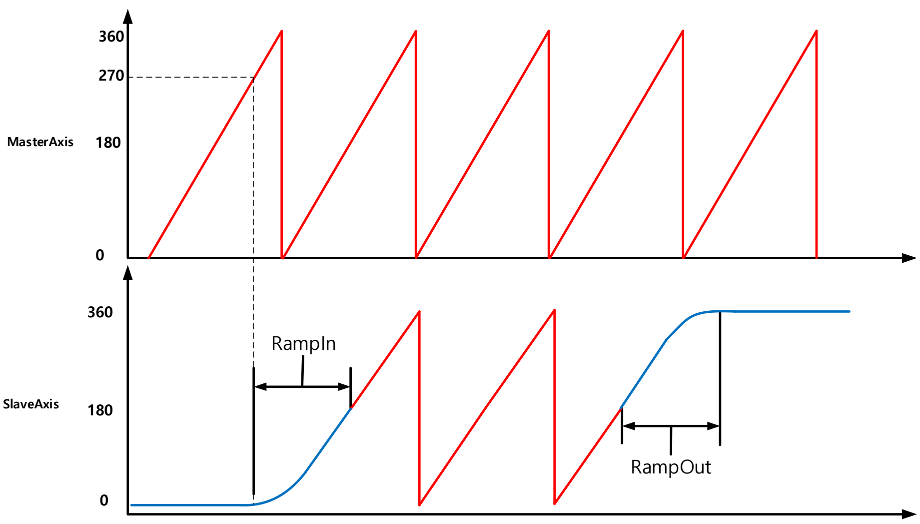
(6) On the generated cam profile, the movement amount of the main axis is 360 in ratio to PartLength. This means that you must set the gear ratio of the motor and the machine in the parameter so that when the main axis moves 360, the real distance equals PartLength.

(7) On the generated cam profile, the movement amount of the serve axis is 360 in ratio to Circumference. This means that you must set the gear ratio of the motor and the machine in the parameter so that when the serve axis moves 360, the real distance equals Circumference.

(8) For SealStart, you cannot enter a value that is less than 1/8 of the Circumference or greater than SealEnd. A "0x1172" error occurs if there is an error in the SealStart value.

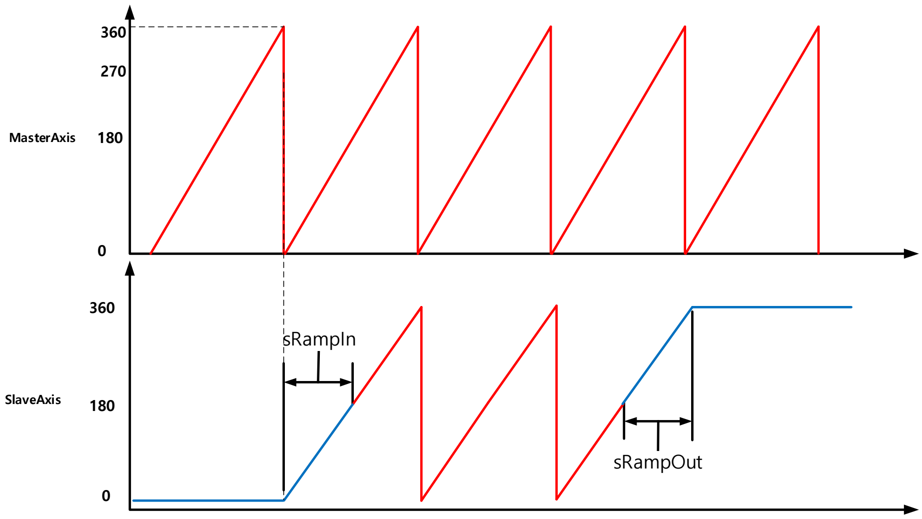
(9) For SealEnd, you cannot enter a value that is greater than 7/8 of the Circumference or smaller than SealEnd. A "0x1172" error occurs if there is an error in the SealEnd value. To set the sealing region to the minimum, set SealEnd and SealStart as equal values.

(10) On the CamType, enter the type of cam profile to generate. Available values are 1:RampIn 2:Running 3:RampOut 5:sRampIn 6:Running 7:sRampOut. If you enter 0, RampIn/Running/RampOut will be generated at once. The Running type generates a cam profile which performs repeated sealing actions. The RampIn type generates a profile that includes the stop state to the action of the Running cam profile performing the sealing action. The RampOut type generates a profile to switch the cross sealer from a running state to a stop state. A "0x1176" error occurs if the CamType value is outside of the range.



(11) The cam profile generated in the LS_CrossSealCamGen function is similar to the cam profile generated in the LS_RotaryCutCamGen. For the RampIn profile, the operation starts when the main axis is at 270 and not at 0. The profile also starts to perform sealing when the main axis is at 180 degrees.

(12) The sRampIn and sRampOut types generate a shortened cam profile of RampIn and RampOut respectively. When operating using sRampIn and sRampOut, the cam operation starts when the main axis is at 0.



(13) On the SealSpdRatio input, set the speed ratio for the sealing region. If SealSpdRatio is set to 100, a cam profile is generated which operates by synchronizing 1:1 with the speed of the main axis in the sealing section. The higher the SealSpdRatio value, the faster the synchronization speed in the cutting region. The setting range of SealSpdRatio is 50-200 and a "0x1174" error occurs if there is an error in the SealSpdRatio value.

(14) On the CamCurve, enter the curve of the cam profile to generate. If you enter 0:Linear, a cam profile is generated using linear interpolation. Once you select linear interpolation, you must specify the number of cam profile points to generate by setting CamPointNum. Take care when setting the number of points as too little can lead to a shock due to the acceleration or deceleration of cam operation and too many can lead to an overload in the program due to the amount of computing resources for saving cam profiles. If you enter 1:Cubic, a cam profile is generated that uses cubic interpolation. A "0x1176" error occurs if the CamCurve value is outside of the range.

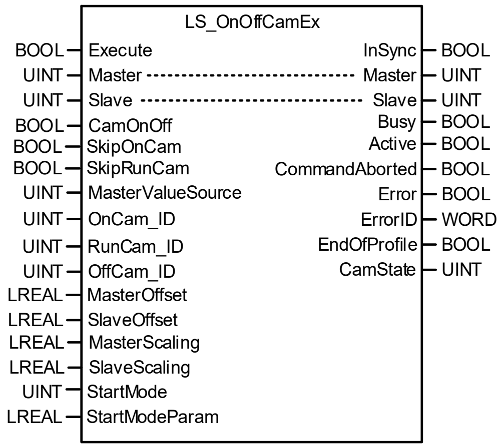
(15) The minimum number of cam points required for CamPointNum is 10 and a "0x1177" error occurs if there is an error in the CamPointNum value.

(16) This motion function block is supported in the following versions:

Category / Product	Module O/S	XG5000
XMC-E32A	V1.20	V4.25

LS_OnOffCamEx	Applied model
Extended OnOff CAM Operation	**XMC**

Motion function block type

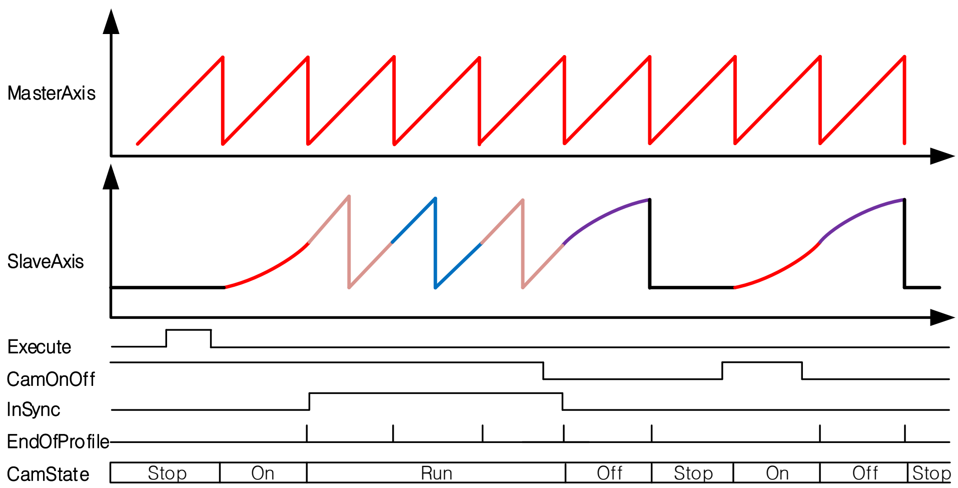


Input-Output		
UINT	Master	Set the main axis. (1-32: real/virtual axis, 33-36: virtual axis, 1001-1002: Encoder)
UINT	Slave	Set the serve axis. (1-32: real/virtual axis, 33-36: virtual axis)
input		
BOOL	Execute	Give the OnOff cam operation command to the relevant axis on the rising Edge.
BOOL	CamOnOff	Set the on/off state of the cam operation. 1: Complete OnCam and switch to RunCam. 0: Complete OffCam in RunCam and switch the cam to the stop status
BOOL	SkipOnCam	Exclude OnCam from OnOff cam operation and carry out RunCam->OffCam in order.
BOOL	SkipRunCam	Exclude RunCam from OnOff cam operation and carry out OnCam->OffCam in order.
UINT	MasterValueSource	Select the source of the main axis for cam operation. 0: Synchronizes to the command position of the main axis. 1: Synchronizes to the current position of the main axis.
UINT	OnCam_ID	Specify the cam table to operate in the OnCam state.
UINT	RunCam_ID	Specify the cam table to operate in the RunCam state.
UINT	OffCam_ID	Specify the cam table to operate in the OffCam state.
LREAL	MasterOffset	Sets the offset value of the main axis.
LREAL	SlaveOffset	Sets the offset value of the serve axis.
LREAL	MasterScaling	Specifies the scale of the main axis.
LREAL	SlaveScaling	Specifies the scale of the serve axis.
UINT	StartMode	Specify the method for starting the cam operation. 0: Start when CamOnOff is set to 1. 1: Start when CamOnOff is set to 1 and the main axis reaches the position set in StartModeParam. 2: Start when CamOnOff is set to 1 and the main axis moves the distance set in StartModeParam.

		3: Use the profile generated with LS_CrossSealCamGen.
LREAL	StartModeParam	Set the parameter according to the method for starting the cam operation.

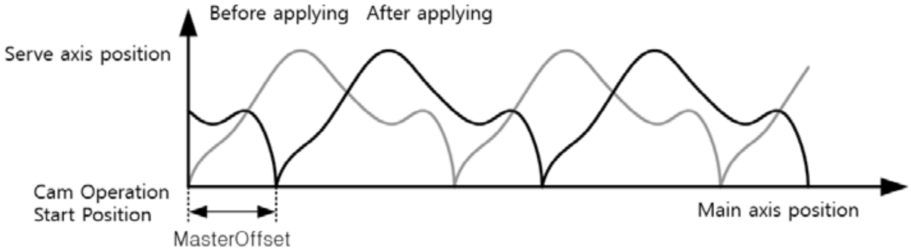
Print		
BOOL	InSync	Indicates that cam operation has entered the RunCam state.
BOOL	Busy	Indicates that the execution of the motion function block is not completed.
BOOL	Active	Indicates that the current motion function block is controlling the relevant axis.
BOOL	CommandAborted	Indicates that the current motion function block is interrupted by another command.
BOOL	Error	Indicates whether an error occurs or not.
WORD	ErrorID	Outputs the error ID that occurred while the motion function block is running.
BOOL	EndOfProfile	Indicates the end of the current cam operation.
UINT	CamState	0: Stop state 1: Executing OnCam 2: Executing RunCam 3: Executing OffCam

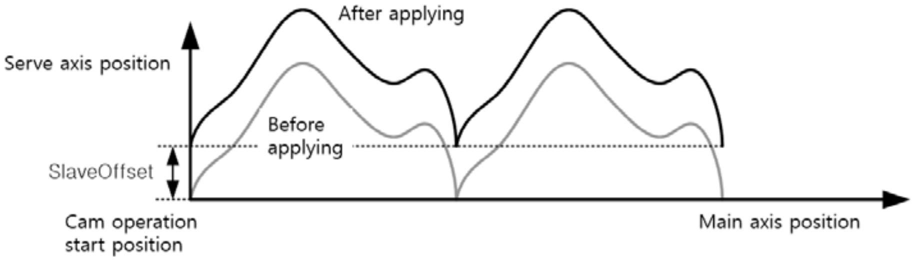
(1) This motion function block is a motion function block that performs cam operation to switch to Stop state -> OnCam -> RunCam or RunCam -> OffCam -> Stop state according to CamOnOff input by using 3 cam tables.



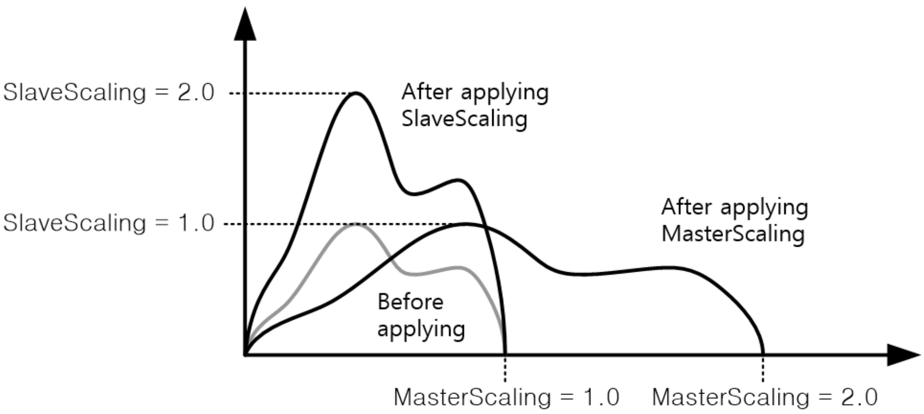
(2) The cam operation is executed while the Execute is at the rising edge. Cam operation does not stop even if Execute is changed to Off during operation. To stop the on-off cam operation, the MC_CamOut command must be issued or another motion function block must be activated.

(3) Set the offset of the cam table to apply to MasterOffset and SlaveOffset. MasterOffset sets offset from main axis starting point, and SlaveOffset sets offset from starting point of subordinate axis. Please refer to the figure below.





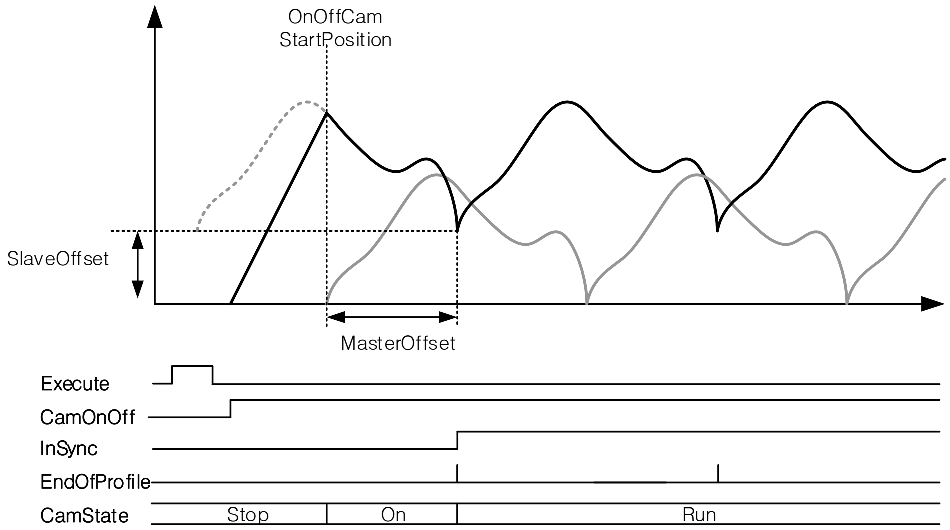
(4) For MasterScaling and SlaveScaling, set the scale of the cam data to be applied. MasterScaling sets the main axis data magnification and SlaveScaling sets the sub axis data magnification. Please refer to the figure below.



(5) If StartMode is set to 0, OnCam will be executed immediately when CamOnOff is set to 1. If StartMode is set to 1, OnCam will be executed when CamOnOff is set to 1 but the OnCam is not executed immediately and the main axis position passes the position set in StartModeParam. If StartMode is set to 2, OnCam will be executed after moving CamOnOff by the distance set in StartModeParam at the position where 1 is entered.

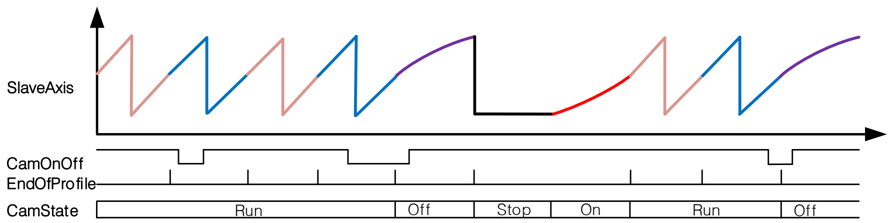
(6) If you use the cam created by LS_CrossSealCamGen function block, set StartMode to 3. If StartMode is set to 3, if OnCam_ID is 270, StartMode = 1 and StartModeParam = 270. If the length of OnCam_ID is 180, it performs the same operation as set StartMode = 1, StartModeParam = 0.

(7) When MasterOffset / SlaveOffset is set, if 1 is input to CamOnOff, operation starts to the OnOffCam start position set to StartMode and StartModeParam. OnOffCam operation is performed when the start position of OnOffCam is reached. If MasterOffset / SlaveOffset is set and StartMode is 0 and OnOffCam operation is performed, a shock may be generated at the start of operation.



(8) The EndOfProfile signal is turned on when the cam profile of OnCam / OffCam / RunCam is run.

(9) If the CamOnOff signal is off, RunCam-> OffCam-> Stop is executed. If the CamOnOff signal changes from Off to On in the RunCam state, the RunCam state is maintained if OffCam is not yet running. When OffCam is running, it switches to the OnCam-> RunCam state after switching to OffCam-> Stop state. (If CamOnOff is turned off in RunCam, it must be maintained until the EndOfProfile signal is generated.)



(10) If the SkipOnCam signal is On, RunCam will run immediately without OnCam. If CamOnOff signal is turned off after RunCam is executed, RunCam-> OffCam-> Stop is executed. When the SkipOnCam signal is ON, it is executed from the middle of RunCam.

(11) If the SkipRunnCam signal is On, RunCam is not executed after OnCam execution but OffCam is executed. At this time, when CamOnOff is ON, operation is repeated in the order of OnCam-> OffCam-> Stop-> OnCam-> OffCam-> Stop.

(12) To stop the on-off cam operation completely, use the Stop (MC_Halt) or Immediate Stop (MC_Stop) Motion Function Block.

(13) Depending on the cam operation status, CamState value is output as Stop (0) / OnCam (1) / RunCam (2) / OffCam (3) value.

(14) InSync output turns on when the cam operation set in RunCam_ID is executed.

(15) MasterValueSource selects the source of the main axis to be synchronized. When set to 0, the command position of the main axis computed by the motion controller is set to 1, and the subordinate axis performs cam operation based on the current position received from the main axis servo drive via communication.

(16) Set the cam profile to be run during running on-off cam to RunCam_ID. Set the cam profile to be executed to OnCam_ID before running RunCam in Stop state. OffCam_ID sets the cam profile to run before RunCam reaches the Stop state. The setting range of each ID is 1 ~ 32. If the input value is out of the setting range, error "0x1115" occurs in Motion Function Block.

(17) The value of MasterValueSource / OnCam_ID / RunCam_ID / OffCam_ID is not reflected even if you change it while driving.

(18) OnCam / RunCam / OffCam You can change the spindle value during operation (V1.5 or later).

(19) When this Motion Function Block is running, the corresponding axis is "Synchronized Motion" status.

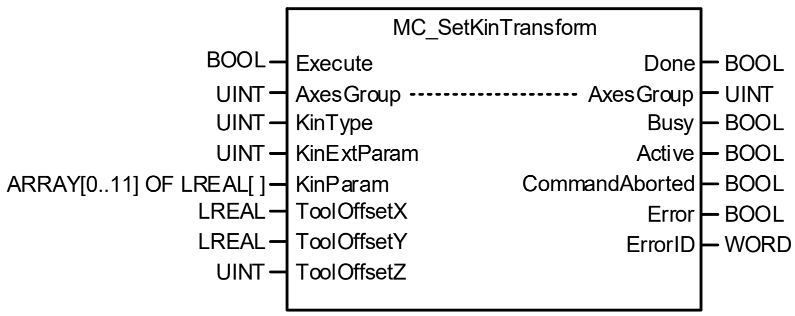
(20) For details, refer to 8.6 RotaryKnife Operation of Chapter 8 Motion Control Function.

(21) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.50	V4. ??

MC_SetKinTransform	**Availability**
Machine information setting	**XMC**

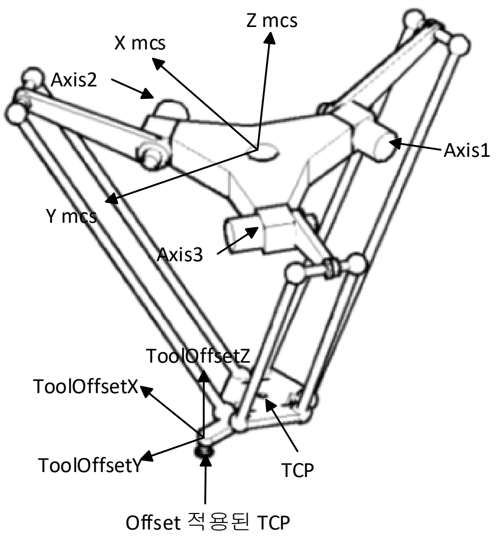
Motion Function Block



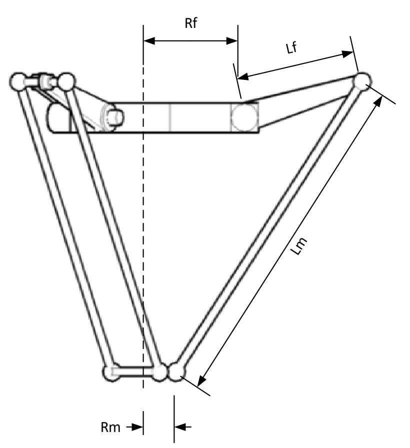
Input-Output		
UINT	AxesGroup	Set the axes group to set the machine information.(1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give the machine information setting command on the axis in the rising Edge.
UINT	KinType	Set the machine type.(0:XYZ, 1:Delta3)
UINT	KinExtParam	Unused
LREAL[]	KinParam	Set the machine information.
LREAL	ToolOffsetX	Set the X axis offset of at the end of the machine.
LREAL	ToolOffsetY	Set the Y axis offset of at the end of the machine.
LREAL	ToolOffsetZ	Set the Z axis offset of at the end of the machine.
Output		
BOOL	Done	Indicate the machine information setting is successfully completed.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that machine information setting of the current axis is running.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block sets the ACS and MCS conversion based on the machine model defined in advance at AxesGroup.

(2) The same setting can be applied to the XG5000 group parameter settings.
(3) The KinType input is used to set the type of the device. You can set the device as shown below.
  1) 0: None
  2) 1: XYZ
  3) 2: Delta3
  4) 3: Delta3R
  5) 4: LinearDelta3
  6) 5: LinearDelta3R
(4) KinParam input is used to set the device information. (It is not set for XYZ type.)
(5) ToolOffsetX / ToolOffsetY / ToolOffsedZ are the functions to set the offset at the end point of the device. In order to cope with the case where a separate device is connected to the end of the TCP of the robot, the tool offset function is provided separately from the device information.

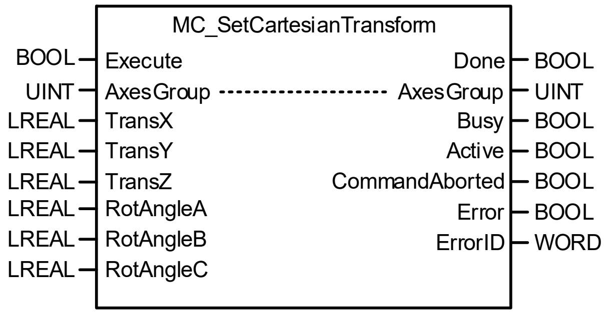


(6) When using Delta3, the device setting information is as follows. For more information, refer to 8.4.4 Machine information setting.

<table>
<tr><td rowspan="5">
</td><td>Parameter</td><td>Description</td></tr>
<tr><td>KinParam[0]</td><td>Lf:: Link length of the fixed frame(mm)</td></tr>
<tr><td>KinParam[1]</td><td>Lm: Link length of the moving frame(mm)</td></tr>
<tr><td>KinParam[2]</td><td>Rf: Distance from center of the fixed frame to the link fo the fixed frame (mm)</td></tr>
<tr><td>KinParam[3]</td><td>Rm: Distance from the center of the moving frame to the link of the moving frame (mm)</td></tr>
</table>

MC_SetCartesianTransform	Availability
PCS setting	XMC

Motion Function Block

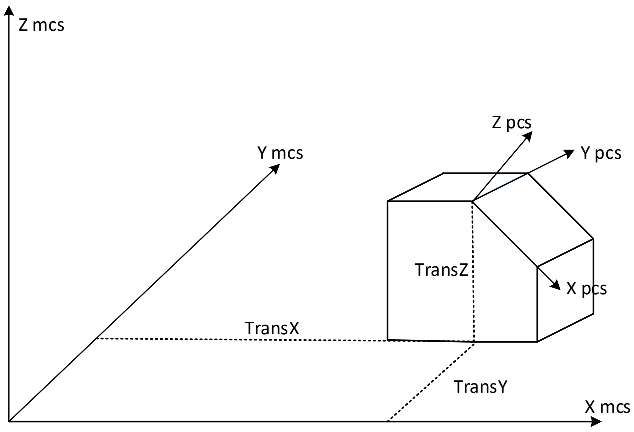


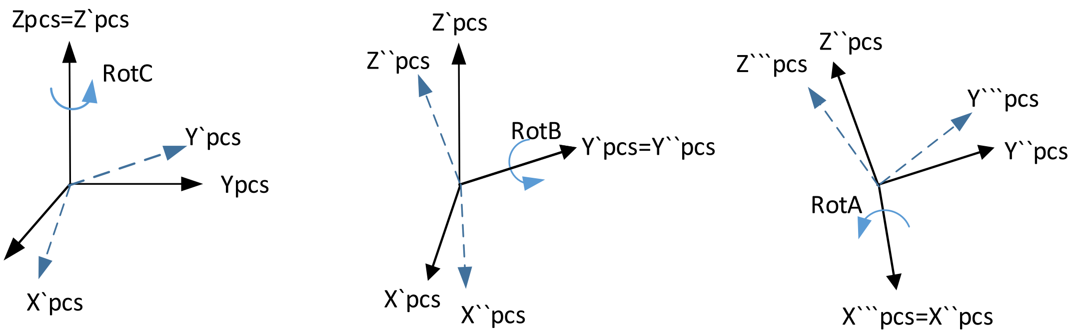
Input-Output		
UINT	AxesGroup	Set the axes group to set the PCS.(1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give the PCS setting command on the axes group in the rising Edge.
LREAL	TransX	Movement from MCS to X Axis(mm)
LREAL	TransY	Movement from MCS to Y Axis(mm)
LREAL	TransZ	Movement from MCS to Z Axis(mm)
LREAL	RotAngleA	X Axis rotation amount (Degree)(reserved)
LREAL	RotAngleB	Y Axis rotation amount (Degree)(reserved)
LREAL	RotAngleC	Z Axis rotation amount (Degree)
Output		
BOOL	Done	Indicate the PCS setting is successfully completed.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that machine information setting of the current axis is running.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block sets the perpendicular coordinate conversion between MCS and PCS at AxesGroup.
(2) Axis group setting can be performed in the same way at XG5000 axis group parameter setting.

PCS Configuration	X-axis feed amount	0 mm
	Y-axis feed amount	0 mm
	Z-axis feed amount	0 mm
	X-axis rotation	0 deg
	Y-axis rotation	0 deg
	Z-axis rotation	0 deg

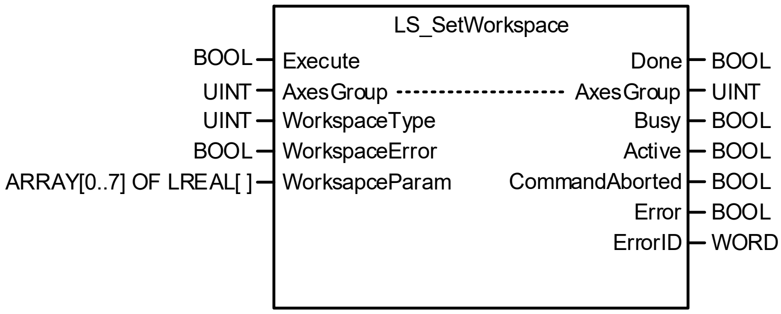
(3) TransX/TransY/TransZ represent the distance of movement from the MCS origin point to the PCS origin point. RotA/RotB/RotCare rotation values for PCS. RotA represents PCS rotation along X-axis. RotB represents PCS rotation along Y-axis. RotC represents PCS rotation along Z-axisPCS rotation is performed in the following order: RotC->RotB->RotA.
Refer to chapter 8.4.3 PCS setting in motion controller's manual for more details.





LS_SetWorkspace	Availability
Work space setting	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the axes group to set the work space.(1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give the work space setting command on the axes group in the rising Edge.
UINT	WorkspaceType	Set the work space type (1:Rectangle 2:Cylinder 3:Delta3 4:Sector)
BOOL	WorkspaceError	Set whether an error occurs or not when a coordinate system operation exceeds the work space.
LREAL[]	WorkspaceParam	Set the parameter of the work space.
Output		
BOOL	Done	Indicate the PCS setting is successfully completed.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that machine information setting of the current axis is running.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block sets the work space based on the coordinate system at the axes group designated by AxesGroup input.
(2) The same setting can be performed in XG5000 group parameter setting.

Workspace configuration	Workspace type	0: Rectangle
	Workspace error check	0: Disable
	Workspace Parameter1	170 mm
	Workspace Parameter2	-170 mm
	Workspace Parameter3	170 mm
	Workspace Parameter4	-170 mm
	Workspace Parameter5	-380 mm
	Workspace Parameter6	-580 mm
	Workspace Parameter7	0
	Workspace Parameter8	0

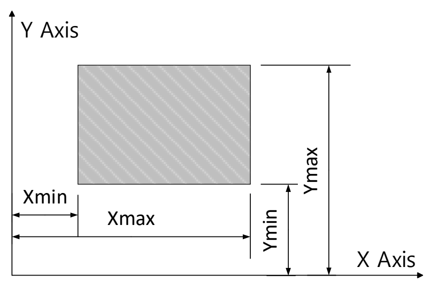
(3) WorkspaceType can be selected from 4 types (1:Rectangle 2:Cylinder 3:Delta3 4:Sector).

(4) WorkspaceError input determines whether an error occurs when a coordinate system operation exceeds the work space.

(5) WorkspaceParam input sets the parameters depending on the work space type.

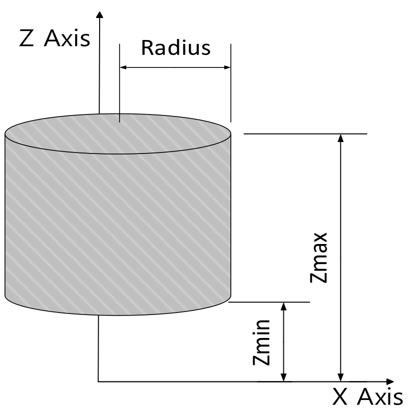
(6) Refer to chapter 8.4.5 Workspace setting in motion controller's manual for more details.

1) Rectangle



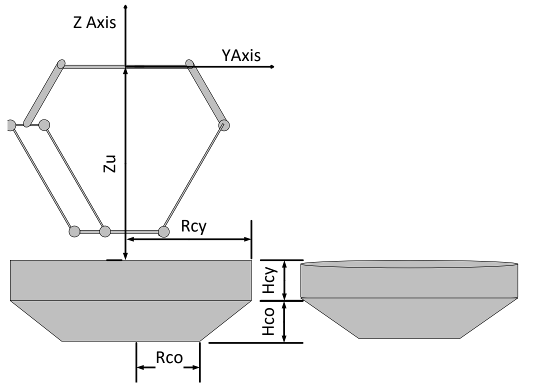
Parameter	Value
WorkspaceParam[0]	X max(mm)
WorkspaceParam[1]	X min(mm)
WorkspaceParam[2]	Y max(mm)
WorkspaceParam[3]	Y min(mm)
WorkspaceParam[4]	Z max(mm)
WorkspaceParam[5]	Z min(mm)

2) Cylinder



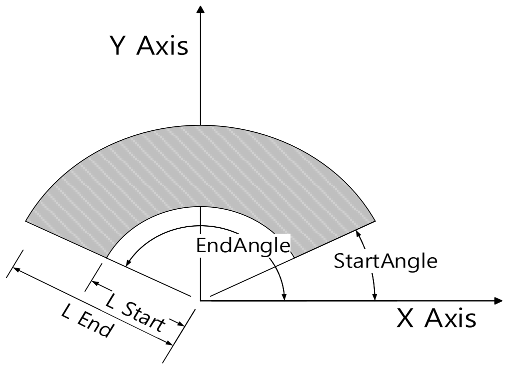
Parameter	Value
WorkspaceParam[0]	Radius(mm)
WorkspaceParam[1]	Z max(mm)
WorkspaceParam[2]	Z min(mm)

3) Delta



Parameter	Value
WorkspaceParam[0]	Zu(mm)
WorkspaceParam[1]	Hcy(mm)
WorkspaceParam[2]	Hco(mm)
WorkspaceParam[3]	Rcy(mm)
WorkspaceParam[4]	Rco(mm)
WorkspaceParam[5]	-

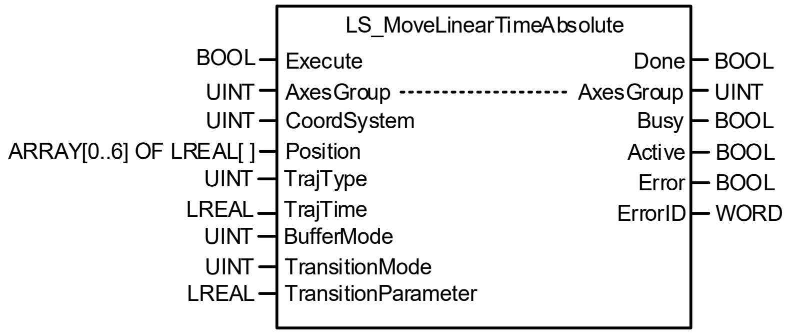
4) Sector



Parameter	Value
WorkspaceParam[0]	L end (mm)
WorkspaceParam[1]	L start(mm)
WorkspaceParam[2]	Z max(mm)
WorkspaceParam[3]	Z min(mm)
WorkspaceParam[4]	EndAngle(degree)
WorkspaceParam[5]	StartAngle(degree)

LS_MoveLinearTimeAbsolute	Availability
Time-Linear interpolation operation for absolute position of coordinate system	**XMC**

Motion Function Block

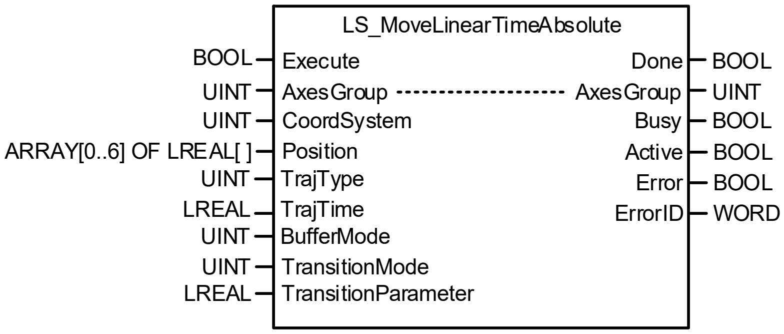


Input-Output		
UINT	AxesGroup	Set the axes group to set the absolute position time linear interpolation.(1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give the time linear interpolation command on the axes group in the rising Edge.
UINT	CoordSystem	Set the coordinate system type (1:MCS 2:PCS)
LREAL[]	Position	Enter the target position of the end point of the machine.
UINT	TrajType	Enter the operation acc/dec type.(0:Trapezoid 1:Sine1 2:Sine2)
LREAL	TrajTime	Set the time taken to reach the target position.(msec)
UINT	BufferMode	Give the sequential operation of the motion function block. (Refer to the 6.1.4 BufferMode input)
UINT	TransitionMode	Unused
LREAL	TransitionParameter	Unused
Output		
BOOL	Done	Indicate the PCS setting is successfully completed.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that machine information setting of the current axis is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block issues absolute position/time linear interpolation command based on coordinate system on the axes group designated by AxesGroup input
(2) When this motion function block is executed, interpolation control is performed in a linear trajectory from the machine end point of each axes group to the target position.
(3) TrajType input sets the type of velocity, acceleration, deceleration of interpolation trajectory. The type can be selected from three types: Trapezoid/Sine1/Sine2.
(4) TrajTime sets the time taken to reach the target position.
(5) Please refer to 8. 4. 6 Coordinate System Absolute Position/Time Linear Interpolation Control further details.

LS_MoveLinearTimeRelative	Availability
Time-Linear interpolation operation for relative position of coordinate system	**XMC**

Motion Function Block

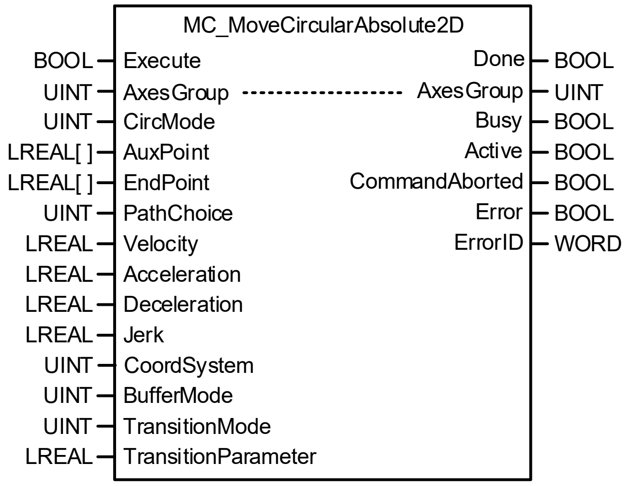


Input-Output		
UINT	AxesGroup	Set the axes group to set the relative position time linear interpolation.(1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	Give the time linear interpolation command on the axes group in the rising Edge.
UINT	CoordSystem	Set the coordinate system type (1:MCS 2:PCS)
LREAL[]	Position	Enter the target position of the end point of the machine.
UINT	TrajType	Enter the operation acc/dec type.(0:Trapezoid 1:Sine1 2:Sine2)
LREAL	TrajTime	Set the time taken to reach the target position.(msec)
UINT	BufferMode	Give the sequential operation of the motion function block. (Refer to the 6.1.4 BufferMode input)
UINT	TransitionMode	Unused
LREAL	TransitionParameter	Unused
Output		
BOOL	Done	Indicate the PCS setting is successfully completed.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that machine information setting of the current axis is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block issues relative position/time linear interpolation command based on coordinate system on the axes group designated by AxesGroup input
(2) When this motion function block is executed, interpolation control is performed in a linear trajectory from the machine end point of each axes group to the target position.
(3) TrajType inputs set the type of velocity, acceleration, deceleration of interpolation trajectory. The type can be selected from three types: Trapezoid/Sine1/Sine2.
(4) TrajTime sets the time taken to reach the target position.
(5) Please refer to 8. 4. 6 Coordinate System Relative Position/Time Linear Interpolation Control for further details.

MC_MoveCircularAbsolute2D	Availability
Circular interpolation operation for absolute position of coordinate system	**XMC**

Motion Function Block



Input-Output

UINT	AxesGroup	Set the axes group to set the absolute position circular interpolation.(1 ~ 16 : Group 1 ~ Group 16)

Input

BOOL	Execute	Give the circular interpolation command on the axes group in the rising Edge.
UINT	CircMode	The way to set the circular interpolation [0: Middle point Aux point, 1: Center point, 2: Radius]
LREAL[ ]	AuxPoint	The auxiliary point position for circular interpolation is designated as an absolute coordinate.
LREAL[ ]	EndPoint	Set the circular end point as an absolute coordinate.
BOOL	PathChoice	Set the circular path. 0: clockwise direction, 1: counter-clockwise direction
LREAL	Velocity	Set the maximum velocity of the path.. [u/s]
LREAL	Acceleration	Set the maximum acceleration. [u/s^2]
LREAL	Deceleration	Set the minimum decleration. [u/s^2]
LREAL	Jerk	Set the maximum acc/dec jerk. [u/s^3]
UINT	CoordSystem	Set the coordinate system's type. (1:MCS 2:PCS)
UINT	BufferMode	the sequential operation of the motion function block. (Refer to the chapter 6.1.4 BufferMode input)

UINT	TransitionMode	Unused
LREAL	TransitionParameter	Unused
Output		
BOOL	Done	Indicate whether to reach the specified point.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that whether or not motion function block is controlling the group.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block issues absolute position circular interpolation command based on coordinate system on the axis group designated by AxesGroup input.

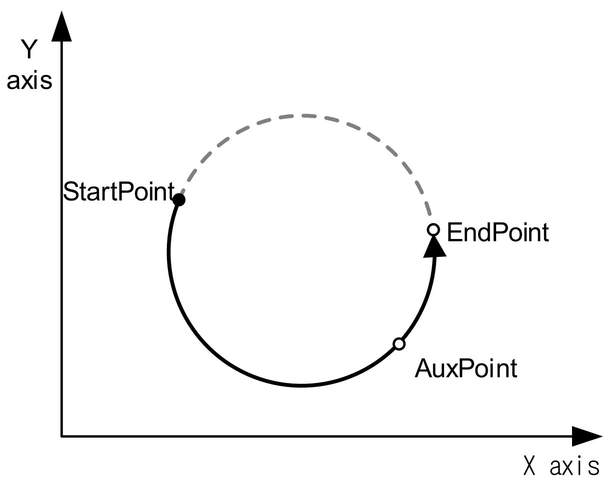
(2) When this motion function block starts, each axis performs circular trajectory interpolation control referring to the auxiliary point input, and the movement direction is determined by Path Choice input. If PathChoice input is set to 0, circular interpolation is operated in a clockwise direction, and if it is set to 1, circular interpolation is operated in a counter-clockwise direction.

(3) At AuxPoint and EndPoint input, designate the arrangement of the absolute position of auxiliary points to refer to for circular interpolation of each axis. The input corresponds in the order of X, Y, Z, unlike MC_MoveCircularAbsolute.

(4) Velocity, Acceleration, Deceleration, Jerk input sets the velocity, acceleration, deceleration, and acceleration/deceleration rate change of the interpolation path, respectively.

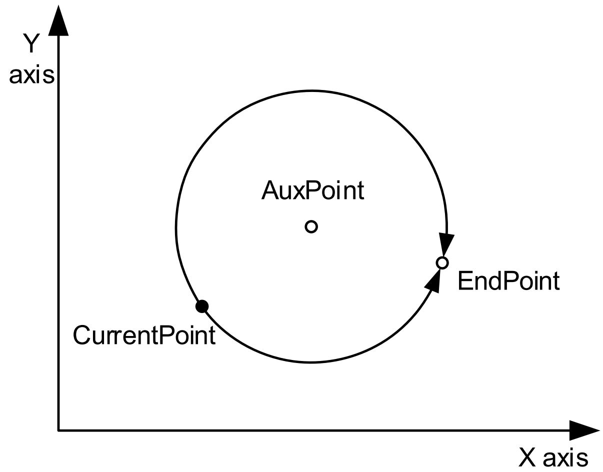
(5) CircMode input sets the circular interpolation method. The circular interpolation methods corresponding to CircMode values are as follows.

(a) Circular Interpolation Using Midpoint Specification (CircMode = 0)
This method performs circular interpolation by starting operation at the start position, passing the designated midpoint, and reaching the target position. In the figure below, the start position corresponds to the axes group coordinate at the start of the command, the midpoint corresponds to the coordinate input for the AuxPoint, and the target position corresponds to the absolute coordinate input for the EndPoint.

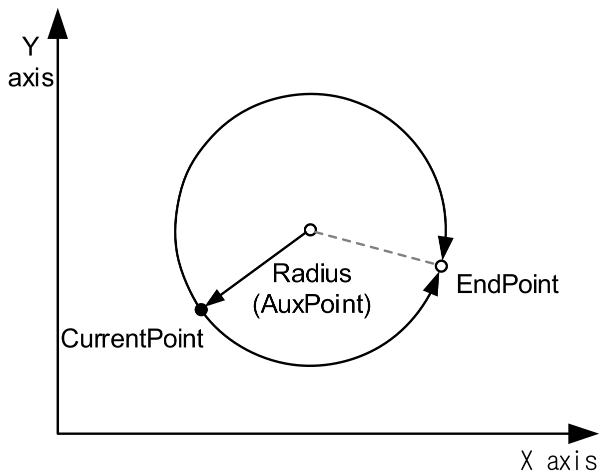


(b) Circular Interpolation Using Center Point Specification (CircMode = 1)

This method performs circular interpolation to the target position by starting operation at the current position, and following a circular trajectory of which diameter corresponds to the distance to the designated center point. In the figure below, the current position corresponds to the axes group coordinate at the start of the command, the center point corresponds to the coordinate input for the AuxPoint, and the target position corresponds to the absolute coordinate input for the EndPoint.



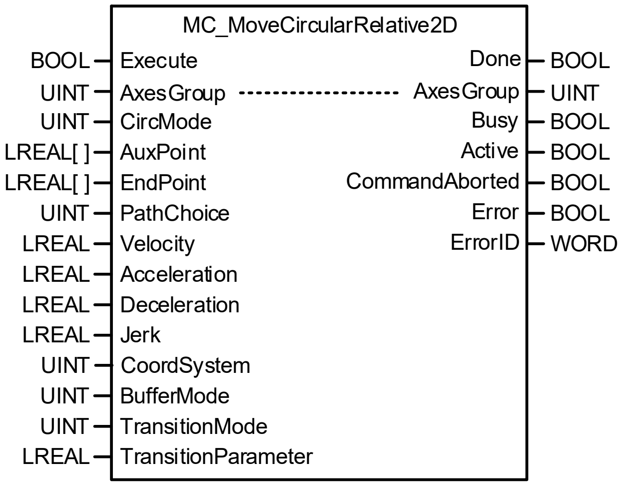
(c) Circular Interpolation using Radius Speciation (CircMode = 2)
This method performs circular interpolation to the target position by starting operation at the current position, and following a circular trajectory with a designated radius from the current position to the target position. In the figure below, the current position corresponds to the axes group coordinate at the start of the command, the radius corresponds to the X coordinate input for the AuxPoint, and the target position corresponds to the absolute coordinate input for the EndPoint.



