Panasonic

Motion Controller

GM1 Series

Reference Manual

Instruction Edition

WUME-GM1PGR-08

2024.4

(MEMO)

Introduction

Thank you for purchasing a Panasonic product. Before you use the product, please carefully read through the installation instructions and the manuals, and understand them in detail to use the product properly.

Types of Manual

- There are different types of manuals for the GM1 series. Refer to the appropriate manual according to your need.
- These manuals can be downloaded from our website:https://industrial.panasonic.com/ac/e/motor/motion-controller/mc/gm1/index.jsp.

Manuals for GM1 series

Manual name	Manual code	Description
GM1 Controller RTEX User's Manual (Setup Edition)	WUME-GM1RTXSU	Explains wiring between the GM1 and its peripheral devices, installation method, and operation check method.
GM1 Controller EtherCAT User's Manual (Setup Edition)	WUME-GM1ETCSU	
GM1 Controller RTEX User's Manual (Operation Edition)	WUME-GM1RTXOP	Explains how to use GM Programmer and PANATERM Lite for GM, set up each function, create projects, and perform other operations.
GM1 Controller EtherCAT User's Manual (Operation Edition)	WUME-GM1ETCOP	
GM1 Series Reference Manual (Hardware Edition)	WUME-GM1H	Explains the functions and performance of each GM1 unit.
GM1 Series Reference Manual (Instruction Edition)	WUME-GM1PGR	Explains the specifications of each instruction that can be used with the GM1 Series.
GM1 Series Reference Manual (Analog I/O Unit)	WUME-GM1AIO	Explains the functions and performance of the GM1 Analog Expansion Unit.
GM1 Series Reference Manual (Pulse Output Unit)	WUME-GM1PG	Explains the functions and performance of the GM1 Pulse Output Unit.

Copyright / Trademarks

(MEMO)

Table of Contents

(MEMO)

1 List of Instructions

1.1 List of Ladder Instructions

The following table lists contact and coil ladder instructions that can be used in ladder diagram programs for GM Programmer.

Name	Code	Description	Simulation (●: Supported, -: Not supported)	Page
NO contact		This instruction outputs a BOOL-type input from the left to the right. If the variable of the contact is TRUE, then the input value from the left is output. If the variable of the contact is FALSE, then FALSE is output.	●	"P.2-2"
NC contact		This instruction outputs the negated value of the BOOL-type input from the left to the right. If the variable of the contact is TRUE, then FALSE is output. If the variable of the contact is FALSE, then the input value from the left is output.	●	"P.2-3"
Rising edge detection		When a rising edge is detected in the BOOL-type input from the left, TRUE is output for one cycle only.	●	"P.2-4"
Falling edge detection		When a falling edge is detected in the BOOL-type input from the left, TRUE is output for one cycle only	●	"P.2-5"
Parallel NO contact	-	NO contacts can be wired in parallel. The contacts wired in parallel are treated as OR logic. If the output of one or more contacts is TRUE, TRUE is output.	●	"P.2-6"
Parallel NC contact	-	NC contacts can be wired in parallel. The contacts wired in parallel are treated as OR logic. If the output of one or more contacts is TRUE, TRUE is output.	●	"P.2-7"
Coil		A BOOL-type input from the left can be saved. If the input is TRUE, then TRUE is saved. If the input is FALSE, then FALSE is saved.	●	"P.2-8"
Negated coil		The negated value of the BOOL-type input from the left can be saved. If the input is TRUE, then FALSE is saved. If the input is FALSE, then TRUE is saved.	●	"P.2-9"
Set coil		If the BOOL-type input from the left becomes TRUE, TRUE is saved. It can be used together with the reset coil.	●	"P.2-10"
Reset coil		If the BOOL-type input from the left becomes TRUE, FALSE is saved. It can be used together with the set coil.	●	"P.2-11"
Execute Box		ST language programming is possible. If "Enter ST code here ..." is clicked, an input field using a multi-line ST will open.	●	"P.2-12"

1.2 List of Function Instructions

This section provides lists of the functions used by the GM Programmer. These functions can be used without declaring them with variables.

■ Basic instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MOVE	Substitution	Substitutes the input argument values with the output argument.	●	●	●	"P.3-4"
SIZEOF	Get the size	Outputs the size (in units of byte) of the input argument.	●	●	●	"P.3-5"
ADR	Get the address	Outputs the address of the input argument.	●	●	●	"P.3-6"

■ Arithmetic operation instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
ADD	Addition	Adds the input arguments.	●	●	●	"P.3-7"
SUB	Subtraction	Subtracts the input arguments.	●	●	●	"P.3-9"
MUL	Multiplication	Multiplies the input arguments.	●	●	●	"P.3-10"
DIV	Division	Divides the input arguments.	●	●	●	"P.3-11"
MOD	Mod	Outputs the remainder of the input argument.	●	●	●	"P.3-12"

■ Boolean operation instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
AND	Logical AND	Outputs the logical AND of the input arguments.	●	●	●	"P.3-13"
OR	Logical OR	Outputs the logical OR of the input arguments.	●	●	●	"P.3-14"
XOR	Exclusive OR	Outputs the Exclusive OR of the input arguments.	●	●	●	"P.3-16"
NOT	Negation	Outputs the negation of the input argument.	●	●	●	"P.3-15"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
AND_THEN	Logical AND	Outputs the logical AND of the input arguments.	●	●	●	"P.3-17"
OR_ELSE	Logical OR	Outputs the logical OR of the input arguments.	●	●	●	"P.3-18"

■ Comparison operation instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
EQ	“Equal” comparison	Compares the two input arguments and, if they are equal to each other, outputs TRUE.	●	●	●	"P.3-19"
NE	“Not Equal” comparison	Compares the two input arguments and, if they are not equal to each other, outputs TRUE.	●	●	●	"P.3-20"
LT	“Less Than” comparison	Compares the two input arguments and, if the first argument is less than the second argument, outputs TRUE.	●	●	●	"P.3-21"
LE	“Less Than or Equal” comparison	Compares the two input arguments and, if the first argument is less than the second argument or equal, outputs TRUE.	●	●	●	"P.3-22"
GT	“Greater Than” comparison	Compares the two input arguments and, if the first argument is greater than the second argument, outputs TRUE.	●	●	●	"P.3-23"
GE	“Greater Than Or Equal” comparison	Compares the two input arguments and, if the first argument is greater than the second argument or equal, outputs TRUE.	●	●	●	"P.3-24"

■ Bit shift instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SHL	Shift left	Shifts the input argument to the left by the specified number of bits. Inserts “0” from the least significant	●	●	●	"P.3-25"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
		bit to the specified bit and outputs the data.				
SHR	Shift right	Shifts the input argument to the right by the specified number of bits. Inserts “0” from the most significant bit to the specified bit and outputs the data.	●	●	●	"P.3-26"
ROL	Rotate left	Shifts the input argument to the left by the specified number of bits. Inserts the value in excess from the most significant bit into the data starting from the least significant bit and outputs the data.	●	●	●	"P.3-27"
ROR	Rotate right	Shifts the input argument to the right by the specified number of bits. Inserts the value in excess from the least significant bit into the data starting from the most significant bit and outputs the data.	●	●	●	"P.3-28"

■ Numerical operation instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
ABS	Absolute value	Outputs the absolute value.	●	●	●	"P.3-29"
SQRT	Square root	Outputs the the square root (√) of a number.	●	●	●	"P.3-30"
LN	Natural logarithm	Outputs the natural logarithm ($\log_e X$) of a number.	●	●	●	"P.3-31"
LOG	Common logarithm	Outputs the common logarithm ($\log_{10} X$) of a number.	●	●	●	"P.3-32"
EXP	Natural exponent	Outputs the natural exponent (e^X) of a number.	●	●	●	"P.3-33"
EXPT	Exponentiation	Outputs the exponentiation of a number (a^n).	●	●	●	"P.3-34"
SIN	Trigonometric function (sine)	Outputs the result of the sine function calculation.	●	●	●	"P.3-35"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
COS	Trigonometric function (cosine)	Outputs the result of the cosine function calculation.	●	●	●	"P.3-36"
TAN	Trigonometric function (tangent)	Outputs the result of the tangent function calculation.	●	●	●	"P.3-37"
ASIN	Trigonometric function (arc sine)	Outputs the result of the arc sine function calculation.	●	●	●	"P.3-38"
ACOS	Trigonometric function (arc cosine)	Outputs the result of the arc cosine function calculation.	●	●	●	"P.3-39"
ATAN	Trigonometric function (arc tangent)	Outputs the result of the arc tangent function calculation.	●	●	●	"P.3-40"
SMC_PI	trigonometric constant	The conversion constants Pi, degree, and Radian are available.	●	●		"P.3-40"

■ Data type conversion instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
<Type 1>_TO_<Type 2>	Data type conversion	Converts type 1 input argument to type 2.	●	●	●	"P.3-41"
TRUNC	Data type conversion	Changes the real number to the DINT-type data.	●	●	●	"P.3-48"
TRUNC_INT	Data type conversion	Changes the real number to the INT-type data.	●	●	●	"P.3-49"
BCD_TO_**	Data type conversion	Converts BCD data to binary data (BYTE type / INT type / WORD type / DWORD type).	●	●	●	"P.3-50"
**_TO_BCD	Data type conversion	Converts binary data (BYTE type / INT type / WORD type / DWORD type) to BCD data.	●	●	●	"P.3-53"
GRAY_TO_**	Data type conversion	Converts a Gray code to binary data (BYTE type / WORD type / DWORD type).	●	●	●	"P.3-55"
**_TO_GRAY	Data type conversion	Converts binary data (BYTE type / WORD type / DWORD type) to a Gray code.	●	●	●	"P.3-57"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
BYTE_TO_HEXinASCII	Data type conversion	Converts a one-byte hexadecimal binary-coded value to a one-word ASCII code.	●	●	●	"P.3-59"
HEXinASCII_TO_BYTE	Data type conversion	Converts a one-word ASCII code to a one-byte hexadecimal binary-coded value.	●	●	●	"P.3-61"
MEM.Decode	Data type conversion	Converts data in units of byte to data in units of DWORD.	●	●	●	"P.3-63"
MEM.Encode	Data type conversion	Converts data in units of DWORD to data in units of byte.	●	●	●	"P.3-64"
MEM.PackArrayOfBoolToArrayOfByte	Data type conversion	Packs a BOOL type array into an array in bytes and copies a specified bit size data.	●	●	●	"P.3-66"
MEM.PackBitsTo**	Data type conversion	Packs BOOL type data and converts it to a BYTE, a WORD, or a DWORD.	●	●	●	"P.3-68"
MEM.PackBytesTo**	Data type conversion	Packs BYTE type data and converts it to one-word or one-dword data.	●	●	●	"P.3-73"
MEM.PackWordsToDword	Data type conversion	Packs WORD type data and converts it to a DWORD.	●	●	●	"P.3-75"
MEM.UnpackArrayOfByte	Data type conversion	Unpacks a BYTE type array to data in bits and copies the data to a specified BOOL array.	●	●	●	"P.3-76"

■ Bit operation instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
EXTRACT	Bit extraction	Outputs a BOOL status at a specified bit of a DWORD.	●	●	●	"P.3-78"
PUTBIT	Bit change	Changes the status of a specified bit of a DWORD.	●	●	●	"P.3-79"
SWITCHBIT	Bit inversion	Inverts the status of a specified bit of a DWORD.	●	●	●	"P.3-80"
MEMUtils.BitCpy	Bit copying	Copies a specified size of bit data.	●	●	●	"P.3-81"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MEM.ReverseBitsIn**	Bit order change	Reverses the order of the bits of BYTE-, WORD-, or DWORD-type data and outputs the data of the bits in reverse order.	●	●	●	"P. 3-83"

■ Data manipulation instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SEL	Binary selector	Outputs “IN0” when the input argument G is FALSE and “IN1” when the input argument G is TRUE.	●	●	●	"P.3-85"
MUX	Multiplexer	Outputs the input argument value depending on the input argument K (0,1,2,…).	●	●	●	"P.3-86"
LIMIT	Limiter	Limits the value of the input argument IN between the input arguments MN and MX and outputs the data.	●	●	●	"P.3-87"
MAX	Maximum value	Outputs the maximum value of the input argument.	●	●	●	"P.3-88"
MIN	Minimum value	Outputs the minimum value of the input argument.	●	●	●	"P.3-89"
MEMUtils.Swap	Byte swapping	Swaps specified bytes (2, 4, or 8 bytes) in order.	●	●	●	"P. 3-90"
MEM.Compare	Memory comparison	Compares two specified memory block data pieces to determine whether they match	●	●	●	"P. 3-91"
MEM.FindBlock	Memory block search	Searches memory block data for specified memory block data.	●	●	●	"P. 3-92"
MEM.FindByte	Find byte data	Searches specified memory block data for specified one-byte data.	●	●	●	"P. 3-94"
MEM.MemFill	Memory fill	Fills a specified size in data memory with a specified data value.	●	●	●	"P. 3-96"
MEM.MemMove	Memory copying	Copies a specified size in data memory onto copy destination data memory.	●	●	●	"P. 3-97"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MEM.High**	High byte/High WORD extraction	Outputs high byte / high WORD of an input value.	●	●	●	"P. 3-119"
MEM.Low**	Low byte/Low WORD extraction	Outputs low byte / low WORD of an input value.	●	●	●	"P. 3-100"
MEM.Reverse BYTEsIn**	Byte order change	Reverses the order of the bytes of WORD-, or DWORD-type data and outputs the data of the bytes in reverse order.	●	●	●	"P. 3-101"
MEM.Reverse WORDsInDWORD	WORD order change	Reverses the order of the bytes of DWORD-type data and outputs the data of the WORD in reverse order.	●	●	●	"P. 3-103"

■ Character string instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
LEN/WLEN	Length of a character string	Outputs the length of a character string.	●	●	●	"P.3-104"
LEFT/WLEFT	Extracting characters from the left end	Extracts a character string consisting of the specified number of characters from the left of the character string.	●	●	●	"P.3-105"
RIGHT/ WRIGHT	Extracting characters from the right end	Extracts a character string consisting of the specified number of characters from the right of the character string.	●	●	●	"P.3-106"
MID/WMID	Extracting characters from the specified position	Extracts a character string consisting of the specified number of characters from the specified position of the character string.	●	●	●	"P.3-107"
CONCAT/ WCONCAT	Concatenating character strings	Concatenates two character strings.	●	●	●	"P.3-108"
INSERT/ WINSERT	Inserting a character string	Inserts another character string into the specified position of one character string.	●	●	●	"P.3-109"
DELETE/ WDELETE	Deleting a character string	Deletes a character string consisting of the specified number of characters from	●	●	●	"P.3-111"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
		the specified position of the character string.				
REPLACE/ WREPLACE	Replacing a character string	Replaces a character string, consisting of the specified number of characters from the specified position of the character string, with another character string.	●	●	●	"P.3-112"
FIND/WFIND	Search for a character string	Searches for a specified character string in the character strings and outputs the position.	●	●	●	"P.3-114"
ConvertUTF16 toUTF8	Character code conversion	Converts a UTF-16 character string into a UTF-8 character string.	●	●	●	"P. 3-115"
ConvertUTF8t oUTF16	Character code conversion	Converts a UTF-8 character string into a UTF-16 character string.	●	●	●	"P. 3-117"

■ SD memory card slot instruction

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SYS_GetSDC overState	Get SD card cover open / close state	Gets an open / close state of the card cover for the SD memory card slot.	●	●	-	"P.3-119"
SYS_GetSDAc cessRdy	Get SD card access ready state	Gets the state whether an access to the SD memory card is allowed.	●	●	-	"P.3-119"

■ CRC operation instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MEM.CRC16_ standard	CRC16	Calculates CRC16 checksum.	●	●	●	"P. 3-120"
MEM.CRC32	CRC32	Calculates CRC32 checksum.	●	●	●	"P. 3-122"

■ System time instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, —: Not supported)			
SysTimeGetMs	Get system time in units of milliseconds	Gets a length of time that has been elapsed since the start of the controller in units of milliseconds.	●	●	●	"P. 3-123"
SysTimeGetUs	Get system time in units of microseconds	Gets a length of time that has been elapsed since the start of the controller in units of microseconds.	●	●	●	"P. 3-123"
SysTimeGetNs	Get system time in units of nanoseconds	Gets a length of time that has been elapsed since the start of the controller in units of nanoseconds.	●	●	●	"P. 3-124"

1.3 List of Function Block Instructions

This section provides lists of the function blocks used by the GM Programmer. These function blocks can be used with declaring the instances with variables.

1.3.1 Basic Instructions

■ Timer instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
TON	Timer ON	Starts the timer when the input argument changes from FALSE to TRUE and, after an elapse of the specified time, the output argument outputs TRUE.	●	●	●	"P.4-2"
TOF	Timer OFF	Starts the timer when the input argument changes from TRUE to FALSE and, after an elapse of the specified time, the output argument outputs FALSE.	●	●	●	"P.4-3"
TP	Timer pulse	Starts the timer when the input argument changes from FALSE to TRUE until the specified time elapses. Outputs TRUE to the output argument while the timer keeps counting.	●	●	●	"P.4-4"
RTC	Realtime clock	Starts counting time from the specified date and time when the input argument changes from FALSE to TRUE. Outputs TRUE to the output argument while the clock keeps counting time.	●	●	●	"P.4-6"

■ Counter instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
CTU	Up counter	Starts incrementing the counter value at the rising edge of the input argument CU and, after counting the specified number of count values, outputs TRUE.	●	●	●	"P.4-7"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
CTD	Down counter	Starts decrementing from the specified number of count value at the rising edge of the input argument CD. Outputs TRUE when it reaches 0.	●	●	●	"P.4-8"
CTUD	Up-down counter	Starts incrementing the counter value at the rising edge of the input argument CU and, after counting the specified number of count values, outputs TRUE. Starts decrementing the counter value at the rising edge of the input argument CD and, when it reaches 0, outputs TRUE.	●	●	●	"P.4-9"

■ Edge detection instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
R_TRIG	Rising edge detection	Outputs TRUE for one cycle only when detecting a rising edge.	●	●	●	"P.4-11"
F_TRIG	Falling edge detection	Outputs TRUE for one cycle only when detecting a falling edge.	●	●	●	"P.4-12"

■ Bistable circuit instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SR	Set-priority bistable circuit	If the input argument SET1 is TRUE, outputs TRUE. If the input argument RESET is TRUE, outputs FALSE. If both SET1 and RESET1 are TRUE, outputs TRUE	●	●	●	"P.4-13"
RS	Reset-priority bistable circuit	If the input argument SET1 is TRUE, outputs TRUE.	●	●	●	"P.4-14"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
		If the input argument RESET is TRUE, outputs FALSE. If both SET1 and RESET1 are TRUE, outputs FALSE.				

■ Data Type Conversion Instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MEM.Unpack**	Data type conversion	Unpacks BYTE- , WORD-, or DWORD-type data to data in bits and outputs the data.	●	●	●	"P. 4-16"

■ Data manipulation instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
LIN_TRAFO	linear conversion	Convert one range of numbers to another linearly.	●	●	●	"P. 4-22"
STATISTIC_REAL	Statistical data	Acquire the maximum, minimum, and average values of the input data (REAL type).	●	●	●	"P. 4-23"
LIMITALARM	Monitoring of input values	Monitor whether the input value is between LOW (lower limit) and HIGH (upper limit)	●	●	●	"P. 4-24"

■ Other instructions

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
BLINK	output of blinking signal	Switch the output argument OUT to TRUE or FALSE according to the setting time.	●	●	●	"P. 4-25"

1.3.2 Motion Control Function Blocks (Single Axis Control)

■ Servo ON

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MC_Power	Servo ON	Sets the axis to the servo ON state to be ready for operation.	●	●	●	"P.5-3"

■ Home return

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
PMC_Home	Home return	Performs home return operation on the axis. Uses the home return function of the amplifier.	●	-	-	"P.5-5"
MC_Home	Home return	Performs home return operation on the axis. Uses the home return function of the amplifier.	-	●	●	"P.5-8"

■ Control switch

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SMC_SetControllerMode	Control mode setup	Sets up the control mode for controlling the position, velocity, and torque.	●	●	-	"P.5-9"

■ Stop

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MC_Stop	Forced stop	Causes the axis to make a deceleration stop. After stopping, the axis remains stopped while Execute is TRUE.	●	●	●	"P.5-11"
MC_Halt	Stop	Causes the axis to make a deceleration stop. After the axis is stopped or while the axis is being decelerated, other motion	●	●	●	"P.5-13"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
		instructions can be executed.				

■ JOG / Inching

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MC_Jog	Jogging	Causes the axis to keep traveling in a forward or reverse direction at a constant velocity while the input is TRUE.	●	●	●	"P.5-17"
SMC_Inch	Inching	Causes the axis to travel in a forward or reverse direction for a specified relative distance when the input turns TRUE.	●	●	●	"P.5-19"

■ Position control

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MC_MoveAbsolute	Absolute value positioning	Causes the axis to travel to a position specified as an absolute position.	●	●	●	"P.5-23"
MC_MoveRelative	Relative value positioning	Causes the axis to travel to a position specified as a relative position.	●	●	●	"P.5-27"
MC_MoveAdditive	Change target position	Adds a relative distance to the target position of the immediately preceding instruction.	●	●	●	"P.5-31"
MC_MoveSuperImposed	Superimposed positioning	Adds a relative distance, a velocity, an acceleration, and a deceleration to the operations of the immediately preceding instruction.	●	●	●	"P.5-34"
MC_PositionProfile	Position profile move	Causes the axis to operate according to the profile data that consists of a combination of position and time.	●	●	●	"P.5-38"
SMC_MoveContinuousAbsolute	Absolute value position velocity move	Executes absolute value positioning and, after the axis reaches the target position, causes the axis	●	●	●	"P.5-43"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
		to keep moving at a specified velocity.				
SMC_MoveContinuousRelative	Relative value position velocity move	Executes relative value positioning and, after the axis reaches the target position, causes the axis to keep moving at a specified velocity.	●	●	●	"P.5-47"

■ Velocity control

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MC_MoveVelocity	Velocity control	Specifies the velocity of the axis.	●	●	●	"P.5-54"
MC_VelocityProfile	Velocity profile move	Causes the axis to operate according to the profile data that consists of a combination of time and velocity.	●	●	●	"P.5-57"
MC_AccelerationProfile	Acceleration profile move	Causes the axis to operate according to the profile data that consists of a combination of time and acceleration.	●	●	●	"P.5-60"

■ Torque control

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
PMC_SetTorque	Torque control	Specifies by % the torque of the axis.	●	●	●	"P.5-65"
SMC_SetTorque	Torque control	Specifies by Nm the torque of the axis.	-	●	●	"P.5-67"

■ Direct commands

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SMC_FollowPosition	Target position command at every control interval	Commands the target position at every control interval.	●	●	●	"P.5-71"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SMC_FollowVelocity	Target velocity command at every control interval	Commands the target velocity at every control interval.	-	●	●	"P.5-73"

■ Axis structure

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
AXIS_REF_SM3 AXIS_REF_VIRTUAL_SM3 FREE_ENCODER_REF	Axis device control	Controls devices with real axis, virtual axis, or encoder axis.	●	●	●	"P.5-92"

1.3.3 Motion Control Function Blocks (Synchronous Control)

■ Cam operation

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MC_CAM_REF	Cam profile	Cam profile structure	●	●	●	"P.6-15"
MC_CamTableSelect	Cam profile selection	Specifies the cam profile for cam synchronous operation.	●	●	●	"P.6-24"
MC_CamIn	Start cam control	Starts cam synchronous operation.	●	●	●	"P.6-27"
MC_CamOut	Cancel cam operation	Cancels cam synchronous operation.	●	●	●	"P.6-33"
SMC_GetTappetValue	Get single tappet information	Outputs single tappet information defined in the cam profile.	●	●	●	"P.6-34"
SMC_CamRegister	Get all tappet information	Outputs all tappet information relative to any master axis according to the cam profile.	●	●	●	"P.6-36"
SMC_CAMBounds	Slave parameter calculation	Calculate the maximum/minimum parameter values that the slave is	●	●	●	"P.6-39"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
		reaching relative to master parameters.				
SMC_GetCamSlaveSetPosition	Slave position calculation	Calculates starting position, velocity, and acceleration values of the slave based on the set cam table.	●	●	●	"P.6-41"

■ Gear operation

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MC_GearIn	Start gear operation	Starts gear synchronous operation.	●	●	●	"P.6-2"
MC_GearInPos	Position specified gear operation	Starts gear synchronous operation from the specified absolute position.	●	●	●	"P.6-5"
MC_GearOut	Cancel gear operation	Cancels the gear synchronous operation.	●	●	●	"P.6-10"

■ Phase correction

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MC_Phasing	Master axis phase correction	Corrects the phase between the master and slave axes.	●	●	●	"P.6-60"

1.3.4 Motion Control Function Blocks (Interpolation Control)

■ Interpolation Control

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
PMC_Interpolator2D	2-axis Interpolation Control	Specify the CNC pattern to perform 2-axis interpolation control.	●	●	●	"P.7-2"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
PMC_ Interpolator3D	3-axis Interpolation Control	Specify the CNC pattern to perform 3-axis interpolation control.	●	●	●	"P.7-4"
PMC_NCDecoder	CNC Table Conversion	Convert the CNC table to executable format.	●	●	●	"P.7-6"

1.3.5 Motion Control Function Blocks (CNC Control)

■ CNC data decoding

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SMC_ NCDecoder	CNC program conversion	Converts the CNC program to executable format.	●	●	●	"P.8-7"
SMC_ReadNCFile2	Read CNC file	Reads a CNC file from an SD card.	●	●	●	"P.8-11"
SMC_NCInterpreter	CNC file conversion	Converts CNC data read through SMC_ReadNCFile2 to executable format.	●	●	●	"P.8-16"
SMC_GEOINFO	CNC executable format data	This is a structure of CNC executable format data.	●	●	●	"P.8-19"

■ Pre-processing after decoding

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SMC_CheckVelocities	Check angle between paths	Check the angle between the paths and switch between P point control and C point control.	●	●	●	"P.8-22"
SMC_SmoothPath	Path smoothing	Smooths the path of the specified CNC program.	●	●	●	"P.8-23"
SMC_RoundPath	Arc correction between paths	Corrects between paths in the specified CNC program with an arc.	●	●	●	"P.8-26"
SMC_ToolRadiusCorr	Tool radius correction for path	Applies tool radius correction to the path of the specified CNC program.	●	●	●	"P.8-29"

■ Control calculation

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SMC_Interpolator	CNC control operation	Calculates discrete data for the control interval from the continuous path.	●	●	●	"P.8-31"
SMC_GetMParameters	Get M-code parameters	Parameters can be received from the CNC processing unit that is paused by the M-code.	●	●	●	"P.8-35"
SMC_PreAcknowledgeMFunction	Deactivate M-code	The operation can continue without pausing by ignoring the M-code description.	●	●	●	"P.8-36"

■ Control command & kinematics conversion

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SMC_ControlAxisByPos	Control command	Command the CNC position information to the motor.	●	●	●	"P.8-37"
SMC_ToolLengthCorr	Tool length correction	Offsets the coordinates by the tool length.	●	●	●	"P.8-38"
SMC_TRAFO_Polar	Conversion to polar coordinates	Converts two-dimensional coordinates to polar coordinates.	●	●	●	"P.8-41"
SMC_TRAFOF_Polar	Conversion to polar coordinates	Converts polar coordinates to two-dimensional coordinates.	●	●	●	"P.8-42"
SMC_TRAFO_Bipod_Arm	Bipod robot conversion	Converts Bipod robot hand XY coordinates to angle information of each axis motor.	●	●	●	"P.8-44"
SMC_TRAFO_Gantry2	XY linear type robot conversion	Converts two-dimensional (X, Y) coordinates to position information of each axis motor.	●	●	●	"P.8-46"
SMC_TRAFOF_Gantry2	XY linear type robot conversion	Converts position information of each axis motor to two-dimensional (X, Y) coordinates.	●	●	●	"P.8-47"
SMC_TRAFO_Gantry3	XYZ linear type robot conversion	Converts three-dimensional (X, Y, Z) coordinates to position information of each axis motor.	●	●	●	"P.8-49"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SMC_TRAFOF_Gantry3	XYZ linear type robot conversion	Converts position information of each axis motor to three-dimensional (X, Y, Z) coordinates.	●	●	●	"P.8-50"
SMC_TRAFO_GantryCutter2	Conversion of XY linear type robot with single rotational axis	Converts two-dimensional (X, Y) gantry coordinates with a rotational axis cutter to position information of each axis motor.	●	●	●	"P.8-52"
SMC_TRAFO_GantryCutter3	Conversion of XYZ linear type robot with single rotational axis	Converts three-dimensional (X, Y, Z) gantry coordinates with a rotational axis cutter to position information of each axis motor.	●	●	●	"P.8-53"

■ CNC program

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, —: Not supported)			
G00, G01	Linear interpolation	Moves from the current position to target coordinates by linear interpolation.	●	●	●	"P.8-59"
G02, G03	Circular interpolation	Moves from the current position to target coordinates by circular interpolation.	●	●	●	"P.8-62"
G04	Dwell time	Sets a time to wait until next processing is executed.	●	●	●	"P.8-69"
G15, G16, G17, G18, G19	Plane specification	Specifies a plane in which interpolation motion is performed.	●	●	●	"P.8-70"
G20, G36, G37	Jump and loop process	Specifies variables that can be used in jump condition settings and loop conditions.	●	●	●	"P.8-73"
G40, G41, G42	Tool radius correction for path	Sets the start and end points of tool radius correction.	●	●	●	"P.8-89"
G43	Tool length correction	Sets the tool length.	●	●	●	"P.8-73"
G50, G51, G52	Path smoothing	Sets smoothing start and end points.	●	●	●	"P.8-98"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, —: Not supported)			
G53, G54, G55, G56	Coordinate conversion	Converts the reference coordinate system under CNC from the machine coordinate system (MCS) to the decoder coordinate system (DCS).	●	●	●	"P.8-102"
G75	Timing synchronization	Synchronizes the timing of interpolation operations in SMC_Interolator.	●	●	●	"P.8-113"
G90, G91	Specification of coordinates	Sets any of absolute coordinate specification and relative coordinate specification.	●	●	●	"P.8-114"
G92	Start position specification	Sets the start position of a CNC program operation.	●	●	●	"P.8-117"
G98, G99	Circular arc coordinate specification	Circular arc coordinates can be specified as either absolute coordinates or relative coordinates.	●	●	●	"P.8-118"
M-code	M-code programming	When the program reaches a line at which the M-code is executed, SMC_Interpolator can be paused to execute a desired process.	●	●	●	"P.8-121"
H-Switch	IO output switching by H-switch	You can turn ON or OFF the IO output during the execution of an interpolation operation with specified timing.	●	●	●	"P.8-125"
CNC program file	Programming by CNC program file	With the method of reading a CNC program from an SD card, you can program code using any of subprograms, variables, operators, and functions.	●	●	●	"P.8-129"

1.3.6 Motion Control Function Blocks (Motion Communication Control)

■ RTEX/EtherCAT Common

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, —: Not supported)			
SetCommunicationState	Change device state setting	Specifies a device state.	-	●	-	"P.9-3"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, —: Not supported)			
CheckSupportedCommunicationState	Check supported state	Checks if the device supports a transition to a requested setting.	-	●	-	"P.9-4"
CheckCurrentSupportedCommunicationState	Check state change availability	Checks if the device in the current state provides a transition to a requested setting.	-	●	-	"P.9-5"

■ RTEX

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
RTEX_ClearAmpAlarm	Clear amplifier alarm	Clears the amplifier's alarm.	●	-	-	"P.9-6"
RTEX_ReadAmpAlarm	Read amplifier alarm	Reads the amplifier's alarm.	●	-	-	"P.9-9"
RTEX_ReadAmpState	Amplifier alarm status	Reads the amplifier's alarm status.	●	-	-	"P.9-10"
RTEX_ReadAmpData	Amplifier monitor	Reads the amplifier's monitor data.	●	-	-	"P.9-11"
RTEX_ReadAmpParameter	Read amplifier parameter	Reads the amplifier's parameters.	●	-	-	"P.9-12"
RTEX_WriteAmpParameter	Write amplifier parameter	Writes the amplifier's parameters.	●	-	-	"P.9-13"
RTEX_WriteAmpEEPROM	Write amplifier EEPROM	Writes parameters of the servo amplifier to EEPROM.	●	-	-	"P.9-14"
RTEX_Reset	Reset RTEX	Resets the entire RTEX network.	●	-	-	"P.9-15"
RTEX_ClearAmpMultiTurnData	Clear the multi-turn data	Clears the multi-turn data of the amplifier.	●	-	-	"P.9-16"
RTEX_ClearAmpPositionalDeviation	Clear amplifier deviation counter	Clears the deviation counter of the amplifier.	●	-	-	"P.9-17"
RTEX_GetTrackingCommandError	Error	Measures the number of sent RTEX commands and the number of lost RTEX commands.	●	-	-	"P.9-19"
RTEX_ReadPot	Read NOT of amplifier	Reads the amplifier's NOT status.	●	-	-	"P.9-20"
RTEX_ReadNot	Read POT of amplifier	Reads the amplifier's POT status.	●	-	-	"P.9-20"

■ EtherCAT

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
ETC_CO_SdoRead	Read slave parameter	Reads EtherCAT slave parameters. Unlike ETC_CO_SdoRead4, this FB supports parameters longer than 4 bytes.	-	●	-	"P.9-21"
ETC_CO_SdoRead4	Read four bytes of slave parameter	Reads EtherCAT slave parameters. Unlike ETC_CO_SdoRead, this FB supports only parameters with 4 bytes or less.	-	●	-	"P.9-22"
ETC_CO_SdoReadDWord	Read double words of slave parameter	Just like ETC_CO_SdoRead4, this FB reads the EtherCAT slave parameters. The read data is stored in DWORD (dwData), not in an array.	-	●	-	"P.9-23"
ETC_CO_SdoRead_Access	Read slave parameter index	Just like ETC_CO_SdoRead, this FB reads the EtherCAT slave parameters. By setting the xCompleteAccess input to TRUE and the bySubIndex input to 0, you can read complete indexes including all entries.	-	●	-	"P.9-24"
ETC_CO_SdoRead_Channel	Read priority specification of slave parameter	Reads all EtherCAT slave parameters.	-	●	-	"P.9-26"
ETC_CO_SdoWrite	Write slave parameter	Writes EtherCAT slave parameters. Unlike ETC_CO_SdoWrite4, this FB supports parameters longer than 4 bytes.	-	●	-	"P.9-27"
ETC_CO_SdoWrite4	Write four bytes of slave parameter	Writes EtherCAT slave parameters. Unlike ETC_CO_SdoWrite, this FB supports only parameters with 4 bytes or less.	-	●	-	"P.9-29"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
ETC_CO_SdoWriteDWord	Write double words of slave parameter	Just like ETC_CO_SdoWrite4, this FB writes the EtherCAT slave parameters. The write data is transferred in DWORD (dwData), not in an array.	-	●	-	"P.9-30"
ETC_CO_SdoWrite_Access	Write slave parameter index	Just like ETC_CO_SdoWrite, this FB writes the EtherCAT slave parameters. By setting the xCompleteAccess input to TRUE and the bySubIndex input to 0, you can write complete indexes including all entries. By using the byChannelPriority (BYTE) input, you can specify the channel and priority using a CoE mailbox message.	-	●	-	"P.9-31"
ReadIdentification	Read slave identification data	Reads identification data from EtherCAT slaves.	-	●	-	"P.9-33"
ReadMemory	Read slave memory	Reads the EtherCAT slave memory.	-	●	-	"P.9-34"
ReadNbrSlaves	Read the number of connected slaves	Reads the number of slaves currently connected.	-	●	-	"P.9-35"
WriteMemory	Write slave memory	Writes the EtherCAT slave memory.	-	●	-	"P.9-36"
PETC_ClearAmpPositionalDeviation	Clear amplifier deviation counter	Clears the deviation counter of the amplifier.	-	●	-	"P.9-37"
IoDrvEtherCAT	EtherCAT master control	Controls EtherCAT master communication.	-	●	-	"P.9-39"
IoDrvEtherCAT.GetStatistics	Get EtherCAT frame statistics information	Reads EtherCAT master statistics data.	-	●	-	"P.9-40"
IoDrvEtherCAT.ClearStatistics	Clear EtherCAT frame statistics information	Clears EtherCAT master statistics data.	-	●	-	"P.9-41"
ETCSlave	EtherCAT slave control	Controls EtherCAT slave communication.	-	●	-	"P.9-41"

1.3.7 Motion Control Function Blocks (Auxiliary Function)

■ Motion auxiliary function (Monitoring)

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MC_ReadActualPosition	Read actual position	Reads the actual position data of the axis.	●	●	●	"P.10-2"
MC_ReadActualVelocity	Read actual velocity	Reads the actual velocity of the axis.	●	●	●	"P.10-2"
PMC_ReadActualTorque	Read actual torque	Read the actual torque value of the axis.	●	●	-	"P.10-3"
MC_ReadActualTorque	Read actual torque	Reads the actual torque value of the axis.	-	●	-	"P.10-4"
MC_ReadAxisError	Read axis error	Gets general axis errors not related to function blocks.	-	●	●	"P.10-5"
MC_ReadStatus	Read status	Reads the status information of the axis.	●	●	●	"P.10-6"
SMC_InPosition	In-position judgment	Compares the actual position of the AMP with the command value and judges whether the position is within the specified range.	●	●	●	"P.10-8"
SMC_ReadFBError	Read oldest error	Reads the oldest function block error information.	●	●	●	"P.10-10"
SMC_ClearFBError	Clear oldest error	Clears the oldest FB error information.	●	●	●	"P.10-11"
SMC_CheckAxisCommunication	Check axis communication state	Checks the communication state of the axis.	●	●	●	"P.10-12"
SMC_CheckLimits	Check exceeding limits	Checks whether the velocity, acceleration, or deceleration is in excess of the dynamic limit set value of the device.	●	●	●	"P.10-13"
SMC_GetMaxSetAccDec	Measure maximum acceleration / deceleration	Measures the maximum value of the axis acceleration / deceleration command.	●	●	●	"P.10-14"
SMC_GetMaxSetVelocity	Measure maximum velocity	Measures the maximum value of the axis velocity command.	●	●	●	"P.10-15"
SMC_GetTrackingError	Measure tracking error	Measures the tracking error of the actual position for the axis command position.	●	●	●	"P.10-16"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SMC_MeasureDistance	Measures turnaround travel distance	Measures the travel distance.	●	●	●	"P.10-17"
SMC_ReadSetPosition	Read axis set position	Reads the set command position of the axis.	●	●	●	"P.10-18"

■ Motion auxiliary function (Change / reset)

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MC_Reset	Reset axis error	Resets the state transition error of the axis.	●	●	●	"P.10-19"
SMC3_ReinitDrive	Reinitialize axis	Restarts the servo amplifier / axis.	●	●	●	"P.10-20"
MC_SetPosition	Change actual position	Changes the actual command position of the axis.	●	●	●	"P.10-21"
SMC_ChangeDynamicLimits	Change axis settings	Change the dynamic limit of the axis.	●	●	●	"P.10-21"
SMC_ChangeGearingRatio	Change axis settings	Change the shaft type and gear ratio.	●	●	●	"P.10-23"
SMC_SetMovementType	Change axis settings	Change the type of the virtual axis.	●	●	●	"P.10-26"
SMC_SetRampType	Change axis settings	Change the speed ramp type of the axis.	●	●	●	"P.10-28"
SMC_SetSoftwareLimits	Change axis settings	Change the soft limit of the axis.	●	●	●	"P.10-29"

■ Motion auxiliary function (Other functions)

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
PMC_ReadLatchPosition	Monitor AMP latch position	Monitors the AMP latch position.	●	-	-	"P.10-31"
PMC_StopLatchPosition	Stop AMP latch monitoring	Stops the axis at the AMP latch position.	●	-	-	"P.10-33"
MC_TouchProbe	Enable AMP latch monitoring	Reads the axis position when a trigger signal occurs.	-	●	●	"P.10-36"
MC_AbortTrigger	Disable AMP latch monitoring	Aborts the trigger event (MC_TouchProbe).	-	●	●	"P.10-38"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
MC_DigitalCamSwitch	Enable digital cam switch	Performs ON / OFF control on the digital output according to the axis position.	●	●	●	"P.10-39"
SMC_BacklashCompensation	Compensate backlash	Compensates the backlash.	●	●	●	"P.10-43"

1.3.8 Function Blocks (Others)

■ COM port (General-purpose communication)

The following table lists the function blocks that are used to perform general-purpose communication with the COM port.

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
COM.Open	Open COM port	Opens the COM port.	●	●	-	"P.11-5"
COM.Close	Close COM port	Closes the COM port.	●	●	-	"P.11-8"
COM.Read	Read COM port	Reads data from the COM port.	●	●	-	"P.11-9"
COM.Write	Write COM port	Writes data to the COM port.	●	●	-	"P.11-10"
COM.ERROR	Error ID	This is an enumeration type error ID that is output when the COM port (general-purpose communication) function block is executed.	●	●	-	"P.11-11"

■ COM port (Modbus COM)

The following table lists the instructions that are used to perform ModbusRTU communication with the COM port.

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
IoDrvModbusComPort	ModbusComPort device	This is a function block that controls the Modbus_Master_COM_Port device.	●	●	-	"P.11-12"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
IoDrvModbus.ModbusChannel	Start sending Modbus command	Sends the command set in the Modbus Slave channel of the ModbusSlaveCOM_Port device.	●	●	-	"P.11-12"
IoDrvModbus.ModbusRequest	Modbus request	Processes the Modbus command specified by I/O without using the ModbusMasterComPort device.	●	●	-	"P.11-13"
IoDrvModbus.ModbusRequest2	Modbus request 2	Like the ModbusRequest, processes the Modbus command specified by I/O without using the ModbusMasterComPort device.	●	●	-	"P.11-15"
IoDrvModbus.ModbusSlaveComPort	ModbusSlave ComPort device	This is a function block that controls the Modbus_Slave_COM_Port device.	●	●	-	"P.11-16"
IoDrvModbus.MB_ErrorCodes	Error code	This is an enumeration type error code that is output when the function block for Modbus communication instruction that uses the COM port is executed.	●	●	-	"P.11-17"

■ LAN port (IoDrvEthernet)

The following table lists the library functions that are used for the network interface to perform communication with the LAN port.

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
IoDrvEthernet	Ethernet device	This is a function block that acquires the status of the LANPort device.	●	●	-	"P.11-18"
IoDrvEthernet.IPARRAY_TO_INADDR	From array type to union type	This is a function that converts an array type IP address to an INADDR (union type).	●	●	●	"P.11-18"
IoDrvEthernet.IPARRAY_TO_IPSTRING	From array type to character string type	This is a function that converts an array type IP address to a character string type.	●	●	●	"P.11-19"
IoDrvEthernet.IPARRAY_TO_UDINT	From array type to UDINT type	This is a function that converts an array type IP address to a UDINT type.	●	●	●	"P.11-19"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
IoDrvEthernet.IPSTRING_TO_UDINT	From character string type to UDINT type	This is a function that converts a character string type IP address to a UDINT type.	●	●	●	"P.11-20"
IoDrvEthernet.UDINT_TO_IPARRAY	From UDINT type to array type	This is a function that converts a UDINT type IP address to an array type.	●	●	●	"P.11-20"
IoDrvEthernet.UDINT_TO_IPSTRING	From UDINT type to character string type	This is a function that converts a UDINT type IP address to an array type.	●	●	●	"P.11-21"

■ LAN port (General-purpose communication)

The following table lists the library functions that are used to perform general-purpose communication with the LAN port using the TCP or UDP protocol.

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
NBS.TCP_Client	Connect to TCP client	Connects to the TCP/IP client.	●	●	-	"P.11-22"
NBS.TCP_Connection	Connect TCP	Establishes the connection of the client connecting to the connection port opened by TCP_Server.	●	●	-	"P.11-23"
NBS.TCP_Read	Receive TCP data	Acquires data received by the connection port that is established by TCP_Connection.	●	●	-	"P.11-24"
NBS.TCP_Server	Connect TCP server	Opens the specified port as a TCP/IP connection port.	●	●	-	"P.11-25"
NBS.TCP_Write	Send TCP data	Sends data to the connection port that is established by TCP_Connection.	●	●	-	"P.11-26"
NBS.UDP_Peer	Open UDP port	Opens the UDP/IP port.	●	●	-	"P.11-27"
NBS.UDP_Receive	Receive UDP data	Receives data to the connection handle acquired by UDP_Peer.	●	●	-	"P.11-28"
NBS.UDP_Send	Send UDP data	Sends data to the connection handle acquired by UDP_Peer.	●	●	-	"P.11-30"
NBS.ERROR	Error code	This is an enumeration type error code that is output when the function	●	●	-	"P.11-29"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
		block for communication instruction that uses the LAN port is executed.				

■ LAN port (Modbus TCP)

The following table lists the library functions that are used to perform ModbusTCP communication with the LAN port.

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
IoDrvModbusTCP	ModbusTCP device	This is a function block that controls the Modbus_TCP_Master device.	●	●	-	"P.11-43"
IoDrvModbusTCP.ModbusChannel	Start sending Modbus command	Sends the command set in the Modbus Slave channel of the ModbusTCP_Slave device.	●	●	-	"P.11-43"
IoDrvModbusTCP.ModbusRequest	Modbus request	Processes the Modbus command specified by I/O without using the Modbus_TCP_Slave device.	●	●	-	"P.11-44"
IoDrvModbusTCPSlave	ModbusTCPSlave device	This is a function block that controls the Modbus__TCP_Slave device.	●	●	-	"P.11-46"
IoDrvModbus.MB_ErrorCodes	Error code	This is an enumeration type error code that is output when the function block for Modbus communication instruction that uses the LAN port is executed.	●	●	-	"P.11-47"

■ LAN port (EtherNet/IP)

The following table lists instructions that are used to control EtherNet/IP scanner and adapter functions using the GM1 controller.

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
IoDrvEtherNetIP	EtherNet/IP scanner device	This is a function block that controls the EtherNet/IP scanner device.	●	●	-	"P.11-48"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
RemoteAdapter	Remote adapter device	This is a function block for the Remote adapter device linked to the EtherNet/IP scanner device.	●	●	-	"P.11-49"
IoDrvEtherNetIPAdapter	EtherNet/IP adapter device	This is a function block that controls the EtherNet/IP adapter device.	●	●	-	"P.11-51"
Module	EtherNet/IP module device	This is a function block that controls the EtherNet/IP module device.	●	●	-	"P.11-53"
Apply_Attributes	Apply_Attributes service	This is a function block that calls Apply_Attributes service of the CIP object instance.	●	●	-	"P.11-54"
Generic_Service	Execute generic service	This is a function block that executes generic services with the EtherNet/IP adapter.	●	●	-	"P.11-55"
Get_Attribute_Single	Inquire specific attributes of a specific instance	This is a function block that inquires specific attributes of a specific instance of the CIP object	●	●	-	"P.11-57"
Get_Attributes_All	Inquire all attributes of a specific instance	This is a function block that inquires all attributes of a specific instance of the CIP object	●	●	-	"P.11-58"
Set_Attribute_Single	Set specific attributes of a specific instance	This is a function block that sets specific attributes of a specific instance of the CIP object	●	●	-	"P.11-59"
Set_Attributes_All	Set all attributes of a specific instance	This is a function block that sets all attributes of a specific instance of the CIP object	●	●	-	"P.11-60"
NOP	NOP service	This is a function block that executes the NOP service of a specific instance of the CIP object	●	●	-	"P.11-61"
Reset	Reset service	This is a function block that executes the Reset service of a specific instance of the CIP object	●	●	-	"P.11-62"
Start	Start service	This is a function block that executes the Start service of a specific instance of the CIP object	●	●	-	"P.11-63"
Stop	Stop service	This is a function block that executes the Stop	●	●	-	"P.11-64"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
		service of a specific instance of the CIP object				
ENIP.ERROR	Message service instruction error code	-	●	●	-	"P.11-65"
ENIP.CIPClass	Service class code	-	●	●	-	"P.11-68"

■ MQTT

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, —: Not supported)			
MQTT.MQTTClient	MQTT client connection	Connects to an MQTT broker server.	●	●	●	"P.11-77"
MQTT.MQTTPublish	MQTT publish function	Sends a message to an MQTT broker server.	●	●	●	"P.11-83"
MQTT.MQTTSubscribe	MQTT subscribe function	Registers subscriptions on an MQTT broker server.	●	●	●	"P.11-86"

■ DNS

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, —: Not supported)			
DNS_GetIPAddress	Name Resolution	Sends the DNS server a query about the IP address of the specified host name.	●	●	-	"P.11-103"

■ SD card operation (File operation)

Files in the SD card inserted in the SD memory card slot can be operated.

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
FILE.Open	Open file	Opens a file or creates a new file.	●	●	-	"P.11-107"
FILE.Close	Close file	Closes a file.	●	●	-	"P.11-108"
FILE.Read	Read file	Reads data from the file opened by the Open instruction.	●	●	-	"P.11-109"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
FILE.Write	Write file	Writes data to the file opened by the Open instruction.	●	●	-	"P.11-110"
FILE.Flush	Flush file	Flushes buffer contents to the file opened by the Open instruction.	●	●	-	"P.11-111"
FILE.Copy	Copy file	Copies a file.	●	●	-	"P.11-112"
FILE.Rename	Rename file	Changes a file name.	●	●	-	"P.11-113"
FILE.Delete	Delete file	Deletes a file.	●	●	-	"P.11-114"
FILE.EOF	EOF of file	Determines whether the current offset of a file is EOF (End Of File) or not.	●	●	-	"P.11-115"
FILE.GetAttribute	Get file attribute	Gets file attributes (compressed, hidden, normal, read only).	●	●	-	"P.11-116"
FILE.GetPos	Get file offset	Gets the current offset of a file.	●	●	-	"P.11-117"
FILE.GetSize	Get file size	Gets the file size.	●	●	-	"P.11-118"
FILE.GetTime	Get file update time	Get the update time of a file	●	●	-	"P.11-119"
FILE.SetPos	Set file offset	Sets the offset of a file.	●	●	-	"P.11-120"

■ SD card operation (Directory operation)

Directories in the SD card inserted in the SD memory card slot can be operated.

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
FILE.DirCreate	Create directory	Creates a directory with a specified name.	●	●	-	"P.11-126"
FILE.DirOpen	Open directory	Opens a directory.	●	●	-	"P.11-127"
FILE.DirClose	Close directory	Closes a directory	●	●	-	"P.11-128"
FILE.DirCopy	Copy directory	Copies a directory.	●	●	-	"P.11-129"
FILE.DirRename	Rename directory	Renames a directory	●	●	-	"P.11-130"
FILE.DirRemove	Delete directory	Deletes a directory.	●	●	-	"P.11-131"
FILE.DirList	Directory list	Outputs a list of directories and files inside the directory.	●	●	-	"P.11-132"

■ SD card operation (CSV file operation)

CSV files in the SD card inserted in the SD memory card slot can be operated (reading, writing).

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, —: Not supported)			
CSV.CSVReaderInit	Specify target CSV file to be read	Specifies a CSV file from which data is read.	●	●	●	"P.11-134"
CSV.ReadAll	Read all file data by batch	Reads all data from a CSV file.	●	●	●	"P.11-136"
CSV.NextElement	Read one element	Reads one element from a CSV file.	●	●	●	"P.11-138"
CSV.NextLine	Read one line	Reads one line from a CSV file.	●	●	●	"P.11-139"
CSV.Init	Specify target CSV file to write.	Specifies a CSV file to which data is written.	●	●	●	"P.11-143"
CSV.Add'Type'	Add data to internal buffer	Adds input data to an internal buffer.	●	●	●	"P.11-145"
CSV.NewLine	Add line separator to internal buffer	Adds a line separator to an internal buffer.	●	●	●	"P.11-147"
CSV.WriteFile	Write, save data to CSV file	Writes data added to an internal buffer to a CSV file and saves the data.	●	●	●	"P.11-148"
CSV.NewFile	Change target to write to new CSV file	Changes the target to write to a new CSV file.	●	●	●	"P.11-149"
CSV.CSVWriter	CSVWriter	This FB is used in the input / output parameters of CSV.Init, Add'Type', NewLine, NewFile, and WriteFile.	●	●	●	"P.11-151"

■ Clock setting

The following table lists the function blocks that are used to set the clock of the GM1 Controller.

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SYS_GetTime	Get time	This is a function block that gets the current local time	●	●	-	"P.11-160"
SYS_SetTime	Set time	This is a function block that sets the current local time.	●	●	-	"P.11-161"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SYS_GetTimezone	Get time zone information	This is a function block that gets the time zone information.	●	●	-	"P.11-162"
SYS_SetTimezone	Set time zone information	This is a function block that sets the time zone information.	●	●	-	"P.11-163"
SYS_DateConcat	Convert from UINT type to DATE type	This is a function that converts a UINT type date to a DATE type.	●	●	-	"P.11-163"
SYS_DateSplit	Convert from DATE type to UINT type	This is a function that converts a DATE type date to a UINT type.	●	●	-	"P.11-164"
SYS_DTConcat	Convert from UINT type to DT type	This is a function that converts a UINT type date and time to a DT type.	●	●	-	"P.11-165"
SYS_DTSplit	Convert from UINT type to DT type	This is a function that converts a UINT type date and time to a DT type.	●	●	-	"P.11-166"
SYS_GetDayOfWeek	Get day of the week	This is a function that gets the day of the week from the DATE type date.	●	●	-	"P.11-167"
SYS_TODConcat	Convert from UINT type to TOD type	This is a function that converts a UINT type time with milliseconds to a TOD type.	●	●	-	"P.11-168"
SYS_TODSplit	Convert from UINT type to TOD type	This is a function that converts a TOD type time with milliseconds to a UINT type.	●	●	-	"P.11-169"
ERROR	Clock instruction error code	-	●	●	-	"P.11-170"
SNTP.SNTPGetUTCTime	Get SNTP time	This is a function block used to communicate with the SNTP server and get the current server time and Main Unit time.	●	●	●	"P.11-170"

■ System data

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SYS_GetSystemError	Get system error	Gets the information of a system error that has occurred in the GM1 Controller.	●	●	-	"P.11-174"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
SYS_ClearSystemError	Clear system error	Clears a system error in the GM1 Controller.	●	●	-	"P.11-174"

■ PID control

This is a function block related to PID control of the GM1 controller.

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
PD	PD control	Performs PD control.	●	●	-	"P.11-175"
PID	PID control	Performs PID control.	●	●	-	"P.11-176"
PID_FIXCYCLE	PID control (any cycle time)	Performs PID control. Cycle time can be manually set.	●	●	-	"P.11-177"

■ Recipe Function

It is a method list of the function block RecipeManCommands of the recipe function.

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
CreateRecipe	Creating a recipe	Create a new recipe.	●	●	●	"P.11-180"
DeleteRecipe	Delete recipe	Delete the recipe.	●	●	●	"P.11-183"
LoadFromAndWriteRecipe	Read recipe file	Reads the value from the recipe file and writes to the recipe and the current value.	●	●	●	"P.11-184"
ReadAndSaveRecipe	Save to recipe file	Save the current value in the recipe and recipe file.	●	●	●	"P.11-186"
prvCompareRecipe	Recipe comparison	Compare the recipe with the current value.	●	●	●	"P.11-187"
ReloadRecipes	Reload recipe file	Read the recipe (inside) from the recipe file in the SD card.	●	●	●	"P.11-189"
GetRecipeCount	Get the number of recipes	Gets the number of recipes that belong to the recipe definition.	●	●	●	"P.11-190"
GetRecipeNames	Get a list of recipe names	Gets a list of recipe names that belong to the recipe definition.	●	●	●	"P.11-191"
GetLastError	Get last error information	Gets the ReturnValues values for last processing.	●	●	●	"P.11-193"

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, -: Not supported)			
GetLastInfo	Get last info information	Gets the InfoValues values for last processing.	●	●	●	"P.11-195"
ResetLastError	Reset last error information	Resets the value of GetLastError.	●	●	●	"P.11-197"
ResetLastInfo	Clear last info information	Resets the value of GetLastInfo.	●	●	●	"P.11-198"

■ Enable/Disable devices

This is a function block related to device enable/disable switching.

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, —: Not supported)			
INode.Enable	Enable/Disable setting	Configure enable/disable setting on a device.	-	●	-	"P.11-200"
Reconfigure	Reconfigure devices	Reads the whole configuration of the specified device and its subdevices and reconfigures the devices.	-	●	-	"P.11-201"

■ Project management function

Name	Function	Overview	RTEX	EtherCAT	Simulation (●: Supported, -: Not supported)	Page
			(●: Supported, —: Not supported)			
SYS_PRJBackup	Backup project	Creates a backup file and a restoration configuration file on an SD memory card.	●	●	-	"P.11-206"
SYS_PRJRestore	Restore project	Restores a backup file from an SD memory card.	●	●	-	"P.11-208"
SYS_GetPRJRestoreResult	Project restoration results	Gets results of the execution of project restoration.	●	●	-	"P.11-211"

1.3.9 Function Blocks (For the GM1 Pulse Output Unit)

The following table lists the function blocks used to control the GM1 Pulse Output Unit.

Name	Function	Overview	RTEX	EtherCAT	Simulation	Page
			(●: Supported, —: Not supported)			
PG_Power	Servo ON or OFF	Performs servo ON/OFF control.	●	●	-	"P.12-4"
PG_Jog	Jogging	Causes the axis to keep traveling in a forward or backward direction.	●	●	-	"P.12-5"
PG_MoveAbsolute	Absolute value positioning	Causes the axis to travel to a position specified as an absolute position.	●	●	-	"P.12-6"
PG_MoveRelative	Relative value positioning	Causes the axis to travel to a position specified as a relative position.	●	●	-	"P.12-7"
PG_LatchPosition	Latch relative positioning	Causes the axis to travel to the relative position specified by an external signal input.	●	●	-	"P.12-9"
PG_Pulser	Pulser operation	Enables constant speed operation for the axes using an external pulse input.	●	●	-	"P.12-11"
PG_Stop	Forced stop and deceleration stop	Causes the axis to make a forced stop or deceleration stop	●	●	-	"P.12-13"
PG_Home	Home return	Causes the axis to make a home return.	●	●	-	"P.12-15"
PG_SetPosition	Elapsed value and feedback counter settings	Sets the elapsed value and the feedback counter to desired values.	●	●	-	"P.12-17"
PG_WriteParameter	Write parameters	Writes the parameters to the pulse output unit.	●	●	-	"P.12-18"
PG_ReadParameter	Read parameters	Reads the parameters from the pulse output unit.	●	●	-	"P.12-22"
PG_ClearError	Clear errors	Clears the limit error or the set value error of the pulse output unit.	●	●	-	"P.12-23"
PG_ReadStatus	Read status	Reads the status from the pulse output unit.	●	●	-	"P.12-24"

1.4 List of Function Block Instructions that Cannot Be Used with the GM1

■ Instructions not available for Modbus

The following function blocks in the IoDrvModbusTCP, IoDrvModbusTCPSlave, IoDrvModbus, and IoDrvModbusSerialSlave libraries are not available for the GM1 Controller.

Name	Function	Alternative function	Page
ModbusTCPSlaveBase	-	-	-
ModbusTCPSlaveUnit	-	-	-
ModbusTCPSlaveUnit_Diag	-	-	-
IoDrvModbusTCP_Diag	-	-	-
ModbusTCPSlave_Diag	-	-	-
ModbusTCPDeviceDiag	-	-	-
IoDrvModbusComPort_Diag	-	-	-
ModbusSlaveComPort_Diag	-	-	-
IoDrvModbusSerialSlave	-	-	-
ModbusSerialDeviceDiag	-	-	-
ModbusServer	-	-	-

■ Instructions not available for general-purpose communication

The following function blocks in the CAA NBS(Net Base Services) library are not available for the GM1 Controller.

Name	Function	Alternative function	Page
TCP_ReadBuffer	-	-	-
TCP_WriteBuffer	-	-	-
UDP_ReceiveBuffer	-	-	-
UDP_SendBuffer	-	-	-
DummyJob	-	-	-

■ Instructions not available for EtherNet/IP

The following function blocks in the IoDrvEtherNetIP and IoDrvEtherNetIPAdapter libraries are not available for the GM1 Controller.

Name	Function	Alternative function	Page
IoDrvEtherNetIP_diag	-	-	-
RemoteAdapter_diag	-	-	-
AdapterDiagnosis	-	-	-
IoDrvEtherNetIPAdapter_Diag	-	-	-
Module_Diag	-	-	-

■ Instructions not available for motion control

The following function blocks in the SM3_Basic library are not available for the GM1 Controller. Alternative functions are listed, if available.

Name	Function	Alternative function	Page
SMC_Commissioning	Commissioning status	Commissioning function of the GM Programmer	-
SMC_SetCustomRampType	Set acceleration / deceleration custom operation	-	-
SMC_CAM_ObjectManager	Manage cam data	-	-
SMC_ReadCAM	Read cam data	-	-
SMC_WriteCAM	Write cam data	-	-
SMC3_CommunicateDriveParameter	Communication setting	RTEX_ReadAmpParameter	"P.9-12"
SMC3_ReadDriveParameter	Read drive parameter	RTEX_ReadAmpParameter	"P.9-12"
SMC3_ReadParameter	Read parameter	RTEX_ReadAmpParameter	"P.9-12"
SMC3_WriteDriveParameter	Write drive parameter	RTEX_WriteAmpParameter	"P.9-13"
SMC3_WriteParameter	Write parameter	RTEX_WriteAmpParameter	"P.9-13"
MC_ReadBoolParameter	Read BOOL-type parameter	RTEX_ReadAmpParameter	"P.9-12"
MC_ReadParameter	Read parameter	RTEX_ReadAmpParameter	"P.9-12"
MC_WriteBoolParameter	Write BOOL-type parameter	RTEX_WriteAmpParameter	"P.9-13"
MC_WriteParameter	Write parameter	RTEX_WriteAmpParameter	"P.9-13"
SMC_VIRTUAL_AXIS	Set virtual axis	-	-
SMC3_BrakeStatus	Get brake status	-	-
SMC3_BrakeControl	Brake control	-	-
SMC3_PersistPosition	Persist actual axis position	-	-
SMC3_PersistPositionLogical	Persist logical axis position	-	-
SMC3_PersistPositionSinglerun	Persist actual axis position with a range	-	-
SMC_PerfStat	Calculate performance statistics	-	-
SMC_SeriesStat	Calculate increment statistics	-	-
SMC_AxisDiagnosticLog	Log axis parameter	-	-
FB_Template_Edge		-	-
FB_Template_EdgeAbort		-	-
FB_Template_EdgeAbortTimeout		-	-
SMC_StartupDrive		-	-
SMC_CAMBounds_Pos		-	-
SMC_CamEditor		-	-
SMC_PerfTimerSum		-	-
SMC_FollowPosition		-	-

Name	Function	Alternative function	Page
SMC_FollowPositionVelocity		-	-
SMC_FollowSetValues		-	-
SMC_FollowVelocity		-	-
SMC_Homing		-	-
ETC_CO_SdoInfoGeEntryDescription	Read object name	-	-
ETC_CO_SdoInfoGetODList	Read object tree	-	-
ETC_CO_SdoInfoGetObjectDescription	Read object information	-	-
ReadEEpromData	Read slave EEPROM	-	-
ReadWriteEEprom	Read / write slave EEPROM	-	-

(MEMO)

2 Ladder Instructions

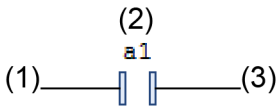
2.1 Ladder Instructions

This section describes ladder instructions that can be used for ladder diagram program (LD program).

2.1.1 NO Contact

If the variable corresponding to the contact is TRUE, then the input value is output. If the variable is FALSE, then FALSE is output.

■ Icon



■ Parameter

No.	Scope	Type	Description
(1)	Input	BOOL	Input to the NO contact
(2)	Variable name	BOOL	Variable that corresponds to the NO contact
(3)	Output	BOOL	Output from the NO contact

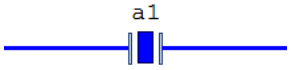
■ Input method

Use one of the following methods to input the NO contact.

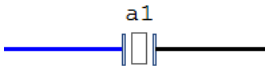
- From the tool box, select **Ladder elements**>**NO contact** and drag to "Start from here".
- Right-click on the network, and, from the displayed menu, select **Insert Contact**.
- Click the icon on the toolbar.
- From the menu, select **FBD / LD / IL**>**Insert Contact**.
- Press the shortcut keys <Ctrl+k> simultaneously.

■ Program example

If the variable (a1) corresponding to the NO contact is TRUE, then the value input to the NO contact (TRUE) is output as is.



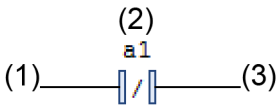
If the variable (a1) corresponding to the contact is FALSE, then FALSE is output.



2.1.2 NC Contact

If the variable corresponding to the contact is TRUE, then FALSE is output. If the variable is FALSE, then the input value is output.

■ Icon



■ Parameter

No.	Scope	Type	Description
(1)	Input	BOOL	Input to the NC contact
(2)	Variable name	BOOL	Variable that corresponds to the NC contact
(3)	Output	BOOL	Output from the NC contact

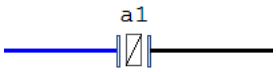
■ Input method

Use one of the following methods to input the NC contact.

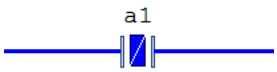
- From the tool box, select **Ladder elements**>**NC contact** and drag to "Start from here".
- Right-click on the network, and, from the displayed menu, select "Insert NC contact".
- Click the icon on the toolbar.
- From the menu, select **FBD / LD / IL**>**Insert NC contact**.

■ Program example

If the variable (a1) corresponding to the NC contact is TRUE, then FALSE is output.



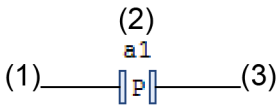
If the variable (a1) corresponding to the NC contact is FALSE, then the value input to the NC contact (TRUE) is output as is.



2.1.3 Rising Edge Detection Contact

If a rising edge is detected in the variable corresponding to the contact, then the input value is output for one cycle only.

■ Icon



■ Parameter

No.	Scope	Type	Description
(1)	Input	BOOL	Input to the contact
(2)	Variable name	BOOL	Variable that corresponds to the rising edge detection contact
(3)	Output	BOOL	Output from the contact

■ Input method

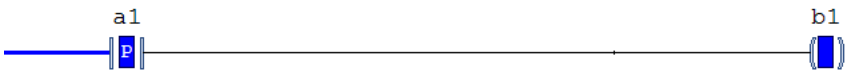
Input the rising edge detection contact by first inputting the NO contact and then changing the NO contact.

Select the NO contact just input and then perform one of the following operations.

- Right-click and, from the displayed menu, select **Edge detection** .
- From the menu, select **FBD / LD / IL**>**Edge detection** .
- Press the shortcut keys <Ctrl+e> simultaneously.
- Click the icon on the toolbar.

■ Program example

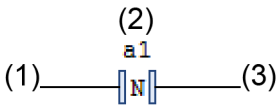
The following program is designed to detect the rising edge with the variable (a1) corresponding to the rising edge detection contact and to output TRUE for one cycle only.



2.1.4 Falling Edge Detection Contact

If a falling edge is detected in the variable corresponding to the contact, then the input value is output for one cycle only.

■ Icon



■ Parameter

No.	Scope	Type	Description
(1)	Input	BOOL	Input to the contact
(2)	Variable name	BOOL	Variable that corresponds to the falling edge detection contact
(3)	Output	BOOL	Output from the contact

■ Input method

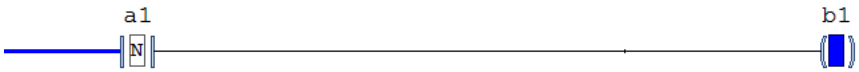
Input the falling edge detection contact by first inputting the NO contact and then changing the NO contact.

Select the NO contact just input and then perform one of the following operations.

- Right-click and, from the displayed menu, select **Edge detection** twice.
- From the menu, select **FBD / LD / IL>Edge detection** twice.
- Press the shortcut keys Ctrl+e simultaneously twice.
- Click the icon on the toolbar twice.

■ Program example

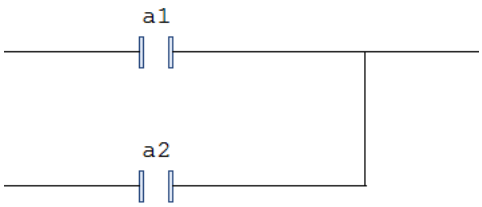
The following program is designed to detect the falling edge with the variable (a1) corresponding to the falling edge detection contact and to output TRUE for one cycle only.



2.1.5 Parallel NO Contact

NO contacts can be input in parallel to the initial contact. Of the contacts wired in parallel, if the output of one or more contacts is TRUE, TRUE is output.

■ Icon



■ Input method

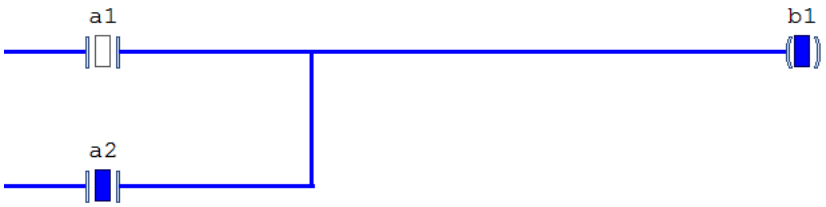
To input a parallel NO contact below the initial contact, select **Ladder elements** >**Parallel NO contact** from the tool box and drag to the position indicated with "▼" next to the contact.

Or, with the contact selected, perform one of the following operations.

- Right-click, and, from the displayed menu, select **Insert contact in parallel (below)**.
- From the menu, select **FBD/LD/IL**>**Insert contact in parallel (below)**.
- Press the shortcut keys <Ctrl+r> simultaneously.
- Click the icon on the toolbar.

■ Program example

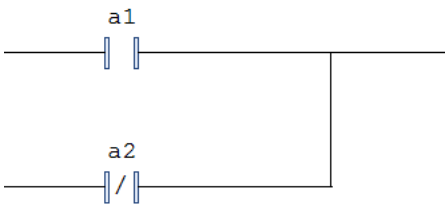
This program is designed to input one NO contact in parallel to the NO contact. TRUE is output because the NO contact below is TRUE.



2.1.6 Parallel NC Contact

NC contacts can be input in parallel to the initial contact. Of the contacts wired in parallel, if the output of one or more contacts is TRUE, TRUE is output.

■ Icon



■ Input method

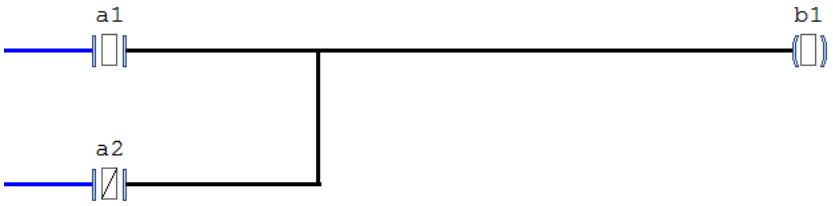
To input a parallel NC contact below the initial contact, select **Ladder elements** >**Parallel NC contact** from the tool box and drag to the position indicated with "▼" next to the contact.

Or, with the contact selected, perform one of the following operations.

- Right-click, and, from the displayed menu, select "Insert NC contact in parallel (below)".
- From the menu, select **FBD / LD / IL**>**Insert NC contact in parallel (below)**.
- Click the icon on the toolbar.

■ Program example

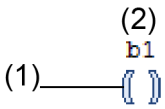
This program is designed to input one NC contact in parallel to the NO contact. FALSE is output because the outputs of both contacts are FALSE.



2.1.7 Coil

The input value is saved in the variable corresponding to the coil. If the input value is TRUE, then TRUE is saved. If the input value is FALSE, then FALSE is saved.

■ Icon



■ Parameter

No.	Scope	Type	Description
(1)	Input	BOOL	Input to the coil
(2)	Variable name	BOOL	Name of the variable that corresponds to the coil

■ Input method

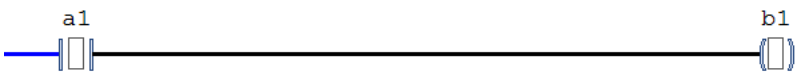
Use one of the following methods to input the coil.

- From the tool box, select **Ladder elements**> **Coil** and drag to "Add output or jump here" (when connecting to a contact).
- Right-click on the network, and, from the displayed menu, select **Insert Coil** .
- Click the icon on the tool bar.
- From the menu, select **FBD / LD / IL**>**Insert Coil** .
- Press the shortcut keys <Ctrl+a> simultaneously.
- Click the icon on the toolbar.

■ Program example

This program is designed to input the output from the NO contact to the coil.
TRUE is saved in the variable (b1) because the input to the coil is TRUE.

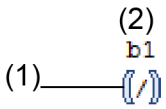
FALSE is saved in the variable (b1) because the input to the coil is FALSE.



2.1.8 Negated Coil

The negated value of the input is saved in the variable corresponding to the coil. If the input value is TRUE, then FALSE is saved. If the input value is FALSE, then TRUE is saved.

■ Icon



■ Parameter

No.	Scope	Type	Description
(1)	Input	BOOL	Input to the negated coil
(2)	Variable name	BOOL	Name of the variable that corresponds to the negated coil

■ Input method

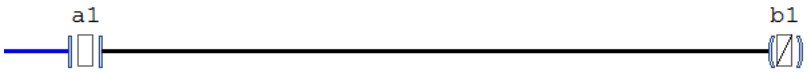
The negated coil can be input by inputting a coil and changing it.
With the input coil selected, perform one of the following operations.

- Right-click and, from the displayed menu, select **Negation**.
- From the menu, select **FBD / LD / IL>Negation**.
- Press the shortcut keys <Ctrl+n> simultaneously.
- Click the icon on the toolbar.

■ Program example

This program is designed to input the output from the NO contact to the negated coil.
FALSE is saved in the variable (b1) because the input to the coil is TRUE.

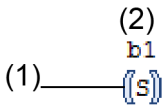
TRUE is saved in the variable (b1) because the input to the coil is FALSE.



2.1.9 Set Coil

When the input value turns TRUE, TRUE is saved in the variable corresponding to the coil. TRUE is held until the input to the reset coil that corresponds to the same variable turns TRUE.

■ Icon



■ Parameter

No.	Scope	Type	Description
(1)	Input	BOOL	Input to the set coil.
(2)	Variable name	BOOL	Name of the variable that corresponds to the set coil

■ Input method

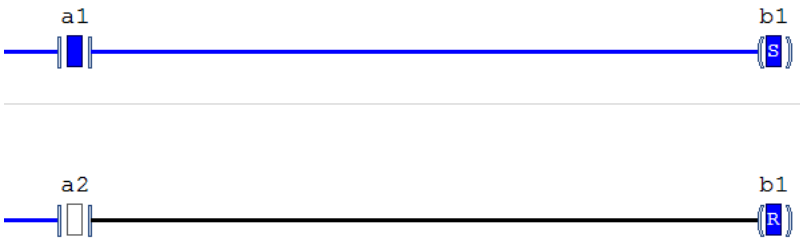
Use one of the following methods to input the set coil.

- From the tool box, select **Ladder elements**>**Set Coil** and drag to "Add output or jump here" (when connecting to a contact).
- Right-click on the network, and, from the displayed menu, select "Insert Set Coil ".
- Click the icon on the toolbar.
- From the menu, select **FBD / LD / IL**>**Insert Set Coil**.

■ Program example

This program is designed to input the output from the NO contact to the set coil and the reset coil.

TRUE is saved in the set coil variable (b1) because the input to the set coil is TRUE.



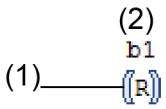
Info.

- Each set coil should be accompanied by a reset coil.

2.1.10 Reset Coil

When the input value turns TRUE, FALSE is saved in the variable corresponding to the coil. FALSE is held until the input to set coil that corresponds to the same variable turns TRUE.

■ Icon



■ Parameter

No.	Scope	Type	Description
(1)	Input	BOOL	Input to the reset coil.
(2)	Variable name	BOOL	Name of the variable that corresponds to the reset coil

■ Input method

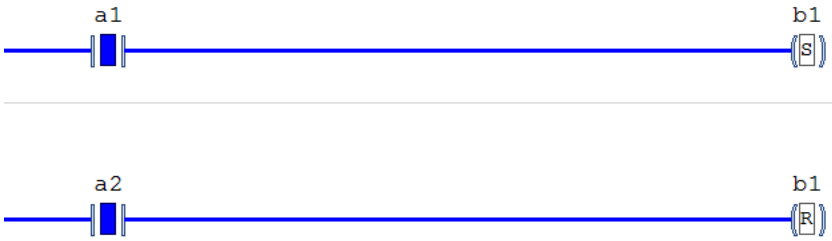
Use one of the following methods to input the reset coil.

- From the tool box, select **Ladder elements**>**Reset Coil** and drag to "Add output or jump here" (when connecting to a contact).
- Right-click on the network, and, from the displayed menu, select **Reset Coil**.
- Click the icon on the toolbar.
- From the menu, select **FBD / LD / IL**>**Insert Reset Coil** .

■ Program example

This program is designed to input the output from the NO contact to the set coil and the reset coil.

FALSE is saved in the variable (b1) because the input to the reset coil is TRUE.



Info.

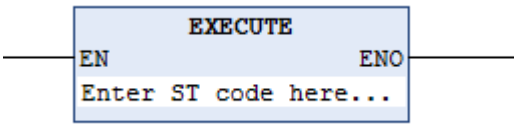
- Each set coil should be accompanied by a reset coil.

2.1.11 Execute Box

You can program in ST language by inserting an execute box in LD language.

If "Enter ST code here ..." is clicked, an input field using a multi-line ST will open.

■ Icon



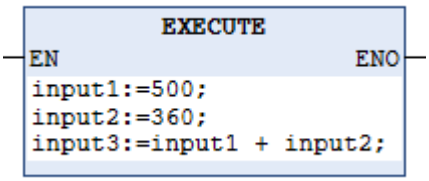
■ Input method

You can enter "Execute Box" by any of the following operations.

- Select **General→Execute** in the toolbox and drag it to the position of ◆ displayed next to the contact.
- Right-click on the network and select Insert "Execute Box" from the menu that appears
- Select **FBD/LD/IL**>**Execute Box** from the menu

■ Program example

When the EN condition is ON, the entered ST language code will be executed.



3 Functions

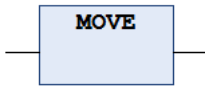
3.1 Basic Instructions

You can use basic instructions to assign the values of other variables to variables, specify addresses, and get sizes.

3.1.1 MOVE (Substitution)

This is a function that substitutes the value of a variable specified in the input for a variable specified in the output.

■ Icon



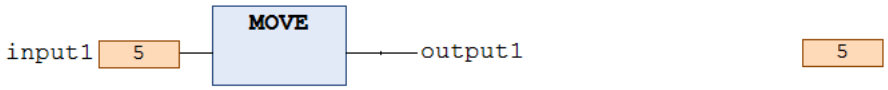
■ Parameter

Scope	Type	Description
Input	All	Specifies the variable of the substitution source.
Output	All	Specifies the variable of the substitution target.

■ Program example

This program is designed to substitute the value of input variable “input1” for the output variable “output1”.

LD program



ST program

```
output1 5 := MOVE(input1 5 );
```

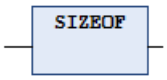
It is also possible to substitute the value using an operator (:=).

```
output1 5 := input1 5 ;
```

3.1.2 SIZEOF (Get the Size)

This is a function that outputs the size (number of bytes) of the input argument.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the argument whose size is to be calculated.
Output	(Note 1)	Outputs the size of (1).

(Note 1) Usable data types

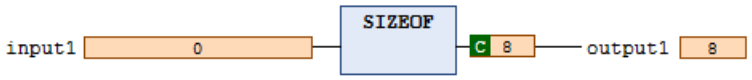
All standard data types

(BOOL, BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, LTIME, DATE, TIME_OF_DAY, DATE_AND_TIME, STRING, WSTRING)

■ Program example

This program is designed to output the size of the ULINT type input variable “input1” to the UINT type output variable “output1”.

LD program



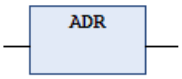
ST program

```
output1  8  := SIZEOF(input1  0  );
```

3.1.3 ADR (Get the Address)

This is a function that outputs the address of the variable.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Input the variable from which to get the address.
Output	(Note 1)	Outputs the address (pointer) of the input variable.

(Note 1) Usable data types

All standard data types

(BOOL, BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, LTIME, DATE, TIME_OF_DAY, DATE_AND_TIME, STRING, WSTRING)

■ Usable data type

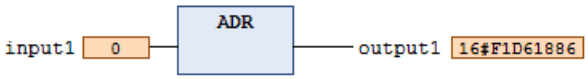
All standard data types

(BOOL, BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, LTIME, DATE, TIME_OF_DAY, DATE_AND_TIME, STRING, WSTRING)

■ Program example

This program is designed to output the address of the input variable “input1” to the output variable “output1”.

LD program



ST program

```
output1 16#F1D61884 := ADR(input1 0 );
```

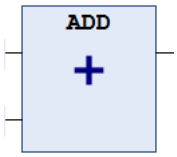
3.2 Arithmetic Operation Instructions

Arithmetic operation instructions can be used to perform calculation such as four arithmetic operations.

3.2.1 ADD (Addition)

This is a function that adds input arguments and outputs the sum.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the variables to be added.
Output	(Note 1)	Outputs the sum of variables specified in the input.

(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, TIME_OF_DAY, DATE_AND_TIME

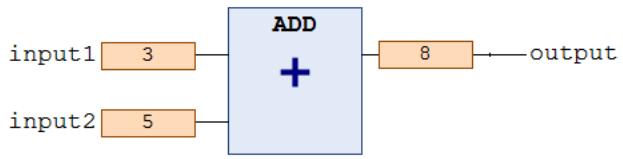
Time type data can be added in the following combinations.

- TIME + TIME = TIME
- TIME_OF_DAY + TIME = TIME_OF_DAY
- DATE_AND_TIME + TIME = DATE_AND_TIME

■ Program example

This program is designed to output the sum of input variables “input1” and “input2” to the output variable “output”.

LD program



ST program

It is possible to add the values using “+” operator.

```
output 8 := input1 3 + input2 5;
```

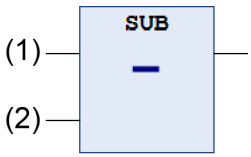
Info.

- If you want to increase input arguments in the LD program, right-click on the ADD function, and, on the displayed menu, select "Add Input".

3.2.2 SUB (Subtraction)

This is a function that subtracts input arguments and outputs the difference.

■ Icon



■ Parameter

Scope	Number	Type	Description
Input	(1), (2)	(Note 1)	Specifies the variables to be subtracted.
Output	-	(Note 1)	Outputs the value obtained by subtracting the input (2) from the input (1).

(Note 1) Usable data types

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, TIME_OF_DAY, DATE, DATE_AND_TIME

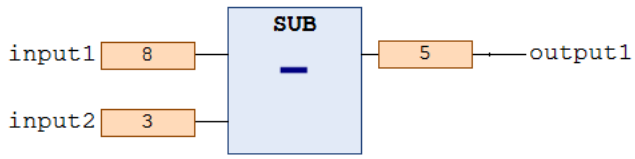
For time type data, subtraction can be performed in the following combinations. Note that negative time cannot be calculated.

- TIME - TIME = TIME
- DATE - DATE = TIME
- TOD - TIME = TOD
- TOD - TOD = TIME
- DT - TIME = DT
- DT - DT = TIME

■ Program example

This program is designed to output the difference between the input variables “input1” and “input2” to the output variable “output1”.

LD program



ST program

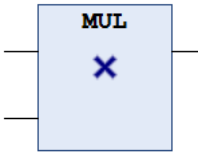
It is possible to subtract the values using “-” operator.

```
output1 5 := input1 8 - input2 3 ;
```

3.2.3 MUL (Multiplication)

This is a function that multiplies input arguments and outputs the product.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the variables to be multiplied.
Output	(Note 1)	Outputs the product of variables specified in the input.

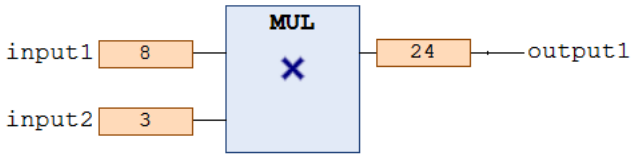
(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME

■ Program example

This program is designed to output the product of the input variables “input1” and “input2” to the output variable “output1”.

LD program



ST program

It is possible to multiply the values using “*” operator.

```
output1 24 := input1 8 * input2 3;
```

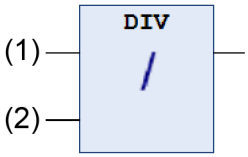
Info.

- If you want to increase input arguments in the LD program, right-click on the MUL function, and, on the displayed menu, select "Add Input".
- TIME type data cannot be multiplied by REAL type, LREAL type, or TIME type data.

3.2.4 DIV (Division)

This is a function that divides input arguments and outputs the quotient.

■ Icon



■ Parameter

Scope	No.	Type	Description
Input	(1), (2)	(Note 1)	Specifies the variables to be divided.
Output	-	(Note 1)	Outputs the quotient obtained by dividing the input (2) by the input (1).

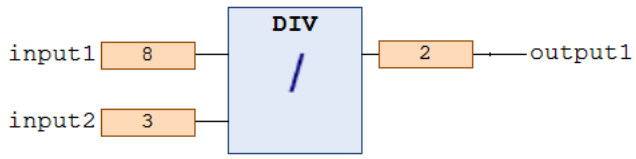
(Note 1) Usable data types

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME

■ Program example

This program is designed to output the quotient of the INT type input variables “input1” and “input2” to the INT type output variable “output1”.

LD program



ST program

It is possible to divide the values using the division operator ("/") .

```
output1 2  := input1 8  / input2 3 ;
```

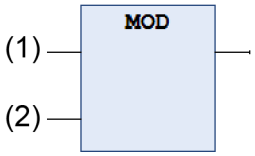
Info.

- TIME type variables can be divided by integer type variables.
- When a variable is divided by a DINT, LINT, REAL, or LREAL type variable, it can be checked if 0 is used in the calculation. (Refer to “Auto Check POU” in the “SMC Tool Introduction Guide”.)

3.2.5 MOD (Remainder)

This is a function that divides input arguments and outputs the remainder.

■ Icon



■ Parameter

Scope	No.	Type	Description
Input	(1), (2)	(Note 1)	Specifies the variables to be divided.
Output	-	(Note 1)	Outputs the remainder obtained by dividing the input (2) by the input (1).

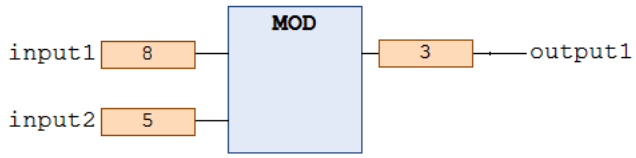
(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT

■ Program example

This program is designed to output the remainder obtained from dividing the INT type input variables “input1” and “input2” to the INT type output variable “output1”.

LD program



ST program

```
output1 3 := input1 8 MOD input2 5;
```

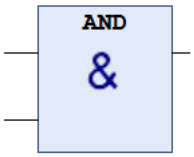
3.3 Boolean Operation Instructions

Boolean operation instructions can be used to perform bool operations such as logical AND or logical OR.

3.3.1 AND (Logical AND)

This is a function that outputs logical AND of the input arguments.

■ **Icon**



■ **Parameter**

Scope	Type	Description
Input	(Note 1)	Specifies the variables to be used to obtain logical AND.
Output	(Note 1)	Outputs the logical AND of the variables specified in the input.

(Note 1) Usable data type

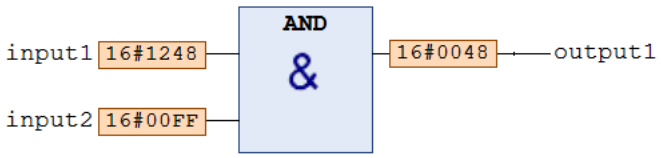
BOOL, BYTE, WORD, DWORD, LWORD

■ **Program example**

This program is designed to output the logical AND of the WORD type input variables “input1” and “input2” to the output variable “output1”.

The execution result is displayed in a hexadecimal number.

LD program



ST program

```
output1 16#0048 := input1 16#1248 AND input2 16#00FF ;
```

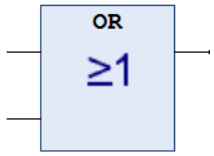
Info.

- If you want to increase input arguments in the LD program, right-click on the AND function, and, on the displayed menu, select "Add Input".

3.3.2 OR (Logical OR)

This is a function that outputs logical OR of the input arguments.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the variables to be used to obtain logical OR.
Output	(Note 1)	Outputs the logical OR of the variables specified in the input.

(Note 1) Usable data type

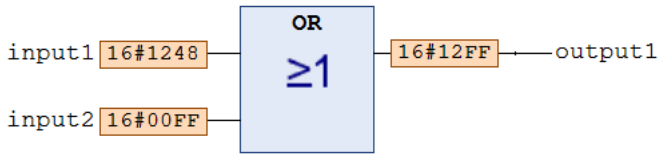
BOOL, BYTE, WORD, DWORD, LWORD

■ Program example

This program is designed to output the logical OR of the WORD type input variables “input1” and “input2” to the output variable “output1”.

The execution result is displayed in a hexadecimal number.

LD program



ST program

```
output1 16#12FF := input1 16#1248 OR input2 16#00FF;
```

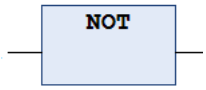
Info.

- If you want to increase input arguments in the LD program, right-click on the OR function, and, on the displayed menu, select "Add Input".

3.3.3 NOT (Negation)

This is a function that outputs the negation of the input argument.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the variable to be used to obtain the negation.
Output	(Note 1)	Outputs the negation of the variable specified in the input.

(Note 1) Usable data type

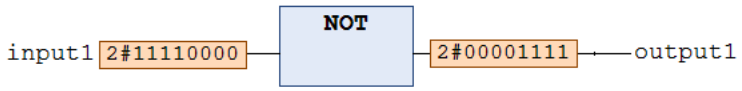
BOOL, BYTE, WORD, DWORD, LWORD

■ Program example

This program is designed to output the negation of the BYTE type input variable “input1” to the output variable “output1”.

The execution result is displayed in a binary number.

LD program



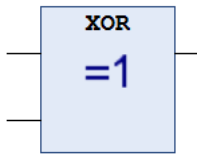
ST program

```
output1 2#00001111 := NOT input1 2#11110000;
```

3.3.4 XOR (Exclusive OR)

This is a function that outputs exclusive OR of the input arguments.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the variables to be used to obtain exclusive OR.
Output	(Note 1)	Outputs the exclusive OR of the variables specified in the input. Outputs 0 if both input bits are 1 or 0. Outputs 1 if one of the two input bits is 1 and the other bit is 0.

(Note 1) Usable data type

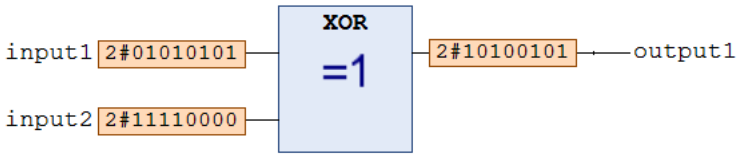
BOOL, BYTE, WORD, DWORD, LWORD

■ Program example

This program is designed to output the exclusive OR of the BYTE type input variables “input1” and “input2” to the output variable “output1”.

The execution result is displayed in a binary number.

LD program



ST program

```
output1 2#10100101 := input1 2#01010101 XOR input2 2#11110000 ;
```

3.3.5 AND_THEN (Logical AND)

This is a conditional AND evaluation function of the input operand.

■ Usable data types

BOOL, BIT

■ Program example

This program is designed to compare the value of the variable accessed by pwAddress (pointer) with wExpected if the pwAddress is not NULL and, if they are the same, substitute with the value of wNewValue.

As default values, “5” is stored in the variable “test1” accessed by pwAddress, “5” in wExpected, and “3” in wNewValue.

As an initial step, judgment is made whether pwAddress is NULL or not. Since it is not NULL, comparison is made between the value of “test1” and the value of wExpected as the next step. Since these two values are both “5”, TRUE is assigned. As a result, the value of wNewValue “3” is stored in the “test1” and the xFlag flag is set to TRUE.

ST program

[Declaration section]

```
VAR
    pwAddress   : POINTER TO WORD;
    wExpected   : WORD := 5;
    wNewValue   : WORD := 3;
    xFlag       : BOOL;
    test1       : WORD := 5;
END_VAR
```

[Implemment section]

```
pwAddress 16#F1D10BBE := ADR(test1 16#0003 );

IF pwAddress 16#F1D10BBE <> 0 AND_THEN pwAddress^ 16#0003 = wExpected 16#0005 THEN
    pwAddress^ 16#0003 := wNewValue 16#0003 ;
    xFlag TRUE := TRUE;
ELSE
    xFlag TRUE := FALSE;
END_IF
```

Info.

- Expressions of other operands are executed only when the first operand is TRUE.

  Therefore, if no value is stored in pwAddress in the above example, the initial NULL judgment turns FALSE. As a result, no judgment is performed on operands after the AND_THEN operator.

3.3.6 OR_ELSE (Logical OR)

This is a conditional OR evaluation function of the input operand.

■ Usable data types

BOOL, BIT

■ Program example

16#000000FF is stored in the variable dw.
”dw.8” that represents bit 8 of dw is FALSE and “dw.1” that represents bit 1 is TRUE.
Therefore, the operation result flag bX is TRUE.
Note that the third input expression is not executed and bEver remains FALSE.

ST program

[Declaration section]

```
VAR
    bEver   : BOOL;
    bX      : BOOL;
    dw      : DWORD := 16#000000FF;
END_VAR
```

[Implemment section]

```
bEver := FALSE;
bX := dw.8 OR_ELSE dw.1 OR_ELSE dw.1 OR_ELSE (bEver := TRUE);
```

Info.

- In case of OR_ELSE, when one of the operands is evaluated TRUE, all other operator expressions are not evaluated.

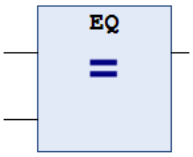
3.4 Comparison Operation Instructions

Comparison operation instructions can be used to compare two arguments.

3.4.1 EQ (“Equal” Comparison)

This is a function that compares two input arguments and determines if they are the same value.

■ **Icon**



■ **Parameter**

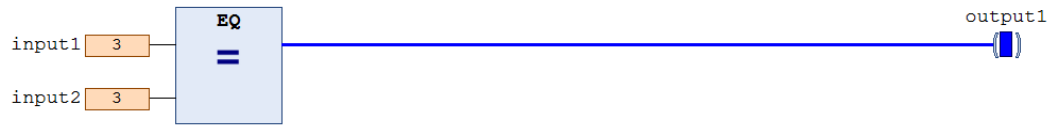
Scope	Type	Description
Input	All	Specifies the variables to be compared.
Output	BOOL	Outputs TRUE if the input variable values are the same. Outputs FALSE if they are different.

■ **Program example**

This program is designed to compare the input variables “input1” and “input2” and output the result to the output variable “output1”.

LD program

TRUE is output because the input variable values “input1” and “input2” are the same.



ST program

Use the operator (=) to compare the values.

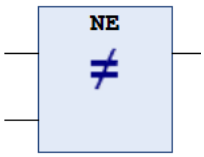
FALSE is output because the input variable values “input1” and “input2” are different.

```
output1 FALSE := (input1 3 = input2 5);
```

3.4.2 NE (“Not Equal” Comparison)

This is a function that compares two input arguments and determines if they are not the same.

■ Icon



■ Parameter

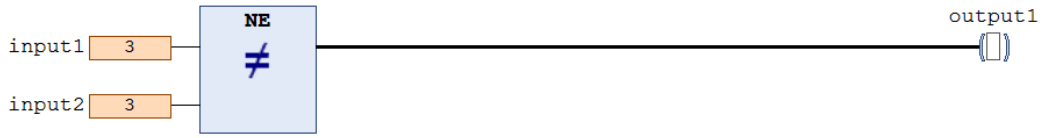
Scope	Type	Description
Input	All	Specifies the variables to be compared.
Output	BOOL	Outputs TRUE if the input variable values are different. Outputs FALSE if they are the same.

■ Program example

This program is designed to compare the input variables “input1” and “input2” and output the result to the output variable “output1”.

LD program

FALSE is output because the input variable values “input1” and “input2” are the same.



ST program

Use the operator (<>) to compare the values.

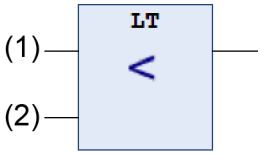
TRUE is output because the input variable values “input1” and “input2” are different.

```
output1 TRUE := input1 3 <> input2 5;
```

3.4.3 LT (“Less Than” Comparison)

This is a function that compares two input arguments and determines if the first argument is less than the second argument.

■ Icon



■ Parameter

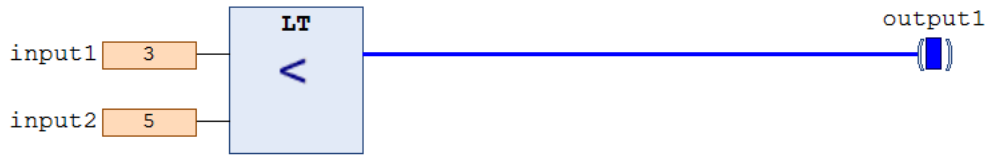
Scope	No.	Type	Description
Input	(1), (2)	All	Specifies the variables to be compared.
Output	-	BOOL	Outputs TRUE if the value of input (1) is less than the value of input (2). Otherwise, outputs FALSE.

■ Program example

This program is designed to compare the input variables “input1” and “input2” and output the result to the output variable “output1”.

LD program

TRUE is output because the input variable “input1” is less than the input variable “input2”.



ST program

Use the operator (<) to compare the values.

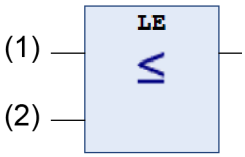
FALSE is output because the input variable “input1” is not less than the input variable “input2”.

```
output1 FALSE := input1 6 < input2 2;
```

3.4.4 LE (“Less Than or Equal” Comparison)

This is a function that compares two input arguments and determines if the first argument is less than or equal to the second argument.

■ Icon



■ Parameter

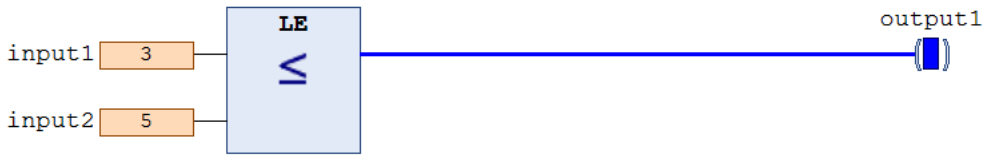
Scope	No.	Type	Description
Input	(1), (2)	All	Specifies the variables to be compared.
Output	-	BOOL	Outputs TRUE if the value of input (1) is less than or equal to the value of input (2). Otherwise, outputs FALSE.

■ Program example

This program is designed to compare the input variables “input1” and “input2” and output the result to the output variable “output1”.

LD program

TRUE is output because the input variable “input1” is less than or equal to the input variable “input2”.



ST program

Use the operator (<=) to compare the values.

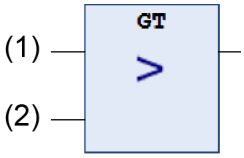
FALSE is output because the input variable “input1” is not less than or equal to the input variable “input2”.

```
output1 FALSE := input1 6 <= input2 2 ;
```

3.4.5 GT (“Greater Than” Comparison)

This is a function that compares two input arguments and determines if the first argument is greater than the second argument.

■ Icon



■ Parameter

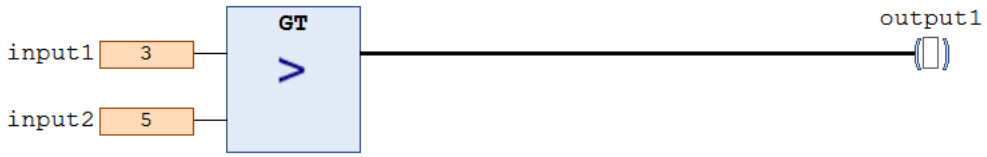
Scope	No.	Type	Description
Input	(1), (2)	All	Specifies the variables to be compared.
Output	-	BOOL	Outputs TRUE if the value of input (1) is greater than the value of input (2). Otherwise, outputs FALSE.

■ Program example

This program is designed to compare the input variables “input1” and “input2” and output the result to the output variable “output1”.

LD program

FALSE is output because the input variable “input1” is not greater than the input variable “input2”.



ST program

Use the operator (>) to compare the values.

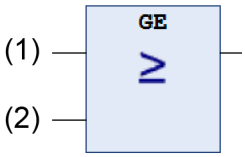
TRUE is output because the input variable “input1” is greater than the input variable “input2”.

```
output1 TRUE := input1 6 > input2 2;
```

3.4.6 GE (“Greater Than Or Equal” Comparison)

This is a function that compares two input arguments and determines if the first argument is greater than or equal to the second argument.

■ Icon



■ Parameter

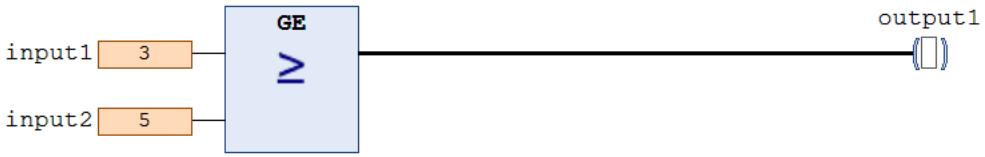
Scope	No.	Type	Description
Input	(1), (2)	All	Specifies the variables to be compared.
Output	-	BOOL	Outputs TRUE if the value of input (1) is greater than or equal to the value of input (2). Otherwise, outputs FALSE.

■ Program example

This program is designed to compare the input variables “input1” and “input2” and output the result to the output variable “output1”.

LD program

FALSE is output because the input variable “input1” is not greater than or equal to the input variable “input2”.



ST program

Use the operator (>=) to compare the values.

TRUE is output because the input variable “input1” is greater than or equal to the input variable “input2”.

```
output1 TRUE := input1 6 >= input2 2 ;
```

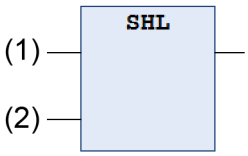
3.5 Bit Shift Instructions

Bit shift instructions can be used to perform bit shift operation on input arguments.

3.5.1 SHL (Shift Left)

This is a function that shifts the input argument to the left by the specified number of bits and outputs the shifted value. “0” is inserted from the least significant bit up to the bit position shifted by the shift quantity.

■ Icon



■ Parameter

Scope	No.	Type	Description
Input	(1)	(Note 1)	Specifies the variable on which bit shift is performed.
	(2)	(Note 1)	Specifies the number of times bit shift is performed (shift quantity).
Output	-	(Note 1)	Outputs the value bit shifted to the left from the value of input (1) by the quantity specified in the input (2).

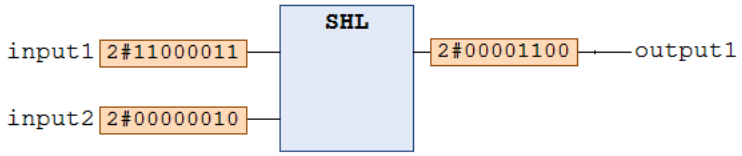
(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT

■ Program example

This program is designed to output the value that is shifted to the left from the value (2#11000011) of input variable “input1” by the number of bits (2 bits) specified in “input2” to the output variable “output1”.

LD program



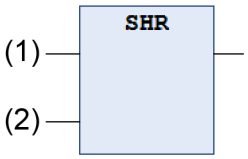
ST program

```
output1 2#00001100 := SHL(input1 2#11000011, input2 2#00000010);
```

3.5.2 SHR (Shift Right)

This is a function that shifts the input argument to the right by the specified number of bits and outputs the shifted value. “0” is inserted from the most significant bit up to the bit position shifted by the shift quantity.

■ Icon



■ Parameter

Scope	No.	Type	Description
Input	(1)	(Note 1)	Specifies the variable on which bit shift is performed.
	(2)	(Note 1)	Specifies the number of times bit shift is performed (shift quantity).
Output	-	(Note 1)	Outputs the value bit shifted to the right from the value of input (1) by the quantity specified in the input (2).

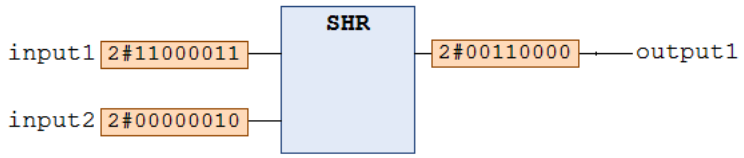
(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT

■ Program example

This program is designed to output the value that is shifted to the right from the value (2#11000011) of input variable “input1” by the number of bits (2 bits) specified in “input2” to the output variable “output1”.

LD program



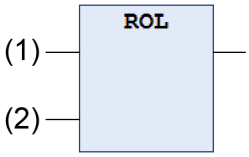
ST program

```
output1 2#00110000 := SHR(input1 2#11000011, input2 2#00000010);
```

3.5.3 ROL (Rotate Left)

This is a function that shifts the input argument to the left by the specified number of bits and outputs the shifted value. The bit value that has overflowed the most significant bit when the bit is shifted is inserted into the data starting from the least significant bit up to the bit position shifted by the shift quantity.

■ Icon



■ Parameter

Scope	No.	Type	Description
Input	(1)	(Note 1)	Specifies the variable on which bit shift is performed.
	(2)	(Note 1)	Specifies the number of times bit shift is performed (shift quantity).
Output	-	(Note 1)	Outputs the value rotated and shifted to the left from the value of input (1) by the quantity specified in the input (2).

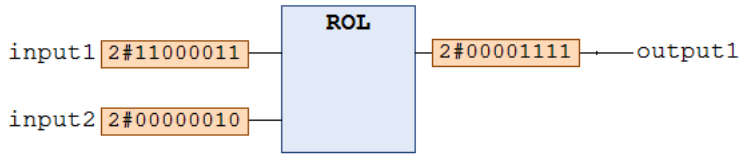
(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT

■ Program example

This program is designed to output the value that is rotated and shifted to the left from the value (2#11000011) of input variable “input1” by the number of bits (2 bits) specified in “input2” to the output variable “output1”.

LD program



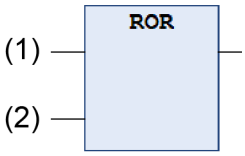
ST program

```
output1 2#00001111 := ROL(input1 2#11000011, input2 2#00000010);
```

3.5.4 ROR (Rotate Right)

This is a function that shifts the input argument to the right by the specified number of bits and outputs the shifted value. The bit value that has overflowed the least significant bit when the bit is shifted is inserted into the data starting from the most significant bit up to the bit position shifted by the shift quantity.

■ Icon



■ Parameter

Scope	No.	Type	Description
Input	(1)	(Note 1)	Specifies the variable on which bit shift is performed.
	(2)	(Note 1)	Specifies the number of times bit shift is performed (shift quantity).
Output	-	(Note 1)	Outputs the value rotated and shifted to the right from the value of input (1) by the quantity specified in the input (2).

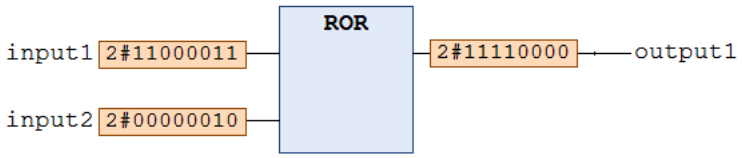
(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT

■ Program example

This program is designed to output the value that is rotated and shifted to the right from the value (2#11000011) of input variable “input1” by the number of bits (2 bits) specified in “input2” to the output variable “output1”.

LD program



ST program

```
output1 2#11110000 := ROR(input1 2#11000011, input2 2#00000010);
```

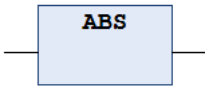
3.6 Numerical Operation Instructions

Numerical operation instructions can be used to perform various numerical calculations.

3.6.1 ABS (Absolute Value)

This is a function that outputs the absolute value of the input argument.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the value from which to obtain the absolute value.
Output	(Note 1)	Outputs the absolute value of the input argument.

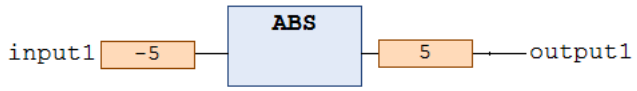
(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL

■ Program example

This program is designed to output the absolute value of the input variable “input1” to the output variable “output1”.

LD program



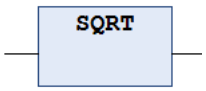
ST program

```
output1 5 := ABS(input1 -5);
```

3.6.2 SQRT (Square Root)

This is a function that outputs the square root (√) of the input argument.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the value from which to obtain the square root.
Output	(Note 2)	Outputs the square root of the input argument.

(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL

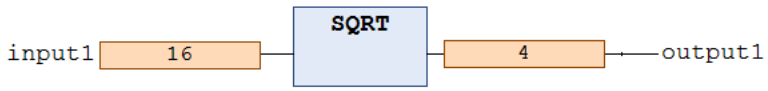
(Note 2) Usable data type

REAL (if the input is REAL), LREAL

■ Program example

This program is designed to output the square root of the input variable “input1” to the output variable “output1”.

LD program



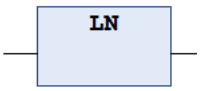
ST program

```
output1 4 := SQRT(input1 16);
```

3.6.3 LN (Natural Logarithm)

This is a function that outputs the natural logarithm ($\log_e X$) of the input argument (X).

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the value from which to obtain the natural logarithm.
Output	(Note 2)	Outputs the natural logarithm of the input argument.

(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL

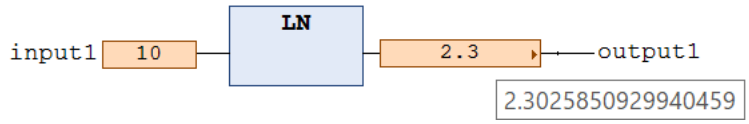
(Note 2) Usable data type

REAL (if the input is REAL), LREAL

■ Program example

This program is designed to output the natural logarithm ($\log_e 10$) of the input variable “input1” (10) to the output variable “output1”.

LD program



ST program

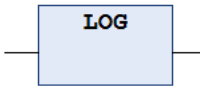
```
output1 2.3 := LN(input1 10);
```

2.3025850929940459

3.6.4 LOG (Common Logarithm)

This is a function that outputs the common logarithm ($\log_{10}X$) of the input argument (X).

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the value from which to obtain the common logarithm.
Output	(Note 2)	Outputs the common logarithm of the input argument.

(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL

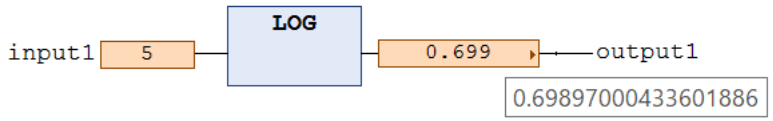
(Note 2) Usable data type

REAL (if the input is REAL), LREAL

■ Program example

This program is designed to output the common logarithm ($\log_{10}5$) of the input variable “input1” (5) to the output variable “output1”.

LD program



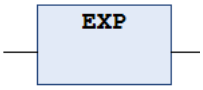
ST program

```
output1 0.699 := LOG(input1 5);
        0.69897000433601886
```

3.6.5 EXP (Natural Exponent)

This is a function that outputs the natural exponent (e^X) of the input argument (X).

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the value from which to obtain the natural exponent.
Output	(Note 2)	Outputs the natural exponent of the input argument.

(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL

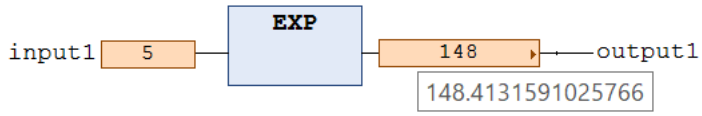
(Note 2) Usable data type

REAL (if the input is REAL), LREAL

■ Program example

This program is designed to output the natural exponent of the input variable “input1” to the output variable “output1”.

LD program



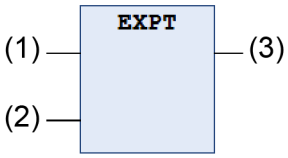
ST program

```
output1 148 := EXP(input1 5);
        148.4131591025766
```

3.6.6 EXPT (Exponentiation)

This is a function that outputs the exponentiation (a^n) of the input arguments (a, n).

■ Icon



■ Parameter

Scope	No.	Type	Description
Input	(1)	(Note 1)	Inputs the base of exponentiation.
	(2)	(Note 1)	Inputs the exponent of exponentiation.
Output	(3)	(Note 2)	Outputs the exponentiation obtained from the input arguments. Outputs a^n in the following case. Input (1): a Input (2): n

(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL

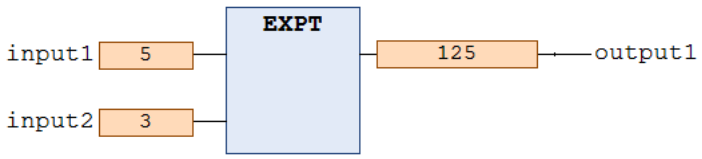
(Note 2) Usable data type

REAL (if the input is REAL), LREAL

■ Program example

This program is designed to output the exponentiation ($5^3 = 125$) obtained from the input variables “input1” and “input2” to the output variable “output1”.

LD program



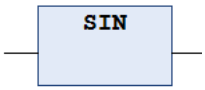
ST program

```
output1 125 := EXPT(input1 5, input2 3);
```

3.6.7 SIN (Trigonometric Function Sine)

This is a function that outputs the value of the trigonometric function sine. The unit of the input argument is radian.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the value (unit: radian) from which to obtain the trigonometric function sine.
Output	(Note 2)	Outputs the value of sine of the input argument.

(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL

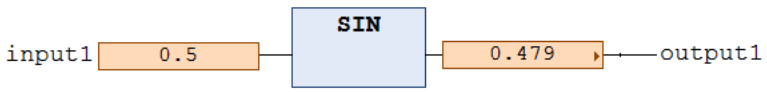
(Note 2) Usable data type

REAL (if the input is REAL), LREAL

■ Program example

This program is designed to output the value of the trigonometric function sine obtained from the input variable “input1” to the output variable ”output1”.

LD program



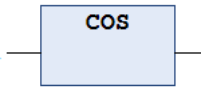
ST program

```
output1 0.479 := SIN(input1 0.5);
```

3.6.8 COS (Trigonometric Function Cosine)

This is a function that outputs the value of the trigonometric function cosine. The unit of the input argument is radian.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the value (unit: radian) from which to obtain the trigonometric function cosine.
Output	(Note 2)	Outputs the value of cosine of the input argument.

(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL

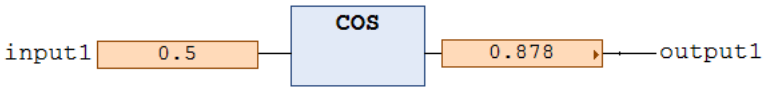
(Note 2) Usable data type

REAL (if the input is REAL), LREAL

■ Program example

This program is designed to output the value of the trigonometric function cosine obtained from the input variable “input1” to the output variable ”output1”.

LD program



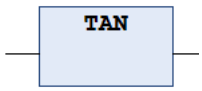
ST program

```
output1 0.878 := COS(input1 0.5);
```

3.6.9 TAN (Trigonometric Function Tangent)

This is a function that outputs the value of the trigonometric function tangent. The unit of the input argument is radian.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the value (unit: radian) from which to obtain the trigonometric function tangent.
Output	(Note 2)	Outputs the value of tangent of the input argument.

(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL

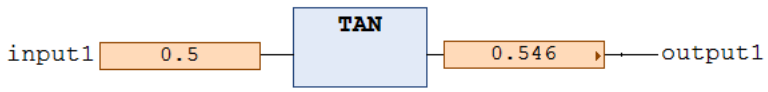
(Note 2) Usable data type

REAL (if the input is REAL), LREAL

■ Program example

This program is designed to output the value of the trigonometric function tangent obtained from the input variable “input1” to the output variable ”output1”.

LD program



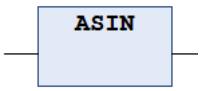
ST program

```
output1 0.546 := TAN(input1 0.5);
```

3.6.10 ASIN (Trigonometric Function Arc Sine)

This is a function that outputs the value of the trigonometric function arc sine. The unit of the input argument is radian.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the value (unit: radian) from which to obtain the trigonometric function arc sine.
Output	(Note 2)	Outputs the value of arc sine of the input argument.

(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL

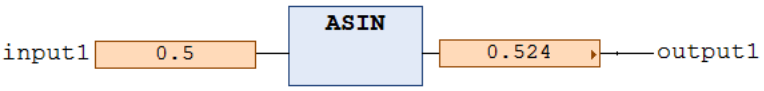
(Note 2) Usable data type

REAL (if the input is REAL), LREAL

■ Program example

This program is designed to output the value of the trigonometric function arc sine obtained from the input variable “input1” to the output variable ”output1”.

LD program



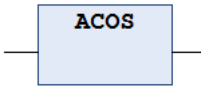
ST program

```
output1 0.524 := ASIN(input1 0.5);
```

3.6.11 ACOS (Trigonometric Function Arc Cosine)

This is a function that outputs the value of the trigonometric function arc cosine. The unit of the input argument is radian.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the value (unit: radian) from which to obtain the trigonometric function arc cosine.
Output	(Note 2)	Outputs the value of arc cosine of the input argument.

(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL

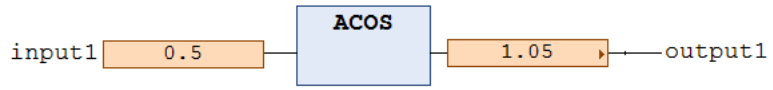
(Note 2) Usable data type

REAL (if the input is REAL), LREAL

■ Program example

This program is designed to output the value of the trigonometric function arc cosine obtained from the input variable “input1” to the output variable ”output1”.

LD program



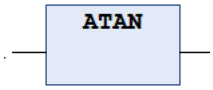
ST program

```
output1 1.05 := ACOS(input1 0.5);
```

3.6.12 ATAN (Trigonometric Function Arc Tangent)

This is a function that outputs the value of the trigonometric function arc tangent. The unit of the input argument is radian.

■ Icon



■ Parameter

Scope	Type	Description
Input	(Note 1)	Specifies the value (unit: radian) from which to obtain the trigonometric function arc tangent.
Output	(Note 2)	Outputs the value of arc tangent of the input argument.

(Note 1) Usable data type

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL

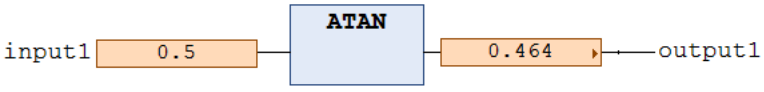
(Note 2) Usable data type

REAL (if the input is REAL), LREAL

■ Program example

This program is designed to output the value of the trigonometric function arc tangent obtained from the input variable “input1” to the output variable ”output1”.

LD program



ST program

```
output1 0.464 := ATAN(input1 0.5);
```

3.6.13 Triangular function operator constant

GM Programmer allows the use of the following constants.

Name	Value	Type	Description
SMC_PI	3.14159265358979	LREAL	Circumference ratio
SMC_FACTOR_DEG_TO_RAD	(SMC_PI/180)	LREAL	Convert angle (deg) to angle (rad)
SMC_FACTOR_RAD_TO_DEG	(180/SMC_PI)	LREAL	Convert angle (rad) to angle (deg)

3.7 Data Type Conversion Instructions

Data type conversion instructions can be used to convert the data type of a variable.

3.7.1 Type 1_TO_Type 2 (Type 1>Type 2 Conversion)

This is a function that converts the data type of the input argument Type 1 to another data type Type 2. Conversion from a larger size data type to a smaller size data type is not performed automatically. It is necessary to convert the data type using this instruction.

Parameter

Value	BOOL/BYTE/WORD/DWORD/LWORD/SINT/USINT INT/UINT/DINT/UDINT/LINT/ULINT/REAL/LREAL
Time	TIME/LTIME/TIME_OF_DAY/DATE/DATE_AND_TIME
Character string	STRING/WSTRING

Input ("Type 1")

Specifies the variable required to be converted

Output ("Type 2")

Outputs the converted variable

■ **Numerical value to numerical value type conversion**

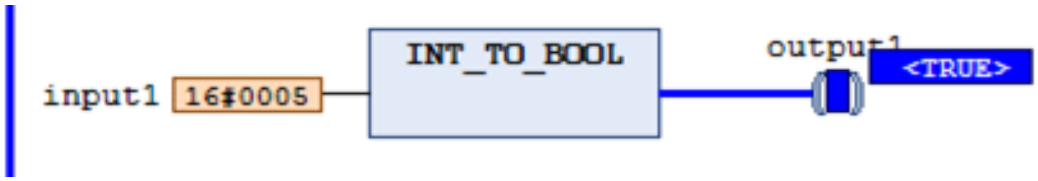
Main conversion examples

Numerical value ⇒ Numerical value	Conversion example		Description
	Input	Output	
INT_TO_BOOL	5	TRUE	If other than 0, outputs TRUE.
UINT_TO_SINT	300(16#012C)	44(16#2C)	Outputs lower eight bits out of the 16 bits of UINT.
REAL_TO_INT	3.5	4	Outputs data after rounding decimals to the nearest whole number.

■ **Program example**

INT_TO_BOOL

LD program

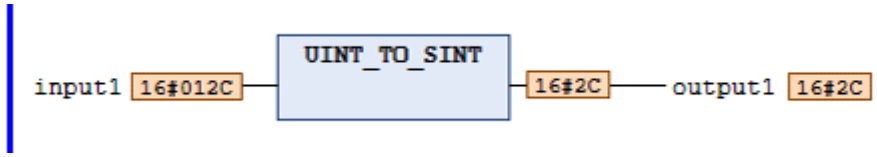


ST program

```
output1 TRUE :=INT_TO_BOOL(input1 16#0005 );
```

UINT_TO_SINT

LD program

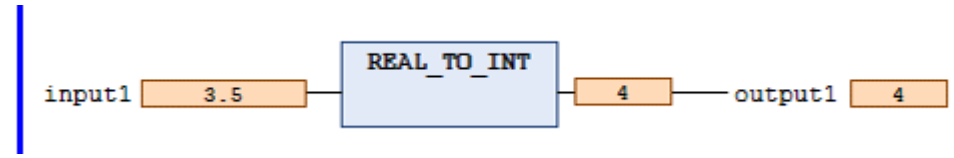


ST program

```
output1 16#2C :=UINT_TO_SINT(input1 16#012C );
```

REAL_TO_INT

LD program



ST program

```
output1 4 :=REAL_TO_INT(input1 3.5 );
```

■ Numerical value to time / Time to numerical value type conversion

Main conversion examples

Numerical value ⇒ Time	Conversion example		Description
	Input	Output	
UDINT_TO_TIME	16#098D 11B2	T#1d20h30m40s50ms	Outputs the TIME constant in the UDINT type.
UDINT_TO_TOD	16#0466 B774	TOD#20:30:40.500	Outputs the TOD constant in the UDINT type.
UDINT_TO_DATE	16#386D 4380	D#2000-1-1	Outputs the DATE constant in the UDINT type.
UDINT_TO_DT	16#386E 63F0	DT#2000-1-1-20:30:40	Outputs the DT constant in the UDINT type.
ULINT_TO_LTIME	16#0000 91BC CB43 3B26	LTIME#1d20h30m40s50ms60us70ns	Outputs the LTIME constant in the LTIME type.

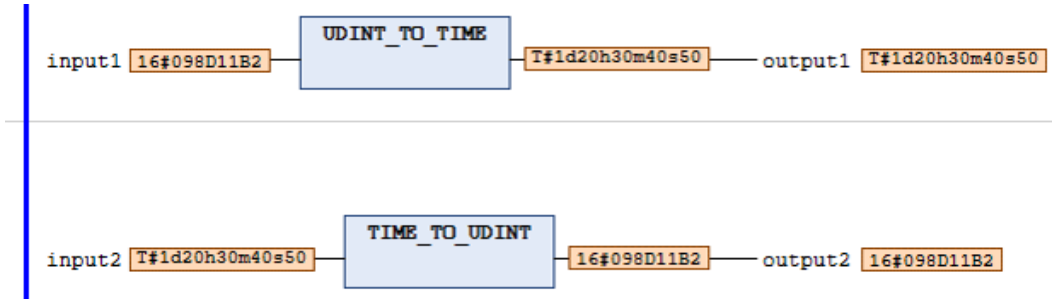
Time ⇒ Numerical value	Conversion example		Description
	Input	Output	
TIME_TO_UDINT	T#1d20h30m40s50ms	16#098D 11B2	Outputs the milliseconds from 0d0h0m0s.
TOD_TO_UDINT	TOD#20:30:40.500	16#0466 B774	Outputs the milliseconds from 00:00:00.

Time ⇒ Numerical value	Conversion example		Description
	Input	Output	
DATE_TO_UDINT	D#2000-1-1	16#386D 4380	Outputs the seconds from 1970-1-1.
DT_TO_UDINT	DT#2000-1-1-20:30:40	16#386E 63F0	Outputs the seconds from 1970-1-1-0:0:0.
LTIME_TO_ULINT	LTIME#1d20h30m40s50ms60us70ns	16#0000 91BC CB43 3B26	Outputs the nanoseconds from 0d0h0m0s0ms0us.

■ Program example

UDINT_TO_TIME/TIME_TO_UDINT

LD program

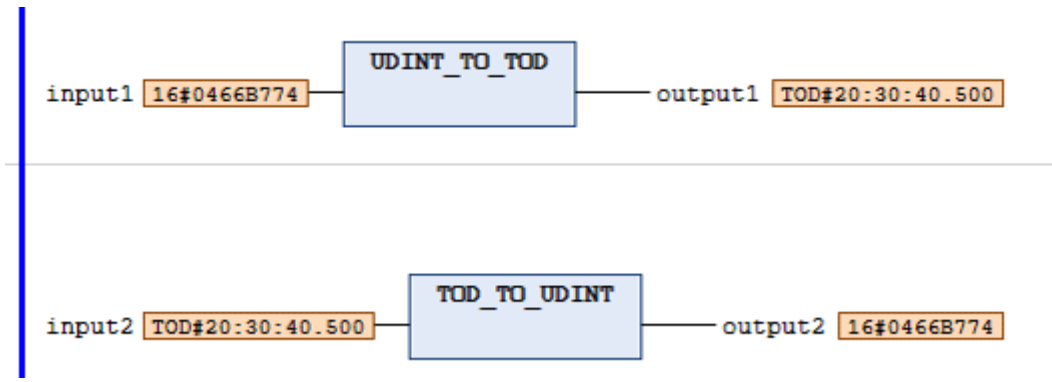


ST program

```
output1 T#1d20h30m40s50ms :=UDINT_TO_TIME(input1 16#098D11B2 );
output2 16#098D11B2 :=TIME_TO_UDINT(input2 T#1d20h30m40s50ms );
```

UDINT_TO_TOD/TOD_TO_UDINT

LD program

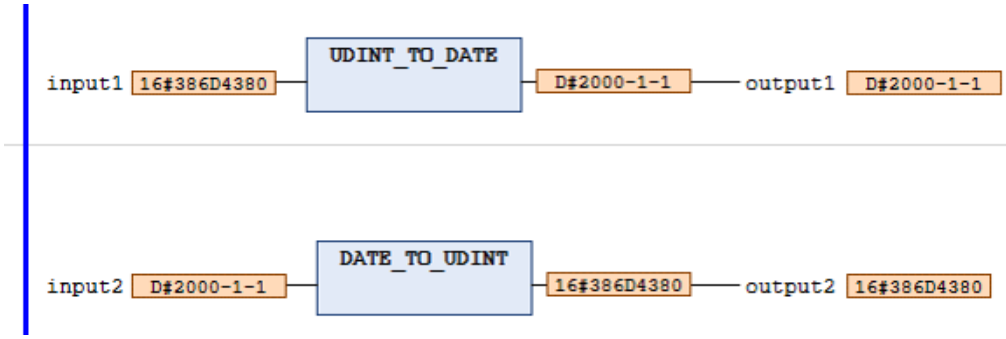


ST program

```
output1 TOD#20:30:40.500 :=UDINT_TO_TOD(input1 16#0466B774 );
output2 16#0466B774 :=TOD_TO_UDINT(input2 TOD#20:30:40.500 );
```

UDINT_TO_DATE/DATE_TO_UDINT

LD program

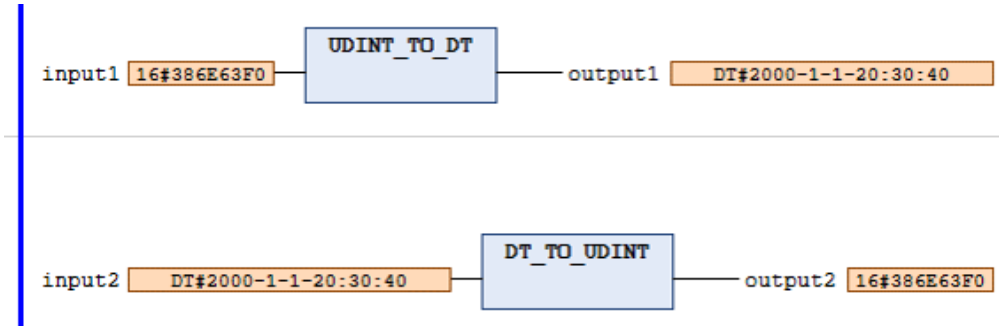


ST program

```
output1 D#2000-1-1 :=UDINT_TO_DATE(input1 16#386D4380);
output2 16#386D4380 :=DATE_TO_UDINT(input2 D#2000-1-1);
```

UDINT_TO_DT/DT_TO_UDINT

LD program

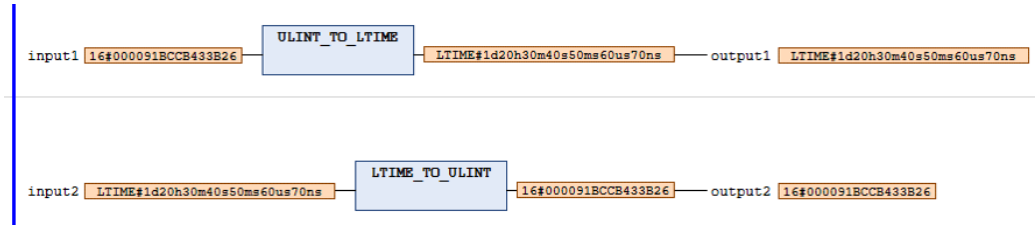


ST program

```
output1 DT#2000-1-1-20:30:40 :=UDINT_TO_DT(input1 16#386E63F0);
output2 16#386E63F0 :=DT_TO_UDINT(input2 DT#2000-1-1-20:30:40);
```

ULINT_TO_LTIME/LTIME_TO_ULINT

LD program



ST program

```
output1 LTIME#1d20h30m40s50ms60us70ns :=ULINT_TO_LTIME(input1 16#000091BCCB433B26 );
output2 16#000091BCCB433B26 :=LTIME_TO_ULINT(input2 LTIME#1d20h30m40s50ms60us70ns );
```

■ Numerical value to character string / Character string to numerical value type conversion

Main conversion examples

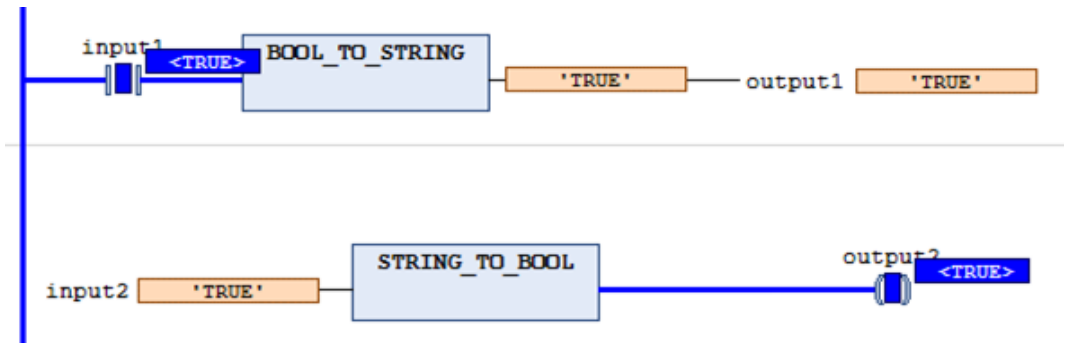
Numerical value ⇒ Character string	Conversion example		Description
	Input	Output	
BOOL_TO_STRING	TRUE	'TRUE'	Outputs 'TRUE' converted from TRUE / Outputs 'FALSE' converted from FALSE.
WORD_TO_STRING	16#3039(10#12345)	'12345'	Outputs the character string '12345' converted from the input value.
INT_TO_WSTRING	16#3039(10#12345)	"12345"	Outputs the character string "12345" converted from the input value.

Character string ⇒ Numerical value	Conversion example		Description
	Input	Output	
STRING_TO_BOOL	'TRUE'	TRUE	Outputs TRUE only when the character string 'TRUE' is input.
STRING_TO_WORD	'12345'	16#3039(10#12345)	Outputs a numerical value converted from the character string of the numerical value.
WSTRING_TO_INT	"12345"	16#3039(10#12345)	Outputs a numerical value converted from the character string of the numerical value.

■ Program example

BOOL_TO_STRING/STRING_TO_BOOL

LD program

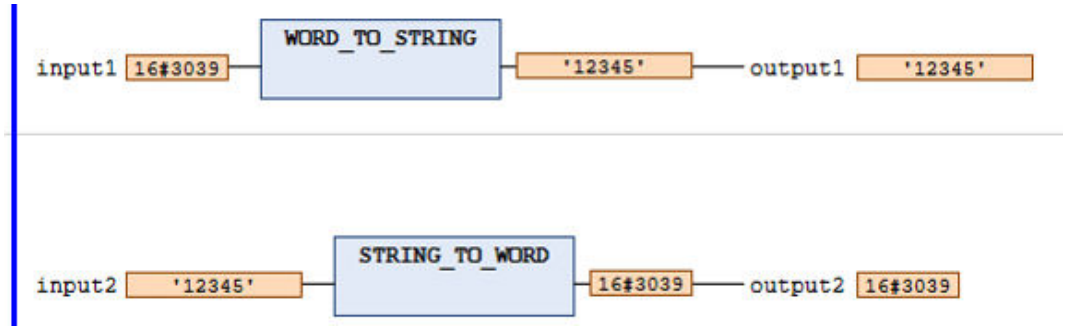


ST program

```
output1 'TRUE' :=BOOL_TO_STRING(input1 TRUE );
output2 TRUE :=STRING_TO_BOOL(input2 'TRUE' );
```

WORD_TO_STRING/STRING_TO_WORD

LD program

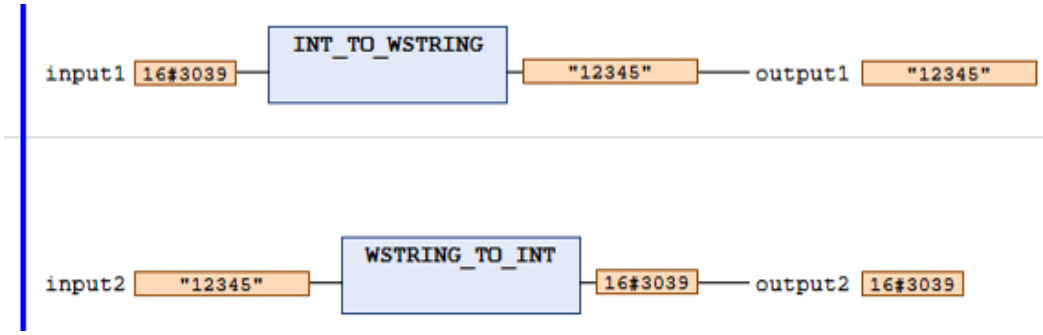


ST program

```
output1 '12345' :=WORD_TO_STRING(input1 16#3039 );
output2 16#3039 :=STRING_TO_WORD(input2 '12345' );
```

INT_TO_WSTRING/WSTRING_TO_INT

LD program



ST program

```
output1 "12345" :=INT_TO_WSTRING(input1 16#3039 );
output2 16#3039 :=WSTRING_TO_INT(input2 "12345" );
```

■ Time to character string / Character string to time type conversion

Main conversion examples

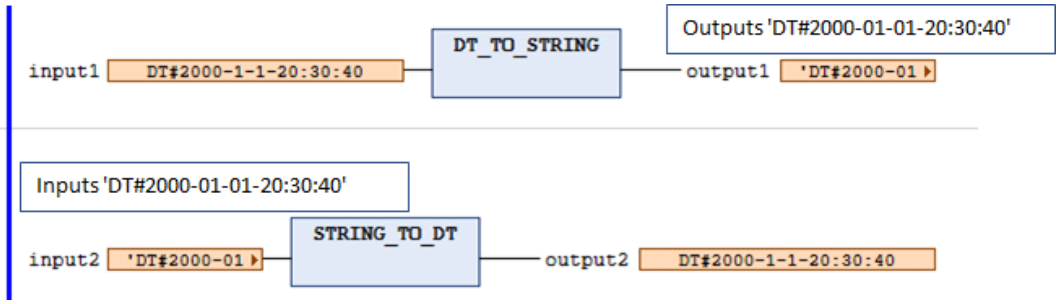
Time ⇒ Character string	Conversion example		Description
	Input	Output	
DT_TO_STRING	DT#2000-1-1-20:30:40	'DT#2000-01-01-20:30:40'	Outputs the DT constant in the STRING type.
TOD_TO_WSTRING	TOD#20:30:40.500	"TOD#20:30:40.500"	Outputs the TOD constant in the WSTRING type.

Character string ⇒ Time	Conversion example		Description
	Input	Output	
STRING_TO_DT	'DT#2000-01-01-20:30:40'	DT#2000-1-1-20:30:40	Outputs time converted from the character string of the time.
WSTRING_TO_TOD	"TOD#20:30:40.500"	TOD#20:30:40.500	Outputs time converted from the character string of the time.

■ **Program example**

DT_TO_STRING/STRING_TO_DT

LD program

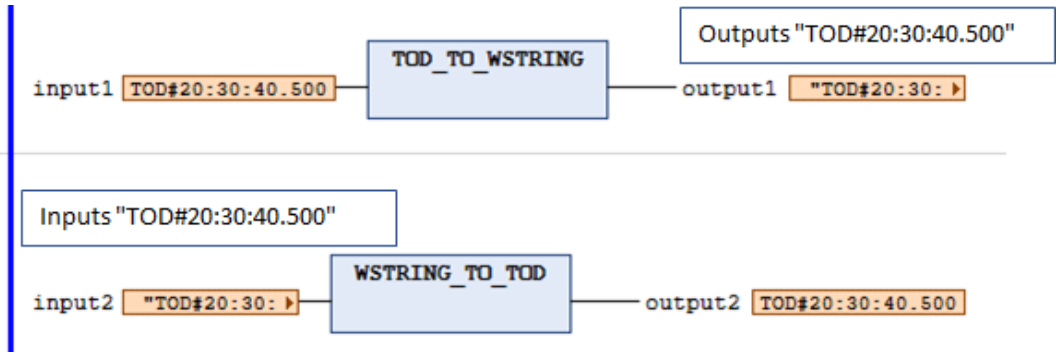


ST program

```
output1 'DT#2000-01 ▸ :=DT_TO_STRING(input1 DT#2000-1-1-20:30:40 );
output2 DT#2000-1-1-20:30:40 :=STRING_TO_DT(input2 'DT#2000-01 ▸ );
```

TOD_TO_WSTRING/WSTRING_TO_TOD

LD program



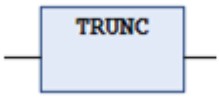
ST program

```
output1 "TOD#20:30: ▸ :=TOD_TO_WSTRING(input1 TOD#20:30:40.500 );
output2 TOD#20:30:40.500 :=WSTRING_TO_TOD(input2 "TOD#20:30: ▸ );
```

3.7.2 TRUNC (Real Number to DINT Conversion)

This is a function that converts a real number type input to a DINT type.

■ Icon



■ Parameter

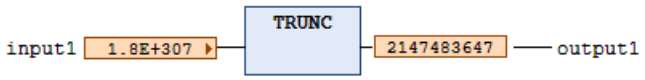
Scope	Type	Description
Input	REAL, LREAL	Real number type value
Output	DINT	Outputs the value converted to the DINT type from the input argument.

■ Program example

This program is designed to convert the LREAL type input variable “input1” to the DINT type output variable “output1” and output the converted data.

Input1 := 1.7976931348623157E+307;

LD program



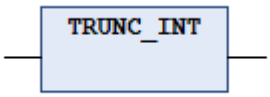
ST program

```
output1 2147483647 := TRUNC(input1 1.8E+307 );
```

3.7.3 TRUNC_INT (Real Number to INT Conversion)

This is a function that converts a real number type input to an INT type.

■ Icon



■ Parameter

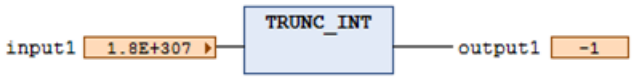
Scope	Type	Description
Input	REAL, LREAL	Real number type value
Output	INT	Outputs the value converted to the INT type from the input argument.

■ Program example

This program is designed to convert the LREAL type input variable “input1” to the INT type output variable “output1” and output the converted data.

Input1 := 1.7976931348623157E+307;

LD program



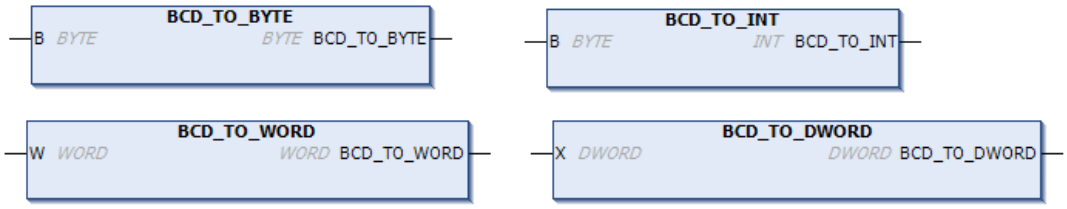
ST program

```
output1 -1 := TRUNC_INT(input1 1.8E+307 );
```

3.7.4 BCD_TO_** (BCD to Binary Conversion)

This is a function that converts an input in BCD format to a binary code of BYTE type, INT type, WORD type, or DWORD type.

■ Icon



■ Parameter

BCD_TO_BYTE

Scope	Name	Type	Description
Input	B	BYTE	The BCD code value to be converted
Output	BCD_TO_BYTE	BYTE	Outputs the value converted to a binary code from the input argument.

BCD_TO_INT

Scope	Name	Type	Description
Input	B	BYTE	The BCD code value to be converted
Output	BCD_TO_INT	INT	Outputs the value converted to a binary code from the input argument. Outputs 10#-1 when a value outside the effective range is input to the input B.

BCD_TO_WORD

Scope	Name	Type	Description
Input	W	WORD	The BCD code value to be converted
Output	BCD_TO_WORD	WORD	Outputs the value converted to a binary code from the input argument.

BCD_TO_DWORD

Scope	Name	Type	Description
Input	X	DWORD	The BCD code value to be converted
Output	BCD_TO_DWORD	DWORD	Outputs the value converted to a binary code from the input argument.

■ Program example

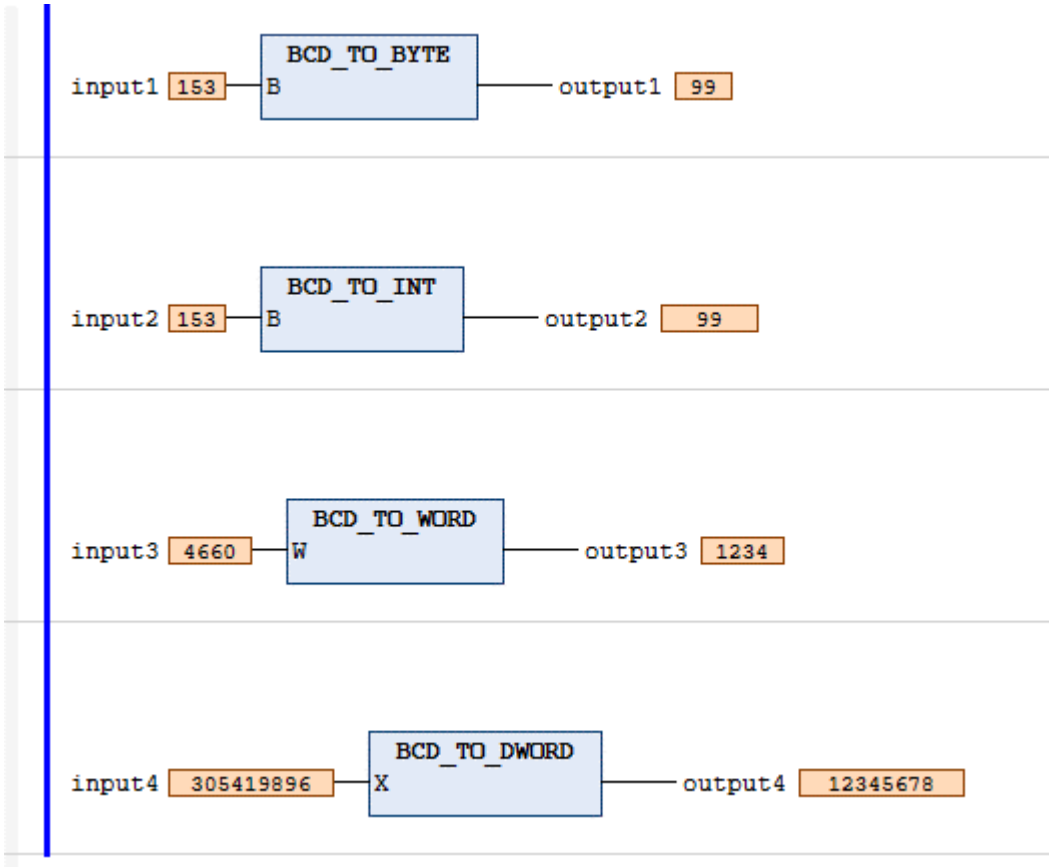
This program is designed to convert the BYTE type input variable “input1” to the BYTE type output variable “output1” and output the converted data.

This program is designed to convert the BYTE type input variable “input2” to the INT type output variable “output2” and output the converted data.

This program is designed to convert the WORD type input variable “input3” to the WORD type output variable “output3” and output the converted data.

This program is designed to convert the DWORD type input variable “input4” to the DWORD type output variable “output4” and output the converted data.

LD program



ST program

```
output1 99  := BCD_TO_BYTE(input1 153 );

output2  99  := BCD_TO_INT(input2 153 );

output3 1234 := BCD_TO_WORD(input3 4660 );

output4  12345678  := BCD_TO_DWORD(input4  305419896  );
```

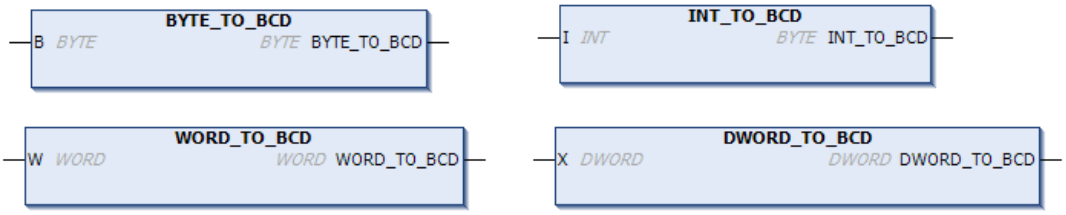
Note

- Do not input a value that is not a BCD array (including A, B, C, D, E, or F in hexadecimal notation).

3.7.5 **_TO_BCD (Binary to BCD Conversion)

This is a function that converts a binary code input of BYTE type, INT type, WORD type, or DWORD type to a BCD format value.

■ Icon



■ Parameter

BYTE_TO_BCD

Scope	Name	Type	Description
Input	B	BYTE	The binary code value to be converted. Effective range: 10#0 to 99
Output	BYTE_TO_BCD	BYTE	Outputs the value converted to the BCD code from the input argument.

INT_TO_BCD

Scope	Name	Type	Description
Input	I	INT	The binary code value to be converted. Effective range: 10#0 to 99
Output	INT_TO_BCD	BYTE	Outputs the value converted to the BCD code from the input argument. Outputs 16#FF when a value outside the effective range is input to the input I.

WORD_TO_BCD

Scope	Name	Type	Description
Input	W	WORD	The binary code value to be converted. Effective range: 10#0 to 9999
Output	WORD_TO_BCD	WORD	Outputs the value converted to the BCD code from the input argument.

DWORD_TO_BCD

Scope	Name	Type	Description
Input	X	DWORD	The binary code value to be converted. Effective range: 10#0 to 99999999
Output	DWORD_TO_BCD	DWORD	Outputs the value converted to the BCD code from the input argument.

■ Program example

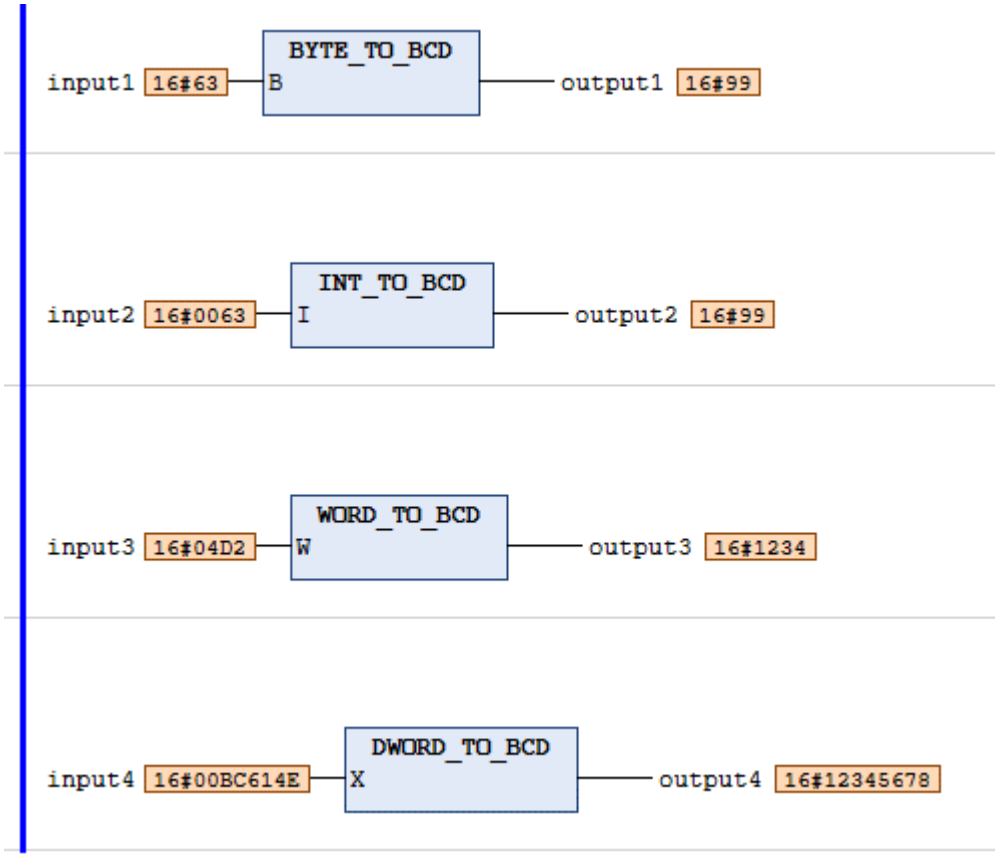
This program is designed to convert the BYTE type input variable “input1” to the BYTE type output variable “output1” and output the converted data.

This program is designed to convert the BYTE type input variable “input2” to the INT type output variable “output2” and output the converted data.

This program is designed to convert the WORD type input variable “input3” to the WORD type output variable “output3” and output the converted data.

This program is designed to convert the DWORD type input variable “input4” to the DWORD type output variable “output4” and output the converted data.

LD program



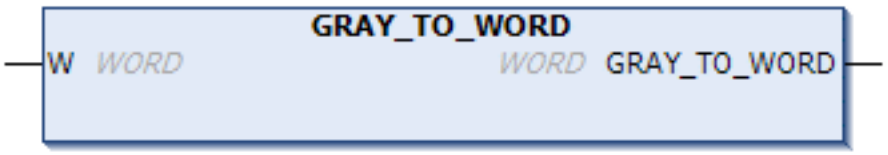
ST program

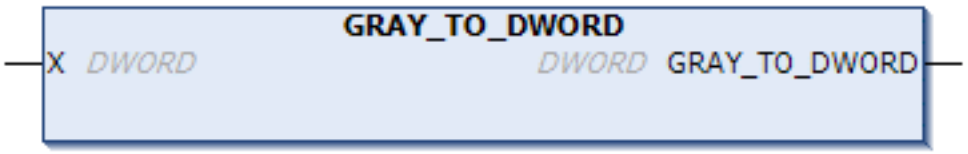
```
output1 16#99 := BYTE_TO_BCD(input1 16#63 );

output2 16#99 := INT_TO_BCD(input2 16#0063 );

output3 16#1234 := WORD_TO_BCD(input3 16#04D2 );

output4 16#12345678 := DWORD_TO_BCD(input4 16#00BC614E );
```

3.7.6 GRAY_TO_** (Gray Code to Binary Conversion)

This is a function that converts a Gray code input to a binary code of BYTE type, WORD type, or DWORD type.

■ **Icon**





■ **Parameter**

GRAY_TO_BYTE

Scope	Name	Type	Description
Input	B	BYTE	The BYTE type Gray code value to be converted
Output	GRAY_TO_BYTE	BYTE	A value converted to a BYTE type binary code from the input argument

GRAY_TO_WORD

Scope	Name	Type	Description
Input	W	WORD	The WORD-type Gray code value to be converted
Output	GRAY_TO_WORD	WORD	A value converted to a WORD-type binary code from the input argument

GRAY_TO_DWORD

Scope	Name	Type	Description
Input	X	DWORD	The DWORD type Gray code value to be converted
Output	GRAY_TO_DWORD	DWORD	A value converted to a DWORD type binary code from the input argument

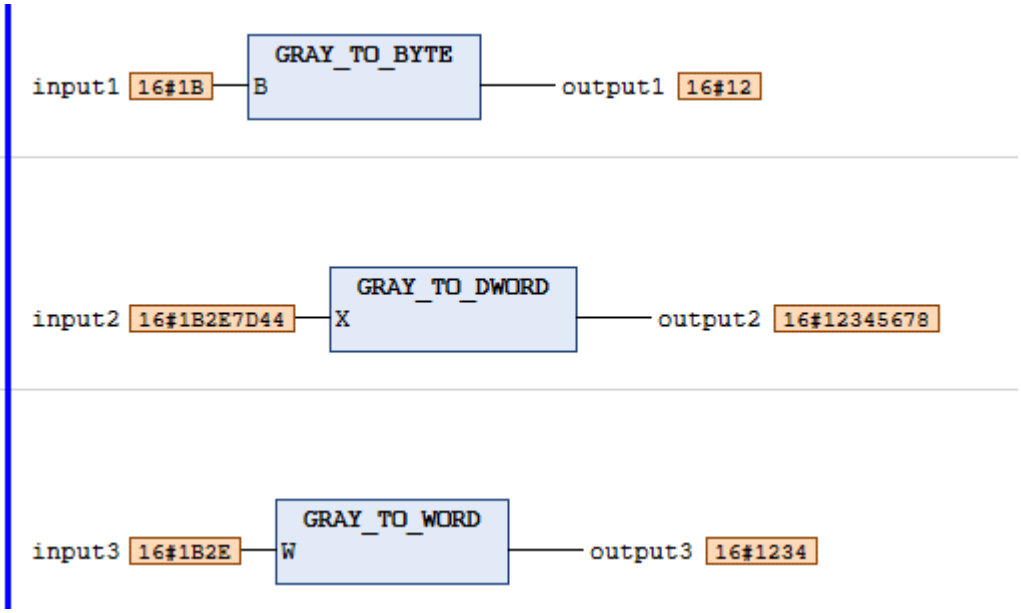
■ Program example

This program is designed to convert the BYTE type Gray code input variable “input1” to the BYTE type binary code output variable “output1” and output the converted data.