(6) Refer to chapter 8.4.7 circular interpolation control in motion controller's manual for more details.
(7) The changed parameters are applied by re-executing the function block (Execute input is On) before the command is completed.
(8) Only, Velocity, Acceleration, Deceleration, Jerk, AuxPoint, Endpoint inpun can be updated.

# MC_MoveCircularRelative2D	Availability
Circular interpolation operation for relative position of coordinate system	**XMC**

Motion Function Block



Input-Output

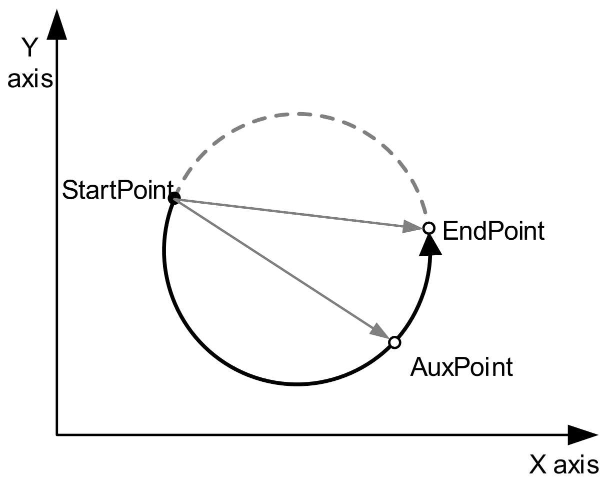
UINT	AxesGroup	Set the group to do relative position circular interpolation operation. (1 ~ 16: Group 1 ~ Group 16)

Input

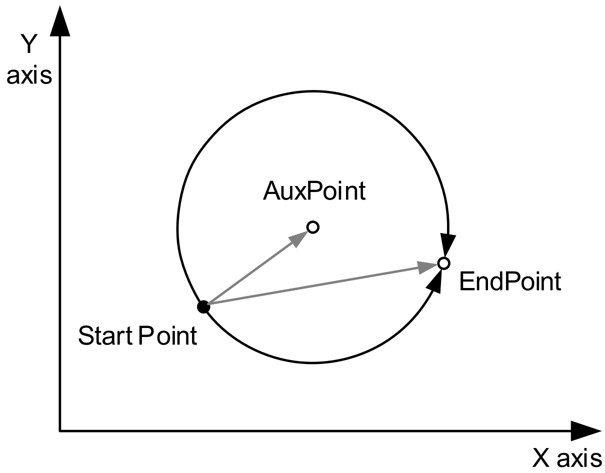
BOOL	Execute	Give relative position circular interpolation operation command on the group in the rising Edge.
UINT	CircMode	Circular interpolation method setting [0: Midpoint, 1: Central point, 2: Radius]
LREAL[ ]	AuxPoint	Specify the position of auxiliary point depending on the circular interpolation method in a relative coordinate.
LREAL[ ]	EndPoint	Specify the end point of the circular trajectory as a relative coordinate from the start point.
BOOL	PathChoice	Set the circular path. 0: clockwise direction, 1: counter-clockwise direction
LREAL	Velocity	Set the maximum velocity of the path. [u/s]
LREAL	Acceleration	Set the maximum acceleration. [u/s^2]
LREAL	Deceleration	Set the minimum decleration. [u/s^2]
LREAL	Jerk	Set the maximum acc/dec jerk. [u/s^3]
UINT	CoordSystem	Set the coordinate system’s type. (1:MCS 2:PCS)
UINT	BufferMode	The sequential operation of the motion function block.

		(Refer to the chapter 6.1.4 BufferMode input)
UINT	TransitionMode	Unused
LREAL	TransitionParameter	Unused
Output		
BOOL	Done	Indicate whether to reach the specified point.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that whether or not motion function block is controlling the group.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

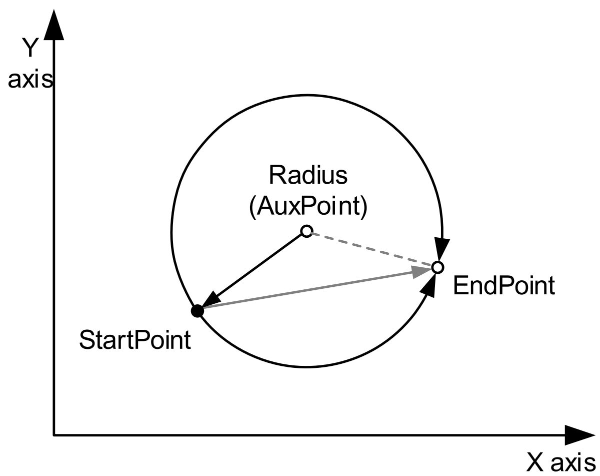
(1) This motion function block issues relative position circular interpolation command on the axes group designated by AxesGroup input.
(2) When this motion function block is executed, each axis performs circular interpolation control referring to the auxiliary point input, and the direction is determined by Path Choice input. If PathChoiceinput is set to 0, circular interpolation is operated in a clockwise direction, and if it is set to 1, circular interpolation is operated in a counter-clockwise direction.
(3) At AuxPoint and EndPoint input, designate the arrangement of the relative position of auxiliary points to refer to for circular interpolation of each axis. The input arrangement and the axes of the group correspond to the designated axis IDs [ID1, ID2, ID3, ···], in that order. (Since the number of axes comprising a group to issue circular interpolation command is 3, arrangements of three sizes should be input for the Position input.)
(4) In Velocity, Acceleration, Deceleration, Jerk inputs, the acceleration, deceleration, change rate of acceleration, velocity of the interpolation path are specified, respectively.
(5) CircMode input sets the circular interpolation method. The circular interpolation methods corresponding to CircMode values are as follows.
   (a) Circular Interpolation Using Midpoint Specification (BORDER, CircMode = 0)
   This method is to perform the circular interpolation to the target position through the midpoint position after starting operation at the current position. In the figure below, the current position corresponds to the axes group coordinate at the start of the command, the midpoint corresponds to the coordinate input for the AuxPoint, and the target position corresponds to the relative coordinate input for the EndPoint.



(b) Circular Interpolation Using Center Point Specification (CircMode = 1)
This method is to perform the circular interpolation to the target position by starting operation at the start position, and following a circular trajectory of which diameter corresponds to the distance to the designated center point. In the figure below, the current position corresponds to the axes group coordinate at the start of the command, the center point corresponds to the coordinate input for the AuxPoint, and the target position corresponds to the relative coordinate input for the EndPoint.



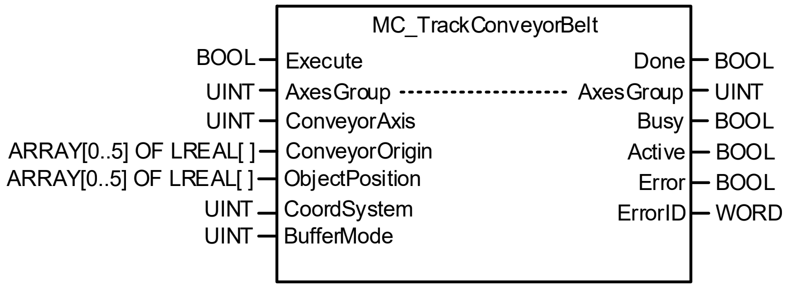
(c) Circular Interpolation using Radius Speciation (CircMode = 2)
This method is to perform the circular interpolation to the target position by starting operation at the current position, passing the designated center point, and reaching the target position. In the figure below, the current position corresponds to the axes group coordinate at the start of the command, the diameter corresponds to the X coordinate input for the AuxPoint, and the target position corresponds to the relative coordinate input for the EndPoint.



(6) Refer to chapter 8.4.7 circular interpolation control in motion controller’s manual for more details.

MC_TrackConveyorBelt	Availability
Synchronization setting of conveyor belt	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group to do conveyor belt synchronized setting.(1 ~ 16: Group 1 ~ Group 16)
Input		
BOOL	Execute	Give the conveyor belt synchronized setting command on the axes group in the rising Edge.
UINT	ConveyorAixs	Set the conveyor axis.(1 ~ 32 : Axis 1~Axis 32)
LREAL[]	ConveyorOrigin	Enter the position from the MCS home position to the conveyor origin point.
LREAL[]	ObjectPosition	Input the conveyor home position to the object to work on.
UINT	CoordSystem	Set the coordinate system type.(2:PCS)
UINT	BufferMode	Set the sequential operation of the motion function block. (Refer to the 6.1.4 BufferMode input)
Output		
BOOL	Done	Indicate the PCS setting is successfully completed.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that machine information setting of the current axis is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block sets conveyor belt synchronized operation for the axes group designated by AxesGroup input.

(2) This motion function block is not directly involved in operation. When this function block is executed, the coordinate system operation using the PCS coordinate system is synchronized to the designated conveyor belt axis.

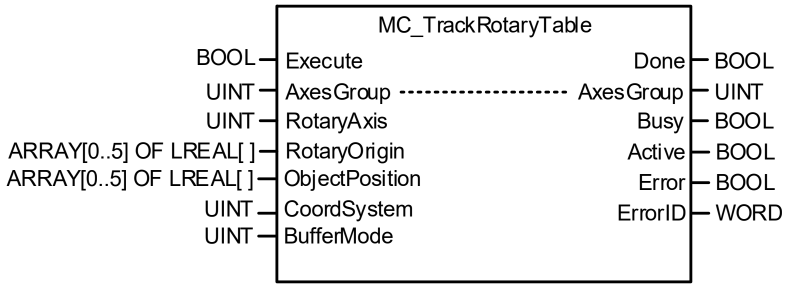
(3) ConveyorAxis can be set to between 1 and 32. An axis belonging to the axes group set as AxesGroup cannot be

designated.

(4) The operation parameter of the axis designated as ConveyorAxis must be in mm/inch.
(5) Infinite running repeat must be set for the operation parameter of the axis designated as ConveyorAxis
(6) Synchronized conveyor operation is terminated by performing coordinate system operation using the PCS coordinate system or performing PCS setting with MC_SetCartesianTransform function block.
(7) Refer to chapter 8.4.9 synchronized conveyor operation in motion controller’s manual for more details

MC_TrackRotaryTable	Availability
Synchronization setting of rotary table	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group to do rotary table synchronized setting.(1 ~ 16: Group 1 ~ Group 16)
Input		
BOOL	Execute	Give the rotary table synchronized setting command on the axes group in the rising Edge.
UINT	RotaryAixs	Set the rotary table axis.(1 ~ 32 : Axis 1~Axis 32)
LREAL[]	RotaryOrigin	Enter the position from the MCS home position to the rotary table origin point.
LREAL[]	ObjectPosition	Input the rotary table home position to the object to work on.
UINT	CoordSystem	Set the coordinate system type.(2:PCS)
UINT	BufferMode	Set the sequential operation of the motion function block. (Refer to the 6.1.4 BufferMode input)
Output		
BOOL	Done	Indicate the PCS setting is successfully completed.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that machine information setting of the current axis is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block sets rotary table synchronized operation for the axes group designated by AxesGroup input.

(2) This motion function block is not directly involved in operation. When this function block is executed, the coordinate system operation using the PCS coordinate system is synchronized to the designated rotary tablet axis.

(3) RotaryAxis can be set to between axis 1 and axis 32 belonging to the axes group set as AxesGroup cannot be

designated.

(4) The operation parameter of the axis designated as RotaryAxis must be in mm/inch.

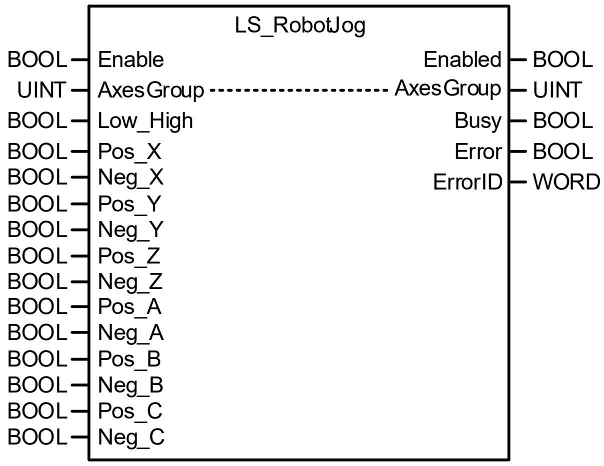
(5) Infinite running repeat must be set for the operation parameter of the axis designated as RotaryAxis

(6) Synchronized rotary table operation is terminated by performing coordinate system operation using the PCS coordinate system or performing PCS setting with MC_SetCartesianTransform function block.

(7) Refer to chapter 8.4.10 synchronized rotary table operation in motion controller's manual for more details

LS_RobotJog	Availability
JOG operation of the coordinate system	**XMC**

Motion Function Block



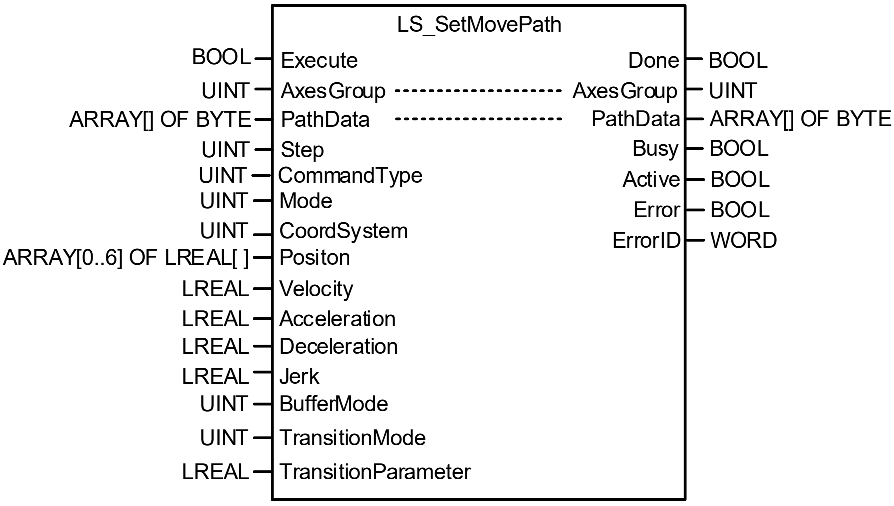
Input-Output		
UINT	AxesGroup	Set the axis group to make the command. (1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Enable	While the input is ON, the JOG operation command is sent to the relevant axis group.
BOOL	Low_High	Set the JOG speed in JOG operation. (0: JOG low-speed operation, 1: JOG high-speed operation)
BOOL	Pos_X	Set the linear operation direction at JOG operation. (X-axis + direction)
BOOL	Neg_X	Set the linear operation direction at JOG operation. (X-axis –direction)
BOOL	Pos_Y	Set the linear operation direction at JOG operation. (Y-axis + direction)
BOOL	Neg_Y	Set the linear operation direction at JOG operation. (Y-axis –direction)
BOOL	Pos_Z	Set the linear operation direction at JOG operation. (Z-axis + direction)
BOOL	Neg_Z	Set the linear operation direction at JOG operation.

		(Z-axis –direction)
BOOL	Pos_A	Set the rotary operation direction at JOG operation. (X-axis counter-clockwise rotation)
BOOL	Neg_A	Set the rotary operation direction at JOG operation. (X-axis clockwise rotation)
BOOL	Pos_B	Set the rotary operation direction at JOG operation. (Y-axis counter-clockwise rotation)
BOOL	Neg_B	Set the rotary operation direction at JOG operation. (Y-axis clockwise rotation)
BOOL	Pos_C	Set the rotary operation direction at JOG operation. (Z-axis counter-clockwise rotation)
BOOL	Neg_C	Set the rotary operation direction at JOG operation. (Z-axis clockwise rotation)
Output		
BOOL	Enabled	It indicates that the axis group is in the process of JOG operation.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block executes the JOG operation of the coordinate system for the corresponding axis group.
(2) The JOG operation is a manual operation function for testing. It is used for checking system operations, wiring status, and position address for teaching. It can be respectively applied to both high speed and low speed.
(3) If you change the value set in Low / High when the Enable input is On (JOG operation status), the speed will change without stopping JOG operation.
(4) If both the forward (Pox_) / reverse (Neg_) inputs are set for the same axis, the axis will stop.

LS_SetMovePath	Availability
Set path operation data	**XMC**

Motion Function Block



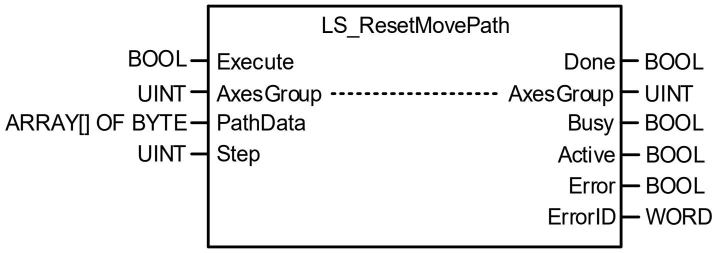
Input-Output		
UINT	AxesGroup	Set the group to set the path operation data (1 ~ 16: Group 1 ~ Group 16)
BYTE[]	PathData	Set the location where the path data is stored.
Input		
BOOL	Execute	In the rising Edge, it sends the command for setting the path operation data to the corresponding axis group.
UINT	Step	Enter the step number of the path data. (The step number is affected by the size of the data set in PathData.)
UINT	CommandType	Select the type of path operation. 0: None 1: Linear interpolation operation for the absolute position of the coordinate system, 2: Linear interpolation operation for the relative position of the coordinate system 3: Circular interpolation operation for the absolute position of the coordinate system, 4: Circular interpolation operation for the relative position of the coordinate system
UINT	Mode	Select the method and path for circular interpolation operation of the coordinate system 0/1/2: Clockwise, Midpoint/Central point/Radius 3/4/5: counter-clockwise Midpoint/Central point/Radius
UINT	CoordSystem	Select the coordinate system type.(1:MCS 2:PCS)

LREAL[]	Position	Enter the target position of the end point of the machine. In the circular interpolation, the Central point/Waypoint should be set in Position [3] Position [4] Position [5]. In the circular interpolation, the Radius should be in Position[3].
LREAL	Velocity	Specify the maximum speed of the path. [u/s]
LREAL	Acceleration	Specify the acceleration. [u/s^2]
LREAL	Deceleration	Specify the deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	Direction	Specify the operation direction. (0~4: 0-Not specified, 1-Forward direction, 2-Shortest distance, 3-Reverse direction, 4-Current direction)
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 6.1.4.BufferMode)
UINIT	TransitionMode	Unused
UREAL	TransitionParameter	Unused
Output		
BOOL	Done	Indicate that the path data setting is done successfully.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that machine information setting of the current axis is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is the function block that sets the path data for the axis group specified in the AxesGroup input.
(2) The step value can be set from 0, and the size of one step is 96 Bytes.
(3) The path data is saved in the area of data set in PathData. The variable set in PathData should be set to 96 times or more of the number of the steps to use.
(4) The CommandType value selects the operation method for the path operation. If the CommandType value is set to 0, it is considered that the data for the corresponding step is not set during path operation.
(5) The Mode value sets the direction of the circular interpolation when performing the circular interpolation operation.
(6) The value of BufferMode should be set to 1(Buffered).
(7) For more details, refer to Section 8.4.11, "Path Operation of the Coordinate System ".

LS_ResetMovePath	**Availability**
Delete path operation data	**XMC**

Motion Function Block

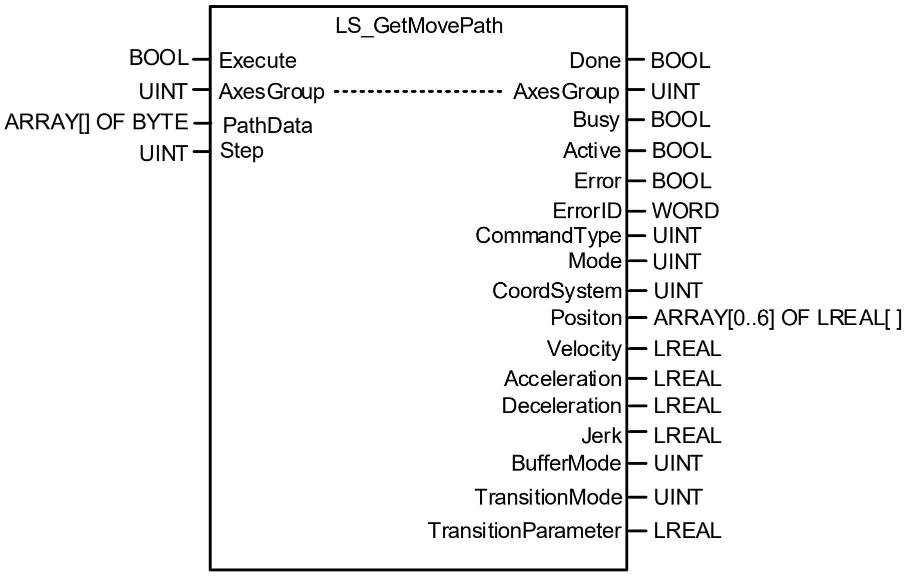


Input-Output		
UINT	AxesGroup	Set the group to set the path operation data (1 ~ 16: Group 1 ~ Group 16)
Input		
BOOL	Execute	In the rising Edge, the command for deleting the path operation data is sent to the corresponding axis group.
BYTE[]	PathData	Set the location where the path data is stored.
UINT	Step	Enter the step number of the path data. (The step number is affected by the size of the data set in PathData.)
Output		
BOOL	Done	Indicate the deleting the path data is done successfully.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that machine information setting of the current axis is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is the function block to delete the path data of the axis group specified in the AxesGroup input.
(2) The step value can be set from 0, and the size of one step is 96 Bytes.
(3) The path data is saved in the area of data set in PathData. The variable set in PathData should be set to 96 times or more of the number of the steps to use.
(4) For more details, refer to Section 8.4.11, "Path Operation of the Coordinate System ".

LS_GetMovePath	Availability
Read path operation data	**XMC**

Motion Function Block



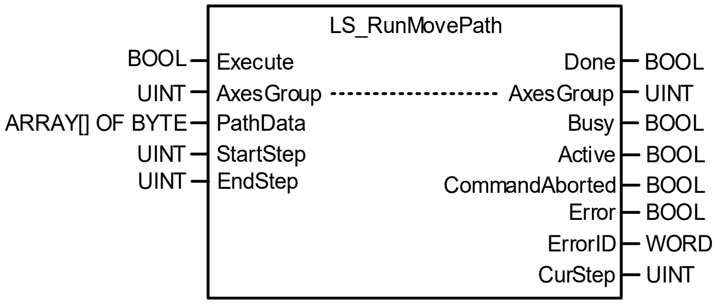
Input-Output		
UINT	AxesGroup	Set the group to set the path operation data(1 ~ 16: Group 1 ~ Group 16).
Input		
BOOL	Execute	In the rising Edge, the command for setting the path operation data is sent to the corresponding axis group.
BYTE[]	PathData	Set the location where the path data is stored.
UINT	Step	Enter the step number of the path data. (The step number is affected by the size of the data set in PathData.)
Output		
BOOL	Done	Indicate that the path data setting is done successfully.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that machine information setting of the current axis is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
UINT	CommandType	Output the type of path operation. 0: None 1: Linear interpolation operation for the absolute position of the coordinate system,

		2: Linear interpolation operation for the relative position of the coordinate system 3: Circular interpolation operation for the absolute position of the coordinate system, 4: Circular interpolation operation for the relative position of the coordinate system
UINT	Mode	Output the operation mode.
UINT	CoordSystem	Output the coordinate system type.(1:MCS 2:PCS)
LREAL[]	Position	Output the target position.
LREAL	Velocity	Output the maximum speed of the path. [u/s]
LREAL	Acceleration	Output the maximum acceleration [u/s^2]
LREAL	Deceleration	Output the maximum deceleration [u/s^2]
LREAL	Jerk	Output the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Output the sequential operation setting of motion function block. (Refer to 6.1.4.BufferMode)
UINT	TransitionMode	Unused
LREAL	TransitionParameter	Unused

(1) This motion function block is the function block to read the path data to the axis group specified in AxesGroup input.
(2) The step value can be set from 0, and the size of one step is 96 Bytes.
(3) The path data is saved in the area of data set in PathData. The variable set in PathData should be set to 96 times or more of the number of the steps to use.
(4) For more details, refer to Section 8.4.11, "Path Operation of the Coordinate System ".

LS_RunMovePath	Availability
Perform path operation	**XMC**

Motion Function Block



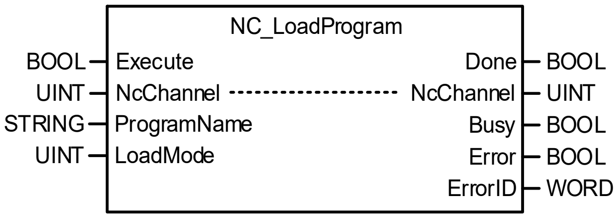
Input-Output		
UINT	AxesGroup	Set the group to execute the path operation data. (1 ~ 16 : Group 1 ~ Group 16)
Input		
BOOL	Execute	In the rising Edge, the command for setting the path operation data is sent to the corresponding axis group.
BYTE[]	PathData	Set the location where the path data is stored.
UINT	StartStep	Enter the start step number of the path data. (The step number is affected by the size of the data set in PathData.)
UINT	EndStep	Enter the end step number of the path data. (The step number is affected by the size of the data set in PathData.)
Output		
BOOL	Done	Indicate that the path data setting is completed successfully.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that machine information setting of the current axis is running.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
UINT	CurStep	Output the currently running step number.

(1) This motion function block is the function block to execute the path operation for the axis group specified in the AxesGroup input.
(2) The step value can be set from 0, and the size of one step is 96 Bytes.
(3) The path data is saved in the area of data set in PathData. The variable set in PathData should be set to 96 times or more of the number of the steps to use.

(4) The difference between StartStep and EndStep cannot be set to 100 or more. (Up to 100 step operations can be executed at one time.)
(5) If the CommandType of path data is 0 during the path operation, the operation is terminated even if EndStep is not reached.
(6) If the path operation is executed, the current step number in operation is output to the CurStep.
(7) For more details, refer to Section 8.4.11, "Path Operation of the Coordinate System ".

NC_LoadProgram	Availability
Specify NC Program	**XMC**

Motion Function Block

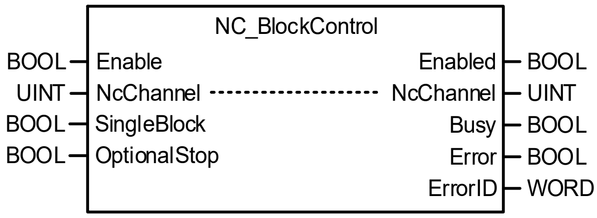


Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Execute	Set the program to be executed in the rising Edge.
STRING	ProgramName	Set the name of the program to be executed.
UINT	LoadMode	Unused (Only'0'is settable.)
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is the function block to specify the NC program to be executed when NC control is performed.
(2) When the program to be operated by the channel set in NC channel is set to ProgramName and the function block is executed, the program is designated as the one to be executed.

NC_BlockControl	Availability
Specify block opeartion	**XMC**

Motion Function Block

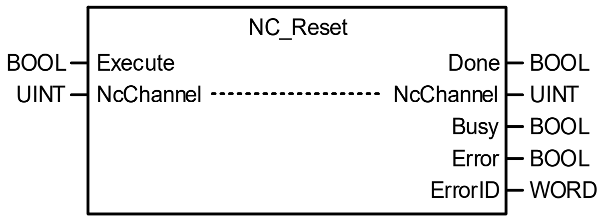


Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Enable	While the input is enabled, the corresponding channel becomes the status of Single Block or Optional Stop.
BOOL	SingleBlock	Set the Single Block operation signal.
BOOL	OptionalStop	Set the Optional Stop operation signal.
Output		
BOOL	Done	Indicate the state of Block Control completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block determines the method to execute the program under the NC control.
(2) If SingleBlock is set to '1', NC_CycleStart executes one block at a time and stops after execution. If SingleBlock becomes '1' during the automatic operation and NC_BlockControl function block is executed, it will be stopped after terminating the currently executing block.
(3) If OptionalStop is set to '1', and M01 is commanded during the program, it will wait until NC_CycleStart function block is executed again.
(4) When both SingleBlock and OptionalStop are set to '1', SingleBlock setting is applied.

NC_Reset	Availability
Reset NC opeartion	**XMC**

Motion Function Block



Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Execute	In the rising Edge, the NC is reset.
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to make the NC reset state under the NC control.
(2) If NC_Reset is executed during the automatic operation, it stops the automatic operation and changes into the reset state.
(3) The Reset state is as follows.

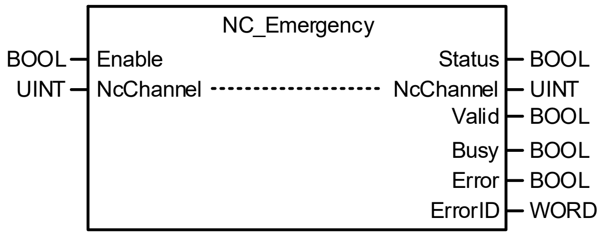
Contents		Status
Setting Data	Offset Value	
	Parameter	
Various Data	Program in Memory	
	Contents in the buffer storage	
	Display of Sequence Number	
	One shot G code	
	Modal G code	
	F	
	S, T, M	
	K (Number of repeats)	

Work coordinate value		
Action in operation	Movement	
	Dwell	
	Issuance of M, S, T code	
	Tool Length compensation	
	Cutter compensation	
	Storing called subprogram number	

항목		상태
Output Signal	CNC Alarm signal AL	Extinguish if there is no cause for the alarm
	Reference position return completion LED	
	S, T, B Code	
	M Code	
	M, S, T strobe signal	
	Spindle revolution signal(S analog signal)	
	CNC ready signal MA	
	Servo ready signal SA	ON
	Cycle Start LED	
	Feed hold LED	

NC_Emergency	Availability
Emergency stop	**XMC**

Motion Function Block

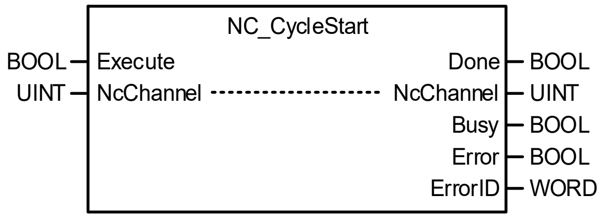


Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Enable	The emergency stop is executed while the input is '1'.
Output		
BOOL	Status	Indicate the status of the emergency stop.
BOOL	Valid	Indicate the validity of the function block output. (Same as the Status output).
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to execute the emergency stop on the corresponding NC channel under the NC control.
(2) If the emergency stop is executed, the current operation must be stopped immediately.

NC_CycleStart	**Availability**
Start automatic opeartion	**XMC**

Motion Function Block

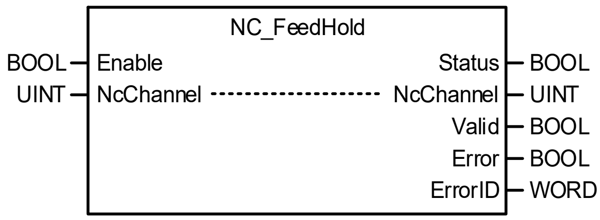


Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Execute	Start the automatic operation in the rising Edge.
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to execute the automatic operation on the corresponding NC channel under the NC control.
(2) The program set in NC_LoadProgram is automatically operated.
(3) When the automatic operation is stopped due to M00, M01(Optional Stop) and single block, the automatic operation is restarted.

NC_FeedHold	**Availability**
Feed hold	**XMC**

Motion Function Block

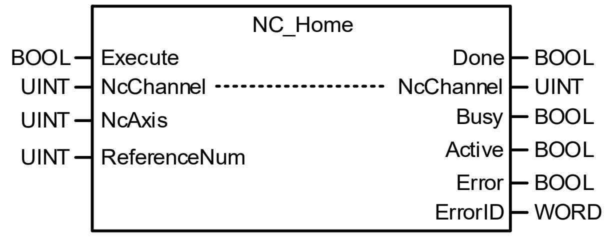


Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Enable	The NC channel will be in Feed Hold status while the input is enabled.
Output		
BOOL	Status	Indicate the Feed Hold status.
BOOL	Valid	Indicate the validity of the function block output.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block is to make the Feed Hold command to the corresponding NC channel under the NC control.
(2) If the NC_FeedHold is executed during the automatic operation, the automatic operation is stopped.
(3) If the NC_CycleStart is performed during the execution of the NC_FeedHold command, the NC_CycleStart command is ignored.

NC_Home	Availability
NC homing	XMC

Motion Function Block

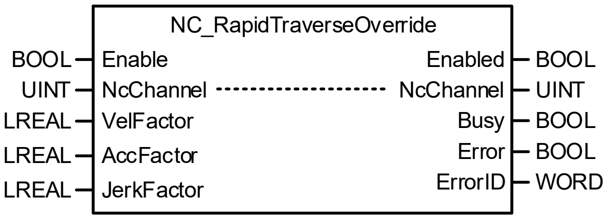


Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Execute	Start the automatic operation in the rising Edge.
UINT	NcAxis	Set the channel axis. (1~10: X=1, Y=2, … B=8, C=9, S=10)
UINT	ReferenceNum	Select the origin type. (1~4: first origin ~ fourth origin)
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current Function Block is controlling the axis.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block performs homing to the corresponding NC channel under the NC control.
(2) Homing to the 1st origin, 2nd origin, 3rd origin, and 4th origin is executed according to the values set in ReferenceNum. The origin coordinates can be set for each axis parameters of NC parameters in XG5000.

NC_RapidTraverseOverride	Availability
Rapid traverse override	**XMC**

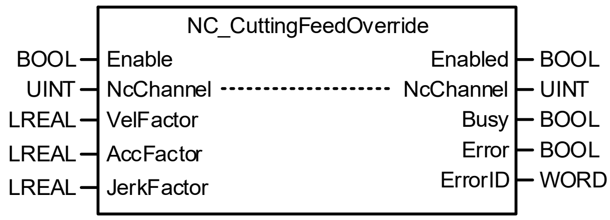
Motion Function Block



Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Enable	Execute the Rapid Traverse Override operation on the channel while the input is enabled.
LREAL	VelFactor	Specify the override rate of the speed. (0 ~ 1.0, 1.0=100%)
LREAL	AccFactor	Specify the override rate of acceleration / deceleration.
LREAL	JerkFactor	Specify the override ratio of the rate of change for acceleration.
Output		
BOOL	Enabled	Indicate that the override rate was applied successfully.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block makes the Rapid Traverse Override command for the corresponding NC channel under the NC control.
(2) Specify the speed override ratio for the VelFactor input. If the specified value is 0.0, the axis stops.
(3) The default value of each factor is 1.0, which means 100% of the command speed of the currently executing function block.
(4) Specify the acceleration / deceleration for the AccFactor input and the override rate of the jerk (rate of change of acceleration) for the JerkFactor input, respectively.
(5) Negative numbers cannot be entered into each factor.

NC_CuttingFeedOverride		**Availability**
Cutting feed override		**XMC**
Motion Function Block		

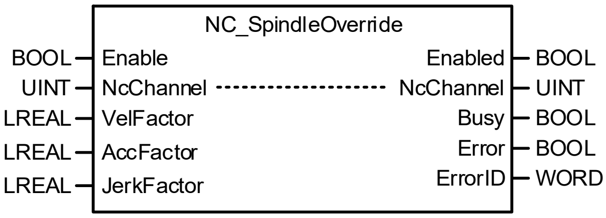


Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Enable	Execute the Cutting Feed Override operation on the channel while the input is enabled.
LREAL	VelFactor	Specify the override rate of the speed. (0 ~ 1.0, 1.0=100%)
LREAL	AccFactor	Specify the override rate of acceleration / deceleration.
LREAL	JerkFactor	Specify the override ratio of the rate of change for acceleration.
Output		
BOOL	Enabled	Indicate that the override rate was applied successfully.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block makes the Cutting Feed Override command for the corresponding NC channel under the NC control.
(2) Specify the speed override ratio for the VelFactor input. If the specified value is 0.0, the axis stops.
(3) The default value of each factor is 1.0, which means 100% of the command speed of the currently executing function block.
(4) Specify the acceleration / deceleration for the AccFactor input and the override rate of the jerk (rate of change of acceleration) for the JerkFactor input, respectively.
(5) Negative numbers cannot be entered into each factor.

NC_SpindleOverride	Availability
Spindle override	**XMC**

Motion Function Block

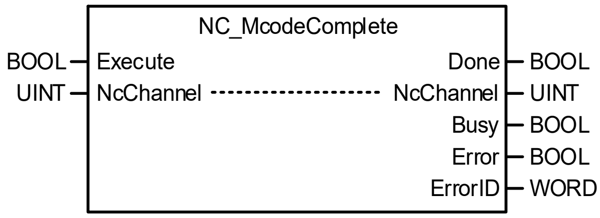


Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Enable	Execute the Spindle Override operation on the channel while the input is enabled.
LREAL	VelFactor	Specify the override rate of the speed. (0 ~ 1.0, 1.0=100%)
LREAL	AccFactor	Specify the override rate of acceleration / deceleration.
LREAL	JerkFactor	Specify the override ratio of the rate of change for acceleration.
Output		
BOOL	Enabled	Indicate that the override rate was applied successfully.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block makes the Spindle Override command for the corresponding NC channel under the NC control.
(2) Specify the speed override ratio for the VelFactor input. If the specified value is 0.0, the axis stops.
(3) The default value of each factor is 1.0, which means 100% of the command speed of the currently executing function block.
(4) Specify the acceleration / deceleration for the AccFactor input and the override rate of the jerk (rate of change of acceleration) for the JerkFactor input, respectively.
(5) Negative numbers cannot be entered into each factor.

NC_McodeComplete	Availability
M code operation completed	**XMC**

Motion Function Block



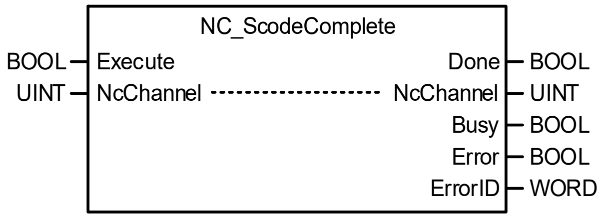
Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Execute	Set the completion of the M Code operation on the corresponding the channel.
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block makes the completion command of the M Code operation for the corresponding NC channel under the NC control.

(2) It is the command to check the M code on the corresponding channel and set that the M code operation is completed.

NC_ScodeComplete	Availability
S code operation completed	**XMC**

Motion Function Block



Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Execute	Set the completion of the S Code operation on the corresponding the channel.
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block makes the completion command of the S Code operation for the corresponding NC channel under the NC control.

(2) It is the command to check the S code on the corresponding channel and set that the S code operation is completed.

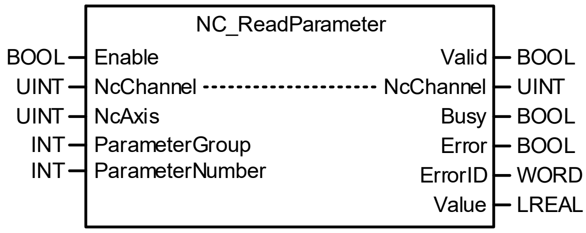
NC_TcodeComplete		Availability
T code operation completed		**XMC**
Motion Function Block		
Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Execute	Set the completion of the T Code operation on the corresponding the channel.
Output		
BOOL	Done	Indicate the state of motion function block completion.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block makes the completion command of the T Code operation for the corresponding NC channel under the NC control.

(2) It is the command to check the T code on the corresponding channel and set that the T code operation is completed.

NC_ReadParameter	Availability
Read NC parameter	**XMC**

Motion Function Block



Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Enable	The relevant parameters are output while the input is enabled.
UINT	NcAxis	Set the channel axis. (1~10: X=1, Y=2, … B=8, C=9, S=10) If it is set to 0, 'Read Channel Parameters' will be executed.
INT	ParameterGroup	Specify the group of the parameters to read.
INT	ParameterNumber	Specify the group number of the parameters to read.
Output		
BOOL	Valid	Indicate the validity of the function block output.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
LREAL	Value	Output the values of the parameters.

(1) This motion function block is to read and output the parameters of the channel and channel / axis of the corresponding channel.
(2) While the Enable input is active, the values of the relevant parameters are output continuously.
(3) ParameterGroup input specifies the parameter group number to read.
(4) ParameterNumber input specifies the number in the group of the parameters to be read.
(5) The group number and the number in the group of each parameter are as follows.

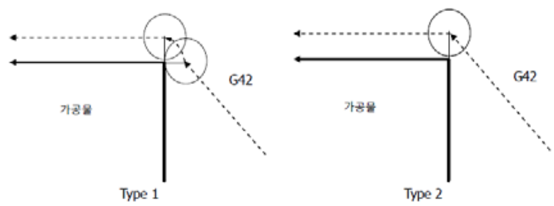
Parameters	Group	No.	Item	Description
1. Channel	1. Basic	1	Target machining quantity	Set the target machining quantity.