This program is designed to convert the DWORD type Gray code input variable “input2” to the DWORD type binary code output variable “output2” and output the converted data.

This program is designed to convert the WORD type Gray code input variable “input3” to the WORD type binary code output variable “output3” and output the converted data.

LD program



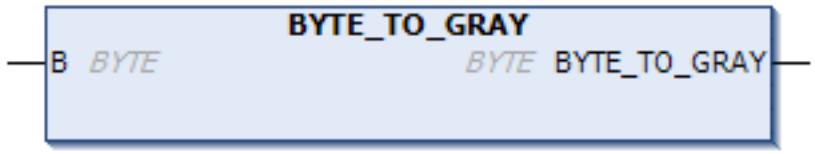
ST program

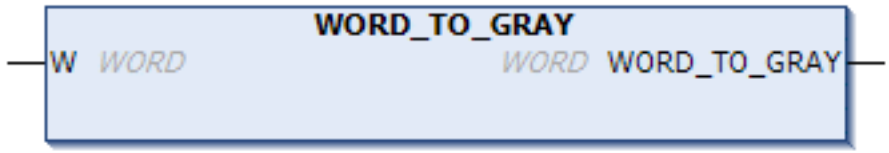
```
output1 16#12 :=GRAY_TO_BYTE(input1 16#1B );

output2 16#12345678 :=GRAY_TO_DWORD(input2 16#1B2E7D44 );

output3 16#1234 :=GRAY_TO_WORD(input3 16#1B2E );
```

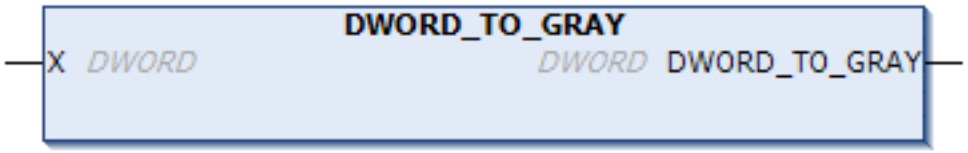
3.7.7 **_TO_GRAY (Binary to Gray Code Conversion)

This is a function that converts a binary code input of BYTE type, WORD type, or DWORD type to a Gray code.

■ Icon







■ Parameter

BYTE_TO_GRAY

Scope	Name	Type	Description
Input	B	BYTE	The BYTE type binary code value to be converted
Output	BYTE_TO_GRAY	BYTE	A value converted to a BYTE type Gray code from the input argument

WORD_TO_GRAY

Scope	Name	Type	Description
Input	W	WORD	The WORD type binary code value to be converted
Output	WORD_TO_GRAY	WORD	A value converted to a WORD type Gray code from the input argument

DWORD_TO_GRAY

Scope	Name	Type	Description
Input	X	DWORD	The DWORD type binary code value to be converted
Output	DWORD_TO_GRAY	DWORD	A value converted to a DWORD type Gray code from the input argument

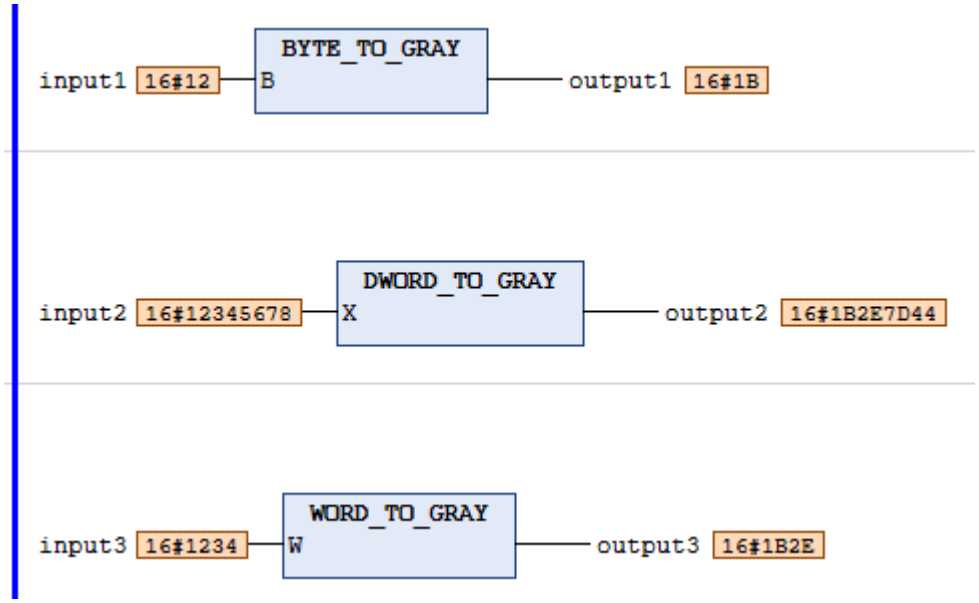
■ Program example

This program is designed to convert the BYTE type binary code input variable “input1” to the BYTE type Gray code output variable “output1” and output the converted data.

This program is designed to convert the DWORD type binary code input variable “input2” to the DWORD type Gray code output variable “output2” and output the converted data.

This program is designed to convert the WORD type binary code input variable “input3” to the WORD type Gray code output variable “output3” and output the converted data.

LD program



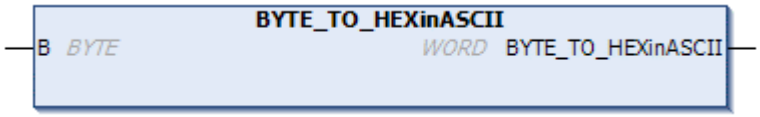
ST program

```
output1 16#1B :=BYTE_TO_GRAY(input1 16#12 );

output2 16#1B2E7D44 :=DWORD_TO_GRAY(input2 16#12345678 );

output3 16#1B2E :=WORD_TO_GRAY(input3 16#1234 );
```

3.7.8 BYTE_TO_HEXinASCII (Binary to ASCII Conversion)

This is a function that converts a one-byte hexadecimal binary-coded value to a one-word ASCII code.

■ Icon



■ Parameter

BYTE_TO_HEXinASCII

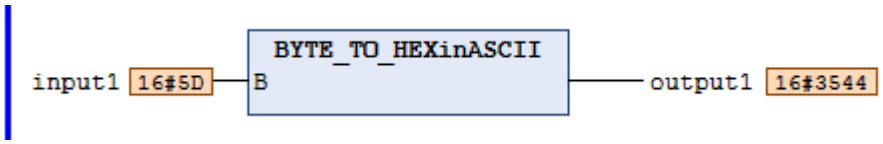
Scope	Name	Type	Description
Input	B	BYTE	The binary code value to be converted.
Output	BYTE_TO_HEXinASCII	WORD	A value converted to an ASCII code from the input argument

■ Program example

This program is designed to convert the BYTE type input variable “input1” to the WORD type output variable “output1” and output the converted data.

input1 := 16#5D

LD program



ST program

```
output1 16#3544  :=  BYTE_TO_HEXinASCII (input1 16#5D );
```

Note

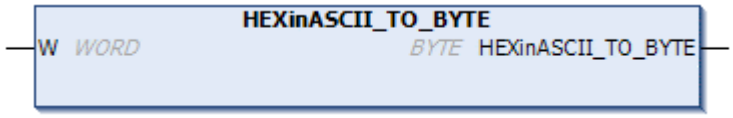
- **Convertible ASCII codes**

ASCII	Hexadecimal
0	0x30
1	0x31
2	0x32
3	0x33
4	0x34
5	0x35
6	0x36
7	0x37
8	0x38
9	0x39
A	0x41
B	0x42
C	0x43
D	0x44
E	0x45
F	0x46

3.7.9 HEXinASCII_TO_BYTE (ASCII to Binary Conversion)

This is a function that converts a one-word ASCII code to a one-byte hexadecimal binary-coded value.

■ Icon



■ Parameter

HEXinASCII_TO_BYTE

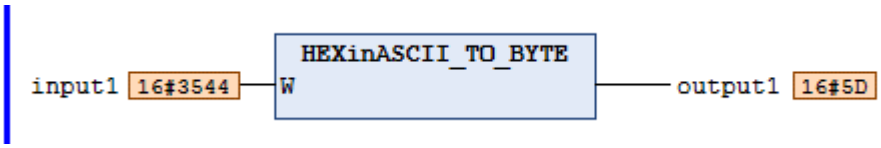
Scope	Name	Type	Description
Input	W	WORD	The ASCII code value to be converted
Output	HEXinASCII_TO_BYTE	BYTE	A value converted to a binary code from the input argument

■ Program example

This program is designed to convert the WORD type input variable “input1” to the BYTE type output variable “output1” and output the converted data.

input1 := 16#3544

LD program



ST program

```
output1 16#5D := HEXinASCII_TO_BYTE (input1 16#3544 );
```

Note

- **Convertible ASCII codes**

  Inputs other than those shown below cause 0 to be output.

Hexadecimal	ASCII
0x30	0
0x31	1
0x32	2
0x33	3
0x34	4
0x35	5
0x36	6
0x37	7
0x38	8
0x39	9
0x41	A
0x42	B
0x43	C
0x44	D
0x45	E
0x46	F

3.7.10 MEM.Decode (4BYTE to DWORD Conversion)

This is a function that decodes data in units of byte to data in units of DWORD. The number of bytes that can be decoded is a multiple of 4 within the effective range 10#4 to 10#65532.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	pSource	POINTER TO BYTE	Start pointer to data in units of byte
Input	pDestination	POINTER TO DWORD	Start pointer to data in units of DWORD
Input	uiNumberOfBytes	UINT	Number of bytes to decode Effective range: 10#4 to 10#65532
Output	Decode	BOOL	Always outputs FALSE

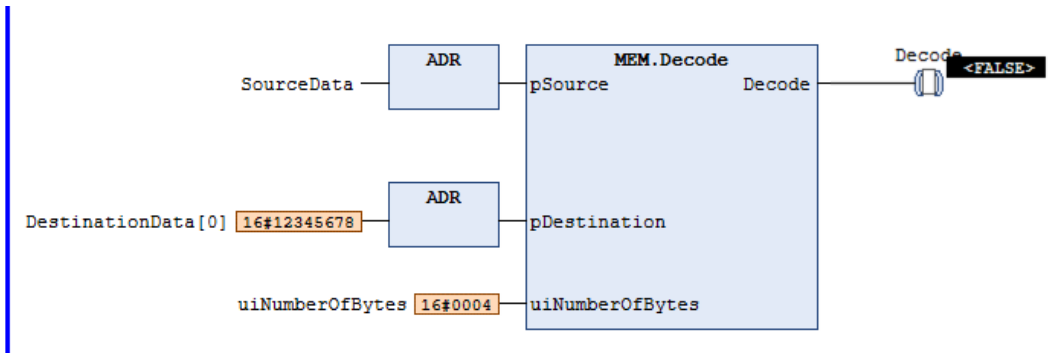
■ Program example

This program is designed to decode four-byte (uiNumberOfBytes) data of an input variable (SourceData[16#78,16#56,16#34,16#12]) into one-dword data (DestinationData[16#12345678]).

SourceData : ARRAY [0..3] OF BYTE := [16#78,16#56,16#34,16#12] (decode source data)

uiNumberOfBytes := 10#4 (16#4)

LD program



ST program

```
SourceData[3] 16#12;
SourceData[2] 16#34;
SourceData[1] 16#56;
SourceData[0] 16#78;

Decode FALSE := MEM.Decode(ADR(SourceData), ADR(DestinationData[0] 16#12345678 ),4);
```

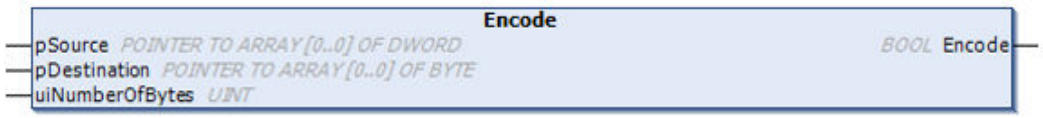
Note

- Use a multiple of 4 for the number of bytes to decode (uiNumberOfBytes). The bytes of data other than a multiple of 4 cannot be decoded and 0 is output.
- Do not set 0 (NULL) in the start pointer to decode source data (pSource) and the start pointer to decode destination data (pDestination). If set to NULL, an exception error occurs.

3.7.11 MEM.Encode (DWORD to 4BYTE Conversion)

This is a function that encodes data in units of DWORD into data in units of bytes. The number of bytes that can be encoded is a multiple of 4 within the effective range 10#4 to 10#65532.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	pSource	POINTER TO DWORD	Start pointer of data in units of DWORD
Input	pDestination	POINTER TO BYTE	Start pointer to data in units of byte
Input	uiNumberOfBytes	UINT	Number of bytes to encode Effective range: 10#4 to 10#65532
Output	Encode	BOOL	Number of bytes to encode Effective range: 10#4 to 10#65532

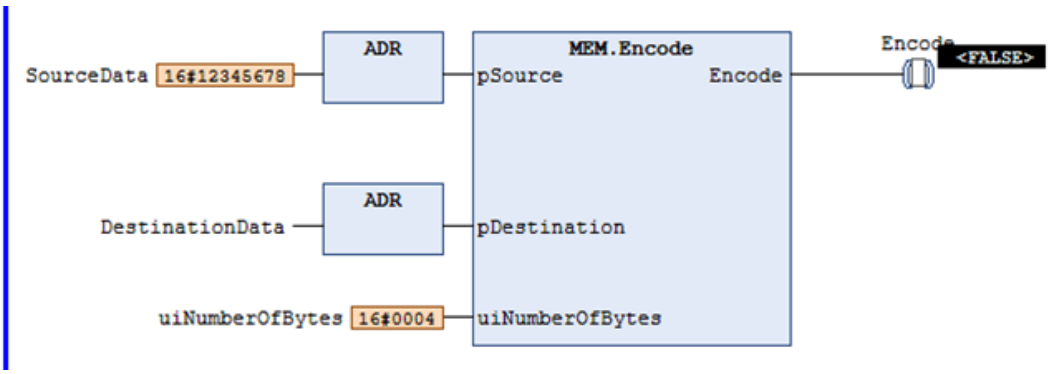
■ Program example

This program is designed to encode four-byte (uiNumberOfBytes) data of an input variable (SourceData[16#12345678]) into four-byte data (DestinationData[16#78,16#56,16#34,16#12]).

SourceData := 16#12345678

uiNumberOfBytes := 10#4 (16#4)

LD program



ST program

```
Encode FALSE := MEM.Encode(ADR(SourceData 16#12345678 ), ADR(DestinationData),4);

        DestinationData[3] 16#12 ;
        DestinationData[2] 16#34 ;
        DestinationData[1] 16#56 ;
        DestinationData[0] 16#78 ;
```

Note

- Use a multiple of 4 for the number of bytes to encode (uiNumberOfBytes). The bytes of data other than a multiple of 4 cannot be encoded and 0 is output.
- Do not set 0 (NULL) in the start pointer to encode source data (pSource) and the start pointer to encode destination data (pDestination). If set to NULL, an exception error occurs.

3.7.12 MEM.PackArrayOfBoolToArrayOfByte (BOOL Array to BYTE Conversion)

This is a function that packs a BOOL type array into an array in bytes and copies a specified bit size data. The function returns the number of bytes required for coping. The maximum copyable size is 65535 bits (approx. 8192 bytes).

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	paxSource	POINTER TO BYTE	Starting address of BOOL type array data
Input	pabyDestination	POINTER TO BYTE	Starting address of Byte type data
Input	uiNumberOfBits	UINT	Number of bits to copy Effective range: 10#1 to 65535
Output	PackArrayOfBoolToArrayOfByte	UINT	Outputs the number of bytes required for coping

■ Program example

This program is designed to pack a 24-bit amount (uiNumberOfBits) of BOOL type copy source data (xbit) in bytes and copy the packed data to the copy destination (ArrayBlock).

The program returns the number of bytes required for coping.

ArrayBlock : ARRAY [0..4] OF BYTE := [5(0)] (copy destination data: default value)

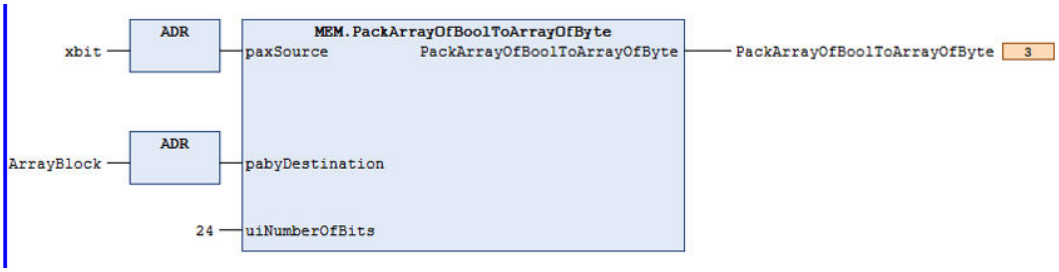
xbit[7:0] = 2#00010010[16#12]

xbit[15:8] = 2#00110100[16#34]

xbit[23:16] = 2#01010110[16#56]

uiNumberOfBits := 10#24

LD program



ST program

```
xbit[7] FALSE :=FALSE;  xbit[6] FALSE :=FALSE;  xbit[5] FALSE :=FALSE;  xbit[4] TRUE :=TRUE;
xbit[3] FALSE :=FALSE;  xbit[2] FALSE :=FALSE; xbit[1] TRUE :=TRUE;    xbit[0] FALSE :=FALSE;
xbit[15] FALSE :=FALSE; xbit[14] FALSE :=FALSE; xbit[13] TRUE :=TRUE;  xbit[12] TRUE :=TRUE;
xbit[11] FALSE :=FALSE; xbit[10] TRUE :=TRUE;  xbit[9] FALSE :=FALSE;  xbit[8] FALSE :=FALSE;
xbit[23] FALSE :=FALSE; xbit[22] TRUE :=TRUE;  xbit[21] FALSE :=FALSE; xbit[20] TRUE :=TRUE;
xbit[19] FALSE :=FALSE; xbit[18] TRUE :=TRUE; xbit[17] TRUE :=TRUE;  xbit[16] FALSE :=FALSE;

PackArrayOfBoolToArrayOfByte 16#0003  := MEM.PackArrayOfBoolToArrayOfByte(ADR(xbit), ADR(ArrayBlock), 24);

ArrayBlock[0] 16#12 ;
ArrayBlock[1] 16#34 ;
ArrayBlock[2] 16#56 ;
```

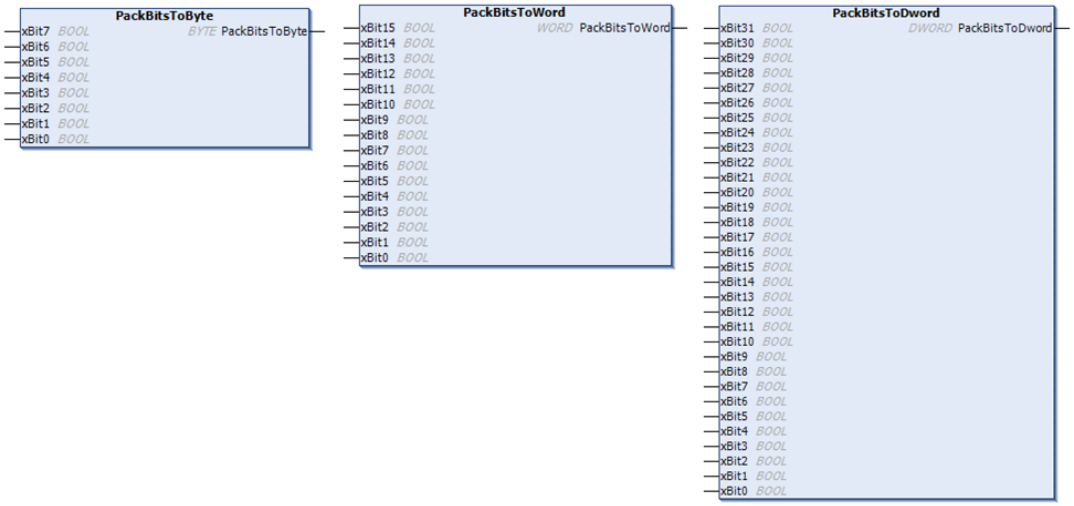
Note

- If the number of bits to be copied uiNumberOfBits = 0, the copying will not be carried out and the return value of the function will be PackArrayOfBoolToArrayOfByte = 0.

3.7.13 MEM.PackBitsTo**(Bit Data to BYTE/WORD/DWORD Conversion)

This is a function that packs input BOOL type data and outputs a BYTE, a WORD, or a DWORD.

■ Icon



■ Parameter

MEM.PackBitsToByte

Scope	Name	Type	Description
Input	xBit0 to xBit7	BOOL	Bits to be packed Bit0 to Bit7
Output	PackBitsToByte	BYTE	A value of the packed input

MEM.PackBitsToWord

Scope	Name	Type	Description
Input	xBit0~xBit15	BOOL	Bits to be packed Bit0 to Bit15
Output	PackBitsToWord	WORD	A value of the packed input

MEM.PackBitsToDword

Scope	Name	Type	Description
Input	xBit0~xBit31	BOOL	Bits to be packed Bit0 to Bit31
Output	PackBitsToDword	DWORD	A value of the packed input

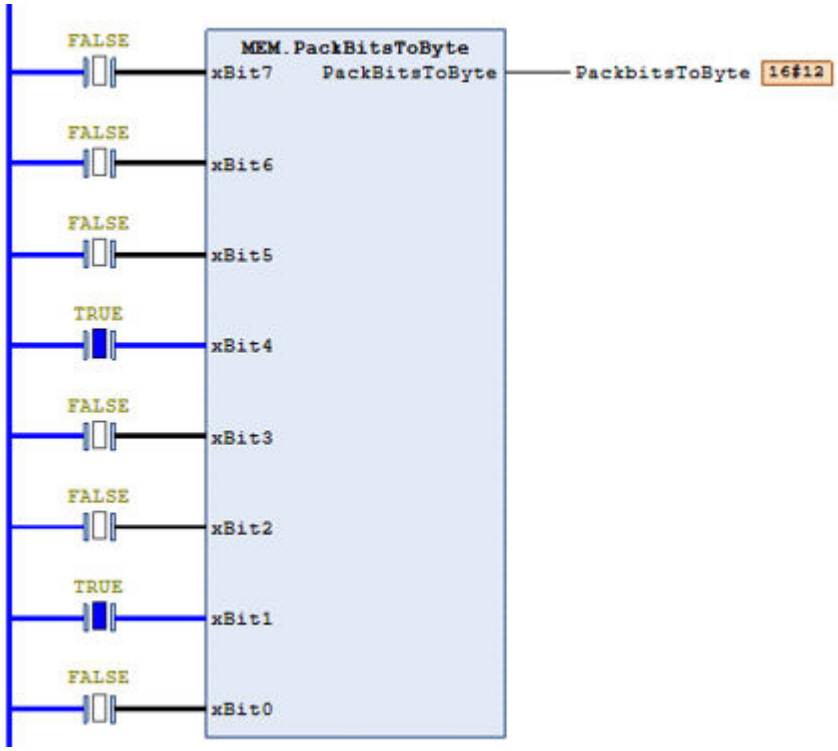
■ Program example 1

This program is designed to pack the BOOL type input variables xBit and output the packed data to the BYTE type output variable PackBitsToByte.

xBit4,xBit1 :=TRUE

Others :=FALSE

LD program



ST program

```
PackbitsToByte 16#12 :=MEM.PackbitsToByte (xbit[7] FALSE ,xbit[6] FALSE ,xbit[5] FALSE ,xbit[4] TRUE ,
                                           xbit[3] FALSE ,xbit[2] FALSE ,xbit[1] TRUE ,xbit[0] FALSE );
```

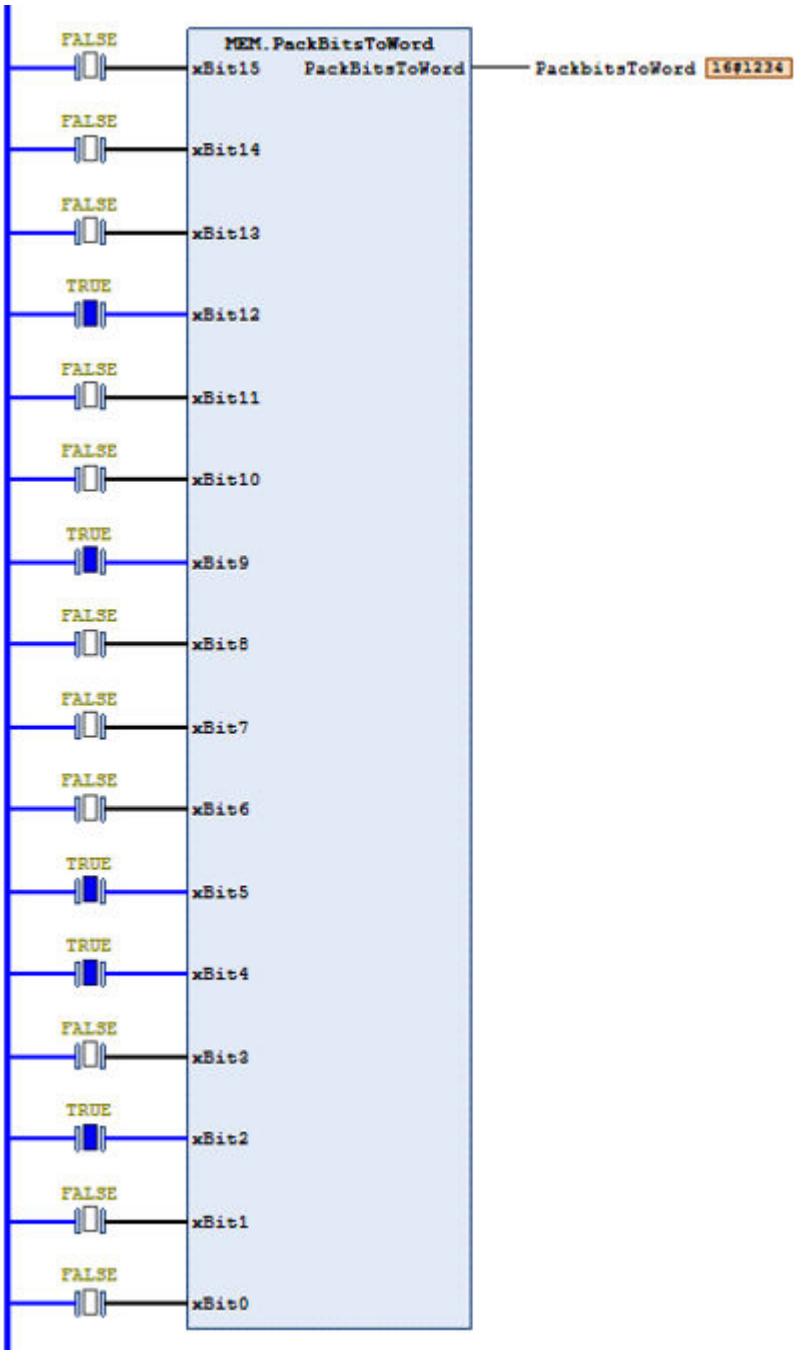
■ Program example 2

This program is designed to pack the BOOL type input variables xBit and output the packed data to the WORD type output variable PackBitsToWord.

xBit12,xBit9,xBit5,xBit4,xBit2 :=TRUE

Others :=FALSE

LD program



ST program

```
PackbitsToWord 16#1234 :=MEM.PackBitsToWord(xbit[15] FALSE ,xbit[14] FALSE ,xbit[13] FALSE ,xbit[12] TRUE ,
                                            xbit[11] FALSE ,xbit[10] FALSE ,xbit[9] TRUE ,xbit[8] FALSE ,
                                            xbit[7] FALSE ,xbit[6] FALSE ,xbit[5] TRUE ,xbit[4] TRUE ,
                                            xbit[3] FALSE ,xbit[2] TRUE ,xbit[1] FALSE ,xbit[0] FALSE );
```

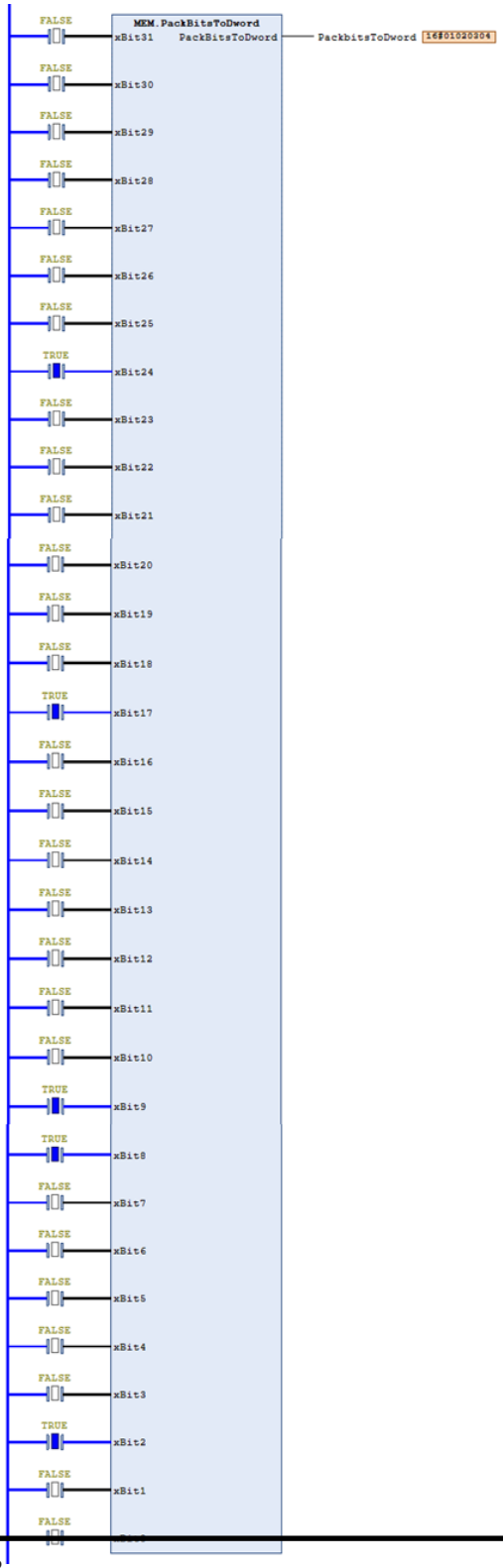
■ Program example 3

This program is designed to pack the BOOL type input variables xBit and output the packed data to the DWORD type output variable PackBitsToDword.

xBit24,xBit17,xBit9,xBit8,xBit2 :=TRUE

Others :=FALSE

LD program


MEM.PackBitsToDword
xBit31
PackBitsToDword
PackbitsToDword
16#01020304
FALSE xBit30
FALSE xBit29
FALSE xBit28
FALSE xBit27
FALSE xBit26
FALSE xBit25
TRUE xBit24
FALSE xBit23
FALSE xBit22
FALSE xBit21
FALSE xBit20
FALSE xBit19
FALSE xBit18
TRUE xBit17
FALSE xBit16
FALSE xBit15
FALSE xBit14
FALSE xBit13
FALSE xBit12
FALSE xBit11
FALSE xBit10
TRUE xBit9
TRUE xBit8
FALSE xBit7
FALSE xBit6
FALSE xBit5
FALSE xBit4
FALSE xBit3
TRUE xBit2
FALSE xBit1
FALSE

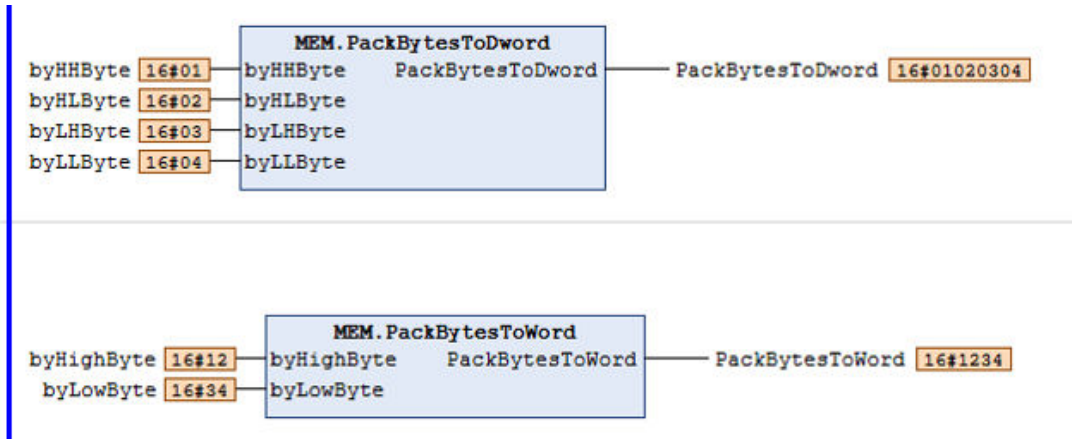
ST program

```
PackbitsToDword 16#01020304 :=MEM.PackBitsToDword(xbit[31] FALSE ,xbit[30] FALSE ,xbit[29] FALSE ,xbit[28] FALSE ,
                                                  xbit[27] FALSE ,xbit[26] FALSE ,xbit[25] FALSE ,xbit[24] TRUE ,
                                                  xbit[23] FALSE ,xbit[22] FALSE ,xbit[21] FALSE ,xbit[20] FALSE ,
                                                  xbit[19] FALSE ,xbit[18] FALSE ,xbit[17] TRUE ,xbit[16] FALSE ,
                                                  xbit[15] FALSE ,xbit[14] FALSE ,xbit[13] FALSE ,xbit[12] FALSE ,
                                                  xbit[11] FALSE ,xbit[10] FALSE ,xbit[9] TRUE ,xbit[8] TRUE ,
                                                  xbit[7] FALSE ,xbit[6] FALSE ,xbit[5] FALSE ,xbit[4] FALSE ,
                                                  xbit[3] FALSE ,xbit[2] TRUE ,xbit[1] FALSE ,xbit[0] FALSE );
```

3.7.14 MEM.PackBytesTo**(BYTE to WORD/DWORD Conversion)

This is a function that packs input BYTE type data and outputs one-word or one-dword data.

■ Icon



■ Parameter

PackBytesToWord

Scope	Name	Type	Description
Input	byHighByte	BYTE	PackBytesToWord
Input	byLowByte	BYTE	Low byte to be packed
Output	PackBytesToWord	WORD	A value of the packed input

PackBytesToDword

Scope	Name	Type	Description
Input	byHHByte	BYTE	HH byte to be packed
Input	byHLByte	BYTE	HL byte to be packed
Input	byLHByte	BYTE	LH byte to be packed
Input	byLLByte	BYTE	LL byte to be packed
Output	PackBytesToDword	DWORD	A value of the packed input

■ Program example

- PackBytesToDword

  This program is designed to pack the byHHByte, byHLByte, byLHByte, and byLLByte input variables of the BYTE type and output the packed data to the PackBytesToDword output variable of the DWORD type.

  byHHByte := 16#01 、byHLByte : = 16#02 、byLHByte := 16#03 、byLLByte := 16#04

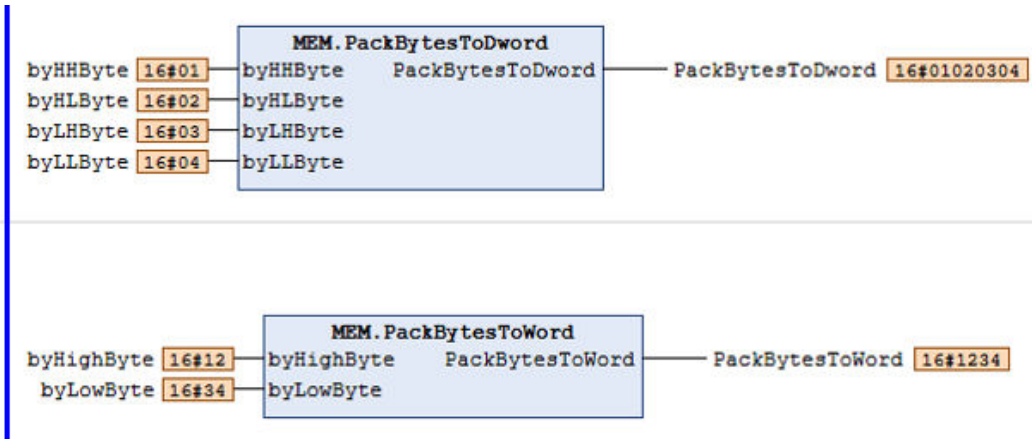
- PackBytesToWord

  This program is designed to pack the byHighByte and byLowByte input variables of the BYTE type and output the packed data to the PackBytesToWord output variable of the WORD type.

  byHighByte := 16#12

  byLowByte := 16#34

LD program



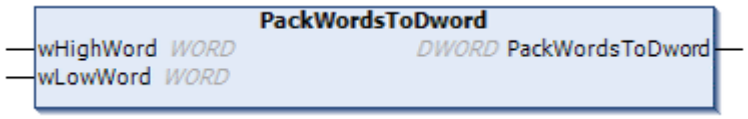
ST program

```
PackBytesToDword 16#01020304 :=
MEM.PackBytesToDword(byHHByte 16#01 ,byHLByte 16#02 ,byLHByte 16#03 ,byLLByte 16#04 );

PackBytesToWord 16#1234 :=
MEM.PackBytesToWord(byHighByte 16#12 ,byLowByte 16#34 );
```

3.7.15 MEM.PackWordsToDword (WORD to DWORD Conversion)

This is a function that packs input WORD type data and outputs a DWORD.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	wHighWord	WORD	High WORD to be packed
Input	wLowWord	WORD	Low WORD to be packed
Output	PackWordsToDword	DWORD	A value of the packed input

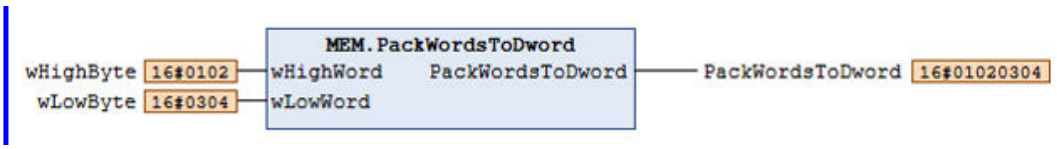
■ Program example

This program is designed to pack the wHighWord and wLowWord input variables of the WORD type and output the packed data to the PackWordsToDword output variable of the DWORD type.

wHighWord := 16#0102

wLowWord := 16#0304

LD program



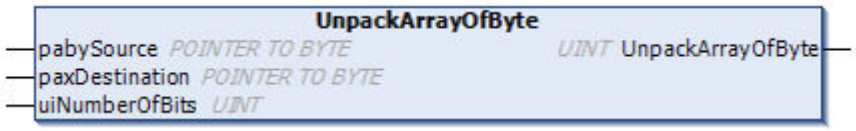
ST program

```
PackWordsToDword 16#01020304 :=MEM.PackWordsToDword(wHighWord 16#0102 ,wLowWord 16#0304 );
```

3.7.16 MEM.UnpackArrayOfByte (BYTE to BOOL Array Conversion)

This is a function that unpacks a BYTE type array to data in units of bits and copies a specified bit size of the data to a destination BOOL array. The function returns the number of bytes required for coping. The maximum copyable size is 65535 bits (approx. 8192 bytes).

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	pabySource	POINTER TO BYTE	Starting address of Byte type data
Input	paxDestination	POINTER TO BYTE	Starting address of BOOL type array data
Input	uiNumberOfBits	UINT	Number of bits to copy Effective range: 10#1 to 65535
Output	UnpackArrayOfByte	UINT	Outputs the number of bytes required for coping

■ Program example

This program is designed to unpack BYTE type copy source data (ArrayBlock) to a 24-bit amount (uiNumberOfBits) in bits and copy the unpacked data to the copy destination (xbit).

The program returns the number of bytes required for coping.

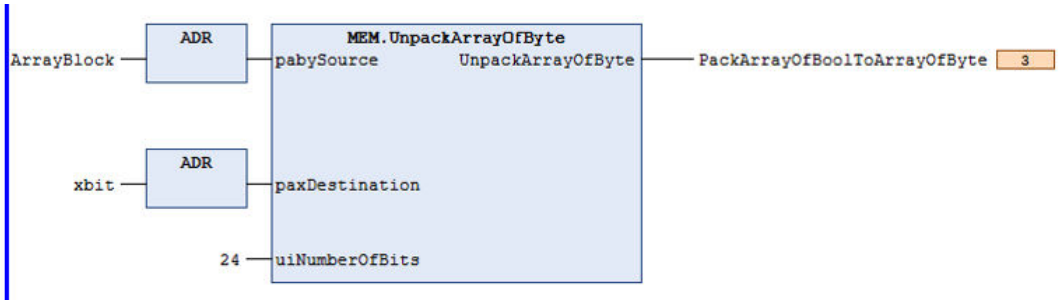
ArrayBlock[0] := 16#12

ArrayBlock[1] := 16#34

ArrayBlock[2] := 16#56

uiNumberOfBits := 10#24 (16#18)

LD program



ST program

```
ArrayBlock[0] 16#12 := 16#12;
ArrayBlock[1] 16#34 := 16#34;
ArrayBlock[2] 16#56 := 16#56;

UnpackArrayOfByte 16#0003 := MEM.UnpackArrayOfByte(ADR(ArrayBlock), ADR(xbit),24);

xbit[7] FALSE;  xbit[6] FALSE;  xbit[5] FALSE;  xbit[4] TRUE;
xbit[3] FALSE;  xbit[2] FALSE;  xbit[1] TRUE;   xbit[0] FALSE;
xbit[15] FALSE; xbit[14] FALSE; xbit[13] TRUE;  xbit[12] TRUE;
xbit[11] FALSE; xbit[10] TRUE;  xbit[9] FALSE;  xbit[8] FALSE;
xbit[23] FALSE; xbit[22] TRUE;  xbit[21] FALSE; xbit[20] TRUE;
xbit[19] FALSE; xbit[18] TRUE;  xbit[17] TRUE;  xbit[16] FALSE;
```

Note

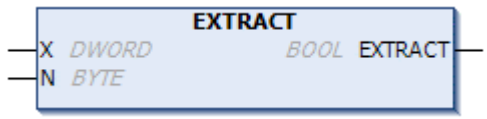
- If the number of bits to be copied uiNumberOfBits = 0, the copying will not be carried out and the return value of the function will be UnpackArrayOfByte = 0.

3.8 Bit operation instructions

3.8.1 EXTRACT (Bit Extraction)

This is a function that outputs the bit number N value (BOOL) of input value X (DWORD).

■ **Icon**



■ **Parameter**

Scope	Name	Type	Description
Input	X	DWORD	Input value
Input	N	BYTE	Number of the bit to be extracted. Effective range: 10#0 to 31
Output	EXTRACT	BOOL	The Nth bit value of the input value X

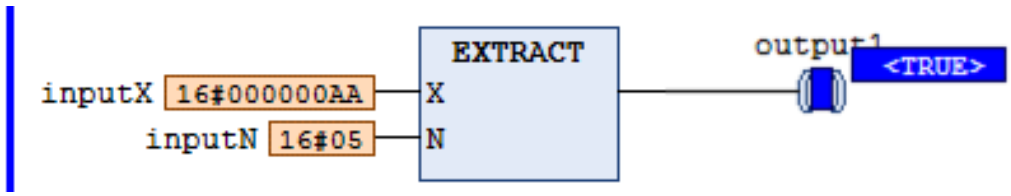
■ **Program example**

This program is designed to output the inputNth bit value of the inputX input variable of the DWORD type to the BOOL type output variable “output1”.

inputX := 16#AA (2#10101010)

inputN := 16#5

LD program



ST program

```
output1 TRUE := EXTRACT(inputX 16#000000AA ,inputN 16#05 );
```

Note

- The allowable range of the input value N (bit number) is 0 to 31 (bits).

3.8.2 PUTBIT (Bit Change)

This is a function that changes the value at bit number N of input value X (DWORD) to the B value and outputs a DWORD with the changed value at the bit number.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	X	DWORD	Input value
Input	N	BYTE	Number of the bit to be changed. Effective range: 10#0 to 31
Input	B	BOOL	Value of specified bit
Output	PUTBIT	DWORD	Value with the Nth bit of the input value X changed to the B value

■ Program example

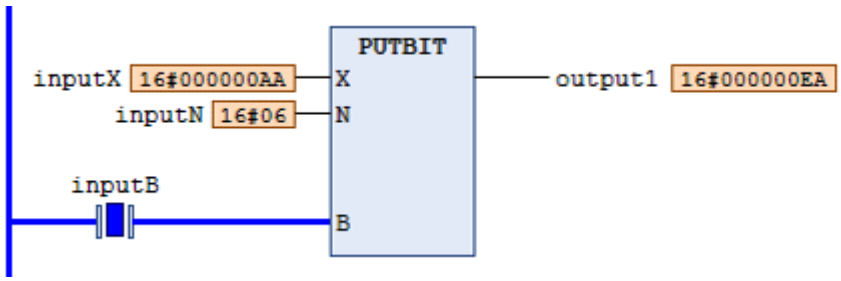
This program is designed to output a value with the inputNth bit value of the DWORD type inputX input variable changed to the inputB value to the DWORD type output variable “output1”.

inputX := 16#AA (2#10101010)

inputN := 16#6

inputB := TRUE

LD program



ST program

```
output1 16#000000EA := PUTBIT(inputX 16#000000AA, inputN 16#06, inputB TRUE);
```

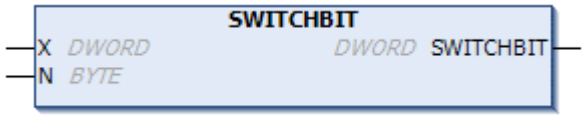
Note

- The allowable range of the input value N (bit number) is 0 to 31 (bits).

3.8.3 SWITCHBIT (Bit Inversion)

This is a function that inverts the bit number N value (0 to 1/1 to 0) of input value X (DWORD) and outputs the DWORD with the inverted value at the bit number.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	X	DWORD	Input value
Input	N	BYTE	Number of the bit to invert. Effective range: 10#0 to 31
Output	SWITCHBIT	DWORD	Value with the Nth bit value of the input value X inverted

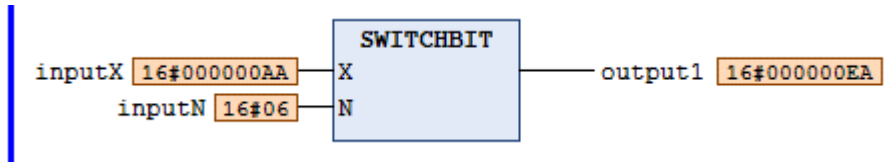
■ Program example

This program is designed to invert the inputNth bit value of the DWORD type inputX input variable and output a value with the inverted bit value to the DWORD type output variable “output1”.

inputX := 16#AA (2#10101010)

inputN := 10#6

LD program



LD program

```
output1 16#000000EA := SWITCHBIT(inputX 16#000000AA,inputN 16#06);
```

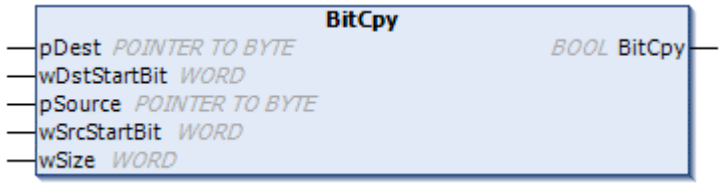
Note

- The allowable range of the input value N (bit number) is 0 to 31 (bits).

3.8.4 MEMUtils.BitCpy (Bit Copying)

This is a function that copies a specified size of bit data from copy source data.The maximum copyable size is 65535 bits (approx. 8191 bytes).

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	pDest	POINTER TO BYTE	Start pointer to copy destination data
Input	wDstStartBit	WORD	Start bit position in copy destination data
Input	pSource	POINTER TO BYTE	Start pointer to copy source data
Input	wSrcStartBit	WORD	Start bit position in copy source data
Input	wSize	WORD	Bit size to copy. Effective range: 10#1 to 65535
Output	BitCpy	BOOL	Always outputs FALSE

■ Program example

This program is designed to copy 40 bits (wSize) in copy source data (SourceData) onto copy destination data (DestinationData).

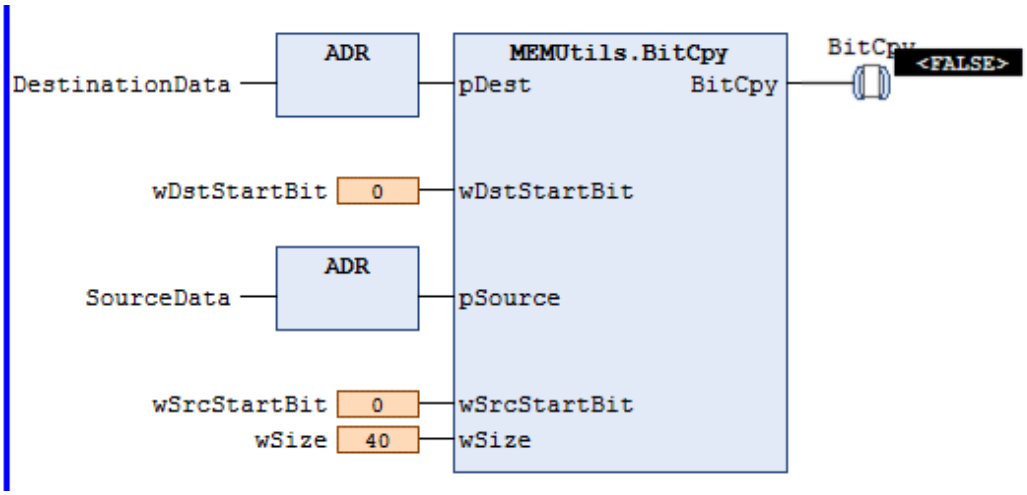
SourceData : ARRAY [0..4] OF BYTE := [1,2,3,4,5] (Copy source data)

DestinationData : ARRAY [0..4] OF BYTE := [5(0)] (Copy destination data)

wDstStartBit := 0 , wSrcStartBit := 0

wSize := 40

LD program



ST program

```
BitCpy FALSE := MEMUtils.BitCpy(ADR(DestinationData),0,ADR(SourceData ),0,40);

SourceData[0] 16#01 ;          DestinationData[0] 16#01 ;
SourceData[1] 16#02 ;          DestinationData[1] 16#02 ;
SourceData[2] 16#03 ;          DestinationData[2] 16#03 ;
SourceData[3] 16#04 ;          DestinationData[3] 16#04 ;
SourceData[4] 16#05 ;          DestinationData[4] 16#05 ;
```

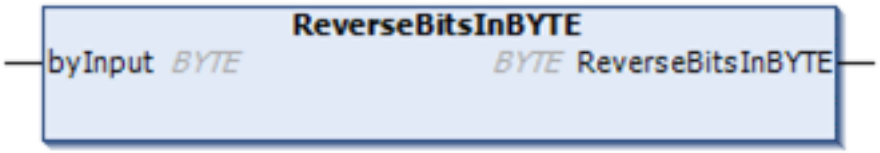
Note

- If the wSize value is 0, the copying will not be carried out.
- If copying in units of byte is required, use the function in ”3.12.10 MEM.MemMove”.
- If any of the start bit positions are set to a value other than 0, the parameters need to be configured such that both the conditions below are satisfied.
  - wSize ≤ 65536 – wDstStartBit
  - wSize ≤ 65536 – wSrcStartBit

3.8.5 MEM.ReverseBitsIn** (Bit Order Change)

This is a function that reverses the order of the bits of input BYTE- , WORD-, or DWORD-type data and outputs the data of the bits in reverse order.

■ Icon



■ Parameter

ReverseBitsInBYTE

Scope	Name	Type	Description
Input	byInput	BYTE	Input value, BYTE type data
Output	ReverseBitsInBYTE	BYTE	Value in reverse bit order

ReverseBitsInWORD

Scope	Name	Type	Description
Input	wInput	WORD	Input value, WORD type data
Output	ReverseBitsInWORD	WORD	Value in reverse bit order

ReverseBitsInDWORD

Scope	Name	Type	Description
Input	dwInput	DWORD	Input value, DWORD type data
Output	ReverseBitsInDWORD	DWORD	Value in reverse bit order

■ Program example

- ReverseBitsInBYTE

This program is designed to reverse the order of bits of the byInput input variable of the BYTE type and outputs the data of the bits in reverse order to the ReverseBitsInBYTE output variable of the BYTE type.

byInput := 16#12

- ReverseBitsInDWORD

  This program is designed to reverse the order of bits of the dwInput input variable of the DWORD type and outputs the data of the bits in reverse order to the ReverseBitsInDWORD output variable of the DWORD type.

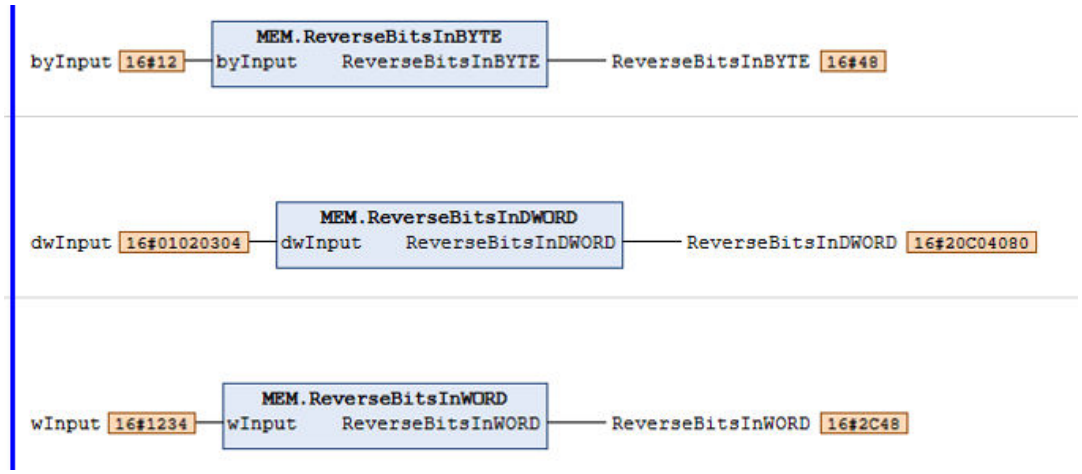
  dwInput := 16#01020304

- ReverseBitsInWORD

  This program is designed to reverse the order of bits of the wInput input variable of the WORD type and outputs the data of the bits in reverse order to the ReverseBitsInWORD output variable of the WORD type.

  wInput := 16#1234

LD program



ST program

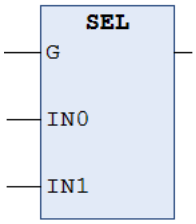
```
ReverseBitsInBYTE 16#48 :=MEM.ReverseBitsInBYTE(byInput 16#12 );

ReverseBitsInDWORD 16#20C04080 :=MEM.ReverseBitsInDWORD (dwInput 16#01020304 );

ReverseBitsInWORD 16#2C48 :=MEM.ReverseBitsInWORD(wInput 16#1234 );
```

3.9 Memory operation instructions

3.9.1 SEL (Binary Selector)

This is a function that outputs the value of the input argument IN0 or IN1 depending on whether the input argument G is true or false.

■ **Icon**



■ **Parameter**

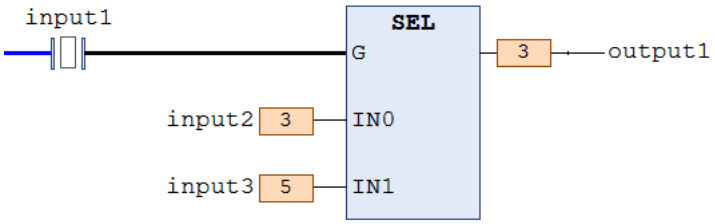
Scope	Name	Type	Description
Input	G	BOOL	Conditions for selecting the contents to be output
	IN0	All	Specifies the variable to be output if G is FALSE.
	IN1	All	Specifies the variable to be output if G is TRUE.
Output	-	All	Outputs the value of IN0 or IN1 depending on the value of G.

■ **Program example**

This program is designed to output the value of the input variable “input2” or “input3” to the output variable “output1” depending on the value of the input variable “input1”.

LD program

This program is designed to output the value of “input2” (IN0) because the value of “input1” is FALSE.



ST program

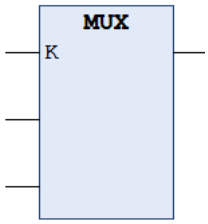
This program is designed to output the value of “input3” (IN1) to the "output1" because the value of “input1” is TRUE.

```
output1  5  := SEL(input1  TRUE ,input2  3 ,input3  5 );
```

3.9.2 MUX (Multiplexer)

This is a function that selectively outputs the input arguments depending on the value of the input argument K.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	K	(Note 1)	Specifies the value (K = 0, 1, 2...) to select the value to output.
	-	All	Specifies the value to be output depending on K.
Output	-	All	Outputs one of the input arguments depending on the value of K.

(Note 1) Usable data type

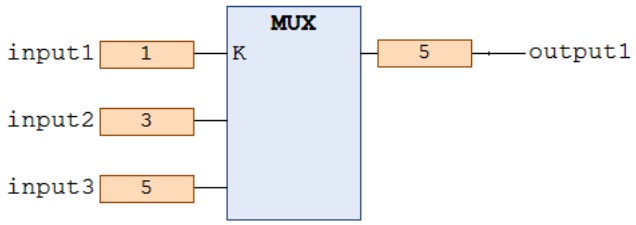
BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT

■ Program example

This program is designed to output the value of the input variable “input2” or “input3” to the output variable “output1” depending on the value of the input variable “input1”.

LD program

This program is designed to output the value of “input3” to “output1” depending on the value (1) of “input1”.



ST program

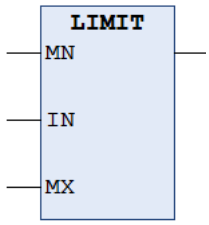
This program is designed to output the value of “input 2” to the “output 1” depending on the value (0) of “input 1”.

```
output1 3 := MUX(input1 0, input2 3, input3 5);
```

3.9.3 LIMIT (Limiter)

This is a function that limits the input value with the lower and upper limit values and outputs a restricted value.

■ Icon



■ Parameter

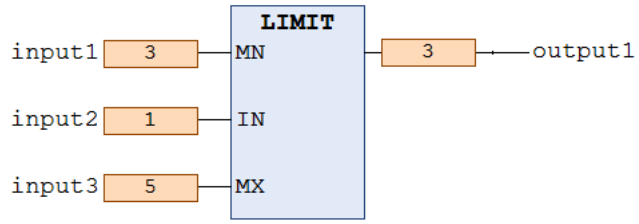
Scope	Name	Type	Description
Input	MN	All	Specifies the lower limit of the value to be output.
	IN	All	Specifies the input values to be restricted.
	MX	All	Specifies the upper limit of the value to be output.
Output	-	All	Outputs values according to the following conditions. IN ≤ MN: Outputs "MN". MN ≤ IN ≤ MX: Outputs "IN". MX ≤ IN: Outputs "MX".

■ Program example

This program is designed to limit the value range of the input variable “input2” with the input variable “input1” (lower limit) and the input variable “input3” (upper limit) and to output the limited value to the output variable “output1”.

LD program

This program is designed to output “3” to “output1” because the value (1) of “input2” (IN) is less than or equal to the lower limit (3) specified in “input1” (MN).



ST program

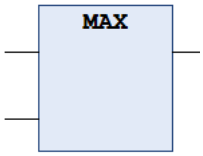
This program is designed to output “5” to “output1” because the value (8) of “input2” is greater than or equal to the upper limit (5) specified in “input3”.

```
output1 5 := LIMIT(input1 3, input2 8, input3 5);
```

3.9.4 MAX (Maximum Value)

This is a function that outputs the maximum value of the input arguments.

■ Icon



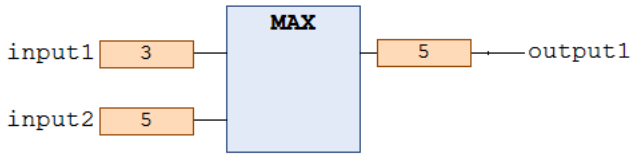
■ Parameter

Scope	Type	Description
Input	All	Specifies the values from which to obtain the maximum value.
Output	All	Outputs the maximum value of the input values.

■ Program example

This program is designed to output the maximum value of the input variables to the output variable “output1”.

LD program



ST program

```
output1 5 := MAX(input1 3, input2 5);
```

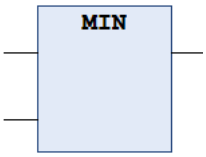
Info.

- If you want to increase input arguments in the LD program, right-click on the MAX function, and, on the displayed menu, select "Add Input".

3.9.5 MIN (Minimum Value)

This is a function that outputs the minimum value of the input arguments.

■ Icon



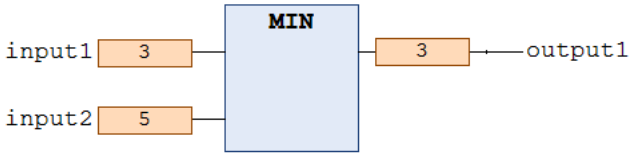
■ Parameter

Scope	Type	Description
Input	All	Specifies the values from which to obtain the minimum value.
Output	All	Outputs the minimum value of the input values.

■ Program example

This program is designed to output the minimum value of the input variables to the output variable “output1”.

LD program



ST program

```
output1 3 := MIN(input1 3, input2 5);
```

Info.

- If you want to increase input arguments in the LD program, right-click on the MIN function, and, on the displayed menu, select "Add Input".

3.9.6 MEMUtils.Swap (Byte Swapping)

This is a function that swaps specified bytes in order at a specified pointer to data in units of byte. The numbers of bytes that can be swapped are 2 bytes, 4 bytes, and 8 bytes.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	puc	POINTER TO BYTE	Start pointer at which byte swapping starts
Input	iSize	INT	The number of bytes to swap. Allowable inputs: 10#2/10#4/10#8
Output	Swap	INT	Status Successfully complete = 1 Error = -1

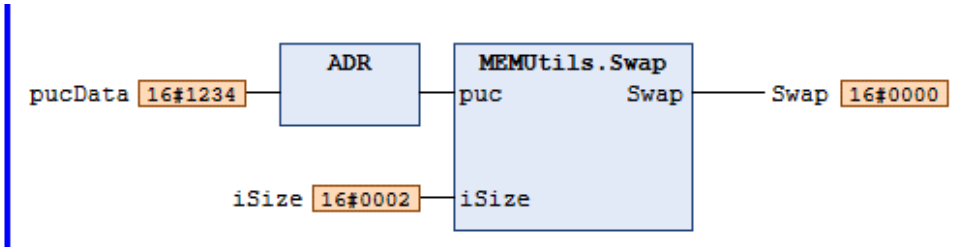
■ Program example

This program is designed to swap 2 bytes (iSize) in order in source data (pucData) in units of byte.

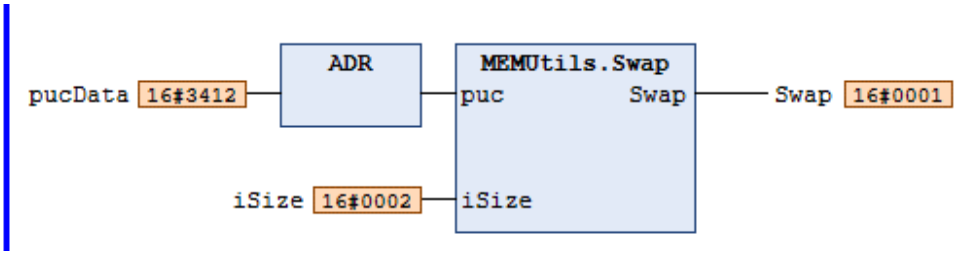
pucData := 16#1234 (Data to swap)

iSize := 2

LD program



Execution result



ST program

```
Swap 16#0001 := MEMUtils.Swap(ADR(pucData 16#1234 ),2);
```

Execution result

```
Swap_before 16#1234 ;
Swap_after 16#3412 ;
```

Note

- If iSize is set to a value other than 2/4/8, byte swapping will not be carried out and the return value (Swap) of the function will be -1.

3.9.7 MEM.Compare (Memory Comparison)

This is a function that compares two specified memory block data pieces. When the memory block data pieces match each other, the function outputs 0. If they do not match, the function outputs the first location at which they differ.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	pMemoryBlockA	POINTER TO BYTE	Start pointer to data A to compare
Input	pMemoryBlockB	POINTER TO BYTE	Start pointer to data B to compare
Input	uiNumberOfBytes	UINT	Number of data bytes to compare Effective range: 10#1 to 10#65534
Output	Compare	UINT	0 = Data pieces match / Other = First location (BYTE) at which data pieces differ

■ Program example

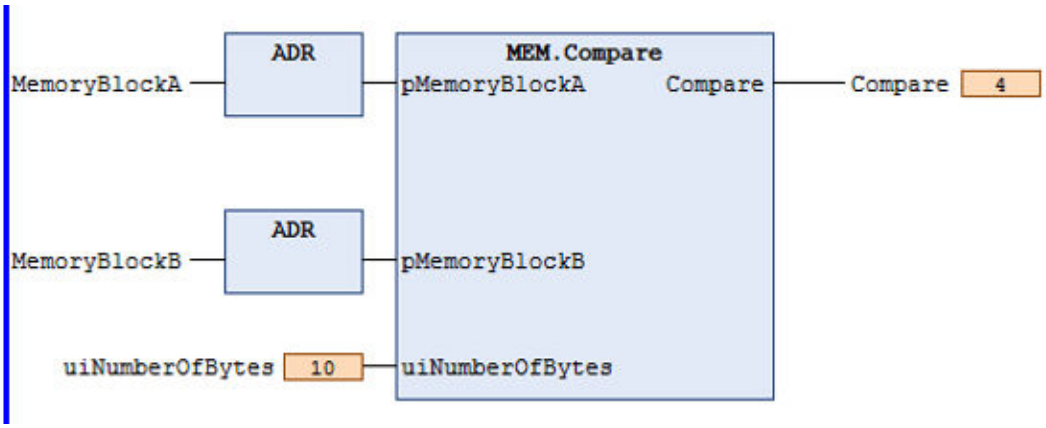
This program is designed to compare two specified pieces of BYTE type data (MemoryBlockA/ MemoryBlockB).

MemoryBlockA : ARRAY[0..9] OF BYTE := 0,1,2,3,4,5,6,7,8,9

MemoryBlockB : ARRAY[0..9] OF BYTE := 0,1,2,0,4,5,0,7,8,9

uiNumberOfBytes := 10

LD program



ST program

```
Compare 4 :=MEM.Compare(ADR(MemoryBlockA),ADR(MemoryBlockB),10);
```

Note

- If the number of data bytes to compare (uiNumberOfBytes) is 0, the function does not compare the two data pieces and returns 0.
- The function does not operate properly if the number of data bytes to compare (uiNumberOfBytes) is set to 65535 bytes. Thus, do not use that byte size.

3.9.8 MEM.FindBlock(Memory block search)

This is a function that searches memory block data A for memory block data B. If the target data is found, the function outputs the location at which the target data starts. If the target data is not found, the function outputs 0.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	pMemoryBlockA	POINTER TO BYTE	Start pointer to the data to search
Input	uiLengthBlockA	UINT	Number of bytes of the data to search Effective range: 10#1 to 10#65535

Scope	Name	Type	Description
Input	pMemoryBlockB	POINTER TO BYTE	Start pointer to the data to find
Input	uiLengthBlockB	UINT	Number of bytes of the data to find Effective range: 10#1 to 10#65535
Output	FindBlock	UINT	0 = Data not found / Other = Location (BYTE) at which the found data starts

■ Program example

This program is designed to search specified BYTE type data (MemoryBlockA) for specified BYTE type data (MemoryBlockB).

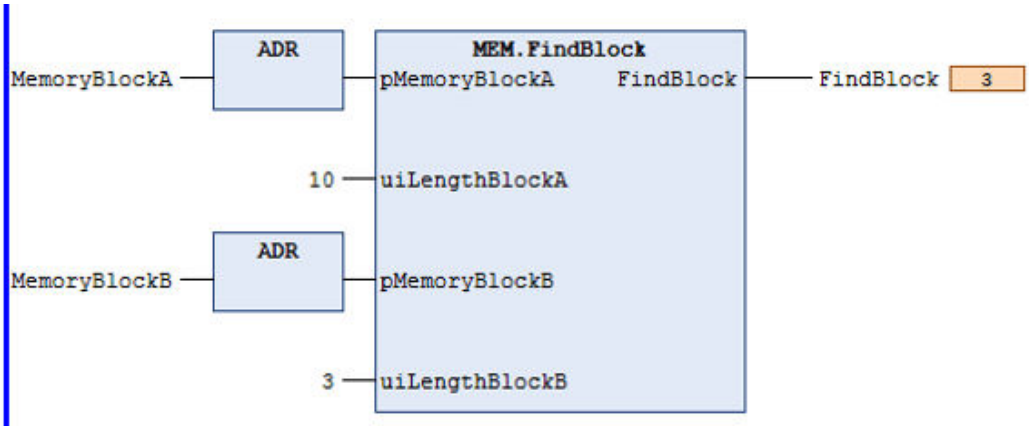
pMemoryBlockA : ARRAY[0..9] OF BYTE := 1,2,3,4,5,1,2,3,4,5

pMemoryBlockB : ARRAY[0..2] OF BYTE := 3,4,5

uiLengthBlockA := 10

uiLengthBlockB := 3

LD program



ST program

```
FindBlock 3 :=MEM.FindBlock(ADR(MemoryBlockA),10,ADR(MemoryBlockB),3);
```

Note

- Do not use this function with 0 set in the number of bytes of the data to find (uiLengthBlockB). If uiLengthBlockB = 0, the return value of the function is 16#FFFF.

3.9.9 MEM.FindByte (Find Byte Data)

This is a function that searches specified memory block data for specified one-byte data. If the target data is found, the function outputs the location at which the target data starts. If the target data is not found, the function outputs 0.

■ Icon

■ Parameter

Parameter	Name	Type	Description
Input	pMemoryBlock	POINTER TO BYTE	Start pointer to the data to search
Input	uiLength	UINT	Number of bytes of the data to search Effective range: 10#1 to 10#65534
Input	byValue	BYTE	Data to find Effective range: 10#0 to 10#255
Output	FindByte	UINT	0 = Data not found / Other = Location (BYTE) at which the found data starts

■ Program example

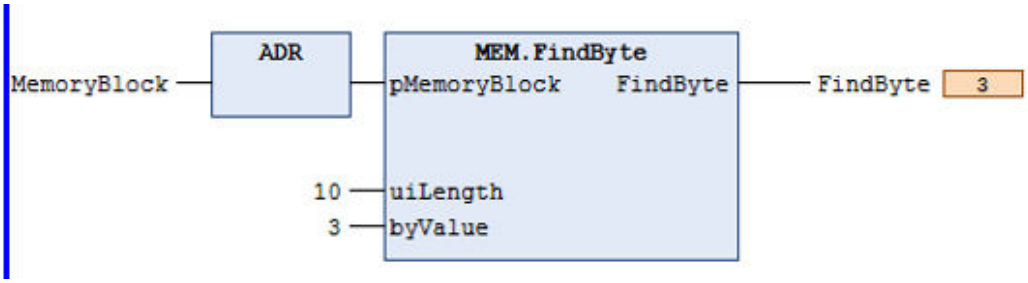
This program is designed to search specified BYTE type data (MemoryBlock) for BYTE type data (byValue).

pMemoryBlock : ARRAY[0..9] OF BYTE := 1,2,3,4,5,1,2,3,4,5

byValue := 3

uiLength := 10

LD program



ST program

```
FindByte 3 :=MEM.FindByte(ADR(MemoryBlock),10,3);
```

Note

- If the number of bytes of the data to search (uiLength) is 0, the function does not search the data and returns 0.
- The function does not operate properly if the number of bytes of the data to search (uiLength) is set to 65535 bytes. Thus, do not use that byte size.

3.9.10 MEM.MemFill (Memory Fill)

This is a function that fills a specified size in data memory with a specified data value.

■ Icon

■ Parameter

MemFill

Scope	Name	Type	Description
Input	pMemoryBlock	POINTER TO BYTE	Starting address of data to fill
Input	uiLength	UINT	Number of bytes to fill Effective range: 10#1 to 65534
Input	byFillValue	BYTE	Data value with which to fill the data Effective range: 10#0 to 255
Output	MemFill	BOOL	TRUE = Filling completed

■ Program example

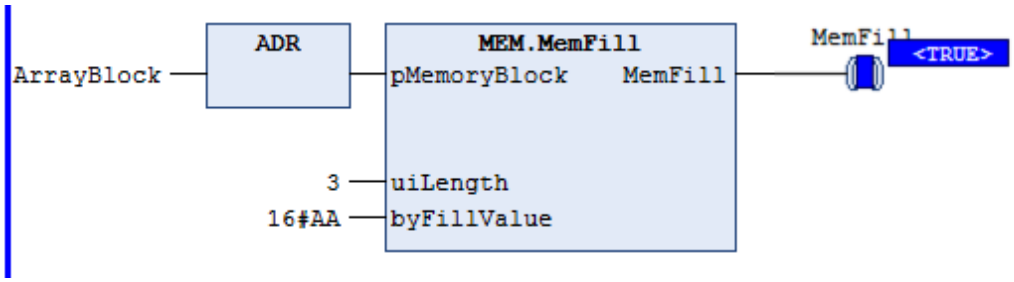
This program is designed to fill a three-byte size (uiLength) in data to fill (ArrayBlock) with the 16#AA data value (byFillValue).

ArrayBlock : ARRAY [0..4] OF BYTE := [5(0)] (data to fill: default value)

uiLength := 10#3

byFillValue := 16#AA (data value with which to fill the data)

LD program



ST program

```
MemFill TRUE := MEM.MemFill(ADR(ArrayBlock[0] 16#AA ),3,16#AA);

ArrayBlock[0] 16#AA ;
ArrayBlock[1] 16#AA ;
ArrayBlock[2] 16#AA ;
ArrayBlock[3] 16#00 ;
ArrayBlock[4] 16#00 ;
```

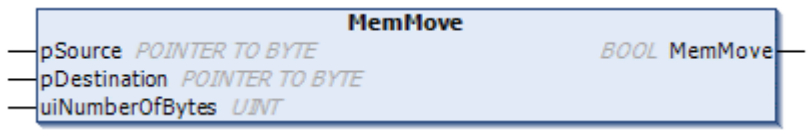
Note

- The function does not operate properly if the number of bytes to fill (uiLength) is set to the maximum 65535 bytes. Thus, do not use that byte size.
- If the number of bytes to fill (uiLength) is 0, the data filling will not be carried out.
- If the start pointer to data to fill (pMemoryBlock) is set to 0 (NULL), the function returns FALSE.

3.9.11 MEM.MemMove (Memory Copying)

This is a function that copies a specified size in data memory onto copy destination data memory.

■ Icon



■ Parameter

MemMove

Scope	Name	Type	Description
Input	pSource	POINTER TO BYTE	Copy source data starting address
Input	pDestination	POINTER TO BYTE	Copy destination data starting address
Input	uiNumberOfBytes	UINT	Number of bytes to copy Effective range: 10#1 to 65534
Output	MemMove	BOOL	TRUE = Copying completed

■ Program example

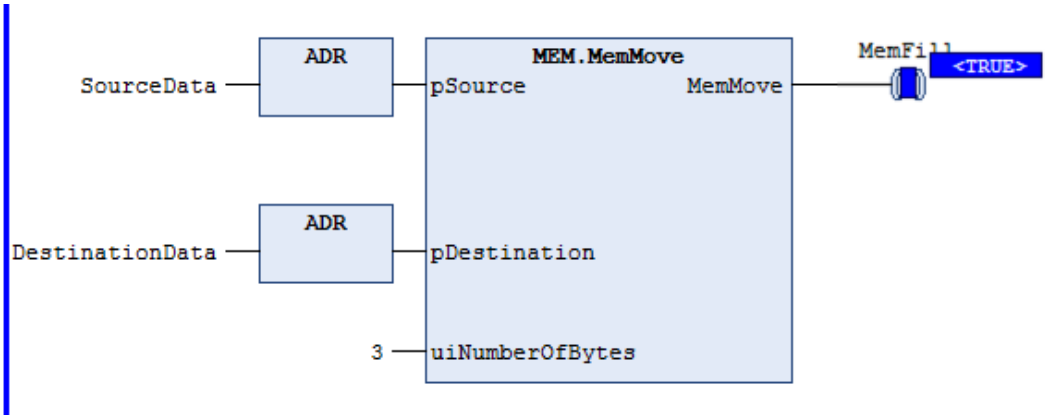
This program is designed to copy 3 bytes (uiNumberOfBytes) in copy source data (SourceData) onto copy destination data (DestinationData).

SourceData : ARRAY [0..4] OF BYTE := [1,2,3,4,5] (Copy source data)

DestinationData : ARRAY [0..4] OF BYTE := [5(0)] (Copy destination data: default value)

uiNumberOfBytes := 3

LD program



ST program

```
MemMove TRUE := MEM.MemMove(ADR(SourceData), ADR(DestinationData),3);

SourceData[0] 16#01;      DestinationData[0] 16#01;
SourceData[1] 16#02;      DestinationData[1] 16#02;
SourceData[2] 16#03;      DestinationData[2] 16#03;
SourceData[3] 16#04;      DestinationData[3] 16#00;
SourceData[4] 16#05;      DestinationData[4] 16#00;
```

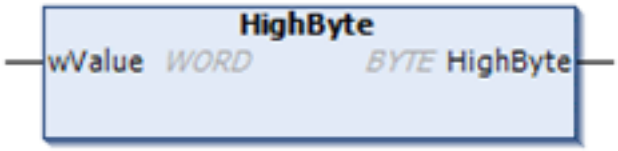
Note

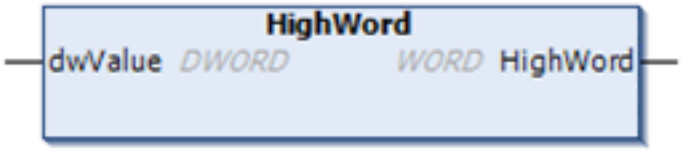
- The function does not operate properly if the number of bytes to copy (uiNumberOfBytes) is set to the maximum 65535 bytes. Thus, do not use that byte size.
- If the number of bytes to copy (uiNumberOfBytes) is 0, the copying will not be carried out.
- If any of the start pointer to copy source data (pSource) and the start pointer to copy destination data (pDestination) are set to 0 (NULL), the function returns FALSE.
- If copying in units of bit is required, use the function in ”3.12.4 MEMUtils.BitCpy”.

3.9.12 EM.High** (High Byte/High WORD Extraction)

This is a function that outputs high byte / high WORD of the input value.

■ Icon





■ Parameter

HighByte

Scope	Name	Type	Description
Input	wValue	WORD	Input value of WORD type
Output	HighByte	BYTE	Outputs high byte of the input value

HighWord

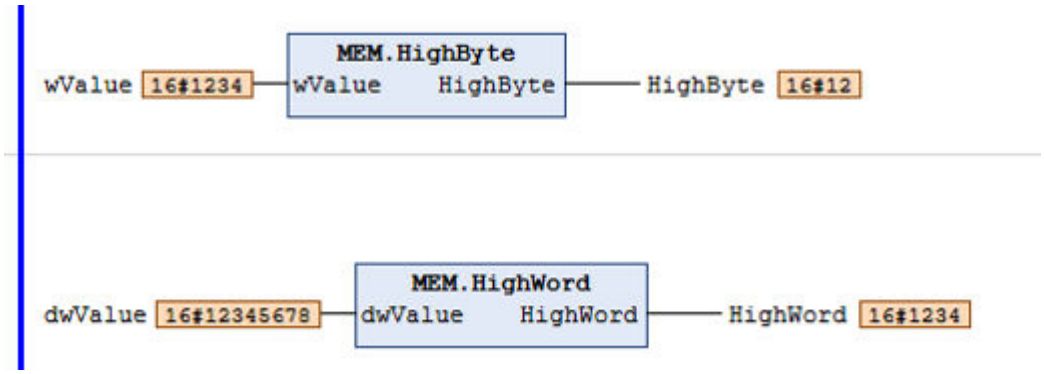
Scope	Name	Type	Description
Input	dwValue	DWORD	Input value of DWORD type
Output	HighWord	WORD	Outputs high WORD of the input value

■ Program example

This program is designed to output the high byte (16#12) of the wValue input variable (16#1234) of the WORD type to the HighByte output variable of the BYTE type.

This program is designed to output the high WORD (16#1234) of the dwValue input variable (16#12345678) of the DWORD type to the HighWord output variable of the WORD type.

LD program



ST program

```
HighByte 16#12 :=MEM.HighByte(wValue 16#1234 );

HighWord 16#1234 :=MEM.HighWord(dwValue 16#12345678 );
```

3.9.13 MEM.Low** (Low Byte/Low WORD Extraction)

This is a function that outputs low byte / low WORD of the input value.

■ **Icon**

■ **Parameter**

LowByte

Scope	Name	Type	Description
Input	wValue	WORD	Input value of WORD type
Output	LowByte	BYTE	Outputs low byte of the input value

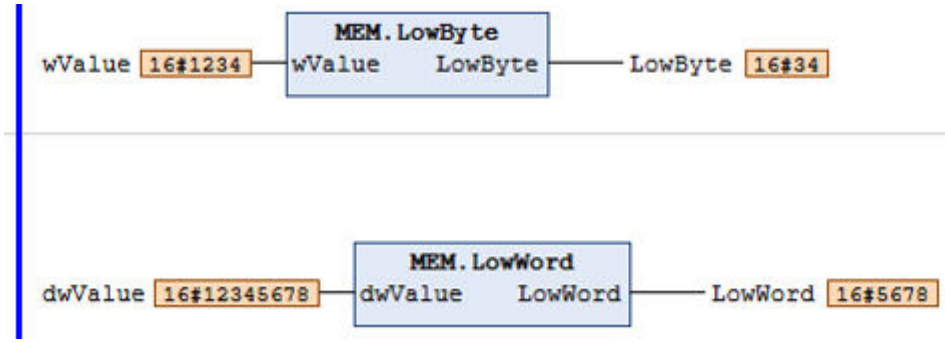
LowWord

Scope	Name	Type	Description
Input	dwValue	DWORD	Input value of DWORD type
Output	LowWord	WORD	Outputs low WORD of the input value

■ **Program example**

This program is designed to output the low byte (16#34) of the wValue input variable (16#1234) of the WORD type to the LowByte output variable of the BYTE type.
This program is designed to output the low WORD (16#5678) of the dwValue input variable (16#12345678) of the DWORD type to the LowWord output variable of the WORD type.

LD program



ST program

```
LowByte 16#34 :=MEM.LowByte (wValue 16#1234 );

LowWord 16#5678 :=MEM.LowWord (dwValue 16#12345678 );
```

3.9.14 MEM.ReverseBYTEsIn** (Byte Order Change)

This is a function that reverses the order of the bytes of input WORD-, or DWORD-type data and outputs the data of the bytes in reverse order.

■ **Icon**

■ Parameter

ReverseBYTEsInWORD

Scope	Name	Type	Description
Input	wInput	WORD	Input value, WORD type data
Output	ReverseBYTEsInWORD	WORD	Value in reverse byte order

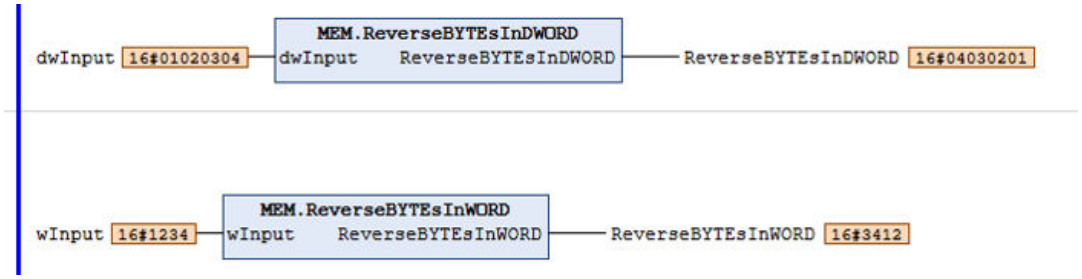
ReverseBYTEsInDWORD

Scope	Name	Type	Description
Input	dwInput	DWORD	Input value, DWORD type data
Output	ReverseBYTEsInDWORD	DWORD	Value in reverse byte order

■ Program example

- ReverseBYTEsInDWORD
  This program is designed to reverse the order of bytes of the dwInput input variable of the DWORD type and outputs the data of the bytes in reverse order to the ReverseBYTEsInDWORD output variable of the DWORD type.
  dwInput := 16#01020304
- ReverseBYTEsInWORD
  This program is designed to reverse the order of bytes of the wInput input variable of the WORD type and outputs the data of the bytes in reverse order to the ReverseBYTEsInWORD output variable of the WORD type.
  wInput := 16#1234

LD program



ST program

```
ReverseBYTEsInDWORD 16#04030201 :=MEM.ReverseBYTEsInDWORD(dwInput 16#01020304 );

ReverseBYTEsInWORD 16#3412 :=MEM.ReverseBYTEsInWORD(wInput 16#1234 );
```

3.9.15 MEM.ReverseWORDsInDWORD (WORD Order Change)

This is a function that reverses the order of the WORDs of input DWORD-type data and outputs the data of the WORDs in reverse order.

■ Icon

■ Parameter

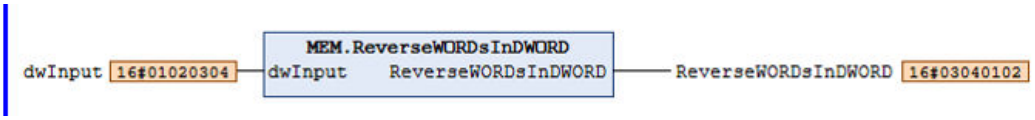
Scope	Name	Type	Description
Input	dwInput	DWORD	Input value, DWORD type data
Output	ReverseWORDsInDWORD	DWORD	Value in reverse WORD order

■ Program example

This program is designed to reverse the order of WORDs of the dwInput input variable of the DWORD type and outputs the data of the WORDs in reverse order to the ReverseWORDsInDWORD output variable of the DWORD type.

dwInput := 16#01020304

LD program



ST program

```
ReverseWORDsInDWORD 16#03040102 :=MEM.ReverseWORDsInDWORD(dwInput 16#01020304);
```

3.10 Character string instructions

Character string instructions can be used to perform various operations on character strings.

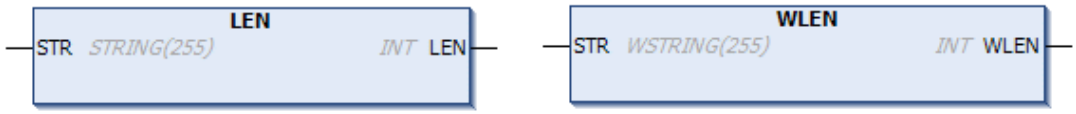
There is no limit to the length of a STRING type string, but the string functions described in this chapter only process lengths of 1 to 255 characters.

Do not use a string longer than 256 characters in the function input.

3.10.1 LEN/WLEN (string length)

This is a function that outputs the length of a character string.

■ Icon



■ Parameter

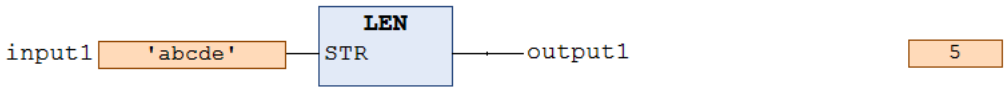
Scope	Name	Type	Description
Input	STR	STRING(255)/ WSTRING(255)	Specifies the character string from which to obtain the length.
Output	LEN/WLEN	INT	Outputs the character string length of the input argument.

■ Program example

This program is designed to output the character string length of the input variable “input1” to the output variable ”output1”.

This is a program example for the function LEN.

LD program

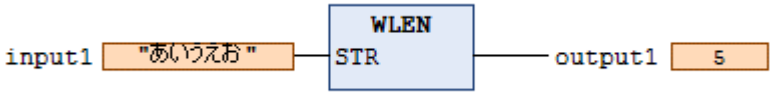


ST program

```
output1 5 := LEN(input1 'abcde');
```

This is a program example for the function WLEN.

LD program



ST program

```
output1  5  :=WLEN(input1  "あいうえお"  );
```

3.10.2 LEFT/WLEFT (extract text from left edge)

This is a function that extracts a character string consisting of the specified number of characters from the left end of the character string and outputs the extracted data.

■ Icon

■ Parameter

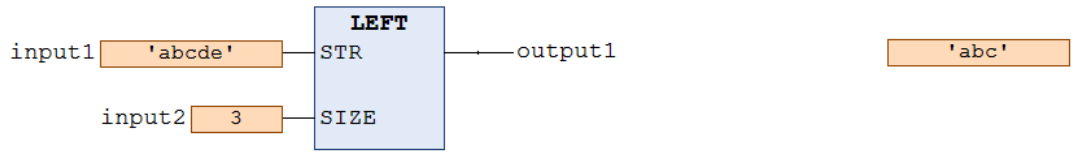
Scope	Name	Type	Description
Input	STR	STRING(255)/ WSTRING(255)	Specifies the character string from which a character string is to be extracted.
	SIZE	INT	Specifies the number of characters to be extracted from the left.
Output	LEFT/ WLEFT	STRING(255)/ WSTRING(255)	Extracts a character string consisting of the number of characters specified in SIZE from STR and outputs the extracted data.

■ Program example

This program extracts the character string of the number of characters (3 characters) specified by input2 from the character string of the input variable input1 from the left end and outputs it to the output variable output1.

This is a program example for the function LEFT.

LD program

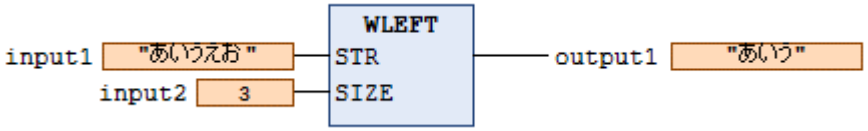


ST program

```
output1  'abc'  := LEFT(input1  'abcde' , input2  3 );
```

This is a program example for the function WLEFT.

LD program



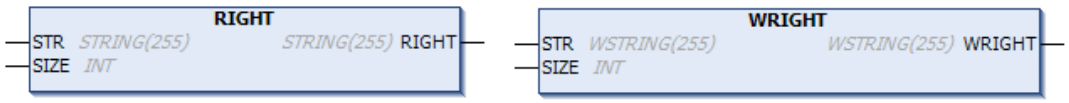
ST program

```
output1 "あいう" :=WLEFT(input1 "あいうえお",input2 3);
```

3.10.3 RIGHT/WRIGHT (Extract text from the right end)

This is a function that extracts a character string consisting of the specified number of characters from the right end of the character string and outputs the extracted data.

■ Icon



■ Parameter

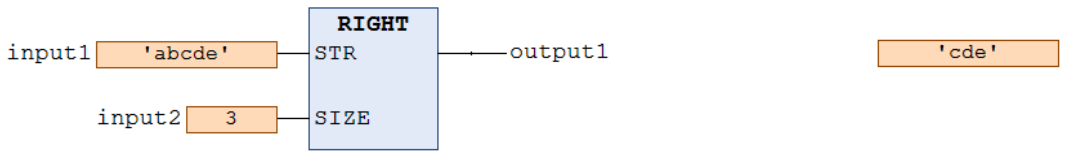
Scope	Name	Type	Description
Input	STR	STRING(255)/ WSTRING(255)	Specifies the character string from which a character string is to be extracted.
	SIZE	INT	Specifies the number of characters to be extracted from the right.
Output	RIGHT/ WRIGHT	STRING(255)/ WSTRING(255)	Extracts a character string consisting of the number of characters specified in SIZE from STR and outputs the extracted data.

■ Program example

This program is designed to extract a character string consisting of the number of characters (3 characters) specified in “input2” from the right end of the character string of the input variable “input1” and to output the extracted character string to the output variable ”output1”.

This is a program example for the function RIGHT.

LD program

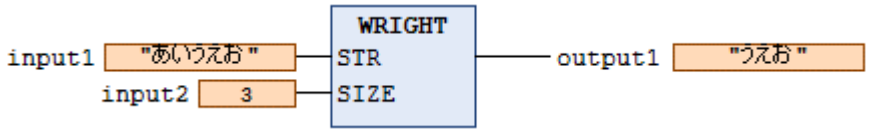


ST program

```
output1 'cde' := RIGHT(input1 'abcde', input2 3);
```

This is a program example for the function WRIGHT.

LD program



ST program

```
output1 "うえお" :=WRIGHT(input1 "あいうえお",input2 3);
```

3.10.4 MID/WMID (extract string from specified position)

This function extracts a specified number of characters from the right end of a character string and outputs it.

■ Icon

■ Parameter

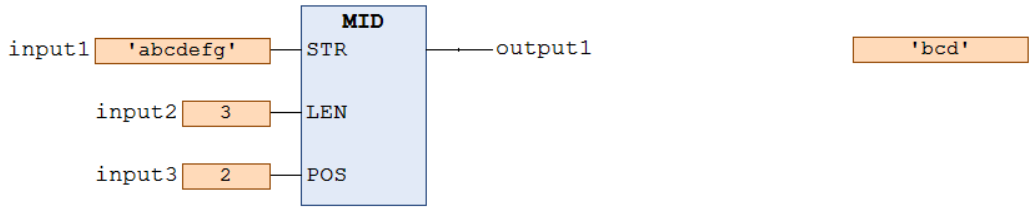
Scope	Name	Type	Description
Input	STR	STRING(255)/ WSTRING(255)	Specifies the character string from which a character string is to be extracted.
	LEN	INT	Specifies the number of characters to be extracted.
	POS	INT	Specified the position from which extraction is to be started.
Output	MID/WMID	STRING(255)/ WSTRING(255)	Extracts a character string consisting of the number of characters specified in LEN from STR starting from the position specified in POS and outputs the extracted data.

■ Program example

This program is designed to extract a character string consisting of the number of characters (3 characters) specified in “input2” from the character string of the input variable “input1”, starting from the position (2nd character from the left end) specified in “input3”, and to output the extracted data to the output variable ”output1”.

This is a program example for the function MID.

LD program

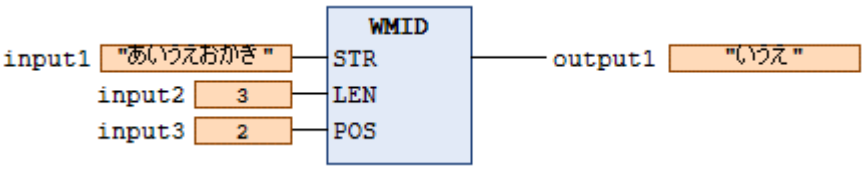


ST program

```
output1 'bcd' := MID(input1 'abcdefg', input2 3, input3 2);
```

This is a program example for the function WMID.

LD program



ST program

```
output1 "いうえ" :=WMID(input1 "あいうえおかき",input2 3,input3 2);
```

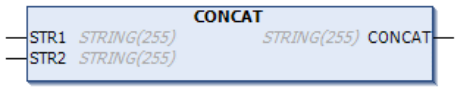
Info.

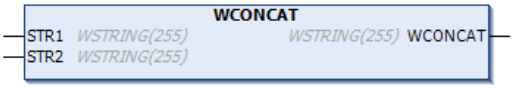
If POS = 0, it is not extracted.

3.10.5 CONCAT/WCONCAT (string concatenation)

This is a function that concatenates the character strings.

■ Icon





■ Parameter

Scope	Name	Type	Description
Input	STR1	STRING(255)/ WSTRING(255)	Specifies the character string to be concatenated.
	STR2	STRING(255)/ WSTRING(255)	Specifies the character string to be concatenated.

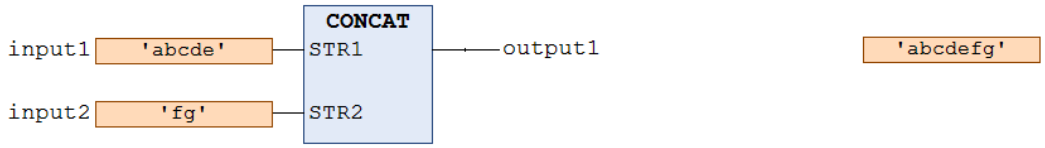
Scope	Name	Type	Description
Output	CONCAT/ WCONCAT	STRING(255)/ WSTRING(255)	Concatenate the STR2 character string to the right of the STR1 character string and output the concatenated data.

■ Program example

This program is designed to concatenate the character string of “input2” to the character string of the input variable “input1” and to output the concatenated data to the output variable ”output1”.

This is a program example for the function CONCAT.

LD program

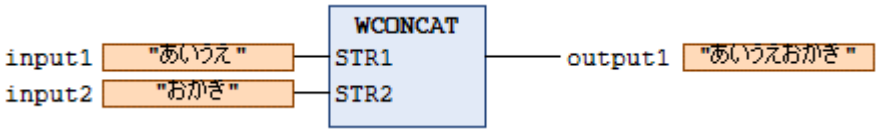


ST program

```
output1 'abcdefg' := CONCAT(input1 'abcde', input2 'fg');
```

This is a program example for the function WCONCAT.

LD program



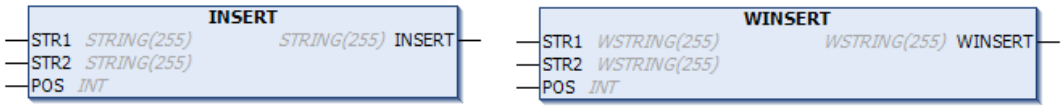
ST program

```
output1 "あいうえおかき" :=WCONCAT(input1 "あいうえ",input2 "おかき");
```

3.10.6 INSERT/WINSERT (Inserting a Character String)

This is a function that inserts a character string in the specified position and outputs the inserted data.

■ Icon



■ Parameter

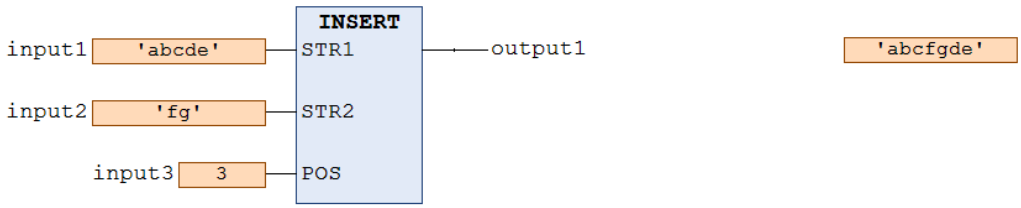
Scope	Name	Type	Description
Input	STR1	STRING(255)/ WSTRING(255)	Specifies the character string in which a character string is to be inserted.
	STR2	STRING(255)/ WSTRING(255)	Specifies the character string to be inserted.
	POS	INT	Specifies the position to be inserted. n-th character from the left
Output	INSERT/ WINSERT	STRING(255)/ WSTRING(255)	Insert the string of STR2 into the position of POS in the string of STR1 and output

■ Program example

This program is designed to insert the character string of “input2” in the position (3rd character from the left end) specified in “input3” from the left of the the character string of the input variable “input1” and to output the inserted data to the output variable ”output1”.

This is a program example for the function INSERT.

LD program

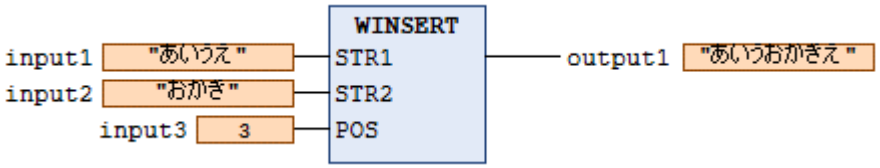


ST program

```
output1 'abcfgde' := INSERT(input1 'abcde', input2 'fg', input3 3);
```

This is a program example for the function WINSERT.

LD program



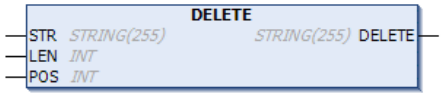
ST program

```
output1 "あいうおかきえ" :=WINSERT(input1 "あいうえ",input2 "おかき",input3 3);
```

3.10.7 DELETE/WDELETE (delete string)

This is a function that deletes a character string from the specified position and outputs the deleted data.

■ Icon



■ Parameter

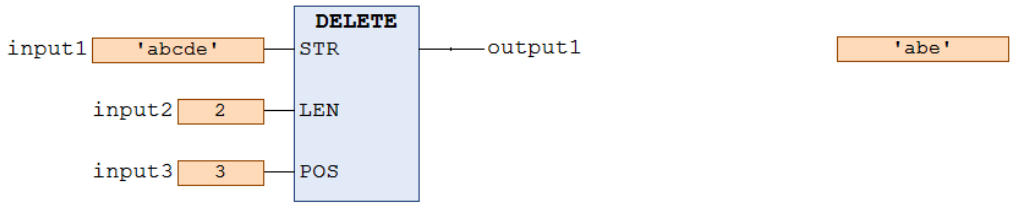
Scope	Name	Type	Description
Input	STR	STRING(255)/ WSTRING(255)	Specifies the character string from which a character string is to be deleted.
	LEN	INT	Specifies the length of the character string to be deleted.
	POS	INT	Specified the position from which deletion is to be started. n-th character from the left
Output	DELETE/ WDELETE	STRING(255)/ WSTRING(255)	Deletes a character string consisting of the number of characters specified in LEN from the left end of the STR character string starting from the position specified in POS and outputs the deleted data.

■ Program example

This program is designed to delete a character string consisting of the number of characters (2 characters) specified in “input2” from the character string of the input variable “input1” starting from the position (3rd character from the left) specified in “input3” and to output the deleted data to the output variable “output1”.

This is a program example for the function DELETE.

LD program

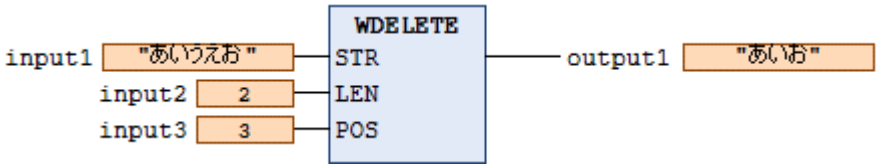


ST program

```
output1 'abe' := DELETE(input1 'abcde', input2 2, input3 3);
```

This is a program example for the function WDELETE.

LD program



ST program

```
output1 "あいお" :=WDELETE(input1 "あいうえお",input2 2,input3 3);
```

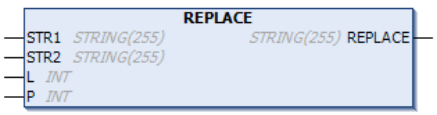
Info.

If POS = 0, LEN is used with a setting of -1.

3.10.8 REPLACE/WREPLACE (replace string)

This is a function that replaces the character strings and outputs the replaced character strings.

■ Icon



■ Parameter

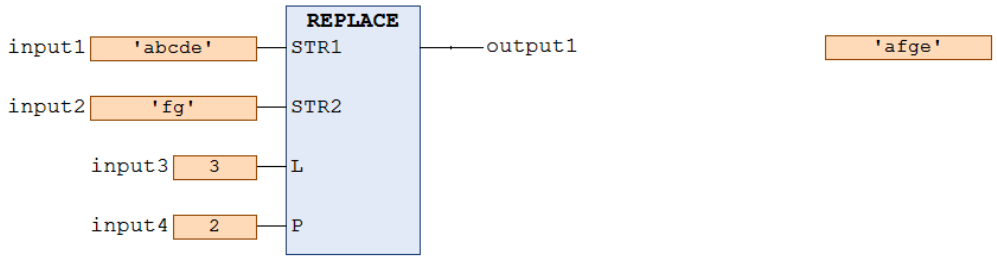
Scope	Name	Type	Description
Input	STR1	STRING(255)/ WSTRING(255)	Specifies the character string to be replaced.
	STR2	STRING(255)/ WSTRING(255)	Specifies the character string to be added by replacement.
	L	INT	Specifies the number of characters to be deleted by replacement.
	P	INT	Specify where to add STR2 text by substitution
Output	REPLACE/ WREPLACE	STRING(255)/ WSTRING(255)	Replaces the number of characters specified in L with the character string specified in STR2 from the left end of the character string specified in STR1 starting from the position specified in P and outputs the replaced data.

■ Program example

This program is designed to replace a character string consisting of the number of characters specified in “input3” with the character string specified in “input2” from the position specified in “input4” in the character string of the input variable “input1” and to output the replaced data to the output variable “output1”.

This is a program example for the function REPLACE.

LD program

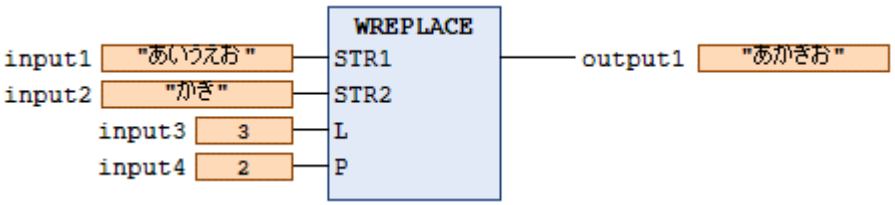


ST program

```
output1 'afge' := REPLACE(input1 'abcde', input2 'fg',
                  input3 3, input4 2);
```

This is a program example for the function WREPLACE.

LD program



ST program

```
output1 "あかきお" :=WREPLACE(input1 "あいうえお", input2 "かき", input3 3, input4 2);
```

Info.

If P = 0, L is used with a setting of -1.

3.10.9 FIND/WFIND (find text)

This is a function that searches for a specified character string and outputs the searched position.

■ Icon

■ Parameter

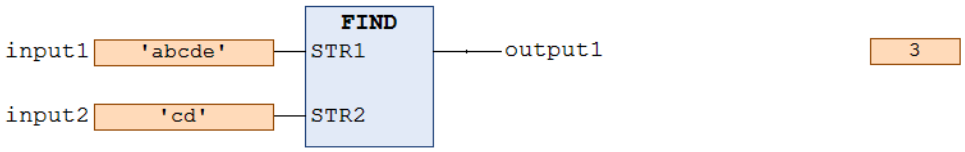
Scope	Name	Type	Description
Input	STR1	STRING(255)/ WSTRING(255)	Specify text to extract
	STR2	STRING(255)/ WSTRING(255)	Specifies the number of characters to extract
Output	FIND/ WFIND	INT	Searches for the character string specified in STR2 in the character string specified in STR1 and outputs the position from the left end.

■ Program example

This program is designed to search for the character string specified in “input2” in the character string of the input variable “input1” and to output the position from the left to the output variable ”output1”.

This is a program example for the function FIND.

LD program

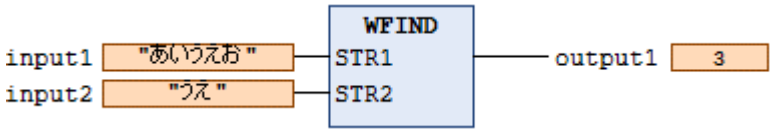


ST program

```
output1 3 := FIND(input1 'abcde', input2 'cd');
```

This is a program example for the function WFIND.

LD program



ST program

```
output1  3  :=WFIND(input1  "おいうえお"  ,input2  "うえ"  );
```

Info.

- Outputs 0 if the character string is not found.
- If the character string is found in multiple places, the position found first (the leftmost position) is output.

3.10.10 ConvertUTF16toUTF8 (UTF-16 → UTF-8)

This is a function that converts a UTF-16 character string into a UTF-8 character string. Input a target storage size (dwTargetBufferSize) based on [(input WORD type data volume x 3) + 1(end code)].

■ Icon

■ Parameter

ConvertUTF16toUTF8

Scope	Name	Type	Description
Input	sourceStart	POINTER TO WORD	Start pointer to the UTF16 character string to be converted
Input	targetStart	POINTER TO BYTE	Start pointer to the converted UTF8 character string
Input	dwTargetBufferSize	DWORD	Target storage size (unit byte)
Input	bStrictConversion	BOOL	TRUE = An error is output when data that is not convertible is present TRUE = An error is output when data that is not convertible is present
Output	ConvertUTF16toUTF8	UDINT	Error identification (refer to ConvertUTF16toUTF8 return values)

ConvertUTF16toUTF8 return values

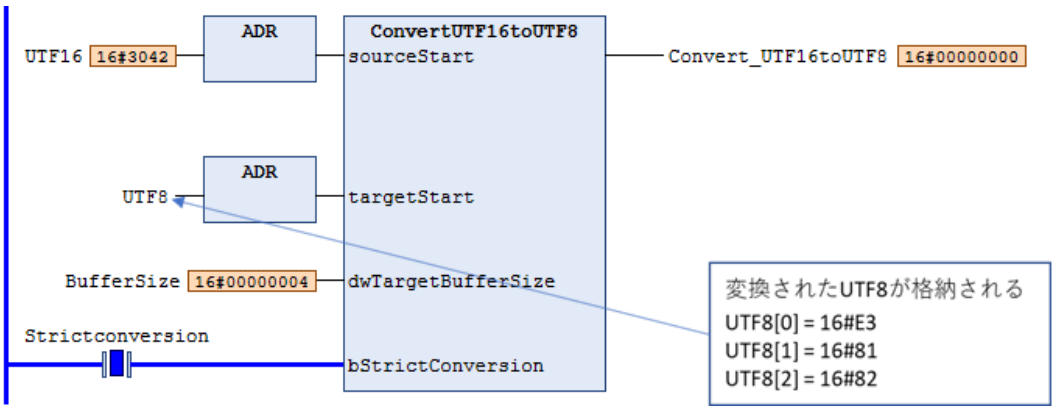
Return value	Name	Description
16#0000	ERR_OK	No error
16#0002	ERR_PARAMETER	Parameter error
16#40A1	ERR_TARGET_EXHAUSTED	Error in stored data buffer size
16#40A2	ERR_SOURCE_ILLEGAL	Data that is not convertible is present

■ Program example

This program is designed to convert a UTF16 input variable of the WORD type to a UTF8 character string.

UTF16 := 16#3042	(UTF16 that represents “あ”)
BufferSize := 4	(input WORD type data volume 1 WORD x 3) + 1 = 4
Strictconversion := TRUE	

LD program



ST program

```
Convert_UTF16toUTF8 16#00000000 :=ConvertUTF16toUTF8(sourceStart:=ADR(UTF16),
                                  targetStart:=ADR(UTF8),
                                  dwTargetBufferSize:=BufferSize 16#00000004 ,
                                  bStrictConversion:=StrictConversion TRUE );

                                  UTF8_0 16#F08DFAC2 :=ADR(UTF8[0] 16#E3 );
                                  UTF8_1 16#F08DFAC3 :=ADR(UTF8[1] 16#81 );
                                  UTF8_2 16#F08DFAC4 :=ADR(UTF8[2] 16#82 );
```

Note

- Take care of input in the Unicode disuse / not used areas. (Otherwise, data may not be output properly.)
- Always set the error detection function for data that cannot be input (bStrictConversion) to TRUE.
- If the target storage buffer size (dwTargetBuffer) is not proper, return value = 16#40A1 is output.
- In the UTF16 data to be converted, 16#0000 serves as end-of-file (EOF). Thus, the UTF16 string from the start pointer (sourceStart) data to 16#0000 is converted.

3.10.11 ConvertUTF8toUTF16(UTF-8 → UTF-16)

This is a function that converts a UTF-8 character string into a UTF-16 character string. Input a target storage size (dwTargetBufferSize) based on [(input BYTE type data volume x 2) + 2(end code)].

■ Icon

■ Parameter

ConvertUTF8toUTF16

Scope	Name	Type	Description
Input	sourceStart	POINTER TO BYTE	Start pointer of the UTF8 character string to be converted
Input	targetStart	POINTER TO WORD	Start pointer of the converted UTF16 character string
Input	dwTargetBufferSize	DWORD	Target storage size (unit byte)
Input	bStrictConversion	BOOL	TRUE = An error is output when data that is not convertible is present FALSE = An error is not output even if data that is not convertible is present
Output	ConvertUTF8toUTF16	UDINT	Error identification (refer to ConvertUTF8toUTF16 return values)

ConvertUTF8toUTF16 return values

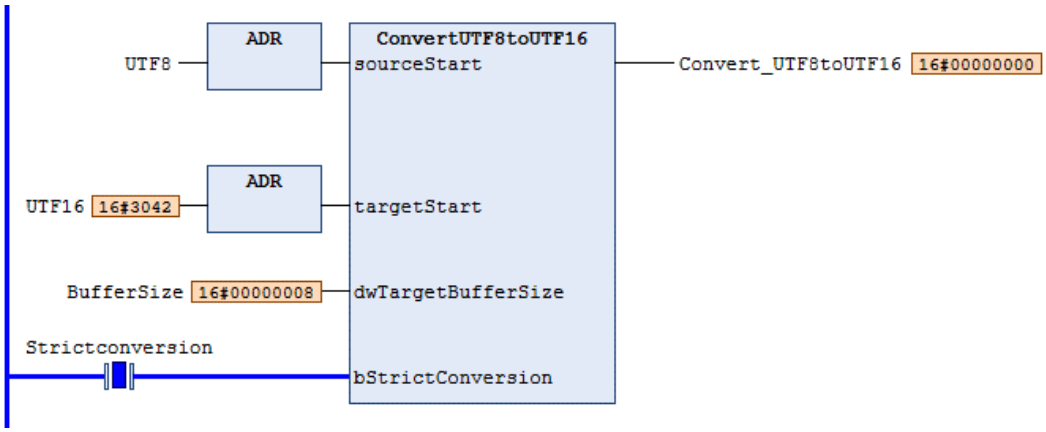
Return value	Name	Description
16#0000	ERR_OK	No error
16#0002	ERR_PARAMETER	Parameter error
16#40A1	ERR_TARGET_EXHAUSTED	Error in stored data buffer size
16#40A2	ERR_SOURCE_ILLEGAL	Data that is not convertible is present

■ Program example

This program is designed to convert a UTF8 input variable of the BYTE type to a UTF16 character string.

```
UTF8[0] := 16#E3          (UTF8 that represents “あ”)
UTF8[1] := 16#81
UTF8[2] := 16#82
BufferSize := 8           (input BYTE type data volume 3 bytes x 2) + 2 = 8
Strictconversion := TRUE
```

LD program



ST program

```
Convert_UTF8toUTF16 16#00000000 :=ConvertUTF8toUTF16(sourceStart:=ADR(UTF8),
                                    targetStart:=ADR(UTF16[0] 16#3042 ),
                                    dwTargetBufferSize:=BufferSize 16#00000008 ,
                                    bStrictConversion:=StrictConversion TRUE );

UTF8_0 16#F08DFADB :=ADR(UTF8[0] 16#E3 );
UTF8_1 16#F08DFADC :=ADR(UTF8[1] 16#81 );
UTF8_2 16#F08DFADD :=ADR(UTF8[2] 16#82 );
```

Note

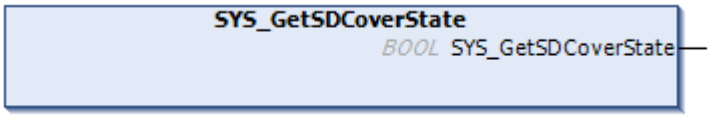
- Take care of input in the Unicode disuse / not used areas. (Otherwise, data may not be output properly.)
- Always set the error detection function for data that cannot be input (bStrictConversion) to TRUE.
- If a UTF8 character string that is not convertible is input, the 16#FFFD data is stored in the converted UTF16 data.
- With bStrictConversion = TRUE, an error (return value: 16#40A2) will occur in response to input of data that cannot be represented.
- If the target storage buffer size (dwTargetBuffer) is not proper, return value = 16#40A1 is output.
- In the UTF8 data to be converted, 16#00 serves as end-of-file (EOF). Thus, the UTF8 string from the start pointer (sourceStart) data to 16#00 is converted.

3.11 SD Memory Card Slot Instruction

3.11.1 SYS_GetSDCoverState (Get SD Card Cover Open / Close State)

This is a function that gets an open / close state of the card cover for the SD memory card slot.

■ Icon



■ Parameter

Scope	Name	Type	Description
Output	SYS_GetSDCoverState	BOOL	TRUE: The card cover is closed. FALSE: The card cover is open.

3.11.2 SYS_GetSDAccessRdy (Get SD Card Access Ready State)

This is a function block that gets the state whether an access to the SD memory card is allowed.

■ Icon

■ Parameter

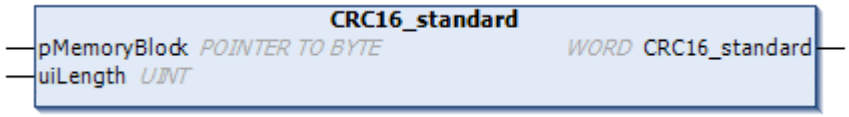
Scope	Name	Type	Description
Output	SYS_GestSDAccessRdy	BOOL	TRUE: Access to the SD memory card is enabled. FALSE: Access to the SD memory card is disabled.

3.12 CRC operation instructions

3.12.1 MEM.CRC16_standard (CRC16)

This is a function that calculates the CRC16 checksum.

■ **Icon**



■ **Parameter**

CRC16_standard

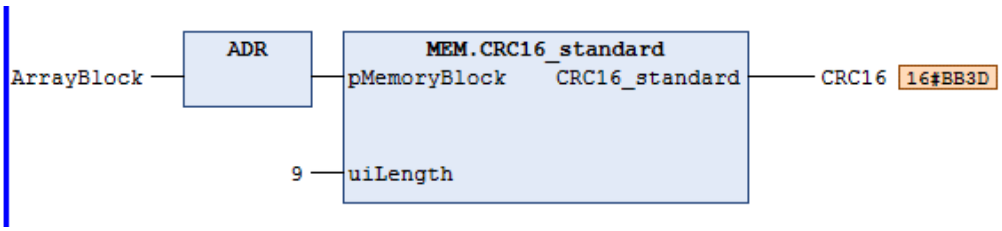
Scope	Name	Type	Description
Input	pMemoryBlock	POINTER TO BYTE	Start pointer to the memory block to calculate the checksum
Input	uiLength	UINT	Number of bytes to be calculated Effective range: 10#1 to 10#65534
Output	CRC16_standard	WORD	Calculated CRC16 result

■ **Program example**

This program is designed to calculate the CRC16 checksum of 9 bytes (uiLength) in memory block data (MemoryBlock) and output the result (16#BB3D) to CRC16_standard.

```
ArrayBlock := ARRAY [0..8] OF BYTE :   = [16#31,16#32,16#33,16#34,16#35,16#36,16#37,16#38,16#39]
                                        (= [STRING(10) := '123456789'])
uiLength :=9
```

LD program



ST program

```
CRC16 16#BB3D := MEM.CRC16_standard(ADR(ArrayBlock), 9);
```

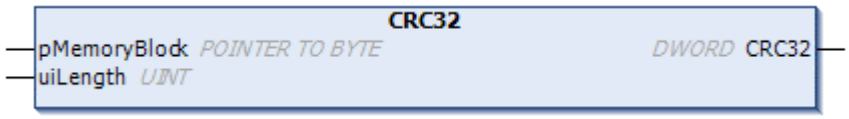
Note

- The function does not operate properly if the number of bytes to calculate (uiLength) is set to 65535 bytes or more. Thus, do not use that byte size.

3.12.2 MEM.CRC32(CRC32)

This is a function that calculates the CRC32 checksum.

■ Icon



■ Parameter

CRC32

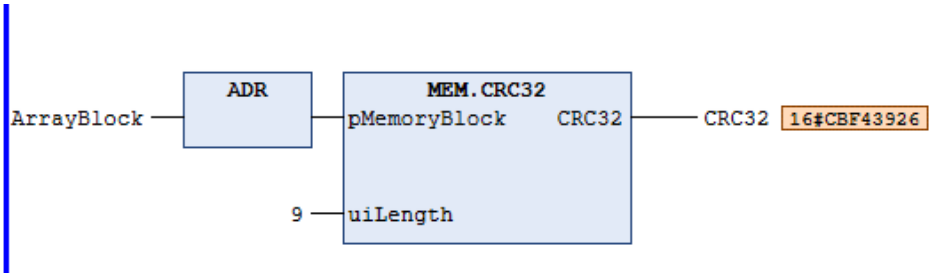
Scope	Name	Type	Description
Input	pMemoryBlock	POINTER TO BYTE	Start pointer to the memory block to calculate the checksum
Input	uiLength	UINT	Number of bytes to be calculated Effective range: 10#1 to 10#65534
Output	CRC32	DWORD	Calculated CRC32 result

■ Program example

This program is designed to calculate the CRC32 checksum of 9 bytes (uiLength) in memory block data (MemoryBlock) and output the result (16#CBF43926) to CRC32.

```
ArrayBlock := ARRAY [0..8] OF BYTE :   = [16#31,16#32,16#33,16#34,16#35,16#36,16#37,16#38,16#39]
                                        (= [STRING(10) := '123456789'])

uiLength :=9
```

LD program



ST program

```
CRC32 16#CBF43926 := MEM.CRC32(ADR(ArrayBlock),9);
```

Note

- The function does not operate properly if the number of bytes to calculate (uiLength) is set to 65535 bytes or more. Thus, do not use that byte size.

3.13 System Time Instructions

A length of time that has been elapsed since the start of the GM1 controller can be acquired.

3.13.1 SysTimeGetMs(Get System Time in units of milliseconds)

This is a function used to output a length of time that has been elapsed since the start of the GM1 controller in units of milliseconds. The power of the GM1 controller can be turned OFF to reset the value.

■ **Icon**

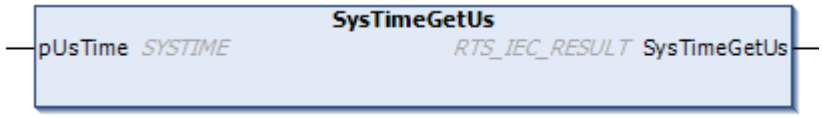
■ **Parameter**

Scope	Name	Type	Default value	Description
Output	SysTimeGetMs	UDINT	0	System time (unit: ms)

3.13.2 SysTimeGetUs(Get System Time in units of microseconds)

This is a function used to output a length of time that has been elapsed since the start of the GM1 controller in units of microseconds. The power of the GM1 controller can be turned OFF to reset the value.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Default value	Description
Input / output	pUsTime	SYSTIME(=ULINT)	0	System time (unit: μs)
Output	SysTimeGetUs	SysTimeCore.RTS_IEC_RESULT	0	An error ID is output.

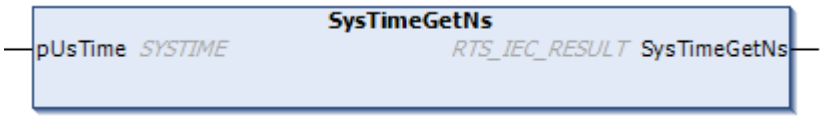
Info.

- For pUsTime, elapsed time of ULINT type is acquired.

3.13.3 SysTimeGetNs(Get System Time in units of nanoseconds)

This is a function used to output a length of time that has been elapsed since the start of the GM1 controller in units of nanoseconds. The power of the GM1 controller can be turned OFF to reset the value.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	pUsTime	SYSTIME(=ULINT)	0	System time (unit: ns)
Output	SysTimeGetNs	SysTimeCore.RTS_IEC_RESULT	0	An error ID is output.

Info.

- For pUsTime, elapsed time of ULINT type is acquired.

4 Function Blocks (Basic Instructions)

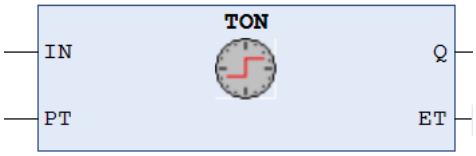
4.1 Timer Instructions

Timer instructions can be used to perform timer operations.

4.1.1 TON (Timer ON)

This is a function block (FB) that starts the timer when the input becomes TRUE. After a specified time elapses, the output becomes TRUE.

■ Icon



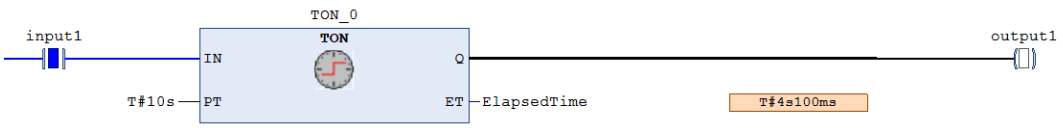
■ Parameter

Scope	Name	Type	Description
Input	IN	BOOL	Starts the timer when FALSE becomes TRUE and the timer continues counting while it remains TRUE. Resets the timer when it becomes FALSE.
	PT	TIME	Specifies the timer time.
Output	Q	BOOL	Outputs TRUE when the time specified in the input argument PT elapses.
	ET	TIME	Specifies the elapsed time of the timer.

■ Program example

This program is designed to start the timer when the input variable “input1” becomes TRUE and, after an elapse of 10 seconds, to cause the output variable “output1” to become TRUE. The instance name is TON_0.

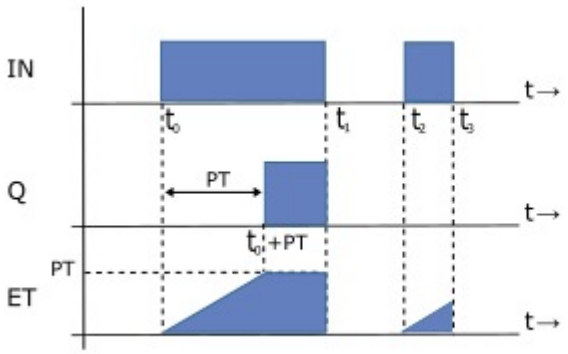
LD program



ST program

```
TON_0(IN TRUE := input1 TRUE, PT T#10s := T#10S,
      Q FALSE => output1 FALSE, ET T#3s659ms => ElapsedTime T#3s659ms );
```

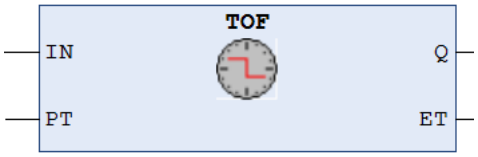
■ **Time-sequence diagram**



4.1.2 TOF (Timer OFF)

This is a function block (FB) that starts the timer when the input becomes FALSE. After a specified time elapses, the output becomes FALSE.

■ **Icon**



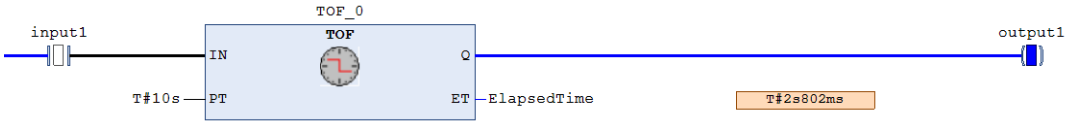
■ **Parameter**

Scope	Name	Type	Description
Input	IN	BOOL	Starts the timer when TRUE becomes FALSE and the timer continues counting while it remains FALSE. Resets the timer when it becomes TRUE.
	PT	TIME	Specifies the timer time.
Output	Q	BOOL	Outputs FALSE when the time specified in the input argument PT elapses.
	ET	TIME	Specifies the elapsed time of the timer.

■ **Program example**

This program is designed to start the timer when the input variable “input1” changes from TRUE to FALSE and, after an elapse of 10 seconds, to cause the output variable “output1” to become FALSE. The instance name is TOF_0.

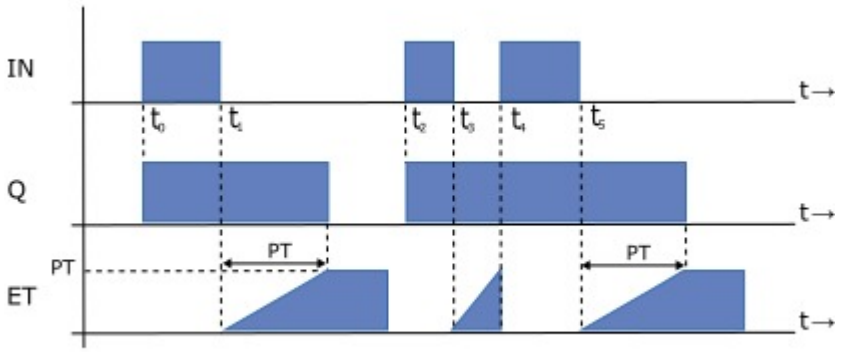
LD program



ST program

```
TOF_0(IN FALSE := input1 FALSE,
      PT T#10s := T#10S,
      Q TRUE => output1 TRUE,
      ET T#2s113ms => ElapsedTime T#2s113ms);
```

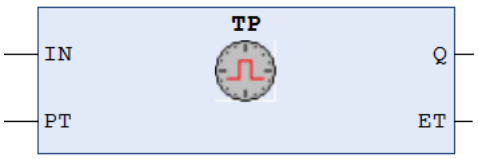
■ **Time-sequence diagram**



4.1.3 TP (Timer Pulse)

This is a function block that starts the timer at the rising edge. The output remains TRUE while the timer keeps counting. After a specified time elapses, the output becomes FALSE.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Description
Input	IN	BOOL	Starts the timer when FALSE changes to TRUE (rising edge).

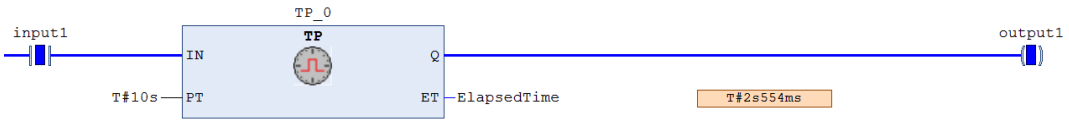
Scope	Name	Type	Description
			Resets the timer when the timer expires and TRUE changes to FALSE .
	PT	TIME	Specifies the timer time.
Output	Q	BOOL	Outputs TRUE from when the timer is started until when the time specified in the input argument PT elapses. Outputs FALSE after the specified time elapses.
	ET	TIME	Specifies the elapsed time of the timer.

■ Program example

This program is designed to start the timer when the input variable “input1” changes from FALSE to TRUE and, during the time from when the timer is started to when the timer expires (for 10 seconds), to cause the output variable “output1” to remain TRUE.

The instance name is TP_0.

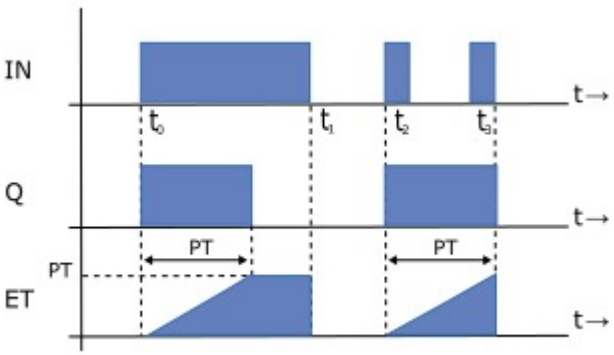
LD program



ST program

```
TP_0(IN TRUE := input1 TRUE,
     PT T#10s := T#10S,
     Q TRUE => output1 TRUE,
     ET T#2s822ms => ElapsedTime T#2s822ms);
```

■ Time-sequence diagram



4.1.4 RTC (Realtime Clock)

This is a function block that starts counting time at the rising edge starting from the specified date and time. The output remains TRUE while the time counting continues. After a specified time elapses, the output becomes FALSE.

■ Icon

■ Parameter

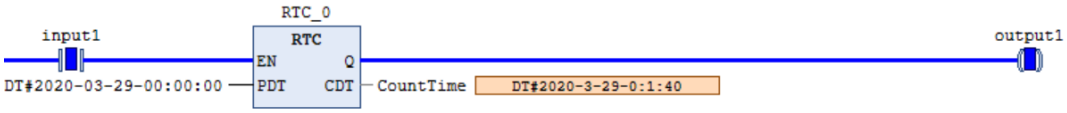
Scope	Name	Type	Description
Input	EN	BOOL	Starts counting time from the date and time specified in the input argument PDT when FALSE changes to TRUE (rising edge). When TRUE changes to FALSE, DT#1970-01-01-00:00:00 is set in the output argument CDT.
	PDT	DATE_AND_TIME	Date and time when time counting starts
Output	Q	BOOL	Outputs TRUE while time counting continues.
	CDT	DATE_AND_TIME	Outputs the time count time from the date and time specified in the input argument PDT.

■ Program example

This program is designed to start counting time, starting from 0 o'clock of March 29, 2020, when the input variable “input1” changes from FALSE to TRUE, and, to cause the output variable “output1” to remain TRUE while time counting continues.

The instance name is RTC_0.

LD program



ST program

```
RTC_0(EN TRUE := input1 TRUE,
      PDT DT#2020-3-29-0:0:0 := DT#2020-03-29-00:00:00,
      Q TRUE => output1 TRUE,
      CDT DT#2020-3-29-0:1:31 => CountTime DT#2020-3-29-0:1:31);
```

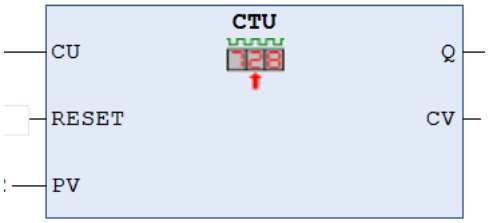
4.2 Counter Instructions

Counter instructions can be used to perform counter operations.

4.2.1 CTU (Up Counter)

This is a function block that increments the counter value by 1 every time the rising edge occurs.

■ Icon



■ Parameter

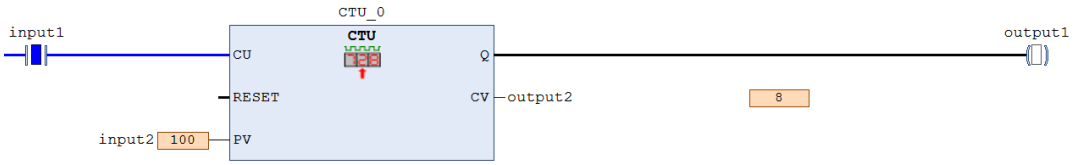
Scope	Name	Type	Description
Input	CU	BOOL	Increments the value of the output argument CV by 1 when FALSE changes to TRUE (rising edge).
	RESET	BOOL	If TRUE, 0 is set in the output argument CV.
	PV	WORD	Target value of CV
Output	Q	BOOL	Outputs TRUE when the CV value reaches the PV value.
	CV	WORD	Outputs the current counter value.

■ Program example

This program is designed to increment the value of the output variable “output2” by 1 every time the input variable “input1” changes from FALSE to TRUE. The program is designed to cause the output variable “output1” to change to TRUE when the value (100) of the input variable “input2” is counted up.

The instance name is CTU_0.

LD program



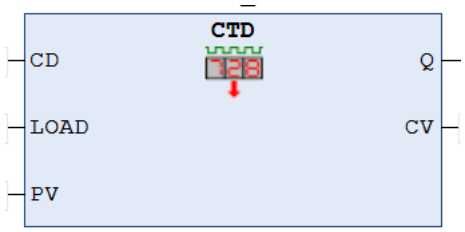
ST program

```
CTU_0(
    CU TRUE := input1 TRUE,
    RESET:= ,
    PV 100 := 100,
    Q FALSE => output1 FALSE,
    CV 8 => output2 8 );
```

4.2.2 CTD (Down Counter)

This is a function block that decrements the counter value by 1 every time the rising edge occurs.

■ Icon



■ Parameter

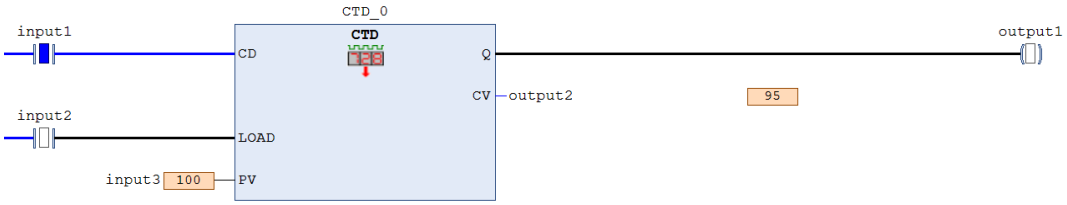
Scope	Name	Type	Description
Input	CD	BOOL	Decrements the value of the output argument CV by 1 when FALSE changes to TRUE (rising edge).
	LOAD	BOOL	If TRUE, the value specified in PV is set in the output argument CV.
	PV	WORD	Initial value of the counter value
Output	Q	BOOL	Outputs TRUE when the CV value becomes 0.
	CV	WORD	Outputs the current counter value.

■ Program example

This program is designed to decrement the value of the output variable “output2” by 1 every time the input variable “input1” changes from FALSE to TRUE, and to cause the output variable “output1” to change to TRUE when the value becomes 0. The initial value (100) to count down from is specified in the input variable “input3”.

The instance name is CTD_0.

LD program



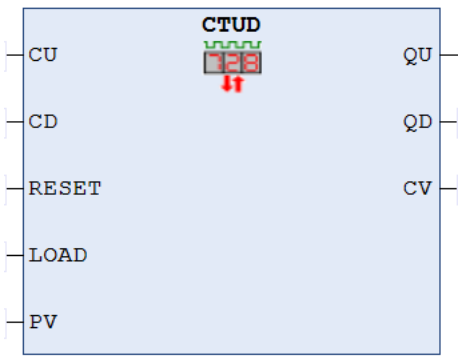
ST program

```
CTD_0(
    CD TRUE := input1 TRUE,
    LOAD FALSE := input2 FALSE,
    PV 100 := input3 100,
    Q FALSE => output1 FALSE,
    CV 95 => output2 95);
```

4.2.3 CTUD (Up-down Counter)

This is a function block that increments or decrements the counter value by 1 every time the rising edge occurs.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	CU	BOOL	Increments the value of the output argument CV by 1 when FALSE changes to TRUE (rising edge).
	CD	BOOL	Decrements the value of the output argument CV by 1 when FALSE changes to TRUE (rising edge).
	RESET	BOOL	If TRUE, 0 is set in the output argument CV.
	LOAD	BOOL	If TRUE, the value specified in PV is set in the output argument CV.
	PV	WORD	Initial value of the counter value
Output	QU	BOOL	Outputs TRUE when the CV value reaches the PV value.

Scope	Name	Type	Description
	QD	BOOL	Outputs TRUE when the CV value becomes 0.
	CV	WORD	Outputs the current counter value.

■ Program example

Every time the input variable “input1” changes from FALSE to TRUE, the value of the output variable “output3” is incremented by 1.

Every time the input variable “input2” changes from FALSE to TRUE, the value of the output variable “output3” is decremented by 1

When the output variable “output3” becomes greater than or equal to the input variable “input5”, the output variable “output1” becomes TRUE.

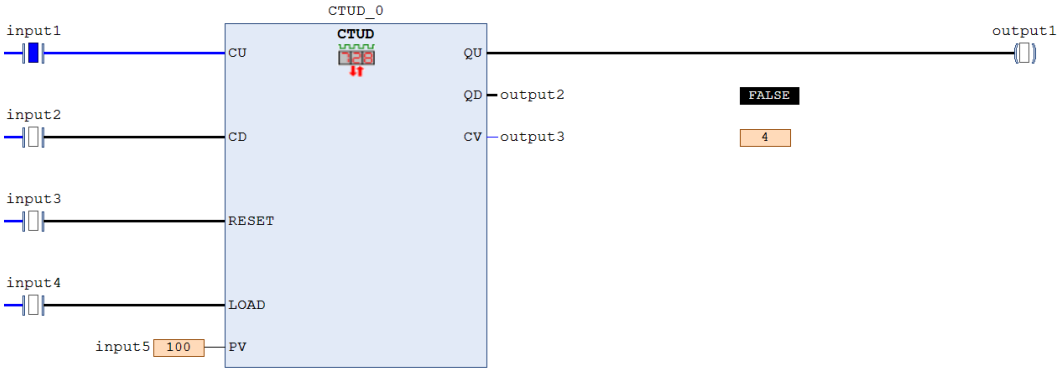
When the output variable “output3” becomes 0, the output variable “output2” becomes TRUE.

When the input variable “input3” becomes TRUE, the output variable “output3” becomes 0.

When the input variable “input4” becomes TRUE, the value (100) of the input variable “input5” is set in the output variable “output3”.

The instance name is CTUD_0.

LD program



ST program

```
CTUD_0(
    CU TRUE := input1 TRUE,
    CD FALSE := input2 FALSE,
    RESET FALSE := input3 FALSE,
    LOAD FALSE := input4 FALSE,
    PV 100 := input5 100,
    QU FALSE => output1 FALSE,
    QD FALSE => output2 FALSE,
    CV 3 => output3 3);
```

4.3 Edge Detection Instructions

Edge detection instructions can be used to perform edge detection.

4.3.1 R_TRIG (Rising Edge Detection)

This is a function block that detects a rising edge.

■ Icon

■ Parameter

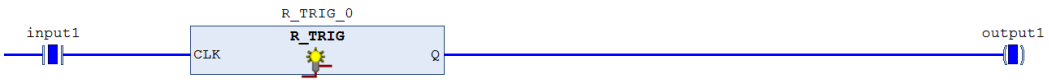
Scope	Name	Type	Description
Input	CLK	BOOL	Input that detects a rising edge
Output	Q	BOOL	Outputs TRUE for one cycle only when a rising edge is detected in the input argument CLK.

■ Program example

When the input variable “input1” changes from FALSE to TRUE, the output variable “output1” becomes TRUE for one cycle only.

The instance name is R_TRIG_0.

LD program



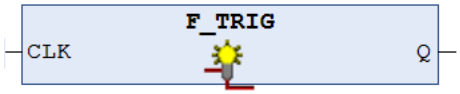
ST program

```
R_TRIG_0(
    CLK TRUE := input1 TRUE,
    Q TRUE => output1 TRUE);
```

4.3.2 F_TRIG (Falling Edge Detection)

This is a function block that detects a falling edge.

■ Icon



■ Parameter

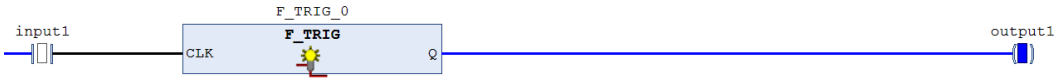
Scope	Name	Type	Description
Input	CLK	BOOL	Input that detects a falling edge
Output	Q	BOOL	Outputs TRUE for one cycle only when a falling edge is detected in the input argument CLK.

■ Program example

When the input variable “input1” changes from FALSE to TRUE, the output variable “output1” becomes TRUE for one cycle only.

The instance name is F_TRIG_0.

LD program



ST program

```
F_TRIG_0(
    CLK FALSE := input1 FALSE,
    Q TRUE => output1 TRUE );
```

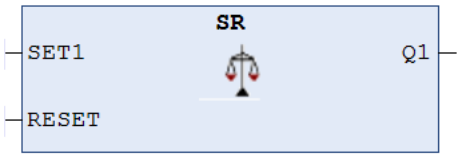
4.4 Bistable Circuit Instructions

Bistable circuit instructions can be used to perform edge detection.

4.4.1 SR (Set-priority Bistable Circuit)

This is a function block that realizes a bistable (flip-flop) circuit. The priority is given to the set input.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	SET1	BOOL	Specifies the set input for a bistable circuit.
	RESET	BOOL	Specifies the reset input for a bistable circuit.
Output	Q1	BOOL	When the input argument SET1 becomes TRUE, outputs and holds TRUE. When the input argument RESET becomes TRUE, outputs and holds FALSE. When both SET1 and RESET1 are TRUE, outputs and holds TRUE.

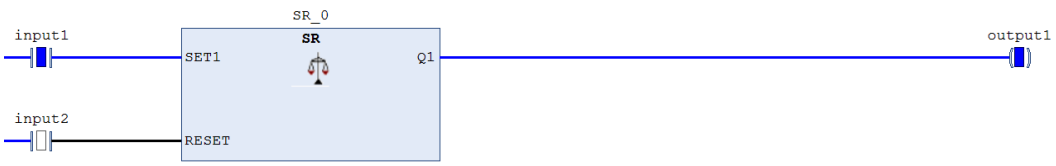
■ Program example

When the input variable “input1” becomes TRUE, the output variable “output1” becomes TRUE.
Even if the input variable “input1” becomes FALSE, “output1” remains TRUE.

When the input variable “input1” is FALSE and if input variable “input2” becomes TRUE, the output variable “output1” becomes FALSE.

The instance name is SR_0.

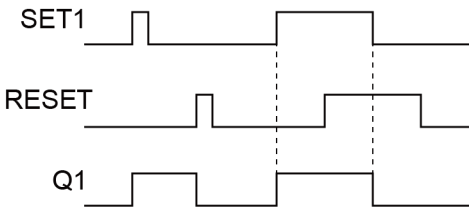
LD program



ST program

```
SR_0(
    SET1 TRUE := input1 TRUE,
    RESET FALSE := input2 FALSE,
    Q1 TRUE => output1 TRUE);
```

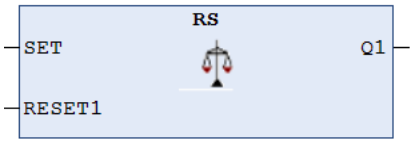
■ **Time-sequence diagram**



4.4.2 RS (Reset-priority Bistable Circuit)

This is a function block that realizes a bistable (flip-flop) circuit. The priority is given to the reset input.

■ **Icon**



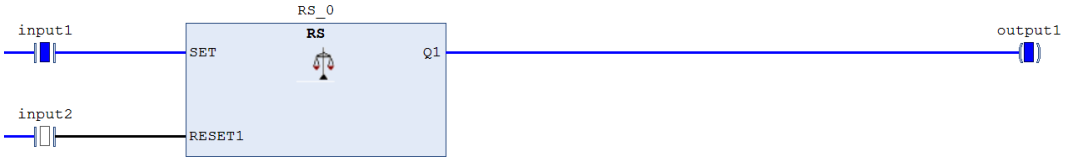
■ **Parameter**

Scope	Name	Type	Description
Input	SET1	BOOL	Specifies the set input for a bistable circuit.
	RESET	BOOL	Specifies the reset input for a bistable circuit.
Output	Q1	BOOL	When the input argument SET1 becomes TRUE, outputs and holds TRUE. When the input argument RESET becomes TRUE, outputs and holds FALSE. When both SET1 and RESET1 are TRUE, outputs and holds FALSE.

■ **Program example**

When the input variable “input1” becomes TRUE, the output variable “output1” becomes TRUE.
Even if the input variable “input1” becomes FALSE, “output1” remains TRUE.
When the input variable “input1” is FALSE and if the input variable “input2” becomes TRUE, the output variable “output1” becomes FALSE.
The instance name is RS_0.

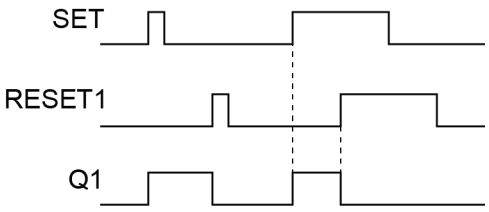
LD program



ST program

```
RS_0(
    SET TRUE := input1 TRUE,
    RESET1 FALSE := input2 FALSE,
    Q1 TRUE => output1 TRUE);
```

■ **Time-sequence diagram**

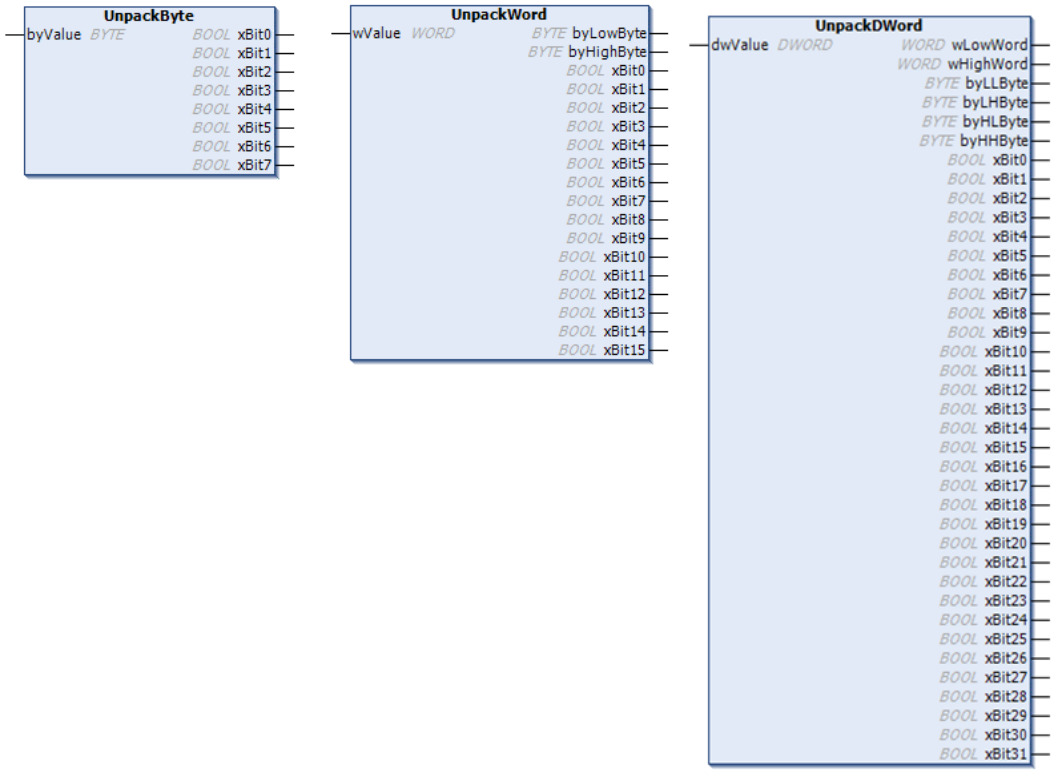


4.5 Data Type Conversion Instructions

4.5.1 MEM.Unpack** (BYTE/WORD/DWORD to Bit Data Conversion)

This is a function that unpacks input BYTE-, WORD-, or DWORD-type data to data in bits and outputs the data.

■ **Icon**



■ **Parameter**

UnpackByte

Scope	Name	Type	Description
Input	byValue	BYTE	BYTE type data to be unpacked
Output	xBit0 to xBit7	BOOL	A value representing the input value unpacked in bits

UnpackWord

Scope	Name	Type	Description
Input	wValue	BYTE	WORD type data to be unpacked
Output	byHighByte	BYTE	High byte unpacked from the input value

Scope	Name	Type	Description
Output	byLowByte	BYTE	Low byte unpacked from the input value
Output	xBit0 to xBit15	BYTE	A value representing the input value unpacked in bits

UnpackDWord

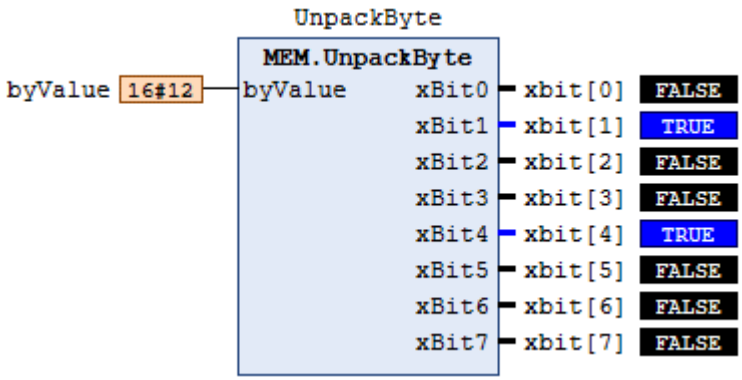
Scope	Name	Type	Description
Input	dwValue	DWORD	DWORD type data to be unpacked
Output	wHighWord	WORD	High WORD unpacked from the input value
Output	wLowWord	WORD	Low WORD unpacked from the input value
Output	byHHByte	BYTE	HH byte unpacked from the input value
Output	byHLByte	BYTE	HL byte unpacked from the input value
Output	byLHByte	BYTE	LH byte unpacked from the input value
Output	byLLByte	BYTE	LL byte unpacked from the input value
Output	xBit0 to xBit31	BOOL	A value representing the input value unpacked in bits

■ Program example 1

This program is designed to unpack the byValue input variable of the BYTE type to pieces of data of the BOOL type and outputs them to the xBit0 to xBit7 output variables of the BOOL type.

byValue := 16#12

LD program



ST program

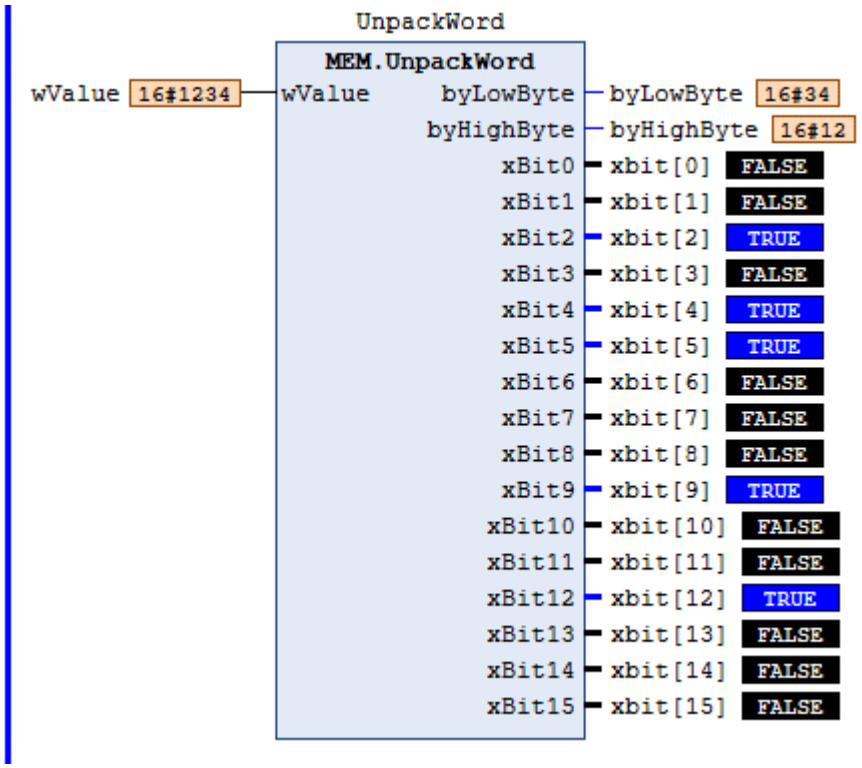
```
UnpackByte(byvalue 16#12 :=byValue 16#12 ,
           xBit7 FALSE =>xbit[7] FALSE ,
           xBit6 FALSE =>xbit[6] FALSE ,
           xBit5 FALSE =>xbit[5] FALSE ,
           xBit4 TRUE =>xbit[4] TRUE ,
           xBit3 FALSE =>xbit[3] FALSE ,
           xBit2 FALSE =>xbit[2] FALSE ,
           xBit1 TRUE =>xbit[1] TRUE ,
           xBit0 FALSE =>xbit[0] FALSE );
```

■ Program example 2

This program is designed to unpack the wValue input variable of the WORD type to pieces of data of the BOOL and BYTE types and outputs them to the xBit0 to xBit15 output variables of the BOOL type and the byHighByte and byLowByte output variables of the BYTE type.

wValue := 16#1234

LD program



ST program

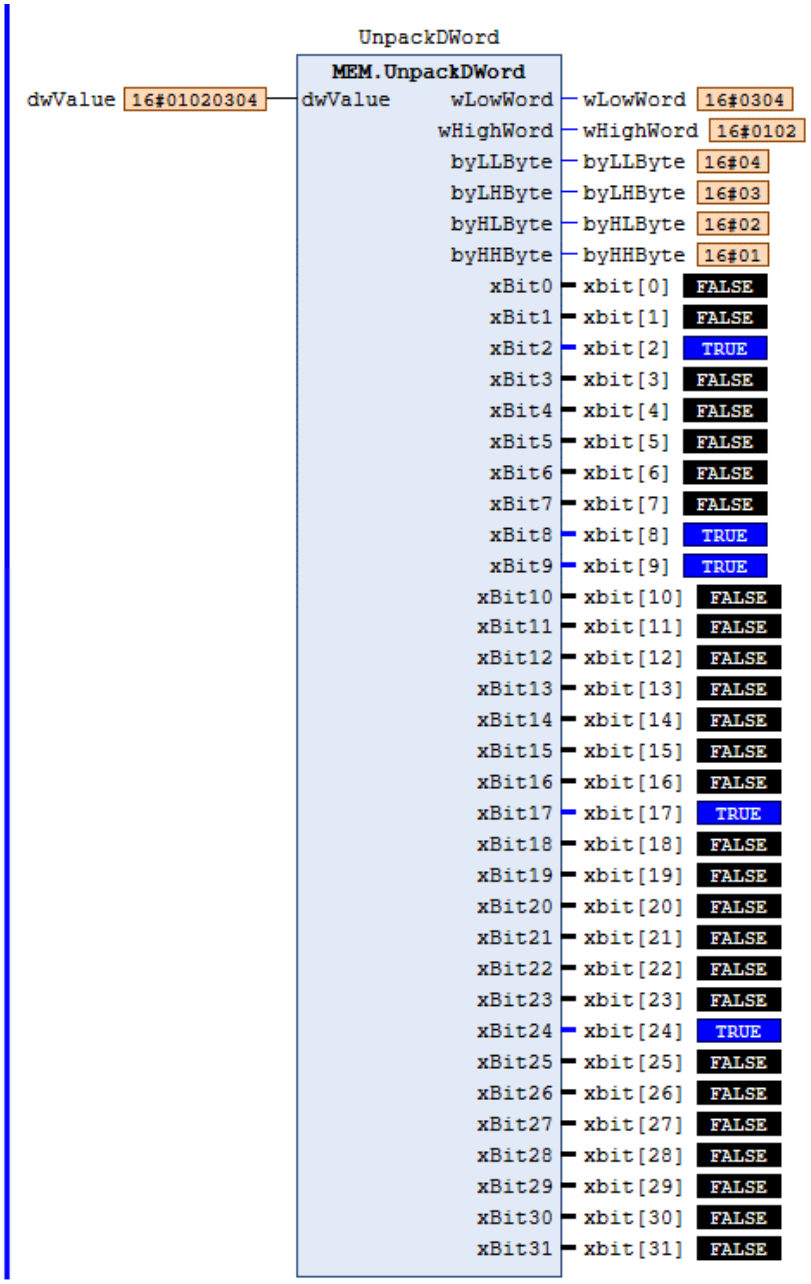
```
UnpackWord(wValue 16#1234 :=wValue 16#1234 ,
           byHighByte 16#12 =>byHighByte 16#12 ,byLowByte 16#34 =>byLowByte 16#34 ,
           xBit15 FALSE =>xbit[15] FALSE ,
           xBit14 FALSE =>xbit[14] FALSE ,
           xBit13 FALSE =>xbit[13] FALSE ,
           xBit12 TRUE =>xbit[12] TRUE ,
           xBit11 FALSE =>xbit[11] FALSE ,
           xBit10 FALSE =>xbit[10] FALSE ,
           xBit9 TRUE =>xbit[9] TRUE ,
           xBit8 FALSE =>xbit[8] FALSE ,
           xBit7 FALSE =>xbit[7] FALSE ,
           xBit6 FALSE =>xbit[6] FALSE ,
           xBit5 TRUE =>xbit[5] TRUE ,
           xBit4 TRUE =>xbit[4] TRUE ,
           xBit3 FALSE =>xbit[3] FALSE ,
           xBit2 TRUE =>xbit[2] TRUE ,
           xBit1 FALSE =>xbit[1] FALSE ,
           xBit0 FALSE =>xbit[0] FALSE );
```

■ Program example 3

This program is designed to unpack the dwValue input variable of the DWORD type to pieces of data of the BOOL, BYTE and WORD types and outputs them to the xBit0 to xBit31 output variables of the BOOL type, the wHighWord and wLowWord output variables of the WORD type, and the byHHByte, byHLByte, byLHByte, and byLLByte output variables of the BYTE type.

dwValue := 16#01020304

LD program


UnpackDWord
MEM.UnpackDWord
dwValue 16#01020304
dwValue
wLowWord wLowWord 16#0304
wHighWord wHighWord 16#0102
byLLByte byLLByte 16#04
byLHByte byLHByte 16#03
byHLByte byHLByte 16#02
byHHByte byHHByte 16#01
xBit0 xbit[0] FALSE
xBit1 xbit[1] FALSE
xBit2 xbit[2] TRUE
xBit3 xbit[3] FALSE
xBit4 xbit[4] FALSE
xBit5 xbit[5] FALSE
xBit6 xbit[6] FALSE
xBit7 xbit[7] FALSE
xBit8 xbit[8] TRUE
xBit9 xbit[9] TRUE
xBit10 xbit[10] FALSE
xBit11 xbit[11] FALSE
xBit12 xbit[12] FALSE
xBit13 xbit[13] FALSE
xBit14 xbit[14] FALSE
xBit15 xbit[15] FALSE
xBit16 xbit[16] FALSE
xBit17 xbit[17] TRUE
xBit18 xbit[18] FALSE
xBit19 xbit[19] FALSE
xBit20 xbit[20] FALSE
xBit21 xbit[21] FALSE
xBit22 xbit[22] FALSE
xBit23 xbit[23] FALSE
xBit24 xbit[24] TRUE
xBit25 xbit[25] FALSE
xBit26 xbit[26] FALSE
xBit27 xbit[27] FALSE
xBit28 xbit[28] FALSE
xBit29 xbit[29] FALSE
xBit30 xbit[30] FALSE
xBit31 xbit[31] FALSE

ST program

```
UnpackDWord(dwValue 16#01020304 :=dwValue 16#01020304 ,
            wHighWord 16#0102 =>wHighWord 16#0102 ,wLowWord 16#0304 =>wLowWord 16#0304 ,
            byHHByte 16#01 =>byHHByte 16#01 ,byHLByte 16#02 =>byHLByte 16#02 ,
            byLHByte 16#03 =>byLHByte 16#03 ,byLLByte 16#04 =>byLLByte 16#04 ,
            xBit31 FALSE =>xbit[31] FALSE ,
            xBit30 FALSE =>xbit[30] FALSE ,
            xBit29 FALSE =>xbit[29] FALSE ,
            xBit28 FALSE =>xbit[28] FALSE ,
            xBit27 FALSE =>xbit[27] FALSE ,
            xBit26 FALSE =>xbit[26] FALSE ,
            xBit25 FALSE =>xbit[25] FALSE ,
            xBit24 TRUE =>xbit[24] TRUE ,
            xBit23 FALSE =>xbit[23] FALSE ,
            xBit22 FALSE =>xbit[22] FALSE ,
            xBit21 FALSE =>xbit[21] FALSE ,
            xBit20 FALSE =>xbit[20] FALSE ,
            xBit19 FALSE =>xbit[19] FALSE ,
            xBit18 FALSE =>xbit[18] FALSE ,
            xBit17 TRUE =>xbit[17] TRUE ,
            xBit16 FALSE =>xbit[16] FALSE ,
            xBit15 FALSE =>xbit[15] FALSE ,
            xBit14 FALSE =>xbit[14] FALSE ,
            xBit13 FALSE =>xbit[13] FALSE ,
            xBit12 FALSE =>xbit[12] FALSE ,
            xBit11 FALSE =>xbit[11] FALSE ,
            xBit10 FALSE =>xbit[10] FALSE ,
            xBit9 TRUE =>xbit[9] TRUE ,
            xBit8 TRUE =>xbit[8] TRUE ,
            xBit7 FALSE =>xbit[7] FALSE ,
            xBit6 FALSE =>xbit[6] FALSE ,
            xBit5 FALSE =>xbit[5] FALSE ,
            xBit4 FALSE =>xbit[4] FALSE ,
            xBit3 FALSE =>xbit[3] FALSE ,
            xBit2 TRUE =>xbit[2] TRUE ,
            xBit1 FALSE =>xbit[1] FALSE ,
            xBit0 FALSE =>xbit[0] FALSE );
```

4.6 Data manipulation instructions

You can process the data using data manipulation instructions.

4.6.1 LIN_TRAFO (linear conversion)

Convert one range of numbers to another linearly.

It is calculated by the following formula.

OUT = OUT_MIN ＋((IN－IN_MIN)×(OUT_MAX-OUT_MIN)/(IN_MAX-IN_MIN))

Set the setting value to IN_MIN ≤ IN ≤ IN_MAX, and set the maximum input value (IN_MAX) to be not equal to the minimum input value (IN_MIN).

■ **Icon**

■ **Parameter**

Scope	Name	Type	Default value	Description
Input	IN	REAL	0	Input value
	IN_MIN	REAL	0	Minimum input range
	IN_MAX	REAL	0	Maximum input range
	OUT_MIN	REAL	0	Minimum output range
	OUT_MAX	REAL	0	Maximum output range
Output	OUT	REAL	0	Converted output value
	ERROR	BOOL	FALSE	TRUE：IN_MIN=IN_MAX Or IN is outside of the input range (IN<IN_MIN or IN >_IN_MAX)

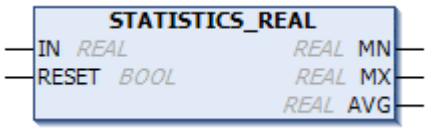
Note

- Do not set the input range (IN_MAX-IN_MIN) ≥ REAL maximum value (3.402823E+38)
- Do not set the output range (OUT_MAX-OUT_MIN) ≥ REAL maximum value (3.402823E+38).
- Do not set the same range (IN_MIN = OUT_MIN and IN_MAX = OUT_MAX).
- The REAL type is divided into the mantissa and an exponent, so if you increase the input range and the output range, an error will occur.

4.6.2 STATISTICS_REAL (maximum, minimum, and average input values)

Acquire the maximum, minimum, and average values of the input data (REAL type). The input value is added and updated for each execution timing. Resetting will return the maximum, minimum, and average values to their default values.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	IN	REAL	0	Input value
	RESET	BOOL	FALSE	TRUE: Reset Set MN, MX, AVG to the default values
Output	MN	REAL	-3.402823466 E+38	Minimum value
	MX	REAL	3.402823466 E+38	Maximum value
	AVG	REAL	0	Average Value

4.6.3 LIMITALARM (Monitoring of input values)

Monitor whether the input value is between LOW (lower limit) and HIGH (upper limit)

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	IN	INT	0	Input value
	HIGH	INT	0	Upper limit value(Note 1)
	LOW	INT	0	Lower limit value(Note 1)
Output	O	BOOL	FALSE	TRUE: Input value (IN) is greater than HIGH, FALSE: IN is equal to or less than HIGH
	U	BOOL	FALSE	TRUE: Input value (IN) is less than LOW, FALSE: IN is equal to or greater than LOW
	IL	BOOL	FALSE	TRUE: Input value (IN) is within the range of LOW to HIGH FALSE: If either the output argument O or U is TRUE

(Note 1) Set LOW < HIGH to use.

4.7 Other instructions

You can use other instructions

4.7.1 BLINK (output of blinking signal)

Switch the output argument OUT to TRUE or FALSE according to the setting time.

■ **Icon**

■ **Parameter**

Scope	Name	Type	Default value	Description
Input	ENABLE	BOOL	FALSE	TRUE: Start the pulse output. FALSE: The pulse output is stopped and the output OUT is maintained.(Note 1)
	TIMELOW	TIME	T#0ms	Time that is FALSE(Note 2)
	TIMEHIGH	TIME	T#0ms	Time that is TRUE(Note 2)
Output	OUT	BOOL	FALSE	Switch between TRUE and FALSE at the specified time. If the output at the start of the pulse is FALSE, it starts with TRUE, and if it is TRUE, it starts with FALSE.

(Note 1) When pulse is stopped (ENABLE = FALSE), the value of the output value OUT at that time is maintained.

(Note 2) When executing with the default value (T#0ms), the timing at which the pulse signal OUT switches is 1 scan.

(MEMO)

5 Motion Control Function Blocks (Single Axis Control)

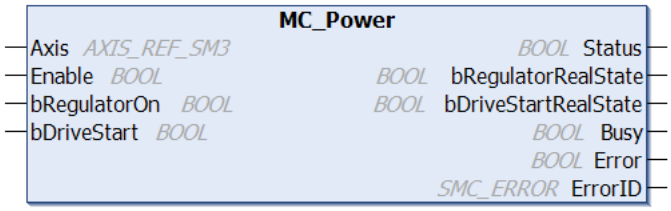
This section describes motion control function blocks for the single axis.

5.1 Servo ON

5.1.1 MC_Power (motion readiness)

This is a function block (FB) that controls the axis readiness for motion.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Enable	BOOL	FALSE	TRUE: Execution of the FB is enabled.
	bRegulatorOn	BOOL	FALSE	TRUE: Switches power to the axis to On. FALSE: Switches power to the axis to Off.
	bDriveStart	BOOL	FALSE	TRUE: Quick stop is disabled. For the GM1 Controller, fix to TRUE.
Output	Status	BOOL	FALSE	TRUE: The axis is ready to move.
	bRegulatorRealState	BOOL	FALSE	TRUE: Power to the axis has been switched to On.
	bDriveStartRealState	BOOL	FALSE	TRUE: Operation is not stopped by quick stop.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ Detail of function

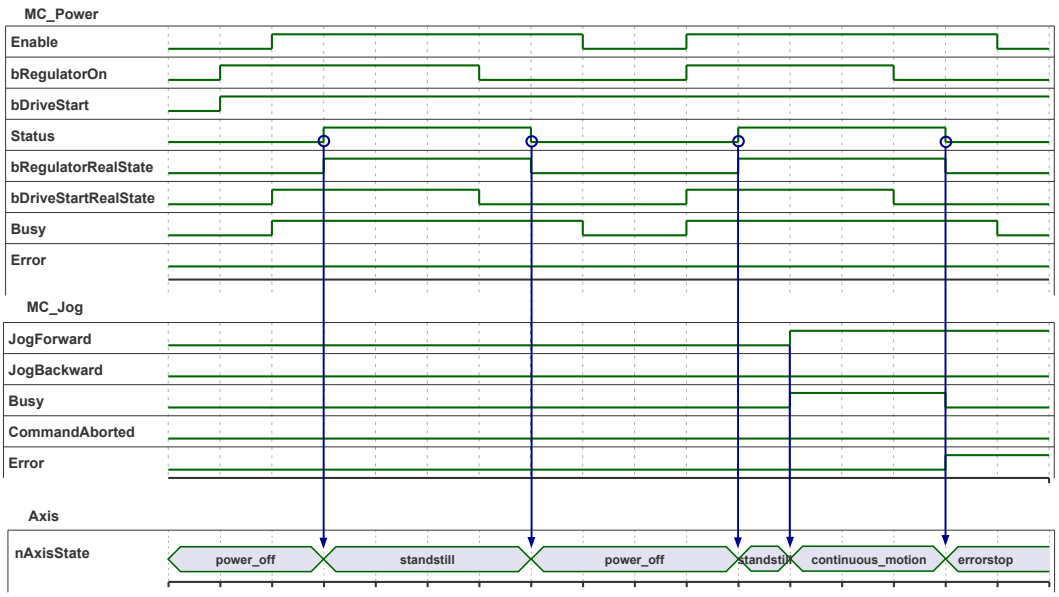
- Description of functions
  - This FB controls the axis readiness for motion.
  - Use this FB with bDriveStart fixed to TRUE.
- Operation start

- The inputs Enable, bRegulatorOn, and bDriveStart are set to TRUE. After that, when the Status goes TRUE, the axis is ready to move.

- Operation stop
  - After the start of operation, when the Enable is TRUE and the bRegulatorOn is set to FALSE, the Status goes FALSE and the motion ready state is canceled. However, if an FB is controlling the same axis, an error occurs in the FB in operation and the FB stops because the nAxisState of the axis is set to errorstop.
  - After the start of operation, even when only the Enable is changed to FALSE, the nAxisState of the axis remains unchanged. Thus, an FB in operation on the same axis does not stop.

■ Timing chart

- When the Enable is set to TRUE, the Busy goes TRUE.
- The bRegulatorOn, bDriveStart, and Enable are set to TRUE. After that, when the Status goes TRUE, the axis is ready for motion.



Info.

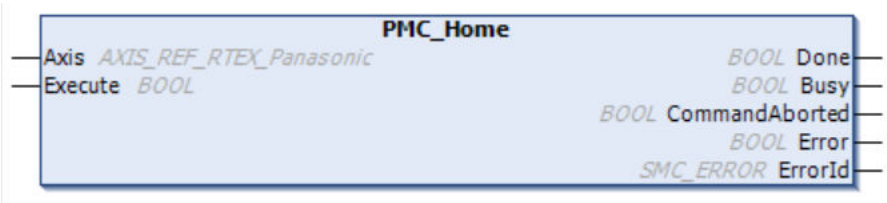
- If the Enable, bRegulatorOn, and bDriveStart are TRUE but the Status remains FALSE without the occurrence of any error, a possible reason is a hardware problem with the motor.

5.2 Home Return

5.2.1 PMC_Home (Home Return)

This is a function block (FB) that performs home return of the axis. The home return function of the servo amplifier is used.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_RTEX_Panasonic	-	Specifies the axis.
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Stops processing.
Output	Done	BOOL	FALSE	TRUE: Execution is completed and transitioned to the Standstill state.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	An error ID is output.

■ **Execution operation**

- Execute = TRUE: Starts the home return mode. Execute = FALSE: Ends the home return mode.
- When PMC_Home is successfully completed (when Done changes to TRUE), the home return mode is automatically ended.
- When PMC_Home is abnormally terminated (when Error changes to TRUE), end the home return mode by setting Execute = FALSE and making a call.

■ Execution errors

The PMC_Home function block outputs the following

Error	Description
SMC_WRONG_CONTROLLER_MODE	Executed in a mode other than the position control mode. Change to SMC_position using SMC_SetControllerMode.
SMC_DI_HOMING_ERROR	The version of the amplifier paired with an absolute encoder is lower than V1.24.
	Trigger setting is incorrect.
	Amplifier parameters (Pr4.00 to Pr4.07) are incorrect.
	Abnormal state in HOME, POT, or NOT is detected.
	The home return cannot be completed even if POT and NOT settings were inverted three times or more.
	The home return was completed at an incorrect position.
SMC_MS_DIRECTION_NOT_APPLICABLE	The return direction setting is incorrect.
SMC_AXIS_NOT_READY_FOR_MOTION	The axis is in a state (Stopping, Disabled, or Errorstop) where PMC_Home cannot be executed.
SMC_REGULATOR_OR_START_NOT_SET	The servo was turns OFF and the brake was applied.
SMC_3SH_INVALID_VELACC_VALUES	The input target velocity, home return creep speed, acceleration, or deceleration is incorrect.
SMC_AXIS_REF_CHANGED_DURING_OPERATION	The Axis was changed during operation.

■ Execution conditions

- As the PMC_Home function block uses the RTEX home return command, it cannot be executed together with PMC_ReadLatchPosition or PMC_StopLatchPosition.
- If PMC_Home is executed while PMC_ReadLatchPosition or PMC_StopLatchPosition is being executed, the CommandAborted parameter becomes TRUE. Furthermore, if PMC_Home of another instance is executed while one PMC_Home is being executed, the CommandAborted parameter of the PMC_Home executed later becomes TRUE.

■ Amplifier parameter conditions

When using PMC_Home, set amplifier parameters as shown in the following table.

Parameter	Parameter name	Setting A	Setting B
Pr4.00	SI1 input selection	SI-MON5	SI-MON5
Pr4.01	SI2 input selection	POT	
Pr4.02	SI3 input selection	NOT	
Pr4.03	SI4 input selection	SI-MON1	SI-MON1
Pr4.04	SI5 input selection	HOME	HOME
Pr4.05	SI6 input selection	EXT2	POT
Pr4.06	SI7 input selection	EXT3	NOT
Pr4.07	SI8 input selection	SI-MON4	SI-MON4

Return methods that can be executed for the settings A and B are as shown in the following table.

Return method	Setting A	Setting B
DOG method 1	○	○(Note 2)
DOG method 2	×(Note 1)	○
DOG method 3	○	○(Note 2)
Limit method 1	○	○(Note 2)
Limit method 2	×(Note 1)	○
Home return method	○	○(Note 2)
Stop-on-contact method 1	○	○(Note 2)
Stop-on-contact method 2	○	○
Data setting method	○	○
High-speed home return method	○	○

(Note 1) When using POT, NOT, or HOME as a home reference trigger, assign them as follows.
HOME: SI5 input selection
POT: SI6 input selection
NOT: SI7 input selection

(Note 2) When EXT2 or EXT3 is used as a home reference trigger, it can be used only for the above setting A.

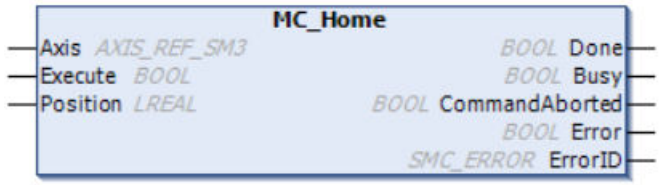
Info.

- Reference manual
  GM1 Controller RTEX User's Manual (Operation Edition)

5.2.2 MC_Home (Home Return)

This is a function block (FB) that performs home return.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge.
	Position	LREAL	0	Set value of the absolute position when the reference signal is detected
Output	Done	BOOL	FALSE	TRUE: Stopping is completed.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	Error ID output

5.3 Control Switch

5.3.1 SMC_SetControllerMode (Control Mode Setting)

This is a function block (FB) that sets the control mode. The default control mode value is position control mode.

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	nControllerMode	SMC_CONTROLLER_MODE	SMC_position	Specifies the control mode.
Output	bDone	BOOL	FALSE	TRUE: Control mode setting is completed.
	bBusy	BOOL	FALSE	TRUE: The FB is in operation.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	nErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ SMC_CONTROLLER_MODE (Enumeration type)

Name	Value	Description
SMC_nocontrol	0	Usage prohibited
SMC_torque	1	Torque control mode
SMC_velocity	2	Velocity control mode
SMC_position	3	Position control mode, default value
SMC_current	4	Usage prohibited

■ Detail of function

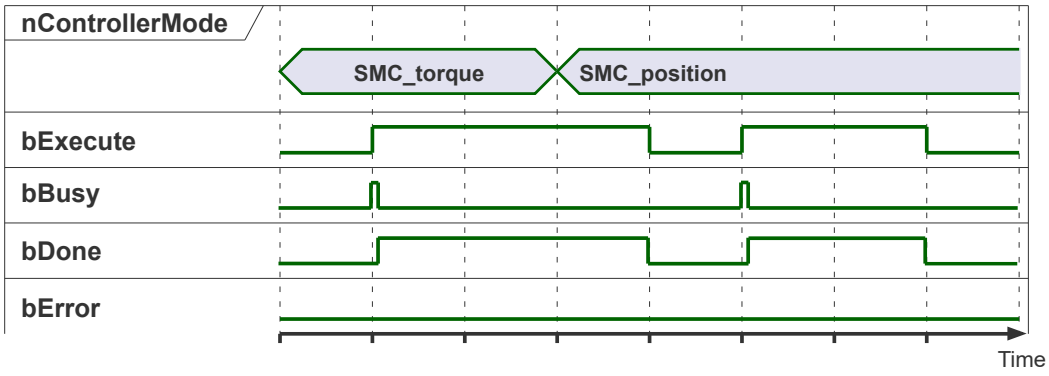
- Description of functions
  - The table below shows a list of FBs that correspond to each control mode. The default value of the control mode is position control mode. MC_MoveVelocity, MC_VelocityProfile, and MC_AccelerationProfile also operate in the position control mode.

Control mode	Compatible FBs
Torque control mode	PMC_SetTorque, SMC_SetTorque
Velocity control mode	MC_MoveVelocity, MC_VelocityProfile, MC_AccelerationProfile
Position control mode	MC_MoveAbsolute, MC_MoveRelative, MC_MoveAdditive, MC_MoveSuperImposed, MC_PositionProfile, SMC_MoveContinuousAbsolute, SMC_MoveContinuousRelative, MC_MoveVelocity, MC_VelocityProfile, MC_AccelerationProfile

- Operation start
  - At the rising edge of Execute, the control mode is set according to the nControllerMode value.
- Operation stop
  - When the control mode setting is completed, the operation stops.

■ Timing chart

- At the rising edge of the bExecute, Busy changes to TRUE.
- When the control mode setting is completed, bDone changes to TRUE and immediately FALSE.



Info.

- If the control mode is changed from torque control mode to velocity control mode, an error occurs. Therefore, change it through the position control mode.
- Whether or not the control mode can be changed for an axis in motion depends on the specifications of the servo amplifier. We recommend that you once stop the axis motion when changing the control mode.

5.4 Stop

5.4.1 MC_Stop (Forced Stop)

This is a function block (FB) that causes the axis to make a deceleration stop. While the axis is being decelerated or while it is stopped, another FB cannot be executed. Use this FB for emergency stop or exception handling.

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge. While it is TRUE, other FB cannot be executed.
	Deceleration	LREAL	0	Specifies the deceleration (u/s^2).
	Jerk	LREAL	0	Specifies the jerk (u/s^3).
Output	Done	BOOL	FALSE	TRUE: The axis velocity has reached 0.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	Error	BOOL	FALSE	TRUE: An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ Detail of function

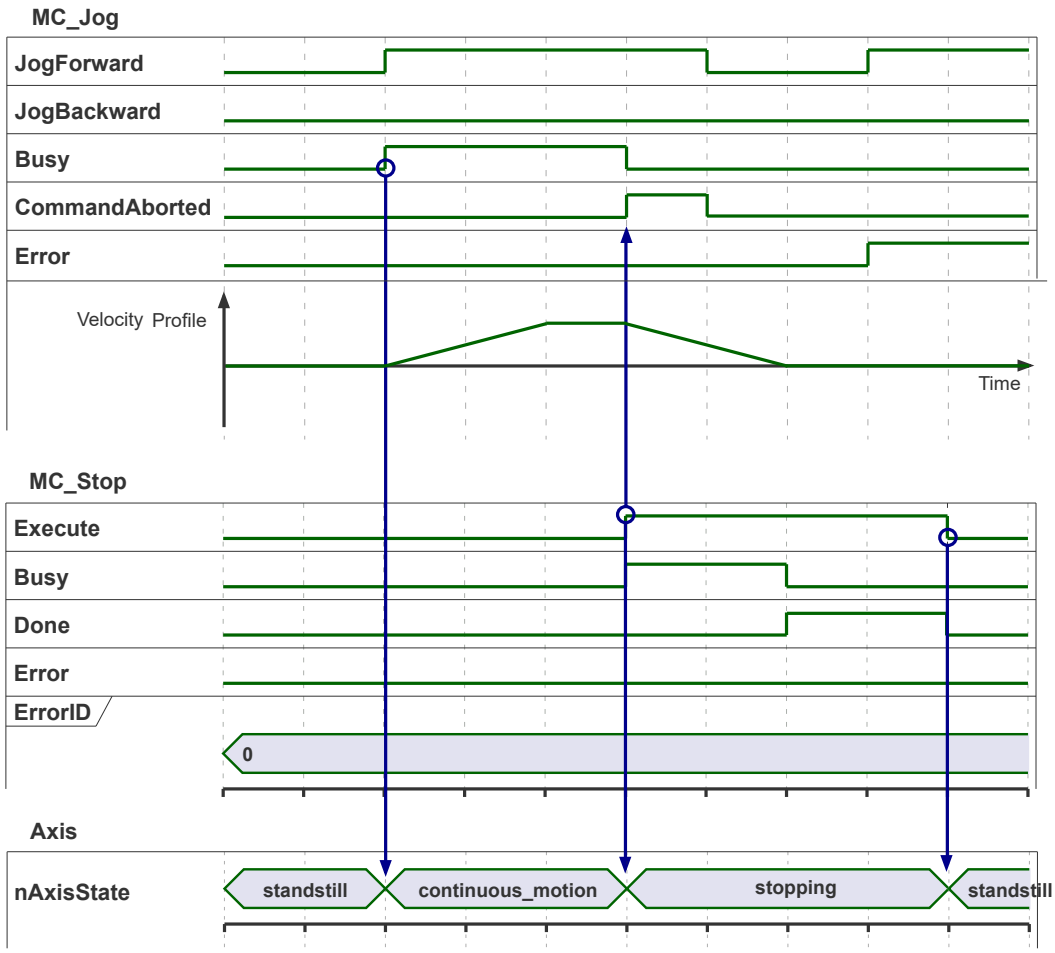
- Description of functions
  - This FB causes the axis in motion to make a deceleration stop according to the Deceleration and Jerk values.
  - Use this FB for emergency stop or exception handling.
  - As long as either the Busy or Execute is TRUE, an FB that controls the same axis cannot be executed. If an FB that controls the same axis is called, an error occurs in the FB and the ErrorID is set to SMC_AXIS_NOT_READY_FOR_MOTION. Even if the Done output is TRUE, as long as the Execute is TRUE, an FB that controls the same axis cannot be executed.
- Operation start
  - At the launch of the Execute, the axis starts making a deceleration stop according to the Deceleration and Jerk values.
- Operation stop

- When the axis completes the deceleration stop and the axis velocity reaches 0, the axis operation stops.
- When the axis operation stops, the Done goes TRUE and the Busy goes FALSE.

- Re-execution
  - Set the Execute to FALSE. Next, specify input values again. When the Execute is launched again, the FB is executed with the new input values.
- Interruption of operation
  - Operation of this FB cannot be interrupted by another FB.

■ Timing chart

- At the launch of the Execute, the Busy goes TRUE.
- When the process is completed, the Busy goes FALSE and the Done goes TRUE. The Done in the TRUE state goes FALSE when the Execute goes FALSE.



Info.

- In the torque control mode (SMC_torque), the axis cannot be stopped using MC_Stop. For stopping methods, refer to PMC_SetTorque or SMC_SetTorque.

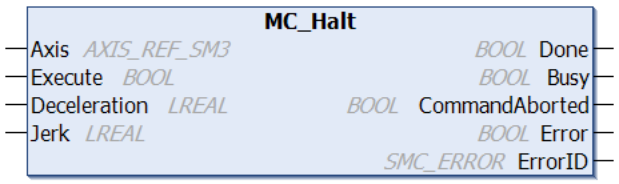
REFERENCE

5.8.1 PMC_SetTorque (Torque Control)

5.4.2 MC_Halt (Halt)

This is a function block (FB) that causes the axis to make a deceleration stop. While the axis is being decelerated or while it is stopped, another FB can be executed.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Deceleration	LREAL	0	Specifies the deceleration (u/s^2).
	Jerk	LREAL	0	Specifies the jerk (u/s^3).
Output	Done	BOOL	FALSE	TRUE: The axis velocity has reached 0.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from another FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

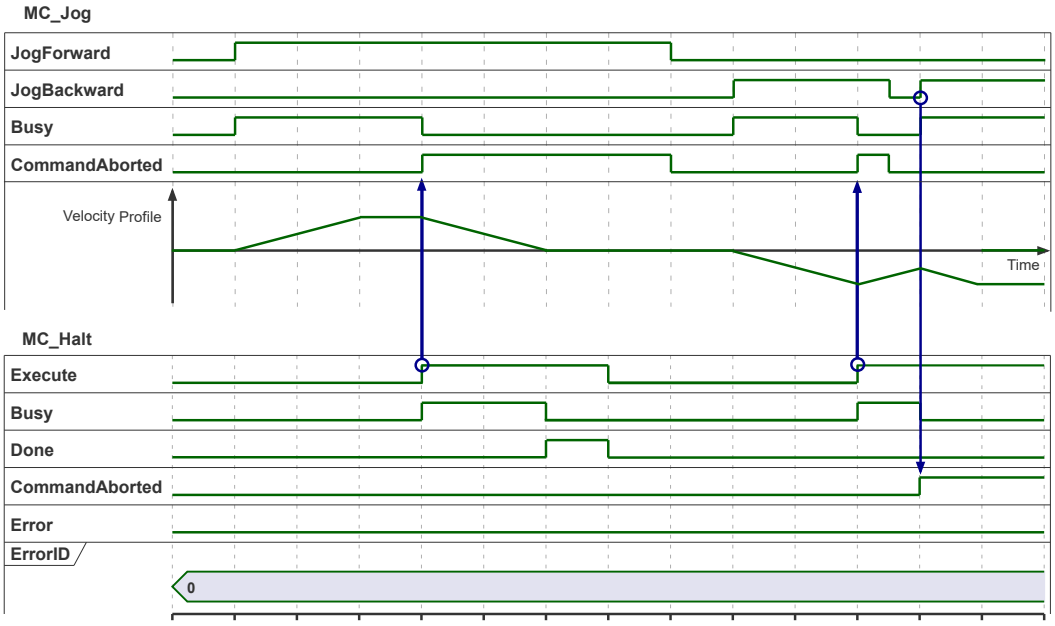
■ Detail of function

- Description of functions
  - This FB causes the axis in motion to make a deceleration stop according to the Deceleration and Jerk values.
  - Use this FB to stop the axis in normal operation other than emergency stop and exception handling.
- Operation start

  - At the launch of the Execute, the axis starts making a deceleration stop according to the Deceleration and Jerk values.
- Operation stop
  - When the axis completes the deceleration stop and the axis velocity reaches 0, the axis operation stops.
- Re-execution
  - Set the Execute to FALSE. Next, specify input values again. When the Execute is launched again, the FB is executed with the new input values.
- Interruption of operation
  - If, during operation of this FB, another FB that controls the same axis is called, the operation of this FB is interrupted.

■ Timing chart

- At the launch of the Execute, the Busy goes TRUE.
- When the operation is completed, the Busy goes FALSE and the Done goes TRUE. The Done in the TRUE state goes FALSE when the Execute is set to FALSE.
- When an FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE.



Info.

- In the torque control mode (SMC_torque), the axis cannot be stopped using MC_Halt. For stopping methods, refer to PMC_SetTorque or SMC_SetTorque.

5.4.3 Example: Stop

Here is an example of stopping the motor during axis movement.

■ Program examples

The following is an ST program example.

- Implementation Section

```
CASE Process OF
    0: // Servo ON
        MC_Power_0(
            Axis := Axisl ,
            Enable := TRUE ,
            bRegulatorOn := TRUE,
            bDriveStart := TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1: // Execute the MC_MoveVelocity
        MC_MoveVelocity_0(
            Axis := Axisl,
            Execute := TRUE,
            Velocity := 360,
            Acceleration := 3600,
            Deceleration := 3600,
            Direction := positive
        );
        IF MC_MoveVelocity_0.InVelocity = TRUE THEN
                MC_MoveVelocity_0(
                    Axis := Axisl,
                    Execute := FALSE,
                );
                Process := 2;
        END_IF
    2: // Execute the MC_stop
        MC_Stop_0(
            Axis := Axisl,
            Execute := TRUE,
            Deceleration := 1800
        );
        IF MC_Stop_0.Done THEN
            MC_Stop_0(
                Axis := Axisl,
                Execute := FALSE
            );
            Process := 3;
        END_IF
END_CASE
```

In the program example, stop operation is started at the rising edge of the "Execute" flag of "MC_Stop".

The "Busy" flag is set to TRUE during execution. While the "Busy" flag is set to TRUE, "MC_Stop_0" must be called every cycle. Otherwise, the operation will terminate with an error.

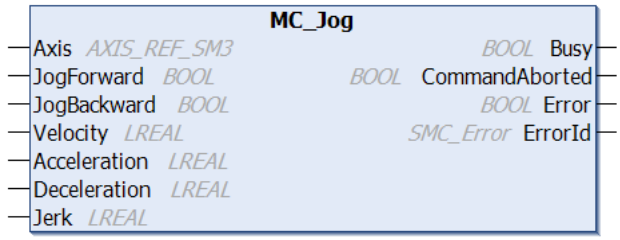
When the stop operation is completed, the "Done" flag is set to TRUE.

5.5 JOG / Inching

5.5.1 MC_Jog (Jogging)

This is a function block (FB) that causes the axis to keep traveling in a forward or reverse direction at a constant velocity.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	JogForward	BOOL	FALSE	While JogForward is TRUE, the axis travels in a forward direction. If JogBackaward is TRUE at the same time, the axis does not operate.
	JogBackward	BOOL	FALSE	While JogBackward is TRUE, the axis travels in a reverse direction. If JogForward is TRUE at the same time, the axis does not operate.
	Velocity	LREAL	0	Specifies the velocity (u/s).
	Acceleration	LREAL	0	Specifies the acceleration (u/s^2).
	Deceleration	LREAL	0	Specifies the deceleration (u/s^2).
	Jerk	LREAL	0	Specifies the jerk (u/s^3).
Output	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from another FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

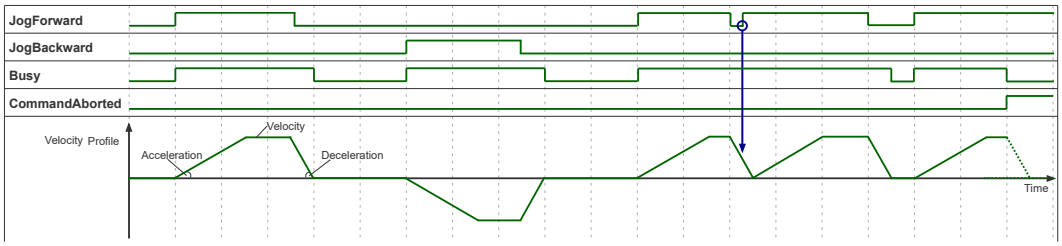
■ Detail of function

- Description of behavior
  - This FB causes the axis to make a jogged movement at a specified Velocity value.

  - To make the axis travel in a forward direction, set the JogForward to TRUE. To make the axis travel in a reverse direction, set the JogBackward to TRUE.
- Operation start
  - When the JogForward or JogBackward is set to TRUE, the axis starts acceleration according to the Acceleration and Jerk values. The axis velocity gets constant when it reaches the specified Velocity value.
  - The Velocity, Acceleration, and Jerk inputs are set to specified values at the launch of the JogForward or JogBackward. Thus, a change made to any of the Velocity, Acceleration, and Jerk values while the JogForward or JogBackward is TRUE does not take effect.
- Operation stop
  - When either the JogForward or JogBackward in the TRUE state is set to FALSE, the axis starts deceleration according to the Deceleration and Jerk values and stops.
  - When both the JogForward and JogBackward are set to TRUE, the axis starts deceleration according to the Deceleration and Jerk values and stops.
  - The Deceleration and Jerk inputs are set to specified values at the launch of the JogForward or JogBackward. Thus, a change made to any of the Deceleration and Jerk values while the JogForward or JogBackward is TRUE does not take effect.
- Restart operation during stop
  - When the JogForward or JogBackward is set to TRUE again during deceleration, the axis velocity reaches 0 once, and after that, the axis restarts acceleration according to the Acceleration and Jerk values, and the axis velocity gets constant when it reaches the specified Velocity value.
- Interruption of operation
  - When an FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE and the Busy goes FALSE.

■ Timing chart

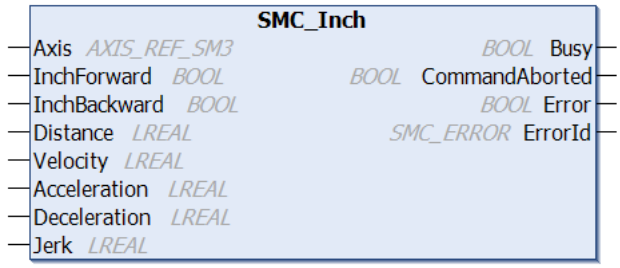
- At the launch of the JogForward or JogBackward, the Busy goes TRUE.
- When either the JogForward or JogBackward in the TRUE state is set to FALSE, the axis starts making a deceleration stop. When the axis velocity reaches 0, the Busy goes FALSE.
- When the CommandAborted goes TRUE, the Busy goes FALSE.



5.5.2 SMC_Inch (Inching)

This is a function block (FB) that causes the axis to travel in a forward or reverse direction for a specified relative distance.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	InchForward	BOOL	FALSE	When the input changes from FALSE to TRUE, the axis starts traveling in a forward direction for the distance specified in Distance. When the input changes to FALSE before the axis travels the specified distance, the axis stops traveling. If InchBackaward is TRUE at the same time, the axis does not operate.
	InchBackward	BOOL	FALSE	When the input changes from FALSE to TRUE, the axis starts traveling in a reverse direction for the distance specified in Distance. When the input changes to FALSE before the axis travels the specified distance, the axis stops traveling. If InchForward is TRUE at the same time, the axis does not operate.
	Distance	LREAL	0	Specifies the travel distance (u).
	Velocity	LREAL	0	Specifies the velocity (u/s).
	Acceleration	LREAL	0	Specifies the acceleration (u/s^2).
	Deceleration	LREAL	0	Specifies the deceleration (u/s^2).
	Jerk	LREAL	0	Specifies the jerk (u/s^3).
Output	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from another FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.

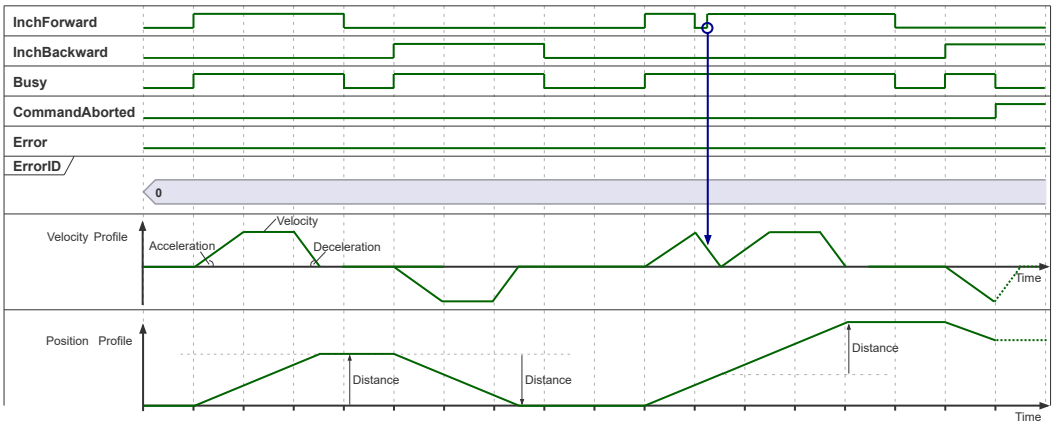
Scope	Name	Type	Default value	Description
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ Detail of function

- Description of behavior
  - This FB causes the axis to travel from start position for a distance specified in Distance at a specified Velocity value and stop.
  - To make the axis travel in a forward direction, set the InchForward to TRUE. To make the axis travel in a reverse direction, set the InchBackward to TRUE.
- Operation start
  - When the InchForward or InchBackward is set to TRUE, the axis starts acceleration according to the Acceleration and Jerk values. When the axis reaches the specified Velocity value, it maintains the reached velocity.
  - The Distance, Velocity, Acceleration, and Jerk inputs are set to specified values at the launch of the InchForward or InchBackward. Thus, a change made to any of the Distance, Velocity, Acceleration, and Jerk values while the InchForward or InchBackward is TRUE does not take effect.
- Operation stop
  - The motion stops when the axis completes traveling from the start position for the distance specified in Distance.
  - The axis starts deceleration according to the Deceleration and Jerk values when it nearly travels the Distance value, and the velocity decreases to 0 when the axis reaches the Distance value.
  - When either the InchForward or InchBackward in the TRUE state is set to FALSE before the axis reaches the Distance value, the axis decelerates according to the Deceleration and Jerk values and stops.
  - When both the InchForward and InchBackward are set to TRUE, the axis decelerates according to the Deceleration and Jerk values and stops.
- Restart operation during stop
  - When the InchForward or InchBackward is set to TRUE again during deceleration, the axis velocity reaches 0 once, and after that, the axis restarts acceleration according to the Acceleration and Jerk values, and the axis velocity gets constant at the specified Velocity value. With the current place taken as a start position, the axis travels a distance specified in Distance and stops.
- Interruption of operation
  - When an FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE and the Busy goes FALSE.

■ Timing chart

- At the launch of the InchForward or InchBackward, the Busy goes TRUE.
- When either the InchForward or InchBackward in the TRUE state is set to FALSE after the axis reaches the Distance value, the Busy goes FALSE.
- When either the InchForward or InchBackward in the TRUE state is set to FALSE before the axis reaches the Distance value, the axis starts making a deceleration stop. When the axis velocity reaches 0, the Busy goes FALSE.
- When the CommandAborted goes TRUE, the Busy goes FALSE.


InchForward
InchBackward
Busy
CommandAborted
Error
ErrorID
0
Velocity Profile
Acceleration
Velocity
Deceleration
Time
Position Profile
Distance
Distance
Distance
Time

5.5.3 Example: JOG Operation

Here is an example of how to execute JOG operation.

■ Program examples

Here is an example of an ST (Structured Text) program.

- Implementation Section

```
CASE Process OF
    0: // Servo ON
        MC_Power_0(
            Axis := Axisl,
            Enable := TRUE,
            bRegulatorOn := SW_power,
            bDriveStart := TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1: // Execute the MC_
        MC_Jog_0(
            Axis := Axisl,
            JogForward := TRUE,
            JogBackward := FALSE,
            Velocity := 360,
            Acceleration := 3600,
            Deceleration := 3600
        );
        MC_ReadActualPosition_0(
            Axis := Axisl,
            Enable := TRUE
        );
        IF MC_ReadActualPosition_0.Valid = TRUE THEN
            IF MC_ReadActualPosition_0.Position > 1000 THEN
                Process := 2;
            END_IF
        END_IF
    2: // Execute the MC_MoveRelative
        MC_Jog_0(
            Axis := Axisl,
            JogForward := FALSE,
            JogBackward := TRUE,
            Velocity := 720,
            Acceleration := 3600,
            Deceleration := 3600
        );
END_CASE
```

JOG operation is started at the rising edge of "JogForward" or "JogBackward".

The "Busy" flag is set to TRUE during execution. While the "Busy" flag is set to TRUE, MC_Jog_0 must be called every cycle. Otherwise, the operation will terminate with an error.

5.6 Position Control

5.6.1 MC_MoveAbsolute (Absolute Value Positioning)

This is a function block (FB) that causes the axis to travel to a specified target position.

■ **Icon**

■ **Parameter**

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Position	LREAL	0	Specifies the target position (u).
	Velocity	LREAL	0	Specifies the maximum velocity (u/s).
	Acceleration	LREAL	0	Specifies the acceleration (u/s^2).
	Deceleration	LREAL	0	Specifies the deceleration (u/s^2).
	Jerk	LREAL	0	Specifies the jerk (u/s^3).
	Direction	MC_Direction	shortest	Specification is valid only when the axis is of the modulo type. Specifies the traveling direction of the axis.
	BufferMode	MC_BUFFER_MODE	Aborting	Specifies a buffer mode. The value is valid when this FB is a second FB.
Output	Done	BOOL	FALSE	TRUE: The target position has been reached.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	Active	BOOL	FALSE	TRUE: The second FB is being controlled.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred.

Scope	Name	Type	Initial	Description
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ MC_Direction (Enumeration type)

Name	Value	Description
positive	1	Travels in the positive direction.
negative	-1	Travels in the negative direction.
shortest	0	Travels in the direction in which the distance is shortest from the current command position at the time of the command execution to the target position.
fastest	3	Travels in the direction in which the time is fastest from the current command position at the time of the command execution to the target position.
current	2	If an FB is controlling the axis, the axis travels by keeping the current direction. If any FB is not controlling the axis, the axis travels in the direction taken by the immediately preceding FB that controlled the axis.

■ MC_BUFFER_MODE (Enumeration type)

On condition that this FB is connected as the second FB, the table below gives a description.

Name	Value	Description
Aborting	0	The operation of the first FB stops, and this FB starts operation instantly.
Buffered	1	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The movement starts at the velocity that the preceding movement has when the end condition is reached.
BlendingLow	2	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The command velocities of the first FB and this FB are compared, and the axis passes through the end position of the first FB operation at the lower command velocity.
BlendingPrevious	3	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The axis passes through the end position of the first FB operation at the velocity of the first FB command.
BlendingNext	4	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The axis passes through the end position of the first FB operation at the velocity of this FB command.
BlendingHigh	5	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The command velocities of the first FB and this FB are compared, and the axis passes through the end position of the first FB operation at the higher command velocity.

(Note 1) Refer to Table "Buffer Mode Operation Conditions."

■ Detail of function

- Description of functions

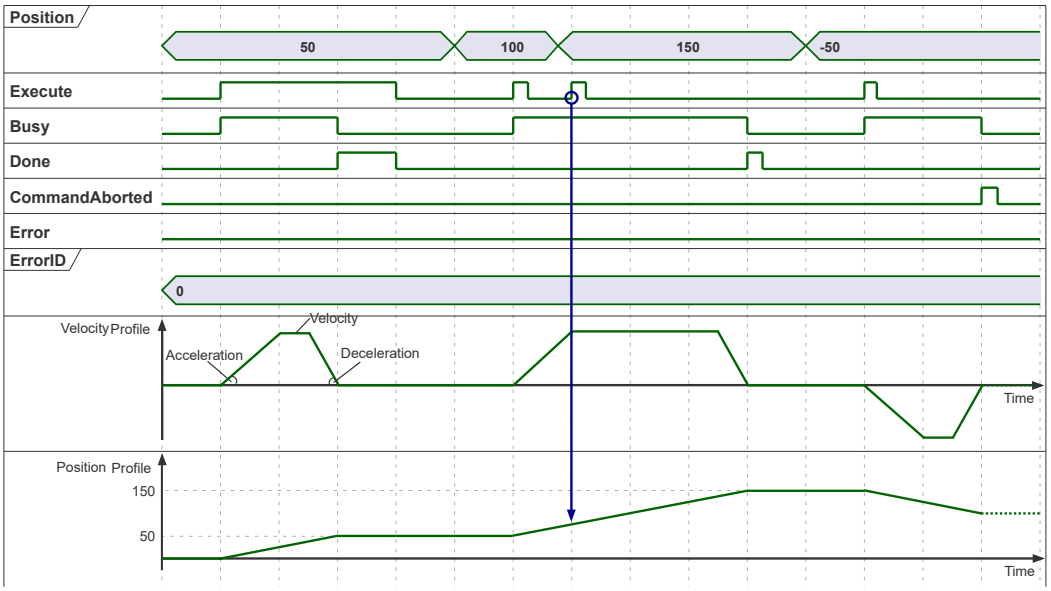
- This FB causes the axis to travel to the Position at the specified Velocity.
- The traveling direction of the axis, which is equivalent to the direction of the Velocity, is determined by the axis type.

Axis type	Axis traveling direction
Finite	The direction is determined by the positional relationship of the current command position and the Position input. If the Position input is larger relative to the current command position, the axis travels in the positive direction. If the Position input is smaller relative to the current command position, the axis travels in the negative direction.
Modulo	MC_Direction specification

- Operation start
  - At the launch of the Execute, the axis starts traveling according to the Position, Velocity, Acceleration, Jerk, and Direction values.
- Operation stop
  - The motion stops when the axis reaches the Position value.
  - Deceleration operation before the axis reaches the Position value behaves according to the Deceleration and Jerk values, and the Velocity level decreases to 0 when the axis reaches the Position value.
- Re-execution
  - Set the Execute to FALSE. Next, specify input values again. When the Execute is launched, the FB is executed with the new input values.
- Interruption of operation
  - If, during operation of this FB, another FB that controls the same axis is called, the operation of this FB is interrupted.

■ Timing chart

- At the launch of the Execute, the Busy goes TRUE.
- When the process is completed, the Busy goes FALSE and the Done goes TRUE. The Done in the TURE state goes FALSE when the Execute is set to FALSE.
- When another FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE. The behavior of this FB after the CommandAborted is TRUE depends on the behavior of other FBs.


Position
50
100
150
-50
Execute
Busy
Done
CommandAborted
Error
ErrorID
0
Velocity Profile
Velocity
Acceleration
Deceleration
Time
Position Profile
150
50
Time

5.6.2 MC_MoveRelative (Relative Value Positioning)

This is a function block (FB) that causes the axis to travel to a target position that is a result of the addition of a travel distance to the current command position.

■ Icon

■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Distance	LREAL	0	Specifies the travel distance (u). The target position is a position that is a result of the addition of a Distance value to the current command position.
	Velocity	LREAL	0	Specifies the maximum velocity (u/s).
	Acceleration	LREAL	0	Specifies the acceleration (u/s^2).
	Deceleration	LREAL	0	Specifies the deceleration (u/s^2).
	Jerk	LREAL	0	Specifies the jerk (u/s^3).
	BufferMode	MC_BUFFER_MODE	Aborting	Specifies a buffer mode. The value is valid when this FB is a second FB.
Output	Done	BOOL	FALSE	TRUE: The target position has been reached.
	Busy	BOOL	FALSE	TRUE: FB operation is in progress.
	Active	BOOL	FALSE	TRUE: The second FB is being controlled.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ MC_BUFFER_MODE (Enumeration type)

On condition that this FB is connected as the second FB, the table below gives a description.

ENUM	Value	Description
Aborting	0	The operation of the first FB stops, and this FB starts operation instantly.
Buffered	1	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The movement starts at the velocity that the preceding movement has when the end condition is reached.
BlendingLow	2	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The command velocities of the first FB and this FB are compared, and the axis passes through the end position of the first FB operation at the lower command velocity.
BlendingPrevious	3	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The axis passes through the end position of the first FB operation at the velocity of the first FB command.
BlendingNext	4	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The axis passes through the end position of the first FB operation at the velocity of this FB command.
BlendingHigh	5	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The command velocities of the first FB and this FB are compared, and the axis passes through the end position of the first FB operation at the higher command velocity.

(Note 1) Refer to Table "Buffer Mode Operation Conditions."

■ Detail of function

- Description of functions
  - This FB causes the axis to travel to a target position at the specified Velocity. The target position is a result of the addition of a Distance value to the current command position.
  - The traveling direction of the axis before the target position is reached is determined by whether the Distance input is positive or negative.

    When the Distance input is positive, the axis travels in the positive direction.

    When the Distance input is negative, the axis travels in the negative direction.
- Operation start
  - At the launch of the Execute, the axis starts traveling according to the Distance, Velocity, Acceleration, Deceleration, and Jerk values. The target position is a position that is a result of the addition of a Distance value to the current command position.
- Operation stop
  - The motion stops when the axis reaches the target position.
  - Deceleration operation before the axis reaches the target position behaves according to the Deceleration and Jerk values, and the Velocity level decreases to 0 when the axis reaches the target position.
- Re-execution

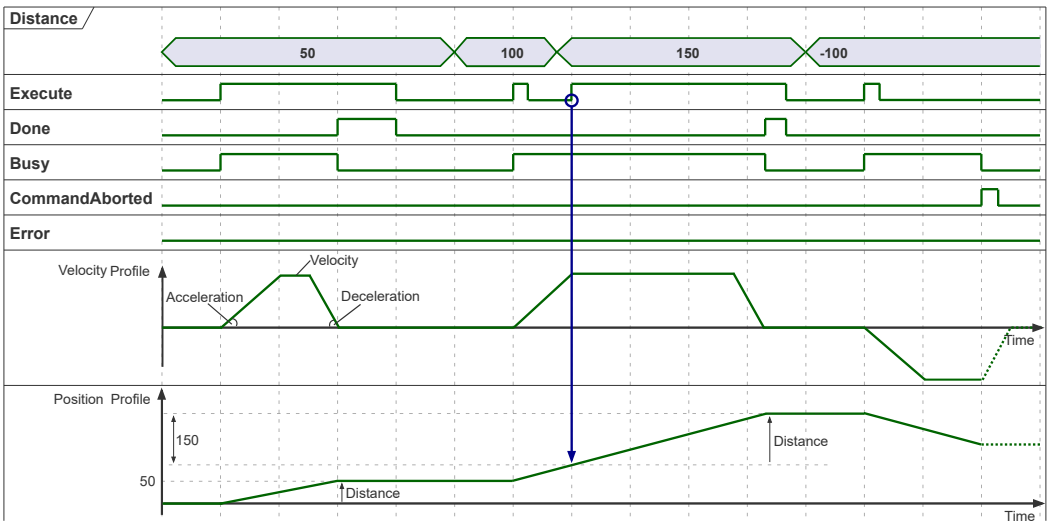
- Set the Execute to FALSE. Next, specify input values again. When the Execute is launched, the FB is executed with the new input values.

● Interruption of operation

- If, during operation of this FB, another FB that controls the same axis is called, the operation of this FB is interrupted.

■ Timing chart

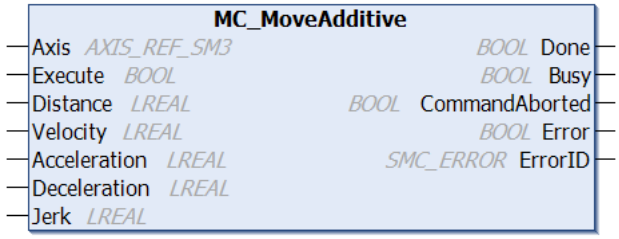
- At the launch of the Execute, the Busy goes TRUE.
- When the process is completed, the Busy goes FALSE and the Done goes TRUE. The Done in the TURE state goes FALSE when the Execute is set to FALSE.
- When another FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE. The behavior of this FB after the CommandAborted is TRUE depends on the behavior of other FBs.



5.6.3 MC_MoveAdditive (Target Position Change)

This is a function block (FB) that causes the axis to travel to a new target position that is a result of the addition of a travel distance to a target position the immediately preceding FB involving control of the axis has aimed to reach.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Distance	LREAL	0	Specifies the travel distance (u). The target position is a position that is a result of the addition of a Distance value to a target position the immediately preceding FB involving control of the axis has aimed to reach.
	Velocity	LREAL	0	Specifies the maximum velocity (u/s).
	Acceleration	LREAL	0	Specifies the acceleration (u/s^2).
	Deceleration	LREAL	0	Specifies the deceleration (u/s^2).
	Jerk	LREAL	0	Specifies the jerk (u/s^3).
Output	Done	BOOL	FALSE	TRUE: The target position has been reached.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ Detail of function

- Description of functions
  - This FB causes the axis to travel to a new target position that is a result of the addition of a travel distance to a target position the immediately preceding FB involving control of the axis has aimed to reach. The velocity, acceleration, deceleration, and jerk in operation by

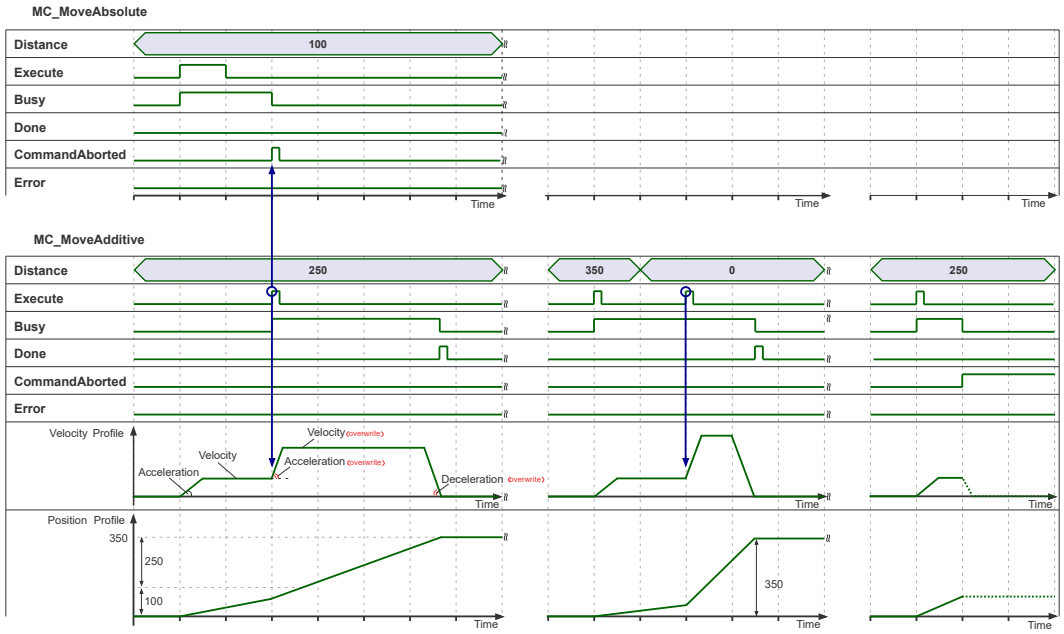
the immediately preceding FB are replaced by the Velocity, Acceleration, Deceleration, and Jerk inputs. Thus, when the Distance is set to 0, the velocity, acceleration, deceleration, and jerk in operation by the immediately preceding FB can be changed.

- When this FB is launched by the Execute input while another FB is controlling the axis, control of the axis by the other FB is interrupted.
- When this FB is launched by the Execute input while any FB involving control of the axis is not executed at all, this FB operates in the same way as the motion of MC_MoveRelative.

- Operation start
  - At the launch of the Execute, the axis starts traveling according to the Distance, Velocity, Acceleration, Deceleration, and Jerk values. The target position is a position that is a result of the addition of a Distance value to a target position the immediately preceding FB involving control of the axis has aimed to reach.
- Operation stop
  - The motion stops when the axis reaches the target position.
  - Deceleration operation before the axis reaches the target position behaves according to the Deceleration and Jerk values, and the Velocity level decreases to 0 when the axis reaches the target position.
- Re-execution
  - Set the Execute to FALSE. Next, specify input values again. When the Execute is launched, the FB is executed with the new input values.
- Interruption of operation
  - If, during operation of this FB, another FB that controls the same axis is called, the operation of this FB is interrupted.

■ Timing chart

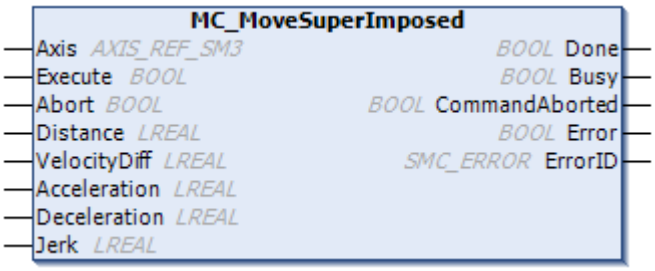
- At the launch of the Execute, the Busy goes TRUE.
- When the process is completed, the Busy goes FALSE and the Done goes TRUE. The Done in the TRUE state goes FALSE when the Execute is set to FALSE.
- When another FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE. The behavior of this FB after the CommandAborted is TRUE depends on the behavior of other FBs.


MC_MoveAbsolute
Distance
100
Execute
Busy
Done
CommandAborted
Error
Time
MC_MoveAdditive
Distance
250
350
0
250
Execute
Busy
Done
CommandAborted
Error
Velocity Profile
Velocity
Acceleration
Velocity (overwrite)
Acceleration (overwrite)
Deceleration (overwrite)
Time
Position Profile
350
250
100
350
Time

5.6.4 MC_MoveSuperImposed (Superimposed positioning)

This is a function block (FB) that adds an axis control command to another FB controlling the axis. Control commands to the target axis are command position, command velocity, command acceleration, command deceleration, and command jerk.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Abort	BOOL	FALSE	TRUE: The value of the axis control command added by this FB is set to 0, and the value of the control command previously executed by the existing FB returns.
	Distance	LREAL	0	Specifies the travel distance (u). The target position is a position that is a result of the addition of a Distance value to the current command position.
	VelocityDiff	LREAL	0	Specifies the maximum velocity (u/s) to be added.
	Acceleration	LREAL	0	Specifies the acceleration (u/s^2) to be added.
	Deceleration	LREAL	0	Specifies the deceleration (u/s^2) to be added.
	Jerk	LREAL	0	Specifies the jerk (u/s^3) to be added.
Output	Done	BOOL	FALSE	TRUE: The target position has been reached.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ Detail of function

- Description of functions
  - This FB adds an axis control command to another FB controlling the axis. Control commands to the target axis are command position, command velocity, command acceleration, command deceleration, and command jerk. Thus, it is assumed that this FB is used while another FB is controlling the axis. When this FB is launched by the Execute input while any other FB is not controlling the axis, this FB operates in the same way as the motion of MC_MoveRelative.
  - When this FB is launched by the Execute input while another FB is controlling the axis, the operation of the other FB continues.
- Operation start
  - At the launch of the Execute, this FB adds the VelocityDiff, Acceleration, Deceleration, and Jerk values to the motion of the axis that is being controlled by another FB. The target position of this FB is a position that is a result of the addition of a Distance value to the current command position. The target position of the other FB is a position that is a result of the addition of the current target position of the other FB to the Distance value.

    Target position = Current command position + Distance

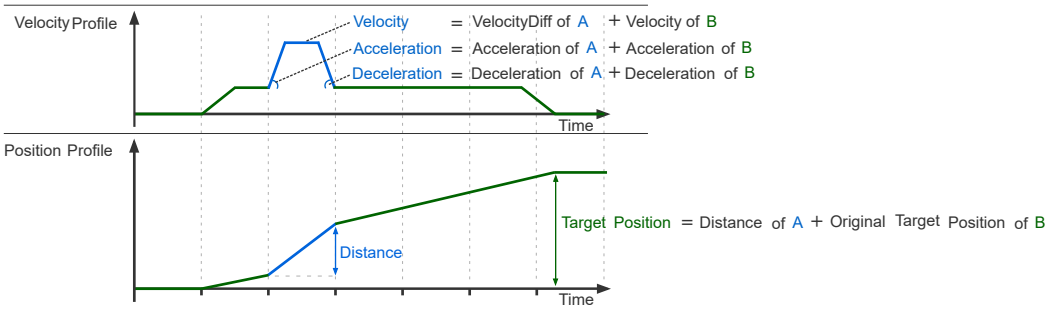
    Command velocity = Command velocity + VelocityDiff

    Command acceleration = Command acceleration + Acceleration

    Command deceleration = Command deceleration + Deceleration

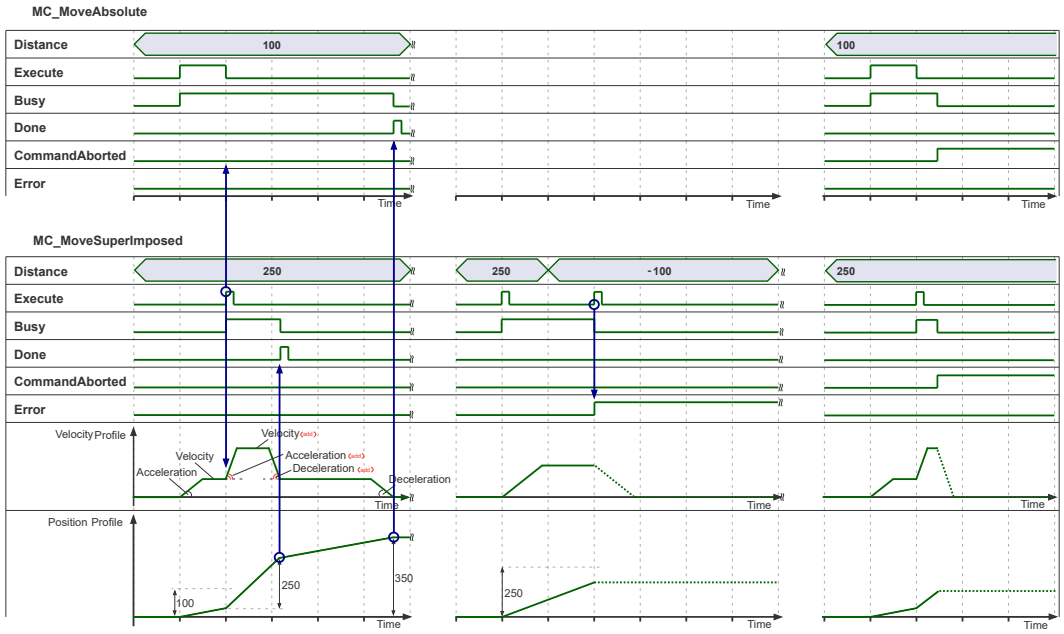
    Command jerk = Command jerk + Jerk
- Operation stop
  - When the target position of this FB is reached, the operation of this FB stops, and the axis returns to motion commanded only by the other FB.
  - Deceleration operation before the axis reaches the target position of this FB behaves according to the Deceleration and Jerk values, and the Velocity level returns to the Velocity value of the other FB when the axis reaches the target position of this FB.
- Re-execution
  - Set the Execute to FALSE. Next, specify input values again. When the Execute is launched, the FB is executed with the new input values.
  - The number of times at which this FB is allowed to be concurrently executed on the same axis is once. Thus, this FB cannot be concurrently executed multiple times while another FB is controlling the axis. If this FB is concurrently executed multiple times, an error occurs in this FB, and the ErrorID is set to SMC_AXIS_ERROR_DURING_MOTION.
- Interruption of operation
  - If, during operation of this FB, another FB that controls the same axis is called, the operation of this FB is interrupted.

- **BLUE LINE is MC_MoveSuperImposed . This is Called A.**
- **GREEN LINE is Original Motion Control FB. This is Called B.**



■ Timing chart

- At the launch of the Execute, the Busy goes TRUE.
- When the process is completed, the Busy goes FALSE and the Done goes TRUE. The Done in the TURE state goes FALSE when the Execute is set to FALSE.
- When another FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE. The behavior of this FB after the CommandAborted is TRUE depends on the behavior of other FBs.



Note

- While the Busy of MC_MoveSuperImposed is TRUE, be sure to call MC_MoveSuperImposed at every interval. If a call is not made, the axis may perform an abnormal operation.

5.6.5 MC_PositionProfile (Position Profile Move)

This is a function block (FB) that causes the axis to operate according to a position profile that consists of a combination of position and time.

■ Icon

■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
	TimePosition	MC_TP_REF	-	Specifies the position profile.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	ArraySize	INT	0	Specifies the number of profile points to be executed.
	PositionScale	LREAL	1	Position scaling Multiplies the position value of the profile by the specified value.
	Offset	LREAL	0	Position offset Adds the specified value to the position value of the profile.
Output	Done	BOOL	FALSE	TRUE: Motion specified by the position profile has been completed.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ MC_TP_REF (Structure)

Member	Type	Description
Number_of_pairs	INT	Not used. Values are ignored.
isAbsolute	BOOL	Methods of specifying profile positions TRUE: Absolute coordinate FALSE: Relative coordinate
MC_TP_Array	ARRAY [1..100] OF SMC_TP	A set of position profile data (1st point to 100th point)

■ SMC_TP (Structure)

Member	Type	Description
delta_time	TIME	Time of the profile Period of time spanned from the time at the last profile position
position	LREAL	Target position of the profile

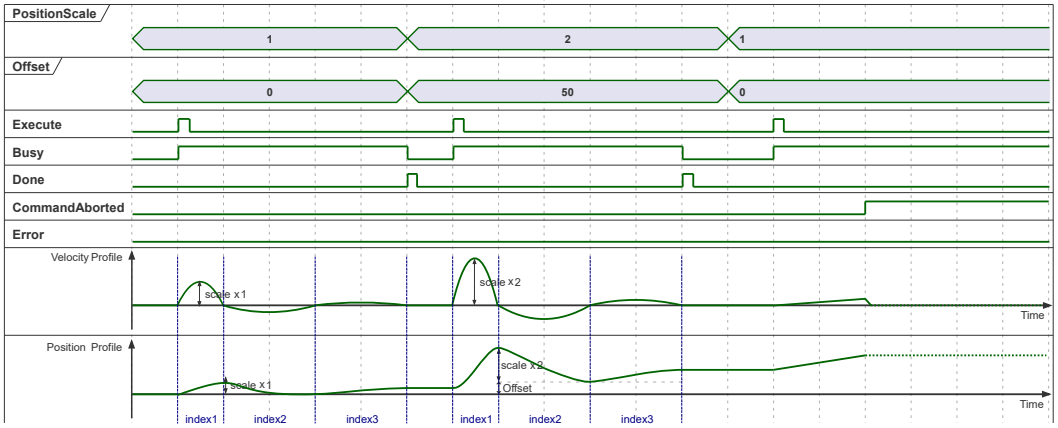
■ Detail of function

- Description of functions
  - This FB causes the axis to operate according to the position profile.
  - Up to 100 points that are each a combination of time and position values can be registered in the position profile.
  - The velocity is calculated by a fifth degree polynomial according to the specified time and positions.
- Operation start
  - At the launch of the Execute, the axis starts traveling according to the TimePosition description.
- Operation stop
  - When the axis has traveled through a trajectory equivalent to the number of profile points specified in ArraySize, the axis stops.
- Re-execution
  - Set the Execute to FALSE. Next, specify input values again. When the Execute is launched, the FB is executed with the new input values.
- Interruption of operation
  - If, during operation of this FB, another FB that controls the same axis is called, the operation of this FB is interrupted.

■ Timing chart

- At the launch of the Execute, the Busy goes TRUE.
- When the axis has traveled through a trajectory equivalent to the number of profile points specified in ArraySize, the Busy goes FALSE and the Done goes TRUE. The Done in the TURE state goes FALSE when the Execute is set to FALSE.
- When another FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE. The behavior of this FB after the CommandAborted is TRUE depends on the behavior of other FBs.
- The timing chart is plotted when MC_TP_Array, a structure member of TimePosition, has values shown in the table below. In this example, isAbsolute, a structure member of TimePosition, is set to TRUE.

index	delta_time	position
1	Time#10s	150
2	Time#20s	0
3	Time#20000ms	50



Note

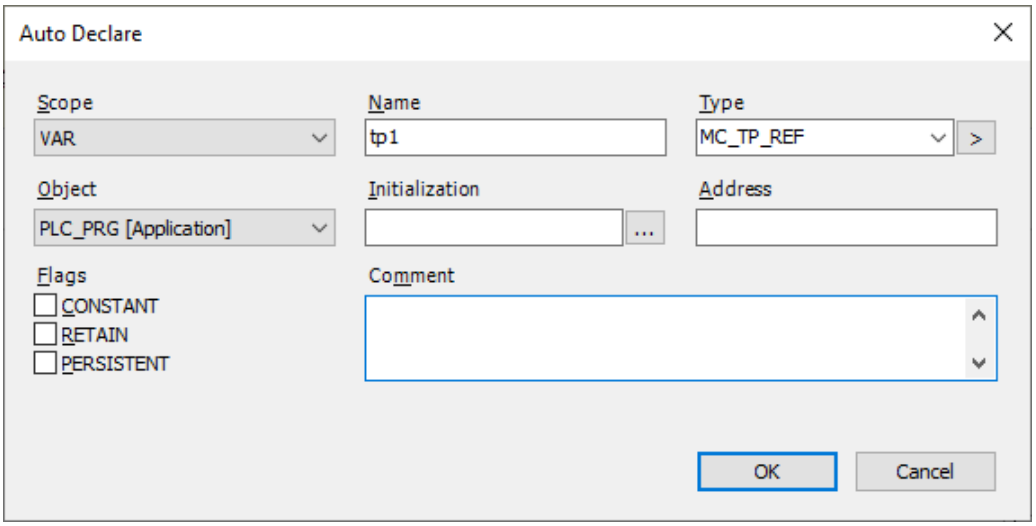
- While the axis keeps driving, do not set delta_time to 0 ms. Otherwise, operation cannot be properly executed in the section for which 0 ms is specified and subsequent sections.

5.6.6 Default Setting for Variables of the MC_TP_REF Type Structure

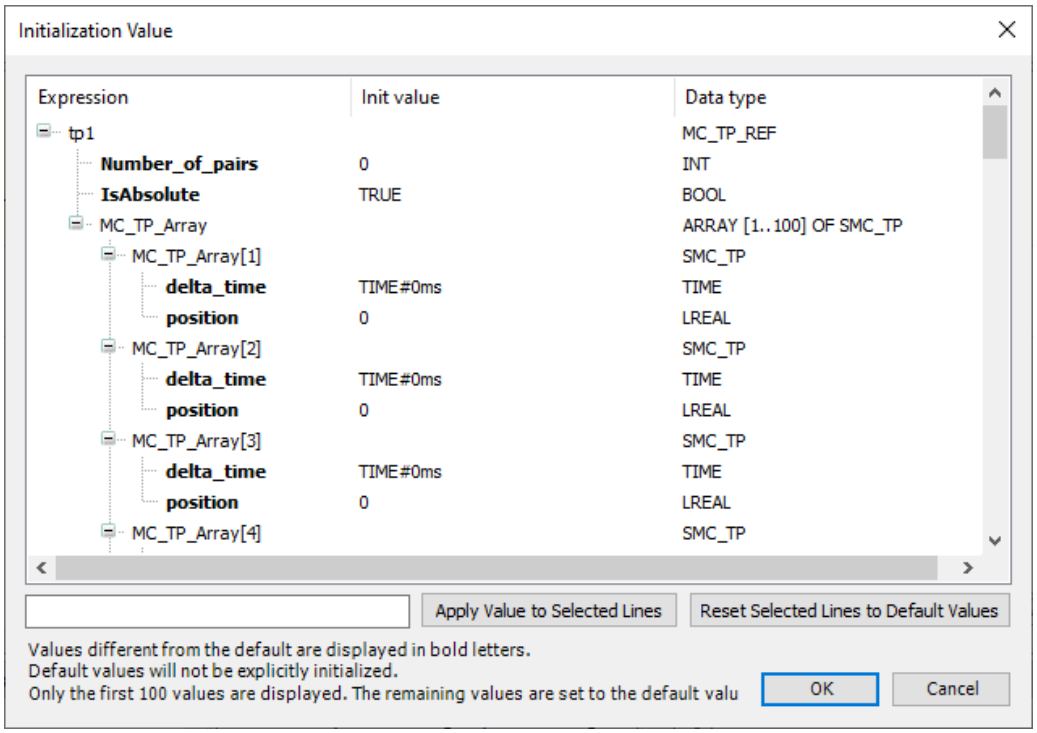
To enter the value of the input TimePosition, it is necessary to make default setting for variables of the MC_TP_REF type structure.

Procedure

1. When the input variable to TimePosition is declared, "Automatic Declaration" dialog box is displayed. Click ... displayed next to the "Initial Value" field.



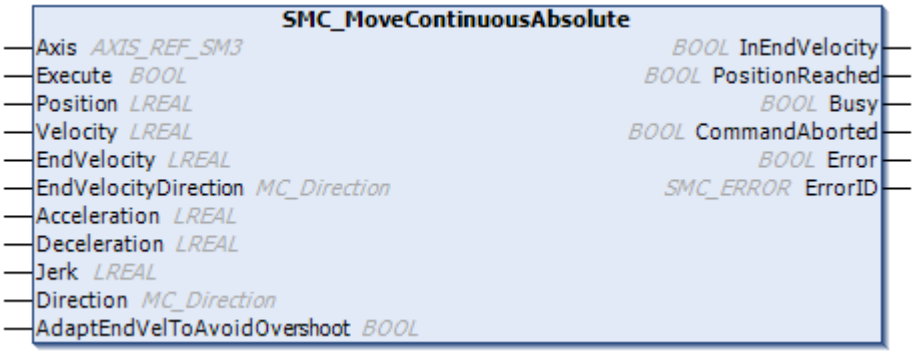
2. The "Initial Value" dialog box is displayed and, on the dialog box, you can set the default value for every member of the variable type (MC_TP_REF).


Initialization Value
Expression
Init value
Data type
tp1
MC_TP_REF
Number_of_pairs
0
INT
IsAbsolute
TRUE
BOOL
MC_TP_Array
ARRAY [1..100] OF SMC_TP
MC_TP_Array[1]
SMC_TP
delta_time
TIME#0ms
TIME
position
0
LREAL
MC_TP_Array[2]
SMC_TP
delta_time
TIME#0ms
TIME
position
0
LREAL
MC_TP_Array[3]
SMC_TP
delta_time
TIME#0ms
TIME
position
0
LREAL
MC_TP_Array[4]
SMC_TP
Apply Value to Selected Lines
Reset Selected Lines to Default Values
Values different from the default are displayed in bold letters.
Default values will not be explicitly initialized.
Only the first 100 values are displayed. The remaining values are set to the default valu
OK
Cancel

5.6.7 SMC_MoveContinuousAbsolute (Absolute Value Position Velocity Move)

This function block (FB) causes the axis to travel to a specified target position. Then, after the axis reaches the target position, this FB causes the axis to keep moving at a specified velocity.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Position	LREAL	0	Specifies the target position (u).
	Velocity	LREAL	0	Specifies the velocity (u/s) until the axis reaches the target position.
	EndVelocity	LREAL	0	Specifies the velocity (u/s) after the axis reaches the target position.
	EndVelocityDirection	MC_Direction	current	Specifies the traveling direction of the axis after the axis reaches the target position. Specifies either “positive”, “negative”, or “current”. If “fastest” or “shortest” is specified, an error occurs.
	Acceleration	LREAL	0	Specifies the acceleration (u/s^2).
	Deceleration	LREAL	0	Specifies the deceleration (u/s^2).
	Jerk	LREAL	0	Specifies the jerk (u/s^3).
	Direction	MC_Direction	shortest	Specification is valid only when the axis is of the modulo type. Specifies the traveling direction of the axis.
	AdaptEndVelToAvoidOvershoot	BOOL	FALSE	Do not use.

Scope	Name	Type	Initial	Description
Output	InEndVelocity	BOOL	FALSE	TRUE: The velocity after the attainment of the target position has been reached.
	PositionReached	BOOL	FALSE	TRUE: The target position has been reached.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ MC_Direction (Enumeration type)

Name	Value	Description
positive	1	Travels in the positive direction.
negative	-1	Travels in the negative direction.
shortest	0	Travels in the direction in which the distance is shortest from the current command position at the time of the command execution to the target position.
fastest	3	Travels in the direction in which the time is fastest from the current command position at the time of the command execution to the target position.
current	2	If an FB is controlling the axis, the axis travels by keeping the current direction. If any FB is not controlling the axis, the axis travels in the direction taken by the immediately preceding FB that controlled the axis.

■ Detail of function

- Description of functions
  - This FB causes the axis to travel to the Position at the specified Velocity. After the axis reaches the Position, this FB then causes the axis to keep moving at the specified EndVelocity value.
  - The traveling direction of the axis before the Position is reached is determined by the axis type.
  - The traveling direction of the axis after the Position is reached is determined by the specified EndVelocityDirection value.
  - The behavior of the axis when the Position is reached is determined by the relationship between the traveling directions of the axis before and after the position is reached.

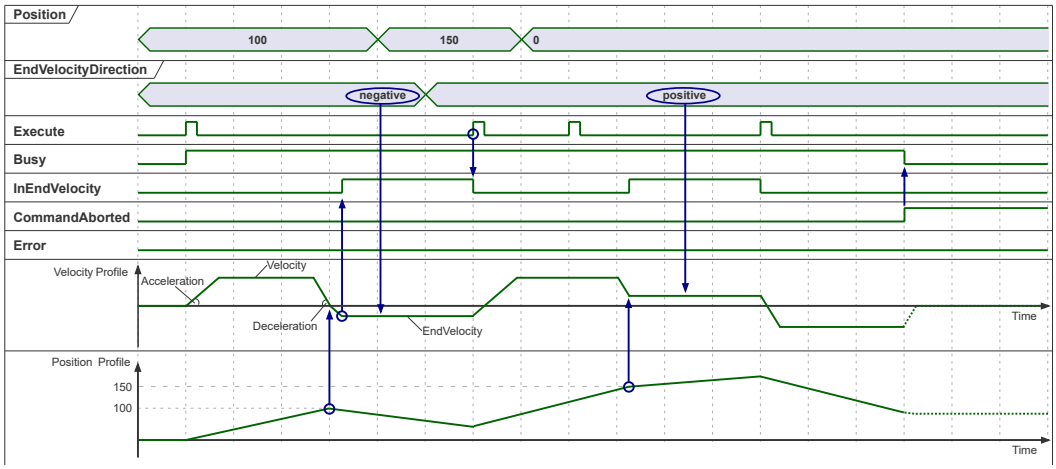
Axis type	Axis traveling direction
Finite	The direction is determined by the positional relationship of the current command position and the Position input. ● If the Position input is larger relative to the current command position, the axis travels in the positive direction. ● If the Position input is smaller relative to the current command position, the axis travels in the negative direction.
Modulo	MC_Direction specification

Traveling direction of the axis before the Position is reached	Traveling direction of the axis after the Position is reached	Behavior of the axis when the Position is reached
Positive direction	Positive direction	Determined by the magnitude relationship between the Velocity and EndVelocity values ● When Velocity < EndVelocity, The axis velocity, before the axis reaches the Position, starts acceleration from the Velocity value toward the EndVelocity value according to the Acceleration and Jerk values. Before the axis reaches the Position, the axis velocity reaches and gets constant at the EndVelocity value. ● When Velocity > EndVelocity, The axis, before it reaches the Position, starts deceleration from the Velocity value toward the EndVelocity value according to the Deceleration and Jerk values. Before the axis reaches the Position, the axis velocity reaches and gets constant at the EndVelocity value.
Negative direction	Negative direction	
Positive direction	Negative direction	The Velocity level, when the axis comes close to the Position, starts deceleration according to the Deceleration and Jerk values and decreases to 0 when the axis reaches the Position. The axis then restarts traveling in the direction specified in the EndVelocityDirection input, and when the velocity reaches the specified EndVelocity value, the axis keeps the velocity value.
Negative direction	Positive direction	

- Operation start
  - At the launch of the Execute, the axis starts traveling according to the Position, Velocity, Acceleration, Jerk, and Direction values.
- Operation stop
  - After the Position is reached, this FB causes the axis to keep moving at the specified EndVelocity value. Thus, the axis does not stop.
- Re-execution
  - While the Busy and InEndVelocity are TRUE, set the Execute to FALSE. Next, specify input values again. When the Execute is launched, the FB is executed with the new input values.
  - If the Execute is changed from FALSE to TRUE before the Position is reached, the new input values do not take effect and re-execution is not performed.
- Interruption of operation
  - If another FB that controls the same axis is called while the Busy is TRUE, the operation of this FB is interrupted.

■ Timing chart

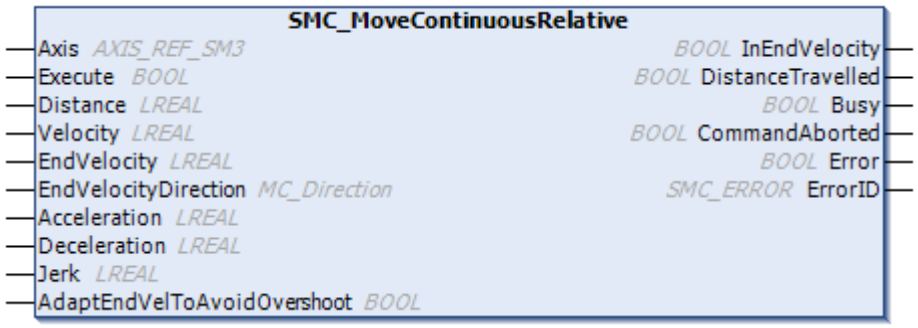
- At the launch of the Execute, the Busy goes TRUE.
- When the velocity reaches the EndVelocity value, the InEndVelocity goes TRUE. At the launch of the Execute, the InEndVelocity in the TRUE state is set to FALSE.
- When another FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE. The behavior of this FB after the CommandAborted is TRUE depends on the behavior of other FBs.


Position
100
150
0
EndVelocityDirection
negative
positive
Execute
Busy
InEndVelocity
CommandAborted
Error
Velocity Profile
Acceleration
Velocity
Deceleration
EndVelocity
Time
Position Profile
150
100
Time

5.6.8 SMC_MoveContinuousRelative (Relative Value Position Velocity Move)

This function block (FB) causes the axis to travel to a target position that is a result of the addition of a travel distance to the current command position. Then, after the axis reaches the target position, this FB causes the axis to keep moving at a specified velocity.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Distance	LREAL	0	Specifies the travel distance (u). The target position is a position that is a result of the addition of a Distance value to the current command position.
	Velocity	LREAL	0	Specifies the velocity (u/s).
	EndVelocity	LREAL	0	Specifies the velocity (u/s) after the axis reaches the target position.
	EndVelocityDirection	MC_Direction	current	Specifies the traveling direction after the axis reaches the target position. Specifies either “positive”, “negative”, or “current”. If “fastest” or “shortest” is specified, an error occurs.
	Acceleration	LREAL	0	Specifies the acceleration (u/s^2).
	Deceleration	LREAL	0	Specifies the deceleration (u/s^2).
	Jerk	LREAL	0	Specifies the jerk (u/s^3).
	AdaptEndVelToAvoidOvershoot	BOOL	FALSE	Do not use.
Output	InEndVelocity	BOOL	FALSE	TRUE: The velocity after the attainment of the target position has been reached.

Scope	Name	Type	Initial	Description
	DistanceTravelled	BOOL	FALSE	TRUE: The target distance has been traveled.
	Busy	BOOL	FALSE	TRUE: FB is operating.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from another FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ Detail of function

- Description of functions
  - This FB causes the axis to travel to a target position at the specified Velocity. The target position is a result of the addition of a Distance value to the current command position. After the axis reaches the target position, this FB then causes the axis to keep moving at the specified EndVelocity value.
  - The traveling direction of the axis before the target position is reached is determined by whether the Distance input is positive or negative.

    When the Distance input is positive, the axis travels in the positive direction.

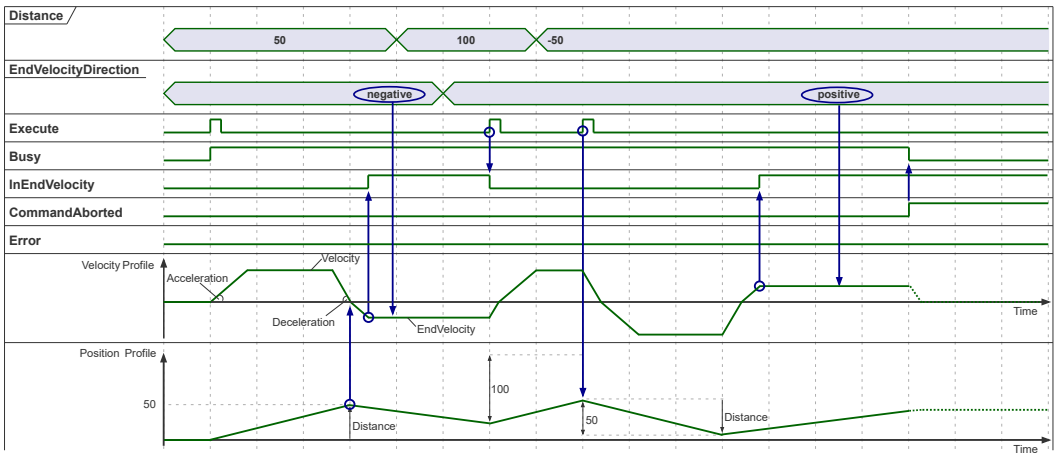
    When the Distance input is negative, the axis travels in the negative direction.
  - The traveling direction of the axis after the target position is reached is determined by the specified EndVelocityDirection value.
  - The behavior of the axis when the target position is reached is determined by the relationship between the traveling directions of the axis before and after the target position is reached.

Traveling direction of the axis before the target position is reached	Traveling direction of the axis after the target position is reached	Behavior of the axis when the target position is reached
Positive direction	Positive direction	Determined by the magnitude relationship between the Velocity and EndVelocity values ● When Velocity < EndVelocity, The axis velocity, before the axis reaches the target position, starts acceleration from the Velocity value toward the EndVelocity value according to the Acceleration and Jerk values. Before the axis reaches the target position, the axis velocity reaches and gets constant at the EndVelocity value. ● When Velocity > EndVelocity, The axis, before it reaches the target position, starts deceleration from the Velocity value toward the EndVelocity value according to the Deceleration and Jerk values. Before the axis reaches the target position, the axis velocity reaches and gets constant at the EndVelocity value.
Negative direction	Negative direction	
Positive direction	Negative direction	The Velocity level, when the axis comes close to the target position, starts deceleration according to the Deceleration and Jerk values and decreases to 0 when the axis reaches the target position. The axis then restarts traveling in the direction specified in the EndVelocityDirection input, and when the velocity reaches the specified EndVelocity value, the axis keeps the velocity value.
Negative direction	Positive direction	

- Operation start
  - At the launch of the Execute, the axis starts traveling according to the Distance, Velocity, Acceleration, and Jerk values. The target position is a result of the addition of a Distance value to the current command position.
- Operation stop
  - After the target position is reached, this FB causes the axis to keep moving at the specified EndVelocity value. Thus, the axis does not stop.
- Re-execution
  - Set the Execute to FALSE. Next, specify input values again. When the Execute is launched, the FB is executed with the new input values.
- Interruption of operation
  - If, during operation of this FB, another FB that controls the same axis is called, the operation of this FB is interrupted.

■ Timing chart

- At the launch of the Execute, the Busy goes TRUE.
- When the velocity reaches the EndVelocity value, the InEndVelocity goes TRUE. At the launch of the Execute, the InEndVelocity in the TRUE state is set to FALSE.
- When another FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE. The behavior of this FB after the CommandAborted is TRUE depends on the behavior of other FBs.



5.6.9 Example: Absolute Positioning, Relative Positioning

Here is an example of a program that uses "MC_MoveAbsolute" to move in the positive direction to position 180, and then uses "MC_MoveRelative" to move a distance of 4000.

■ Program examples

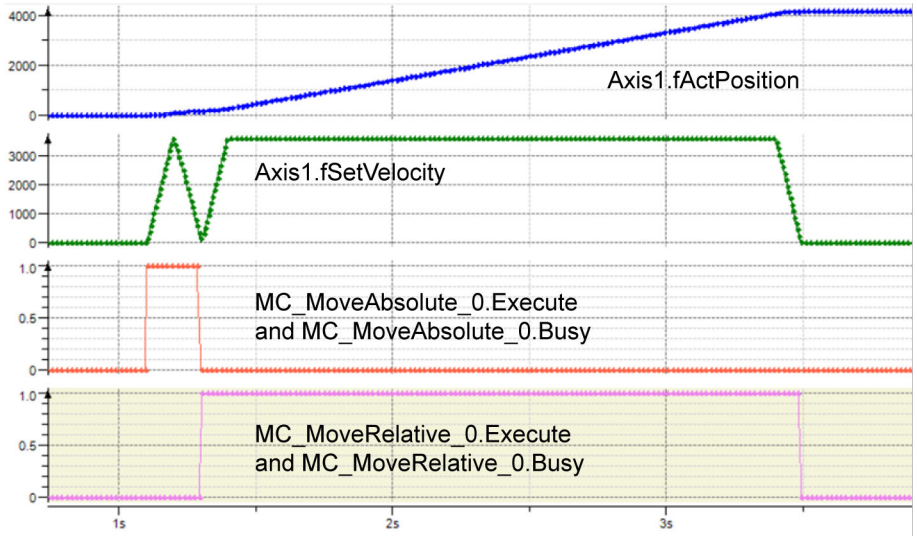
- Implementation Section

```
CASE Process OF
    0: // Servo ON
        MC_Power_0(
            Axis := Axis1,
            Enable := TRUE,
            bRegulatorOn := TRUE,
            bDriveStart := TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1: // Execute the MC_MoveAbsolute
        MC_MoveAbsolute_0(
            Axis := Axis1,
            Execute := TRUE,
            Position := 180,
            Velocity := 3600,
            Acceleration := 72000,
            Deceleration := 72000,
            Direction := positive
        );
        IF MC_MoveAbsolute_0.Done = TRUE THEN
                MC_MoveAbsolute_0(
                    Axis := Axis1,
                    Execute := FALSE,
                );
                Process := 2;
        END_IF
    2: // Execute the MC_MoveRelative
        MC_MoveRelative_0(
            Axis := Axis1,
            Execute := TRUE,
            Distance := 4000,
            Velocity := 3600,
            Acceleration := 72000,
            Deceleration := 72000
        );
        IF MC_MoveRelative_0.Done = TRUE THEN
            MC_MoveRelative_0(
                Axis := Axis1,
                Execute := FALSE
            );
            Process := 3;
        END_IF
    3: //End
```

```
        //No operation
END_CASE
```

"MC_MoveAbsolute" and "MC_MoveRelative" are started at the rising edge of the "Execute" flag. If processing is completed normally, the "Done" flag will be set to TRUE.

In this program example, the actual position eventually becomes 4180.



The "Busy" flag is set to TRUE during execution. While the "Busy" flag is set to TRUE, an instance of the function block must be called every cycle. Otherwise, the operation will terminate with an error.

5.6.10 Example: Target Position Change

Here's an example of changing the Position of "MC_MoveAbsolute" from 90 to 180. It is possible to switch the parameter while "MC_MoveAbsolute" is in execution (Busy=TRUE).

■ Program example

- Implementation Section

```
CASE Process OF
    0: // Servo ON
        MC_Power_0(
            Axis := Axisl ,
            Enable := TRUE ,
            bRegulatorOn := TRUE,
            bDriveStart := TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1: // Execute the MC_MoveAbsolute
        MC_MoveAbsolute_0(
            Axis := Axisl,
            Execute := TRUE,
```

```
            Position := 90,
            Velocity := 360,
            Acceleration := 3600,
            Deceleration := 3600,
            Direction := positive
        );
        IF MC_MoveAbsolute_0.Busy = TRUE THEN
                MC_MoveAbsolute_0(
                    Axis := Axis1,
                    Execute := FALSE,
                );
                Process := 2;
        END_IF
    2: // Change the position of the MC_MoveAbsolute_0
        MC_MoveAbsolute_0(
            Axis := Axis1,
            Execute := TRUE,
            Position := 180
        );
        IF MC_MoveAbsolute_0.Done = TRUE THEN
                MC_MoveAbsolute_0(
                    Axis := Axis1,
                    Execute := FALSE,
                );
                Process := 3;
        END_IF
    3: //End
        //No operation
END_CASE
```

Info.

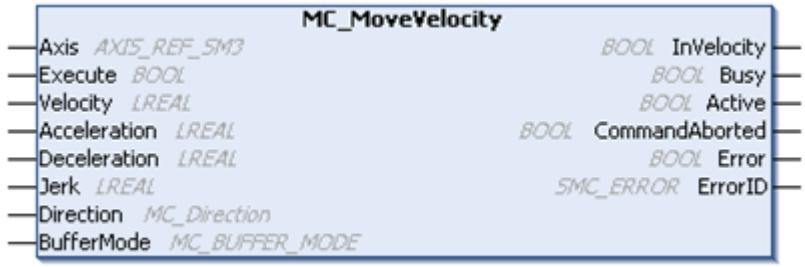
- If you need to change parameters such as the target position (position) during operation, please set Execute to FALSE once. After changing the parameters and setting Execute to TRUE again, the changes will be reflected.

5.7 Velocity Control

5.7.1 MC_MoveVelocity (Velocity Control)

This is a function block (FB) that controls the axis at a specified velocity.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Velocity	LREAL	0	Specifies the velocity (u/s).
	Acceleration	LREAL	0	Specifies the acceleration (u/s^2).
	Deceleration	LREAL	0	Specifies the deceleration (u/s^2).
	Jerk	LREAL	0	Specify the jerk (u/s^3)
	Direction	MC_Direction	current	Specifies the traveling direction of the axis. Specifies either “positive”, “negative”, or “current”. If “fastest” or “shortest” is specified, an error occurs.
	BufferMode	MC_BUFFER_MODE	Aborting	Specifies a buffer mode. The value is valid when this FB is a second FB.
Output	InVelocity	BOOL	FALSE	TRUE: The specified velocity has been reached for the first time.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	Active	BOOL	FALSE	TRUE: The second FB is being controlled.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred.

Scope	Name	Type	Initial	Description
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ MC_Direction (Enumeration type)

Name	Value	Description
positive	1	Travels in the positive direction.
negative	-1	Travels in the negative direction.
shortest	0	Not available. Do not specify this.
fastest	3	Not available. Do not specify this.
current	2	Travels in the positive direction if the axis is stopped. Travels in the current direction if the axis is in motion.

■ MC_BUFFER_MODE (Enumeration type)

On condition that this FB is connected as the second FB, the table below gives a description.

Name	Value	Description
Aborting	0	The operation of the first FB stops, and this FB starts operation instantly.
Buffered	1	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The movement starts at the velocity that the preceding movement has when the end condition is reached.
BlendingLow	2	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The command velocities of the first FB and this FB are compared, and the axis passes through the end position of the first FB operation at the lower command velocity.
BlendingPrevious	3	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The axis passes through the end position of the first FB operation at the velocity of the first FB command.
BlendingNext	4	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The axis passes through the end position of the first FB operation at the velocity of this FB command.
BlendingHigh	5	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The command velocities of the first FB and this FB are compared, and the axis passes through the end position of the first FB operation at the higher command velocity.

(Note 1) Refer to Table "Buffer Mode Operation Conditions."

■ Detail of function

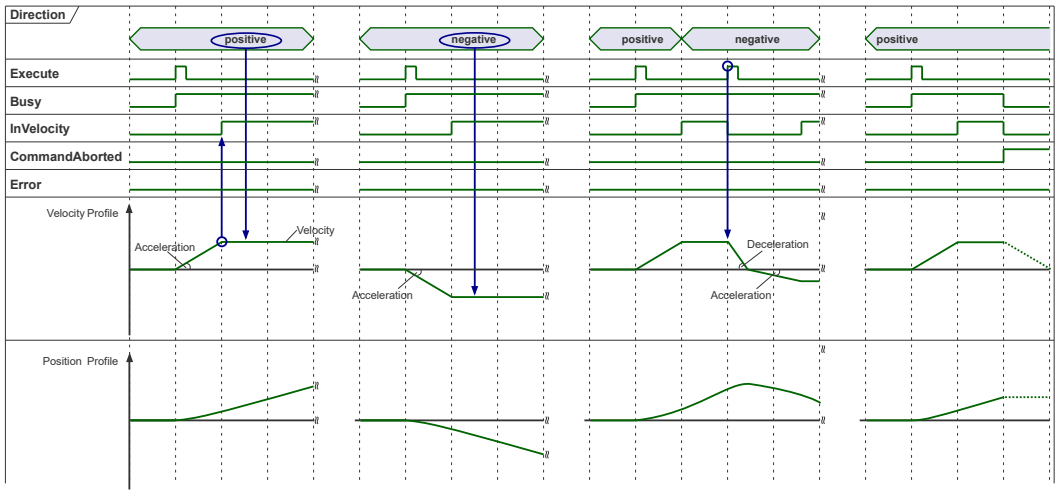
- Description of functions
  - The axis starts acceleration according to the Acceleration and Jerk values, and the axis velocity gets constant when it reaches the specified Velocity value.

- The Direction input is specified for the traveling direction of the axis, which is equivalent to the direction of the Velocity.
- When the axis is in motion and the current traveling direction of the axis is different from the Direction value during execution of this FB, the axis starts deceleration according to the Deceleration and Jerk values, and the Velocity level decreases to 0. The axis then starts acceleration according to the Acceleration and Jerk values, and the axis velocity gets constant when it reaches the specified Velocity value.
- This FB can be used in position control mode and in speed control mode.

- Operation start
  - At the launch of the Execute, the axis starts traveling according to the Velocity, Acceleration, Jerk, and Direction values.
- Operation stop
  - The axis velocity gets constant when it reaches the specified Velocity value. Thus, the axis does not stop. To stop the axis, set Velocity to 0 and execute again.
- Re-execution
  - Set the Execute to FALSE. Next, specify input values again. When the Execute is launched, the FB is executed with the new input values.
- Interruption of operation
  - If, during operation of this FB, another FB that controls the same axis is called, the operation of this FB is interrupted.

■ Timing chart

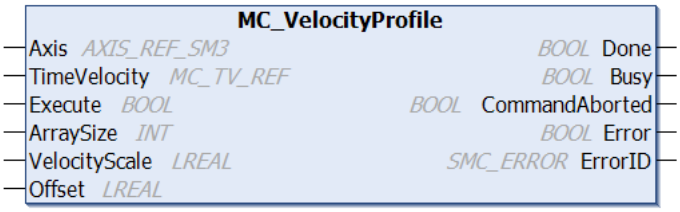
- At the launch of the Execute, the Busy goes TRUE.
- When the velocity reaches the Velocity value, the InVelocity goes TRUE. At the launch of the Execute, the InVelocity in the TRUE state is set to FALSE.
- When another FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE. If the InVelocity is in the TRUE state, it is set to FALSE. The behavior of this FB after the CommandAborted is TRUE depends on the behavior of other FBs.



5.7.2 MC_VelocityProfile (Velocity Profile Movement)

This is a function block (FB) that causes the axis to operate according to a velocity profile that consists of a combination of time and velocity.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
	TimeVelocity	MC_TV_REF	-	Specifies the velocity profile.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	ArraySize	INT	0	Specifies the number of profile points to be executed.
	VelocityScale	LREAL	1	Velocity scaling Multiplies the velocity value of the profile by the specified value.
	Offset	LREAL	0	Velocity offset (u/s) Adds the specified value to the velocity value of the profile.
Output	Done	BOOL	FALSE	TRUE: Motion specified by the velocity profile has been completed.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ MC_TV_REF (Structure)

Member	Type	Description
Number_of_pairs	INT	Not used. Values are ignored.
isAbsolute	BOOL	Methods of specifying profile velocities TRUE: Absolute value FALSE: Relative value
MC_TV_Array	ARRAY [1..100] OF SMC_TV	A set of velocity profile data

Member	Type	Description
		(1st point to 100th point)

■ SMC_TV (Structure)

Member	Type	Description
delta_time	TIME	Time of the profile Period of time spanned from the time at the last profile acceleration
velocity	LREAL	Velocity of profile data

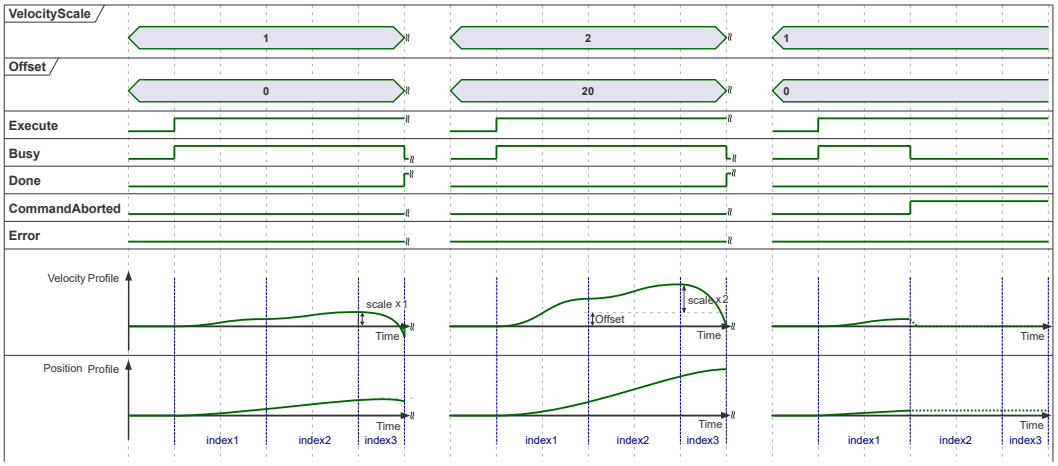
■ Detail of function

- Description of functions
  - This FB causes the axis to operate according to the velocity profile.
  - Up to 100 points that are each a combination of time and velocity values can be registered in the velocity profile.
  - This FB can be used in position control mode and in speed control mode.
- Start operation
  - At the launch of the Execute, the axis starts traveling according to the TimeVelocity description.
- Operation stop
  - When the axis has traveled through a trajectory equivalent to the number of profile points specified in ArraySize, the axis immediately stops. By specifying 0 for the velocity in the last section of the profile, the axis can make a deceleration stop.
- Re-execution
  - Set the Execute to FALSE. Next, specify input values again. When the Execute is launched, the FB is executed with the new input values.
- Interruption of operation
  - If, during operation of this FB, another FB that controls the same axis is called, the operation of this FB is interrupted.

■ Timing chart

- At the launch of the Execute, the Busy goes TRUE.
- When the axis has traveled through a trajectory equivalent to the number of profile points specified in ArraySize, the Busy goes FALSE and the Done goes TRUE. The Done in the TURE state goes FALSE when the Execute is set to FALSE.
- When another FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE. The behavior of this FB after the CommandAborted is TRUE depends on the behavior of other FBs.
- The timing chart is plotted when MC_TV_Array, a structure member of TimeVelocity, has values shown in the table below. In this example, isAbsolute, a structure member of TimeVelocity, is set to TRUE.

index	delta_time	velocity
1	Time#2s	10
2	Time#2s	20
3	Time#1000ms	-10



Info.

- While the axis keeps driving, do not set delta_time to 0 ms. Otherwise, operation cannot be properly executed in the section for which 0 ms is specified and subsequent sections.

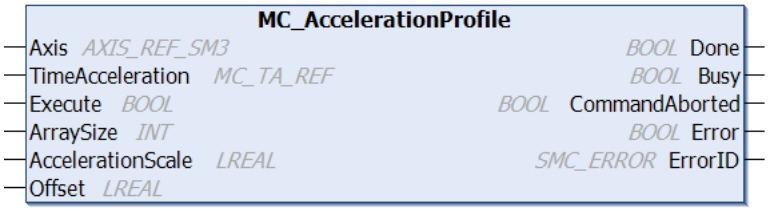
REFERENCE

5.6.6 Default Setting for Variables of the MC_TP_REF Type Structure

5.7.3 MC_AccelerationProfile (Acceleration Profile Movement)

This is a function block (FB) that causes the axis to operate according to acceleration profile data that consists of a combination of time and acceleration.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
	TimeAcceleration	MC_TA_REF	-	Specifies the acceleration profile.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	ArraySize	INT	0	Specifies the number of profile points to be executed.
	AccelerationScale	LREAL	1	Acceleration scaling Multiplies the acceleration value of the profile by the specified value.
	Offset	LREAL	0	Acceleration offset (u/s^2) Adds the specified value to the acceleration value of the profile.
Output	Done	BOOL	FALSE	TRUE: Motion specified by the acceleration profile has been completed.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ MC_TA_REF (Structure)

Member	Type	Description
Number_of_pairs	INT	Not used. Values are ignored.
isAbsolute	BOOL	Methods of specifying profile accelerations TRUE: Absolute value FALSE: Relative value

Member	Type	Description
MC_TA_Array	ARRAY [1..100] OF SMC_TA	A set of acceleration profile data (1st point to 100th point)

■ SMC_TA (Structure)

Member	Type	Description
delta_time	TIME	Time of the profile Period of time spanned from the time at the last profile acceleration
Acceleration	LREAL	Acceleration of profile data

■ Detail of function

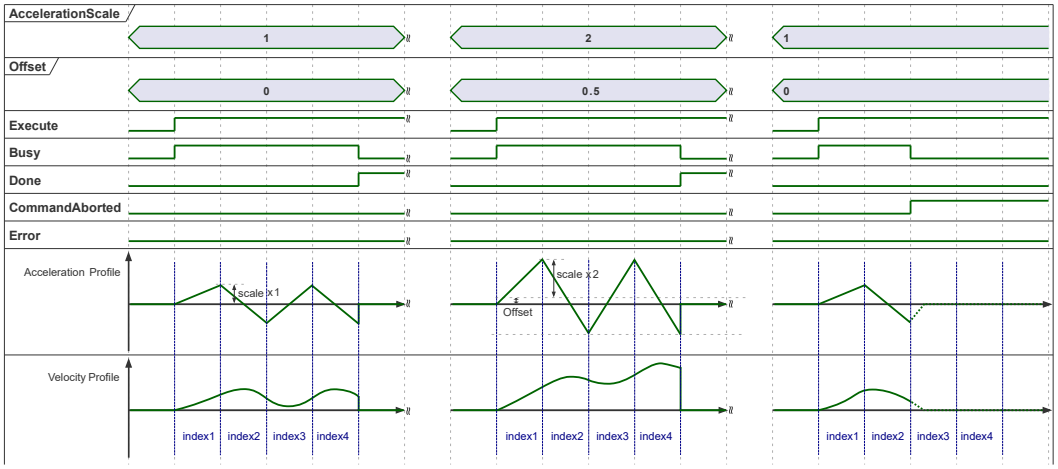
- Description of functions
  - This FB causes the axis to operate according to the acceleration profile.
  - Up to 100 points that are each a combination of time and acceleration values can be registered in the acceleration profile.
  - This FB can be used in position control mode and in speed control mode.
- Start operation
  - At the launch of the Execute, the axis starts traveling according to the TimeAcceleration description.
- Operation stop
  - When the axis has traveled through a trajectory equivalent to the number of profile points specified in ArraySize, the axis immediately stops.
- Re-execution
  - Set the Execute to FALSE. Next, specify input values again. When the Execute is launched, the FB is executed with the new input values.
- Interruption of operation
  - If, during operation of this FB, another FB that controls the same axis is called, the operation of this FB is interrupted.

■ Timing chart

- At the launch of the Execute, the Busy goes TRUE.
- When the axis has traveled through a trajectory equivalent to the number of profile points specified in ArraySize, the Busy goes FALSE and the Done goes TRUE. The Done in the TURE state goes FALSE when the Execute is set to FALSE.
- When another FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE. The behavior of this FB after the CommandAborted is TRUE depends on the behavior of other FBs.
- The timing chart is plotted when MC_TP_Array, a structure member of TimeAcceleration, has values shown in the table below. In this example, isAbsolute, a structure member of TimeAcceleration, is set to TRUE.

index	delta_time	Acceleration
1	Time#1s	1
2	Time#1s	-1
3	Time#1000ms	1

index	delta_time	Acceleration
4	Time#1000ms	-1



Info.

- While the axis keeps driving, do not set delta_time to 0 ms. Otherwise, operation cannot be properly executed in the section for which 0 ms is specified and subsequent sections.

REFERENCE

5.6.6 Default Setting for Variables of the MC_TP_REF Type Structure

5.7.4 Example: Speed Control

Here is an example of a program that performs velocity control in the positive direction with a speed of 360 u/s, acceleration of 3600 u/s^2, and deceleration of 3600 u/s^2.

■ Program example

- Implementation Section

```
CASE Process OF
    0: // Servo ON
        MC_Power_0(
            Axis := Axisl ,
            Enable := TRUE ,
            bRegulatorOn := TRUE,
            bDriveStart := TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1: // Change controller mode to SMC_velocity
        SMC_SetControllerMode_0(
            Axis := Axisl,
            bExecute := TRUE,
            nControllerMode := SMC_velocity
        );
        IF SMC_SetControllerMode_0.bDone = TRUE THEN
            SMC_SetControllerMode_0(
                Axis := Axisl,
                bExecute := FALSE
            );
            Process := 2;
        END_IF
    2: // Execute the PMC_SetTorque
        MC_MoveVelocity_0(
            Axis := Axisl,
            Execute := TRUE,
            Velocity := 360,
            Acceleration := 3600,
            Deceleration := 3600,
            Direction := positive
        );
        IF MC_MoveVelocity_0.InVelocity =TRUE THEN
            MC_MoveVelocity_0(
                Axis := Axisl,
                Execute :=  FALSE
            );
            Process := 3;
        END_IF
    3: // Change Torque to 0
        MC_MoveVelocity_0(
            Axis := Axisl,
            Execute := TRUE,
            Velocity := 0,
            Acceleration := 3600,
```

```
            Deceleration := 3600,
            Direction :=positive
        );
END_CASE
```

Info.

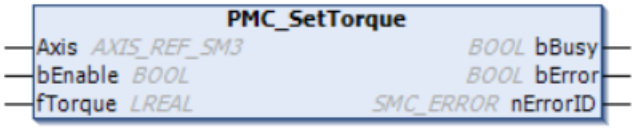
- To change parameter settings such as target velocity ("Velocity") during operation, temporarily set the "Execute" flag to FALSE beforehand. After parameter settings have been changed, if the "Execute" flag is set back to TRUE, the changed parameter settings will be applied.

5.8 Torque Control

5.8.1 PMC_SetTorque (Torque Control)

This is a function block (FB) that controls torque. Torque is specified as a percentage (%) of the rated torque value of the servo amplifier. When using this FB, set the control mode to torque control mode in advance using SMC_SetControllerMode.

■ Icon



■ Parameter

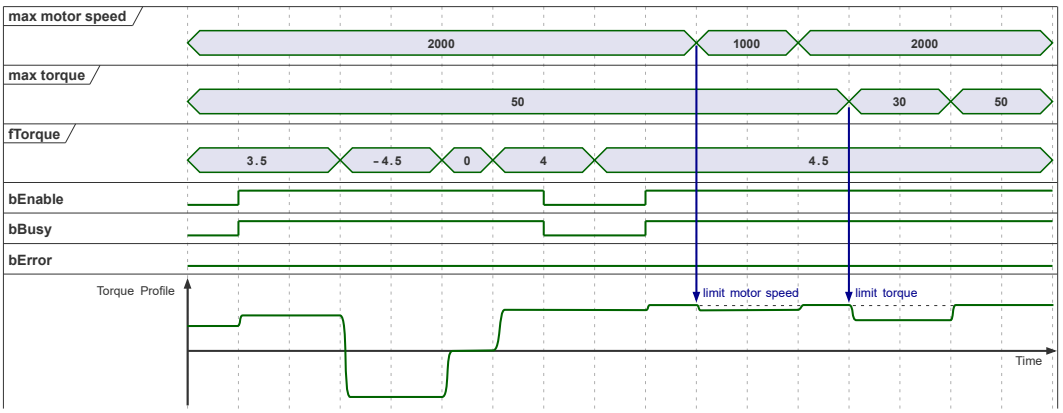
Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bEnable	BOOL	FALSE	TRUE: Torque output can be changed according to fTorque.
	fTorque	LREAL	0	Specifies torque as a percentage (%) of the rated torque value of the servo amplifier.
Output	bBusy	BOOL	FALSE	TRUE: The FB is in operation.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	nErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ Detail of function

- Description of functions
  - fTorque is specified as a percentage (%) of the rated torque value of the servo amplifier. The rated torque value can be checked using the IEC object fFactorTor for the axis.
- Torque command value
  - The torque command value is the value of fTorque. The torque command value can be changed when Enable is set to TRUE. It cannot be changed when Enable is set to FALSE. The torque command continues with the current fTorque value even when Enable is set to FALSE.
  - When fTorque is specified as 0, the FB operates according to the mechanism and load application in the operating environment.
- Interruption of operation
  - If, during operation of this FB, another FB that controls torque is called for the same axis, the operation of this FB is interrupted.

■ Timing chart

- When the Enable is set to TRUE, Busy changes to TRUE.
- If the value of fTorque is changed while Enable is TRUE, the value is immediately reflected.
- In the figure below, the motor speed is limited when max motor speed is 1000, causing the torque output to be limited. Similarly, the torque command value is limited when max troque is 30, causing the torque output to be limited.



Info.

- When using this FB, set the control mode to torque control mode in advance using SMC_SetControllerMode.
- The operation of this FB is not stopped by MC_Stop or MC_Halt.
- When performing torque control using the MINAS A5B/A6B, select PDO mapping 2 or 4. In addition, preset the parameters shown in the table below. With the default value, the axis does not move because the maximum value is 0.

PDO mapping	Parameter	Description	Default value
2	Max motor speed (16#6080:00)	Maximum speed	0
4	Max torque (16#6072:00)	Maximum torque	0
	Max motor speed (16#6080:00)	Maximum speed	0

REFERENCE

5.3.1 SMC_SetControllerMode (Control Mode Setting)
5.4.1 MC_Stop (Forced Stop)

5.8.2 SMC_SetTorque (Torque Control)

This is a function block (FB) that controls torque. Torque is specified in N-m (Newton-meters). When using this FB, set the control mode to torque control mode in advance using SMC_SetControllerMode.

■ Icon

■ Parameter

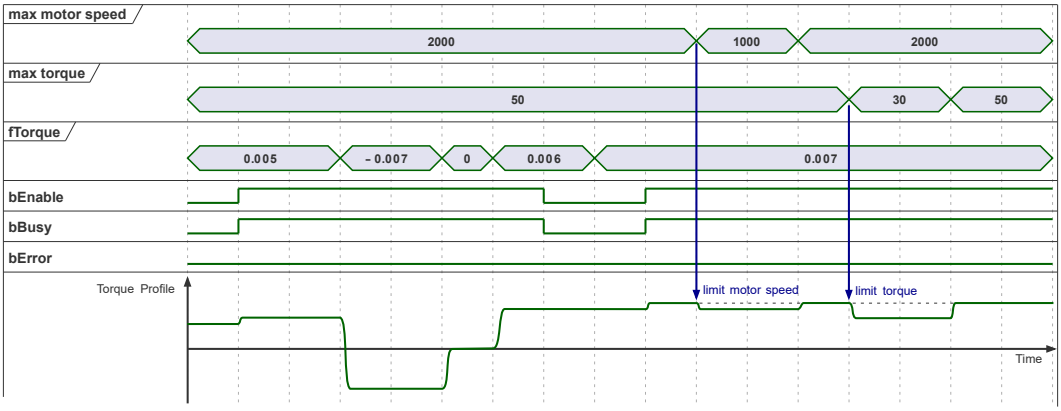
Scope	Name	Type	Default	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bEnable	BOOL	FALSE	TRUE: Torque output can be changed according to fTorque.
	fTorque	LREAL	0	Specifies the torque (N·m, N)
Output	bBusy	BOOL	FALSE	TRUE: The FB is in operation.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	nErrorID	SMC_ERROR	SMC_NO_ERROR	Error ID output

■ Detail of function

- Description of functions
  - fTorque is specified in N-m (Newton-meters). The rated torque value can be checked using the IEC object fFactorTor for the axis.
- Torque command value
  - The torque command value is the value of fTorque. The torque command value can be changed when Enable is set to TRUE. It cannot be changed when Enable is set to FALSE. The torque command continues with the current fTorque value even when Enable is set to FALSE.
  - When fTorque is specified as 0, the FB operates according to the mechanism and load application in the operating environment.
- Interruption of operation
  - If, during operation of this FB, another FB that controls torque is called for the same axis, the operation of this FB is interrupted.

■ Timing chart

- When the Enable is set to TRUE, Busy changes to TRUE.
- If the value of fTorque is changed while Enable is TRUE, the value is immediately reflected.
- In the figure below, the motor speed is limited when max motor speed is 1000, causing the torque output to be limited. Similarly, the torque command value is limited when max troque is 30, causing the torque output to be limited.



Info.

- When using this FB, set the control mode to torque control mode in advance using SMC_SetControllerMode.
- The operation of this FB is not stopped by MC_Stop or MC_Halt.
- When performing torque control using the MINAS A5B/A6B, select PDO mapping 2 or 4. In addition, preset the parameters shown in the table below. With the default value, the axis does not move because the maximum value is 0.

PDO mapping	Parameter	Description	Default value
2	Max motor speed (16#6080:00)	Maximum speed	0
4	Max torque (16#6072:00)	Maximum torque	0
	Max motor speed (16#6080:00)	Maximum speed	0

5.8.3 Example: Torque Control

Here is an example of a program that executes torque control at 30% of the rated torque.

■ Program example

- Implementation Section

```
CASE Process OF
    0: // Servo ON
        MC_Power_0(
            Axis := Axisl ,
            Enable := TRUE ,
            bRegulatorOn := TRUE,
            bDriveStart := TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1: // Change controller mode to SMC_velocity
        SMC_SetControllerMode_0(
            Axis := Axisl,
            bExecute := TRUE,
            nControllerMode := SMC_torque
        );
        IF SMC_SetControllerMode_0.bDone = TRUE THEN
            SMC_SetControllerMode_0(
                Axis := Axisl,
                bExecute := FALSE
            );
            Process := 2;
        END_IF
    2: // Execute the PMC_SetTorque
        PMC_SetTorque_0(
            Axis := Axisl,
            bEnable := TRUE,
            fTorque := 30
        );
        PMC_ReadActualTorque_0(
            Axis := Axisl,
            Enable := TRUE
        );
        IF PMC_ReadActualTorque_0.Valid = TRUE THEN
            IF PMC_ReadActualTorque_0.Torque >= 30 THEN
                PMC_SetTorque_0(
                    Axis := Axisl,
                    bEnable := FALSE
                );
                Process := 3;
            END_IF
        END_IF
    3: // Change Torque to 0
        PMC_SetTorque_0(
            Axis := Axisl,
            bEnable := TRUE,
```

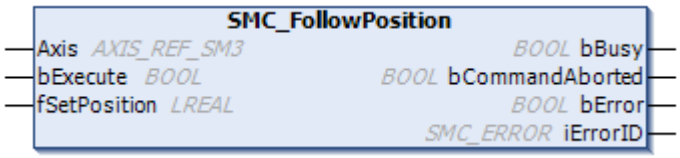
```
            fTorque := 0
        );
END_CASE
```

5.9 Direct commands

5.9.1 SMC_FollowPosition (Target Position Command at Every Interval)

This is a function block (FB) that writes the target position at every control period, causing the axis to travel.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Axis	AXIS_REF_SM3	-	Reference to the axis
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	fSetPosition	LREAL	0	Sets the target position (u).
Output	bBusy	BOOL	FALSE	TRUE: FB is in progress.
	bCommandAborted	BOOL	FALSE	TRUE: An interruption is caused by another FB.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	iErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ Detail of function

- Description of functions
  - This FB causes the axis to travel to the specified fSetPosition.
  - The traveling direction of the axis before the target position is reached is determined by whether the fSetPosition input is positive or negative.

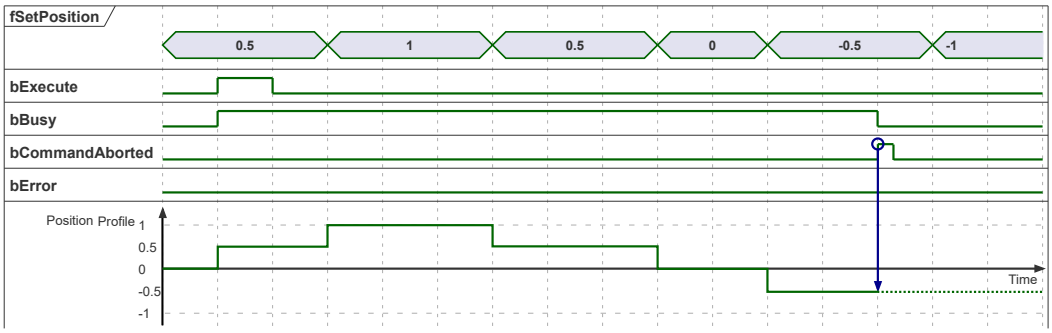
    When the fSetPosition input is positive, the axis travels in the positive direction.
    When the fSetPosition input is negative, the axis travels in the negative direction.

    However, in the case of the modulo axis, the direction in which the axis travels from the current value to the target position in the shortest distance will be used.
  - Even if the bExecute input is changed from TRUE to FALSE, the bBusy output remains TRUE.
  - This FB can be used in position control mode and in speed control mode.
- Operation start
  - At the launch of the bExecute, the axis starts traveling toward the specified fSetPosition.
- Re-setting
  - When the FB is in operation (bBusy is TRUE), set fSetPosition again.
- Interruption of operation

- If, during operation of this FB, another FB that controls the same axis is called, the operation of this FB is interrupted.

■ Timing chart

- At the launch of the bExecute, the bBusy changes to TRUE.
- While bBusy is TRUE, this FB writes the target position specified by fSetPosition to the axis.
- When another FB that controls the same axis is called while the bBusy is TRUE, the bCommandAborted changes to TRUE. The behavior of this FB after the bCommandAborted is TRUE depends on the behavior of other FBs.



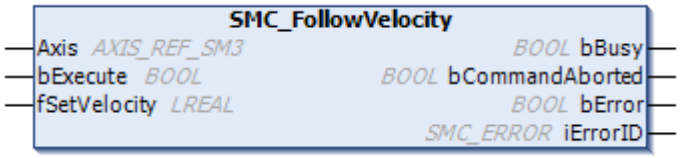
Note

- If there is a large difference between the current value and the target command value, fSetPosition may cause sudden movements since it commands the target position as it is at every control interval. Set the argument so as to ensure smooth control command movements.

5.9.2 SMC_FollowVelocity (Target Velocity Command at Every Interval)

This is a function block (FB) that writes the target velocity at every control period, causing the axis to travel.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Axis	AXIS_REF_SM3	-	Reference to the axis
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	fSetVelocity	LREAL	0	Specifies the target velocity (u/s).
Output	bBusy	BOOL	FALSE	TRUE: FB is in progress.
	bCommandAborted	BOOL	FALSE	TRUE: An interruption is caused by another FB.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	iErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ Detail of function

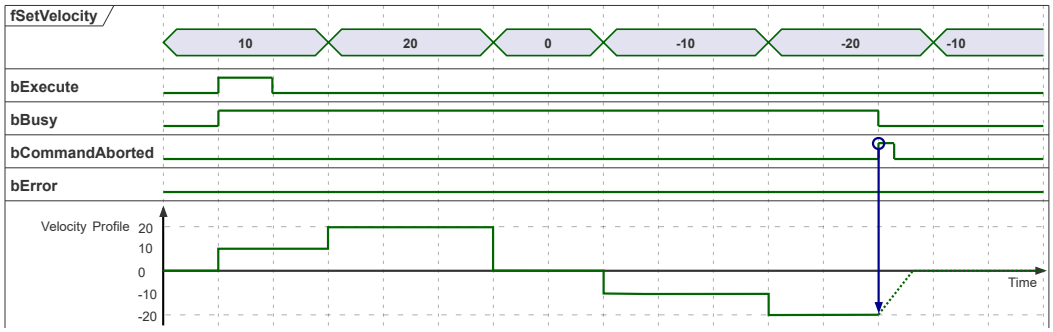
- Description of functions
  - This FB causes the axis to travel at the specified fSetVelocity value.
  - The traveling direction of the axis is determined by whether the fSetVelocity input is positive or negative.

    When the fSetVelocity input is positive, the axis travels in the positive direction.
    When the fSetVelocity input is negative, the axis travels in the negative direction.
  - Even if the bExecute input is changed from TRUE to FALSE, the bBusy output remains TRUE.
  - This FB can be used in position control mode and in speed control mode.
- Operation start
  - At the launch of the bExecute, the axis starts traveling according to the specified fSetVelocity value.
- Operation stop
  - The axis velocity gets constant when it reaches the specified fSetVelocity value. Thus, the axis does not stop. To stop the axis, set fSetVelocity to 0.
- Re-setting
  - When the FB is in operation (bBusy is TRUE), set fSetVelocity again.
- Interruption of operation

- If, during operation of this FB, another FB that controls the same axis is called, the operation of this FB is interrupted.

■ Timing chart

- At the launch of bExecute, the bBusy changes to TRUE.
- While bBusy is TRUE, this FB writes the target velocity specified by fSetVelocity.
- When another FB that controls the same axis is called while the bBusy is TRUE, the bCommandAborted changes to TRUE. The behavior of this FB after the bCommandAborted is TRUE depends on the behavior of other FBs.



Note

- If there is a large difference between the current velocity and the target command value, fSetVelocity may cause sudden movements since it commands the target velocity as it is at every control interval. Set the argument so as to ensure smooth control command movements.

5.10 Buffer Mode

The buffer mode is a function that controls the operation start timing when an axis control function block is executed while another axis control function block (FB) is being executed. For convenience, the FB executed first is called the first FB and the FB executed later is called the second FB.

5.10.1 Buffer Mode Execution Rules

The following rules apply to the buffer mode.

- The buffer mode is enabled when the second FB is executed while the first FB is being executed or before it reaches the end condition.
- The operation start timing of the second FB is specified in MC_BUFFER_MODE described later.
- If the first FB is provided with an Active signal port, the second FB can be operated in the buffer mode by connecting the Execute signal port of the second FB.
- If the buffer mode not usable for the second FB is specified, the second FB does not operate and an error is output.
- When performing Buffered operation or Blending operation, the first FB and the second FB need to be executed in the order as described on the POU. If they are executed in a different order, the second FB does not operate and an error is output.

  OK example: Described and executed in the following order: MC_MoveRelative_0 → MC_MoveRelative_1.

```
MC_MoveRelative_0(
    Axis:=Axis1,
    Execute:=bExe_mr0,
    Distance:=100,
    Velocity:=20,
    Acceleration:=100,
    Deceleration:=100
);

MC_MoveRelative_1(
    Axis:=Axis1,
    Execute:=bExe_mr1,
    Distance:=150
    Velocity:=5,
    Acceleration:=50,
    Deceleration:=50,
    BufferMode:=MC_BUFFER_MODE.BlendingHigh
);

bExe_mr0:=TRUE;
IF MC_MoveRelative_0.Active = TRUE THEN
    bExe_mr1:=TRUE;
END_IF
```

  NG example: Described as follows: MC_MoveRelative_0 → MC_MoveRelative_1, but executed as follows: MC_MoveRelative_1 → MC_MoveRelative_0

```
MC_MoveRelative_0(
    Axis:=Axis1,
    Execute:=bExe_mr0,
    Distance:=100,
    Velocity:=20,
    Acceleration:=100,
    Deceleration:=100,
    BufferMode:=MC_BUFFER_MODE.BlendingLow
);

MC_MoveRelative_1(
    Axis:=Axis1,
    Execute:=bExe_mr1,
    Distance:=150
    Velocity:=5,
    Acceleration:=50,
    Deceleration:=50
);

bExe_mr1:=TRUE;
IF MC_MoveRelative_1.Active = TRUE THEN
    bExe_mr0:=TRUE;
END_IF
```

■ Second FB in buffer mode

The following FBs are supported as the second FB in the buffer mode.

FB name	Settable buffer mode (MC_BUFFER_MODE)
MC_MoveAbsolute	Aborting, Buffered, BlendingLow, BlendingPrevious, BlendingNext, BlendingHigh
MC_MoveRelative	
MC_MoveVelocity	
MC_GearIn	Aborting, Buffered, BlendingPrevious
MC_GearInPos	

■ First FB in the buffer mode and the end condition

The following FBs are supported as the first FB in the buffer mode. The table also shows the output value indicating that the operation of the first FB has ended and the buffer mode for which the second FB can be set.

FB name	Output value indicating that the operation of the first FB has ended	Buffer mode for which the second FB can be set (MC_BUFFER_MODE)
MC_MoveAbsolute	Done	Aborting, Buffered, BlendingLow, BlendingPrevious, BlendingNext, BlendingHigh
MC_MoveRelative		
MC_MoveVelocity	InVelocity	Aborting, Buffered
MC_GearIn	InGear	
MC_GearInPos	Insync	
MC_MoveAdditive	Done	
SMC_MoveContinuousAbsolute	InEndVelocity	
SMC_MoveContinuousRelative		
MC_PositionProfile	Done	
MC_VelocityProfile		
MC_AccelerationProfile		
MC_CamIn	EndOfProfile	
MC_CamOut	Done	
MC_GearOut		

5.10.2 MC_BUFFER_MODE (Enumeration type)

The following table lists the buffer mode that can be set for the second FB. Set a buffer mode in the BufferMode input of FB.

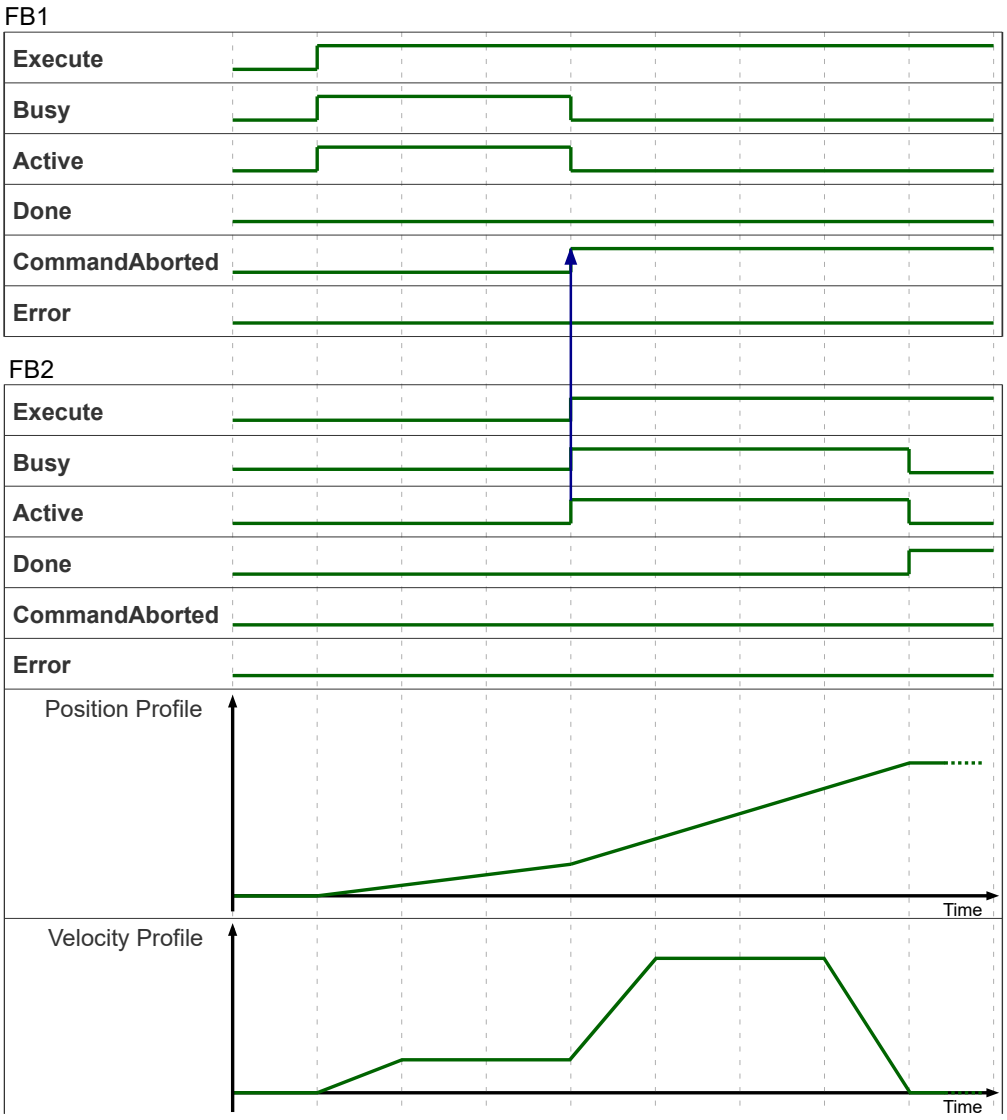
Name	Value	Description
Aborting	0	The operation of the first FB stops, and the second FB starts operation instantly. Default value
Buffered	1	When the first FB operation satisfies the end condition(Note 1), the second FB starts operation instantly. The velocity at which the second FB starts operation is the velocity at which the first FB has reached the end condition.
BlendingLow	2	When the first FB operation satisfies the end condition(Note 1), the second FB starts operation instantly. The command velocities of the first FB and the second FB are compared, and the axis passes through the end position of the first FB operation at the lower velocity.
BlendingPrevious	3	When the first FB operation satisfies the end condition(Note 1), the second FB starts operation instantly. The axis passes through the end position of the first FB operation at the velocity of the first FB command.
BlendingNext	4	When the first FB operation satisfies the end condition(Note 1), the second FB starts operation instantly. The axis passes through the end position of the first FB operation at the velocity of the second FB command.
BlendingHigh	5	When the first FB operation satisfies the end condition(Note 1), the second FB starts operation instantly. The command velocities of the first FB and the second FB are compared, and the axis passes through the end position of the first FB operation at the higher velocity.

(Note 1) Refer to "5.10.1 Buffer Mode Execution Rules".

■ Timing chart

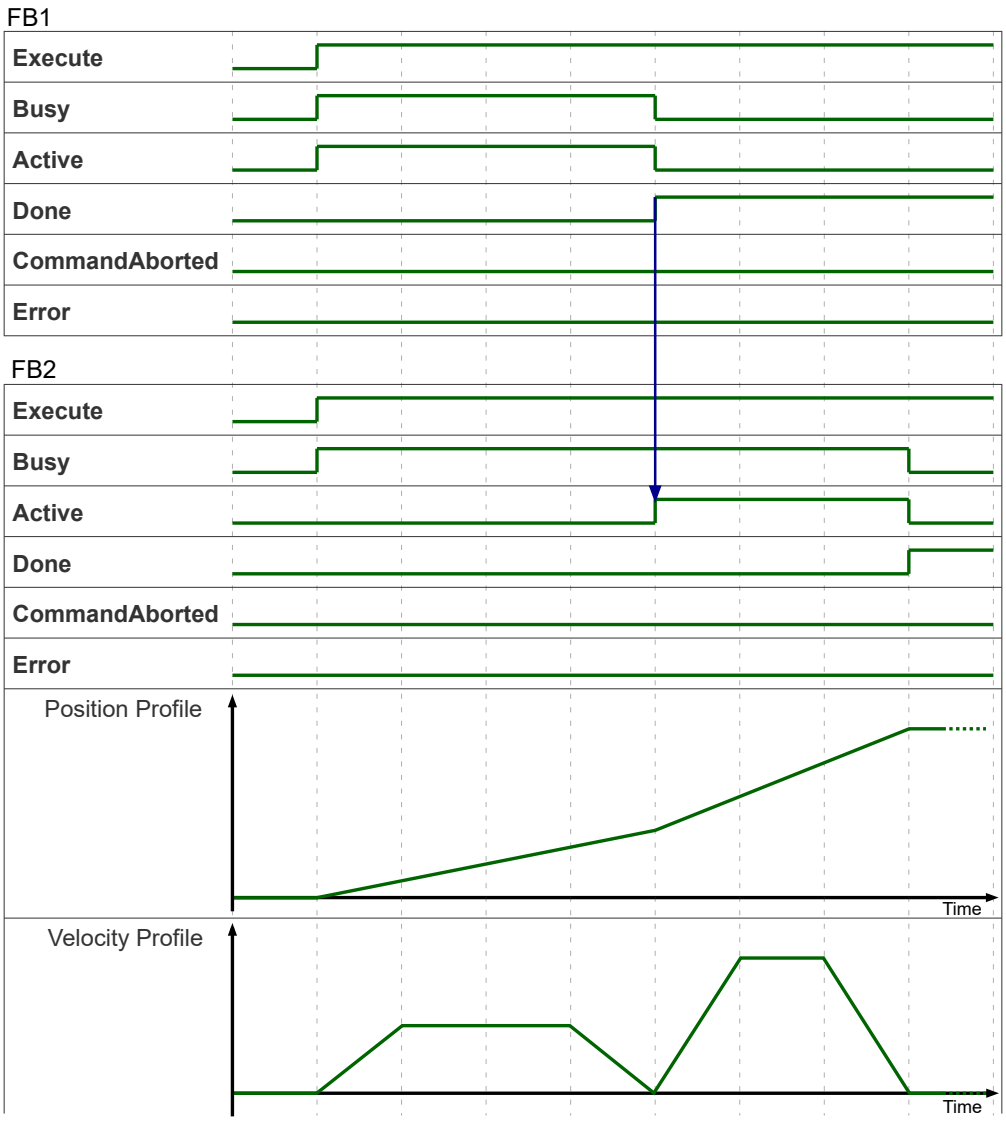
The following section shows the timing chart for each buffer mode. Assume that FB1 is executed as the first FB and FB2 is executed as the second FB.

- Aborting
  - The timing chart shows the case where the second FB is executed with BufferMode = Aborting.
  - When the second FB is executed, the first FB is immediately interrupted (CommandAborted = TRUE) and the operation transitions to the second FB operation as is.



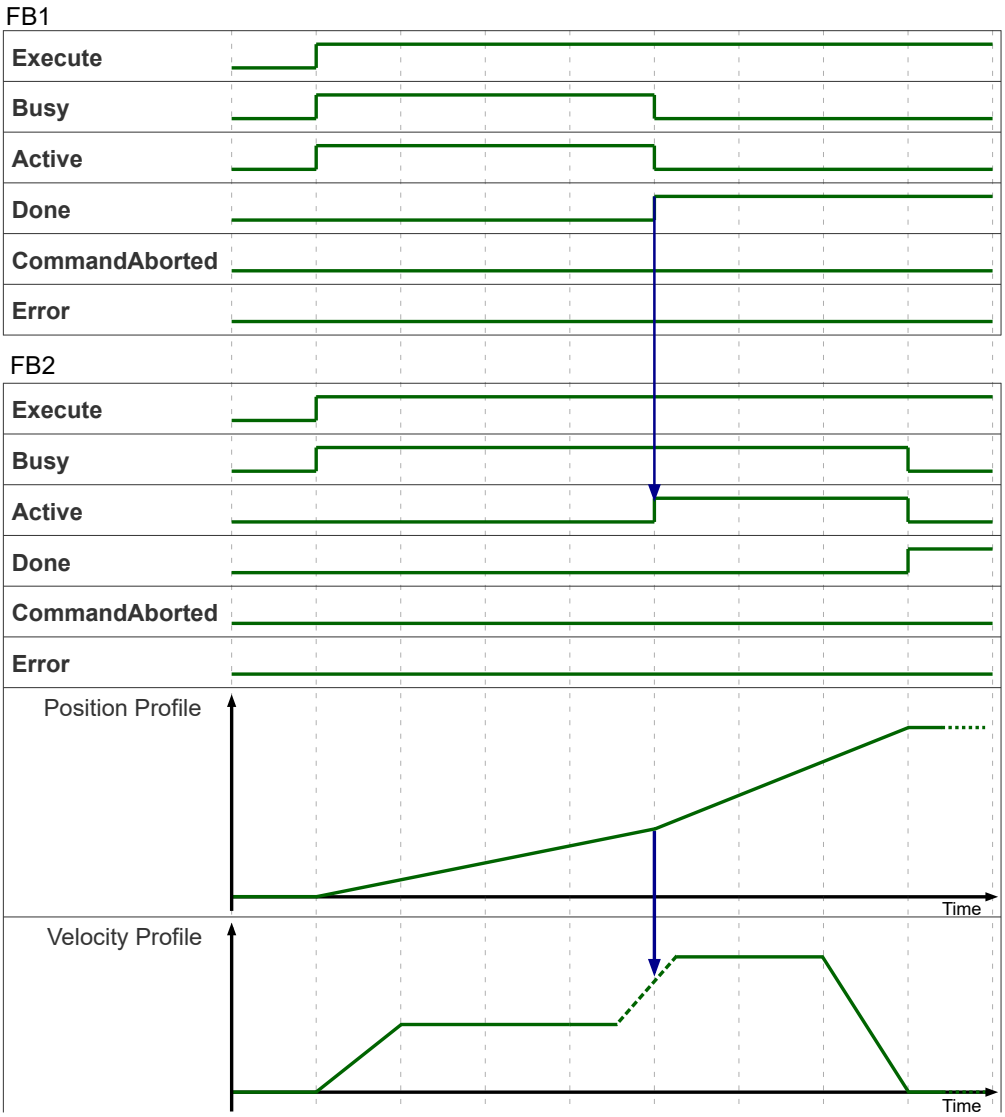
- Buffered

- The timing chart shows the case where the second FB is executed with BufferMode = Buffered.
- When the second FB is executed, the function block stays in Busy state (Busy = TRUE) and waits until the first FB operation ends.
- When the first FB operation ends (Done = TRUE), the second FB starts operation.

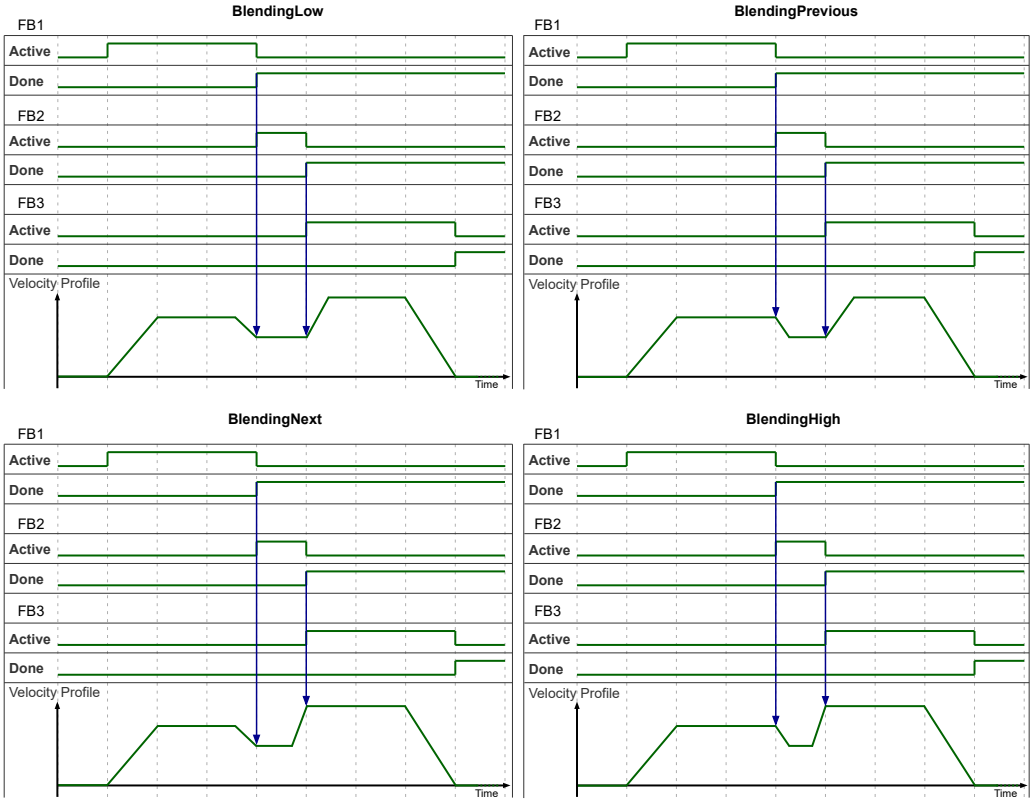


● Blending action（BlendingLow、BlendingPrevious、BlendingNext、BlendingHigh）

- The timing chart shows the case where the second FB is executed in Blending operation.
- As in the case with the Buffered mode, the second FB waits in the Busy state until the first FB operation ends.
- When the operation of the first FB ends, the second FB starts operation instantly.
- The velocity at the end position of the first FB operation (= starting position of the second FB operation) varies depending on which Blending mode is set.



The velocity (Blending velocity) at the FB transition position in each Blending mode is as follows. The timing chart below shows the case where the velocities are set as follows: command velocity of the third FB (=FB3) > command velocity of the first FB (=FB1) > command velocity of the second FB (=FB2).

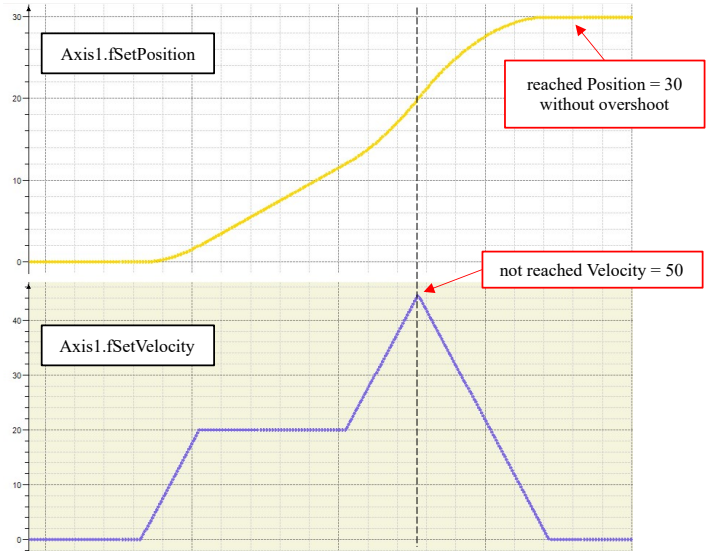

BlendingLow
FB1
Active
Done
FB2
Active
Done
FB3
Active
Done
Velocity Profile
Time
BlendingPrevious
FB1
Active
Done
FB2
Active
Done
FB3
Active
Done
Velocity Profile
Time
BlendingNext
FB1
Active
Done
FB2
Active
Done
FB3
Active
Done
Velocity Profile
Time
BlendingHigh
FB1
Active
Done
FB2
Active
Done
FB3
Active
Done
Velocity Profile
Time

Info.

- If the movement direction of the first FB is different from that of the second FB, the Blending velocity becomes 0.
- If the distance that the subsequent feedback (FB) action needs to travel is shorter than the blending speed, the blending speed will be automatically adjusted to prevent overshooting of the subsequent FB movement.