Parameters	Group	No.	Item	Description
parameters	setting			(0 ~ 2,147,483,647)
		2	Target machining quantity at M99 repeated machining	Set the target machining quantity for repeated machining with M99. If the set value matches the current machining quantity, the cycle automatically stops. (0 ~ 2,147,483,647)
		3	Check of decimal point	Set whether to check decimal point of the NC program. 0: Decimal point check (Mm if there is a decimal point, um if there is no decimal point) 1: No decimal point check (mm)
		4	Keep Product Coordinate System	Set whether to keep the Product Coordinate System when resetting. 0: Keep 1: Do not keep
		No.	Item	Description
		5	Whether to call the macro when the T code is commanded	Set whether to call the macro program (9000.nc ~ 9009.nc) when the T code is commanded. 0: Do not call 1: Call
		6	DWELL Method	Set the DWELL function (G04) to use the data corresponding to X, P as time or the number of revolutions of the spindle. If the data is set to the number of revolutions of the spindle, it is applied in the status of feed per revolution (G95). 0: Time 1: Number of revolutions
		7	Select a progress block at reset	Set whether to initialize to the start block of the program at reset. ※ If you want to set to 0 (keep the current block), the parameters of "Keep Product Coordinate System" should be set to 0 (keep). 0: Keep the current block 1: Initialize to the start block of the main program

Parameters	Group	No.	Item	Description
				2: Initialize to the current block of the main program
		8	Whether or not to search the Statement Number	The number of buffers that can store the program's Statement Number (N__) is limited to 1,000 in the system. This buffer is needed if the program changes the sequence using a GOTO statement. If more than 1,000 blocks have the N__ command, an alarm will occur. This parameter is used to input whether or not to execute such Statement Number search. Because high- capacity CAM programs do not have GOTO using the Statement Number and in the majority of cases, there are more than 1,000 Statement Numbers, you should set this parameter as 1. 0: Search 1: Do not search
		12	Minimum command unit	When decimal point check is applied, set the minimum unit of the commanded value. (0 ~ 0.999mm)
		18	Whether to use G22 No Travelling Area	0: 'No Travelling Area' is valid. 1: 'No Travelling Area' is invalid.
		19	Set the inner/outer side of G22 No Travelling Area	0: Inner side 1: Outer side
		20	Whether to use the 3rd 'No Travelling Area'	0: 'No Travelling Area' is valid. 1: 'No Travelling Area' is invalid.
		No.	Item	Description
		22	Rotary axis of Cylindrical interpolation	In the cylindrical interpolation mode, the axis maps the axis of rotation during the circular interpolation.The axes are X, Y, Z and perform the circular interpolation by mapping the axis of rotation to the selected axis. For example, if the axis of rotation is mapped to the X axis under the state of the XY plane (G17), the width becomes the axis of rotation and the height becomes Y axis. When ZX (G18) is selected as the plane, the width becomes the Z axis and the height becomes

Parameters	Group	No.	Item	Description
				the axis of rotation. However, if you set the plane to YZ (G19), you cannot perform the circular interpolation on the commanded axis of rotation. 0: X-axis, 1: Y-axis, 2: Z-axis
		23	Linear axis for interpolating the polar coordinate	0: Unused 1: X, 2: Y, 3: Z, 4: A, 5: B, 6: C, 7: U, 8: V, 9: W
		24	Rotary axis for interpolating the polar coordinate	0: Unused 1: X, 2: Y, 3: Z, 4: A, 5: B, 6: C, 7: U, 8: V, 9: W
		33	Monitoring time for in-position completion	0 ~ 65,535ms
		1	Regenerate the circular center when the circular alarm occurs	Set whether to recreate the central point of the arc without generating an arc alarm when the distance between the start point and the end point exceeds the tolerance of the difference between the two radii under the I, J, K circular commands. 0: An alarm occurs. 1: The central point of the arc is regenerated.
		2	Speed-limiting function for the circular milling ON/OFF	0: Unused 1: Used
		3	Tolerance of arc radius	Set the tolerance of the difference between the two radii at the start point and the end point under the circular arc command. If this value is large, the accuracy of the end part of the arc may be degraded. When set to 0, it is recognized as 0.001. (0~ 1 unit, real number)
		5	Circular radius with the speed-limiting function for the arc machining	(0 ~ 10,000 unit, real number)
		6	Upper cutting speed limit of the circular milling	The maximum speed is limited to the set value for the circular arc below "Circular radius with the speed-limiting function for the circular milling " . (0 ~ 10,000 unit/min, real number)
		7	Lower cutting speed limit of the circular milling	If “Speed-limiting function for the circular milling ON/OFF” is set to ON, the cutting speed is limited to

Parameters	Group	No.	Item	Description
				the set value or more. (0 ~ 10,000 unit/min, real number)
		9	Circular milling acceleration	Set the acceleration at the circular milling.
		10	Circular milling deceleration	Set the deceleration at the circular milling.
		11	Circular milling jerk	Set the jerk at the circular milling.
		No.	Item	Description
		1	Set the upper speed limit of the cutting feed	If the cutting speed exceeding the set value is commanded, the cutting speed is limited to the set value and an alarm occurs. (0 ~ 100,000 unit/min, real number)
		2	Set the lower speed limit of the cutting feed	It is applied only when the cutting speed is not commanded in the feed mode per minute. (0 ~ 100,000 unit/min, real number)
		4	Acceleration / deceleration method of the interpolation operation	1: Acceleration / deceleration before interpolation
		7	Operating method of the continuous blocks for acceleration / deceleration before interpolation	When executing the consecutive blocks, it creates the connecting trajectory that draws an arc on the corner of the connecting trajectory with the speed set with the next block. 1: When it is set to Buffered, the circular arc is not inserted. 1: Buffered 2: Blending Low 3: Blending Previous 4: Blending Next 5: Blending High
		9	Acceleration at the time of cutting feed (before interpolation)	Acceleration at the time of cutting feed
		10	Deceleration at the time of cutting feed (before interpolation)	Deceleration at the time of cutting feed
		11	Jerk at the time of cutting feed (before interpolation)	Jerk at the time of cutting feed

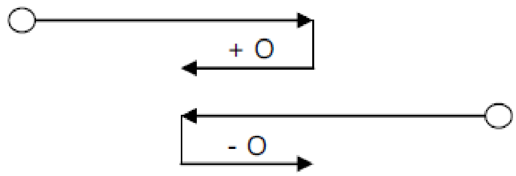
Parameters	Group	No.	Item	Description
		129	How to apply the compensation value of the tool diameter	Set the method of applying the compensation amount of the tool diameter when compensating the tool diameter. 0: Apply the diameter value 1: Apply the radius value
		130	Compensation type of the tool diameter	Tool diameter Sets the type of traversing method at the beginning and end of the calibration.  0: Type 1(Bypass traverse) 1: Type 2(Direct traverse)
		131	Whether to check the tool interference during tool diameter compensation	Set whether to check the tool interference during tool diameter compensation 0: Do not check 1: Check
		1	Compensation amount of the tool diameter 1	Compensation amount 1 to be used to compensate the tool diameter
		……	……	……
		128	Compensation amount of the tool diameter 128	Compensation amount 128 to be used to compensate the tool diameter
		1	Compensation amount 1 of the tool length	Compensation amount 1 to be used to compensate the tool length
		……	……	……
		128	Compensation amount 128 of the tool length	Compensation amount 128 to be used to compensate the tool length
		1	Whether to use the Product Coordinate System Shift amount.	Set whether to use the Product Coordinate System Shift amount. 0: Unused 1: Used
		No.	Item	Description
		11	Product Coordinate System Shift amount 1	Set the Product Coordinate System Shift amount for the X axis.
		……	……	Set the Product Coordinate System Shift amount for

Parameters	Group	No.	Item	Description
				the 7 axes; Y, Z, A, B, C, U, V.
		19	Product Coordinate System Shift amount 9	Set the Product Coordinate System Shift amount for the W axis.
		41	G54 Product Coordinate System value 1	Set the Product Coordinate System value for the X axis.
		……	……	Set the G54 Product Coordinate System values for the 7 axes; Y, Z, A, B, C, U, V.
		49	G54 Product Coordinate System value 9	Set the G54 Product Coordinate System value for the W axis.
		51	G55 Product Coordinate System value 1	Set the G55Product Coordinate System value for the X axis.
		……	……	Set the G55 Product Coordinate System values for the 7 axes; Y, Z, A, B, C, U, V.
		59	G55 Product Coordinate System value 9	Set the G55 Product Coordinate System values for the W axis.
		61	G56 Product Coordinate System value 1	Sets the G56 Product Coordinate System values for the X axis.
		……	……	Set the G56 Product Coordinate System values for the 7 axes; Y, Z, A, B, C, U, V
		69	G56 Product Coordinate System value 9	Set the G56 Product Coordinate System values for the W axis.
		71	G57 Product Coordinate System value 1	Set the G57 Product Coordinate System values for the X axis.
		……	……	Sets the G57 Product Coordinate System values for the 7 axes; Y, Z, A, B, C, U, V
		79	G57 Product Coordinate System value 9	Set the G57 Product Coordinate System values for the W axis.
		81	G58 Product Coordinate System value 1	Set the G58 Product Coordinate System values for the X axis.
		……	……	Set the G58 Product Coordinate System values for the 7 axes; Y, Z, A, B, C, U, V
		89	G58 Product Coordinate System value 9	Set the G58 Product Coordinate System values for the W axis.
		91	G59 Product Coordinate System value 1	Set the G59 Product Coordinate System values for the X axis.
		……	……	Set the G59 Product Coordinate System values for

Parameters	Group	No.	Item	Description
				the 7 axes; Y, Z, A, B, C, U, V
		99	G59 Product Coordinate System value 9	Set the G59 Product Coordinate System values for the W axis.
		1	Whether to apply the single block stop function to the macro program	Set whether to apply the single block stop function to the macro program(9000.nc ~ 9999.nc) 0: Stop 1: Do not stop
		2	Display the macro program block	Set whether to display the progress status of the block on the screen when operating the macro program (9000.nc ~ 9999.nc). 0: Do not display 1: Display
		10	Macro program call G code (9010.nc)	Set the G code number to call the macro program (9010.nc ~ 9019.nc) that can be called by the G code. ※ The setting values 0, 1, 2, 3 are ignored. (0~255.9, real number)
		……	……	
		19	Macro program call G code (9019.nc)	Set the G code number to call the macro program (9010.nc ~ 9019.nc) that can be called by the G code. ※ The setting values 0, 1, 2, 3 are ignored. (0~255.9, real number)
		No.	Item	Description
		20	Macro program call M code (9020.nc)	Assign the M code number to call the macro program (9020.nc ~ (9020.nc ~ 9029.nc) with the M code. ※ 0, 30 of the input values are ignored. (0~255, integer)
		……	……	
		29	Macro program call M code (9029.nc)	Assign the M code number to call the macro program (9020.nc ~ (9020.nc ~ 9029.nc) with the M code. ※ 0, 30 of the input values are ignored.

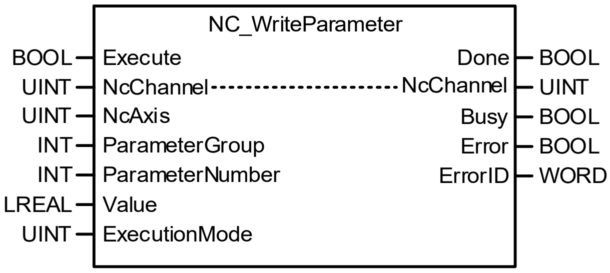
Parameters	Group	No.	Item	Description
				(0~255, integer)
		9	T code call Macro program number	Enter the number of the macro program (9000.nc ~ 9009.nc) to be called when the T code is commanded. (9000 ~ 9009, integer)
		1	Modal traverse of default settings	If there is no G00 or G01, select the G code to be applied as the default modal. 0: Rapid Traverse(G00) 1: cutting feed(G01)
		2	Modal plane of default settings	If there is no G code instruction for G17, G18, G19 group, select the G code to be applied as the default modal. 0: XY plane(G17) 1: XZ plane(G18) 2: YZ plane(G19)
		3	Modal absolute / increment with default settings	If there is no G code instruction for G90, G91 group, select the G code to be applied as the default modal. 0: Absolute command (G90) 1: Incremental command (G91)
		5	Check the modal prohibited area with default settings	If there is no G code instruction for G22, G23 group, select the G code to be applied as the default modal. 0: Stroke On(G22) 1: Stroke Off(G23)
		1	Relative coordinate's offset value #1	Set the relative coordinate's offset value for the X axis.
		2	Relative coordinate's offset value #2	Set the relative coordinate's offset value for the Y axis.
		3	Relative coordinate's offset value #3	Set the relative coordinate's offset value for the Z axis.
		4	Relative coordinate's offset value #4	Set the relative coordinate's offset value for the A axis.
		5	Relative coordinate's offset value #5	Set the relative coordinate's offset value for the B axis.
		6	Relative coordinate's offset value #6	Set the relative coordinate's offset value for the C axis.
		7	Relative coordinate's offset	Set the relative coordinate's offset value for the U

Parameters	Group	No.	Item	Description
			value #7	axis.
		8	Relative coordinate's offset value #8	Set the relative coordinate's offset value for the V axis.
		9	Relative coordinate's offset value #9	Set the relative coordinate's offset value for the W axis.
		2	Setting the direction for the modular axis	Set the traverse command for the axis set as the modular axis. 0: One-way 1: Two-way
		No.	Item	Description
		1	Coordinates of the 2nd origin	Set the coordinates of the 2nd origin.
		2	Coordinates of the 3rd origin	Set the coordinates of the 3rd origin.
		3	Coordinates of the 4th origin	Set the coordinates of the 4th origin.
		2	Rapid traverse acceleration	The set value is used as the acceleration of the G00 block.
		3	Rapid traverse deceleration	The set value is used as the de**celeration** of the G00 block.
		4	Rapid traverse jerk	The set value is used as the jerk of the G00 block.
		5	Rapid traverse speed	The set value is used as the traverse speed of the G00 block. (0~100000 unit/min, real number)
		1	Minimum value of the G22 Traverse-Prohibited Area range for the X, Y, and Z axis.	Set the minimum value of the G22 Traverse-Prohibited Area range for the X, Y, and Z axis. (-100,000~100,000 unit, real number)
		2	Maximum value of the G22 Traverse-Prohibited Area range for the X, Y, and Z axis.	Set the maximum value of the G22 Traverse-Prohibited Area range for the X, Y, and Z axis. (-100,000~100,000 unit, real number)
		3	Minimum value of the 3rd Traverse-Prohibited Area range for the X, Y, and Z axis.	Set the minimum value of the 3rd Traverse-Prohibited Area range for the X, Y, and Z axis. (-100,000~100,000 unit, real number)

Parameters	Group	No.	Item	Description
		4	Maximum value of the 3rd Traverse-Prohibited Area range for the X, Y, and Z axis.	Set the maximum value of the 3rd Traverse-Prohibited Area range for the X, Y, and Z axis. (-100,000~100,000 unit, real number)
		2	Overrun feed rare of single direction positioning	Set the overrun feed rate of the 9 axes; X, Y, Z, A, B, C, U, V, W when using the single direction positioning function (G60). After stopping at the position separated by the set value for the G60 command block's axis, it moves to the command position to eliminate the effect of backlash.  (-100 ~ 100 unit, real number)

NC_WriteParameter	Availability
Read NC parameter	**XMC**

Motion Function Block



Input-Output		
UINT	NC channel	Set the NC channel to make the command.
Input		
BOOL	Execute	The NC parameter is written in the rising Edge of the input.
UINT	NcAxis	Set the channel axis. (1~10: X=1, Y=2, … B=8, C=9, S=10) When set to 0, 'Write Channel Parameters' is executed.
INT	ParameterGroup	Specify the group of the parameter to be written.
INT	ParameterNumber	Specify the number in the group of the parameter to be written.
LREAL	Value	Specify the value of the parameter to be written.
UNIT	ExecutionMode	Reserved
Output		
BOOL	Valid	Indicate the validity of the function block output.
BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.
LREAL	Value	Output the values of the parameters.

(1) This motion function block is the function block that writes the values specified in the parameters of the NC channel and channels/axes.

(2) The parameters will be written in the rising edge of the Execute input.

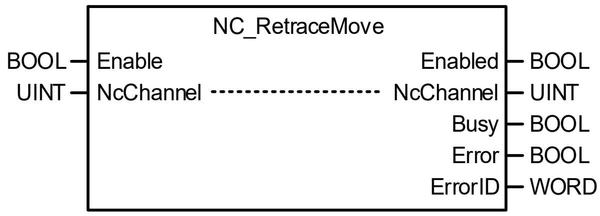
(3) ParameterGroup input specifies the group number of the parameter to be written.

(4) ParameterNumber input specifies the number in the group of the parameter to be written. If the value that cannot be set is applied, "Error 16 # 000B" occurs.

(5) In the Value input, specify the value to be written in the parameter.

(6) For the group number and the number in the group of each parameter, refer to NC_ReadParameter.

NC_RetraceMove		Applied model
Reverse operation		**XMC**
Motion function block type		



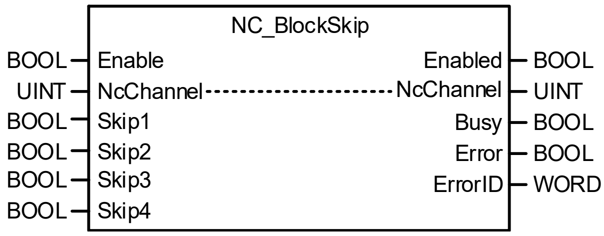
Input-Output		
UINT	NcChannel	Specify the NC channel to set the command (1 to 4: 1 to 4 channels)
input		
BOOL	Enable	Reverse operation command is executed on the channel while the input is active.
Print		
BOOL	Enabled	Indicates that the function block has been successfully applied.
BOOL	Busy	Indicates that function block execution is not completed.
BOOL	Error	Indicates whether an error occurred.
WORD	ErrorID	The error number generated during function block output is output.

(1) This motion function block is a function block that gives reverse run command in corresponding NC channel.
(2) Enable Runs the operation in the opposite direction while the input is active.
(3) Reverse operation is possible only for G00, G01, G02, G03 blocks.
(4) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.30	V4.28

NC_BlockSkip	Applied model
Block skip	**XMC**

Motion function block type



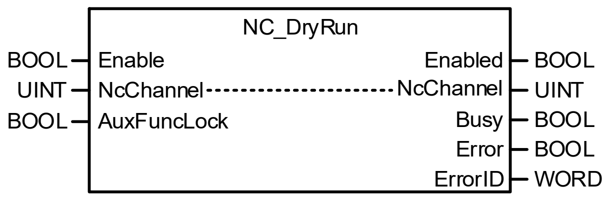
Input-Output		
UINT	NcChannel	Specify the NC channel to set the command (1 to 4: 1 to 4 channels)
input		
BOOL	Enable	Executes the specified block skip operation of the channel while the input is active.
BOOL	Skip1	Specify the block skip 1. (G31 / G31.1 / G37 / G37.1)
BOOL	Skip2	Specifies block skip 2. (G31.2 / G37.2)
BOOL	Skip3	Specify block skip 3. (G31.3 / G37.3)
BOOL	Skip4	Specify block skip 4. (G31.4 / G37.4)
Print		
BOOL	Enabled	Indicates that the function block has been successfully applied.
BOOL	Busy	Indicates that function block execution is not completed.
BOOL	Error	Indicates whether an error occurred.
WORD	ErrorID	The error number generated during function block output is output.

(1) This motion function block is a function block that issues a block skip or automatic tool length measurement command in the corresponding NC channel.
(2) Skip Skip (G31 / G31.1), Skip2 (G31.2), Skip3 (G31.3) and Skip4 (G31.4) blocks while the Enable input is active.
(3) If there is a G31 / G31.1 (Skip1), G31.2 (Skip2), G31.3 (Skip3) or G31.4 (Skip4) instruction at the time of enabling the Enable input, If there is an M / S / T code, the next block is executed after the corresponding code is executed.
(4) If there is a G37 / G37.1 (Skip1), G37.2 (Skip2), G37.3 (Skip3) or G37.4 (Skip4) instruction at the time of Enable input activation, the automatic tool length measurement operation .
(5) When the function block is executed, the current machine position is stored in each NC channel / axis flag and the skipped position can be known.
(6) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.30	V4.28

NC_DryRun	Applied model
Dry run	**XMC**

Motion function block type



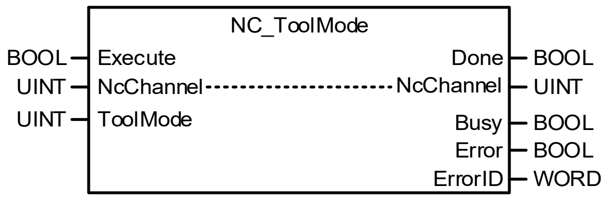
Input-Output		
UINT	NcChannel	Specify the NC channel to set the command (1 to 4: 1 to 4 channels)
input		
BOOL	Enable	The corresponding parameter is output while the input is active.
BOOL	AuxFuncLock	When the input is activated, the auxiliary function code (M / S / T) is ignored.
Print		
BOOL	Enabled	Indicates that the function block is being executed.
BOOL	Busy	Indicates that function block execution is not completed.
BOOL	Error	Indicates whether an error occurred.
WORD	ErrorID	The error number generated during function block output is output.

(1) This motion function block is a function block that performs the dry run operation in the corresponding NC channel.
(2) Perform the dry run operation while the Enable input is active.
(3) During dry run operation, according to the parameter set in G00, 0: Dry run speed operation, 1: Rapid traverse speed operation.
(4) When the AuxFuncLock input is activated, the strobe signal of the auxiliary function code (M / S / T) except for M00, M01, M02, M30, M98 and M99 is not output.
(5) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.30	V4.28

NC_ToolMode	Applied model
Tool Escape / Return Operation	**XMC**

Motion function block type

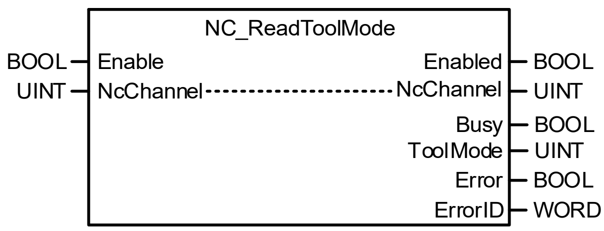


Input-Output		
UINT	NcChannel	Specify the NC channel to set the command (1 to 4: 1 to 4 channels)
input		
BOOL	Execute	A tool escape or return operation command is issued to the rising edge of the input.
UINT	ToolMode	Reduce the tool run (1) or return (2) run command.
Print		
BOOL	Done	Indicates that the function block has been successfully applied.
BOOL	Busy	Indicates that function block execution is not completed.
BOOL	Error	Indicates whether an error occurred.
WORD	ErrorID	The error number generated during function block output is output.

(1) This motion function block is a function block that issues a tool escape or tool return operation command to the corresponding NC channel.
(2) Execute A tool exit or a return run command is issued to the ToolMode at the rising edge of the input.
(3) Jog operation is required for escape operation during tool escape operation, and the position is memorized at the point when the operation axis is changed during escape operation by jog operation. Up to 10 positions are memorized.
(4) Jog operation must be created so that two axes or more are not selected at the same time during tool escape operation.
(5) When returning to the tool, it returns to the point memorized.
(6) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.30	V4.28

NC_ReadToolMode		Applied model
Read tool escape / return mode		**XMC**
Motion function block type		



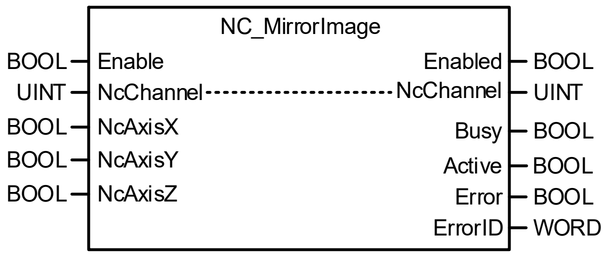
Input-Output		
UINT	NcChannel	Specify the NC channel to set the command (1 to 4: 1 to 4 channels)
input		
BOOL	Enable	Check the state of tool escape / return while input is active.
Print		
BOOL	Enabled	Indicates that the function block is being executed.
BOOL	Busy	Indicates that function block execution is not completed.
BOOL	Error	Indicates whether an error occurred.
WORD	ErrorID	The error number generated during function block output is output.
UINT	ToolMode	Indicates whether the tool is being moved (1) or returning (2).

(1) This motion function block is a function block that issues a command to check the state of tool escape / return in the corresponding NC channel.
(2) While the Enable input is active, the ToolMode output shows the status of tool escape (1) or tool return (2).
(3) During the tool escape, make sure that no more than two axes are running.
(4) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.30	V4.28

NC_MirrorImage		Applied model
Mirror image		**XMC**

Motion function block type



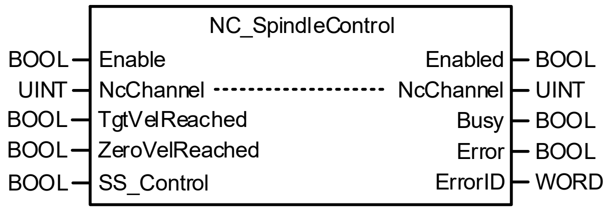
Input-Output		
UINT	NcChannel	Specify the NC channel to set the command (1 to 4: 1 to 4 channels)
input		
BOOL	Enable	Executes an inverse operation on the specified axis of the channel while the input is active.
BOOL	NcAxisX	Give reverse operation signal to X axis.
BOOL	NcAxisY	It gives reverse operation signal to Y axis.
BOOL	NcAxisZ	It gives reverse operation signal to Z axis.
Print		
BOOL	Enabled	Indicates that the function block is being executed.
BOOL	Busy	Indicates that function block execution is not completed.
BOOL	Error	Indicates whether an error occurred.
WORD	ErrorID	The error number generated during function block output is output.

(1) This motion function block is a function block that performs the operation to reverse the feed position on the NC axis (X, Y, Z) of the corresponding NC channel.
(2) While the Enable input is active, the traversing position of the set axis is reversed and the operation is performed.
(3) Inverted operation is performed only for G00, G01, G02, G03, G31.x, G37.x among the specified G code.
(4) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.30	V4.28

NC_SpindleControl		Applied model
Spindle operation control		**XMC**

Motion function block type



Input-Output		
UINT	NcChannel	Specify the NC channel to set the command (1 to 4: 1 to 4 channels)
input		
BOOL	Enable	While the input is active, it is assigned to the main spindle of that channel Perform the action.
BOOL	TgtVelReached	The target spindle speed is transmitted to the NC function module. 0: Target Speed Not Reached 1: Target speed is reached
BOOL	ZeroVelReached	Transfers to the NC function module whether or not the main spindle zero speed has been reached. 0: Zero speed does not reach 1: Zero speed reached
BOOL	SS_Control	Start (end) SS control mode of the main spindle. (Future support) 0: Start SS control 1: SS control end
Print		
BOOL	Enabled	Indicates that the function block is being executed.
BOOL	Busy	Indicates that function block execution is not completed.
BOOL	Error	Indicates whether an error occurred.
WORD	ErrorID	The error number generated during function block output is output.

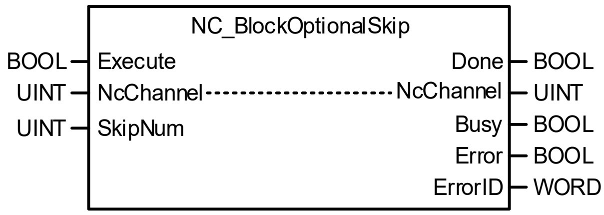
(1) This motion function block performs the action specified by the user for the main spindle of the NC channel specified by the function block that operates when the spindle control is executed in NC.
(2) If the spindle axis of the channel is not activated automatically in the NC function module, '0x36D0' error occurs.
(3) If the axis specified as the main spindle of the channel is not ready for operation, a '0x36D1' error will occur.
(4) For details on automatic operation in the NC function module, refer to '9.5.1 How to operate the spindle axis'.

(5) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.30	V4.28

NC_BlockOptionalSkip	Applied model
NC Optional block skip	**XMC**

Motion function block type



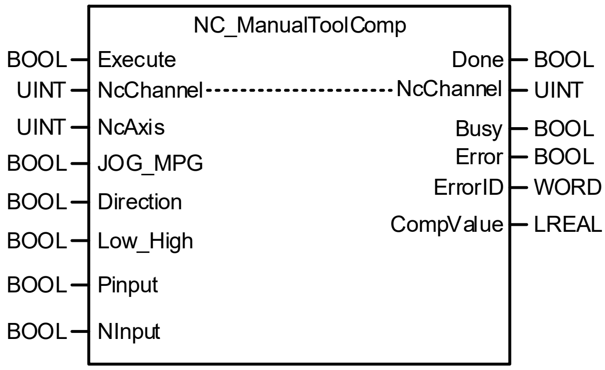
Input-Output		
UINT	NcChannel	Specify the NC channel to set the command (1 to 4: 1 to 4 channels)
input		
BOOL	Execute	Give SkipNum a skip signal at the rising edge of the input.
UINT	SkipNum	Signal designation for block skip (1 to 9) (1: / or / 1, 2: / 2, 3: / 3, 4/4, 5/5, 6/6, 7/7, If set to 0, skip function is canceled.
Print		
BOOL	Enabled	Indicates that the function block is being executed.
BOOL	Busy	Indicates that function block execution is not completed.
BOOL	Error	Indicates whether an error occurred.
WORD	ErrorID	The error number generated during function block output is output.

(1) This motion function block is a function block that outputs an optional skip instruction to the NC channel.
(2) Skip the block with "/ n" in front of the NC program block according to the SkipNum input value at the rising edge of the Execute input. For example, if SkipNum is 3, skip blocks with / 3 before the block. At this time, the current block is skipped and the next block is executed. If there is an M / S / T code, the next block is executed after the corresponding code is executed.
(3) When SkipNum is set to 0 and the command is executed, the skip function is disabled.
(4) If a value other than 0 to 9 is set to SkipNum, a "0x36A0" error will occur.
(5) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.30	V4.28

NC_ManualToolComp	Applied model
Manual measurement of the NC correction amount	**XMC**

Motion function block type



Input-Output		
UINT	NcChannel	Specify the NC channel to set the command (1 to 4: 1 to 4 channels)
UINT	NcAxis	Sets the channel axis. (1 to 3: X = 1, Y = 2, Z = 3)
input		
BOOL	Execute	Decrease the command for inputting the correction amount input mode to the rising edge of the input.
BOOL	JOG_MPG	Operation method selection (0: JOG, 1: MPG)
BOOL	Direction	Jog operation direction (0: Forward, 1: Reverse direction)
BOOL	Low_High	Jog operation speed (0: Low speed, 1: high speed)
BOOL	PInput	Forward measurement input signal
BOOL	NInput-	Reverse measurement input signal
Print		
BOOL	Done	Indicates that the function block has been successfully applied.
BOOL	Busy	Indicates that function block execution is not completed.
BOOL	Error	Indicates whether an error occurred.
WORD	ErrorID	The error number generated during function block output is output.
LREAL	CompValue	The calculated correction amount is output.

(1) This motion function block is a function block that outputs a manual tool compensation amount measurement command to the axis set in NcAxis of the corresponding NC channel
(2) Execute manual tool compensation amount measurement run command at the rising edge of the input.
(3) When the command is executed, the operation selected in JOG_MPG starts. When the signal selected in PInput

or NInput becomes 1, operation is stopped and the compensation value is calculated by using the value of the corresponding position.

(4) The correction amount is calculated by the formula below.

Amount of correction = PInput / NInput On axis position - Measurement reference position

(5) The measurement reference position is selected from the channel parameter "measuring reference distance X of automatic tool offset" - "measuring reference distance Z of automatic tool offset" according to the axis. For example, if NcAxis is selected as Y and NInput is On, the value set for "Measuring distance Y of automatic tool offset" becomes the measurement reference position.

(6) The calculated compensation amount is output to CompValue and Done becomes 1.

(7) If both PInput and NInput are on at the same time, they are recognized as PInput.

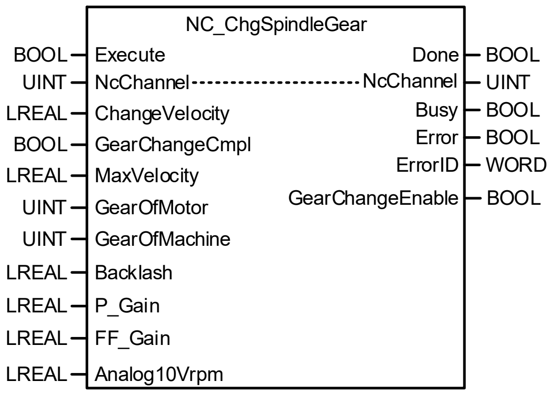
(8) If an axis other than X to Z is set in NcAxis and a function block is executed, "0x36B0" error will occur.

(9) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.30	V4.28

# NC_ChgSpindleGear	**Applied model**
NC spindle gear conversion	**XMC**

Motion function block type



Input-Output		
UINT	NcChannel	Specify the NC channel to set the command (1 to 4: 1 to 4 channels)
input		
BOOL	Execute	Give the spindle gear conversion command to the rising edge of the input.
LREAL	ChangeVelocity	Set the speed value to change
BOOL	GearChangeCmpl	A signal that the gear change is complete. On, the set values of each operand are set to the corresponding parameters.
LREAL	MaxVelocity	Maximum speed parameter setting value
UINT	GearOfMotor	Motor side gear ratio parameter
UINT	GearOfMachine	Machine side gear ratio parameter
LREAL	Backlash	Backlash value
LREAL	P_Gain	P gain setting value
LREAL	FF_Gain	Feed Forward gain setting value
LREAL	Analog 10Vrpm	Unapplied
Print		
BOOL	Done	Indicates that the function block has been successfully applied.
BOOL	Busy	Indicates that function block execution is not completed.
BOOL	Error	Indicates whether an error occurred.
WORD	ErrorID	The error number generated during function block output is output.
BOOL	GearChangeEnable	Indicates whether gear change is possible.

(1) This motion function block is a function block that issues a spindle gear change command to the corresponding NC

channel.
(2) Execute Changes the spindle gear change command at the rising edge of the input.
(3) When the command is executed, change the current spindle speed to the value set in ChangeVelocity, which enables gear conversion.
(4) If the spindle axis speed is changed to less than the value set in ChangeVelocity and the GearChangeEnable output is On, the user operates the sequence program to perform gear conversion and inputs On to GearChangeCmpl when gear conversion is completed.
(5) If GearChangeCmpl is On, set the values of the following items set in the function block as parameters and operate the spindle with the changed settings.

Speed limit (MaxVelocity)

Motor side gear ratio (GearOfMotor)

Machine gear ratio (GearOfMachine)

Backlash correction amount (Backlash)

Position mode P gain (P_Gain)

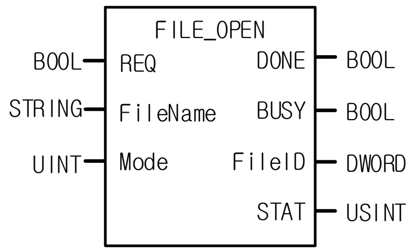
Position Mode Feed Forward Gain (FF_Gain)

(6) If you set the value of ChageVelocity to a value larger than the speed limit of the axis and execute the function block, "0x36C0" error occurs.
(7) When the value of MaxVelocity is set to a value less than 0 and the function block is executed, "0x36C1" error occurs.
(8) If you set the value of GearOfMotor to a value less than 0 or a value larger than 65535 and execute the function block, "0x36C2" error occurs.
(9) If you set the value of GearOfMachine to a value less than 0 or a value larger than 65535 and execute the function block, "0x36C3" error occurs.
(10) If you set the backlash value to a value less than 0 and execute the function block, "0x36C4" error occurs.
(11) If you set the value of P_Gain to a value less than 0 or a value larger than 500 and execute the function block, "0x36C5" error occurs.
(12) If you set the value of FF_Gain to a value less than 0 or a value larger than 100 and execute the function block, "0x36C6" error occurs.
(13) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.30	V4.28

FILE_OPEN	Applied model
Open file in SD memory card	**XMC**

Motion function block type



input		
BOOL	REQ	Set the program to run on the rising edge.
STRING	FileName	Set the file name to be specified.
UINT	Mode	File open mode.
Print		
BOOL	Done	Indicates completion of function block completion.
BOOL	Busy	Indicates that function block execution is not completed.
DWORD	FileID	The ID of the file that was opened.
USINT	STAT	The error number generated during function block output is output.

(1) This motion function block is a function block that issues a spindle gear change command to the corresponding NC channel.
(2) When executing Open, motion is classified according to Mode setting value.

Mode	action
0	Open the file for read and write. If a file does not exist, it is created as a new one. If a file with the same name exists, the contents of the file are deleted and a new one is created from scratch.
1	Open the file for reading and writing. If the file does not exist, it is newly created. If there is a file with the same name, the write operation is continuously performed from the end of the file.
2	Open the file as read-only.

(3) It reads from the beginning of the file at FILE_READ after FILE_OPEN. However, when FILE_READ is executed after FILE_WRITE, it is read from the end of file. Therefore, it should be moved to FILE_SEEK and read must be performed.
(4) File Open The ID of the opened file is displayed as 'FileID' when it is executed normally.
(5) 'FileID' is used when executing FILE_WRITE, FILE_READ, FILE_SEEK, and FILE_CLOSE commands.
(6) STAT = 0 when executing FILE_OPEN normally, and STAT information when error occurs in other cases.
(7) The maximum number of FILE_OPEN is 50. (Including data log file)

STAT	Error state
0	normal
1	SD memory card access failed
2	File is already open
3	Mode 2 and there is no file in the Inst folder, If the SD card is not installed
4	More than 50 files open
5	If Mode is a value other than 0 to 2

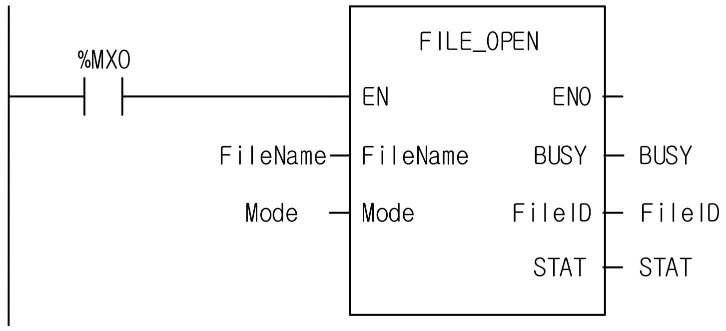
(8) FILE_OPEN One file must be FILE_CLOSE command after use to close the file.

(9) Even if the PLC mode is changed, the file is still open, so FILE_OPEN must be performed after closing the file.

■ **Example Program**

(1) LD

- FileName = 'ABC', Mode = 0



(a) If execution condition (% MX0) is On, FILE_OPEN function will be executed.

(b) If the SD card is properly inserted, open the file that can be read and written with the file name FileName = 'ABC'. If ABC file with the same name exists, it deletes the file contents and opens from scratch.

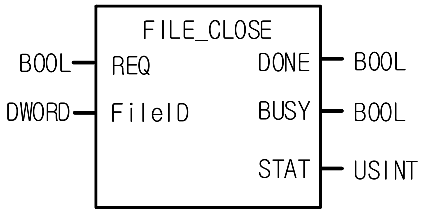
(c) Depending on the status of the SD card or the status of the file, an error is displayed in STAT. In normal operation, 0 is output.

(2) ST

INST_FILE_OPEN (REQ: =% MX0, FileName: = 'ABC', Mode: = 0, DONE => DONE, BUSY => BUSY, FileID => FileID, stat => stat);

FILE_CLOSE	Applied model
Close files in SD memory card	**XMC**

Motion function block type



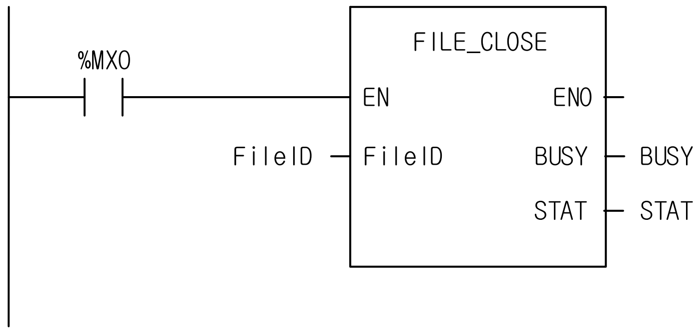
input		
BOOL	REQ	Set the program to run on the rising edge.
DWORD	FileID	The ID of the file that was opened.
Print		
BOOL	Done	Indicates completion of function block completion.
BOOL	Busy	Indicates that function block execution is not completed.
USINT	STAT	The error number generated during function block output is output.

(1) Close the file specified as 'FileID' on the SD memory card
(2) STAT = 0 when executing FILE_CLOSE normally, and STAT information when error occurs

STAT	Error state
0	normal
1	SD memory card access failed
2	If you do not have any open files

■ **Example Program**

(1) LD



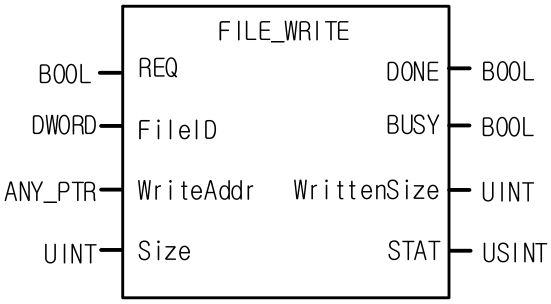
(a) FILE_OPEN After this is successfully done, you must enter the output value FileID.
(b) When execution condition (% MX0) is On, FILE_CLOSE function is executed.
(c) Depending on the status of the SD card or the status of the file, an error is displayed in STAT. In normal operation, 0 is output.

(2) ST

INST_FILE_CLOSE (REQ: =% MX0, FileID: = FileID, DONE => DONE, BUSY => BUSY, stat => stat);

FILE_WRITE	Applied model
Write files to SD memory card	**XMC**

Motion function block type



input		
BOOL	REQ	Set the program to run on the rising edge.
DWORD	FileID	The ID of the file that was opened.
ANY_PTR	WriteAddr	The address of the data to be written.
UINT	Size	The number of data to write.
Print		
BOOL	Done	Indicates completion of function block completion.
BOOL	Busy	Indicates that function block execution is not completed.
UINT	WrittenSize	The number of data that has been written.
USINT	STAT	The error number generated during function block output is output.

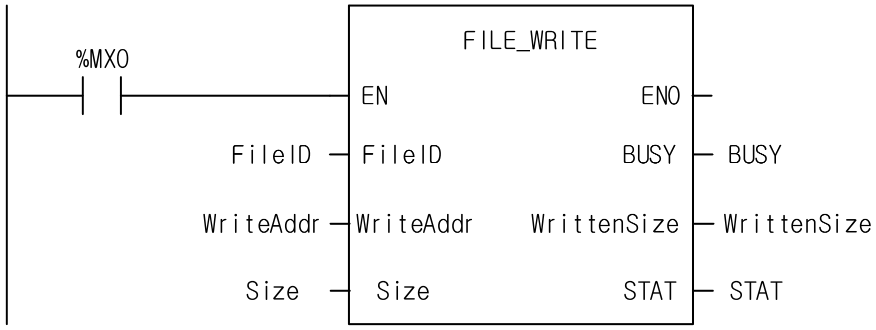
(1) Write to a file opened with 'FileID' on the SD memory card.
(2) The write data is the contents of WriteAddr, and write is performed for the number of size.
(3) When WriteAddr is declared as an Array type, data in the array is written by the size to be written.
(4) Write data size is Array type, WriteAddr data type x Size. (In case of Byte, data type is 1)
(5) When WriteAddr is declared as a data type, only the corresponding data value is written regardless of the value of Size.
(6) BUSY = 1 when writing, BUSY = 0 when completed and DONE = 1.
(7) Normally, the data size that is actually written when FILE_WRITE is executed is output to WrittenSize.
(8) STAT information is STAT = 0 at normal completion, and STAT information at the time of error occurrence is as follows.
(9) FILE_CLOSE Data is not saved normally when you remove the SD card previously.

STAT	Error state

0	normal
1	SD memory card access failed
2	FileID is not open
3	The file is opened as read-only.
4	If the size is 0 (for the Array type) and the size is 65535 or larger

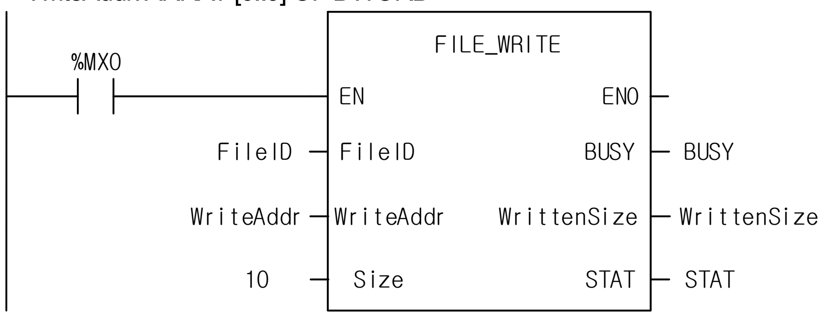
■ **Example Program**

(1) LD



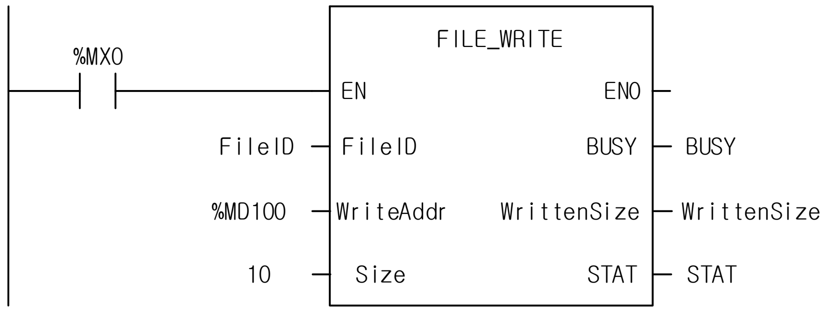
(a) FILE_OPEN After this is successfully done, you must enter the output value FileID.
(b) When execution condition (% MX0) is On, FILE_WRITE function is executed.
(c) WriteAddr can be set to array type or data type.
(d) When set as an array type, data can be written to the SD card within the array range. For example, if 10 DWORD arrays are set, 10 array values can be written from [0] to [9] using Size.
(e) When set to data type, only the corresponding data value is written to the SD card, and the size value is meaningless.
(f) In normal operation, WrittenSize shows the actual size of the data written.
(g) Depending on the status of the SD card or the status of the file, an error is displayed in STAT. In normal operation, 0 is output.

※ **WriteAddr array type example**
- WriteAddr: ARRAY [0..9] OF DWORD



(a) When execution condition (% MX0) is On, FILE_WRITE function is executed.
(b) Since WriteAddr is an array type and Size is 10, WriteAddr [0] to [9] data write operations.
(c) After writing 10 DWORD data, WrittenSize is displayed as 40 and STAT is output as 0 after writing.

※ **WriteAddr data type example**
- WriteAddr:% MD100



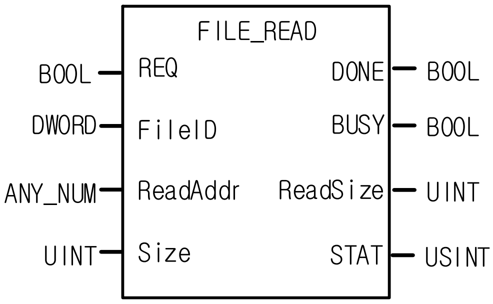
(a) When execution condition (% MX0) is On, FILE_WRITE function is executed.
(b) Since the size is 10 or WriteAddr is the data type, only the set% MD100 value will be written.
(c) Since it is DWORD data, WrittenSize is 4 and STAT is 0 after writing is completed.

(2) ST

INST_FILE_WRITE (REQ: =% MX0, FileID: = FileID, WriteAddr: = WriteAddr, Size: = Size, DONE => DONE, BUSY => BUSY, WrittenSize => WrittenSize, stat => stat)

FILE_READ	Applied model
Reading files in SD memory card	XMC

Motion function block type



input		
BOOL	REQ	Set the program to run on the rising edge.
DWORD	FileID	The ID of the file that was opened.
ANY_NUM	ReadAddr	The starting address of the data to read.
UINT	Size	The number of data to read.
Print		
BOOL	Done	Indicates completion of function block completion.
BOOL	Busy	Indicates that function block execution is not completed.
UINT	ReadSize	The number of data read completed.
USINT	STAT	The error number generated during function block output is output.

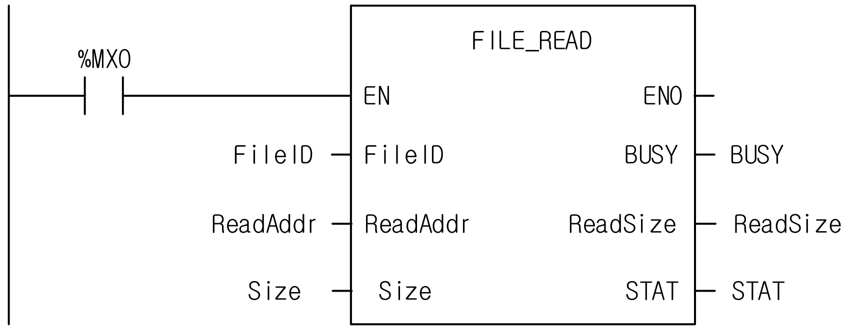
(1) Read from file opened with 'FileID' on SD memory card.
(2) Read after FILE_OPEN is read from the beginning of the file. When FILE_WRITE is executed, file pointer is read from the last position.
(3) If you need to move the location, you must move it with FILE_SEEK command.
(4) The read data is stored in ReadAddr and is read as many as the size.
(5) If ReadAddr is declared as an Array type, it will be read as array by the size to be read.
(6) Read data size is Array type, ReadAddr data type x Size. (In case of Byte, data type is 1)
(7) When ReadAddr is declared as a data type, it is read only by the data type size regardless of the value of Size.
(8) BUSY = 1 when reading, BUSY = 0 when completed and DONE = 1.
(9) When FILE_READ is executed normally, the data size that is actually read is output to ReadSize.
(10) STAT information is STAT = 0 at normal completion, and STAT information at the time of error occurrence is as follows.

STAT	Error state
0	normal
1	SD memory card access failed
2	FileID is not open
3	If Size is 0 (Array type) or if there is no actual data to read

(11) Even if the file pointer is at the end of the file, STAT = 3 is output because there is no data to read.

■ **Example Program**

(1) LD



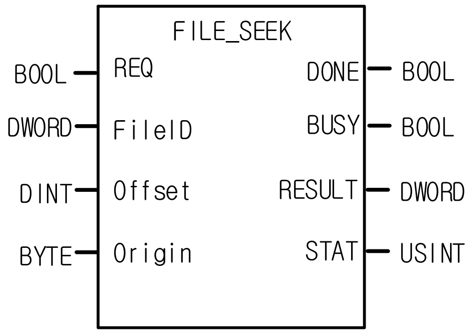
(a) FILE_OPEN After this is successfully done, you must enter the output value FileID.
(b) When the execution condition (% MX0) is On, FILE_READ function is executed.
(c) ReadAddr can be set to array type or data type.
(d) When set as an array type, the data of the file stored on the SD card can be read as an array with the set size. For example, if you set 10 DWORD array, the data stored in SD card will be read as array of size. When set as data type, only the corresponding data value is read. Size value is meaningless.
(e) During normal operation ReadSize shows the actual read data size.
(f) Depending on the status of the SD card or the status of the file, an error is displayed in STAT. In normal operation, 0 is output.