First FB
MC_MoveAbsolute_0
- Position : 20
- Velocity : 20

Second FB
MC_MoveAbsolute_1
- Position : 30
- Velocity : 50
- BufferMode : BlendingHigh

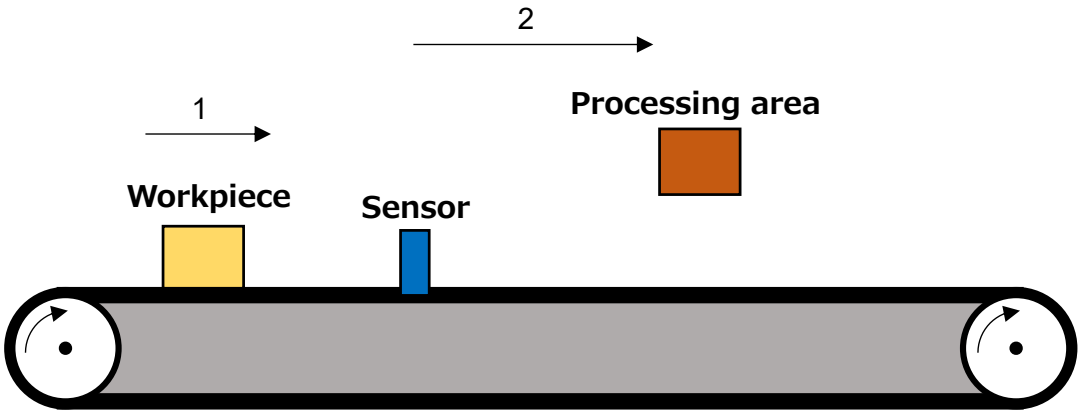


5.10.3 Usage Example of Buffer Mode

■ Operation example 1: Latch positioning operation (Aborting operation)

- Overview

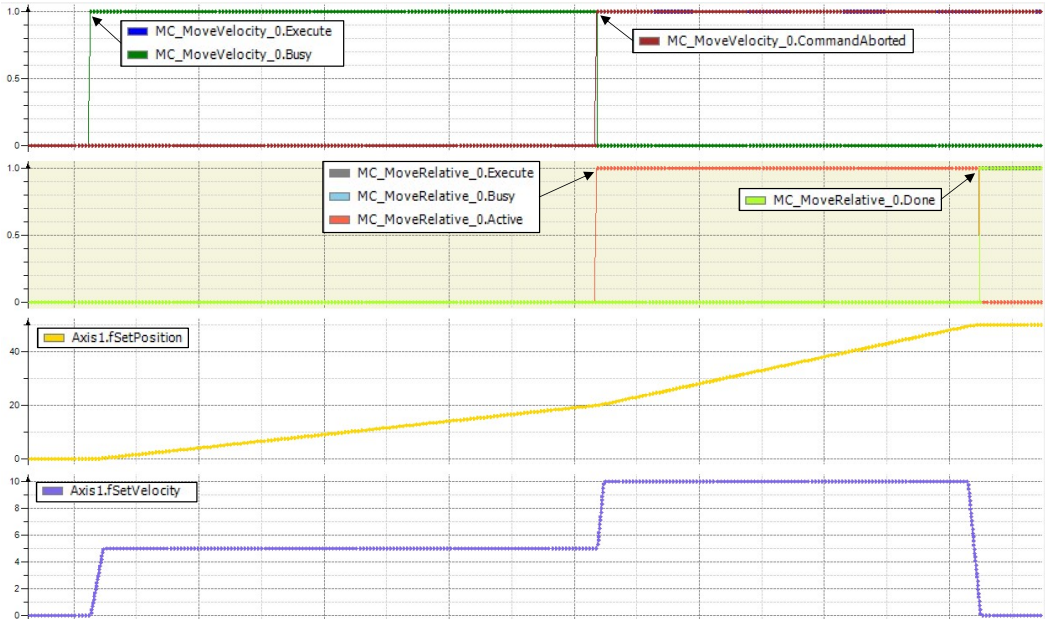
1. Using MC_MoveVelocity, operate the belt conveyor at a constant velocity to move the target workpiece.
2. When the target workpiece passes the sensor position, use MC_MoveRelative to perform the interrupt control to move the workpiece from the sensor position to the processing area.



- Implementation section (excerpt)

```
MC_MoveVelocity_0(
                    Axis:=Axis1,
                    Execute:=bExe_mv,
                    Velocity:=5,
                    Acceleration:=50,
                    Deceleration:=50
);

MC_MoveRelative_0(
                    Axis:=Axis1,
                    Execute:=bExe_mr,
                    Distance:=30,
                    Velocity:=10,
                    Acceleration:=100,
                    Deceleration:=100,
                    BufferMode:=MC_BUFFER_MODE.Aborting
);

bExe_mv:=TRUE;
IF bLatched = TRUE THEN // Workpiece passed sensor
    bExe_mr:=TRUE;
END_IF
```

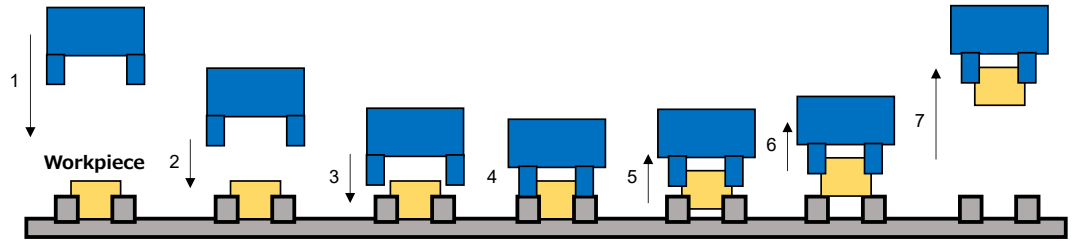
- Operation patterns


MC_MoveVelocity_0.Execute
MC_MoveVelocity_0.Busy
MC_MoveVelocity_0.CommandAborted
MC_MoveRelative_0.Execute
MC_MoveRelative_0.Busy
MC_MoveRelative_0.Active
MC_MoveRelative_0.Done
Axis1.fSetPosition
Axis1.fSetVelocity

■ Operation example 2: Pick and place operation (Blending operation using two FBs)

● Overview

1. Using the first MC_MoveAbsolute, lower the axis at high speed to the place before the pick position of the target workpiece.
2. Using BlendingNext, pass the axis through the end position of the first FB at the velocity of the second MC_MoveAbsolute.
3. Using the second MC_MoveAbsolute, lower the axis to the pick position at low speed.
4. Pick the target workpiece.
5. Using the first MC_MoveAbsolute, raise the axis at low speed until it passes through the groove where the target workpiece is to be placed.
6. Using BlendingLow, pass the axis through the end position of the first FB at the velocity (low speed) of the first MC_MoveAbsolute.
7. Using the second MC_MoveAbsolute, raise the axis at high speed.

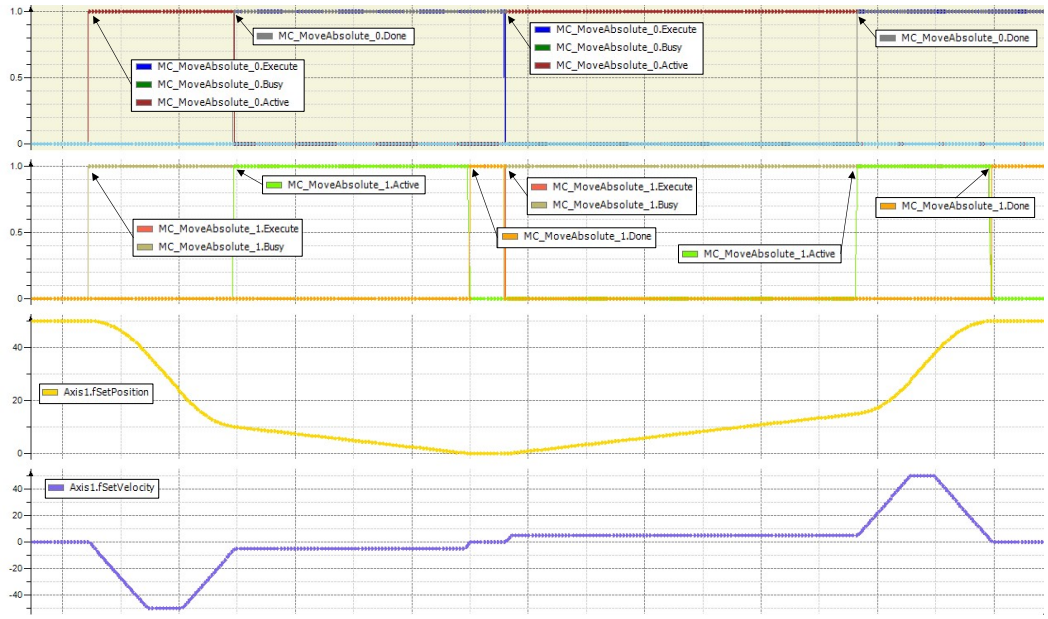


● Implementation section (excerpt)

```
MC_MoveAbsolute_0(
                    Axis:=Axis1,
                    Execute:=bExe_ma0,
                    Position:=Position0,
                    Velocity:=Velocity0,
                    Acceleration:=100,
                    Deceleration:=100,
                    BufferMode:=BufferMode0
);

MC_MoveAbsolute_1(
                    Axis:=Axis1,
                    Execute:=bExe_ma1,
                    Position:=Position1,
                    Velocity:=Velocity1,
                    Acceleration:=100,
                    Deceleration:=100,
                    BufferMode:=BufferMode1
);

CASE iStep OF
    0: // Set parameters during descent and execute FBs
        Position0:=10;
        Velocity0:=50;
        Position1:=0;
        Velocity1:=5;
        BufferMode1:=MC_BUFFER_MODE.BlendingNext;
```

```
        bExe_ma0:=TRUE;
        IF MC_MoveAbsolute_0.Active = TRUE THEN
            bExe_ma1:=TRUE;
            iStep:=1;
        END_IF

    1: // Pick up and reset FBs
        IF MC_MoveAbsolute_1.Done = TRUE THEN
            bChuckClose:=TRUE;    // For Pick up flag
            bExe_ma0:=FALSE;
            bExe_ma1:=FALSE;
        END_IF
        IF bCucked = TRUE THEN
            iStep:=2;
        END_IF

    2: // Set parameters when rising and execute FBs
        Position0:=15;
        Velocity0:=5;
        Position1:=50;
        Velocity1:=50;
        BufferMode1:=MC_BUFFER_MODE.BlendingLow;
        bExe_ma0:=TRUE;
        IF MC_MoveAbsolute_0.Active = TRUE THEN
            bExe_ma1:=TRUE;
        END_IF

END_CASE
```

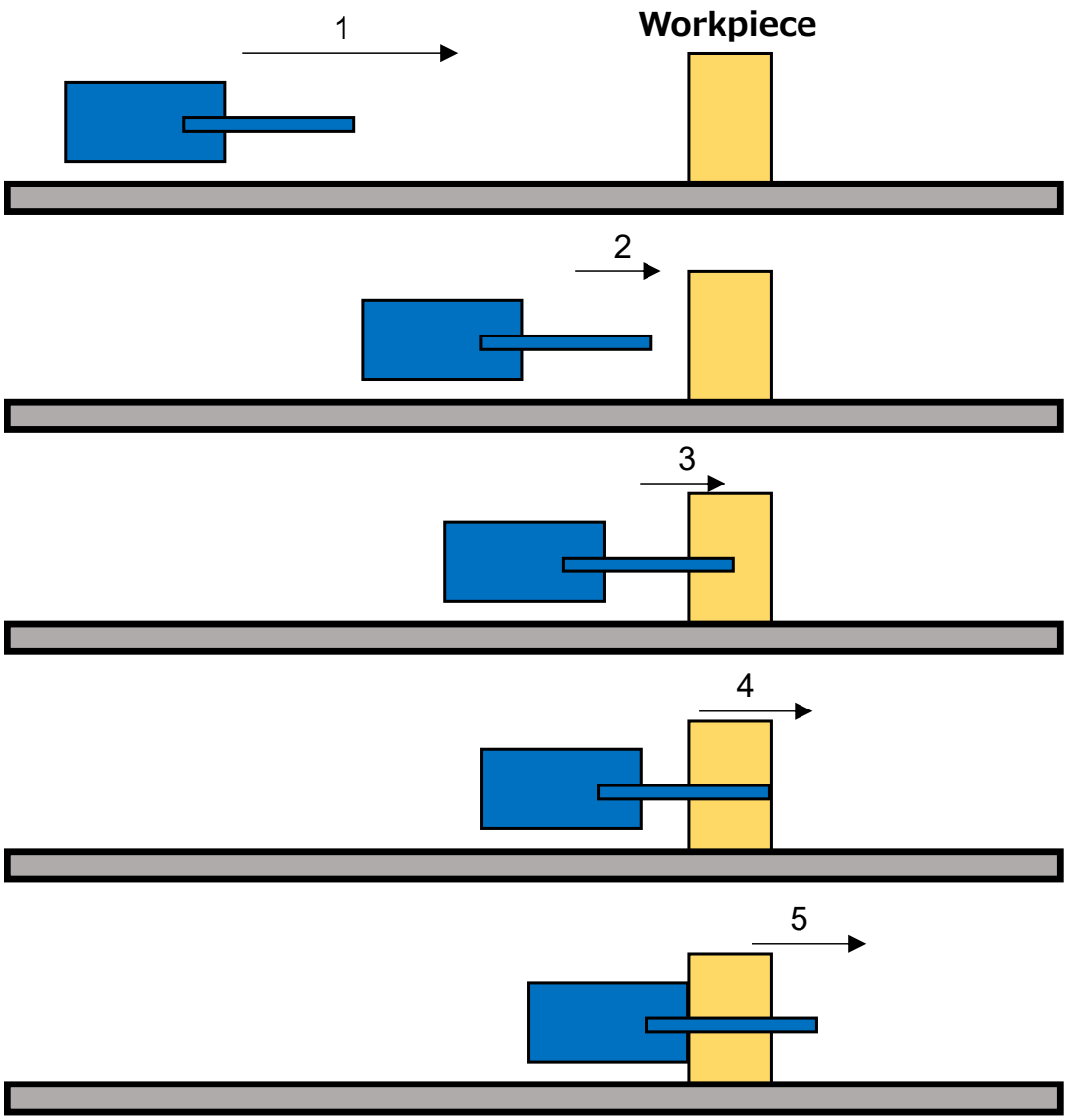
- Motion waveform



■ Operation example 3: Insertion operation (Blending operation using three FBs)

● Overview

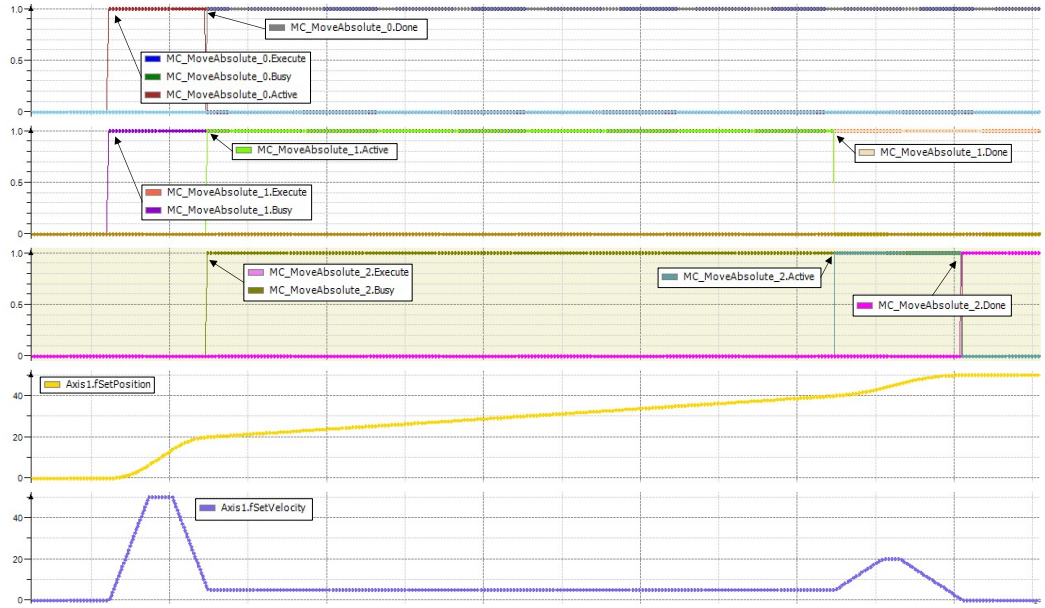
1. Using the first MC_MoveAbsolute, move the axis at high speed to the place before the insertion position of the target workpiece.
2. Using BlendingLow, pass the axis through the end position of the first FB at the velocity (low speed) of the second MC_MoveAbsolute.
3. Using the second MC_MoveAbsolute, move the axis at low speed to the position where the workpiece is inserted through.
4. Using BlendingLow, pass the axis through the end position of the second FB at the velocity (low speed) of the second MC_MoveAbsolute.
5. Using the third MC_MoveAbsolute, move the axis at medium speed to the insertion completion position.



- Implementation section (excerpt)

```
MC_MoveAbsolute_0(
                    Axis:=Axis1,
                    Execute:=bExe_ma0,
                    Position:=20,
                    Velocity:=50,
                    Acceleration:=100,
                    Deceleration:=100
);

MC_MoveAbsolute_1(
                    Axis:=Axis1,
                    Execute:=bExe_ma1,
                    Position:=40,
                    Velocity:=5,
                    Acceleration:=100,
                    Deceleration:=100,
                    BufferMode:=MC_BUFFER_MODE.BlendingLow
);

MC_MoveAbsolute_2(
                    Axis:=Axis1,
                    Execute:=bExe_ma2,
                    Position:=50,
                    Velocity:=20,
                    Acceleration:=50,
                    Deceleration:=50,
                    BufferMode:=MC_BUFFER_MODE.BlendingLow
);

bExe_ma0:=TRUE;
IF MC_MoveAbsolute_0.Active = TRUE THEN
    bExe_ma1:=TRUE;
ELSIF MC_MoveAbsolute_0.Done = TRUE AND MC_MoveAbsolute_1.Active = TRUE THE
N
    bExe_ma2:=TRUE;
END_IF
```

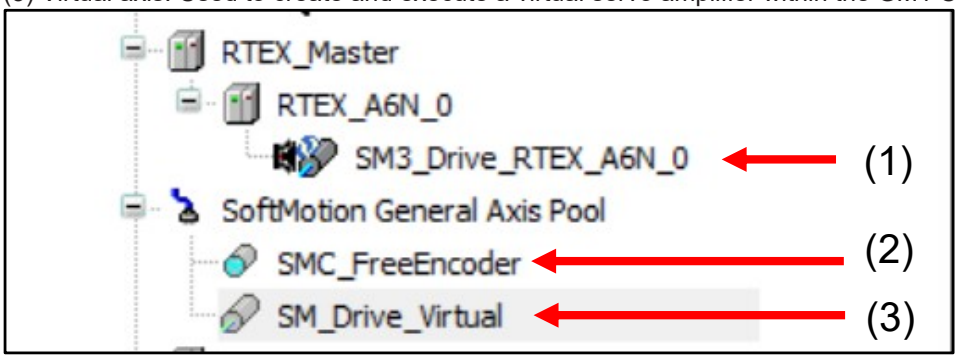
- Motion waveform


MC_MoveAbsolute_0.Done
MC_MoveAbsolute_0.Execute
MC_MoveAbsolute_0.Busy
MC_MoveAbsolute_0.Active
MC_MoveAbsolute_1.Active
MC_MoveAbsolute_1.Done
MC_MoveAbsolute_1.Execute
MC_MoveAbsolute_1.Busy
MC_MoveAbsolute_2.Execute
MC_MoveAbsolute_2.Busy
MC_MoveAbsolute_2.Active
MC_MoveAbsolute_2.Done
Axis1.fSetPosition
Axis1.fSetVelocity

5.11 Axis Structure

This is a function block that controls devices with real axis (AXIS_REF_SM3), virtual axis (AXIS_REF_VIRTUAL_SM3), or encoder axis (FREE_ENCODER_REF).

(1) Real axis: Used to actually control the servo amplifier.
(2) Encoder axis: Used when high precision control is required.
(3) Virtual axis: Used to create and execute a virtual servo amplifier within the GM1 Controller.



For the detailed setting procedure, refer to the "User's Manual (Operation Edition)".

■ Axis information

Member	Type	Default value	Description
nAxisState	SMC_AXIS_STATE	power_off	Axis (drive) state 0: power_off 1: errorstop 2: stopping 3: standstill 4: discrete_motion 5: continuous_motion 6: synchronized_motion 7: homing
nDirection	MC_DIRECTION	positive	Rotation direction of the axis (drive) 0:shortest: Travels to the rotation direction in the shortest distance (only for the modulo axis). 1:positive: Travels in the positive direction. 2:current: Maintains the current rotation direction (only for the modulo axis). 3:fastest: Travels to the fastest direction up to the target position (only for the modulo axis). -1:negative: Travels in the negative direction.
bRegulatorOn	BOOL	FALSE	Axis (drive) power ON/OFF: Possible to set using MC_Power(FB). TRUE:ON FALSE:OFF

Member	Type	Default value	Description
bDriveStart	BOOL	FALSE	Axis (drive) quick stop (software) control ON/OFF: Possible to set using MC_Power(FB). TRUE: Quick stop control control is OFF (possible to operate). FALSE: Quick stop control is ON (operation stopped)
byControllerMode	BYTE	3	Control mode of the axis (drive): Possible to set using SMC_SetControllerMode(FB). 1: SMC_torque: Torque control mode 2: SMC_velocity: Velocity control mode 3: SMC_position: Position control mode
bRegulatorRealState	BOOL	FALSE	Actual axis (drive) power state
bDriveStartRealState	BOOL	FALSE	State of the actual axis (drive) quick stop (software) control
byRealControllerMode	BYTE	3	State of the actual axis (drive) control mode
bRestarting	BOOL	FALSE	Re-initialization flag of the axis (drive) TRUE: Initialization in progress
usiSWEndSwitchState	USINT	0	Soft limit function state **(only for the finite axis)** 0: Soft limit invalid 2: Soft limit valid
strDriver	STRING(16)	''	Driver name is output.
bCommunication	BOOL	FALSE	Do not use.
wCommunicationState	WORD	16#FFFF	Do not use.
dwDriverVersion	DWORD	0	Do not use.

■ Position information

Member	Type	Default value	Description
fSetPosition	LREAL	0	Command position [u] on the program
fActPosition	LREAL	0	Actual position [u] on the program
fAimPosition	LREAL	0	Target position [u] on the program Target position set in a function block such as MC_MoveAbsolute(FB)
fScalefactor	LREAL	1	Factor used to calculate the command position (diSetPosition) of the actual axis (drive)
diSetPosition	DINT	0	Command position of the actual axis (drive)
diActPosition	DINT	0	Actual position of the actual axis (drive)
fSetActTimeLagCycles	LREAL	3	Time lag (number of cycles) between the fSetPosition value and the fActPosition value
fOffsetPosition	LREAL	0	Offset position

Member	Type	Default value	Description
			Stores the offset value (difference between the current position and changed position) when the command position is changed using MC_SetPosition(FB).

■ Velocity information

Member	Type	Default value	Description
fSetVelocity	LREAL	0	Command velocity [u/s] on the program
fActVelocity	LREAL	0	Actual velocity [u/s] on the program
fFactorVel	LREAL	1	Factor used to calculate command velocity (diSetVelocity) of the actual axis (drive)
diSetVelocity	DINT	0	Command velocity [u/s] of the actual axis (drive)
diActVelocity	DINT	0	Actual velocity [u/s] of the actual axis (drive)(Note 1)
bConstantVelocity	BOOL	FALSE	TRUE: The axis is moving at a constant velocity or is stopped.

(Note 1) Values cannot be obtained using the RTEX.

■ Acceleration / deceleration / jerk

Member	Type	Default value	Description
fSetAcceleration	LREAL	0	Command acceleration [u/s^2] on the program
fActAcceleration	LREAL	0	Actual acceleration [u/s^2] on the program
fFactorAcc	LREAL	1	Factor used to calculate command acceleration of the actual axis (drive)
bAccelerating	BOOL	FALSE	TRUE: The axis is moving in acceleration.
bDecelerating	BOOL	FALSE	TRUE: The axis is moving in deceleration.
fSetJerk	LREAL	0	Command jerk [u/s^3] on the program
fFactorJerk	LREAL	1	Factor used to calculate command jerk of the actual axis (drive)
fActJerk	LREAL	0	Do not use.
diSetAcceleration	DINT	0	Do not use.
diActAcceleration	DINT	0	Do not use.

■ Torque information

Member	Type	Default value	Description
fSetTorque	LREAL	0	Command torque [Nm] on the program

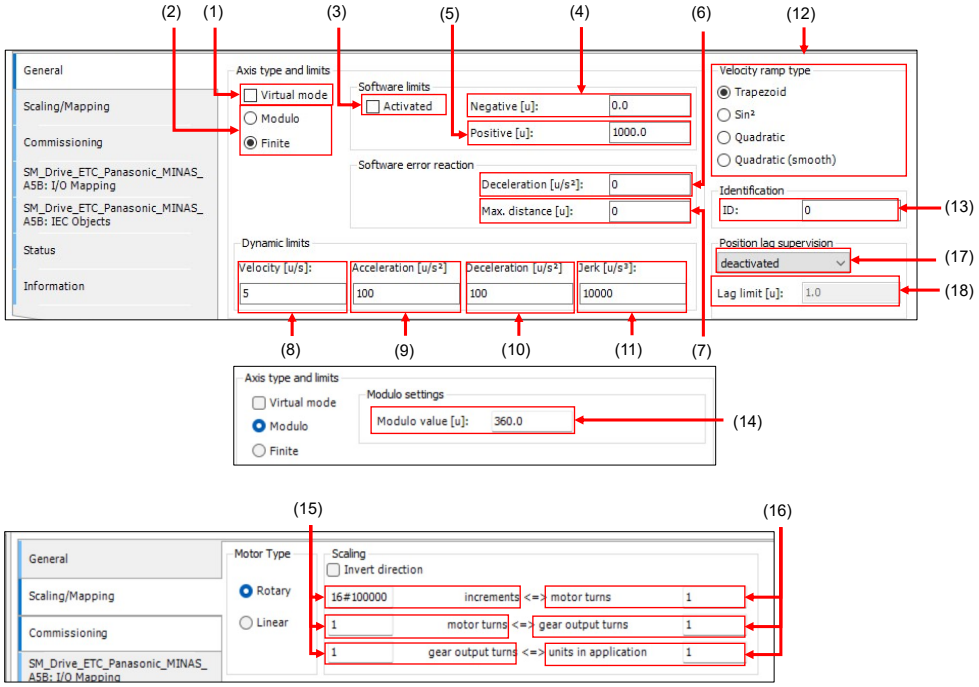
Member	Type	Default value	Description
fActTorque	LREAL	0	Actual torque [Nm] on the program
fFactorTor	LREAL	0	Factor used to calculate the command torque (diSetTorque) of the actual axis (drive)
diSetTorque	DINT	0	Command torque [Nm] of the actual axis (drive)
diActTorque	DINT	0	Actual torque [Nm] of the actual axis (drive)

■ Tab setting locations

Member	Type	Default value	Description
bVirtual	BOOL	FALSE	TRUE: Virtual mode/Simulation mode ((1) in figure below)
iMovementType	INT	1	Axis type: Possible to set using SMC_ChangeGearingRatio(FB) ((2) in figure below) 0: Modulo 1: Finite
bSWLimitEnable	BOOL	FALSE	Function (position limit) setting under “Soft limit” ((3) in figure below): Possible to set using SMC_SetSoftwareLimits(FB). FALSE: Disable TRUE: Enable
fSWLimitNegative	LREAL	0	Position limit value [u] to the negative direction under “Soft limit” ((4) in figure below): Possible to set using SMC_SetSoftwareLimits(FB).
fSWLimitPositive	LREAL	0	Position limit value [u] to the positive direction under “Soft limit” ((5) in figure below): Possible to set using SMC_SetSoftwareLimits(FB).
fSWLimitDeceleration	LREAL	0	Deceleration [u/s^2] under “Software error reaction” ((6) in figure below): Possible to set using SMC_SetSoftwareLimits(FB).
fSWErrorMaxDistance	LREAL	0	Maximum distance [u] under “Software error reaction” ((7) in figure below): Possible to set using SMC_SetSoftwareLimits(FB).
fSWMaxVelocity	LREAL	0	Maximum velocity on the program [u/s] under “Dynamic limit” ((8) in figure below): Possible to set using SMC_ChangeDynamicLimits(FB).
fSWMaxAcceleration	LREAL	0	Maximum acceleration on the program [u/s^2] under “Dynamic limit” ((9) in figure below): Possible to set using SMC_ChangeDynamicLimits(FB).
fSWMaxDeceleration	LREAL	0	Maximum deceleration on the program [u/s^2] under “Dynamic limit” ((10) in

Member	Type	Default value	Description
			figure below): Possible to set using SMC_ChangeDynamicLimits(FB).
fSWMaxJerk	LREAL	0	Maximum jerk on the program [u/s^3] under “Dynamic limit” ((11) in figure below): Possible to set using SMC_ChangeDynamicLimits(FB).
eRampType	SMC_RAMPTYPE	trapez	Function setting under “Velocity ramp type” ((12) in figure below): Possible to set using SMC_SetRampType(FB). 0: trapez 1: sinsquare 2: quadratic_ramp 3: quadratic_smooth_ramp
wDriveId	WORD	0	ID number of the drive ((13) in figure below)
fPositionPeriod	LREAL	1000	Modulo setting value ((14) in figure below): Possible to set using SMC_ChangeGearingRatio(FB)
dwRatioTechUnitsDenom	DWORD	1	Gear ratio denominator under internal variables ((15) in figure below): Possible to set using SMC_ChangeGearingRatio(FB) increments × motor turns × gear output turns
iRatioTechUnitsNum	DINT	1	Gear ratio numerator under internal variables ((16) in figure below): Possible to set using SMC_ChangeGearingRatio(FB) motor turns × gear output turns × units in application
eCheckPositionLag	SMC3_Check PositionLagMode	0	Position lag monitoring setting ((17) in figure below) 0: SMC3_PCL_OFF (deactivating) 1: SMC3_PCL_DESABLE (disabling the drive) 2: SMC3_PCL_HALT (using quick stop) 3: SMC3_PCL_ENABLE (leave drive enabled)
fMaxPositionLag	LREAL	0	Maximum allowance [u] of position lag ((18) in figure below)

■ Setting locations



■ Error information

Member	Type	Default value	Description
bError	BOOL	FALSE	Presence/absence of error
dwErrorID	DWORD	0	Error ID unique to the drive
fbeFBError[0..5]	ARRAY OF SMC_FBERROR	SMC_NO_ERROR	FB error: SMC_FBERROR(STRUCT) wID(SMC_ERROR): Error ID pbyErrorInstance(POINTER TO BYTE): Pointer to the FB instance that detected an error strErrorInstance(STRING): FB instance that detected an error tTimestamp(TIME): Elapsed time from when the GM1 power is turned on till when an error has occurred
uiDriveInterfaceError	UINT	0	Do not use.
strDriveInterfaceError	STRING	''	Do not use.
bOldError	BOOL	FALSE	Do not use.
bErrorAckn	BOOL	FALSE	Do not use.
bDisableErrorLogging	BOOL	FALSE	Do not use.
diFollowingError	DINT	0	Do not use.
fFollowingError	LREAL	0	Do not use.

■ Others

Member	Type	Default value	Description
dwOneTurn	DWORD	0	Number of pulses per one modulo period
bPositionLagActive	BOOL	FALSE	TRUE: Position lag exceeds the maximum allowance.
iTurn	INT	0	Do not use.
bAvoidReversalOnHaltStop	BOOL	FALSE	Do not use.
bConsiderLimitsOfAbortedMotionOnHaltStop	BOOL	FALSE	Do not use.
bStartReference	BOOL	FALSE	Do not use.
fReference	LREAL	0	Do not use.
bStartReferenceRealState	BOOL	FALSE	Do not use.
xWaitForHaltWhenStopInterruptsHome	BOOL	FALSE	Do not use.
iOwner	INT	0	Do not use.
iNoOwner	INT	0	Do not use.
fCycleTimeSpent	LREAL	0	Do not use.
fTaskCycle	LREAL	0.005	Do not use.
bHWLimitEnable	BOOL	TRUE	Do not use.
nAbortCounter	UDINT	0	Do not use.
iLastSinSquareOwner	INT	-3	Do not use.
bSetValuesModifiedByMoveSuperimposed	BOOL	FALSE	Do not use.
eBrakeControl	SMC3_BrakeSetState	SMC_BRAKE_AUTO	Do not use.
bBrakeClosedRealState	BOOL	FALSE	Do not use.
xPersistentDataLoaded	BOOL	FALSE	Do not use.
wAxisStructID	WORD	16#FE12	Do not use.
fFactorCur	LREAL	0	Do not use.
fMarkPosition	LREAL	0	Do not use.
fSavePosition	LREAL	0	Do not use.
fMaxVelocity	LREAL	0	Do not use.
fMarkVelocity	LREAL	0	Do not use.
fSaveVelocity	LREAL	0	Do not use.
fMaxAcceleration	LREAL	0	Do not use.
fMarkAcceleration	LREAL	0	Do not use.
fSaveAcceleration	LREAL	0	Do not use.
fMaxDeceleration	LREAL	0	Do not use.
fSaveDeceleration	LREAL	0	Do not use.

Member	Type	Default value	Description
fMaxJerk	LREAL	0	Do not use.
fRampJerk	LREAL	100000	Do not use.
fMarkJerk	LREAL	0	Do not use.
fSaveJerk	LREAL	0	Do not use.
fSetCurrent	LREAL	0	Do not use.
fActCurrent	LREAL	0	Do not use.
fMaxCurrent	LREAL	100	Do not use.
diSetCurrent	DINT	0	Do not use.
diActCurrent	DINT	0	Do not use.
fSWMaxCurrent	LREAL	0	Do not use.
fMaxTorque	LREAL	0	Do not use.
dwPosOffsetForResiduals	DWORD	0	Do not use.
dwLastPosition	DWORD	0	Do not use.
aCaptDesc	ARRAY OF SMC3_CaptureDescription		Do not use.
adatAcyclic	ARRAY OF SMC3_DriveAcyclicTel		Do not use.
bySwitchingState	SMC_SWITCHING_STATE	SMC_ST_INITIALIZING	Do not use.
iRestNumerator	DINT	0	Do not use.
dwPosOffsetForResidualsHoming	DWORD	0	Do not use.
dwActPosition	DWORD	0	Do not use.
dwBusBandWidth	DWORD	0	Do not use.
dwBusModuloMask	DWORD	0	Do not use.
bModuloDoneByDrive	BOOL	FALSE	Do not use.
bLogical	BOOL	FALSE	Do not use.
bUpdateIOsInStop	BOOL	FALSE	Do not use.
vMinRequiredVersion	VERSION	0	Do not use.
iRampType1	SMC_TG_IRAMPTYPE		Do not use.
iRampType2	SMC_TG_IRAMPTYPE		Do not use.

■ FREE_ENCODER_REF only

Member	Type	Default value	Description
diEncoderPosition	DINT		Do not use.

(MEMO)

6 Motion Control Function Blocks (Synchronous Control)

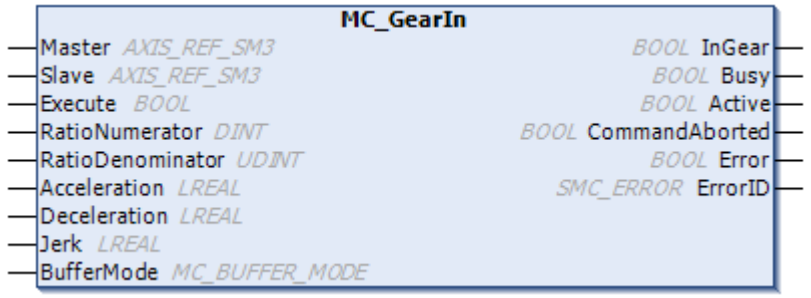
This section describes motion control function blocks to perform synchronous processing.

6.1 Gear Operation

6.1.1 MC_GearIn (Start Gear Operation)

This is a Function Block (FB) to initiate gear synchronous operation.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Master	AXIS_REF_SM3	-	Specifies the master axis.
	Slave	AXIS_REF_SM3	-	Specifies the slave axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	RatioNumerator	DINT	1	Specifies the gear ratio (numerator).
	RatioDenominator	UDINT	1	Specifies the gear ratio (denominator).
	Acceleration	LREAL	0	Maximum acceleration (u/s^2) until gear synchronization is completed
	Deceleration	LREAL	0	Maximum deceleration (u/s^2) until gear synchronization is completed
	Jerk	LREAL	0	Maximum jerk (u/s^3) until gear synchronization is completed
	BufferMode	MC_BUFFER_MODE	Aborting	Specifies a buffer mode. The value is valid when this FB is a second FB.
Output	InGear	BOOL	FALSE	TRUE: Gear synchronization is completed.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	Active	BOOL	FALSE	TRUE: The second FB is being controlled.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred.

Scope	Name	Type	Initial	Description
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ MC_BUFFER_MODE (Enumeration type)

On condition that this FB is connected as the second FB, the table below gives a description.

Name	Value	Description
Aborting	0	The operation of the first FB stops, and this FB starts operation instantly.
Buffered	1	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The movement starts at the velocity that the preceding movement has when the end condition is reached.
BlendingLow	2	Not available. Do not specify this.
BlendingPrevious	3	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The axis passes through the end position of the first FB operation at the velocity of the first FB command.
BlendingNext	4	Not available. Do not specify this.
BlendingHigh	5	Not available. Do not specify this.

(Note 1) Refer to Table "Buffer Mode Operation Conditions."

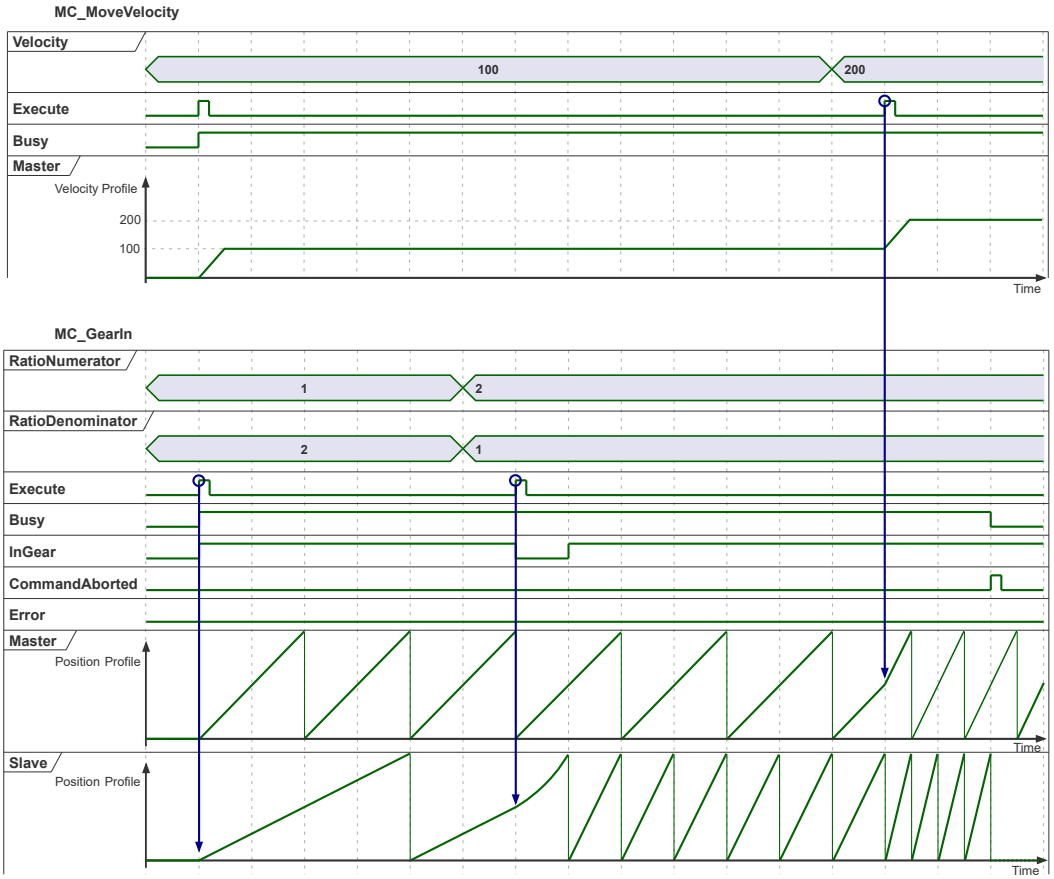
■ Detail of function

- Description of functions
  - The slave axis will have a speed that is the product of the master axis speed and the gear ratio.

    The speed of the slave axis = the speed of the master axis * (RatioNumerator / RatioDenominator)
  - The slave axis will accelerate or decelerate based on the gear ratio towards gear synchronization. Once the slave axis reaches the speed of the specified gear ratio, gear synchronization is complete.
  - After the gear synchronization is complete, if the speed of the master axis changes, the slave axis will maintain the gear ratio relationship and follow the speed.
  - During gear synchronization operation, this FB (function block) must be called at each control cycle.
- Gear Synchronization Start
  - Upon the rising edge of Execute, following the Acceleration, Deceleration, and Jerk settings, the slave axis will begin gear synchronization.
- Gear Synchronization Release
  - This FB cannot be used to release gear synchronization; use MC_GearOut to release it.
- Re-execution
  - Set Execute to FALSE. Then, reconfigure the input values. By raising Execute, you run it with the new input values.
- Interruption of operation
  - If you call an FB to control an axis for the slave axis during operation, the operation of this FB will be interrupted.

■ Timing chart

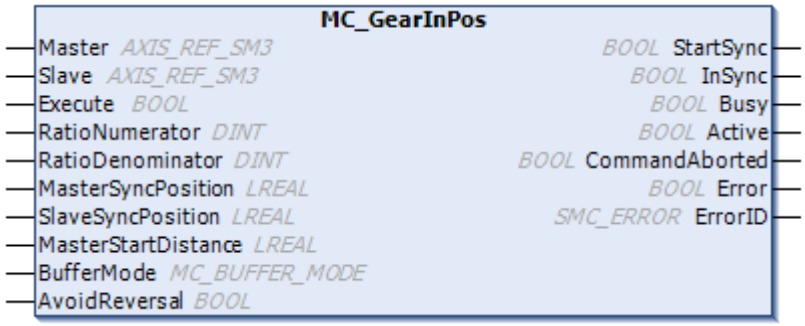
- At the launch of the Execute, the Busy goes TRUE.
- When the slave axis reaches the speed of the specified gear ratio, InGear becomes TRUE, indicating gear synchronization is complete.
- When another FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE. The behavior of this FB after the CommandAborted is TRUE depends on the behavior of other FBs.



6.1.2 MC_GearInPos (Position Specified Gear Operation)

This is a function block (FB) that starts synchronous operation of the gears from the specified absolute position.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Master	AXIS_REF_SM3	-	Specifies the master axis.
	Slave	AXIS_REF_SM3	-	Specifies the slave axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	RatioNumerator	DINT	1	Specifies the gear ratio (numerator).
	RatioDenominator	DINT	1	Specifies the gear ratio (denominator).
	MasterSyncPosition	LREAL	0	Position of the master axis to start gear synchronization
	SlaveSyncPosition	LREAL	0	Position of the slave axis to start gear synchronization
	MasterStartDistance	LREAL	0	The travel distance from the start of the movement towards synchronization of the slave axis until synchronization is complete. The starting position for the slave axis synchronization is specified by MasterSyncPosition - MasterStartDistance, based on the position of the master axis. If MasterStartDistance is 0, the slave axis will immediately start moving towards synchronizatio
	BufferMode	MC_BUFFER_MODE	Aborting	Specifies a buffer mode. The value is valid when this FB is a second FB.
	AvoidReversal	BOOL	FALSE	● Axis setting: When set to "Finite" AvoidReversal = TRUE

Scope	Name	Type	Initial	Description
				● Axis setting: When set to "Modulo" AvoidReversal = FALSE
Output	StartSync	BOOL	FALSE	TRUE: Start gear synchronization
	InSync	BOOL	FALSE	TRUE: Gear synchronization is completed.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	Active	BOOL	FALSE	TRUE: The second FB is being controlled.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ MC_BUFFER_MODE (Enumeration type)

On condition that this FB is connected as the second FB, the table below gives a description. If Buffered or BlendingPrevious is specified, set MasterStartDistance to 0.

Name	Value	Description
Aborting	0	The operation of the first FB stops, and this FB starts operation instantly.
Buffered	1	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The movement starts at the velocity that the preceding movement has when the end condition is reached.
BlendingLow	2	Not available. Do not specify this.
BlendingPrevious	3	When the first FB operation satisfies the end condition(Note 1), this FB starts operation instantly. The axis passes through the end position of the first FB operation at the velocity of the first FB command.
BlendingNext	4	Not available. Do not specify this.
BlendingHigh	5	Not available. Do not specify this.

(Note 1) Refer to Table "Buffer Mode Operation Conditions."

■ Detail of function

● Description of functions

- The slave axis will have a speed that is the product of the master axis speed and the gear ratio.

  The speed of the slave axis = speed of the master axis * (RatioNumerator / RatioDenominator)
- The position of the master axis to start gear synchronization is specified by MasterSyncPosition, and the position of the slave axis is specified by SlaveSyncPosition.
- The position at which the slave axis starts moving is specified by MasterSyncPosition - MasterStartDistance, based on the position of the master axis. The slave axis will stop

until it is able to start moving. However, if MasterStartDistance is set to 0, the slave axis will start moving at the rising edge of Execute.

- The slave axis begins its operation towards gear synchronization based on MasterSyncPosition and MasterStartDistance, and gear synchronization is completed when the position of the master axis reaches MasterSyncPositiont and the position of the slave axis reaches SlaveSyncPosition.
- The speed, acceleration, and deceleration of the slave axis from the start to the completion of gear synchronization are determined automatically.
- After the completion of gear synchronization, if the speed of the master axis changes, the slave axis will adjust its speed while maintaining the gear ratio relationship.
- During gear synchronization operation, this FB (function block) must be called at each control cycle.

- Gear Synchronization Start
  - At the rising edge of Execute, the slave axis starts synchronization based on MasterSyncPosition, SlaveSyncPosition, and MasterStartDistance.
- Gear Synchronization Release
  - This FB cannot be used to release gear synchronization; use MC_GearOut to release it.
- Re-execution
  - Set Execute to FALSE. Then, reconfigure the input values. By raising Execute, you run it with the new input values.
- Interruption of operation
  - If you call an FB to control an axis for the slave axis during operation, the operation of this FB will be interrupted.

■ Operations when the function block is executed

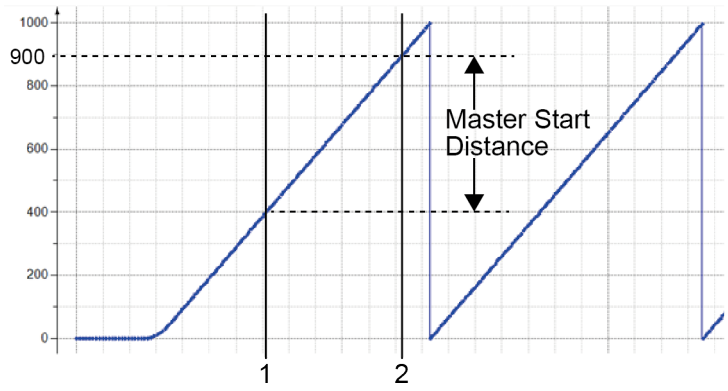
This example shows the trace when the MC_GearInPos function block is executed with the following conditions.

Execution condition

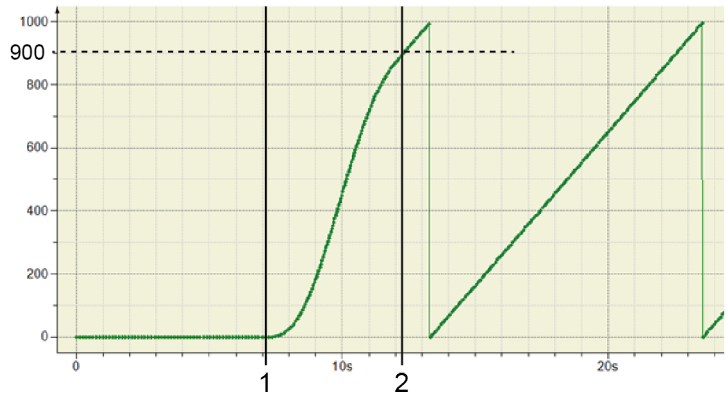
Item	Dis
Master axis type	Modulo (modulo value = 1000)
Slave axis type	Modulo (modulo value = 1000)
Gear ratio	1 : 1
Input MasterSyncPostion	900
Input SlaveSyncPositon	900
Input MasterStartDistance	500

When the master axis position reaches 900 and the salve axis position reaches 900, the master axis starts to synchronize with the slave axis. When the master axis passes the position 400, which is obtained by deducting 500 (MasterStartDistane) from 900 (synchronization start position of the master axis), the slave axis starts traveling to synchronize with the master axis. At this time, velocity, acceleration, and deceleration are automatically determined.

Position of the master axis



Position of the slave axis



■ AvoidReversal

By setting AvoidReversal, the slave axis can be restricted on reverse rotation. If AvoidReversal is set to TRUE, an error occurs under the following conditions.

1. Gear ratio is negative.

   If the gear ratio is negative (for example, RatioNumerator = -1, RatioDenominator = 1), when the axis reaches the position set in GearInPos.StartSync while the slave axis is operating in forward rotation, an error (SMC_GIP_SLAVE_REVERSAL_CANNOT_BE_AVOIDED) occurs
2. The slave axis is rotating in reverse to the rotation of the master axis before the start of synchronization

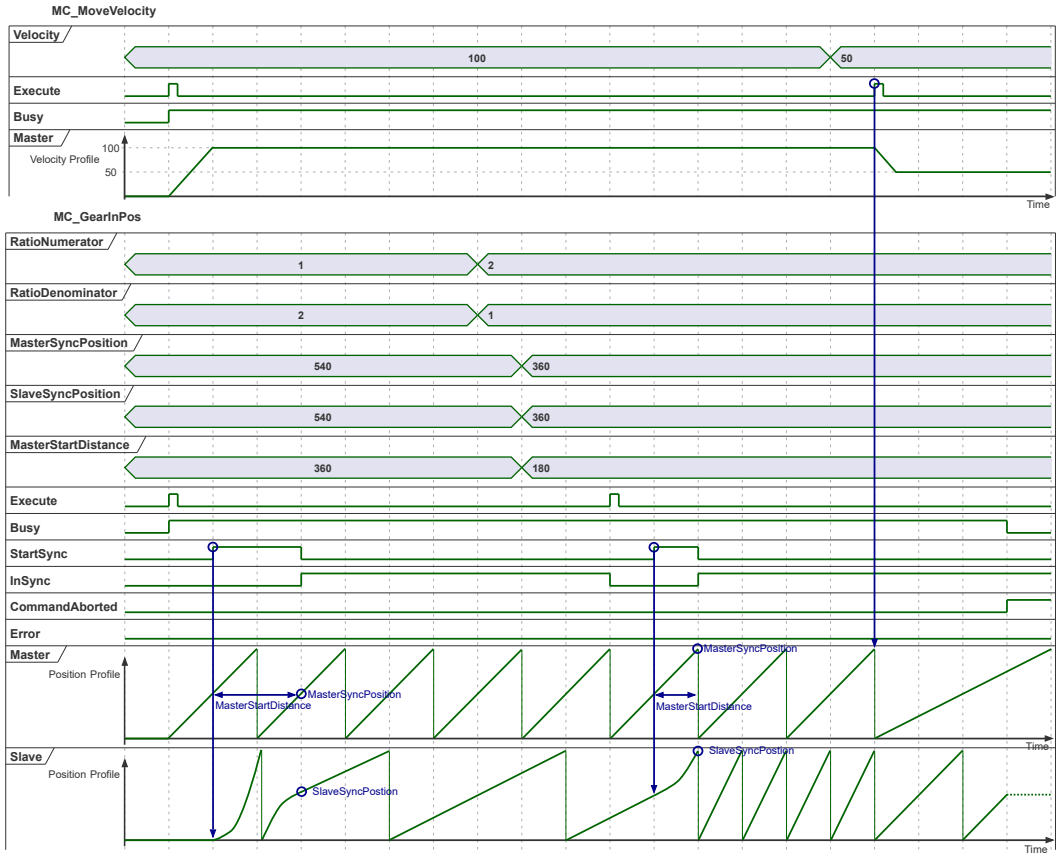
   When the axis reaches the gear synchronization start position set in StartSync while the slave axis is operating in reverse rotation, an error (SMC_GIP_SLAVE_REVERSAL_CANNOT_BE_AVOIDED) occurs
3. Correction of the slave axis is not completed within five cycles.

   Gear synchronization completion (InSync) is not achieved within five cycles after reaching the gear synchronization start (StartSync), an error occurs.

■ Timing chart

- At the launch of the Execute, the Busy goes TRUE.

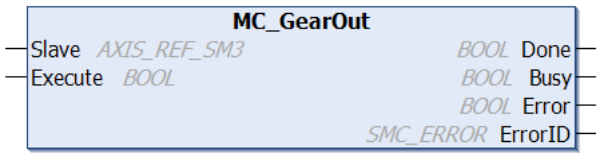
- If the slave axis has started moving towards synchronization, StartSync becomes TRUE.
- When the position of the master axis reaches MasterSyncPositiont, and the position of the slave axis reaches SlaveSyncPosition, StartSync becomes FALSE, InSync becomes TRUE, and gear synchronization is completed.
- When another FB that controls the same axis is called while the Busy is TRUE, the CommandAborted goes TRUE. The behavior of this FB after the CommandAborted is TRUE depends on the behavior of other FBs.



6.1.3 MC_GearOut (Cancel Gear Operation)

This is a function block (FB) to release gear synchronization. After gear synchronization is released, the state of the slave axis will become continuous motion. If you want to return the state of the axis to standstill, please execute MC_Halt or MC_Stop.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Slave	AXIS_REF_SM3	-	Specifies the slave axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
Output	Done	BOOL	FALSE	TRUE: Synchronization cancellation is completed.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	Error	BOOL	FALSE	TRUE: An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

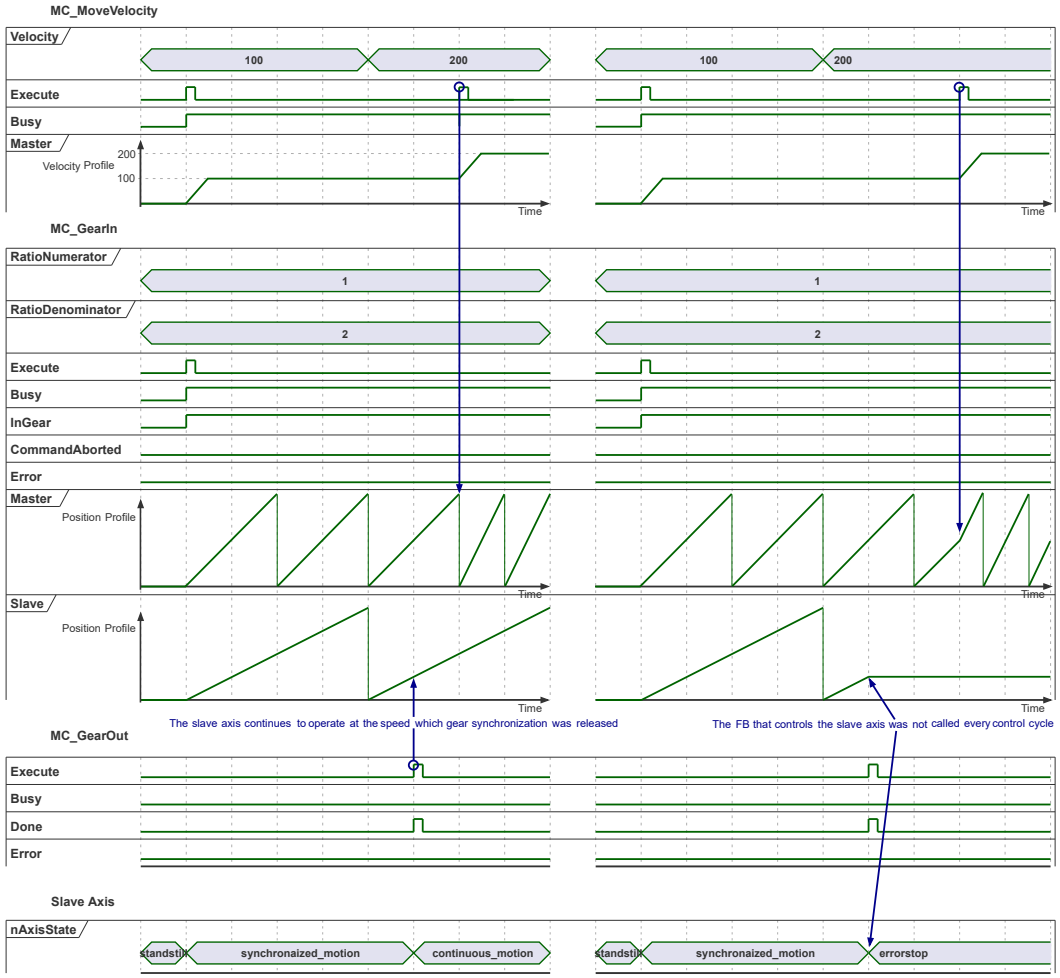
■ Detail of function

- Function Description
  - Releases the gear synchronization of the slave axis with respect to the master axis.
  - After the gear synchronization is released, the state of the slave axis becomes Continuous Motion (continuous_motion).

    Since the axis is in operation during Continuous Motion, it is necessary to call this FB at every control cycle even after the gear synchronization is complete.
  - Since the state of the slave axis continues in Continuous Motion after the gear synchronization is released, the slave axis continues to control speed at the velocity command value that was set at the time of gear synchronization release.
  - If you wish to stop the slave axis after the gear synchronization is released, please execute MC_Halt or MC_Stop.
- Operation Start
  - The gear synchronization of the slave axis with respect to the master axis is released at the rising edge of Execute.

■ Timing chart

- After the gear synchronization is released, Done becomes TRUE. If Execute is set to FALSE, the Done state will become FALSE.



6.1.4 Example: Gear Synchronization

Here's an example of a program for gear synchronization control with a gear ratio of 2:1. To start synchronization, please use "MC_GearIn" or "MC_GearInpos".

To end gear synchronization control, use "MC_GearOut".

■ Program example

- Implementation Section

```
CASE Process OF
    0: // Servo ON
        MC_Power_0(
            Axis := Master ,
            Enable := TRUE ,
            bRegulatorOn := TRUE,
```

```
            bDriveStart := TRUE
        );
        MC_Power_1(
            Axis := Slave ,
            Enable := TRUE ,
            bRegulatorOn:= TRUE,
            bDriveStart:= TRUE
        );
        IF MC_POWER_0.Status = TRUE AND MC_Power_1.Status = TRUE THEN
            Process := 1;
        END_IF
    1: // Controll the master axis with MC_MOveVelocity and Control the ala
ve axis aynchhronously
        MC_GearIn_0(
            Master := Master,
            Slave  := Slave,
            Execute := TRUE,
            RatioNumerator := 2,
            RatioDenominator :=1,
            Acceleration := 3600,
            Deceleration := 3600
            );
        MC_MoveVelocity_0(
            Axis := Master,
            Execute := TRUE,
            Velocity := 360,
            Acceleration := 3600,
            Deceleration := 3600,
            Direction := positive
            );
        IF MC_GearIn_0.InGear = TRUE THEN
            // Gear In OK
            MC_GearIn_0(
                Master := Master,
                Slave  := Slave,
                Execute := FALSE
            );
            MC_MoveVelocity_0(
                Axis := Master,
                Execute := FALSE
            );
        END_IF
        IF  Master.fActposition > 100 THEN
            MC_Halt_0(
            Axis := Master,
            Execute := TRUE,
            Deceleration := 1800
            );
            IF MC_Halt_0.Done THEN
                Process := 2;
            END_IF
        END_IF
    2: // Call MC GearOut to end aynchronization
        MC_GearOut_0(
            Slave  := Slave,
            Execute := TRUE
```

```
        );
        IF MC_GearOut_0.Done THEN
            // Gear Out OK
            MC_Halt_1(
                Axis := Slave,
                Execute := TRUE,
                Deceleration := 1800
            );
        END_IF
END_CASE
```

By using "MC_GearOut", synchronization is released, and the slave axis will continue to operate maintaining its speed at the time of synchronization release. Therefore, it is important to note that the slave axis needs to be stopped using something like "MC_Halt".

You can safely stop by disengaging the gear synchronization after stopping the master axis.

6.2 Cam Synchronous Control

Cam Synchronous Control is control executed to get a target axis that constitutes the slave axis to be synchronized in response to the motion of the master axis in accordance with a cam profile (time-series waveform information).

6.2.1 Overview of Cam Synchronous Control

Through use of function blocks, you can perform cam synchronous control of the axes using a cam table created by a tool or a cam profile created by POU.

The following function blocks related to cam synchronous control can be used.

- MC_CAM_REF: A cam profile used for cam control can be created by POU.
- MC_CamTableSelect: This is used to set a cam profile used for cam control, and the master axis and slave axis.
- MC_CamIn: The slave axis gets synchronized with the master axis through the cam profile.
- MC_CamOut: This cancels cam synchronization of the slave axis.
- SMC_GetTappetValue / SMC_CamRegister: This gets switch information (tappet information) that causes ON/OFF in response to positional information.
- SMC_CAMBounds: This is used to get the minimum/maximum velocity and acceleration values of the slave under cam synchronous control in advance.
- SMC_GetCamSlaveSetPosition: This calculates starting position, velocity, and acceleration values of the slave relative to the position of the master.

A procedure for performing cam synchronous control using a cam profile (MC_CAM_REF) created by the Cam editor on the GM Programmer is described.

1 2 Procedure

1. Create a cam by the Cam editor

   With the Cam editor, specify a cam table and a tappet table used for synchronous operation.

2. While cam synchronous control is in progress, calculate whether the slave motion is in an allowable range in advance.

   Execute SMC_CAMBounds to calculate the minimum/maximum velocity and acceleration values that the slave is reaching during motion.

3. Set a cam profile (MC_CAM_REF) used to execute cam control, and the master axis and slave axis.

   Execute MC_CamTableSelect to set a cam profile (MC_CAM_REF) used to execute cam control, and the master axis and slave axis.

4. Before executing cam synchronization, calculate data about the starting position of the slave in advance.

   Execute SMC_GetCamSlaveSetPosition to calculate position and velocity values of the slave at the start of cam synchronous operation relative to the current position of the master.

5. Synchronize the master axis and the slave axis.

Execute MC_CamIn so that the slave axis gets synchronized with the master axis through the cam profile. In accordance with the master position and the cam profile, the slave axis is controlled to get synchronized with the master axis.

6. Configure setting to read tappet information.
Get tappet information by using SMC_GetTappetValue or SMC_CamRegister.

7. Cancel synchronization between the master axis and slave axis.
After cam synchronization is completed, execute MC_CamOut to cancel synchronization between the master axis and slave axis.

8. After canceling synchronization, stop the slave axis motion.
Since the slave axis continues to operate at velocity after the cancellation of synchronization, execute MC_Halt or MC_Stop to stop the slave motion.

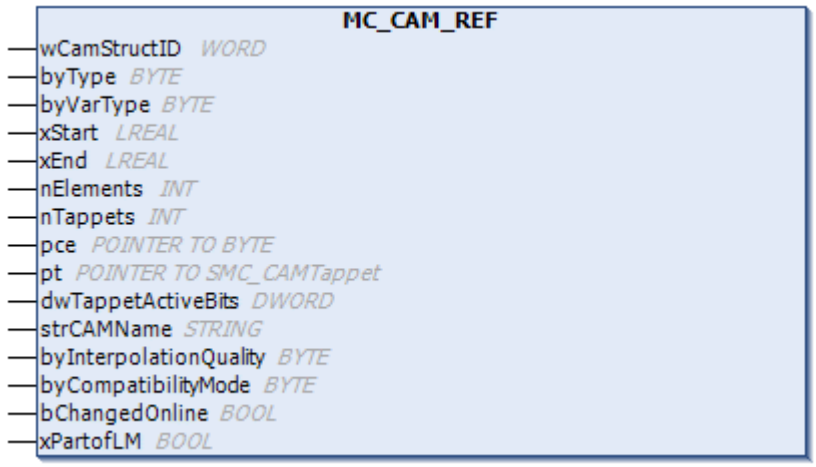
Info.

- For the procedure for creating MC_CAM_REF using the Cam editor, refer to the GM1 Series Reference Manual (Operation Edition).
- For an example of the procedure for creating MC_CAM_REF through a program, refer to"6.2.12 Sample Example: Create MC_CAM_REF by POU".

6.2.2 MC_CAM_REF (Cam Profile)

This is a function block (FB) that specifies a cam table and a tappet table used for cam control and creates a cam profile (MC_CAM_REF). MC_CAM_REF created by POU can be used in MC_CamTableSelect and other FBs in the similar way as ones created by the Cam editor.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	wCamStructID	WORD	16#DC34	Fixed to 16#DC34
	byType	BYTE	0	A parameter that sets the cam type(Note 1)(Details will be described later) ● 1: One-dimensional table of slave positions ● 2: Two-dimensional table of master and slave positions ● 3: Polynomial array table of points consisting of master position, slave position, slave velocity and slave acceleration (XYVA)
	byVarType	BYTE	0	A parameter that defines the type of variables entered in the cam table(Note 1) Only used if byType = 1 or 2 0: used if byType = 3 1: INT 2: UINT 3: DINT 4: UDINT 5: REAL 6: LREAL
	xStart	LREAL	0	Start position of the master axis on the cam table(Note 1)
	xEnd	LREAL	0	End position of the master axis on the cam table(Note 1)
	nElements	INT	0	Number of data elements in the cam table array(Note 1)
	nTappets	INT	0	Number of data elements in the tappet table array(Note 2)
	pce	POINTER TO BYTE	-	Specifies the address of a data element in the cam table array
	pt	POINTER TO SMC_CAMTappet	-	Specifies the address of a data element in the tappet table array
	dwTappetActiveBits	DWORD	0	Do not use.
	strCAMName	STRING	''	Name of the created cam
	byInterpolationQuality	BYTE	1	Interpolation format parameter for the cam table array Only used if byType = 1 or 2 ● 1: Linear interpolation ● 3: Cubic interpolation
	byCompatibilityMode	BYTE	0	1: Periodically executes cam tables with master cycle(Note 3)

Scope	Name	Type	Default value	Description
	bChangedOnline	BOOL	-	Do not use.
	xPartofLM	BOOL	FALSE	Do not use.

(Note 1) Specify a value to suit the format of the cam table array you input. If a value different from input data is specified, the system does not operate properly.

(Note 2) Specify a value to suit the format of the tappet table array you input. If a value different from input data is specified, the system does not operate properly.

(Note 3) This periodic operation is similar to that performed when Periodic of MC_CamTableSelect is set to TRUE.

■ List of data structures of cam tables

- For byType = 1

  SMC_CAMTable_***_128_1, SMC_CAMTable_***_256_1 (Structure: *** is LREAL, REAL, UDINT, or UINT)

  PMC_CAMTable_***_N_1 (Structure: N is 128, 256, 1001, 3601, 10001, or 30001 and *** is LREAL, REAL, UDINT, or UINT)

MC_CAM_REF with byType = 1 has a one-dimensional cam table of slave positions that are arranged at equal intervals. In each structure, the following parameters exist.

Name	Type	Description
Table	ARRAY [0..N] OF *** N: 127, 255, 1000, 3600, 10000, 30000	Array data storing slave positions The number of array elements and the variable type differ depending on the structure. The available variable types are LREAL, REAL, UDINT, and UINT.
fEditorMasterMin	REAL	Parameters representing master and slave motion scales(Note 1)
fEditorMasterMax	REAL	
fEditorSlaveMin	REAL	
fEditorSlaveMax	REAL	
fTableMasterMin	REAL	
fTableMasterMax	REAL	
fTableSlaveMin	REAL	
fTableSlaveMax	REAL	

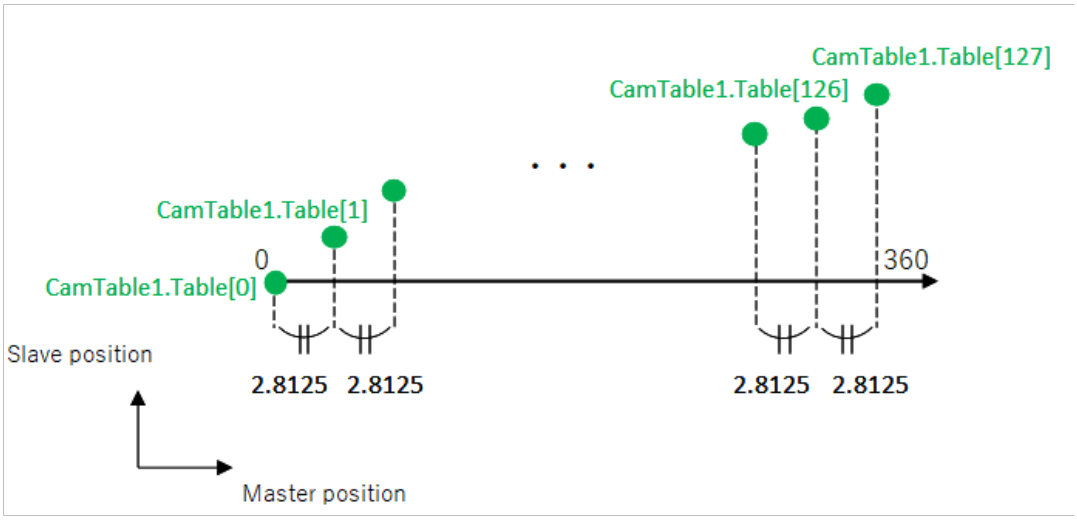
(Note 1) Set each value so as to agree with the actual motion range. The settings must be configured such that fEditorMasterMin = fTableMasterMin, fEditorMasterMax = fTableMasterMax, fEditorSlaveMin = fTableSlaveMin, and fEditorSlaveMax = fTableSlaveMax.

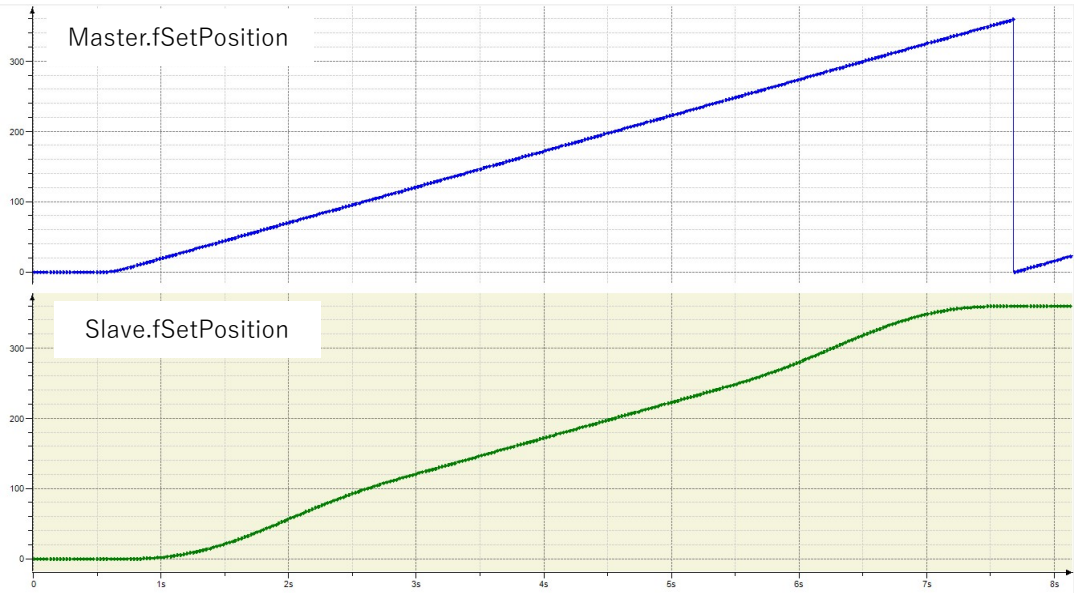
In the Table parameter, set slave positions as many as the number of elements in the array data. These values are equally arranged within the master axis motion range (the range between the xStart and xEnd inputs of MC_CAM_REF).

In one example, parameters of SMC_CAMTable_LREAL_128_1 that are configured so as to have the following waveform are shown. A cam is defined with the settings byVarType = 6, xStart = 0, xEnd = 360, and nElements = 128. At this time, the slave positions are arranged from the beginning of the Table data array at intervals that are each separated by a master position width of (360 - 0) / 128 = 2.8125.

```
CamTable1 : SMC_CAMTable_LREAL_128_1:=( fEditorMasterMin:=0, fEditorMasterMax
:=360, fTableMasterMin:=0, fTableMasterMax:=360, fEditorSlaveMin:=0, fEditorS
```

```
laveMax:=360, fTableSlaveMin:=0, fTableSlaveMax:=360, Table:=[ 0, 0.0092, 0.0
712, ..., 359.9287, 359.9908, 360]);
```





Info.

- It is possible for you to set a structure that has a desired number of array elements. For details, refer to "6.2.12 Sample Example: Create MC_CAM_REF by POU".
- For byType = 2

  SMC_CAMTable_***_128_2, SMC_CAMTable_***_256_2 (Structure: *** is LREAL, REAL, UDINT, or UINT)

  PMC_CAMTable_***_N_2 (Structure: N is 128, 256, 1001, 3601, 10001, or 30001 and *** is LREAL, REAL, UDINT, or UINT)

MC_CAM_REF with byType = 2 has a two-dimensional cam table in which master positions and corresponding slave positions are specified. In each structure, the following parameters exist.

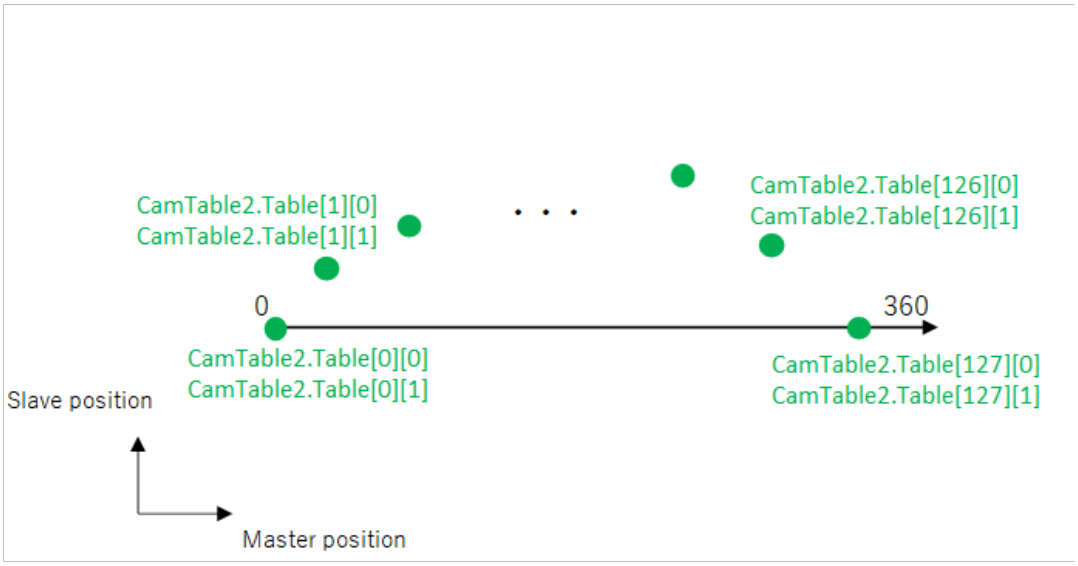
Name	Type	Description
Table	ARRAY [0..N] OF ARRAY [0..1] OF *** N: 127, 255, 1000, 3600, 10000, 30000	Array data storing master positions and slave positions The number of array elements and the variable type differ depending on the structure. The available variable types are LREAL, REAL, UDINT, and UINT.
fEditorMasterMin	REAL	Parameters representing master and slave motion scales(Note 1)
fEditorMasterMax	REAL	
fEditorSlaveMin	REAL	
fEditorSlaveMax	REAL	
fTableMasterMin	REAL	
fTableMasterMax	REAL	
fTableSlaveMin	REAL	
fTableSlaveMax	REAL	

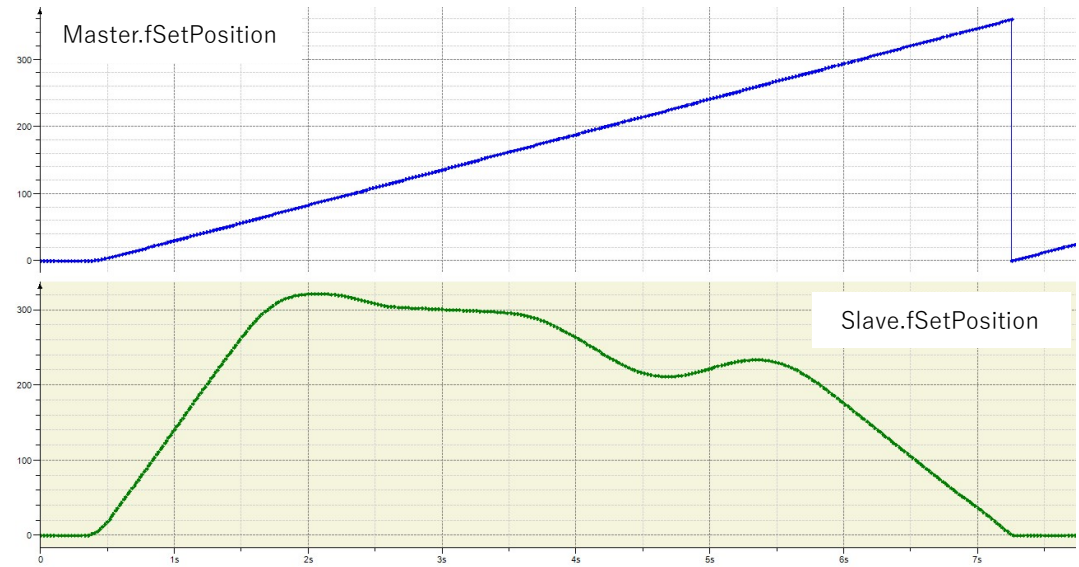
(Note 1) Set each value so as to agree with the actual motion range. The settings must be configured such that fEditorMasterMin = fTableMasterMin, fEditorMasterMax = fTableMasterMax, fEditorSlaveMin = fTableSlaveMin, and fEditorSlaveMax = fTableSlaveMax.

In the Table parameter, set master position and slave positions as many as the number of elements in the array data.

In one example, parameters of SMC_CAMTable_LREAL_128_2 that are configured so as to have the following waveform are shown. A cam is defined with the settings byVarType = 6, xStart = 0, xEnd = 360, and nElements = 128.

```
CamTable2 : SMC_CAMTable_LREAL_128_2:=( fEditorMasterMin:=0, fEditorMasterMax
:=360, fTableMasterMin:=0, fTableMasterMax:=360, fEditorSlaveMin:=0, fEditorS
laveMax:=360, fTableSlaveMin:=0, fTableSlaveMax:=360, Table:=[ [0, 0], [2.834
6, 13.2733], [5.6693, 26.5467], ..., [354.3307, 15.0461], [357.1654, 7.5231],
 [360, 360]]);
```





Info.

- It is possible for you to set a structure that has a desired number of array elements. For details, refer to "6.2.12 Sample Example: Create MC_CAM_REF by POU".

- For byType = 3

  SMC_CAMXYVA (Structure)

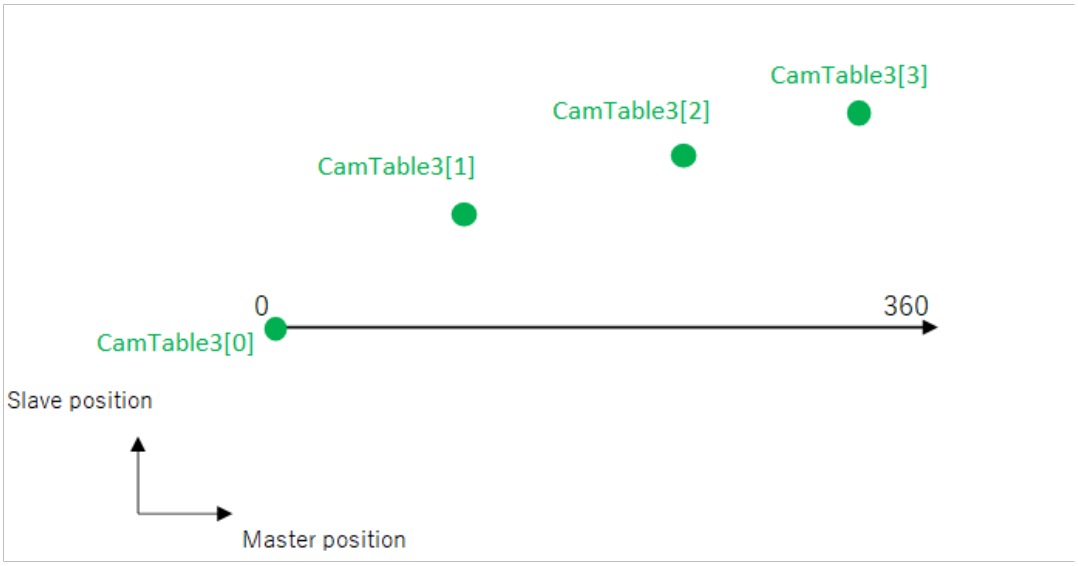
MC_CAM_REF with byType = 3 has a cam table in which master positions (X) and corresponding slave positions (Y), slave velocity values (V), and slave acceleration values (A) are specified. In this type, regardless of the byInterpolationQuality setting, cubic interpolation is applied to the path between adjacent data points along the slave axis. In the structure, the following parameters exist.

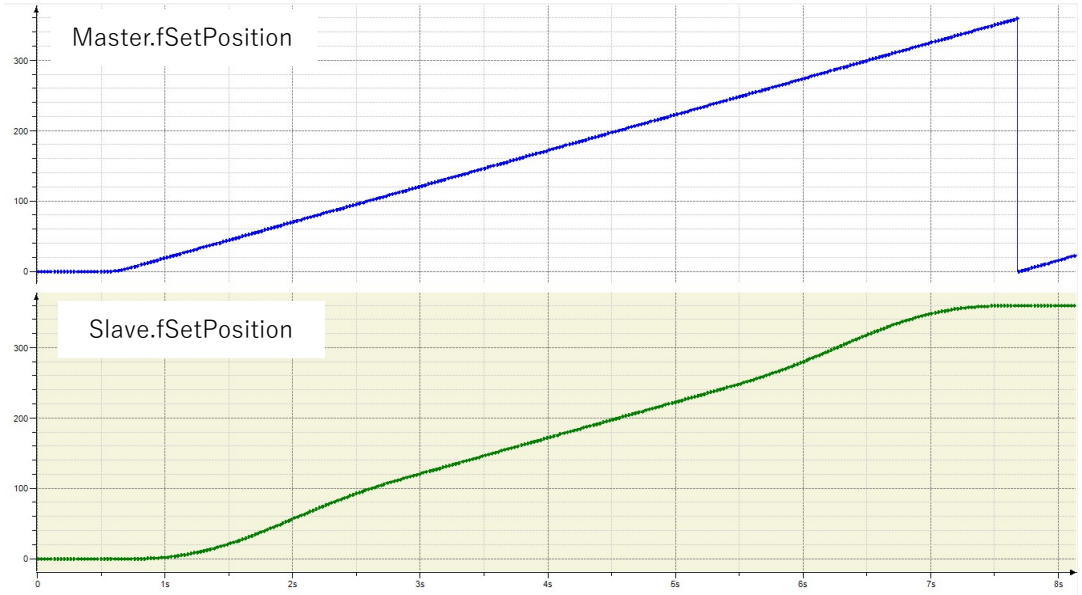
Name	Type	Description
dX	LREAL	Master position
dY	LREAL	Slave position
dV	LREAL	Slave velocity
dA	LREAL	Slave acceleration

For byType = 3, the cam table has array data of the SMC_CAMXYVA type.

In one example, parameters that are configured so as to have the following waveform are shown. A cam is defined with the settings xStart = 0, xEnd = 360, and nElements = 4.

```
CamTable3 : ARRAY[0..3] OF SMC_CAMXYVA:=[ (dX:=0, dY:=0, dV:=0, dA:=0), (dX:=
120, dY:=120, dV:=1, dA:=0), (dX:=240, dY:=240, dV:=1, dA:=0), (dX:=360, dY:=
360, dV:=0, dA:=0)];
```





■ List of data structures of tappet tables

● SMC_CAMTappet (Structure)

This structure describes the tappets of a table referenced by MC_CAM_REF.

Name	Type	Default value	Description
ctt	SMC_CAMTAPPETTYPE	0	Specifies the direction in which the tappet point is passed through by the axis to make tappet action active.
cta	SMC_CAMTAPPETACTION	0	Specifies what switching action is executed when the tappet is passed.
dwDelay	DWORD	0	Specifies the delay time in µs before the tappet is switched to ON following the axis having passed through the tappet point to make tappet action active. This setting is enabled when cta = TAPPETACTION_time.
dwDuration	DWORD	0	Specifies the time in µs, for which the tappet is switched to on.(Note 1) This setting is enabled when cta = TAPPETACTION_time.
iGroupID	INT	0	Tack ID for tappet arrangement
x	INT	0	Axis position where tappet is switched to ON.
dwActive	DWORD	16#FFFFFFFF	This is used as an internal variable. Do not set this.

(Note 1) When this parameter is set to 0, the tappet does not operate.

● SMC_CAMTAPPETTYPE (Enumeration type)

This determines the direction in which the tappet has to be passed through by the axis to make tappet action active.

Name	Value	Description
TAPPET_pos	0	Action of the tappet is made active when the axis passes through the position of the tappet in positive direction.
TAPPET_all	1	Action of the tappet is made active when the axis passes through the position of the tappet in any of positive and negative directions.
TAPPET_neg	2	Action of the tappet is made active when the axis passes through the position of the tappet in negative direction.

- SMC_CAMTAPPETACTION (Enumeration type)

This determines what switching action is executed when tappet action is made active.

Name	Value	Description
TAPPETACTION_on	0	Switches on tappet.
TAPPETACTION_off	1	Switches off tappet.
TAPPETACTION_inv	2	Inverts tappet switching on and off.
TAPPETACTION_time	3	Enables the dwDelay and dwDuration settings of SMC_CAMTappet.

Info.

- Up to three tappets can be specified for a shared point x irrespective of track ID. When the axis reaches the shared point for which four or more tappets are specified, FBs output the SMC_AXIS_NOT_READY_FOR_MOTION error to MC_CamIn and the SMC_CI_TOO_MANY_TAPPETS_PER_CYCLE error to the slave axis.
- For examples that show the way of creating a tappet table by programing or the way of using dwDelay and dwDuration, refer to"6.2.12 Sample Example: Create MC_CAM_REF by POU".

■ byInterpolationQuality

An interpolation format for slave positions during cam control can be specified through byInterpolationQuality. This function is activated on cam tables (one-dimensional tables, two-dimensional tables) with byType = 1 or 2.

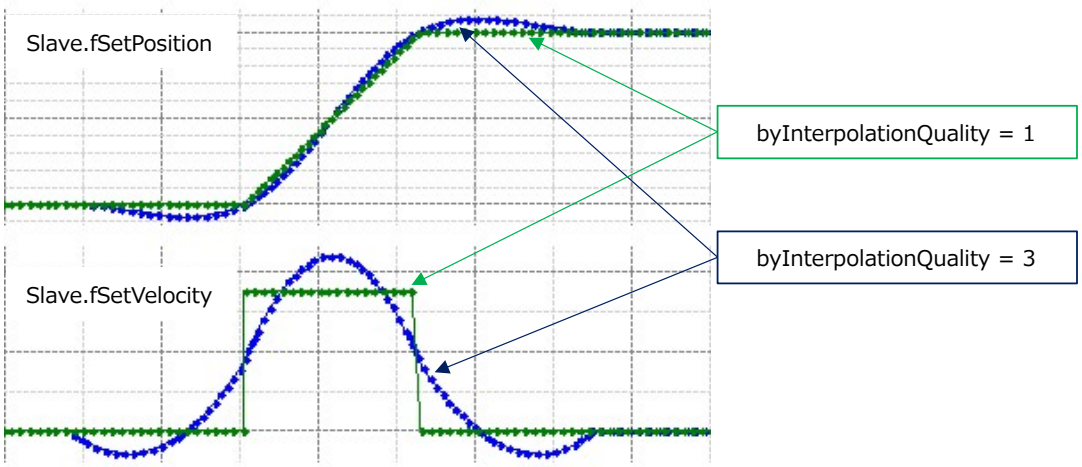
- byInterpolationQuality = 1: Linear interpolation

With this interpolation format, linear interpolation is applied to points on the cam table and thus the position, velocity, and other data about slave axis commands change discretely.

- byInterpolationQuality = 3: Cubic interpolation

With this interpolation format, polynomial interpolation is applied to points on the cam table. Thus, data about slave axis commands changes smoothly. However, an overshoot is likely to occur in position and velocity.

Motion curves by the respective interpolation formats are as shown below.



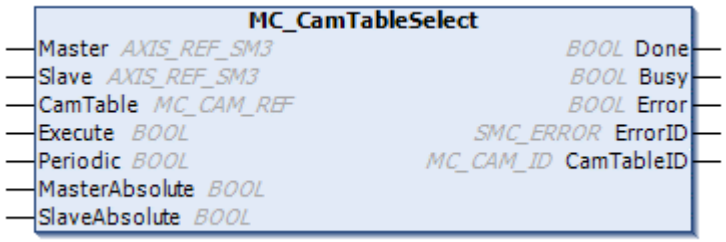
Info.

- For the XYVA table format with byType = 3, interpolation is performed between the tables regardless of the byInterpolationQuality setting such that the specified velocity and acceleration are attained.

6.2.3 MC_CamTableSelect (Select Cam Profile)

This is a function block (FB) that specifies a cam profile (MC_CAM_REF) for cam synchronous operation. When the cam profile to be used is selected, a cam table ID is output. The cam profile can be created by tools or through a program.

■ Icon



■ Parameter

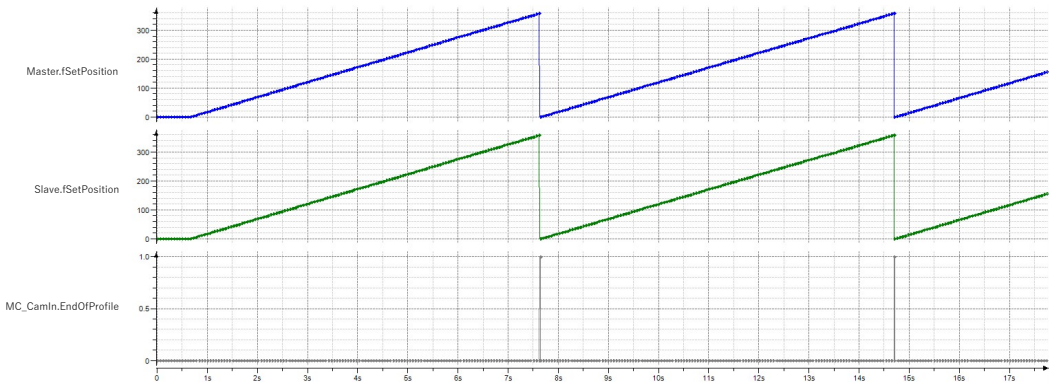
Scope	Name	Type	Default value	Description
Input / output	Master	AXIS_REF_SM3	-	Specifies the master axis for cam synchronous operation.
	Slave	AXIS_REF_SM3	-	Specifies the slave axis for cam synchronous operation.

Scope	Name	Type	Default value	Description
	CamTable	MC_CAM_REF	-	Specifies the cam profile."6.2.2 MC_CAM_REF (Cam Profile)"
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Periodic	BOOL	TRUE	Sets periodic cam synchronous operation. TRUE: Repeat execution FALSE: 1-period execution
	MasterAbsolute	BOOL	TRUE	TRUE: Cam refers to absolute master position FALSE: Cam refers to relative master position
	SlaveAbsolute	BOOL	TRUE	TRUE: Cam refers to absolute slave position FALSE: Cam refers to relative slave position
Output	Done	BOOL	FALSE	TRUE: Execution of the FB is completed.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.
	CamTableID	MC_CAM_ID	-	Cam table ID Specify this ID for use by CamTableID of MC_CamIn.

■ Periodic (Periodic cam control)

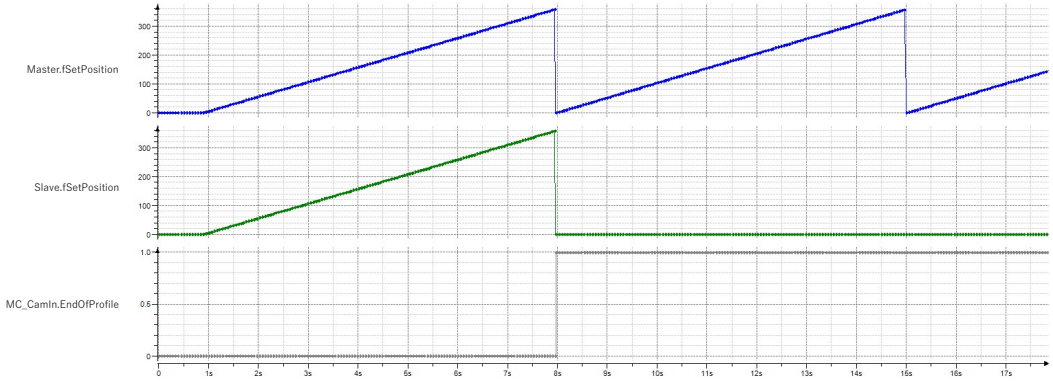
- Periodic = TRUE

If Periodic of MC_CamTableSelect is set to TRUE, cam synchronous operation is periodically repeated. Synchronous operation is automatically restarted when the axis position reaches the end position on the cam table.



- Periodic = FALSE

If Periodic is set to FALSE, when the axis position reaches the master end position of cam synchronous operation, EndOfProfile of MC_CamIn changes to TRUE and the slave stops at the current position. Meanwhile, the master keeps the motion as is.

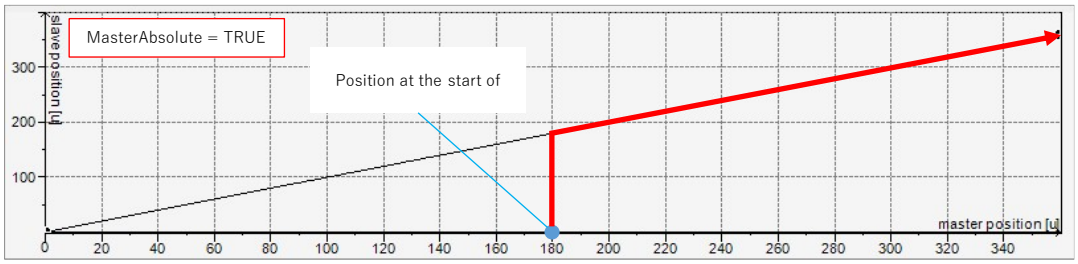


■ MasterAbsolute

● MasterAbsolute = TRUE

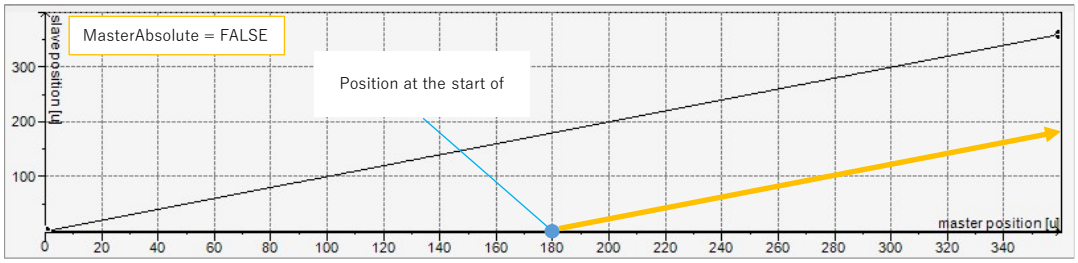
If MasterAbsolute is set to TRUE, cam synchronous operation starts from the current master position on the cam table. Any master position within the cam table range can be set as the starting position. The motion cycle of this operation conforms with the master cycle in the specified cam table (whenever the master reaches the xEnd input of MC_CAM_REF, EndOfProfile of MC_CamIn is set to TRUE).

If the master position at the time of start of cam synchronous operation is outside the cam table range, an error occurs.



● MasterAbsolute = FALSE

If MasterAbsolute is set to FALSE, cam synchronous operation starts, with the current master position being as the zero point. This operation can be executed only when the value 0 is included in the master range in the cam table. The motion cycle of this operation conforms with the cycle of the specified cam table (width of the cam table) (whenever one cycle of the cam table is completed, EndOfProfile of MC_CamIn is set to TRUE).



■ SlaveAbsolute

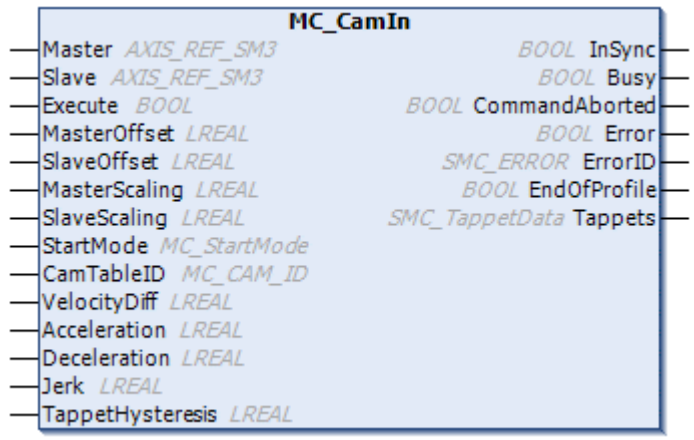
The SlaveAbsolute input affects StartMode at the time of start of cam synchronous operation. StartMode applied to cam synchronous operation is determined by a combination of the StartMode input of MC_CamIn and the SlaveAbsolute input of MC_CamTableSelec. Mode varieties determined by the respective combinations are as shown below.

MC_CamIn.StartMode	MC_CamTableSelect.SlaveAbsolute	Applied StartMode
absolute	TRUE	absolute
absolute	FALSE	relative
relative	TRUE	relative
relative	FALSE	relative
ramp_in	TRUE	ramp_in absolute
ramp_in	FALSE	ramp_in relative
ramp_in_pos	TRUE	ramp_in_pos absolute
ramp_in_pos	FALSE	ramp_in_pos relative
ramp_in_neg	TRUE	ramp_in_neg absolute
ramp_in_neg	FALSE	ramp_in_neg relative

6.2.4 MC_CamIn (Start Cam Synchronization)

This is a function block (FB) that starts cam synchronous operation. The master axis and the slave axis in synchronization operate according to the cam table.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Master	AXIS_REF_SM3	-	Specifies the master axis for cam synchronous operation.
	Slave	AXIS_REF_SM3	-	Specifies the slave axis for cam synchronous operation.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	MasterOffset	LREAL	0	Offset on master profile
	SlaveOffset	LREAL	0	Offset on slave profile
	MasterScaling	LREAL	1	Scaling factor for master profile
	SlaveScaling	LREAL	1	Scaling factor for slave profile
	StartMode	MC_StartMode	absolute	Specifies operation mode at the time of start of cam synchronous operation.
	CamTableID	MC_CAM_ID	-	Dynamic cam table ID Specifies the CamTableID output of MC_CamTableSelect.
	VelocityDiff	LREAL	0	Maximum velocity difference (u/s) for ramp_in mode when StartMode = ramp_in
	Acceleration	LREAL	0	Acceleration (u/s^2) for ramp_in mode when StartMode = ramp_in
	Deceleration	LREAL	0	Deceleration (u/s^2) for ramp_in mode when StartMode = ramp_in
	Jerk	LREAL	0	Jerk (u/s^3) for ramp_in mode when StartMode = ramp_in
	TappetHysteresis	LREAL	0	Hysteresis value for tappet action in (u) Performs action when the value is 0 or higher.
Output	InSync	BOOL	FALSE	TRUE: Cam synchronization is completed.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE: An interruption to operation from another FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.
	EndOfProfile	BOOL	FALSE	A pulse is output (TRUE) every time the cam profile period of the slave ends.
	Tappets	SMC_TappetData	-	Tappet data from the active cam table Specified for the Tappets input/ output of SMC_GetTappetValue.

■ Changing the scale and offset in the cam table

- MasterOffset, MasterScaling

The master position is converted by the MasterOffset and MasterScaling inputs. The cam table operates relative to the converted master position. The conversion formula is as follows.

Cam master position after conversion = Cam master position before conversion × MasterScaling + MasterOffset

- SlaveOffset, SlaveScaling

The slave position is converted by the SlaveOffset and SlaveScaling inputs. The conversion formula is as follows.

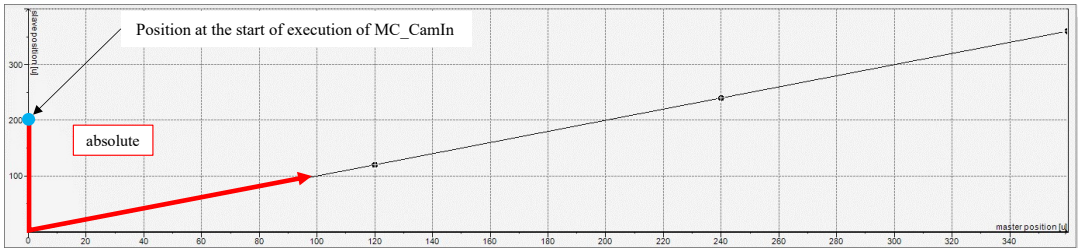
Cam slave position after conversion = Cam slave position before conversion × SlaveScaling + SlaveOffset

■ MC_StartMode (Enumeration type)

- absolute

A command value to the slave is such that the slave position is adjusted to a position relative to the master position according to the cam profile.

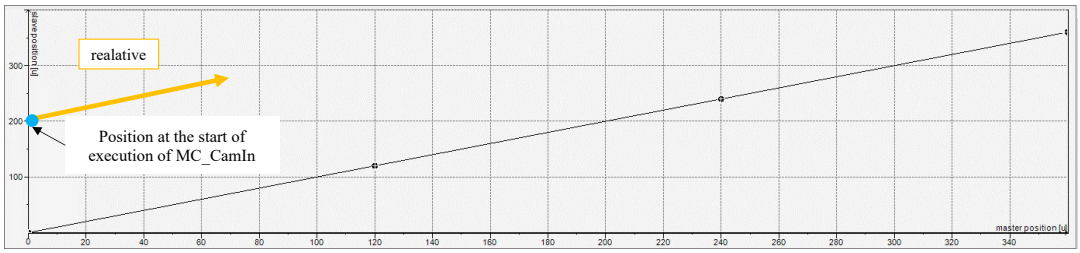
If the slave position is different from the start position on the cam table when MC_CamIn is executed, the position jumps to agree with the synchronous position on the cam table.



- relative

A command value relatively follows the cam profile such that the slave position is adjusted from the current slave position irrespective of the master position.

When MC_CamIn is executed, cam synchronous operation starts with the current slave position regarded as the start position. The slave motion position is the slave position at the time of execution of MC_CamIn + the original cam table value. It must be noted, however, that the position jumps if the slave start position on the cam table is not 0.

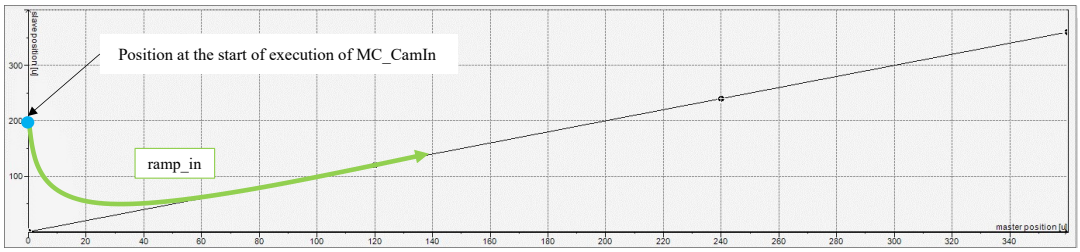


- ramp_in, ramp_in_neg, ramp_in_pos

When MC_CamIn is executed, the slave position goes to the position on the cam table. The velocity and acceleration during travel are the values on which VelocityDiff, Acceleration, Deceleration, and Jerk are superimposed for the velocity profile of the slave axis relative to the current position of the master axis.

When the slave position reaches the position on the cam table, the cam gets into the synchronized state (InSync = TRUE). If the slave axis is a modulo axis, correction is made only

in the positive direction when the mode is set to ramp_in_pos, while correction is made only in the negative direction when the mode is set to ramp_in_neg. With the finite axis, ramp_in_pos and ramp_in_neg are treated as ramp_in.



Info.

The final StartMode is determined by MC_CamIn.StartMode and MC_CamTableSelect.SlaveAbsolute. For details, refer to"6.2.3 MC_CamTableSelect (Select Cam Profile)".

■ Tappet

By setting TappetHysteresis, tappet action chattering can be filtered.

When the master reaches the tappet position, tappet processing is performed. After that, the master needs to move away from the tappet by at least the distance specified for TappetHysteresis before the same tappet processing is performed again. When the distance by which the master has traveled is smaller than the distance specified for TappetHysteresis, the tappet processing is not performed even if the master reaches the tappet action position. Even for the tappet at the same position, if tappet actions differ between positive pass and negative pass, distances traveled are determined in either of the travel directions. (In other words, this is the function of filtering an identical array element in the tappet table.)

Examples are shown below.

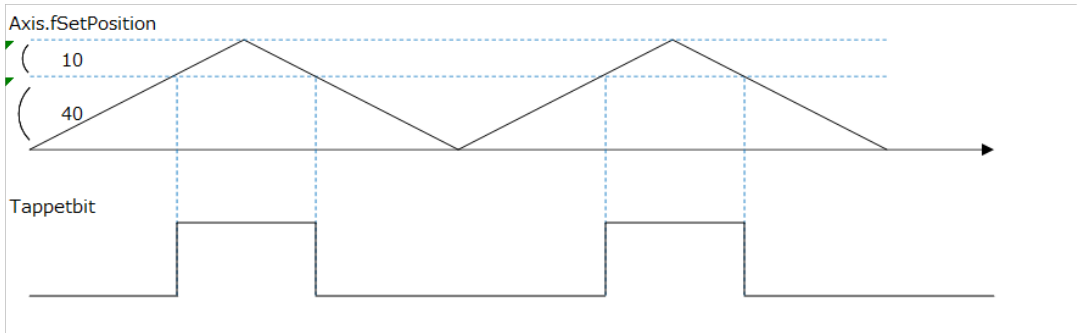
- TappetHysteresis = 0: Tappet processing consists of invert for both positive pass and negative pass

Example of tappet table

```
TappetTable : ARRAY[0..0] OF SMC_CAMTappet:=[(x:=40, ctt:=1, iGroupID:=1, cta
:=2)];
```

Because of TappetHysteresis = 0, tappet processing is not restricted in travel distance. When the master reaches the tappet position, processing is performed.

Example of motion curves



- TappetHysteresis = 30: Tappet processing consists of invert for both positive pass and negative pass

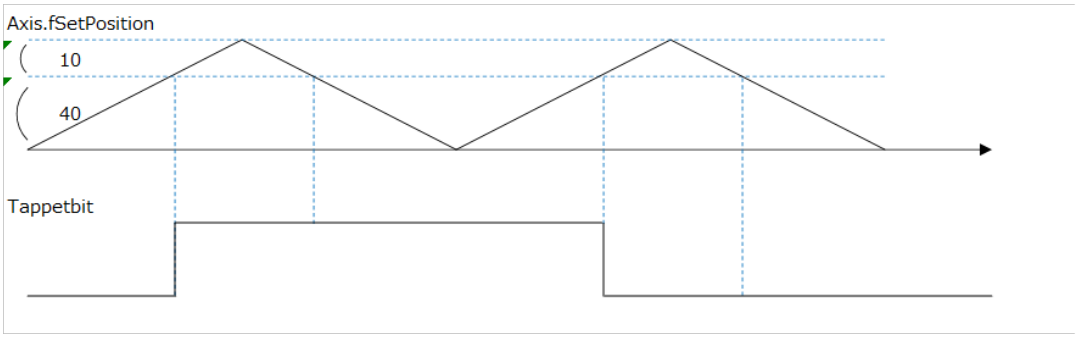
Example of tappet table

```
TappetTable : ARRAY[0..0] OF SMC_CAMTappet:=[(x:=40, ctt:=1, iGroupID:=1, cta
:=2)];
```

Because of TappetHysteresis = 30, the same tappet processing is not performed unless the master travels to 30 or more. In this example, the tappet position is set to 40. The tappet first performs action (switch invert ON) at the place of 40 and does not perform next switch action unless the master travels to a place of 40 + 30 = 70 or more or to a place of 40 - 30 = 10 or less and then reaches the tappet position 40 again.

For the next curve, after the tappet first performs action (switch invert ON) at the place of 40, the master moves away by 10 and reaches the tappet action position. Since the distance by which the master has moved away is smaller than the TappetHysteresis value, tappet processing is not performed. When the master reaches the place of 40 again, tappet processing (switch invert OFF) is performed because the master has traveled once to 0 (= at a distance of 40) and consequently moved away by the TappetHysteresis value or more.

Example of motion curves



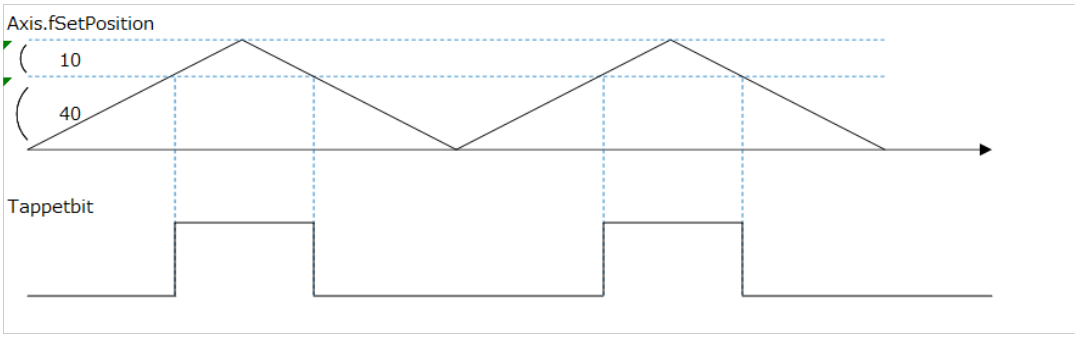
- TappetHysteresis = 30: Tappet processing consists of switch ON for positive pass and switch off for negative pass

Example of tappet table

```
TappetTable : ARRAY[0..1] OF SMC_CAMTappet:=[ (x:=40, ctt:=2, iGroupID:= 1, c
ta:=1), (x:=40, ctt:=0, iGroupID:=1, cta:=0)];
```

Since contents of the tappet processing differ between the positive pass and negative pass, whether or not the TappetHysteresis criterion is satisfied is assessed in either of the motion directions. Thus, since the master has traveled by the TappetHysteresis value or more, the tappet performs action even at the same position.

Example of motion curves



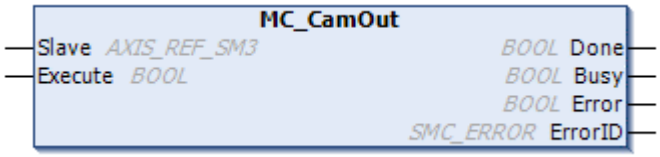
- SMC_TappetData (Structure)

Name	Type	Description
pTaps	ARRAY [0..2] OF POINTER TO SMC_CAMTappet	Used internally for the output of MC_CamIn and for the input of SMC_GetTappetValue.
dwCycleTime	DWORD	
byChannels	BYTE	
bRestart	BOOL	

6.2.5 MC_CamOut (Cancel Cam Synchronization)

This is a function block (FB) to release cam synchronization. After cam synchronization is released, the state of the slave axis will become continuous motion. If you want to return the state of the axis to standstill, please execute MC_Halt or MC_Stop.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Slave	AXIS_REF_SM3	-	Specifies the slave axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
Output	Done	BOOL	FALSE	TRUE: Synchronization cancellation is completed.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	Error	BOOL	FALSE	TRUE: An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

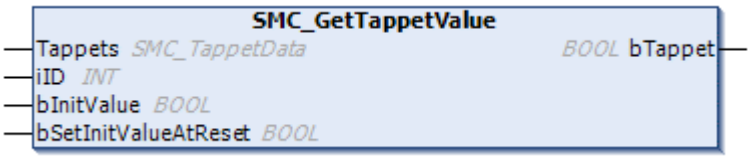
Info.

- If this FB is executed for a slave axis that is not in cam synchronization state, an error SMC_AXIS_NOT_READY_FOR_MOTION will occur.
- After cam synchronization is released, the state of the slave axis will continue to be continuous motion. Since the axis is in an operating state during continuous motion, it is necessary to call this FB at every control cycle even after cam synchronization is complete.
- After the release of cam synchronization, the state of the slave axis continues to be in continuous motion, therefore the slave axis continues speed control at the speed command value at the time of cam synchronization release.
- If you want to stop the slave axis after releasing cam synchronization, please execute MC_Halt or MC_Stop.

6.2.6 SMC_GetTappetValue (Get Single Tappet Information)

This is a function block (FB) that gets the status of the tappet performing switching action relative to the specified current master position by batch based on tappet information defined by MC_CAM_REF. Specify the Tappets output of MC_CamIn for the input of SMC_GetTappetValue to perform tappet output. The tappet that can be output is only one track for one instance.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Tappets	SMC_TappetData	-	Specifies the output Tappets of MC_CamIn.
Input	iID	INT	0	Track ID of the tappet to be gotten(Note 1)
	bInitValue	BOOL	FALSE	The value of the tappet to be set by the bSetInitValueAtReset function
	bSetInitValueAtReset	BOOL	FALSE	TRUE: Sets the bTappet output to the initial value of bInitValue and starts. FALSE: Starts with the current tappet value.
Output	bTappet	BOOL	FALSE	Tappet switch output

(Note 1) The track ID can be changed during cam synchronous operation, but if you want to monitor multiple tappet statuses, prepare multiple instances.

Info.

- Call this FB together with MC_CamIn concurrently. If this FB is called while MC_CamIn is in progress, the system does not operate properly.
- This FB cannot be used concurrently with SMC_CamRegister. Do not call SMC_CamRegister when SMC_GetTappetValue is used.
- If the same MC_CAM_REF is used to perform more than one cam synchronous operation, tappet processing is performed only in the first synchronous operation called. To perform tappet processing with more than one synchronous operation, MC_CamTableSelect must be set as follows.

  Example: Using Cam1 in two synchronous operations to perform tappet processing in each of them

```
MC_CAM_REF_0 := Cam1;
MC_CAM_REF_1 := Cam1;
MC_CamTableSelect_0.CamTable := MC_CAM_REF_0;
MC_CamTableSelect_1.CamTable := MC_CAM_REF_1;
```

  Unacceptable example

```
MC_CamTableSelect_0.CamTable := Cam1;
MC_CamTableSelect_1.CamTable := Cam1;
```

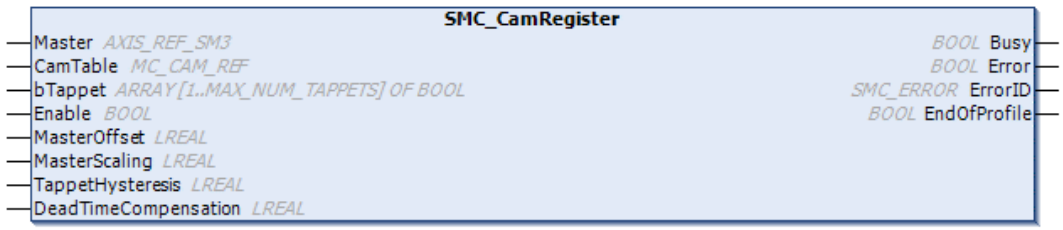
■ SMC_TappetData (Structure)

Name	Type	Description
pTaps	ARRAY [0..2] OF POINTER TO SMC_CAMTappet	Used internally for the output of MC_CamIn and for the input of SMC_GetTappetValue.
dwCycleTime	DWORD	
byChannels	BYTE	
bRestart	BOOL	

6.2.7 SMC_CamRegister (Get All Tappet Information)

This is a function block (FB) that gets the status of any tappet performing switching action relative to any specified current axis position by batch based on tappet information defined by MC_CAM_REF.

■ Icon



■ Parameter

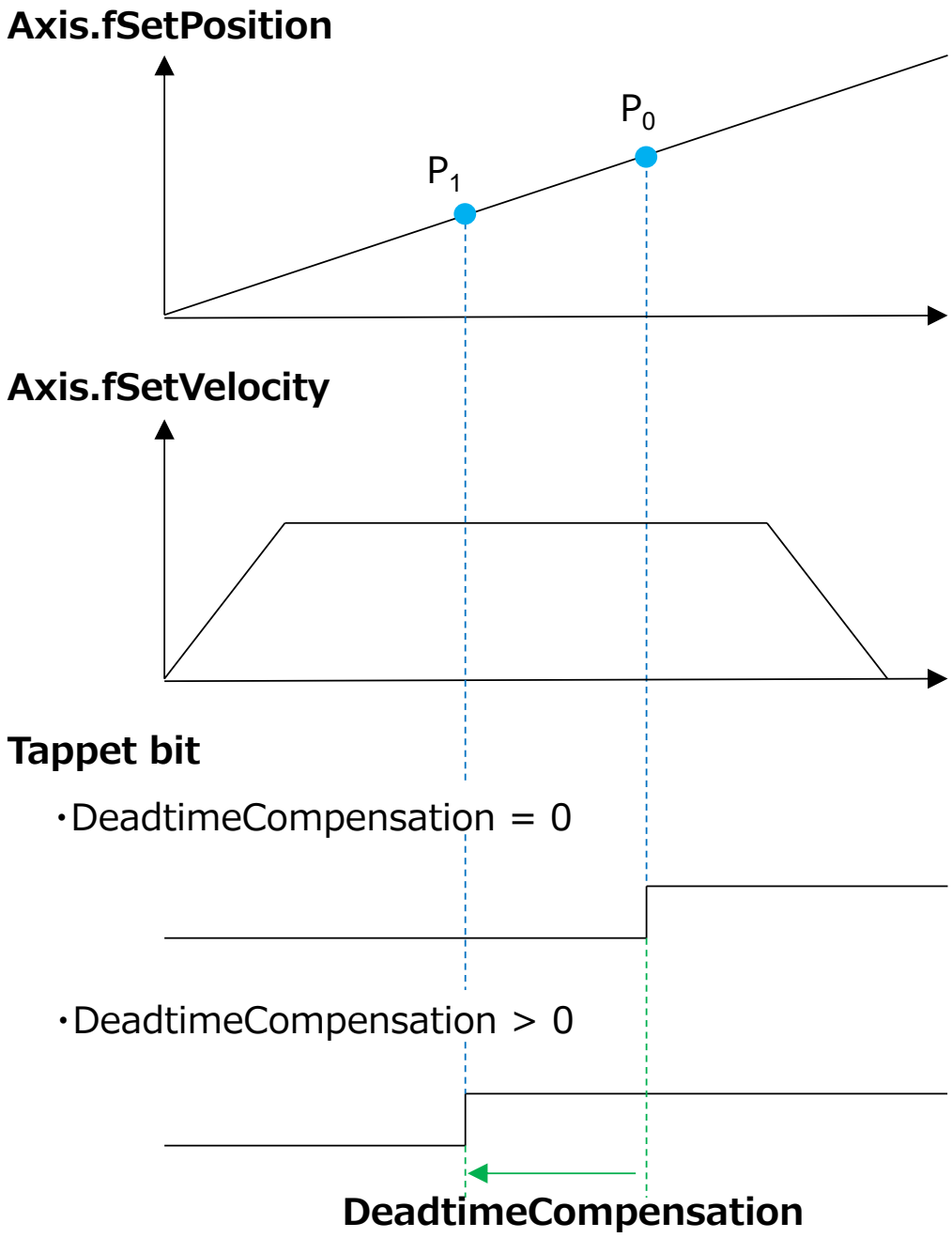
Scope	Name	Type	Default value	Description
Input / output	Master	AXIS_REF_SM3	-	Reference to master axis
	CamTable	MC_CAM_REF	-	Reference to a cam profile that causes tappet motion"6.2.2 MC_CAM_REF (Cam Profile)"
	bTappet	ARRAY [1..MAX_NUM_TAPPETS] OF BOOL	-	A bit array storing tappet information
Input	Enable	BOOL	FALSE	TRUE: Starts execution of function block.
	MasterOffset	LREAL	0	Offset to master position
	MasterScaling	LREAL	1	Scaling factor for the master
	TappetHysteresis(Note 1)	LREAL	0	Hysteresis value for tappet action in (u) Performs action when the value is 0 or higher.
	DeadtimeCompensation	LREAL	0	Dead time compensation (s)
Output	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.
	EndOfProfile	BOOL	FALSE	A pulse is output (TRUE) every time the cam profile period ends.

(Note 1) For details of TappetHysteresis, refer to "6.2.4 MC_CamIn (Start Cam Synchronization)".

■ DeadtimeCompensation

By setting a DeadtimeCompensation value, compensation can be provided for tappet dead time. The compensation is calculated by linear extrapolation. The formula is DeadtimeCompensation × master velocity.

An example of motion curves is shown below. In this example, when dead time compensation is not provided, the tappet performs action at a master position of P_0. When compensation is provided using DeadtimeCompensation, the tappet performs action at a master position of P_1. According to the formula above, the equation $P_1 = P_0$ - DeadtimeCompensation × V_1 is formulated.


Axis.fSetPosition
P0
P1
Axis.fSetVelocity
Tappet bit
・DeadtimeCompensation = 0
・DeadtimeCompensation > 0
DeadtimeCompensation

Info.

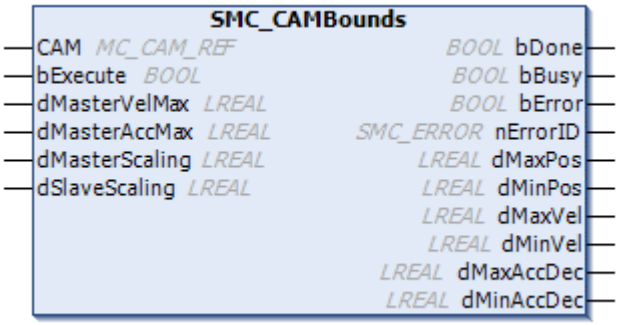
- While SMC_GetTappetValue is used to perform tappet processing relative to the master position specified by MC_CamIn, SMC_CamRegister is used to perform tappet processing relative to any axis position.

  With SMC_GetTappetValue, instances need to be created as many as the number of track IDs that are required to be monitored. Meanwhile, if input arguments of SMC_CamRegister are set to parameter values shared with MC_CamIn, you can get ON/OFF changes in all track IDs as array data by batch.
- This FB cannot be used concurrently with SMC_GetTappetValue. Do not call SMC_GetTappetValue when SMC_CamRegister is used.

6.2.8 SMC_CAMBounds (Calculate Maximum/Minimum Parameters of Slave)

This is a function block (FB) that calculates minimum/maximum position, velocity, and acceleration/deceleration values from information on the slave locus. This enables you to assess whether cam velocity and acceleration/deceleration settings are appropriate before cam synchronous operation. The cam compile format that is allowed to be used is only XYVA table.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	CAM	MC_CAM_REF	-	Reference to cam cam table(Note 1)
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	dMasterVelMax	LREAL	1	Maximum velocity of the master
	dMasterAccMax	LREAL	0	Maximum acceleration/deceleration of the master
	dMasterScaling	LREAL	1	Scaling factor for the master
	dSlaveScaling	LREAL	1	Scaling factor for the slave
Output	bDone	BOOL	FALSE	TRUE: Execution of the FB is completed.
	bBusy	BOOL	FALSE	TRUE: Execution of the FB is not completed.

Scope	Name	Type	Default value	Description
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	nErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.
	dMaxPos	LREAL	0	Maximum slave position value calculated in (u)
	dMinPos	LREAL	0	Minimum slave position value calculated in (u)
	dMaxVel	LREAL	0	Maximum slave velocity value calculated in (u/s)
	dMinVel	LREAL	0	Minimum slave velocity value calculated in (u/s)
	dMaxAccDec	LREAL	0	Maximum slave acceleration/deceleration value calculated in (u/s^2)
	dMinAccDec	LREAL	0	Minimum slave acceleration/deceleration value calculated in (u/s^2)

(Note 1) Specify MC_CAM_REF with byType = 3 (XYVA table)."6.2.2 MC_CAM_REF (Cam Profile)"

■ Detail of function

- Description of functions
  - By specifying the maximum velocity of the master (dMasterVelMax) and the maximum acceleration of the master (dMasterAccMax) for the cam table (MC_CAM_REF) set by CAM, this FB calculates the minimum/maximum slave position values (dMinPos/dMaxPos), minimum/maximum slave velocity values (dMinVel/dMaxVel), and minimum/maximum slave acceleration/deceleration values (dMinAccDec/dMaxAccDec).
  - The minimum/maximum slave position values (dMinPos/dMaxPos) change in proportion to the dSlaveScaling input.
  - The minimum/maximum slave velocity values (dMinVel/dMaxVel) change in proportion to the dMasterVelMax, dMasterScaling, and dSlaveScaling inputs.
  - The minimum/maximum slave acceleration/deceleration values (dMinAccDec/dMaxAccDec) change in proportion to the dMAsterAccMax and dSlaveScaling inputs and change in proportion to the square of the dMasterVelMax and dMasterScaling inputs.
- Usage example
  - To implement cam synchronous control by controlling the master at a constant velocity through the MC_MoveVelocity FB, configure the following settings in SMC_CAMBounds. If any of the dMax/dMinPos, dMax/dMinVel, and dMax/dMinAccDec outputs acquired through the implementation exceed the slave limitation parameter, take steps such as changing the motion velocity of the master or changing the slave limitation parameter.

    dMasterVelMax = the Velocity setting in MC_MoveVelocity

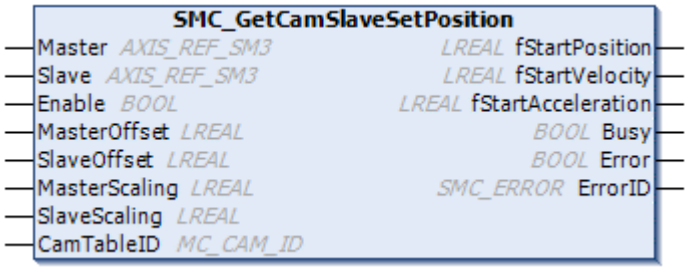
    dMasterAccMax = larger one of the Acceleration and Deceleration settings in MC_MoveVelocity

    dMasterScaling and dSlaveScaling = the MasterScaling and SlaveScaling settings in MC_CamIn

6.2.9 SMC_GetCamSlaveSetPosition (Calculate Condition for Slave Synchronization Start)

This is a function block (FB) that calculates position, velocity, and acceleration values of the slave axis relative to the current position of the master axis. If you want to start synchronization of the slave at a desired position of the master axis, this FB can be used to get a command value for control of the slave at the synchronization start position.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Master	AXIS_REF_SM3	-	Specifies the master axis for cam synchronous operation.
	Slave	AXIS_REF_SM3	-	Specifies the slave axis for cam synchronous operation.
Input	Enable	BOOL	FALSE	TRUE: Starts the execution of the FB.
	MasterOffset	LREAL	0	Offset on master profile
	SlaveOffset	LREAL	0	Offset on slave profile
	MasterScaling	LREAL	1	Scaling factor for master profile
	SlaveScaling	LREAL	1	Scaling factor for slave profile
	CamTableID	MC_CAM_ID	-	Cam table ID Specifies the CamTableID output of MC_CamTableSelect.
Output	fStartPosition	LREAL	0	Slave position set in (u) according to current master position at the start of cam operation
	fStartVelocity	LREAL	0	Slave velocity set in (u/s) according to current master position at the start of cam operation
	fStartAcceleration	LREAL	0	Slave acceleration set in (u/s^2) according to current master position at the start of cam operation
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.

Scope	Name	Type	Default value	Description
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

6.2.10 Sample Example: Allow Different MC_CAM_REF Profiles to Work

This is an example of a program designed to continue cam control by changing a cam profile (MC_CAM_REF) used during the cam control to another.

■ [Program example 1]

This program runs cam control by switching Cam1 with which the cam control is in progress to Cam2 with which the cam control continues.

- Declaration section

```
// Variables
    iProcess             : INT:=0;
    bExe_Movevel         : BOOL;
    bExe_Halt1           : BOOL;
    bExe_Tabsel0         : BOOL;
    bExe_Camin0          : BOOL;
    bExe_Tabsel1         : BOOL;
    bExe_Camin1          : BOOL;

// FB instances
    MC_MoveVelocity_1     : MC_MoveVelocity;
    MC_Halt_1             : MC_Halt;
    MC_CamTableSelect_0   : MC_CamTableSelect;
    MC_CamIn_0            : MC_CamIn;
    MC_CamTableSelect_1   : MC_CamTableSelect;
    MC_CamIn_1            : MC_CamIn;
    MC_CAM_REF_0          : MC_CAM_REF;
```

- Implementation section

```
// CAM FBs Settings
MC_MoveVelocity_1(
                    Axis:=Drive_Master,
                    Execute:=bExe_Movevel,
                    Velocity:=100,
                    Acceleration:=1000,
                    Deceleration:=1000,
);

MC_CamTableSelect_0(
                    Master:=Drive_Master,
                    Slave:=Drive_Slave,
                    CamTable:=Cam1,
                    Execute:=bExe_Tabsel0,
                    Periodic:=FALSE,
                    MasterAbsolute:=TRUE,
                    SlaveAbsolute:=FALSE,
);

MC_CamIn_0(
            Master:=Drive_Master,
            Slave:=Drive_Slave,
            Execute:=bExe_Camin0,
            MasterOffset:=0,
```

```
            SlaveOffset:=0,
            MasterScaling:=1,
            SlaveScaling:=1,
            StartMode:=relative,
            CamTableID:=MC_CamTableSelect_0.CamTableID,
            VelocityDiff:=100,
            Acceleration:=1000,
            Deceleration:=1000,
            Jerk:=10000,
            TappetHysteresis:=0,
);

MC_CamTableSelect_1(
                    Master:=Drive_Master,
                    Slave:=Drive_Slave,
                    CamTable:=Cam2,
                    Execute:=bExe_Tabsel1,
                    Periodic:=FALSE,
                    MasterAbsolute:=TRUE,
                    SlaveAbsolute:=FALSE,
);

MC_CamIn_1(
            Master:=Drive_Master,
            Slave:=Drive_Slave,
            Execute:=bExe_Camin1,
            MasterOffset:=0,
            SlaveOffset:=0,
            MasterScaling:=1,
            SlaveScaling:=1,
            StartMode:=relative,
            CamTableID:=MC_CamTableSelect_1.CamTableID,
            VelocityDiff:=100,
            Acceleration:=1000,
            Deceleration:=1000,
            Jerk:=10000,
            TappetHysteresis:=0,
);

MC_Halt_1(
            Axis:=Drive_Master,
            Execute:=bExe_Halt1,
            Deceleration:=10000,
            Jerk:=10000,
);

CASE iProcess OF
    0:// Load CamTable
        bExe_Tabsel0:=TRUE;
        IF MC_CamTableSelect_0.Done = TRUE THEN
            iProcess:=1;
        END_IF

    1:// Start Cam Sync and moving
        bExe_Camin0:=TRUE;
        bExe_Movevel:=TRUE;
```

```
        iProcess:=2;

    2:// Start 2nd Cam Loading
        bExe_Tabsel1:=TRUE;
        IF MC_CamTableSelect_1.Done = TRUE THEN
            iProcess:=3;
        END_IF

    3:// Start 2nd Cam sync
        IF Drive_Master.fSetPosition >= 100 THEN
            bExe_Camin1:=TRUE;
            iProcess:=4;
        END_IF

    4:// If finish one cycle, stop Drive_Master
        IF MC_CamIn_1.EndOfProfile = TRUE THEN
            bExe_Halt1:=TRUE;
        END_IF
END_CASE
```

Info.

- Note that if a program is not designed to make a smooth transition between cam profiles, a sudden change occurs in position or velocity.

6.2.11 Sample Example: Adjust Phase of Cam Control Using MC_Phasing

This is an example of a program designed to adjust the phase of ongoing cam control using MC_Phasing.

■ Program example

To adjust the phase of ongoing cam control using MC_Phasing, preparing an axis aside from the cam control is necessary. In this example, the program adjusts the phase of Drive_Slave, which is the slave axis under cam control, by using Virtual_Master and Virtual_Slave for MC_Phasing.

- MC_Phasing
  Master axis: Virtual_Master
  Slave axis: Virtual_Slave
- MC_CamIn
  Master axis: Virtual_Master
  Slave axis: Drive_Slave

The following program, after the completion of one cycle of cam control, sifts the phase forward by 10 and continues the cam control. After that, the program shifts the phase backward by 55.5.

- Declaration section

```
// Variables
    iProcess         : INT:=0;
    bExe_Movevel     : BOOL;
    bExe_Tabsel0     : BOOL;
    bExe_Camin0      : BOOL;
```

```
    bExe_Phasing0    : BOOL;
    bExe_Phasing1    : BOOL;

// FB instances
    MC_MoveVelocity_1      : MC_MoveVelocity;
    MC_MoveVelocity_2      : MC_MoveVelocity;
    MC_CamTableSelect_0    : MC_CamTableSelect;
    MC_CamIn_0             : MC_CamIn;
    MC_Phasing_0           : MC_Phasing;
    MC_Phasing_1           : MC_Phasing;
```

- Implementation section

```
// CAM FBs Settings
MC_MoveVelocity_1(
                    Axis:=Virtual_Master,
                    Execute:=bExe_Movevel,
                    Velocity:=50,
                    Acceleration:=1000,
                    Deceleration:=1000,
);
MC_MoveVelocity_2(
                    Axis:=Virtual_Slave,
                    Execute:=bExe_Movevel,
                    Velocity:=50,
                    Acceleration:=1000,
                    Deceleration:=1000,
);

MC_Phasing_0(
            Master:=Virtual_Master,
            Slave:=Virtual_Slave,
            Execute:=bExe_Phasing0,
            PhaseShift:=10,
            Velocity:=1,
            Acceleration:=1000,
            Deceleration:=1000,
            Jerk:=10000,
);
MC_Phasing_1(
            Master:=Virtual_Master,
            Slave:=Virtual_Slave,
            Execute:=bExe_Phasing1,
            PhaseShift:=-55.5,
            Velocity:=20,
            Acceleration:=10,
            Deceleration:=10,
            Jerk:=10000,
);

MC_CamTableSelect_0(
                    Master:=Virtual_Slave,
                    Slave:=Drive_Slave,
                    CamTable:=Cam2,
                    Execute:=bExe_Tabsel0,
                    Periodic:=TRUE,
```

```
                    MasterAbsolute:=TRUE,
                    SlaveAbsolute:=FALSE,
);

MC_CamIn_0(
            Master:=Virtual_Slave,
            Slave:=Drive_Slave,
            Execute:=bExe_Camin0,
            MasterOffset:=0,
            SlaveOffset:=0,
            MasterScaling:=1,
            SlaveScaling:=1,
            StartMode:=relative,
            CamTableID:=MC_CamTableSelect_0.CamTableID,
            VelocityDiff:=5,
            Acceleration:=1000,
            Deceleration:=1000,
            Jerk:=10000,
            TappetHysteresis:=0,
);

CASE iProcess OF
    1:// Load CamTable
        bExe_Tabsel0:=TRUE;
        IF MC_CamTableSelect_0.Done = TRUE THEN
            iProcess:=2;
        END_IF

    2:// Start Virtual Axis moving and Cam
        bExe_Movevel:=TRUE;
        bExe_Camin0:=TRUE;
        iProcess:=3;

    3:// Start Phasing
        IF MC_CamIn_0.EndOfProfile = TRUE THEN
            bExe_Phasing0:=TRUE;
            IF MC_Phasing_0.Done = TRUE THEN
                iProcess:=4;
            END_IF
        END_IF

    4:// re-Phasing
        IF Virtual_Slave.fSetPosition > 200 THEN
            bExe_Phasing1:=TRUE;
            IF MC_Phasing_1.Done = TRUE THEN
                iProcess:=5;
            END_IF
        END_IF
END_CASE
```

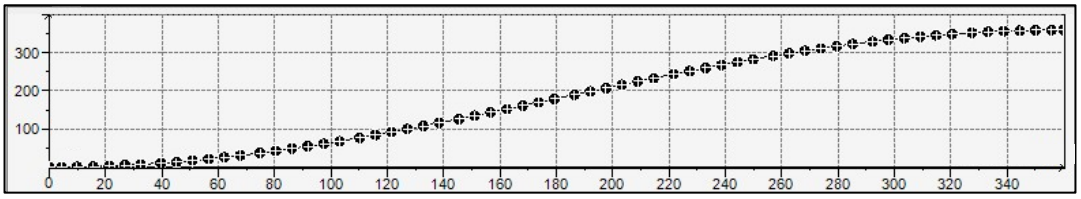
Info.

- If MC_Phasing is executed on master and slave axes (axes specified for the master axis and slave axis of MC_CamIn) under cam control, MC_CamIn is aborted. Thus, MC_Phasing can make a phase correction on cam synchronous control by using two virtual axes as described above.

6.2.12 Sample Example: Create MC_CAM_REF by POU

[Program example 1] One-dimensional table format with byType = 1

This program creates MC_CAM_REF that has a curve as shown below. Execute this program by UserTask.



- Global variable declaration section

```
MC_CAM_REF_1 : MC_CAM_REF:=(xPartofLM:=TRUE);
```

- Declaration section

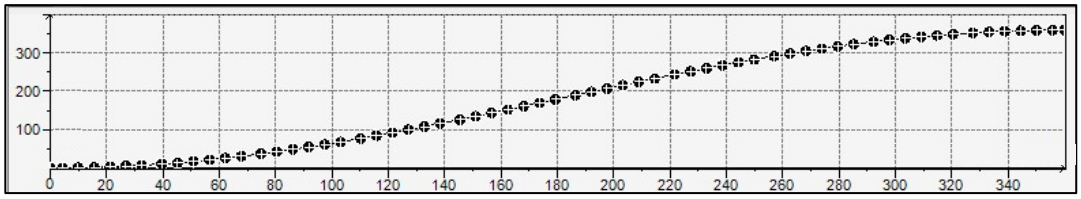
```
bMake : BOOL; i : INT; nTable1 : SMC_CAMTable_LREAL_128_1;
```

- Implementation section

```
IF bMake = FALSE THEN // Set Editor and Table parameters nTable1.fEditorMas
terMin:=0.0; nTable1.fEditorMasterMax:=360.0; nTable1.fEditorSlaveMin:=0.0;
 nTable1.fEditorSlaveMax:=360.0; nTable1.fTableMasterMin:=0.0; nTable1.fTab
leMasterMax:=360.0; nTable1.fTableSlaveMin:=0.0; nTable1.fTableSlaveMax:=36
0.0; // Set Cam Table values FOR i:=0 TO 127 DO nTable1.Table[i]:=360.0 * 0
.5 * (1 - COS(SMC_PI * i / 127)); END_FOR MC_CAM_REF_1.byType:=1; MC_CAM_RE
F_1.byVarType:=6; MC_CAM_REF_1.xStart:=0.0; MC_CAM_REF_1.xEnd:=360.0; MC_CA
M_REF_1.nElements:=128; MC_CAM_REF_1.nTappets:=0; MC_CAM_REF_1.pce:=ADR(nTa
ble1); MC_CAM_REF_1.pt:=Math_Globals.NULL; MC_CAM_REF_1.strCAMName:='Camexa
mple01'; MC_CAM_REF_1.byInterpolationQuality:=1; MC_CAM_REF_1.byCompatibili
tyMode:=0; bMake:=TRUE; END_IF
```

[Program example 2] Two-dimensional table format with byType = 2

This program creates MC_CAM_REF that has a tappet table, declares a cam table structure, and has a curve as shown below. Execute this program by UserTask.



- Global variable declaration section

```
MC_CAM_REF_2 : MC_CAM_REF:=(xPartofLM:=TRUE);
```

- Structure (TYPE CAMTable_REAL_500_2)

```
TYPE CAMTable_REAL_500_2 :
STRUCT
    Table                                   : ARRAY[0..499] OF ARRAY[0..1] OF
 REAL;
    fEditorMasterMin, fEditorMasterMax    : REAL;
    fEditorSlaveMin, fEditorSlaveMax      : REAL;
    fTableMasterMin, fTableMasterMax      : REAL;
    fTableSlaveMin, fTableSlaveMax        : REAL;
END_STRUCT
END_TYPE
```

- Declaration section

```
    bMake       : BOOL;
    i           : INT;
    nTable2     : CAMTable_REAL_500_2;
    nTappet2    : ARRAY[0..3] OF SMC_CAMTappet:=[
                    (ctt:=0, cta:=0, iGroupID:=1, x:=90),
                    (ctt:=0, cta:=1, iGroupID:=1, x:=130),
                    (ctt:=0, cta:=2, iGroupID:=1, x:=150),
                    (ctt:=0, cta:=1, iGroupID:=1, x:=300)
                ];
```

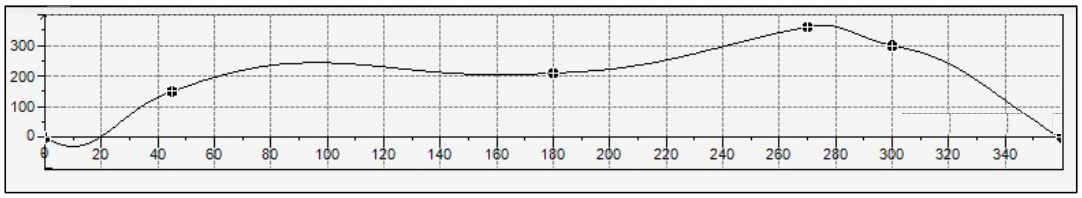
- Implementation section

```
IF bMake = FALSE THEN
// Set Editor and Table parameters
    nTable2.fEditorMasterMin:=0.0;
    nTable2.fEditorMasterMax:=360.0;
    nTable2.fEditorSlaveMin:=0.0;
    nTable2.fEditorSlaveMax:=360.0;
    nTable2.fTableMasterMin:=0.0;
    nTable2.fTableMasterMax:=360.0;
    nTable2.fTableSlaveMin:=0.0;
    nTable2.fTableSlaveMax:=360.0;

// Set Cam Table values
    FOR i:=0 TO 499 DO
        nTable2.Table[i][0]:=360.0 * i / 499;
        nTable2.Table[i][1]:=TO_REAL(360.0 * 0.5 * (1 - COS(SMC_PI * nTable
2.Table[i][0] / 360.0)));
    END_FOR
```

```
    MC_CAM_REF_2.byType:=2;
    MC_CAM_REF_2.byVarType:=5;
    MC_CAM_REF_2.xStart:=0.0;
    MC_CAM_REF_2.xEnd:=360.0;
    MC_CAM_REF_2.nElements:=512;
    MC_CAM_REF_2.nTappets:=4;
    MC_CAM_REF_2.pce:=ADR(nTable2);
    MC_CAM_REF_2.pt:=ADR(nTappet2);
    MC_CAM_REF_2.strCAMName:='Camexample02';
    MC_CAM_REF_2.byInterpolationQuality:=3;
    MC_CAM_REF_2.byCompatibilityMode:=0;

    bMake:=TRUE;
END_IF
```

■ [Program example 3] XYVA table format with byType = 3

This program creates MC_CAM_REF that has a tappet table and has a cam table as shown below. Execute this program by UserTask.



- Global variable declaration section

```
MC_CAM_REF_3 : MC_CAM_REF:=(xPartofLM:=TRUE);
```

- Declaration section

```
bMake : BOOL; i : INT; nTable3 : ARRAY[0..5] OF SMC_CAMXYVA; nTappet3 : ARR
AY[0..32767] OF SMC_CAMTappet; a_x1 : ARRAY [0..7] OF LREAL:=[10, 50, 70, 1
00, 110, 180, 200, 200]; a_ID1 : ARRAY [0..7] OF INT:=[1, 1, 1, 1, 1, 1, 1,
 1]; a_ctt1 : ARRAY [0..7] OF INT:=[0, 0, 0, 0, 1, 0, 0, 2]; a_cta1 : ARRAY
 [0..7] OF INT:=[0, 1, 2, 2, 3, 3, 0, 3]; a_Delay1 : ARRAY [0..7] OF DWORD:
=[0, 0, 0, 0, 0, 500000, 100000, 100000]; a_Duration1 : ARRAY [0..7] OF DWO
RD:=[0, 0, 0, 0, 500000, 200000, 100000, 100000];
```

- Implementation section

```
IF bMake = FALSE THEN // Set Cam Table XY values nTable3[0].dX:=0.0; nTable
3[0].dY:=0.0; nTable3[1].dX:=45.0; nTable3[1].dY:=150.0; nTable3[2].dX:=180
.0; nTable3[2].dY:=210.0; nTable3[3].dX:=270.0; nTable3[3].dY:=360.0; nTabl
e3[4].dX:=300.0; nTable3[4].dY:=300.0; nTable3[5].dX:=360.0; nTable3[5].dY:
=0.0; // Calculate dV nTable3[0].dV:=TO_LREAL((nTable3[5].dY - nTable3[4].d
Y) / (nTable3[5].dX - nTable3[4].dX)); FOR i:=1 TO 4 DO nTable3[i].dV:=TO_L
REAL((nTable3[i].dY - nTable3[i - 1].dY) / (nTable3[i].dX - nTable3[i - 1].
dX)); END_FOR nTable3[5].dV:=nTable3[0].dV; // Calculate dA nTable3[0].dA:=
TO_LREAL((nTable3[5].dV - nTable3[4].dV) / (nTable3[5].dX - nTable3[4].dX))
; FOR i:=1 TO 4 DO nTable3[i].dA:=TO_LREAL((nTable3[i].dV - nTable3[i - 1].
dV) / (nTable3[i].dX - nTable3[i - 1].dX)); END_FOR nTable3[5].dA:=nTable3[
0].dA; // Set Tappet Table FOR i:=0 TO 7 DO nTappet3[i].x:=a_x1[i]; nTappet
3[i].iGroupID:=a_ID1[i]; nTappet3[i].ctt:=a_ctt1[i]; nTappet3[i].cta:=a_cta
```

```
1[i]; nTappet3[i].dwDelay:=a_Delay1[i]; nTappet3[i].dwDuration:=a_Duration
1[i]; END_FOR MC_CAM_REF_3.byType:=3; MC_CAM_REF_3.byVarType:=0; MC_CAM_REF
_3.xStart:=0.0; MC_CAM_REF_3.xEnd:=360.0; MC_CAM_REF_3.nElements:=6; MC_CAM
_REF_3.nTappets:=i; MC_CAM_REF_3.pce:=ADR(nTable3); MC_CAM_REF_3.pt:=ADR(nT
appet3); MC_CAM_REF_3.strCAMName:='Camexample03'; MC_CAM_REF_3.byCompatibil
ityMode:=0; bMake:=TRUE; END_IF
```

Info.

- Programs that create MC_CAM_REF must be executed by UserTask.
- MC_CAM_REF does not output information about errors. Thus, in order to check whether a proper cam profile is created, monitor it to detect any error at the time of execution of MC_CamTableSelect.
- In the XYVA format, dV can be calculated by finding the first derivative of dY with respect to dX (dY/dX) and dA can be calculated by finding the second derivative of dY with respect to dX (d^2Y/dX^2) or the first derivative of dV with respect to dX (dV/dX).

■ fEditorMasterMin, fTableMasterMin, fEditorMasterMax, and fTableMasterMax

In the Type2 format, the slave operation start and end positions relative to the master axis are shifted and scaled by the variables fEditorMasterMin, fTableMasterMin, fEditorMasterMax, and fTableMasterMax of the cam table strcture and the start and end positions (xStart, xEnd) of MC_CAM_REF. If the slave motion range after the conversion is smaller than the range between the start and end positions of MC_CAM_REF, the system does not operate properly.

In the Type1 format, these parameters exist, but such scaling and shift do not work because the slave positions are equally arranged on the cam table.

The formulas are given as described below.

Scaling factor = (fEdirtorMasterMax - fEditorMasterMin) / (fTableMasterMax - fTableMasterMin)

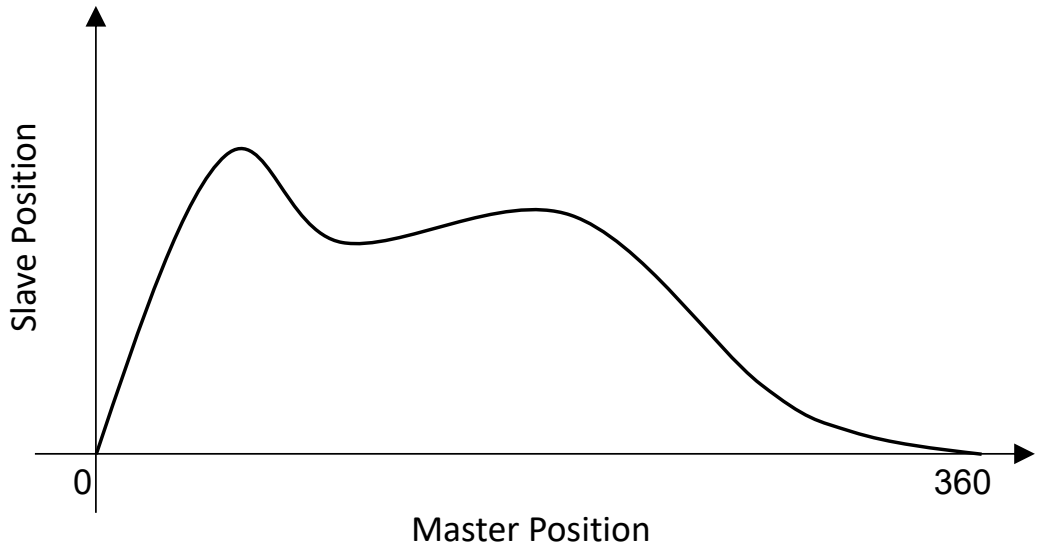
Slave operation start position = fEditorMasterMin - fTableMasterMin × Scaling factor

Slave operation end position = Slave operation start position + (xEnd - xStart) × Scaling factor

With MC_CAM_REF in the Type2 format, when xStart = 0 and xEnd = 360, the following operations are thought.

- Example 1: fEditorMasterMin = 0, fEditorMasterMax = 360, fTableMasterMin = 0, and fTableMasterMax = 360

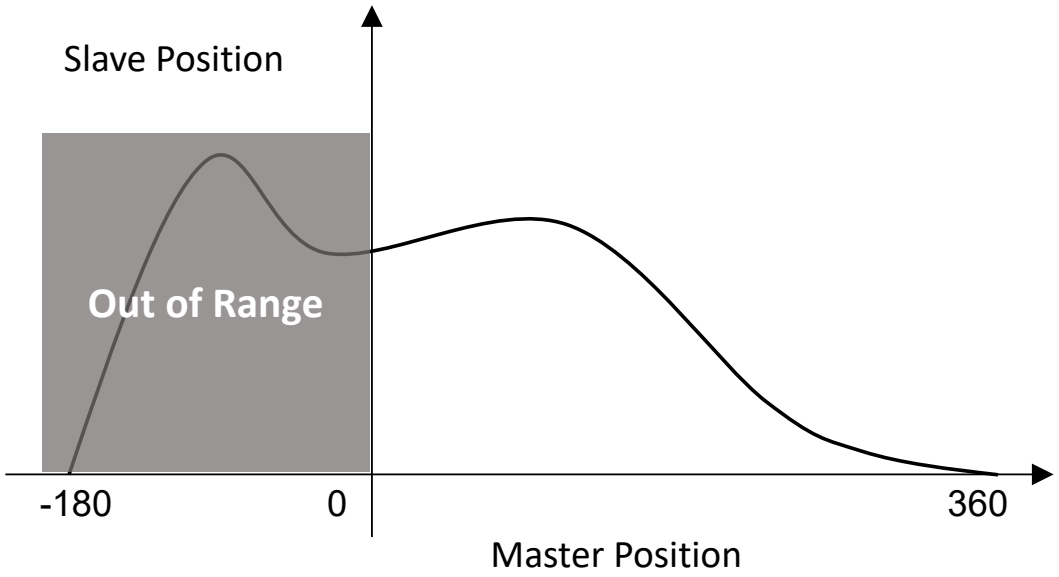
According to the formulas, Scaling factor = 1, Slave operation start position = 0, and Slave operation end position = 360. Since the motion range of MC_CAM_REF is set such that xStart = 0 and xEnd = 360, the entire created cam table represents a motion range.



- Example 2: fEditorMasterMin = -180, fEditorMasterMax = 360, fTableMasterMin = 0, and fTableMasterMax = 360

According to the formulas, Scaling factor = 1.5, Slave operation start position = -180, and Slave operation end position = 360. Hence, with the created cam table, the range of the master position is set from -180 to 360.

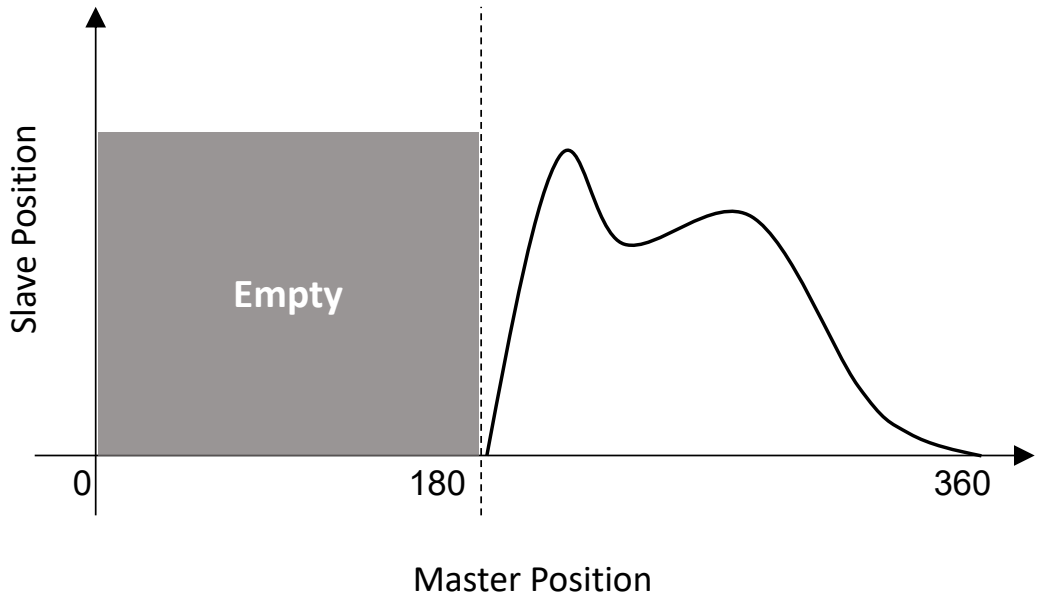
However, the motion range of MC_CAM_REF is set such that xStart = 0 and xEnd = 360 and thus, in the created cam table, only the range of the master position from 0 to 360 is used for the cam to operate.



- Example 3: fEditorMasterMin = 0, fEditorMasterMax = 360, fTableMasterMin = -360, and fTableMasterMax = 360

According to the formulas, Scaling factor = 0.5, Slave operation start position = 180, and Slave operation end position = 360. Hence, with the created cam table, the range of the master position is set from 180 to 360.

However, the motion range of MC_CAM_REF is set such that xStart = 0 and xEnd = 360 and thus, when the master position is in the range from 0 to 180, the cam table does not exist. As a result, for this period, the cam operates with the parameter of the slave axis being indefinite.



■ fEditorSlaveMin, fTableSlaveMin, fEditorSlaveMax, and fTableSlaveMax

The home position, maximum value, and minimum value of the slave operation are shifted and scaled by fEditorSlaveMin, fTableSlaveMin, fEditorSlaveMax, fTableSlaveMax of the variables of the cam table structure in either of the Type1 format and Type2 format.

The formulas are given as described below.

Scaling factor = (fEdirtorSlaveMax - fEditorSlaveMin) / (fTableSlaveMax - fTableSlaveMin)

Slave operation home position = fEditorSlaveMin - fTableSlaveMin × Scaling factor

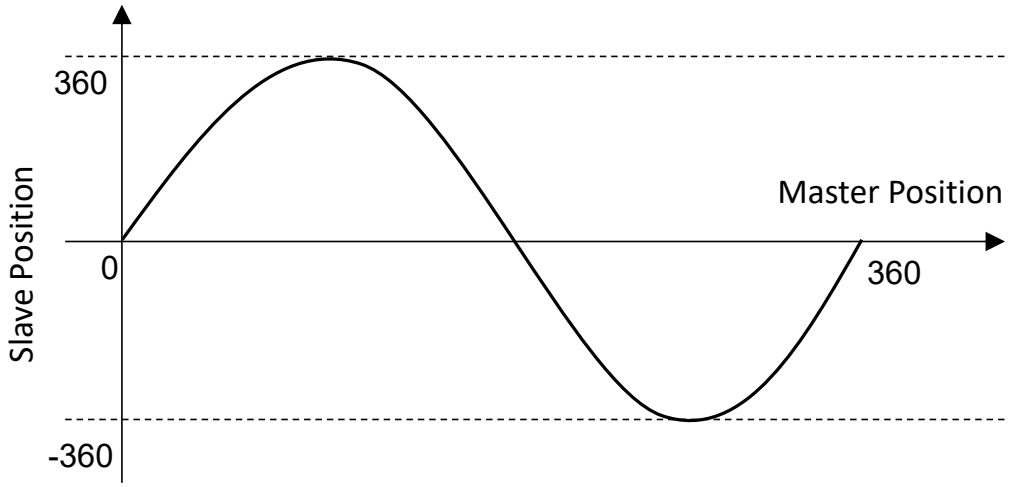
Maximum slave operation position = Slave operation home position + Maximum slave position × Scaling factor

Minimum slave operation position = Slave operation home position + Minimum slave position × Scaling factor

With MC_CAM_REF in the Type1 format, when xStart = 0 and xEnd = 360 and when the slave axis value ranges from -360 to 360, the following operations are thought.

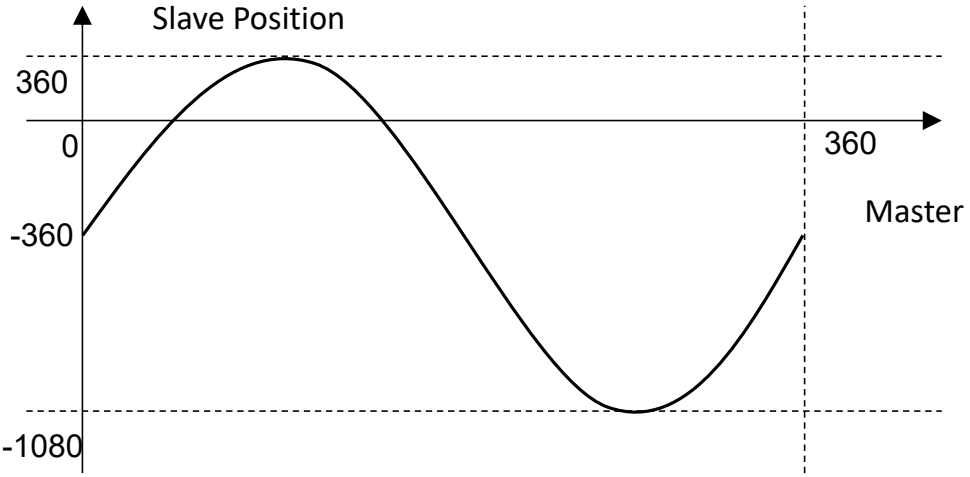
- Example 1: fEditorSlaveMin = 0, fEditorSlaveMax = 360, fTableSlaveMin = 0, and fTableSlaveMax = 360

According to the formulas, Scaling factor = 1, Slave operation home position = 0, Maximum slave operation position = 360, and Minimum slave operation position = -360.



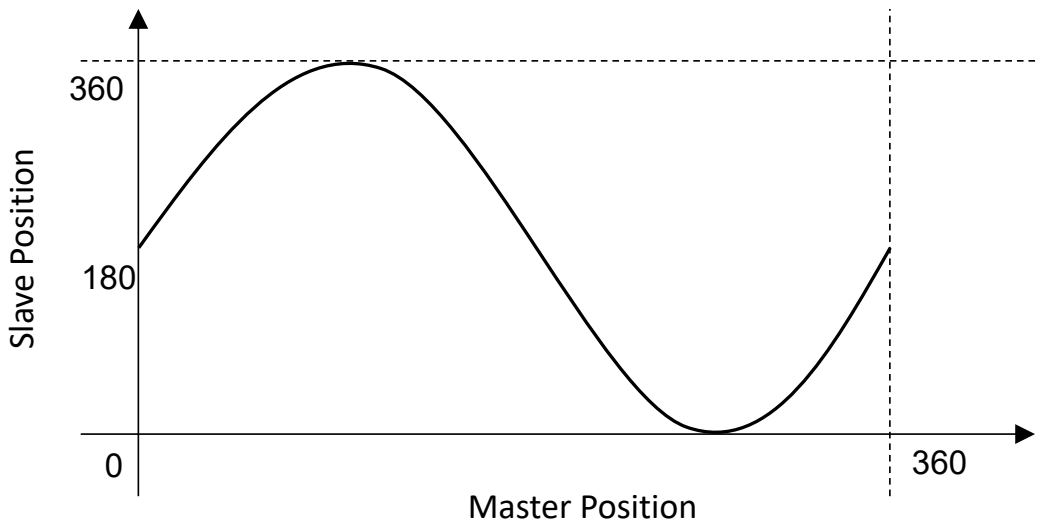
- Example 2 : fEditorSlaveMin = 0, fEditorSlaveMax = 360, fTableSlaveMin = 180, and fTableSlaveMax = 360

According to the formulas, Scaling factor = 2, Slave operation home position = -360, Maximum slave operation position = 360, and Minimum slave operation position = -1080. The slave operation range has doubled from that in the original cam table.



- Example 3 : fEditorSlaveMin = 180, fEditorSlaveMax = 360, fTableSlaveMin = 0, and fTableSlaveMax = 360

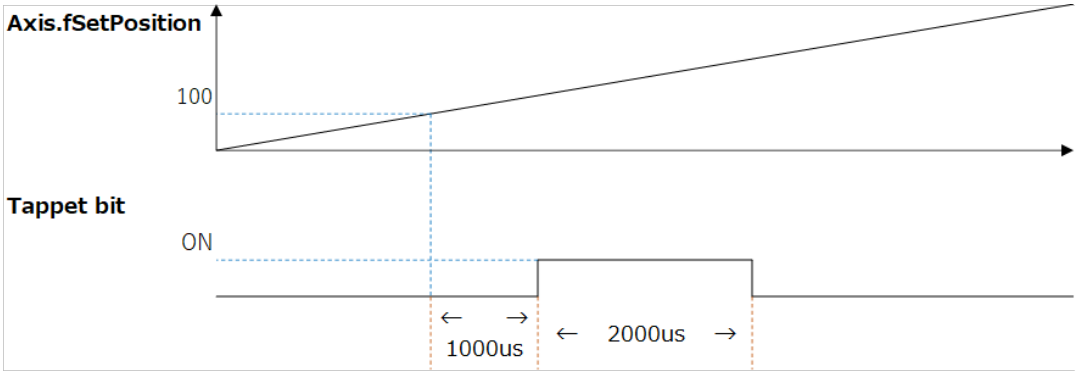
According to the formulas, Scaling factor = 0.5, Slave operation home position = 180, Maximum slave operation position = 360, and Minimum slave operation position = 0. The slave operation range has decreased by half from that in the original cam table.



■ Tappet table

● Example for use of dwDelay and dwDuration

The tappet structure is set such that x = 100, dwDelay = 1000, and dwDuration = 2000. With these settings, the tappet switches to ON after the elapse of 1000 µs following the time when the master passes through a place of 100. The tappet remains ON for a duration of 2000 µs.



● Example in which three tappets with a shared track ID are specified for a shared point

If two or more tappets with the shared track ID are specified for the shared point, processing is performed starting from the last data element of the tappet table array.

```
nTappet : ARRAY [0..2] OF SMC_CAMTappet;
a_x : ARRAY [0..2] OF LREAL:=[100, 100, 100];
a_ID : ARRAY [0..2] OF INT:=[1, 1, 1];
a_ctt : ARRAY [0..2] OF INT:=[0, 0, 0];
a_cta : ARRAY [0..2] OF INT:=[0, 2, 1];
```

With these settings configured, when the master passes through the point x = 100 with the tappet switch being OFF, the tappets are processed in the order of a_cta[2] → a_cta[1] → a_cta[0]. In other words, the tappets are processed such that the current switch OFF tappet changes as follows: OFF(a_cta[2] = 1) → ON(a_cta[1] = 2) → ON(a_cta[0] = 0). Consequently, the tappet switch changes to ON.

6.2.13 Sample Example: Create MC_CAM_REF Using Recipe Function

This is an example of a program for creating a cam profile (MC_CAM_REF) from cam data created by spreadsheet software through the use of a recipe definition.

■ [Program example 1] Reading cam array data stored in a recipe file to create MC_CAM_REF

It is assumed that 256 master positions (Rp_Cam_X.txtrecipe) and slave positions (Rp_Cam_Y.txtrecipe) are written in the recipe file to be read. Execute this program by UserTask.

- Recipe definition settings

  CAM_X (PersistentVars.Cam_dX[0] to [255] are registered)

  CAM_Y (PersistentVars.Cam_dY[0] to [255] are registered)

- PersistentVars variable declaration section

```
    Cam_dX    : ARRAY [0..255] OF LREAL;
    Cam_dY    : ARRAY [0..255] OF LREAL;
```

- Global variable declaration section

```
    MC_CAM_REF_4    : MC_CAM_REF:=(xPartofLM:=TRUE);
```

- Declaration section

```
    iProcess                  : INT:=0;
    bMake                     : BOOL;
    i                         : INT;
    nTable4                   : SMC_CAMTable_LREAL_256_2;
    RecipeManCommands_0       : RecipeManCommands;
    LoadFromAndWriteRecipe    : DWORD;
    CreateRecipe              : DWORD;
```

- Implementation section

```
IF bMake = FALSE THEN
// Set Editor and Table parameters
    nTable4.fEditorMasterMin:=0.0;
    nTable4.fEditorMasterMax:=360.0;
    nTable4.fEditorSlaveMin:=0.0;
    nTable4.fEditorSlaveMax:=360.0;
    nTable4.fTableMasterMin:=0.0;
    nTable4.fTableMasterMax:=360.0;
    nTable4.fTableSlaveMin:=0.0;
    nTable4.fTableSlaveMax:=360.0;

    // Set Cam Table values
    FOR i:=0 TO 255 DO
        nTable4.Table[i][0]:=PersistentVars.Cam_dX[i];
        nTable4.Table[i][1]:=PersistentVars.Cam_dY[i];
    END_FOR

    MC_CAM_REF_4.byType:=2;
    MC_CAM_REF_4.byVarType:=6;
```

```
    MC_CAM_REF_4.xStart:=0.0;
    MC_CAM_REF_4.xEnd:=360.0;
    MC_CAM_REF_4.nElements:=256;
    MC_CAM_REF_4.nTappets:=0;
    MC_CAM_REF_4.pce:=ADR(nTable4);
    MC_CAM_REF_4.pt:=Math_Globals.NULL;
    MC_CAM_REF_4.strCAMName:='Camexample04';
    MC_CAM_REF_4.byInterpolationQuality:=1;
    MC_CAM_REF_4.byCompatibilityMode:=0;

    bMake:=TRUE;
END_IF

CASE iProcess OF
    0:// Load Master Axis data
        CreateRecipe:=RecipeManCommands_0.CreateRecipe(RecipeDefinitionName
:='CAM_X', RecipeName:='Rp_Cam_X');
        LoadFromAndWriteRecipe:=RecipeManCommands_0.LoadFromAndWriteRecipe(
RecipeDefinitionName:='CAM_X', RecipeName:='Rp_Cam_X', FileName:='Rp_Cam_X'
);
        iProcess:=1;

    1:// Load Slave Axis data
        CreateRecipe:=RecipeManCommands_0.CreateRecipe(RecipeDefinitionName
:='CAM_Y', RecipeName:='Rp_Cam_Y');
        LoadFromAndWriteRecipe:=RecipeManCommands_0.LoadFromAndWriteRecipe(
RecipeDefinitionName:='CAM_Y', RecipeName:='Rp_Cam_Y', FileName:='Rp_Cam_Y'
);
        iProcess:=2;

    2:// Set values to Cam Table
        bMake:=FALSE;
        iProcess:=3;
END_CASE
```

■ [Program example 2] Save a MC_CAM_REF cam table created by a program in a recipe file

This program saves a MC_CAM_REF cam table created by a program in a recipe file. Execute this program by UserTask.

- Recipe definition settings

  CAM_X (PersistentVars.Cam_dX[0] to [255] are registered)
  CAM_Y (PersistentVars.Cam_dY[0] to [255] are registered)
  CAM_V (PersistentVars.Cam_dV[0] to [255] are registered)
  CAM_A (PersistentVars.Cam_dA[0] to [255] are registered)

- PersistentVars variable declaration section

```
    Cam_dX    : ARRAY [0..255] OF LREAL;
    Cam_dY    : ARRAY [0..255] OF LREAL;
    Cam_dV    : ARRAY [0..255] OF LREAL;
    Cam_dA    : ARRAY [0..255] OF LREAL;
```

- Global variable declaration section

```
    MC_CAM_REF_5    : MC_CAM_REF:=(xPartofLM:=TRUE);
```

- Declaration section

```
    iProcess             : INT:=0;
    bMake                : BOOL;
    i                    : INT;
    nTable5              : ARRAY [0..63] OF SMC_CAMXYVA;
    RecipeManCommands_0  : RecipeManCommands;
    CreateRecipe         : DWORD;
    ReadAndSaveRecipe    : DWORD;
```

- Implementation section

```
IF bMake = FALSE THEN
// Set Cam Table values
    FOR i:=0 TO 63 DO
        nTable5[i].dX:=360.0 * i / 63;
        nTable5[i].dY:=TO_REAL(360.0 * 0.5 * (1 - COS(SMC_PI * nTable5[i].d
X / 360.0)));
    END_FOR

    // Calculate dV
    nTable5[0].dV:=0.0;
    FOR i:=1 TO 62 DO
        nTable5[i].dV:=TO_LREAL((nTable5[i].dY - nTable5[i - 1].dY) / (nTab
le5[i].dX - nTable5[i - 1].dX));
    END_FOR
    nTable5[63].dV:=nTable5[0].dV;

    // Calculate dA
    nTable5[0].dA:=TO_LREAL((nTable5[63].dV - nTable5[62].dV) / (nTable5[63
].dX - nTable5[62].dX));
    FOR i:=1 TO 62 DO
        nTable5[i].dA:=TO_LREAL((nTable5[i].dV - nTable5[i - 1].dV) / (nTab
le5[i].dX - nTable5[i - 1].dX));
    END_FOR
    nTable5[63].dA:=nTable5[0].dA;

    MC_CAM_REF_5.byType:=3;
    MC_CAM_REF_5.byVarType:=0;
    MC_CAM_REF_5.xStart:=0.0;
    MC_CAM_REF_5.xEnd:=360.0;
    MC_CAM_REF_5.nElements:=64;
    MC_CAM_REF_5.nTappets:=0;
    MC_CAM_REF_5.pce:=ADR(nTable5);
    MC_CAM_REF_5.pt:=Math_Globals.NULL;
    MC_CAM_REF_5.strCAMName:='Camexample05';
    MC_CAM_REF_5.byInterpolationQuality:=1;
    MC_CAM_REF_5.byCompatibilityMode:=0;

    bMake:=TRUE;
END_IF

CASE iProcess OF
    0:// Make MC_CAM_REF
```

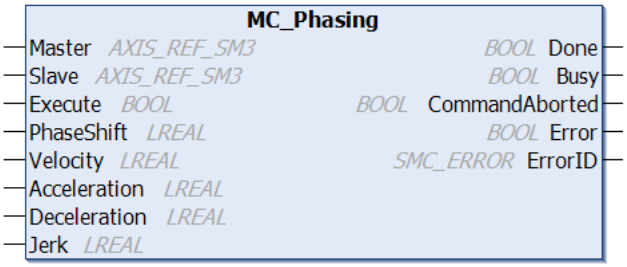
```
        IF bMake = TRUE THEN
            FOR i:= 0 TO 63 DO
                PersistentVars.Cam_dX[i]:=nTable5[i].dX;
                PersistentVars.Cam_dY[i]:=nTable5[i].dY;
                PersistentVars.Cam_dV[i]:=nTable5[i].dV;
                PersistentVars.Cam_dA[i]:=nTable5[i].dA;
            END_FOR
            iProcess:=1;
        END_IF

    1:// Save dX Values
        CreateRecipe:=RecipeManCommands_0.CreateRecipe(RecipeDefinitionName
:='CAM_X', RecipeName:='Rp_MCCam_X');
        ReadAndSaveRecipe:=RecipeManCommands_0.ReadAndSaveRecipe(RecipeDefi
nitionName:='CAM_X', RecipeName:='Rp_MCCam_X');
        iProcess:=2;

    2:// Save dY Values
        CreateRecipe:=RecipeManCommands_0.CreateRecipe(RecipeDefinitionName
:='CAM_Y', RecipeName:='Rp_MCCam_Y');
        ReadAndSaveRecipe:=RecipeManCommands_0.ReadAndSaveRecipe(RecipeDefi
nitionName:='CAM_Y', RecipeName:='Rp_MCCam_Y');
        iProcess:=3;

    3:// Save dV Values
        CreateRecipe:=RecipeManCommands_0.CreateRecipe(RecipeDefinitionName
:='CAM_V', RecipeName:='Rp_MCCam_V');
        ReadAndSaveRecipe:=RecipeManCommands_0.ReadAndSaveRecipe(RecipeDefi
nitionName:='CAM_V', RecipeName:='Rp_MCCam_V');
        iProcess:=4;

    4:// Save dA Values
        CreateRecipe:=RecipeManCommands_0.CreateRecipe(RecipeDefinitionName
:='CAM_A', RecipeName:='Rp_MCCam_A');
        ReadAndSaveRecipe:=RecipeManCommands_0.ReadAndSaveRecipe(RecipeDefi
nitionName:='CAM_A', RecipeName:='Rp_MCCam_A');
        iProcess:=5;
END_CASE
```

6.3 Phase Correction

6.3.1 MC_Phasing (Master Axis Phase Correction)

This is a function block (FB) that performs phase correction between the master axis and slave axis. Phase synchronous operation can be performed by making phase correction for the master axis.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Initial	Description
Input / output	Master	AXIS_REF_SM3	-	Specifies the master axis.
	Slave	AXIS_REF_SM3	-	Specifies the slave axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	PhaseShift	LREAL	0	Specify the phase difference between the master axis and the slave axis.
	Velocity	LREAL	0	Specify the maximum speed for phase correction (u/s), which is the relative speed from the master axis.
	Acceleration	LREAL	0	Specify the maximum acceleration for phase correction (u/s^2)
	Deceleration	LREAL	0	Specify the maximum deceleration for phase correction (u/s^2)
	Jerk	LREAL	0	Specify the maximum jerk for phase correction (u/s^3)
Output	Done	BOOL	FALSE	TRUE: Phase correction is completed.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ Detail of function

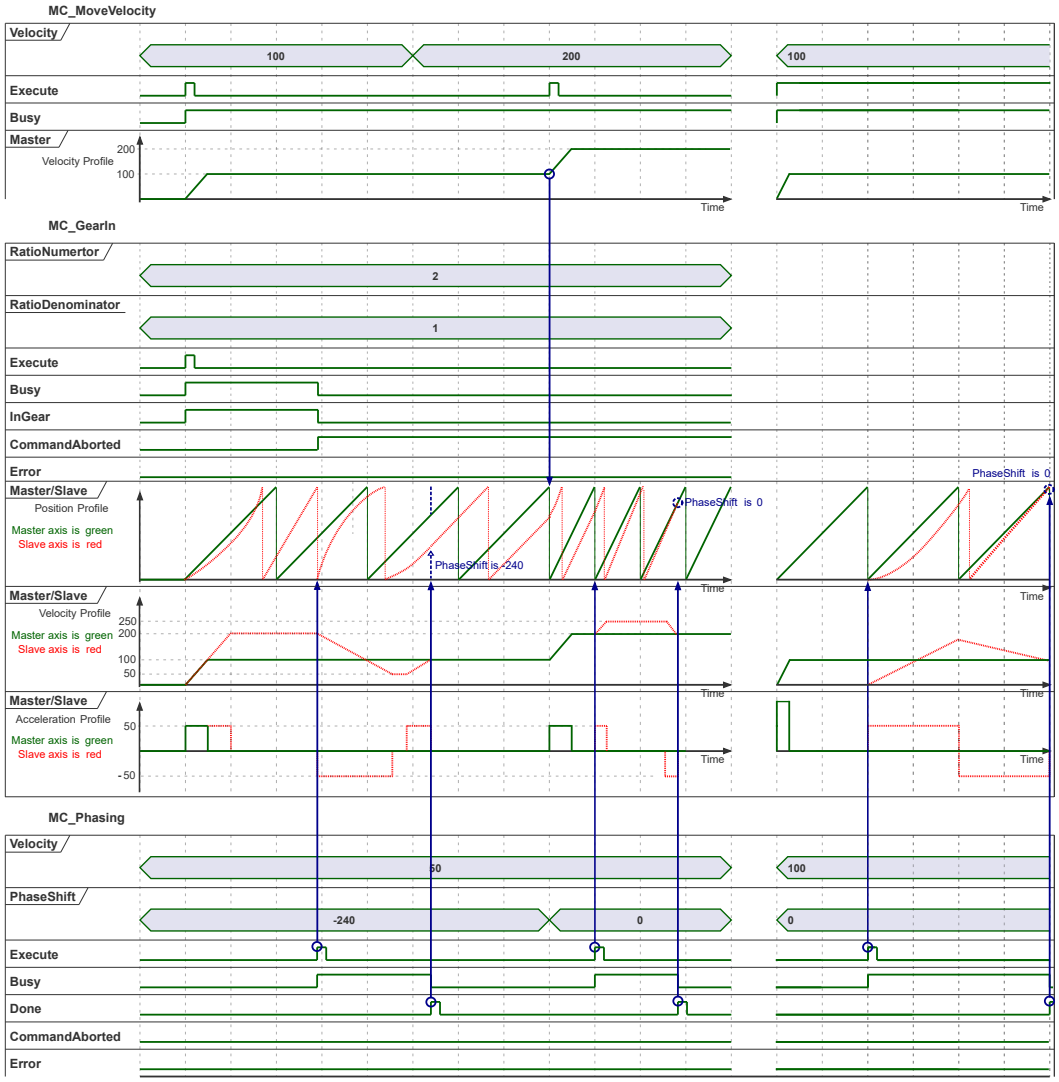
- Function Description
  - If this Function Block (FB) is executed during gear synchronous operation, the ongoing gear synchronous operation will be interrupted and will switch to the behavior of this FB.

    If this FB is executed while not in synchronization control, it will perform synchronization control and phase correction.
  - At the rising edge of this FB's Execute, if there is a difference in speed between the main and follower axes, the speed of the slave axis will be synchronized to the speed of the master axis in accordance with this FB's Acceleration and Deceleration.

    After that, the slave axis begins phase correction following the Acceleration and Deceleration parameters.
  - PhaseShift indicates the phase difference between the main axis and the slave axis after the completion of phase correction.Furthermore, the position of the slave axis after the completion of phase correction will be **Master axis position - PhaseShift**.
  - During phase correction, the speed of the slave axis changes to **Master axis speed + Velocity** or **Master axis speed - Velocity** to ensure that phase correction is completed in the shortest possible time. Additionally, if **Master axis speed - Velocity** becomes a negative value, the slave axis may move in the negative direction
  - Phase correction is completed when the phase difference between the master axis and the slave axis reaches PhaseShift, and the slave axis has matched the speed of the master axis.
  - After the completion of phase correction, if the speed of the master axis changes, the slave axis will follow the speed of the master axis.
  - This FB must be called every control cycle during phase correction.
  - If you want to reset the phase difference to 0 after phase correction is complete, re-execute with PhaseShift = 0.
- Operation Start
  - At the rising edge of Execute, the slave axis starts phase correction following Velocity, Acceleration, Deceleration, and Jerk.
- Re-execution
  - Set Execute to FALSE. Then, reset the input values. By triggering Execute, it will run with the new input values.
- Operation Interruption
  - If a Function Block (FB) that controls the axis is called for the slave axis during operation, the operation of this FB will be interrupted.

■ Timing chart

- If Execute is triggered while gear synchronous operation is in progress, CommandAborted will occur in MC_GearIn, and the Busy of this FB will become TRUE.
- If the phase correction is completed, Busy will turn to FALSE, and Done will become TRUE. The TRUE state of Done will become FALSE if Execute is set to FALSE.


MC_MoveVelocity
Velocity
100
200
100
Execute
Busy
Master
Velocity Profile
200
100
Time
Time
MC_GearIn
RatioNumertor
2
RatioDenominator
1
Execute
Busy
InGear
CommandAborted
Error
Master/Slave
Position Profile
Master axis is green
Slave axis is red
PhaseShift is 0
PhaseShift is -240
PhaseShift is 0
Time
Time
Master/Slave
Velocity Profile
Master axis is green
Slave axis is red
250
200
100
50
Time
Time
Master/Slave
Acceleration Profile
Master axis is green
Slave axis is red
50
-50
Time
Time
MC_Phasing
Velocity
50
100
PhaseShift
-240
0
0
Execute
Busy
Done
CommandAborted
Error

7 Motion Control Function Blocks (Interpolation Control)

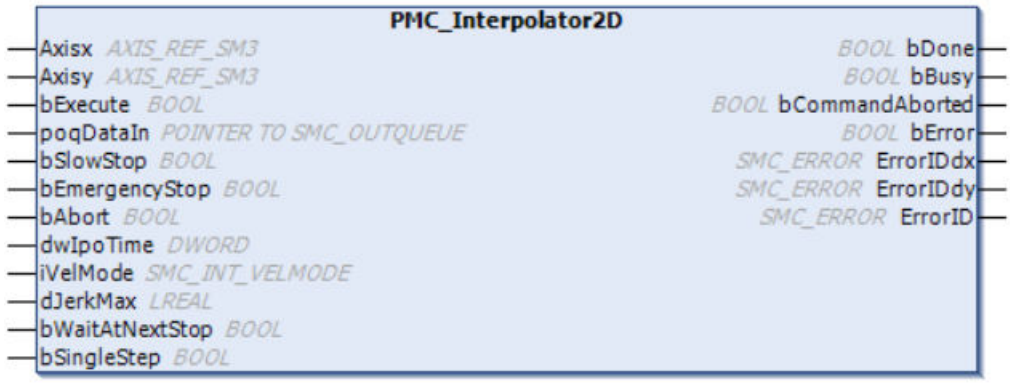
This section describes function blocks used to perform interpolation control using the CNC program.

7.1 Interpolation Control

7.1.1 PMC_Interpolator2D (2-axis Interpolation Control)

This function block (FB) performs 2-axis interpolation control according to the specified CNC table.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Default value	Description
I/O	Axisx	AXIS_REF_SM3	-	Specifies the x-axis.
	Axisy	AXIS_REF_SM3	-	Specifies the y-axis.
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	poqDataIn	POINTER TO SMC_OUTQUEUE	-	Specifies a pointer to the CNC table.
	bSlowStop	BOOL	FALSE	TRUE: A pause is executed. Deceleration stop is executed according to the velocity profile (iVelMode). FALSE: The pause is canceled.
	bEmergencyStop	BOOL	FALSE	TRUE: An emergency stop is executed. FALSE: The emergency stop is canceled.
	bAbort	BOOL	FALSE	TRUE: Execution of the FB is stopped.
	dwIpoTime	DWORD	0	MotionTask interval (µsec)
	iVelMode	SMC_INT_VELMODE	TRAPEZOID	Specifies a velocity profile.
	dJerkMax	LREAL	LREAL	Specifies the maximum value of jerk.

Scope	Name	Type	Default value	Description
				This parameter must be specified when QUADRATIC is selected for the velocity profile (iVelMode).
	bWaitAtNextStop(Note 1)	BOOL	BOOL	TRUE: A pause is executed in the table where the velocity between paths becomes zero. The conditions that cause the velocity between paths to become zero are set in bSingleStep or dAngleMode. FALSE: The pause is canceled.
	bSingleStep(Note 1)	BOOL	BOOL	TRUE: All connections between paths are established through deceleration stop.
Output	bCommandAborted	BOOL	FALSE	TRUE: An interruption is caused by another FB.
	bBusy	BOOL	-	TRUE: Execution of the FB is not completed.
	bDone	BOOL	FALSE	TRUE: Output is completed.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorIDdx	SMC_ERROR	SMC_NO_ERROR	Error ID output during x-axis movement processing
	ErrorIDdy	SMC_ERROR	SMC_NO_ERROR	Error ID output during y-axis movement processing
	ErrorID	SMC_ERROR	SMC_NO_ERROR	Error ID output during interpolation control operation

(Note 1) When both bWaitAtNextStop and bSingleStep are set to TRUE, they may not work properly, so please do not use them together.

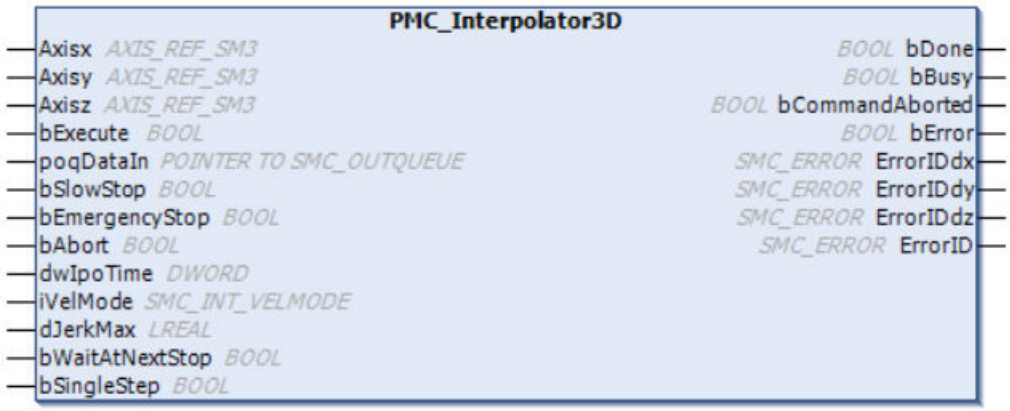
SMC_INT_VELMODE (Enumeration type)

Name	Value	Description
TRAPEZOID	0	Trapezoid
SIGMOID	1	Sin2
SIGMOID_LIMIT	2	Sin2 (limit)
QUADRATIC	3	Quadratic
QUADRATIC_SMOOTH	4	Quadratic (smooth)

7.1.2 PMC_Interpolator3D (3-axis Interpolation Control)

This function block (FB) performs 3-axis interpolation control according to the specified CNC table.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
I/O	Axisx	AXIS_REF_SM3	-	Specifies the x-axis.
	Axisy	AXIS_REF_SM3	-	Specifies the y-axis.
	Axisz	AXIS_REF_SM3	-	Specifies the z-axis.
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	poqDataIn	POINTER TO SMC_OUTQUEUE	-	Specifies a pointer to the CNC table.
	bSlowStop	BOOL	FALSE	TRUE: A pause is executed. Deceleration stop is executed according to the velocity profile (iVelMode). FALSE: The pause is canceled.
	bEmergencyStop	BOOL	FALSE	TRUE: An emergency stop is executed. FALSE: The emergency stop is canceled.
	bAbort	BOOL	FALSE	TRUE: Execution of the FB is stopped.
	dwIpoTime	DWORD	0	MotionTask interval (µsec)
	iVelMode	SMC_INT_VELMODE	TRAPEZOID	Specifies a velocity profile.
	dJerkMax	LREAL	LREAL	Specifies the maximum value of jerk.

Scope	Name	Type	Default value	Description
				This parameter must be specified when QUADRATIC is selected for the velocity profile (iVelMode).
	bWaitAtNextStop(Note 1)	BOOL	BOOL	TRUE: A pause is executed in the table where the velocity between paths becomes zero. The conditions that cause the velocity between paths to become zero are set in bSingleStep or dAngleMode. FALSE: The pause is canceled.
	bSingleStep(Note 1)	BOOL	BOOL	TRUE: All connections between paths are established through deceleration stop.
Output	bCommandAborted	BOOL	FALSE	TRUE: An interruption is caused by another FB.
	bBusy	BOOL	-	TRUE: Execution of the FB is not completed.
	bDone	BOOL	FALSE	TRUE: Output is completed.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorIDdx	X-axis error ID	SMC_NO_ERROR	Error ID output during x-axis movement processing
	ErrorIDdy	Y-axis error ID	SMC_NO_ERROR	Error ID output during y-axis movement processing
	ErrorIDdz	Z-axis error ID	SMC_NO_ERROR	Error ID output during z-axis movement processing
	ErrorID	SMC_ERROR	SMC_NO_ERROR	Error ID output during interpolation control operation

(Note 1) When both bWaitAtNextStop and bSingleStep are set to TRUE, they may not work properly, so please do not use them together.

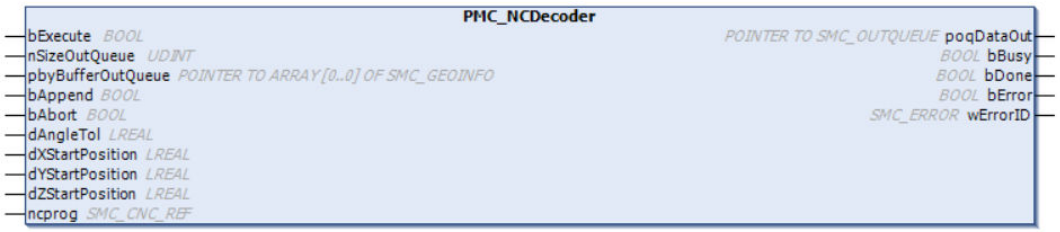
SMC_INT_VELMODE (Enumeration type)

Name	Value	Description
TRAPEZOID	0	Trapezoid
SIGMOID	1	Sin2
SIGMOID_LIMIT	2	Sin2 (limit)
QUADRATIC	3	Quadratic
QUADRATIC_SMOOTH	4	Quadratic (smooth)

7.1.3 PMC_NCDecoder (CNC Table Conversion)

This function block (FB) decodes the specified SMC_CNC_REF value to SMC_OUTQUEUE.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
I/O	ncprogIn	SMC_CNC_REF	-	Specifies the SMC_CNC_REF value to be decoded.
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	nSizeOutQueue	UDINT	-	Specifies a buffer size. We recommend that a buffer be created and the sizeof operator be specified as shown in the following example. ExampleBuf: ARRAY [0..50] OF SMC_GEOINFO; nSizeOutQueue:=sizeof(ExampleBuf)
	pbyBufferOutQueue	POINTER TO ARRAY [0..0] OF SMC_GEOINFO	-	Specifies the memory space for SMC_OUTUEUE. We recommend that array SMC_GEOINFO be defined and an address be specified as shown in the following example. ExampleBuf: ARRAY [0..50] OF SMC_GEOINFO; (Buffer that can store 50 path elements) pbyBufferOutQueue:=ADR(ExampleBuf)
	dXstartPosition	LREAL	0	Specifies the position of the x-axis at the start of movement(Note 1).
	dYstartPosition	LREAL	0	Specifies the position of the y-axis at the start of movement(Note 1).
	dZstartPosition	LREAL	0	Specifies the position of the z-axis at the start of movement(Note 1).
	bAppend	BOOL	FALSE	TRUE: Decoded data of ncprogIn is appended to the end of poqDataOut without resetting the poqDataOut

Scope	Name	Type	Default value	Description
				data within the FB at the rising edge as specified by bExecute.
	bAbort	BOOL	FALSE	TRUE: Execution of the FB is stopped.
Output	poqDataOut	POINTER TO SMC_OUTQUEUE	-	Pointer to SMC_OUTQUEUE which manages decoded SMC_GEOINFO objects
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Done	BOOL	FALSE	TRUE: Output is completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	Error ID output

(Note 1) We recommend that fSetPosition be entered. If the entered value and the actual position differ, there is a risk that the axis may move suddenly.

(MEMO)

8 Motion Control Function Blocks (CNC Control)

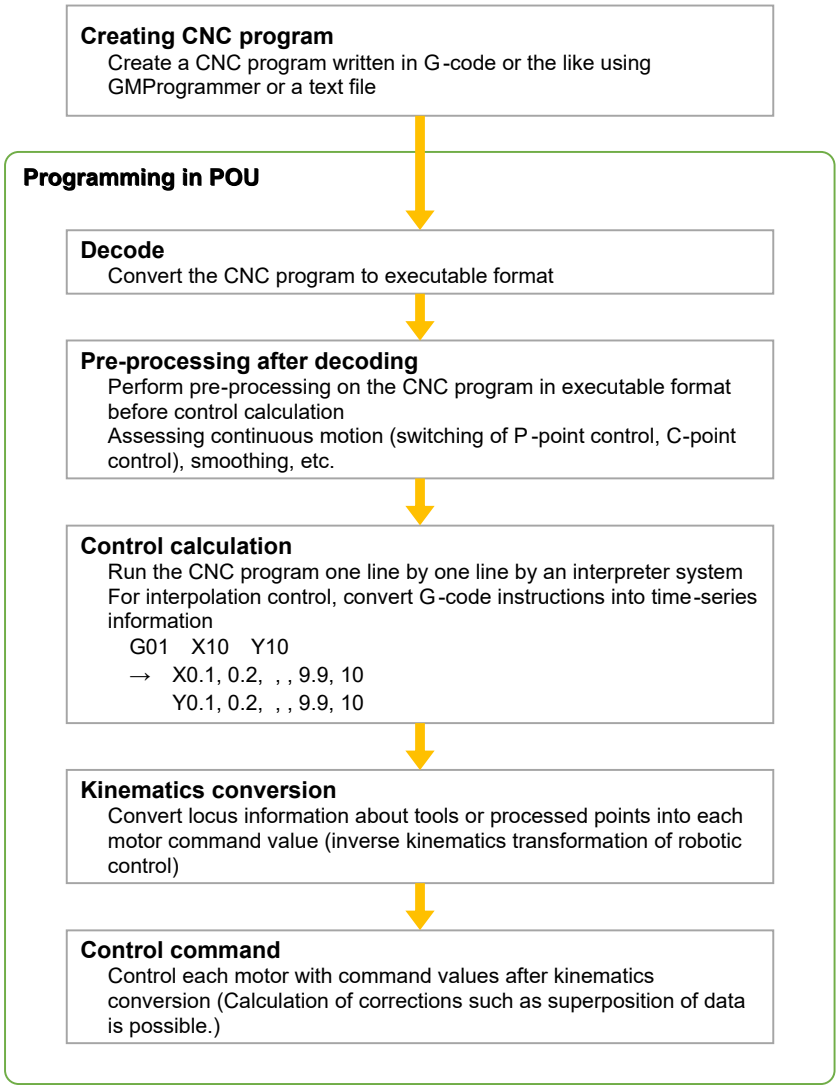
This section describes function blocks used to perform control using the CNC program.

8.1 Overview of CNC Control and How to Use It

This section describes an overview of CNC control and how to use CNC control with GM Programmer.

You must execute the following series of processes to perform CNC control with the GM1 controller.

- Create a CNC program written in G-code using the CNC editor or other tools.
- Decode the CNC program to executable format.
- Compute command data from the decoded CNC program at every cycle of motor control to control the motor.

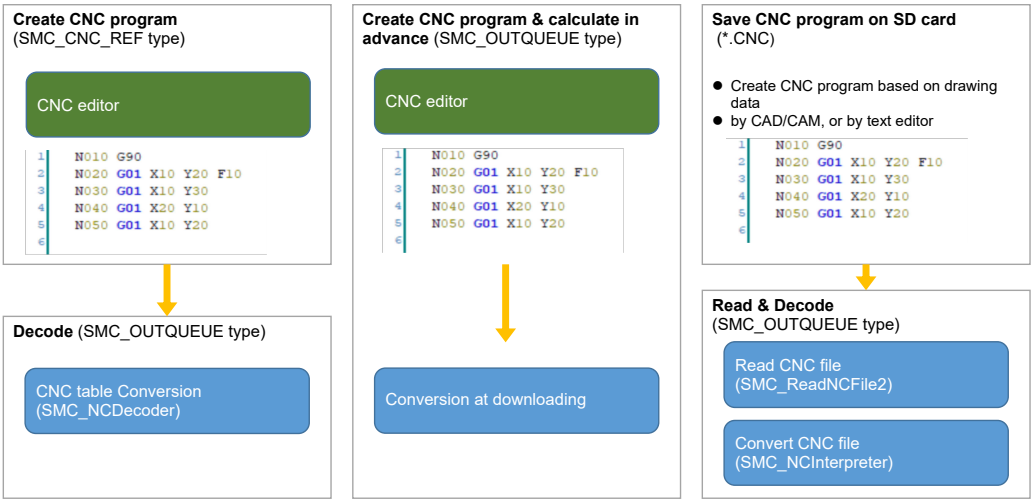


Procedure

1. Create a CNC program and decode it

A CNC program (SMC_OUTQUEUE type) written in executable format is necessary to program CNC control using GM Programmer. The CNC program in executable format can be created by any of the following three methods.

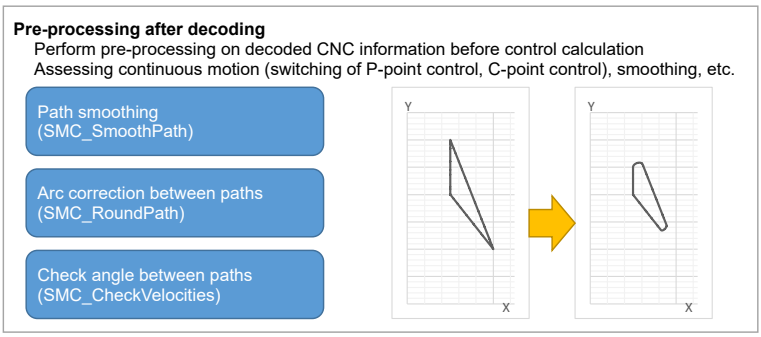
- After creating a CNC program (SMC_CNC_REF type) using the CNC editor, decoding the CNC program to that of the SMC_OUTQUEUE type by"8.2.1 SMC_NCDecoder (CNC Program Conversion)".
- Creating a CNC program (SMC_OUTQUEUE type) using the CNC editor. The created CNC program is converted into executable format when it is downloaded.
- Reading a CNC program file (*.CNC), which is created by a text editor or CAD/CAM, via a SD card using"8.2.2 SMC_ReadNCFile2 (Read CNC File)"and decoding the CNC file to that of the SMC_OUTQUEUE type by"8.2.3 SMC_NCInterpreter (Convert CNC File)".



For details on how to create a CNC program using the CNC editor, refer to the*GM1 Series User's Manual (Operation Edition)*.

2. Pre-processing after decoding

As pre-processing including assessing continuous motion and smoothing, process the CNC program in executable format.



Perform smoothing by"8.3.2 SMC_SmoothPath (path smoothing)"or perform arc correction between decoded paths by"8.3.3 SMC_RoundPath (Arc correction between paths)".

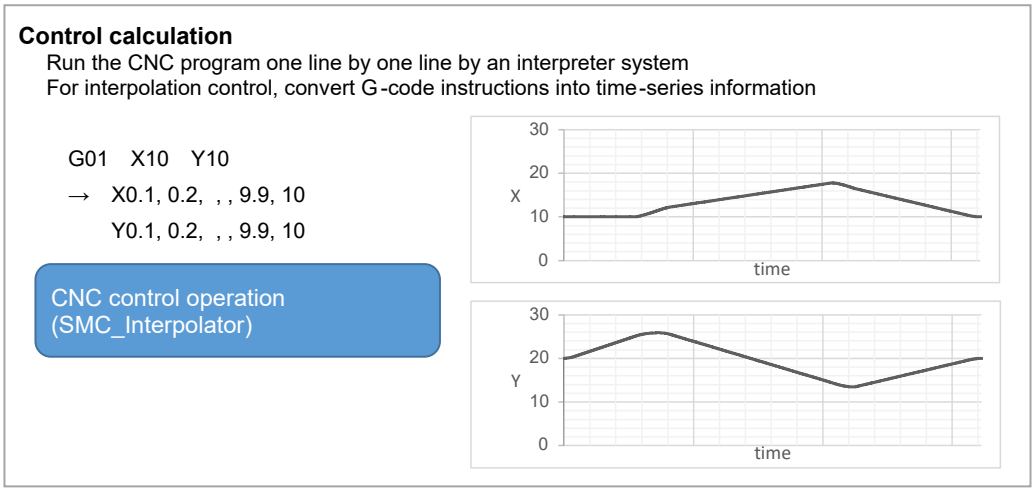
Using "8.3.1 SMC_CheckVelocities (Check Angle between Paths)", check an angle formed by decoded paths. If the angle is larger than or equal to a set threshold, axial motion instantaneously stops between the paths (C-point motion).

Using "8.3.4 SMC_ToolRadiusCorr (Tool Radius Correction for Path)", apply tool radius correction to decoded paths.

3. Control calculation

Run the CNC program in executable format, i.e., the CNC information, one line by one line by an interpreter system through"8.4.1 SMC_Interpolator (CNC Control Operation)".

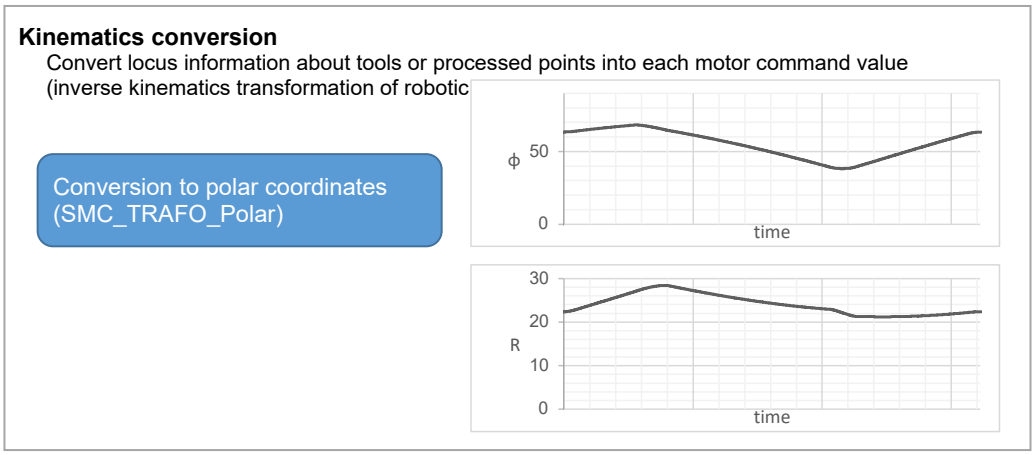
For interpolation control, calculate time-series information for every control cycle from G-code instructions.



4. Kinematics conversion

Convert locus information about tools or processed points into each motor command value.

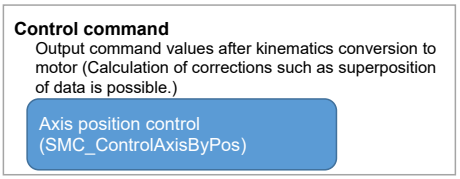
By"8.5.3 SMC_TRAFO_Polar (Conversion from Two-dimensional (X, Y) Coordinates to Polar Coordinates)", locus information can be converted to data on the polar coordinate system.



Function blocks used in kinematics conversion include "8.5.5 SMC_TRAFO_Bipod_Arm (Bipod robot hand XY coordinates → conversion of each axis position)" and "8.5.2 SMC_ToolLengthCorr (Tool Length Correction)".

5. Control command

Using"8.5.1 SMC_ControlAxisByPos (Axis Position Control)", control each motor with command values after kinematics conversion.

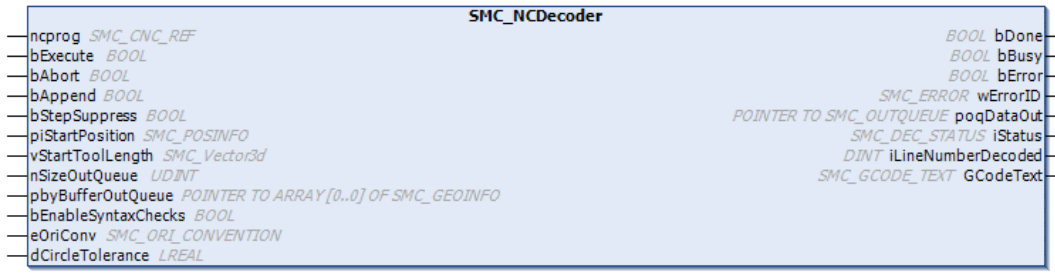


8.2 CNC Data Decoding

8.2.1 SMC_NCDecoder (CNC Program Conversion)

This function block (FB) decodes a specified CNC program (SMC_CNC_REF) to data (SMC_OUTQUEUE) used to manage an array list of CNC executable format data (SMC_GEOINFO). In each cycle, one line of the program is decoded. Execute the function block by MotionTask.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
I/O	ncprog	SMC_CNC_REF	-	Specifies the CNC program (SMC_CNC_REF) to be decoded.
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	bAbort	BOOL	FALSE	TRUE: Execution of the FB is stopped.
	bAppend	BOOL	FALSE	TRUE: At the rising edge as specified by bExecute, the poqDataOut data within the FB is not reset.(Note 1) Decoded data of ncprogIn is appended to the end of poqDataOut.
	bStepSuppress	BOOL	FALSE	TRUE: Lines of the CNC program starting with "/" are ignored.
	piStartPosition	SMC_POSINFO	-	Start position of the path(Note 2)
	vStartToolLength	SMC_Vector3d	dX=0, dY=0, dZ=0	Start tool length
	nSizeOutQueue	UDINT	0	Specifies the size of the data buffer to which the list of SMC_GEOINFO structure objects will be written. This buffer must be able to hold at least five SMC_GEOINFO objects. If the size of the buffer is not satisfactory, no error occurs and the FB is not executed.

Scope	Name	Type	Default value	Description
				The buffer size may be predefined, but may be changed only during a reset. [Declaration example] ExampleBuf: ARRAY[0..49] OF SMC_GEOINFO; (An array of five or more elements is required) [Writing example]nSizeOutQueue:=SIZEOF(ExampleBuf);
	pbyBufferOutQueue	POINTER TO ARRAY [0..0] OF SMC_GEOINFO	-	Specifies the address of the memory space for SMC_OUTUEUE.(Note 4) We recommend that array SMC_GEOINFO be defined and an address be specified as shown in the following example. [Declaration example] ExampleBuf: ARRAY[0..49] OF SMC_GEOINFO; (An array of five or more elements is required) [Writing example] pbyBufferOutQueue:=ADR(Exampl eBuf);
	bEnableSyntaxChecks	BOOL	FALSE	TRUE: Detects invalid G-code and wrong CNC program, and stops with the occurrence of an error.
	eOriConv	SMC_ORI_CON VENTION	ADDAXES	A definition for the order in which Euler angles specified by coordinate system conversion G54/G55/G56 rotate
	dCircleTolerance	LREAL	0	Tolerance to determine whether the definition of a circle makes sense(Note 3)
Output	bDone	BOOL	FALSE	TRUE: Decode output is completed.
	bBusy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	wErrorID	SMC_ERROR	SMC_NO_ERR OR	Error ID output
	poqDataOut	POINTER TO SMC_OUTQ UEUE	-	Pointer to the CNC program which manages decoded SMC_GEOINFO objects Specify this output for the poqDataIn input of"8.3 Pre-processing after decoding".
	iStatus	SMC_DEC_STA TUS	WAIT_PROG	Current status
	iLineNumberDecoded	DINT	0	The 0-based line number of the CNC file that is completely decoded

Scope	Name	Type	Default value	Description
	GCodeText	SMC_GCODE_TEXT	-	The G-Code text that is currently decoded

(Note 1) To decode the CNC program including circular interpolation written by center-point approach with bAppend = TRUE, explicitly specify G98/G99 in the CNC program.

(Note 2) We recommend that fSetPosition be entered. If the entered value and the current value greatly differ, there is a risk that the axis may move suddenly.

(Note 3) This is determined according to the following rules.

Definition via target position and radius: If the distance between start- and end-positions is greater than 2 * the radius + MAX (dCircleTolerance, 1e-06), the circle will be converted into a line.

Definition via target- and center-position: Let x be the maximum of the distance between start- and center-positions and the distance between target- and center-positions. If those distances differ by more than MAX (dCircleTolerance, 0.1 * x), the circle will be converted into a line.

(Note 4) Do not set SMC_GEOINFO objects that are specified in other function blocks.

Info.

- To use the bAppend function, set bAppend to TRUE after decoding of the first CNC program is completed, and then decode the second and subsequent CNC programs.
- While decoding of a CNC program that uses G20, do not set bExecute to FALSE. Loop processing will not be executed correctly.
- For the tool length (vStartToolLength), refer to "8.5.2 SMC_ToolLengthCorr (Tool Length Correction)".

SMC_POSINFO (Structure)

This is a structure that describes the positions of coordinate axes including additional axes for a particular position point.

Information on the path written in G code is output as position information for control at every cycle from the SMC_Interpolator.

Information about the output is written by this structure.

Name	Type	Default value	Description
iFrameNo	INT	0	Frame number (Additional information not relevant for the SoftMotion modules may be stored by the user.)
wAuxData	WORD	7	Axes to be calculated by the Interpolator: TRUE = Enabled, FALSE = Disabled bit0 = X axis, bit1 = Y axis, bit2 = Z axis, bit3 and subsequent bits are not used.
wSProfile	WORD	0	Not used
dX	LREAL	0	X-position in coordinate system
dY	LREAL	0	Y-position in coordinate system
dZ	LREAL	0	Z-position in coordinate system
dA	LREAL	0	Not used
dB	LREAL	0	Not used
dC	LREAL	0	Not used
dA1	LREAL	0	Not used
dA2	LREAL	0	Not used

Name	Type	Default value	Description
dA3	LREAL	0	Not used
dA4	LREAL	0	Not used
dA5	LREAL	0	Not used
dA6	LREAL	0	Not used

■ Example

- For 2-axis interpolation control, wAuxData = 10#3 and values are set in dX: target position X for next cycle and dY: target position Y for next cycle. The other parameters are not used.
- For 3-axis interpolation control, wAuxData = 10#7 and values are set in dX: target position X for next cycle, dY: target position Y for next cycle, and dZ: target position Z for next cycle. The other parameters are not used.

SMC_ORI_CONVENTION (Enumeration type)

Input values need to be specified when coordinate conversion (G54, G55, G56) is executed.

While parallel translation (X, Y, Z) of the coordinate system is executed with any input value, rotation of the coordinate system requires an input value that sets the derided order in rotation to be specified.

Name	Type	Value	Description
ADDAXES	INT	0	Rotation of the coordinate system is not executed (default value).
ZYZ	INT	1	The coordinate system rotates around the Z axis > The coordinate system rotates around the Y axis > The coordinate system rotates around the Z axis
ZYX	INT	2	The coordinate system rotates around the Z axis > The coordinate system rotates around the Y axis > The coordinate system rotates around the X axis
XYZ	INT	3	The coordinate system rotates around the X axis > The coordinate system rotates around the Y axis > The coordinate system rotates around the Z axis

SMC_DEC_STATUS (Enumeration type)

Name	Type	Value	Description
WAIT_PROG	INT	0	Waiting program
READ_WORD	INT	1	Program decoding in progress
PROG_READ	INT	2	Program decoding completed

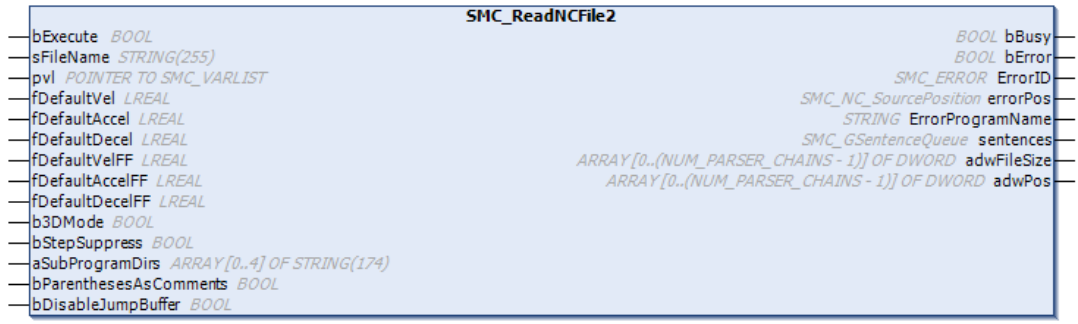
SMC_GCODE_TEXT (Structure)

Name	Type	Default value	Description
str	STRING(80)	''	Outputs the G-Code text that is currently decoded
iLineNumber	DINT	0	Line number
bNewLine	BOOL	FALSE	TRUE: A new line has been decoded.
bClearList	BOOL	FALSE	TRUE: When the NCDecoder has been started from new, a buffer that may store the last lines needs to be emptied.

8.2.2 SMC_ReadNCFile2 (Read CNC File)

This function block reads a CNC file in an SD card and outputs data (SMC_GSentenceQueue) read from the CNC file. Execute the function block by UserTask.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	sFileName	STRING(255)	''	Specifies the file name with an absolute path. (Note 1)
	pvl	POINTER TO SMC_VARLIST	-	Specifies the address of an SMC_VARLIST object. (Note 2) Setting of variables written in the CNC file
	fDefaultVel	LREAL	0	Default velocity(u/s)(Note 3)
	fDefaultAccel	LREAL	0	Default acceleration(u/s^2)(Note 3)
	fDefaultDecel	LREAL	0	Default deceleration(u/s^2)(Note 3)
	fDefaultVelFF	LREAL	0	Default velocity (u/s) for fast forward (G0)(Note 3)
	fDefaultAccelFF	LREAL	0	Default acceleration (u/s^2) for fast forward (G0)(Note 3)
	fDefaultDecelFF	LREAL	0	Default deceleration (u/s^2) for fast forward (G0)(Note 3)
	b3DMode	BOOL	TRUE	TRUE: XYZ 3-axis interpolation can be used without G16 to G19 plane specification.(Note 3) If this input is FALSE, default 2D mode is enabled.
	bStepSuppress	BOOL	FALSE	TRUE: In the CNC file, lines starting with "/" will be ignored.
	aSubProgramDirs	ARRAY [0..4] OF STRING(174)	-	Directories where subprograms are stored (up to 5 directories can be specified). (Note 4)

Scope	Name	Type	Default value	Description
	bParenthesesAsComments	BOOL	TRUE	TRUE: Treats parentheses as comments. (Note 5)
	bDisableJumpBuffer	BOOL	FALSE	Do not use.
Output	bBusy	BOOL	FALSE	TRUE: FB is in progress.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.
	errorPos	SMC_NC_SourcePosition	-	Detects invalid G-code or incorrect writing and outputs the position where an error has occurred, as well as the character length.
	ErrorProgramName	STRING	''	Absolute path to the name of the CNC file where an error has occurred
	sentences	SMC_GSentenceQueue	-	Data read from the CNC file Specify this output for the input of "8.2.3 SMC_NCInterpreter (Convert CNC File)".
	adwFileSize	ARRAY [0..(NUM_PARSER_CHAINS - 1)] OF DWORD	-	Outputs the number of characters written in the CNC file. (Note 6)
	adwPos	ARRAY [0..(NUM_PARSER_CHAINS - 1)] OF DWORD	-	Output the current position of the cursor in the CNC file. (Note 6)

(Note 1) Please specify the filename within 237 characters, including the extension ('.cnc').

(Note 2) Do not set it to 0(NULL) if variables are written in the CNC file.

(Note 3) If velocity or acceleration (F, E, FF, EF) is not written, or plane specification (G17, G18, G19) is not written in the CNC file, the corresponding setting is used.

This setting does not apply to called subprograms.

(Note 4) To specify the root directory, specify '.' or './'.

(Note 5) Parentheses "(,)" used to group expressions and for subprogram calls are also treated as comments.

Regardless of this setting, it is recommended that curly braces "{,}" be used to group expressions and for subprogram calls.

(Note 6) 'RN' is handled as two characters, and '$R', '$N', and a space are each handled as one character.

Info.

- You cannot use full size characters and the following symbols in file and directory names: [\], [/], [:], [*], [?], [”], [<], [>], [|].
- This function block's instance declaration uses a significant amount of memory. Limit instance declarations to a maximum of two, and if you need more than two, reusing instances is recommended.
- The directories specified for the aSubProgramDirs argument are used to search for the CNC file for a subprogram to be called. The CNC file is searched for in the specified directories, starting with directory aSubProgramDirs[0], in order. The CNC file that matches first is used.
- The subroutine to be read should have a filename that is within 237 characters, including the directory name and extension.
- The adwFileSize/adwPos outputs are output as follows.

  For adwFileSize[0]/adwPos[0], the number of characters/the reading position in the main program are output.

  For adwFileSize[1]/adwPos[1], the number of characters/the reading position written in the last subprogram called by the main program are output.

  For each of adwFileSize[2]/adwPos[2] and subsequent elements, the number of characters/the reading position written in the last subprogram called by each subprogram are output.

■ SMC_VARLIST (Structure)

This structure stores settings for variables written in the CNC file.

Name	Type	Default value	Description
wVarListID	WORD	16#BBFA	Fixed to 16#BBFA
wNumberVars	WORD	0	Specifies the number of SMC_SINGLEVAR array elements used
psvVarList	POINTER TO SMC_SINGLEVAR	-	Specifies the address of an element of the SMC_SINGLEVAR array

Info.

- If multiple global variables are written in the CNC file, specify the address of a data element of the SMC_SINGLEVAR array for the psvVarList argument.

  Starting from the array element specified for the psvVarList argument, the number of array elements equal to the number specified for the wNumberVars argument is used.

■ SMC_SINGLEVAR (Structure)

This structure defines the global variable used in the CNC file.

Name	Type	Default value	Description
strVarName	STRING	''	Specifies the name of the global variable as written in the CNC program file in capital letters. (Note 1)
pAdr	POINTER TO BYTE	-	Pointer to the program variable used with the name of the global

Name	Type	Default value	Description
			variable specified for strVarName. (Note 2)
eVarType	SMC_VARTYPE	SMC_TYPE_UNKNOWN	Specifies the type of the specified variable. (Note 3)
diValue	DINT	0	0 (Fixed)
fValue	LREAL	0	0 (Fixed)

(Note 1) The set character string and the name of the global variable written in the CNC file do not distinguish between upper- and lowercase letters.

(Note 2) If you specify 0 (NULL) for the pointer, the function does not operate properly, and thus do not specify so.

(Note 3) Set a type identical to that of the variable specified for the pAdr argument. If you set a type different from that of the variable specified for the pAdr argument, the function does not operate properly.

Info.

- The character string specified for the strVarName argument is used as the global variable, and the variable/type specified for the pAdr/eVarType arguments are specified for the global variable.
- For an SMC_VARLIST structure in which the SMC_SINGLEVAR object is set as an array, the pAdr/eVarType arguments are specified for the global variable when the character string specified for the strVarName argument matches the name of the global variable and are not specified when they do not match each other.

■ SMC_VARTYPE (Enumeration type)

Name	Type
SMC_TYPE_INT	1
SMC_TYPE_BYTE	2
SMC_TYPE_WORD	3
SMC_TYPE_DINT	4
SMC_TYPE_DWORD	5
SMC_TYPE_REAL	6
SMC_TYPE_SINT	14
SMC_TYPE_USINT	15
SMC_TYPE_UINT	16
SMC_TYPE_UDINT	17
SMC_TYPE_LREAL	22

■ SMC_NC_SourcePosition (Structure)

This structure detects invalid G-code or incorrect writing and outputs the position where an error has occurred, as well as the character length.

Name	Type	Default value	Description
diLine	DINT	-1	Outputs the line number. (Note 1)(Note 3)

Name	Type	Default value	Description
diColumn	DINT	-1	Outputs the position from the left end.(Note 2)(Note 3)
diLength	DINT	-1	Outputs the character length.(Note 3)

(Note 1) The uppermost line in the CNC file is regarded as 0th line, and 'RN' is handled as a line separator.

(Note 2) A character at the leftmost end in the CNC file is regarded as the 0th character, and '$R', '$N', and a space are each handled as one character.

(Note 3) Outputs -1 if unknown.

Example 1 A CNC file in which G-code word X is written without writing of the number ('01') of G-code ('G01')

- CNC File to be read

  N000 G X10 Y20

  N010 G01 X30 Y30

- Output result

  errorPos.diLine=0 (0th line)

  errorPos.diColumn=7 (7th character)

  errorPos.diLength=1 (1 character)

  When you write G-code, G + 'number' must be written. Since G-code word X is written without writing of any number for G-code, an error has occurred in G-code Word 'X'.

Example 2 A CNC file in which N number is written as maximum value (DWORD) + 1

- CNC File to be read

  N000 F10 E100 E-100

  N4294967296 G01 X10 Y20

- Output result

  errorPos.diLine=1 (1st line)

  errorPos.diColumn=1 (1st character)

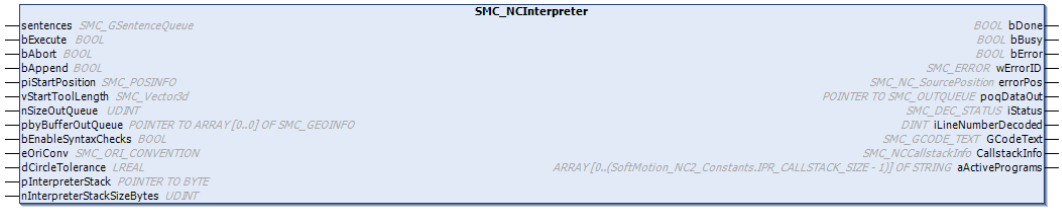
  errorPos.diLength=10 (10 characters)

  N number that is written must be a numerical value in the range from 0 to 4294967295. Since the written N-code number is 4294967296 (outside the effective range), an error has occurred at '4294967296'.

8.2.3 SMC_NCInterpreter (Convert CNC File)

This is a function block (FB) used to decode data (SMC_GSentenceQueue) read from the CNC file to data (SMC_OUTQUEUE) that is managed in the form of an array list of CNC executable format data (SMC_GEOINFO). Execute the function block by MotionTask.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	sentences	SMC_GSentenceQueue	-	CNC file data Specify the output of SMC_ReadNCFile2.
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	bAbort	BOOL	FALSE	TRUE: Execution of the FB is stopped.
	bAppend	BOOL	FALSE	TRUE: At the rising edge as specified by bExecute, the poqDataOut data within the FB is not reset.(Note 1) The sentences data is appended to the end of the poqDataOut output.
	piStartPosition	SMC_POSINFO	-	Start position of the motion in the CNC Program(Note 2)
	vStartToolLength	SMC_Vector3d	dX=0,dY=0,dZ=0	Start tool length in the CNC program
	nSizeOutQueue	UDINT	0	Specifies the size of the data buffer to which the list of SMC_GEOINFO structure objects will be written. This buffer must be able to hold at least five SMC_GEOINFO objects. If the size of the buffer is not satisfactory, no error occurs and the FB is not executed. The buffer size may be predefined, but may be changed only during a reset. [Declaration example] ExampleBuf: ARRAY[0..49] OF SMC_GEOINFO; (An array of five or more elements is required) [Example of acquiring appropriate buffer size] nSizeOutQueue:=SIZEOF(ExampleBuf);

Scope	Name	Type	Default value	Description
	pbyBufferOutQueue	POINTER TO ARRAY [0..0] OF SMC_GEOINFO	-	Specifies the address of the memory space for SMC_OUTQUEUE.(Note 5) We recommend that array SMC_GEOINFO be defined and an address be specified as shown in the following example. [Declaration example] ExampleBuf: ARRAY[0..49] OF SMC_GEOINFO; (An array of five or more elements is required) [Writing example] pbyBufferOutQueue:=ADR(ExampleBuf);
	bEnableSyntaxChecks	BOOL	TRUE	TRUE: Detects invalid G-code and incorrect CNC file, and stops with the occurrence of an error.
	eOriConv	SMC_ORI_CONVENTION	ADDAXES	A definition for the order in which Euler angles specified by coordinate system conversion G54/G55/G56 rotate Refer to"8.2.1 SMC_NCDecoder (CNC Program Conversion)"
	dCircleTolerance	LREAL	0	Tolerance to determine whether the definition of a circle makes sense(Note 3)
	pInterpreterStack	POINTER TO BYTE	-	Specifies the address of a buffer for the interpreter stack. If 0, a default buffer of size 10240 bytes is used.
	nInterpreterStackSizeBytes	UDINT	0	Specifies the size of the buffer pointed to by pInterpreterStack. At least 1024 bytes are required for the size.
Output	bDone	BOOL	FALSE	TRUE: Decode output is completed.
	bBusy	BOOL	FALSE	TRUE: FB is in progress.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	wErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.
	errorPos	SMC_NC_SourcePosition	-	Detects invalid G-code or incorrect CNC program and outputs the position where an error has occurred, as well as the character length.
	poqDataOut	POINTER TO SMC_OUTQUEUE	-	An address to the CNC program which manages decoded SMC_GEOINFO objects Specify this output for the poqDataIn input of"8.3 Pre-processing after decoding".

Scope	Name	Type	Default value	Description
	iStatus	SMC_DEC_STATUS	WAIT_PROG	Current status
	iLineNumberDecoded	DINT	0	The 0-based line number of the CNC file that is completely decoded(Note 4)
	GCodeText	SMC_GCODE_TEXT	-	The G-Code text that is currently decoded
	CallstackInfo	SMC_NCCallstackInfo	-	Not supported
	aActivePrograms	ARRAY [0..(SoftMotion_NC2_Constants.IPR_CALLSTACK_SIZE - 1)] OF STRING	-	The name of the currently decoded program (CNC file name) is output.

(Note 1) If a subprogram is decoded with the parameter set to TRUE, the function does not operate properly, and thus do not use this setting in that way.

(Note 2) We recommend that fSetPosition be entered. If the entered value and the current value greatly differ, there is a risk that the axis may move suddenly.

(Note 3) Definition via target-position and radius: If the distance between start- and end-positions is greater than 2 * the radius + MAX (dCircleTolerance, 1e-06), the circle will be converted into a line.

Definition via target- and center-position: Let x be the maximum of the distance between start- and center-positions and the distance between target- and center-positions.
If those distances differ by more than MAX (dCircleTolerance, 0.1 * x), the circle will be converted into a line.

(Note 4) If the input arguments b3DMode, fDefault** of SMC_ReadNCFile2 are used, the line number is output starting from -1.

(Note 5) Do not set SMC_GEOINFO objects that are specified in other function blocks.

Info.

- To use the bAppend function, set bAppend to TRUE after decoding of the first CNC file is completed, and then decode the second and subsequent CNC files.
- If the CNC file containing 65 or more M-codes written in a row is decoded, an error occurs. G4-elements are also counted as M-codes.

  To decode a CNC file containing 65 or more M-codes written in a row, write a G75 before the first M-code of the sequence.
- Program names are output to the elements of the aActivePrograms argument array as shown below.

  For the argument element aActivePrograms[0], the name of the currently decoded CNC file (program name) is output.

  For each of the argument element aActivePrograms[1] and subsequent elements, the name of each calling CNC file (program name) is output.
- For the tool length (vStartToolLength), refer to "8.5.2 SMC_ToolLengthCorr (Tool Length Correction)".

8.2.4 SMC_GEOINFO (CNC Executable Format Data)

This is a structure of CNC program data stored line by line in the executable format. Data such as movement types (linear interpolation, circular interpolation) as well as motion path parameters such as start position, target position, velocity, and acceleration are stored.

■ Parameter

InOut

Name	Type	Default value	Description
iObjNo	DINT	0	Identification ID
iSourceLine_No	DINT	0	The 0-based line number in the CNC program
diSentenceNo	DINT	0	CNC program N number
iMoveType	SMC_MOVTYP	-	Movement type such as linear interpolation and circular interpolation
piStartPos	SMC_POSINFO	-	Start position of the travel path
piDestPos	SMC_POSINFO	-	Target position (end position of the travel path)
dP1	LREAL	0	Described later
dP2	LREAL	0	Described later
dP3	LREAL	0	Described later
dP4	LREAL	0	Described later
dP5	LREAL	0	Described later
dP6	LREAL	0	Described later
dP7	LREAL	0	Described later
dP8	LREAL	0	Described later
dP9	LREAL	0	Described later
dP10	LREAL	0	Described later
dP11	LREAL	0	Described later
dP12	LREAL	0	Described later
dP13	LREAL	0	Described later
dP14	LREAL	0	Described later
dP15	LREAL	0	Described later
dP16	LREAL	0	Described later
dP17	LREAL	0	Described later
dP18	LREAL	0	Described later
vX	SMC_Vector3D	STRUCT(dX := 1, dY := 0, dZ := 0)	Do not use.
vY	SMC_Vector3D	STRUCT(dX := 0, dY := 1, dZ := 0)	Do not use.

Name	Type	Default value	Description
vN	SMC_Vector3D	STRUCT(dX := 0, dY := 0, dZ := 1)	Do not use.
dT1	LREAL	0	Described later
dT2	LREAL	1	Described later
dToolRadius	LREAL	0	Tool radius
dVel	LREAL	0	Target velocity [u/sec]
dVelEnd	LREAL	0	Velocity when the target position is reached [u/sec]
dVelEndStored	LREAL	-	Do not use.
dVelEndSafe	LREAL	0	Do not use.
dAccel	LREAL	100	Maximum allowable acceleration [u/sec^2]
dDecel	LREAL	100	Maximum allowable deceleration [u/sec^2]
dLength	LREAL	0	Path length
wInternMark2	WORD	0	Do not use.
byInternMark	BYTE	0	Do not use.
dwFeatureFlags	DWORD	-	TRUE: Operates in 3D mode
b3DMode	BOOL	FALSE	Stores feature bits set by G38/G39.
dHelpPos	ARRAY [0..MAX_IPOSWITCHES] OF LREAL	-	Do not use.
iHelpID	ARRAY [0..MAX_IPOSWITCHES] OF INT	-	Do not use.
adVelAddAx	ARRAY [0..7] OF LREAL	-	Do not use.
adAccAddAx	ARRAY [0..7] OF LREAL	-	Do not use.
adDecAddAx	ARRAY [0..7] OF LREAL	-	Do not use.
aAdditionalParams	ARRAY [0..(SMC_MAX_ADDITIONAL_PARAMS - 1)] OF LREAL	-	Do not use.
adToolLength	ARRAY [0..2] OF LREAL	-	Parameters for tool length compensation (set by G43 I/J/K)

For the parameters dP1 to dP18, information that varies with the iMoveType is defined.

- LIN, LINPOS (G00, G01)
  Information stored in piStartPos and piDestPos
- CLW, CCLW (G02, G03)
  The coordinates are stored in the coordinate system (vX, vY, vN).
  - dP1: X-axis coordinate of circle center

- dP2: Y-axis coordinate of circle center
- dP3: Circle radius
- dP4: Y-axis coordinate of circle

● INITPOS (M-codes)

MCOMMAND: for M-commands (iMoveType = 120), the M-code number is 120 during halt.

dT1 and dT2 are the start and end parameters for circle/spline.

● CLW（G02）
- dT1: Start angle [in degree] (0=east, 90=north, 180=west, 270=south)
- dT2: Apex angle of circle [in degree] (Example: 90=quarter of circle, 180 = semicircle)

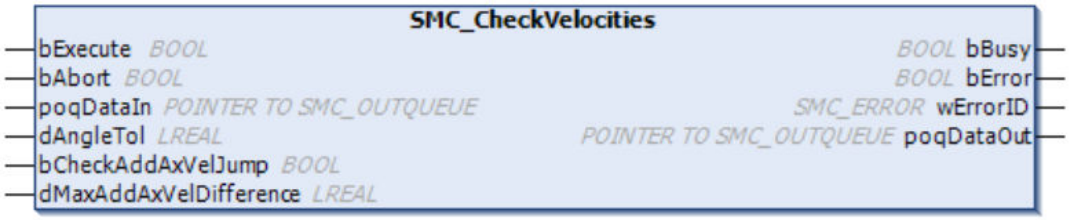
● CCLW（G03）
- dT1: Start angle [in degree] (0=east, 90=north, 180=west, 270=south)
- dT2: Negative apex angle of circle [in degree] (Example: 90=quarter of circle, 180 = semicircle)

8.3 Pre-processing after decoding

8.3.1 SMC_CheckVelocities (Check Angle between Paths)

This function block (FB) is used to check an angle between paths and perform P-point control (without deceleration stop between paths) or C-point control (with deceleration stop between paths) according to the formed angle. If the SMC_OUTQUEUE has not been created by the editor, but by the program (e.g. SMC_NCDecoder), this FB has to be called straight before each call to the SMC_Interpolator. Execute the function block by MotionTask.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Default value	Description
Input	bExecute	BOOL	FALSE	Starts execution of the FB at the rising edge.
	bAbort	BOOL	FALSE	TRUE: Execution of the FB is stopped.(Note 1) Do not use this if you want to stop movement in midstream.
	poqDataIn	POINTER TO SMC_OUTQUEUE	-	A pointer to the CNC program Input the poqDataOut output of "8.2 CNC Data Decoding".
	dAngleTol	LREAL	0.001	Tolerance angle up to which P-point control is performed
	bCheckAddAxVelJump	BOOL	FALSE	TRUE: Additional axes velocities are checked. Even if this parameter is set, nothing is reflected in operation. Do not use.
	dMaxAddAxVelDifference	LREAL	0	Maximum allowed velocity difference (u/s) Even if this parameter is set, nothing is reflected in operation. Do not use.
Output	bBusy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.

Scope	Name	Type	Default value	Description
	wErrorID	SMC_ERROR	SMC_NO_ERROR	Error ID output
	poqDataOut	POINTER TO SMC_OUTQUEUE	-	A pointer to SMC_OUTQUEUE that has checked the angle between the paths(Note 2) Specify this output for the poqDataIn input of "8.4.1 SMC_Interpolator (CNC Control Operation)".

(Note 1) The abort function operates only before the completion of SMC_NCDecoder or when the G code ""8.6.12 G75: Timing Synchronization"" is used.

If you want to stop axial movement in midstream, do not use bAbort but use the argument described in "8.4.1 SMC_Interpolator (CNC Control Operation)".

(Note 2) For the poqDataIn value, do not specify a pointer to values other than the CNC program. Otherwise, SMC_CheckVelocities and subsequent processes will not be executed.

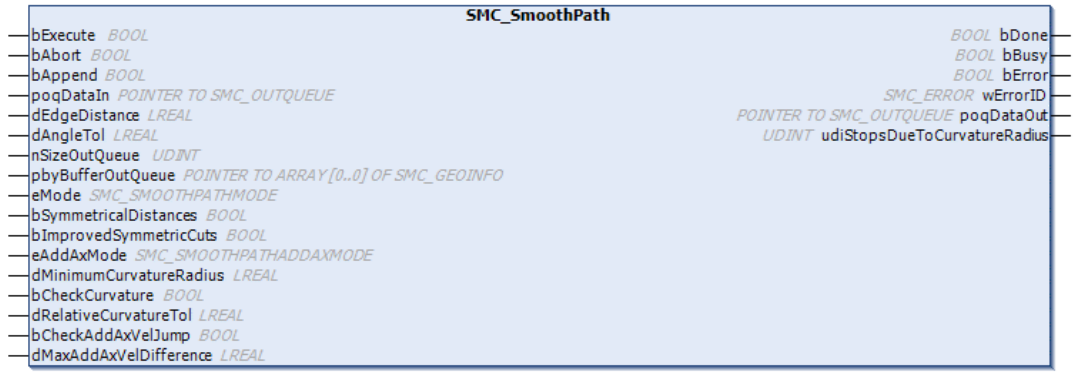
Info.

- See "8.7.2 Example of Use: C-point Control and P-point Control" for examples of use. An explanation of the definition of the angle between the paths is also described.

8.3.2 SMC_SmoothPath (path smoothing)

This function block can smooth bends in the path of the specified CNC program. G51 and G50 in the G-code are used to perform smoothing. Unlike SMC_RoundPath, the entire path is also subject to smoothing, not just between paths. This FB must be run before running SMC_Interpolator. Execute the function block by MotionTask.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.

Scope	Name	Type	Default value	Description
	bAbort	BOOL	FALSE	TRUE: Execution of the FB is stopped.
	bAppend	BOOL	FALSE	TRUE: At the rising edge as specified by bExecute, the poqDataOut data within the FB is not reset. Decoded data of ncprogIn is appended to the end of poqDataOut. (Note 1)
	poqDataIn	POINTER TO SMC_OUTQUEUE	-	Specifies a pointer to the CNC program.(Note 2) Input the poqDataOut output of"8.2 CNC Data Decoding".
	dEdgeDistance	LREAL	0	Set the radius of curvature of the smoothing process to be added to parameter D of G-code G51.
	dAngleTol	LREAL	'0.001	Set the tolerance for the path-to-path angle at which smoothing is not performed
	nSizeOutQueue	UDINT	0	Specifies the size of the data buffer to which the list of SMC_GEOINFO structure objects will be written. This buffer must be able to hold at least five SMC_GEOINFO objects. If the size of the buffer is not satisfactory, no error occurs and the FB is not executed. The buffer size may be predefined, but may be changed only during a reset. [Declaration example] ExampleBuf: ARRAY[0..49] OF SMC_GEOINFO; (An array of five or more elements is required) [Example of acquiring appropriate buffer size] nSizeOutQueue:=SIZEOF(ExampleBuf);
	pbyBufferOutQueue	POINTER TO ARRAY [0..0] OF SMC_GEOINFO	-	Points to the first byte of the memory space assigned to the structure SMC_OUTQUEUE, which must be at least the same size as that defined by nSizeOutQueue. (Note 2)(Note 4) The buffer size may be predefined, but may be changed only during a reset. [Declaration example] ExampleBuf: ARRAY[0..49] OF SMC_GEOINFO; (An array of five or more elements is required) [Writing example] pbyBufferOutQueue:=ADR(ExampleBuf);

Scope	Name	Type	Default value	Description
	eMode	SMC_SMOOTHPATHMODE	SP_SPLINE3	Element type applied to path smoothing
	bSymmetricalDistances	BOOL	TRUE	Half the length of the short side of the two sides that form the angle and the smoothed radius of curvature D are compared, and the smaller value is taken as D'. TRUE: The smoothed radius of curvature is set to the value of D' FALSE: The value set in D is used
	bImprovedSymmetricCuts	BOOL	FALSE	The setting is reflected when bSymmetricalDistances = TRUE. TRUE:For comparison of the radius of curvature of bSymmetrical Distances at the second and subsequent turns, the radius of curvature used for the judgment at the first corner is applied.
	eAddAxMode	SMC_SMOOTHPATHADDAXMODE	SPAA_LATE	It does not affect the operation even if set. Do not use.
	dMinimumCurvatureRadius	LREAL	0	If the spline inserted in the smoothing path contains a position in which the radius of curvature is less than this parameter, it is not smoothed and the original path bend is used
	bCheckCurvature	BOOL	FALSE	TRUE: Check whether the curvature of adjacent elements is equal. If not equal, the path is smoothed.
	dRelativeCurvatureTol	LREAL	0.001	It does not affect the operation even if set. Do not use.
	bCheckAddAxVelJump	BOOL	FALSE	It does not affect the operation even if set. Do not use.
	dMaxAddAxVelDifference	LREAL		It does not affect the operation even if set. Do not use.
Output	bDone	BOOL	FALSE	TRUE：Completion of smoothing of input data
	bBusy	BOOL	FALSE	TRUE： Execution of the FB is not completed.
	bError	BOOL	FALSE	TRUE： An error has occurred within the FB.
	wErrorID	SMC_ERROR	SMC_NO_ERROR	Error ID output
	poqDataOut	POINTER TO SMC_OUTQUEUE	-	Pointer to the SMC_GEOINFO object that has executed the smoothing process Specify this output for the poqDataIn input of"8.4.1 SMC_Interpolator (CNC Control Operation)".

Scope	Name	Type	Default value	Description
	udiStopsDueToCurvatureRadius	UDINT	0	Number of bends that could not be smoothed due to the dMinimumCurvatureRadius setting

(Note 1) To use the bAppend function, set bAppend to TRUE simultaneously with bAppend of SMC_NCDecoder.

(Note 2) If the input variables are not set correctly, bBusy remains TRUE and the function block will not be executed without error.

(Note 3) For details, refer to “G50, G51, G52”

(Note 4) Do not set SMC_GEOINFO objects that are specified in other function blocks.

SMC_SMOOTHPATHMODE (Enumeration type)

Name	Description
SP_SPLINE3	A 3rd order spline with different tangent lengths is inserted to define the spline. The length is dependent on the length of the adjacent object.
SP_SPLINE5	A 5th order spline is inserted.
SP_SPLINE3_CV	A 3rd order spline with different tangent lengths is inserted to define the spline. The length is dependent on the length of the portion of the cut adjacent object. For two adjacent line objects, SP_SPLINE3_CV stays inside the convex groove of the original path.
SP_SPLINE5_CV	A 5th order spline with different tangent lengths is inserted to define the spline. The length is dependent on the length of the portion of the cut adjacent object.
SP_SPLINE5_MIN_CURVATURE	Multiplication with the 5th order polynomial of the minimum curvature.

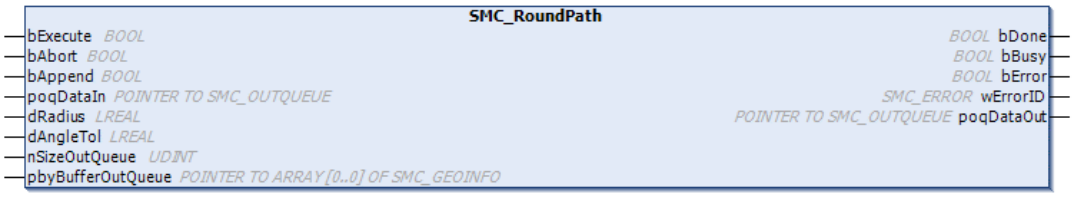
Info.

- For the method of using G50, G51, refer to"8.6.10 G50, G51, G52: Path Smoothing".

8.3.3 SMC_RoundPath (Arc correction between paths)

This function block can correct between paths in the specified CNC program with an arc. To perform smoothing, use G52 and G50 in G-code. Unlike SMC_SmoothPath, arc correction is applied only between the paths in the specified section. This FB must be run before running SMC_Interpolator. Execute the function block by MotionTask.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	bAbort	BOOL	FALSE	TRUE: Execution of the FB is stopped.
	bAppend	BOOL	FALSE	TRUE: At the rising edge as specified by bExecute, the poqDataOut data within the FB is not reset. Decoded data of ncprogIn is appended to the end of poqDataOut. (Note 1)
	poqDataIn	POINTER TO SMC_OUTQUEUE	-	Specifies a pointer to the CNC program.(Note 2) Input the poqDataOut output of"8.2 CNC Data Decoding".
	dRadius	LREAL	0	Set the radius of curvature of arc correction to be added to parameter D of G code G52.(Note 3)
	dAngleTol	LREAL	'0.001	Sets the tolerance for the angle between paths where arc correction is not performed.
	nSizeOutQueue	UDINT	0	Specifies the size of the data buffer to which the list of SMC_GEOINFO structure objects will be written. This buffer must be able to hold at least five SMC_GEOINFO objects. If the size of the buffer is not satisfactory, no error occurs and the FB is not executed. The buffer size may be predefined, but may be changed only during a reset. [Declaration example] ExampleBuf: ARRAY[0..49] OF SMC_GEOINFO; (An array of five or more elements is required) [Example of acquiring appropriate buffer size] nSizeOutQueue:=SIZEOF(ExampleBuf);
	pbyBufferOutQueue	POINTER TO ARRAY [0..0] OF SMC_GEOINFO	-	Points to the first byte of the memory space assigned to the structure SMC_OUTQUEUE, which must be at least the same size as that defined by nSizeOutQueue. (Note 2)(Note 4) The buffer size may be predefined, but may be changed only during a reset. [Declaration example] ExampleBuf: ARRAY[0..49] OF SMC_GEOINFO;

Scope	Name	Type	Default value	Description
				(An array of five or more elements is required) [Writing example] pbyBufferOutQueue:=ADR(ExampleBuf);
Output	bDone	BOOL	FALSE	TRUE：Completion of smoothing of input data
	bBusy	BOOL	FALSE	TRUE： Execution of the FB is not completed.
	bError	BOOL	FALSE	TRUE： An error has occurred within the FB.
	wErrorID	SMC_ERROR	SMC_NO_ERROR	Error ID output
	poqDataOut	POINTER TO SMC_OUTQUEUE	-	Pointer to the SMC_GEOINFO object that performed arc correction Specify this output for the poqDataIn input of"8.4.1 SMC_Interpolator (CNC Control Operation)".

(Note 1) To use the bAppend function, set bAppend to TRUE simultaneously with bAppend of SMC_NCDecoder.

(Note 2) If the input variables are not set correctly, bBusy remains TRUE and the function block will not be executed without error.

(Note 3) If the radius specification is 0, subsequent arc correction will not be performed.

(Note 4) Do not set SMC_GEOINFO objects that are specified in other function blocks.

Info.

- For the method of using G50, G52, refer to"8.6.10 G50, G51, G52: Path Smoothing".

8.3.4 SMC_ToolRadiusCorr (Tool Radius Correction for Path)

This is a function block (FB) that performs tool radius correction. This FB converts the section of the path specified by G41 to G40 or G42 to G40 in the CNC program so that it is offset by the tool radius. When specified by G41, the path will be corrected to the right for the radius. When specified by G42, the path will be corrected to the left for the radius. This FB must be run before running SMC_Interpolator. Execute the function block by MotionTask.

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	bAbort	BOOL	FALSE	TRUE: Execution of the FB is stopped.
	bAppend	BOOL	FALSE	TRUE: At the rising edge as specified by bExecute, the poqDataOut data within the FB is not reset. The poqDataIn data is appended to the end of the poqDataOut output.
	poqDataIn	POINTER TO SMC_OUTQUEUE	-	Specifies the address of the CNC table.
	nSizeOutQueue	UDINT	0	Specifies the size of the data buffer to which the list of SMC_GEOINFO structure objects will be written. This buffer must be able to hold at least five SMC_GEOINFO objects. If the size of the buffer is not satisfactory, no error occurs and the FB is not executed. The buffer size may be predefined, but may be changed only during a reset. [Declaration example] ExampleBuf: ARRAY[0..49] OF SMC_GEOINFO; (An array of five or more elements is required) [Example of acquiring appropriate buffer size] nSizeOutQueue:=SIZEOF(ExampleBuf);
	pbyBufferOutQueue	POINTER TO ARRAY [0..0] OF SMC_GEOINFO	-	Specifies the address of the memory space for SMC_OUTQUEUE.(Note 1)(Note 2) We recommend that array SMC_GEOINFO be defined and an

Scope	Name	Type	Default value	Description
				address be specified as shown in the following example. [Declaration example] ExampleBuf: ARRAY[0..49] OF SMC_GEOINFO; (An array of five or more elements is required) [Writing example] pbyBufferOutQueue:=ADR(ExampleBuf);
Output	bDone	BOOL	FALSE	TRUE : Processing of input data from poqDataIn is completed.
	bBusy	BOOL	FALSE	TRUE: FB is in progress.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	wErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.
	poqDataOut	POINTER TO SMC_OUTQUEUE	-	An address to the CNC table which manages corrected SMC_GEOINFO objects

(Note 1) If the input variables are not set correctly, bBusy remains TRUE and the function block will not be executed without error.

(Note 2) Do not set SMC_GEOINFO objects that are specified in other function blocks.

Info.

- Use this function block with the bAppend input of SMC_NCDecoder and SMC_NCInterpreter set to FALSE.
- To perform coordinate conversion when using this function block, set 3D mode using G17, etc. However, it cannot be used in combination with coordinate rotation conversion.
- Timing synchronization by G75 cannot be used during tool radius correction. Execute G75 after movement to the line following G40 (after radius correction is canceled).

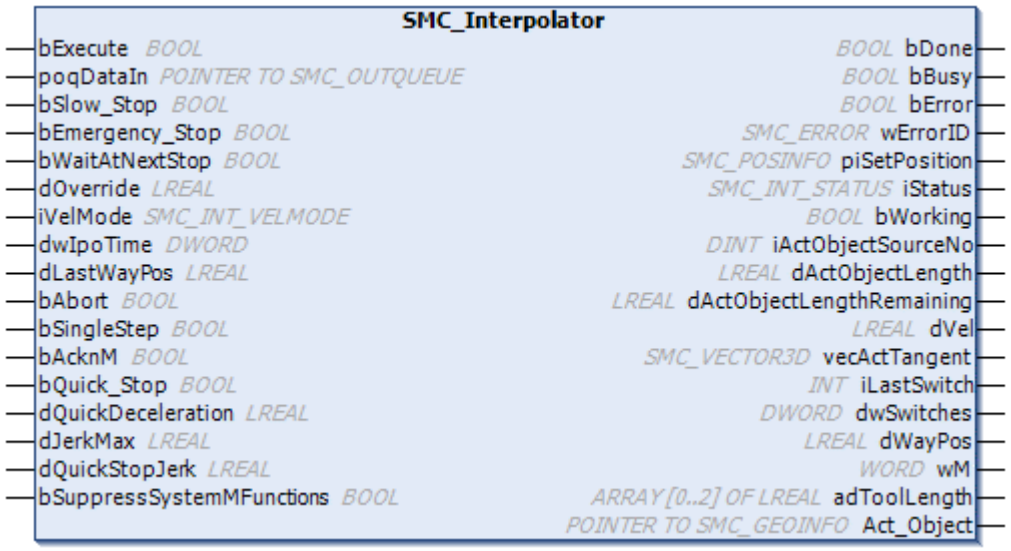
  For details, refer to "Example: Combined use of timing synchronization by G75 and tool correction".
- For the method of using G40, G41, and G42, refer to "8.6.8 G40, G41, G42: Tool Radius Correction for Path".

8.4 Control calculation

8.4.1 SMC_Interpolator (CNC Control Operation)

This is a function block (FB) that converts a continuous path described by SMC_GEOINFO objects into discrete path position points taking into account a defined velocity profile and time pattern. Execute the function block by MotionTask.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Default value	Description
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	poqDataIn	POINTER TO SMC_OUTQUEUE	-	Specifies a pointer to the CNC file. Input the poqDataOut output of "8.3 Pre-processing after decoding".
	bSlow_Stop	BOOL	FALSE	TRUE: Executes deceleration stop according to the velocity profile (iVelMode). FALSE: The pause is canceled.
	bEmergency_Stop	BOOL	FALSE	TRUE: Causes an emergency stop, so that piSetPosition will be retained at the current value and the velocity will be set to 0.(Note 1) FALSE: The emergency stop is canceled.
	bWaitAtNextStop(Note 4)	BOOL	FALSE	TRUE: Executes a pause at points in the CNC program where the velocity between paths becomes zero.

Scope	Name	Type	Default value	Description
				FALSE: The pause is canceled.
	dOverride	LREAL	1	This variable can be used to handle the override. The velocity and acceleration/deceleration get scaled by dOverride. （0.01～）(Note 2) The modified override will only be applied if axis acceleration or deceleration is not in progress.
	iVelMode	SMC_INT_VEL MODE	TRAPEZOID	Specifies a velocity profile.(Note 3)
	dwIpoTime	DWORD	0	MotionTask interval (μsec)
	dLastWayPos	LREAL	0	The total length of the path generated by the CNC control operation can be measured. To use this, dLastWayPos needs to be connected to dWayPos.
	bAbort	BOOL	FALSE	TRUE: Execution of the FB is stopped.
	bSingleStep(Note 4)	BOOL	FALSE	TRUE: All connections between paths are established through deceleration stop.
	bAcknM	BOOL	FALSE	TRUE：Output wM is cleared and processing resumes from the paused state. Since the processing resumes when the launch of TRUE is detected, keep the input at TRUE for one cycle and return it to FALSE.
	bQuick_Stop	BOOL	FALSE	TRUE: Reduces the velocity of the object to zero and stops it. The velocity is reduced according to the velocity profile specified in iVelMode and the deceleration given by the maximum of the values specified in dQuickDeceleration and programmed in the path. If a quadratic velocity profile is used, the jerk is limited by max(dJerkMax, dQuickStopJerk). FALSE: Cancels deceleration stop.
	dQuickDeceleration	LREAL	0	Specifies a deceleration value used for bQuick_Stop.
	dJerkMax	LREAL	0	Magnitude of the maximum allowed jerk used for quadratic velocity profiles Must be positive and cannot be changed while performing a CNC control operation
	dQuickStopJerk	LREAL	0	Specifies the jerk used for bQuick_Stop.

Scope	Name	Type	Default value	Description
	bSuppressSystemMFunctions	BOOL	FALSE	TRUE: The output wM is not set when G75 or G4 command is executed.
Output	bDone	BOOL	FALSE	TRUE: Output is completed.
	bBusy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	wErrorID	SMC_ERROR	SMC_NO_ERROR	Error ID output during CNC control operation
	piSetPosition	SMC_POSINFO	-	The target coordinates of the next position set by CNC control operation (Cartesian coordinate system)
	iStatus	SMC_INT_STATUS	IPO_INIT	The status of CNC control operation
	bWorking	BOOL	FALSE	TRUE: The process is underway. Specify this output for the bEnable input of SMC_ControlAxisByPos.
	iActObjectSourceNo	DINT	-1	Outputs a value of iSourceLine_No of active SMC_GEOINFO object of poqDataIn-queue. When bWorking = FALSE, the value is set to -1.
	dActObjectLength	LREAL	0	The length of the current object. Valid if bWorking = TRUE. A correct value may not be output. Do not use.
	dActObjectLengthRemaining	LREAL	0	The remaning length of the current object. Valid if bWorking = TRUE. A correct value may not be output. Do not use.
	dVel	LREAL	0	Current path velocity
	vecActTangent	SMC_VECTOR3D	-	Current path tangent, a unit vector
	iLastSwitch	INT	0	The number of the last switch H passed
	dwSwitches	DWORD	0	The current switch status of H switches 1 to 32, in bit notation
	dWayPos	LREAL	0	Refer to dLastWayPos.
	wM	WORD	0	Number of M-code where CNC control operation is paused.
	adToolLength	ARRAY [0..2] OF LREAL	-	Parameters for tool length compensation
	Act_Object	POINTER TO SMC_GEOINFO	0	Pointer to the CNC program in executable format currently in progress The line number in progress and G-code information can be acquired.

(Note 1) Make sure that bEmergency_Stop is connected to SMC_ControlAxisByPos.bStopIpo.

(Note 2) Set dOverride to a numerical value greater than 0.01.
If set to a smaller value, no axis movement starts and no error occurs.

(Note 3) Axis velocity ramp type settings do not apply to CNC control using SMC_Interpolator.
Specify the velocity profile in iVelMode.

(Note 4) Please do not use bWaitAtNextStop and bSingleStep inputs together as setting both to TRUE may cause the function block to operate incorrectly.

SMC_INT_VELMODE (Enumeration type)

Name	Value	Description
TRAPEZOID	0	Trapezoid
SIGMOID	1	Sin2
SIGMOID_LIMIT	2	Sin2 (limit)
QUADRATIC	3	Quadratic
QUADRATIC_SMOOTH	4	Quadratic (smooth)

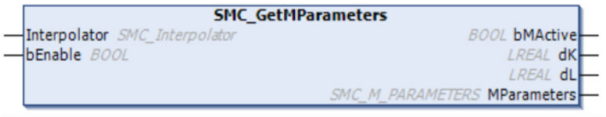
SMC_INT_STATUS (Enumeration type)

Name	Value	Description
IPO_UNKNOWN	0	Internal state that may not occur after a complete pass of the SMC_Interpolator
IPO_INIT	1	Initialization state, movement not started yet
IPO_ACCEL	2	Currently accelerating
IPO_CONSTANT	3	Movement ongoing with constant velocity
IPO_DECEL	4	Currently decelerating
IPO_FINISHED	5	The execution of the CNC program is done. From then on, SMC_GEOINFO object input in poqDataIn is not processed.
IPO_WAIT	6	Currently waiting by a stop input in SMC_Interpolator or a M-code process
IPO_INCREASING_ACCEL	7	Currently increasing the acceleration
IPO_DECREASING_ACCEL	8	Currently decreasing the acceleration
IPO_INCREASING_DECEL	9	Currently increasing the deceleration
IPO_DECREASING_DECEL	10	Currently decreasing the deceleration

8.4.2 SMC_GetMParameters (Get M-code Parameters)

If SMC_Interpolator is paused by an M-code, this function block (FB) can be used to get the parameters that have been set for this M-code (K, L, O words).

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	Interpolator	SMC_Interpolator	-	Instance of the SMC_Interpolator
Input	bEnable	BOOL	FALSE	TRUE: Execution of the FB is enabled.
Output	bMActive	BOOL	FALSE	TRUE: Processing is paused by an M-code
	dK	LREAL	0	Parameter specified via word K
	dL	LREAL	0	Parameter specified via word L
	MParameters	SMC_M_PARAMETERS	-	Parameters specified by a variable of type SMC_M_PARAMETERS set by the O-word

SMC_M_PARAMETERS (STRUCT)

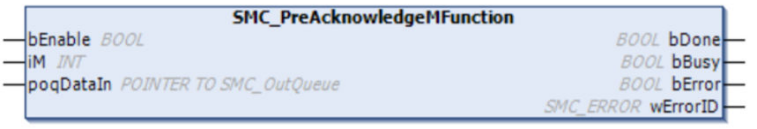
This data type structure allows you to define additional parameters for the currently active M-code that can be gotten by SMC_GetMParameters.

Name	Value	Default value
dP1	LREAL	0
dP2	LREAL	0
dP3	LREAL	0
dP4	LREAL	0
dP5	LREAL	0
dP6	LREAL	0
dP7	LREAL	0
dP8	LREAL	0

8.4.3 SMC_PreAcknowledgeMFunction (Deactivate M-code)

This function block (FB) is designed to deactivate the process of an M-code before SMC_Interpolator executes the M-code. This FB must be executed in the same task as the SMC_Interporalter. By letting this FB acknowledge the M-code in advance, you can switch between a pause by the M-code and nonpause.

■ Icon



■ Parameter

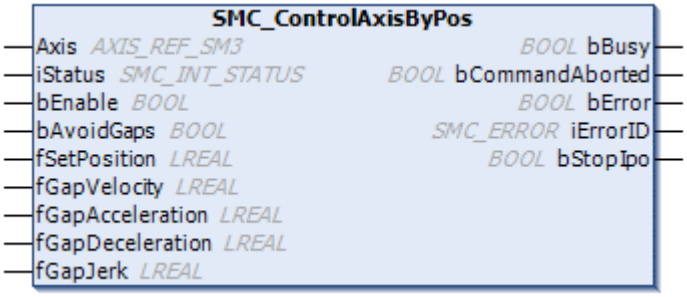
Scope	Name	Type	Default value	Description
Input	bEnable	BOOL	FALSE	TRUE: Execution of the FB is enabled. Whenever this FB is executed once, the single M-code is deactivated and the bDone output changes to TRUE. To execute the FB consecutively, immediately after the bDone output changes to TRUE, set the bEnable input to FALSE and set it to TRUE again.
	iM	INT	0	M-code to be deactivated
	poqDataIn	POINTER TO SMC_OUTQUEUE	-	Specifies a pointer to the CNC file. Input the poqDataOut output of "8.3 Pre-processing after decoding".
Output	bDone	BOOL	FALSE	TRUE: M-code deactivation is completed.
	bBusy	BOOL	FALSE	TRUE: FB is in progress.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	wErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

8.5 Control command & kinematics conversion

8.5.1 SMC_ControlAxisByPos (Axis Position Control)

This function block (FB) writes the input variable fSetPosition to the drive structure (AXIS_REF_SM3) and controls axis movement. Performs position control while monitoring whether the axis speed exceeds the dynamic limit setting. Execute the function block by MotionTask.

■ **Icon**



■ **Parameter**

The FB monitors if the axis velocity exceeds the dynamic limit setting. If the velocity exceeds the limit setting, the FB outputs bStopIpo = TRUE to decelerate it. The FB adapts the axis to the position at which the velocity exceeded the limit setting. When adaptation to the position is completed, the FB outputs bStopIpo = FALSE and then returns to its original control.

Scope	Name	Type	Default value	Description
I/O	Axis	AXIS_REF_SM3	-	Reference to the axis
Input	iStatus	SMC_INT_STATUS	IPO_INIT	Substitute iStatus of SMC_Interpolator
	bEnable	BOOL	FALSE	TRUE: The FB can be executed.
	bAvoidGaps	BOOL	TRUE	TRUE：Enable monitoring of dynamic limits The axis is moved to the position according to the values set in fGapVelocity, fGapAcceleration, and fGapDeceleration.
	fSetPosition	LREAL	-	Set position of the axis in (u).
	fGapVelocity	LREAL	1	Follow-up speed when the dynamic limit is exceeded (u/s) Set the value within the dynamic limit.(Note 1)
	fGapAcceleration	LREAL	1	Follow-up acceleration when the dynamic limit is exceeded (u/s^2) Set the value within the dynamic limit.(Note 1)

Scope	Name	Type	Default value	Description
	fGapDeceleration	LREAL	1E+15	Follow-up deceleration when the dynamic limit is exceeded (u/s^2) Set the value within the dynamic limit.(Note 1) Deceleration is also used if bAvoidGaps = FALSE, for stopping after the change of bEnable from TRUE to FALSE.
	fGapJerk	LREAL	1E+16	Follow-up jerk when dynamic limit is exceeded (u/s^3) Enabled when the axis velocity ramp type is set to a value other than “Trapezoid”
Output	bBusy	BOOL	FALSE	TRUE: FB operation is in progress.
	bCommandAborted	BOOL	FALSE	TRUE: An interruption is caused by another FB.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	iErrorID	SMC_ERROR	SMC_NO_ERROR	Error ID output
	bStopIpo	BOOL	FALSE	TRUE：The speed of the axis exceeds the set value of the dynamic limit, and follow-up control is being executed.(Note 2)

(Note 1) Do not set fGapVelocity, fGapAcceleration, and fGapDeceleration to values that exceed the axis dynamic limit settings.

(Note 2) Make sure that bStopIpo is connected to SMC_Interpolator.bEmergency_Stop.

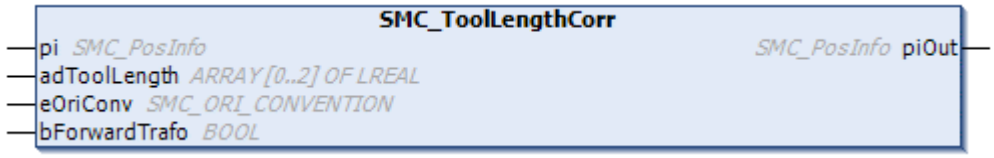
8.5.2 SMC_ToolLengthCorr (Tool Length Correction)

This is a function block (FB) that performs tool length correction. This FB converts the path of the CNC program so that it is offset by the tool length. The tool length can be specified by setting the adToolLength I/O in this FB, by setting the vStartToolLength input in the decoder FB, or by using G43 in the CNC program file. Execute the function block by MotionTask.

■ Conversion formula

- When bForwardTrafo is set to FALSE:
  piOut.dX = pi.dX - adToolLength[0]
  piOut.dY = pi.dY - adToolLength[1]
  piOut.dZ = pi.dZ - adToolLength[2]
- When bForwardTrafo is set to TRUE:
  piOut.dX = pi.dX + adToolLength[0]
  piOut.dY = pi.dY + adToolLength[1]
  piOut.dZ = pi.dZ + adToolLength[2]

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	pi	SMC_PosInfo	-	Inputs SMC_Interpolator.piSetPosition.
	adToolLength	ARRAY [0..2] OF LREAL	-	Tool length to be offset x-, y-, and z-axis directions in order from the smallest to the largest in the array
Input	eOriConv	SMC_ORI_CONVENTION	ZYZ	When set to ADDAXES, tool length correction is not executed.(Note 1)
	bForwardTrafo	BOOL	FALSE	FALSE: Executes backward conversion in which the tool length is subtracted from the path coordinates. TRUE: Executes forward conversion in which the tool length is added to the path coordinates.
Output	piOut	SMC_PosInfo	-	Inputs to the function block for kinematics conversion.

(Note 1) Set the eOriconv input of the decoder (SMC_NCDecoder or SMC_NCInterpreter) and that of SMC_ToolLengthCorr to the same value.

■ Setting the tool length

There are the following two methods to set the tool length.

- Method 1: Specifying only the tool number in the CNC program to perform tool length correction

Set and manage tool information (SMC_Vector3d, etc.) in advance in the program according to the tool to be used. To change the tool under CNC program control, use the M-code to call the required tool information.

Set the tool length information in the adToolLength I/O of SMC_ToolLengthCorr in the program and specify the tool length to be corrected.

- Method 2: Specifying the tool length correction value in the CNC program to perform correction

Set the tool length at the start of operation in the vStartToolLength input of SMC_NCDecoder or SMC_NCInterpreter. To change the correction tool length later under CNC program control, use G43 to specify the tool length in the CNC program file.

Note that the value of vStartToolLength is used only at the start of decoding. This means that changes made under CNC program control will not be reflected in the operation.

For details on each method, refer to "Example: Tool length correction in z-axis direction".

Info.

- Use SMC_ControlAxisByPos with bAvoidGaps set to TRUE. Depending on the current position or the target position, sudden movements such as jumps in the path may occur due to the effect of tool length correction.
- The tool length correction value is not affected by coordinate conversion. With the tool length is set in the x-, y-, and z-axis directions in the coordinate system before coordinate conversion, tool length correction is applied to the path after coordinate conversion.
- For the method of using G43, refer to "8.6.9 G43: Tool Length Correction".

8.5.3 SMC_TRAFO_Polar (Conversion from Two-dimensional (X, Y) Coordinates to Polar Coordinates)

This function block (FB) converts two-dimensional (X, Y) coordinates into polar (R, φ) coordinates. The calculation of and conversion to R and φ are performed as follows.Please execute it as a motion task.

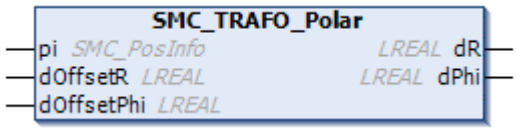
■ Conversion formula

R = (X^2 ＋ Y^2)^0.5 + offsetR　φ=Atan(Y / X) + offsetφ

When X = 0, φ = 90 deg (Y > 0) or -90 deg (Y≦ 0)

When Y =0, φ= 0 deg (X > 0) or -180 deg (X < 0)

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	pi	SMC_PosInfo	-	Two-dimensional (X, Y) coordinates
	dOffsetR	LREAL	0	Offset for radial distance axis R
	dOffsetPhi	LREAL	0	Offset for angular direction axis φ in degree
Output	dR	LREAL	-	Position of radial distance axis R after conversion
	dPhi	LREAL	-	Position of angular direction axis φ in degree after conversion(Note 1)

(Note 1)　An angle that forms φ = 180 deg is converted as an angle of φ = -180 deg.

8.5.4 SMC_TRAFOF_Polar (Conversion from Polar Coordinates to Two-dimensional (X, Y) Coordinates)

This function block (FB) converts polar (R, φ) coordinates into two-dimensional (X, Y) coordinates. The calculation of and conversion to X and Y are performed as follows.Please execute it as a motion task.

■ Conversion formula

X = (R + offsetR) * cos(φ + offsetφ) Y = (R + offsetR) * sin(φ + offsetφ)

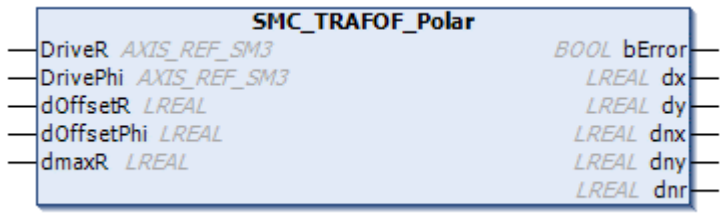
The norm after normalization with the maximum radius is calculated below.

nX = X / (dmaxR - offsetR)

nY = Y / (dmaxR - offsetR)

nR = (R - offsetR) / (dmaxR - offsetR)

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
I/O	DriveR	AXIS_REF_SM3	-	Reference to current position of radial distance axis R, fActPotion
	DrivePhi	AXIS_REF_SM3	-	Reference to current position of angular direction axis φ, fActPotion
Input	dOffsetR	LREAL	0	Offset for radial distance axis R
	dOffsetPhi	LREAL	0	Offset for angular direction axis φ in degree
	dmaxR	LREAL	0	Maximum radius R (> 0) used in normalization(Note 1)
Output	bError	BOOL	FALSE	TRUE if conversion is not possible
	dx	LREAL	0	X-coordinate after conversion
	dy	LREAL	0	Y-coordinate after conversion
	dnx	LREAL	0	A position vector to the X-coordinate after conversion (after normalized by maximum radius)
	dny	LREAL	0	A position vector to the Y-coordinate after conversion (after normalized by maximum radius)

Scope	Name	Type	Default value	Description
	dnr	LREAL	0	Norm after normalized by maximum radius

(Note 1) If the function block operates under the initial conditions of dmaxR = 0 and dOffsetR = 0, an exception error occurs.

8.5.5 SMC_TRAFO_Bipod_Arm (Bipod robot hand XY coordinates → conversion of each axis position)

It is a function block (FB) that converts the XY coordinates of the hand of the Bipod robot into the angle information of each axis motor. Please execute it as a motion task.

■ Conversion formula

dA ={-180-atan(Yo/Xa)} + acos {(L1^2 +Xa^2+Yo^2-L2^2)/(2*L1*(Xa^2+Yo^2)^1/2)} +dOffsetA

dB = atan(Yo/Xb) + acos {(L1^2 +Xb^2+Yo^2-L2^2)/(2*L1*(Xb^2+Yo^2)^1/2)} +dOffsetB

・When L2 > L1+D/2, Yo=Y-(L2^2-(L1+D/2)^2)^1/2 Other than that, Yo=Y

・Xa= X+1/2*D、Xb= X-1/2*D、L1=dArmLength1、L2=dArmLength2、D=dDistance

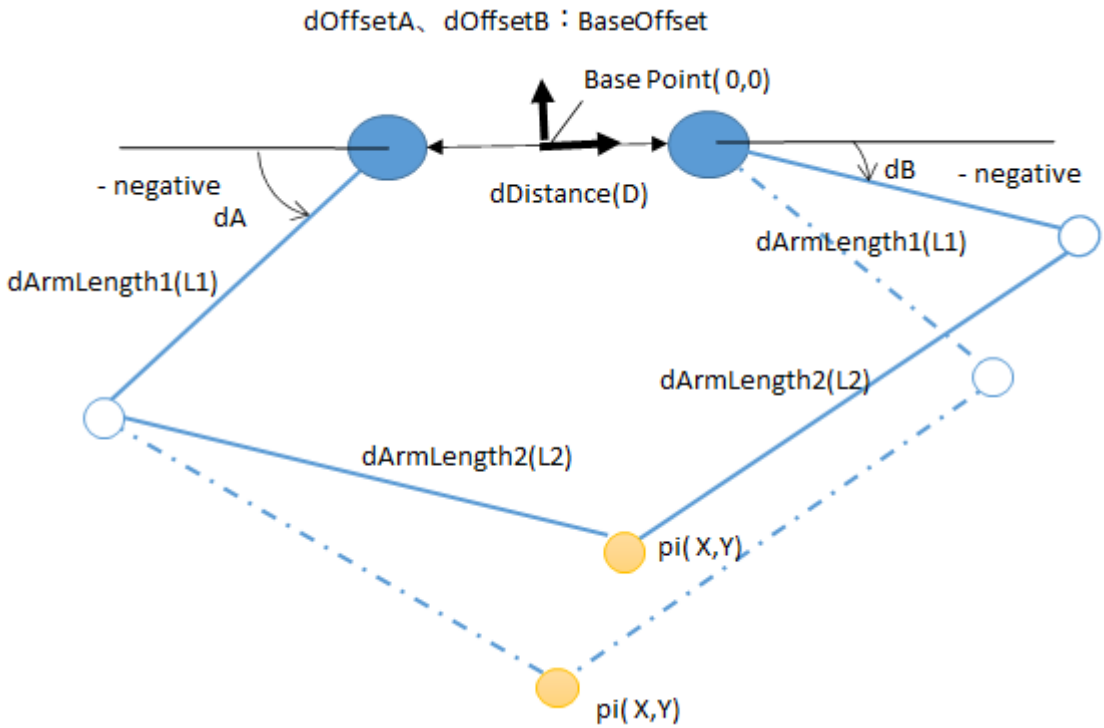
・When Xa = 0, atan (Yo / Xa) => -90, when Xb = 0, atan (Yo / Xb) => -90

・An error will occur if any of the following conditions are met:

1. Yo > 0
2. L1 ≦ 0 or L2 ≦ 0 or D < 0
3. (Xa^2+Yo^2)^1/2 > L1+L2 or (Xb^2+Yo^2)^1/2 > L1+L2
4. Posture that cannot be taken due to the mechanism

Note

- Please check the operating range according to the parameters in advance before use.
- Note that when L2> L1 + 1 / 2D, the origin position shifts in the Y direction by (L2 ^ 2- (L1 + D / 2) ^ 2) ^ 1/2 minutes.



■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	pi	SMC_PosInfo	-	XY2D coordinates of the hand
	dArmLength1(Note 1)	LREAL	0	Length from motor to joint(Note 2) dArmLength1>0
	dArmLength2(Note 1)	LREAL	0	Length from joint to hand(Note 2) dArmLength2>0
	dDistance(Note 1)	LREAL	0	Distance between two motors dDistance≧0
	dOffsetA(Note 1)	LREAL	0	Reference offset angle of left motor
	dOffsetB(Note 1)	LREAL	0	Reference offset angle of right motor
Output	bError	BOOL	FALSE	TRUE: Argument or error in calculation process
	dA	LREAL	0	Angle of left motor to hand position
	dB	LREAL	0	Angle of right motor to hand position

(Note 1) Even if the values of dArmLength1, dArmLength2, dDistance, dOffsetA, and dOffsetB are changed after executing FB of the same instance, they are not reflected.

(Note 2) If the function block is operated with the initial value of dArmLength1 = 0 and dArmLength2 = 0, an exception error will occur.

8.5.6 SMC_TRAFO_Gantry2 (Convert XY Gantry Coordinates to Positions of Axes)

This is a function block (FB) that converts XY gantry coordinates (Xin, Yin) to positions of the motor axes. Execute the function block by MotionTask.

■ Conversion formula

dx = Xin + dOffsetX

dy = Yin + dOffsetY

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	pi	SMC_PosInfo	-	Two-dimensional (X, Y) coordinates
	dOffsetX	LREAL	0	Offset for x-axis
	dOffsetY	LREAL	0	Offset for y-axis
Output	dx	LREAL	0	X-coordinate after conversion
	dy	LREAL	0	Y-coordinate after conversion

8.5.7 SMC_TRAFOF_Gantry2 (Conversion Positions of Axes -> XY Gantry Coordinates)

This is a function block (FB) that converts positions of the motor axes to XY gantry coordinates (Xout, Yout). Execute the function block by MotionTask.

■ Conversion formula

dx = X - dOffsetX

dy = Y - dOffsetY

When maxX - minX > 0 and maxY - minY > 0,

ratio = (maxX-minX)/(maxY-minY)

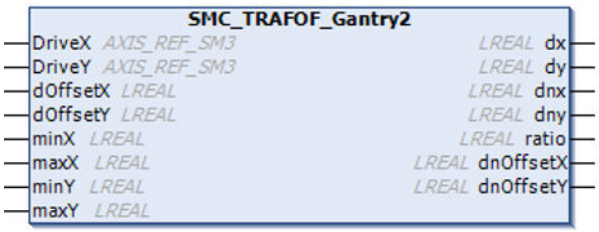
When ratio ≤ 1, dnOffsetX = ratio, dnOffsetY = 1

When ratio > 1, dnOffsetX = 1, dnOffsetY = 1 / ratio

dnx = dnOffsetX*(X-minX-dOffsetX)/(maxX-minX)

dny = dnOffsetY*(Y-minY-dOffsetY)/(maxY-minY)

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	DriveX	AXIS_REF_SM3	-	Specifies the x-axis.
	DriveY	AXIS_REF_SM3	-	Specifies the y-axis.
Input	dOffsetX	LREAL	0	Offset x (x-axis component)
	dOffsetY	LREAL	0	Offset y (y-axis component)
	minX	LREAL	0	Lower bound of move range along x-axis
	maxX	LREAL	0	Upper bound of move range along x-axis
	minY	LREAL	0	Lower bound of move range along y-axis
	maxY	LREAL	0	Upper bound of move range along y-axis
Output	dx	LREAL	0	X-coordinate after conversion
	dy	LREAL	0	Y-coordinate after conversion
	dnx	LREAL	0	A position vector to the X-coordinate after conversion (after normalized by move range)

Scope	Name	Type	Default value	Description
	dny	LREAL	0	A position vector to the Y-coordinate after conversion (after normalized by move range)
	ratio	LREAL	0	Ratio between x-axis and y-axis move ranges
	dnOffsetX	LREAL	0	X-offset for normalization
	dnOffsetY	LREAL	0	Y-offset for normalization

8.5.8 SMC_TRAFO_Gantry3 (Convert XYZ Gantry Coordinates to Positions of Axes)

This is a function block (FB) that converts XYZ gantry coordinates (Xin, Yin, Zin) to positions of the motor axes. Execute the function block by MotionTask.

■ Conversion formula

dx = Xin + dOffsetX

dy = Yin + dOffsetY

dz = Zin + dOffsetZ

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	pi	SMC_PosInfo	-	Three-dimensional XYZ coordinates
	dOffsetX	LREAL	0	Offset for x-axis
	dOffsetY	LREAL	0	Offset for y-axis
	dOffsetZ	LREAL	0	Offset for z-axis
Output	dx	LREAL	0	X-coordinate after conversion
	dy	LREAL	0	Y-coordinate after conversion
	dz	LREAL	0	Z-coordinate after conversion

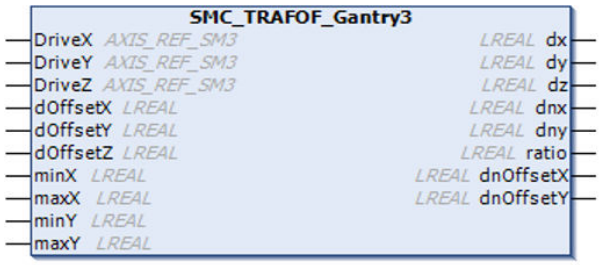
8.5.9 SMC_TRAFOF_Gantry3 (Conversion Positions of Axes -> XYZ Gantry Coordinates)

This is a function block (FB) that converts positions of the motor axes to XYZ gantry coordinates (Xout, Yout, Zout). Execute the function block by MotionTask.

■ Conversion formula

dx = Xin - dOffsetX
dy = Yin - dOffsetY
dz = Zin - dOffsetZ
When maxX - minX > 0 and maxY - minY > 0,
ratio = (maxX-minX)/(maxY-minY)
When ratio ≤ 1, dnOffsetX = ratio, dnOffsetY = 1
When ratio > 1, dnOffsetX = 1, dnOffsetY = 1 / ratio
dnx = dnOffsetX*(Xin-minX-dOffsetX)/(maxX-minX)
dny = dnOffsetY*(Yin-minY-dOffsetY)/(maxY-minY)

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input / output	DriveX	AXIS_REF_SM3	-	Specifies the x-axis.
	DriveY	AXIS_REF_SM3	-	Specifies the y-axis.
	DriveZ	AXIS_REF_SM3	-	Specifies the z-axis.
Input	dOffsetX	LREAL	0	Offset x (x-axis component)
	dOffsetY	LREAL	0	Offset y (y-axis component)
	dOffsetZ	LREAL	0	Offset y (z-axis component)
	minX	LREAL	0	Lower bound of move range along x-axis
	maxX	LREAL	0	Upper bound of move range along x-axis
	minY	LREAL	0	Lower bound of move range along y-axis
	maxY	LREAL	0	Upper bound of move range along y-axis
Output	dx	LREAL	0	X-coordinate after conversion

Scope	Name	Type	Default value	Description
	dy	LREAL	0	Y-coordinate after conversion
	dz	LREAL	0	Z-coordinate after conversion
	dnx	LREAL	0	A position vector to the X-coordinate after conversion (after normalized by move range)
	dny	LREAL	0	A position vector to the Y-coordinate after conversion (after normalized by move range)
	ratio	LREAL	0	Ratio between x-axis and y-axis move ranges
	dnOffsetX	LREAL	0	X-offset for normalization
	dnOffsetY	LREAL	0	Y-offset for normalization

8.5.10 SMC_TRAFO_GantryCutter2 (Convert XY Gantry Coordinates with Tool rotation to Positions of Axes)

This is a function block (FB) that converts XY gantry coordinates (Xin, Yin) with a tool rotation axis to positions of the motor axes. Execute the function block by MotionTask.

■ Conversion formula

dx = Xin + dOffsetX

dy = Yin + dOffsetY

dr = 180.0/π * iDirectionR * atan(Vy/Vx) + dOffsetR

wherein when Vx = 0 and Vy = 0, dr = 0, and calculation results are corrected in the range 0 ≤ dr < 360.

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	pi	SMC_PosInfo	-	Two-dimensional (X, Y) coordinates
	v	SMC_VECTOR3D	-	Current path tangent, a unit vector
	dOffsetX	LREAL	0	Offset for x-axis
	dOffsetY	LREAL	0	Offset for y-axis
	dOffsetR	LREAL	0	Offset for rotation axis
	iDirectionR	INT	1	Rotation factor The rotation direction can be set to either a positive or negative value.
Output	dx	LREAL	0	X-coordinate after conversion
	dy	LREAL	0	Y-coordinate after conversion
	dr	LREAL	0	Rotation angle (deg) after conversion

8.5.11 SMC_TRAFO_GantryCutter3 (Convert XYZ Gantry Coordinates with Tool rotation to Positions of Axes)

This is a function block (FB) that converts XYZ gantry coordinates (Xin, Yin, Zin) with a tool rotation axis to positions of the motor axes. Execute the function block by MotionTask.

■ Conversion formula

dx = Xin + dOffsetX

dy = Yin + dOffsetY

dz = Zin + dOffsetZ

dr = 180.0/π * iDirectionR * atan(Vy/Vx) + dOffsetR

wherein when Vx = 0 and Vy = 0, dr = 0, and calculation results are corrected in the range 0 ≤ dr < 360.

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	pi	SMC_PosInfo	-	Two-dimensional (X, Y) coordinates
	v	SMC_VECTOR3D	-	Current path tangent, a unit vector
	dOffsetX	LREAL	0	Offset for x-axis
	dOffsetY	LREAL	0	Offset for y-axis
	dOffsetZ	LREAL	0	Offset for z-axis
	dOffsetR	LREAL	0	Offset for rotation axis
	iDirectionR	INT	1	Rotation factor The rotation direction can be set to either a positive or negative value.
Output	dx	LREAL	0	X-coordinate after conversion
	dy	LREAL	0	Y-coordinate after conversion
	dz	LREAL	0	Z-coordinate after conversion
	dr	LREAL	0	Rotation angle R (deg) after conversion

8.6 CNC Program Operation and Setting Method

This chapter explains CNC program operation.

8.6.1 CNC Editor and Coding Rules

There are the following rules on coding of CNC programs.

■ Line number (N number)

G-codes must be written in lines that begin with an N-number (N**) in the range of 0 to 4294967295. Although duplicate N numbers do not cause any problem in operation, it is difficult to distinguish the N number being executed.

Example:

N10 G01 X0 Y0 Z0 F10

N20 G01 X100 Y100 Z100 F10

By selecting **CNC**>**Renumber CNC Program** from the GM Programmer menu, you can renumber the line numbers in increments of 10 starting from N000.

■ Only one G-code per line

It is not permitted to write more than one G-code or the same G-code word in a single line. Split the G-codes as shown below in coding.

- Acceptable coding

  N01 G17

  N02 G01 X10 Y10 F10

  N03 G01 X20

- Unacceptable coding

  N01 G17 G01 X10 Y10 F10

  or

  N01 G17

  N02 G01 X10 Y10 F10 X20

■ G-code omitted writing

If a G-code is omitted in a line, the previously executed G-code is assumed and executed. G-codes that can be omitted are G00 and G01 for linear interpolation and G02 and G03 for circular interpolation.

Example: G01 is omitted in N20 and N30 lines.

N10 G01 X0 Y0 Z0 F10

N20 X100 Y100

N30 Z100

The above coding and the following coding

N10 G01 X0 Y0 Z0 F10

N20 G01 X100 Y100

N30 G01 Z100

are equivalent.

■ G-code word omitted writing

Some G-code words can be omitted when writing a CNC program, such as target position X, Y, Z, etc.

G-code word	Set value	Description
X、Y、Z	Target position in interpolated movement Position shift amount in coordinate conversion Start position specification	Can be omitted Writing only X, Y, or Z is acceptable.
I、J、K	Center point of circular interpolation in center-point approach Tool length	Can be omitted Writing only I, J, or K is acceptable.
	Normal vector for plane designation	Cannot be omitted All of I, J, and K must be specified
	Scaling factor for coordinate conversion	All of I, J, and K must be specified, if written

Parameters using other G-code words must be written. For details, refer to the description of each G-code.

■ Velocity, acceleration, and deceleration settings

Motion velocity, acceleration, and deceleration settings can be omitted. If they are omitted in the SMC_CNC_REF type or SMC_OUTQUEUE type program, the "default values" in the CNC program setting properties will be applied to movements. For CNC program files, the default values set in "SMC_ReadNCFile2" will be applied to movements.

To change these settings in a CNC program, write the program with the following G-code words.

Set value	Code	Remarks
Velocity	Fxxx (xxx: Velocity)	The unit is u/s.
Acceleration, deceleration	Exxx (xxx: Acceleration, deceleration)	Xxx > 0: Specifies acceleration. Xxx < 0: Specifies deceleration.

Velocity, acceleration, and deceleration can be specified in the following ways.

- Example 1: Batch specification

  Once you set velocity, acceleration, and deceleration, the same velocity, acceleration, and deceleration will be applied until you change the values.

  N00 F100 E10 E-10

  N10 G01 X100 Y50 Z10

- Example 2: Sequential setting

  Specify velocity, acceleration, and deceleration for every interpolation operation. When you change velocity, acceleration, or deceleration settings in the CNC program more than once, using this input method can prevent input mistakes.

  N10 G01 X100 Y50 Z10 F100 E10 E-10

  N20 G01 X150 Y100 Z50 F200 E5 E-5

■ Use of variables (global variables) in CNC programs

Variables declared in POU can be used in CNC programs. To use a variable in a CNC program, specify it between $ marks. This method can be used only when the CNC program of SMC_CNC_REF type is created.

Example: Using a variable declared as Variable in POU for the variable to be set with G36

```
N10 G01 X0 Y0 Z0 F10
N20 G36 O$POU.Variable$ D5
```

Info.

- For CNC programs, only half size alphanumeric characters can be used. You cannot use full size characters.

  In addition, symbols can be used only in global variable names.
- Separate each G-code and parameter with a half size space.
- Do not include half size spaces between the letter G of each G-code and its number or between the word X, Y, or Z and its number.
- The portion enclosed in parentheses ( ) is regarded as a comment.

8.6.2 List of G-codes

G-code is a coding system used for machine NC programming. The GM1 controller complies with the"Din66025"standards.

Each G-code operation will be described later.

G-code	Function	Description	Remarks
G00	Fast-forward linear interpolation	Executes rapid linear interpolation.	End point: X, Y, Z, Velocity: FF, Acceleration: FE
G01	Linear interpolation	Executes linear interpolation	End point: X, Y, Z, Velocity: F, Acceleration: E
G02	Circular interpolation (clockwise)	Executes circular interpolation (clockwise)	End point: X, Y, Z Center point: I (X), J (Y), K (Z) Radius: R
G03	Circular interpolation (counterclockwise)	Executes circular interpolation (counterclockwise)	
G04	Dwell time	Sets a time to wait until next movement is started	Time specified in seconds: T
G15	Plane specification (X/Y)· 2D mode switch	Deactivates 3D mode, and activates 2D mode for XY-plane specification.	Default
G16	Arbitrary plane specification	Specifies arbitrary plane with normal vector (I, J, K).	Activate 3D mode.(Note 2)
G17	XY plane specification	Executes circular interpolation in XY plane.	
G18	XZ plane selection	Executes circular interpolation in XZ plane.	
G19	YZ plane selection	Executes circular interpolation in YZ plane.	
G20	Conditional jump	Jumps to the line number specified by the G code and executes the described	Jump condition (K) Jump target line (L)

G-code	Function	Description	Remarks
		content in a loop unless the jump condition is other than 0.	Writing of jump conditional expression(Note 1)
G36	Variable setting	Writes a value to the variable. The write variable can be used to specify the number of jumps for G20. Written to the internal variable unless a variable is specified. (The internal variable is 32-bit variable type: 0 to 4294967295.)	Set value (D) Set variable (O)
G37	Variable increment/ decrement	Increments or decrements the variable set with G36 by the specified value. Applies to the internal variable unless a variable is specified.	Increment/decrement value (D) Increment/decrement variable (O)
G40	End of tool radius correction	Ends the tool radius correction by SMC_ToolRadiusCorr.	
G41	Start of tool radius correction to left in direction of motion	Starts tool radius correction by SMC_ToolRadiusCorr to the left in the direction of motion.	Tool radius (D)
G42	Start of tool radius correction to right in direction of motion	Starts tool radius correction by SMC_ToolRadiusCorr to the right in the direction of motion.	
G43(Note 1)	Tool length correction	Executes tool length correction by SMC_ToolLengthCorr.	Tool length in x-, y-, or z-axis direction (I,J,or K)
G50	End of smoothing and arc correction	Finishes the smoothing and arc correction processing of SMC_SmoothPath and SMC_RoundPath.	
G51	Start smoothing the path	The path smoothing process by SMC_SmoothPath starts.	Radius of curvature of the spline(D)
G52	Start arc correction between paths	The arc correction process between paths by SMC_RoundPath starts.	Radius of arc(D)
G53	Coordinate conversion resetting	Returns to the reference coordinate system.	
G54	Absolute coordinate conversion	Converts coordinates from the reference coordinate system using an absolute value.	Coordinate system shift values (X, Y, Z) Coordinate system rotation values (A, B, C) Scaling factors (I, J, K)(Note 1)
G55	Relative coordinate conversion	Converts coordinates from the current coordinate system using a relative value. Combined use of G54 and G55 and use of multiple G55 codes are possible.	
G56	Coordinate reference point setting(Note 1)	Sets the current orientation and position of the reference coordinate system in the specified coordinate system.	
G75	Timing synchronization	Synchronizes decoding and CNC operation timing.	
G90	Absolute coordinate specification	Specifies target coordinates as absolute coordinates.	If any of G90 and G91 are not specified, the program runs with G90 absolute coordinates.
G91	Relative coordinate specification	Specifies target coordinates as relative coordinates	

G-code	Function	Description	Remarks
G92	Start position specification	Sets the start position of CNC program operation.	
G98	Absolute coordinates specification (center point)	Specifies the center point of circular interpolation as absolute coordinates	If any of G98 and G99 are not specified, the program runs with G99 relative coordinates.
G99	Relative coordinates specification (center point)	Specifies the center point of circular interpolation as relative coordinates.	

(Note 1) Only CNC program files can be used.

(Note 2) To execute XYZ three-axis interpolation, specify 3D mode. The mode is by default put in the state selected by G15 due to tool or FB argument.

8.6.3 G00, G01: Linear Interpolation

The path moves from the current coordinates to target coordinates by linear interpolation. All the axes reach the target coordinates simultaneously. To perform continuous motion (P-point control) through interpolation control, use G01.

Setting rules for linear interpolation

- Specifying linear interpolation

G-code	Function
G00	Fast-forward linear interpolation
G01	Linear interpolation

- Parameters set for linear interpolation

Parameter name	Input value
X-axis	X xxx (xxx: target coordinate)
Y-axis	Y xxx (xxx: target coordinate)
Z-axis	Z xxx (xxx: target coordinate) * In the 2D mode, Z-axis is not subject to interpolation control and thus is controlled at a specified velocity.
Velocity	G00 FF xxx (xxx: Composite velocity [u/sec]) G01 F xxx (xxx: Composite velocity [u/sec])
Acceleration / deceleration	G00 EF xxx (xxx > 0: Acceleration [u/sec^2]), (xxx < 0: Deceleration [u/sec^2]) G01 E xxx (xxx > 0: Acceleration [u/sec^2]), (xxx < 0: Deceleration [u/sec^2]) * When xxx = 0, an error occurs.

Example: 2-axis linear interpolation

Examples of setting linear interpolation in the XY-plane are shown below. In this example, target coordinates are set as absolute coordinates.

[Setting example]

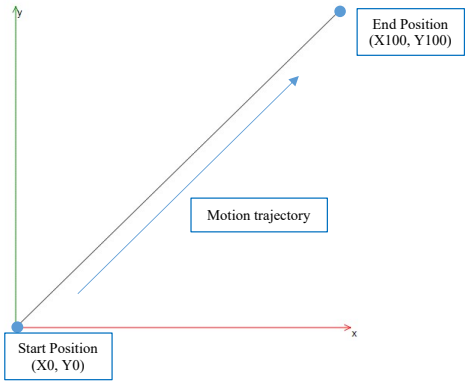
```
G-code example:
N01 G01 X100 Y100 F100 E500 E-500
```

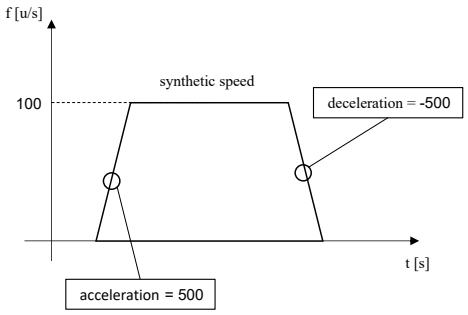
- Explanation of G-code

  N01: The X-axis and Y-axis are specified to form an XY-plane. Linear interpolation can be set in the XY-plane according to the following values.

  Current position (X0, Y0), end point (X100, Y100)

  Velocity 100 [u/sec], Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]





Example: 3-axis linear interpolation

Examples of setting linear interpolation in the XYZ-plane are shown below. In this example, target coordinates are set as absolute coordinates.

[Setting example]

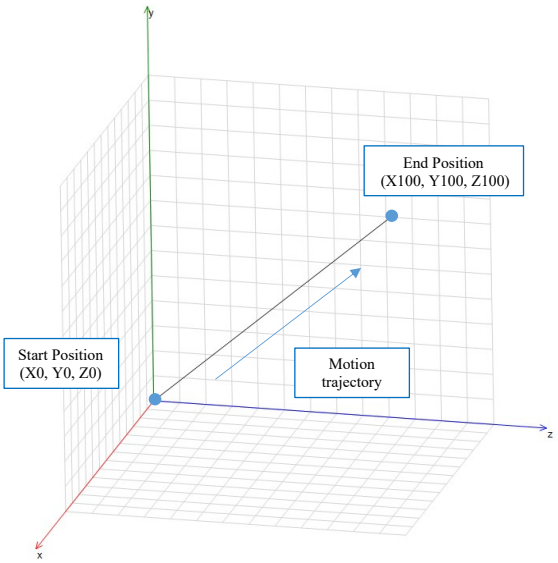
G-code example:
N11 G01 X100 Y100 Z100 F100 E500 E-500

- Explanation of G-code

  N11: Linear interpolation is performed in the XYZ-plane according to the following values.

  Current value (X0, Y0, Z0), end point (X100, Y100, Z100)

  Velocity 100 [u/sec], Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]



Info.

- To perform 3-axis linear interpolation, 3D mode must be activated.
  For details on how to activate 3D mode, refer to "8.6.6 G15, G16, G17, G18, G19: Plane Specification".

8.6.4 G02, G03: Circular Interpolation

The path moves from the current coordinates to target coordinates by circular interpolation. All the axes reach the target coordinates simultaneously.

Setting rules for circular interpolation

- Specifying a rotation direction for circular arc

  The rotational direction of a circular arc can be switched by specifying a G-code.

G-code	Function
G02	Circular interpolation (clockwise)
G03	Circular interpolation (counterclockwise)

- Parameters set for circular interpolation

Parameter name		Input value
Target coordinates	X-axis	X xxx (xxx: Target coordinate)
	Y-axis	Y xxx (xxx: target coordinate)
	Z-axis	Z xxx (xxx: target coordinate)
Center point	X-axis	I xxx (xxx: Center point coordinate)
	Y-axis	J xxx (xxx: Center point coordinate)
	Z-axis	K xxx (xxx: Center point coordinate)
Radius		R xxx (xxx: Circle radius)
Velocity		F xxx (xxx: Composite velocity [u/sec])
Acceleration / deceleration		E xxx (xxx > 0: Acceleration [u/sec^2]), (xxx < 0: Deceleration [u/sec^2]) * When xxx = 0, an error occurs.

Info.

- For circular arcs, it is necessary to specify a start point and target position, as well as a radius (R) or center point (I, J, K). Set either a center point or radius.
- The target position and center point of circular interpolation can be specified as relative coordinates or absolute coordinates. For details on how to set relative coordinates or absolute coordinates, refer to "Setting rules for coordinate specification".

- Specifying the coordinate plane for circular interpolation

  For circular interpolation, it is necessary to determine the plane targeted for interpolation. If no plane is specified, the XY-plane will be set by default.

  The coordinate plane can be switched according to the G-code specification. For details of the setting method, refer to "Setting rules for plane selection".

- For axes untargeted for interpolation

  Note that if a control amount is specified for the axis that is regarded as an axis untargeted for circular interpolation because of the plane specification, "helical interpolation" will be set. For details on helical interpolation, refer to "Example: Helical interpolation".

Example: Circular interpolation

■ Example of setting circular interpolation with center point specified

Examples of setting circular interpolation with G-code are shown below. In these examples, target coordinates are set as absolute coordinates and center point coordinates are set as relative coordinates.

For circular interpolation, it is necessary to specify the coordinate plane targeted for interpolation. When using circular interpolation, specify a coordinate plane.

- Circular interpolation by center-point approach (XY-plane)

  [Setting example 1] Locus moving along semicircle

```
G-code example:
N00 G17
N10 G02 X100 Y0 I50 J0 F100 E500 E-500
```

  - Explanation of G-code

    N00: An XY-plane is selected. (This can be omitted when there is no need to change the plane.)

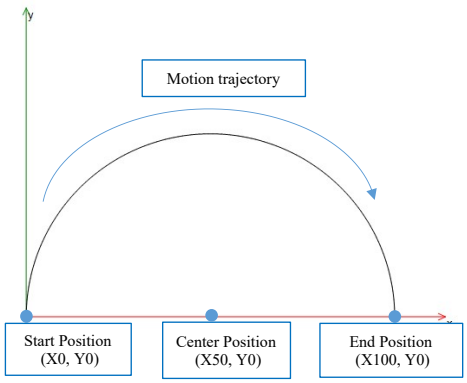
    N10: Circular interpolation is performed in the XY-plane according to the following values.

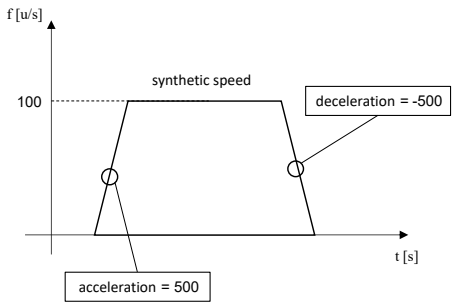
    Current value (X0, Y0), end point (X100, Y0)

    Center point (X50, Y0)

    Velocity 100

    Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]





- Circular interpolation by center-point approach (round circle)

  A round circle can be specified by omitting an end point or entering the same coordinates for the start point and end point as below. (For center point specification only)

  [Setting example 2] Locus moving along perfect circle (When end point is omitted)

```
G-code example:
N00 G17
N20 G02 I50 J0 F100 E500 E-500
```

  [Setting example 3] Locus moving along perfect circle (When start point = end point)

```
G-code example:
N00 G17
```

```
N20 G02 X0 Y0 I50 J0 F100 E500 E-500
```

- Explanation of G-code

  N00: An XY-plane is selected.

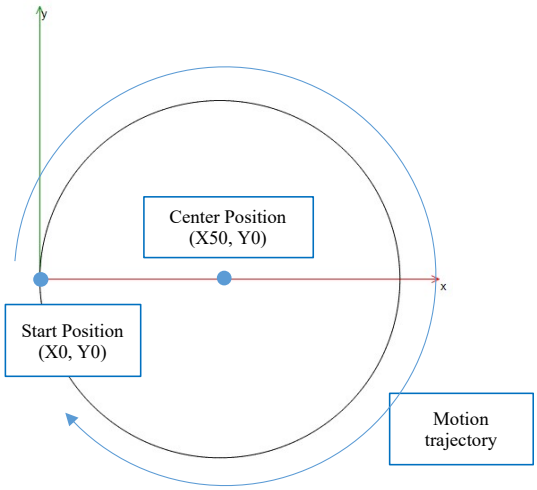
  N20: Circular interpolation is performed clockwise in the XY-plane according to the following values.

  Current value (X0, Y0), end point (X0, Y0 or omitted)

  Center point (X50, Y0)

  Velocity 100

  Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]



- Circular interpolation by radius approach (XY-plane)

  [Setting example 4] Locus moving along semicircle

```
G-code example:
N00   G17
N30   G02 X100 Y0 R50 F100 E500 E-500
```

- Explanation of G-code

  N03: An XY-plane is selected.

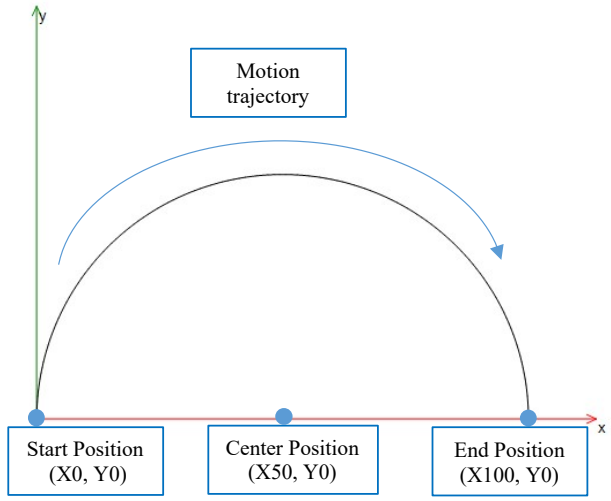
  N04: Circular interpolation is performed in the XY-plane according to the following values.

  Current value (X0, Y0), end point (X100, Y0)

  Radius 50

  Velocity 100

  Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]



- Circular interpolation by center-point approach (ZX-plane)

  [Setting example 5] Locus moving along semicircle

```
G-code example:
N00 G18
N40 G03 X100 Z0 I50 K0 F100 E500 E-500
```

  - Explanation of G-code

    N00: A ZX-plane is selected.

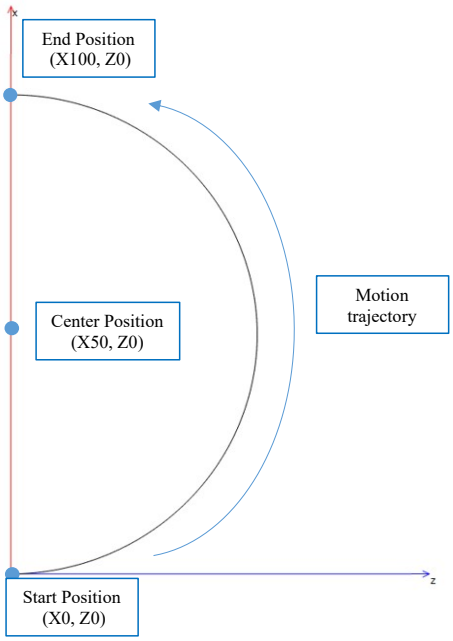
    N40: Circular interpolation is performed counterclockwise in the ZX-plane according to the following values.

    Current value (X0, Z0), end point (X100, Z0)

    Center point (X50, Z0)

    Velocity 100

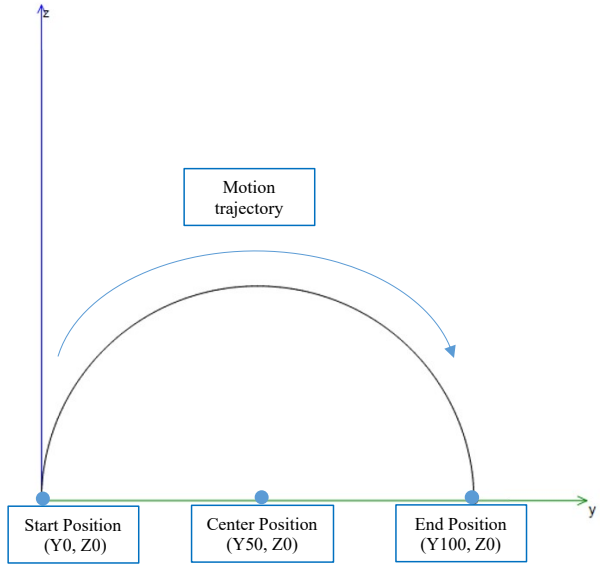
    Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]



- Circular interpolation by center-point approach (YZ-plane)
  [Setting example 6] Locus moving along semicircle

G-code example: N00 G19 N50 G02 Y100 Z0 J50 K0 F100 E500 E-500

  - Explanation of G-code
    N00: An YZ-plane is selected.
    N50: Circular interpolation is performed clockwise in the YZ-plane according to the following values.
    Current value (Y0, Z0), end point (Y100, Z0)
    Center point (Y50, Z0)
    Velocity 100
    Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]


z
Motion trajectory
y
Start Position (Y0, Z0)
Center Position (Y50, Z0)
End Position (Y100, Z0)

Example: Helical interpolation

Axes other than those in the target plane are selected and used, helical interpolation will be performed.

[Setting example]

```
G-code example:
N00 G17
N60 G02 X0 Y0 Z100 I50 J0 F100 E500 E-500
```

- Explanation of G-code

  N00: An XY-plane is selected.

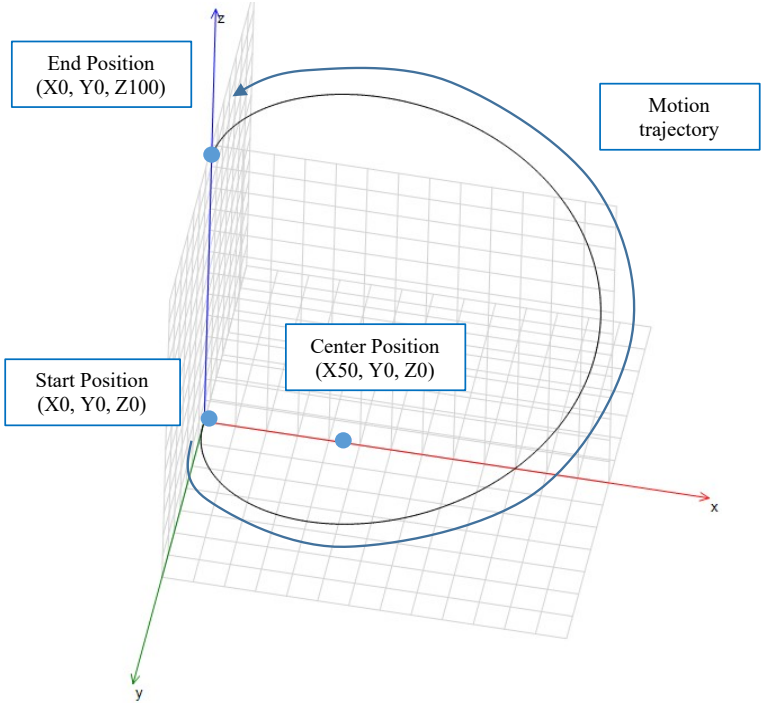
  N60: Circular interpolation is performed clockwise in the XY-plane according to the following values.

  Current value (X0, Y0, Z0), end point (X0, Y0, Z100)

  Center point (X50, Y0)

  Velocity 100

  Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]



8.6.5 G04: Dwell Time

Dwell time is a time to wait until next processing is executed. It is used for purposes such as waiting for a particular operation.

Setting rules for dwell time

- Specifying dwell time

G-code	Function
G04	Dwell time

- Parameter set for dwell time

Parameter name	Input value
Dwell time	T xxx (xxx: Dwell time [sec])

Example: Dwell time setting

[Setting example]

```
G-code example:
N00 E500 E-500
N10 G01 X100 Y100 F100
N20 G04 T1.5
N30 G01 X100 Y0 F100
```

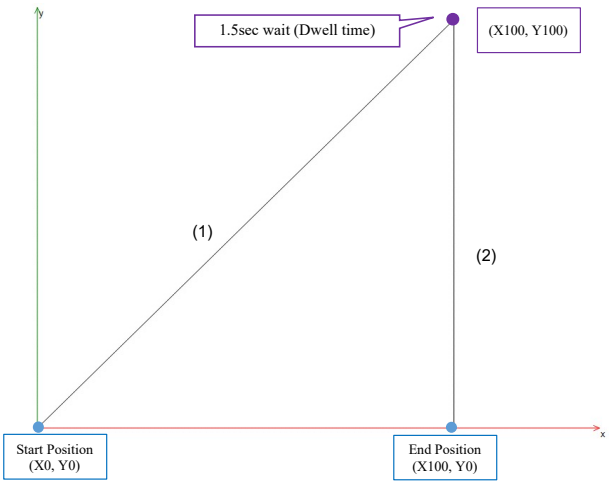
- Explanation of G-code

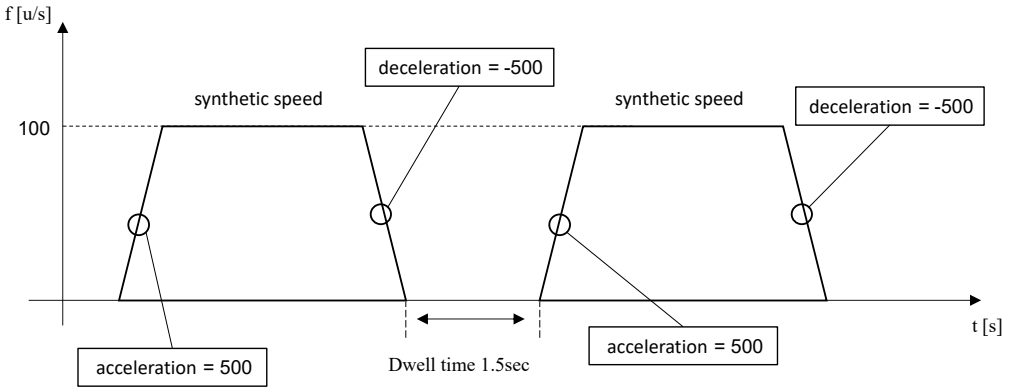
  N00: Acceleration and deceleration (acceleration 500 [u/sec^2] and deceleration -500 [u/sec^2]) are set collectively.