(2) ST

INST_FILE_READ (REQ: =% MX0, FileID: = FileID, ReadAddr: = ReadAddr, Size: = Size, DONE => DONE, BUSY => BUSY, ReadSize => ReadSize, stat => stat);

FILE_SEEK	Applied model
Move SD memory card inside	**XMC**

Motion function block type



input		
BOOL	REQ	Set the program to run on the rising edge.
DWORD	FileID	The ID of the file that was opened.
DINT	Offset	The offset position from Origin.
BYTE	Origin	This is the base location.
Print		
BOOL	Done	Indicates completion of function block completion.
BOOL	Busy	Indicates that function block execution is not completed.
DWORD	Result	Outputs the changed position.
USINT	STAT	The error number generated during function block output is output.

(1) Specify the location to access the file opened with 'FileID' on the SD memory card.
(2) The reference position is set in 3 modes as below.

Origin value	Origin Location
0	In front of file
1	Current file pointer location
2	End of file

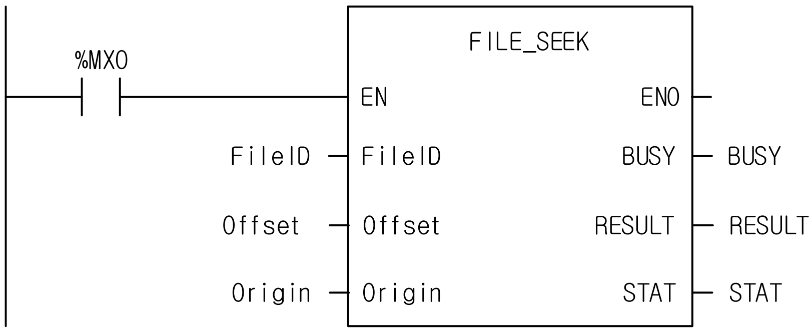
(3) Moves the file pointer position by adding the reference position setting value and the input offset value.
(4) When operating, BUSY = 1. When completed, BUSY = 0 and DONE = 1.
(5) The STAT information is STAT = 0, and the STAT information when an error occurs is as follows.

STAT	Error state

0	normal
1	SD memory card access failed
2	FileID is not open
3	When the position value to move is smaller than the origin value

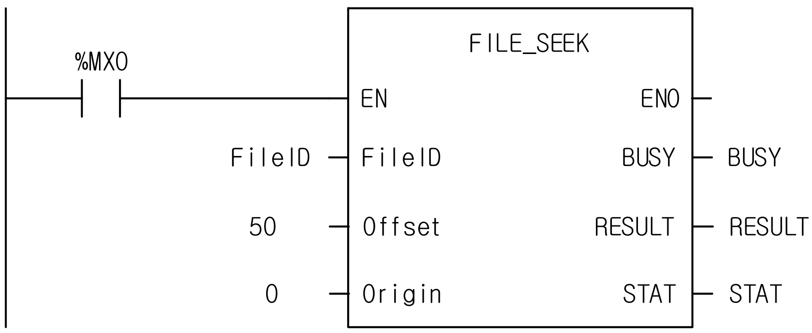
■ **Example Program**

(1) LD

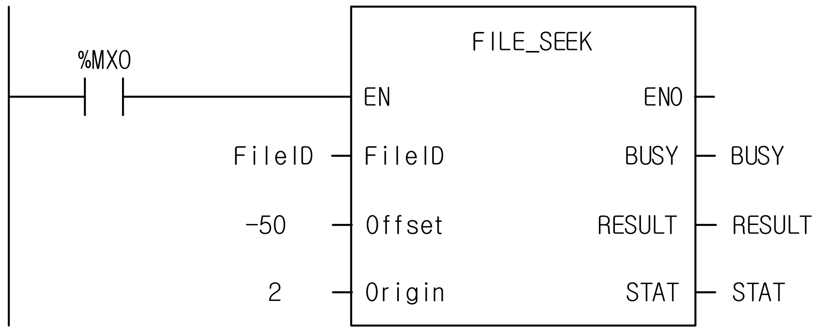


(a) FILE_OPEN After this is successfully done, you must enter the output value FileID.
(b) When the execution condition (% MX0) is On, FILE_SEEK function is executed.
(c) Move the file pointer by adding the Offset value to the Origin setting. For example, if you want to move to the beginning of the file, set Offset = 0, Origin = 0, and set Offset = 20, Origin = 0 to move to the 20 bytes from the beginning.
(d) During normal operation RESULT displays the current file pointer.
(e) Depending on the status of the SD card or the status of the file, an error is displayed in STAT. In normal operation, 0 is output.

※ **Moving to 50 byte position when file size is 100 bytes Example**



(a) When the execution condition (% MX0) is On, FILE_SEEK function is executed.
(b) Since Origin = 0, it moves to the starting point of the file and moves to offset position by Offset = 50.
(c) The 50 bytes moved to RESULT are output.
(d) It is also possible to go backwards from the end of the file as shown below.

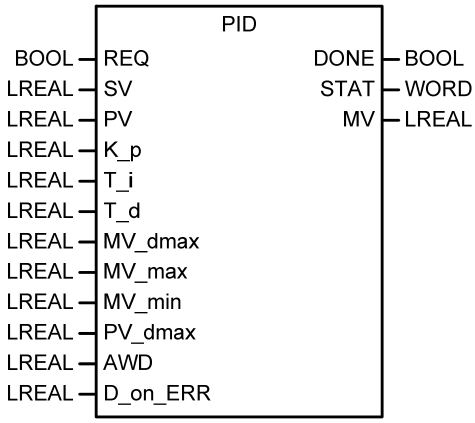


(2) ST

INST_FILE_SEEK (REQ: =% MX0, FileID: = FileID, Offset: = Offset, Origin: = Origin, DONE => DONE, BUSY => BUSY, RESULT => RESULT, stat => stat);

PID	Applied model
PID operator	XMC

Motion function block type



input		
BOOL	REQ	Execute the function block.
LREAL	SV	Target value (SV)
LREAL	PV	Current value (PV)
LREAL	K_p	P constant (K_p)
LREAL	T_i	I constant (T_i) [sec]
LREAL	T_d	D constant (T_d) [sec]
LREAL	MV_dmax	MV variation limit
LREAL	MV_max	MV max limit
LREAL	MV_min	MV minimum limit
LREAL	PV_dmax	PV variation limit
BOOL	AWD	Anti Wind-up prohibited (0: operation, 1: prohibited)
BOOL	D_on_ERR	Differential calculation source selection (0: PV, 1: ERR)
Print		
BOOL	DONE	Indicates that the PID operation is normally performed.
WORD	STAT	PID status alarm
LREAL	MV	Output value (MV)

(1) This function block is a function block that receives the target value (SV) and the current value (PV) of the control target and performs PID operation to output to MV.

(2) Target value SV input is the current status of the control target. This state is represented by a number, and it should be converted to the PV reference according to the gain of the system. For example, in a system where the temperature is 50 ° C and the PV is sensed at 5000, set SV to 5000 when controlling the temperature to 50 ° C.

(3) Current value The PV input is an indicator of the current state of the control object. In general, the input from the sensor is stored in the CPU device via an input device such as an A / D conversion module, You must give.

(4) The K_p input sets the proportional constant of the current PID operator. Since K_p is multiplied by P, I, D (proportional, integral, derivative) of the PID control effect, the proportional and differential effects become large and the integral effect decreases when K_p becomes large. Especially when K_p input is 0, PID control is not performed.

(5) The T_i input sets the integral time constant of the loop. Since T_i divides the I (integral) term of the PID control effect, the integral effect becomes smaller when T_i becomes larger. If T_i input is 0, I control is not performed.

(6) The T_d input sets the derivative time constant of the loop. T_d is multiplied by the D (derivative) term of the PID control effect, so the larger the T_d, the greater the differential effect. If T_d input is 0, D control is not performed.

(7) The PV_dmax input limits the PV variation of the loop. In actual control, PV does not always reflect the exact state of the system. Unwanted signals such as sensor malfunction, noise, disturbance, etc. may be mixed and reflected in the PV. In such a case, the PV may suddenly change suddenly, causing a large change in the PID output. In order to prevent this phenomenon, if the PV changes more than the value set in _PID [B] _ [L] dPV_max, it prevents it from changing more than the setting value.

On the other hand, if PV_dmax is set too small, the change of the system may be delayed and the convergence time may take a long time. Especially when the corresponding setting value is set to 0, the function of limiting the PV change amount does not work.

(8) The MV_dmax input limits the amount of MV change in the loop. If the output of the control system suddenly changes, the system may become unstable, or the actuator may be loaded with a large load, resulting in a malfunction or unstable operation. This is an item that limits the amount of change in the controller output to prevent this. This function does not work if the corresponding setting value is set to 0.

(9) The MV_max input limits the maximum MV of the loop. Limits the maximum value of the controller output delivered to the output device to prevent overload and prevent system error in advance. It also prevents overflow and other undesired values from being delivered.

(10) The MV_min input limits the minimum MV of the loop. Limits the minimum value of the controller output delivered to the output device to prevent system faults in advance. It also prevents overflow and other undesired values from being delivered.

(11) The D_on_ERR input sets the D operation source of the corresponding PID loop to ERR. D operation is calculated by ERR or PV. When D operation is performed using ERR, the D response changes suddenly at the moment when the SV is changed by the user, so that excessive input may be applied to the actuator momentarily. In order to prevent this, PV method is used in D operation and default value is set to D operation using PV. If ERR is used without this algorithm, this bit turns on. If the corresponding bit is Off, PID performs D operation with PV value, and

when it is On status, it performs D operation with ERR value.

(12) AWD input is the input to enable or disable Anti Wind-up function. If the input is turned on, the Anti Wind-up function is disabled.

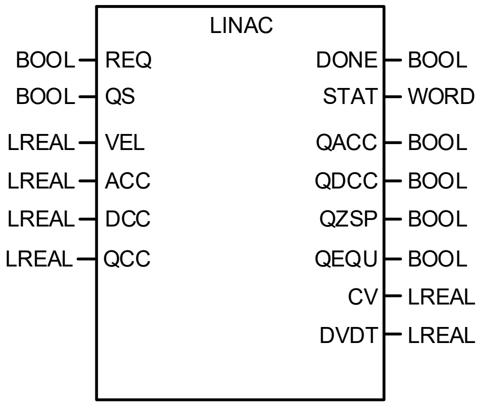
(13) Each bit of the STAT output indicates the status of the corresponding PID controller or an abnormal condition. Each bit is ON only when the corresponding operation occurs, and returns to OFF when the corresponding operation is released.

The lower 8 bits of STAT indicate various abnormal conditions of the loop, and the upper 8 bits indicate the control status of the corresponding loop. The assignment of each bit is as follows.

beat	condition
0	T_s setting is too small to indicate that the operation is skipping
One	Signals that the K_p value is zero.
2	Notice that PV variation is limited.
3	Notice that MV variation is limited.
4	Signals that the MV maximum value is limited.
5	Signals that the MV minimum is limited.
8	PID operation is being performed.
15	Indicates that Anti Wind-up is in operation during PID operation

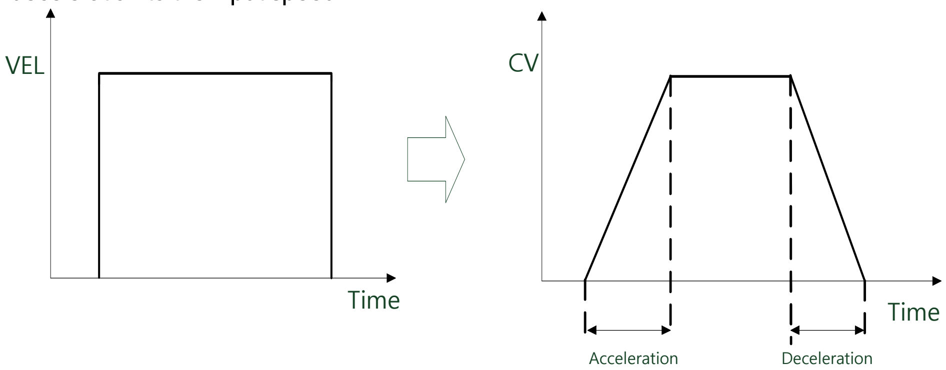
LINAC	**Applied model**
Acceleration / deceleration command	**XMC**

Motion function block type



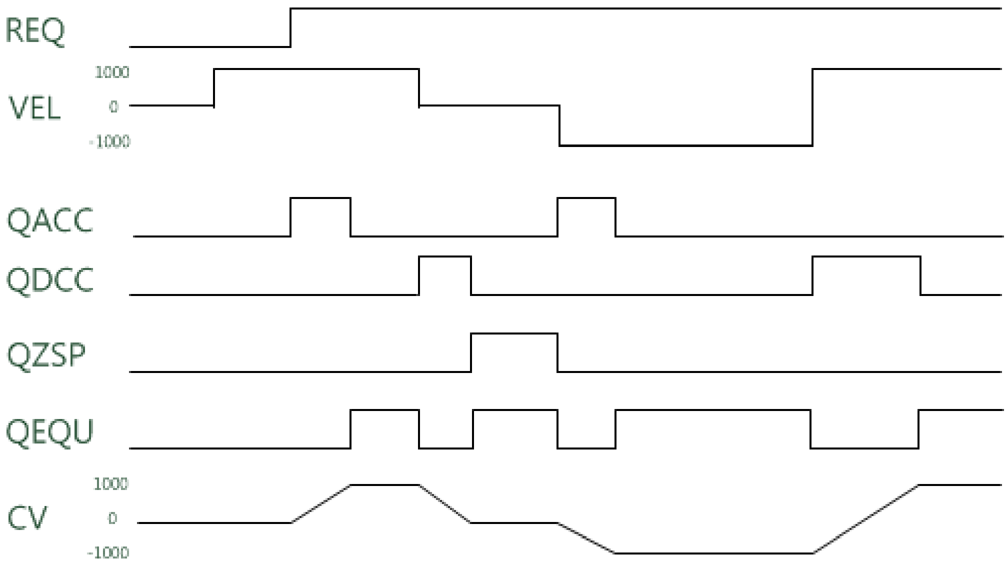
input		
BOOL	REQ	Run the LINAC command.
BOOL	QS	Enter emergency stop
LREAL	VEL	Specify the target speed [u / s]
LREAL	ACC	Specify acceleration [u / s^2]
LREAL	DCC	Specify the deceleration [u / s^2]
LREAL	QCC	Specify rapid stop deceleration. [u / s^2]
Print		
BOOL	DONE	Indicates completion of function block execution.
WORD	STAT	Indicates the error value of the function block.
BOOL	QACC	Indicates whether acceleration is in progress.
BOOL	QDCC	Indicates whether deceleration is in progress.
BOOL	QZSP	Indicates whether the speed of the current speed is zero or not.
BOOL	QEQU	Indicates whether the target speed matches the current speed.
LREAL	CV	Indicates the current speed.
LREAL	DVDT	Indicates current acceleration / deceleration speed.

(1) This function block is a function block that outputs the reached speed value by applying constant acceleration / deceleration to the input speed.

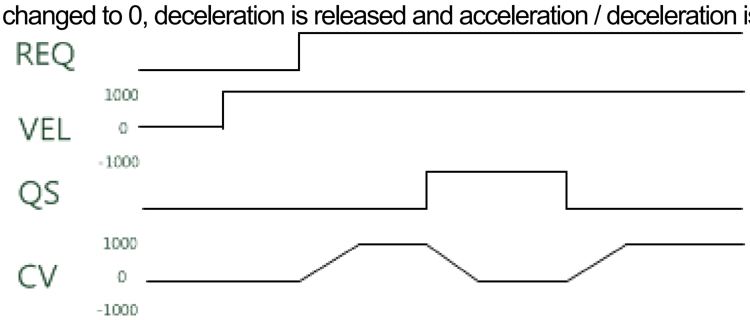


(2) REQ input At this rising edge, the ACC / DCC / QCC value is used in the function block and the ACC / DCC / QCC value is not changed during operation.

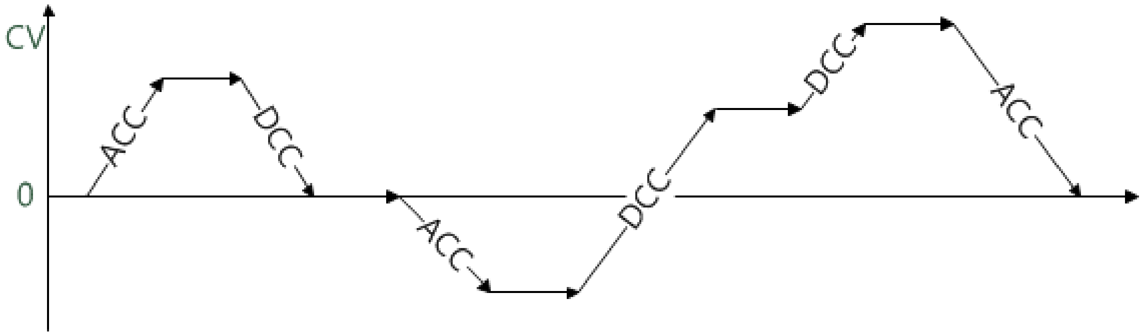
(3) QACC / QDCC / QZSP / QEQU output during operation is as follows.



(4) If the QS value is 1, deceleration (deceleration) is set at the deceleration set by QCC. When the QS value is changed to 0, deceleration is released and acceleration / deceleration is performed to the input target speed.



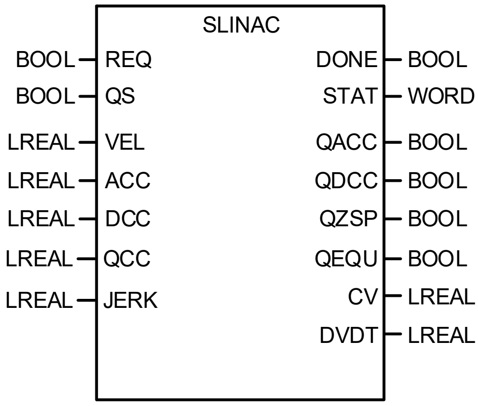
(5) When the stop status is 0, it accelerates in the direction of the input target speed and decelerates in the opposite direction. In case of stop operation at zero speed, the direction of acceleration / deceleration is changed.



When a negative number is input to ACC, QCC, DCC, 11 (0x000B) error is output to STAT.

SLINAC	Applied model
S-Curve Accelerometer Command	**XMC**

Motion function block type



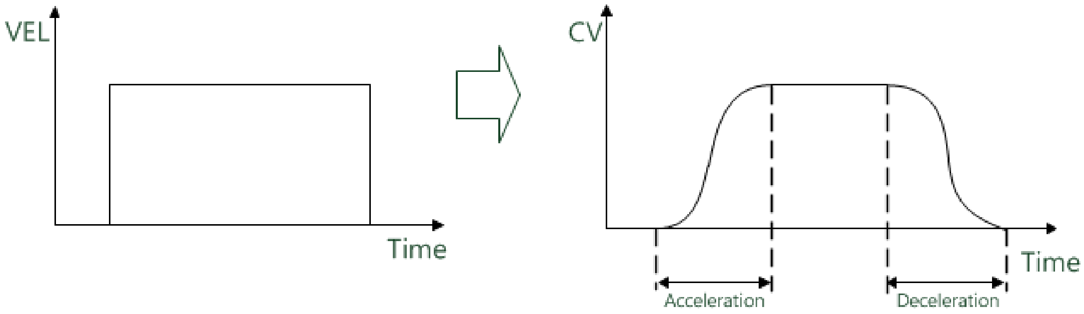
input

BOOL	REQ	Run the SLINAC command.
BOOL	QS	Enter emergency stop
LREAL	VEL	Specify the target speed [u / s]
LREAL	ACC	Specify acceleration [u / s^2]
LREAL	DCC	Specify the deceleration [u / s^2]
LREAL	QCC	Specify rapid stop deceleration. [u / s^2]
LREAL	JERK	Specifies the rate of acceleration / deceleration change. [u / s^3]

Print

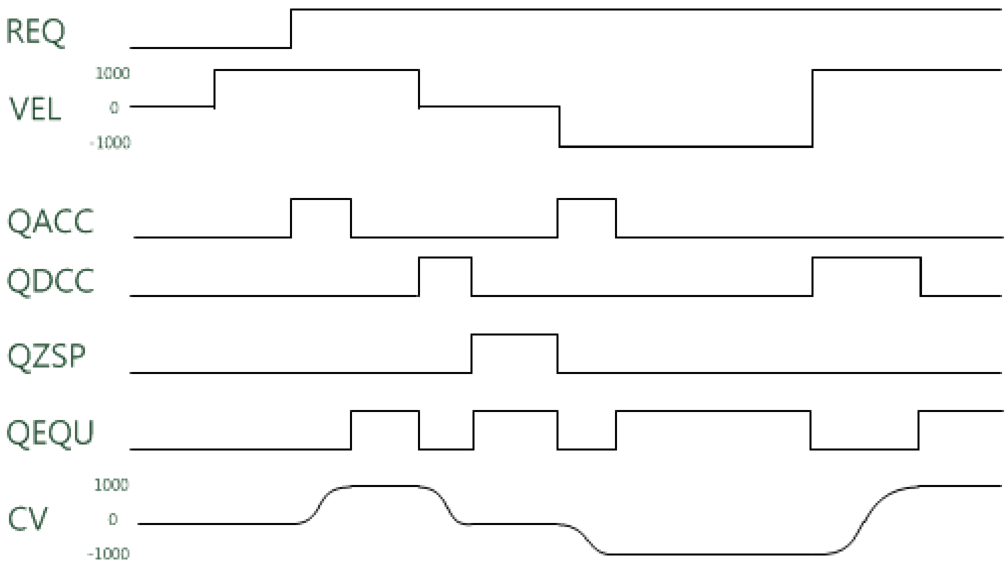
BOOL	DONE	Indicates completion of function block execution.
WORD	STAT	Indicates the error value of the function block.
BOOL	QACC	Indicates whether acceleration is in progress.
BOOL	QDCC	Indicates whether deceleration is in progress.
BOOL	QZSP	Indicates whether the speed of the current speed is zero or not.
BOOL	QEQU	Indicates whether the target speed matches the current speed.
LREAL	CV	Indicates the current speed.
LREAL	DVDT	Indicates current acceleration / deceleration speed.

(1) This function block is a function block which outputs the reached speed value by applying acceleration / deceleration applied JERK up to input speed.



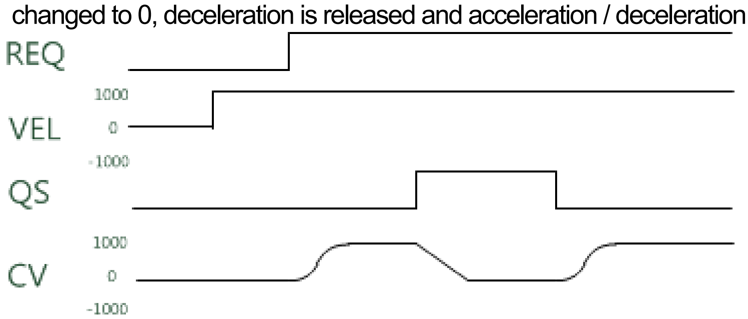
(2) REQ input At this rising edge, the ACC / DCC / QCC value is used in the function block and the ACC / DCC / QCC value is not changed during operation.

(3) QACC / QDCC / QZSP / QEQU output during operation is as follows.



(4) Overshoot or undershoot may occur if the target speed changes before the target speed is reached.

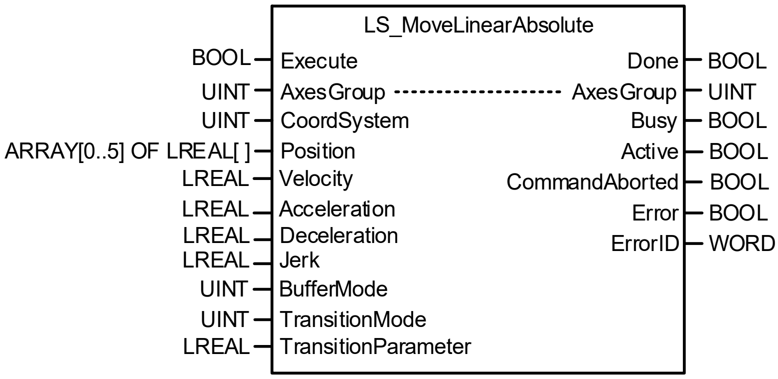
(5) If the QS value is 1, deceleration (deceleration) is set at the deceleration set by QCC. When the QS value is changed to 0, deceleration is released and acceleration / deceleration is performed to the input target speed.



(6) When the values of ACC, DCC, QCC and JERK are negative, 11 (0x000B) error is output to STAT.

LS_MOVELINEARABSOLUTE	Applied model
Coordinate system absolute positioning linear interpolation operation	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group to perform coordinate system absolute position linear interpolation operation. (1 ~ 16: Group 1 ~ Group 16)
Input		
BOOL	Execute	Give coordinate system absolute position linear interpolation operation command to the relevant group in the rising Edge.
UINT	CoordSystem	Set the coordinate system type (1:MCS 2:PCS)
LREAL[]	Position	Enter the target position of the end point of the machine.
LREAL	Velocity	Specify the maximum speed of the route. [u/s]
LREAL	Acceleration	Specify the maximum acceleration. [u/s^2]
LREAL	Deceleration	Specify the maximum deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4. BufferMode)
UINT	TransitionMode	Specify the route change mode of group operation. (Refer to 16.1.6. TransitionMode)
LREAL	TransitionParameter	Specify the parameter of the route change setting of group operation. (Refer to 16.1.6. TransitionMode)
Output		
BOOL	Done	Indicate that the execution of motion function block is completed.

BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant group.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block issues absolute positioning linear interpolation command based on coordinate system on the axes group designated by AxesGroup input.

(2) When this motion function block is executed, interpolation control is performed in a linear path from the current position to the target position of the end point of the machine.

(3) Specify the speed, acceleration, deceleration, and the change rate of acceleration/deceleration of interpolation route in Velocity, Acceleration, Deceleration, and Jerk inputs respectively.

(4) Velocity is to set the maximum interpolation speed of the machine respect to the combined distance of current position to target position value(Position[0], Position[1], Position[2]). If the position value of the target position is the same as the current position, it is the speed relative to the composite angle of the angle values (Position [3], Position [4], Position [5]).

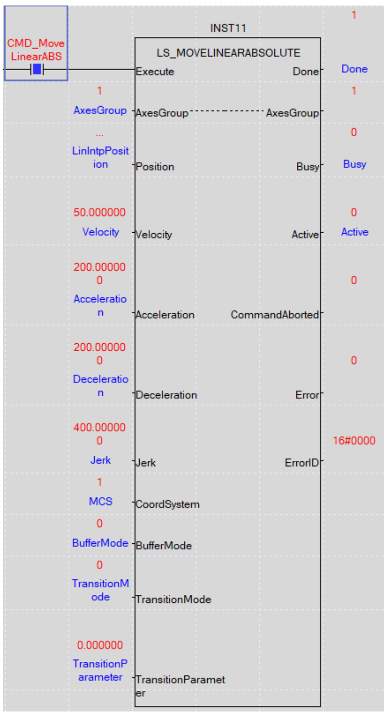
(5) The changed parameters can be applied by re-executing the function block (Execute input is On) before the command is completed. Only Velocity, Acceleration, Deceleration, Jerk, Position input can be updated.

(6) Velocity input can be set to 0 or changed.

(7) Example program

This example shows the linear interpolation to the target position (100, 200, 0) when the current command position is (0, 0, 0).

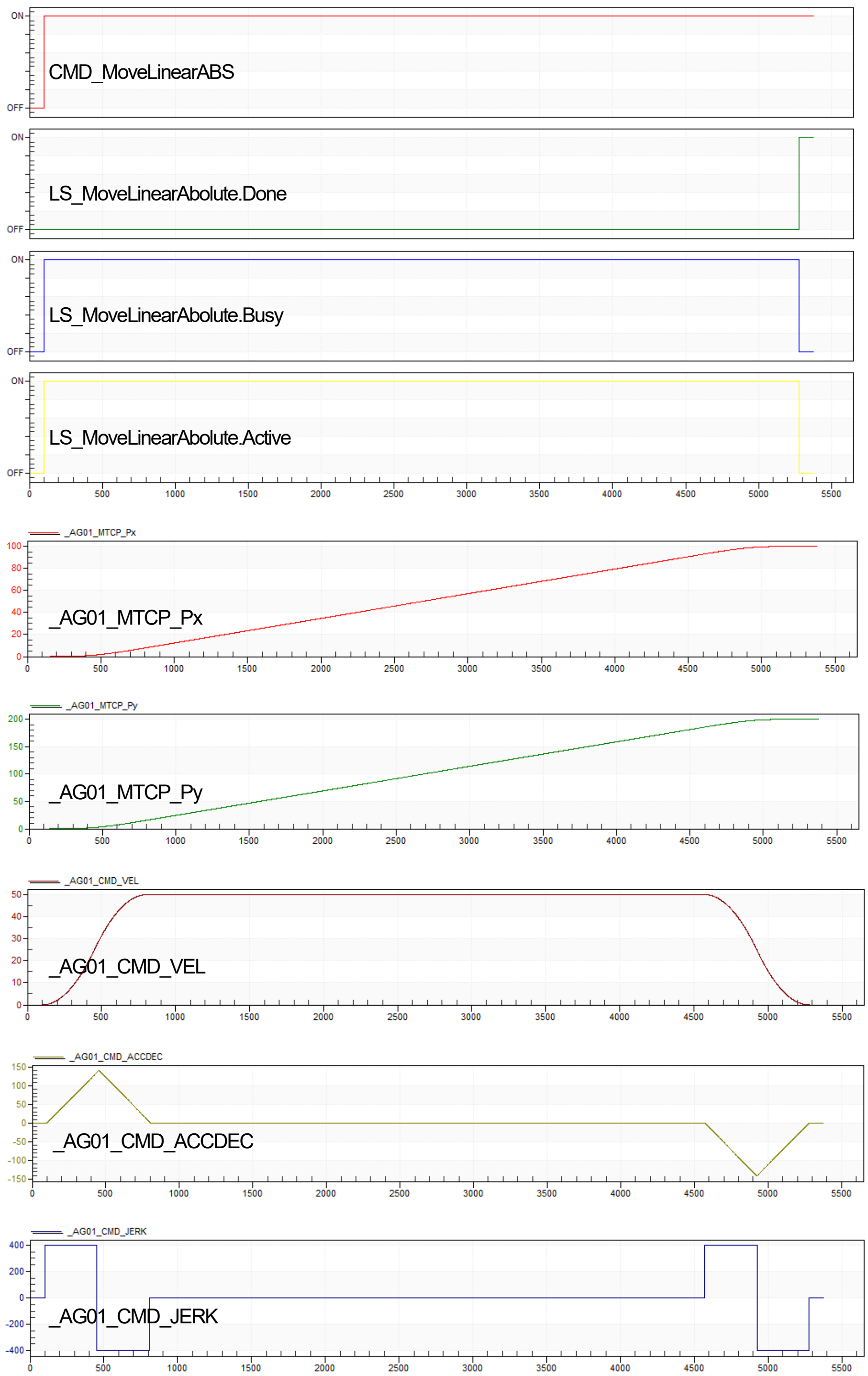
(a) Function block setting



	Variable/Device	Value	Type	Device/Variable	Comment
1	_AG01_MTCP_Px	1.0000000000000000e+002	LREAL	%FL3367	Axis group 01 X axis position(MCS)
2	_AG01_MTCP_Py	2.0000000000000000e+002	LREAL	%FL3368	Axis group 01 Y axis position(MCS)
3	_AG01_MTCP_Pz	0.0000000000000000e+000	LREAL	%FL3369	Axis group 01 Z axis position(MCS)
4	⊟ LinIntpPosition		ARRAY[0..		
5	LinIntpPosition[0]	1.0000000000000000e+002	LREAL		
6	LinIntpPosition[1]	2.0000000000000000e+002	LREAL		
7	LinIntpPosition[2]	0.0000000000000000e+000	LREAL		
8	LinIntpPosition[3]	0.0000000000000000e+000	LREAL		
9	LinIntpPosition[4]	0.0000000000000000e+000	LREAL		
10	LinIntpPosition[5]	0.0000000000000000e+000	LREAL		

Target Position

(b) Timing diagram

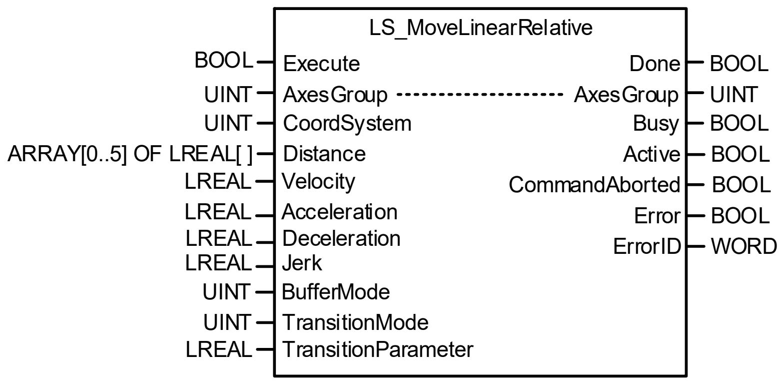


(8) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.50	V4.30
XMC-E16A	V1.50	V4.30
XMC-E08A	V1.50	V4.30
XMC-E32C	V1.50	V4.30

LS_MOVELINEARRELATIVE	Applied model
Coordinate system relative positioning linear interpolation operation	**XMC**

Motion Function Block



Input-Output		
UINT	AxesGroup	Set the group to perform coordinate system relative position linear interpolation operation. (1 ~ 16: Group 1 ~ Group 16)
Input		
BOOL	Execute	Give coordinate system absolute position linear interpolation operation command to the relevant group in the rising Edge.
UINT	CoordSystem	Set the coordinate system type (1:MCS 2:PCS)
LREAL[]	Distance	Enter the target distance of the end point of the machine.
LREAL	Velocity	Specify the maximum speed of the route. [u/s]
LREAL	Acceleration	Specify the maximum acceleration. [u/s^2]
LREAL	Deceleration	Specify the maximum deceleration. [u/s^2]
LREAL	Jerk	Specify the change rate of acceleration/deceleration. [u/s^3]
UINT	BufferMode	Specify the sequential operation setting of motion function block. (Refer to 16.1.4. BufferMode)
UINT	TransitionMode	Specify the route change mode of group operation. (Refer to 16.1.6. TransitionMode)
LREAL	TransitionParameter	Specify the parameter of the route change setting of group operation. (Refer to 16.1.6. TransitionMode)
Output		
BOOL	Done	Indicate that the execution of motion function block is completed.

BOOL	Busy	Indicate that the execution of motion function block is not completed.
BOOL	Active	Indicate that the current motion function block is controlling the relevant group.
BOOL	CommandAborted	Indicate that the current motion function block is interrupted while it is running.
BOOL	Error	Indicate whether an error occurs or not.
WORD	ErrorID	Output the number of error occurred while motion function block is running.

(1) This motion function block issues relative positioning linear interpolation command based on coordinate system on the axes group designated by AxesGroup input.
(2) When this motion function block is executed, interpolation control is performed in a linear path from the current position to the target position of the end point of the machine.
(3) Specify the speed, acceleration, deceleration, and the change rate of acceleration/deceleration of interpolation route in Velocity, Acceleration, Deceleration, and Jerk inputs respectively.
(4) Velocity is to set the maximum interpolation speed of the machine respect to the combined distance of target distance value(Distance[0], Distance[1], Position[2]). If the distance value is zero, it is the speed relative to the composite angle of the angle values (Distance[3], Distance[4], Distance[5]).
(5) The changed parameters can be applied by re-executing the function block (Execute input is On) before the command is completed. Only Velocity, Acceleration, Deceleration, Jerk, Position input can be updated.
(6) Velocity input can be set to 0 or changed.
(7) The available version information of this Motion Function Block is as follows.

Item / product name	Module O / S	XG5000
XMC-E32A	V1.50	V4.30
XMC-E16A	V1.50	V4.30
XMC-E08A	V1.50	V4.30
XMC-E32C	V1.50	V4.30

Chapter 17. IL (Instruction List)

17.1. summary

1) IL programs are portable, with all text editors available.
2) It executes one command per line and can be applied to simple PLC program.
3) It is easy to program by someone familiar with computer assembly language.

```

        // USER FUNCTION BLOCK example
        CAL INST_CMD_TMR(IN:=%IX5.0.0, PT:=T#300ms)
        LD  INST_CMD_TMR.Q
        ST  BOOL1

        // Arithmetic statement example((1.000e+3*1.000e+3)-(4*1.0*2.0))
        LD  1.000e+3
        ST  REAL1
        MUL REAL1
        SUB( 4
            MUL( 1.0
                MUL2_REAL((*IN1:=CR(REAL),*) IN2:=(2.0))
            )
        )
        ST REAL2

        //IF statement example
        GT  0
        JMPN    ELSEIF
        LD  0
        ST  NROOTS
        JMP END
    ELSEIF:
        LD  1
        ST  NROOTS
    END:
```

17.2. Current Result: CR)

1) IL has the operation results up to that point in the calculation process, which is called the current value (CR).
2) There is only one CR in the IL operation.
3) CR is available in all data types and does not have a fixed size.
4) LD (Load) is the operator that determines the data type of the CR while putting a certain value in CR.
5) Operator performs operations on defined CRs and operands. Therefore, operators except LD, LDN, JMP, CAL, RET, and SCAL can not perform operations unless CR is defined.
6) The operator defines (creates) or changes CRs according to each group of operators and makes them unaffected or undefined.

Operator group	Abbreviation	Explanation
Create	C	Defines the CR. Existing CRs are replaced.
Process	P	The CR type or value is changed by the operation result.
Leave unchanged	U	The result of the operation does not affect the CR.
Set to undefined	-	Change the CR to an undefined state after the operation is finished.

<Table 1> CR conversion according to operator group

Example	Explanation
LD% IX0.0.0 LD VAL_INT (* INT *)	Put the variable% IX0.0.0 value in CR. At this time, since the data type of the direct variable represented by X is BOOL, the data type of CR is BOOL. If the variable VAL_INT is declared as INT, VAL_INT value is put into CR, and the data type of CR is INT
LD% IX0.0.0 ST VAL_INT (* INT *)	On the first line, the CR is specified as BOOL. On the second line, CR is set as INT I tried to use it, so I get an error at compile time.
LD TRUE ST% QX0.0.0 LD 20 ST VAL_INT (* INT *)	This is a normal program because the data types for storing the specified CRs are the same.

17.3. Expression

1) An expression consists of an operator that can have a modifier and an operand, a label, and an annotation that are the subject of the operator. The operands are defined characters (numeric characters, strings, and time characters), defined variables (general variables, direct variables) It may be a defined function (function, function block)

```
//lable     Operator    Operand     Annotation
START:      LD          %IX1        (*PUSH BUTTON  *)
            ANDN        %MX5        (*NOT INHIBITED*)
            ST          %QX2        (*FAN ON       *)
```

2) Each instruction starts on a new line, each line contains an operator with a selectable modifier, one or more operands separated by commas if necessary for a particular operation, CR, the result of the previous operation, and the result of the operation It affects the CR.

17.4. Label

1) Labels are displayed in the operator area with a colon (:) after the label name.
2) The label is used as the destination of the jump instruction.

3) The label initializes the CR.

17.5. Modifier

1) The modifier is used immediately after the operator and is performed by modifying the original arithmetic function.
2) Modifiers include N, C, and (.
3) The modifier 'N' indicates the BOOL inverse of the operand (Boolean Negation).
4) Modifier 'C' indicates that the specified operation will only work if the currently computed CR is TRUE (1).

Example	Explanation
ANDN% IX2.0.0	CR: = CR AND NOT% IX2.0.0
AND (% IX1.0.0 OR% IX2.0.0)	CR: = CR AND (% IX1.0.0 OR% IX2.0.0) The execution of the AND is deferred until a). As a result, it means that% IX1.0.0 OR% IX2.0.0 in the parentheses is executed first, and then the operation is performed with the result.
JMPC THERE	If CR is TRUE (1), it means jump to THERE.

Note

If the modifier N is followed by a bitwise operator (LDN, STN, ANDN, ORN), it means BOOL inversion of the operation result CR, and if it follows the execution operator (JMP, CAL, RET, SCAL) Conditional Execution. In this case, if N is a modifier for conditional select, it means it works when CR is FALSE (0) as opposed to C.

5) Modifier The parentheses '(' indicates that the operation of the operator is delayed until ')' is encountered. Since there is only one CR in the operation of IL, it is possible to perform a delay operation in which the CR is held for a while and another operation is performed, and the result and the stored CR value are calculated.
6) Modifier The parentheses after the '(' operand are used after the LD. Please refer to <Table 1> of 15.5 which expresses the same expression.

Technology	**example**
Expressions in parentheses that begin with an explicit operator	AND(LD %IX1 OR %IX2)
Expressions in parentheses (short forms)	AND(%IX1 OR %IX2)

<Table 1> IL language expression in parentheses

17.6. Basic operator

1) The basic operators are:

number	Operator	Modifier	Operator group	Explanation
One	LD	N	C	Put the operand into the CR.
2	ST	N	U	Store the CR in the operand.
3 4	SET RST		U U	If CR value is BOOL type TRUE (1), set BOOL type operand to TRUE (1). If CR value is BOOL type TRUE (1), BOOL type operand is FALSE (0).
5 6 7	AND OR XOR	N, (N, ((	P P P	Logic AND Operation Logic OR operation Logic XOR operation
8 9 10 11	ADD SUB MUL DIV	((((	P P P P	Arithmetic addition operation Arithmetic subtraction operation Arithmetic multiplication operation Arithmetic division operation
12 13 14 15 16 17	GT GE EQ NE LE LT	((((((	P P P P P P	Comparison operation:> (large) Compare operation:> = (equal to or greater than) Comparison operation: = (same) Comparison operation: <> (not equal) Comparison operation: <= (same or smaller) Comparison operation: <(small)
18 19 20 21	JMP CAL RET SCAL	C, N C, N C, N C, N	- - - -	Jump to label Function call without function block or return value Return from function or function block Subroutine call
22	)		U	Use '(' with modifier to do deferred operations.

2) Operators 5 through 17 are replaced with the current result (CR) used by the operator (OP) in relation to the operand as shown below.
Current Result <= Current Result Operand Operand
It computes the CR and operand values using the operator's arithmetic function and stores the result back into the CR

3) The comparison operator compares the CR on the left with the operand on the right and stores the BOOL result in CR

Example	Explanation

AND% IX1.0.0	CR: = CR AND% IX1.0.0
GT% MW10	If CR is greater than the value in internal memory% MW10, the value of CR is BOOL type TRUE (1); otherwise, it is FALSE (0).
LD VAL_INT1 (a) EQ VAL_INT2 (b) AND% IX0.0.0 (c) ST% QX0.0.0 (d)	In line (a), place an INT value named VAL_INT1 in the CR. In the line (b), this CR is compared with the INT value of VAL_INT2. If it is the same, the value of BOOL type TRUE (1) is added. If it is different, the value of FALSE (0) is put into CR. At this time, the data type of CR changes from INT to BOOL. Therefore, no compile error occurs when using commands (c) and (d).

Note

Most of the operation instructions do not change the data type of CR even after the operation is finished. However, unlike this, the data type of CR is different for comparison instructions, function, and JMP / CAL / RET / SCAL operators.

For details, refer to <Table 1> CR conversion according to operator group in 15.2.

17.6.1. LD

1) Put the operand into the CR. At this time, the data type of CR is changed to the data type of the operand.
2) Modifier N: If the operand is BOOL, the operand value is inverted and placed in the CR

Operator group	Modifier			operand
	C	N	(	
C (Create)		○		All data types are available (ANY type). Water is available.

Example	Explanation
LD TRUE	Put the value of BOOL 1 into CR. At this time, the data type of CR is BOOL.
LD INT_VALUE	Put INT variable INT_VALUE into CR. At this time, the data type of CR is INT.
LD T # 1S	Put the elapsed time constant T # 1S in CR. At this time, the data type of CR is TIME.
LDN B_VALUE	Invert the B_VALUE value, which is a BOOL variable, into the CR. At this time, the data type of CR is BOOL.

Note

ANY types include all types. For details, refer to the data type hierarchy diagram in 3.2.2.

17.6.2. ST

1) Put the CR value into the operand. At this time, the data type of CR and the data type of operand must be the same data type.
2) CR value does not change.
3) Modifier N: If the CR data type is BOOL, the CR value is inverted and put in the operand. At this time, the value of CR does not change

Operator group	Modifier			operand
U (Leave unchanged)	C	N	(	All data types are available. Constants are not allowed. Must be the same as the data type of CR
		○		

Example	Explanation
LD FALSE	BOOL Put the value of 0 into CR. At this time, the data type of CR is BOOL.
ST B_VALUE1	Put CR value 0 into B_VALUE1 variable whose data type is BOOL.
STN B_VALUE2	Inverts the CR value (1) and places it in B_VALUE2 whose data type is BOOL.
LD INT_VALUE	Put INT variable INT_VALUE into CR.
	At this time, the data type of CR is INT.
ST I_VALUE1	Put CR value into I_VALUE1 variable with data type INT.
LD D # 1995-12-25	Put the date constant D # 1995-12-25 in the CR.
	At this time, the data type of CR is DATE.
ST D_VALUE1	Put the CR value into the D_VALUE1 variable whose data type is DATE.

17.6.3. SET

1) If the CR value is BOOL 1, the operand whose data type is BOOL is set to 1.
2) If the CR value is BOOL 0, no operation is performed.
3) CR value does not change.
4) There is no change.

Operator group	Modifier			operand
U (Leave unchanged)	C	N	(	Only BOOL data type is available. Constants are not allowed.

Example	Explanation
LD FALSE	BOOL Put the value of 0 into CR. At this time, the data type of CR is BOOL.
S B_VALUE1	The CR value is 0, so no action is taken.
	The value of the B_VALUE1 variable does not change.
LD TRUE	Put the value of BOOL 1 into CR. At this time, the data type of CR is BOOL.

S B_VALUE2	Since the CR value is 1, set the value of the B_VALUE2 variable whose data type is BOOL to 1.

17.6.4. RST (Reset)

1) If the CR value is BOOL 1, the value of the operand whose data type is BOOL is set to 0.
2) If the CR value is BOOL 0, no operation is performed.
3) CR value does not change.
4) There is no change.