  N10: Linear interpolation (X100, Y100) is performed. (Section (1) in the figure below)

  N20: The system waits for the dwell time (1.5 seconds).

  N30: Linear interpolation (X100, Y0) is performed. (Section (2) in the figure below)





8.6.6 G15, G16, G17, G18, G19: Plane Specification

A plane in which circular interpolation motion is performed can be specified.

If no plane is specified, the XY-plane will be set by default. Once a plane is set, motion continues on the same specified plane unless another plane is specified. The plane setting is related to linear interpolation, circular interpolation, and coordinate conversion functions.

Setting rules for plane selection

- Specifying plane

G-code	Function
G15	Deactivates 3D mode, and activates 2D mode for XY-plane specification.
G16	Activates 3D mode in the plane specified using normal vector (I, J, K).
G17	Activate 3D mode with the XY-plane specified.
G18	Activate 3D mode with the XZ-plane specified.
G19	Activate 3D mode with the YZ-plane specified.

Info.

- With the GM1 Controller, operation modes available for CNC control are 2D mode and 3D mode.

  In 2D mode, the system manages velocity and acceleration only along the x-axis and y-axis. In 3D mode, the system manages velocity and acceleration along the x-axis, y-axis, and z-axis.

  Operation in 3D mode is allowed if G16, G17, G18, or G19 is executed and the following conditions are satisfied.
  - To read the CNC program as an external file, execute SMC_ReadNCFile2 with b3DMode:=TRUE.
  - If the CNC program is created on the CNC editor, select **Application**>**CNC program**>**Properties** in this order by right-clicking in the Device tree of GM Programmer and select the [3D mode] checkbox on the displayed [CNC tab].
- Parameters used for plane specification with G16 normal vector

Parameter name		Input value
Normal vector(Note 1)	X-axis component	I xxx (xxx : -2147483648 to 2147483647)
	Y-axis component	J xxx (xxx : -2147483648 to 2147483647)
	Z-axis component	K xxx (xxx : -2147483648 to 2147483647)

(Note 1) All I, J, and K values need to be written. However, when all I, J, and K are set to 0, a build error occurs.

Info.

- If only an x-axis component is set like I=1, J=0, and K=0, the YZ-plane is specified and operation will be similar to that with G19. Likewise, operation will be similar to that with G18 when I=0, J=1, and K=0 and be similar to that with G17 when I=0, J=0, and K=1.
- When two axis components of the normal vector are set, a plane is determined according to the specified normal vector components, and motion is performed in the plane.
  If the settings are I=0, J=1, and K=1, for example, a plane inclined at an angle of 45 degrees with respect the x-axis is specified.

Example: Plane specification

- Examples of setting plane with normal vector specification

```
G-code example:
N00 F10 E100 E-100
N10 G91
N20 G16 I0 J0 K1
N30 G02 X10 Y0 R5
N40 G16 I0 J1 K1
N50 G02 X10 Y0 R5
N60 G16 I0 J1 K0
N70 G02 X10 Y0 R5
```

- Explanation of G-code

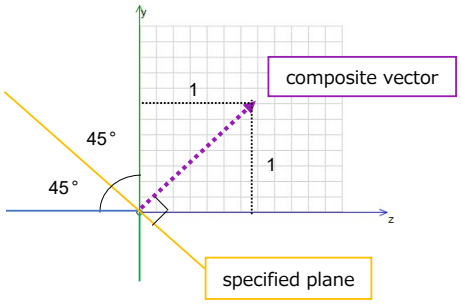
  N00: Velocity, acceleration, and deceleration (velocity 10 [u/sec], acceleration 100 [u/sec^2] and deceleration -100 [u/sec^2]) are set collectively.

  N10: Relative coordinate specification is set.

  N20: A plane is specified with normal vector components I=0, J=0, and K=1.
  Since the normal vector faces along the z-axis, the XY-plane is specified.

  N30: Arc correction is performed in the XY-plane according to the following values. (Section (1) in the figure below)
  Current position (X0, Y0, Z0), end point (X10, Y0, Z0), radius 5

  N40: A plane is specified with normal vector components I=0, J=1, and K=1.
  Since the size of the y- and z-axis components of the normal vector is set to 1, a plane inclined at 45 degrees is specified.



N50: Arc correction is performed in the plane inclined at 45 degrees according to the following values. (Section (2) in the figure below)

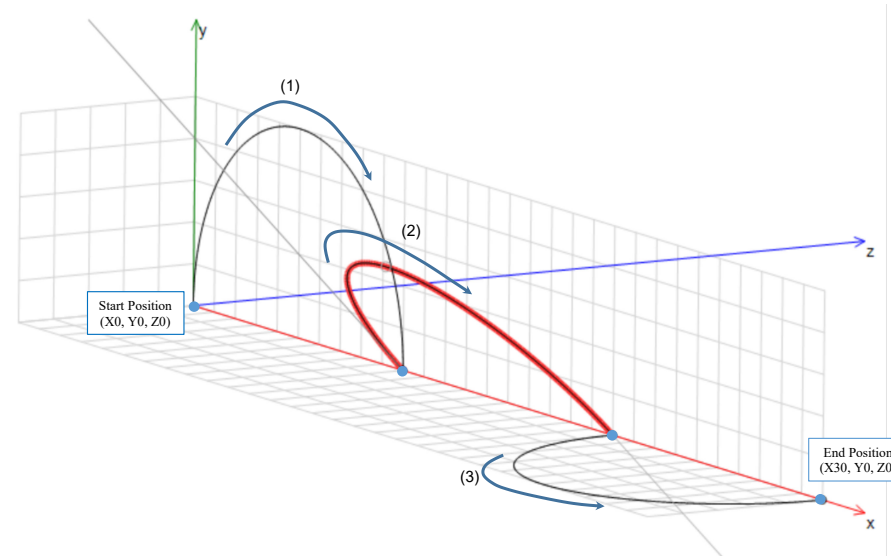
Current position (X10, Y0, Z0), end point (X20, Y0, Z0), radius 5

N60: A plane is specified with normal vector components I=0, J=1, and K=0.

Since the normal vector faces along the y-axis, the XZ-plane is specified.

N70: Circular interpolation is performed in the XZ-plane according to the following values. (Section (3) in the figure below)

Current position (X20, Y0, Z0), end point (X30, Y0, Z0), radius 5



- Examples of setting circular interpolation
  For the setting examples, refer to "8.6.4 G02, G03: Circular Interpolation".
- Examples of setting coordinate conversion
  For the setting examples, refer to "8.6.11 G53, G54, G55, G56: Coordinate Conversion".

8.6.7 G20, G36, G37: Jump and Loop Process

With a jump executed by G-code, a content written in the CNC program can be repeated.
You can specify the number of repetitive runs by setting a jump condition and a loop counter.
To use such a counter, configure counter settings and increments or decrements a counter variable using G36 and G37.
You can also perform branching by combining a jump condition and a jump label.

Setting rules for jump

- Specifying jump condition

G-code	Function
G20	Conditional jump
G36	Variable settings
G37	Variable increment/decrement

- Parameters used for jump

G-code	Parameter name	Input value
G20	Jump condition	K xxx While K is other than 0, a jump process is executed by G20. The jump condition can be handled from the program by the use of a variable value declared for K. If K is not defined, a CNC internal variable acts as a jump condition and is configured using G36 and G37.
	Jump target line	L xxx Specifies a jump destination, the line number (N) in the CNC program.
	Example: Execution of a jump to N010 when the CNC internal variable for counter is not 0: N020 G20 L10 Example: Execution of a jump to N010 when the global variable g_x is not 0: N020 G20 L10 Kg_x	
	Jump label, jump index(Note 1)	L!x、L?x From the jump index (eg, L? 4), the jumping process is performed from the given line with the jump label (eg, L! 4). The row that grants a jump label must be behind the Jump index line.
	Example: Execution of a jump from L?3 to L!3 line: N020 G20 L?3 ... N080 G01 X0 Y0 L!3	
G36	Value that is specified	D xxx (CNC internal variable) Sets the counter to the specified value. The O variable or CNC internal variable is changed.
	Variable that is written	O xxx(global variable(Note 2))

G-code	Parameter name	Input value
		Specifies a global variable to which a value is assigned by the parameter D. If O is not used, the value is assigned to the CNC internal variable.
	Example: Set the CNC internal variable for counter to 3: N020 G36 D3 Example: Set the global variable g_x to 3: N020 G36 Og_x D3	
G37	Incremental value	D xxx (CNC internal variable) Increments the counter by the specified value. If a negative value is specified, the counter is decremented by the value. The O variable or CNC internal variable is changed.
	Variable that is incremented	O xxx (global variable) Specifies a global variable that is incremented or decremented by the value of the parameter D. If O is not used, the CNC internal variable is incremented or decremented.
	Example: Reduce CNC internal variables for counter by 1: N050 G37 D-1 Example: Reduce global variables g_x by 1: N050 G37 Og_x D-1	

(Note 1) It can be used only for "8.6.18 CNC Program File".

(Note 2) If the CNC program is created with SMC_OUTQUEUE type, global variables cannot be used. (The same is true for other G code.)

Info.

- The CNC internal variable is handled as one common variable.
  If the G20 loop process is used two or more times, assign a value to the CNC internal variable before the loop process is executed.
- To use a CNC internal variable, use it of the 32-bit variable type in the range of 0 to 4294967295.
- To use a global variable counter, declare a variable of the WORD type.
  The variable that is used must have a positive integer in the range from 0 to 65535.
- A parallel or nested use of multiple G20 commands is possible. However, use separate global variables for respective jump conditions.
  Shared use of CNC internal variables and other variables may result in improper operation such as an infinite loop.

Example: Loop Process with a conditional jump

- Loop Process with a conditional jump
  [Setting example 1]

```
G-code example:
N000 G91
N010 G01 X10 Y5 F5
N020 G20 L10 K$K$
```

```
Global variable declaration:(GVL)
K : REAL := 1;
```

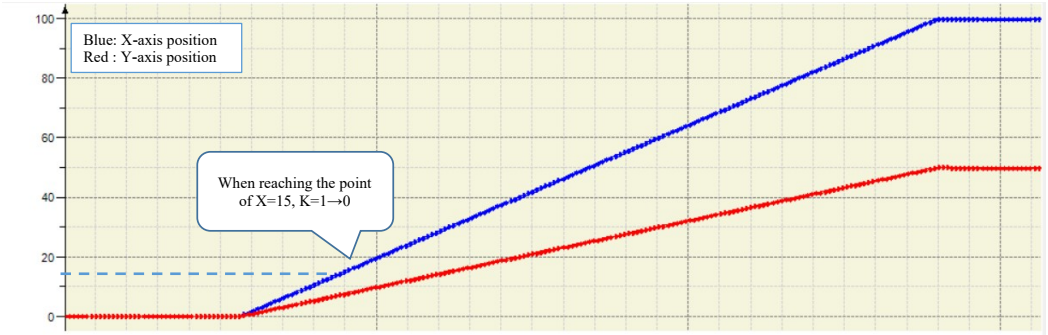
```
Control section (excerpt):
IF X_Drive.fSetPositon >= 15 THEN
K := 0;
END_IF
```

- Explanation of G-code

  N000: Relative coordinate specification is set.

  N010: The path is moved (linearly interpolated) from the current value by (X10, Y5).

  N020: If K<>0, the program returns to N010 and repeats the process at N010 and subsequent lines.



When the path reaches the position X = 15, K is changed to 0 and the jump process ends, and in the meantime the CNC program is decoded.

As a result, the target position is set at a point out of sync with the time when jump condition K = 0.
In order to make the point in sync with the timing, decoding must be synchronized with CNC operation timing using G75.

- Loop process with a conditional jump (with synchronized timing)

  [Setting example 2]

```
G-code example:
N000 G91
N010 G01 X10 Y5 F5
N020 G75
N030 G20 L10 K$K$
```

```
Global variable declaration:(GVL)
K : REAL := 1;
```

```
Control section (excerpt):
IF X_Drive.fSetPositon >= 15 THEN
K := 0;
END_IF
```

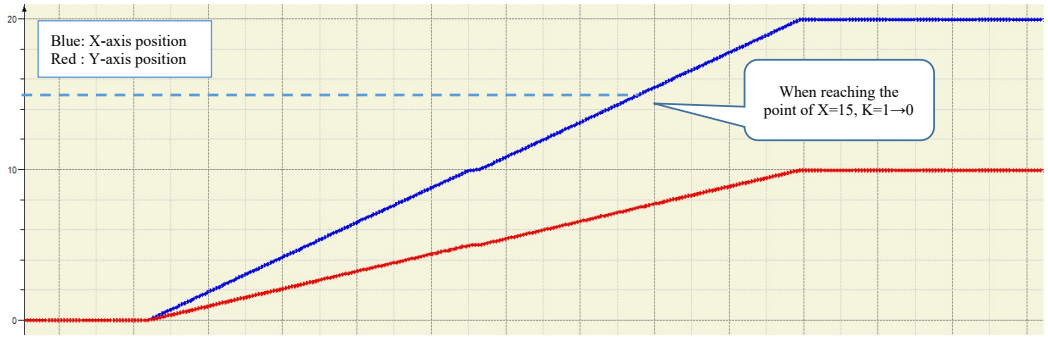
- Explanation of G-code

  N000: Relative coordinate specification is set.

  N010: The path is moved (linearly interpolated) from the current value by (X10, Y5).

  N020: CNC program decoding and interpolation operation process are waited.

N030: If K<>0, the program returns to N010 and repeats the process at N010 and subsequent lines.



When the path reaches the position X = 15, K is changed to 0 and the jump process ends.

While interpolation operation is performed, decoding is put in the wait state. Thus, the loop process by G20 ends when operation at the time of jump condition K = 0 is completed. However, to synchronize timing by G75, C-point control is executed to make a pause every time.

Example: Loop Process Using Variable

- Loop process using a CNC internal variable

  [Setting example 3]

```
G-code example:
N000 G91
N010 G01 X5 Y5 F5
N020 G36 D3
N030 G01 X2
N040 G01 Y3
N050 G37 D-1
N060 G20 L30
N070 G01 X5 Y5
```

  - Explanation of G-code

    N000: Relative coordinate specification is set.

    N010: The path is moved (linearly interpolated) from the current value by (X+5, Y+5).

    N020: The CNC internal variable is set to 3 by G36.

    N030: The path is moved (linearly interpolated) from the current value by (X+2, Y+0).

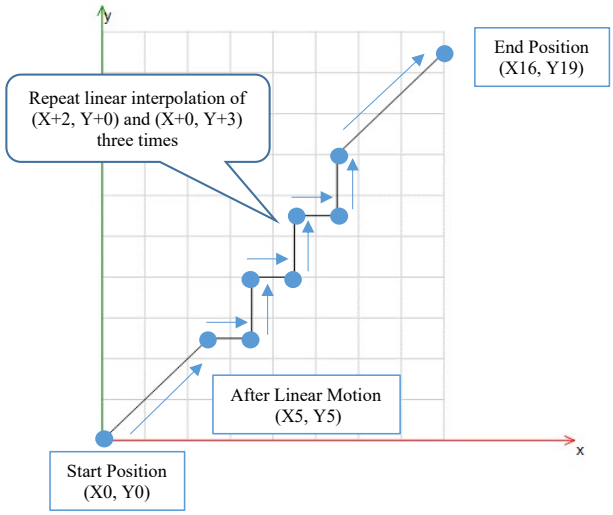
    N040: The path is moved (linearly interpolated) from the current value by (X+0, Y+3).

    N050: By G37, -1 is added to the CNC internal variable.

    N060: The program returns to N030 and repeats the process at N030 and subsequent lines until the CNC internal variable reaches 0.

    At the time when the CNC internal variable reaches 0, the program transitions to N070 without executing the jump.

    N070: The path is moved (linearly interpolated) from the current value by (X+5, Y+5).



- Loop process using global variable counter
  [Setting example 4]

```
G-code example:
N000 G91
N010 G01 X5 Y5 F5
N020 G36 O$O$ D3
N030 G01 X2
N040 G01 Y3
N050 G37 O$O$ D-1
N060 G20 L30 K$O$
N070 G01 X5 Y5
```

```
Global variable declaration:(GVL)
O : WORD;
```

  - Explanation of G-code

    N000: Relative coordinate specification is set.

    N010: The path is moved (linearly interpolated) from the current value by (X+5, Y+5).

    N020: The global variable O is set to 3 by G36.

    N030: The path is moved (linearly interpolated) from the current value by (X+2, Y+0).

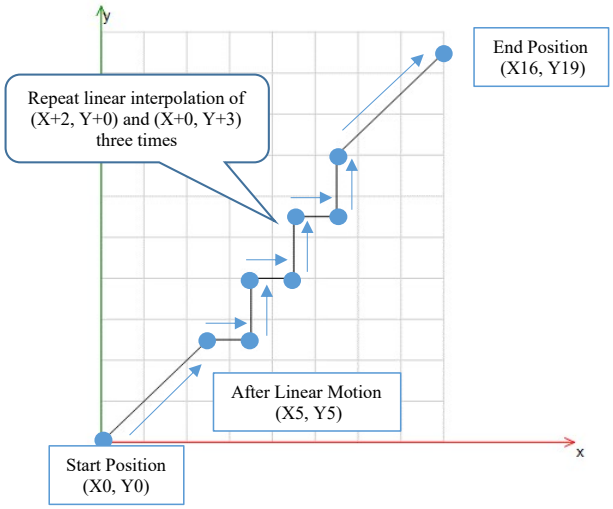
    N040: The path is moved (linearly interpolated) from the current value by (X+0, Y+3).

    N050: By G37, -1 is added to the global variable O.

    N060: The program returns to N030 and repeats the process at N030 and subsequent lines until the global variable O reaches 0.

    At the time when the variable O reaches 0, the program transitions to N070 without executing the jump.

    N070: The path is moved (linearly interpolated) from the current value by (X+5, Y+5).



- Parallel use of loop processes [Unacceptable example]

```
N000 G91
N010 G01 X5 Y5 F5
N020 G36 D3
N030 G01 X2
N040 G01 Y3
N050 G37 D-1
N060 G20 L30
N070 G01 X10
N080 G01 Y20
N090 G37 D-1
N100 G20 L70
N110 G01 X5 Y5
```

Description of unacceptable example operation

- At N020, the CNC internal variable is set to 3.
- After a loop process at N030 to N060, the CNC internal variable is changed to 0.
- When a loop process at N070 to N100 is executed, the CNC internal variable is set to 0-1=-1 at N090. The CNC internal variable will not reach 0 hereafter, and the end condition for G20 will not be satisfied, causing an infinite loop.

- Parallel use of loop processes [Acceptable example]

```
N000 G91
N010 G01 X5 Y5 F5
N020 G36 D3
N030 G01 X2
N040 G01 Y3
N050 G37 D-1
N060 G20 L30
N070 G36 D3
N080 G01 X10
N090 G01 Y20
N100 G37 D-1
```

```
N110 G20 L80
N120 G01 X5 Y5
```

Description of acceptable example operation

- A command at N70 is added, and a new number of repetitive run is specified for the CNC internal variable, which has reached 0. After execution of a loop process at N080 to N110, the program transitions to N120.

Example: Repeated processing by jump label

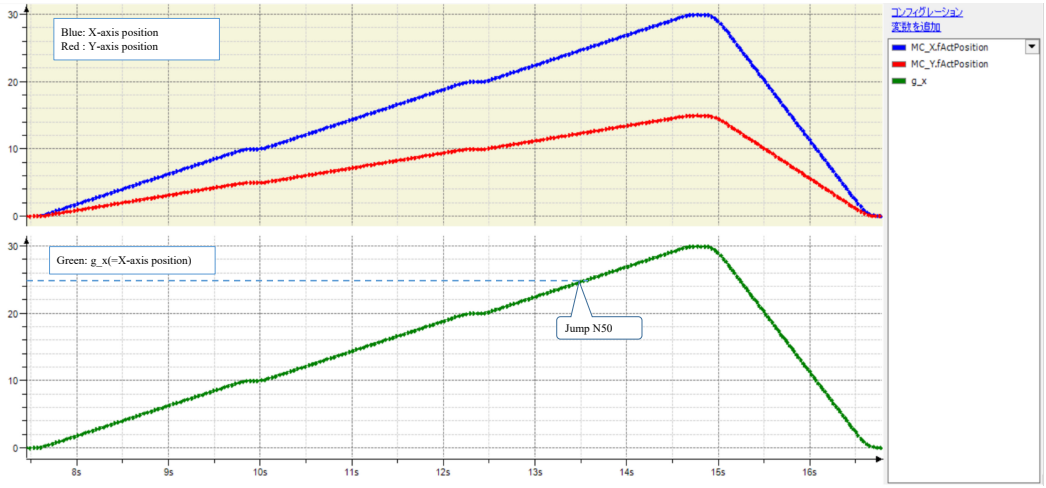
- [Setting example]

```
G-code example:
N000 G91
N010 G01 X10 Y5 F5
N020 G75
N030 G20 L?1 K{$g_x$>=25}
N040 G20 L10
N050 G90 L!1
N060 G01 X0 Y0 F20
```

```
Global variable declaration:(GVL)
g_x : LREAL := 0;
```

```
Control unit (excerpt):
g_x := X_Drive.fSetPositon;
```

- Explanation of G-code
  N000: Relative coordinate specification is set.
  N010: Move (X10, Y5) from the current position (straight interpolation).
  N020: Wait for CNC decoding and interpolation operation.
  N030: In the case of g_x>=25 , jump to the L!1(N050).
  N040: Jump to N010 and repeat the operation from the N10 repeatedly.
  N050: Absolute coordinate specification is set.
  N060: Move from the current position to (X0, Y0).



During the x<25, N040 is executed and returns to the N010 because the Jump condition (g_x>=25) of the N030 are not satisfied.

The Jump condition are satisfied at the point of X = 25, jump to the L!1(N050) and perform the processing afterwards.

However, to synchronize timing by G75, C-point control is executed to make a pause every time.

Info.

- The jump label (L?1), jump index (L!1), and jump condition by variable ({global variable condition}) can be written only for CNC program files described in "8.6.18 CNC Program File".

Example: Conditional branching by jump label

- [Setting example]

```
G-code example:
N000 G91
N010 G01 Z10 F50
N020 G75
N030 G20 L?1 K {$ProgramNo$ <> 1}
N040 G02 X0 Y0 I10 J0
N050 G20 L?2 K {$ProgramNo$ <> 2} L!1
N060 G02 X0 Y0 I20 J0
N070 G20 L?3 K {$ProgramNo$ <> 3} L!2
N080 G02 X0 Y0 I30 J0
N090 G20 L010 K {$ProgramNo$ <> 9} L!3
N100 G90
N110 G01 X0 Y0 Z0
```

```
Global variable declaration:(GVL)
ProgramNo : INT := 0;
```

```
Control section (excerpt):
IF Z_Drive.fSetPosition >= 100 THEN
```

```
GVL.ProgramNo:=9;
ELSIF Z_Drive.fSetPosition >= 80 THEN
GVL.ProgramNo:=2;
ELSIF Z_Drive.fSetPosition >= 50 THEN
GVL.ProgramNo:=3;
ELSIF Z_Drive.fSetPosition >= 20 THEN
GVL.ProgramNo:=1;
END_IF
```

- Explanation of G-code

  N000: Relative coordinate specification is set.

  N010: The tool is moved relatively (linearly interpolated) from the current position by (Z = +5).

  N020: CNC program decoding and interpolation operation process are waited.

  N030: When ProgramNo is not 1, the program jumps to L!1 (N050).

  N040: Circular interpolation (circle with a radius of 10) is performed with the center at (X10,Y0) from the current position.

  N050: When ProgramNo is not 2, the program jumps to L!2 (N070).

  N060: Circular interpolation (circle with a radius of 20) is performed with the center at (X20,Y0) from the current position.

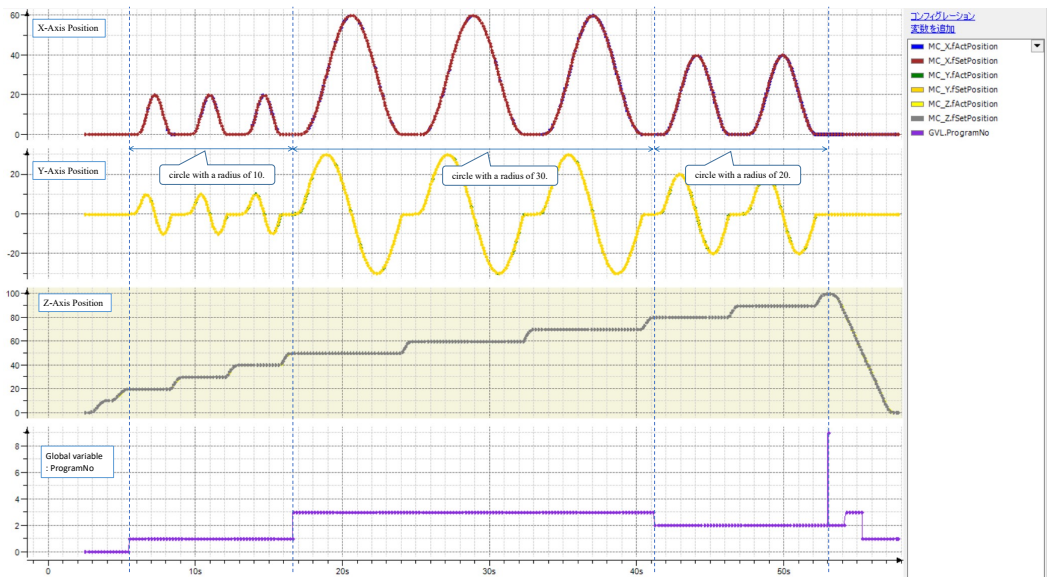
  N070: When ProgramNo is not 3, the program jumps to L!3 (N090).

  N080: Circular interpolation (circle with a radius of 30) is performed with the center at (X30,Y0) from the current position.

  N090: When ProgramNo is not 9, the program jumps to N010 and repeats operation from N010.

  N100: Absolute coordinate specification is set.

  N110: The tool is moved from the current position to (X0,Y0,Z0).



The program implemented in the control section changes the value of the global variable ProgramNo in the order 0, 1, 3, 2, and 9 according to the change in the value of the z-axis position.

If ProgramNo is 0, N030, N050, and N070 are set to TRUE and circular interpolation operation is not performed. Then, N090 changes to TRUE, and the program jumps to N010, and the z-axis position is incremented by +5.

If ProgramNo is 1, N030 is set to FALSE and circular interpolation operation with a radius of 10 is performed in N040. Since N050 and N070 also change to TRUE, N060 and N080 are not executed. Then, N090 changes to TRUE, and the program jumps to N010, and the z-axis position is incremented by +5.

If ProgramNo is 2, only N060 is executed. If ProgramNo is 3, only N080 is executed. Then, N090 changes to TRUE, and the program jumps to N010, and the z-axis position is incremented by +5.

If ProgramNo is 9, N030, N050, and N070 are set to TRUE and circular interpolation operation is not performed. Then, N090 changes to FALSE, the program changes to N100, and the processing ends at the absolute position (X0,Y0,Z0) in N110.

Thus, branching based on the value of the global variable ProgramNo can be achieved by combining the jump conditions and jump labels written in N030 to N090.

Info.

- Jump labels (L?n), jump indexes (L!n), and jump conditions by variables ({global variable conditions}) can be written only for CNC program files described in "8.6.18 CNC Program File".

8.6.8 G40, G41, G42: Tool Radius Correction for Path

Tool radius correction allows the programmed path to be corrected according to the radius of the tool in use without change to the CNC program.

Rules for Tool Radius Correction

- Specification of tool radius correction

To perform tool radius correction using SMC_ToolRadiusCorr, specify G40, G41, and G42. Tool radius correction converts a path in the specified range so that it is offset by the tool radius.

G-code	Function
G40	Ends tool radius correction.
G41	D > 0: Executes tool radius correction to the left in the direction of motion. D < 0: Executes tool radius correction to the right in the direction of motion.
G42	D > 0: Executes tool radius correction to the right in the direction of motion. D < 0: Executes tool radius correction to the left in the direction of motion.

- Parameters used for tool radius correction

Parameter name	Input value
Tool radius	D xxx: Tool radius to be corrected

Example: Tool radius correction in XY plane

- Apply tool radius correction to outside of the square.

[Setting example 1]

G-code example:

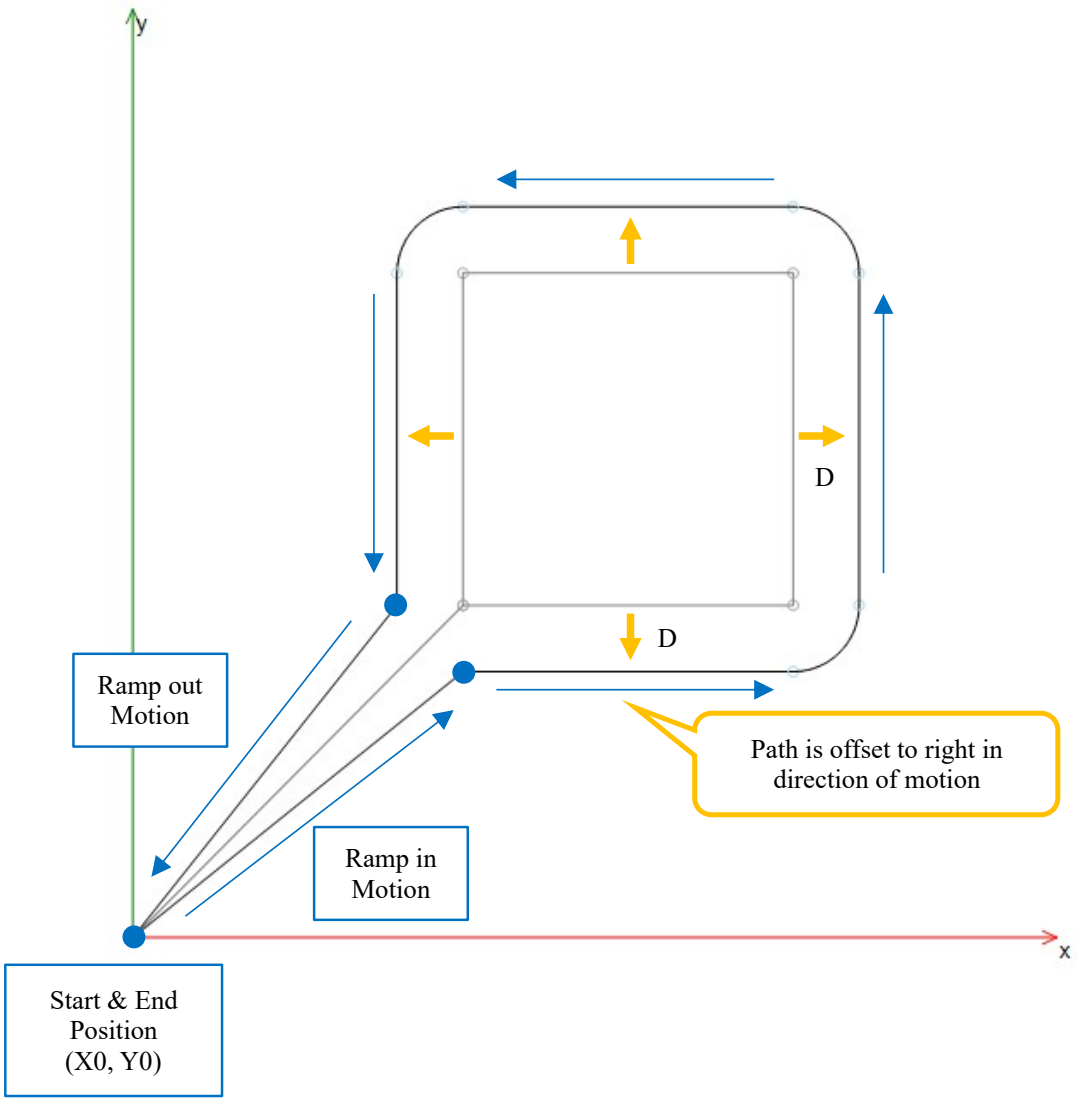
```
N000 G42 D1
N010 G01 X5 Y5 F10
N020 G01 X10 Y5
N030 G01 Y10
N040 G01 X5
N050 G01 Y5
N060 G40
N070 G01 X0 Y0
```

- Explanation of G-code

N000: Tool radius correction with a tool radius of 1 is performed from the next path. The path is corrected to the right in the direction of motion.
N010: The tool is moved (linearly interpolated) to the next start position in ramp-in motion for tool radius correction.

N020 to N050: The tool is moved (linearly interpolated) on the path corrected to the right for the tool radius (offset conversion) in the direction of motion. Between each path, circular interpolation is inserted.

N060: The application of tool radius correction ends.
N070: The tool is moved (linearly interpolated) to (X0, Y0) in ramp-out motion.



- Apply tool radius correction to inside of the square.

[Setting example 2]

G-code example:

```
N000 G17
N010 G01 Z5 F10
N020 G41 D1
N030 G01 X5 Y7
N040 G01 Y5
N050 G01 Z0
N060 G01 X10 Y5
N070 G01 Y10
N080 G01 X5
N090 G01 Y5
N100 G01 Z5
N110 G01 X7
```

```
N120 G40
N130 G01 X0 Y0
N140 G01 X0 Y0 Z0
```

• Explanation of G-code

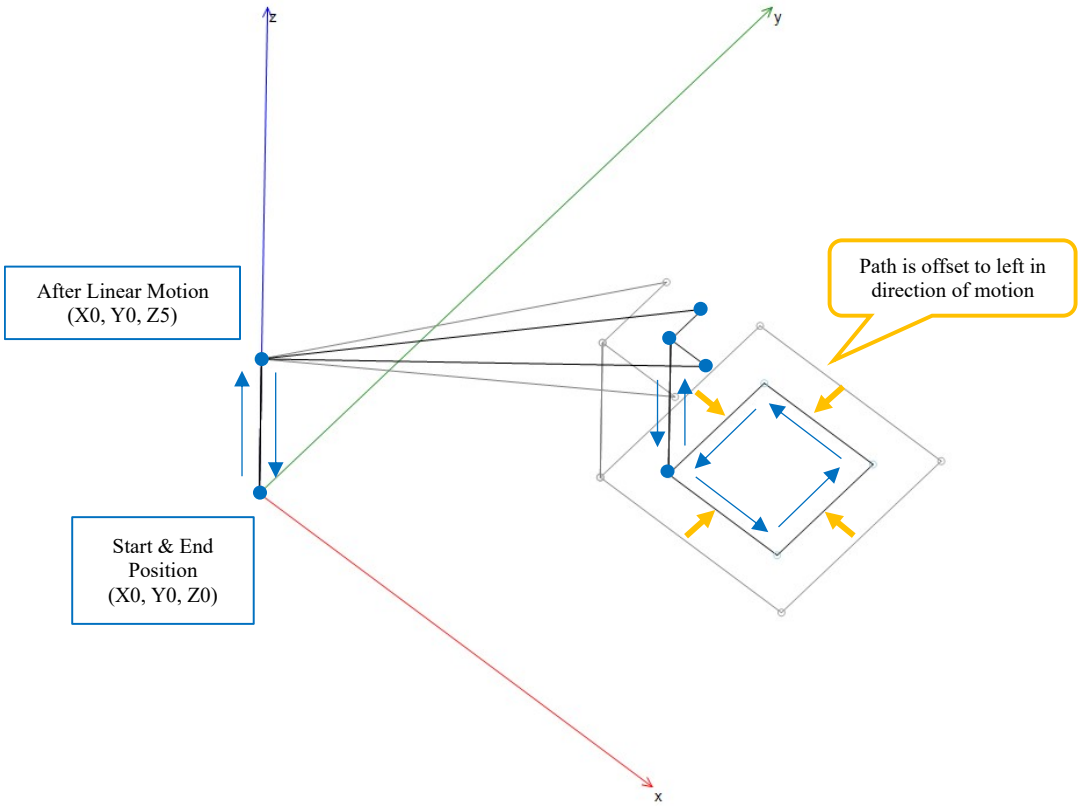
N000: 3D mode is activated for three-dimensional operation. Since the intended tool radius correction is executed in XY plane, G17 is used.
N010: The tool is moved (linearly interpolated) to (X0, Y0, Z5).

N020: Tool radius correction with a tool radius of 1 is performed from the next path. The path is corrected to the left in the direction of motion.
N030: The tool is moved (linearly interpolated) to the next start position in ramp-in motion for tool radius correction.

N040 to N110: The tool is moved (linearly interpolated) on the path corrected to the left for the tool radius in the direction of motion.

N120: The application of tool radius correction ends.
N130: The tool is moved (linearly interpolated) to (X0, Y0, Z5) in ramp-out motion.

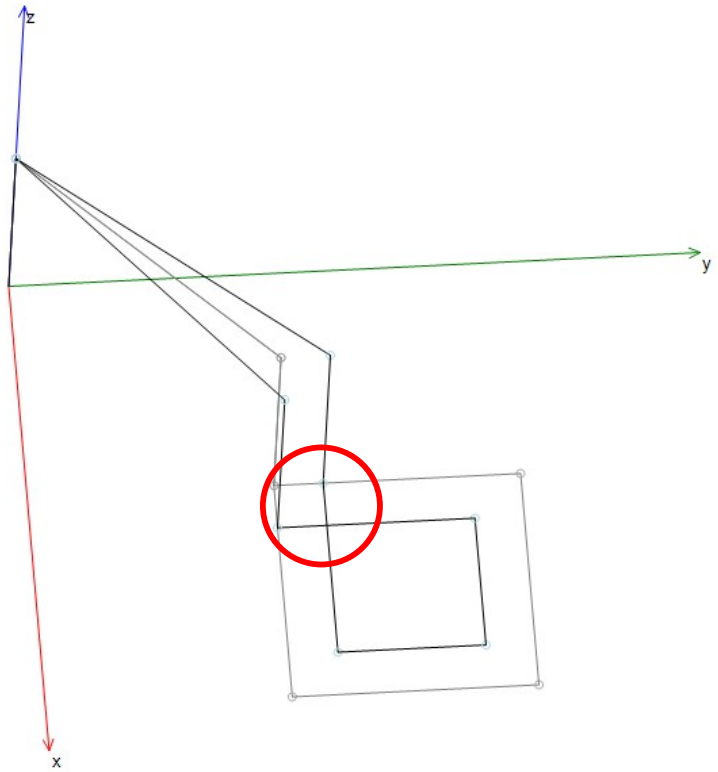
N140: The tool is moved (linearly interpolated) to (X0, Y0, Z0).



Info.

- In the case of an inner surface processing such as Setting example 2, attention must be paid to operations performed at the start and end points. These operations include, for example, setting a path for ramp-in motion at a position away from the processing section according to the offset tool radius (which corresponds to N030 in Setting Example 2). Performing tool radius correction without setting such a path will lead to motions such as locus intersections.

```
N000 G17
N010 G01 Z5 F10
N020 G41 D1
N030 G01 X5 Y5
N040 G01 Z0
N050 G01 X10 Y5
N060 G01 Y10
N070 G01 X5
N080 G01 Y5
N090 G01 Z5
N100 G40
N110 G01 X0 Y0
```



Example: Change of corrected plane

- Apply tool radius correction to the three-dimensional arc.

[Setting example]

G-code example:

```
N000 G41 D1
N010 G16 I1 J1 K1
N020 G01 X5 F10
N030 G02 X0 Y5 R5
N040 G40
N050 G01 X-1
N060 G92
N070 G42 D1
N080 G16 I1 J1 K-1
N090 G01 X0
N100 G01 Y10
N110 G03 X10 Y0 R15
```

N120 G40

- Explanation of G-code

  N000: Tool radius correction with a tool radius of 1 is performed from the next path. The path is corrected to the left in the direction of motion.
  N010: A plane is specified with normal vector components I=1, J=1, and K=1.

  N020: The tool is moved (linearly interpolated) to the next start position in ramp-in motion for tool radius correction.
  N030: The tool is moved (circularly interpolated) on the path corrected to the left for the tool radius in the direction of motion on the plane specified by the normal vector.

  N040: The application of tool radius correction ends.
  N050: The tool is moved (linearly interpolated) to (X-1, Y5) in ramp-out motion.

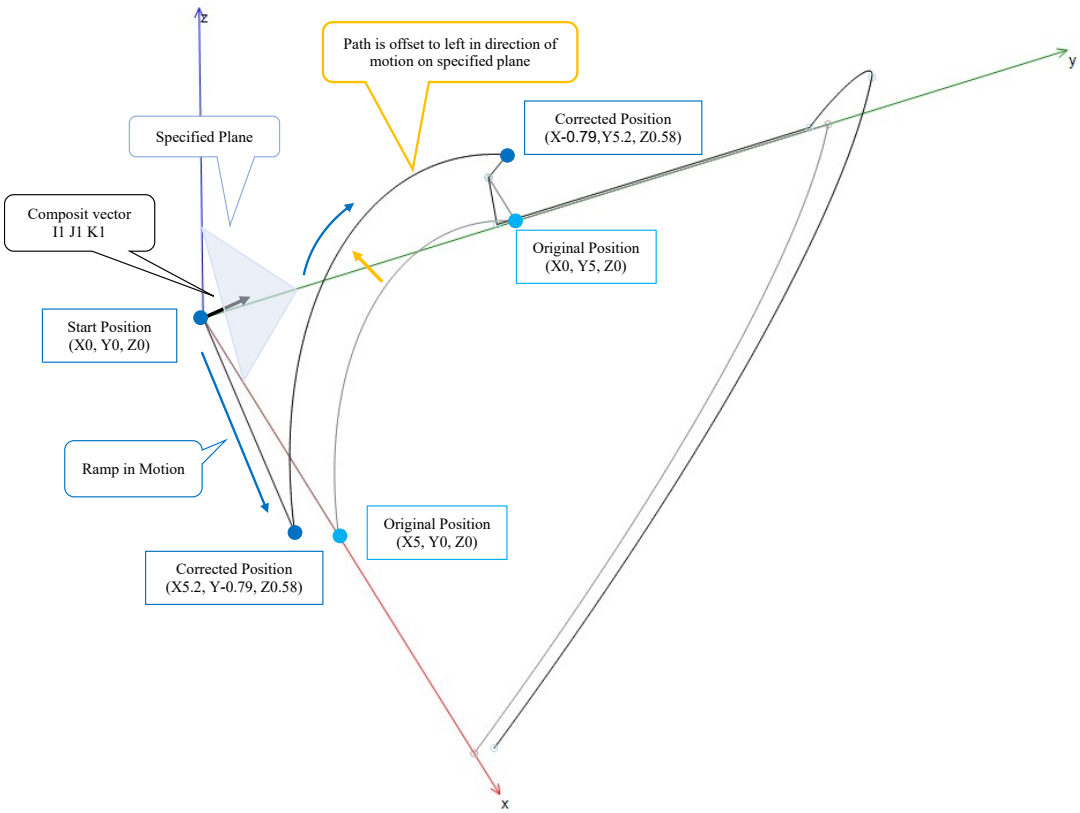
  N060: The motion start position (X-1, Y5) is set to change the corrected plane.
  N070: Tool radius correction with a tool radius of 1 is performed from the next path. The path is corrected to the right in the direction of motion.

  N080: A plane is specified with normal vector components I=1, J=1, and K=-1.
  N090: The tool is moved (linearly interpolated) to the next start position in ramp-in motion for tool radius correction.

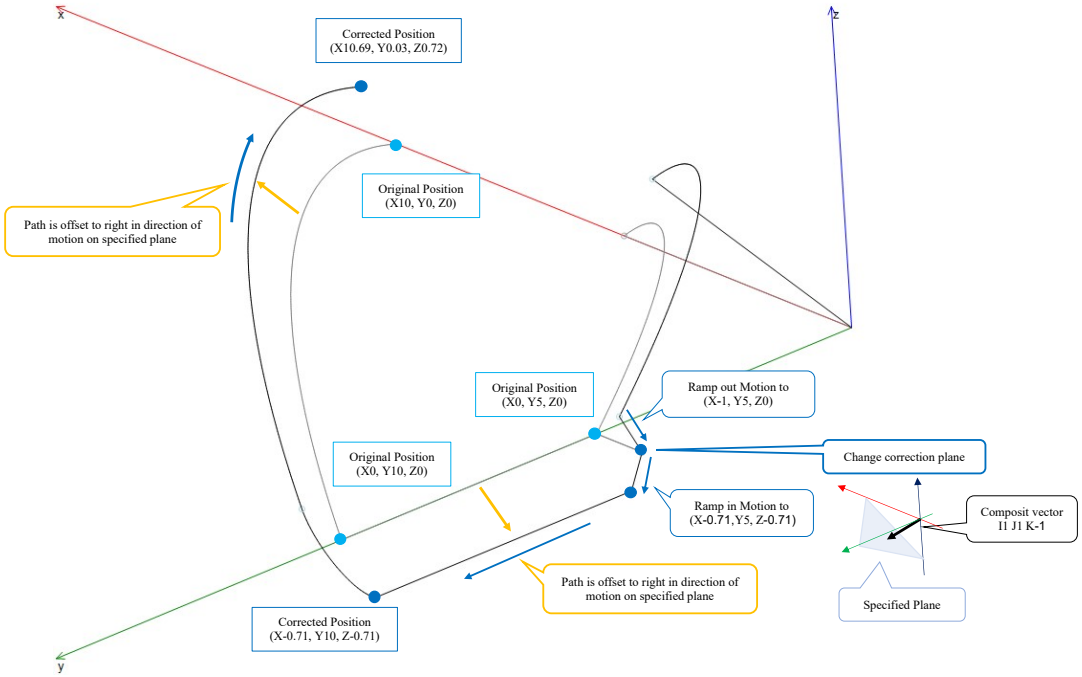
  N100: The tool is moved (linearly interpolated) on the path corrected to the right for the tool radius in the direction of motion on the plane specified by the normal vector.
  N110: The tool is moved (circularly interpolated) on the path corrected to the right for the tool radius in the direction of motion on the plane specified by the normal vector.

  N120: The application of tool radius correction ends.

- Behavior from N000 to N040



- Behavior from N050 onwards



Info.

- To change the plane to be corrected, G92 must be used, as shown in N060 in the above setting example.

8.6.9 G43: Tool Length Correction

Tool length correction allows the programmed path to be corrected according to the length of the tool in use without change to the CNC program. G43 can be used only in CNC program files.

Rules for Tool Length Correction

- Specification of tool length correction

To perform tool length correction by SMC_ToolLengthCorr, include G43 in the CNC program file. Tool length correction converts the path so that it is offset by the specified tool length.

G-code	Function
G43	Tool length correction

- Parameters used for tool length correction

Parameter name		Input value	Overview
Tool length	X-axis	I xxx: Tool length to be corrected	Correct the path by the specified tool length in the direction of each axis.
	Y-axis	J xxx: Tool length to be corrected	
	Z-axis	K xxx: Tool length to be corrected	

Example: Tool length correction in z-axis direction

- Apply tool length correction to the coordinate system that is offset by the tool length when using a tool with a length in the z-axis direction.

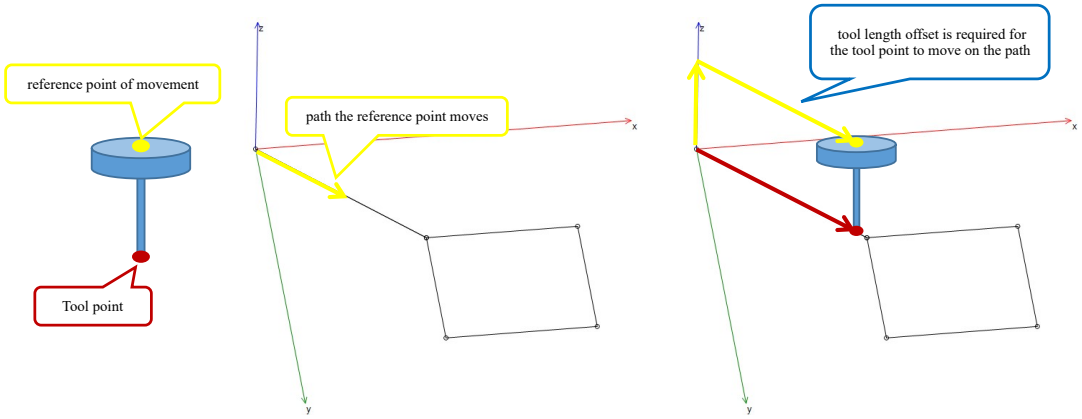
[Setting example]

G-code example:

```
N000 G17
N010 G43 K-5
N020 G01 X5 Y5 F10
N030 G01 X10 Y5
N040 G01 Y10
N050 G01 X5
N060 G01 Y5
N070 G01 X0 Y0
```

- Explanation of G-code

N000: 3D mode is activated.
N010: Tool length correction is activated. The tool length is 5 (K-5) in the z-axis direction.
N020: The tool is moved (linearly interpolated) to (X5, Y5). At the same time, the specified tool length K-5 is corrected and the tool is moved to Z=5.
N030 and subsequent lines: The tool is moved (linearly interpolated) to the specified coordinate position. Z is always in a position that is offset by the tool length K-5.



- To change the tool length during operation

[Setting example 1]: Specifying the tool number in the CNC program to perform tool length correction

```
G-code example:
N000 G17
N010 G01 X10 Y20 F10
N020 M100 K1
N030 G75
N040 G01 X20 Y30 F30
```

```
Declaration section (excerpt):
SMC_ToolLengthCorr_0 : SMC_ToolLengthCorr;
a_lrToolLength : ARRAY [0..2] OF LREAL := [0, 0, 5];
ToolNo : INT;
avTool : ARRAY [1..10] OF SMC_VECTOR3D;
```

```
Control section (excerpt):
SMC_ToolLengthCorr_0.adToolLength := a_lrToolLength;
ToolNo := TO_INT(SMC_GetMParameters_0.dK);
IF SMC_Interpolator0.wM = 100 THEN
   a_lrToolLength[0] := avTool[ToolNo].dX;
   a_lrToolLength[1] := avTool[ToolNo].dY;
   a_lrToolLength[2] := avTool[ToolNo].dZ;
END_IF
```

- Explanation of G-code

  N000: 3D mode is activated.
  N010: The tool is moved (linearly interpolated) to (X10, Y20).
  N020: Since the M-code parameter K word (tool number) is 1, the tool length correction is performed using the tool length preset in avTool[1].
  N030: Timing synchronization causes decoding to wait until CNC operation has worked through previous objects.
  N040: The tool is moved (linearly interpolated) to (X20, Y30).

[Setting example 2]: Setting the offset value in the CNC program to perform tool length correction

```
G-code example:
N000 G17
N010 G01 X10 Y20 F10
```

```
N020 G43 K1
N030 G01 X20 Y30 F30
```

```
Declaration section (excerpt):
SMC_NCInterpreter_0 : SMC_NCInterpreter;
aToolVector : SM3_Math.SMC_VECTOR3D := (dZ := 5);
SMC_ToolLengthCorr_0 : SMC_ToolLengthCorr;
```

```
Control section (excerpt):
SMC_NCInterpreter_0.vStartToolLength := aToolVector;
SMC_ToolLengthCorr_0.adToolLength := SMC_Interpolator_0.adToolLength;
```

- Explanation of G-code

  N000: 3D mode is activated.
  N010: The tool is moved (linearly interpolated) to (X10, Y20).

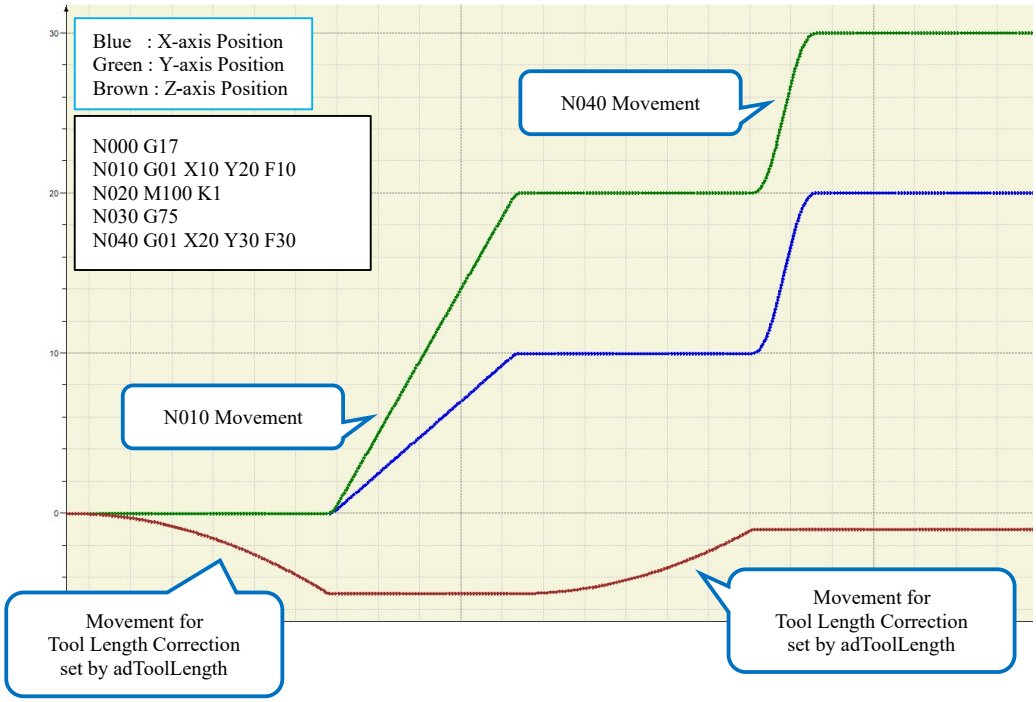
  N020: Tool length correction is activated. The tool length is -1 (K1) in the z-axis direction.
  N030: The tool is moved (linearly interpolated) to (X20, Y30). At the same time, the specified tool length K1 is corrected.

Info.

The timing at which the movement to correct the tool length occurs differs depending on the method of setting the tool length. For example, the figure below shows the trace waveforms in Setting example 1 and Setting example 2 above, where the start tool length is set to Z=5 and then the tool is changed to another tool with a length of Z=1.

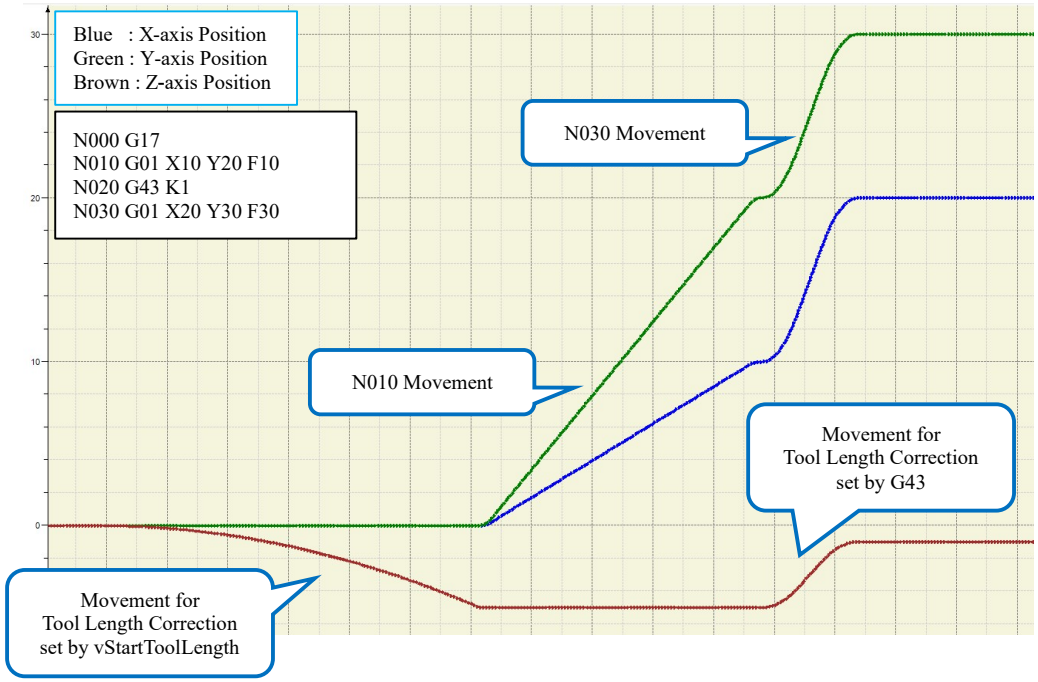
- Trace waveform in Setting example 1

  When a tool length is set directly in SMC_ToolLengthCorr.adToolLength, the tool moves according to the CNC program after the movement to correct the tool length is completed. As shown in Setting example 1, the movement by G01 (N010, N040) occurs after completion of the movement for tool length correction.



- Trace waveform in Setting example 2

  When a tool length is set in the decoder's vStartToolLength, the correction movement occurs before the operation written in the CNC program is performed. On the other hand, for the tool length specified by G43, the correction movement occurs simultaneously with the motion specified by the next G-code. As shown in Setting Example 2, the movement by G01 (N010) occurs after the completion of the movement for tool length correction using vStartToolLength, but the movement for tool length correction set by G43 occurs simultaneously with the movement by G01 in N030.



- To activate the H switch during tool length correction

[Setting example]

G-code example:

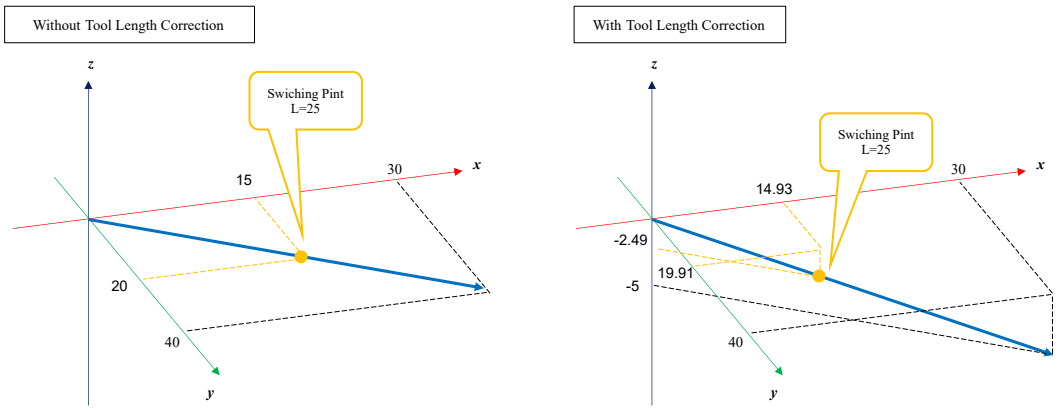
```
N000 G17
N010 G43 K5
N020 G01 X30 Y40 H1 L25 F10
```

- Explanation of G-code

  N000: 3D mode is activated.
  N010: Tool length correction is activated. The tool length is -5 (K5) in the z-axis direction.
  N020: The tool is moved (linearly interpolated) to (X30, Y40). At the same time, the specified tool length K5 is corrected.

  H1 is switched ON at timing when the travel distance reaches 25.

  The position where H1 is switched ON is (X14.93, Y19.91, Z-2.49), which is determined including the movement for tool length correction.



Info.

When linear interpolation and H switch are set in the G-code that follows G43, the movement of tool length correction is also taken into account to determine the position where the switch is activated.

Example: Tool length correction during coordinate conversion

- Applying tool length correction to correct the coordinate system in the y-axis direction in combination with coordinate conversion using G54

[Setting example]

G-code example:

```
N000 G17
N010 G01 X5 Y5 Z5 F10
N020 G54 X10 Y10 Z10 A180 B0 C0
N030 G43 J-5
N040 G01 X5 Y5 Z5
```

- NCInterpreter.eOriConv

  The convention is set to ZYZ (G54 parameters A and C are interpreted as rotation angles around the z-axis, and the parameter B is interpreted as a rotation angle around the y-axis).

- Explanation of G-code

  N000: 3D mode is activated.
  N010: The tool is moved (linearly interpolated) to (X5, Y5, Z5).
  N020: Absolute coordinate conversion is performed by G54.

  The origin of the DCS coordinate system is converted from (X0, Y0, Z0) to (X10, Y10, Z10).

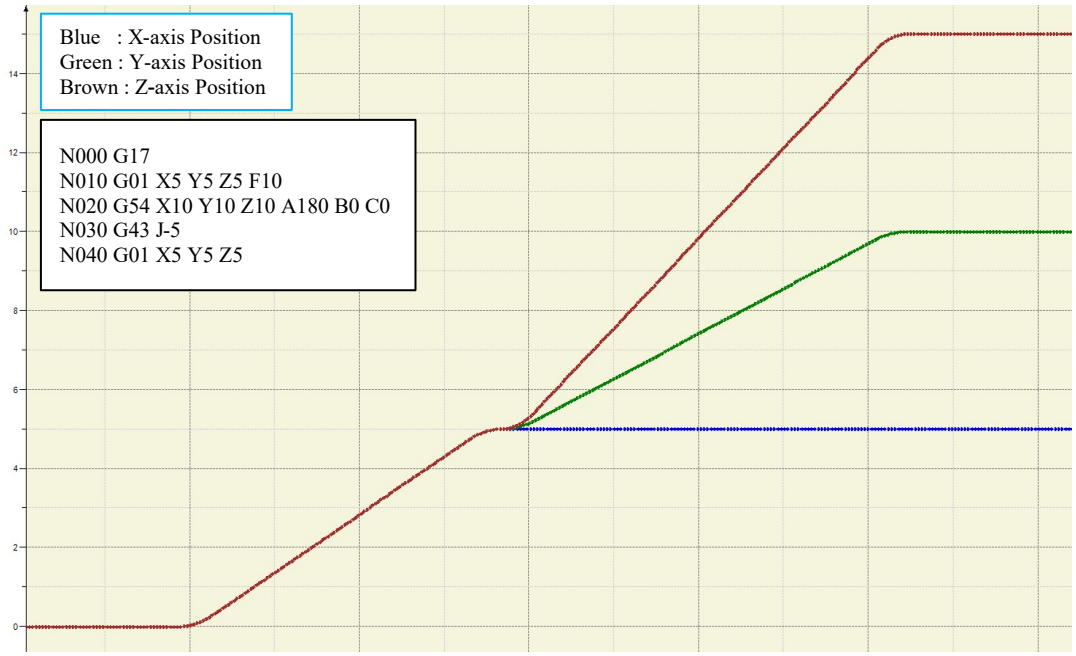
  According to the rotation convention ZYZ, coordinates are rotated 180° around the z-axis.

  N030: Tool length correction is activated. The tool length is 5 (J-5) in the y-axis direction.

  The offset amount and orientation are applied to the original coordinate system, since tool length correction is not affected by coordinate conversion.

  N040: Linear interpolation according to the absolute coordinate system is performed. For the Y-coordinate, the amount of travel is offset by the tool length correction.

The final motion path can be checked as shown in traces below.



Example: Combined use of timing synchronization by G75 and tool correction

- Using timing synchronization by G75 allows you to perform tool correction after a tool change even when CNC control is active by changing parameters according to the new tool. This example shows G-codes for combined use of tool radius correction and tool length correction.

[Setting example]

```
G-code example:
N000 G91 F10
N010 G17
N020 M902 K1
N030 G75
N040 G43 I$OffsetX$ J$OffsetY$ K$OffsetZ$
N050 G00 Z5
N060 G41 D$Radius$
N070 G00 X5 Y7
~
N150 G00 X2
N160 G40
N170 G00 X4 Y6
N180 M902 K2
N190 G75
N200 G43 I$OffsetX$ J$OffsetY$ K$OffsetZ$
N210 G42 D$Radius$
N220 G00 X1 Y1
~
N290 G40
```

```
N300 G00 X-2
```

- Explanation of G-code

  N000: Relative coordinate specification is set.
  N010: 3D mode is activated for three-dimensional operation. Since the intended tool radius correction is executed in XY plane, G17 is used.

  N020: The preset tool information corresponding to tool No. 1 is acquired.

  The acquired tool information is set to OffsetX in N040 and Radius in N060.
  N030: Timing synchronization causes decoding to wait until CNC operation has worked through previous objects.

  N040: Tool length correction is applied using the value acquired and set from the tool number.
  N050: The tool is moved (linearly interpolated) in the z-axis direction.

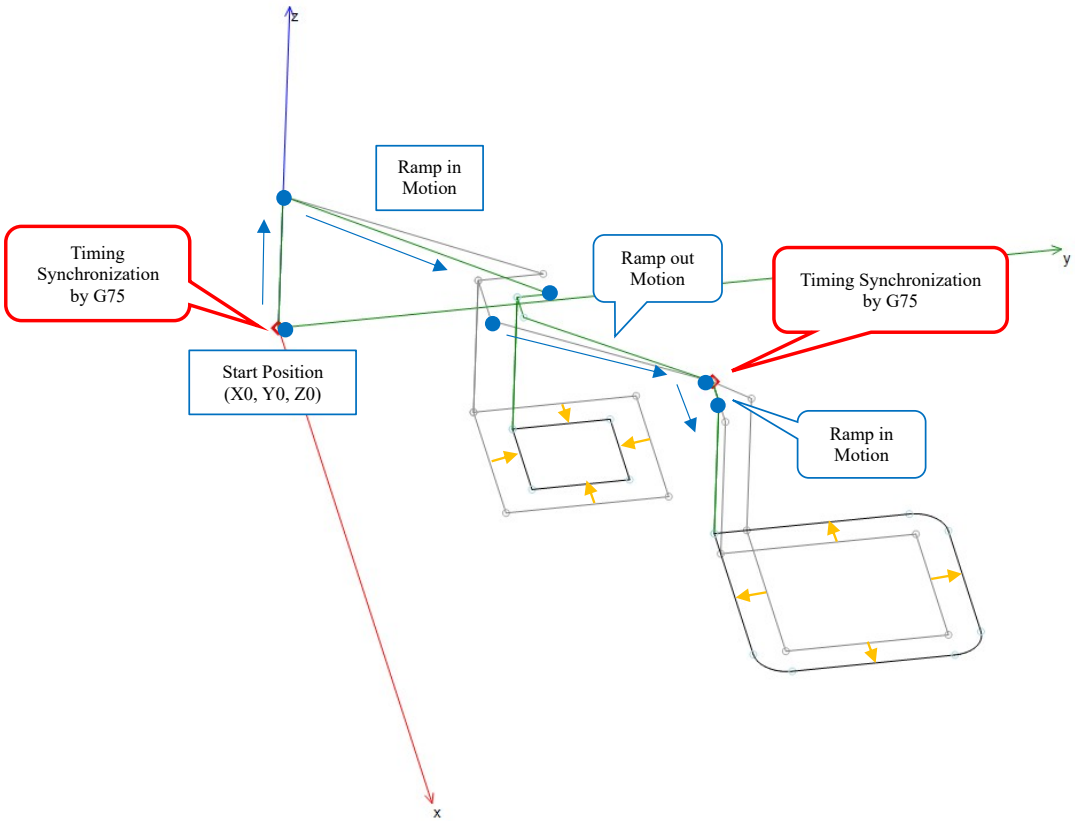
  N060: Tool radius correction is performed from the next path. The correction radius is a value acquired and set from the tool number and the path is corrected to the left in the direction of motion.
  (Lines omitted)

  N160: The application of tool radius correction ends.
  N170: The tool is moved (linearly interpolated) in ramp-out motion for tool radius correction.

  N180: The preset tool information corresponding to tool No. 2 is acquired and set in the variables.
  N190: Timing synchronization causes decoding to wait until CNC operation has worked through previous objects.

  N200: Tool length correction is applied using the value acquired and set from the tool number.
  N210: Tool radius correction is performed from the next path. The correction radius is a value acquired and set from the tool number and the path is corrected to the right in the direction of motion.

  (The rest omitted)



Info.

- The processing of tool radius correction includes up to the movement (ramp-out motion) by the line following G40. Therefore, a CNC program such as the following considers that tool radius correction is in progress, so G75 cannot be used in combination.

  Unacceptable example

  N150 G00 X2

  N160 G40

  N170 M902 K2

  N180 G75

  N190 G00 X4 Y6

8.6.10 G50, G51, G52: Path Smoothing

With path smoothing, a connection between paths written in the CNC program can be changed to a smooth path. An angle between paths subject to smoothing can be set to change C-point control motion to P-point control motion.

Setting rules for smoothing

- Specifying smoothing properties

  To perform smoothing by means of SMC_SmoothPath, write G50/G51 commands and to perform arc correction by means of SMC_RoundPath, write G50/G52 commands.

G-code	Function
G50	Ends smoothing.
G51	Starts path smoothing by SMC_SmoothPath.
G52	Starts arc correction between paths by SMC_RoundPath.

- Parameters used for smoothing

Parameter name	Input value
Radius	D xxx G51 xxx: The radius of curvature of a spline path for smoothing G52 xxx: The radius of an arc for arc correction

Example: Smoothing by SMC_SmoothPath

- When bSymmetricalDistances is set to TRUE

  [Setting example 1]

```
G-code example:
N000 G01 X10 Y0 F10
N010 G51 D10
N020 G01 X10 Y20
N030 G01 X20 Y20
N040 G01 X20 Y0
N050 G50
N060 G01 X30 Y0
```

  - Explanation of G-code

    N000: The path is moved (linearly interpolated) to (X10, Y0).

    N010: Smoothing is performed on the path at the succeeding lines with a radius of curvature of 10.

    N020: The path is moved (linearly interpolated) to (X10, Y20).

    N030: The path is moved (linearly interpolated) to (X20, Y20).

    Smoothing is applied to a segment between the paths for N020 and N030.

    Because bSymmetricalDistances = TRUE, the G-code set value D10 and half the length of the short side of the path sides, 10 × 0.5 = 5, are compared to determine a radius of curvature.

The smaller value “5” is assigned to the radius of curvature D’ for smoothing.

N040: The path is moved (linearly interpolated) to (X20, Y0).

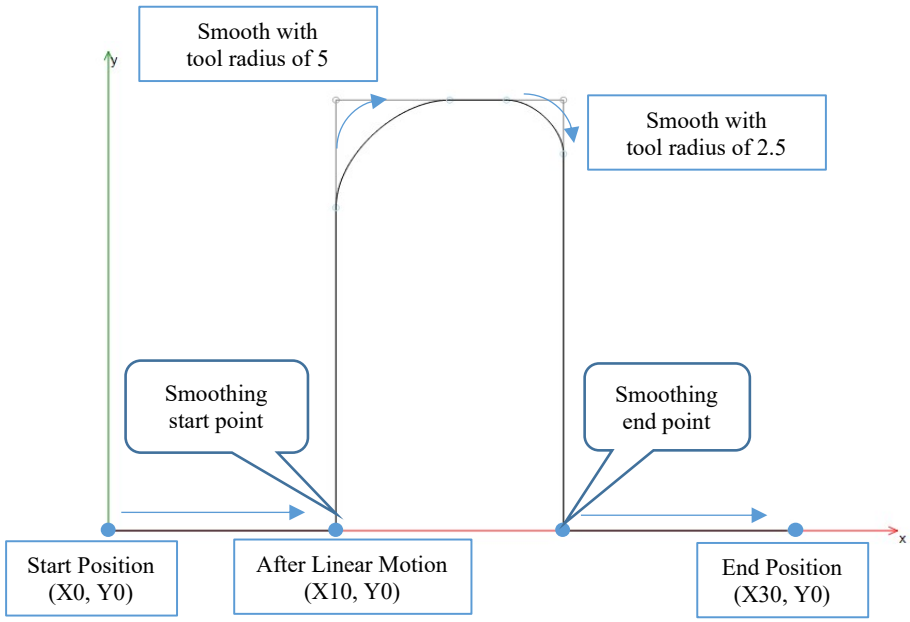
Smoothing is applied to a segment between the paths for N030 and N040.

Because bSymmetricalDistances = TRUE, the 5 assigned earlier to D’ and half the length of the short side of the path sides are compared to determine a radius of curvature.

The short side is 5, from which a length equivalent to smoothing at N30 is subtracted. Thus, the 5 for D’ and the half the length of the short side of the path sides, 5 × 0.5 = 2.5, are compared, and the smaller value “2.5” is assigned to the radius of curvature for smoothing.

N050: The applied smoothing ends.

N060: The path is moved (linearly interpolated) to (X30, Y0).



- When both bSymmetricalDistances and bImprovedSymmetricCuts are set to TRUE

[Setting example 2]

```
G-code example:
N000 G01 X10 Y0 F10
N010 G51 D10
N020 G01 X10 Y20
N030 G01 X20 Y20
N040 G01 X20 Y0
N050 G50
N060 G01 X30 Y0
```

- Explanation of G-code

N000: The path is moved (linearly interpolated) to (X10, Y0).

N010: Smoothing is performed on the path at the succeeding lines with a radius of curvature of 10.

N020: The path is moved (linearly interpolated) to (X10, Y20).

N030: The path is moved (linearly interpolated) to (X20, Y20).

Smoothing is applied to a segment between the paths for N020 and N030.

Because bSymmetricalDistances = TRUE, the set value D10 and half the length of the short side of the path sides, 10 × 0.5 = 5, are compared to determine a radius of curvature. The smaller value “5” is assigned to the radius of curvature D’ for smoothing.

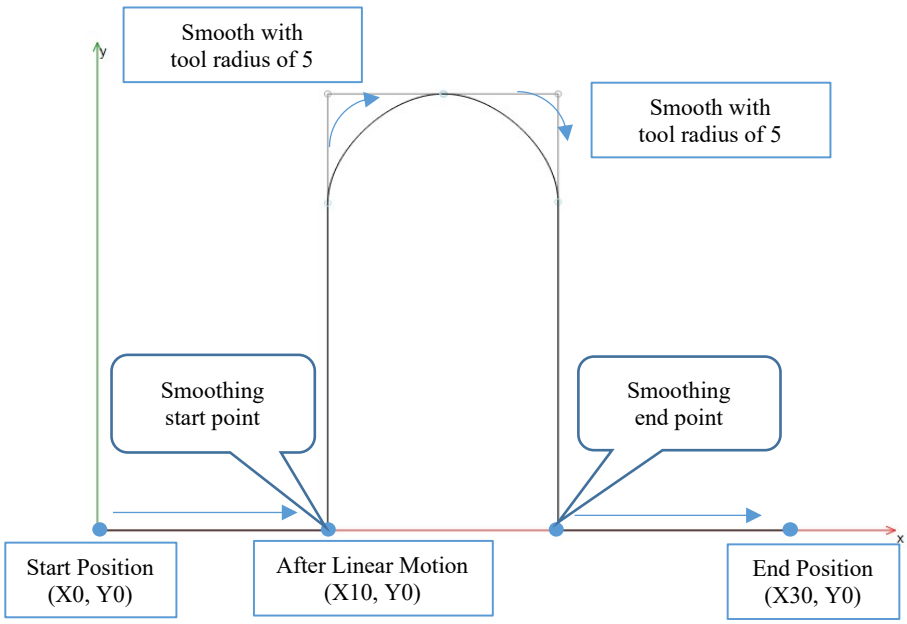
N040: The path is moved (linearly interpolated) to (X20, Y0).

Smoothing is applied to a segment between the paths for N030 and N040.

Because bSymmetricalDistances = TRUE, the 5 assigned earlier to D’ and half the length of the short side of the path sides are compared to determine a radius of curvature. Since bImprovedSymmetricCuts is set to TRUE, the half the length of the short side of the path sides, 10 × 0.5 = 5, and the D’ are compared. The smaller value “5” is assigned to the radius of curvature for smoothing.

N050: The applied smoothing ends.

N060: The path is moved (linearly interpolated) to (X30, Y0).



Example: Arc correction by SMC_RoundPath

- [Setting example]

```
G-code example:
N000 G01 X10 Y0 F10
N010 G52 D10
N020 G01 X10 Y20
N030 G01 X20 Y20
N040 G01 X20 Y0
N050 G50
N060 G01 X30 Y0
```

  - Explanation of G-code

    N000: The path is moved (linearly interpolated) to (X10, Y0).

    N010: Arc correction is performed on the path at the succeeding lines with a radius of 10.

N020: The path is moved (linearly interpolated) to (X10, Y20).

N030: The path is moved (linearly interpolated) to (X20, Y20).

Arc correction is applied to a segment between the paths for N020 and N030.

The set value D10 and half the short side of the path sides, 10 × 0.5 = 5, are compared to determine a radius. The smaller value “5” is assigned to the radius for arc correction.

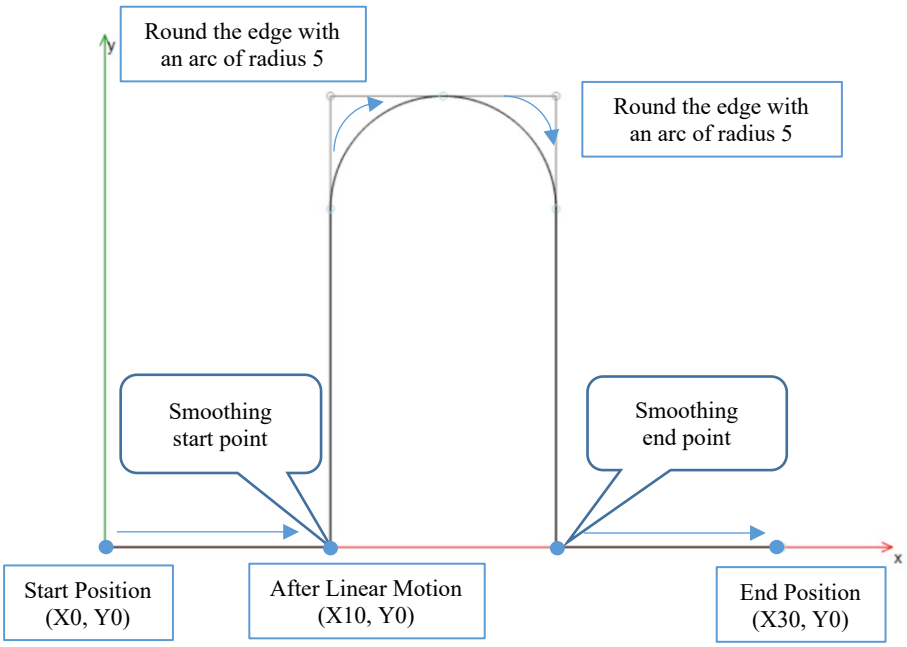
N040: The path is moved (linearly interpolated) to (X20, Y0).

Smoothing is applied to a segment between the paths for N030 and N040.

The set value D10 and half the short side of the path sides, 10 × 0.5 = 5, are compared to determine a radius. The smaller value “5” is assigned to the radius of curvature for smoothing.

N050: The applied arc correction ends.

N060: The path is moved (linearly interpolated) to (X30, Y0).



8.6.11 G53, G54, G55, G56: Coordinate Conversion

This chapter describes the method of coordinate conversion in CNC control.

■ Overview of coordinate conversion

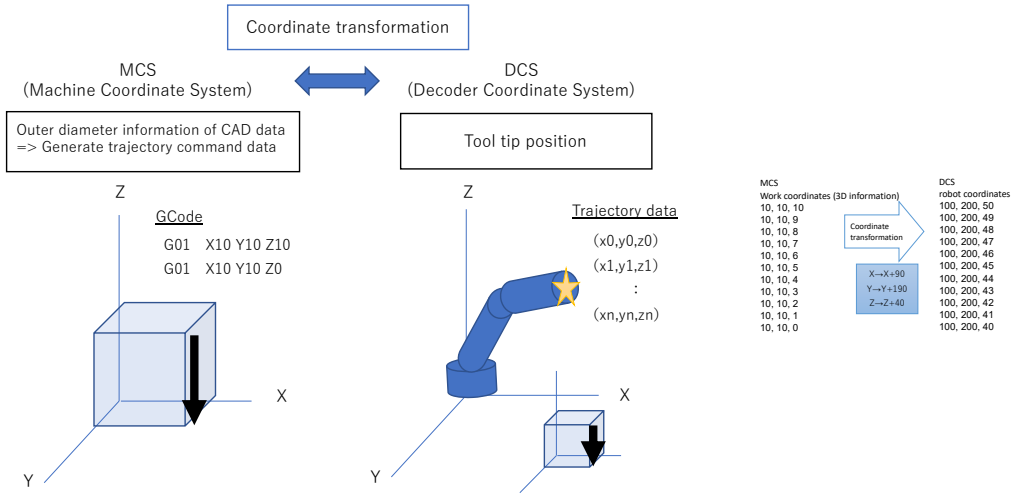
After equipment is installed, offset or rotation amounts of a unit inside the equipment in XYZ directions are corrected with respect to a coordinate system that serves as a basis, in some cases.

On that occasion, replacing the original coordinate system with the coordinate system that serves as a basis is called coordinate conversion.

For GM Programmer, the reference coordinate system under CNC is called the machine coordinate system (MCS), and a coordinate system converted from the MCS is called the decoder coordinate system (DCS).

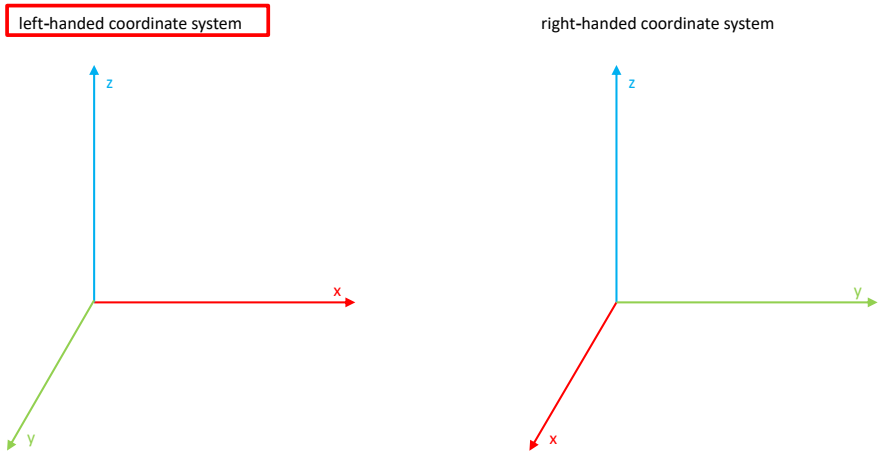
- For instance, as shown below, information on a motion path made in a workpiece coordinate system is converted to coordinates of an end of a robot (a robot coordinate system) and is used to control the robot.

  Example) The end of a tool of the robot is moved along a path from (X10, Y10, Z10) to (X10, Y10, Z0) in a cubic coordinate system on the left side.



■ Coordinate system

- A left-handed coordinate system is adopted for GM Programmer.

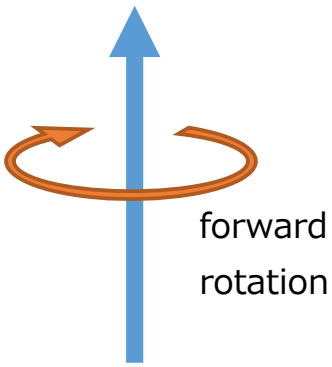


■ Rotation of coordinate system

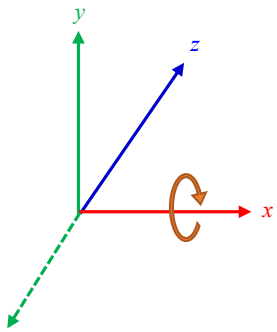
- Rotation of the left-handed coordinate system in the forward direction is clockwise rotation relative to the rotation axis.
- Rotation angles are specified in degrees.

  Rotations can be set in a range from -180 degrees to +180 degrees.

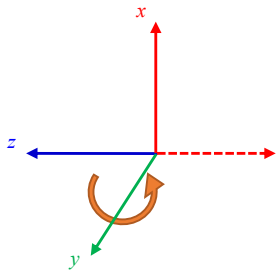
  If you specify an angle outside this range, the coordinate system rotates by a difference between the angle and 360 degrees.
- Example) If an angle of 350 degrees is specified, a rotation of 350 degrees - 360 degrees = -10 degrees is performed.



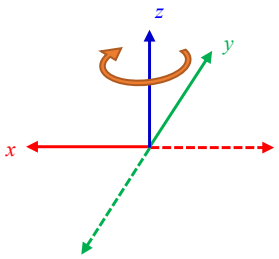
Rotate 90 degrees around the x-axis



Rotate -90 degrees around the y-axis



Rotate 180 degrees around the z-axis



If rotation around an axis is performed for coordinate conversion, the following input parameters must be specified in function blocks for decoding.

Hence, PMC_NCDecoder cannot be used for coordinate rotation. Use SMC_NCDecoder or SMC_NCInterpreter.

Parameter	Type
SMC_NCDecoder.eOriConv SMC_NCInterpreter.eOriConv	SMC_ORI_CONVENTION

■ **SMC_ORI_CONVENTION(ENUM)**

Scope	Description
ADDAXES	Rotation of the coordinate system is not executed. (Default)
ZYZ	Specifies rotation with Euler angles. The coordinate system is rotated in the order of z-axis, y-axis, and z-axis. The G-code parameters A, B, and C are interpreted as rotation angles around the z-axis, y-axis, and z-axis, respectively.
ZYX	Specifies rotation with Euler angles. The coordinate system is rotated in the order of z-axis, y-axis, and x-axis. The G-code parameters A, B, and C are interpreted as rotation angles around the z-axis, y-axis, and x-axis, respectively.
XYZ	Specifies rotation with Euler angles. The coordinate system is rotated in the order of x-axis, y-axis, and z-axis. The G-code parameters A, B, and C are interpreted as rotation angles around the x-axis, y-axis, and z-axis, respectively.

Setting rules for coordinate conversion resetting

A command for coordinate conversion resetting is used to reset the decoder coordinate system (DCS) and return to the coordinate system (MCS), which served as a basis before the coordinate conversion was executed.

- Specifying coordinate conversion resetting

G-code	Function
G53	Coordinate conversion resetting

Setting rules for absolute coordinate conversion

An absolute coordinate conversion method is used to convert the coordinate system (MCS), which serves as a basis, to the decoder coordinate system (DCS) by shifting and rotating using an absolute value.

This method can be used for purposes such as adjusting the positional relationship between a workpiece coordinate system and a robot coordinate system.

- Specifying absolute coordinate conversion

G-code	Function
G54	Absolute coordinate conversion

- Parameters set for absolute coordinate conversion

Parameter name		Input value	Overview
Coordinate shift values	X-axis	X xxx (xxx: shift value)	Shifts the origin of the coordinate system (MCS), which serves as a basis, to specified coordinates.
	Y-axis	Y xxx (xxx: shift value)	
	Z-axis	Z xxx (xxx: shift value)	
Coordinate rotation values	1st axis	A xxx (xxx: rotation angle)	Shifts the origin of the coordinate system (MCS), which serves as a basis, to specified coordinates. (Note 1)
	2nd axis	B xxx (xxx: rotation angle)	
	3rd axis	C xxx (xxx: rotation angle)	
Scaling factors	X-axis	I xxx (xxx: scaling factor)	Scales up or down along each axis by a specified factor. This can be used only in CNC program files.
	Y-axis	J xxx (xxx: scaling factor)	
	Z-axis	K xxx (xxx: scaling factor)	

(Note 1) Axes on which rotation angle parameters A, B, and C act differ depending on the eOriConv input.

Setting rules for relative coordinate conversion

A relative coordinate conversion method is used to convert the decoder coordinate system (DCS) with the current position and orientation to the decoder coordinate system (DCS) by shifting and rotating using a relative value.

This method can be used for purposes such as changing the tool offset amount to any value by G55 in combination with G54 when an tool on the end of a robot is switched to another tool.

- Specifying relative coordinate conversion

G-code	Function
G55	Relative coordinate conversion

- Parameters set for relative coordinate conversion

Parameter name		Input value	Overview
Coordinate shift values	X-axis	X xxx (xxx: shift value)	Shifts the origin of the current coordinate system (DCS) by a specified shift value.
	Y-axis	Y xxx (xxx: shift value)	
	Z-axis	Z xxx (xxx: shift value)	
Coordinate rotation values	1st axis	A xxx (xxx: rotation angle)	Rotates the current coordinate system (DCS) around the axis by a specified angle (degrees). (Note 1)
	2nd axis	B xxx (xxx: rotation angle)	
	3rd axis	C xxx (xxx: rotation angle)	
Scaling factors	X-axis	I xxx (xxx: scaling factor)	Scales up or down along each axis by a specified factor. This can be used only in CNC program files.
	Y-axis	J xxx (xxx: scaling factor)	
	Z-axis	K xxx (xxx: scaling factor)	

(Note 1) Axes on which rotation angle parameters A, B, and C act differ depending on the eOriConv input.

Setting rules for Coordinate reference point resetting

A coordinate reference point resetting method is used to convert to a decoder coordinate system (DCS) by specifying shift and rotation values to the current orientation and position of the coordinate system (MCS), which serves as a basis.

This method can be used for purposes such as setting the current position to a zero point or any position when the robot position is calibrated.

- Specifying coordinate reference point resetting

G-code	Function
G56	Coordinate reference point resetting

- Parameters set for coordinate reference point resetting

Parameter name		Input value	Overview
Coordinate shift values	X-axis	X xxx (xxx: shift value)	Defines the current position in the reference coordinate system (MCS) as a specified coordinate position.
	Y-axis	Y xxx (xxx: shift value)	
	Z-axis	Z xxx (xxx: shift value)	
Coordinate rotation values	1st axis	A xxx (xxx: rotation angle)	Defines the current coordinate direction as a specified rotation angle state. (Note 1)
	2nd axis	B xxx (xxx: rotation angle)	
	3rd axis	C xxx (xxx: rotation angle)	

Parameter name		Input value	Overview
Scaling factors	X-axis	I xxx (xxx: scaling factor)	Scales up or down along each axis by a specified factor. This can be used only in CNC program files.
	Y-axis	J xxx (xxx: scaling factor)	
	Z-axis	K xxx (xxx: scaling factor)	

(Note 1) Axes on which rotation angle parameters A, B, and C act differ depending on the eOriConv input.

Example: Absolute coordinate conversion

- [Setting example]

```
G-code example:
N00 G01 X10 Y10 Z10 F100 E500 E-500
N01 G54 X10 Y10 Z10
N02 G01 X10 Y10 Z10
```

- Explanation of G-code

  N00: Linear interpolation according to the absolute coordinate system is performed.

  Current position (X0, Y0, Z0), end point (X10, Y10, Z10)

  Velocity 100, Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]

  N01: Absolute coordinate conversion is performed.

  The origin of the DCS coordinate system is converted from (X0, Y0, Z0) to (X10, Y10, Z10).

  N02: Linear interpolation according to the absolute coordinate system is performed.

  - DCS coordinate system: Current position (X0, Y0, Z0), end point (X10, Y10, Z10)
  - MCS coordinate system: Current position (X10, Y10, Z10), end point (X20, Y20, Z20)

    Velocity 100 [u/sec], Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]

Note

When G54 is executed in a condition in which coordinate conversion has been performed through G54, G55, or G56, the coordinates are converted according to the settings in the G54 command that is executed last irrespective of the result of the immediately preceding coordinate conversion.

Example: Absolute coordinate conversion and relative coordinate conversion

- [Setting example]

```
G-code example:
N00 G54 X10 Y10 Z10
N10 G01 X10 Y10 Z10 F100 E500 E-500
N20 G55 X-10 Y10 Z-20
N30 G01 X10 Y10 Z10
```

  - Explanation of G-code

    N00: Absolute coordinate conversion is performed.

    The origin of the DCS coordinate system is converted from (X0, Y0, Z0) to (X10, Y10, Z10).

    N10: Linear interpolation according to the absolute coordinate system is performed.

    - DCS coordinate system: Current position (X-10, Y-10, Z-10), end point (X10, Y10, Z10)
    - MCS coordinate system: Current position (X0, Y0, Z0), end point (X20, Y20, Z20)

      Velocity 100, Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]

    N20: Relative coordinate conversion is performed.

    The origin of the DCS coordinate system is converted from (X10, Y10, Z10) to (X0, Y20, Z-10).

    N30: Linear interpolation according to the absolute coordinate system is performed.

    - DCS coordinate system: Current coordinates (X20, Y0, Z30), end point (X10, Y10, Z10)
    - MCS coordinate system: Current coordinates (X20, Y20, Z20), end point (X10, Y30, Z0)

      Velocity 100, Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]

Note

When G55 is executed in a condition in which coordinate conversion has been performed through G54, G55, or G56, the coordinates are converted in the coordinate system after the immediately preceding coordinate conversion because relative values are specified for G55. When only G55 is executed, the coordinates are converted in the same way as G54.

Example: Absolute coordinate conversion and coordinate conversion resetting

- [Setting example]

```
G-code example:
N000 G54 X10 Y10 Z10
N001 G01 X10 Y10 Z10 F100 E500 E-500
N002 G53
N003 G01 X10 Y10 Z10
```

  - Explanation of G-code

    N00: Absolute coordinate conversion is performed.

    The origin of the DCS coordinate system is converted from (X0, Y0, Z0) to (X10, Y10, Z10).

N10: Linear interpolation according to the absolute coordinate system is performed.
- DCS coordinate system: Current position (X-10, Y-10, Z-10), end point (X10, Y10, Z10)
- MCS coordinate system: Current position (X0, Y0, Z0), end point (X20, Y20, Z20)

Velocity 100, Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]

N002: Coordinate conversion resetting is performed.

The origin of the DCS coordinate system is reset to (X0, Y0, Z0).

N003: Linear interpolation is performed according to the following values.

Current position (X20, Y20, Z20), end point (X10, Y10, Z10)

Velocity 100, Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]

Example: Coordinate conversion combination

After absolute coordinate conversion is performed by G54, relative coordinate conversion can be performed by executing G55 two or more times.

- [Setting example]

```
G-code example:
N000 G17
N010 G54 X10 Y10 Z10 A180 B0 C0
N020 G55 X5 Y5 Z5 A0 B90 C0
N030 G55 X5 Y-5 Z0 A0 B0 C-90
N040 G01 X0 Y0 Z0 F100
N050 G01 X100 Y0 Z0
N060 G01 X100 Y100 Z0
N070 G01 X0 Y100 Z0
N080 G01 X0 Y0 Z0
N090 G01 X0 Y0 Z100
N100 G01 X100 Y0 Z100
N110 G01 X100 Y0 Z0
N120 G01 X0 Y0 Z0
N130 G01 X0 Y0 Z100
N140 G01 X0 Y100 Z100
N150 G01 X0 Y100 Z0
N160 G01 X0 Y0 Z0
```

- NCDecoder.eOriConv

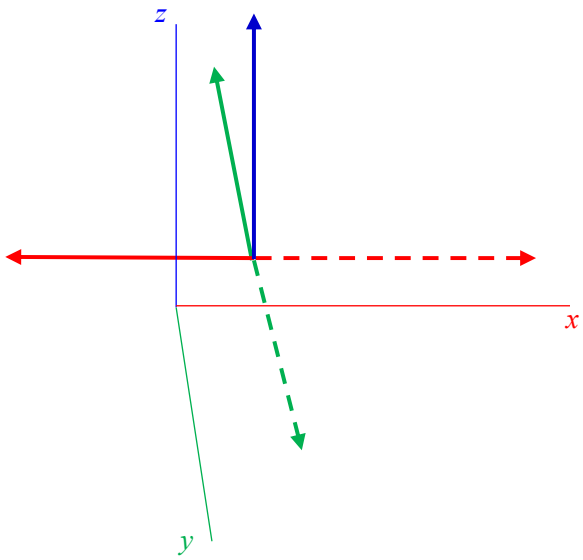
  The convention is set to ZYZ (G54, G55 parameters A, C are interpreted as rotation angles around the z-axis, and the parameter B is interpreted as a rotation angle around the y-axis).

- Explanation of G-code

  N010: Absolute coordinate conversion is performed.

  The origin of the DCS coordinate system is converted from (X0, Y0, Z0) to (X10, Y10, Z10).

  According to the rotation convention ZYZ, coordinates are rotated 180° around the z-axis.

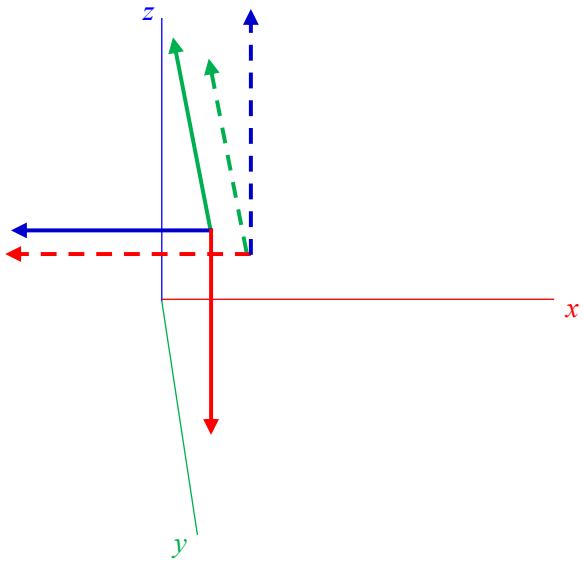


N020: Relative coordinate conversion is performed.

The origin of the DCS coordinate system is converted from (X10, Y10, Z10) to (X5, Y5, Z15).

According to the rotation convention ZYZ, coordinates are rotated 90° around the y-axis.

- From the viewpoint of the original coordinate system, X is shifted in -X direction and thus the relative coordinate conversion results in X = 10 - 5 = 5.
- From the viewpoint of the original coordinate system, Y is shifted in -Y direction and thus the relative coordinate conversion results in Y = 10 - 5 = 5.
- From the viewpoint of the original coordinate system, Z is shifted in +Z direction and thus the relative coordinate conversion results in Z = 10 + 5 = 15.

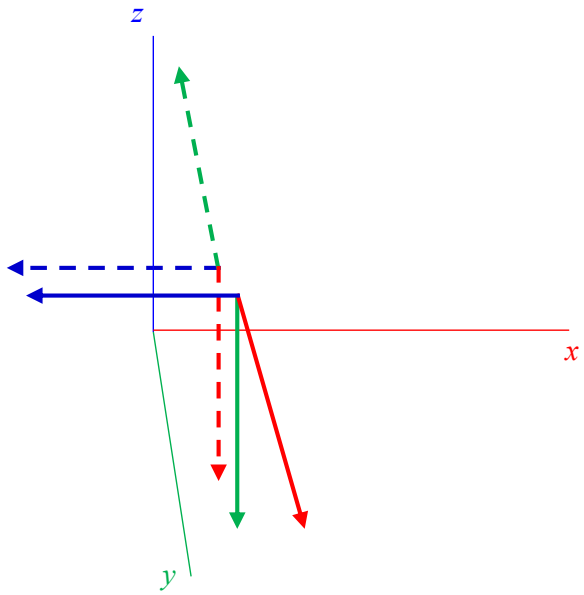


N030: Relative coordinate conversion is performed.

The origin of the DCS coordinate system is converted from (X5, Y5, Z15) to (X5, Y10, Z10).

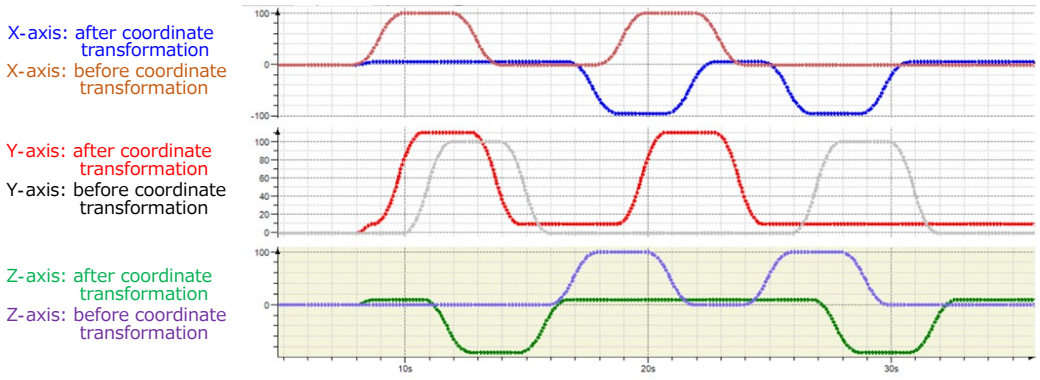
According to the rotation convention ZYZ, coordinates are rotated -90° around the z-axis.

- From the viewpoint of the original coordinate system, X is shifted in -Z direction and thus the relative coordinate conversion results in Z = 15 - 5 = 10.
- From the viewpoint of the original coordinate system, Y is shifted in -Y direction and thus the relative coordinate conversion results in Y = 5 - (-5) = 10.
- From the viewpoint of the original coordinate system, Z is shifted in -X direction and thus the relative coordinate conversion results in X = 5 - 0 = 5.



N040 and subsequent lines: In the converted coordinate system, linear interpolation according to the absolute coordinate system is performed.

The final motion path can be checked as shown in traces below.



Settings for coordinate reference point resetting

- [Setting example]

```
G-code example:
N00 G01 X10 Y-10 Z0 F100 E500 E-500
N10 G56 X0 Y0 Z10
N20 G01 X10 Y10 Z20 F100
```

  - Explanation of G-code

    N00: Linear interpolation according to the absolute coordinate system is performed.

    Current position (X0, Y0, Z0), end point (X10, Y-10, Z0)

    Velocity 100, Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]

    N10: Coordinate reference point resetting is performed.

    The current position in the DCS coordinate system is converted from (X10, Y-10, Z0) to (X0, Y0, Z10).

    N20: Linear interpolation is performed.

    - DCS coordinate system: Current position (X0, Y0, Z10), end point (X10, Y10, Z20)
    - MCS coordinate system: Current position (X10, Y-10, Z0), end point (X20, Y0, Z10)

      Velocity 100, Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]

Note

Since the current position in the reference coordinate system (MCS) is converted to the specified coordinate position, the coordinates are converted according to the settings in the G56 command that is executed last irrespective of the result of the immediately preceding coordinate conversion.

Confirmation items on coordinate conversion

Note the following points when using coordinate conversion.

- Parameter settings for coordinate conversion
  - When programming rotation, the angles of rotation should always be specified in A, B, and C for all three axes.
  - A missing angle of rotation causes an error when decoding.
- Effective range of coordinate conversion

  Coordinate conversion is effective in the range of each CNC program (SMC_CNC_REF). Thus, if processes of the same sort are executed, the similar processes can be generalized, for example, by implementing a common process that includes a subprogram using a CNC program file.

8.6.12 G75: Timing Synchronization

Decoding by SMC_Ncdecoder or SMC_NCInterpreter is synchronized with the timing of SMC_Interolator interpolation operation.

Setting rules for timing synchronization

- Specifying timing synchronization

G-code	Function
G75	Timing synchronization

Example: Settings for timing synchronization

[Setting example]

```
G-code example:
N00 E500 E-500
N10 G01 X100 Y100 F100
N20 G75
N30 G01 X$g_x$ Y$g_y$ F100
```

- Explanation of G-code

  N00: Acceleration and deceleration (acceleration 500 [u/sec^2] and deceleration -500 [u/sec^2]) are set collectively.

  N10: Linear interpolation (X100, Y100) is performed. (Section (1) in the figure below)

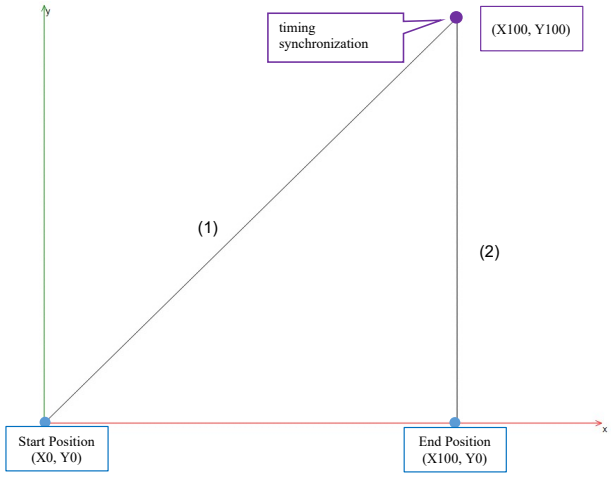
  N20: Timing synchronization causes decoding to wait until CNC operation has worked through previous objects.

  The program refers to a value indicating the time at which code with defined variables is executed at N030.

  N30: Linear interpolation (g_x, g_y) is performed. (Section (2) in the figure below)

  The figure shows a case in which g_x = 100 and g_y = 0.

  Point: With G75 written, decoding is required to wait. This allows the final travel position to be changed during execution of the N10 line.



Info.

- When timing synchronization is performed with G75, the program always makes a pause after an interpolation operation that is written immediately before G75. It should be noted that if continuous motion (P-point control) is necessary, G75 must be executed before the start of P-point control motion.
- For an example of combined use of G75 and M-codes, refer to ""Example: Updating variables"".

8.6.13 G90, G91: Coordinate Specification

Set any of absolute coordinate specification and relative coordinate specification.

Absolute coordinate specification is a method that specifies coordinates as absolute coordinates based on an origin. Meanwhile, relative coordinate specification is a method that specifies a movement distance (relative coordinates) from the current position.

A set of absolute coordinates is the default value for target position settings. Once a method of coordinate specification is set, motion continues under the same coordinate specification unless the other method of coordinate specification is set.

Setting rules for coordinate specification

- Specifying coordinates

G-code	Function
G90	Absolute coordinates specification
G91	Relative coordinates specification

Example: Absolute coordinates specification

[Setting example]

G-code example:

```
N10 G90
N20 G01 X50 Y50 F100 E500 E-500
N30 G01 X100 Y50 F100 E500 E-500
```

- Explanation of G-code

  N10: Absolute coordinate specification is set.

  N20: Linear interpolation is performed according to the following values.

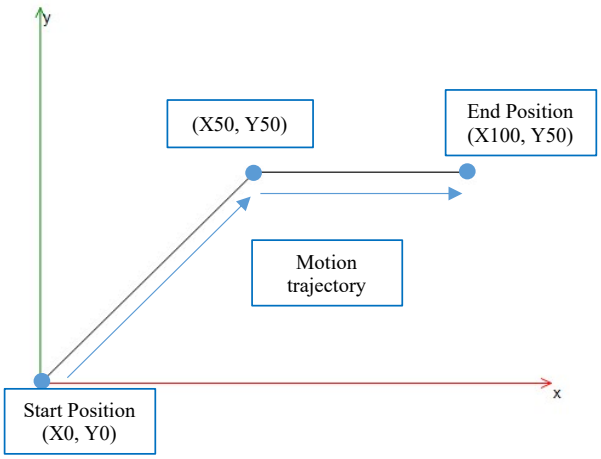
  Current position (X0, Y0), end point (X50, Y50)

  Velocity 100, Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]

  N30: Linear interpolation is performed according to the following values.

  Current position (X50, Y50), end point (X100, Y50)

  Velocity 100, Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]



Example: Relative coordinates specification

[Setting example]

```
G-code example:
N11 G91
N21 G01 X50 Y50 F100 E500 E-500
N31 G01 X50 Y0 F100 E500 E-500
```

- Explanation of G-code

  N11: Relative coordinate specification is set.

  N21: Linear interpolation is performed according to the following values.

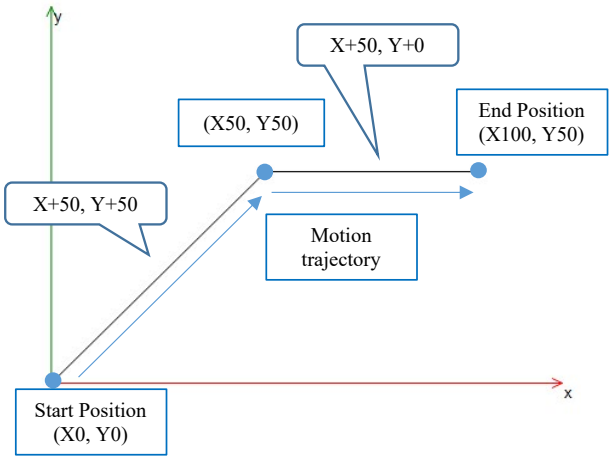
  Current position (X0, Y0), movement amount (X+50, Y+50), end point (X50, Y50)

  Velocity 100, Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]

  N31: Linear interpolation is performed according to the following values.

  Current position (X50, Y50), movement amount (X+50, Y+0), end point (X100, Y50)

  Velocity 100, Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]


y
X+50, Y+0
(X50, Y50)
End Position
(X100, Y50)
X+50, Y+50
Motion
trajectory
x
Start Position
(X0, Y0)

8.6.14 G92: Start position specification

The start position of a CNC program operation is set.

Setting rules for start position specification

- Specifying start position

G-code	Function
G92	Start position specification

- Parameters set for start position specification

Parameter name	Input value
X-axis	X xxx (xxx: starting coordinate)
Y-axis	Y xxx (xxx: starting coordinate)
Z-axis	Z xxx (xxx: starting coordinate)

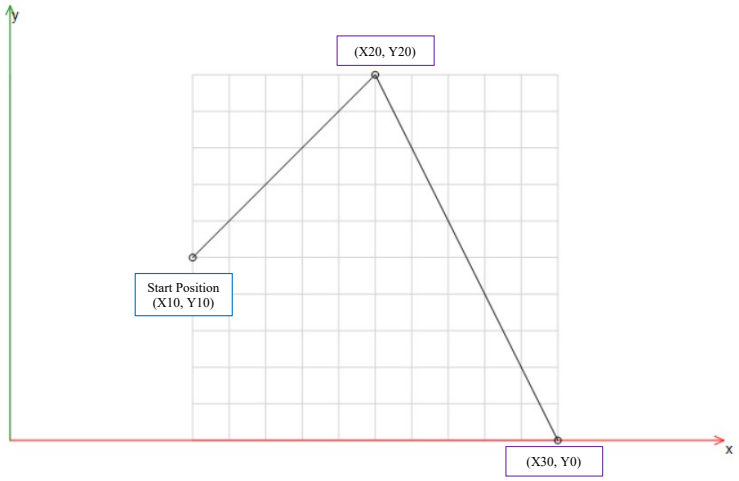
Example: Setting start position

[Setting example]

```
G-code example:
N00 F100 E500 E-500
N10 G92 X10 Y10
N20 G1 X20 Y20
N30 G1 X30 Y0
```

- Explanation of G-code

  N00: Velocity, acceleration, and deceleration (velocity 100 [u/sec], acceleration 500 [u/sec^2] and deceleration -500 [u/sec^2]) are set collectively.
  N10: The start position of the CNC program operation (X10, Y10) is set.
  N20: From the start position (X10, Y10), linear interpolation (X20, Y20) is performed.
  N30: Linear interpolation (X30, Y0) is performed.



8.6.15 G98, G99: Circular arc coordinate specification

Circular arc coordinates can be specified as either absolute coordinates or relative coordinates.

With circular arc coordinates set, center point coordinates can be specified by selecting relative coordinates or absolute coordinates.

A set of relative coordinates is the default value for circular arc coordinates. Once a method of coordinate specification is set, motion continues under the same coordinate specification unless the other method of coordinate specification is set.

Setting rules for coordinate specification

- Specifying circular arc coordinates

G-code	Function
G98	Absolute coordinate specification (center point)
G99	Relative coordinate specification (center point)

Info.

- For relative and absolute coordinate specifications, different G-codes are used for target coordinates and center point coordinates.
- It is strongly recommended that relative coordinates (default setting) be used for center point coordinates, as using relative coordinates makes input easier.

Example: Circular arc relative coordinate specification

Center points can be specified as relative coordinates, as shown below. Coordinates other than center point coordinates are specified as absolute coordinates.

[Setting example]

```
G-code example:
N00 G90
N01 G01 X100 Y100 F100 E500 E-500
N02 G99
N03 G17
N04 G02 X200 Y100 I50 J0 F100
```

- Explanation of G-code

  N00: The target position is specified as absolute coordinates. (This specification can be omitted.)

  N01: Movement to X100, Y100 coordinates (linear interpolation) is performed.

  N02: A center point is specified as relative coordinates.

  N03: An XY-plane is selected.

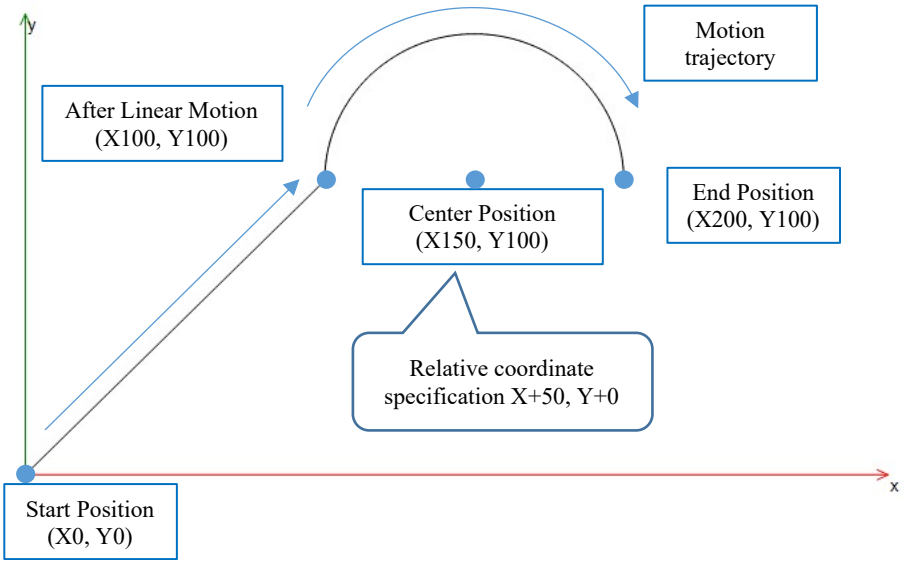
  N04: Circular interpolation is performed in the XY-plane according to the following values.

  Current value (X100, Y100), end point (X200, Y100)

  Center point entered as relative coordinates (X50, Y0)

  Velocity 100

  Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]



Example: Circular arc absolute coordinate specification

Center points can be specified as absolute coordinates, as shown below. Coordinates for all movements including center point coordinates are specified as absolute coordinates.

[Setting example]

```
G-code example:
N00 G90
N01 G01 X100 Y100 F100 E500 E-500
N02 G98
N03 G17
N04 G02 X200 Y100 I150 J100 F100 E500 E-500
```

- Explanation of G-code

  N00: Absolute coordinate specification is set. (This specification can be omitted.)

  N01: Movement to X100, Y100 coordinates (linear interpolation) is performed.

  N02: The center point is specified as absolute coordinates. (This specification can be omitted.)

  N03: An XY-plane is selected.

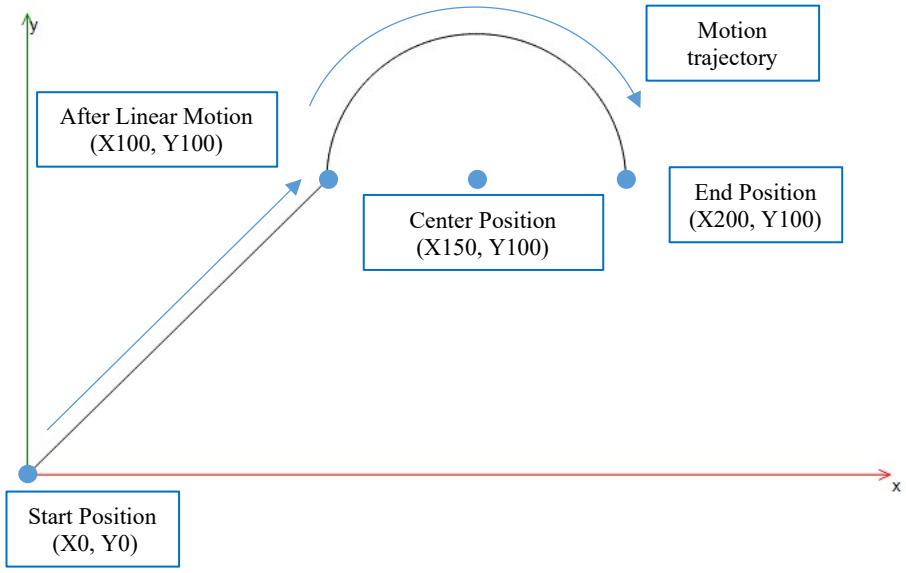
  N04: Circular interpolation is performed in the XY-plane according to the following values.

  Current value (X100, Y100), end point (X200, Y100)

  Center point (X150, Y100)

  Velocity 100

  Acceleration 500 [u/sec^2], Deceleration -500 [u/sec^2]



8.6.16 M-code

With M-code, you can freely program a process in a user program.

When the program reaches a line at which the M-code is executed, the iStatus status of SMC_Interpolator changes to IPO_WAIT and the M-code number sent to wM is output.

After that, the process in the user program ends and when the bAcknM input of SMC_Interpolator is set to TRUE, the CNC control operation restarts.

Setting rules for M-code

- Specifying M-code

M-code	Function
Mxx	xx: A number from 0 to 32766 can be set. For G75 and G04, the number is 32767. Thus, do not use it.

- If the bSuppressSystemMFunctions argument of SMC_Interpolator is set to TRUE, the wM output is not set to 32767 but remains 0 during a pause with G04 or G75.

Example: Updating variables

This is an example of a program designed to detect when the program reaches the M-code line during CNC control, call a separate process and then update the values of variables.

[Implementation example]

- CNC program:

```
(CNC program: CNC_01)
N010 G01 X0 Y0 F10
N020 G01 X10 Y10 F10
N030 M1 K1
N040 G75
N050 G01 X$g_x$ Y$g_y$
N060 M1 K2
N070 G75
N080 G01 X$g_x$ Y$g_y$
```

- MotionTask declaration section:

```
SMC_NCDecoder0          : SMC_NCDecoder;
buf                     : ARRAY[0..10] OF SMC_GEOINFO;
SMC_CheckVelocities0    : SMC_CheckVelocities;
SMC_Interpolator0       : SMC_Interpolator;
SMC_GetMParameters0     : SMC_GetMParameters;
SMC_TRAFO_Gantry2_0     : SMC_TRAFO_Gantry2;
SMC_ControlAxisByPos0   : SMC_ControlAxisByPos;
SMC_ControlAxisByPos1   : SMC_ControlAxisByPos;
R_TRIG_0: R_TRIG;
RS_0: RS;
```

```
bExe                    : BOOL:=TRUE;
bEnaGetMPara            : BOOL;
```

- MotionTask:

```
// CNC Control
SMC_NCDecoder0( nSizeOutQueue:=SIZEOF(buf),
                    pbyBufferOutQueue:=ADR(buf),
                    ncprog:=CNC_01,
                    bExecute:=bExe);
SMC_CheckVelocities0(bExecute:=bExe,
                    poqDataIn:=SMC_NCDecoder0.poqDataOut,
                    dAngleTol:=5);
SMC_Interpolator0(bExecute:=bExe,
                    poqDataIn:=SMC_CheckVelocities0.poqDataOut,
                    dwIpoTime:=1000,
                    wM=>GVL.McodeNo,
                    iStatus=>GVL.IntStatus,
                    bAcknM:=GVL.McodeFinish);
SMC_GetMParameters0(Interpolator:=SMC_Interpolator0,
                    bEnable:=bEnaGetMPara,
                    dK=>GVL.ProcessNo);
SMC_TRAFO_Gantry2_0(pi:=SMC_Interpolator0.
                    piSetPosition);
SMC_ControlAxisByPos0(Axis:=X_Drive,
                    bEnable:=SMC_Interpolator0.bWorking,
                    iStatus:=SMC_Interpolator0.iStatus,
                    fSetPosition:=SMC_TRAFO_Gantry2_0.dx);
SMC_ControlAxisByPos1(Axis:=Y_Drive,
                    bEnable:=SMC_Interpolator0.bWorking,
                    iStatus:=SMC_Interpolator0.iStatus,
                    fSetPosition:=SMC_TRAFO_Gantry2_0.dy);
IF bExe THEN
    R_TRIG_0(CLK:=(GVL.IntStatus = IPO_WAIT));
    RS_0(SET:=R_TRIG_0.Q,
         RESET1:=(GVL.McodeFinish OR NOT(GVL.IntStatus = IPO_WAIT) ),
         Q1=> bEnaGetMPara);
END_IF
```

- Global variable declaration section:

```
// GVL
bStart      : BOOL  := FALSE;
g_x         : DINT  := 0;
g_y         : DINT  := 0;
ProcessNo   : LREAL := 0;
IntStatus   : SMC_INT_STATUS    := IPO_INIT;
McodeNo     : WORD;
McodeFinish : BOOL              := FALSE;
```

- User task declaration section:

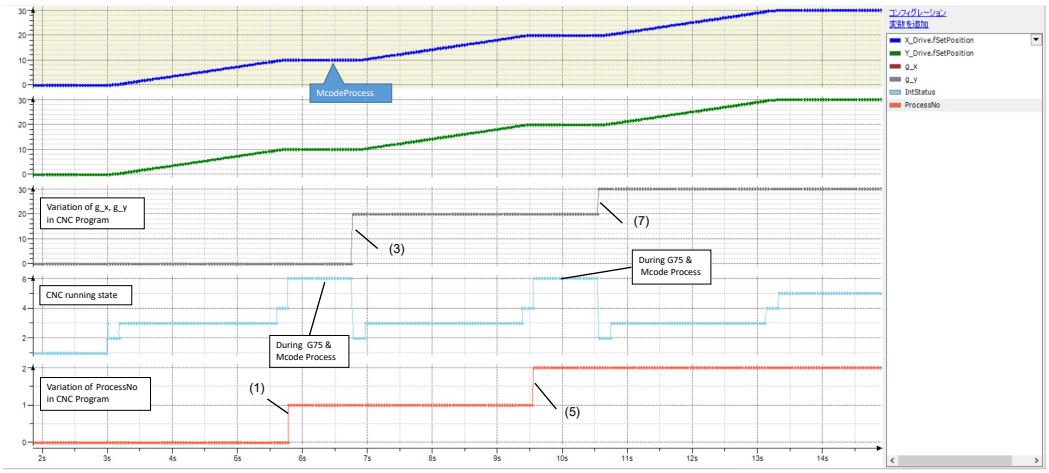
```
// Declare a task section that calls a separate process
TON0        : TON;
TON1        : TON;
```

```
bIN0         : BOOL  := FALSE;
bIN1         : BOOL  := FALSE;
```

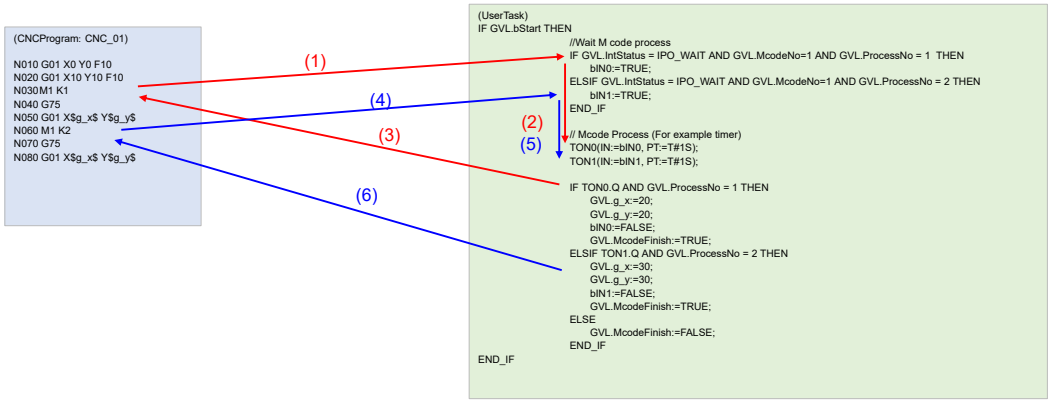
- User task:

```
IF GVL.bStart THEN
    //Wait M code process
    IF GVL.IntStatus = IPO_WAIT AND GVL.McodeNo=1 AND GVL.ProcessNo = 1  TH
EN
        bIN0:=TRUE;
    ELSIF GVL.IntStatus = IPO_WAIT AND GVL.McodeNo=1 AND GVL.ProcessNo = 2
THEN
        bIN1:=TRUE;
    END_IF

    // Mcode Process (For example timer)
    TON0(IN:=bIN0, PT:=T#1S);
    TON1(IN:=bIN1, PT:=T#1S);

    IF TON0.Q AND GVL.ProcessNo = 1 THEN
        GVL.g_x:=20;
        GVL.g_y:=20;
        bIN0:=FALSE;
        GVL.McodeFinish:=TRUE;
    ELSIF TON1.Q AND GVL.ProcessNo = 2 THEN
        GVL.g_x:=30;
        GVL.g_y:=30;
        bIN1:=FALSE;
        GVL.McodeFinish:=TRUE;
    ELSE
        GVL.McodeFinish:=FALSE;
    END_IF
END_IF
```

- Trace results



- Outline of behavior



- Description of operation
  1. When the programs makes a pause at N030 for M-code in the CNC table,
     the status of SMC_Interpolator changes to IPO_WAIT and the M-code number from the wM output is input. ((1) in the figure)
  2. The user task side detects that the program is paused for the M-code, and then a separate process is executed. ((2) in the figure, this example shows a simple process for waiting the elapse of time.)
  3. After the separate process ends, the g_x, g_y values are updated, and the M-code process is completed. ((3) in the figure, The operation resumes by setting GVL.McodeFinish(SMC_Interpolator input bAcknM):=TRUE.)
  4. The rest of the program is similar to the process in (1) to (3).

8.6.17 H-Switch

By using H-switches, you can turn ON or OFF the IO output during the execution of an interpolation operation (e.g., G01, G02, or G03) at the time when the travel distance of the interpolation operation reaches a specified amount.

This function can be useful when processing is necessary during coating or other operation.

Setting rules for H-switch

- Specifying H-switch

Parameter name	Input value	Function
H-switch number	Hxx	The status of switches 1 to 32 can be constantly monitored through the dwSwitches output of SMC_Interpolator. As for the way of writing, switch ON can be controlled with the H number between 1 and 32, and switch OFF can be controlled with the H number between -1 and -32.
Switch condition	Oxx	For the relative position of the switching point in the travel path, specify a value between 0 and 1. Switching point = start point + {(end point - start point)* percentage O}
		To turn the switch function ON in the middle of the travel path, write code as follows. Example: G01 X10 Y10 H1 O0.1 With X0, Y0 set as the start point, H1 switch is turned ON at the point X1, Y1.
	Lxx	You can specify the travel distance from the start point. For L > 0, the distance from the start point is specified. Switching point X = start point + (distance L * cosθ) Switching point Y = start point + (distance L * sinθ) For L < 0, the distance to the end point is specified. Switching point X = end point + (distance L * cosθ) Switching point Y = end point + (distance L * sinθ) * θ is an angle formed by the path
		To turn the switch function ON when the travel distance from the start point is 2, write code as follows. Example: G01 X10 Y10 H1 L2 With X0, Y0 set as the start point, H1 switch is turned ON at the point X1.4142, Y14142.

Note

- Up to three switches can be written in one line. If four or more switches need to be written, write it in the next line.
- Note that if a switch is written in the next line, ON/OFF control cannot be executed during interpolation operation at the preceding line. ON/OFF control is executed with timing of processing at the present line.

Example: Specifying relative position in travel path

[Setting example]

```
CNC program example:
N00 F10 E100 E-100
N10 G1 X10 Y20 H4 O0.5
N20 G1 X30 Y30 H-4 O0.25
```

- Explanation of the CNC program

  N00: Velocity, acceleration, and deceleration (velocity 10 [u/sec], acceleration 100 [u/sec^2] and deceleration -100 [u/sec^2]) are set collectively.

  N10: Linear interpolation (X10, Y20) is performed. H4 is switched ON at timing (X5, Y10) when the relative position reaches 50%.

  X-axis switching point = start point(0) + {travel path(10-0) * position(0.5)} = 5

  Y-axis switching point = start point(0) + {travel path(20-0) * position(0.5)} = 10

  N20: Linear interpolation (X30, Y30) is performed. H4 is switched OFF at timing (X15, Y225) when the relative position reaches 25%.

  X-axis switching point = start point(10) + {travel path(30 - 10) * position(0.25)} = 15

  Y-axis switching point = start point(20) + {travel path(30 - 20) * position(0.25)} = 22.5

Example: Specifying travel distance

[Setting example 1] Specify a travel distance from the start point

```
CNC program example:
N00 F10 E100 E-100
N10 G1 X10 Y10 H4 L2
```

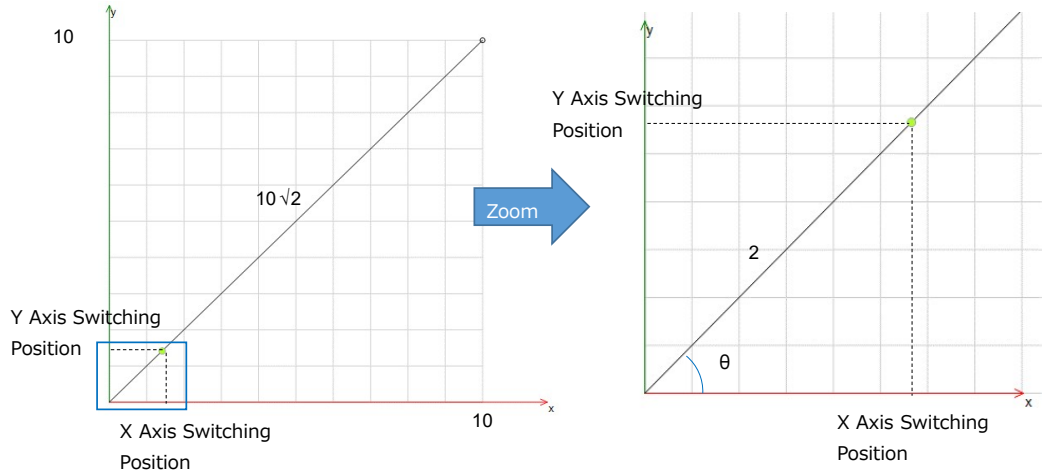
- Explanation of the CNC program

  N00: Velocity, acceleration, and deceleration (velocity 10 [u/sec], acceleration 100 [u/sec^2] and deceleration -100 [u/sec^2]) are set collectively.

  N10: Linear interpolation (X10, Y10) is performed. H4 is switched ON at timing (X5, Y10) when the travel distance reaches 2.

  X-axis switching point = start point + (distance L * cosθ)
  = start point + distance L * (travel distance / oblique side distance)
  = 0 + 2*(10/√(10^2+10^2))
  = 1.414213...

  Y-axis switching point = start point + (distance L * sinθ)
  = start point + distance L * (travel distance / oblique side distance)
  = 0 + 2*(10/√(10^2+10^2))
  = 1.414213...



[Setting example 2] Specify a travel distance to the end point

CNC program example: N00 F10 E100 E-100 N10 G1 X10 Y10 H4 L-2

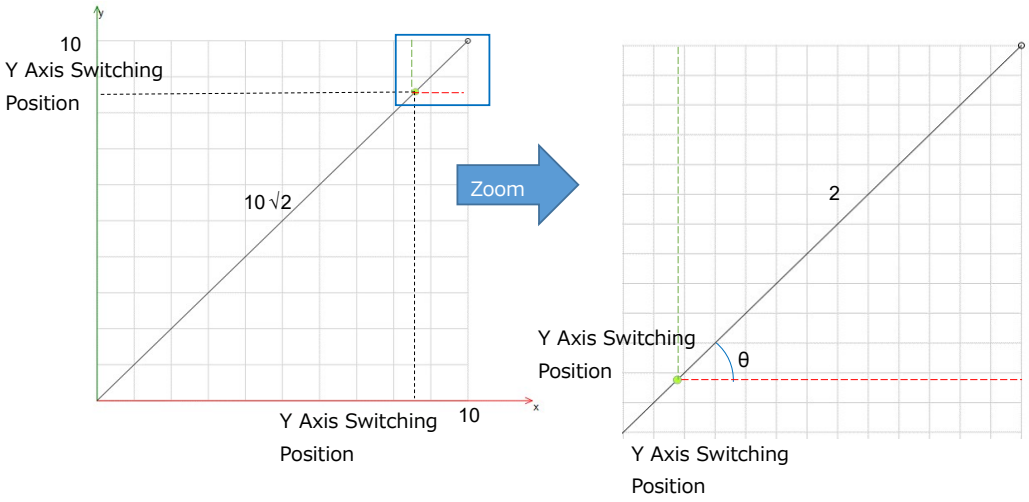
- Explanation of the CNC program

  N00: Velocity, acceleration, and deceleration (velocity 10 [u/sec], acceleration 100 [u/sec^2] and deceleration -100 [u/sec^2]) are set collectively.

  N10: Linear interpolation (X10, Y10) is performed. H4 is switched ON at timing (X5, Y10) when the distance to the end point reaches 2.

  X-axis switching point = end point + (distance L * cosθ)
  = end point + distance L * (travel distance / oblique side distance)
  = 10 - 2*(10/√(10^2+10^2))
  = 10 - 1.414213... = 8.585786...

  Y-axis switching point = end point + (distance L * sinθ)
  = end point + distance L * (travel distance / oblique side distance)
  = 10 - 2*(10/√(10^2+10^2))
  = 10 - 1.414213... = 8.585786...


y
10
Y Axis Switching
Position
10 √2
Zoom
Y Axis Switching
Position
10
x
2
Y Axis Switching
Position
θ
Y Axis Switching
Position

8.6.18 CNC Program File

In addition to creating a CNC program with the CNC editor of the GM Programmer, you can create a CNC program on CAD/CAM, PC, or other tools in text format and read it from an SD card.

CNC programs read from SD cards in such a way are called CNC program files.

The following operations are possible with a CNC program file.

- G20 jump label function (refer to "8.6.7 G20, G36, G37: Jump and Loop Process")
- G53 to 56 coordinate conversion and scaling (refer to "8.6.11 G53, G54, G55, G56: Coordinate Conversion")
- Use of subprograms
- Use of variables (global variables, local variables)
- Use of operators and functions ('+', '-', 'sin', etc.)

Coding rules for CNC program files

From one CNC program file, another CNC program file can be read and executed.

The calling CNC program file is termed a main program, and the called CNC program file is termed a subprogram.

■ Overview of subprograms

For using a subprogram, it is necessary to configure settings in a function block and write a main program and the subprogram in accordance with rules. A subprogram can be called from another subprogram. Up to 12 recursive calls are permitted.

■ Location of setting in function block

Specify a directory with an absolute path where the subprogram is stored in the aSubProgramDirs input of the SMC_ReadNCFile2 function block.

The file is searched for in the specified directories, starting with the directory that has the lowest index in the array. A subprogram in the directory that matches first is run.

* The root directory can be specified with ‘./’ or ‘.’.

■ Coding rules for main program

N number Subprogram name {a value for argument 1, a value for argument 2,,,}

- Example of code (without argument):
  N020 SUBCNC1{}
- Example of code (with argument):
  N020 SUBCNC1{10, 5.5}

■ Coding rules for sub program

SUBPROGRAM Subprogram name {name of argument 1, name of argument 2,,, } RESTORE option

A CNC program for the subprogram is written

END_SUBPROGRAM

- Example of code:

```
SUBPROGRAM SUBCNC1{#a : LREAL, #b : LREAL} RESTORE_MODES
N100 G91 F#a E100 E-100
N110 G01 X10 Y#b
N120 G01 X20 Y10
END_SUBPROGRAM
```

- Subprogram arguments

  Following #, a variable name and a variable type are declared.

  For the variable name, up to 80 half-width alphabetic characters and underscores can be used.

  The available variable types are LREAL, BOOL, and STRING(255).

■ RESTORE option

When returning to the calling program from the subprogram, modal states are restored to the values that it had at the call.

If RESTORE_MODES is specified, the following G-code states are restored.

- Relative/Absolute coordinates specification (G90 / G91, G98 / G99)
- Plane specification and 2D / 3D mode (G15 / G16 / G17 / G18 / G19)
- Coordinate Conversion (G53 / G54 / G55 / G56)
- Velocity and acceleration F, FF, E EF
- Tool radius (D word)

■ RETURN instruction

If RETURN is written in the subprogram, the system can return to the main program before reaching END.

Although the code after RETURN is not executed, it must conform to the coding rules.

- Example of code:

```
SUBPROGRAM CNC_RETURN1{}
N110 G01 X10 Y20
N120 RETURN
N130 G01 X10 Y20
END_SUBPROGRAM
```

- Example of code that causes an error (END_SUBPROGRAM is not at the end of the text):

```
SUBPROGRAM CNC_RETURN2{}
N110 G01 X10 Y20
N120 RETURN
N130 G01 X10 Y20
```

- Example of code that causes an error (there is a syntax error in the code after RETURN):

```
SUBPROGRAM CNC_RETURN2{}
N110 G01 X10 Y20
N120 RETURN
N130 G01 X10 Y20
N140 G01 X((20) Y10
END_SUBPROGRAM
```

Use of variables in CNC program files

When CNC control is executed using a CNC program file, variables can be used in the G-code and values can be assigned to the variables.

Variables of the following three types can be declared.

Scope	Main program	Subprogram
Global variables	○(Available)	○(Available)
Local variables	○(Available)	○(Available)
Subprogram arguments	×(Not available)	○(Available)

- Global variables

  In the CNC program file, as in the CNC editor, global variables can be used and values can be assigned to the variables while the program is in operation. For using a global variable, you must set up the pvl argument of SMC_ReadNCFile2. For details, refer to "8.2.2 SMC_ReadNCFile2 (Read CNC File)".

- Declaration of local variables

  Variables can be used locally in main programs and subprograms on a temporary basis.

  Declarations must be inserted at the beginning of the programs. Declarations of local variables are not required in POU programs and thus are completed inside CNC programs.

- Example of declarations in main program

```
LET #x : LREAL :=10
LET #y : LREAL :=20
LET #b : BOOL :=FALSE
N000 G91 F10 E100 E-100
N010 G01 X#x Y#y
N030 G20 L10 K#b
```

- Example of declarations in subprogram

```
SUBPROGRAM CNC_LET3{}
LET #x : LREAL :=10
LET #y : LREAL :=20
N100 G91 F20 E100 E-100
N110 G01 X#x Y#y
END_SUBPROGRAM
```

Operators and functions

In CNC program files, numerical calculations can be performed using operators and functions.

- Available elements
  - Numerical values and character strings
  - Global variables and local variables
  - Commas and parentheses

  * Attention should be paid to the writing of parentheses because they are excluded from decoding by the SMC_ReadNCFile2.bParenthesesAsComments argument.

- Available operators

Character	Type	Arguments
MOD	LREAL	LREAL, LREAL
*	LREAL	LREAL, LREAL
/	LREAL	LREAL, LREAL
+	LREAL	LREAL, LREAL
-	LREAL	LREAL, LREAL
=	BOOL	BOOL, BOOL
=	BOOL	LREAL, LREAL
=	BOOL	STRING, STRING
<>	BOOL	BOOL, BOOL
<>	BOOL	LREAL, LREAL
<>	BOOL	STRING, STRING
>	BOOL	LREAL, LREAL
<	BOOL	LREAL, LREAL
>=	BOOL	LREAL, LREAL
<=	BOOL	LREAL, LREAL
AND	BOOL	BOOL, BOOL
XOR	BOOL	BOOL, BOOL
OR	BOOL	BOOL, BOOL

- Available functions

Character	Type	Arguments
-	LREAL	LREAL
ABS	LREAL	LREAL
MAX	LREAL	LREAL, LREAL
MIN	LREAL	LREAL, LREAL
NOT	BOOL	BOOL
TRUE	BOOL	-
FALSE	BOOL	-
SIN	LREAL	LREAL
COS	LREAL	LREAL
TAN	LREAL	LREAL
ASIN	LREAL	LREAL
ACOS	LREAL	LREAL
ATAN	LREAL	LREAL
EXP	LREAL	LREAL
LN	LREAL	LREAL
SQRT	LREAL	LREAL
EXPT	LREAL	LREAL, LREAL

Character	Type	Arguments
FLOOR	LREAL	LREAL
CEIL	LREAL	LREAL
PI	LREAL	-
LEN	LREAL	STRING
CONCAT	STRING	STRING, STRING

Notes on the use of CNC program files

Check the CNC program file for any error by bEnableSyntaxChecks of NCInterpreter since build cannot be executed in advance.

However, it should be noted that if the CNC program file is any of the following cases, a syntactical error cannot be detected even by the bEnableSyntaxChecks error checking and decoding remains stopped in midstream.

- G-code is not written.
- The CNC program file is empty.
- Necessary code is not written with G01.
- Command R that is not permitted for G01 is written.
- The Gvl type and the actual variable type differ.
  Example: eVarType=SMC_TYPE_BYTE, ProgramNo= 1000
- More than 22 local variables and subprogram are used.
- The variable name has 81 or more characters.
- A variable type other than the supported variable types LREAL, BOOL, and STRING(255) is used.

The following points should also be noted.

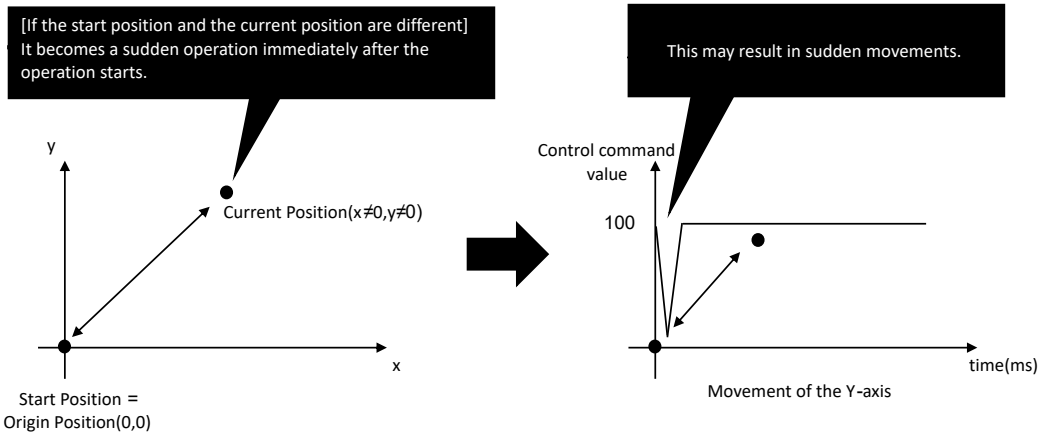
- If no default value is declared for the variable, the program runs with the default value set to 0.
- Local variables cannot be used for jump labels (x of L ! x).
- If a variable name identical to that of a subprogram argument is used for a global variable or a local variable, an error occurs.

8.7 Example of Use of CNC Control

8.7.1 Example of USE: Specifying Starting Coordinates

In CNC programs written in G-code, by default, the starting coordinates are defined as the origin (0, 0). Normally, if the starting coordinates in the CNC program are identical with the operation starting coordinates (current coordinates), normal operation will occur.

If the operation starting position (current coordinates) is not the origin, when operation is started from the current coordinates, rapid movements may occur from the current coordinates through to the origin immediately after startup, thereby preventing normal operation from being performed.



To achieve normal operation, therefore, if the starting coordinates in the CNC program differ from the operation starting coordinates (current coordinates), they must be matched.

Start coordinates are different for smc_cnc_ref, smc_outqueue, and CNC program files.

It can also be specified by "8.6.14 G92: Start position specification".

* In the following descriptions, the starting coordinates in the CNC program are referred to as "starting coordinates" and the coordinates at the start of CNC control operation are referred to as "operation starting coordinates".

Note

- If the specified starting coordinates differ from the operation starting coordinates, there is a risk that rapid movement may occur.

■ For SMC_CNC_REF

Specify starting coordinates in SMC_NCDecoder when decoding is executed.

[Setting example]

- Variable declaration section:

```
// FBs
    SMC_NCDecoder_0         : SMC_NCDecoder;
    SMC_CheckVelocities_0   : SMC_CheckVelocities;
    SMC_Interpolator_0      : SMC_Interpolator;
    SMC_ControlAxisByPos_0  : SMC_ControlAxisByPos;
```

```
    SMC_ControlAxisByPos_1  : SMC_ControlAxisByPos;
    SMC_ReadSetPosition_0   : SMC_ReadSetPosition;  //Read Current Position
    SMC_ReadSetPosition_1   : SMC_ReadSetPosition;
// Variables
    iState          : INT   :=0;            // Case Number
    buf             : ARRAY[0..10] OF SMC_GEOINFO;  // Decord buffer
    piStartpos      : SMC_POSINFO;          // Start Position
    dwTime          : DWORD :=1000;
    xAvoidgaps_0    : BOOL  :=TRUE;
    xAvoidgaps_1    : BOOL  :=TRUE;
    bExe            : BOOL  :=FALSE;        // Execute for FBs
    bStart          : BOOL  :=FALSE;        // Start Flag
```

- Control section:

```
// CNC FBs Settings
SMC_NCDecoder_0(
                ncprog:=CNC_Sample,
                bExecute:=bExe,
                piStartPosition:=piStartpos,
                nSizeOutQueue:=SIZEOF(buf),
                pbyBufferOutQueue:=ADR(buf)
);
SMC_CheckVelocities_0(
                        bExecute:=bExe,
                        poqDataIn:=SMC_NCDecoder_0.poqDataOut
);
SMC_Interpolator_0(
                    bExecute:=bExe,
                    poqDataIn:=SMC_CheckVelocities_0.poqDataOut,
                    dwIpoTime:=dwTime
);
SMC_ControlAxisByPos_0(
                        Axis:=Axis1,
                        iStatus:=SMC_Interpolator_0.iStatus,
                        bEnable:= SMC_Interpolator_0.bWorking,
                        bAvoidGaps:=xAvoidgaps_0,
                        fSetPosition:=SMC_Interpolator_0.piSetPosition.dX
);
SMC_ControlAxisByPos_1(
                        Axis:=Axis2,
                        iStatus:=SMC_Interpolator_0.iStatus,
                        bEnable:= SMC_Interpolator_0.bWorking,
                        bAvoidGaps:=xAvoidgaps_1,
                        fSetPosition:=SMC_Interpolator_0.piSetPosition.dY
);

// When Unreasonable movement is occured
IF SMC_ControlAxisByPos_0.bError = TRUE OR SMC_ControlAxisByPos_0.bStopIpo
= TRUE OR SMC_ControlAxisByPos_1.bError = TRUE OR SMC_ControlAxisByPos_1.bS
topIpo = TRUE THEN
    SMC_Interpolator_0.bEmergency_Stop:=TRUE;
ELSE
    SMC_Interpolator_0.bEmergency_Stop:=FALSE;
END_IF
```

```
IF bStart = TRUE THEN
    CASE iState OF
        0: // Read Current Position of Axises
            SMC_ReadSetPosition_0(
                                    Axis:=Axis1,
                                    Enable:=TRUE,
                                    Position=>piStartPos.dX
            );
            SMC_ReadSetPosition_1(
                                    Axis:=Axis2,
                                    Enable:=TRUE,
                                    Position=>piStartPos.dY
            );
            iState:=1;

        1: // Start CNC motion
            bExe:=TRUE;
    END_CASE
END_IF
```

Explanation of control section

- When the bStart flag is set to TRUE, the current coordinates of each axis are read (SMC_ReadSetPosition).
- The read current coordinates are stored in the variable piStartPos so that they are written in the input variable piStartPosition of SMC_NCDecoder.
- CNC control is executed with the read current coordinates taken as the start position of the path (bExe is set to TRUE).

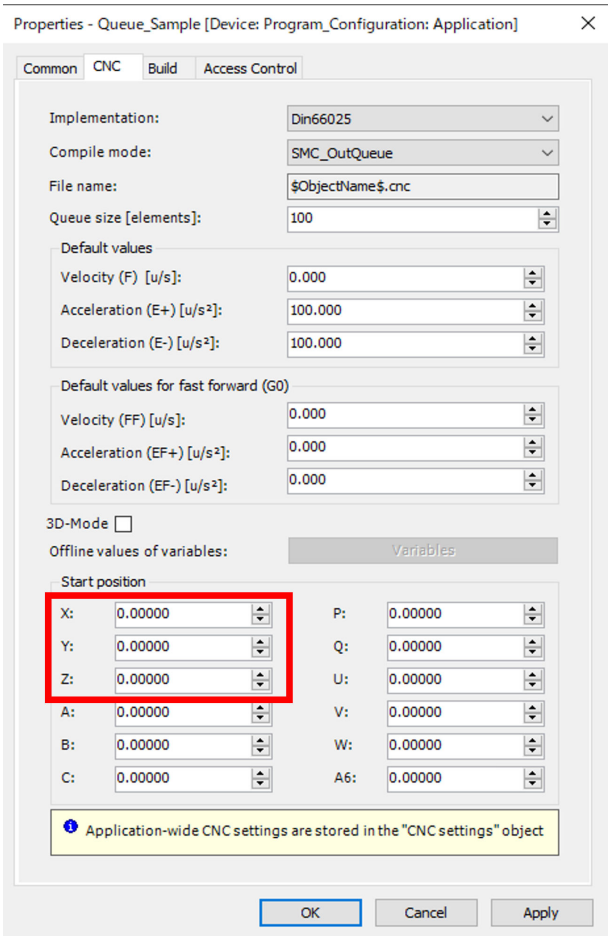
Note

- In the case of a CNC program file, you can set the start position by entering piStartPosition in SMC_NCInterpreter instead of SMC_NCDecoder.

■ For SMC_OUTQUEUE

In the Properties window for the CNC program, set the start position.

In the "Start position" section, specify the respective operation starting coordinates for the X, Y, and Z axes and then click [OK].



[Setting example]

- Variable declaration section:

```
// FBs
    SMC_Interpolator_0      : SMC_Interpolator;
    SMC_ControlAxisByPos_0  : SMC_ControlAxisByPos;
    SMC_ControlAxisByPos_1  : SMC_ControlAxisByPos;
// Variables
    iState              : INT   :=0;            // Case Number
    dwTime              : DWORD :=1000;
    xAvoidgaps_0        : BOOL  :=TRUE;
    xAvoidgaps_1        : BOOL  :=TRUE;
    bExe                : BOOL  :=FALSE;        // Execute for FBs
    bStart              : BOOL  :=FALSE;        // Start Flag
```

- Control section:

```
// CNC FBs Settings
SMC_Interpolator_0(
                    bExecute:=bExe,
                    poqDataIn:=ADR(Queue_Sample),
                    dwIpoTime:=dwTime
```

```
);
SMC_ControlAxisByPos_0(
                        Axis:=Axis1,
                        iStatus:=SMC_Interpolator_0.iStatus,
                        bEnable:= SMC_Interpolator_0.bWorking,
                        bAvoidGaps:=xAvoidgaps_0,
                        fSetPosition:=SMC_Interpolator_0.piSetPosition.dX
);
SMC_ControlAxisByPos_1(
                        Axis:=Axis2,
                        iStatus:=SMC_Interpolator_0.iStatus,
                        bEnable:= SMC_Interpolator_0.bWorking,
                        bAvoidGaps:=xAvoidgaps_1,
                        fSetPosition:=SMC_Interpolator_0.piSetPosition.dY
);

// When Unreasonable movement is occured
IF SMC_ControlAxisByPos_0.bError = TRUE OR SMC_ControlAxisByPos_0.bStopIpo
= TRUE OR SMC_ControlAxisByPos_1.bError = TRUE OR SMC_ControlAxisByPos_1.bS
topIpo = TRUE THEN
    SMC_Interpolator_0.bEmergency_Stop:=TRUE;
ELSE
    SMC_Interpolator_0.bEmergency_Stop:=FALSE;
END_IF

IF bStart = TRUE THEN
    CASE iState OF
        0: // Start CNC motion
            bExe:=TRUE;
    END_CASE
END_IF
```

■ When using G92

See "Example: Setting start position" for the use of G92.

Note

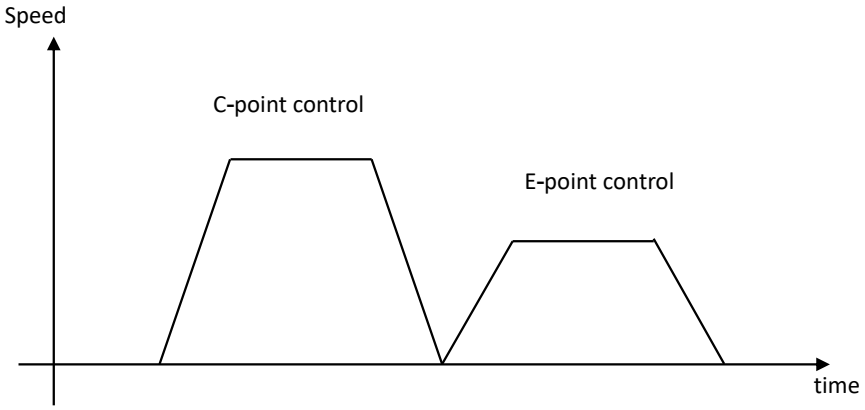
- In the case of absolute statement specification, the set value by the G92 is used, and in the case of a relative statement, the additional value of both sides is used.
- SMC_CNC_REF, SMC_OUTQUEUE, CNC program files can be set up with the start position settings by POU programming and the start position settings by G92, but the operation differs depending on the direction specification method.

8.7.2 Example of Use: C-point Control and P-point Control

Interpolation control is basically executed by C-point Control and is executed by P-point Control only under specific conditions.

■ C-point control

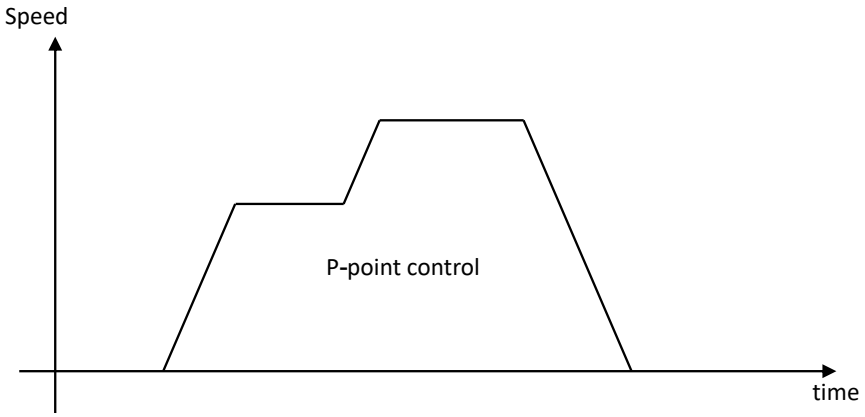
C-point control refers to control passing through a "Continuance Point". In this manual, this control is referred to as"C-point control"for the sake of convenience. This method is used to execute consecutive E-point controls by one-time startup.



■ P-point control

P-point control refers to control passing through a "Pass Point". In this manual, this control is referred to as"P-point control"for the sake of convenience.

This method is used when target multi-stage velocities are specified in a sequence of motions.



P-point motion is performed under any of the following conditions.

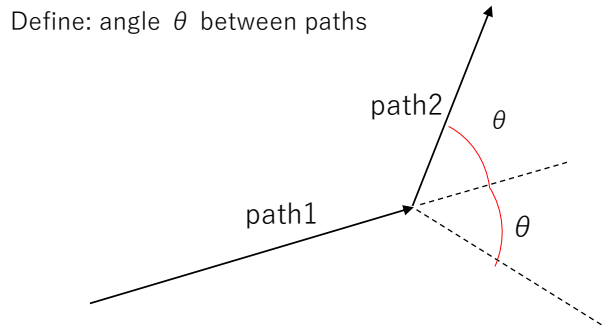
- Adjacent paths are linearly connected to each other.
- An angle that requires C-point control/P-point control is specified in the input variable dAngleTol of SMC_CheckVelocities.
- Smoothing is performed by means of SMC_SmoothPath or SMC_RoundPath.