Operator group	Modifier			operand
U (Leave unchanged)	C	N	(	Only BOOL data type is available. Constants are not allowed.

Example	Explanation
LD FALSE	BOOL Put the value of 0 into CR. At this time, the data type of CR is BOOL.
R B_VALUE1	The CR value is 0, so no action is taken. The value of the B_VALUE1 variable does not change.
LD TRUE	Put the value of BOOL 1 into CR. At this time, the data type of CR is BOOL.
R B_VALUE2	Since the CR value is 1, the value of the B_VALUE2 variable whose data type is BOOL is set to 0. CR value does not change.
ST B_VALUE3	Put CR value (1) in B_VALUE3 variable whose data type is BOOL.

17.6.5. AND

1) Logically ANDs the CR value and operand value and puts the result in CR. At this time, the data type of the CR and the data type of the operand must be the same.
2) The value of the operand does not change.
3) Modifier N: If the data type of the operand is BOOL, the value of the operand is inverted and computed with the CR value.
4) Modifier (: If the data type of the operand is BOOL, keep the current CR value somewhere else and put the value of the operand in CR. (Delay calculation)

Operator group	Modifier			operand
P (Process)	C	N	(	Only BOOL, BYTE, WORD, DWORD, and LWORD data types are allowed. Water is also available.
		○	○	

Example	Explanation

LD B_VALUE1	Put the value of B_VALUE1 whose data type is BOOL into CR. At this time, the data type of CR is BOOL.
AND B_VALUE2	ANDs the CR value with the value of B_VALUE2 whose data type is BOOL, and places the result in CR.
ANDN B_VALUE3	The CR value and the value of B_VALUE3 whose data type is BOOL are inverted and ANDed, and the result is put into CR.
ST B_VALUE4	Put CR value into B_VALUE4 variable whose data type is BOOL. ***B_VALUE4 <== B_VALUE1 AND B_VALUE2 AND NOT (B_VALUE3)***
LD W_VALUE1	Put the WORD variable W_VALUE1 into CR. At this time, the data type of CR is WORD.
AND W_VALUE2	The CR value and the value of W_VALUE2 whose data type is WORD are ANDed, and the result is put into CR.
ST W_VALUE3	Put CR value into W_VALUE3 variable whose data type is WORD. ***W_VALUE3 <== W_VALUE1 AND W_VALUE2***
LD B_VALUE1	Put the value of B_VALUE1 whose data type is BOOL into CR. At this time, the data type of CR is BOOL.
AND (B_VALUE2	Keep the CR value elsewhere and put the value of B_VALUE2 whose data type is BOOL into CR.
OR B_VALUE3	ORs the CR value and the value of B_VALUE3 whose data type is BOOL, and places the result in CR.
)	ANDs the current CR value with the CR value stored elsewhere and places the result in CR.
ST B_VALUE4	Put CR value into B_VALUE4 variable whose data type is BOOL. ***B_VALUE4 <== B_VALUE1 AND (B_VALUE2 OR B_VALUE3)***

17.6.6. OR

1) Logically ORs the CR value with the value of the operand and places the result in CR. At this time, the data type of the CR and the data type of the operand must be the same.
2) The value of the operand does not change.
3) Modifier N: If the data type of the operand is BOOL, the value of the operand is inverted and computed with the CR value.
4) Modifier (: If the data type of the operand is BOOL, keep the current CR value somewhere else and put the value of the operand in CR. (Delay calculation)

Operator group	Modifier			operand
P (Process)	C	N	(	Only BOOL, BYTE, WORD, DWORD, and LWORD data types are allowed. Water is also available.
		○	○	

Example	Explanation
LD B_VALUE1	Put the value of B_VALUE1 whose data type is BOOL into CR. At this time, the data type of CR is BOOL.
OR B_VALUE2	ORs the CR value with the value of B_VALUE2 whose data type is BOOL and places the result in CR.
ORN B_VALUE3	The CR value and the value of B_VALUE3 whose data type is BOOL are inverted and ORed, and the result is put into CR.
ST B_VALUE4	Put CR value into B_VALUE4 variable whose data type is BOOL. ***B_VALUE4 <== B_VALUE1 OR B_VALUE2 OR NOT (B_VALUE3)***
LD W_VALUE1	Put the WORD variable W_VALUE1 into CR. At this time, the data type of CR is WORD.
OR W_VALUE2	ORs the CR value and the value of W_VALUE2 whose data type is WORD and puts the result in CR.
ST W_VALUE3	Put CR value into W_VALUE3 variable whose data type is WORD. ***W_VALUE3 <== W_VALUE1 OR W_VALUE2***
LD B_VALUE1	Put the value of B_VALUE1 whose data type is BOOL into CR. At this time, the data type of CR is BOOL.
OR (B_VALUE2	Keep the CR value elsewhere and put the value of B_VALUE2 whose data type is BOOL into CR.
AND B_VALUE3	ANDs the CR value and the value of B_VALUE3 whose data type is BOOL, and places the result in CR.
)	ORs the current CR value with the CR value stored elsewhere and places the result in CR.
ST B_VALUE4	Put CR value into B_VALUE4 variable whose data type is BOOL. ***B_VALUE4 <== B_VALUE1 OR (B_VALUE2 AND B_VALUE3)***

17.6.7. XOR

1) Logically XORs the CR value and operand value and puts the result in CR. At this time, the data type of the CR and the data type of the operand must be the same.
2) The value of the operand does not change.
3) Modifier (: If the data type of the operand is BOOL, keep the current CR value somewhere else and put the value of the operand in CR. (Delay calculation)

Operator group	Modifier			operand
P (Process)	C	N	(	Only BOOL, BYTE, WORD, DWORD, and LWORD data types are allowed. Water is also available.
			○	

Example	Explanation
LD B_VALUE1	Put the value of B_VALUE1 whose data type is BOOL into CR. At this time, the data type of CR is BOOL.
XOR B_VALUE2	XORs the CR value and the value of B_VALUE2 whose data type is BOOL, and places the result in CR.
XORN B_VALUE3	The CR value and the value of B_VALUE3 whose data type is BOOL are inverted, XORed, and the result is put into CR.
ST B_VALUE4	Put CR value into B_VALUE4 variable whose data type is BOOL. ***B_VALUE4 <== B_VALUE1 XOR B_VALUE2 XOR NOT (B_VALUE3)***
LD W_VALUE1	Put the WORD variable W_VALUE1 into CR. At this time, the data type of CR is WORD.
XOR W_VALUE2	XORs the CR value and the value of W_VALUE2 whose data type is WORD and puts the result in CR.
ST W_VALUE3	Put CR value into W_VALUE3 variable whose data type is WORD. ***W_VALUE3 <== W_VALUE1 XOR W_VALUE2***
LD B_VALUE1	Put the value of B_VALUE1 whose data type is BOOL into CR. At this time, the data type of CR is BOOL.
XOR (B_VALUE2	Keep the CR value elsewhere and put the value of B_VALUE2 whose data type is BOOL into CR.
AND B_VALUE3	ANDs the CR value and the value of B_VALUE3 whose data type is BOOL, and places the result in CR.
)	XORs the current CR value and the CR value stored elsewhere, and places the result in CR.
ST B_VALUE4	Put CR value into B_VALUE4 variable whose data type is BOOL. ***B_VALUE4 <== B_VALUE1 XOR (B_VALUE2 AND B_VALUE3)***

17.6.8. ADD

1) Performs an arithmetic operation on the CR value and the value of the operand, and places the result in CR. At this time, the data type of the CR and the data type of the operand must be the same.
2) The value of the operand does not change.
3) Modifier (: Keep the CR value somewhere else and put the value of the operand in CR. (Delay calculation)

Operator group	Modifier			operand
P (Process)	C	N	(	SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, REAL, LREAL data types are possible. Water is also available.
			○	

Example	Explanation
LD I_VALUE1	Put the value of I_VALUE1 whose data type is INT into CR. At this time, the data type of CR is INT.
ADD I_VALUE2	Adds the CR value and the value of I_VALUE2 whose data type is INT, and adds the result to CR.
ST I_VALUE3	Put CR value into I_VALUE3 variable whose data type is INT. ***I_VALUE3 <== I_VALUE1 + I_VALUE2***
LD D_VALUE1	Put the value of D_VALUE1 whose data type is DINT into CR. At this time, the data type of CR is DINT.
ADD (D_VALUE2	Keep the CR value elsewhere and put the value of D_VALUE2 with data type DINT in CR.
DIV D_VALUE3	The CR value and the value of D_VALUE3 whose data type is DINT are subjected to arithmetic division and the result is put into CR.
)	Performs an arithmetic operation on the current CR value and the CR value stored elsewhere, and places the result in CR.
ST D_VALUE4	Place the CR value in the B_VALUE4 variable with a data type of DINT. ***D_VALUE4 <== D_VALUE1 + (D_VALUE2 / D_VALUE3)***

17.6.9. SUB

1) Subtracts the CR value and the value of the operand by arithmetic operation and puts the result into CR. At this time, the data type of the CR and the data type of the operand must be the same.
2) The value of the operand does not change.
3) Modifier (: Keep the CR value somewhere else and put the value of the operand in CR (deferred operation).

Operator group	Modifier			operand
P (Process)	C	N	(	SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, REAL, LREAL data types are possible. Water is also available.
			○	

Example	Explanation
LD I_VALUE1	Put the value of I_VALUE1 whose data type is INT into CR. At this time, the data type of CR is INT.
SUB I_VALUE2	Subtracts the value of I_VALUE2 whose CR value and data type are INT, and puts the result into CR.
ST I_VALUE3	Put CR value into I_VALUE3 variable whose data type is INT.

	I_VALUE3 <== I_VALUE1 - I_VALUE2
LD D_VALUE1	Put the value of D_VALUE1 whose data type is DINT into CR. At this time, the data type of CR is DINT.
SUB (D_VALUE2	Keep the CR value elsewhere and put the value of D_VALUE2 with data type DINT in CR.
MUL D_VALUE3	The CR value and the value of D_VALUE3 whose data type is DINT are arithmetically multiplied and the result is put into CR.
)	Subtracts the current CR value from the CR value stored elsewhere and places the result in CR.
ST D_VALUE4	Place the CR value in the B_VALUE4 variable with a data type of DINT. ***D_VALUE4 <== D_VALUE1 - (D_VALUE2 * D_VALUE3)***

17.6.10. MUL

1) Arithmically multiplies the CR value and operand value and puts the result in CR. At this time, the data type of the CR and the data type of the operand must be the same.
2) The value of the operand does not change.
3) Modifier (: Keep the CR value somewhere else and put the value of the operand in CR (deferred operation).

Operator group	Modifier			operand
	C	N	(	
P (Process)			○	SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, REAL, LREAL data types are possible. Water is also available.

Example	Explanation
LD I_VALUE1	Put the value of I_VALUE1 whose data type is INT into CR. At this time, the data type of CR is INT.
MUL I_VALUE2	The CR value and the value of I_VALUE2 whose data type is INT are arithmetically multiplied and the result is put into CR.
ST I_VALUE3	Put CR value into I_VALUE3 variable whose data type is INT. ***I_VALUE3 <== I_VALUE1 * I_VALUE2***
LD D_VALUE1	Put the value of D_VALUE1 whose data type is DINT into CR. At this time, the data type of CR is DINT.
MUL (D_VALUE2	Keep the CR value elsewhere and put the value of D_VALUE2 with data type DINT in CR.
SUB D_VALUE3	Subtracts the CR value and the value of D_VALUE3 whose data type is DINT, and

)	inserts the result into CR. Multiply the current CR value and the CR value stored elsewhere by arithmetic, and put the result in CR.
ST D_VALUE4	Place the CR value in the B_VALUE4 variable with a data type of DINT. ***D_VALUE4 <== D_VALUE1 * (D_VALUE2 - D_VALUE3)***

17.6.11. DIV

1) The arithmetic operation is performed on the CR value and the operand value, and the quotient is put into the CR. At this time, the data type of the CR and the data type of the operand must be the same.
2) The value of the operand does not change.
3) Modifier (: Keep the CR value somewhere else and put the value of the operand in CR (deferred operation).

Operator group	Modifier			operand
	C	N	(	
P (Process)			○	SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, REAL, LREAL data types are possible. Water is also available.

Example	Explanation
LD I_VALUE1	Put the value of I_VALUE1 whose data type is INT into CR. At this time, the data type of CR is INT.
DIV I_VALUE2	The CR value and the value of I_VALUE2 whose data type is INT are subjected to arithmetic division and the result is put into CR.
ST I_VALUE3	Put CR value into I_VALUE3 variable whose data type is INT. ***I_VALUE3 <== I_VALUE1 / I_VALUE2***
LD D_VALUE1	Put the value of D_VALUE1 whose data type is DINT into CR. At this time, the data type of CR is DINT.
DIV (D_VALUE2	Keep the CR value elsewhere and put the value of D_VALUE2 with data type DINT in CR.
ADD D_VALUE3	The CR value and the value of D_VALUE3 whose data type is DINT are arithmetically added and the result is put into CR.
)	Divides the current CR value and the CR value stored elsewhere by an arithmetic operation, and puts the result into the CR.
ST D_VALUE4	Place the CR value in the B_VALUE4 variable with a data type of DINT. ***D_VALUE4 <== D_VALUE1 / (D_VALUE2 + D_VALUE3)***

17.6.12. GT

1) Compare the CR value with the operand value and put the BOOL result in CR.
2) CR is 1 only if CR is greater than operand. Otherwise, the CR value is 0.
3) The data type of CR and operand must be the same.
4) The value of the operand does not change.
5) After the operation, the data type of CR is BOOL regardless of the data type of the operand.
6) Modifier (: Keep the CR value somewhere else and put the value of the operand in CR (deferred operation).

Operator group	Modifier			operand
P (Process)	C	N	(	All data types except ARRAY are possible. Water is also available.
			○	

Example	Explanation
	Ex) I_VAL1 = 50, I_VAL2 = 100 IVAL_3 = 70
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
GT I_VAL2	Compares the value of CR with the value of I_VAL2 whose data type is INT and puts the result into CR (since I_VAL1 <I_VAL2, CR is 0)
ST B_VAL1	Put CR value into B_VAL1 variable whose data type is BOOL. ***B_VAL1 <== FALSE***
LD I_VAL2	Put the value of I_VAL2 whose data type is INT into CR.
GT I_VAL1	Compares the value of CR with the value of I_VAL1 whose data type is INT and puts the result into CR (I_VAL1 <I_VAL2, CR is 1)
ST B_VAL2	Put CR value into B_VAL2 variable whose data type is BOOL. ***B_VAL2 <== TRUE***
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
GT (I_VAL2	Keep the CR value elsewhere and put the value of I_VAL2 with data type INT in CR.
SUB I_VAL3	The value of I_VAL3 with CR value and data type INT is subtracted and the result is put into CR.
)	Compares the CR value stored elsewhere with the current CR value and puts the result into the CR (storage CR> 1 because CR is the current CR).
ST B_VAL3	Put CR value into B_VAL3 variable whose data type is BOOL. ***B_VAL3 <== TRUE***

17.6.13. GE

1) Compare the CR value with the operand value and put the BOOL result in CR.
2) If CR is greater than or equal to the operand, CR is 1. Otherwise, the CR value will be zero.
3) The data type of CR and operand must be the same.
4) The value of the operand does not change.
5) After the operation, the data type of CR is BOOL regardless of the data type of the operand.
6) Modifier (: Keep the CR value somewhere else and put the value of the operand in CR (deferred operation).

<table>
<tr><th>Operator group</th><th colspan="3">Modifier</th><th>operand</th></tr>
<tr><td rowspan="2">P (Process)</td><td>C</td><td>N</td><td>(</td><td rowspan="2">All data types except ARRAY are possible. Water is also available.</td></tr>
<tr><td></td><td></td><td>○</td></tr>
</table>

Example	Explanation
	Ex) I_VAL1 = 50, I_VAL2 = 100 IVAL_3 = 70
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
GE I_VAL2	Compares the value of CR with the value of I_VAL2 whose data type is INT and puts the result into CR (since I_VAL1 <I_VAL2, CR is 0)
ST B_VAL1	Put CR value into B_VAL1 variable whose data type is BOOL. ***B_VAL1 <== FALSE***
LD I_VAL2	Put the value of I_VAL2 whose data type is INT into CR.
GE I_VAL1	Compares the value of CR with the value of I_VAL1 whose data type is INT and puts the result into CR (I_VAL1 <I_VAL2, CR is 1)
ST B_VAL2	Put CR value into B_VAL2 variable whose data type is BOOL. ***B_VAL2 <== TRUE***
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
GE (I_VAL2	Keep the CR value elsewhere and put the value of I_VAL2 with data type INT in CR.
SUB I_VAL3	The value of I_VAL3 with CR value and data type INT is subtracted and the result is put into CR.
)	Compares the CR value stored elsewhere with the current CR value and puts the result into the CR (storage CR> 1 because CR is the current CR).
ST B_VAL3	Put CR value into B_VAL3 variable whose data type is BOOL. ***B_VAL3 <== TRUE***

17.6.14. EQ

1) Compare the CR value with the operand value and put the BOOL result in CR.
2) CR is 1 only if CR is equal to operand. Otherwise, the CR value is 0.
3) The data type of CR and operand must be the same.
4) The value of the operand does not change.
5) After the operation, the data type of CR is BOOL regardless of the data type of the operand.
6) Modifier (: Keep the CR value somewhere else and put the value of the operand in CR (deferred operation).

Operator group	Modifier			operand
P (Process)	C	N	(	All data types except ARRAY are possible. Water is also available.
			○	

Example	Explanation
	Ex) I_VAL1 = 50, I_VAL2 = 100 IVAL_3 = 50
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
EQ I_VAL2	Compares the value of CR with the value of I_VAL2 whose data type is INT and puts the result into CR (since I_VAL1 <I_VAL2, CR is 0)
ST B_VAL1	Put CR value into B_VAL1 variable whose data type is BOOL.
	B_VAL1 <== FALSE
LD I_VAL1	Put the value of I_VAL2 whose data type is INT into CR.
EQ I_VAL3	Compare the value of CR with the value of I_VAL1 whose data type is INT and put the result into CR (I_VAL1 = I_VAL3 so CR is 1)
ST B_VAL2	Put CR value into B_VAL2 variable whose data type is BOOL.
	B_VAL2 <== TRUE
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
EQ (I_VAL2	Keep the CR value elsewhere and put the value of I_VAL2 with data type INT in CR.
SUB I_VAL3	The value of I_VAL3 with CR value and data type INT is subtracted and the result is put into CR.
)	Compares the CR value stored elsewhere with the current CR value and puts the result into CR (CR = 1 because the storage CR = current CR).
	Put CR value into B_VAL3 variable whose data type is BOOL.
ST B_VAL3	***B_VAL3 <== TRUE***

17.6.15. NE

1) Compare the CR value with the operand value and put the BOOL result in CR.
2) If CR is different from the operand, CR is 1. Otherwise, the CR value is 0.
3) The data type of CR and operand must be the same.
4) The value of the operand does not change.
5) After the operation, the data type of CR is BOOL regardless of the data type of the operand.
6) Modifier (: Keep the CR value somewhere else and put the value of the operand in CR (deferred operation).

<table>
<tr><th>Operator group</th><th colspan="3">Modifier</th><th>operand</th></tr>
<tr><td rowspan="2">P (Process)</td><td>C</td><td>N</td><td>(</td><td rowspan="2">All data types except ARRAY are possible. Water is also available.</td></tr>
<tr><td></td><td></td><td>○</td></tr>
</table>

Example	Explanation
	Ex) I_VAL1 = 50, I_VAL2 = 100 IVAL_3 = 50
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
NE I_VAL3	Compare the value of CR with the value of I_VAL3 whose data type is INT and put the result into CR (I_VAL1 = I_VAL3, CR is 0)
ST B_VAL1	Put CR value into B_VAL1 variable whose data type is BOOL.
	B_VAL1 <== FALSE
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
NE I_VAL2	Compare the value of CR with the value of I_VAL2 whose data type is INT and put the result into CR (I_VAL1 <> I_VAL2, so CR is 1)
ST B_VAL2	Put CR value into B_VAL2 variable whose data type is BOOL.
	B_VAL2 <== TRUE
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
NE (I_VAL2	Keep the CR value elsewhere and put the value of I_VAL2 with data type INT in CR.
SUB I_VAL3	The value of I_VAL3 with CR value and data type INT is subtracted and the result is put into CR.
)	Compares the CR value stored elsewhere with the current CR value and puts the result into CR (CR = 0 because the storage CR = current CR)
ST B_VA3	Put CR value into B_VAL3 variable whose data type is BOOL. ***B_VAL2 <== FALSE***

17.6.16. LE

1) Compare the CR value with the operand value and put the BOOL result in CR.
2) If CR is less than or equal to the operand, CR is 1. Otherwise, the CR value will be zero.
3) The data type of CR and operand must be the same.
4) The value of the operand does not change.
5) After the operation, the data type of CR is BOOL regardless of the data type of the operand.
6) Modifier (: Keep the CR value somewhere else and put the value of the operand in CR (deferred operation).

<table>
<tr><th>Operator group</th><th colspan="3">Modifier</th><th>operand</th></tr>
<tr><td rowspan="2">P (Process)</td><td>C</td><td>N</td><td>(</td><td rowspan="2">All data types except ARRAY are possible. Water is also available.</td></tr>
<tr><td></td><td></td><td>○</td></tr>
</table>

Example	Explanation
	Ex) I_VAL1 = 50, I_VAL2 = 100 IVAL_3 = 70
LD I_VAL2	Put the value of I_VAL2 whose data type is INT into CR.
LE I_VAL1	Compare the value of CR with the value of I_VAL1 whose data type is INT and put the result into CR (I_VAL1 <I_VAL2, so CR is 0)
ST B_VAL1	Put CR value into B_VAL1 variable whose data type is BOOL.
	B_VAL1 <== FALSE
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
LE I_VAL2	Compares the value of CR with the value of I_VAL2 whose data type is INT and puts the result into CR (I_VAL1 <I_VAL2, so CR is 1)
ST B_VAL2	Put CR value into B_VAL2 variable whose data type is BOOL.
	B_VAL2 <== TRUE
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
LE (I_VAL2	Keep the CR value elsewhere and put the value of I_VAL2 with data type INT in CR.
SUB I_VAL3	The value of I_VAL3 with CR value and data type INT is subtracted and the result is put into CR.
)	Compares the CR value stored elsewhere with the current CR value and puts the result into CR (storage CR> CR is 0 because it is the current CR).
ST B_VA3	Put CR value into B_VAL3 variable whose data type is BOOL.
	B_VAL2 <== FALSE

17.6.17. LT

1) Compare the CR value with the operand value and put the BOOL result in CR.
2) CR is 1 only if CR is less than operand. Otherwise, the CR value is 0.
3) The data type of CR and operand must be the same.
4) The value of the operand does not change.
5) After the operation, the data type of CR is BOOL regardless of the data type of the operand.
6) Modifier (: Keep the CR value somewhere else and put the value of the operand in CR (deferred operation).

Operator group	Modifier			operand
P (Process)	C	N	(	All data types except ARRAY are possible. Water is also available.
			○	

Example	Explanation
	Ex) I_VAL1 = 50, I_VAL2 = 100 IVAL_3 = 70
LD I_VAL2	Put the value of I_VAL2 whose data type is INT into CR.
LT I_VAL1	Compare the value of CR with the value of I_VAL1 whose data type is INT and put the result into CR (I_VAL1 <I_VAL2, so CR is 0)
ST B_VAL1	Put CR value into B_VAL1 variable whose data type is BOOL. ***B_VAL1 <== FALSE***
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
LT I_VAL2	Compares the value of CR with the value of I_VAL2 whose data type is INT and puts the result into CR (I_VAL1 <I_VAL2, so CR is 1)
ST B_VAL2	Put CR value into B_VAL2 variable whose data type is BOOL. ***B_VAL2 <== TRUE***
LD I_VAL1	Put the value of I_VAL1 whose data type is INT into CR.
LT (I_VAL2	Keep the CR value elsewhere and put the value of I_VAL2 with data type INT in CR.
SUB I_VAL3	The value of I_VAL3 with CR value and data type INT is subtracted and the result is put into CR.
)	Compares the CR value stored elsewhere with the current CR value and puts the result into CR (storage CR> CR is 0 because it is the current CR).
ST B_VA3	Put CR value into B_VAL3 variable whose data type is BOOL. ***B_VAL2 <== FALSE***

17.6.18. JMP

1) Moves the execution flow to the label described in the operand section.
2) Modifier C: If the CR value whose data type is BOOL is TRUE (1), it moves to the label.
   If the CR value whose data type is BOOL is FALSE (0), the next command is executed without moving.
3) Modifier N: If the CR value whose data type is BOOL is FALSE (0), it moves to the label.
   If the CR value whose data type is BOOL is TRUE (1), the next instruction is executed without moving.
4) If there is no modifier, it moves to the label regardless of the CR value.

Operator group	Modifier			operand
	C	N	(	
- (Set to undefined)	○	○		Label name.

Example	Explanation
	Depending on the value of B_VAL1 whose data type is BOOL, I_VAL1 or
	It is a program that puts the value of I_VAL2 in I_VAL3.
LD B_VAL1	Put the value of B_VAL1 whose data type is BOOL into CR.
JMPC THERE1	If the CR value is 1, move to the label THERE1; if it is 0, do the following statement.
LD I_VAL1	CR <== I_VAL1
JMP THERE2	I go to the THERE2 label unconditionally.
THERE1:	THERE1 label
LD I_VAL2	CR <== I_VAL2
THERE2:	THERE2 label
ST I_VAL3	I_VAL3 <== CR
	If the value of B_VAL2 whose data type is BOOL is 1, it executes the SEL function.
	CR <== B_VAL2
LD B_VAL2	If CR is 0 (FALSE), it moves to the label THERE3.
JMPN THERE3	CR <== B_VALUE
LD B_VALUE	Invokes the SEL function.
SEL (	
(* G: = CR, *)	
IN1: = I_VAL1	
IN2: = I_VAL2	
)	I_VAL3 <== CR
ST I_VAL3	THERE3 label
THERE3:	

17.6.19. CAL

1) The function block with the name described in the operand part is called.
2) Modifier C: If the CR value whose data type is BOOL is TRUE (1), the function block is called.
   If the CR value whose data type is BOOL is FALSE (0), the function block is not called.
3) Modifier N: If the CR value whose data type is BOOL is FALSE (0), the function block is called.
   If the CR value whose data type is BOOL is TRUE (1), the function block is not called.
4) If there is no modifier, the function block is called irrespective of the CR value.

<table>
<tr><th rowspan="1">Operator group</th><th colspan="3">Modifier</th><th>operand</th></tr>
<tr><td rowspan="2">- (Set to undefined)</td><td>C</td><td>N</td><td>(</td><td rowspan="2">Function name without function block name or return value</td></tr>
<tr><td>○</td><td>○</td><td></td></tr>
</table>

Example	Explanation
LD B_VAL1 CALC INST_TON (IN: = T_INPUT PT: = PRE_TIME)	**When the value of B_VAL1 whose data type is BOOL is 1 (TRUE), it is a program that calls on-delay timer TON.** Put the value of B_VAL1 whose data type is BOOL into CR. If the CR value is 1, the instance calls the on-delay timer TON with INST_TON.
LD B_VAL2 CALN INST_CTU (CU: = B_UP R: = B_RESET PV: = I_VAL1)	**If the value of B_VAL2 whose data type is BOOL is 0 (FALSE), it is a program that calls up counter CTU_INT.** Put the value of B_VAL2 whose data type is BOOL into CR. If the CR value is [0], the up counter CTU_INT whose instance is INST_CTU is called.
LD B_VAL1 CAL XCHG ((* SRC1: = CR, *) SRC2: = B_VAL2)	**It is a program to call data exchange XCHG function unconditionally regardless of CR value.** Invokes XCHG, a function with no return value.

17.6.20. RET

1) Return from function or function block.
2) Modifier C: If the CR value whose data type is BOOL is TRUE (1), it returns.
   If the CR value whose data type is BOOL is FALSE (0), it does not return.
3) Modifier N: If the CR value whose data type is BOOL is FALSE (0), it returns.
   If the CR value whose data type is BOOL is TRUE (1), it does not return.
4) If there is no modifier, it returns regardless of the CR value.

Operator group	Modifier			operand
	C	N	(	
- (Set to undefined)	○	○		There is not.

Example	Explanation
	This function multiplies the value of I_VAL1 whose data type is INT by the value of I_VAL2 whose data type is INT and puts the result into I_VAL3. In this case, if an operation error occurs in the multiply operation, 0 is returned to I_VAL3.
LD I_VAL1	
MUL I_VAL2	
ST I_VAL3	
LD _ERR	CR <== System error flag
RETN	If the CR value is 0, the instance returns.
LD 0	I_VAL3 <== 0
ST I_VAL3	
RET	I will return unconditionally.

17.6.21. SCAL

1) Calls a subroutine with the name described in the operand section.
2) Modifier C: If the CR value whose data type is BOOL is TRUE (1), the subroutine is called.
   If the CR value whose data type is BOOL is FALSE (0), the subroutine is not called.
3) Modifier N: If the CR value whose data type is BOOL is FALSE (0), it calls the subroutine.
   If the CR value whose data type is BOOL is TRUE (1), the subroutine is not called.
4) If there is no modifier, the subroutine is called regardless of the CR value.

Operator group	Modifier			operand
	C	N	(	
- (Set to undefined)	○	○		Subroutine name.

Example	Explanation
	If the value of B_VAL1 whose data type is BOOL is 1 (TRUE), it is a program that calls subroutine SBRT1.
LD B_VAL1	Put the value of B_VAL1 whose data type is BOOL into CR.
SCALC SBRT1	If the CR value is 1, the subroutine SBRT1 is called.
	If the value of B_VAL2 whose data type is BOOL is 0 (FALSE), it is a program that calls subroutine SBRT2.
LD B_VAL2	Put the value of B_VAL2 whose data type is BOOL into CR.
SCALN SBRT1	If the CR value is 0, the subroutine SBRT0 is called.
	It is a program that calls subroutine SBRT1 unconditionally regardless of CR value.
SCAL SBRT1	Calls subroutine SBRT1.
END_PROGRAM	
SBRT SBRT1	
LD B_VAL1	Declare SBRT1.
ST B_VAL2	
RET	
	Return to RET.

Note
A subroutine (SBRT) can declare the subroutine name after END_PROGRAM and define its contents. The subroutine returns via the RET command.

17.6.22.)

1) Use '(' to perform deferred operations.

Operator group	Modifier			operand
U (Leave unchanged)	C	N	(	There is not.

Example	Explanation
LD I_VAL1 ADD I_VAL2 MUL I_VAL3 ST I_VAL4	**I_VAL4 <== (I_VAL1 + IVAL2) * I_VAL3**

LD I_VAL1 ADD (I_VAL2 MUL I_VAL3) ST I_VAL4	**I_VAL4 <== I_VAL1 + (IVAL2 * I_VAL3)**
LD L_VAL1 ADD (L_VAL2 MUL (L_VAL3 SUB L_VAL4) ADD L_VAL5) DIV L_VAL6 ST L_VAL7	**L_VAL7 <== (L_VAL1 + (L_VAL2 * (L_VAL3 - L_VAL4) + L_VAL5)) / L_VAL6**

Note
There can not be JMP, CAL, RET, SCAL, or label between the parentheses '(' and ')'. You can call parentheses back inside parentheses. The maximum depth for this is 32, including the top-level body.

17.7. Non-executable statements (comments)

1) Non-executable statements (comments) provide two forms. There are two types of non-executing statements and non-executing statements.
2) One line non-executable statement uses "//" and is executed until the end of the line.
3) Block non-executable statements process non-executable characters between "(*" and "*)".

Yes)

```
// One line comment

(* Multiple
    line comment
*)
```

17.8. Function and function block

17.8.1. Function

1) The function is called with the function name as an operator.
2) When calling a function, the CR enters the first input of the function
3) If there is more than one input of the function, specify the remaining input values and call the function.
4) The output value of the function enters CR.
5) The data type of CR is the output value data type of the function.

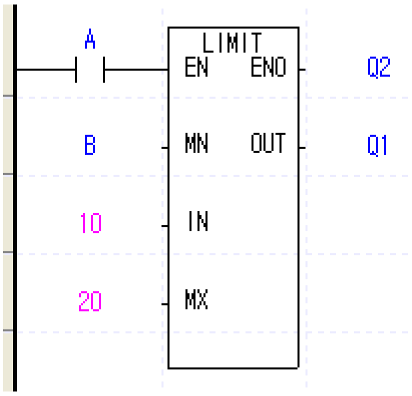
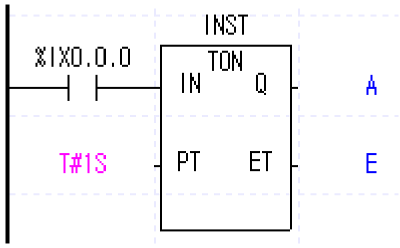
Example	Explanation
LD R_VAL1	Put the value of R_VAL1 (REAL) variable in CR.
SIN	When the SIN function is called, the CR at that time enters the first input of the SIN function. Since the SIN function has only one input, the input value is no longer required. After executing the SIN function, the output value is assigned to CR.
ST R_VAL2	The CR is stored in the R_VAL2 (REAL) variable.
LD% IX0.0.0	**An example of a function with multiple inputs**. % IX0.0.0 (BOOL) has been set on the CR.
SEL ((* G: = CR (BOOL), *) IN0: = VAL1, IN1: = VAL1)	The CR is entered as the first input value of the SEL function. For the rest of the inputs, set the value and set the SEL function If you call it, the result of the execution is also input to the CR.
ST VAL3	Store the CR in the VAL3 variable.

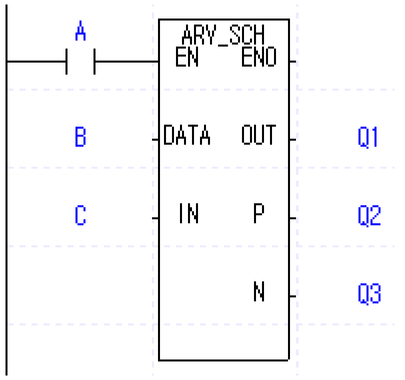
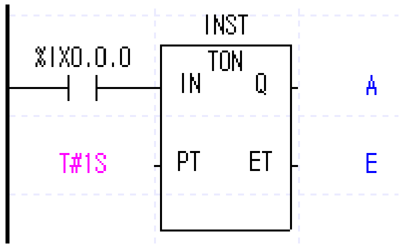
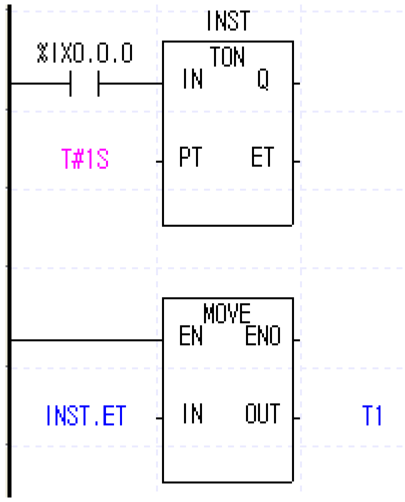
17.8.2. Function block

1) The call to the function block uses the CAL operator and the operand is the instance name of the function block declared in advance.
2) Function block does not enter CR as input of function block. Therefore, all necessary input values must be specified in the function block. Also, the output value is not displayed as CR.
3) Can not be used between '(' and ')' modifiers.
4) Please refer to 15.6.19 Example of CAL for the function block calling method using CAL.

17.8.3. Stereotyped form

1) There are two types of function and function block input methods: formal and non-formalized. Either form can be used depending on the situation.
2) Formalization type is a form to display the input and output parameter names of function and function block.

parameter	Function	Function block
common	Parameter order can be used in any order. LD B LIMIT (MX: = 20, IN: = 10) LIMIT (IN: = 10, MX: = 20) EN, ENO can be used or omitted LD B LIMIT (**EN: = A,** MX: = 20, IN: = 10, ENO = & gt; Q2) ST Q1 	Parameter order can be used in any order. INST (IN: =% IX0.0.0, PT: = T # 1s, Q => A, ET => E) INST (PT: = T # 1s, IN: = IX0.0.0, Q => A, ET => E) 
input	Input: Use = symbol for input / output parameter assignment. LIMIT (**MX: = 20**, IN: = 10)	Input: Use = symbol for input / output parameter assignment. INST (**IN: =% IX0.0.0, PT: = T # 1s**, Q = & gt; A, ET = & gt; B)
Print	If the output parameter name is OUT or Y (user defined function is function name), return value is assigned. The remaining output parameter assignments use the => symbol. LD B	Use => symbol to assign all output parameters Output parameter assignment can be omitted. INST (IN: =% IX0.0.0, PT: = T # 1s, **Q => A, ET => E**)

parameter	Function	Function block
	ARY_SCH (IN: = C, **P => Q2, N => Q3**)  However, output parameters that are not used as shown below can be omitted. (Q2 and Q3 are omitted) LD B ARY_SCH (IN: = C)	 INST (IN: =% IX0.0.0, PT: = T # 1s) LD INST.ET ST T1 

Note

Use the function block as the instance name. That is, declare a function block as a variable and set the variable name (instance name)

Should be used.

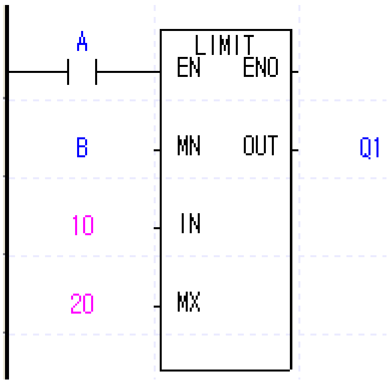
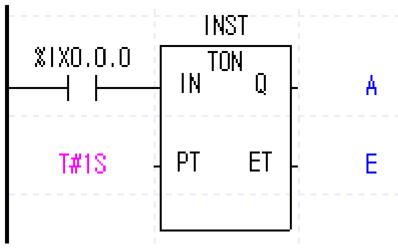
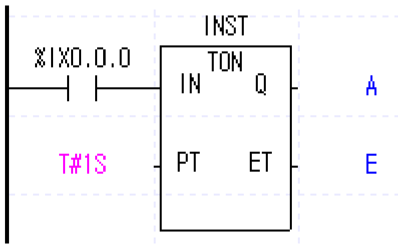
Example: Using a timer

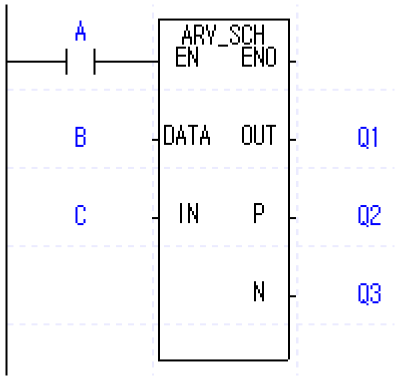
	Variable Kind	Variable	Type
1	VAR	INST_TON1	TON

INST_TON1(IN: = TRUE, PT: = T # 100MS, Q => Q_OUT, ET => ET_OUT)

17.8.4. Nonformatted form

1) It is a form to omit input and output parameter names of function and function block.

parameter	Function	Function block
common	All parameter sequences can not be changed. All parameters are not omissible LD B LIMIT (20, 10) ST Q1  EN, ENO can not be used.	All parameter sequences can not be changed. All parameters can not be omitted. INST (% IX0.0.0, T # 1s, A, E) 
input	Input parameter order can not be changed. LD B LIMIT (**20**, IN: = 10)	Input parameter order can not be changed. INST (**% IX0.0.0, T # 1s**, A, E)
Print	Assign the return value to CR when the output parameter name is OUT or Y (user-defined function is function name). The remaining output parameter assignments are entered in order. LD B ARY_SCH (C, **Q2, Q3**) ST **Q1**	All output parameter assignments are entered in order. INST (% IX0.0.0, T # 1s, **A, E**) 

Note

Functions with variable parameter types are not supported by IL.

To operate normally, enter one of the following methods.

Example	Explanation
LD **INT # 1** ADD 2	You can set the type to a constant.
LD **INT_VAL** ADD 2	Variable (INT_VAL) can be used.
LD 1 **ADD_INT**(2)	You can use the type-set function.

Note

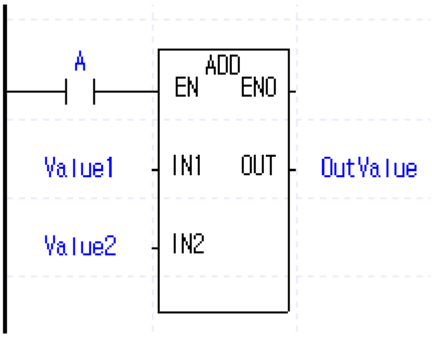
1. Input parameter EN is a condition for executing the function. If EN is used as follows, the value of A is 1 day Only the LIMTIT function is executed.
   LD B
   LIMIT (EN: = A, MX: = 20, IN: = 10)
   ST OUT
2. The ENO parameter is set to 1 when the function is executed without error.

Note

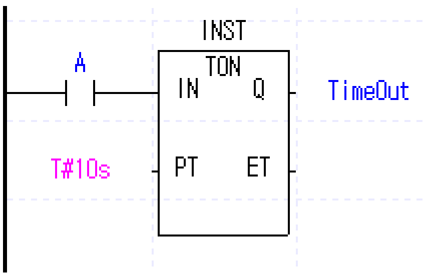
1. IL does not support extended instructions (BREAK, CALL, END, FOR, INIT_DONE, JMP, NEXT, RET, SBRT) but supports JMP, RET and SBRT in operators.
2. A function with the same name as an operator name can not be used (ADD, OR, XOR, AND, GT, etc.)

17.8.5. Example

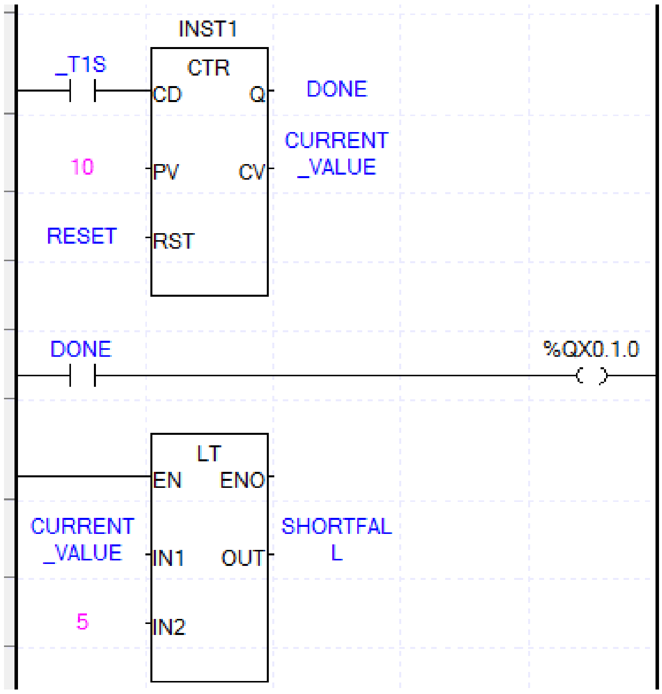
1) Function

LD example	Examples of using IL
	1) Typical form Using EN LD Value1 ADD (EN: = A, IN2: = Value2) ST OutValue Disable EN LD Value1 ADD (IN2: = Value2); ST OutValue 2) Unstructured form LD Value1 ADD (Value2) ST OutValue EN, ENO can not be used.

2) Function block

LD example	Examples of using IL
	1) Typical form INST (IN: = A, PT: = T # 10S, Q => TimeOut) 2) Unstructured form INST (A, T # 10S, TimeOut, TimeValue) Output variables can not be omitted. Therefore, it is necessary to connect the variable corresponding to the output parameter ET. (TimeValue)

3) Application

<table>
<tr><th>LD example</th><th>Examples of using IL</th></tr>
<tr><td>
</td><td>INST1 (CD: = _T1S, PV: = 10, RST: = reset, Q => completed,
CV => current value)

LD completed
ST% QX0.1.0

LD current value
LT (IN2: = 5)
ST under</td></tr>
</table>

Appendix 1 Numerical System and Data Structure

A1.1 Numerical (data) Representation

PLC CPU remembers and processes every data as the states of on and off or ‘1’and ’0’. Therefore, any numerical operation is processed by binary system (1 or 0). On the other hand, we conveniently use the decimal system, so decimal or hexadecimal number systems must be converted to hexadecimal or decimal number systems, respectively in order to write or read numerical data to/from PLC. This chapter describes the representation of decimal, binary, hexadecimal and binary-coded decimal notation and the relations.

1) Decimal

Decimal number system means the “number expressing an order or size (volume) using 0~9. And, followed by 0, 1, 2, 3, 4…9, it is carried to ‘10’ and keeps counting. For instance, a decimal number, 153 can be expressed as follows in the view of line and “weight of line.”

$$153 = 100 + 50 + 3$$

$$= 1*100 + 5*10 + 3*1$$

$$= \underline{1}*\underline{10}^2 + \underline{5}*\underline{10}^1 + \underline{3}*\underline{10}^0$$

Decimal number system symbols (0~9)

Weight of line

2) Binary

Binary numeral presents a numeral meaning an order and size by using two symbols, 0 and 1. Therefore, it is carried to ‘10’ followed by 0 and 1 and keeps counting. That is, a cipher of 0, 1 is called bit.

Binary	Decimal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
……	……

For instance, let us think that the given binary numeral can be expressed in decimal number system.

"10011101"

As considering line number and the weight of line in decimal number system, try to attach bit number and bit weight from the very right.

7	6	5	4	3	2	1	0	← Bit number
1	0	0	1	1	1	0	1	binary numeral
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
128	64	32	16	8	4	2	1	weight of bit

How about summing the multiplication of weights of each bit code like decimal number system?

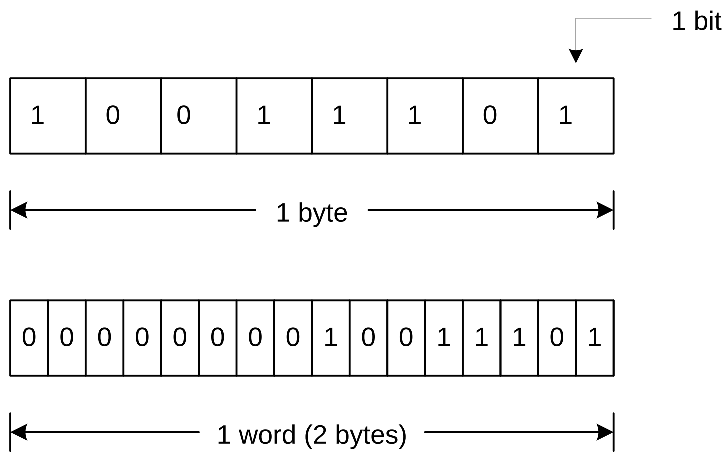
$= 1 \times 128 + 0 \times 64 + 0 \times 32 + 1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$

$= 128 + 16 + 8 + 4 + 1$

$= 157$

That is, as the above, a binary numeral is converted to a decimal numeral by adding the weights of bits of which code is 1.