Even if any of the P-point motion conditions is met, the connection between paths is established by C-point motion if the input variable bSingleStep of SMC_Interpolator is set or dwell time (G04) is used.

Info.

- The definition of the angle θ between the paths in the GM Programmer is as follows, indicating the angle of pass 1 and pass 2.

  The angle takes a value in the range of 0° ≤ θ ≤ 180°.



  If θ < dAngleTol, P-point control is applied.

  - Although dAngleTol allows for settings exceeding 180°, operation using such settings will be based on the value of 360° - dAngleTol.
  - Furthermore, if dAngleTol is set exceeding 360°, the remainder by dividing it by 360° is used.
  - For example, assume that dAngleTol is set to 700°. Since dAngleTol exceeds 360°, a remainder is calculated and 340° is obtained. Since the obtained value exceeds 180°, the control motion is the same as when it is set to 20° (360° - 340°).

  The angle θ between the paths used for judgment by SMC_SmoothPath or SMC_RoundPath dAngleTol is also the same.

■ Example of operation

G-code

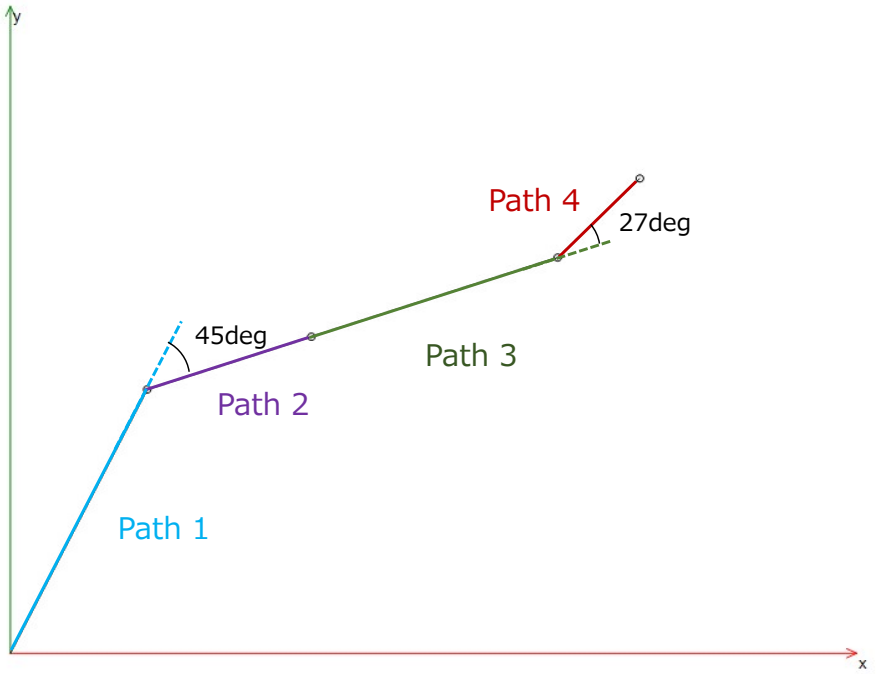
```
N000 G91 N010 G01 X5 Y10 F5 (Path 1) N020 G01 X6 Y2 (Path 2) N030 G01 X9 Y3 F
10 (Path 3) N040 G01 X3 Y3 (Path 4)
```

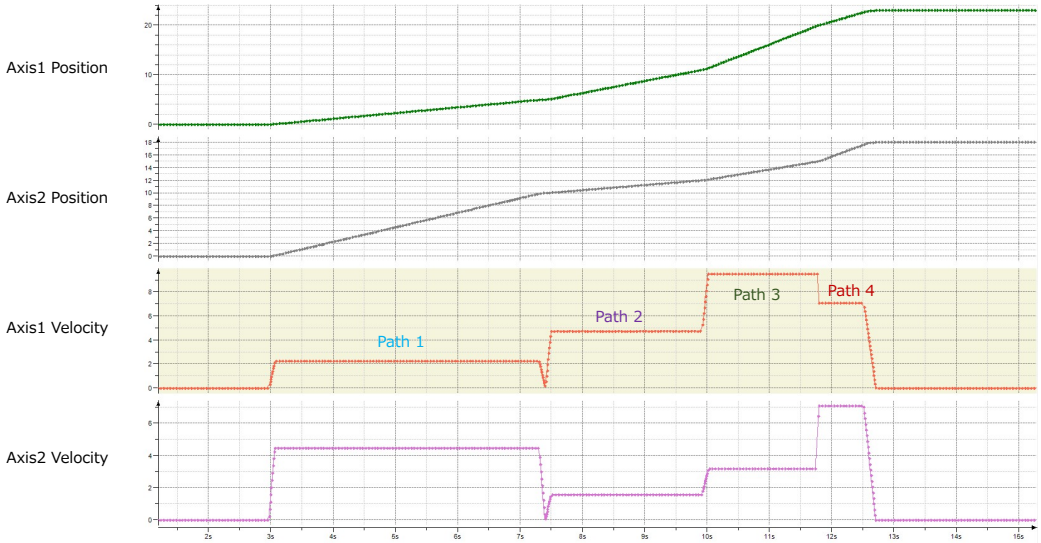
- [Setting example 1] The dAngleTol input is set to 30 (P-point motion if the angle between the paths is 30 degrees or less)
  - Variable declaration section:

```
// FBs ～ // Variables ～ lrAngletol : LREAL :=30; ～
```

  - Control section:

```
// CNC FBs Settings ～ SMC_CheckVelocities_0( bExecute:=bExe, poqDataIn:=
SMC_NCDecoder_0.poqDataOut, dAngleTol:=lrAngletol ); ～
```



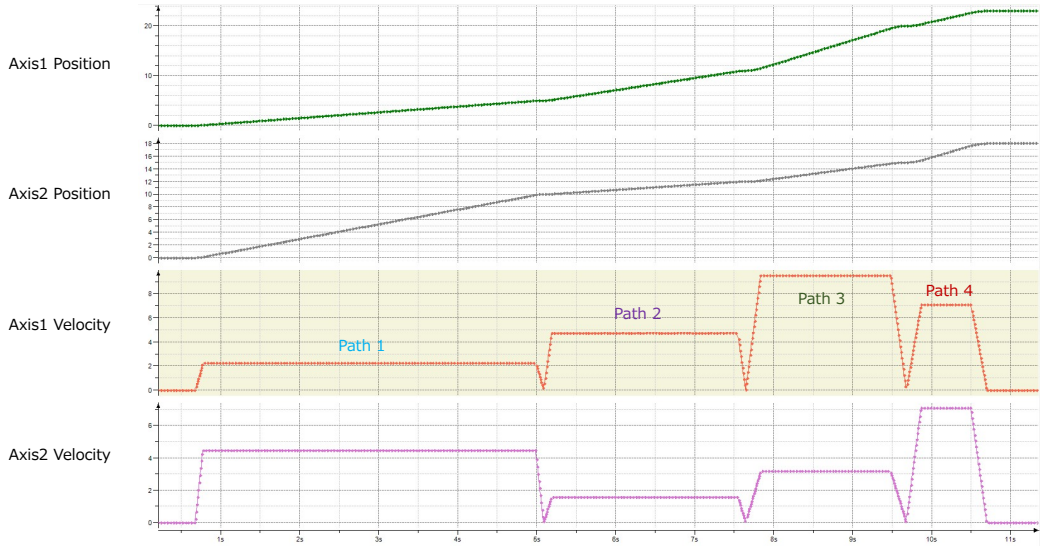


- In cases like Path 1-Path 2 in which the angle between the paths is the dAngleTol setting or greater, C-point motion is performed.
- In cases like Path 2-Path 3, for no angle between the paths that forms a linear connection, P-point motion is performed.
- In cases like Path 3-Path 4 in which the angle between the paths is the dAngleTol setting or less, P-point motion is performed.

● [Setting example 2] The bSingleStep input of SMC_Interpolator is set to TRUE

- Control section:

```
// CNC FBs Settings ～ SMC_Interpolator_0( bExecute:=bExe, poqDataIn:=SMC
_CheckVelocities_0.poqDataOut, dwIpoTime:=dwTime, bSingleStep:=TRUE ); ～
```



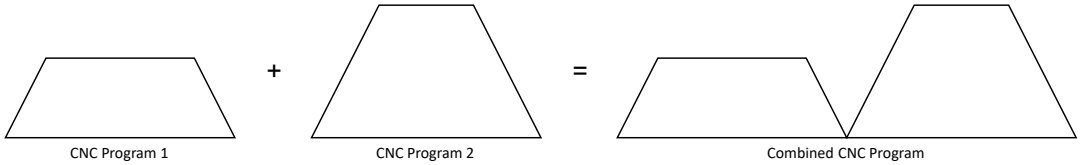
When the bSingleStep input of SMC_Interpolator is set to TRUE, every connection between the paths is established by C-point motion irrespective of the dAngleTol setting or a linear connection.

8.7.3 Example of Use: Repeating Processes

CNC programs can be joined by setting the bAppend argument of SMC_NCDecoder to TRUE and decoding each CNC program to be joined in order.

Motion between the joined CNC programs is performed by C-point control.

■ Joining CNC programs



[Setting example]

- G-code: CNC_Program1

```
N000 G01 X10 Y10 F30
```

- G-code: CNC_Program2

```
N000 G01 X30 Y30 F50
```

- Variable declaration section:

```
// FBs
    SMC_NCDecoder_0         : SMC_NCDecoder;
    SMC_CheckVelocities_0   : SMC_CheckVelocities;
    SMC_Interpolator_0      : SMC_Interpolator;
    SMC_ControlAxisByPos_0  : SMC_ControlAxisByPos;
    SMC_ControlAxisByPos_1  : SMC_ControlAxisByPos;
    SMC_ReadSetPosition_0   : SMC_ReadSetPosition;  //Read Current Position
    SMC_ReadSetPosition_1   : SMC_ReadSetPosition;
// Variables
    iState          : INT   :=0;          // Case Number
    Ncprogin        : SMC_CNC_REF;
    buf             : ARRAY[0..10] OF SMC_GEOINFO;  // Decord buffer
    piStartpos      : SMC_POSINFO;        // Start Position
    dwTime          : DWORD :=1000;
    xAvoidgaps_0    : BOOL  :=TRUE;
    xAvoidgaps_1    : BOOL  :=TRUE;
    bExe_ncd        : BOOL  :=FALSE;      // Execute for SMC_NCDecoder
    bExe_cv         : BOOL  :=FALSE;      // Execute for SMC_CheckVelocities
    bExe_ip         : BOOL  :=FALSE;      // Execute for SMC_Interpolator
    bStart          : BOOL  :=FALSE;      // Start Flag
```

- Control section:

```
// CNC FBs Settings
SMC_NCDecoder_0(
                ncprog:=Ncprogin,
                bExecute:=bExe_ncd,
                bAppend:=TRUE,
                piStartPosition:=piStartpos,
                nSizeOutQueue:=SIZEOF(buf),
```

```
                pbyBufferOutQueue:=ADR(buf)
);
SMC_CheckVelocities_0(
                        bExecute:=bExe_cv,
                        poqDataIn:=SMC_NCDecoder_0.poqDataOut
);
SMC_Interpolator_0(
                    bExecute:=bExe_ip,
                    poqDataIn:=SMC_CheckVelocities_0.poqDataOut,
                    dwIpoTime:=dwTime
);
SMC_ControlAxisByPos_0(
                        Axis:=Axis1,
                        iStatus:=SMC_Interpolator_0.iStatus,
                        bEnable:= SMC_Interpolator_0.bWorking,
                        bAvoidGaps:=xAvoidgaps_0,
                        fSetPosition:=SMC_Interpolator_0.piSetPosition.dX
);
SMC_ControlAxisByPos_1(
                        Axis:=Axis2,
                        iStatus:=SMC_Interpolator_0.iStatus,
                        bEnable:= SMC_Interpolator_0.bWorking,
                        bAvoidGaps:=xAvoidgaps_1,
                        fSetPosition:=SMC_Interpolator_0.piSetPosition.dY
);

// When Unreasonable movement is occured
IF SMC_ControlAxisByPos_0.bError = TRUE OR SMC_ControlAxisByPos_0.bStopIpo
= TRUE OR SMC_ControlAxisByPos_1.bError = TRUE OR SMC_ControlAxisByPos_1.bS
topIpo = TRUE THEN
    SMC_Interpolator_0.bEmergency_Stop:=TRUE;
ELSE
    SMC_Interpolator_0.bEmergency_Stop:=FALSE;
END_IF

IF bStart = TRUE THEN
    CASE iState OF
        0: // Read Current Position of Axises
            SMC_ReadSetPosition_0(
                                    Axis:=Axis1,
                                    Enable:=TRUE,
                                    Position=>piStartPos.dX
            );
            SMC_ReadSetPosition_1(
                                    Axis:=Axis2,
                                    Enable:=TRUE,
                                    Position=>piStartPos.dY
            );
            iState:=1;

        1: // Start decording
            Ncprogin:=CNC_Program1;
            bExe_ncd:=TRUE;
            iState:=2;

        2: // Change CNC Program
```

```
            IF SMC_NCDecoder_0.bDone = TRUE THEN
                bExe_ncd:=FALSE;
                Ncprogin:=CNC_Program2;
                iState:=3;
            END_IF

        3: // Start CNC motion
            bExe_ncd:=TRUE;
            bExe_cv:=TRUE;
            bExe_ip:=TRUE;

    END_CASE
END_IF
```

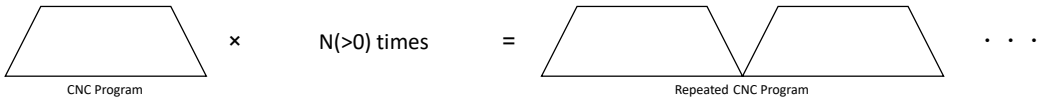
- The first CNC program is specified in SMC_NCDecoder and is decoded.
- After decoding of the first CNC program is completed, the second CNC program is specified and is decoded again. At the same time, other function blocks are performed and CNC control starts.

■ Repeating CNC programs

To repeat an identical CNC program, decode the CNC program for a number of repetitions you want, in a similar way to joining CNC programs together.

Motion between the joined CNC programs is performed by C-point control.

When decoding is executed, set the bAppend input of SMC_NCDecoder to TRUE.



[Setting example]

- G-code: CNC_Program

```
N000 G01 X10 Y0 F10
N010 G01 X10 Y10
N020 G01 X0 Y0
```

- Variable declaration section:

```
// FBs
～
// Variables
    iState          : INT   :=0;        // Case Number
    iCounter        : INT   :=0;        // Repeat counter
    iRepetition     : INT   :=3;        // Number of repetitions
～
```

- Control section:

```
// CNC FBs Settings
SMC_NCDecoder_0(
                ncprog:=CNC_Program,
                bExecute:=bExe_ncd,
                bAppend:=TRUE,
                piStartPosition:=piStartpos,
                nSizeOutQueue:=SIZEOF(buf),
```

```
                pbyBufferOutQueue:=ADR(buf)
);
～
IF bStart = TRUE THEN
    CASE iState OF
        0: // Read Current Position of Axises
            SMC_ReadSetPosition_0(
                                    Axis:=Axis1,
                                    Enable:=TRUE,
                                    Position=>piStartPos.dX
            );
            SMC_ReadSetPosition_1(
                                    Axis:=Axis2,
                                    Enable:=TRUE,
                                    Position=>piStartPos.dY
            );
            iState:=1;

        1: // Start CNC motion
            bExe_ncd:=TRUE;
            bExe_cv:=TRUE;
            bExe_ip:=TRUE;
            iState:=2;

        2: // Repeat decording
            IF SMC_NCDecoder_0.bDone = TRUE THEN
                iCounter:=iCounter + 1;
                IF iCounter >= iRepetition THEN
                    iState:=3;
                ELSE
                    bExe_ncd:=FALSE;
                    iState:=1;
                END_IF
            END_IF

    END_CASE
END_IF
```

- SMC_NCDecoder is executed for a number of repetitions (repeated three times in the example above).
- Other examples include a method for repeating SMC_CheckVelocities and SMC_Interpolator and a method for repeating programs by G20.

  For repetitive motion using G20, refer to "8.6.7 G20, G36, G37: Jump and Loop Process".

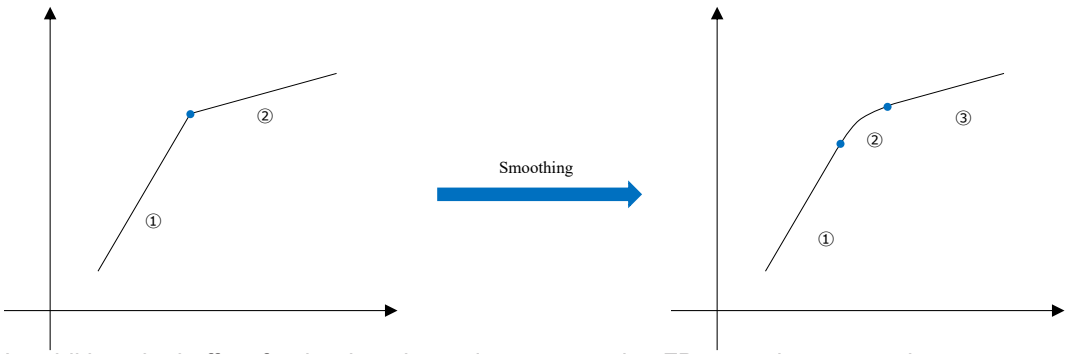
Info.

- If processes are repeated by the bAppend argument, a satisfactory buffer size must bef ensured. If the buffer size is small, a decoding error occurs.

8.7.4 Example of use: Pre-processing and tool correction

For CNC programs written with G-codes, pre-processing such as smoothing and tool radius correction can be performed. To perform pre-processing, use a combination of the required G-codes and function blocks.
Pre-processing creates and inserts path elements in the elements written in the CNC program. For example, if smoothing is performed on a linear motion path with two elements, the path will be converted to a path with a total of three elements, which consists of straight line and a curved line portions.

This requires more data arrays (buffers) for SMC_GEOINFO than when pre-processing is not performed.



In addition, the buffers for the decoder and pre-processing FBs must be prepared independently of each other. Setting the same buffer data array for different FBs will not produce a correct path due to mixed processing data for each FB.

- Acceptable example

```
SMC_NCDecoder_0(
    ～
nSizeOutQueue:=SIZEOF(buf_ncd),
pbyBufferOutQueue:=ADR(buf_ncd),
    ～
SMC_SmoothPath_0(
    ～
nSizeOutQueue:=SIZEOF(buf_sp),
pbyBufferOutQueue:=ADR(buf_sp),
```

- Unacceptable example

```
SMC_NCDecoder_0(
    ～
nSizeOutQueue:=SIZEOF(buf_ncd),
pbyBufferOutQueue:=ADR(buf_ncd),
    ～
SMC_SmoothPath_0(
    ～
nSizeOutQueue:=SIZEOF(buf_ncd),
pbyBufferOutQueue:=ADR(buf_ncd),
```

■ Tool correction using CNC programs

As an example of pre-processing, an example of tool correction is shown below.

By using SMC_ToolRadiusCorr, which executes pre-processing for tool radius correction and SMC_ToolLengthCorr, which executes tool length correction, it is possible to perform offset conversion of the motion path according to the tool without making major changes to the CNC program.

[Setting example]

- Variable declaration section:

```
// FBs
    SMC_NCDecoder_0         : SMC_NCDecoder;
    SMC_ToolRadiusCorr_0    : SMC_ToolRadiusCorr;
    SMC_CheckVelocities_0   : SMC_CheckVelocities;
    SMC_Interpolator_0      : SMC_Interpolator;
    SMC_ToolLengthCorr_0    : SMC_ToolLengthCorr;
    SMC_TRAFO_Gantry3_0        : SMC_TRAFO_Gantry3;
    SMC_ControlAxisByPos_0    : SMC_ControlAxisByPos;
    SMC_ControlAxisByPos_1    : SMC_ControlAxisByPos;
    SMC_ControlAxisByPos_2    : SMC_ControlAxisByPos;
    SMC_ReadSetPosition_0   : SMC_ReadSetPosition;  //Read Current Position
    SMC_ReadSetPosition_1   : SMC_ReadSetPosition;
    SMC_ReadSetPosition_2   : SMC_ReadSetPosition;
// Variables
    iState              : INT   :=0;         // Case Number
    buf_ncd             : ARRAY[0..19] OF SMC_GEOINFO;  // Decord buffer
    buf_trc             : ARRAY[0..19] OF SMC_GEOINFO;  // ToolRadiusCorr b
uffer
    piStartpos          : SMC_POSINFO;         // Start Position
    vStartToolLen       : SMC_VECTOR3D;        // Tool Length
    eOriConv            : SMC_ORI_CONVENTION;
    dwTime              : DWORD :=1000;
    bExe                : BOOL  :=FALSE;     // Execute for FBs
    bStart              : BOOL  :=FALSE;     // Start Flag
```

- Control section:

```
// CNC FBs Settings
SMC_NCDecoder_0(
                ncprog:=CNC_Sample,
                bExecute:=bExe,
                piStartPosition:=piStartpos,
                vStartToolLength:=vStartToolLen,
                nSizeOutQueue:=SIZEOF(buf_ncd),
                pbyBufferOutQueue:=ADR(buf_ncd),
                eOriConv:=eOriConv
);
SMC_ToolRadiusCorr_0(
                bExecute:=bExe,
                poqDataIn:=SMC_NCDecoder_0.poqDataOut,
                nSizeOutQueue:=SIZEOF(buf_trc),
                pbyBufferOutQueue:=ADR(buf_trc)
);
SMC_CheckVelocities_0(
                        bExecute:=bExe,
                        poqDataIn:=SMC_ToolRadiusCorr_0.poqDataOut
);
SMC_Interpolator_0(
```

```
                    bExecute:=bExe,
                    poqDataIn:=SMC_CheckVelocities_0.poqDataOut,
                    dwIpoTime:=dwTime
);
SMC_ToolLengthCorr_0(
                        pi:=SMC_Interpolator_0.piSetPosition,
                        adToolLength:=SMC_Interpolator_0.adToolLength,
                        eOriConv:=eOriConv,
                        bForwardTrafo:=FALSE
);
SMC_TRAFO_Gantry3_0(
                    pi:=SMC_ToolLengthCorr_0.piOut
);
SMC_ControlAxisByPos_0(
                        Axis:=Axis1,
                        iStatus:=SMC_Interpolator_0.iStatus,
                        bEnable:=SMC_Interpolator_0.bWorking,
                        bAvoidGaps:=TRUE,
                        fSetPosition:=SMC_TRAFO_Gantry3_0.dX,
                        fGapVelocity:=5,
);
SMC_ControlAxisByPos_1(
                        Axis:=Axis2,
                        iStatus:=SMC_Interpolator_0.iStatus,
                        bEnable:=SMC_Interpolator_0.bWorking,
                        bAvoidGaps:=TRUE,
                        fSetPosition:=SMC_TRAFO_Gantry3_0.dY,
                        fGapVelocity:=5,
);
SMC_ControlAxisByPos_2(
                        Axis:=Axis3,
                        iStatus:=SMC_Interpolator_0.iStatus,
                        bEnable:=SMC_Interpolator_0.bWorking,
                        bAvoidGaps:=TRUE,
                        fSetPosition:=SMC_TRAFO_Gantry3_0.dZ,
                        fGapVelocity:=5,
);

// When Unreasonable movement is occured
IF SMC_ControlAxisByPos_0.bError = TRUE OR SMC_ControlAxisByPos_0.bStopIpo
= TRUE
    OR SMC_ControlAxisByPos_1.bError = TRUE OR SMC_ControlAxisByPos_1.bStop
Ipo = TRUE
        OR SMC_ControlAxisByPos_2.bError = TRUE OR SMC_ControlAxisByPos_2.b
StopIpo = TRUE THEN
    SMC_Interpolator_0.bEmergency_Stop:=TRUE;
ELSE
    SMC_Interpolator_0.bEmergency_Stop:=FALSE;
END_IF

IF bStart = TRUE THEN
    CASE iState OF
        0: // Read Current Position of Axises
            SMC_ReadSetPosition_0(
                                    Axis:=Axis1,
                                    Enable:=TRUE,
```

```
                                    Position=>piStartPos.dX
            );
            SMC_ReadSetPosition_1(
                                    Axis:=Axis2,
                                    Enable:=TRUE,
                                    Position=>piStartPos.dY
            );
            SMC_ReadSetPosition_2(
                                    Axis:=Axis3,
                                    Enable:=TRUE,
                                    Position=>piStartPos.dZ
            );
            iState:=1;

        1: // Prepare parameters
            vStartToolLen.dX:=0;
            vStartToolLen.dY:=0;
            vStartToolLen.dZ:=10;
            eOriConv:=SM3_CNC.SMC_ORI_CONVENTION.ZYZ;
            iState:=2;

        2: // Start CNC motion
            bExe:=TRUE;
    END_CASE
END_IF
```

Explanation of control section

- SMC_ToolRadiusCorr executes tool radius correction on the CNC program decoded by SMC_NCDecoder.
- SMC_Interpolator executes interpolation operations on the path offset by the tool radius by SMC_ToolRadiusCorr.
- SMC_ToolLengthCorr executes tool length correction and conversion by the kinematics function on the interpolation data calculated by SMC_Interpolator.

9 Motion Control Function Blocks (Motion Communication Control)

This section describes function blocks used to perform communication control.

9.1 RTEX/EtherCAT Common

9.1.1 SetCommunicationState (Set Device Communication State)

This is a method used to change the communication state of a device. The state can be changed to any of start, stop, and reset. Add the name of the device before this method and write like DeviceName.SetCommunicationState.

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	eRequestedState	DEVICE_TRANSITION_STATE	2(STOP)	Specifies a device state to be set.
Output	SetCommunicationState	DED.ERROR	NO_ERROR	An error ID is output. "11.16.4 DED.ERROR (Error Code)"

■ DEVICE_TRANSITION_STATE (Enumeration type)

States to which the device transitions are shown.

Name	Value	Description
START	1	Puts the device in run mode.
STOP	2	Puts the device in stop mode.
RESET	3	Restarts the device and reconfigures it.

■ List of functions provided by devices

The following table shows a list of devices that support SetCommunicationState.

Scope	Device name	Transition to START	Transition to STOP	Transition to RESET
ECAT	EtherCAT_Master_SoftMotion	○	○	○
	Servo amplifier	×	×	×
	Real axis	×	×	×
RTEX	RTEX_Master	×	×	×
	Servo amplifier	×	×	×
	Real axis	×	×	×

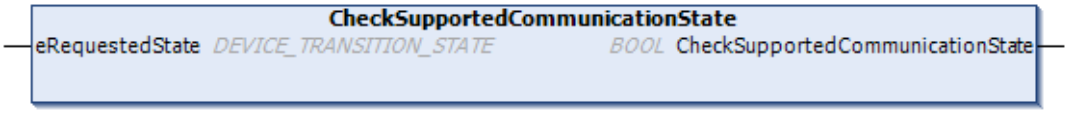
Info.

- To use this method, it is necessary to select “Enable diagnosis for device” checkbox on the “PLC Settings” tab in the Device object.
- For examples of use, refer to "9.4.8 Sample Example: Stop/Restart EtherCAT Master Communication".

9.1.2 CheckSupportedCommunicationState (Check if Device Provides Communication State Setting)

This method is used to query a device to check whether or not it provides a transition to a requested setting using SetCommunicationState(Method). TRUE is output if the device supports the transition to the specified communication state. Add the name of the device before this method and write like DeviceName.CheckSupportedCommunicationState.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	eRequestedState	DEVICE_TRANSITION_STATE	2(STOP)	Specifies a communication state you want the device to transition to.
Output	CheckSupportedCommunicationState	BOOL	FALSE	TRUE: Supported FALSE: Not supported

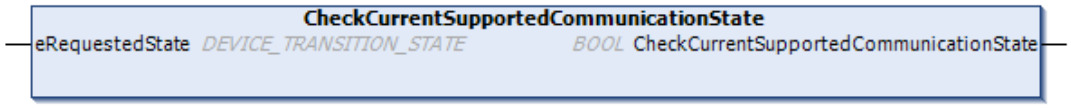
Info.

- To use this method, it is necessary to select “Enable diagnosis for device” checkbox on the “PLC Settings” tab in the Device object.
- For examples of use, refer to "9.4.8 Sample Example: Stop/Restart EtherCAT Master Communication".

9.1.3 CheckCurrentSupportedCommunicationState (Check if Device in Current State Provides Communication State Setting)

This method is used to query a device to check whether or not the device in the current state provides a transition to a requested setting using SetCommunicationState(Method). TRUE is output if the device supports the transition to the specified communication state. Add the name of the device before this method and write like DeviceName.CheckCurrentSupportedCommunicationState.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	eRequestedState	DEVICE_TRANSITION_STATE	2(STOP)	Specifies a communication state you want the device to transition to.
Output	CheckSupportedCommunicationState	BOOL	FALSE	TRUE: Allowed to transition FALSE: Not allowed to transition

Info.

- To use this method, it is necessary to select “Enable diagnosis for device” checkbox on the “PLC Settings” tab in the Device object.
- For examples of use, refer to "9.4.8 Sample Example: Stop/Restart EtherCAT Master Communication".

9.2 RTEX

9.2.1 Types of Data To Be Handled by AMP Function Blocks

Item	Description	Related function blocks
AMP alarm	This is an AMP alarm that occurs in AMP operation.	RTEX_ClearAmpAlarm RTEX_ReadAmpAlarm RTEX_ReadAmpState
AMP warning	This is an AMP warning that occurs in AMP operation. This occurs before the AMP alarm. If the situation worsens, an AMP alarm occurs.	
Monitor data	This is monitor data (position deviation, load percentage, etc.) of the RTEX communication data.	RTEX_ReadAmpData
AMP parameter	This is configuration data of the AMP device itself.	RTEX_ReadAmpParameter RTEX_WriteAmpParameter
Multi-turn data	There are two types of data in the data read by the absolute encoder (23 bit/r): one type is single-turn data that indicates the position within one motor rotation and the other is multi-turn data that counts one for one turn.	RTEX_ClearAmpMultiTurnData
Deviation counter	This is a processing part in the AMP that receives move commands to the AMP. The motor moves according to the commands accumulated in the deviation counter. The commands used for the motor movement are deleted from the deviation counter. The amount of commands accumulated in the deviation counter is called the position deviation.	RTEX_ClearAmpPositionalDeviation
Limit switch	This data is collected to monitor the POT and NOT states of the AMP.	RTEX_ReadNot, RTEX_ReadPot

9.2.2 RTEX_ClearAmpAlarm (Clear Amplifier Alarm)

This is a function block (FB) that clears the AMP alarm. It deletes the alarm or warning that has occurred in the AMP.

■ Icon

■ Parameter

Scope	Name	Type	Initial	Description
Input / Output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Stops processing.
Output	Done	BOOL	FALSE	TRUE:Clear processing completed
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE:Suspension from other FB occurred
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorId	SMC_ERROR	0	An error ID is output.
	Alarm	ALARM_CODE	-	A deleted alarm code is output.
	Warning	WARNING_CODE	-	A deleted warning code is output.

■ ALARM_CODE (Union)

Member	Type	Description
uiAlarmCode	UINT	Alarm code
tAlarmCodeMember	ALARM_WARNING_CODES	Main alarm code and sub alarm code

■ WARNING_CODE (Union)

Member	Type	Description
uiWarningCode	UINT	Warning code
tWarningCodeMember	ALARM_WARNING_CODES	Main warning code (warning number) and sub warning code (0)

■ ALARM_WARNING_CODES (Structure)

Member	Type	Description
byMainCode	BYTE	Main code
bySubCode	BYTE	Sub code

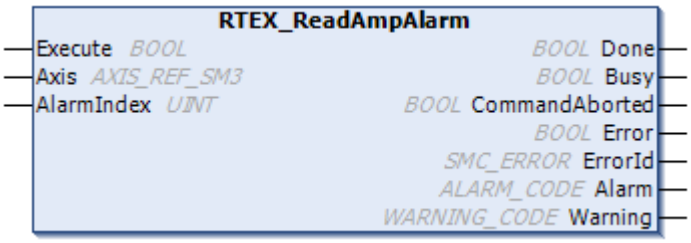
REFERENCE

13.2.2 Alarm Codes
13.2.3 Warning Codes

9.2.3 RTEX_ReadAmpAlarm (Read Amplifier Alarm)

This is a function block (FB) that reads the AMP alarm. It reads the information of the alarm or warning that has occurred in the AMP.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Stops processing.
	AlarmIndex	UINT	-	Specifies the history number (0 to 14). 0 is given for the latest history.
Output	Done	BOOL	FALSE	TRUE: Reading is completed.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from another FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorId	SMC_ERROR	0	An error ID is output.
	Alarm	ALARM_CODE"13.2.2 Alarm Codes"	-	A read alarm code is output.
	Warning	WARNING_CODE"13.2.3 Warning Codes"	-	A read warning code is output.

REFERENCE

13.2.2 Alarm Codes
13.2.3 Warning Codes

9.2.4 RTEX_ReadAmpState (Amplifier Alarm Status)

This is a function block (FB) that reads the AMP alarm state. It outputs the information and state of the axis where the AMP alarm or warning has occurred.

■ Icon

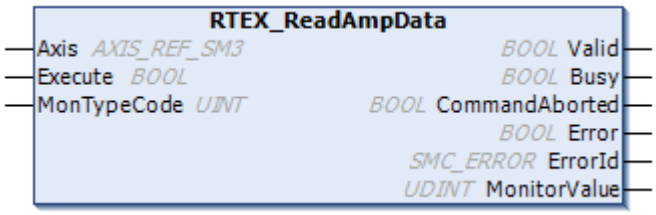
■ Parameter

Scope	Name	Type	Initial	Description
Output	NumberOfSlaves	UDINT	-	The number of axes connected (1 to 32) is output.
	AlarmState	DWORD	-	The MAC-ID (0 to 31) where the AMP alarm has occurred is output.
	WarningState	DWORD	-	The MAC-ID (0 to 31) where the AMP warning has occurred is output.

9.2.5 RTEX_ReadAmpData (Amplifier Monitor)

This is a function block (FB) that reads the monitor data of the AMP. It reads various monitor data of the AMP.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Stops processing.
	MonTypeCode	UINT		Specifies the type code for the monitor command.
Output	Valid	BOOL	FALSE	TRUE:Monitor processing completed
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE:Suspension from other FB occurred
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorId	SMC_ERROR	0	An error ID is output.
	MonitorValue	UDINT	-	Read monitor command

Info.

- This function block is designed specifically for RTEX 32-byte mode.
- Set the project file and servo amplifier to 32-byte mode. If they are in 16-byte mode, an error will occur.
- Declare only one instance for this function block. Due to RTEX communication specifications, multiple instances cannot be simultaneously called.

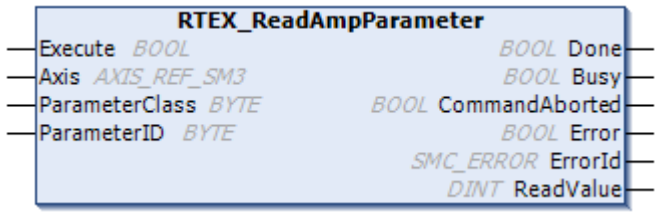
REFERENCE

13.4 Monitor Commands

9.2.6 RTEX_ReadAmpParameter (Read Amplifier Parameter)

This is a function block (FB) that reads the AMP parameter.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Stops processing.
	ParameterClass	BYTE	-	Specifies the AMP parameter classification.
	ParameterID	BYTE	-	Specifies the AMP parameter number.
Output	Done	BOOL	FALSE	TRUE:Reading processing completed
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE:Suspension from other FB occurred
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorId	SMC_ERROR	0	An error ID is output.
	ReadValue	DINT	-	Read AMP parameter value

REFERENCE

13.2.1 RTEX Error ID
13.3 List of AMP Parameters

9.2.7 RTEX_WriteAmpParameter (Write Amplifier Parameter)

This is a function block (FB) that writes the AMP parameter.

■ Icon

■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Stops processing.
	ParameterClass	BYTE	-	Specifies the AMP parameter classification.
	ParameterID	BYTE	-	Specifies the AMP parameter number.
	WriteValue	DINT	-	Value to be written in the AMP parameter
Output	Done	BOOL	FALSE	TRUE:Writing processing completed
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE:Suspension from other FB occurred
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorId	SMC_ERROR	0	An error ID is output.

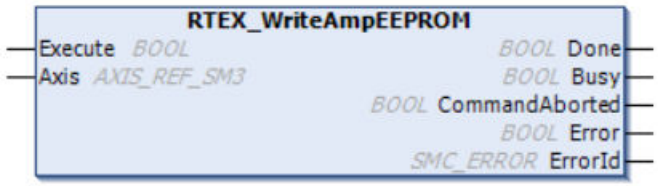
REFERENCE

13.2.1 RTEX Error ID
13.3 List of AMP Parameters

9.2.8 RTEX_WriteAmpEEPROM (Write Amplifier EEPROM)

This is a function block (FB) that writes the servo amplifier parameters to EEPROM.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Stops processing.
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Output	Done	BOOL	FALSE	TRUE: Writing is completed.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	Error ID output

9.2.9 RTEX_Reset (Reset RTEX)

Resets the entire RTEX network.

■ Icon

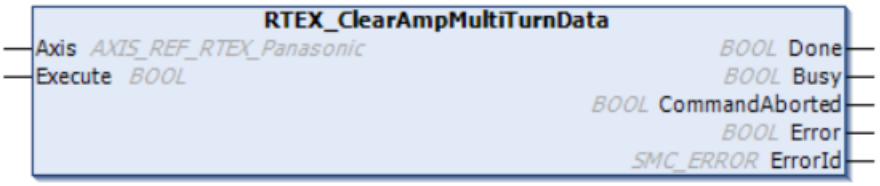
■ Parameter

Scope	Name	Type	Default	Description
Input / output	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge.
Output	Done	BOOL	FALSE	TRUE: Reset done
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	Error ID output

9.2.10 RTEX_ClearAmpMultiTurnData (Clear Amplifier Multi-turn Data)

This is a function block (FB) that clears the multi-turn data of the AMP.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_RTEX_Panasonic	-	Specifies the axis.
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Stops processing.
Output	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Done	BOOL	FALSE	TRUE: Clearing is completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	An error ID is output.

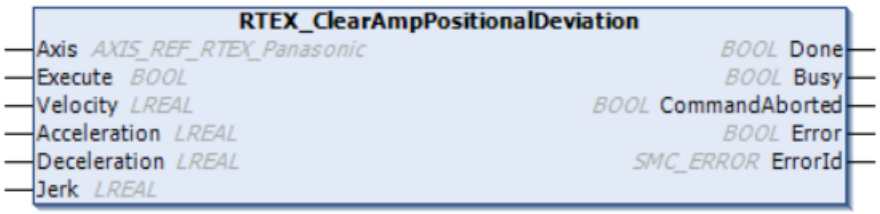
The PMC_ClearAmpMultiTurnData function block outputs the following errors.

Error	Description
SMC_WRONG_CONTROLLER_MODE	Executed in a mode other than the position control mode. Change to SMC_position using SMC_SetControllerMode.
SMC_DI_HOMING_ERROR	The encoder used is an Incremental encoder.
SMC_AXIS_NOT_READY_FOR_MOTION	The axis is in a state where RTEX_ClearAmpMultiTurnData cannot be executed. It can be executed only when set to Disabled or Errorstop.
SMC_REGULATOR_OR_START_NOT_SET	The axis is in a servo ON state.
SMC_AXIS_REF_CHANGED_DURING_OPERATION	The Axis was changed during operation.

9.2.11 RTEX_ClearAmpPositionalDeviation (Clear Amplifier Deviation Counter)

This is a function block (FB) that clears the deviation counter of the AMP. It deletes the position deviation data in the deviation counter of the AMP.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_RTEX_Panasonic	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Velocity(Note 1)	LREAL		Information required to execute MC_MoveAbsolute
	Acceleration(Note 1)	LREAL		Information required to execute MC_MoveAbsolute
	Deceleration(Note 1)	LREAL		Information required to execute MC_MoveAbsolute
	Jerk(Note 1)	LREAL		Information required to execute MC_MoveAbsolute
Output	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Done	BOOL	FALSE	TRUE: Clearing is completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	An error ID is output.

(Note 1) This function block internally substitutes the command position with an actual position to call MC_MoveAbsolute and, therefore, requires parameters including Velocity, Acceleration, Deceleration, and Jerk.

The RTEX_ClearAmpPositionalDeviation function block outputs the following errors.

Error	Description
SMC_WRONG_CONTROLLER_MODE	Executed in a mode other than the position control mode. Change to SMC_position using SMC_SetControllerMode.

Error	Description
SMC_AXIS_NOT_READY_FOR_MOTION	The axis is in a state where RTEX_ClearAmpPositionalDeviation cannot be executed. It can be executed only at the standstill state.
SMC_REGULATOR_OR_START_NOT_SET	The axis is in a servo ON state.
SMC_PP_WRONG_AXIS_TYPE	The axis is a virtual axis.

9.2.12 RTEX_GetTrackingCommandError (Read RTEX Command Send Statistics Information)

The RTEX periodically sends commands. With the GM1 specifications, when the MotionTask cycle time exceeds the control cycle, the command position for the servo amplifier is not updated for that cycle. (This is called a lost RTEX command.) This function block measures the number of sent RTEX commands and the number of lost RTEX commands. Using this function, you can check if the command position is updated normally for every cycle.

■ Icon

■ Parameter

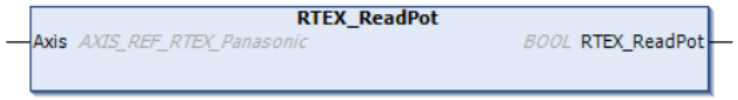
Scope		Definition	Value	Description
Input	Execute	---		Execute = TRUE: Starts measurement when triggered. Execute = FALSE: Clears output.
	ContinuousLossCondtion	Continuous command loss condition	0: Disabled	If the command loss continuously occurs at ContinuousLossCondtion cycle, bContinuousLossError turns TRUE.
	MesurementCycleCondition	Command loss statistical measurement cycle condition	0: Disabled	If the command loss occurs as many times as specified in LossRateCondition during the MesurementCycleCondition period, bLossRateError turns TRUE.
	LossRateCondition	Under measurement	0 to 100%	
Output	Busy	Under measurement		---
	NumberOfSendCommand	Total number of commands sent		Returns a value when Execute is TRUE. Clears when Execute is FALSE.
	NumberOfLossCommand	Total number of commands lost		Returns a value when Execute is TRUE. Clears when Execute is FALSE.
	bContinuousLossError	Occurrence of a continuous command loss error		Occurrence of a condition error of ContinuousLossCondtion
	bLossRateError	Occurrence of a command loss statistics error		Occurrence of a condition error of MesurementCycleCondition or LossRateCondition

(Note 1) If the number of frames exceeds 32 bits, normal value is not returned.

9.2.13 RTEX_ReadPot (Read POT of Amplifier)

This is a function that reads the POT state of the amplifier.

■ Icon



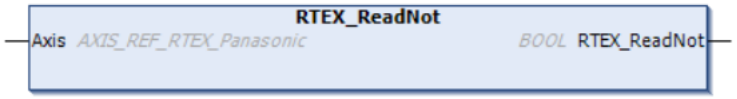
■ Parameter

Type	Parameter name	Type	Description
I/O	Axis	AXIS_REF_RTEX_Panasonic	Specifies the axis.
Output	RTEX_ReadPot	BOOL	TRUE: POT is ON.

9.2.14 RTEX_ReadNot (Read NOT of Amplifier)

This is a function that reads the NOT state of the amplifier.

■ Icon



■ Parameter

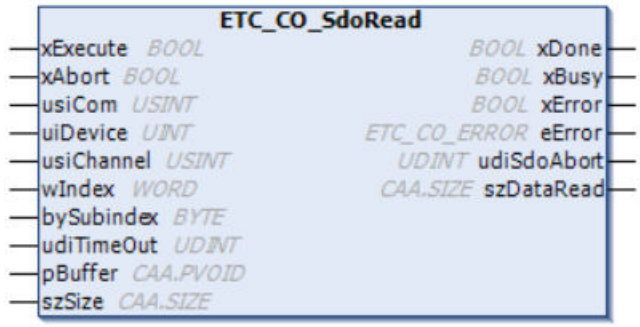
Type	Parameter name	Type	Description
I/O	Axis	AXIS_REF_RTEX_Panasonic	Specifies the axis.
Output	RTEX_ReadNot	BOOL	TRUE: NOT is ON.

9.3 EtherCAT

9.3.1 ETC_CO_SdoRead (Read Slave Parameter)

This is a function block (FB) that reads the EtherCAT slave parameters. Unlike ETC_CO_SdoRead4, this FB supports parameters longer than 4 bytes. Specify parameters to be read using the index and sub-index used for the object directory.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	TRUE: Interrupts processing and resets output.
	usiCom	USINT	1	1 (Fixed)
	uiDevice	UINT	0	Physical slave address
	usiChannel	USINT	1	Reserved
	wIndex	WORD	0	Parameter index in the object directory(Note 1)
	bySubIndex	BYTE	0	Parameter sub-index in the object directory(Note 1)
	udiTimeout	UDINT	0	Timeout (Unit: ms)
	pBuffer	CAA.PVOID	0	Pointer to the buffer that stores read data
	szSize	CAA.SIZE	0	Size of the buffer that stores data
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.

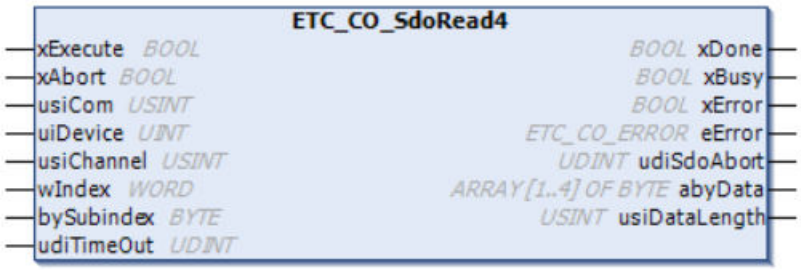
Scope	Name	Type	Default	Description
	eError	ETC_CO_ERROR	ETC_CO_NO_ERROR	Error ID output
	udiSdoAbort	UDINT	0	Abort code received from the slave device
	szDataRead	CAA.SIZE	0	Number of bytes read normally

(Note 1) The parameter content differs according to the slave. Refer to the manuals of corresponding slave devices.

9.3.2 ETC_CO_SdoRead4 (Read Four Bytes of Slave Parameter)

This is a function block (FB) that reads the EtherCAT slave parameters. Unlike ETC_CO_SdoRead, this FB supports only parameters with 4 bytes or less. Specify parameters to be read using the index and sub-index used for the object directory.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	TRUE: Interrupts processing and resets output.
	usiCom	USINT	1	1 (Fixed)
	uiDevice	UINT	1	Physical slave address
	usiChannel	USINT	1	Reserved
	wIndex	WORD	0	Parameter index in the object directory(Note 1)
	bySubIndex	BYTE	0	Parameter sub-index in the object directory(Note 1)
	udiTimeout	UDINT	0	Timeout (Unit: ms)
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.

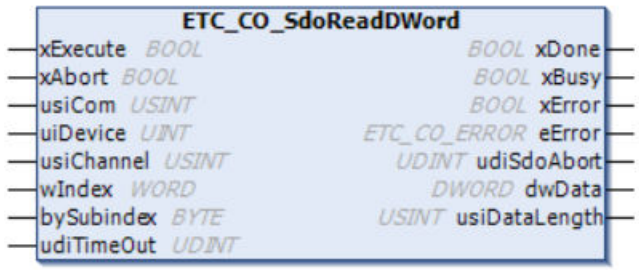
Scope	Name	Type	Default	Description
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	ETC_CO_ERROR	ETC_CO_NO_ERROR	Error ID output
	udiSdoAbort	UDINT	0	Abort code received from the slave device
	abyData	ARRAY [1..4] OF BYTE	-	Read data storage location
	usiDataLength	USINT	0	Number of read bytes

(Note 1) The parameter content differs according to the slave. Refer to the manuals of corresponding slave devices.

9.3.3 ETC_CO_SdoReadDWord (Read Double Word of Slave Parameter)

Just like ETC_CO_SdoRead4, this is a function block (FB) that reads the EtherCAT slave parameters. The read data is stored in DWORD (dwData), not in an array. Since byte swapping is automatically executed, read data can be directly used.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	TRUE: Interrupts processing and resets output.
	usiCom	USINT	1	1 (Fixed)
	uiDevice	UINT	0	Physical slave address
	usiChannel	USINT	1	Reserved
	wIndex	WORD	0	Parameter index in the object directory(Note 1)

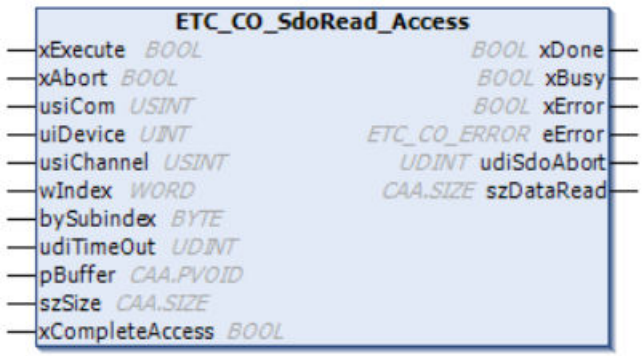
Scope	Name	Type	Default	Description
	bySubIndex	BYTE	0	Parameter sub-index in the object directory(Note 1)
	udiTimeout	UDINT	0	Timeout (Unit: ms)
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	ETC_CO_ERROR	ETC_CO_NO_ERROR	Error ID output
	udiSdoAbort	UDINT	0	Abort code received from the slave device
	dwData	DWORD	0	Read data storage location
	usiDataLength	USINT	0	Number of read bytes

(Note 1) The parameter content differs according to the slave. Refer to the manuals of corresponding slave devices.

9.3.4 ETC_CO_SdoRead_Access (Read Slave Parameter Index)

Just like ETC_CO_SdoRead, this is a function block (FB) that reads the EtherCAT slave parameters. By setting the xCompleteAccess input to TRUE and the bySubIndex input to 0, you can read complete indexes including all entries.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	TRUE: Interrupts processing and resets output.

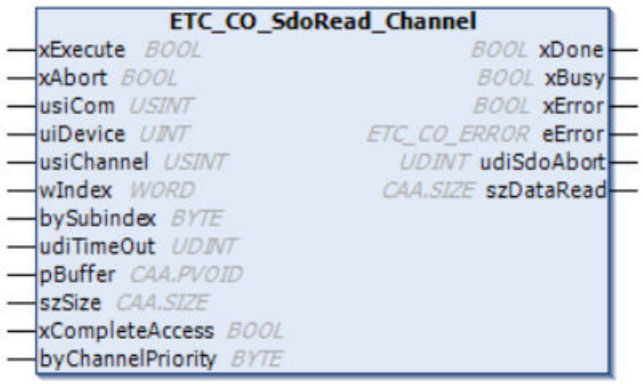
Scope	Name	Type	Default	Description
	usiCom	USINT	1	1 (Fixed)
	uiDevice	UINT	0	Physical slave address
	usiChannel	USINT	1	Reserved
	wIndex	WORD	0	Parameter index in the object directory(Note 1)
	bySubIndex	BYTE	0	Parameter sub-index in the object directory(Note 1)
	udiTimeout	UDINT	0	Timeout (Unit: ms)
	pBuffer	CAA.PVOID	0	Pointer to the buffer that stores read data
	szSize	CAA.SIZE	0	Size of the buffer that stores data
	xCompleteAccess	BOOL	FALSE	TRUE: Accesses all sub-indexes within the specified index.
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	ETC_CO_ERROR	ETC_CO_NO_ERROR	Error ID output
	udiSdoAbort	UDINT	0	Abort code received from the slave device
	szDataRead	CAA.SIZE	0	Number of read bytes

(Note 1) The parameter content differs according to the slave. Refer to the manuals of corresponding slave devices.

9.3.5 ETC_CO_SdoRead_Channel (Read Priority Specification of Slave Parameter)

Just like ETC_CO_SdoRead_Access, this is a function block (FB) that reads the EtherCAT slave parameters. By using the byChannelPriority (BYTE) input, you can specify the channel and priority using a CoE mailbox message. Specify the channel with the first 6 bits (bit0 to bit5) and the priority with the last 2 bits (bit6 and bit7).

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	TRUE: Interrupts processing and resets output.
	usiCom	USINT	1	1 (Fixed)
	uiDevice	UINT	0	Physical slave address
	usiChannel	USINT	1	Reserved
	wIndex	WORD	0	Parameter index in the object directory(Note 1)
	bySubIndex	BYTE	0	Parameter sub-index in the object directory(Note 1)
	udiTimeout	UDINT	0	Timeout (Unit: ms)
	pBuffer	CAA.PVOID	0	Pointer to the buffer that stores read data
	szSize	CAA.SIZE	0	Size of the buffer that stores data
	xCompleteAccess	BOOL	FALSE	TRUE: Accesses all sub-indexes within the specified index.
	byChannelPriority	BYTE	0	Specifies the channel and priority using a CoE mailbox message.
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.

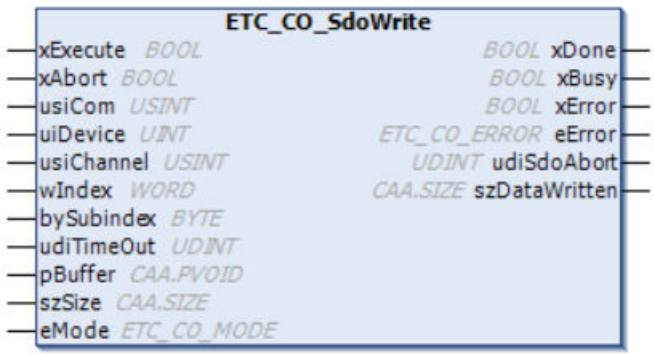
Scope	Name	Type	Default	Description
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	ETC_CO_ERROR	ETC_CO_NO_ERROR	Error ID output
	udiSdoAbort	UDINT	0	Abort code received from the slave device
	szDataRead	CAA.SIZE	0	Number of read bytes

(Note 1) The parameter content differs according to the slave. Refer to the manuals of corresponding slave devices.

9.3.6 ETC_CO_SdoWrite (Write Slave Parameter)

This is a function block (FB) that writes the EtherCAT slave parameters. Unlike ETC_CO_SdoWrite4, this FB supports parameters longer than 4 bytes. Specify parameters to be written using the index and sub-index used for the object directory.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	TRUE: Interrupts processing and resets output.
	usiCom	USINT	1	1 (Fixed)
	uiDevice	UINT	0	Physical slave address
	usiChannel	USINT	1	Reserved
	wIndex	WORD	0	Parameter index in the object directory(Note 1)

Scope	Name	Type	Default	Description
	bySubIndex	BYTE	0	Parameter sub-index in the object directory(Note 1)
	udiTimeout	UDINT	0	Timeout (Unit: ms)
	pBuffer	CAA.PVOID	0	Pointer to the buffer storing write data
	szSize	CAA.SIZE	0	Number of written bytes
	eMode	ETC_CO_MODE	ETC_CO_AUTO	Transmission mode
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	ETC_CO_ERROR	ETC_CO_NO_ERROR	Error ID output
	udiSdoAbort	UDINT	0	Abort code received from the slave device
	szDataWritten	CAA.SIZE	0	Number of bytes written normally

(Note 1) The parameter content differs according to the slave. Refer to the manuals of corresponding slave devices.

■ ETC_CO_ERROR (Union type)

Member	Type	Description
ETC_CO_NO_ERROR	WORD	No error
ETC_CO_FIRST_ERROR	WORD	Check udiSdoAbort for the cause of errors.
ETC_CO_OTHER_ERROR	WORD	The master is not found.
ETC_CO_DATA_OVERFLOW	WORD	ETC_CO_Expedited and size exceed 4.
ETC_CO_TIMEOUT	WORD	The time limit is exceeded.
ETC_CO_FIRST_MF	WORD	Not used
ETC_CO_LAST_ERROR	WORD	Not used

■ ETC_CO_MODE (Union type)

Member	Type	Description
ETC_CO_AUTO	WORD	Mode is selected automatically.
ETC_CO_EXPEDITED	WORD	Expedited transfer
ETC_CO_SEGMENTED	WORD	Segmented transfer

9.3.7 ETC_CO_SdoWrite4 (Write Four Bytes of Slave Parameter)

This is a function block (FB) that writes the EtherCAT slave parameters. Unlike ETC_CO_SdoWrite, this FB supports only parameters with 4 bytes or less. Specify parameters to be written using the index and sub-index used for the object directory.

■ Icon

■ Parameter

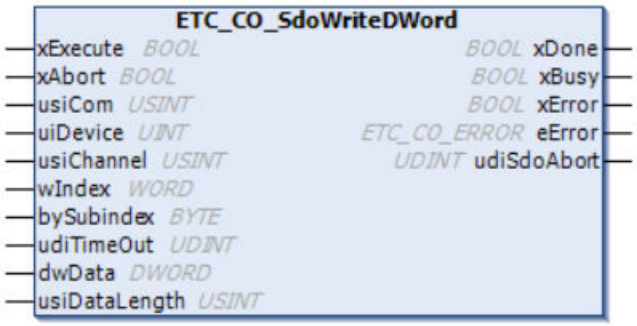
Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	TRUE: Interrupts processing and resets output.
	usiCom	USINT	1	1 (Fixed)
	uiDevice	UINT	0	Physical slave address
	usiChannel	USINT	1	Reserved
	wIndex	WORD	0	Parameter index in the object directory(Note 1)
	bySubIndex	BYTE	0	Parameter sub-index in the object directory(Note 1)
	udiTimeout	UDINT	0	Timeout (Unit: ms)
	abyData	ARRAY [1..4] OF BYTE	-	Write data storage location
	usiDataLength	USINT	0	Number of written bytes
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	ETC_CO_ERROR	ETC_CO_NO_ERROR	Error ID output
	udiSdoAbort	UDINT	0	Abort code received from the slave device

(Note 1) The parameter content differs according to the slave. Refer to the manuals of corresponding slave devices.

9.3.8 ETC_CO_SdoWriteDWord (Write Double Words of Slave Parameter)

Just like ETC_CO_SdoWrite4, this is a function block (FB) that writes the EtherCAT slave parameters. The write data is transferred in DWORD (dwData), not in an array. Since byte swapping is automatically executed, write data can be directly used.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	TRUE: Interrupts processing and resets output.
	usiCom	USINT	1	1 (Fixed)
	uiDevice	UINT	0	Physical slave address
	usiChannel	USINT	1	Reserved
	wIndex	WORD	0	Parameter index in the object directory(Note 1)
	bySubIndex	BYTE	0	Parameter sub-index in the object directory(Note 1)
	udiTimeout	UDINT	0	Timeout (Unit: ms)
	dwData	DWORD	0	Write data storage location
	usiDataLength	USINT	0	Number of written bytes
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.

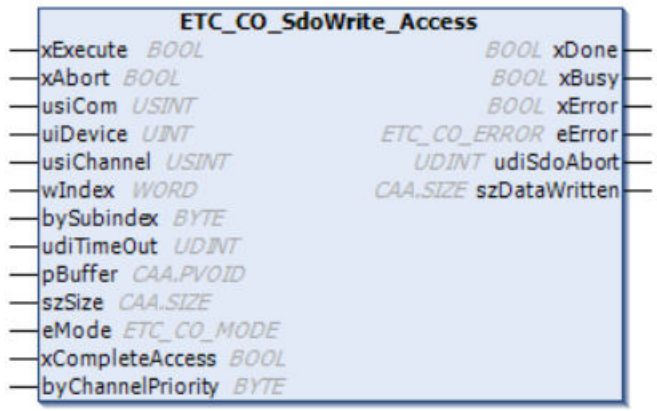
Scope	Name	Type	Default	Description
	eError	ETC_CO_ERROR	ETC_CO_NO_ERROR	Error ID output
	udiSdoAbort	UDINT	0	Abort code received from the slave device

(Note 1) The parameter content differs according to the slave. Refer to the manuals of corresponding slave devices.

9.3.9 ETC_CO_SdoWrite_Access (Write Slave Parameter Index)

Just like ETC_CO_SdoWrite, this is a function block (FB) that writes the EtherCAT slave parameters. By setting the xCompleteAccess input to TRUE and the bySubIndex input to 0, you can write complete indexes including all entries. By using the byChannelPriority (BYTE) input, you can specify the channel and priority using a CoE mailbox message. Specify the channel with the first 6 bits (bit0 to bit5) and the priority with the last 2 bits (bit6 and bit7).

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	TRUE: Interrupts processing and resets output.
	usiCom	USINT	1	1 (Fixed)
	uiDevice	UINT	0	Physical slave address
	usiChannel	USINT	1	Reserved
	wIndex	WORD	0	Parameter index in the object directory(Note 1)

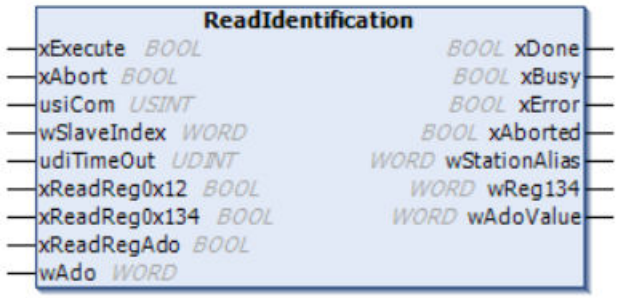
Scope	Name	Type	Default	Description
	bySubIndex	BYTE	0	Parameter sub-index in the object directory(Note 1)
	udiTimeout	UDINT	0	Timeout (Unit: ms)
	pBuffer	CAA.PVOID	0	Pointer to the buffer storing write data
	szSize	CAA.SIZE	0	Number of written bytes
	eMode	ETC_CO_MODE	ETC_CO_AUTO	Transmission mode
	xCompleteAccess	BOOL	FALSE	TRUE: Accesses all sub-indexes within the specified index.
	byChannelPriority	BYTE		Specifies the channel and priority using a CoE mailbox message.
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	ETC_CO_ERROR	ETC_CO_NO_ERROR	Error ID output
	udiSdoAbort	UDINT	0	Abort code received from the slave device
	szDataWritten	CAA.SIZE	0	Number of bytes written normally

(Note 1) The parameter content differs according to the slave. Refer to the manuals of corresponding slave devices.

9.3.10 ReadIdentification (Read Slave Identification Data)

Reads identification data from EtherCAT slaves.

■ Icon



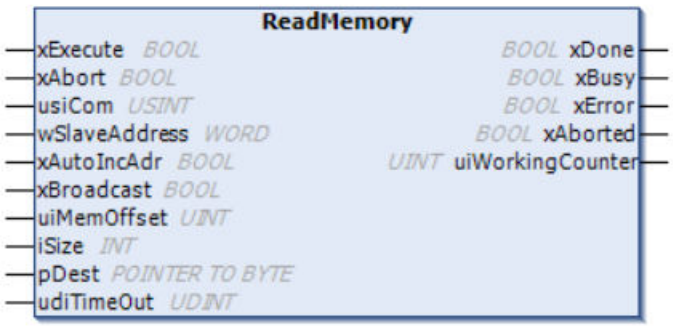
■ Parameter

Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	Do not use.
	usiCom	USINT	1	1 (Fixed)
	wSlaveIndex	WORD	0	Specifies EtherCAT slaves. EtherCAT slave numbers are allocated in ascending order from 0 to the one closest to the master.
	udiTimeout	UDINT	0	Timeout (Unit: ms) When executed with the default value 0, it will timeout immediately.
	xReadReg0x12	BOOL	FALSE	Register 16#12 (Station alias) read flag
	xReadReg0x134	BOOL	FALSE	Register 16#134 (Explicit Device ID) read flag
	xReadRegAdo	BOOL	FALSE	Register Ado read flag
	wAdo	WORD	0	Ado ID address
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	xAborted	BOOL	FALSE	TRUE: An interruption has occurred in FB processing.
	wStationAlias	WORD	0	Value of register 16#12
	wReg134	WORD	0	Value of register 16#0x134
	wAdoValue	WORD	0	Value of Ado ID

9.3.11 ReadMemory (Read Slave Memory)

This is a function block (FB) that reads the EtherCAT slave memory.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	Do not use.
	usiCom	USINT	1	Master device index (1 onwards)
	wSlaveAddress	WORD	0	Either the automatic incremental address or the device's physical address
	xAutoIncAdr	BOOL	FALSE	TRUE: Uses the automatic incremental address. When set to TRUE, specify the automatic incremental address for wSlaveAddress.
	xBroadcast	BOOL	FALSE	TRUE: Uses the broadcast read. If set to TRUE, wSlaveAddress and bAutoIncAdr are not used.
	uiMemOffset	UINT	0	Offset of the memory
	iSize	INT	0	Number of read bytes
	pDest	POINTER TO BYTE	0	Pointer to the buffer that stores read data
	udiTimeOut	UDINT	0	Timeout (Unit: ms) When executed with the default value 0, it will timeout immediately.
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.

Scope	Name	Type	Default	Description
	xAborted	BOOL	FALSE	TRUE: An interruption has occurred in FB processing.
	uiWorkingCounter	UINT	0	Working counter of received commands

9.3.12 ReadNbrSlaves (Read the Number of Connected Slaves)

Reads the number of slaves currently connected.

■ Icon

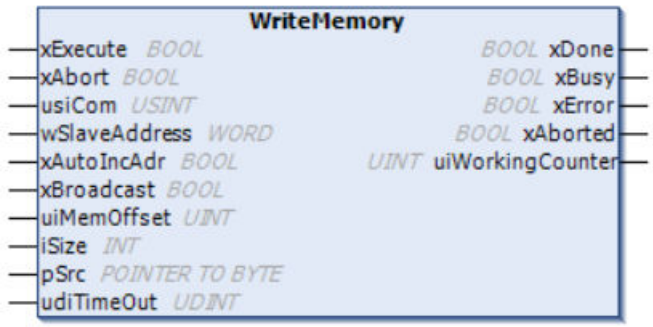
■ Parameter

Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	Do not use.
	usiCom	USINT	1	1 (Fixed)
	udiTimeOut	UDINT	0	Timeout (Unit: ms) When executed with the default value 0, it will timeout immediately.
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	xAborted	BOOL	FALSE	TRUE: An interruption has occurred in FB processing.
	wNumberSlaves	WORD	0	Number of slaves connected

9.3.13 WriteMemory (Write Slave Memory)

This is a function block (FB) that writes the EtherCAT slave memory. Accesses"ESC address space".

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	xExecute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Resets output after execution is completed.
	xAbort	BOOL	FALSE	Do not use.
	usiCom	USINT	1	1 (Fixed)
	wSlaveAddress	WORD	0	When set to the physical slave address or to bAutoIncrementAddress, specify the automatic incremental address.
	xAutoIncAdr	BOOL	FALSE	TRUE: Uses the automatic incremental address. When set to TRUE, specify the automatic incremental address for wSlaveAddress.
	xBroadcast	BOOL	FALSE	TRUE: Uses the broadcast read. If set to TRUE, wSlaveAddress and bAutoIncAdr are not used.
	uiMemOffset	UINT	0	Offset of the memory
	iSize	INT	0	Number of bytes to be written
	pSrc	POINTER TO BYTE	0	Address of the data buffer to be written
	udiTimeOut	UDINT	0	Timeout (Unit: ms) When executed with the default value 0, it will timeout immediately.
Output	xDone	BOOL	FALSE	TRUE: FB processing is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.

Scope	Name	Type	Default	Description
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	xAborted	BOOL	FALSE	TRUE: An interruption has occurred in FB processing.
	uiWorkingCounter	UINT	0	Working counter of received commands

9.3.14 PETC_ClearAmpPositionalDeviation (Clear Amplifier Deviation Counter)

This is a function block (FB) that clears the deviation counter of the AMP. It deletes the position deviation data in the deviation counter of the AMP.

■ Icon

■ Parameter

Scope	Name	Type	Default	Description
Input / output	Axis	AXIS_REF_RTEX_Panasonic	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Velocity(Note 1)	LREAL		Information required to execute MC_MoveAbsolute
	Acceleration(Note 1)	LREAL		Information required to execute MC_MoveAbsolute
	Deceleration(Note 1)	LREAL		Information required to execute MC_MoveAbsolute
	Jerk(Note 1)	LREAL		Information required to execute MC_MoveAbsolute
Output	Done	BOOL	FALSE	TRUE: Clearing is completed.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	Error ID output

(Note 1) This function block internally substitutes the command position with an actual position to call MC_MoveAbsolute and, therefore, requires parameters including Velocity, Acceleration, Deceleration, and Jerk.

The PETC_ClearAmpPositionalDeviation function block outputs the following errors.

Error	Description
SMC_WRONG_CONTROLLER_MODE	Executed in a mode other than the position control mode. Change to SMC_position using SMC_SetControllerMode.
SMC_AXIS_NOT_READY_FOR_MOTION	The axis is in a state where PMC_ClearAmpMultiTurnData cannot be executed. It can be executed only at the standstill state.
SMC_REGULATOR_OR_START_NOT_SET	The axis is in a servo OFF state.
SMC_PP_WRONG_AXIS_TYPE	The axis is a virtual axis.

9.4 EtherCAT Master/Slave

9.4.1 EtherCAT Master/Slave Communication Control and Monitoring

With the GM1 controller, you can control EtherCAT master communication and monitor communication between the EtherCAT master and the slave.

With GM Programmer, the following four function blocks and methods are available in checking the EtherCAT state.

- IoDrvEtherCAT(FB): This FB provides functions for the restart or the bus stop of the EtherCAT master, allowing you to check a transition to normal communication through the completion of the master configuration and the completion of synchronization with the slave.
- IoDrvEtherCAT.GetStatistics(METH): This enables you to get EtherCAT frame statistics information. Through the number of lost EtherCAT frames (udiLostFrameCount), you can monitor if communication is properly performed.
- IoDrvEtherCAT.ClearStatistics(METH): This enables you to clear EtherCAT frame statistics information.
- ETCSlave(FB): This enables you to check the communication state of EtherCAT slaves.

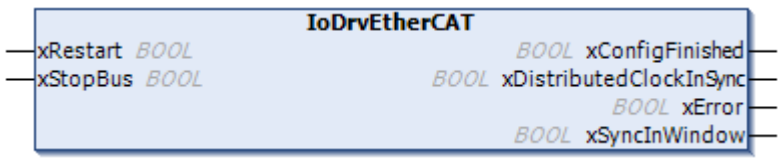
Info.

- For an example process for monitoring the communication state of the EtherCAT master, refer to "9.4.6 Sample Example: Process for Monitoring EtherCAT Master Communication".
- For an example process for monitoring the communication state of EtherCAT slave devices, refer to "9.4.7 Sample Example: Process for Monitoring EtherCAT Slave Communication".
- For an example process for stopping/restarting EtherCAT master device communication, refer to "9.4.8 Sample Example: Stop/Restart EtherCAT Master Communication".

9.4.2 IoDrvEtherCAT (Control EtherCAT Master Communication)

This is a function block that controls EtherCAT master communication. Since the EtherCAT master FB (EtherCAT_Master_SoftMotion) is automatically created, no declaration is required.

■ **Icon**



■ **Parameter**

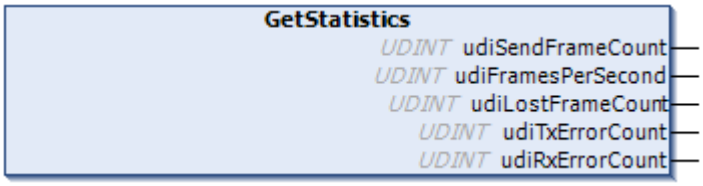
Scope	Name	Type	Default value	Description
Input	xRestart	BOOL	FALSE	Rising edge: Restarts the EtherCAT master communication.
	xStopBus	BOOL	FALSE	TRUE: Stops the EtherCAT master communication.

Scope	Name	Type	Default value	Description
Output	xConfigFinished	BOOL	FALSE	TRUE: Configuration has been completed successfully
	xDistributedClockInSync	BOOL	FALSE	TRUE: Synchronization with the EtherCAT slave with the DC option activated has been completed successfully.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	xSyncInWindow	BOOL	FALSE	Do not use.

9.4.3 IoDrvEtherCAT.GetStatistics (Get EtherCAT Communication Statistics Information)

This method is used to get statistics information such as the number of sent frames, the number of lost frames or error counts during EtherCAT communication. Write EtherCAT_Master_SoftMotion.GetStatistics. Through the acquired lost EtherCAT frame count, you can monitor whether communication is properly performed.

■ Icon



■ Parameter

Scope	Name	Type	Description
Output	udiSendFrameCount	UDINT	Number of total EtherCAT frames sent
	udiFramesPerSecond	UDINT	Number of EtherCAT send frames per second
	udiLostFrameCount	UDINT	Number of lost EtherCAT frames
	udiTxErrorCount	UDINT	Number of send errors
	udiRxErrorCount	UDINT	Number of receive errors

9.4.4 IoDrvEtherCAT.ClearStatistics (Clear EtherCAT Communication Statistics Information)

This method is used to clear statistics information about EtherCAT communication. Write EtherCAT_Master_SoftMotion.ClearStatistics to set every statistics information counter to zero. This has no argument.

■ Icon

9.4.5 ETCSlave (Control EtherCAT Slave Communication)

This is a function block designed to start establishing EtherCAT slave communication and monitor the state of communication. An FB instance is automatically generated for each EtherCAT slave. With DeviceName.wState, the communication state of the slave can be acquired.

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	xSetOperational	BOOL	FALSE	Rising edge: An attempt is made to switch to the ETC_SLAVE_OPERATIONAL mode.
Output	wState	ETC_SLAVE_STATE	3(ETC_SLAVE_BOOT)	Outputs the communication state of the slave device.

■ ETC_SLAVE_STATE (Enumeration type)

The states of the EtherCAT slave dependent on the EtherCAT State Machine (ESM) are shown.

Name	Value	Description
ETC_SLAVE_BOOT	3	Bootstrap state
ETC_SLAVE_INIT	1	Init state
ETC_SLAVE_PREOPERATIONAL	2	Pre-Operational state
ETC_SLAVE_SAVEOPERATIONAL	4	Safe-Operational state
ETC_SLAVE_OPERATIONAL	8	Operational state

9.4.6 Sample Example: Process for Monitoring EtherCAT Master Communication

This is a program coded to monitor the communication state of the master via EtherCAT_Master_SoftMotion.

- Description of process

  In response to the start of operation, the process for monitoring the communication gets started to acquire statistics information. The program for communication monitoring determines a communication error when the number of lost EtherCAT frames gets greater than or equal to a specified number.

  After the occurrence of the communication error, if xClear is set to TRUE, the statistics information is cleared and communication of the EtherCAT master can be monitored again.

- Declaration section

```
VAR
// Execution of ClearStatistics
xClear                  : BOOL  := FALSE;
// communication check flag
xCommunicateOK          : BOOL  := FALSE;
xCommunicateNG          : BOOL  := FALSE;

// Variables
udiECAT_SendFrameCount  : UDINT := 0; //sending frame count
udiECAT_FramesPerSecond : UDINT := 0; //sending frame count per second
udiECAT_LostFrameCount  : UDINT := 0; //lost frame count
udiECAT_TxErrorCount    : UDINT := 0; //Tx Error frame count
udiECAT_RxErrorCount    : UDINT := 0; //Rx Error frame count
END_VAR

VAR CONSTANT
udiECAT_ERRORCOUNT      : UDINT := 3; //threshold of abnormal Communicate
END_VAR
```

- Implementation section

```
// Get EtherCAT communication log statistics
EtherCAT_Master_SoftMotion.GetStatistics(
        udiSendFrameCount   =>  udiECAT_SendFrameCount,
        udiFramesPerSecond  =>  udiECAT_FramesPerSecond,
        udiLostFrameCount   =>  udiECAT_LostFrameCount,
        udiTxErrorCount     =>  udiECAT_TxErrorCount,
        udiRxErrorCount     =>  udiECAT_RxErrorCount );

// EtherCAT communication check
IF ( udiECAT_LostFrameCount > udiECAT_ERRORCOUNT ) THEN
    xCommunicateNG     := TRUE;
    xCommunicateOK     := FALSE;
ELSE
    xCommunicateOK     := TRUE;
    xCommunicateNG     := FALSE;
END_IF

// Clear EtherCAT communication log statistics
```

```
IF ( xClear=TRUE ) THEN
    EtherCAT_Master_SoftMotion.ClearStatistics();
    xClear              := False;
END_IF
```

9.4.7 Sample Example: Process for Monitoring EtherCAT Slave Communication

This is a program coded to monitor the communication state of the slave devices connected to the EtherCAT master. This program allows you to check how many slave devices are properly communicating.

- Description of process

  When the case number (iStep) is set to 1, the communication monitoring process starts. A description of the process is given with the following two execution results taken as examples.

Execution result 1

Expression	Type	Value
xECAT_SlaveOK	BOOL	TRUE
iECAT_SlaveCount	INT	3
axSlaveState	ARRAY [1..32] OF BOOL	
axSlaveState[1]	BOOL	TRUE
axSlaveState[2]	BOOL	TRUE
axSlaveState[3]	BOOL	TRUE
axSlaveState[4]	BOOL	FALSE
axSlaveState[5]	BOOL	FALSE

Execution result 2

Expression	Type	Value
xECAT_SlaveOK	BOOL	FALSE
iECAT_SlaveCount	INT	3
axSlaveState	ARRAY [1..32] OF BOOL	
axSlaveState[1]	BOOL	TRUE
axSlaveState[2]	BOOL	FALSE
axSlaveState[3]	BOOL	FALSE
axSlaveState[4]	BOOL	FALSE
axSlaveState[5]	BOOL	FALSE

Errors can be detected as shown below from the execution result 1 and execution result 2.

- The case of execution result 1

  If three slave devices are connected to the EtherCAT master, they are properly communicating.

- The case of execution result 2

  If three slave devices are connected to the EtherCAT master, the first device is properly communicating, whereas a communication error is detected in the second and succeeding devices.

  A failure in LAN cable connection between the first and second devices or the occurrence of an error in the second slave can be detected.

- Declaration section

```
VAR
// Change to 1 : To Start
iStep               : INT    := 0;
// Finish Flag
xFinish             : BOOL   := FALSE;

// Variables
pSlave              : POINTER TO ETCSlave;   //Pointer of slave information
xECAT_SlaveOK       : BOOL   := FALSE;       //All slave state OK
iECAT_SlaveCount    : INT    := 0;           //Number of slaves
axSlaveState        : ARRAY[1..32] OF BOOL;  //Slave state

END_VAR
```

- Implementation section

```
// Communication check of EtherCAT_Slave
CASE iStep OF
  1: // Initial setting
    xECAT_SlaveOK    := TRUE;
    iECAT_SlaveCount := 0;
    pSlave           := EtherCAT_Master_SoftMotion.FirstSlave;
    iStep            := 2;
  2: // Check of EtherCAT_Slave
    WHILE ( pSlave <> 0 ) DO
      iECAT_SlaveCount := iECAT_SlaveCount + 1;
      pSlave^();
      IF ( EtherCAT_Master_SoftMotion.xDistributedClockInSync = TRUE ) AND
         ( pSlave^.wState = ETC_SLAVE_STATE.ETC_SLAVE_OPERATIONAL ) THEN
          axSlaveState[iECAT_SlaveCount] := TRUE;
      ELSE
          axSlaveState[iECAT_SlaveCount] := FALSE;
          xECAT_SlaveOK                  := FALSE;
      END_IF
      pSlave           := pSlave^.NextInstance;
    END_WHILE
    iStep            := 3;
  3: // Check completed
    xFinish          := TRUE;
    iStep            := 0;
END_CASE
```

9.4.8 Sample Example: Stop/Restart EtherCAT Master Communication

This program is designed to stop or restart the communication of the EtherCAT master.

To use this program, it is necessary to select “Enable diagnosis for device” checkbox on the “PLC Settings” tab in the Device object.

The communication automatically restarts after the communication is stopped unless the “Automatically Start Slaves” checkbox is deselected in the general settings of EtherCAT_Master_SoftMotion.

- Description of process

  When the case number (iStep) is set to 1, the communication stops. When the stop processing is completed, the xStop_Communication parameter goes TRUE.

  When the case number (iStep) is set to 4, the communication restarts. When the restart processing is completed, the xRestart_Communication parameter goes TRUE.

- Declaration section

```
VAR
// Change to 1 or 4 : To Start
iStep                      : INT    := 0;
// Finish Flag
xStop_Communication        : BOOL   := FALSE;
xRestart_Communication     : BOOL   := FALSE;
END_VAR
```

- Implementation section

```
// Stop and restart communication of EtherCAT_Master_SoftMotion
CASE iStep OF
  1: // Stop communication Bus
    EtherCAT_Master_SoftMotion(xStopBus := TRUE);
    iStep                       := 2;
  2: // Stopping communication
    EtherCAT_Master_SoftMotion(xStopBus := FALSE);
    IF ( EtherCAT_Master_SoftMotion.xDistributedClockInSync=FALSE ) THEN
      iStep                     := 3;
    END_IF
  3: // Chaging Device state to STOP from RUN.
    IF ( EtherCAT_Master_SoftMotion.CheckCurrentSupportedCommunicationStat
e(eRequestedState:=DEVICE_TRANSITION_STATE.STOP) )
    AND ( EtherCAT_Master_SoftMotion.CheckSupportedCommunicationState(eRequ
estedState:=DEVICE_TRANSITION_STATE.STOP) ) THEN
      EtherCAT_Master_SoftMotion.SetCommunicationState(eRequestedState:=DEV
ICE_TRANSITION_STATE.STOP);
    ELSE
     // Complete stop process
      IF ( EtherCAT_Master_SoftMotion.xDistributedClockInSync=FALSE ) THEN
          xStop_Communication  := TRUE;
          iStep                := 0;
      END_IF
    END_IF
  4: //Restart communication of EtherCAT_Master_SoftMotion
    EtherCAT_Master_SoftMotion(xRestart := TRUE);
     //If complete communication
    IF ( EtherCAT_Master_SoftMotion.xDistributedClockInSync=TRUE ) THEN
      EtherCAT_Master_SoftMotion(xRestart := FALSE);
      xRestart_Communication   := TRUE;
      iStep                    := 0;
    END_IF
END_CASE
```

(MEMO)

10 Motion Control Function Blocks (Auxiliary Function)

10.1 Motion Auxiliary Function (Monitoring)

10.1.1 MC_ReadActualPosition (Read Current Position)

This is a function block (FB) that reads the actual position data of the axis.

■ Icon

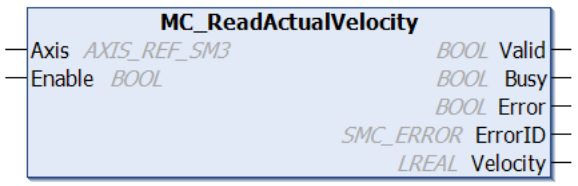
■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Enable	BOOL	FALSE	Reads the actual position while Enable is set to TRUE.
Output	Valid	BOOL	FALSE	TRUE: Valid output
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	An error ID is output.
	Position	LREAL	0	Actual position (u) that is read out

10.1.2 MC_ReadActualVelocity (Read Current Velocity)

This is a function block (FB) that reads the actual velocity of the axis.

■ Icon



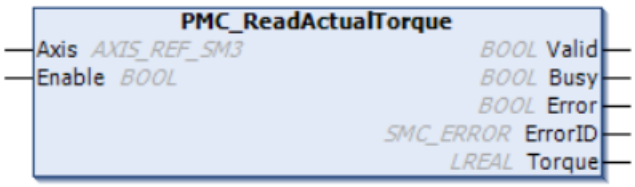
■ **Parameter**

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Enable	BOOL	FALSE	Reads the actual velocity while Enable is set to TRUE.
Output	Valid	BOOL	FALSE	TRUE: Valid output
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	An error ID is output.
	Velocity	LREAL	0	Current actual velocity (u/s) that is read out

10.1.3 PMC_ReadActualTorque (Read Current Torque)

This is a function block (FB) that reads the actual torque value of the axis.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Enable	BOOL	FALSE	Reads the actual torque value while Enable is set to TRUE.
Output	Valid	BOOL	FALSE	TRUE: Valid output
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	An error ID is output.
	Torque	LREAL	0	Current actual torque (%) that is read out

10.1.4 MC_ReadActualTorque (Read Current Torque)

This is a function block (FB) that reads the current torque value of the axis.

■ Icon

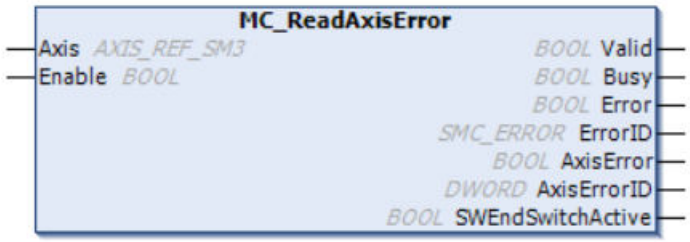
■ Parameter

Scope	Name	Type	Default	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Enable	BOOL	FALSE	Reads the actual torque value while Enable is set to TRUE.
Output	Valid	BOOL	FALSE	TRUE: Valid output
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	Error ID output
	Torque	LREAL	0	Current actual torque that is read out (N·m, N)

10.1.5 MC_ReadAxisError (Read Axis Error)

This is a function block that gets general axis errors not related to function blocks.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Enable	BOOL	FALSE	Reads the state while Enable is set to TRUE.
Output	Valid	BOOL	FALSE	TRUE: Valid output
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	Error ID output
	AxisError	BOOL	FALSE	TRUE: An axis error has occurred.
	AxisErrorID	DWORD	0	Axis error ID Servo amplifier alarms can be acquired from AxisErrorID. If an Err27.4 error occurs when connected to the MINAS, 27 (16#1B) is set and AxisErrorID becomes 16#0000FF1B.
	SWEndSwitchActive	BOOL	FALSE	TRUE: The software limit has been exceeded.

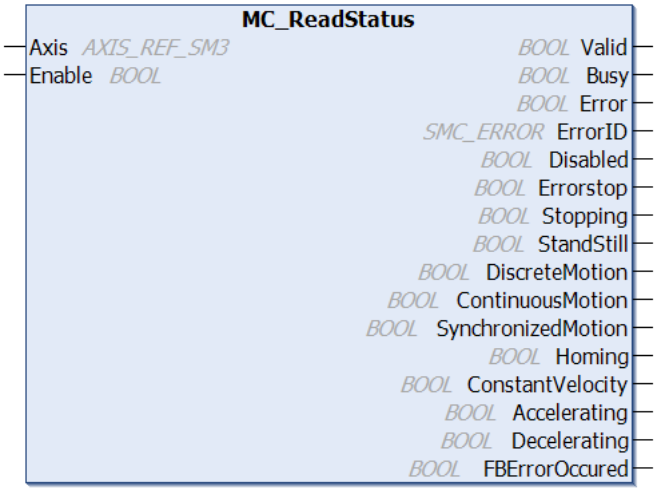
Info.

- Do not execute MC_ReadAxisError while SMC3_ReinitDrive is running.

10.1.6 MC_ReadStatus (Read Status)

This is a function block (FB) that reads the status information of the axis. It reads detailed information about the axis state.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Enable	BOOL	FALSE	Reads the status information while the input is TRUE.
Output	Valid	BOOL	FALSE	TRUE: Valid output
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	An error ID is output.
	Disabled	BOOL	FALSE	TRUE: The axis is in the Disabled state.
	ErrorStop	BOOL	FALSE	TRUE: The axis is in the ErrorStop state.
	Stopping	BOOL	FALSE	TRUE: The axis is in the Stopping state.
	StandStill	BOOL	FALSE	TRUE: The axis is in the StandStill state.
	DiscreteMotion	BOOL	FALSE	TRUE: The axis is in the DiscreteMotion state.
	ContinuousMotion	BOOL	FALSE	TRUE: The axis is in the ContinuousMotion state.

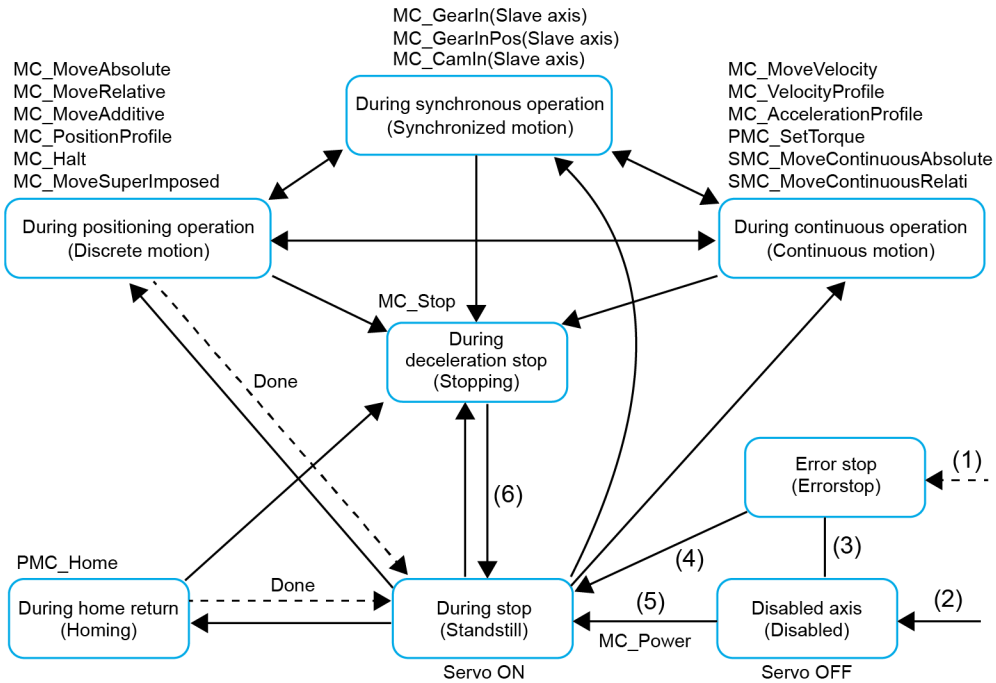
Scope	Name	Type	Initial	Description
	SynchronizedMotion	BOOL	FALSE	TRUE: The axis is in the SynchronizedMotion state.
	Homing	BOOL	FALSE	TRUE: The axis is in the Homing state.
	ConstantVelocity	BOOL	FALSE	TRUE: The axis is moving at a constant velocity.
	Accelerating	BOOL	FALSE	TRUE: The axis is moving in acceleration.
	Decelerating	BOOL	FALSE	TRUE: The axis is moving in deceleration.
	FBErrorOccured	BOOL	FALSE	TRUE: An FB error has occurred.

■ Axis state

The following section describes state transition diagram of the axis when the motion function blocks are executed.

State transition diagram

- The blue frame indicates the state.
- When the function block indicated above the state is executed, the state transitions to the direction indicated by the solid-line arrow.
- When the execution is completed or when an error occurs, the state transitions to the state indicated at the tip of the broken-line arrow.
- The terms in parentheses are defined in PLCopen.

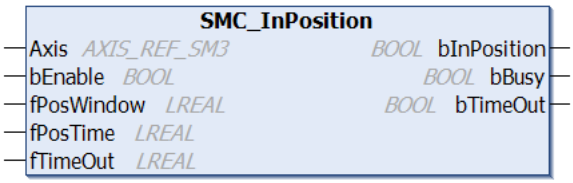


Number	Transition conditions
(1)	Regardless of the state, when an error occurs in the axis
(2)	Regardless of the state, when Enable of MC_Power is TRUE, bRegulator is FALSE, and there is no error in the axis
(3)	When Status of MC_Reset and Status of MC_Power are FALSE
(4)	When Enable of MC_Reset and Enable of MC_Power are TRUE, bRegulator is TRUE, and Status is TRUE
(5)	When Enable of MC_Power is TRUE, bRegulator is TRUE, and Status is TRUE
(6)	When Done of MC_Stop is TRUE and Execute of MC_Stop is FALSE

10.1.7 SMC_InPosition (In-position Judgment)

This is a function block (FB) that compares the actual position of the AMP with the command value and judges whether the position is within the specified range. The maximum difference between the actual position of the AMP and the command value as well as the dwell time are specified to judge (in-position judgment) whether the specified values are satisfied.

■ Icon



■ Parameter

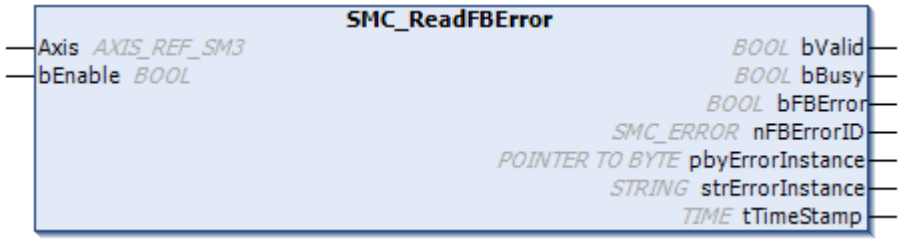
Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bEnable	BOOL	FALSE	TRUE: Executes the FB.
	fPosWindow	LREAL	0	The maximum difference between the actual position and the command value to judge whether the target position has been reached.
	fPosTime	LREAL	0	The dwell time (s) to judge whether the axis has reached the position
	fTimeOut	LREAL	0	The time (s) from when the FB is enabled to when judgment is made that timeout has occurred When the value is “0”, the timeout judgment is not made yet.
Output	bInPosition	BOOL	FALSE	TRUE: The target position is reached. While the difference between the actual position and the command

Scope	Name	Type	Initial	Description
				value is within the time specified in fPosTime, it is within the fPosWindow.
	bBusy	BOOL	FALSE	TRUE: The FB is in operation.
	bTimeOut	BOOL	FALSE	TRUE: Timeout has occurred.

10.1.8 SMC_ReadFBError (Read Oldest Error)

This is a function block (FB) that reads the oldest function block error information.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bEnable	BOOL	FALSE	TRUE: The FB can be executed.
Output	bValid	BOOL	FALSE	TRUE: Error information is acquired.
	bBusy	BOOL	FALSE	TRUE: The FB is in operation.
	bFBError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	nFBErrorID	SMC_ERROR	0	An error ID is output.
	pbyErrorInstance	POINTER TO BYTE	0	FB instance of the error acquisition source
	strErrorInstance	STRING	''	FB instance name of the error acquisition source
	tTimeStamp	TIME	TIME#0ms	Time stamp of the error information

Note

- The error information is cleared when SMC_ClearFBError is executed. When and error occurs again, SMC_ReadFBError reads the error.

REFERENCE

10.1.9 SMC_ClearFBError (Clear Oldest Error)

10.1.9 SMC_ClearFBError (Clear Oldest Error)

This function clears the oldest FB error information.

■ Icon

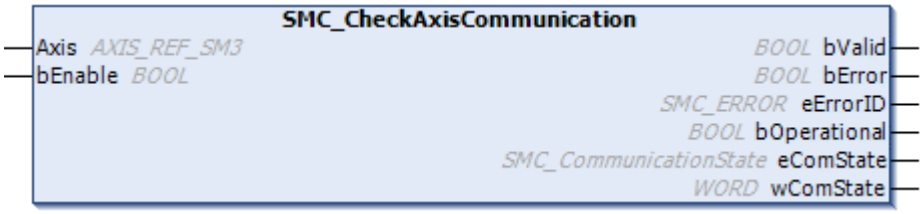
■ Parameter

Type	Parameter name	Type	Default	Description
Input	pDrive	POINTER TO AXIS_REF_SM3	-	Specifies the axis.
Return	SMC_ClearFBError	BOOL		This function always returns FALSE even for normal completion.

10.1.10 SMC_CheckAxisCommunication (Check Axis Communication Status)

This is a function block (FB) that checks the communication state of the axis.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bEnable	BOOL	FALSE	TRUE: The FB can be executed.
Output	bValid	BOOL	FALSE	TRUE: The output value is valid.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eErrorID	SMC_ERROR	0	An error ID is output.
	bOperational	BOOL	FALSE	TRUE: Communication state is operational. (100)
	eComState	SMC_CommunicationState	SMC_COMSTATE_NOT_STARTED	Communication state
	wComState	WORD	0	Internal value of the communication state

■ SMC_CommunicationState (Enumeration type)

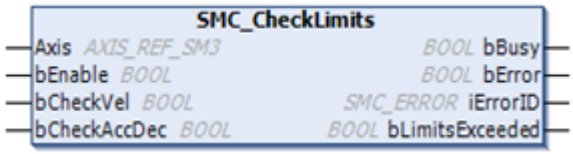
Name	Value	Description
SMC_COMSTATE_NOT_STARTED	0	Stop
SMC_COMSTATE_VARIABLE_INITIALIZATION	1	Initialization of variables
SMC_COMSTATE_BASE_COM_INITIALIZATION	2	Initialization of base communication settings
SMC_COMSTATE_DRIVE_INITIALIZATION	3	Initialization of drive settings
SMC_COMSTATE_DRIVE_WAITING_FOR_SYNC	4	Waiting for drive synchronization
SMC_COMSTATE_INITIALIZATION_DONE	5	Initialization completed
SMC_COMSTATE_OPERATIONAL	6	Operational
SMC_COMSTATE_REINITIALIZATION	7	Re-initialization

Name	Value	Description
SMC_COMSTATE_ERROR	8	Error
SMC_COMSTATE_UNKNOWN	9	Unknown

10.1.11 SMC_CheckLimits (Check Exceeding Limits)

This is a function block (FB) that checks whether the velocity, acceleration, or deceleration is in excess of the dynamic limit set value of the device.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bEnable	BOOL	FALSE	TRUE: The FB can be executed.
	bCheckVel	BOOL	TRUE	TRUE: Checks the velocity setting.
	bCheckAccDec	BOOL	FALSE	TRUE: Checks the acceleration and deceleration settings.
Output	bBusy	BOOL	FALSE	TRUE: The FB is in operation.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	iErrorID	SMC_ERROR	0	An error ID is output.
	bLimitsExceeded	BOOL	FALSE	TRUE: Limits are exceeded.

Info.

- Reference manual
  GM1 Controller RTEX User's Manual (Operation Edition)
  GM1 Controller EtherCAT User's Manual (Operation Edition)

10.1.12 SMC_GetMaxSetAccDec (Measure Maximum Acceleration / Deceleration)

This is a function block (FB) that measures the maximum value of the axis acceleration/ deceleration command.

■ Icon

■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bEnable	BOOL	FALSE	TRUE: The FB can be executed.
	dwTimeStamp	DWORD	0	Time stamp
Output	bValid	BOOL	FALSE	TRUE: The output value is valid.
	bBusy	BOOL	FALSE	TRUE: The FB is in operation.
	fMaxAcceleration	LREAL	0	Maximum acceleration (u/s^2).
	dwTimeAtMax	DWORD	0	dwTimeStamp value at the maximum acceleration

Note

- It is possible to check when the maximum acceleration or deceleration has occurred by entering a call counter value in the input variable "dwTimeStamp".

10.1.13 SMC_GetMaxSetVelocity (Measure Maximum Velocity)

This is a function block (FB) that measures the maximum value of the axis velocity command.

■ Icon

■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bEnable	BOOL	FALSE	TRUE: The FB can be executed.
	dwTimeStamp	DWORD	0	Time stamp
Output	bValid	BOOL	FALSE	TRUE: The output value is valid.
	bBusy	BOOL	FALSE	TRUE: The FB is in operation.
	fMaxVelocity	LREAL	0	Maximum velocity (u/s).
	dwTimeAtMax	DWORD	0	dwTimeStamp value at the maximum acceleration

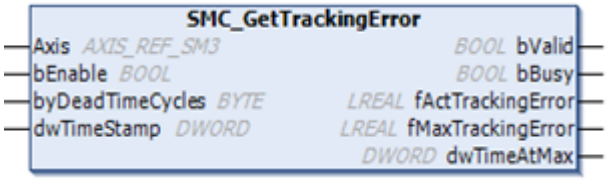
Note

- It is possible to check when the maximum velocity has occurred by entering a call counter value in the input variable "dwTimeStamp".

10.1.14 SMC_GetTrackingError (Measure Tracking Error)

This is a function block (FB) that measures the tracking error of the actual position for the axis command position.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bEnable	BOOL	FALSE	TRUE: The FB can be executed.
	byDeadTimeCycles	BYTE	2	Number of dead time cycles Compares the command position and actual position between the specified cycles.
	dwTimeStamp	DWORD	0	Time stamp
Output	bValid	BOOL	FALSE	TRUE: The output value is valid.
	bBusy	BOOL	FALSE	TRUE: The FB is in operation.
	fActTrackingError	LREAL	0	Actual tracking error
	fMaxTrackingError	LREAL	0	Maximum tracking error while the function block is being executed
	dwTimeAtMax	DWORD	0	dwTimeStamp value when the maximum tracking error is detected

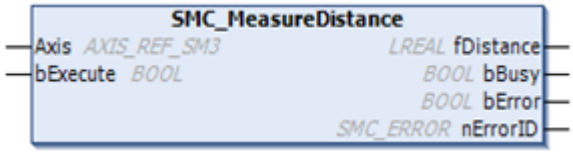
Note

- It is possible to check when the maximum tracking error has occurred by entering a call counter value in the input variable "dwTimeStamp".

10.1.15 SMC_MeasureDistance (Measure Turnaround Travel Distance)

This is a function block (FB) that measures the travel distance. For the modulo axis, the cover distance can be measured considering the laps.

■ Icon



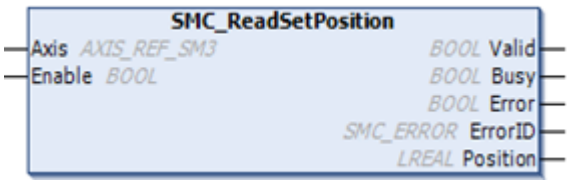
■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bExecute	BOOL	FALSE	TRUE: Starts measurement at the rising edge. FALSE: Ends measurement.
Output	fDistance	LREAL	0	Distance traveled from the start of measurement
	bBusy	BOOL	FALSE	TRUE: The FB is in operation.
	bError	BOOL	0	TRUE: An error has occurred within the FB.
	nErrorID	SMC_ERROR	0	An error ID is output.

10.1.16 SMC_ReadSetPosition (Read Axis Set Position)

This is a function block (FB) that acquires the command position of the axis.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Enable	BOOL	FALSE	TRUE: Executes the FB.
Output	Valid	BOOL	FALSE	TRUE: The output value is valid.
	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	An error ID is output.
	Position	LREAL	0	Axis position

10.2 Motion Auxiliary Function (Change / Reset)

10.2.1 MC_Reset (Axis Error Reset)

This is a function block (FB) that resets the state transition error of the axis. It reset the axis error and transitions the state from the ErrorStop state to the StandStill state.

■ **Icon**

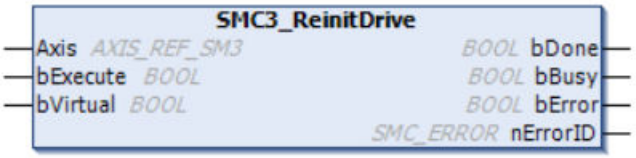
■ **Parameter**

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
Output	Done	BOOL	FALSE	TRUE: Reset done
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	An error ID is output.

10.2.2 SMC3_ReinitDrive (Reinitialize Axis)

This is a function block that restarts the drive / axis. It means that the startup phase is executed again and the application cannot control the drive until bDone of the FB is set to TRUE.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	bVirtual	BOOL	FALSE	If bVirtual is set to TRUE, the axis is set to the virtual mode.
Output	bDone	BOOL	FALSE	TRUE: Reset is completed.
	bBusy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	nErrorID	SMC_ERROR	0	Error ID output

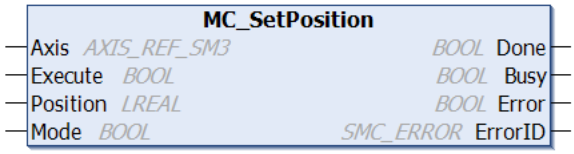
Info.

- Do not execute MC_ReadAxisError while SMC3_ReinitDrive is running.
- If MC_ReadAxisError is running, set Enable of MC_ReadAxisError to FALSE to stop the processing and then execute SMC3_ReinitDrive.

10.2.3 MC_SetPosition (Change Current Position)

This is a function block (FB) that changes the current command position of the axis.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Position	LREAL	0	Specifies the position when the mode is set to ABSOLUTE. Specifies the distance when the mode is set to RELATIVE.
	Mode	BOOL	FALSE	TRUE: RELATIVE (Relative position) FALSE: ABSOLUTE (Absolute position)
Output	Done	BOOL	FALSE	TRUE: Position change is completed.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred.
	ErrorID	SMC_ERROR	0	An error ID is output.

10.2.4 SMC_ChangeDynamicLimits(Dynamic limit change)

This is a function block that changes the dynamic limits (velocity, acceleration, deceleration, jerk) of the real and virtual axes.When the axis state is power_off or standstill,this FB can be used.

■ Icon

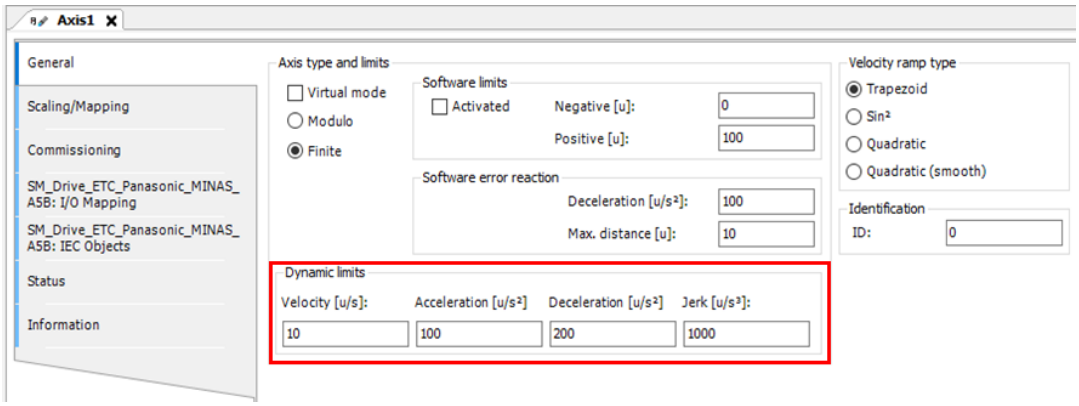
■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	fMaxVelocity	LREAL	0	Dynamic limit velocity to set [u / s] (Note 1)
	fMaxAcceleration	LREAL	0	Dynamic limit acceleration to set [u / s^2] (Note 1)
	fMaxDeceleration	LREAL	0	Dynamic limit deceleration to set [u / s^2] (Note 1)
	fMaxJerk	LREAL	0	Dynamic limit Jerk to set[u / s^3] (Note 1)
Output	bDone	BOOL	FALSE	TRUE：FB execution completed
	bError	BOOL	FALSE	TRUE: An error has occurred.
	bBusy	BOOL	FALSE	TRUE：The FB is in operation.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

(Note 1) An error occurs when 0 is set for each argument. Set a positive value.

■ Setting location

The dynamic limit in the figure below can be changed while GM1 is running.



Info.

- When the GM1 is powered off, cold reset, or warm reset, it returns to the previous setting value.
- For dynamic restrictions, refer to the GM1 Series Reference Manual (Operation).

10.2.5 SMC_ChangeGearingRatio(Gear ratio and axis type change)

This is a function block that changes the gear ratio and Axis type (Finite / Modulo) of the real and virtual axes.On change, the axis must be restarted by SMC3_ReinitDrived.When the axis state is power_off ,this FB can be used.

■ Icon

■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	dwRatioTechUnitsDenom	DWORD	0	use the multiplication of three parameters in the figures (1) and (3) below. (Note 1)
	iRatioTechUnitsNum	DINT	0	use the multiplication of three parameters in the figures (2) and (4) below. (Note 2)
	fPositionPeriod	LREAL	0	Effective for the modulo value and the modulo axis. (Note 3)
	iMovementType	SMC_MOVEMENTTYPE	0	Specify axis type
Output	bDone	BOOL	FALSE	TRUE：FB execution completed.
	bBusy	BOOL	FALSE	TRUE：The FB is in operation.
	bError	BOOL	FALSE	TRUE：An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

(Note 1) Set other than 0.
When iMovementType is 0, do not set a value higher than 16 # 7FFFFF.

(Note 2) Set other than 0.When a negative value is set, the axis is reversed.
When iMovementType is 0, set a multiple of 360.

(Note 3) Set a positive value.
When iMovementType is 0, set a multiple of 360.

■ SMC_MOVEMENTTYPE（Enumeration type）

Name	Value	Description
rotary	0	the modulo axis

Name	Value	Description
linear	1	the finite axis<

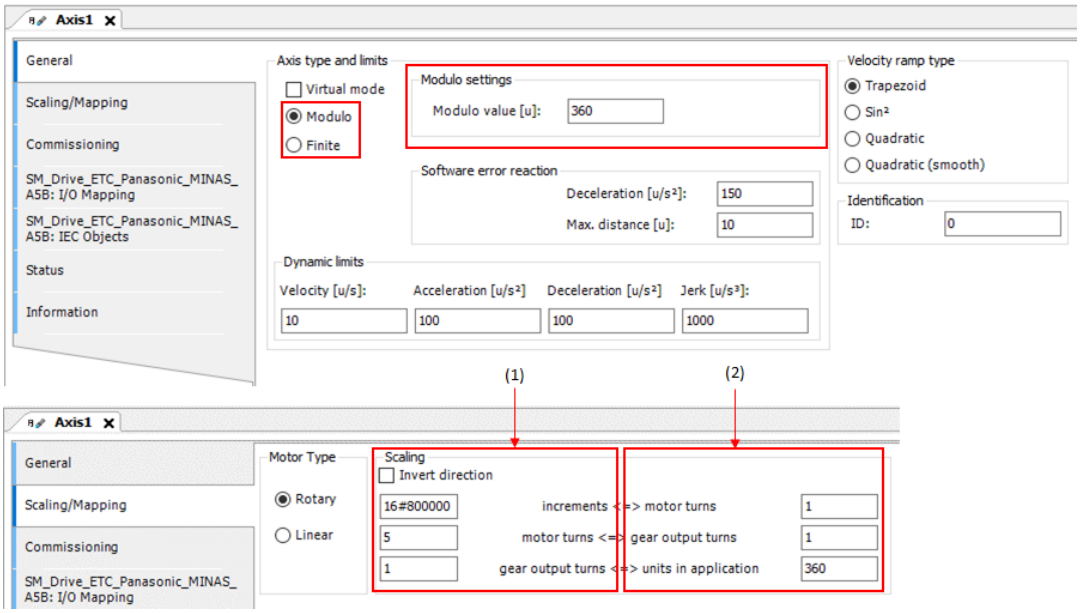
Info.

- When the execution is completed, the axis must be restarted by SMC3_ReinitDrived.

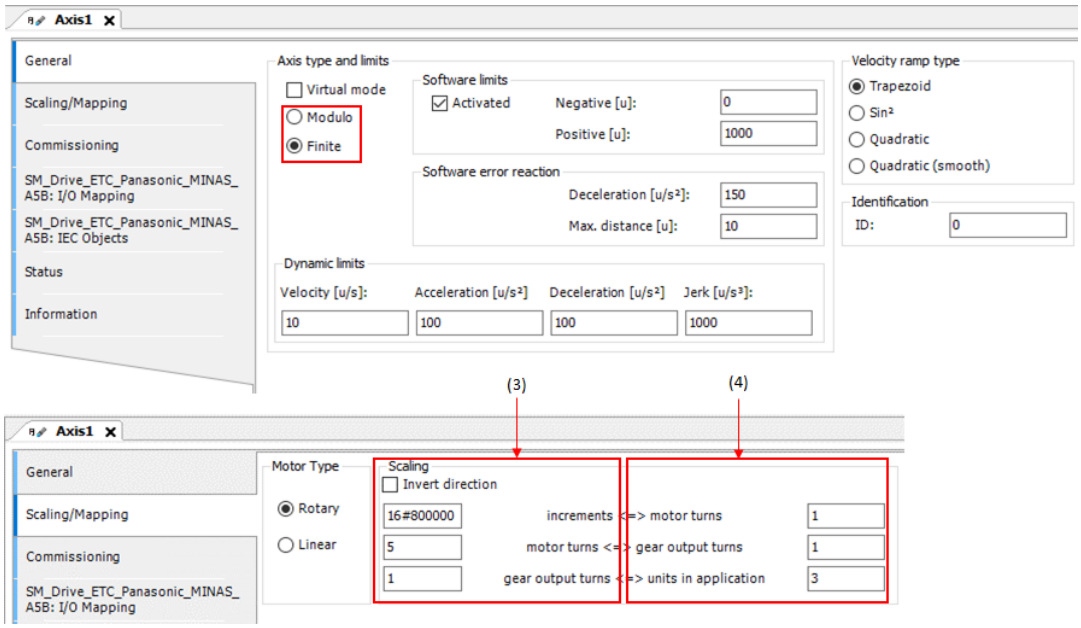
■ Setting location

The axis type, the modulo value and the scaling value in the figure below can be changed while GM1 is running.

- In the case of the modulo axis



- In the case of the finite axis



Info.

- When the GM1 is powered off, cold reset, or warm reset, it returns to the previous setting value.
- For scaling settings (gear ratio, encoder resolution), refer to the GM1 Series Reference Manual (Operation).
- When setting the modulo axis, set as follows.

```
dwRatioTechUnitsDenom    = "increments"×"motor turns"×"gear output turns" …in
the figure below(1)
                         = 16#800000 × 5 × 1 = 41943040
iRatioTechUnitsNum       = "motor turns"×"gear output turns"×"units in
application" …in the figure below(2)
                         = 1 × 1 × 360 = 360
fPositionPeriod          = the module value= 360
iMovementType            = 0(modulo)
```

- When setting the finite axis, set as follows.

```
dwRatioTechUnitsDenom    = "increments"×"motor turns"×"gear output turns" …in
the figure below(3)
                         = 16#800000 × 5 × 1 = 41943040
iRatioTechUnitsNum       = "motor turns"×"gear output turns"×"units in
application" …in the figure below(4)
                         = 1 × 1 × 3 = 3
iMovementType            = 1(finite)

fPositionPeriod is the same as the modulo value.
```

10.2.6 SMC_SetMovementType(Virtual axis type change)

This is a function block that changes the axis type and modulo value of a virtual axis.When the axis state is power_off or standstill,this FB can be used.

■ Icon

■ Parameter

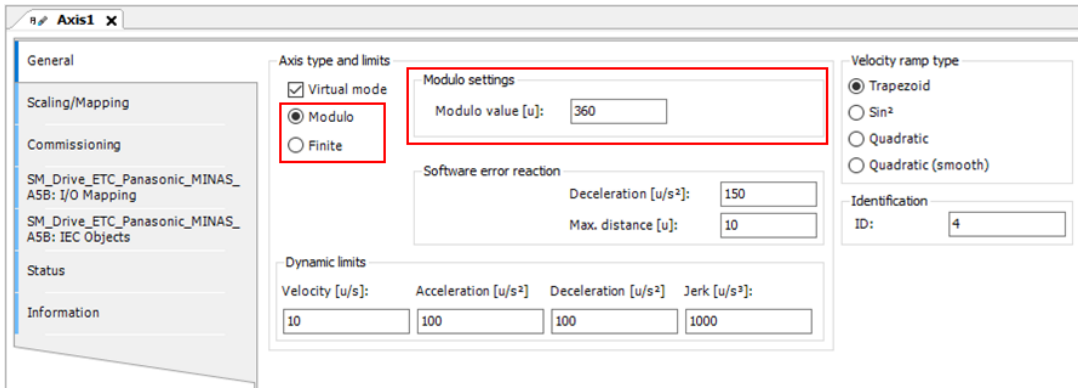
Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.(Note 1)
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	iMovementType	INT	0	Specify the axis type.0 is modulo.1 is finite.Other than error.
	fPositionPeriod	LREAL	1	the modululo value (Note 2)
Output	bDone	BOOL	FALSE	TRUE：The FB is in operation.
	bError	BOOL	FALSE	TRUE：An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

(Note 1) Supports the virtual axes. If used on the real axis, an error will occur.
When using on real axis, enable virtual mode.

(Note 2) Set a positive value.

■ Setting location

The axis type, the modulo value and the scaling value in the figure below can be changed while GM1 is running.



Info.

- When the GM1 is powered off, cold reset, or warm reset, it returns to the previous setting value.
- For the axis type, refer to the GM1 Series Reference Manual (Operation).

10.2.7 SMC_SetRampType(Velocity ramp type change)

This is a function block that changes the velocity ramp type of the real and virtual axes.When the axis state is power_off or standstill,this FB can be used.

■ Icon

■ Parameter

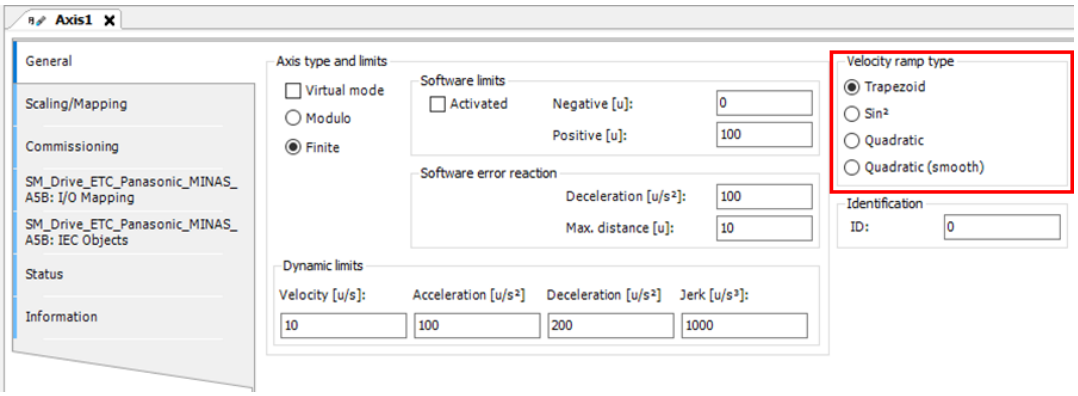
Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	eRampType	SMC_RAMPTYPE	0	Specifies the velocity ramp type.
Output	bDone	BOOL	FALSE	TRUE：The FB is in operation.
	bError	BOOL	FALSE	TRUE：An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

■ SMC_RAMPTYPE（Enumeration type）

Name	Value	Description
trapez	0	Trapezoid
sinsquare	1	sin^2
quadratic_ramp	2	Quadratic
quadratic_smooth_ramp	3	Quadratic (smooth)

■ Setting location

The velocity ramp type in the figure below can be changed while GM1 is running.



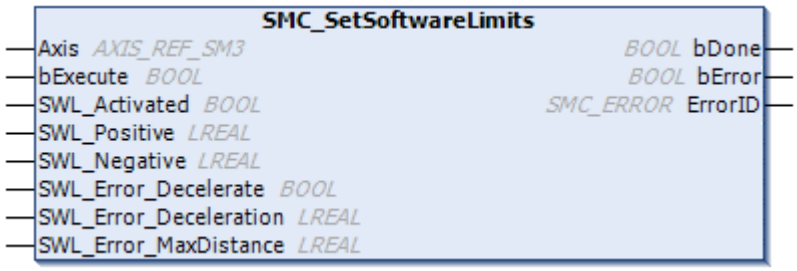
Info.

- When the GM1 is powered off, cold reset, or warm reset, it returns to the previous setting value.
- When CNC controlled by SMC_Interpolator, This FB has no effect.
- Check the GM1 Controller User's Manual (Operation Edition) for Axis operation specifications of the velocity ramp type.

10.2.8 SMC_SetSoftwareLimits(Soft limit change)

This is a function block that changes the enable / disable of soft limit of the real and virtual axes. It can be set without depending on the state of the axis.When the axis type is finite, the soft limit function is effective.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge.
	SWL_Activated	BOOL	FALSE	TRUE : Soft limit is valid. FALSE : Soft limit is invalid.
	SWL_Positive	LREAL	0	Soft limit for the positive direction[u] (Note 1)
	SWL_Negative	LREAL	0	Soft limit for negative direction[u] (Note 1)
	SWL_Error_Decelerate	BOOL	FALSE	Please do not use it.
	SWL_Error_Deceleration	LREAL	0	Deceleration when software error occurs[u /s²] (Note 2)
	SWL_Error_MaxDistance	LREAL	0	Maximum distance when a software error occurs[u] (Note 2)
Output	bDone	BOOL	FALSE	TRUE : The FB is in operation.
	bError	BOOL	FALSE	TRUE : An error has occurred.
	ErrorID	SMC_ERROR	SMC_NO_ERROR	An error ID is output.

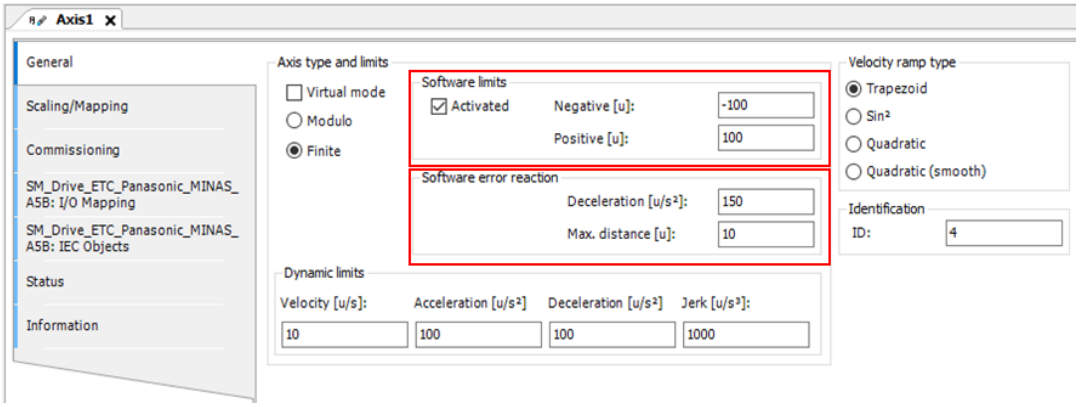
(Note 1) Set SWL_Positive to be larger than SWL_Negative.

(Note 2) Set a positive value.

When SWL_Error_Deceleration and WL_Error_MaxDistance are 0, Deceleration is fMaxDeceleration of SMC_ChangeDynamicLimits.

■ Setting location

The velocity ramp type in the figure below can be changed while GM1 is running.



Info.

- When the GM1 is powered off, cold reset, or warm reset, it returns to the previous setting value.
- For the soft limit, refer to the GM1 series reference manual (operation).

10.3 Motion Auxiliary Function (Other Functions)

10.3.1 PMC_ReadLatchPosition (Amplifier Latch Monitor)

This is a function block (FB) that monitor the AMP latch position. It reads the axis position when a trigger signal occurs.

■ **Icon**

■ **Parameter**

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Stops processing.
	nStartCancelState	IoDRVRTEX.LATCH_STATE	MONITOR_LATCH	Specifies the start and cancellation of the latch mode.
	nLatchTrg1	IoDRVRTEX.LATCH_TRIGGER	Z_PHASE	Selects the trigger signal for latch position 1
	nLatchTrg2	IoDRVRTEX.LATCH_TRIGGER	-	Selects the trigger signal for latch position 2
	nMonitorSel	IoDRVRTEX.MONITOR_SELECT		Selects the latch position to be output as the output MonitorData.
Output	Done	BOOL	FALSE	TRUE: Output is completed.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	An error ID is output.
	MonitorData	LREAL		Axis position is output.
	bLatchComp1	BOOL	FALSE	TRUE : Latch completed at latch position 1 (CH1).

Scope	Name	Type	Initial	Description
	bLatchComp2	BOOL	FALSE	TRUE : Latch completed at latch position 2 (CH2).

■ IoDRVRTEX.LATCH_STATE (Enumeration type)

Name	Value	Description
MONITOR_LATCH	80	Monitors the position latch state. Monitors the state without newly starting or canceling.
START_LATCH1	81	Starts the position latch 1 (CH1).
START_LATCH2	82	Starts the position latch 2 (CH2).
START_LATCH1_AND2	83	Starts the position latch 1 (CH1) and position latch 2 (CH2).
CANCEL_LATCH1	84	Cancels the position latch 1 (CH1).
CANCEL_LATCH2	88	Cancels the position latch 2 (CH2).
CANCEL_LATCH1_AND2	92	Cancels the position latch 1 (CH1) and position latch 2 (CH2).

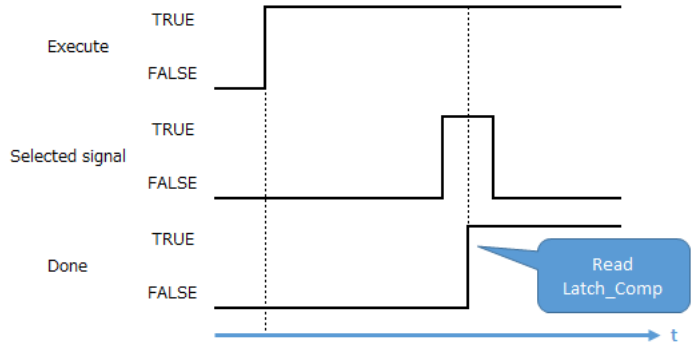
■ IoDRVRTEX.LATCH_TRIGGER (Enumeration type)

Name	Value	Description
Z_PHASE	0	Z phase
EXT1_RISING_EDGE	1	Rising edge of EXT1
EXT2_RISING_EDGE	2	Rising edge of EXT2
EXT3_RISING_EDGE	3	Rising edge of EXT3
PR7_111_RISING_EDGE	7	Not used for this FB.
EXT1_FALLING_EDGE	9	Falling edge of EXT1
EXT2_FALLING_EDGE	10	Falling edge of EXT2
EXT3_FALLING_EDGE	11	Falling edge of EXT3
PR7_111_FALLING_EDGE	15	Not used for this FB.

■ IoDRVRTEX.MONITOR_SELECT (Enumeration type)

Name	Value	Description
LPOS1	9	Latch position 1
LPOS2	10	Latch position 2

■ **Operations when the function block is executed**



The PMC_ReadLatchPosition function block outputs the following error.

Error	Description
SMC_WRONG_CONTROLLER_MODE	Executed in a mode other than the position control mode. Change to SMC_position using SMC_SetControllerMode.
SMC_RP_DRIVE_PARAMETER_NOT_MAPPED	Specified nLatchTrg1 and nLatchTrg2 to not use.
	Allocation of EXT1, EXT2, and EXT3 to the servo amplifier is faulty. Change the settings for Pr4.04 to Pr4.06.

As the PMC_ReadLatchPosition function block uses the RTEX home return command, it cannot be executed together with PMC_Home.

If PMC_ReadLatchPosition is executed while PMC_Home is being executed, the CommandAborted parameter of PMC_ReadLatchPosition becomes TRUE.

When using EXT1, EXT2, and EXT3 for nLatchTrg1 and nLatchTrg2, set amplifier parameters as shown in the following table.

Parameter	Parameter name	Settings
Pr4.04	SI5 input selection	EXT1
Pr4.05	SI6 input selection	EXT2
Pr4.06	SI7 input selection	EXT3

10.3.2 PMC_StopLatchPosition (Stop Amplifier Latch)

This is a function block (FB) that stops the axis at the AMP latch position. Stops the axis when a trigger event occurs.

■ Icon

■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge. FALSE: Stops processing.
	nLatchTrg1	IoDRVRTEX.LATCH_TRIGGER	EXT1_RISING_EDGE	Selects the trigger signal for latch position
	Velocity	LREAL	0	Specifies the velocity (u/s).
	Acceleration	LREAL	0	Specifies the acceleration (u/s^2).
	Deceleration	LREAL	0	Specifies the deceleration (u/s^2).
	Jerk	LREAL	0	Specifies the jerk (u/s^3).
	Direction	MC_Direction	negative	Specifies the traveling direction of the axis.
Output	InVelocity	BOOL	FALSE	TRUE: The axis has reached the specified velocity for the first time.
	CommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Done	BOOL	FALSE	TRUE: Stopping is completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	An error ID is output.
	MonitorData	LREAL		Axis position is output.

■ IoDRVRTEX.LATCH_TRIGGER (Enumeration type)

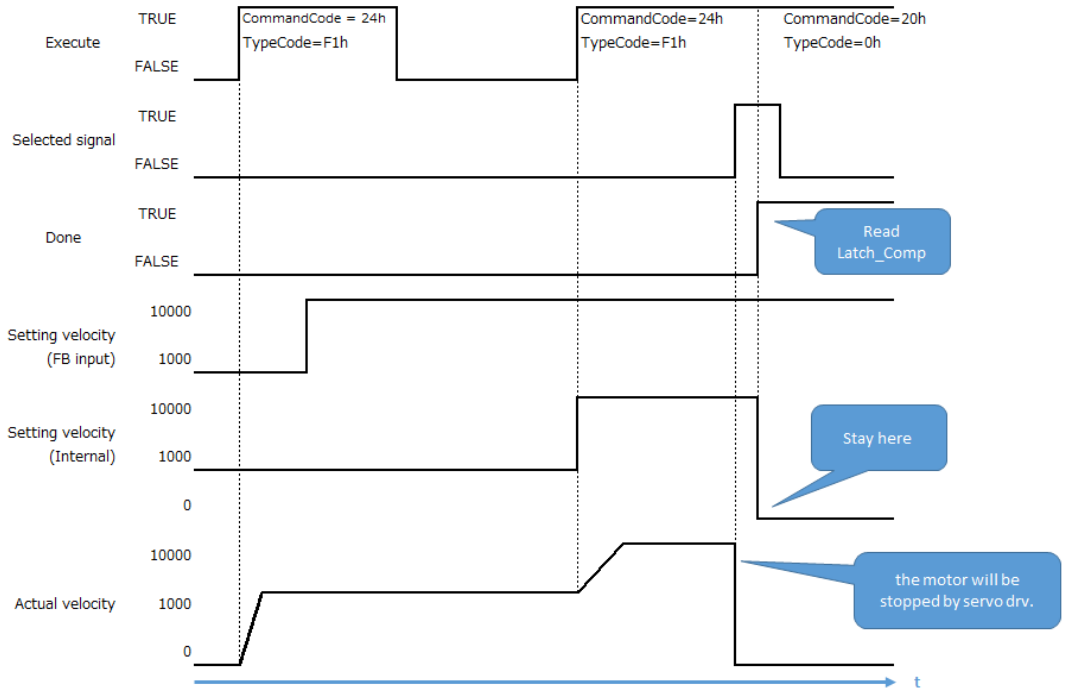
Name	Value	Description
Z_PHASE	0	Not used for this FB.
EXT1_RISING_EDGE	1	Rising edge of EXT1
EXT2_RISING_EDGE	2	Rising edge of EXT2

Name	Value	Description
EXT3_RISING_EDGE	3	Rising edge of EXT3
PR7_111_RISING_EDGE	7	Condition set by MINAS amplifier parameter Pr7.111
EXT1_FALLING_EDGE	9	Falling edge of EXT1
EXT2_FALLING_EDGE	10	Falling edge of EXT2
EXT3_FALLING_EDGE	11	Falling edge of EXT3
PR7_111_FALLING_EDGE	15	Condition set by MINAS amplifier parameter Pr7.111

■ MC_Direction (Enumeration type)

Name	Value	Description
positive	1	Travels in the positive direction.
negative	-1	Travels in the negative direction.
shortest	0	Not available. Do not specify this.
fastest	3	Not available. Do not specify this.
current	2	Travels to the current direction. Possible to use only for the modulo axis.

■ Operations when the function block is executed



- Execute = TRUE: Starts the latch mode. Execute = FALSE: Ends the latch, however, the axis operation continues as long as PMC_StopLatchPosition is called. Stop the axis using either MC_Stop or MC_Halt.

- When a trigger signal is input, the PMC_StopLatchPosition function block ignores the command value from the GM1 and stops at the latch position.

■ Execution errors

The PMC_StopLatchPosition function block outputs the following error.

Error	Description
SMC_WRONG_CONTROLLER_MODE	Executed in a mode other than the position control mode. Change to SMC_position using SMC_SetControllerMode.
SMC_RP_DRIVE_PARAMETER_NOT_MAPPED	Specified nLatchTrg1 to not use.
	Allocation of EXT1, EXT2, and EXT3 to the servo amplifier is faulty. Change the settings for Pr4.04 to Pr4.06.
SMC_DI_HOMING_ERROR	Servo amplifier version is lower than V1.24.
SMC_AXIS_NOT_READY_FOR_MOTION	The axis is in a state (Stopping, Disabled, or Errorstop) where PMC_StopLatchPosition cannot be executed.
SMC_REGULATOR_OR_START_NOT_SET	The servo OFF or brake is applied.
SMC_3SH_INVALID_VELACC_VALUES	The input (Velocity, Acceleration, or Deceleration) is faulty.
SMC_AXIS_REF_CHANGED_DURING_OPERATION	The Axis was changed during operation.

■ Execution conditions

- As the PMC_StopLatchPosition function block uses the RTEX home return command, it cannot be executed together with PMC_Home.
- To use the PMC_StopLatchPosition function block, the MINAS version must be V1.23 or higher.
- The function block supports only the control cycle of 1.0 ms and communication cycle of 0.5 ms.

■ Amplifier parameter conditions

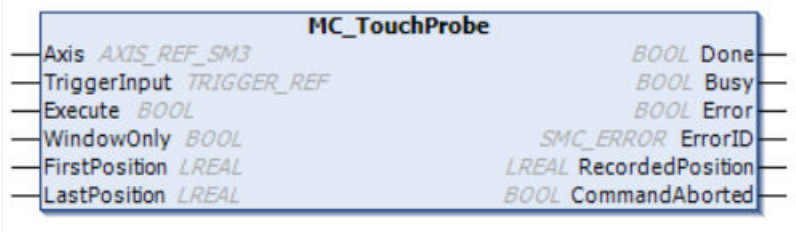
When using EXT1, EXT2, and EXT3 for nLatchTrg1, set amplifier parameters as shown in the following table.

Parameter	Parameter name	Settings
Pr4.04	SI5 input selection	EXT1
Pr4.05	SI6 input selection	EXT2
Pr4.06	SI7 input selection	EXT3

10.3.3 MC_TouchProbe (Enable AMP Latch Monitoring)

This a function block (FB) that reads the axis position when a trigger signal occurs.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
	TriggerInput	TRIGGER_REF	0	Specifies the trigger signal.
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge.
	WindowOnly	BOOL	FALSE	TRUE: The trigger event is accepted only in the specified window.
	FirstPosition	LREAL	0	The trigger event is accepted from the start position (in the positive direction). (Unit: [u])
	LastPosition	LREAL	0	The last position up to which the trigger event is accepted. (Unit: [u])
Output	Done	BOOL	FALSE	TRUE: Halt is completed.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	Error ID output
	RecordedPosition	LREAL	0	The position where the trigger event has occurred. (Unit: [u])
	CommandAborted	BOOL	FALSE	TRUE: An interruption is caused by another FB.

■ TRIGGER_REF (Structure)

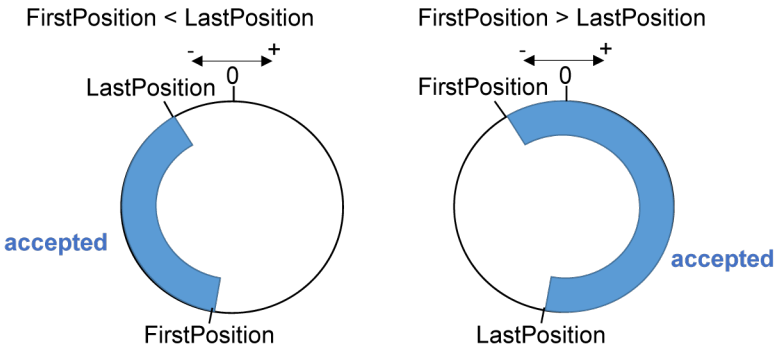
Member	Type	Default	Description
iTriggerNumber	INT	-1	Trigger channel: Defined by the driver. (Used only when bFastLatching is TRUE.)
bFastLatching	BOOL	TRUE	When bFastLatching is set to TRUE, latch is performed by the servo amplifier. When bFastLatching is set to FALSE, bInput is used as the trigger signal.
bInput	BOOL		When bFastLatching is set to FALSE, the trigger signal is input.
bActive	BOOL	FALSE	Internal variable. Do not set the value.

When using the MINAS, set iTriggerNumber as follows.

iTriggerNumber	Description
0	Rising edge of EXT1
1	Falling edge of EXT1
2	Rising edge of EXT2
3	Falling edge of EXT2

As for pin assignment, assign EXT1 to SI5 and EXT2 to SI6.

The range where the trigger event is accepted (WindowOnly) is as follows (in case of the modulo).



Note that hardware latch (EXT1 or EXT2 trigger) is not supported. Only software latch (bInput trigger) is supported.

10.3.4 MC_AbortTrigger (Disable AMP Latch Monitoring)

This is a function block (FB) that aborts the trigger event (MC_TouchProbe).

■ Icon

■ Parameter

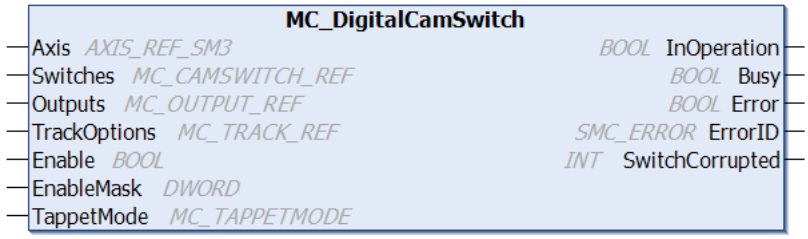
Scope	Name	Type	Default	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis.
	TriggerInput	TRIGGER_REF	0	Specifies the trigger signal.
Input	Execute	BOOL	FALSE	TRUE: Starts execution at the rising edge.
Output	Done	BOOL	FALSE	TRUE: Stopping is completed.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.

Scope	Name	Type	Default	Description
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	SMC_ERROR	0	Error ID output

10.3.5 MC_DigitalCamSwitch (Enable Digital Cam Switch)

This is a function block (FB) that performs ON / OFF control on the digital output according to the axis position. It assigns digital cam switches to tracks (maximum of 32). Switching operations can be controlled by specifying the ON / OFF position for each digital camp switch.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Axis	AXIS_REF_SM3	-	Specifies the axis where the switch is connected.
	Switches	MC_CAMSWITCH_REF	-	Specifies the switching operation.
	Outputs	MC_OUTPUT_REF	-	ON or OFF of the switch is output. ARRAY [1..32] OF BOOL
	TrackOptions	MC_TRACK_REF	-	Specifies the property of the track. ARRAY [1..32] OF MC_TRACK_TR
Input	Enable	BOOL	FALSE	TRUE: The FB can be executed.
	EnableMask	DWORD	16#FFFFFFFF	Specifies the track to be enabled. 1: Enabled, 0: Disabled The least significant bit is the 1st track. The most significant bit is the 32nd track.
	TappetMode	MC_TAPPETMODE	tp_mode_auto	Specifies the tappet mode.
Output	InOperation	BOOL	FALSE	TRUE: The track is enabled.
	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.

Scope	Name	Type	Initial	Description
	ErrorID	SMC_ERROR	0	An error ID is output.
	SwitchCorrupted	INT	-1	Index output of a faulty switch -1: No problem 0 to 31: A problem has occurred in switches 1 to 32.

■ MC_CAMSWITCH_REF (Structure)

Member	Type	Description
NoOfSwitches	BYTE	Number of switches Specifies the number of switches to be enabled when the FB is executed in the MC_CAMSWITCH_TR type array (1 to 32).
CamSwitchPtr	POINTER TO MC_CAMSWITCH_TR	Pointer to the first element of the MC_CAMSWITCH_TR type array

■ MC_CAMSWITCH_TR (Structure)

Member	Type	Description
TrackNumber	INT	Switch track number (1 to 32)
FirstOnPosition	LREAL	Position where the switch turns ON when the axis is moving in the positive direction
LastOnPosition	LREAL	Position where the switch turns OFF when the axis is moving in the positive direction Not used when CamSwitchMode is set to 1.
AxisDirection	INT	Movement direction where the switch is enabled 0: Both positive and negative directions 1: Only positive direction 2: Only negative direction
CamSwitchMode	INT	Control method that performs switch ON / OFF control 0: ON and OFF are both controlled by the position. 1: ON is controlled by the position and OFF is controlled by the time.
Duration	TIME	Specifies the time during which the switch remains ON for when CamSwitchMode is set to 1.
bOn	BOOL	Used within the FB.
CounterOff	INT	Used within the FB.

■ MC_TRACK_REF (Structure)

Member	Type	Description
OnCompensation	LREAL	Specifies the switch ON delay time in seconds. When a positive value is specified, the switch turns ON later by the time specified. When a negative value is specified, the switch turns ON earlier by the time specified.
OffCompensation	LREAL	Specifies the switch OFF delay time in seconds.

Member	Type	Description
		When a positive value is specified, the switch turns OFF after a delay of the time specified. When a negative value is specified, the switch turns OFF earlier by the time specified.
Hysteresis	LREAL	Specifies the hysteresis value (position).

■ MC_TAPPETMODE (Enumeration type)

Name	Value	Description
tp_mode_auto	0	Automatically determined according to the state. Servo ON state: Command position (fSetPosition) of the master axis Servo OFF state: Actual position (fActPosition) of the master axis
tp_mode_demandposition	1	Command state (fSetPosition) of the master axis
tp_mode_actualposition	2	Actual state (fActPosition) of the master axis

Regarding the method for entering defaults for variables of the MC_CAMSWITCH_TR type structure, refer to “Default Setting for Variables of the MC_TP_REF Type Structure”.

■ Operations when the function block is executed

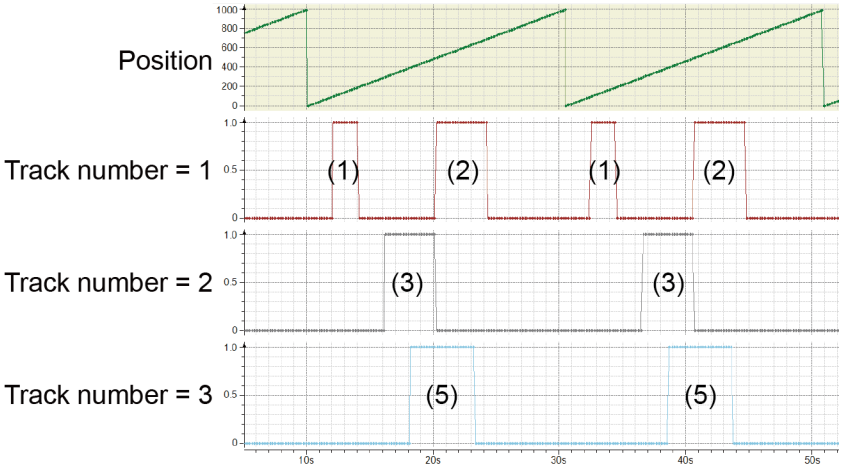
The following sections shows switching operations (Outputs) of each track when the function block is executed after setting the parameter as follows. The axis is set to the modulo (modulo value: 1000).

Function block input parameters

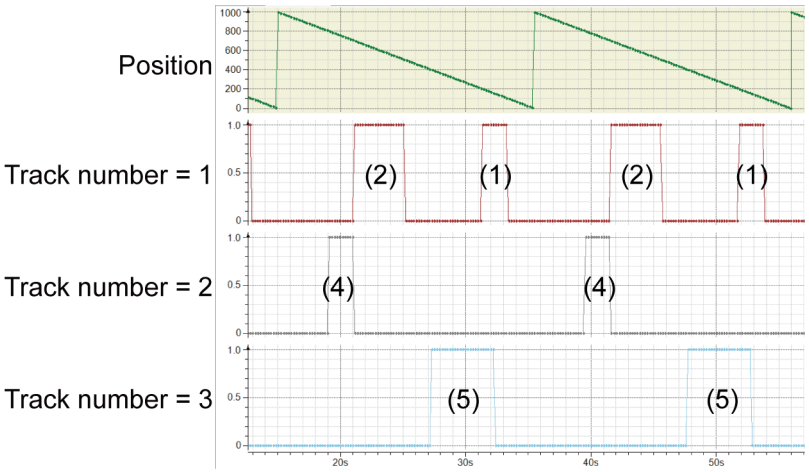
Five switches (CamSwitchPtr) are set.

Switch	Index	Track Number	FirstOn Position	LastOn Position	Axis Direction	Cam SwitchMode	Duration
(1)	1	1	100	200	0 (Both)	0 (Position)	T#0ms
(2)	2	1	500	700	0 (Both)	0 (Position)	T#0ms
(3)	3	2	300	500	1 (Positive direction)	0 (Position)	T#0ms
(4)	4	2	700	800	2 (Negative direction)	0 (Position)	T#0ms
(5)	5	3	400	0	0 (Both)	1 (Time)	T#5s

Switching operations when the axis is moved in the positive direction
(1) to (5) are switch numbers.



Switching operations when the axis is moved in the negative direction



■ Detection of faulty switch operation (SwitchCorrupted)

SwitchCorrupted occurs when the switch does not turn ON/OFF as set.

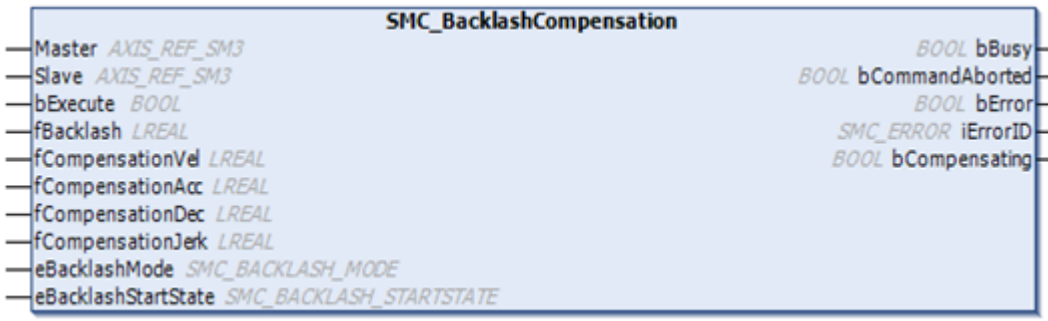
REFERENCE

5.6.6 Default Setting for Variables of the MC_TP_REF Type Structure

10.3.6 SMC_BacklashCompensation (Compensate Backlash)

This is a function block (FB) that compensates the backlash.

■ Icon



■ Parameter

Scope	Name	Type	Initial	Description
Input / output	Master	AXIS_REF_SM3	-	Specifies the master axis.
	Slave	AXIS_REF_SM3	-	Specifies the slave axis.
Input	bExecute	BOOL	FALSE	Starts execution at the rising edge. Remains enabled until the slave axis is interrupted by another operation or until an error occurs.
	fBacklash	LREAL	0	Distance to compensate (backlash)
	fCompensationVel	LREAL	0	Additional velocity used when compensation is performed (A value to be added to the master axis velocity)
	fCompensationAcc	LREAL	0	Additional acceleration used when compensation is performed (A value to be the maximum acceleration when compensation is performed)
	fCompensationDec	LREAL	0	Additional deceleration used when compensation is performed. (A value to be the maximum deceleration when compensation is performed)
	fCompensationJerk	LREAL	0	Additional jerk used when compensation is performed (Even if any value is set, the setting is disabled.)
	eBacklashMode	SMC_BACKLASH_MODE	SMC_BL_AUTO	Backlash compensation mode
	eBacklashStartState	SMC_BACKLASH_STARTSTATE	SMC_BL_START_NONE	Specifies the start conditions whether compensation is required or not when starting the backlash compensation.

Scope	Name	Type	Initial	Description
Output	bBusy	BOOL	FALSE	TRUE: The FB is in operation.
	bCommandAborted	BOOL	FALSE	TRUE: An interruption from other FB has occurred.
	bError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	iErrorID	SMC_ERROR	0	An error ID is output.
	bCompensating	BOOL	FALSE	TRUE: Backlash compensation in operation

■ MC_BACKLASH_MODE (Enumeration type)

Name	Value	Description
SMC_BL_AUTO	2	Compensation in the traveling direction of the master axis
SMC_BL_POSITIVE	1	Compensation in the positive direction
SMC_BL_NEGATIVE	-1	Compensation in the negative direction
SMC_BL_OFF	0	No backlash compensation

■ SMC_BACKLASH_STARTSTATE (Enumeration type)

Name	Value	Description
SMC_BL_START_NEGATIVE	-1	If the slave axis is driven in the negative direction when compensation is started: ● To make the axis travel in the positive direction, compensation is required for the backlash distance (fBacklash). ● No compensation is required for the travels in the negative direction.
SMC_BL_START_NONE	0	If the slave axis is not driven in either direction when compensation is started: To make the axis travel in the positive or negative direction, compensation is required for half the amount of the backlash distance (fBacklash).
SMC_BL_START_POSITIVE	1	If the slave axis is driven in the positive direction when compensation is started: ● No compensation is required for the travels in the positive direction. ● To make the axis travel in the negative direction, compensation is required for the backlash distance (fBacklash).

- When starting operation, make sure that both the master axis and slave axis are in the same position. If they are not set at the same position, the slave axis travels to the master axis position at the moment when SMC_BacklashCompensation is executed.
- SMC_BacklashCompensation functions in the same way as the phase synchronous operation (MC_Phasing) and the phase depends on the master axis direction.

11 Other Function Blocks

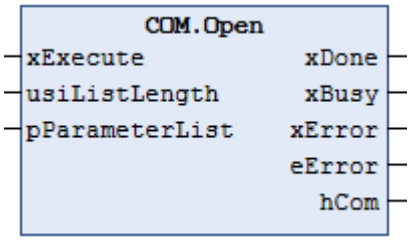
11.1 COM Port (General-purpose Communication)

This section describes function blocks that are used to perform general-purpose communication with the COM port.

11.1.1 COM.Open (Open COM port)

This is a function block that opens a COM port. It reads from and writes to the COM port using the output handle. Close the opened COM port using the COM.Close instruction.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	usiListLength	USINT	Number of pParameterList entries
	pParameterList	COM.CAA.PVOID	A pointer to the communication setting parameter list for the COM port. Specifies the pointer to the COM.PARAMETER structure array.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	COM.ERROR	An error ID is output. Refer to "11.1.5 COM.ERROR (Error ID)".
	hCom	COM.CAA.HANDLE	Handle of the opened COM port.

■ COM.PARAMETER (Structure)

Member	Type	Description
udiParameterId	UDINT	Parameter ID to be set in the COM port. For a list of parameters, refer to "COM.CAA_Parameter_Constants (Constants)".
udiValue	UDINT	Value to be set in the COM port

■ COM.CAA_Parameter_Constants (Constants)

Name	Value	Support	Description
udiPort	16#1	Supported	Port number (Fixed to 1.)
udiStopBits	16#2	Supported	Stop bit Refer to "COM.STOPBIT (Enumeration type)".
udiParity	16#3	Supported	Parity Refer to "COM.PARITY (Enumeration type)".
udiBaudrate	16#4	Supported	Baud rate (Can be selected from 9600, 19200, 38400, 57600, and 115200)
udiTimeout	16#5	Not supported	Timeout
udiBufferSize	16#6	Not supported	Buffer size parameter Specifies a serial buffer size.
udiByteSize	16#7	Supported	Byte size parameter Sets the number of data bits to 4 to 8. (Specify 7 or 8 for the GM1 Controller.)
udiBinary	16#8	Not supported	Binary parameter Enables the binary mode. (With the GM1 Controller, it is fixed to 0 (binary mode).)
udiOutxCtsFlow (Note 1)	16#9	Not supported	CTS handshake for the output parameter
udiOutxDsrFlow (Note 1)	16#A	Not supported	DSR handshake for the output parameter
udiDtrControl (Note 1)	16#B	Not supported	DTR flow control parameter
udiDsrSensitivity (Note 1)	16#C	Not supported	DSR sensitivity parameter
udiRtsControl (Note 1)	16#D	Not supported	Rts flow control parameter
udiTXContinueOnXoff (Note 1)	16#E	Not supported	XOFF continues Tx parameter.
udiOutX (Note 1)	16#F	Not supported	XON / XOFF output flow control parameter
udiInX (Note 1)	16#10	Not supported	XON / XOFF of the flow control parameter
udiXonChar (Note 1)	16#11	Not supported	Tx AND Rx XON character parameter
udiXoffChar (Note 1)	16#12	Not supported	Tx AND Rx XOFF character parameter
udiXonLim (Note 1)	16#13	Not supported	Sends XON threshold parameter
udiXoffLim (Note 1)	16#14	Not supported	Sends XOFF threshold parameter

(Note 1) The GM1 Controller does not support the flow control.

■ COM.STOPBIT (Enumeration type)

Name	Value	Description
ONESTOPBIT	0	1 stop bit
ONE5STOPBITS	1	1.5 stop bit (Not available)
TWOSTOPBITS	2	2 stop bit

■ COM.PARITY (Enumeration type)

Name	Value	Description
EVEN	0	Even
ODD	1	Odd
NONE	2	None

■ ST Program Example

Declaration section

```
VAR
Open : COM.Open;
OpenParam : ARRAY [1..7] OF COM.PARAMETER := [
    (udiParameterID := COM.CAA_Parameter_Constants.udiPort,     udiValue := 2
),
    (udiParameterID := COM.CAA_Parameter_Constants.udiBaudrate, udiValue := 1
15200),
    (udiParameterID := COM.CAA_Parameter_Constants.udiParity,   udiValue := I
NT_TO_UDINT(COM.PARITY.ODD)),
    (udiParameterID := COM.CAA_Parameter_Constants.udiStopBits, udiValue := I
NT_TO_UDINT(COM.STOPBIT.ONESTOPBIT)),
    (udiParameterID := COM.CAA_Parameter_Constants.udiTimeout,  udiValue := 0
),
    (udiParameterID := COM.CAA_Parameter_Constants.udiByteSize, udiValue := 8
),
    (udiParameterID := COM.CAA_Parameter_Constants.udiBinary,   udiValue := 1
)
];
END_VAR
```

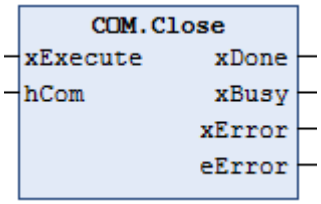
Implementation section

```
Open( xExecute := TRUE , pParameterList := ADR(OpenParam) , usiListLength :=
SIZEOF(OpenParam) / SIZEOF(COM.PARAMETER) );
```

11.1.2 COM.Close (Close COM Port)

This is a function block that closes the COM port.