In general, 1 byte consists of 8 bits while 1 word consists of 16 bits (2 bytes).

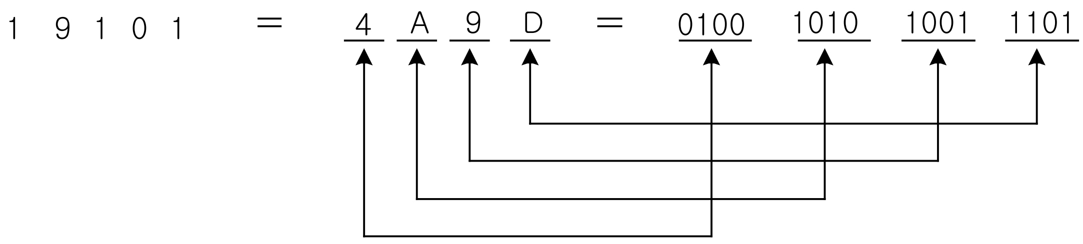


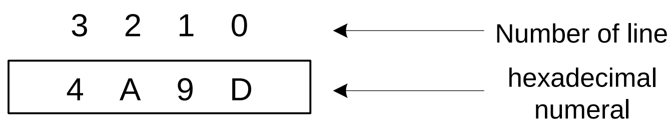
3) Hexadecimal

Like decimal or binary numeral, hexadecimal numeral means the 'number representing an order and size by using 0~9 and A~F."

Then, followed by 0, 1, 2, …D, E, F, it is carried to '10' and keeps counting.

Decimal	Hexadecimal	Binary
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
18	12	10010





= (4) × 163 + (A) × 162 + (9) × 161 + (D) × 160

= 4 × 4096 + 10 × 2568 + 9 × 16 + 13 × 1

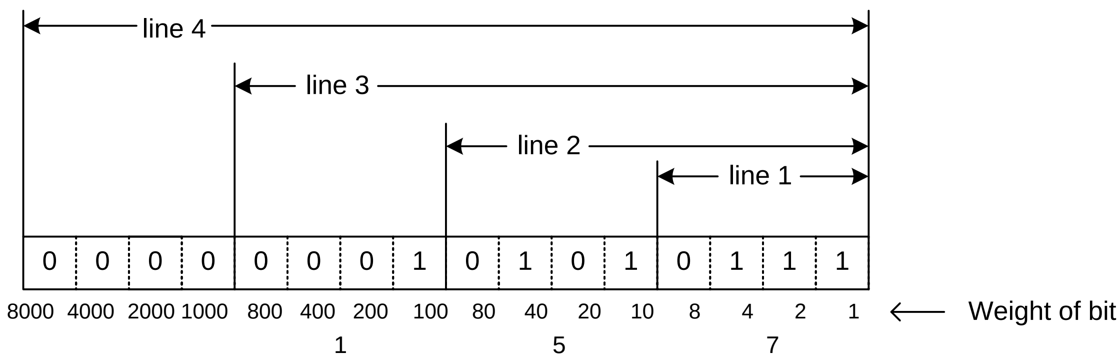
= 19101

A digit of hexadecimal number corresponds to 4 bits of binary numeral.

4) Binary Coded Decimal (BCD)

Binary coded decimal means the “number expressing each line of a decimal numeral in binary number system.” Therefore, binary coded decimal represents 0 ~ 9,999(max of 4 lines) of decimal numeral in 16 bits.

For instance, a decimal numeral, 157 can be expressed as follows and the weight of each bit can be also expressed as follows.



5) Table of Numeral Systems

Binary coded Decimal (BCD)	Binary (BIN)	Decimal	Hexadecimal (H)
00000000 00000000	00000000 00000000	0	0000
00000000 00000001	00000000 00000001	1	0001
00000000 00000010	00000000 00000010	2	0002
00000000 00000011	00000000 00000011	3	0003
00000000 00000100	00000000 00000100	4	0004
00000000 00000101	00000000 00000101	5	0005
00000000 00000100	00000000 00000100	6	0006
00000000 00000111	00000000 00000111	7	0007
00000000 00001000	00000000 00001000	8	0008
00000000 00001001	00000000 00001001	9	0009
00000000 00010000	00000000 00001010	10	000A
00000000 00010001	00000000 00001011	11	000B
00000000 00010010	00000000 00001100	12	000C
00000000 00010011	00000000 00001101	13	000D
00000000 00010100	00000000 00001110	14	000E
00000000 00010101	00000000 00001111	15	000F
00000000 00000110	00000000 00010000	16	0010
00000000 00000111	00000000 00010001	17	0011
00000000 00001000	00000000 00010010	18	0012
00000000 00001001	00000000 00010011	19	0013
00000000 00100000	00000000 00010100	20	0014
00000000 00100001	00000000 00010101	21	0015
00000000 00100010	00000000 00010110	22	0016
00000000 00100011	00000000 00010111	23	0017
00000001 00000000	00000000 01100100	100	0064
00000001 00100111	00000000 01111111	127	007F
00000010 01010101	00000000 11111111	255	00FF
00010000 00000000	00000000 11100000	1,000	03E8
00100000 01000111	00000000 11111111	2,047	07FF
01000000 10010101	00000000 11111111	4,095	0FFF
10011001 10011001	00000111 00001111	9,999	270F
	00100111 00010000	10,000	2710
	01111111 11111111	32,767	7FFF

A1.2 Integer Representation

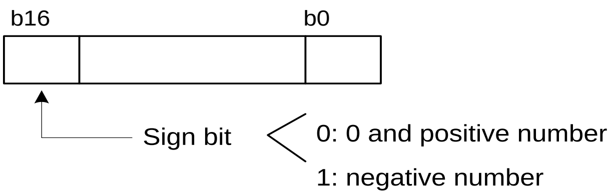
XGI command is based on negative number system operation (Signed)

If the top level bit (MSB) is 0, it represents 'positive number' while if it is 1, it is expressed as 'negative number'.

The top level bit expressing negative/positive is called 'sign bit.'

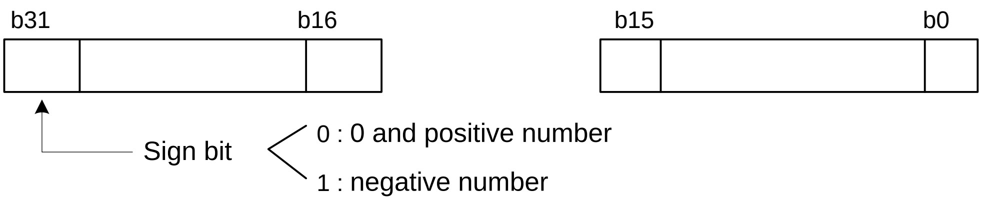
Because of different position of MSB in 16 or 32 bits, be cautious of sign bit position.

✷ If 16 bits



Representation ran: -32,768 ~ 32,767

✷ If 32 bits



Representation range: -2,147,483,648 ~ 2,147,483,647

A1.3 Negative Number Representation

Ex) How to express – 0001

(1) Represent 0001 in case of negative number (b15 = 1).

b15				b0
1	0	~	0	1

(2) Reverse the result of (1) (b15 = excluded).

b15				b0
1	1	~	1	0

(3) Plus 1 to the result of (2).

b15							b0
1	1				~	1	1

-0001 = 16#FFFF

Appendix 2 Flag List (XGI)

A2.1 Modes and Status

Reserved Variable	Data Type	Description
_SYS_STATE	BOOL	PLC mode and operation status
_RUN	BOOL	RUN status
_STOP	BOOL	STOP status
_ERROR	BOOL	ERROR status
_DEBUG	BOOL	DEBUG status
_LOCAL_CON	BOOL	Local control mode
_REMOTE_CON	BOOL	Remote control mode
_RUN_EDIT_ST	BOOL	Downloading edit program during run
_RUN_EDIT_CHK	BOOL	Processing edit program during run
_RUN_EDIT_DONE	BOOL	Complete edit program during run
_RUN_EDIT_NG	BOOL	Abnormally complete edit program during run
_CMOD_KEY	BOOL	Run mode changed by key
_CMOD_LPADT	BOOL	Run mode changed by local PADT
_CMOD_RPADT	BOOL	Run mode changed by remote PADT
_CMOD_RLINK	BOOL	Run mode changed by remote COM module
_FORCE_IN	BOOL	Forced input status
_FORCE_OUT	BOOL	Forced output status
_SKIP_ON	BOOL	I/O skip
_EMASK_ON	BOOL	Error mask on
_MON_ON	BOOL	Monitor on
_USTOP_ON	BOOL	Stop by STOP function
_ESTOP_ON	BOOL	Stop by ESTOP function
_INIT_RUN	BOOL	Initialization task is running
_PB1	BOOL	Select program code 1
_PB2	BOOL	Select program code 2
_USER_WRITE_F	WORD	Contact available by program
_RTC_WR	BOOL	Write/read data in RTC
_SCAN_WR	BOOL	Initialize scan value
_CHK_ANC_ERR	BOOL	Error detection from external device
_CHK_ANC_WAR	BOOL	Warning detection from external device

Reserved Variable	Data Type	Description
_INIT_DONE	BOOL	Initialization task complete
_KEY	DWORD	Current status of local key

A2.2 System Error

Reserved Variable	Data Type	Description
_CNF_ER	WORD	System warning
_AB_SD_ER	BOOL	Stop by abnormal operation
_IO_TYER	BOOL	Module type inconsistence error
_IO_DEER	BOOL	Module installation error
_IO_TYER_N	WORD	Slot number of module type inconsistence error
_IO_DEER_N	WORD	Slot number of module installation error
_FUSE_ER	BOOL	Fuse disconnection
_FUSE_ER_N	WORD	Slot number of fuse blown
_FUSE_ERR	ARRARY [0..7] OF WORD	Detail information of fuse blown (base and slot number)
_ANNUM_ER	BOOL	Heavy trouble detection error of external device
_BPRM_ER	BOOL	Basic parameter error
_IOPRM_ER	BOOL	IO configuration parameter error
_SPPRM_ER	BOOL	Special module parameter error
_CPPRM_ER	BOOL	Communication module parameter error
_PGM_ER	BOOL	Program error
_CODE_ER	BOOL	Program code error
_SWDT_ER	BOOL	System watch-dog on
_BASE_POWER_ER	BOOL	Base power error
_WDT_ER	BOOL	Scan watch-dog timer on
_IO_TYERR	ARRARY [0..7] OF WORD	Main base and extension base module type error
_IO_DEERR	ARRARY [0..7] OF WORD	Main base and extension base module installation error

A2.3 System Warning

Reserved Variable	Data Type	Description
_CNF_WAR	DWORD	System error status
_RTC_ER	BOOL	RTC data error
_TASK_ER	BOOL	Task conflict
_BAT_ER	BOOL	Battery error
_ANNUM_WAR	BOOL	External device warning detected
_BASE_INFO_ER	BOOL	Base information error
_HS_WAR1	BOOL	Over high-speed link parameter 1
_HS_WAR2	BOOL	Over high-speed link parameter 2
_HS_WAR3	BOOL	Over high-speed link parameter 3
_HS_WAR4	BOOL	Over high-speed link parameter 4
_HS_WAR5	BOOL	Over high-speed link parameter 5
_HS_WAR6	BOOL	Over high-speed link parameter 6
_HS_WAR7	BOOL	Over high-speed link parameter 7
_HS_WAR8	BOOL	Over high-speed link parameter 8
_HS_WAR9	BOOL	Over high-speed link parameter 9
_HS_WAR10	BOOL	Over high-speed link parameter 10
_HS_WAR11	BOOL	Over high-speed link parameter 11
_HS_WAR12	BOOL	Over high-speed link parameter 12
_P2P_WAR1	BOOL	Over P2P – parameter 1
_P2P_WAR2	BOOL	Over P2P – parameter 2
_P2P_WAR3	BOOL	Over P2P – parameter 3
_P2P_WAR4	BOOL	Over P2P – parameter 4
_P2P_WAR5	BOOL	Over P2P – parameter 5
_P2P_WAR6	BOOL	Over P2P – parameter 6
_P2P_WAR7	BOOL	Over P2P – parameter 7
_P2P_WAR8	BOOL	Over P2P – parameter 8
_CONSTANT_ER	BOOL	Fixed cycle error
_ANC_ERR	WORD	Error info of external device
_ANC_WAR	WORD	Warning info of external device

A2.4 User Flag

Reserved Variable	Data Type	Description
_T20MS	BOOL	20ms cycle clock
_T100MS	BOOL	100ms cycle clock
_T200MS	BOOL	200ms cycle clock
_T1S	BOOL	1s cycle clock
_T2S	BOOL	2s cycle clock
_T10S	BOOL	10s cycle clock
_T20S	BOOL	20s cycle clock
_T60S	BOOL	60s cycle clock
_ON	BOOL	All time on bit
_OFF	BOOL	All time off bit
_1ON	BOOL	The only first scan on bit
_1OFF	BOOL	The only first scan off bit
_STOG	BOOL	Reversal at every scanning

A2.5 Operation Result Flag

Reserved Variable	Data Type	Description
_ERR	BOOL	Operation error flag
_LER	BOOL	On for 1 scan if any operation error
_ARY_IDX_ERR	BOOL	Out of arrangement index error flag
_ARY_IDX_LER	BOOL	Out of arrangement index latch error flag
_ALL_OFF	BOOL	On if every output is off
_PUTGET_ERR	WORD	PUT/GET error
_PUTGET_NDR	WORD	PUT/GET complete

A2.6 System Run Status Information

Reserved Variable	Data Type	Description
_CPU_TYPE	WORD	CPU type information
_CPU_VER	WORD	CPU version
_OS_VER	DWORD	OS version
_OS_DATE	DWORD	OS distribution date
_SCAN_MAX	WORD	Max. scan time after run in 0.1ms
_SCAN_MIN	WORD	Min. scan time after run in 0.1ms
_SCAN_CUR	WORD	Present scan time in 0.1ms
_RTC_TIME	ARRARY [0..7] OF BYTE	Present time data of PLC
_RTC_TIME[0]	BYTE	Year data of present time
_RTC_TIME[1]	BYTE	Month data of present time
_RTC_TIME[2]	BYTE	Day data of present time
_RTC_TIME[3]	BYTE	Hour data of present time
_RTC_TIME[4]	BYTE	Minute data of present time
_RTC_TIME[5]	BYTE	Second data of present time
_RTC_TIME[6]	BYTE	Day of the week data of present time
_RTC_TIME[7]	BYTE	Year of hundred data of present time
_RTC_TIME_USER	ARRARY [0..7] OF BYTE	Time data to set
_RTC_TIME_USER[0]	BYTE	Year data of time to set
_RTC_TIME_USER[1]	BYTE	Month data of time to set
_RTC_TIME_USER[2]	BYTE	Day data of time to set
_RTC_TIME_USER[3]	BYTE	Hour data of time to set
_RTC_TIME_USER[4]	BYTE	Minute data of time to set
_RTC_TIME_USER[5]	BYTE	Second data of time to set
_RTC_TIME_USER[6]	BYTE	Day of the week data of time to set
_RTC_TIME_USER[7]	BYTE	Year of hundred data of time to set
_RTC_DATE	WORD	Present data of RTC
_RTC_WEEK	WORD	Present a day of the week of RTC
_RTC_TOD	DWORD	Present time of RTC (ms unit)
_BASE_INFO	ARRARY [0..7] OF WORD	Slot information of main and extension base

Reserved Variable	Data Type	Description
_RBANK_NUM	WORD	Block number currently used
_AC_F_CNT	WORD	Instantaneous AC failure frequency
_FALS_NUM	WORD	FALS number

A2.7 High-speed Link Flag (* = 0 ~ 12, *** = 000 ~ 127)

Reserved Variable	Data Type	Description
_HS*_RLINK	BOOL	Every station of high speed link no.* normally works
_HS*_LTRBL	BOOL	Abnormal status after _HS*RLINK on
_HS*_STATE***	BOOL	General status of *** block of high speed link no.*
_HS*_MOD***	BOOL	Run operation mode of *** block of high speed link no.*
_HS*_TRX***	BOOL	Normal communication with *** block station of high speed link no.*
_HS*_ERR***	BOOL	Run error mode of *** block station of high speed link no.*
_HS*_SETBLOCK***	BOOL	*** block setting of high speed link no.*

A2.8 P2P Flag (* = 0 ~ 8, ** = 0 ~ 63)

Reserved Variable	Data Type	Description
_P2P*_NDR**	BOOL	** block service of P2P no.* completed successfully
_P2P*_ERR**	BOOL	** block service of P2P no.* completed abnormally
_P2P*_STATUS**	WORD	Error code in case of ** block service of P2P no.*
_P2P*_SVCCNT**	DWORD	** block normal service frequency of P2P no.*
_P2P*_ERRCNT**	DWORD	** block abnormal service frequency of P2P no.*

A2.9 PID Flag (* = 0 ~ 7, ** = 0 ~ 31)

Reserved Variable	Data Type	Description
_PID*_MAN	DWORD	PID output selection(0:auto ,1:manual) – block*
PID***MAN	BOOL	PID output selection(0:auto ,1:manual) - block* loop**
_PID*_PAUSE	DWORD	PID pause (0:STOP/RUN ,1:PAUSE) – block*
PID***PAUSE	BOOL	PID pause (0:STOP/RUN ,1:PAUSE) – block* loop**
_PID*_REV	DWORD	PID operation selection(0:forward ,1:reverse) – block*
PID***REV	BOOL	PID operation selection(0:forward ,1:reverse) – block* loop**
_PID*_AW2D	DWORD	PID Anti Wind-up2 prohibited(0:enable ,1:disable) – block*
PID***AW2D	BOOL	PID Anti Wind-up2 prohibited(0:enable ,1:disable) – block* loop**
_PID*_REM_RUN	DWORD	PID remote(HMI) execution bit (0:STOP ,1:RUN) – block*
PID***REM_RUN	DWORD	PID remote(HMI) execution bit (0:STOP ,1:RUN) – block* loop**
_PID*_P_on_PV	DWORD	PID proportional(P) cal source selection (0:ERR, 1:PV) – block*
PID***P_on_PV	BOOL	PID proportional(P) cal source selection (0:ERR, 1:PV) - block* loop**
_PID*_D_on_ERR	DWORD	PID differential(D) cal source selection (0:PV, 1:ERR) – block*

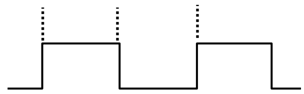
Reserved Variable	Data Type	Description
PID***D_on_ERR	BOOL	PID differential(D) cal source selection (0:PV, 1:ERR) - block* loop**
_PID*_AT_EN	DWORD	PID auto tuning setting (0:Disable, 1:Enable) – block*
PID***AT_EN	BOOL	PID auto tuning setting (0:Disable, 1:Enable) – block* loop**
_PID*_MV_BMPL	DWORD	PID mode change(A/M) - MV no impact change setting (0:Disable, 1:Enable) – block*
PID***MV_BMPL	BOOL	PID mode change(A/M) - MV smoothing setting (0:Disable, 1:Enable) – block* loop**
PID***SV	INT	PID target value (SV) – block* loop**
PID***T_s	WORD	PID operation cycle (T_s)[0.1ms] – block* loop**
PID***K_p	REAL	PID P - constant (K_p) – block* loop**
PID***T_i	REAL	PID I - constant (T_i)[sec] – block* loop**
PID***T_d	REAL	PID D - constant (T_d)[sec] – block* loop**
PID***d_PV_max	WORD	PID PV variation limit – block* loop**
PID***d_MV_max	WORD	PID MV variation limit – block* loop**
PID***MV_max	INT	PID MV max. value limit – block* loop**
PID***MV_min	INT	PID MV min. value limit – block* loop**
PID***MV_man	INT	PID manual output (MV_man) – block* loop**
PID***STATE	WORD	PID State – block* loop**
PID***ALARM0	BOOL	PID Alarm 0 (1:T_s setting is low) – block* loop**
PID***ALARM1	BOOL	PID Alarm 1 (1:K_p is 0) – block* loop**
PID***ALARM2	BOOL	PID Alarm 2 (1:PV variation is limited) – block* loop**
PID***ALARM3	BOOL	PID Alarm 3 (1:MV variation is limited) – block* loop**
PID***ALARM4	BOOL	PID Alarm 4 (1:MV max. value is limited) – block* loop**
PID***ALARM5	BOOL	PID Alarm 5 (1:MV min. value is limited) – block* loop**
PID***ALARM6	BOOL	PID Alarm 6 (1:AT abnormal cancellation) – block* loop**
PID***ALARM7	BOOL	PID Alarm 7 – block* loop**
PID***STATE0	BOOL	PID State 0 (0:PID_STOP, 1:PID_RUN) – block* loop**
PID***STATE1	BOOL	PID State 1 (0:AT_STOP, 1:AT_RUN) – block* loop**
PID***STATE2	BOOL	PID State 2 (0:AT_UNDONE, 1:DONE) – block* loop**
PID***STATE3	BOOL	PID State 3 (0:REM_STOP, 1:REM_RUN) – block* loop**
PID***STATE4	BOOL	PID State 4 (0:AUTO_OUT, 1:MAN_OUT) – block* loop**
PID***STATE5	BOOL	PID State 5 (0:CAS_STOP, 1:CAS_RUN) – block* loop**
PID***STATE6	BOOL	PID State 6 (0:SLV/SINGLE, 1:CAS_MST) – block* loop**
PID***STATE7	BOOL	PID State 7 (0:AW_STOP, 1:AW_ACT) – block* loop**

Reserved Variable	Data Type	Description
PID***PV	INT	PID present value (PV) – block* loop**
PID***PV_old	INT	PID previous value (PV_old) – block* loop**
PID***MV	INT	PID output value (MV) – block* loop**
PID***MV_BMPL_val	INT	PID no impact operation memory (user setting prohibited) – block* loop**
PID***ERR	DINT	PID control error value – block* loop**
PID***MV_p	REAL	PID output P element – block* loop**
PID***MV_i	REAL	PID output I element – block* loop**
PID***MV_d	REAL	PID output D element – block* loop**
PID***DB_W	WORD	PID deadband setting (operation after stabilization) – block* loop**
PID***Td_lag	WORD	PID differential function LAG filter – block* loop**
PID***AT_HYS_val	WORD	PID auto tuning hysteresis setting – block* loop**
PID***AT_SV	INT	PID auto tuning SV setting – block* loop**
PID***AT_step	WORD	PID auto tuning status (user setting prohibited) – block* loop**
PID***INT_MEM	WORD	PID internal memory (user setting prohibited) – block* loop**

Appendix 3 Flag list (XGR)

Appendix 3.1 User Flag

1. User flag

Address	Flag name	Type	Writable	Contents	Description
%FX6144	_T20MS	BOOL	-	20ms cycle clock	Clock signal used in user program reverses on/off per half cycle Use more enough long clock signal than PLC scan time. Clock signal starts from off condition when initialization program starts or scan program starts. _T100ms clock example 50ms 50ms 
%FX6145	_T100MS	BOOL	-	100ms cycle clock	
%FX6146	_T200MS	BOOL	-	200ms cycle clock	
%FX6147	_T1S	BOOL	-	1s cycle clock	
%FX6148	_T2S	BOOL	-	2s cycle clock	
%FX6149	_T10S	BOOL	-	10s cycle clock	
%FX6150	_T20S	BOOL	-	20s cycle clock	
%FX6151	_T60S	BOOL	-	60s cycle clock	
%FX6153	_ON	BOOL	-	Ordinary time On	Always on state flag, used when writing user program.
%FX6154	_OFF	BOOL	-	Ordinary time Off	Always off state flag, used when writing user program.
%FX6155	_1ON	BOOL	-	1'st scan On	Only first scan on after operation start
%FX6156	_1OFF	BOOL	-	1'st scan Off	Only first scan off after operation start
%FX6157	_STOG	BOOL	-	Reversal every scan (scan toggle)	On/Off reversed flag per every scan when user program is working. (on state for first scan)
%FX6163	_ALL_OFF	BOOL	-	All output Off	On in case all outputs are off
%FX30720	_RTC_WR	BOOL	Available	Writing data to RTC	Write data to RTC and read
%FX30721	_SCAN_WR	BOOL	Available	Initialize scan value	Initialize scan value
%FX30722	_CHK_ANC_ERR	BOOL	Available	Request for detecting heavy fault of external device	Flag that requests detecting heavy fault of external
%FX30723	_CHK_ANC_WAR	BOOL	Available	Request for detecting light fault of external device	Flag that requests detecting light fault (warning) of external
%FX30724	_MASTER_CHG	BOOL	Available	Master/Standby switching	Flag used when switching master/standby
%FW3860	_RTC_TIME_USER	ARRAY[0..7] OF BYTE	Available	Time to set	Flag for user to set time (year, month, hour, minute, second, day, century available)

Appendix 3.2 System Error Representative Flag

Master CPU system error representative flag

Address	Flag name	Type	Bit position	Contents	Description
%FD65	_CNF_ER	DWORD	Representative flag	System error (heavy fault error)	Handles error flags about non-operation fault error as below.
%FX2081	_IO_TYER	BOOL	BIT 1	Error when Module type mismatched	Representative flag displays when I/O configuration parameter for each slot is not matched with practical module configuration or a specific module is applied in the wrong location. (Refer to "_IO_TYER_N, _IO_TYER[n]")
%FX2082	_IO_DEER	BOOL	BIT 2	Module detachment error	Representative flag displays when the module configuration for each slot is changed while running. (Refer to "_IO_DEER_N, _IO_DEER[n]")
%FX2083	_FUSE_ER	BOOL	BIT 3	Fuse cutoff error	Representative flag displays when the fuse of module is cut off. (Refer to "_FUSE_ER_N, _FUSE_ER[n]")
%FX2086	_ANNUM_ER	BOOL	BIT 6	Heavy fault detection error in external device	Representative flag displays when heavy fault error detected by user program is recorded in "_ANC_ERR[n]".
%FX2088	_BPRM_ER	BOOL	BIT 8	Basic parameter error	Basic parameter does not match CPU type.
%FX2089	_IOPRM_ER	BOOL	BIT 9	I/O parameter error	It is abnormal to the I/O configuration parameter.
%FX2090	_SPPRM_ER	BOOL	BIT 10	Special module parameter error	It is abnormal to the special module parameter.
%FX2091	_CPPRM_ER	BOOL	BIT 11	Communication module parameter error	It is abnormal to the communication module parameter.
%FX2092	_PGM_ER	BOOL	BIT 12	Program error	Indicates that there is problem with user-made program.
%FX2093	_CODE_ER	BOOL	BIT 13	Program code error	Indicates that while user program is running, the program code cannot be interpreted.
%FX2094	_SWDT_ER	BOOL	BIT 14	CPU abnormal ends.	Displays when the saved program gets damages by an abnormal end of CPU or program does not work.
%FX2095	_BASE_POWER_ER	BOOL	BIT 15	Abnormal base power	Base power off or power module error
%FX2096	_WDT_ER	BOOL	BIT 16	Scan watchdog error	Indicates that the program scan time exceeds the scan watchdog time specified by a parameter.
%FX2097	_BASE_INFO_ER	BOOL	BIT 17	Base information error	Base information is abnormal
%FX2102	_BASE_DEER	BOOL	BIT 22	Extension base detachment error	Extension base is detatched
%FX2103	_DUPL_PRM_ER	BOOL	BIT 23	Redundant parameter error	Abnormal Redundant parameter
%FX2104	_INSTALL_ER	BOOL	BIT 24	Module attachment position error	The module which cannot be inserted into main base is inserted in to main base or The module which cannot be

Address	Flag name	Type	Bit position	Contents	Description
					inserted into extension base is inserted in to extension base
%FX2105	_BASE_ID_ER	BOOL	BIT 25	Overlapped extension base number	extension base number is overlapped
%FX2106	_DUPL_SYNC_ER	BOOL	BIT 26	Redundant operation Sync. error	Synchronization between master and standby CPU is abnormal
%FX2107	_AB_SIDEKEY_ER	BOOL	BIT 27	A/B SIDE key overlap error	A,B side key of master, standby CPU are overlapped. They should be different.

Standby CPU System error representative flag

Address	Flag name	Type	Bit position	Contents	Description
%FD129	_SB_CNF_ER	DWORD	Representative flag	System error (heavy fault error)	Handles error flags about non-operation fault error .
%FX4129	_SB_IO_TYER	BOOL	BIT 1	Module type mismatch error	Attached module is different with I/O parameter or some module which cannot be inserted into some slot is inserted some slot. Representative flag that detects them and displays (refer to _SB_IO_TYER_N, _SB_IO_TYERR)
%FX4130	_SB_IO_DEER	BOOL	BIT 2	Module detachment error	Representative flag displays when the module configuration for each slot is changed while running. (refer to _SB_IO_DEER_N,_SB_IO_DEERR]
%FX4131	_SB_FUSE_ER	BOOL	BIT 3	Fuse cutoff error	Representative flag displays when the fuse of module is cut off.
%FX4134	_SB_ANNUM_ER	BOOL	BIT 6	Heavy fault detection error in external device	Representative flag displays when heavy fault error detected by user program is recorded in "_ANC_ERR[n]".
%FX4136	_SB_BPRM_ER	BOOL	BIT 8	Basic parameter error	Basic parameter does not match CPU type.
%FX4137	_SB_IOPRM_ER	BOOL	BIT 9	I/O parameter error	It is abnormal to the I/O configuration parameter
%FX4138	_SB_SPPRM_ER	BOOL	BIT 10	Special module parameter error	It is abnormal to the special module parameter.
%FX4139	_SB_CPPRM_ER	BOOL	BIT 11	Communication module parameter error	It is abnormal to the communication module parameter.
%FX4141	_SB_CODE_ER	BOOL	BIT 13	Program code error	Indicates that while user program is running, the program code cannot be interpreted.
%FX4142	_SB_SWDT_ER	BOOL	BIT 14	CPU abnormal ends.	Displays when the saved program gets damages by an abnormal end of CPU or program cannot work.
%FX4143	_SB_BASE_POWER_ER	BOOL	BIT 15	Abnormal base power	Base power off or power module error
%FX4144	_SB_WDT_ER	BOOL	BIT 16	Scan watchdog error	Indicates that the program scan time exceeds the scan watchdog time specified by a parameter.
%FX4145	_SB_BASE_INFO_	BOOL	BIT 17	Base information error	Base information is abnormal

Address	Flag name	Type	Bit position	Contents	Description
	ER				
%FX4150	_SB_BASE_DEER	BOOL	BIT 22	Extension base detachment error	Extension base is detached.
%FX4151	_SB_DUPL_PRM_ER	BOOL	BIT 23	Abnormal redundant parameter	Redundant parameter is abnormal
%FX4152	_SB_INSTALL_ER	BOOL	BIT 24	Module attachment position error	The module which cannot be inserted into main base is inserted in to main base or the module which cannot be inserted into extension base is inserted in to extension base
%FX4153	_SB_BASE_ID_ER	BOOL	BIT 25	Overlapped extension base number	Extension base number overlaps.
%FX4154	_SB_DUPL_SYNC_ER	BOOL	BIT 26	Redundant operation Sync. error	Synchronization between master and standby CPU is abnormal
%FX4156	_SB_CPU_RUN_ER	BOOL	BIT 28	Standby CPU run error	Standby CPU fails to join redundant operation when MASTER CPU is error

Appendix 3.3 System Error Detail Flag

Master CPU system error detail flag

Address	Flag name	Type	Writable	Contents	Description
%FW424	_IO_TYERR	ARRAY[0..31] OF WORD	-	Module type mismatch error	Indicates slot and base where module mismatch error occurs.
%FW456	_IO_DEERR	ARRAY[0..31] OF WORD	-	Module detachment error	Indicates slot and base where module detachment error occurs.
%FW488	_FUSE_ERR	ARRAY[0..31] OF WORD	-	Fuse cutoff error	Indicates slot and base where fuse cutoff error occurs.
%FD83	_BASE_DEERR	DWORD	-	Extension base detachment error	Indicates base where extension base is detached.
%FD574	_BASE_POWER_FAIL	DWORD	-	Information of base where power module error occurs	Indicates base where power module error occurs.
%FW416	_IO_TYER_N	WORD	-	Module type mismatch slot number	Indicates slot number where module type mismatch error occurs. When two or more occurs, first slot indicates.
%FW417	_IO_DEER_N	WORD	-	Module detachment slot number	Indicates slot number where module detachment error occurs. When two or more occurs, first slot indicates.
%FW418	_FUSE_ER_N	WORD	-	Fuse cutoff slot number	Indicates slot number Fuse cutoff error occurs. When two or more occurs, first slot indicates.
%FW1922	_ANC_ERR	WORD	Available	Heavy fault information of external device	Classifies the type of user-defined error and writes value except 0. If detection of heavy fault is requested, it develops an external heavy fault detection error. By monitoring this flag, the user can know a reason of heavy fault.

2. Standby CPU system error detail flag

Address	Flag name	Type	Writable	Contents	Description
%FD147	_SB_BASE_DEERR	DWORD	-	Extension base detachment error	Indicates base where extension base is detached.
%FW588	_SB_IO_TYERR	WORD	-	Module type mismatch error	Indicates slot and base where module mismatch error occurs.
%FW589	_SB_IO_DEERR	WORD	-	Module detachment error	Indicates slot and base where module detachment error occurs.

Appendix 3.4 System Warning Representative Flag

MASTER CPU System warning representative flag

Address	Flag name	Type	Bit position	Contents	Description
%FD66	_CNF_WAR	DWORD	Representative flag	System warning	Representative flag displayed the system warning state.
%FX2112	_RTC_ER	BOOL	BIT 0	RTC error	Indicates that RTC data is abnormal.
%FX2114	_BASE_EXIST_WAR	BOOL	BIT 2	Not joined base	Warns there is base which doesn't join operation.
%FX2115	_AB_SD_ER	BOOL	BIT 3	Stop by operation error	Stopped by abnormal operation.
%FX2116	_TASK_ER	BOOL	BIT 4	Task collision	It is collided to the task.
%FX2117	_BAT_ER	BOOL	BIT 5	Battery error	It has the error in the battery state.
%FX2118	_ANNUM_WAR	BOOL	BIT 6	External device fault	Indicates that the light fault in the external device is detected.
%FX2120	_HS_WAR	BOOL	BIT 8	High speed link	Abnormal HS parameter
%FX2121	_REDUN_WAR	BOOL	BIT 9	Redundant configuration warning	The single CPU RUN mode and redundant configuration is not configured
%FX2122	_OS_VER_WAR	BOOL	BIT 10	O/S version mismatch	OS versions between CPUs, extension managers, extension drive modules are different.
%FX2123	_RING_WAR	BOOL	BIT 11	Ring topology configuration warning	Configure an extension cable as the Ring topology.
%FX2132	_P2P_WAR	BOOL	BIT 20	P2P parameter	Abnormal P2P parameter
%FX2140	_CONSTANT_ER	BOOL	BIT 28	Fixed cycle error	Fixed cycle error
%FX2141	_BASE_POWER_WAR	BOOL	BIT 29	Power module error warning	One or two power module is error
%FX2142	_BASE_SKIP_WAR	BOOL	BIT 30	Base skip cancelation warning	In case of canceling the base skip, base is different with IO parameter
%FX2143	_BASE_NUM_OVER_WAR	BOOL	BIT 31	Base number setting error	Base number of extension drive module is not 1~31

Standby CPU System warning representative flag

Address	Flag name	Type	Bit position	Contents	Description
%FD130	_SB_CNF_WAR	DWORD	Representative flag	System warning	Representative flag displayed the system warning state
%FX4160	_SB_RTC_ER	BOOL	BIT 0	RTC error	Indicates that RTC data is abnormal
%FX4162	_SB_BASE_EXIST_WAR	BOOL	BIT 2	Not joined base	Warns there is base which does not join operation.
%FX4163	_SB_AB_SD_ER	BOOL	BIT 3	Stop by operation error	Stopped by abnormal operation
%FX4164	_SB_TASK_ER	BOOL	BIT 4	Task collision	It is collided to the task
%FX4165	_SB_BAT_ER	BOOL	BIT 5	Battery error	It is to the error in the battery state
%FX4166	_SB_ANNUM_WAR	BOOL	BIT 6	External device fault	Indicates that the light fault in the external device is detected.
%FX4168	_SB_HS_WAR	BOOL	BIT 8	High speed link	Abnormal HS parameter
%FX4170	_SB_OS_VER_WAR	BOOL	BIT 10	O/S version mismatch	OS versions between CPUs, extension managers, extension drive modules are different
%FX4171	_SB_RING_WAR	BOOL	BIT 11	Ring topology configuration warning	Configure an extension cable as the Ring topology
%FX4180	_SB_P2P_WAR	BOOL	BIT 20	P2P parameter	Abnormal P2P parameter
%FX4188	_SB_CONSTANT_ER	BOOL	BIT 28	Fixed cycle error	Fixed cycle error
%FX4189	_SB_BASE_POWER_WAR	BOOL	BIT 29	Power module error warning	One or two power module is error
%FX4190	_SB_BASE_SKIP_WAR	BOOL	BIT 30	Base skip cancelation warning	In case of canceling the base skip, base is different with IO parameter
%FX4191	_SB_BASE_NUM_OVER_WAR	BOOL	BIT 31	Base number setting error	Base number of extension drive module is not 1~31

Appendix 3.5 System Warning Detail Flag

Master CPU system warning detail flag

Address	Flag name	Type	Writable	Contents	Description
%FX2624	_HS_WARN	ARRAY[0..11] OF BOOL	-	Abnormal HS parameter	Relevant flag is on in case Hs parameter is abnormal
%FX2640	_P2P_WARN	ARRAY[0..7] OF BOOL	-	Abnormal P2P parameter	Relevant flag is on in case P2P parameter is abnormal P2P
%FD587	_BASE_ACPF_WAR	DWORD	-	Instantaneous power cutoff occurrence warning information	Indicates base where Instantaneous power cutoff occurs
%FW164	_HS_WAR_W	WORD	-	Abnormal HS parameter	Indicates abnormal HS link number by bit
%FW165	_P2P_WAR_W	WORD	-	Abnormal P2P parameter	Indicates abnormal P2P link number by bit
%FW1923	_ANC_WAR	WORD	-	Light fault information external device	Classifies the type of user-defined error and writes value except 0. If detection of heavy fault is requested, it develops an external light fault detection error. By monitoring this flag, the user can know the reason of light fault.

Standby CPU system warning detail flag

Address	Flag name	Type	Writable	Contents	Description
%FX4672	_SB_HS_WARN	ARRAY[0..11] OF BOOL	-	Abnormal HS parameter	Relevant flag is on, in case Hs parameter is abnormal
%FX4688	_SB_P2P_WARN	ARRAY[0..7] OF BOOL	-	Abnormal P2P parameter	Relevant flag is on, in case P2P parameter is abnormal P2P
%FW292	_SB_HS_WAR_W	WORD	-	Abnormal HS parameter	Indicates abnormal HS link number by bit
%FW293	_SB_P2P_WAR_W	WORD	-	Abnormal P2P parameter	Indicates abnormal P2P link number by bit

Appendix 3.6 System Operation Status Information Flag

Master CPU system operation status information flag

Address	Flag name	Type	Bit position	Contents	Description
%FD64	_SYS_STATE	DWORD	Representative flag	PLC Mode and operation state	Indicates PLC mode and operation state of system.
%FX2048	_RUN	BOOL	BIT 0	RUN	Indicates CPU's operation status
%FX2049	_STOP	BOOL	BIT 1	STOP	
%FX2050	_ERROR	BOOL	BIT 2	ERROR	
%FX2051	_DEBUG	BOOL	BIT 3	DEBUG	
%FX2052	_LOCAL_CON	BOOL	BIT 4	Local control	Indicates operation mode changeable state only by the Mode key and XG5000.
%FX2054	_REMOTE_CON	BOOL	BIT 6	Remote Mode On	It is Remote control mode
%FX2058	_RUN_EDIT_DONE	BOOL	BIT 10	Editing during Run completed	Indicates completion of editing during Run
%FX2059	_RUN_EDIT_NG	BOOL	BIT 11	Editing during Run abnormally completed	Edit is ended abnormally during Run
%FX2060	_CMOD_KEY	BOOL	BIT 12	Operation mode change by key	Indicates Operation mode change by key
%FX2061	_CMOD_LPADT	BOOL	BIT 13	Operation mode change by local PADT	Indicates operation mode change by local PADT
%FX2062	_CMOD_RPADT	BOOL	BIT 14	Operation mode change by remote PADT	Indicates operation mode change by remote PADT
%FX2063	_CMOD_RLINK	BOOL	BIT 15	Operation mode change by remote communication module	Indicates operation mode change by remote communication module
%FX2064	_FORCE_IN	BOOL	BIT 16	Forced Input	Forced On/Off state about input contact
%FX2065	_FORCE_OUT	BOOL	BIT 17	Forced Output	Forced On/Off state about output contact
%FX2066	_SKIP_ON	BOOL	BIT 18	Input/Output Skip	I/O Skip on execution
%FX2067	_EMASK_ON	BOOL	BIT 19	Fault mask	Fault mask on execution
%FX2069	_USTOP_ON	BOOL	BIT 21	Stopped by STOP function	Stopped after scan completion by 'STOP' function while RUN mode operation.
%FX2070	_ESTOP_ON	BOOL	BIT 22	Stopped by ESTOP function	Instantly stopped by 'ESTOP' function while RUN mode operation.
%FW192	_SL_OS_VER	ARRAY[0..31] OF WORD	-	O/S version of extension drive module	Indicates O/S version of extension drive module
%FW600	_BASE_INFO	ARRAY[0..31] OF WORD	-	Base information	Indicates how many base is installed
%FB12	_RTC_TIME	ARRAY[0..7] OF BYTE	-	Current clock	Indicates current clock
%FX2072	_INIT_RUN	BOOL	-	Initialization task on execution	User-defined Initialization program on execution.

Address	Flag name	Type	Bit position	Contents	Description
%FX2074	_AB_SIDE	BOOL	-	CPU position	CPU position (A-SIDE: ON, B-SIDE: OFF)
%FX2076	_PB1	BOOL	-	Program Code 1	Program code 1 is selected
%FX2077	_PB2	BOOL	-	Program Code 2	Program code 1 is selected
%FX30736	_INIT_DONE	BOOL	writable	Initialization task execution completion	If this flag is set by user's initial program, it is started to execution of scan program after initial program completion.
%FW584	_RTC_DATE	DATE	-	RTC's current date	Indicates RTC's current date
%FD67	_OS_VER	DWORD	-	O/S version	Indicates CPU O/S version
%FD68	_OS_DATE	DWORD	-	O/S data	Indicates CPU O/S data
%FD69	_CP_OS_VER	DWORD	-	Extension manager O/S version	Indicates extension manager O/S version
%FD573	_OS_TYPE	DWORD	-	For PLC classification	Whether it is provided to other division
%FW1081	_FALS_NUM	INT	-	FALS number	Indicates FALS number
%FD293	_RTC_TOD	TIME_OF_DAY	-	RTC's current clock	Indicates RTC's current clock RTC. (ms unit)
%FD582	_RUN_EDIT_CNT	UDINT	-	The no. of editing during Run	Indicates the no. of editing during Run
%FW140	_AC_F_CNT	UINT	-	The no. of instantaneous power cutoff	Indicates the no. of instantaneous power cutoff
%FW158	_POWER_OFF_CNT	UINT	-	The no. of power cutoff	Indicates the no. of power cutoff
%FW386	_SCAN_MAX	UINT	writable	Max. scan time	Indicates max. scan time after(unit: 0.1ms)
%FW387	_SCAN_MIN	UINT	writable	Min. scan time	Indicates min. scan time after Run
%FW388	_SCAN_CUR	UINT	writable	Current scan time	Indicates current scan time (unit 0.1ms)
%FW585	_RTC_WEEK	UINT	-	RTC's current day	Indicates RTC's current day
%FW141	_CPU_TYPE	WORD	-	CPU ID (XGR - 0xA801)	Indicates CPU type
%FW633	_RBANK_NUM	WORD	-	Currently used block no.	Indicates currently used block no.

Standby CPU system operation status information flag

Address	Flag name	Type	Bit position	Contents	Description
%FD128	_SB_SYS_STATE	DWORD	Representative flag	System information	Handles system information
%FX4096	_SB_RUN	BOOL	BIT 0	RUN	Indicates CPU's operation status
%FX4097	_SB_STOP	BOOL	BIT 1	STOP	
%FX4098	_SB_ERROR	BOOL	BIT 2	ERROR	
%FX4100	_SB_LOCAL_CON	BOOL	BIT 4	Local control	Local control mode

Address	Flag name	Type	Bit position	Contents	Description
%FX4102	_SB_REMOTE_CON	BOOL	BIT 6	Remote mode On	Remote control mode
%FX4106	_SB_RUN_EDIT_DONE	BOOL	BIT 10	Editing during Run completed	Indicates completion of editing during Run
%FX4107	_SB_RUN_EDIT_NG	BOOL	BIT 11	Editing during Run abnormally completed	Edit is ended abnormally during Run
%FX4108	_SB_CMOD_KEY	BOOL	BIT 12	Operation mode change by key	Indicates Operation mode change by key
%FX4109	_SB_CMOD_LPADT	BOOL	BIT 13	Operation mode change by local PADT	Indicates operation mode change by local PADT
%FX4110	_SB_CMOD_RPADT	BOOL	BIT 14	Operation mode change by remote PADT	Indicates operation mode change by remote PADT
%FX4111	_SB_CMOD_RLINK	BOOL	BIT 15	Operation mode change by remote communication module	Indicates operation mode change by remote communication module
%FX4112	_SB_FORCE_IN	BOOL	BIT 16	Forced Input	Forced On/Off state about input contact
%FX4113	_SB_FORCE_OUT	BOOL	BIT 17	Forced Output	Forced On/Off state about output contact
%FX4114	_SB_SKIP_ON	BOOL	BIT 18	Input/Output Skip	I/O Skip on execution
%FX4115	_SB_EMASK_ON	BOOL	BIT 19	Fault mask	Fault mask on execution
%FX4117	_SB_USTOP_ON	BOOL	-	Stopped by STOP function	Stopped after scan completion by 'STOP' function while RUN mode operation.
%FX4118	_SB_ESTOP_ON	BOOL	-	Stopped by ESTOP function	Instantly stopped by 'ESTOP' function while RUN mode operation.
%FD131	_SB_OS_VER	DWORD	-	O/S version	Indicates CPU O/S version
%FD132	_SB_OS_DATE	DWORD	-	O/S data	Indicates CPU O/S data
%FD133	_SB_CP_OS_VER	DWORD	-	O/S version of extension drive module	Indicates O/S version of extension drive module
%FW286	_SB_POWER_OFF_CNT	UINT	-	The no. of power cutoff	Indicates the no. of power cutoff
%FW269	_SB_CPU_TYPE	WORD	-	CPU ID (XGR - 0xA801)	Indicates CPU type
%FW632	_SB_BASE_INFO	WORD	-	Base information	Indicates how many base installed.

Appendix 3.7 Redundant Operation Mode Information Flag

Redundant operation mode information

Address	Flag name	Type	Bit position	Contents	Description
%FD0	_REDUN_STATE	DWORD	Representative flag	Redundant operation information	Representative flag that indicates Redundant operation information
%FX0	_DUAL_RUN	BOOL	BIT 0	Redundant operation	Now Redundant operation　CPU A, CPU B are normal
%FX1	_RING_TOPOLOGY	BOOL	BIT 1	Ring topology status	Extension base is configure as ring
%FX2	_LINE_TOPOLOGY	BOOL	BIT 2	Line topology status	Extension base is configure as line
%FX4	_SINGLE_RUN_A	BOOL	BIT 4	A-SIDE single Run mode	Indicates A-SIDE single Run mode
%FX5	_SINGLE_RUN_B	BOOL	BIT 5	B-SIDE single Run mode	Indicates B-SIDE single Run mode
%FX6	_MASTER_RUN_A	BOOL	BIT 6	A-SIDE is master Run mode (Incase standby CPU exists)	Indicates A-SIDE is master Run mode
%FX7	_MASTER_RUN_B	BOOL	BIT 7	B-SIDE is master Run mode (Incase standby CPU exists)	Indicates B-SIDE is master Run mode

Appendix3.8 Operation Result Information Flag

Operation Result Information Flag

Address	Flag name	Type	Writable	Contents	Description
%FX672	_ARY_IDX_ERR	BOOL	Writable	Index range excess error in case of using array	In case of using array, index is out of setting value's range
%FX704	_ARY_IDX_LER	BOOL	Writable	Index range excess error latch in case of using array	Error occurred when index is out of setting value's range, in case of using array, is kept and the user erases this by program
%FX6160	_ERR	BOOL	Writable	Operation error flag	As an operation error flag by unit of operation function (FN) or function block (FB), it is renewed every operation
%FX6165	_LER	BOOL	Writable	Operation error latch flag	Operation error latch flag by program block (PB) unit. Error is kept until relevant program ends and the user erases this by program.

Appendix 3.9 Operation mode Key Status Flag

Operation mode key status flag

Address	Flag name	Type	Writable	Contents	Description
%FX291	_REMOTE_KEY	BOOL	-	Remote key status information	CPU key position status information (remote: off, not remote: On)
%FX294	_STOP_KEY	BOOL	-	Stop key status information	CPU key position status information (Stop: off, not stop: On)
%FX295	_RUN_KEY	BOOL	-	Run key status information	CPU key position status information (Run: off, not Run: On)

Appendix 3.10 Link Flag (L) List

It describes data link (L) flag

[Table 1.10.1] Communication Flag List according to High speed link no. (High speed link no. 1 ~ 12)

Item	Keyword	Type	Content	Description
HS link	_HSn_RLINK	Bit	High speed link parameter “n” normal operation of all station	Indicates normal operation of all station according to parameter set in High speed link, and on under the condition as below. 1. In case that all station set in parameter is RUN mode and no error. 2. All data block set in parameter is communicated normally. 3. The parameter set in each station itself is communicated normally. Once RUN_LINK is On, it keeps On unless stopped by LINK_DISABLE.
	_HSn_LTRBL	Bit	Abnormal state after _HSn_RLINK ON	In the state of _HSmRLINK flag On, if communication state of the station set in the parameter and data block is as follows, this flag shall be on. 1. In case that the station set in the parameter is not RUN mode, or 2. There is an error in the station set in the parameter, or 3. The communication state of data block set in the parameter is not good. LINK TROUBLE shall be on if the above 1, 2 & 3 conditions occur, and if the condition return to the normal state, it shall be off again.
	_HSn_STATE[k] (k=000~127)	Bit Array	High speed link parameter “n”, k block general state	Indicates the general state of communication information for each data block of setting parameter. HS1STATEk=HS1MODk&_HS1TR X k&(~_HSnERRk)
	_HSn_MOD[k] (k=000~127)	Bit Array	High speed link parameter “n”, k block station RUN operation mode	Indicates operation mode of station set in k data block of parameter.
	_HSn_TRX[k] (k=000~127)	Bit Array	Normal communication with High speed link parameter “n”, k block station	Indicates if communication state of k data of parameter is communicated smoothly according to the setting.
	_HSn_ERR[k] (k=000~127)	Bit Array	High speed link parameter “n”, k block station operation error mode	Indicates if the error occurs in the communication state of k data block of parameter.
	_HSn_SETBLOCK[k]	bit Array	High speed link parameter “n”, k block setting	Indicates whether or not to set k data block of the parameter.

Notes

High Speed Link no.	L area address	Remarks
1	L000000~L00049F	Comparing with High speed link 1 from [Table 1], the flag address of different high speed link station no. is as follows by a simple calculation formula. ∗ Calculation formula : L area address = L000000 + 500 x (High speed link no. – 1) In case of using high speed line flag for program and monitoring, you can use the flag map registered in XG5000 conveniently.
2	L000500~L00099F	
3	L001000~L00149F	
4	L001500~L00199F	
5	L002000~L00249F	
6	L002500~L00299F	
7	L003000~L00349F	
8	L003500~L00399F	
9	L004000~L00449F	
10	L004500~L00499F	
11	L005000~L00549F	

k means block no. and appears 8 words by 16 per 1 word for 128 blocks from 000~127.
For example, mode information (_HS1MOD) appears from block 0 to block 15 for L00010, and block 16~31, 32~47, 48~63, 64~79, 80~95, 96~111, 112~127 information for L00011, L00012, L00013, L00014, L00015, L00016, L00017. Thus, mode information of block no. 55 appears in L000137.

[Table 2] Communication Flag List according to P2P Service Setting

P2P parameter no.(n) : 1~8, P2P block(xx) : 0~63

No.	Keyword	Type	Contents	Description
P2P	_P2Pn_NDRxx	Bit	P2P parameter n, xx Block service normal end	Indicates P2P parameter n, xx Block service normal end
	_P2Pn_ERRxx	Bit	P2P parameter n, xx Block service abnormal end	Indicates P2P parameter n, xx Block service abnormal end
	_P2Pn_STATUSxx	Word	P2P parameter n, xx Block service abnormal end error Code	Indicates error code in case of P2P parameter n, xx Block service abnormal end
	_P2Pn_SVCCNTxx	Double word	P2P parameter n, xx Block service normal count	Indicates P2P parameter n, xx Block service normal count
	_P2Pn_ERRCNTxx	Double word	P2P parameter n, xx Block service abnormal count	Indicates P2P parameter n, xx Block service abnormal count

Appendix 3.11 Communication Flag (P2P) List

Link Register List according to P2P No. P2P Parameter No. (n) : 1~8, P2P Block(xx) : 0~63

No.	Flags	Type	Contents	Description
N00000	_PnBxxSN	Word	P2P parameter n, xx block another station no	Saves another station no. of P2P parameter 1, 00 block. In case of using another station no. at XG-PD, it is possible to edit during RUN by using P2PSN command.
N00001 ~ N00004	_PnBxxRD1	Device structure	Area device 1 to read P2P parameter n, xx block	Saves area device 1 to read P2P parameter n, xx block.
N00005	_PnBxxRS1	Word	Area size 1 to read P2P parameter n, xx block	Saves area size 1 to read P2P parameter n, xx block.
N00006 ~ N00009	_PnBxxRD2	Device structure	Area device 2 to read P2P parameter n, xx block	Saves area device 2 to read P2P parameter n, xx block.
N00010	_PnBxxRS2	Word	Area size 2 to read P2P parameter n, xx block	Saves area size 2 to read P2P parameter n, xx block.
N00011 ~ N00014	_PnBxxRD3	Device structure	Area device 3 to read P2P parameter n, xx block	Saves area device 3 to read P2P parameter n, xx block.
N00015	_PnBxxRS3	Word	Area size 3 to read P2P parameter n, xx block	Saves area size 3 to read P2P parameter n, xx block.
N00016 ~ N00019	_PnBxxRD4	Device structure	Area device 4 to read P2P parameter n, xx block	Saves area device 4 to read P2P parameter n, xx block.
N00020	_PnBxxRS4	Word	Area size 4 to read P2P parameter n, xx block	Saves area size 4 to read P2P parameter n, xx block.
N00021 ~ N00024	_PnBxxWD1	Device structure	Area device 1 to save P2P parameter n, xx block	Saves area device 1 to save P2P parameter n, xx block.
N00025	_PnBxxWS1	Word	Area size 1 to save P2P parameter n, xx block	Saves area size 1 to save P2P parameter n, xx block.
N00026 ~ N00029	_PnBxxWD2	Device structure	Area device 2 to save P2P parameter n, xx block	Saves area device 2 to save P2P parameter n, xx block.
N00030	_PnBxxWS2	Word	Area size 2 to save P2P parameter n, xx block	Saves area size 2 to save P2P parameter n, xx block.
N00031 ~ N00034	_PnBxxWD3	Device structure	Area device 3 to save P2P parameter n, xx block	Saves area device 3 to save P2P parameter n, xx block.
N00035	_PnBxxWS3	Word	Area size 3 to save P2P parameter n, xx block	Saves area size 3 to save P2P parameter n, xx block.
N00036 ~ N00039	_PnBxxWD4	Device structure	Area device 4 to save P2P parameter n, xx block	Saves area device 4 to save P2P parameter n, xx block.
N00040	_PnBxxWS4	WORD	Area size 4 to save P2P parameter n, xx block	Saves area size 4 to save P2P parameter n, xx block.

Notes

N area shall be set automatically when setting P2P parameter by using XG-PD and available to modify during RUN by using P2P dedicated command.

N area has a different address classified according to P2P parameter setting no., block index. The area not used by P2P service as address is divided and can be used by internal device.

Appendix 3.12 Reserved Word

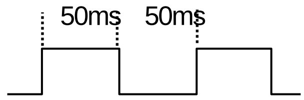
The reserved words are predefined words to use in the system.
Therefore, it is impossible to use them as the identifier.

Reserved Words
ACTION ... END_ACTION
ARRAY ... OF
AT
CASE ... OF ... ELSE ... END_CASE CONFIGURATION ... END_CONFIGURATION
Name of Data Type
DATE#, D# DATE_AND_TIME#, DT#
EXIT
FOR ... TO ... BY ... DO ... END_FOR
FUNCTION ... END_FUNCTION
FUNCTION_BLOCK ... END_FUNCTION_BLOCK
Names of Function Block
IF ... THEN ... ELSIF ... ELSE ... END_IF
OK
Operator (IL Language) Operator (ST Language)
PROGRAM
PROGRAM ... END_PROGRAM
REPEAT ... UNTIL ... END_REPEAT
RESOURCE ... END_RESOURCE
RETAIN
RETURN
STEP ... END_STEP
STRUCTURE ... END_STRUCTURE
T#
TASK ... WITH
TIME_OF_DAY#, TOD#
TRANSITION ... FROM... TO ... END_TRANSITION
TYPE ... END_TYPE
VAR ... END_VAR VAR_INPUT ... END_VAR VAR_OUTPUT ... END_VAR VAR_IN_OUT ... END_VAR VAR_EXTERNAL ... END_VAR
VAR_ACCESS ... END_VAR
VAR_GLOBAL ... END_VAR
WHILE ... DO ... END_WHILE
WITH

Appendix 4 Flag List (XEC)

A4.1 Special Relay (F) List

Reserved variable	Data type	Contents
_SYS_STATE	Mode and state	Indicates PLC mode and operation State.
_RUN	Run	Run state.
_STOP	Stop	Stop state.
_ERROR	Error	Error state.
_DEBUG	Debug	Debug state.
_LOCAL_CON	Local control	Local control mode.
_REMOTE_CON	Remote mode	Remote control mode.
_RUN_EDIT_ST	Online editing	Editing program download during RUN.
_RUN_EDIT_CHK		Internal edit processing during RUN.
_RUN_EDIT_DONE		Edit is done during RUN.
_RUN_EDIT_NG		Edit is ended abnormally during RUN.
_CMOD_KEY	Change Operation Mode	Operation mode changed by key.
_CMOD_LPADT		Operation mode changed by local PADT.
_CMOD_RPADT		Operation mode changed by Remote PADT.
_CMOD_RLINK		Operation mode changed by Remote communication module.
_FORCE_IN	Forced input	Forced input state.
_FORCE_OUT	Forced output	Forced output state.
_MON_ON	Monitor	Monitor on execution.
_USTOP_ON	Stop by STOP function	PLC stops by STOP function after finishing current scan
_ESTOP_ON	Stop by Estop function	PLC stops by ESTOP function promptly
_INIT_RUN	Initialize	Initialization task on execution.
_PB1	Program Code 1	Select Program Code 1.
_PB2	Program Code 2	Select Program Code 2.
_CB1	Compile Code 1	Select Compile Code 1.
_CB2	Compile Code2	Select Compile Code 2.
_CNF_ER	System error	Reports heavy error state of system.
_IO_TYER	Module Type error	Module Type does not match.
_IO_DEER	Module detachment error	Module is detached.
_IO_RWER	Module I/O error	Module I/O error.
_IP_IFER	Module interface error	Special/communication module interface error.

<table>
<tr><th>Reserved variable</th><th>Data type</th><th>Contents</th></tr>
<tr><td>_ANNUM_ER</td><td>External device error</td><td>Detected heavy error in external device.</td></tr>
<tr><td>_BPRM_ER</td><td>Basic parameter</td><td>Basic parameter error.</td></tr>
<tr><td>_IOPRM_ER</td><td>IO parameter</td><td>I/O configuration parameter error.</td></tr>
<tr><td>_SPPRM_ER</td><td>Special module parameter</td><td>Special module parameter is abnormal.</td></tr>
<tr><td>_CPPRM_ER</td><td>Communication module parameter</td><td>Communication module parameter is abnormal.</td></tr>
<tr><td>_PGM_ER</td><td>Program error</td><td>There is error in Check Sum of user program</td></tr>
<tr><td>_CODE_ER</td><td>Program code error</td><td>Meets instruction can not be interpreted</td></tr>
<tr><td>_SWDT_ER</td><td>CPU abnormal stop
Or malfunction</td><td>The saved program is damaged because of CPU abnormal end or program can not be executed.</td></tr>
<tr><td>_WDT_ER</td><td>Scan watchdog</td><td>Scan watchdog operated.</td></tr>
<tr><td>_CNF_WAR</td><td>System warning</td><td>Reports light error state of system.</td></tr>
<tr><td>_RTC_ER</td><td>RTC data error</td><td>RTC data Error occurred</td></tr>
<tr><td>_DBCK_ER</td><td>Backup error</td><td>Data backup error.</td></tr>
<tr><td>_HBCK_ER</td><td>Restart error</td><td>Hot Restart is not available</td></tr>
<tr><td>_ABSD_ER</td><td>Operation shutdown error</td><td>Stop by abnormal operation.</td></tr>
<tr><td>_TASK_ER</td><td>Task collision</td><td>Tasks are under collision</td></tr>
<tr><td>_BAT_ER</td><td>Battery error</td><td>There is error in battery status</td></tr>
<tr><td>_ANNUM_WAR</td><td>External device error</td><td>Detected light error of external device.</td></tr>
<tr><td>_HS_WAR1</td><td>High speed link 1</td><td>High speed link – parameter 1 error.</td></tr>
<tr><td>_HS_WAR2</td><td>High speed link 2</td><td>High speed link – parameter 2 error.</td></tr>
<tr><td>_P2P_WAR1</td><td>P2P parameter 1</td><td>P2P – parameter 1 error.</td></tr>
<tr><td>_P2P_WAR2</td><td>P2P parameter 2</td><td>P2P – parameter 2 error.</td></tr>
<tr><td>_P2P_WAR3</td><td>P2P parameter 3</td><td>P2P – parameter 3 error.</td></tr>
<tr><td>_CONSTANT_ER</td><td>Constant error</td><td>Constant error.</td></tr>
<tr><td>_USER_F</td><td>User contact</td><td>Timer used by user.</td></tr>
<tr><td>_T20MS</td><td>20ms</td><td rowspan="8">As a clock signal available at user program, it reverses on/off every half period. Since clock signal is dealt with at the end of scan, there may be delay or distortion according to scan time. So use clock that’s longer than scan time. Clock signal is Off status at the start of scan program and task program.
_T100ms clock
_T100ms clock
</td></tr>
<tr><td>_T100MS</td><td>100ms</td></tr>
<tr><td>_T200MS</td><td>200ms</td></tr>
<tr><td>_T1S</td><td>1s Clock</td></tr>
<tr><td>_T2S</td><td>2 s Clock</td></tr>
<tr><td>_T10S</td><td>10 s Clock</td></tr>
<tr><td>_T20S</td><td>20 s Clock</td></tr>
<tr><td>_T60S</td><td>60 s Clock</td></tr>
</table>

Reserved variable	Data type	Contents
	Ordinary time On	Always on state Bit.
_Off	Ordinary time Off	Always off state Bit.
_1On	1scan On	First scan on Bit.
_1Off	1scan Off	First scan off bit.
_STOG	Reversal	Reversal every scan.
_USER_CLK	User Clock	Clock available for user setting.
_USR_CLK0	Setting scan repeat	On/off as much as set scan Clock 0.
_USR_CLK1	Setting scan repeat	On/off as much as set scan Clock 1.
_USR_CLK2	Setting scan repeat	On/off as much as set scan Clock 2.
_USR_CLK3	Setting scan repeat	On/off as much as set scan Clock 3.
_USR_CLK4	Setting scan repeat	On/off as much as set scan Clock 4.
_USR_CLK5	Setting scan repeat	On/off as much as set scan Clock 5.
_USR_CLK6	Setting scan repeat	On/off as much as set scan Clock 6.
_USR_CLK7	Setting scan repeat	On/off as much as set scan Clock 7.
_LOGIC_RESULT	Logic result	Indicates logic results.
_ERR	operation error	On during 1 scan in case of operation error.
_LER	Operation error latch	Continuously on in case of operation error
_FALS_NUM	FALS no.	Indicates FALS no.
_PUTGET_ERR0	PUT/GET error 0	Main base Put / Get error.
_PUTGET_NDR0	PUT/GET end 0	Main base Put/Get end.
_CPU_TYPE	CPU Type	Indicates information for CPU Type.
_CPU_VER	CPU version	Indicates CPU version.
_OS_VER	OS version	Indicates OS version.
_OS_DATE	OS date	Indicates OS distribution date.
_SCAN_MAX	Max. scan time	Indicates max. scan time.
_SCAN_MIN	Min. scan time	Indicates min. scan time.
_SCAN_CUR	Current scan time	Current scan time.
_MON_YEAR	Month/year	Clock data (month/year)
_TIME_DAY	Hour/date	Clock data (hour/date)
_SEC_MIN	Second/minute	Clock data (Second/minute)
_HUND_WK	Hundred year/week	Clock data (Hundred year/week)
_REF_COUNT	Refresh count	Increase when module Refresh.
_REF_OK_CNT	Refresh OK	Increase when module Refresh is normal.

Reserved variable	Data type	Contents
_REF_NG_CNT	Refresh NG	Increase when module Refresh is abnormal.
_REF_LIM_CNT	Refresh Limit	Increase when module Refresh is abnormal (Time Out).
_REF_ERR_CNT	Refresh Error	Increase when module Refresh is abnormal.
_BUF_FULL_CNT	Buffer Full	Increase when CPU internal buffer is full.
_PUT_CNT	Put count	Increase when Put count.
_GET_CNT	Get count	Increase when Get count.
_KEY	Current key	Indicates the current state of local key.
_KEY_PREV	Previous key	Indicates the previous state of local key
_IO_TYER_N	Mismatch slot	Module Type mismatched slot no.
_IO_DEER_N	Detach slot	Module detached slot no.
_IO_RWER_N	RW error slot	Module read/write error slot no.
_IP_IFER_N	IF error slot	Module interface error slot no.
_IO_TYER0	Module Type 0 error	Main base module Type error.
_IO_DEER0	Module Detach 0 error	Main base module Detach error.
_IO_RWER0	Module RW 0 error	Main base module read/write error.
_IO_IFER_0	Module IF 0 error	Main base module interface error.
_AC_FAIL_CNT	Current time of RTC (unit: ms)	As time data based on 00:00:00 within one day, unit is ms
_ERR_HIS_CNT	Power shutdown times	Saves the times of power shutdown.
_MOD_HIS_CNT	Error occur times	Saves the times of error occur.
_SYS_HIS_CNT	Mode conversion times	Saves the times of mode conversion.
_LOG_ROTATE	History occur times	Saves the times of system history.
_BASE_INFO0	Slot information 0	Main base slot information.
_RBANK_NUM	Currently used block No.	Indicates currently used block no.
_RBLOCK_STATE	Currently used block status	Indicates Currently used block status (Read/Write/Error)
_RBLOCK_RD_FLAG	Read flash N block	When reading data of flash N block, Nth bit is on.
_RBLOCK_WR_FLAG	Write flash N block	When writing data of flash N block, Nth bit is on.
_RBLOCK_ER_FLAG	Flash N block error	When error occurs during flash N block service, Nth bit is on.
_USER_WRITE_F	Available contact point	Contact point available in program.
_RTC_WR	RTC RW	Data write and read in RTC.
_SCAN_WR	Scan WR	Initializing the value of scan.
_CHK_ANC_ERR	Request detection of external serious error	Request detection of external error.

Reserved variable	Data type	Contents
_CHK_ANC_WAR	Request detection of external slight error (warning)	Request detection of external slight error (warning).
_USER_STAUS_F	User contact point	User contact point.
_INIT_DONE	Initialization completed	Initialization complete displayed.
_ANC_ERR	Display information of external serious error	Display information of external serious error
_ANC_WAR	Display information of external slight error (warning)	Display information of external slight error (warning)
_MON_YEAR_DT	Month/year	Clock data (month/year)
_TIME_DAY_DT	Hour/date	Clock data (hour/date)
_SEC_MIN_DT	Second/minute	Clock data (Second/minute)
_HUND_WK_DT	Hundred year/week	Clock data (Hundred year/week)
_ARY_IDX_ERR	Array –index- range exceeded- error flag	Error flag is indicated when exceeding the no. of array
_ARY_IDX_LER	Array –index- range exceeded- latch-error flag	Error latch flag is indicated when exceeding the no. of array

A4.2 High Speed Link Flag (* = 1~2, *** = 000~063)

Reserved variable	Data type	Contents
_HS*_RLINK	BOOL	Every station of high speed link no.* normally works
_HS*_LTRBL	BOOL	Abnormal status after _HS*RLINK on
_HS*_STATE***	BOOL	General status of *** block of high speed link no.*
_HS*_MOD***	BOOL	Run operation mode of *** block of high speed link no.*
_HS*_TRX***	BOOL	Normal communication with *** block station of high speed link no.*
_HS*_ERR***	BOOL	Run error mode of *** block station of high speed link no.*
_HS*_SETBLOCK***	BOOL	*** block setting of high speed link no.*

A4.3 P2P Flag (* = 0 ~ 8, ** = 0 ~ 63)

Reserved variable	Data type	Contents
_P2P*_NDR**	BOOL	** block service of P2P no.* completed successfully
_P2P*_ERR**	BOOL	** block service of P2P no.* completed abnormally
_P2P*_STATUS**	WORD	Error code in case of ** block service of P2P no.*
_P2P*_SVCCNT**	DWORD	** block normal service frequency of P2P no.*
_P2P*_ERRCNT**	DWORD	** block abnormal service frequency of P2P no.*

A4.4 PID flag (* = 0 ~ 15, ** = 0 ~ 15)

Reserved variable	Data type	Contents
_PID_MAN	WORD	PID output selection(0:auto ,1:manual)
_PID*_MAN	BOOL	PID output selection(0:auto ,1:manual) - loop**
_PID_PAUSE	WORD	PID pause (0:STOP/RUN ,1:PAUSE)
_PID*_PAUSE	BOOL	PID pause (0:STOP/RUN ,1:PAUSE) - loop**
_PID_REV	WORD	PID operation selection(0:forward ,1:reverse)
_PID*_REV	BOOL	PID operation selection(0:forward ,1:reverse) - loop**
_PID_AW2D	WORD	PID Anti Wind-up2 prohibited (0:enable ,1:disable)
_PID*_AW2D	BOOL	PID Anti Wind-up2 prohibited (0:enable ,1:disable) - loop**
_PID_REM_RUN	WORD	PID remote (HMI) execution bit (0:STOP ,1:RUN)
_PID*_REM_RUN	BOOL	PID remote (HMI) execution bit (0:STOP ,1:RUN) - loop**
_PID_P_on_PV	WORD	PID proportional(P) cal source selection (0:ERR, 1:PV)
_PID*_P_on_PV	BOOL	PID proportional(P) cal source selection (0:ERR, 1:PV) - loop**
_PID_D_on_ERR	WORD	PID differential(D) cal source selection (0:PV, 1:ERR)
_PID*_D_on_ERR	BOOL	differential(D) cal source selection (0:PV, 1:ERR) - loop**
_PID_AT_EN	WORD	PID auto tuning setting (0:Disable, 1:Enable)
_PID*_AT_EN	BOOL	PID auto tuning setting (0:Disable, 1:Enable) –loop**
_PID_PWM_EN	WORD	PID PWM operation enable (0:Disable, 1:Enable)
_PID*_PWM_EN	BOOL	PID PWM operation enable (0:Disable, 1:Enable) - loop**
_PID_STD	WORD	PID operation status indication (0:Stop, 1:Run)
_PID*_STD	WORD	PID operation status indication (0:Stop, 1:Run) – loop 00**
_PID_ALARM	BOOL	PID P - constant (K_p) - block* loop**
_PID*_ALARM	REAL	PID I - constant (T_i)[sec] - loop**
_PID_ERROR	WORD	PID error occurs (0: normal 1: error occurs)
_PID*_ERROR	BOOL	PID error occurs (0: normal 1: error occurs) – loop 01
_PID*_SV	INT	PID Set value (SV) - loop**
_PID*_T_s	WORD	PID operation period (T_s)[0.1msec] - loop**
_PID*_K_p	REAL	PID P - constant (K_p) - loop**
_PID*_T_i	REAL	PID I - constant (T_i)[sec] - loop**
_PID*_T_d	REAL	PID D - constant (T_d)[sec] - loop**
_PID*_d_PV_max	WORD	PID PV change limit - loop**
_PID*_d_MV_max	WORD	PID MV change limit - loop**
_PID*_MV_max	INT	PID MV Max limit - loop**

Reserved variable	Data type	Contents
_PID*_MV_min	INT	PID MV Min limit – loop**
_PID*_MV_man	INT	PID manual output (MV_man) - loop**
_PID*_PV	INT	PID present value (PV) - loop**
_PID*_PV_old	INT	PID previous present value (PV_old) - loop**
_PID*_MV	INT	PID Manipulated value (MV) - loop**
_PID*_ERR	DINT	PID control error value - loop**
_PID*_MV_p	REAL	PID MV P component - loop**
_PID*_Mv_i	REAL	PID MV I component - loop**
_PID*_MV_d	REAL	PID MV D component - loop**
_PID*_DB_W	WORD	PID dead band setting (operation after stabilization) - loop**
_PID*_Td_lag	WORD	PID derivative function LAG filter - loop**
_PID*_PWM	WORD	PID PWM contact point setting value - loop**
_PID*_PWM_Prd	WORD	PID PWM output period - loop**
_PID*_SV_RAMP	WORD	PID Set value ramp value - loop**
_PID*_PV_Track	WORD	PID Set value track value - loop**
_PID*_PV_MIN	INT	PID Present value input Min. limit – loop**
_PID*_PV_MAX	INT	PID Present value input Min. limit – loop**
_PID*_ALM_CODE	WORD	PID alarm code – loop**
_PID*_ERR_CODE	WORD	PID error code - loop**
_PID00_CUR_SV	INT	PID current Set value (SV) – loop**
_AT_REV	WORD	AT operation selection (0:Forward, 1:Reverse)
_AT*_REV	BOOL	AT operation selection (0:Forward, 1:Reverse) - loop**
_AT_PWM_EN	WORD	AT PWM operation enable (0:Disable, 1:Enable)
_AT*_PWM_EN	BOOL	AT PWM operation enable (0:Disable, 1:Enable) - loop**
_AT_ERROR	WORD	AT error occurrence indication (0:normal, 1:error occurrence)
_AT*_ERROR	BOOL	AT error occurrence indication (0:normal, 1:error occurrence) - loop**
_AT*_SV	INT	AT Set value (SV) – loop**
_AT*_T_s	WORD	AT operation period (T_s)[0.1msec] – loop**
_AT*_MV_max	INT	AT MV Max. limit – loop**
_AT00_MV_min	INT	AT MV Min. limit – loop**
_AT*_PWM	WORD	AT PWM contact point setting value – loop**
_AT*_PWM_Prd	WORD	AT PWM output period – loop **
_AT*_HYS_val	WOPD	AT hysteresis setting– loop**
_AT*_STATUS	WORD	AT auto-tuning status indication (prohibited for user to set) – loop**

Reserved variable	Data type	Contents
_AT*_ERR_CODE	WORD	AT error code - (prohibited for user to set) – loop**
_AT*_K_p	REAL	AT result P – constant (K_p) – loop**
_AT*_T_i	REAL	AT result I - constant (T_i)[sec] – loop**
_AT*_T_d	REAL	AT result D - constant (T_d)[sec] - loop00
_AT*_PV	INT	AT present value – loop**
_AT*_MV	INT	AT manipulated value – loop**

A4.5 High Speed Counter flag (* = 0 ~ 7, ** = 0 ~ 7)

Reserved variable	Data type	Contents
_HSC*_Cnt_En	BOOL	CH** enable Counter
_HSC*_IntPrs_En	BOOL	CH** use counter internal preset
_HSC*_DecCnt_En	BOOL	CH** set decreasing counter
_HSC*_Cmp0_En	BOOL	CH** enable comparison output 0
_HSC*_Rpu_En	BOOL	CH** use revolution per unit time
_HSC*_Latch_En	BOOL	CH** use latch counter
_HSC*_Cmp1_En	BOOL	CH** enable comparison output
_HSC*_Carry	BOOL	CH** carry signal
_HSC*_Borrow	BOOL	CH** borrow signal
_HSC*_CmpOut0	BOOL	CH** comparison output 0 signal
_HSC*_CmpOut1	BOOL	CH** comparison output 1 signal
_HSC*_CurCnt	DINT	CH** current count value
_HSC*_CurRpu	DINT	CH** revolution per unit time
_HSC*_ErrCode	DINT	CH** error code
_HSC*_CntMode	INT	CH** counter mode
_HSC*_PlsMode	INT	CH** pulse input mode
_HSC*_CmpMode0	WORD	CH** comparison output 0 type
_HSC*_CmpMode1	WORD	CH** comparison output 1 type
_HSC*_IntPrs_Val	DINT	CH** internal preset setting value
_HSC*_ExtPrs_Val	DINT	CH** external preset setting value
_HSC*_RingMin_Val	DINT	CH** ring counter min. setting value
_HSC*_RingMax_Val	DINT	CH** ring counter max. setting value
_HSC*_CmpMin_Val0	DINT	CH** comparison output 0 min. setting value
_HSC*_CmpMax_Val0	DINT	CH** comparison output 0 max. setting value

Reserved variable	Data type	Contents
_HSC*_CmpMin_Val1	DINT	CH** comparison output 1 min. setting value
_HSC*_CmpMax_Val1	DINT	CH** comparison output 1 max. setting value
_HSC*_CmpContact0	WORD	CH** designate comparison output 0 output contact point
_HSC*_CmpContact1	WORD	CH** designate comparison output 1 output contact point
_HSC*_UnitTime	WORD	CH** unit time setting value
_HSC*_PlsPerRev	INT	CH** pulse number per revolution

A4.6 Positioning flag (* = 0 ~ 80, ** = 0 ~ 80)

Reserved variable	Data type	Contents
_POS_X_Busy	BOOL	X axis BUSY
_POS_Y_Busy	BOOL	Y axis BUSY
_POS_X_Err	BOOL	X axis error
_POS_Y_Err	BOOL	Y axis error
_POS_X_Done	BOOL	X axis position complete
_POS_Y_Done	BOOL	Y axis position complete
_POS_X_McodeOn	BOOL	X axis M code on
_POS_Y_McodeOn	BOOL	Y axis M code on
_POS_X_OriginFix	BOOL	X axis origin fix
_POS_Y_OriginFix	BOOL	Y axis origin fix
_POS_X_OutInhibit	BOOL	X axis output inhibit
_POS_Y_OutInhibit	BOOL	Y axis output inhibit
_POS_X_Stop	BOOL	X axis stop
_POS_Y_Stop	BOOL	Y axis stop
_POS_X_ULimit	BOOL	X axis upper limit detection
_POS_Y_ULimit	BOOL	Y axis upper limit detection
_POS_X_LLimit	BOOL	X axis lower limit detection
_POS_Y_LLimit	BOOL	Y axis lower limit detection
_POS_X_Estop	BOOL	X axis emergency stop
_POS_Y_Estop	BOOL	Y axis emergency stop
_POS_X_Dir	BOOL	X axis CW/CCW
_POS_Y_Dir	BOOL	Y axis CW/CCW
_POS_X_Acc	BOOL	X axis move status (acceleration)
_POS_Y_Acc	BOOL	Y axis move status (acceleration)

Reserved variable	Data type	Contents
_POS_X_Const	BOOL	X axis move status (constant)
_POS_Y_Const	BOOL	Y axis move status (constant)
_POS_X_Dec	BOOL	X axis move status (deceleration)
_POS_Y_Dec	BOOL	Y axis move status (deceleration)
_POS_X_Dwell	BOOL	X axis move status (dwell)
_POS_Y_Dwell	BOOL	Y axis move status (dwell)
_POS_X_Position	BOOL	X axis control pattern (Position)
_POS_Y_Position	BOOL	Y axis control pattern (Position)
_POS_X_Speed	BOOL	X axis control pattern (Speed)
_POS_Y_Speed	BOOL	Y axis control pattern (Speed)
_POS_X_LinearInt	BOOL	X axis control pattern (Linear Int.)
_POS_Y_LinearInt	BOOL	Y axis control pattern (Linear Int.)
_POS_X_Home	BOOL	X axis home return
_POS_Y_Home	BOOL	Y axis home return
_POS_X_PosSync	BOOL	X axis position sync.
_POS_Y_PosSync	BOOL	Y axis position sync.
_POS_X_SpdSync	BOOL	X axis speed sync
_POS_Y_SpdSync	BOOL	Y axis speed sync
_POS_X_JogLow	BOOL	X axis JOG low speed
_POS_Y_JogLow	BOOL	Y axis JOG low speed
_POS_X_JogHigh	BOOL	X axis JOG high speed
_POS_Y_JogHigh _POS_X_Inching	BOOL BOOL	Y axis JOG high speed X axis inching
_POS_Y_Inching	BOOL	Y axis inching
_POS_X_CurPos	DWORD	X axis current position
_POS_Y_CurPos	DWORD	Y axis current position
_POS_X_CurSpd	DWORD	X axis current speed
_POS_Y_CurSpd	DWORD	Y axis current speed
_POS_X_CurStep	WORD	X axis step number
_POS_Y_CurStep	WORD	Y axis step number
_POS_X_ErrCode	WORD	X axis error code
_POS_Y_ErrCode	WORD	Y axis error code
_POS_X_Mcode	WORD	X axis M code
_POS_Y_Mcode	WORD	Y axis M code

Reserved variable	Data type	Contents
_POS_X_Start	BOOL	X axis start
_POS_Y_Start	BOOL	Y axis start
_POS_X_CwJogStart	BOOL	X axis CW JOG START
_POS_Y_CwJogStart	BOOL	Y axis CW JOG START
_POS_X_CcwJogStart	BOOL	X axis CCW JOG START
_POS_Y_CcwJogStart	BOOL	Y axis CCW JOG START
_POS_X_JogLowHigh	BOOL	X axis JOG Low Speed/High Speed
_POS_Y_JogLowHigh	BOOL	Y axis JOG Low Speed/High Speed
_POS_X_BiasSpd	DWORD	X axis bias speed
_POS_Y_BiasSpd	DWORD	X axis bias speed
_POS_X_SpdLimit	DWORD	X axis speed limit
_POS_Y_SpdLimit	DWORD	Y axis speed limit
_POS_X_AccTime1	WORD	X axis acceleration time 1
_POS_Y_AccTime1	WORD	Y axis acceleration time 1
_POS_X_DecTime1	WORD	X axis deceleration time 1
_POS_Y_DecTime1	WORD	Y axis deceleration time 1
_POS_X_AccTime2	WORD	X axis acceleration time 2
_POS_Y_AccTime2	WORD	Y axis acceleration time 2
_POS_X_DecTime2	WORD	X axis deceleration time 2
_POS_Y_DecTime2	WORD	Y axis deceleration time 2
_POS_X_AccTime3	WORD	X axis acceleration time 3
_POS_Y_AccTime3	WORD	Y axis acceleration time 13
_POS_X_DecTime3	WORD	X axis deceleration time 3
_POS_Y_DecTime3	WORD	Y axis deceleration time 3
_POS_X_AccTime4	WORD	X axis acceleration time 4
_POS_Y_AccTime4	WORD	Y axis acceleration time 4
_POS_X_DecTime4	WORD	X axis deceleration time 4
_POS_Y_DecTime4	WORD	Y axis deceleration time 4
_POS_X_SwULimit	DWORD	X axis S/W upper limit
_POS_Y_SwULimit	DWORD	Y axis S/W upper limit
_POS_X_SwLLimit	DWORD	X axis S/W lower limit
_POS_Y_SwLLimit	DWORD	Y axis S/W lower limit
_POS_X_Backlash	WORD	X axis backlash compensation
_POS_Y_Backlash	WORD	Y axis backlash compensation

Reserved variable	Data type	Contents
_POS_X_McodeMode_L	BOOL	X axis M-Code output mode (Low Bit)
_POS_Y_McodeMode_L	BOOL	Y axis M-Code output mode (Low Bit)
_POS_X_McodeMode_H	BOOL	X axis M-Code output mode (High Bit)
_POS_Y_McodeMode_H	BOOL	Y axis M-Code output mode (High Bit)
_POS_X_LimitDetect	BOOL	X axis S/W limit detection
_POS_Y_LimitDetect	BOOL	Y axis S/W limit detection
_POS_X_HomeAddr	DWORD	X axis Home Address
_POS_Y_HomeAddr	DWORD	Y axis Home Address
_POS_X_HomeHSpd	DWORD	X axis Home High Speed
_POS_Y_HomeHSpd	DWORD	Y axis Home High Speed
_POS_X_HomeLSpd	DWORD	X axis Home Low Speed
_POS_Y_HomeLSpd	DWORD	Y axis Home Low Speed
_POS_X_HomeAccTime	WORD	X axis Homing acceleration time
_POS_Y_HomeAccTime	WORD	Y axis Homing acceleration time
_POS_X_HomeDccTime	WORD	X axis Homing deceleration time
_POS_Y_HomeDccTime	WORD	Y axis Homing deceleration time
_POS_X_HomeDwlTime	WORD	X axis Homing dwell time
_POS_Y_HomeDwlTime	WORD	Y axis Homing dwell time
_POS_X_HomeMethod_L	BOOL	X axis Homing Method (Low Bit)
_POS_Y_HomeMethod_L	BOOL	Y axis Homing Method (Low Bit)
_POS_X_HomeMethod_H	BOOL	X axis Homing Method (High Bit)
_POS_Y_HomeMethod_H	BOOL	Y axis Homing Method (High Bit)
_POS_X_HomeDir	BOOL	X axis homing direction
_POS_Y_HomeDir	BOOL	Y axis homing direction
_POS_X_JogHSpd	DWORD	X axis JOG high speed
_POS_Y_JogHSpd	DWORD	Y axis JOG high speed
_POS_X_JogLSpd	DWORD	X axis JOG low speed
_POS_Y_JogLSpd	DWORD	Y axis JOG low speed
_POS_X_JogAccTime	WORD	X axis JOG Acceleration Time
_POS_Y_JogAccTime	WORD	Y axis JOG Acceleration Time
_POS_X_JogDecTime	WORD	X axis JOG Deceleration Time
_POS_Y_JogDecTime	WORD	Y axis JOG Deceleration Time
_POS_X_JogInchSpd	WORD	X axis inching speed
_POS_Y_JogInchSpd	WORD	Y axis inching speed

Reserved variable	Data type	Contents
_POS_X_Position_En	BOOL	X axis position enable
_POS_Y_Position_En	BOOL	Y axis position enable
_POS_X_OutLevel	BOOL	X axis pulse output level
_POS_Y_OutLevel	BOOL	Y axis pulse output level
_POS_X_Limit_En	BOOL	X axis upper limit/lower limit enable
_POS_Y_Limit_En	BOOL	Y axis upper limit/lower limit enable
_POS_X_OutMode	BOOL	X axis pulse output mode
_POS_Y_OutMode	BOOL	Y axis pulse output mode
_POS_X_ST*_Addr	DWORD	X axis step** position
_POS_Y_ST*_Speed	DWORD	Y axis step** speed
_POS_X_ST*_Dwell	WORD	X axis step** dwell time
_POS_Y_ST*_Dwell	WORD	Y axis step** dwell time
_POS_X_ST*_Mcode	WORD	X axis step** M code number
_POS_Y_ST*_Mcode	WORD	Y axis step** M code number
_POS_X_ST*_Method	BOOL	X axis step** method
_POS_Y_ST*_Method	BOOL	Y axis step** method
_POS_X_ST*_Control	BOOL	X axis step** control
_POS_Y_ST*_Control	BOOL	Y axis step** control
_POS_X_ST*_Pattern_L	BOOL	X axis step** pattern (Low Bit)
_POS_Y_ST*_Pattern_L	BOOL	Y axis step** pattern (Low Bit)
_POS_X_ST*_Pattern_H	BOOL	X axis step** pattern (High Bit)
_POS_Y_ST*_Pattern_H	BOOL	Y axis step** pattern (High Bit)
_POS_X_ST*_Cordi	BOOL	X axis step**coordinates
_POS_Y_ST*_Cordi	BOOL	Y axis step**coordinates
_POS_X_ST*_AccDecN_L	BOOL	X axis step** AEC/DEC number (Low Bit)
_POS_Y_ST*_AccDecN_L	BOOL	Y axis step**AEC/DEC number (Low Bit)
_POS_X_ST*_AccDecN_H	BOOL	X axis step** AEC/DEC number (High Bit)
_POS_Y_ST*_AccDecN_H	BOOL	Y axis step** AEC/DEC number (High Bit)
_POS_X_ST01_RptStep	BOOL	X axis step**Repeat Step
_POS_Y_ST01_RptStep	BOOL	Y axis step**Repeat Step

Appendix 5 Flag List (XMC)

A5.1 System Flag List

This flag indicates the operation, state, and information of motion controller

Variable	Type	Address	Description
_SYS_STATE	DWORD	%FD0	PLC mode and states
_RUN	BOOL	%FX0	RUN
_STOP	BOOL	%FX1	STOP
_ERROR	BOOL	%FX2	ERROR
_LOCAL_CON	BOOL	%FX4	Local control
_REMOTE_CON	BOOL	%FX6	Remote mode ON
_RUN_EDIT_ST	BOOL	%FX8	Downloading a program at online editing mode
_RUN_EDIT_CHK	BOOL	%FX9	Processing online editing internally
_RUN_EDIT_DONE	BOOL	%FX10	Online editing done
_RUN_EDIT_NG	BOOL	%FX11	Online editing abnormal termination
_CMOD_KEY	BOOL	%FX12	Change operation mode by the switch
_CMOD_LPADT	BOOL	%FX13	Change operation mode by the local PADT
_FORCE_IN	BOOL	%FX16	Force input
_FORCE_OUT	BOOL	%FX17	Force output
_MON_ON	BOOL	%FX20	Monitoring mode
_USTOP_ON	BOOL	%FX21	STOP by STOP Function
_ESTOP_ON	BOOL	%FX22	STOP by ESTOP Function
_INIT_RUN	BOOL	%FX24	Executing the initial task
_PB1	BOOL	%FX28	Program code 1
_PB2	BOOL	%FX29	Program code 2
_CNF_ER	DWORD	%FD2	System errors(Significant error)
_ANNUM_ER	BOOL	%FX70	Significant error detection in external device
_BPRM_ER	BOOL	%FX72	Basic parameter error
_IOPRM_ER	BOOL	%FX73	IO configuration parameter error
_SPPRM_ER	BOOL	%FX74	Parameter error in Special module
_CPPRM_ER	BOOL	%FX75	Local Ethernet parameter error
_PGM_ER	BOOL	%FX76	Program error
_SWDT_ER	BOOL	%FX78	CPU abnormal ends
_ENCPRM_ER	BOOL	%FX85	Encoder parameter error
_AXISPRM_ER	BOOL	%FX86	Axis parameter error
_GROUPPRM_ER	BOOL	%FX87	Axis group parameter error
_ECPRM_ER	BOOL	%FX88	EtherCAT parameter error

Variable	Type	Address	Description
_NCPRM_ER	BOOL	%FX89	NC Parameter Error
_NCPGM_ER	BOOL	%FX90	NC Program Check Error
_PTASK_CYCLE_ER	BOOL	%FX91	Main Task Period Error
_CTASK_CYCLE_ER	BOOL	%FX92	Cycle Task Period Error
_SYSTEM_ER	BOOL	%FX93	System Error
_TASK_PRM_USAGE_OVER_ER	BOOL	%FX94	Task Program Occupancy Excess Error
_CNF_WAR	DWORD	%FD4	System warnings(Minor error)
_RTC_ER	BOOL	%FX128	Abnormal RTC data
_PTASK_CYCLE_WAR	BOOL	%FX129	Main Task Period Exceeded Warning
_CTASK_CYCLE_WAR	BOOL	%FX130	Cycle Task Period Exceeded Warning
_AB_SD_ER	BOOL	%FX131	Stop from abnormal operation
_MOTION_CONTROL_WAR	BOOL	%FX132	Motion Control Abnormal Warning
_ANNUM_WAR	BOOL	%FX134	Minor error detection in external device
_TASK_PRM_USAGE_OVER_WAR	BOOL	%FX135	Task Program Occupancy Excess Warning
_T20MS	BOOL	%FX192	20ms CLOCK
_T100MS	BOOL	%FX193	100ms CLOCK
_T200MS	BOOL	%FX194	200ms CLOCK
_T1S	BOOL	%FX195	1s CLOCK
_T2S	BOOL	%FX196	2s CLOCK
_T10S	BOOL	%FX197	10s CLOCK
_T20S	BOOL	%FX198	20s CLOCK
_T60S	BOOL	%FX199	60s CLOCK
_ON	BOOL	%FX201	Always ON
_OFF	BOOL	%FX202	Always OFF
_1ON	BOOL	%FX203	1 scan ON
_1OFF	BOOL	%FX204	1 scan OFF
_STOG	BOOL	%FX205	Every scan Toggle
_ERR	BOOL	%FX224	Calculation error flag
_ALL_OFF	BOOL	%FX227	All output OFF
_LER	BOOL	%FX229	Latch flag for calculation error
_ARY_IDX_ERR	BOOL	%FX247	Exceeding error from Index range when using array
_ARY_IDX_LER	BOOL	%FX248	Latch for exceeding error on Index range when using array
_UDF_STACK_ERR	BOOL	%FX249	UDF Stack Over Error Flag
_UDF_STACK_LER	BOOL	%FX250	UDF Stack Over Error Latch Flag
_CPU_TYPE	WORD	%FW18	CPU type
_CPU_VER	WORD	%FW19	CPU version
_OS_VER	DWORD	%FD10	OS version

Variable	Type	Address	Description
_OS_DATE	DWORD	%FD11	OS date
_OS_VER_PATCH	DWORD	%FD12	OS patch version
_RTC_TIME	ARRAY[0..7] OF BYTE	%FB52	RTC Time
_RTC_DATE	DATE	%FW30	Current RTC date
_RTC_WEEK	UINT	%FW31	Current RTC day
_RTC_TOD	TIME_OF_DAY	%FD16	Current time of RTC(ms unit)
_KEY	DWORD	%FD17	Current state of the local key switch
_AC_F_CNT	UINT	%FW36	Short power interruptions count
_FALS_NUM	UINT	%FW37	FALS Command Usage Area
_SYS_ERR_TYPE	WORD	%FW38	System Error Detailed Flag
_ENCODER_HW_ERR	BOOL	%FX608	Encoder Input Handling H/W Setting Error
_BACKPLANE_IF_ERR	BOOL	%FX609	Backplane Interface Error
_SERIAL_NUM	ARRAY[0..19] OF BYTE	%FB80	Serial Number
_PTASK_SCAN_MAX	UINT	%FW512	Main Task Max. Scan Time(Unit:100us)
_PTASK_SCAN_MIN	UINT	%FW513	Main Task Min. Scan Time(Unit:100us)
_PTASK_SCAN_CUR	UINT	%FW514	Main Task Current Scan Time(Unit:100us)
_CTASK_SCAN_MAX	UINT	%FW515	Cycle Task Max. Scan Time(Unit:100us)
_CTASK_SCAN_MIN	UINT	%FW516	Cycle Task Min. Scan Time(Unit:100us)
_CTASK_SCAN_CUR	UINT	%FW517	Cycle Task Current Scan Time(Unit:100us)
_PROGRAM_RATIO_MAX	UINT	%FW518	User Program Maximum Execution Occupancy (1sec)
_PROGRAM_RATIO_MIN	UINT	%FW519	User Program Minimum Execution Occupancy (1sec)
_PROGRAM_RATIO_CUR	UINT	%FW520	User Program Current Execution Occupancy (1sec)
_PTASK_CYCLE_WAR_NUM	UINT	%FW748	Main Task Period Exceeded Warning Count
_CTASK_CYCLE_WAR_NUM	UINT	%FW749	Cycle Task Period Exceeded Warning Count
_RTC_WR	BOOL	%FX20480	User RTC Setting Request
_CHK_ANC_ERR	BOOL	%FX20482	Request for significant error detection in external device
_CHK_ANC_WAR	BOOL	%FX20483	Request for minor error detection in external device
_PTASK_SCAN_WR	BOOL	%FX20486	Main Task Scan Value Initialization
_CTASK_SCAN_WR	BOOL	%FX20487	Cycle Task Scan Value Initialization
_INIT_DONE	BOOL	%FX20496	Completion of initialization task
_ANC_ERR	WORD	%FW1282	Significant error information in external device
_ANC_WAR	WORD	%FW1283	Minor error information in external device
_RTC_TIME_USER	ARRAY[0..7] OF BYTE	%FB2568	User RTC Time

A5.2 Motion Flag List

The flag displayed following areas follows. It displays the state and data of the motion controller.
The flag related to axis is displayed as "_AXxx_..."(xx indicates the relevant axis No. : Decimal) and the flag related to axis group is displayed as "_AGyy_..."(yy indicates the axis group No. : Decimal).

1) Motion Common Flag

Variable	Type	Address	Description
_MC_RUN	BOOL	%FX65536	MC RUN
_MC_STOP	BOOL	%FX65537	MC STOP
_MC_TEST	BOOL	%FX65538	MC TEST
_MC_WARNING	BOOL	%FX65539	MC Common warning occurrence
_MC_ALARM	BOOL	%FX65540	MC Common alarm occurrence
_MC_COM_ERR	BOOL	%FX65541	MC Common error occurrence
_MC_COM_ERR_CODE	WORD	%FW4097	MC Common error code
_EC_LINKUP_INFO	BOOL	%FX65600	EtherCAT Link Up/Down Information
_EC_COMM	BOOL	%FX65601	EtherCAT Communication connection state
_EC_COMM_ERR	BOOL	%FX65602	EtherCAT Communication timeout error
_EC_PDO_ERR_CNT	UINT	%FW4102	EtherCAT PDO error count
_EC_SLAVE_RDY	ARRAY[0..63] OF BOOL	%FX65664	EtherCAT Slave ready
_EC_SDO_BUSY	ARRAY[0..63] OF BOOL	%FX65792	EtherCAT Slave SDO processing busy
_EC_SDO_ERR	ARRAY[0..63] OF BOOL	%FX65920	EtherCAT Slave SDO processing error
_EC_LINE_FAIL	ARRAY[0..63] OF BOOL	%FX66048	EtherCAT Cable disconnection state
_EC_MASTER_STATE	BYTE	%FB8264	EtherCAT master STATE
_EC_SLAVE_NUM	WORD	%FW4133	Number of connected EtherCAT Slave
_EC_ERR_INFO1	STRING	%FB8272	EtherCAT error information1
_EC_ERR_INFO2	STRING	%FB8304	EtherCAT error information2
_EC_TRANSMITTED_OK	UDINT	%FD2084	EtherCAT Number of frames transmitted
_EC_RECEIVED_OK	UDINT	%FD2085	EtherCAT Number of frames received
_EC_CRCERR_CNT	UDINT	%FD2086	EtherCAT Receive CRC error frame
_EC_COLLISION_CNT	UDINT	%FD2087	EtherCAT Number of collision frames
_EC_CARRIER_SENSE_ERR	UDINT	%FD2088	EtherCAT Carrier sense error
_EC_LINKOFF_CNT	UDINT	%FD2089	EtherCAT Number of Link Off
_EC_OVERSIZE_FRAME	UDINT	%FD2090	EtherCAT Receive oversize frames

Variable	Type	Address	Description
_EC_UNDERSIZE_FRAME	UDINT	%FD2091	EtherCAT Receive undersize frames
_EC_JABBER_FRAME	UDINT	%FD2092	EtherCAT Receive jabber frame
_EC_PDO_CUR_TRANSCYCLE	UDINT	%FD2093	EtherCAT PDO transfer cycle ns
_EC_PDO_MAX_TRANSCYCLE	UDINT	%FD2094	EtherCAT Maximum PDO transfer cycle ns
_EC_PDO_MIN_TRANSCYCLE	UDINT	%FD2095	EtherCAT Minimum PDO transfer cycle ns
_EC_PDO_TRANS_JITTER	UDINT	%FD2096	EtherCAT PDO frame transfer jitter ns
_EC_PDO_ERR_CNT_TOTAL	UDINT	%FD2097	PDO working counter error number
_EC_LOST_FRAME	UDINT	%FD2098	EtherCAT Packet Loss
_EC_PDO_ERR_CNT_MAX	UDINT	%FD2099	EtherCAT PDO Error Count(Max.)
_EC_ERR_INFO3	STRING	%FB8424	EtherCAT Error3

Reference) The flags of _AXxx_HOME(Flag used at home return command) and _AXxx_Homing(Operation status of PLC open standard) indicate the same state.

2) Motion Axis Flag

The address information is the flag memory of axis 01. The address has 2,048bit (32LREAL) offsets per axis.

Variable	Type	Address	Description
_AXxx_RDY	BOOL	%FX73728	Axis xx ready
_AXxx_WARNING	BOOL	%FX73729	Axis xx warning occurrence
_AXxx_ALARM	BOOL	%FX73730	Axis xx alarm occurrence
_AXxx_SV_ON	BOOL	%FX73731	Axis xx servo On/Off
_AXxx_SV_RDY	BOOL	%FX73732	Axis xx servo ready
_AXxx_MSTSLV_STS	BOOL	%FX73733	Axis xx master/slave status
_AXxx_NC	BOOL	%FX73734	Axis xx NC operation
_AXxx_MST_INFO	UINT	%FW4609	Axis xx master axis information
_AXxx_AXIS_TYPE	UINT	%FW4610	Axis xx axis type
_AXxx_LINKED_NODE	UINT	%FW4611	Axis xx connected node information
_AXxx_LINKED_SLOT	UINT	%FW4612	Axis xx connected slot information
_AXxx_UNIT	UINT	%FW4613	Axis xx axis unit
_AXxx_VEL_UNIT	UINT	%FW4614	Axis xx speed unit
_AXxx_AX_ERR	WORD	%FW4615	Axis xx error code
_AXxx_SVON_INCMPL	BOOL	%FX73856	Axis xx servo on incomplete
_AXxx_COMM_WARN	BOOL	%FX73857	Axis xx communication warning
_AXxx_DEV_WARN	BOOL	%FX73858	Axis xx deviation warning
_AXxx_SV_ERR	BOOL	%FX73872	Axis xx servo drive error
_AXxx_HW_POT	BOOL	%FX73873	Axis xx positive limit detection
_AXxx_HW_NOT	BOOL	%FX73874	Axis xx negative limit detection
_AXxx_SW_POT	BOOL	%FX73875	Axis xx S/W positive limit detection

Variable	Type	Address	Description
_AXxx_SW_NOT	BOOL	%FX73876	Axis xx S/W negative limit detection
_AXxx_SV_OFF	BOOL	%FX73877	Axis xx execution error of operation command in servo-off state
_AXxx_POS_OVR	BOOL	%FX73878	Axis xx exceeds the set range of positioning travel amount
_AXxx_VEL_OVR	BOOL	%FX73879	Axis xx exceeds the maximum velocity
_AXxx_DEV_ERR	BOOL	%FX73880	Axis xx deviation alarm
_AXxx_HOME_INCMPL	BOOL	%FX73881	Axis xx Execution of absolute position command in undetermined HOME
_AXxx_COMM_ERR	BOOL	%FX73882	Axis xx communication alarm
_AXxx_BUSY	BOOL	%FX73888	Axis xx busy state of motion command
_AXxx_PAUSE	BOOL	%FX73889	Axis xx pause state of motion command (velocity is zero)
_AXxx_STOP	BOOL	%FX73890	Axis xx stop state by the stop command
_AXxx_CMD_FAIL	BOOL	%FX73891	Axis xx abnormal completion of motion command
_AXxx_CMD_CMPL	BOOL	%FX73892	Axis xx normal completion of motion command

Variable	Type	Address	Description
_AXxx_DIR	BOOL	%FX73893	Axis xx operation direction
_AXxx_JOG	BOOL	%FX73894	Axis xx JOG operation
_AXxx_HOME	BOOL	%FX73895	Axis xx Homing operation
_AXxx_POS_CTRL	BOOL	%FX73896	Axis xx position control operation
_AXxx_VEL_CTRL	BOOL	%FX73897	Axis xx velocity control operation
_AXxx_TRQ_CTRL	BOOL	%FX73898	Axis xx torque control operation
_AXxx_LINTP	BOOL	%FX73899	Axis xx linear interpolation operation
_AXxx_CINTP	BOOL	%FX73900	Axis xx circular interpolation operation
_AXxx_SYNC	BOOL	%FX73901	Axis xx synchronous control operation
_AXxx_COORD	BOOL	%FX73902	Axis xx coordinated operation
_AXxx_POS_CMPL	BOOL	%FX73920	Axis xx positioning completion
_AXxx_INPOS	BOOL	%FX73921	Axis xx inposition detection
_AXxx_LATCH_CMPL	BOOL	%FX73922	Axis xx latch completion
_AXxx_HOME_CMPL	BOOL	%FX73923	Axis xx homing completion
_AXxx_Disabled	BOOL	%FX73936	Axis xx Disabled state
_AXxx_Standstill	BOOL	%FX73937	Axis xx Standstill state
_AXxx_Discrete	BOOL	%FX73938	Axis xx Discrete state
_AXxx_Continuous	BOOL	%FX73939	Axis xx Continuous state
_AXxx_Synchronized	BOOL	%FX73940	Axis xx Synchronized state
_AXxx_Homing	BOOL	%FX73941	Axis xx Homing state

Variable	Type	Address	Description
_AXxx_Stopping	BOOL	%FX73942	Axis xx Stopping state
_AXxx_ErrorStop	BOOL	%FX73943	Axis xx ErrorStop state
_AXxx_CMD_TPOS	LREAL	%FL1156	Axis xx target position
_AXxx_CMD_CPOS	LREAL	%FL1157	Axis xx command position of current scan
_AXxx_CMD_VEL	LREAL	%FL1158	Axis xx command velocity
_AXxx_CMD_ACCDEC	LREAL	%FL1159	Axis xx command acceleration/deceleration
_AXxx_CMD_JERK	LREAL	%FL1160	Axis xx command jerk
_AXxx_CMD_TRQ	LREAL	%FL1161	Axis xx command torque
_AXxx_ACT_POS	LREAL	%FL1162	Axis xx actual current position
_AXxx_ACT_VEL	LREAL	%FL1163	Axis xx actual current velocity
_AXxx_ACT_TRQ	LREAL	%FL1164	Axis xx actual current torque
_AXxx_POS_DEV	LREAL	%FL1165	Axis xx position deviation
_AXxx_DRV_ALARM	BOOL	%FX74624	Axis xx drive alarm state
_AXxx_DRV_WARNING	BOOL	%FX74625	Axis xx drive warning state
_AXxx_DRV_SV_ON	BOOL	%FX74626	Axis xx servo on status
_AXxx_DRV_POT	BOOL	%FX74627	Axis xx positive limit input
_AXxx_DRV_NOT	BOOL	%FX74628	Axis xx negative limit input
_AXxx_DRV_HOME	BOOL	%FX74629	Axis xx home input
_AXxx_DRV_LATCH1	BOOL	%FX74630	Axis xx LATCH1 input
_AXxx_DRV_LATCH2	BOOL	%FX74631	Axis xx LATCH2 input
_AXxx_DRV_PARAMBUSY	BOOL	%FX74632	Axis xx read/write operations of the SDO parameter
_AXxx_DRV_IN	DWORD	%FD2333	Axis xx drive inputs
_AXxx_DRV_ERR	WORD	%FW4668	Axis xx drive error code
_AXxx_CMDBUF_FULL	BOOL	%FX73951	Axis xx Buffered Command Buffer Full
_AXxx_CMDBUF_QUEUED	UINT	%FW4622	Axis xx Buffered Command Queued Count
_AXxx_CMDBUF_FREE	UINT	%FW4623	Axis xx Buffered command execution count

Reference) The flags of _AXxx_HOME(Flag used at home return command) and _AXxx_Homing(Operation status of PLC open standard) indicate the same state.

3) Motion Axis Group Flag

The address information is the flag memory of axis 01. The address has 5,120bit (80LREAL) offsets per axis.

Variable	Type	Address	Description
_AGxx_RDY	BOOL	%FX212992	Axis group xx ready
_AGxx_WARNING	BOOL	%FX212993	Axis group xx warning occurrence
_AGxx_ALARM	BOOL	%FX212994	Axis group xx alarm occurrence
_AGxx_SV_ON	BOOL	%FX212995	Axis group xx servo On/Off
_AGxx_SV_RDY	BOOL	%FX212996	Axis group xx servo ready
_AGxx_ERR	WORD	%FW13313	Axis group xx error code

Variable	Type	Address	Description
_AGxx_BUSY	BOOL	%FX213024	Axis group xx busy state of motion command
_AGxx_PAUSE	BOOL	%FX213025	Axis group xx pause state of motion command (velocity is zero)
_AGxx_STOP	BOOL	%FX213026	Axis group xx stop state by the stop command
_AGxx_CMD_FAIL	BOOL	%FX213027	Axis group xx command error exit status
_AGxx_CMD_CMPL	BOOL	%FX213028	Axis group xx command execution complete
_AGxx_LINTP	BOOL	%FX213029	Axis group xx linear interpolation operation
_AGxx_CINTP	BOOL	%FX213030	Axis group xx circular interpolation operation
_AGxx_HOME	BOOL	%FX213031	Axis group xx homing operation
_AGxx_SYNC	BOOL	%FX213032	Axis group xx synchronization operation
_AGxx_TLINTP	BOOL	%FX213033	Axis group xx coordinated time operation
_AGxx_CDMOVE	BOOL	%FX213034	Axis group xx coordinated direct operation
_AGxx_CCINTP	BOOL	%FX213035	Axis group xx coordinated circular interpolation operation
_AGxx_POS_CMPL	BOOL	%FX213056	Axis group xx positioning completion
_AGxx_Disabled	BOOL	%FX213072	Axis group xx Disabled state
_AGxx_Standby	BOOL	%FX213073	Axis group xx Standby state
_AGxx_Moving	BOOL	%FX213074	Axis group xx Moving state
_AGxx_Homing	BOOL	%FX213075	Axis group xx Homing state
_AGxx_Stopping	BOOL	%FX213076	Axis group xx Stopping state
_AGxx_ErrorStop	BOOL	%FX213077	Axis group xx ErrorStop state
_AGxx_CMD_TPOS	ARRAY[0..9] OF LREAL	%FL3330	Axis group xx target position
_AGxx_CMD_CPOS	ARRAY[0..9] OF LREAL	%FL3340	Axis group xx command position of current scan
_AGxx_CMD_VEL	LREAL	%FL3350	Axis group xx target velocity
_AGxx_CMD_ACCDEC	LREAL	%FL3351	Axis group xx command acc./dec.
_AGxx_CMD_JERK	LREAL	%FL3352	Axis group xx command jerk
_AGxx_ACT_POS	ARRAY[0..9] OF LREAL	%FL3353	Axis group xx actual current position
_AGxx_ACT_VEL	LREAL	%FL3363	Axis group xx actual current velocity
_AGxx_CFG_AX_NUM	UINT	%FW13456	Axis group xx number of axes
_AGxx_CMDBUF_FULL	BOOL	%FX213087	Axis group xx Buffered Command Buffer Full
_AGxx_CMDBUF_QUEUED	UINT	%FW13318	Axis group xx Buffered Command Queued Count
_AGxx_CMDBUF_FREE	UINT	%FW13319	Axis group xx Buffered command execution count
_AGxx_CFG_A1	UINT	%FW13458	Axis group xx axis number of composition axis1
_AGxx_CFG_A2	UINT	%FW13459	Axis group xx axis number of composition axis2
_AGxx_CFG_A3	UINT	%FW13460	Axis group xx axis number of composition axis3
_AGxx_CFG_A4	UINT	%FW13461	Axis group xx axis number of composition axis4
_AGxx_CFG_A5	UINT	%FW13462	Axis group xx axis number of composition axis5
_AGxx_CFG_A6	UINT	%FW13463	Axis group xx axis number of composition axis6
_AGxx_CFG_A7	UINT	%FW13464	Axis group xx axis number of composition axis7
_AGxx_CFG_A8	UINT	%FW13465	Axis group xx axis number of composition axis8

Variable	Type	Address	Description
_AGxx_CFG_A9	UINT	%FW13466	Axis group xx axis number of composition axis9
_AGxx_CFG_A10	UINT	%FW13467	Axis group xx axis number of composition axis10
_AGxx_MTCP_Px	LREAL	%FL3367	Axis group xx X axis position(MCS)
_AGxx_MTCP_Py	LREAL	%FL3368	Axis group xx Y axis position(MCS)
_AGxx_MTCP_Pz	LREAL	%FL3369	Axis group xx Z axis position(MCS)
_AGxx_MTCP_A	LREAL	%FL3370	Axis group xx X axis rotation(MCS)
_AGxx_MTCP_B	LREAL	%FL3371	Axis group xx X axis rotation(MCS)
_AGxx_MTCP_C	LREAL	%FL3372	Axis group xx Z axis rotation(MCS)
_AGxx_PTCP_Px	LREAL	%FL3373	Axis group xx X axis position(PCS)
_AGxx_PTCP_Py	LREAL	%FL3374	Axis group xx Y axis position(PCS)
_AGxx_PTCP_Pz	LREAL	%FL3375	Axis group xx Z axis position(PCS)
_AGxx_PTCP_A	LREAL	%FL3376	Axis group xx X axis rotation(PCS)
_AGxx_PTCP_B	LREAL	%FL3377	Axis group xx Y axis rotation(PCS)
_AGxx_PTCP_C	LREAL	%FL3378	Axis group xx Z axis rotation(PCS)

4) Slave Flag

Variable	Type	Address	Description
_SLVxx_EC_STATE	SINT	%FB47104	EtherCAT Slave xx STATE
_SLVxx_LINK_STATUS	BYTE	%FB47105	EtherCAT Slave xx link information
_SLVxx_ERROR	WORD	%FW23553	EtherCAT Slave xx error
_SLVxx_VENDOR_ID	DWORD	%FD11777	EtherCAT Slave xx Vendor ID
_SLVxx_PRODUCT_CODE	DWORD	%FD11778	EtherCAT Slave xx Product Code
_SLVxx_REVISION_NUMBER	DWORD	%FD11779	EtherCAT Slave xx Revision Number
_SLVxx_ALStatus	WORD	%FW23563	EtherCAT slave xx AL state
_SLVxx_ALStatusCode	WORD	%FW23564	EtherCAT Slave xx AL error code
_SLVxx_DLStatus	WORD	%FW23565	EtherCAT Slave xx link state
_SLVxx_LinkLostCount	DWORD	%FD11783	A Port link disconnection count
_SLVxx_InValidFrameCounterA	BYTE	%FB47136	EtherCAT Slave xx A port abnormal frame counter
_SLVxx_RxErrorCounterA	BYTE	%FB47137	EtherCAT Slave xx A port physical layer error number
_SLVxx_InValidFrameCounterB	BYTE	%FB47138	EtherCAT Slave xx B port abnormal frame counter
_SLVxx_RxErrorCounterB	BYTE	%FB47139	EtherCAT Slave xx B port physical layer error number
_SLVxx_InValidFrameCounterC	BYTE	%FB47140	EtherCAT Slave xx C port abnormal frame counter
_SLVxx_RxErrorCounterC	BYTE	%FB47141	EtherCAT Slave xx C port physical layer error number

_SLVxx_InValidFrameCounterD	BYTE	%FB47142	EtherCAT Slave xx D port abnormal frame counter
_SLVxx_RxErrorCounterD	BYTE	%FB47143	EtherCAT Slave xx D port physical layer error number
_SLVxx_ForwardedRXErrCounter	DWORD	%FD11786	Number of abnormal frames delivered

5) NC Channel Flag

It displays the state of NC channel. NC channel flag is displayed as "_NCyy_..."

(yy indicates the NC channel No.(Decimal))

Variable	Type	Address	Description
_NCyy_Ready	BOOL	%FX524288	NC Ch. yy NC ready
_NCyy_Warning	BOOL	%FX524289	NC Ch. yy warning occurrence
_NCyy_Alarm	BOOL	%FX524290	NC Ch. yy alarm occurrence
_NCyy_ResetStatus	BOOL	%FX524291	NC Ch. yy reset state
_NCyy_CycStartBegin	BOOL	%FX524292	NC Ch. yy cycle start begin information
_NCyy_CycStartFinish	BOOL	%FX524293	NC Ch. yy cycle start finish information
_NCyy_TargetQtyCmpl	BOOL	%FX524294	NC Ch. yy target quantity reached signal
_NCyy_PrgmNormalCmpl	BOOL	%FX524295	NC Ch. yy normal completion of program execution
_NCyy_PwrFailInAuto	BOOL	%FX524296	NC Ch. yy power failure in automatic operation
_NCyy_ErrorCode	WORD	%FW32770	NC Ch. yy error code
_NCyy_IPR_HeartBeat	UDINT	%FD16386	NC Ch. yy IPR HeartBeat
_NCyy_IPR_Run	BOOL	%FX524384	NC Ch. yy IPR operation state (0:stop, 1:running)
_NCyy_IPR_WaitEoM	BOOL	%FX524400	NC Ch. yy waiting end of motion state (0: not waiting, 1:waiting)
_NCyy_IPR_EndOfMot	UINT	%FW32776	NC Ch. yy end of motion
_NCyy_IPR_AfBufSts	UINT	%FW32777	NC Ch. yy AutoFIFO buffer state (0: empty, another: buffer usage)
_NCyy_IPR_ErrorCode	UINT	%FW32778	NC Ch. yy IPR error code
_NCyy_PA_ErrorCode	UINT	%FW32779	NC Ch. yy program access error code
_NCyy_IPR_AlarmSts	ARRAY[0..4] OF DWORD	%FD16390	NC Ch. yy IPR alarm information
_NCyy_CycleStart	BOOL	%FX524672	NC Ch. yy cycle start state
_NCyy_FeedHold	BOOL	%FX524673	NC Ch. yy feed hold state
_NCyy_AutoOperation	BOOL	%FX524674	NC Ch. yy automatic operation state
_NCyy_RetraceMove	BOOL	%FX524675	NC Ch. yy retrace move state
_NCyy_RapidTrvsOpr	BOOL	%FX524736	NC Ch. yy rapid traverse operation
_NCyy_CuttingFeedOpr	BOOL	%FX524737	NC Ch. yy cutting feed operation

_NCyy_ConstSurfSpeed	BOOL	%FX524738	NC Ch. yy constant surf speed
_NCyy_TargetVelocity	LREAL	%FL8200	NC Ch. yy target velocity (F command value)
_NCyy_CmdVelocity	LREAL	%FL8201	NC Ch. yy command velocity
_NCyy_TVelOfSpindle	LREAL	%FL8203	NC Ch. yy spindle target velocity (S command value)
_NCyy_CVelOfSpindle	LREAL	%FL8204	NC Ch. yy spindle command velocity
_NCyy_FeedOverride	LREAL	%FL8206	NC Ch. yy feed override
_NCyy_RapidOverride	LREAL	%FL8207	NC Ch. yy rapid override

Variable	Type	Address	Description
_NCyy_SpindleOverride	LREAL	%FL8208	NC Ch. yy spindle override
_NCyy_SpindleStop	BOOL	%FX525376	NC Ch. yy spindle stop state
_NCyy_SpindleCW	BOOL	%FX525377	NC Ch. yy spindle CW operation
_NCyy_SpindleCCW	BOOL	%FX525378	NC Ch. yy spindle CCW operation
_NCyy_SpindleOrient	BOOL	%FX525379	NC Ch. yy spindle orientation operation
_NCyy_SpindleCVelAgr	BOOL	%FX525380	NC Ch. yy spindle command velocity reached signal
_NCyy_SpindleZeroVel	BOOL	%FX525381	NC Ch. yy spindle zero velocity reached signal
_NCyy_SpindlePosCtrl	BOOL	%FX525382	NC Ch. yy spindle position control signal
_NCyy_SpindleSSCtrl	BOOL	%FX525383	NC Ch. yy master axis SS control signal
_NCyy _MainSpindle	UDINT	%FW32840	NC Ch. yy main spindle axis number
_NCyy_DwellCount	UDINT	%FD16422	NC Ch. yy dwell count
_NCyy_ErrorBlockNum	UDINT	%FD16423	NC Ch. yy error block number
_NCyy_BlockCmdType	UINT	%FW32848	NC Ch. yy command type of current block
_NCyy_CurrentToolNum	UINT	%FW32856	NC Ch. yy current tool number
_NCyy_ToolRadiusComp	UINT	%FW32857	NC Ch. yy offset number of current tool radius compensation
_NCyy_ToolLengthComp	UINT	%FW32858	NC Ch. yy offset number of current tool length compensation
_NCyy_McodeStrobe	BOOL	%FX526080	NC Ch. yy M code output strobe signal
_NCyy_McodeDistCmpl	BOOL	%FX526081	NC Ch. yy M code distribution complete signal
_NCyy_McodeM00	BOOL	%FX526082	NC Ch. yy special M code output signal(M00)
_NCyy_McodeM01	BOOL	%FX526083	NC Ch. yy special M code output signal(M01)
_NCyy_McodeM02	BOOL	%FX526084	NC Ch. yy special M code output signal(M02)
_NCyy_McodeM30	BOOL	%FX526085	NC Ch. yy special M code output signal(M30)
_NCyy_McodeData	UDINT	%FD16441	NC Ch. yy M code data output
_NCyy_ScodeStrobe	BOOL	%FX526144	NC Ch. yy S code output strobe signal
_NCyy_ScodeDistCmpl	BOOL	%FX526145	NC Ch. yy S code distribution complete signal

_NCyy_ScodeData	UDINT	%FD16443	NC Ch. yy S code data output
_NCyy_TcodeStrobe	BOOL	%FX526208	NC Ch. yy T code output strobe signal
_NCyy_TcodeDistCmpl	BOOL	%FX526209	NC Ch. yy T code distribution complete signal
_NCyy_TcodeData	UDINT	%FD16445	NC Ch. yy T code data output
_NCyy_CycleTime	REAL	%FD16446	NC Ch. yy machining cycle time
_NCyy_TotalRunTime	REAL	%FD16447	NC Ch. yy total machining cycle time
_NCyy_PartCount	UDINT	%FD16448	NC Ch. yy machining quantity
_NCyy_PartCountByM99	UDINT	%FD16449	NC Ch. yy M99 machining quantity at repeat machining
_NCyy_MainProgram	STRING	%FB65800	NC Ch. yy main program name
_NCyy_CurrentProgram	STRING	%FB65832	NC Ch. yy current running program name
_NCyy_MainBlkNum	UDINT	%FD16466	NC Ch. yy block number of main program
_NCyy_CurrentBlkNum	UDINT	%FD16468	NC Ch. yy block number of current running program

Variable	Type	Address	Description
_NCyy_ModalG_OneShot	REAL	%FD16476	NC Ch. yy G code modal value group 0 - One shot
_NCyy_ModalG_Motion	REAL	%FD16477	NC Ch. yy G code modal value group 1 - Motion
_NCyy_ModalG_CmdMode	REAL	%FD16479	NC Ch. yy G code modal value group 3 - Command mode (ABS or INC)
_NCyy_ModalG_Mirror	REAL	%FD16480	NC Ch. yy G code modal value group 4 - Mirror
_NCyy_ModalG_Feed	REAL	%FD16481	NC Ch. yy G code modal value group 5 - Feed mode
_NCyy_ModalG_Unit	REAL	%FD16482	NC Ch. yy G code modal value group 6 - Unit
_NCyy_ModalG_TRComp	REAL	%FD16483	NC Ch. yy G code modal value group 7 - Tool radius compensation
_NCyy_ModalG_Stroke	REAL	%FD16485	NC Ch. yy G code modal value group 9 - Stroke check
_NCyy _ModalG_Scale	REAL	%FD16487	NC Ch. yy G code modal value group 11 - Scale
_NCyy _ModalG_Macro	REAL	%FD16488	NC Ch. yy G code modal value group 12 - Macro
_NCyy_ModalG_TLComp	REAL	%FD16489	NC Ch. yy G code modal value group 13 - Tool length compensation
_NCyy_ModalG_WpCoord	REAL	%FD16490	NC Ch. yy G code modal value group 14 - Workpiece coordinate system
_NCyy_ModalG_CutMode	REAL	%FD16491	NC Ch. yy G code modal value group 15 - CutMode
_NCyy_ModalG_Plane	REAL	%FD16492	NC Ch. yy G code modal value group 16 - Circular plane
_NCyy_ModalG_RPolar	REAL	%FD16496	NC Ch. yy G code modal value group 20 - Reverse polar coordinate interpolation
_NCyy_ModalG_CylIntp	REAL	%FD16498	NC Ch. yy G code modal value group 22 - Cylindrical interpolation

_NCyy _ModalG_Skip	REAL	%FD16499	NC Ch. yy G code modal value group 23 - Skip
_NCyy_ModalFeed	LREAL	%FL8254	NC Ch. yy modal feed
_NCyy_ModalScode	UDINT	%FD16510	NC Ch. yy modal S code
_NCyy_ModalSpindleM	UDINT	%FD16511	NC Ch. yy modal spindle M code
_NCyy_ModelMcode	UDINT	%FD16512	NC Ch. yy Modal M Code
_NCyy_ModelHcode	UDINT	%FD16513	NC Ch. yy Modal H Code
_NCyy_ModalWorkCoord	UDINT	%FD16514	NC Ch. yy Modal Workpiece Coordinate

6) NC Channel Flag

It displays the state of axis configured on the NC channel. NC channel/axis flag is displayed as “_NCyy_X...”, “NCyy_Y…”

(yy indicates the NC channel No.(Decimal) and X,Y,Z,A,B,C,U,V,W is the assigned axis)

Variable	Type	Address	Description
_NC01X_Ready	BOOL	%FX532480	NC Ch. 01 axis X ready
_NC01X_Warning	BOOL	%FX532481	NC Ch. 01 axis X warning occurrence
_NC01X_Alarm	BOOL	%FX532482	NC Ch. 01 axis X alarm occurrence
_NC01X_ServoOn	BOOL	%FX532483	NC Ch. 01 axis X servo On/Off
_NC01X_ServoReady	BOOL	%FX532484	NC Ch. 01 axis X servo ready
_NC01X_ServoAlarm	BOOL	%FX532485	NC Ch. 01 axis X servo alarm occurrence
_NC01X_OprRdy	BOOL	%FX532544	NC Ch. 01 axis X operation ready
_NC01X_FeedMode	BOOL	%FX532552	NC Ch. 01 axis X axis feed mode (0: linear axis, 1: rotation axis)
_NC01X_LinkedAxNum	UINT	%FW33285	NC Ch. 01 axis X actual axis number of IPR axis
_NC01X_Busy	BOOL	%FX532608	NC Ch. 01 axis X busy state
Variable	Type	Address	Description
_NC01X_Direction	BOOL	%FX532609	NC Ch. 01 axis X operation direction
_NC01X_ForwardRun	BOOL	%FX532610	NC Ch. 01 axis X running to positive direction
_NC01X_ReverseRun	BOOL	%FX532611	NC Ch. 01 axis X running to negative direction
_NC01X_RapidTraverse	BOOL	%FX532612	NC Ch. 01 axis X rapid traverse operation
_NC01X_CuttingFeed	BOOL	%FX532613	NC Ch. 01 axis X cutting feed operation
_NC01X_Homing	BOOL	%FX532614	NC Ch. 01 axis X homing operation
_NC01X_SpindleRun	BOOL	%FX532615	NC Ch. 01 axis X spindle operation
_NC01X_PosCmpl	BOOL	%FX532672	NC Ch. 01 axis X positioning completion
_NC01X_Inposition	BOOL	%FX532673	NC Ch. 01 axis X in-position detection
_NC01X_HomeCmpl	BOOL	%FX532675	NC Ch. 01 axis X homing completion
_NC01X_Mirror	BOOL	%FX532736	NC Ch. 01 axis X mirror signal
_NC01X_CmdPosInWC	LREAL	%FL8325	NC Ch. 01 axis X command position in workpiece coordinate system
_NC01X_CmdPosInRC	LREAL	%FL8326	NC Ch. 01 axis X command position in relative coordinate system
_NC01X_ActualVel	LREAL	%FL8327	NC Ch. 01 axis X actual current velocity

_NC01X_RemDistance	LREAL	%FL8329	NC Ch. 01 axis X remaining distance
_NC01X_PosDeviation	LREAL	%FL8330	NC Ch. 01 axis X servo position deviation (tracking error)
_NC01X_WcOffset	LREAL	%FL8334	NC Ch. 01 axis X offset value of workpiece coordinate system
_NC01X_WcBasicOffset	LREAL	%FL8335	NC Ch. 01 axis X basic offset value of workpiece coordinate system
_NC01X_WcShiftOffset	LREAL	%FL8336	NC Ch. 01 axis X shift offset value of workpiece coordinate system
_NC01X_LocalWcOffset	LREAL	%FL8337	NC Ch. 01 axis X offset value of local workpiece coordinate system
_NC01X_CmdPosInMC	LREAL	%FL8339	NC Ch. 01 axis X command position in machine coordinate system
_NC01X_ActualPosInMC	LREAL	%FL8341	NC Ch. 01 axis X actual current position in machine coordinate system
_NC01X_SkipPosInMC	LREAL	%FL8342	NC Ch. 01 axis X skip position in machine coordinate system
_NC01X_AxErr	WORD	%FW33372	NC Ch. 01 axis X error code
_NC01X_DrvErr	WORD	%FW33373	NC Ch. 01 axis X drive error code

7) SD Memory Flag

Variable	Type	Address	Description
_SD_Attach	BOOL	%KX8256	SD attachment state
_SD_Rdy	BOOL	%KX8257	SD memory ready
_SD_Err	BOOL	%KX8258	SD memory error
_SD_Init	BOOL	%KX8259	SD memory initializing state
_SD_Closing	BOOL	%KX8260	SD memory closing state
_SD_FATErr	BOOL	%KX8261	File System Error
_SD_AutoLogAct	BOOL	%KX8262	Act Auto-logging
_SD_Busy	BOOL	%KX8263	SD memory busy state
_SD_SpaceWarn	BOOL	%KX8264	SD memory insufficient state
_SD_Detach	BOOL	%KX8265	SD memory detachment state
_SD_VolTot	UDINT	%KD259	SD memory storage capacity(GB)
_SD_VolAvail	UDINT	%KD260	Available storage capacity(KB)
_SD_Ecode	WORD	%KW522	SD memory error code
_SD_FmtInfo	WORD	%KW523	SD memory format information
_SD_FmtRun	BOOL	%KX8368	SD memory format operation state
_SD_FmtDone	BOOL	%KX8369	SD memory format complete state
_SD_FmtErr	BOOL	%KX8370	SD memory format fail state
_SD_FmtEcode	WORD	%KW524	SD memory format error code
_SD_FmtProgress	WORD	%KW525	SD memory format progress ratio(%)

Variable	Type	Address	Description
_SD_AttachCnt	WORD	%KW526	SD memory attachment count
_SD_DetachCnt	WORD	%KW527	SD memory detachment count
_SD_AddfuncAct	BOOL	%KX8640	SD additional function operation state
_SD_AddfuncErr	BOOL	%KX8641	SD additional function error state
_SD_AddfuncDone	BOOL	%KX8642	SD additional function complete state
_SD_CmpResult	BOOL	%KX8643	SD result of comparison
_SD_AddfuncKind	WORD	%KW541	SD type of additional function
_SD_AddfuncEcode	WORD	%KW542	SD additional function error code

8) Data Log Flag

Variable	Type	Address	Description
_DL00_Enable	BOOL	%KX8224	Group 00 datalog enable state
_DL00_Rdy	BOOL	%KX8960	Group 00 datalog ready
_DL00_Act	BOOL	%KX8961	Group 00 datalog operation state
_DL00_Err	BOOL	%KX8962	Group 00 datalog error state
_DL00_Stoping	BOOL	%KX8963	Group 00 datalog stoping state
_DL00_Finish	BOOL	%KX8964	Group 00 datalog finish state
_DL00_Trig	BOOL	%KX8965	Group 00 trigger occurrence state
_DL00_TrigDone	BOOL	%KX8966	Group 00 trigger complete state
_DL00_Evt	BOOL	%KX8967	Group 00 event occurrence state
_DL00_Ovf	BOOL	%KX8968	Group 00 buffer overflow state
_DL00_Ecode	WORD	%KW561	Group 00 datalog error code
_DL00_FileIdx	WORD	%KW562	Group 00 datalog file index number
_DL00_FileRollcnt	WORD	%KW563	Group 00 overwrite count
_DL00_FileSize	UDINT	%KD282	Group 00 file size(Byte)
_DL00_DataRow	UDINT	%KD283	Group 00 data row number
_DL00_RemainBuf	UDINT	%KD284	Group 00 remaining buffer size(Byte)
_DL00_WaitingData	UDINT	%KD285	Group 00 waiting data size(Byte)
_DL00_OvfCnt	WORD	%KW572	Group 00 buffer overflow count
_DL00_TrigCnt	WORD	%KW573	Group 00 trigger occurrence count
_DL00_TrigOvlap	WORD	%KW574	Group 00 trigger overlap count
_DL00_EvtgCnt	WORD	%KW575	Group 00 event occurrence count

9) Encoder Flag

Variable	Type	Address	Description
_ENC1_POS	LREAL	%KL0	Encoder1 input position
_ENC2_POS	LREAL	%KL1	Encoder2 input position
_ENC1_UNIT	UINT	%KW8	Encoder1 unit (0:pulse, 1:mm, 2:inch, 3:degree)
_ENC2_UNIT	UINT	%KW9	Encoder2 unit (0:pulse, 1:mm, 2:inch, 3:degree)

Variable	Type	Address	Description
_ENC1_VEL	LREAL	%KL3	Encoder1 Speed
_ENC2_VEL	LREAL	%KL4	Encoder2 Speed
_ENC1_POS_LATCH	LREAL	%KL5	Encoder1 input position latch value
_ENC2_POS_LATCH	LREAL	%KL6	Encoder2 input position latch value

10) P2P Flag

Variable	Type	Address	Description
_P2Pn_NDRxx	BOOL	XG5000 Global/ Direct Variable P2P	P2P parameter n / xx block service normal completion
_P2Pn_ERRxx	BOOL		P2P parameter n / xx block service abnormal completion
_P2Pn_STATUSxx	WORD		P2P parameter n / xx error code of block service abnormal completion
_P2Pn_SVCCNTxx	DWORD		P2P parameter n / xx number of block service normal completion
_P2Pn_ERRCNTxx	DWORD		P2P parameter n / xx number of block service abnormal completion

Warranty

1. Warranty Period
   The product you purchased will be guaranteed for 18 months from the date of manufacturing.

2. Scope of Warranty
   Any trouble or defect occurring for the above-mentioned period will be partially replaced or repaired. However, please note the following cases will be excluded from the scope of warranty.

   (1) Any trouble attributable to unreasonable condition, environment or handling otherwise specified in the manual,
   (2) Any trouble attributable to others' products,
   (3) If the product is modified or repaired in any other place not designated by the company,
   (4) Due to unintended purposes
   (5) Owing to the reasons unexpected at the level of the contemporary science and technology when delivered.
   (6) Not attributable to the company; for instance, natural disasters or fire

3. Since the above warranty is limited to PLC unit only, make sure to use the product considering the safety for system configuration or applications.

Environmental Policy

LS ELECTRIC Co., Ltd supports and observes the environmental policy as below.

Environmental Management

LS ELECTRIC considers the environmental preservation as the preferential management subject and every staff of LS ELECTRIC use the reasonable endeavors for the pleasurably environmental preservation of the earth.

About Disposal

LS ELECTRIC' PLC unit is designed to protect the environment. For the disposal, separate aluminum, iron and synthetic resin (cover) from the product as they are reusable.

www.lselectric.co.kr

LS ELECTRIC Co., Ltd.

■ Headquarter

LS-ro 127(Hogye-dong) Dongan-gu, Anyang-si, Gyeonggi-Do, 14119, Korea

■ Seoul Office

LS Yongsan Tower, 92, Hangang-daero, Yongsan-gu, Seoul, 04386, Korea

Tel: 82-2-2034-4033, 4888, 4703 Fax: 82-2-2034-4588

E-mail: automation@lselectric.co.kr

■ Factory

56, Samseong 4-gil, Mokcheon-eup, Dongnam-gu, Cheonan-si, Chungcheongnam-do, 31226, Korea

■ Overseas Subsidiaries

- **LS ELECTRIC Japan Co., Ltd. (Tokyo, Japan)**
  Tel: 81-3-6268-8241 E-Mail: jschuna@lselectric.biz
- **LS ELECTRIC (Dalian) Co., Ltd. (Dalian, China)**
  Tel: 86-411-8730-6495 E-Mail: jiheo@lselectric.com.cn
- **LS ELECTRIC (Wuxi) Co., Ltd. (Wuxi, China)**
  Tel: 86-510-6851-6666 E-Mail: sblee@lselectric.co.kr
- **LS ELECTRIC Shanghai Office (China)**
  Tel: 86-21-5237-9977 E-Mail: tsjun@lselectric.com.cn
- **LS ELECTRIC Vietnam Co., Ltd.**
  Tel: 84-93-631-4099 E-Mail: jhchoi4@lselectric.biz (Hanoi)
  Tel: 84-28-3823-7890 E-Mail: sjbaik@lselectric.biz (Hochiminh)
- **LS ELECTRIC Middle East FZE (Dubai, U.A.E.)**
  Tel: 971-4-886-5360 E-Mail: salesme@lselectric.biz
- **LS ELECTRIC Europe B.V. (Hoofddorf, Netherlands)**
  Tel: 31-20-654-1424 E-Mail: europartner@lselectric.biz
- **LS ELECTRIC America Inc. (Chicago, USA)**
  Tel: 1-800-891-2941 E-Mail: sales.us@lselectricamerica.com

LS ELECTRIC
Technical Question or After-sales Service
Customer Center – Quick Responsive Service, Excellent technical support
TEL. 82-1644-5481 | www.lselectric.co.kr

2020.05