■ Icon



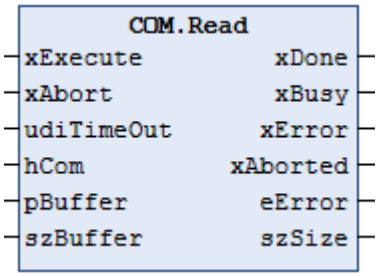
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	hCom	COM.CAA.HANDLE	Handle of the COM port to be closed
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	COM.ERROR	An error ID is output. Refer to "11.1.5 COM.ERROR (Error ID)".

11.1.3 COM.Read (Read COM Port)

This is a function block that reads data from the COM port.

■ Icon



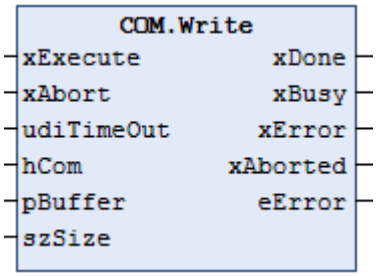
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	xAbort	BOOL	TRUE: Stops execution and resets all outputs.
	udiTimeOut	UDINT	Timeout time until the execution is stopped (μs)
	hCom	COM.CAA.HANDLE	Handle of the COM port
	pBuffer	CAA.PVOID	Pointer to the buffer that acquires data read from the COM port
	szBuffer	CAA.SIZE	Maximum byte of pBuffer
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	xAborted	BOOL	TRUE: Execution is stopped by the user.
	eError	COM.ERROR	An error ID is output. Refer to "11.1.5 COM.ERROR (Error ID)".
	szSize	COM.CAA.SIZE	Data size (bytes) acquired by the pBuffer

11.1.4 COM.Write (Write COM Port)

This is a function block that writes data to the COM port.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	xAbort	BOOL	TRUE: Stops execution and resets all outputs.
	udiTimeOut	UDINT	Timeout time until the execution is stopped (μs)
	hCom	COM.CAA.HANDLE	Handle of the COM port
	pBuffer	CAA.PVOID	Pointer to the buffer of the data written to the COM port
	szSize	COM.CAA.SIZE	Data size (bytes) of the pBuffer to be written to the COM port
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	xAborted	BOOL	TRUE: Execution is stopped by the user.
	eError	COM.ERROR	An error ID is output. Refer to "11.1.5 COM.ERROR (Error ID)".

11.1.5 COM.ERROR (Error ID)

This is an enumeration type error ID that is output when the COM port (general-purpose communication) function block is executed.

■ COM .ERROR (Enumeration type)

Name	Value	Description
NO_ERROR	0	No error
TIME_OUT	5001	Timeout error
ABORT	5002	xAbort input enabled
HANDLE_INVALID	5003	Invalid handle
ERROR_UNKNOWN	5004	Unknown error
WRONG_PARAMETER	5005	Wrong parameter
WRITE_INCOMPLETE	5006	Incomplete write

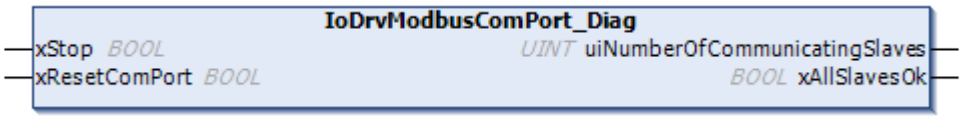
11.2 COM port (Modbus COM)

This section describes the instructions that are used to perform ModbusRTU communication with the COM port.

11.2.1 IoDrvModbusComPort

This is a function block that controls the Modbus_Master_COM_Port device.

■ **Icon**



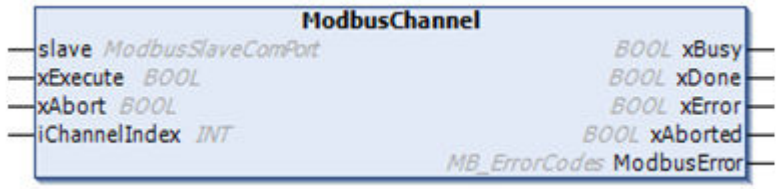
■ **Parameter**

Scope	Name	Type	Description
Input	xStop	BOOL	TRUE: Stops sending a new request to the slave. FALSE: Continues the current request.
	xResetComPort	BOOL	Closes the COM port at a rising edge.
Output	uiNumberOfCommunicatingSlaves	UINT	Number of remote slaves under communication.
	xAllSlavesOk	BOOL	TRUE: All slaves are communicating normally. FALSE: An error has occurred in one of the slaves.

11.2.2 IoDrvModbus.ModbusChannel(Start Sending Modbus Command)

This is a function block that sends the commands set in the Modbus Slave channel of the ModbusSlaveCOM_Port device.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts sending commands at the rising edge.
	xAbort	BOOL	TRUE: Stops execution and resets all outputs.

Scope	Name	Type	Description
	iChannelIndex	INT	Channel number where commands to be sent are set
I/O	slave	ModbusSlaveComPort	Handle of the ModbusSlaveComPort device
Output	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xDone	BOOL	TRUE: Processing is completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	xAborted	BOOL	TRUE: Execution is stopped by the user's xAbort input.
	ModbusError	MB_ErrorCodes	An error code is output. Refer to "11.5.5 IoDrvModbus.MB_ErrorCodes (Error Codes)".

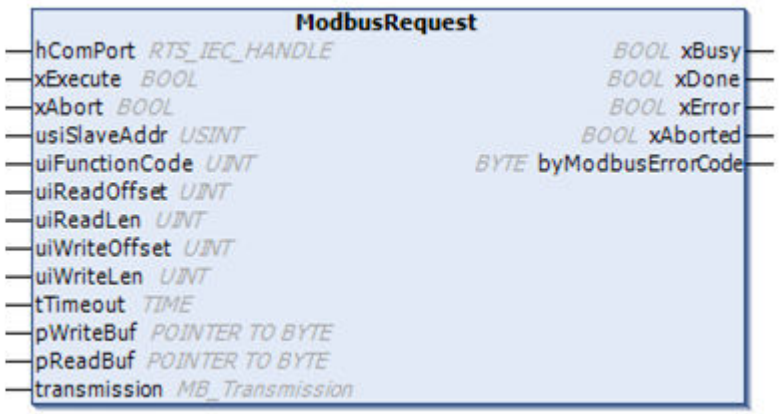
11.2.3 IoDrvModbus.ModbusRequest (Modbus Request)

This is a function block that processes the Modbus command specified by I/O without using the ModbusMasterComPort device.

■ Supported commands

- Command 1 (Read multi-point coil state)
- Command 2 (Read multi-point input state)
- Command 3 (Read multi-point holding register)
- Command 4 (Read multi-point input register)
- Command 5 (Write single-point coil)
- Command 6 (Write single-point holding register)
- Command 15 (Write multi-point coil)
- Command 16 (Write multi-point holding register)
- Command 23 (Read / write multi-point holding register)

■ Icon



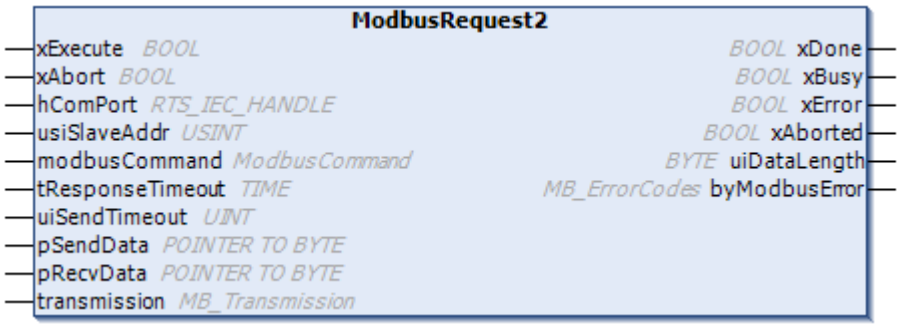
■ Parameter

Scope	Name	Type	Description
Input	hComPort	RTS_IEC_HANDLE	COM port handle acquired by COM.Open
	xExecute	BOOL	Starts sending commands at the rising edge.
	xAbort	BOOL	TRUE: Stops execution and resets all outputs.
	usiSlaveAddr	USINT	Slave address 1 to 247
	uiFunctionCode	UINT	Modbus function code
	uiReadOffset	UINT	Read address offset (0 to 65535)
	uiReadLen	UINT	Read length (1 to 125)
	uiWriteOffset	UINT	Write address offset (0 to 65535)
	uiWriteLen	UINT	Write length (1 to 121)
	tTimeout	UINT	Timeout value (in ms units)
	pWriteBuf	POINTER TO BYTE	Pointer to the send buffer.
	pReadBuf	POINTER TO BYTE	Pointer to the receive buffer
	transmission	MB_Transmission	Transmission type (RTU / ASCII) * Supports only RTU.
Output	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xDone	BOOL	TRUE: Processing is completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	xAborted	BOOL	TRUE: Execution is stopped by the user's xAbort input.
	byModbusErrorCode	BYTE	An error code is output. Refer to "11.2.6 IoDrvModbus.MB_ErrorCodes (Error Codes)".

11.2.4 IoDrvModbus.ModbusRequest 2 (Modbus Request 2)

This is a function block that processes, like the ModbusRequest, the Modbus command specified by I/O without using the ModbusMasterComPort device. It is different from ModbusRequest in that the structure type is used to specify the Modbus command.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts sending commands at the rising edge.
	xAbort	BOOL	TRUE: Stops execution and resets all outputs.
	hComPort	RTS_IEC_HANDLE	COM port handle acquired by COM.Open
	usiSlaveAddr	USINT	Slave address 1 to 247
	modbusCommand	ModbusCommnad	Modbus command
	tResponseTimeout	TIME	Timeout (in ms units) of the response for a request
	uiSendTimeout	UINT	Transmission timeout
	pSendData	UINT	Pointer to the send data
	pRecvData	UINT	Pointer to the receive data
	transmission	MB_Transmission	Transmission type (RTU / ASCII) * Supports only RTU.
Output	xDone	BOOL	TRUE: Processing is completed.
	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	xAborted	BOOL	TRUE: Execution is stopped by the user's xAbort input.
	uiDataLength	BYTE	Received data length (byte)
	byModbusError	MB_ErrorCodes	An error code is output. Refer to "11.2.6 IoDrvModbus.MB_ErrorCodes (Error Codes)".

■ ModbusCommand (Structure)

Name	Type	Description
uiFunctionCode	UINT	Modbus command code
uiReadOffset	UINT	Read address 0 to 65535
uiReadLen	UINT	Range in the number of read instances varies depending on commands.
uiWriteOffset	UINT	Write address 0 to 65535
uiWriteLen	UINT	Range in the number of write instances varies depending on commands.

11.2.5 IoDrvModbus.ModbusSlaveComPort

This is a function block that controls the Modbus_Slave_COM_Port device.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	xTrigger	BOOL	Sends all the commands of the Modbus channel at the rising edge.
	xReset	BOOL	Resets xError and byModbusError and resumes communication.
	xAcknowledge	BOOL	Resumes communication without resetting xError and byModbusError.
	xDoInit	BOOL	TRUE: Sends a slave initialization command when communication is resumed.
Output	xInitDone	BOOL	TRUE: Modbus slave initialization command is fully completed.
	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xDone	BOOL	TRUE: Processing is completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	byModbusError	MB_ErrorCodes	An error code is output. Refer to "11.2.6 IoDrvModbus.MB_ErrorCodes (Error Codes)".
	iChannelIndex	INT	Channel index

11.2.6 IoDrvModbus.MB_ErrorCodes (Error Codes)

This is an enumeration type error code that is output when the function block for Modbus communication instruction that uses the COM port is executed.

■ IoDrvModbus.MB_ErrorCodes (Enumeration type)

Name	Value	Description
RESPONSE_SUCCESS	16#0	Succeeded
ILLEGAL_FUNCTION	16#1	Function code not supported by the slave
ILLEGAL_DATA_ADDRESS	16#2	Register offset not supported by the slave
ILLEGAL_DATA_VALUE	16#3	Illegal data writing
SLAVE_DEVICE_FAILURE	16#4	Non-recoverable error
ACKNOWLEDGE	16#5	Start operation
SLAVE_DEVICE_BUSY	16#6	During operation
MEMORY_PARITY_ERROR	16#8	Memory parity error
GATEWAY_PATH_UNAVAILABLE	16#A	Gateway path unavailable
GATEWAY_DEVICE_FAILED_TO_RESPOND	16#B	Gateway device failed to respond
RESPONSE_TIMEOUT	16#A1	Timeout
RESPONSE_CRC_FAIL	16#A2	CRC error
RESPONSE_WRONG_SLAVE	16#A3	Wrong response
RESPONSE_WRONG_FUNCTIONCODE	16#A4	Wrong function code in the response
REQUEST_FAILED_TO_SEND	16#A5	Request not sent
RESPONSE_INVALID_DATA	16#A6	Invalid response data
RESPONSE_INVALID_PROTOCOL	16#A7	Invalid response protocol
RESPONSE_INVALID_HEADER	16#A8	Invalid response header
UNDEFINED	16#FF	Undefined

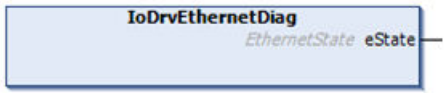
11.3 LAN port (IoDrvEthernet)

This section describes the library functions that are used for the network interface to perform communication with the LAN port.

11.3.1 IoDrvEthernet

This is a function block that acquires the status of the LANPort device.

■ Icon



■ Parameter

Scope	Name	Type	Description
Output	eState	EthernetState	Ethernet state Refer to "EthernetState (Enumeration type)".

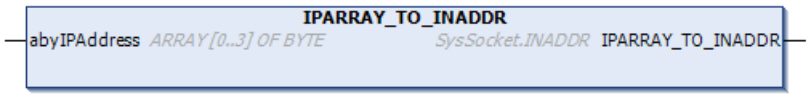
■ EthernetState (Enumeration type)

Name	Value	Description
NOT_CONFIGURED	0	Before configuration
CONFIGURED	1	After configuration
DISCONNECTED	2	Disconnected
RUNNING	3	Being executed
ERROR	4	An error has occurred.
SET_IP_ERROR	5	An IP error has occurred.

11.3.2 IoDrvEthernet.IPARRAY_TO_INADDR (Array Type to Union Type)

This is a function that converts an array type IP address to an INADDR (union type).

■ Icon



■ Parameter

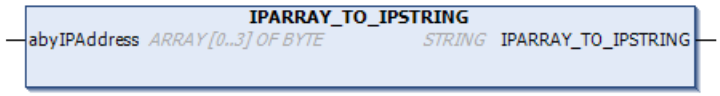
Scope	Name	Type	Description
Input	abyIPAddress	ARRY[0..3] OF BYTE	IP address array

Scope	Name	Type	Description
Output	IPARRAY_TO_INADDR	SysSocket.INADDR	Union type IP address

11.3.3 IoDrvEthernet.IPARRAY_TO_IPSTRING (Array Type to Character String Type)

This is a function that converts an array type IP address to a character string type.

■ Icon



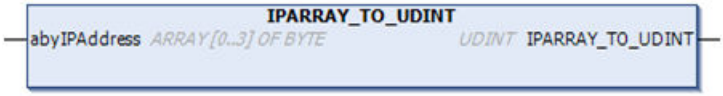
■ Parameter

Scope	Name	Type	Description
Input	abyIPAddress	ARRY[0..3] OF BYTE	IP address array
Output	IPARRAY_TO_STRING	STRING	Character string type IP address

11.3.4 IoDrvEthernet.IPARRAY_TO_UDINT (Array Type to UDINT Type)

This is a function that converts an array type IP address to a UDINT type.

■ Icon



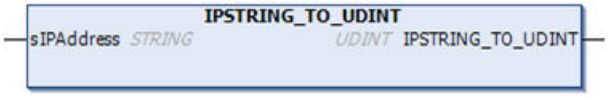
■ Parameter

Scope	Name	Type	Description
Input	abyIPAddress	ARRY[0..3] OF BYTE	IP address array
Output	IPARRAY_TO_UDINT	UDINT	UDINT type IP address

11.3.5 IoDrvEthernet.IPSTRING_TO_UDINT (Character String Type to UDINT Type)

This is a function that converts a character string type IP address to a UDINT type.

■ Icon



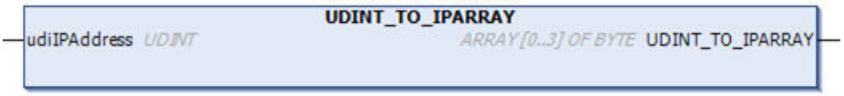
■ Parameter

Scope	Name	Type	Description
Input	abyIPAddress	STRING	Character string type IP address
Output	IPARRAY_TO_UDINT	UDINT	UDINT type IP address

11.3.6 IoDrvEthernet.UDINT_TO_IPARRAY (UDINT Type to Array Type)

This is a function that converts a UDINT type IP address to an array type.

■ Icon



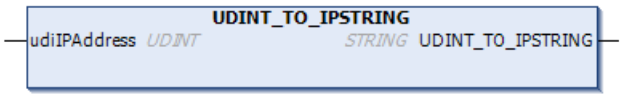
■ Parameter

Scope	Name	Type	Description
Input	abyIPAddress	UDINT	UDINT type IP address
Output	UDINT_TO_IPARRAY	ARRY[0..3] OF BYTE	IP address array

11.3.7 IoDrvEthernet.UDINT_TO_IPSTRING (UDINT Type to Character String Type)

This is a function that converts a UDINT type IP address to an array type.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	abyIPAddress	UDINT	UDINT type IP address
Output	UDINT_TO_STRING	STRING	Character string type IP address

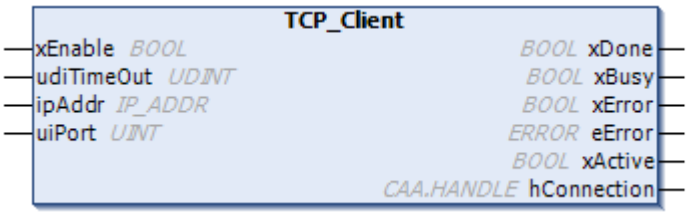
11.4 LAN Port (General-purpose Communication)

This section describes the library functions that are used to perform general-purpose communication with the LAN port using the TCP or UDP protocol.

11.4.1 NBS.TCP_Client (Connect to TCP Client)

This is a function block that connects to the TCP/IP client.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Description
Input	xEnable	BOOL	TRUE: Active
	udiTimeOut	UDINT	Connection timeout (us) No timeout when set to 0.
	ipAddr	NBS.IP ADDR	Server IP address (character string type)
	uiPort	UINT	Server port No.
Output	xDone	BOOL	TRUE: Processing is completed.
	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	NBS.ERROR	Connection result Refer to "11.4.8 NBS.ERROR (Error Code)".
	xActive	BOOL	TRUE: Connection is established.
	hConnection	CAA.HANDLE	Connection handle (Valid when xActive = TRUE)

11.4.2 NBS.TCP_Connection (Connect TCP)

This is a function block that establishes the connection of the client connecting to the connection port opened by TCP_Server.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	xEnable	BOOL	TRUE: Active
	hServer	CAA.HANDLE	Connection port handle acquired by TCP_Server
Output	xDone	BOOL	TRUE: Processing is completed.
	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	NBS.ERROR	Connection result Refer to "11.4.8 NBS.ERROR (Error Code)".
	xActive	BOOL	TRUE: Connection is established.(Note 1)
	hConnection	CAA.HANDLE	Connection handle (Valid when xActive = TRUE)

(Note 1) To detect a disconnection from the client after the line is connected, it is necessary to periodically call TCP_Read.

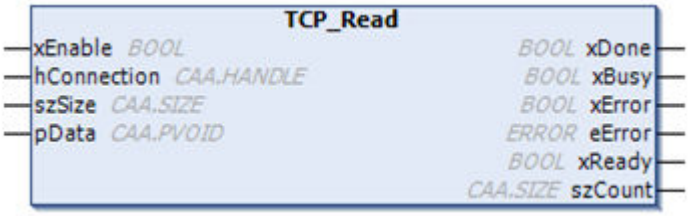
Info.

- When multiple clients are connected simultaneously to the same port, multiple TCP_Connection instances are created.
- The hServer handle acquired by one TCP_Server is set to the multiple TCP_Connection instances.

11.4.3 NBS.TCP_Read (Receive TCP Data)

This is a function block that acquires data received by the connection port that is established by TCP_Connection.

■ Icon



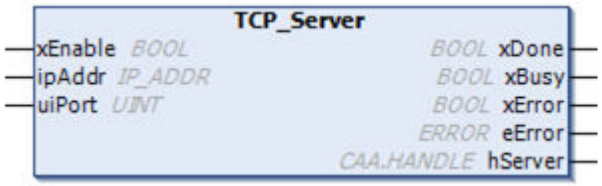
■ Parameter

Scope	Name	Type	Description
Input	xEnable	BOOL	TRUE: Active
	hConnection	CAA.HANDLE	Connection port handle acquired by TCP_Connection
	szSize	CAA.SIZE	Received buffer size (byte)
	pData	CAA.PVOID	Pointer to the receive buffer
Output	xDone	BOOL	Always FALSE
	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	NBS.ERROR	Connection result Refer to "11.4.8 NBS.ERROR (Error Code)".
	xReady	BOOL	TRUE: Data is received.
	szCount	CAA.SIZE	Received data size (byte)

11.4.4 NBS.TCP_Server (Connect TCP Server)

This is a function block that opens the specified port as a TCP/IP connection port.

■ Icon



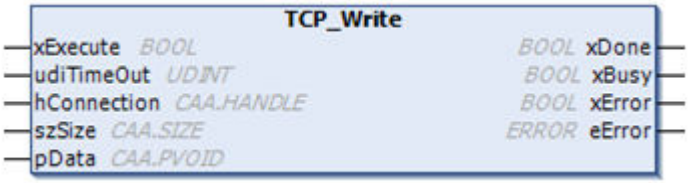
■ Parameter

Scope	Name	Type	Description
Input	xEnable	BOOL	TRUE: Active
	ipAddr	NBS.IP_ADDR	Home IP address (character string), LANPort1 or LANPort2 IP address
	uiPort	UINT	Home waiting port number
Output	xDone	BOOL	TRUE: Processing is completed.
	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	NBS.ERROR	Connection result Refer to "11.4.8 NBS.ERROR (Error Code)".
	hServer	CAA.HANDLE	Connection handle used by TCP_Connection

11.4.5 NBS.TCP_Write (Send TCP Data)

This is a function block that sends data to the connection port that is established by TCP_Connection.

■ Icon



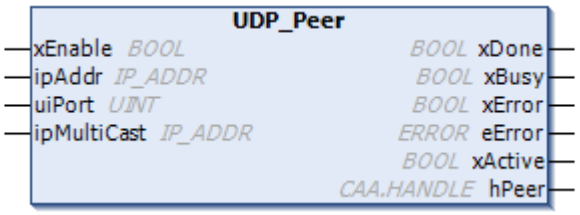
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	TRUE: Send started (edge) FALSE: Processing ended (edge)
	udiTimeOut	UDINT	Timeout (us)
	hConnection	CAA.HANDLE	Connection port handle acquired by TCP_Connection
	szSize	CAA.SIZE	Send data size (byte)
	pData	CAA.PVOID	Pointer to the send data buffer.
Output	xDone	BOOL	TRUE: Processing is completed.
	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	NBS.ERROR	Connection result Refer to "11.4.8 NBS.ERROR (Error Code)".

11.4.6 NBS.UDP_Peer (Open UDP Port)

This is a function block that opens the UDP/IP port.

■ Icon



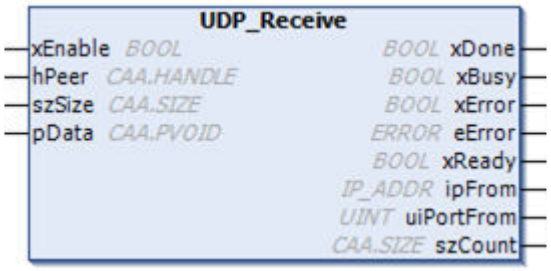
■ Parameter

Scope	Name	Type	Description
Input	xEnable	BOOL	TRUE: Active FALSE: Stop (xDone, xBusy, and xError are reset.)
	ipAddr	NBS.IP_ADDR	Home IP address (character string), LANPort1 or LANPort2 IP address
	uiPort	UINT	Home port number; Not possible to set to 0
	ipMultiCast	NBS.IP_ADDR	Multicast address ("255.255.255.255"=> INADDR_NONE)
Output	xDone	BOOL	TRUE: Processing is completed.
	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	NBS.ERROR	Connection result Refer to "11.4.8 NBS.ERROR (Error Code)".
	xActive	BOOL	TRUE: Connection is established.
	hPeer	CAA.HANDLE	Connection handle (Valid when xActive = TRUE)

11.4.7 NBS.UDP_Receive (Receive UDP Data)

This is a function block that receives data to the connection handle acquired by UDP_Peer.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xEnable	BOOL	TRUE: Active FALSE: Stop (xDone, xBusy, and xError are reset.)
	hPeer	CAA.HANDLE	Connection handle acquired by UDP_Peer
	szSize	CAA.SIZE	Receive data buffer size (byte)
	pData	CAA.PVOID	Pointer to the receive data buffer
Output	xDone	BOOL	Always FALSE
	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	NBS.ERROR	Connection result Refer to "11.4.8 NBS.ERROR (Error Code)".
	xReady	BOOL	TRUE: Data acquired, FALSE: No received data
	ipFrom	NBS.IP_ADDR	Data sending source IP address
	uiPortFrom	UINT	Data sending source port No.
	szCount	CAA.SIZE	Received data size (byte)

(Note 1) If the szSize (receive data buffer size) is smaller than the received data size, only the data equivalent to the size specified by szSize is stored in pData and the data exceeding the size specified by szSize is discarded.

11.4.8 NBS.ERROR (Error Code)

This is an enumeration type error code that is output when the function block for communication instruction that uses the LAN port is executed.

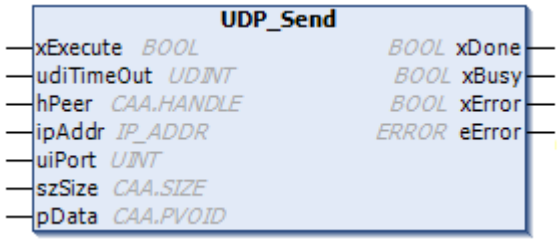
■ NBS.ERROR (Enumeration type)

Name	Value	Description
NO_ERROR	0	No error is occurring.
FIRST_ERROR	6000	Reserved
TIME_OUT	6001	Reserved
INVALID_ADDR	6002	IP address is invalid.
INVALID_HANDLE	6003	Handle is invalid.
INVALID_DATAPOINTER	6004	Data pointer is invalid.
INVALID_DATASIZE	6005	Data size is invalid.
UDP_RECEIVE_ERROR	6006	UDP datagram cannot be received.
UDP_SEND_ERROR	6007	UDP datagram cannot be sent.
UDP_SEND_NOT_COMPLETE	6008	Reserved
UDP_OPEN_ERROR	6009	Port cannot be opened.
UDP_CLOSE_ERROR	6010	Port cannot be released.
TCP_SEND_ERROR	6011	TCP message cannot be sent.
TCP_RECEIVE_ERROR	6012	TCP message cannot be received.
TCP_OPEN_ERROR	6013	TCP port cannot be created.
TCP_CONNECT_ERROR	6014	TCP connection cannot be established.
TCP_CLOSE_ERROR	6015	TCP port cannot be released.
TCP_SERVER_ERROR	6016	Reserved
WRONG_PARAMETER	6017	The parameter contains an invalid value.
ERROR_UNKNOWN	6018	Reserved
TCP_NO_CONNECTION	6019	There is no TCP connection.
IOCTL_ERROR	6020	Internal error (IOCTL is not supported.)
FIRST_MF	6050	Reserved
LAST_ERROR	6099	Reserved

11.4.9 NBS.UDP_Send (Send UDP Data)

This is a function block that sends data to the connection handle acquired by UDP_Peer.

■ Icon



■ Parameter

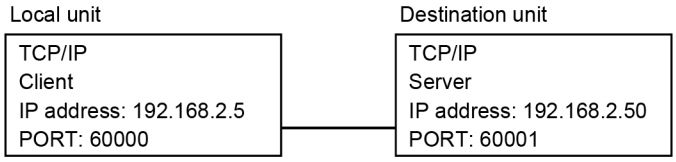
Scope	Name	Type	Description
Input	xExecute	BOOL	TRUE: Send started (edge) FALSE: Processing ended (edge)
	udiTimeOut	UDINT	Timeout (us)
	hPeer	CAA.HANDLE	Connection port handle acquired by UDP_Peer
	ipAddr	NBS.IP_ADDR	Destination IP address
	uiPort	UINT	Destination port No.
	szSize	CAA.SIZE	Send data size (byte)
	pData	CAA.PVOID	Pointer to the send data buffer.
Output	xDone	BOOL	TRUE: Processing is completed.
	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	NBS.ERROR	Connection result Refer to "11.4.8 NBS.ERROR (Error Code)".

11.4.10 Program example: General communication (Ethernet) TCP CLIENT processing

■ TCP CLIENT processing example

The following is a processing example of data transmission / reception via TCP when the local unit is TCP CLIENT.

This processing example assumes the following operating environment.



Processing for data transmission / reception

The processing for data transmission / reception is as follows:

- TCP client connection processing
- Reception start processing
- Transmission processing

Explanation of variables

Process

When the value is rewritten, the following processing is executed. After the execution is completed, the variable is set to 0 (invalid value).

1 = TCP client connection processing

2 = Reception start processing

3 = Transmission processing

Result

The result of processing execution is stored. (TRUE: Error occurrence, FALSE: Normal termination)

If the result of processing execution is abnormal, check the error code of each processing.

- NBS_ClientError: Result of TCP client connection processing
- NBS_WriteError: Result of transmission processing
- NBS_ReadError: Result of reception start processing

Operation example

The TCP client connects to the TCP server.

- The value of "Process" is changed to 1.

The local unit is ready to receive data. In this state, the local unit can receive data from the destination unit.

- The value of "Process" is changed to 2.

The local unit sends data to the destination unit. 10-byte data is sent to the destination unit.

- The value of "Process" is changed to 3.

Declaration section

```
PROGRAM TCP_Client
VAR
    Process         : UINT := 0;    // 1=TCP client connection , 2=Start receiving , 3=Send
    Result          : BOOL;         // Implementation result (FALSE=normal , TRUE=abnormal)

    ClientAddr : NBS.IP_ADDR := (sAddr:='192.168.2.50');    // Partner station IP address

    // FB Declaration
    NBS_Client  : NBS.TCP_Client;
    NBS_Write   : NBS.TCP_Write;
    NBS_Read    : NBS.TCP_Read;
    NBS_Handle  : NBS.CAA.HANDLE;

    NBS_ClientError : NBS.ERROR;
    NBS_WriteError  : NBS.ERROR;
    NBS_ReadError   : NBS.ERROR;

    ClientEnable    : BOOL := FALSE;
    TimeOut         : UDINT := 1000000;                 // Timeout 1second
    Port            : UINT := 60001;                    // Partner station port number

    // Transmission data
    SndEnable   : BOOL := FALSE;
    SendData    : ARRAY [1..10] OF BYTE := [1,2,3,4,5,6,7,8,9,10];

    RevEnable   : BOOL := FALSE;
    RecvBuf     : ARRAY [1..10] OF BYTE;                // Receive buffer
    RecvSize    : NBS.CAA.SIZE;                         // Receive size
    RecvCount   : UINT := 0;                            // Receive count
END_VAR
```

Implementation section (ST programming language)

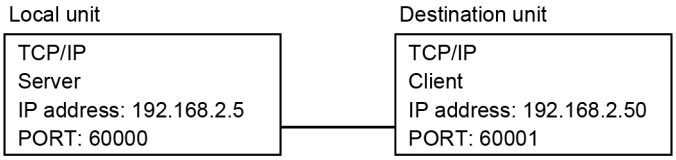
```
// TCP client connection check
IF ClientEnable = TRUE THEN
    NBS_Client();
    IF (NBS_Client.xBusy = TRUE)  AND (NBS_Client.xActive = TRUE) THEN
        // TCP client connection
        NBS_Handle := NBS_Client.hConnection;
    ELSIF NBS_Client.xDone = TRUE THEN
        // TCP client connection cutoff
        ClientEnable := FALSE;
        NBS_Client( xEnable := ClientEnable);
    ELSIF NBS_Client.xError = TRUE THEN
        // Error occurred
        NBS_ClientError := NBS_Client.eError;
        ClientEnable := FALSE;
        NBS_Client( xEnable := ClientEnable);
        Result := TRUE;
    END_IF
END_IF

// Reception enabled
IF RevEnable = TRUE THEN
    // Receipt confirmation
    NBS_Read();
    IF NBS_Read.xReady = TRUE THEN
        RecvCount := RecvCount + 1;         // Received number update
        RecvSize := NBS_Read.szCount;       // Receive size
    ELSIF NBS_Read.xError = TRUE THEN       // Error occurred
        NBS_ReadError := NBS_Read.eError;   // Error information storage
        RevEnable := FALSE;
        NBS_Read( xEnable := RevEnable );   // Stop receiving
        Result := TRUE;
    END_IF
END_IF

IF SndEnable = TRUE THEN
    // Transmission completion process
    NBS_Write();
    IF NBS_Write.xDone = TRUE THEN          // send completely
        NBS_Write( xExecute := FALSE );
        Process := 0;
    ELSIF NBS_Write.xError = TRUE THEN      // Error occurred
        NBS_WriteError := NBS_Write.eError; // Error information storage
        NBS_Write( xExecute := FALSE );     // Stop sending
        Result := TRUE;
        Process := 0;
    END_IF
END_IF

CASE Process OF
    1:  // TCP client connection
        Result := FALSE;
        ClientEnable := TRUE;

        NBS_Client( xEnable := ClientEnable ,   // Execute
            udiTimeOut := TimeOut ,             // Time-out setting
            ipAddr := ClientAddr ,              // Partner station IP address
            uiPort := Port );                   // Partner station port number
        Process := 0;

    2:  // Reception start processing
        Result := FALSE;
        RevEnable := TRUE;
        NBS_Read( xEnable := RevEnable ,        // Start receiving
                hConnection := NBS_Handle ,     // Connection handle
                pData := ADR(RecvBuf) ,         // Receive buffer
                szSize := SIZEOF(RecvBuf));     // Receive size
        Process := 0;

    3:  // Transmission process
        Result := FALSE;
        NBS_Write( xExecute := TRUE ,           // Start sending
                hConnection := NBS_Handle ,     // Connection handle
                pData := ADR(SendData) ,        // Send buffer
                szSize := SIZEOF(SendData));    // Send size
        SndEnable := TRUE;
        Process := 0;                           // Transmission completion wait processing
END_CASE
```

11.4.11 Program example: General communication (Ethernet) TCP SERVER processing

■ TCP SERVER processing example

The following is a processing example of data transmission / reception via TCP when the local unit is TCP SERVER.

This processing example assumes the following operating environment.



Processing for data transmission / reception

The processing for data transmission / reception is as follows:

- TCP server open processing
- TCP connection processing
- Reception start processing
- Transmission processing

Explanation of variables

Process

When the value is rewritten, the following processing is executed. After the execution is completed, the variable is set to 0 (invalid value).

1 = TCP server open processing

2 = TCP connection processing

3 = Reception start processing

4 = Transmission processing

Result

The result of processing execution is stored. (TRUE: Error occurrence, FALSE: Normal termination)

If the result of processing execution is abnormal, check the error code of each processing.

- NBS_ServError: Result of TCP server open processing
- NBS_ConErrorr: Result of TCP connection processing
- NBS_ReadError: Result of reception start processing
- NBS_WriteError: Result of transmission processing

Operation example

The TCP server is opened and connected to the TCP client.

- The value of "Process" is changed from 1 to 2.

The local unit is ready to receive data. In this state, the local unit can receive data from the destination unit.

- The value of "Process" is changed to 3.

The local unit sends data to the destination unit. 10-byte data is sent to the destination unit.

- The value of "Process" is changed to 4.

Declaration section

```
PROGRAM TCP_Server
VAR
    Process : UINT := 0;    // 1=server open , 2=connect , 3=Start receiving , 4=Send
    Result  : BOOL;         // Implementation result (FALSE=normal , TRUE=abnormal)

    MyAddr  : NBS.IP_ADDR := (sAddr:='192.168.2.5');    // Own station IP address
    MyPort  : UINT := 60000;                            // Own station port number

    // FB Declaration
    NBS_Server      : NBS.TCP_Server;
    NBS_Connection  : NBS.TCP_Connection;
    NBS_Read        : NBS.TCP_Read;
    NBS_Write       : NBS.TCP_Write;

    NBS_Handle      : NBS.CAA.HANDLE;
    NBS_ServError   : NBS.ERROR;
    NBS_ConError    : NBS.ERROR;
    NBS_ReadError   : NBS.ERROR;
    NBS_WriteError  : NBS.ERROR;

    // Transmission data
    SendData    : ARRAY [1..10] OF BYTE := [1,2,3,4,5,6,7,8,9,10];

    ServerEnable: BOOL := FALSE;            // Server port open process in progress
    ConEnable   : BOOL := FALSE;            // In process of connecting
    SndEnable   : BOOL := FALSE;            // Sending in progress
    RevEnable   : BOOL := FALSE;            // Receive processing
    RecvBuf     : ARRAY [0..10] OF BYTE;    // Receive buffer
    RecvCount   : UINT := 0;                // Number of receptions
    RecvSize    : NBS.CAA.SIZE;             // Received data size
    ClientAddr  : NBS.SysSocket.INADDR;     // IP address of the client
    sClientAddr : STRING;                   // Destination IP address
END_VAR
```

Implementation section (ST programming language)

```
// Server port open process
NBS_Server( xEnable:=ServerEnable , ipAddr:=MyAddr , uiPort := MyPort );
IF ServerEnable = TRUE THEN
    // Connection confirmation
    IF NBS_Server.xError = TRUE THEN            // Error occurred
        NBS_ServError := NBS_Server.eError;     // Error information storage
        ServerEnable := FALSE;
        Result := TRUE;
    END_IF
END_IF

// Waiting for connection completion
NBS_Connection( xEnable:=ConEnable , hServer:=NBS_Server.hServer );
IF NBS_Connection.xActive = TRUE THEN
    NBS_Handle := NBS_Connection.hConnection;   // Connection handle
    ClientAddr := NBS_Connection.IPAddress;     // Get the IP address of the connection destination
    sClientAddr := NBS.UDINT_TO_IPSTRING( udiIPAddress := ClientAddr.ulAddr );
                                                // IP address translation
ELSIF NBS_Connection.xError = TRUE THEN         // Error occurred
    NBS_ConError := NBS_Connection.eError;      // Error information storage
    ConEnable := FALSE;
    Result := TRUE;
END_IF

// Reception enabled
NBS_Read( xEnable:=RevEnable , hConnection:=NBS_Handle , pData := ADR(RecvBuf) , szSize:= SIZEOF(RecvBuf));
IF RevEnable = TRUE THEN
    // Receipt confirmation
    IF NBS_Read.xReady = TRUE THEN
        RecvCount := RecvCount + 1;         // Received number update
        RecvSize := NBS_Read.szCount;       // Receive size
    ELSIF NBS_Read.xError = TRUE THEN       // Error occurred
        NBS_ReadError := NBS_Read.eError;   // Error information storage
        RevEnable := FALSE;
        Result := TRUE;
    END_IF
END_IF

// Transmission completion process
NBS_Write( xExecute:=SndEnable , hConnection:=NBS_Handle , pData:=ADR(SendData) , szSize:=SIZEOF(SendData));
IF SndEnable = TRUE THEN
    IF NBS_Write.xDone = TRUE THEN          // send completely
        SndEnable := FALSE;
    ELSIF NBS_Write.xError = TRUE THEN      // Error occurred
        NBS_WriteError := NBS_Write.eError; // Error information storage
        SndEnable := FALSE;
        Result := TRUE;
    END_IF
END_IF

CASE Process OF
    1:  // TCP server open
        ServerEnable := TRUE;
        Result := FALSE;
        Process := 0;

    2:  // TCP connect
        ConEnable := TRUE;
        Result := FALSE;
        Process := 0;

    3:  // Reception start processing
        RevEnable := TRUE;
        Result := FALSE;
        Process := 0;

    4:  // Transmission process
        SndEnable := TRUE;
        Result := FALSE;
        Process := 0;

END_CASE
```

When multiple clients are connected simultaneously to the same port, multiple TCP_Connection instances are created. The hServer handle acquired by one TCP_Server is set to the multiple TCP_Connection instances.

Example: When two clients are connected simultaneously to the same port

Declaration section

```
iServer: NBS.TCP_Server;// TCP_Server instance
iConnection: ARRAY [0..1] OF NBS.TCP_Connection;    // TCP_Connection instance (two instances)
```

Implementation section

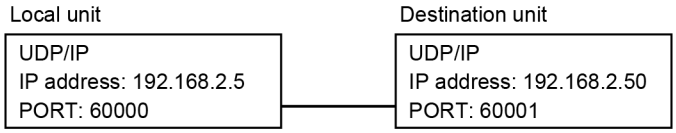
```
iServer( xEnable:=TRUE , ipAddr:=ipAddr , uiPort:=uiPort );    // Server opened
// Omitted (Waiting for TCP_Server completion)
iConnection[0]( xEnable := TRUE , hServer := iServer.hServer );    // For 1st client
iConnection[1]( xEnable := TRUE , hServer := iServer.hServer );    // For 2nd client
```

11.4.12 Program example: General communication (Ethernet) UDP processing

■ UDP processing example

An example of processing for data transmission / reception via UDP is as follows:

This processing example assumes the following operating environment.



Processing for data transmission / reception

The processing for data transmission / reception is as follows:

- Port open processing
- Reception start processing
- Transmission processing

Explanation of variables

Process

When the value is rewritten, the following processing is executed. After the execution is completed, the variable is set to 0 (invalid value).

1 = Port open processing

2 = Reception start processing

3 = Transmission processing

Result

The result of processing execution is stored. (TRUE: Error occurrence, FALSE: Normal termination)

If the result of processing execution is abnormal, check the error code of each processing.

- NBS_PeerError: Result of port open processing
- NBS_RecError: Result of reception start processing
- NBS_ReadError: Result of transmission processing

Operation example

The port is opened and the local unit is ready to receive data. In this state, the local unit can receive data from the destination unit.

- The value of "Process" is changed from 1 to 2.

The local unit sends data to the destination unit. 10-byte data is sent to the destination unit.

- The value of "Process" is changed to 3.

Declaration section

```
PROGRAM UDP
VAR
    Process     : UINT := 0;    // 1=Port open , 2=Receive start , 3=Send
    Result      : BOOL;         // Implementation result (FALSE=normal , TRUE=abnormal)

    MyipAddr : NBS.IP_ADDR := (sAddr:='192.168.2.5');   // Own station IP address
    SendAddr : NBS.IP_ADDR := (sAddr:='192.168.2.50');  // Partner station IP address
    MyPort   : UINT := 60000;                           // Own station PORT number
    SendPort : UINT := 60001;                           // Partner station PORT number

    // FB Declaration
    NBS_Peer    : NBS.UDP_Peer;
    NBS_Receive : NBS.UDP_Receive;
    NBS_Send    : NBS.UDP_Send;

    NBS_Handle      : NBS.CAA.HANDLE;       //  PORT handle
    NBS_PeerError   : NBS.ERROR;            // UDP_Peer Error information
    NBS_RecError    : NBS.ERROR;            // UDP_Receive Error information
    NBS_SendError   : NBS.ERROR;            // UDP_Send Error information

    // Transmission data
    SendData : ARRAY [1..10] OF BYTE := [1,2,3,4,5,6,7,8,9,10];

    PeerEnable  : BOOL := FALSE;
    SndEnable   : BOOL := FALSE;
    RevEnable   : BOOL := FALSE;

    RecvBuf : ARRAY [1..10] OF BYTE;    // Receive buffer
    RecvCount : UINT := 0;              // Receive count
    RecvPort : UINT;                    // Receive port
    RecvSize : NBS.CAA.SIZE;            // Receive size
    RecvIpAddr : NBS.IP_ADDR;           // Destination IP address
END_VAR
```

Implementation section (ST programming language)

```
// Port open processing
NBS_Peer( xEnable:=PeerEnable , ipAddr:=MyipAddr , uiPort:=MyPort );
IF PeerEnable = TRUE THEN
    IF NBS_Peer.xActive = TRUE THEN         // Successful port opening
        NBS_Handle := NBS_Peer.hPeer;       // Get handle
    ELSIF NBS_Peer.xError = TRUE THEN       // Error occurred
        NBS_PeerError := NBS_Peer.eError ;  // Error information storage
        PeerEnable := FALSE;
        Result := TRUE;
    END_IF
END_IF

// Reception processing
NBS_Receive( xEnable:=RevEnable , hPeer:=NBS_Handle , pData:=ADR(RecvBuf) ,
             szSize:=SIZEOF(RecvBuf));
IF RevEnable = TRUE THEN
    IF NBS_Receive.xReady = TRUE THEN       // Received data available
        RecvCount := RecvCount + 1;         // Received number update
        RecvPort := NBS_Receive.uiPortFrom; // Destination PORT
        RecvSize := NBS_Receive.szCount;    // Receive size
        RecvIpAddr := NBS_Receive.ipFrom;   // Destination IP address
    ELSIF NBS_Receive.xError = TRUE THEN    // Error occurred
        NBS_RecError := NBS_Receive.eError; // Error information storage
        RevEnable := FALSE;
        Result := TRUE;
    END_IF
END_IF

// Transmission process
NBS_Send( xExecute:=SndEnable , hPeer:=NBS_Handle , ipAddr:=SendAddr ,
          uiPort:=SendPort , pData:=ADR(SendData) , szSize:=SIZEOF(SendData));
IF SndEnable = TRUE THEN
    IF NBS_Send.xDone = TRUE THEN           // Successful transmission
        SndEnable := FALSE;                 // Transmission processing stopped
    ELSIF NBS_Send.xError = TRUE THEN       // Error occurred
        NBS_SendError := NBS_Send.eError;   // Error information storage
        SndEnable := FALSE;                 // Transmission processing stopped
        Result := TRUE;
    END_IF
END_IF

CASE Process OF
    1: // Port open processing
        PeerEnable := TRUE;
        Result := FALSE;
        Process := 0;

    2:  // Reception processing
        RevEnable := TRUE;
        Result := FALSE;
        Process := 0;

    3: // Transmission process
        SndEnable := TRUE;
        Result := FALSE;
        Process := 0;
END_CASE
```

11.4.13 Program example:General-purpose Communication(Serial)COM transmission / reception processing

■ COM transmission / reception processing example

Send and receive data via SerialCom.

Specify communication settings as below.

COM number	1
Baud rate	115200bps
Data bits	8
Parity bit	odd
Stop bit	1

Processing for data transmission / reception

The processing for data transmission / reception is as follows:

- Serial port open processing
- Serial port close processing
- Reception processing
- Transmission processing

Explanation of variables

Process

When the value is rewritten, the following processing is executed. After the execution is completed, the variable is set to 0 (invalid value).

1 = Serial port open processing

2 = Reception processing

3 = Transmission processing

4 = Serial port close processing

Result

The result of processing execution is stored. (TRUE: Error occurrence, FALSE: Normal termination)

If the result of processing execution is abnormal, check the following error code.

- ComErr: COM processing result

Operation example

Serial port is opened.

- The value of "Process" is changed to 1.

Received data is read.

- The value of "Process" is changed to 2.

10-byte data is sent.

- The value of "Process" is changed to 3.

Serial port is closed.

- The value of "Process" is changed to 4.

Declaration section

```
PROGRAM ST_PRG
VAR
    Process         : UINT := 0;    // 1=COM_OPEN 2=RECV 3=SEND  4=COM_CLOSE
    Result          : BOOL;         // Implementation result (FALSE=normal / TRUE=abnormal)

    ComOpen  : COM.Open;
    ComClose : COM.Close;
    ComSend  : COM.Write;
    ComRecv  : COM.Read;
    ComHandle: COM.CAA.HANDLE := 0;     // COM device handle
    ComErr   : COM.ERROR;               // COM error code

    // Communication parameters
    OpenParam : ARRAY [1..7] OF COM.PARAMETER := [
        (udiParameterId := COM.CAA_Parameter_Constants.udiPort,     udiValue := 1),
        (udiParameterId := COM.CAA_Parameter_Constants.udiBaudrate, udiValue := 115200),
        (udiParameterId := COM.CAA_Parameter_Constants.udiParity,   udiValue := INT_TO_UDINT(COM.PARITY.ODD)),
        (udiParameterId := COM.CAA_Parameter_Constants.udiStopBits, udiValue := INT_TO_UDINT(COM.STOPBIT.ONESTOPBIT)),
        (udiParameterId := COM.CAA_Parameter_Constants.udiTimeout,  udiValue := 0),
        (udiParameterId := COM.CAA_Parameter_Constants.udiByteSize, udiValue := 8),
        (udiParameterId := COM.CAA_Parameter_Constants.udiBinary,   udiValue := 0)
    ];

    OpenExe     : BOOL := FALSE;
    RecvExe     : BOOL := FALSE;
    SendExe     : BOOL := FALSE;
    CloseExe    : BOOL := FALSE;

    ReadBuf     : ARRAY [1..10] OF BYTE;        // Read buffer
    ReadSize    : UDINT;                        // Read data size

    SendBuf : ARRAY [1..10] OF BYTE := [1,2,3,4,5,6,7,8,9,10];  // Transmission data
END_VAR
```

Implementation section (ST programming language)

```
// Serial port open processing
IF Process = 1 THEN                         // Open processing started
    Result := FALSE;
    OpenExe := TRUE;
    Process := 101;
END_IF
ComOpen( xExecute:=OpenExe , pParameterList:=ADR(OpenParam) , usiListLength:=SIZEOF(OpenParam)/SIZEOF(COM.PARAMETER));
IF Process = 101 THEN                       // Opening process
    IF ComOpen.xDone = TRUE OR ComOpen.xError = TRUE THEN
        IF ComOpen.xError = FALSE THEN      // Opening completed
            ComHandle := ComOpen.hCom;      // Get COM handle
        ELSE                                // Error occurred
            ComErr := ComOpen.eError;       // Error information storage
            Result := TRUE;
        END_IF
        OpenExe := FALSE;                   // Stop open processing
        Process := 0;
    END_IF
END_IF

// Reception processing
IF Process = 2 THEN                         // Start receiving process
    Result := FALSE;
    RecvExe := TRUE;
    Process := 102;
END_IF
ComRecv( xExecute:=RecvExe , hCom:=ComHandle , szBuffer:=SIZEOF(ReadBuf) , pBuffer := ADR(ReadBuf));
IF Process = 102 THEN                       // Receive processing
    IF ComRecv.xDone = TRUE THEN            // Message received
        ReadSize := ComRecv.szSize;         // Get reception size
        RecvExe := FALSE;                   // Stop receiving
        Process := 0;
    ELSIF ComRecv.xError = TRUE THEN        // Error occurred
        ComErr := ComRecv.eError;           // Error information acquisition
        RecvExe := FALSE;                   // Stop receiving
        Result := TRUE;
        Process := 0;
    END_IF
END_IF

// Transmission process
IF Process = 3 THEN                         // Start transmission process
    Result := FALSE;
    SendExe := TRUE;
    Process := 103;
END_IF
ComSend( xExecute:=SendExe , hCom:=ComHandle , szSize:=SIZEOF(SendBuf) , pBuffer:=ADR(SendBuf));
IF Process = 103 THEN                       // Sending in progress
    IF ComSend.xDone = TRUE THEN            //  send completely
        SendExe := FALSE;                   // Stop sending
        Process := 0;
    ELSIF ComSend.xError = TRUE THEN        // Error occurred
        ComErr := ComSend.eError;           // Error information acquisition
        SendExe := FALSE;                   // Stop sending
        Result := TRUE;
        Process := 0;
    END_IF
END_IF

// Serial port closed
IF Process = 4 THEN                         // Serial port close processing started
    Result := FALSE;
    CloseExe := TRUE;
    Process := 104;
END_IF
ComClose( xExecute:=CloseExe , hCom:=ComHandle);
IF Process = 104 THEN                       // closing in progress
    IF ComClose.xDone = TRUE OR ComClose.xError = TRUE THEN
        IF ComClose.xError = FALSE THEN     // Close processing completed
            ComHandle := 0;
        ELSE                                // Error occurred
            ComErr := ComClose.eError;      // Error information storage
            Result := TRUE;
        END_IF
        CloseExe := FALSE;                  // Stop closing process
        Process := 0;
    END_IF

END_IF
```

11.5 LAN Port (Modbus TCP)

This section describes the library functions that are used to perform ModbusTCP communication with the LAN port.

It is created from Modbus master TCP available in the device tree.

11.5.1 IoDrvModbusTCP

This is a function block that controls the Modbus_TCP_Master device.

■ **Icon**

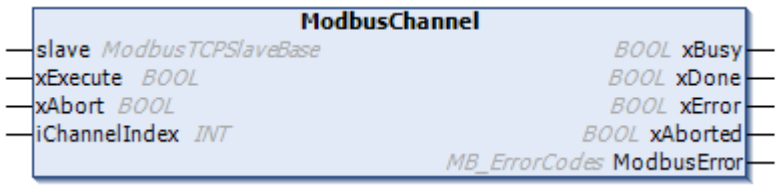
■ **Parameter**

Scope	Name	Type	Description
I/O	xStop	BOOL	TRUE: Stops sending commands to the slave.
Output	xSlaveError	BOOL	There is an error in the slave function
	uiConnectesSlaves	UINT	Number of slaves connected via TCP/IP

11.5.2 IoDrvModbusTCP.ModbusChannel (Start Sending Modbus Command)

This is a function block that sends the commands set in the Modbus Slave channel of the ModbusTCP_Slave device.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts sending commands at the rising edge.
	xAbort	BOOL	TRUE: Stops execution and resets all outputs.
	iChannelIndex	INT	Channel number where commands to be sent are set
I/O	slave	ModbusTCPSlaveBase	Handle of the Modbus_TCP_Slave device Output

Scope	Name	Type	Description
Output	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xDone	BOOL	TRUE: Processing is completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	xAborted	BOOL	TRUE: Execution is stopped by the user's xAbort input
	ModbusError	MB_ErrorCodes	An error code is output. Refer to "11.5.5 IoDrvModbus.MB_ErrorCodes (Error Codes)".

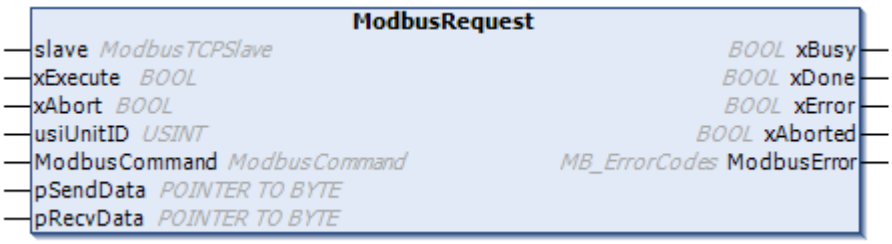
11.5.3 IoDrvModbusTCP.ModbusRequest (Modbus Request)

This is a function block that processes the Modbus command specified by I/O without using the Modbus_TCP_Slave device.

■ Supported commands

- Command 1 (Read multi-point coil state)
- Command 2 (Read multi-point input state)
- Command 3 (Read multi-point holding register)
- Command 4 (Read multi-point input register)
- Command 5 (Write single-point coil)
- Command 6 (Write single-point holding register)
- Command 15 (Write multi-point coil)
- Command 16 (Write multi-point holding register)
- Command 23 (Read / write multi-point holding register)

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	slave	ModbusTCP Slave	Handle of the Modbus_TCP_Slave device
	xExecute	BOOL	Starts sending commands at the rising edge.
	xAbort	BOOL	TRUE: Stops execution and resets all outputs.
	usiUnitID	USINT	Slave address 1 to 247
	ModbusCommand	ModbusCommnand	Structure that stores parameters of the commands issued.

Scope	Name	Type	Description
	pSendData	POINTER TO BYTE	Pointer to the send data buffer.
	pRecvData	POINTER TO BYTE	Pointer to the receive data buffer
Output	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xDone	BOOL	TRUE: Processing is completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	xAborted	BOOL	TRUE: Execution is stopped by the user's xAbort input
	ModbusError	BYTE	An error code is output. Refer to "11.5.5 IoDrvModbus.MB_ErrorCodes (Error Codes)". Also possible to convert the type and use as enumeration type MB_ErrorCodes.

■ ModbusCommand (Structure)

Name	Type	Description
uiFunctionCode	UINT	Modbus command code
uiReadOffset	UINT	Read address 0 to 65535
uiReadLen	UINT	Range in the number of read instances varies depending on commands.
uiWriteOffset	UINT	Write address 0 to 65535
uiWriteLen	UINT	Range in the number of write instances varies depending on commands.

11.5.4 IoDrvModbusTCPSlave

This is a function block that controls the Modbus_TCP_Slave device.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	xConfirmError	BOOL	Resets xError and byModbusError and resumes communication.
	xDoInit	BOOL	TRUE: Sends a slave initialization command when communication is resumed.
Output	xInitDone	UINT	TRUE: Modbus slave initialization command is fully completed.
	xDone	BOOL	TRUE: Processing is completed.
	xBusy	BOOL	TRUE: Processing of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	byModbusError	MB_ErrorCodes	An error code is output. Refer to "11.5.5 IoDrvModbus.MB_ErrorCodes (Error Codes)".
	ComSettings	ModbusTCPComSettings	IP address and port number registered in the Modbus_TCP_Slave device.
	ComState	ModbusTCPComState	Communication status
	iChannelIndex	INT	Channel number

11.5.5 IoDrvModbus.MB_ErrorCodes (Error Codes)

This is an enumeration type error code that is output when the function block for Modbus communication instruction that uses the COM port is executed.

■ IoDrvModbus.MB_ErrorCodes (Enumeration type)

Name	Value	Description
RESPONSE_SUCCESS	16#0	Succeeded
ILLEGAL_FUNCTION	16#1	Function code not supported by the slave
ILLEGAL_DATA_ADDRESS	16#2	Register offset not supported by the slave
ILLEGAL_DATA_VALUE	16#3	Illegal data writing
SLAVE_DEVICE_FAILURE	16#4	Non-recoverable error
ACKNOWLEDGE	16#5	Start operation
SLAVE_DEVICE_BUSY	16#6	During operation
MEMORY_PARITY_ERROR	16#8	Memory parity error
GATEWAY_PATH_UNAVAILABLE	16#A	Gateway path unavailable
GATEWAY_DEVICE_FAILED_TO_RESPOND	16#B	Gateway device failed to respond
RESPONSE_TIMEOUT	16#A1	Timeout
RESPONSE_CRC_FAIL	16#A2	CRC error
RESPONSE_WRONG_SLAVE	16#A3	Wrong response
RESPONSE_WRONG_FUNCTIONCODE	16#A4	Wrong function code in the response
REQUEST_FAILED_TO_SEND	16#A5	Request not sent
RESPONSE_INVALID_DATA	16#A6	Invalid response data
RESPONSE_INVALID_PROTOCOL	16#A7	Invalid response protocol
RESPONSE_INVALID_HEADER	16#A8	Invalid response header
UNDEFINED	16#FF	Undefined

11.6 LAN Port (EtherNet/IP)

This section describes the instructions that are used to control EtherNet/IP scanner and adapter functions using the SMC.

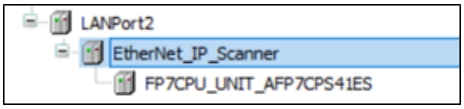
11.6.1 IoDrvEtherNetIP (EtherNet/IP Scanner Device)

This is a function block (FB) that controls the EtherNet/IP scanner device.

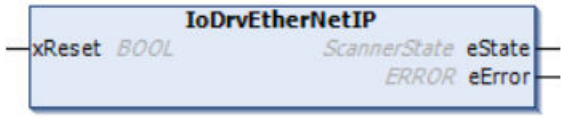
This function block is automatically generated by adding an EtherNet/IP scanner device and the name of the device that is added is used as the instance name.

Example

Adding an EtherNet/IP scanner device named "EtherNet_IP_Scanner" to LANPort2



■ **Icon**



■ **Parameter**

Scope	Name	Type	Description
Input	xReset	BOOL	Resets the scanner function at the rising edge
Output	eState	ScannerState	EtherNet/IP scanner device state
	eError	ERROR	Error state code of EtherNet/IP scanner

■ **ScannerState (EtherNet/IP scanner device state)**

Name	Description
INITIALIZING	The device is setting up a CIP object. It is continuing IP_CONFIG.
IP_CONFIG	The device creates an IP configuration for Ethernet interface and waits until it enters a RUNNING state.
UDP_CONFIG	The device opens the socket for UDP default port 2222.
ENCAPSULATION_CONFIG	The encapsulation server for the scanner is started via the default TCP port (44818).
ADAPTER_CONFIG	The device is in an empty state. It is continuing OPEN_CONNECTIONS.
OPEN_CONNECTIONS	The CIP ID status is set to "configured" and the RUNNING state continues.
RUNNING	The device opens a connection to the adapter and processes explicit messages with I/O communication.

Name	Description
DIAGNOSTIC_AVAILABLE	There are diagnostic messages from the configurator or editor.
BUS_ERROR	The UDP or TCP port failed to open.
RESET	xReset for the CIP ID object was received.
ERROR	When the network interface is in a continued state, the scanner enters the INITIALIZING state.

■ ERROR (Error state code of EtherNet/IP scanner)

Name	Description
NO_ERROR	No error is occurring.
INVALID_COMMAND	The command is invalid.
OUT_OF_MEMORY	A memory shortage occurred.
INVALID_DATA	The data is invalid.
INVALID_SESSION_HANDLE	The session handle is invalid.
INVALID_LENGTH	The data length is invalid.
UNSUPPORTED_PROTOCOL_VERSION	The protocol version is unsupported.
NBS_ERROR	An NBS error occurred.
NBS_RCV_ERROR	Data cannot be received via NBS.
NBS_SND_ERROR	Data cannot be sent via NBS.
ENCAPSULATION_ERROR	An encapsulation error occurred.
TCPIP_CONFIG_ERROR	TCP IP settings are incorrect.
UDP_CONFIG_ERROR	UDP settings are incorrect.
UDP_RECV_ERROR	UDP datagrams cannot be received.
UDP_SEND_ERROR	UDP datagrams cannot be sent.
UDP_CLOSE_ERROR	UDP ports cannot be released.
NULL_POINTER	This is a null pointer.
DEVICE_STATE_ERROR	An error is occurring on the device.
RECONFIGURATION_FAILED	Reconfiguration failed.
PERFORMANCE_MONITOR_DISABLED	The performance monitor is disabled.
INVALID_MEASURING_POINT	Measuring points are invalid.
IP_CONFIG_ERROR	IP settings are faulty.

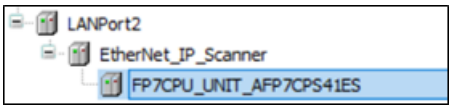
11.6.2 RemoteAdapter (Remote Adapter Device)

This is a function block (FB) for the remote adapter device linked to the EtherNet/IP scanner device.

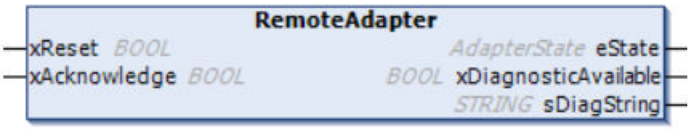
This function block is automatically generated by adding an EtherNet/IP remote adapter device and the name of the device that is added is used as the instance name.

Example

Adding a remote adapter device named "FP7CPU_UNIT_AFP7CPS41ES" to EtherNet_IP_Scanner



■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xReset	BOOL	Resets the remote adapter function at the rising edge
	xAcknowledge	BOOL	Acknowledges the diagnostic information at the rising edge
Output	eState	Adapter State	Remote adapter state
	xDiagnosticAvailable	BOOL	The output remains TRUE when there is diagnostic information
	sDiagString	STRING	Diagnosis string

■ AdapterState (Adapter device state)

Name	Description
DISABLED	The device is disabled in device tree
NOT_CONFIGURED	Parameters are being loaded
IP_CONFIG	The device has configured a TCP object and is waiting for an Ethernet node
ENCAPSULATION_CONFIG	Encapsulation is being configured
LIST_SERVICES	List services are being executed
REGISTER_SESSION	Register session is in progress
PARAMETER_CONFIG	Parameters are being configured
CONFIGURED	The device is in configuration completion state
RUNNING	The device is in running state
IDLE	The device is in idle state
RESET	UDP and TCP connection is closing
RESET_SERVICE	Reset service is being executed
CONNECTIVITY_CHECK	Connectivity check is in progress
BUS_ERROR	Bus error is occurring

Name	Description
ERROR	Error is occurring

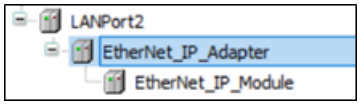
11.6.3 IoDrvEtherNetIPAdapter (EtherNet/IP adapter device)

This is a function block (FB) that controls the EtherNet/IP adapter device.

This function block is automatically generated by adding an EtherNet/IP adapter device and the name of the device that is added is used as the instance name.

Example

Adding an EtherNet/IP adapter device named "EtherNet_IP_Adapter" to LANPort2



■ Icon

■ Parameter

Scope	Name	Type	Description
Input	xReset	BOOL	Resets the adapter function at the rising edge
Output	eState	ADAPTERSTATE	EtherNet/IP adapter device state
	eError	ERROR	Error state code of EtherNet/IP adapter

■ ADAPTERSTATE (EtherNet/IP adapter device state)

Name	Description
UPDATE_CONFIGURATION	Startup phase
NOT_CONFIGURED	Parameters are being loaded
DISABLED	The device is disabled in device tree
CONFIGURED	A CIP object has been created
IP_CONFIG	The device has configured a TCP object and is waiting for an Ethernet node
IMPLICITMESSAGING_CONFIG	UDP port has been opened
EXPLICITMESSAGING_CONFIG	TCP port has been opened
NO_CONNECTION	The protocol stack has been started, but the scanner is unconnected.
RUNNING	The protocol stack is running, and the scanner is connected.

Name	Description
STOPPED	The Ethernet node is inactive, and the device is waiting for the Ethernet node to return.
RESET	UDP and TCP connection is closing.
SCANNER_EXTENSION	If the scanner registered this adapter as an I/O extension, the adapter is active in this state.
ERROR	Critical error
BUS_ERROR	Ethernet is not ready yet or is unavailable.

■ ERROR (EtherNet/IP adapter error state)

Name	Description
NO_ERROR	No error
TIME_OUT	Timeout
CONFIGURATION_FAILED	Failed to initialize resources, load connector parameters, or communicate with sub-connectors (modules)
IP_CONFIG_FAILED	The Ethernet node issued an error
IMPLICITMESSAGING_CONFIG_FAILED	Failed to create UDP port "CIP_ENC.ParameterList.gc_uiUDPPort" (default: 2222)
EXPLICITMESSAGING_CONFIG_FAILED	Failed to create TCP / UDP port "IP_ENC.ParameterList.gc_uiTCPPort" (default: 44818)
EXPLICITMESSAGE_RECEIVE_FAILED	Problem related to TCP or UDP port socket | CIP_ENC.ParameterList.gc_uiTCPPort | (default: 44818)
EXPLICITMESSAGE_SEND_FAILED	Problem related to TCP or UDP port socket | CIP_ENC.ParameterList.gc_uiTCPPort | (default: 44818)
LICENSE_MISSING	No license

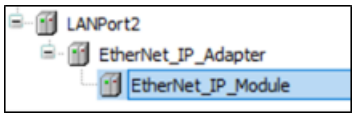
11.6.4 Module (EtherNet/IP Module Device)

This is a function block (FB) that controls the EtherNet/IP module device.

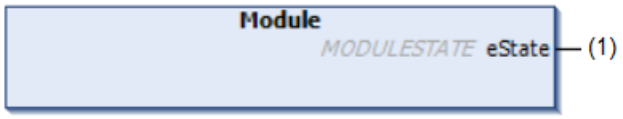
This function block is automatically generated by adding an EtherNet/IP module device and the name of the device that is added is used as the instance name.

Example

Adding an EtherNet/IP module device named "EtherNet_IP_Module" to EtherNet/IP adapter device



■ Icon



■ Parameter

Scope	Name	Type	Description
Output	eState	MODULESTATE	Module device state

■ MODULESTATE (EtherNet/IP module device state)

Name	Description
NOT_CONFIGURED	Parameters are being loaded.
CONFIGURED	A CIP object has been created.
NO_CONNECTION	The protocol stack has been started, but the scanner is unconnected.
RUNNING	The protocol stack is running, and the scanner is connected.
STOPPED	The Ethernet node is inactive, and the device is waiting for the Ethernet node to return.
DISABLED	The device is disabled in device tree.
ERROR	Critical error

11.6.5 Apply_Attributes (Apply_Attributes Service)

This is a function block (FB) that calls the "Apply_Attributes" service of the CIP object instance.

The attribute set in "Get_Attribut_Single" or "Get_Attribut_All" is adopted and saved in the adapter.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Execution flag
	itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP device that implements the EtherNet/IP service
	eClass	ENIP.CIPClass	Class that executes the service
	dwInstance	DWORD	Instance that executes the service (0: Class level, 1-x: Instance level)
Output	xDone	BOOL	Completion flag
	xBusy	BOOL	Busy flag
	xError	BOOL	Error flag
	eError	ENIP.ERROR	Error (0-255: CIP error, 256-x: Library error)

11.6.6 Generic_Service (Generic Service Execution)

This is a function block (FB) that executes generic services with the EtherNet/IP adapter.

Messages are sent as unconnected explicit message requests.

Note

- The endianness of data to be sent or received must be exchanged by devices.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Execution flag
	itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP device that implements the EtherNet/IP service
	eClass	ENIP.CIPClass	Class that executes the service
	dwInstance	DWORD	Instance that executes the service (0: Class level, 1-x: Instance level)
	wAttribute	WORD	Attribute corresponding to the service
	eService	ENIP.CIPCommonService	CIPCommonService member service code or vendor-specific service code
	pWriteData	POINTER TO BYTE	Pointer to data to be written to the EtherNet/IP adapter. The parameter is set to 0 when no data is sent.
	udiWriteDataSize	UDINT	Size of data to be written to the EtherNet/IP adapter. The parameter is set to 0 when no data is sent.
	pReadData	POINTER TO BYTE	Storage pointer to data received from the EtherNet/IP adapter. The parameter is set to 0 when no data is received.
	udiReadDataSize	UDINT	Size of storage buffer for data received from the EtherNet/IP adapter. The parameter is set to 0 when no data is received.
Output	xDone	BOOL	Completion flag
	xBusy	BOOL	Busy flag
	xError	BOOL	Error flag

Scope	Name	Type	Description
	eError	ENIP.ERROR	Error (0-255: CIP error, 256-x: Library error)
	udiReceivedDataSize	UDINT	Size of received data

■ ENIP.CIPCommonService (CIPCommonService member service code)

Name	Value
None	16#0
GET_ATTRIBUTES_ALL	16#1
SET_ATTRIBUTES_ALL	16#2
RESET	16#5
START	16#6
STOP	16#7
APPLY_ATTRIBUTES	16#D
GET_ATTRIBUTE_SINGLE	16#E
SET_ATTRIBUTE_SINGLE	16#10
NO_OPERATION	16#17

11.6.7 Get_Attribute_Single (Inquire Specific Attributes of a Specific Instance)

This is a function block (FB) that inquires specific attributes of a specific instance of the CIP object.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Execution flag
	itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP device that implements the EtherNet/IP service
	eClass	ENIP.CIPClass	Class that executes the service
	dwInstance	DWORD	Instance that executes the service (0: Class level, 1-x: Instance level)
	wAttribute	WORD	Attribute corresponding to the service
	pData	POINTER TO BYTE	Storage pointer to data received from the EtherNet/IP adapter
	udiDataSize	UDINT	Size of storage buffer for data received from the EtherNet/IP adapter
Output	xDone	BOOL	Completion flag
	xBusy	BOOL	Busy flag
	xError	BOOL	Error flag
	eError	ENIP.ERROR	Error (0-255: CIP error, 256-x: Library error)
	udiReceivedDataSize	UDINT	Size of received data

11.6.8 Get_Attributes_All (Inquire All Attributes of a Specific Instance)

This is a function block (FB) that inquires all attributes of a specific instance of the CIP object.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Execution flag
	itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP device that implements the EtherNet/IP service
	eClass	ENIP.CIPClass	Class that executes the service
	dwInstance	DWORD	Instance that executes the service (0: Class level, 1-x: Instance level)
	pData	POINTER TO BYTE	Storage pointer to data received from the EtherNet/IP adapter
	udiDataSize	UDINT	Size of storage buffer for data received from the EtherNet/IP adapter
Output	xDone	BOOL	Completion flag
	xBusy	BOOL	Busy flag
	xError	BOOL	Error flag
	eError	ENIP.ERROR	Error (0-255: CIP error, 256-x: Library error)
	udiReceivedDataSize	UDINT	Size of received data

11.6.9 Set_Attribute_Single (Set Specific Attributes of a Specific Instance)

This is a function block (FB) that sets specific attributes of a specific instance of the CIP object

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Execution flag
	itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP device that implements the EtherNet/IP service
	eClass	ENIP.CIPClass	Class that executes the service
	dwInstance	DWORD	Instance that executes the service (0: Class level, 1-x: Instance level)
	wAttribute	WORD	Attribute corresponding to the service
	pData	POINTER TO BYTE	Pointer to data to be written
	udiDataSize	UDINT	Size of data to be written
Output	xDone	BOOL	Completion flag
	xBusy	BOOL	Busy flag
	xError	BOOL	Error flag
	eError	ENIP.ERROR	Error (0-255: CIP error, 256-x: Library error)

11.6.10 Set_Attributes_All (Set All Attributes of a Specific Instance)

This is a function block (FB) that sets all attributes of a specific instance of the CIP object.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Execution flag
	itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP device that implements the EtherNet/IP service
	eClass	ENIP.CIPClass	Class that executes the service
	dwInstance	DWORD	Instance that executes the service (0: Class level, 1-x: Instance level)
	pData	POINTER TO BYTE	Pointer to data to be written
	udiDataSize	UDINT	Size of data to be written
Output	xDone	BOOL	Completion flag
	xBusy	BOOL	Busy flag
	xError	BOOL	Error flag
	eError	ENIP.ERROR	Error (0-255: CIP error, 256-x: Library error)

11.6.11 NOP (NOP Service)

This is a function block (FB) that executes the NOP service of a specific instance of the CIP object.

Normally, this service is used to check whether the adapter can still be used in the network.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Execution flag
	itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP device that implements the EtherNet/IP service
	eClass	ENIP.CIPClass	Class that executes the service
	dwInstance	DWORD	Instance that executes the service (0: Class level, 1-x: Instance level)
Output	xDone	BOOL	Completion flag
	xBusy	BOOL	Busy flag
	xError	BOOL	Error flag
	eError	ENIP.ERROR	Error (0-255: CIP error, 256-x: Library error)

11.6.12 Reset (Reset Service)

This is a function block (FB) that executes the Reset service of a specific instance of the CIP object.

The effects of this service differ according to the CIP object.

■ Icon

■ Parameter

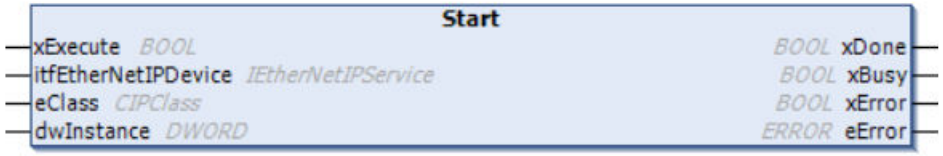
Scope	Name	Type	Description
Input	xExecute	BOOL	Execution flag
	itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP device that implements the EtherNet/IP service
	eClass	ENIP.CIPClass	Class that executes the service
	dwInstance	DWORD	Instance that executes the service (0: Class level, 1-x: Instance level)
Output	xDone	BOOL	Completion flag
	xBusy	BOOL	Busy flag
	xError	BOOL	Error flag
	eError	ENIP.ERROR	Error (0-255: CIP error, 256-x: Library error)

11.6.13 Start (Start Service)

This is a function block (FB) that executes the Start service of a specific instance of the CIP object.

The effects of this service differ according to the CIP object.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Execution flag
	itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP device that implements the EtherNet/IP service
	eClass	ENIP.CIPClass	Class that executes the service
	dwInstance	DWORD	Instance that executes the service (0: Class level, 1-x: Instance level)
Output	xDone	BOOL	Completion flag
	xBusy	BOOL	Busy flag
	xError	BOOL	Error flag
	eError	ENIP.ERROR	Error (0-255: CIP error, 256-x: Library error)

11.6.14 Stop (Stop Service)

This is a function block (FB) that executes the Stop service of a specific instance of the CIP object.

The effects of this service differ according to the CIP object.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Execution flag
	itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP device that implements the EtherNet/IP service
	eClass	ENIP.CIPClass	Class that executes the service
	dwInstance	DWORD	Instance that executes the service (0: Class level, 1-x: Instance level)
Output	xDone	BOOL	Completion flag
	xBusy	BOOL	Busy flag
	xError	BOOL	Error flag
	eError	ENIP.ERROR	Error (0-255: CIP error, 256-x: Library error)

11.6.15 ENIP.ERROR (Message Service Instruction Error Code)

Name	Value	Description
NO_ERROR	0	The service was executed normally by the specified object.
CONNECTION_FAILURE	16#1	The connection-related service failed due to the connection path.
RESOURCE_UNAVAILABLE	16#2	The object was unable to use the resources that it required to execute the requested service.
INVALID_PARAM_VALUE	16#3	Refer to status code 16#20 that is an appropriate value to be used in this situation.
PATH_SEGMENT_ERROR	16#4	The path segment identifier or segment syntax was not recognized by the processing node. Path processing stops when a path segment error occurs.
PATH_DESTINATION_UNKNOWN	16#5	The path refers to an object class, instance, or structure element that is unknown or not included in the processing node. If an unknown path destination error occurs, path processing will stop.
PARTIAL_TRANSFER	16#6	Only part of the expected data was transferred.
CONNECTION_LOST	16#7	The messaging connection was lost.
SERVICE_NOT_SUPPORTED	16#8	The requested service is not implemented or defined for this object class or instance.
INVALID_ATTRIBUTE_VALUE	16#9	Invalid attribute data was detected.
ATTRIBUTE_LIST_ERROR	16#A	The status of the attribute of Get_Attribute_List or Set_Attribute_List response is other than zero.
ALREADY_IN_REQUEST_STATE	16#B	The object is already in the mode or state requested by the service.
OBJECT_STATE_ERROR	16#C	The object cannot execute the requested service in the current mode or state.
OBJECT_ALREADY_EXISTS	16#D	An instance requested for the object to be created already exists.
ATTRIBUTE_NOT_SETTABLE	16#E	A request to change a read-only attribute was received.
PRIVILEGE_VIOLATION	16#F	An authority / privilege check failed.
DEVICE_STATE_ERROR	16#10	The current mode or state of the device prohibits the requested service from being executed.
REPLY_DATA_TOO_LARGE	16#11	The size of data to be sent via a response buffer is larger than the capacity of the allocated response buffer.
FRAGMENTATION_OF_VALUE	16#12	The service specifies an operation that fragmentates half of primitive data values which are a REAL data type.
NOT_ENOUGH_DATA	16#13	The service did not provide enough data to execute the specified operation.
ATTRIBUTE_NOT_SUPPORTED	16#14	The attribute specified in the request is not supported.
TOO_MUCH_DATA	16#15	The service provided more data than expected.
OBJECT_DOES_NOT_EXIST	16#16	The specified object does not exist in the device.
SERVICE_FRAGMENTATION_ SEQUENCE_NOT_IN_PROGRESS	16#17	The fragmentation sequence of this service is currently not active for this data.

Name	Value	Description
NO_STORED_ATTRIBUTE_DATA	16#18	The attribute data of this object has not been saved before the requested service is executed.
STORE_OPERATION_FAILURE	16#19	The attribute data of this object has not been saved because an error occurred during the attempt to save the data.
ROUTING_FAILURE_REQUEST_ PACKET_TOO_LARGE	16#1A	The service request packet was too large to send through the network existing in the path to the destination. The routing device forcibly canceled the service.
ROUTING_FAILURE_RESPONSE_ PACKET_TOO_LARGE	16#1B	The service response packet was too large to send through the network existing in the path from the destination. The routing device forcibly canceled the service.
MISSING_ATTRIBUTE_LIST_ ENTRY_DATA	16#1C	The service did not provide attributes in the list of attributes that it requires to execute the requested operation.
INVALID_ATTRIBUTE_VALUE_LIST	16#1D	The service returned a list of provided attributes together with status information of invalid attributes.
EMBEDDED_SERVICE_ERROR	16#1E	An error occurred in the embedded service.
VENDOR_SPECIFIC_ERROR	16#1F	A vendor-specific error occurred. The additional code field for error response is used to define a specific error that occurred. Use this field only if the error in question does not apply to any of the error codes shown in these tables or those shown in the object class definition.
INVALID_PARAMETER	16#20	The parameter associated with the request is invalid. This code is used when the parameter does not meet the requirements of this specification or the requirements defined in the application object specification.
WRITE_ONEC_VALUE_OR_ MEDIUM_ALREADY_WRITTEN	16#21	An attempt was made to write to a write-once medium (such as WORM drive or PROM) to which data has already been written or to change a value that cannot be changed once set.
INVALID_REPLY_RECEIVED	16#22	An invalid response was received (for example, the response service code does not match the request service code or the response message is shorter than the expected minimum response size). This status code is useful to investigate other causes of invalid responses.
BUFFER_OVERFLOW	16#23	The size of the received message exceeds the maximum size of messages that can be handled by the receiver buffer. The entire message was discarded.
MESSAGE_FORMAT_ERROR	16#24	The format of the received message is not supported by the server.
KEY_FAILURE_IN_PATH	16#25	The key segment included as the first segment of the path does not match the destination module. The object-specific status indicates which part of the key check has failed.
PATH_SIZE_INVALID	16#26	The size of the path sent with the service request is not large enough to route the request to the object or routing data included in the path is too much.
UNEXPECTED_ATTRIBUTE_IN_LIST	16#27	An attempt was made to set an attribute that cannot currently be set.
INVALID_MEMBER_ID	16#28	The member ID specified in the request does not exist in the specified class, instance, or attribute.

Name	Value	Description
MEMBER_NOT_SETTABLE	16#29	A request to change an unchangeable member was received.
GROUP_2_ONLY_SERVER_ GENERAL_FAILURE	16#2A	This error code is issued only by DeviceNet Group 2 Only servers with 4K or less code space and is supported only instead of the server. Attributes are not supported and cannot be set.
UNKNOWN_MODBUS_ERROR	16#2B	The program for conversion from CIP to Modbus received an unknown Modbus exception code.
ATTRIBUTE_NOT_GETTABLE	16#2C	A request to read an unreadable attribute was received.
INSTANCE_NOT_DELETABLE	16#2D	The requested object instance cannot be deleted.
SERVICE_NOT_SUPPORTED_ FOR_SPECIFIED_PATH	16#2E	The object supports the service but does not support the specified application path (such as attributes). Note: Do not use this code for the set service. (Instead, use general status code 16#0E or 16#29.)
TIME_OUT	16#100	The request has timed out.
INTERFACE_MISSING		IEtherNetIPService is not implemented.
REMOTE_CALL_FAILED		There is no physical connection.
NULL_POINTER		A null value was entered by mistake.
INVALID_DATA_SIZE		The data size is invalid.
WRONG_INTERFACE_VERSION		The versions do not match. The device is not equipped with the same version of interface as the called method.
NO_MEMORY		There is not enough memory.
UNKNOWN_ERROR		An unknown error occurred.
ABORTED		The service was aborted.

11.6.16 ENIP.CIPClass (Service Class Code)

Name	Value
IdentityObject	16#1
MessageRouterObject	16#2
DeviceNetObject	16#3
AssemblyObject	16#4
ConnectionObject	16#5
ConnectionManagerObject	16#6
RegisterObject	16#7
DiscreteInputPointObject	16#8
DiscreteOutputPointObject	16#9
AnalogInputPointObject	16#A
AnalogOutputPointObject	16#B
PresenceSensingObject	16#E
ParameterObject	16#F
ParameterGroupObject	16#10
GroupObject	16#12
DiscreteInputGroupObject	16#1D
DiscreteOutputGroupObject	16#1E
DiscreteGroupObject	16#1F
AnalogInputGroupObject	16#20
AnalogOutputGroupObject	16#21
AnalogGroupObject	16#22
PositionSensorObject	16#23
PositionControllerSupervisorObject	16#24
PositionControllerObject	16#25
BlockSequencerObject	16#26
CommandBlockObject	16#27
MotorDataObject	16#28
ControlSupervisorObject	16#29
ACDCDriveObject	16#2A
AcknowledgeHandlerObject	16#2B
OverloadObject	16#2C
SoftstartObject	16#2D
SelectionObject	16#2E
S_DeviceSupervisorObject	16#30
S_AnalogSensorObject	16#31

Name	Value
S_AnalogActuatorObject	16#32
S_SingleStageControllerObject	16#33
S_GasCalibrationObject	16#34
TripPointObject	16#35
FileObject	16#37
S_PartialPressureObject	16#38
SafetySupervisorObject	16#39
SafetyValidatorObject	16#3A
SafetyDiscreteOutputPointObject	16#3B
SafetyDiscreteOutputGroupObject	16#3C
SafetyDiscreteInputPointObject	16#3D
SafetyDiscreteInputGroupObject	16#3E
SafetyDualChannelOutputObject	16#3F
S_SensorCalibrationObject	16#40
EventLogObject	16#41
MotionDeviceAxisObject	16#42
TimeSyncObject	16#43
ModbusObject	16#44
OriginatorConnectionListObject	16#45
ModbusSerialLinkObject	16#46
DeviceLevelRingObject	16#47
QoSObject	16#48
SafetyAnalogInputPointObject	16#49
SafetyAnalogInputGroupObject	16#4A
SafetyDualChannelAnalogInputObject	16#4B
SERCOSIIILinkObject	16#4C
TargetConnectionListObject	16#4D
EnergyObject	16#4E
ElectricalEnergyObject	16#4F
Non_ElectricalEnergyObject	16#50
BaseSwitchObject	16#51
SNMPObject	16#52
PowerManagementObject	16#53
ControlNetObject	16#F0
ControlNetKeeperObject	16#F1
ControlNetSchedulingObject	16#F2
ConnectionConfigurationObject	16#F3

Name	Value
PortObject	16#F4
TCPIPInterfaceObject	16#F5
EthernetLinkObject	16#F6
CompoNetLink	16#F7
CompoNetRepeater	16#F8

11.7 LAN Port (MQTT)

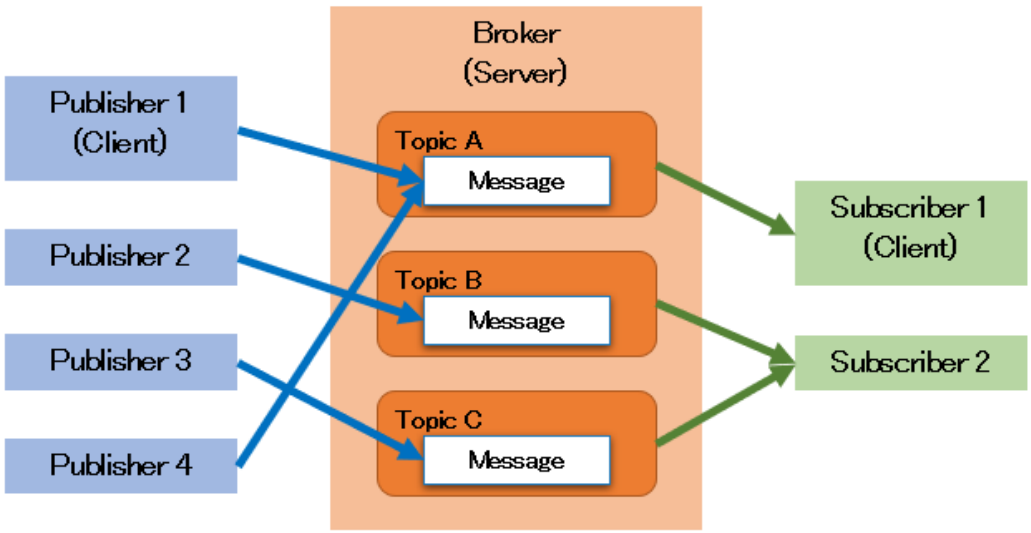
This section describes the instructions that are used to perform communication with the LAN port using the MQTT protocol.

11.7.1 What is MQTT?

MQTT stands for Message Queuing Telemetry Transport. It is a simple and lightweight publish/subscribe messaging protocol.

This protocol allows asynchronous many-to-many communication by a mechanism called “topic” designed to identify messages. Messages are sent and received through an intermediary called a broker server, and thus MQTT enables a device to communicate with another device on the opposite side without being conscious of the opposite device. In addition, since the intermediary is responsible for most of message management, the number of connected client devices can be readily increased. Another feature is that client devices can be freely connected to and disconnected from the broker server. Generally, the amount of information necessary for exchange is small, and this helps to reduce the burden on CPUs and power consumption.

Because of these features, MQTT is widely adopted for IoT applications and is a protocol effective in a system for control among a large number of devices, as well as data logging, traceability, and other communication with a host system.



11.7.2 MQTT Client Specifications

MQTT client specifications with the GM1 controller are described below.

- MQTT Client Specifications

Item	Details
Usable port	LAN ports 1, 2
MQTT protocol version	Version 3.1.1 Version 5.0
Data size	Max. 6000 bytes per packet（payload part Max. 4096 Bytes）(Note 1)
Topic	Topic name：Max.1024 characters Topic level：Max.10
Communication constraints	Max. 20 publishes/subscribes per connection Max. 3 connections
Supported QoS	● QoS0 (publish at most once) ● QoS1 (publish at least once) ● QoS2 (publish exactly once)

(Note 1) This applies to the MQTT protocol packet size out of the total packet.

- List of supported functions

Type	Function	Overview	MQTT Version	
			3.1.1	5.0
Connection	KeepAlive	Specifies an interval during which connection closing is judged	○	○
	Will Message	Specifies a message that is sent when connection is closed	○	○
	Will Retain	Specifies if the Will Message is to be retained	○	○
	Will QoS	Specifies the QoS level for the Will Message	○	○
	Clean Session	Specifies a session used for connection with the broker server	○	○
	User authentication	Connection using a user name and password	○	○
	Client ID	Specifies a client identifier	○	○
	Ping Interval	Specifies an interval at which a ping request (existance check) is sent	○	○
	TLS connection	Connection encrypted by TLS	×	×
	WebSocket connection	Connection using WebSocket	×	×
	Reason Code	An output value of the detailed result of an operation	×	○
	Session Expiry Interval	Specifies how long to retain the session after a disconnect	×	○
	Enhanced authentication	Using other forms of authentication	×	×

Type	Function	Overview	MQTT Version	
			3.1.1	5.0
	Request Problem Information	Specifies the way an operation result is received	×	○
	Request Response Information	Requests the server to return Response Information (runs on request/response format)	×	×
	Receive Maximum	Specifies the number of messages that the client can process concurrently	×	○
	Topic Alias Maximum	Specifies the number of Topic Aliases that the client can receive	×	×
	Maximum Packet Size	Specifies a Maximum Packet Size value	×	○
	Payload Format Indicator	Specifies a format for the Will Message	×	○
	Message Expiry Interval	Specifies an interval for the expiry of the message	×	○
	Content Type	Specifies a type of the content of the Will Message	×	×
	Response Topic	The topic name for a response message (runs on request/response format)	×	○
	Correlation Data	Specifies correlation data (runs on request/response format)	×	○
	Will Delay Interval	Specifies a delay that occurs before the Will Message is sent	×	○
	User Property	User-defined properties	×	○
Publish	Re Delivery	Specifies the re-delivery flag (DUP Flag)	○	○
	Retain	Specifies a message store setting	○	○
	Payload Format Indicator	Specifies a format for the message	×	○
	Message Expiry Interval	Specifies an interval for the expiry of the message	×	○
	Content Type	Specifies a type of the content of the message	×	×
	Response Topic	The topic name for a response message (runs on request/response format)	×	○
	Correlation Data	Specifies correlation data (runs on request/response format)	×	○
	Subscription ID	The identifier of the subscription (for the broker)	×	×
	Topic Alias	Specifies a Topic Alias value	×	○
	User Property	User-defined properties	×	○
Subscribe	Subscription ID	Specifies the identifier of the subscription	×	○
	Correlation Data	Reception of correlation data (runs on request/response format)	×	○
	No Local Option	The setting of reception of messages from the same client	×	○
	Retain As Published	The setting of the Retain flag in a forwarded message	×	×

Type	Function	Overview	MQTT Version	
			3.1.1	5.0
	RetainHandling	Setting of whether or not to receive retained messages at the time of subscribing	×	○
	User Property	User-defined properties	×	○

11.7.3 Overview of MQTT Functions

Main MQTT functions will be introduced.

The GM1 controller supports MQTT protocol versions 3.1.1 and 5.0 and allows you to readily implement message publishing and subscribing using function blocks.

Key MQTT functions are described below.

- Topic

  Topic refers to a string used to filter messages. The publisher creates or specifies a topic related to a message and sends the message. The subscriber(s) specifies a topic it wants to receive and receives messages. Topics can be specified like the path of a file using the forward slash (/) as a delimiter. Sections separated by a forward slash are topic levels.

  Factory/Area_A/Room1

  Factory/Area_A/Room2

  Factory/Area_B/Room1

  A client can use two types of wildcards "+" and "#" to subscribe to topics.

  "+" is used to specify any name for a specific topic level. If the "Factory/+/Room1" topic filter is used for the three topics above, the subscriber can receive messages related to the "Factory/Area_A/Room1" and "Factory/Area_B/Room1" topics.

  "#" is used to specify any topic levels following a specific topic level pattern. If the "Factory/#" topic filter is used for the three topics above, the subscriber can receive messages related to all the topics.

- QoS（Quality of Service）

  QoS in MQTT refers to the quality of service when exchanging messages with the broker server. MQTT defines three levels of QoS: 0 to 2, which can be set individually for each subscription and for each broker server. Please refer to "11.7.4 MQTT.MQTTClient (MQTT Client Connection)" for the characteristics of each QoS level.

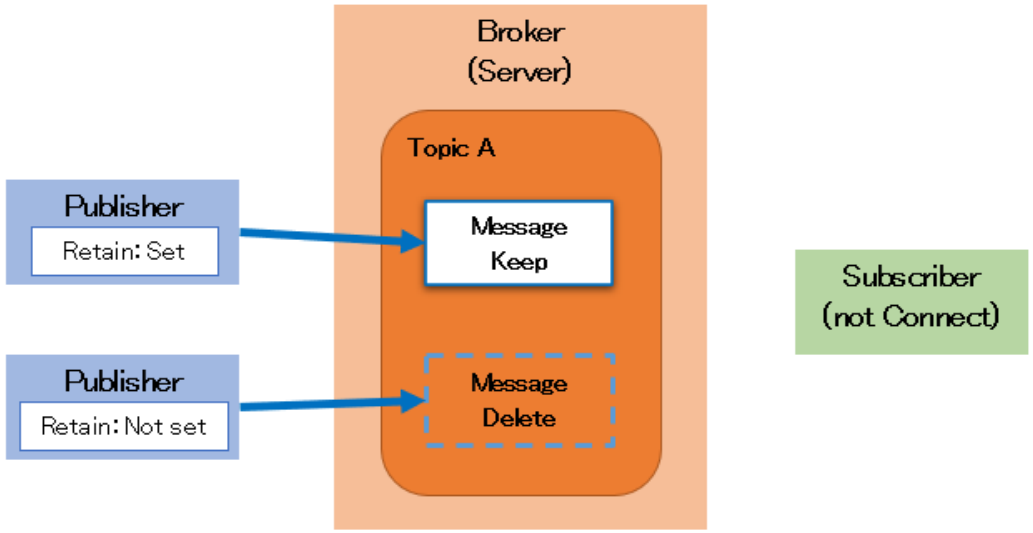
  Note that if the QoS levels differ between the publishing client and the subscribing client, the QoS level at which the subscribing client receives messages is less than or equal to the QoS level used by the publishing client.

QoS settings		Actual operation at subscribing client
Publishing client	Subscribing client	
QoS0	QoS0	QoS0
	QoS1	QoS0
	QoS2	QoS0
QoS1	QoS0	QoS0
	QoS1	QoS1
	QoS2	QoS1
QoS2	QoS0	QoS0
	QoS1	QoS1
	QoS2	QoS2

- Retain function

  The Retain function is a function used to allow the broker server to keep the last message per topic. A basic MQTT operation is to forward a message to subscribers that are connected when the message is sent. Thus, you cannot receive messages sent before you

subscribe. If a publisher sends a message with the Retain flag set to true, subscribers can receive the message later when they subscribe.



- Will message

  The Will message is a message that is forwarded when a client (irrespective of the publishing client or subscribing client) is unexpectedly disconnected due to an error in the network or power supply. Each client can register its Will message in advance when it connects to the broker server, and in the event of an abnormal disconnection, other client (the subscribing client) can be notified about the disconnection. If the client disconnects normally, the Will message is not forwarded.

  Like normal messages, the Will message is used with a topic and QoS level specified. The Will message can also be used in combination with other functions such as the Retain function.

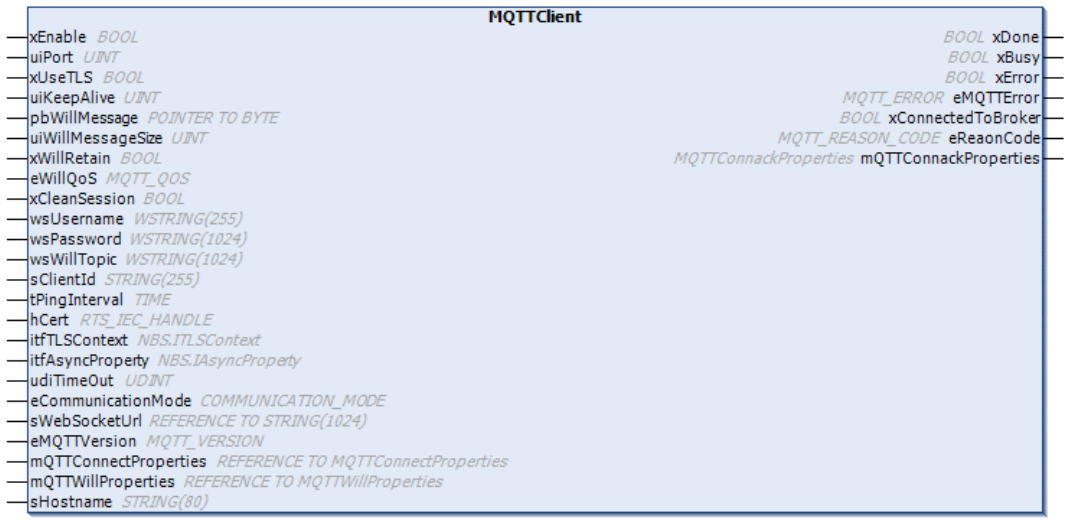
- Property

  Properties refer to optional features available in MQTT protocol version 5.0. While setting properties is not mandatory, it allows the use of various convenient options such as topic aliases and the delay in forwarding Will messages. By setting limits on properties, such as the number of messages that can be processed simultaneously or the maximum packet size that can be received, it is possible to retrieve the limitations of the connecting party during the broker server and client connection. For details on property settings, please refer to the descriptions of each function block.

11.7.4 MQTT.MQTTClient (MQTT Client Connection)

This is a function block used to connect to an MQTT broker server.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	xEnable	BOOL	FALSE	TRUE: Executes the FB. (connection) FALSE: Stops the FB. (disconnection)
	uiPort	UINT	0	Broker server port No.
	xUseTLS	BOOL	FALSE	Setting for using encryption in communication with the broker server TRUE: Encrypts communication with the broker server.
	uiKeepAlive	UINT	5	Keep alive time (unit: s)(Note 1)
	pbWillMessage	POINTER TO BYTE	-	An address to the Will message
	uiWillMessageSize	UINT	0	Size of the Will message (unit: byte) (Note 2)
	xWillRetain	BOOL	FALSE	Will message Retain (keep message on sever) setting TRUE: Enables Will message Retain (keep message on sever) function.
	eWillQoS	MQTT_QOS	MQTT_QOS.QoS0	Specifies the QoS level for the Will Message.

Scope	Name	Type	Default value	Description
	xCleanSession	BOOL	FALSE	Specifies a session used for connection with the broker server. TRUE: Creates a new session. FALSE: Use the existing session.
	wsUsername	WSTRING(255)	""	User name(Note 3)
	wsPassword	WSTRING(1024)	""	Password(Note 3)
	wsWillTopic	WSTRING(1024)	""	Will message topic name
	sClientId	STRING(255)	"	Client ID(Note 4) If not specified, an ID is automatically created. The created ID varies with the MQTT protocol version. ● Version 3.1.1: An ID is created using 23 random half-width alphabetic characters and numbers. ● Version 5.0: An ID is created by the broker server. For details, refer to your broker server specifications.
	tPingInterval	TIME	TIME#2s0ms	Interval at which a ping is transmitted (unit: ms)(Note 5)
	hCert	RTS_IEC_HANDLE	RTS_INVALID_HANDLE	Client certificate handle Do not use.
	itfTLSContext	NBS.ITLSContext	0	TLS connection context Do not use.
	itfAsyncProperty	NBS.IasyncProperty	0	Background task property (for connection process) Do not change form the default value.
	udiTimeOut	UDINT	0	Time to judge that connection process times out (unit: µs)
	eCommunicationMode	COMMUNICATION_MODE	COMMUNICATION_MODE.TCP	Setting of communication mode to be used Do not change form the default value.
	sWebSocketUrl	REFERENCE TO STRING(1024)	-	WebSocket server URI Do not use.
	eMQTTVersion	MQTT_VERSION	MQTT_VERSION.V3_1_1	MQTT protocol version
	MQTTConnectProperties	REFERENCE TO MQTTConnectProperties	-	Optional data used for connection with the broker server (valid only for protocol version 5.0)
	mQTTWillProperties	REFERENCE TO MQTTWillProperties	-	Optional data used for sending Will message (valid only for protocol version 5.0)

Scope	Name	Type	Default value	Description
Input / output	sHostname	STRING(80)	''	Broker server host name (IP address)
Output	xDone	BOOL	FALSE	TRUE: Connection is completed. (Note 6)
	xBusy	BOOL	FALSE	TRUE: Function block is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred in the function block.
	eMQTTError	MQTT_ERROR	MQTT_ERROR.NO_ERROR	Refer to output error codes in"11.7.8 MQTT.MQTT_ERROR (Error Code)".
	xConnectedToBroker	BOOL	FALSE	Indicates the state of connection with the broker server. TRUE: Connected with the broker server FALSE: Not connected with the broker server
	eReaonCode	MQTT_REASON_CODE	-	Refer to output reason codes (valid only for protocol version 5.0) in"11.7.7 MQTT.MQTT_REASON_CODE (Reason Code)".
	mQTTConnackProperties	REFERENCE TO MQTTConnackProperties	-	Response data from the the broker server (valid only for protocol version 5.0)

(Note 1) Do not specify 0.

(Note 2) For the maximum message size that can be sent, refer to"11.7.2 MQTT Client Specifications".

(Note 3) Configure these settings if the broker server provides user name/ password settings.

(Note 4) The ID must be up to 23 characters.

(Note 5) If “0” is specified, no ping is sent. Set the ping transmission interval within the time specified for uiKeepAlive.

(Note 6) The parameter turns TRUE at the completion of connection and remains only for one cycle.

■ COMMUNICATION_MODE (Enumeration type)

Name	Value	Description
TCP	0	Standard TCP/IP communication
WEB_SOCKET	1	TCP/IP communication via WebSocket (MQTT over WebSocket)

■ MQTT_QOS (Enumeration type)

Name	Value	Description
QoS0	0	A message is sent only once. Even if the message has not reached the recipient, a message is not retransmitted.
QoS1	1	A message will be sent at least once. It is possible for a message to be sent or delivered multiple times

Name	Value	Description
		because the sender retransmits the message until it gets an acknowledgment from the receiver.
QoS2	2	A message will be delivered exactly once to the receiver. It takes more time to send a message at this level compared to the QoS1 level, but duplication of the message is less likely to occur.

■ MQTT_VERSION (Enumeration type)

Name	Value	Description
V3_1_1	0	Protocol version 3.1.1
V5	1	Protocol version 5.0

■ MQTTConnectProperties (Structure)

Optional data settings for connection with the MQTT broker server can be configured. This is valid only for the protocol version 5.0.

Name	Type	Description
udiSessionExpiryInterval	UDINT	Sets a session expiry interval after disconnection in seconds. 1 to 4,294,967,294: expiry interval (unit: s) 0: immediately discarded, 16#FFFFFFFF: stored permanently
wsAuthenticationMethod	WSTRING	Authentication method name Do not specify any value.
pbAuthenticationData	POINTER TO BYTE	Authentication data Do not specify any value.
udiAuthenticationDataSize	UDINT	Authentication data size Do not specify any value.
bRequestProblemInformation	BYTE	Reason code reception setting Use this setting with the value fixed to 1.
bRequestResponseInformation	BYTE	Response information Do not specify any value.
uiReceiveMaximum	UINT	The number of QoS1 and QoS2 messages that the client can process concurrently
uiTopicAliasMaximum	UINT	The number of Topic Aliases that the client can accept Do not specify any value.
udiMaximumPacketSize	UDINT	The maximum packet size the client can accept(Note 1) 1 to 4,294,967,295: packet size, 0: no limit
userProperties	ARRAY [0..9] OF MQTTStringPair	User-defined properties

(Note 1) This applies to the MQTT protocol packet size out of the total received packet.

■ MQTTWillProperties (Structure)

Optional data settings for sending the Will message can be configured. This is valid only for the protocol version 5.0.

Name	Type	Description
bPayloadFormatIndicator	BYTE	Payload format Use this setting with the value fixed to 1.
udiMessageExpiryInterval	UDINT	Interval for which the broker server keeps the Will message(Note 1) 1 to 4,294,967,295: keeping time (unit: s) 0, not specified: kept permanently
wsContentType	WSTRING	Payload content type Do not specify any value.
wsResponseTopic	WSTRING	Topic name for a response message(Note 2)
udiCorrelationDataSize	UDINT	Correlation data size (unit: byte)(Note 2)
paCorrelationData	POINTER TO BYTE	Address to correlation data(Note 2)
udiWillDelayInterval	UDINT	A delay in publishing the Will message 1 to 4,294,967,295: delay (unit: s) 0, not specified: immediately published
userProperties	ARRAY [0..9] OF MQTTStringPair	User-defined properties

(Note 1) To use this, enable the Retain function.

(Note 2) If you want to implement request/response type communication using MQTT, you can use it. The maximum size of the correlation data that can be set is 256 bytes.

■ MQTTStringPair (Structure)

This is used when setting user-specific properties. It is effective in protocol version 5.0.

Name	Type	Description
wsKey	WSTRING	Property name
wsValue	WSTRING	Value(Note 1)

(Note 1) If you use it, you need to set it together with wsKey.

■ MQTTConnackProperties (Structure)

This is the option data of the broker server. The value is stored when the connection with the broker server is completed. Since it contains the constraints of the broker server, it can be used to determine the validity of the input values of MQTTPublish/MQTTSubscribe. It is effective in protocol version 5.0.

Name	Type	Description
udiSessionExpiryInterval	UDINT	A session expiry interval after disconnection 1 to 4,294,967,294: expiry interval (unit: s) 0: immediately discarded, 16#FFFFFFFF: stored permanently
wsAssignedClientIdentifier	WSTRING	Client ID issued by the broker server

Name	Type	Description
		Issued when sClientId is not set.
uiServerKeepAlive	UINT	Keep alive time assigned by the broker server (unit: s)(Note 1)
xAuthPacketReceived	BOOL	Property name
wsAuthenticationMethod	WSTRING	Authentication method name
bAuthenticationData	ARRAY [0..256] OF BYTE	Data used for authentication
wsResponseInformation	WSTRING	Data for the Request/Response function
wsReasonString	WSTRING	Refer to reason codes in"11.7.7 MQTT.MQTT_REASON_CODE (Reason Code)".
uiReceiveMaximum	UINT	The number of QoS1 and QoS2 messages that the broker server can process concurrently
uiTopicAliasMaximum	UINT	The number of Topic Aliases that the broker server can accept
bMaximumQoS	BYTE	QoS level the broker server can use 0: only QoS0, 1: up to QoS1, 2, 255: all QoS
bRetainAvailable	BYTE	Retain function availability setting 0: Retain not permitted, 1, 255: Retain permitted
udiMaximumPacketSize	UDINT	The maximum packet size the broker server can accept 1 to 4,294,967,295: packet size, 0: no limit
bWildcardSubscriptionAvailable	BYTE	Setting of the availability of wildcards in topic name 0: wildcards not permitted, 1, 255: wildcards permitted
bSubscriptionIdentifierAvailable	BYTE	Subscription ID availability setting 0: subscription ID not permitted, 1, 255: subscription ID permitted
bSharedSubscriptionAvailable	BYTE	Shared subscription availability setting 0: shared subscriptions not permitted, 1, 255: shared subscriptions permitted
userProperties	ARRAY [0..9] OF MQTTStringPair	User-defined properties

(Note 1) If the uiServerKeepAlive value is specified, the client overwrites the uiKeepAlive setting and operates. At the same time, the ping transmission interval changes to half the uiServerKeepAlive value.

Info.

- If an error occurs, the output of eMQTTError will remain even if xEnable is set to FALSE. After removing the cause of the error, executing the function block again will reset eMQTTError to NO_ERROR.

11.7.5 MQTT.MQTTPublish (MQTT Publish Function)

This is a function block for sending messages to the MQTT broker server. To use it, you must first execute MQTTClient and establish a connection with the broker server.

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	xExecute	BOOL	FALSE	At rising edge: Transmission of a message
	udiTimeOut	UDINT	1000000	Timeout time for transmission process (unit: μs) 0: Do not specify.
	eQoS	MQTT_QOS	MQTT_QOS.QoS0	Specifies QoS level between the MQTTPublish and broker server. Refer to"11.7.4 MQTT.MQTTClient (MQTT Client Connection)".
	xReDelivery	BOOL	FALSE	Specifies the re-delivery flag (DUP Flag).(Note 1) TRUE: Flag ON FALSE: Flag OFF
	xRetain	BOOL	FALSE	Message Retain (keep message on sever) setting TRUE: Retain enabled FALSE: Retain disabled
	pbPayload	POINTER TO BYTE	-	An address to the sent message(Note 2)
	udiPayloadSize	UDINT	0	Size of the sent message (unit: byte)(Note 2)
	mQTTPublishProperties	REFERENCE TO MQTTPublishProperties	-	Optional data used for sending messages (valid only for MQTT protocol version 5.0)
Input / output	mqttClient	MQTTClient	-	Reference to MQTTClient sending the message
	wsTopicName	WSTRING(1024)	""	Topic name of the message to be sent

Scope	Name	Type	Default value	Description
Output	xDone	BOOL	FALSE	TRUE: Transmission is completed. (Note 3)
	xBusy	BOOL	FALSE	TRUE: Function block is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred in the function block.
	eMQTTError	MQTT_ERROR	MQTT_ERROR. NO_ERROR	Refer to output error codes in"11.7.8 MQTT.MQTT_ERROR (Error Code)".

(Note 1) To send QoS0 messages, specify FALSE.

(Note 2) For the maximum message size that can be sent, refer to"11.7.2 MQTT Client Specifications".

(Note 3) The parameter turns TRUE at the completion of transmission and remains only for one cycle.

■ MQTTPublishProperties (Structure)

Optional data settings for sending messages can be configured. This is valid only for the protocol version 5.0.

Name	Type	Description
bPayloadFormatIndicator	BYTE	Payload format Use this setting with the value fixed to 1.
udiMessageExpiryInterval	UDINT	Interval for which the broker server keeps the message(Note 1) 1 to 4,294,967,295: keeping time (unit: s) 0, not specified: kept permanently
wsContentType	WSTRING	Payload content type Do not specify any value.
wsResponseTopic	WSTRING	Topic name for a response message(Note 2)
udiCorrelationDataSize	UDINT	Correlation data size (unit: byte)(Note 2)
paCorrelationData	POINTER TO BYTE	Address to correlation data(Note 2)
udiSubscriptionIdentifier	UDINT	Subscription ID of the last received message (corresponding to the MQTTSubscribe output) Do not specify this when an MQTTPublish instance is executed.
uiTopicAlias	UINT	Topic name alias(Note 3)
userProperties	ARRAY [0..9] OF MQTTStringPair	Refer to user-defined properties in"11.7.4 MQTT.MQTTClient (MQTT Client Connection)".

(Note 1) To use this, enable the Retain function.

(Note 2) If you want to implement request/response type communication using MQTT, you can use it. The maximum size of the correlation data that can be set is 256 bytes.

(Note 3) Specify a value less than or equal to the maximum Topic Alias number accepted by the broker server. The maximum Topic Alias number accepted by the broker server is output to the variable below when an MQTTClient instance is executed.

MQTTClient.mQTTConnackProperties.uiTopicAliasMaximum

Info.

- If you are already using 20 MQTTSubscribe instances with the same client, the eMQTTError of MQTTPublish will output MAX_NUMBER_OF_PUBLISHER_AND_SUBSCRIBER_EXCEEDED, and it will not execute. At this time, xError will not become TRUE.
- After an error occurs, if the cause of the error is resolved while xExecute remains TRUE, the output of eMQTTError may return to NO_ERROR. At this time, xError will remain TRUE.
- Use the parameters in accordance with the broker server settings you use.
- For the protocol version 5.0, it is possible to determine whether input values are valid using the MQTTClient.mQTTConnackProperties value.

11.7.6 MQTT.MQTTSubscribe (MQTT Subscribe Function)

This is a function block for registering a subscription with the MQTT broker server. To use it, you must first execute MQTTClient and establish a connection with the broker server.

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	xEnable	BOOL	FALSE	TRUE: Execution of the FB is enabled. FALSE: Stops the FB. (Cancels the subscription)
	eSubscribeQoS	MQTT_QOS	MQTT_QOS.QoS0	Specifies QoS level between the broker server and MQTTSubscribe. Refer to"11.7.4 MQTT.MQTTClient (MQTT Client Connection)".
	pbPayload	POINTER TO BYTE	0	An address to the received message storage destination
	udiMaxPayloadSize	UDINT	0	Size of the received message storage destination (unit: byte)
	eFilterMode	FILTER_MODE	FILTER_MODE.FILTER_ON	Topic name filter setting
	mQTTCSubscribeProperties	REFERENCE TO MQTTSubscribeProperties	-	Optional data used for registering subscriptions (valid only for protocol version 5.0)
	udiTimeout	UDINT	1000000	Timeout time for subscription registration process (unit: μs)
Input / output	mqttClient	MQTTClient	-	Reference to MQTTClient registering the subsctiption
	wsTopicFilter	WSTRING(1024)	""	Topic name of the message to be received(Note 1)
Output	xDone	BOOL	FALSE	TRUE: Completion of the subscription registration
	xBusy	BOOL	FALSE	TRUE: Function block is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred in the function block.

Scope	Name	Type	Default value	Description
	eMQTTError	MQTT_ERROR	MQTT_ERROR.NO_ERROR	Refer to output error codes in"11.7.8 MQTT.MQTT_ERROR (Error Code)".
	xReceived	BOOL	FALSE	TRUE: A message has been received.(Note 2)
	udiPayloadSize	UDINT	0	Size of the received message
	xSubscribeActive	BOOL	FALSE	Status of subscription registration TRUE: Registered (waiting to receive messages) FALSE: Unregistered
	wsLastTopic	WSTRING(1024)	""	Topic name of the last received message
	mQTTPublishProperties	REFERENCE TO MQTTPublishProperties	-	Optional data of the last received message. Refer to"11.7.5 MQTT.MQTTPublish (MQTT Publish Function)".

(Note 1) For the maximum length of the receivable topic name and the maximum number of topic levels, please refer to"11.7.2 MQTT Client Specifications". Also, do not use the same topic name between MQTTSubscribes using the same MQTTClient instance.

(Note 2) The parameter turns TRUE at the completion of message reception and remains only for one cycle.

■ FILTER_MODE (Enumeration type)

Name	Value	Description
FILTER_ON	0	Receives only messages under the same topic name specified in wsTopicFilter.
FILTER_OFF	1	Receives all messages addressed to the same client irrespective of the topic name specified in wsTopicFilter.(Note 1)
FILTER_NONE	2	Not receive all messages irrespective of the topic name specified in wsTopicFilter.(Note 2)

(Note 1) Only messages published to other subscription topics in which mqttClient is set to the same MQTTClient instance can be received.

(Note 2) In this mode, you are only allowed to register/cancel subscriptions to topic names.

■ MQTTSubscribeProperties (Structure)

Optional data settings for registering subscriptions can be configured. This is valid only for the protocol version 5.0.

Name	Type	Description
udiSubscriptionIdentifier	UDINT	Subscription ID
udiCorrelationDataSize	UDINT	Size of correlation data (unit: byte)(Note 1)(Note 2)
paCorrelationData	POINTER TO BYTE	Address to correlation data(Note 1)
userProperties	ARRAY [0..9] OF MQTTStringPair	Refer to user-defined properties in"11.7.4 MQTT.MQTTClient (MQTT Client Connection)".

Name	Type	Description
xNoLocalOption	BOOL	TRUE: Reception of messages from the same client is permitted. FALSE: Reception of messages from the same client is not permitted.
xRetainAsPublished	BOOL	The setting of the Retain flag in a forwarded message Do not specify any value.
eRetainHandling	MQTT_RETAIN_HANDLING	Setting of whether or not to receive retained messages at the time of subscription registration

(Note 1) If you want to implement request/response type communication using MQTT, you can use it. The maximum size that can be set is 256 bytes.

(Note 2) Please note that if the size of the received message's correlation data exceeds udiCorrelationDataSize, the correlation data of the received message will be discarded.

■ MQTT_RETAIN_HANDLING (Enumeration type)

Name	Value	Description
Time_Of_Subscribe	0	The client receives retained messages at the time of subscription registration.
Not_Exists	1	The client receives retained messages only when a new subscription is registered.
Do_Not_Send	2	The client does not receive retained messages at the time of subscription registration.

Info.

- If a single client is already using 20 instances of MQTTSubscribe, any subsequent MQTTSubscribe instances beyond the 21st will output MAX_NUMBER_OF_PUBLISHER_AND_SUBSCRIBER_EXCEEDED in eMQTTError and will not execute. At this time, xError will not become TRUE.
- If you are receiving a string type message, be sure to set NULL (0) at the end of the message.
- Use the parameters in accordance with the broker server settings you use.
- For the protocol version 5.0, it is possible to determine whether input values are valid using the MQTTClient.mQTTConnackProperties value.

11.7.7 MQTT.MQTT_REASON_CODE (Reason Code)

This is an enumeration type code that is output when a function block of the MQTT function is executed. Each error code indicates the result of an operation. The reason codes are output only if the protocol version 5.0 is used.

■ MQTT.MQTT_ERROR (Enumeration type)

Name	Value	Description
Success	0	Successful operation, granted QoS “0” (at the time of MQTTSubscribe execution)
Granted_QoS_1	1	Granted QoS “1”

Name	Value	Description
Granted_QoS_2	2	Granted QoS “2”
Disconnect_with_Will_Message	4	Disconnected with Will message
No_matching_subscribers	16	Subscribers that match the filter name are unregistered.
No_subscription_existed	17	The specified subscription does not exist.
Continue_authentication	24	Continued authentication
Re_authenticate	25	Re-authentication
Unspecified_error	128	An unspecified error
Malformed_Packet	129	A malformed packet
Protocol_Error	130	Protocol error
Implementation_specific_error	131	An implementation-specific error resulting from a valid packet
Unsupported_Protocol_Version	132	Unsupported protocol version
Client_Identifier_not_valid	133	Client ID not valid
Bad_User_Name_or_Password	134	Bad user name or password
Not_authorized	135	Not authorized
Server_unavailable	136	Broker server is unavailable.
Server_busy	137	Broker Server is busy.
Banned	138	Banned connection
Server_shutting_down	139	Broker sever shutting down
Bad_authentication_method	140	Bad authentication method
Keep_Alive_timeout	141	Keep alive timeout
Session_taken_over	142	Session has been taken over.
Topic_Filter_invalid	143	Topic filter is invalid.
Topic_Name_invalid	144	Topic name is invalid.
Packet_Identifier_in_use	145	Packet ID in use
Packet_Identifier_not_found	146	Packet ID cannot not be found.
Receive_Maximum_exceeded	147	Reached Receive Maximum limit.
Topic_Alias_invalid	148	Topic Alias is invalid.
Packet_too_large	149	The packet size is too large.
Message_rate_too_high	150	The received data rate is too high.
Quota_exceeded	151	Quota exceeded
Administrative_action	152	Disconnection due to an administrative action
Payload_format_invalid	153	Payload format is invalid.
Retain_not_supported	154	Retain not supported
QoS_not_supported	155	QoS not supported
Use_another_server	156	Use another broker server (temporary change).
Server_moved	157	Move to another broker server location (permanent change).

Name	Value	Description
Shared_Subscriptions_not_supported	158	Shared Subscriptions not supported
Connection_rate_exceeded	159	The connection data rate is too high.
Maximum_connect_time	160	The maximum connection time has been exceeded.
Subscription_Identifiers_not_supported	161	Subscription IDs not supported
Wildcard_Subscriptions_not_supported	162	Wildcard Subscriptions not supported

11.7.8 MQTT.MQTT_ERROR (Error Code)

This is an enumeration type error code that is output when a function block of the MQTT function is executed.

■ MQTT.MQTT_ERROR (Enumeration type)

Name	Value	Description
NO_ERROR	0	No error
TCP_INIT_ERROR	1	TCP socket initialization has failed.
TCP_READ_ERROR	2	An error has occurred while received data is read.
TCP_WRITE_ERROR	3	An error has occurred while data is transmitted.
MAX_RESPONSE_SIZE_EXCEEDED	4	The packet size of the received data is greater than the maximum packet size.
DECODE_REMAINING_LENGTH_MALFORMED	5	An invalid packet format is detected.
RESPONSE_PACKET_EMPTY	6	Empty received data is detected.
INVALID_PACKET_TYPE	7	An invalid packet type is detected inside the fixed header.
INVALID_PACKET_BIT_FLAGS	8	An invalid packet bit flag is detected inside the fixed header.
INVALID_PACKET	9	Invalid packet
KEEP_ALIVE_TIME_EXCEEDED	10	Exceeded keep alive time
WRONG_SESSION_PRESENT_CONNACK	11	A wrong session is detected in the CONNACK packet.
UNACCEPTABLE_PROTOCOL_VERION	12	Connection is rejected due to an unacceptable protocol version.
IDENTIFIER_REJECTED	13	The client ID is rejected and so connection is rejected.
SERVER_UNAVAILABLE	14	Connection is rejected because the broker server is unavailable.
BAD_USER_NAME_PASSWORD	15	Connection is rejected because the user name or password is not correct.
NOT_AUTHORIZED	16	Connection is rejected due to unauthorized access.
TOPIC_FILTER_EMPTY	17	The topic filter is empty.
TOPIC_NAME_NOT_ALLOWED_WILDCARD	18	A wildcard is contained in the topic name.

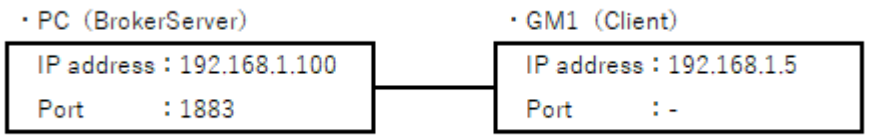
Name	Value	Description
TOPIC_INVALID_LENGTH	19	The topic length is outside the effective range.
TOPIC_IS_EMPTY	20	The topic name is empty.
SUBSCRIBE_FAILURE	21	Failed to register subscription.
ADD_MQTT_PACKET_COLLECTION_ERROR	22	A collection error is detected when an MQTT packet is added to the stack.
ADD_SUBSCRIBER_COLLECTION_ERROR	23	A collection error is detected when a subscriber is added to the stack.
REMOVE_SUBSCRIBER_COLLECTION_ERROR	24	A collection error is detected when a subscriber is removed from the stack.
ACKNOWLEDGE_TIMEOUT	25	Ping packet response was not within the specified time interval (tPingInterval × 2).
ALLOCATED_PAYLOAD_SIZE_EXCEEDED	26	The payload size of the received data is greater than the allocated memory size.
MAX_NUMBER_OF_PACKETS_EXCEEDED	27	Exceeded the maximum packet size.
CAN_NOT_ADD_ELEMENT_TO_QUEUE	28	The element cannot be added to the queue (it may exceed the maximum size).
QUERYINTERFACE_ERROR	29	Failed to call the “QUERYINTERFACE” function (an internal error).
TIME_OUT	30	Timeout is detected.
INVALID_LICENSE	31	A valid license is not found, or the demo mode period expired.
CLIENT_NOT_CONNECTED	32	MQTTClient is not connected to the broker server.
RESOLVE_HOSTNAME_FAILED	33	Host name cannot be resolved.
MAX_REQUEST_SIZE_EXCEEDED	34	The size of the issued packet is greater than the maximum packet size.
UNSUPPORTED_VERSION	35	Unsupported MQTT version
OPERATION_NOT_SUCCESSFUL	36	The operation failed. For details, refer to eReaonCode.
SEND_QUOTA_LIMIT_REACHED	37	Reached Receive Maximum limit.
INVALID_REASON_CODE	38	The reason code is invalid.
MAX_RECEIVE_BUFFER_SIZE_EXCEEDED	39	A packet exceeding the receive buffer size has been received.
MAX_TOPIC_LEVEL_EXCEEDED	40	Exceeded the maximum value of topic level
MAX_STRING_LENGTH_EXCEEDED	41	Exceeded the maximum length of TRING (1024)
MAX_CORRELATION_DATA_LENGTH_EXCEEDED	42	Exceeded the maximum size of correlation data
MAX_NUMBER_OF_PUBLISHER_AND_SUBSCRIBER_EXCEEDED	43	Exceeded the maximum number of publish and subscribe that can be executed simultaneously

11.7.9 Sample Example: MQTT Communication

A publish/subscribe mode communication example using the MQTT functions is described below.

In this program example, message publishing (sending) and subscribing (receiving) are performed by one client. (Either of MQTT protocol version 3.1.1 and 5.0 is used.)

- System configuration



- Description of process

  Registering an MQTT client on a broker server to send and receive a message on the topic below.

  - Topic name: Test
  - Message: test123456789

  1. Set bMQTT_Connect to TRUE to establish connection with the broker server.

     To change the MQTT protocol version you use, change the eVersion value before this step.
  2. Set bMQTT_Receive to TRUE to register a subscriber.
  3. Set bMQTT_Send to TRUE to send the message to the broker server.

     The sent message is forwarded from the broker server to the subscriber registered in the preceding step. (In this example, the message is stored in sMQTT_Receivemessage.)

- Declaration section

```
PROGRAM MQTT_Connection
VAR

//FB Instance
MQTTClient_0     : MQTT.MQTTClient;
MQTTPublisher_0  : MQTT.MQTTPublish;
MQTTSubscriber_0 : MQTT.MQTTSubscribe;

//MQTT Parameter
sHostname  : STRING(255)       := '192.168.1.100';      // Hostname or ip
address or URL
uiPort     : UINT              := 1883;                 // Port of the MQT
T broker
eVersion   : MQTT.MQTT_VERSION := MQTT.MQTT_VERSION.V5; // MQTT protocol v
ersion

//MQTT Properties (Set value if necessary)
sConnectProperties     : MQTT.MQTTConnectProperties;
sWillProperties        : MQTT.MQTTWillProperties;
sPublishProperties     : MQTT.MQTTPublishProperties;
sSubscribeProperties   : MQTT.MQTTSubscribeProperties;
```

```
 bMQTT_Send             : BOOL;   // Publish exe
 bMQTT_Connect          : BOOL;   // Client exe
 bMQTT_Receive          : BOOL;   // Subscribe exe
 sMQTT_Receivemessage   : STRING; // Receive Buffer

 wsMQTT_Topic      :WSTRING(1024)  := "Test";          // Topic Name/Filter
 sMQTT_sendmessage :STRING         := 'test123456789'; // Send Buffer

 bConnect     : BOOL;
 bsendOK      : BOOL;
 breceiveOK   : BOOL;

 END_VAR
```

- Implementation section

```
// MQTTClient Connect
    MQTTClient_0(xEnable               := bMQTT_Connect,
                 uiPort                := uiPort,
                 eMQTTVersion          := eVersion,
                 mQTTConnectProperties := sConnectProperties,
                 sHostname             := sHostname );

    bConnect     := MQTTClient_0.xConnectedToBroker; // Get server connecti
on status


IF bConnect = TRUE THEN
    // publish a message
    MQTTPublisher_0(xExecute              := bMQTT_Send,
                    eQoS                  := MQTT.MQTT_QOS.QoS0,
                    pbPayload             := ADR(sMQTT_sendmessage),
                    udiPayloadSize        := DINT_TO_UDINT(Stu.StrLenA(ADR(
sMQTT_sendmessage))),
                    mQTTPublishProperties := sPublishProperties,
                    mqttClient            := MQTTClient_0,
                    wsTopicName           := wsMQTT_Topic );

    IF MQTTPublisher_0.xDone = TRUE THEN
        bsendOK    := TRUE;
        bMQTT_Send := FALSE;
    END_IF

    // Subscribe registration
    MQTTSubscriber_0(xEnable                 := bMQTT_Receive,
                     eSubscribeQoS           := MQTT.MQTT_QOS.QoS0,
                     pbPayload               := ADR(sMQTT_Receivemessage),
                     udiMaxPayloadSize       := SIZEOF(sMQTT_Receivemessage
),
                     mQTTSubscribeProperties := sSubscribeProperties,
                     mqttClient              := MQTTClient_0,
                     wsTopicFilter           := wsMQTT_Topic);

    // Set the terminating character when receiving a message
    IF MQTTSubscriber_0.xReceived = TRUE THEN
```

```
        breceiveOK := TRUE;
        TerminateString(psIn      := ADR(sMQTT_Receivemessage),
                        udiLength := MQTTSubscriber_0.udiPayloadSize);
    END_IF

END_IF
```

- * Inside FUNCTION TerminateString

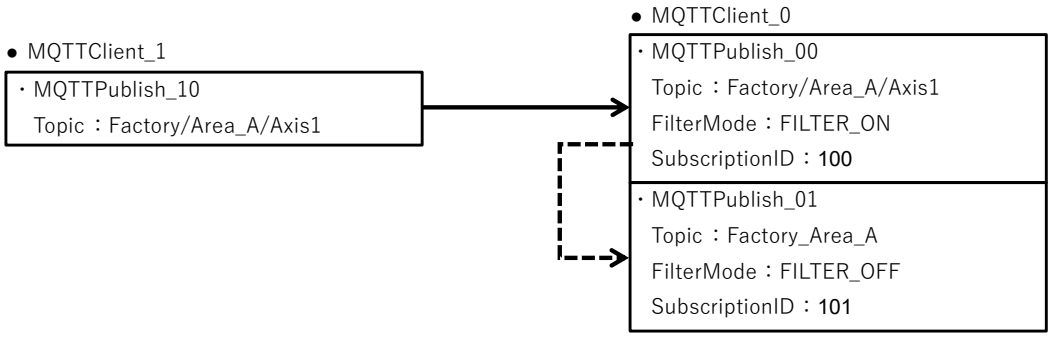
```
// Terminates the STRING psIn at position udiLength
FUNCTION TerminateString : BOOL
VAR_INPUT
    psIn     : POINTER TO BYTE; // Pointer to STRING
    udiLength : UDINT;          // Length of psIn
END_VAR
VAR
END_VAR

psIn[udiLength] := 0;
TerminateString := TRUE;
```

11.7.10 Example: MQTT Communication Using Filter Mode

Below is an example of an MQTT communication program using Filter Mode.

In this program example, FILTER_MODE is used to mirror all messages intended for the same client. Additionally, by utilizing the Subscription ID feature, it is possible to discern to which subscribe the messages belong. (The Subscription ID is only supported in MQTT protocol version 5.0.)

- Similar to system configuration
  "11.7.9 Sample Example: MQTT Communication"
- Processing Content
  Send a message (axis data) to the topic of MQTTSubscribe_00.



1. Set bCliEna_0 to TRUE to register a subscription with the broker server.
2. Set bCliEna_1 and bPubExe_10 to TRUE to send a message to the broker server.
   The message sent is forwarded from the broker server to the subscription registered in the previous step. Although the message is sent to the topic of MQTTSubscriber_00, it is also stored in MQTTSubscriber_01. Additionally, the subscription ID of

MQTTSubscriber_00 is stored in MQTTSubscriber_01.MQTTPublishProperties.udiSubscriptionIdentifier.

- Declaration section

```
PROGRAM MQTT_FilterMode
VAR

//FB Instance
MQTTClient_0        : MQTT.MQTTClient;
MQTTSubscribe_00    : MQTT.MQTTSubscribe;
MQTTSubscribe_01    : MQTT.MQTTSubscribe;

MQTTClient_1        : MQTT.MQTTClient;
MQTTPublish_10      : MQTT.MQTTPublish;

//MQTT Properties (Set SubscriptionID)
sSubProp_00   : MQTT.MQTTSubscribeProperties   := (udiSubscriptionIdentifi
er := 100);
sSubProp_01   : MQTT.MQTTSubscribeProperties   := (udiSubscriptionIdentifi
er := 101);

//MQTT Parameter
sHostname   : STRING(255)   := '192.168.1.100'; // Hostname or IPaddress o
r URL
uiPort      : UINT          := 1883; // Port of the MQTT broker

bCliEna_0   : BOOL   := FALSE;
bSubEna_00  : BOOL   := TRUE;
bSubEna_01  : BOOL   := TRUE;
eVersion_0  : MQTT.MQTT_VERSION := MQTT.MQTT_VERSION.V5;
wsTopic_00  : WSTRING(1024)     := "Factory/Area_A/Axis1";
fAxisValue  : LREAL             := 0;
wsTopic_01  : WSTRING(1024)     := "Factory_Area_A";
abyRecvData : ARRAY[0..SIZEOF(LREAL)-1] OF BYTE; // Payload data Check

bCliEna_1   : BOOL   := FALSE;
bPubExe_10  : BOOL   := FALSE;
eVersion_1  : MQTT.MQTT_VERSION   := MQTT.MQTT_VERSION.V5;
fSendData   : LREAL   := 0;

END_VAR
```

- Implementation section

```
(* Subscribe side *)
// MQTTClient Connect
MQTTClient_0(
               xEnable        := bCliEna_0,
               uiPort         := uiPort,
               sClientId      := 'A',
               sHostname      := sHostname,
               eMQTTVersion   := eVersion_0,
            );
```

```
// Subscribe registration (If MQTTClient_0 succeeds, run automatically)
MQTTSubscribe_00(
                    xEnable                 := bSubEna_00 AND MQTTClient_0.
xConnectedToBroker,
                    udiMaxPayloadSize       := SIZEOF(fAxisValue),
                    pbPayload               := ADR(fAxisValue),
                    eFilterMode             := MQTT.FILTER_MODE.FILTER_ON,
                    mqttClient              := MQTTClient_0,
                    wsTopicFilter           := wsTopic_00,
                    mQTTSubscribeProperties := sSubProp_00,
                );

MQTTSubscribe_01(
                    xEnable                 := bSubEna_01 AND MQTTClient_0.
xConnectedToBroker,
                    pbPayload               := ADR(abyRecvData),
                    udiMaxPayloadSize       := SIZEOF(abyRecvData),
                    eFilterMode             := MQTT.FILTER_MODE.FILTER_OFF,
                    mqttClient              := MQTTClient_0,
                    wsTopicFilter           := wsTopic_01,
                    mQTTSubscribeProperties := sSubProp_01,
                );


(* Publish side *)
// MQTTClient Connect
MQTTClient_1(
                xEnable        := bCliEna_1,
                uiPort         := uiPort,
                sClientId      := 'B',
                sHostname      := sHostname,
                eMQTTVersion   := eVersion_1,
            );

fSendData   := Axis1.fActPosition;

MQTTPublish_10(
                  xExecute       := bPubExe_10 AND MQTTClient_1.xConnectedTo
Broker,
                  pbPayload      := ADR(fSendData),
                  udiPayloadSize := SIZEOF(fSendData),
                  mqttClient     := MQTTClient_1,
                  wsTopicName    := wsTopic_00,
             );
```

11.7.11 MQTT Communication: Request/Response Type Communication

Below is an example of the Request/Response type of communication using MQTT functionality.

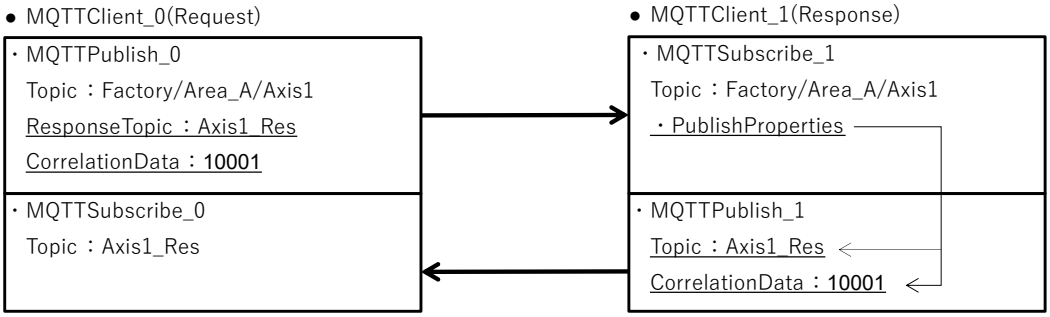
In this program example, properties are used to perform message publish (send) and subscribe (receive) in a request/response manner. (This is only supported in MQTT protocol version 5.0.)

- Similar to system configuration

"11.7.9 Sample Example: MQTT Communication"

- Processing Content

  The request side (MQTTClient_0) sends a message to the response side with the response destination topic name (ResponseTopic) and an ID (CorrelationData) to identify the request. When the response side (MQTTClient_1) receives a message with a ResponseTopic set, it reads the attached ResponseTopic and CorrelationData and returns a message (axis data) addressed to ResponseTopic. The request side, upon receiving the response message, compares the CorrelationData of the response message with the initially provided CorrelationData to confirm that it is a response to the request they sent.



1. Set bCliEna_0 and bCliEna_1 to TRUE to register a subscription with the broker server.
2. Set bPubExe_0 to TRUE to send a message to the broker server.

   If the response side receives a message with a ResponseTopic set, it automatically returns a message using the received message's ResponseTopic and CorrelationData. When the request side successfully receives a response, bComplete becomes TRUE.

- Declaration section

```
 PROGRAM MQTT_RequestResponse
 VAR
    //FB Instance
    MQTTClient_0    : MQTT.MQTTClient;
    MQTTPublish_0   : MQTT.MQTTPublish;
    MQTTSubscribe_0 : MQTT.MQTTSubscribe;

    MQTTClient_1    : MQTT.MQTTClient;
    MQTTPublish_1   : MQTT.MQTTPublish;
    MQTTSubscribe_1 : MQTT.MQTTSubscribe;

    //MQTT Properties (Set value if necessary)
    sPubProp_0   : MQTT.MQTTPublishProperties;
    sPubProp_1   : MQTT.MQTTPublishProperties;

    //MQTT Parameter
    sHostname    : STRING(255)      := '192.168.1.100'; // Hostname or IPadd
ress or URL
    uiPort       : UINT             := 1883; // Port of the MQTT broker
    eVersion     : MQTT.MQTT_VERSION:= MQTT.MQTT_VERSION.V5; // MQTT protoco
l version

    bCliEna_0    : BOOL   := FALSE;
    bPubExe_0    : BOOL   := FALSE;
    bSubEna_0    : BOOL   := TRUE;
```

```
    bSendData   : BOOL   := FALSE;

    bCliEna_1   : BOOL   := FALSE;
    bSubEna_1   : BOOL   := TRUE;
    bPubExe_1   : BOOL   := FALSE;

    wsTopic_1   : WSTRING(1024)   := "Factory/Area_A/Axis1";
    abyRecvData : ARRAY[0..(MQTT.MQTTParam.g_udiMaxPayloadSize-1)] OF BYTE;

    // RequestResponse variable
    wRequestID     : WORD         := 10001;
    wsRequestTopic : WSTRING(1024):= "Axis1_Res";
    bRequestData   : BOOL         := 0;
    pwRecvReqID    : POINTER TO WORD;

    wsResponseTopic: WSTRING(1024):= "";
    bResponseData  : BOOL         := FALSE;

    bComplete      : BOOL := FALSE;
 END_VAR
```

- Implementation section

```
(* Request side *)
// MQTTClient Connect
MQTTClient_0(
                xEnable         := bCliEna_0,
                uiPort          := uiPort,
                sClientId       := 'Request',
                sHostname       := sHostname,
                eMQTTVersion    := eVersion,
            );

// Prepare SubscriberFB to receive response data (If MQTTClient_0 succeeds,
 run automatically)
MQTTSubscribe_0(
                    xEnable              := bSubEna_0 AND MQTTClient_0.xCon
nectedToBroker,
                    pbPayload            := ADR(bRequestData),
                    udiMaxPayloadSize    := SIZEOF(bRequestData),
                    mqttClient           := MQTTClient_0,
                    wsTopicFilter        := wsRequestTopic,
                );

// Prepare a SubscribeFB for Response and then execute a PublishFB
IF MQTTSubscribe_0.xSubscribeActive = TRUE THEN

    // Set the receiving topic and identification number
    // Use "CorrelationData" to link request and response messages
    sPubProp_0.wsResponseTopic          := wsRequestTopic;
    sPubProp_0.paCorrelationData        := ADR(wRequestID);
    sPubProp_0.udiCorrelationDataSize   := SIZEOF(wRequestID);

    MQTTPublish_0(
                        xExecute              := bPubExe_0,
```

```
                        pbPayload             := ADR(bSendData),
                        udiPayloadSize        := SIZEOF(bSendData),
                        xRetain               := true,
                        mqttClient            := MQTTClient_0,
                        wsTopicName           := wsTopic_1,
                        mQTTPublishProperties := sPubProp_0,
                    );

    IF MQTTPublish_0.xDone = TRUE THEN
        bPubExe_0 := FALSE;
    END_IF
END_IF

//When you receive a Response message, check the identification number
IF MQTTSubscribe_0.xReceived = TRUE THEN
    pwRecvReqID    := MQTTSubscribe_0.mQTTPublishProperties.paCorrelationDa
ta;
    IF pwRecvReqID^ = wRequestID THEN
        bComplete  := TRUE;
    END_IF
END_IF


(* Response side *)
// MQTTClient Connect
MQTTClient_1(
                xEnable         := bCliEna_1,
                uiPort          := uiPort,
                sClientId       := 'Response',
                sHostname       := sHostname,
                eMQTTVersion    := eVersion,
            );

// Subscribe registration (If MQTTClient_1 succeeds, run automatically)
MQTTSubscribe_1(
                    xEnable              := bSubEna_1 AND MQTTClient_1.xCon
nectedToBroker,
                    pbPayload            := ADR(abyRecvData),
                    udiMaxPayloadSize    := SIZEOF(abyRecvData),
                    mqttClient           := MQTTClient_1,
                    wsTopicFilter        := wsTopic_1,
                );

// Automatically respond when ResponseTopic is included in received Message
IF MQTTSubscribe_1.xReceived = TRUE
AND MQTTSubscribe_1.mQTTPublishProperties.wsResponseTopic <> "" THEN

    bResponseData := Axis1.bError;

    // Get response destination topic and identification number from receiv
ed Message
    sPubProp_1.udiCorrelationDataSize   := MQTTSubscribe_1.mQTTPublishPrope
rties.udiCorrelationDataSize;
    sPubProp_1.paCorrelationData        := MQTTSubscribe_1.mQTTPublishPrope
rties.paCorrelationData;
    wsResponseTopic                     := MQTTSubscribe_1.mQTTPublishPrope
```

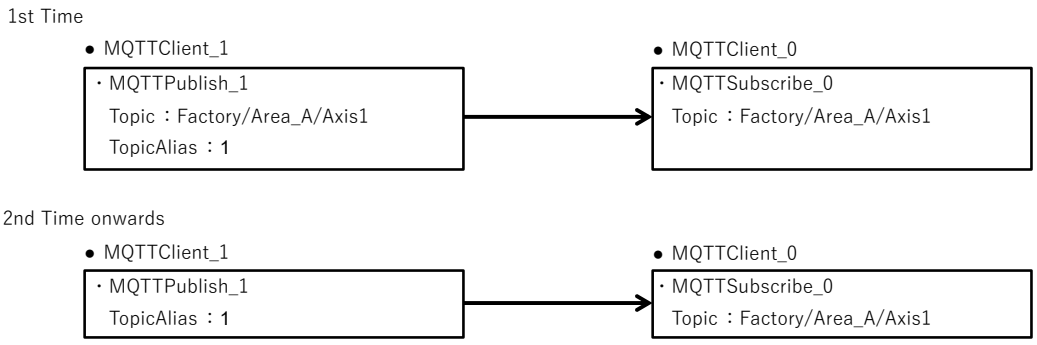
```
rties.wsResponseTopic;

    bPubExe_1 := TRUE;

END_IF

MQTTPublish_1(
                    xExecute              := bPubExe_1,
                    pbPayload             := ADR(bResponseData),
                    udiPayloadSize        := SIZEOF(bResponseData),
                    mqttClient            := MQTTClient_1,
                    wsTopicName           := wsResponseTopic,
                    mQTTPublishProperties := sPubProp_1,
              );

IF MQTTPublish_1.xDone = TRUE THEN
    bPubExe_1 := FALSE;
END_IF
```

11.7.12 Example: MQTT Communication Using Topic Alias

Below is an example of MQTT communication using the Topic Alias feature.

In this program example, topic aliases are used to perform message publishing (sending) without specifying the topic name. (Topic aliases are only supported in MQTT protocol version 5.0.) To use topic aliases, it is necessary to execute a publish with the topic name and topic alias set at least once.

- Similarly to the system configuration
  "11.7.9 Sample Example: MQTT Communication".
- Processing Content
  After the first publish execution, clear the topic name, and from the second time onwards, execute sending and receiving messages (axis data) using only the topic alias.



1. Set bCliEna_0 to TRUE to register a subscription with the broker server.
2. Set bCliEna_1 and bPubExe_1 to TRUE to send a message to the broker server.
   After sending the message, bPubExe_1 becomes FALSE and the topic name that was set in wsTopic_1 is cleared.
3. Set bPubExe_1 to TRUE again to send another message to the broker server.

Although MQTTPublish_1.wsTopicName is left blank, the message sent is successfully forwarded to the subscription.

- Declaration section

```
PROGRAM MQTT_TopicAlias
VAR
   //FB Instance
   MQTTClient_0        : MQTT.MQTTClient;
   MQTTSubscribe_0     : MQTT.MQTTSubscribe;

   MQTTClient_1        : MQTT.MQTTClient;
   MQTTPublish_1       : MQTT.MQTTPublish;

   //MQTT Properties (Set TopicAlias)
   sPubProp_1  : MQTT.MQTTPublishProperties := (uiTopicAlias := 1);

   //MQTT Parameter
   sHostname   : STRING(255)       := '192.168.1.100'; // Hostname or IPad
dress or URL
   uiPort      : UINT              := 1883; // Port of the MQTT broker
   eVersion    : MQTT.MQTT_VERSION := MQTT.MQTT_VERSION.V5; // MQTT protoc
ol version

   bCliEna_0   : BOOL          := FALSE;
   bSubEna_0   : BOOL          := TRUE;
   wsTopic_0   : WSTRING(1024) := "Factory/Area_A/Axis1";
   fAxisValue  : LREAL         := 0;

   bCliEna_1   : BOOL          := FALSE;
   bPubExe_1   : BOOL          := FALSE;
   wsTopic_1   : WSTRING(1024) := "Factory/Area_A/Axis1";
   fSendData   : LREAL         := 0;

   bAliasSet   : BOOL := FALSE;
END_VAR
```

- Implementation section

```
(* Subscribe side *)
// MQTTClient Connect
MQTTClient_0(
               xEnable         := bCliEna_0,
               uiPort          := uiPort,
               sClientId       := 'A',
               sHostname       := sHostname,
               eMQTTVersion    := eVersion,
           );

// Subscribe registration (If MQTTClient_0 succeeds, run automatically)
MQTTSubscribe_0(
                    xEnable              := bSubEna_0 AND MQTTClient_0.xCon
nectedToBroker,
                    pbPayload            := ADR(fAxisValue),
                    udiMaxPayloadSize    := SIZEOF(fAxisValue),
```

```
                    mqttClient           := MQTTClient_0,
                    wsTopicFilter        := wsTopic_0,
                );


(* Publish side *)
// MQTTClient Connect
MQTTClient_1(
                xEnable         := bCliEna_1,
                uiPort          := uiPort,
                sClientId       := 'B',
                sHostname       := sHostname,
                eMQTTVersion    := eVersion,
            );

// Publish execution (If MQTTClient_1 succeeds, run automatically only once
)
IF MQTTClient_1.xConnectedToBroker = TRUE THEN

    fSendData    := Axis1.fActPosition;

    MQTTPublish_1(
                    xExecute              := bPubExe_1,
                    pbPayload             := ADR(fSendData),
                    udiPayloadSize        := SIZEOF(fSendData),
                    mqttClient            := MQTTClient_1,
                    wsTopicName           := wsTopic_1,
                    mQTTPublishProperties := sPubProp_1,
                );

    // Empty the TopicName after first published succeeds
    IF bAliasSet = FALSE AND MQTTPublish_1.xDone = TRUE THEN
        wsTopic_1    := "";
        bPubExe_1    := FALSE;
        bAliasSet    := TRUE;
    END_IF

END_IF
```

11.8 LAN Port (DNS)

This section describes the instructions that are used to perform communication with the LAN port using the DNS protocol.

11.8.1 What is DNS?

DNS stands for the Domain Name System and refers to a system that manages a mapping between the name of a domain or a host on the network and its IP address. A DNS server has information about mappings between domain names and IP addresses. In response to a query containing a “host name” as a key from a DNS client, the DNS server sends back a corresponding “IP address”.

The GM1 controller can obtain an IP address corresponding to a domain name from a DNS server through an FB of a DNS client.

11.8.2 DNS_GetIPAddress (Name Resolution)

This is used to send the DNS server a query about the IP address of the specified host name.

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	Execute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	HostName	STRING(255)	-	Host name on which a query is sent to the DNS server (Note 1)(Note 2)
	DNSIPAddress	STRING(255)	-	IP address of the DNS server (Note 2)
	Timeout	UINT	20	Connection timeout 1 to 60 (s)
	Retry	UINT	0	Number of connection retries: 0 to 3
Output	Busy	BOOL	FALSE	TRUE: The FB is in operation.
	Done	BOOL	FALSE	TRUE: Execution is completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	DNS_CLI_ERROR	NO_ERROR	Error ID output

Scope	Name	Type	Default value	Description
	IPAddress	STRING(255)	-	IP address corresponding to the host name

(Note 1) The domain name must be up to 253 characters.

(Note 2) If no value is specified for any of HostName and DNSIPAddress, an error occurs.

Info.

- For a program example, refer to "11.8.4 Sample Example: DNS Name Resolution".

11.8.3 DNS_CLI_ERROR (Enumeration Type)

This is an enumeration type error code that is output when an DNS function block is executed.

■ DNS_CLI_ERROR (Enumeration type)

Name	Value	Description
NO_ERROR	0	No error
DNS_SYSTEM_ERROR	1	Internal error
DNS_WRONG_PARAMTER	2	Incorrect input parameter
DNS_MULTIPLE_EXECUTION	3	Multiple execution error occurs.
DNS_RESOLUTION_FAILED	4	Name resolution failure
DNS_CONNECTION_TIMEOUT	5	Communication timeout
DNS_ILLEGAL_NAME	6	Illegal host name

11.8.4 Sample Example: DNS Name Resolution

This is a program coded to send the DNS server a query about the IP address corresponding to the host name.

- Description of process

  When the case number (byStep) is set to 1, a name resolution process is executed. Before execution of the process, specify values for the variable sSendSrv_ip (IP address of the DNS server) and variable sHostName (host name whose IP address is to be acquired). Details of the process for each case number in the implementation section are as described below.

  1. DNS_GetIPAddress is executed to acquire the IP address corresponding to the host name from the DNS server.

  2. When the process is completed, the variable bFinish goes TRUE.

- Declaration section

```
VAR
// Start DNS function
byStep                  : BYTE := 0;                    //Process No
bFinish                 : BOOL := FALSE;

// FB instance
DNS_GetIPAddress_0      : DNS_GetIPAddress;

// Variables
bDNS_GetIPAddress_exe   : BOOL   := FALSE;
sHostName               : STRING := 'www.Srv01.local';  //HostName
sSendSrv_ip             : STRING := '192.168.1.100';    //DNS Server IP
uiTimeout               : UINT   := 10;                 //Timeout[s]
uiRetry                 : UINT   := 1;                  //Retry
sGETIPAddress           : STRING;                       //Host IP

END_VAR
```

- Implementation section

```
//FunctionBlock
DNS_GetIPAddress_0( Execute        := bDNS_GetIPAddress_exe,
                    HostName       := sHostName,
                    DNSIPAddress   := sSendSrv_ip,
                    Timeout        := uiTimeout,
                    Retry          := uiRetry,
                    IPAddress      => sGETIPAddress );

CASE byStep OF
    1: // Host name resolution
        bFinish := FALSE;
        bDNS_GetIPAddress_exe := TRUE;
        IF (DNS_GetIPAddress_0.Done = TRUE) THEN
            byStep := 2;
        END_IF

    2: //Finish
        bDNS_GetIPAddress_exe := FALSE;
        bFinish := TRUE;
```

```
        byStep := 0;
END_CASE
```

11.9 SD Card Operation (File Operation)

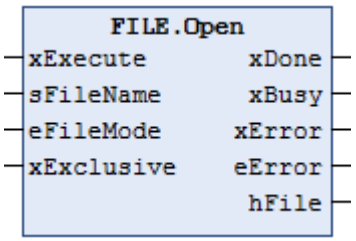
Files in the SD card inserted in the SD memory card slot can be operated.

In file operation using the GM1 Controller, WSTRING (kanji) cannot be used in the file name and directory name.

11.9.1 FILE.Open (Open File)

This is a function block that opens a file or creates a new file.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	sFileName	FILE.CAA.FILENAME	Specifies the file name with an absolute path or relative path.
	eFileMode	FILE.MODE	File mode
	xExclusive	BOOL	TRUE: Exclusive access mode FALSE: Multiple access mode xExclusive is not supported.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".
	hFile	FILE.CAA.HANDLE	Handle of a file

■ FILE.MODE (Enumeration type)

Name	Value	Description
MWRITE	0	Overwrite mode (When the specified file does not exist, a new file is created.)
MREAD	1	Read mode
MRDWR	2	Read / write mode (When the specified file does not exist, a new file is created.)

Name	Value	Description
MAPPD	3	Append write mode

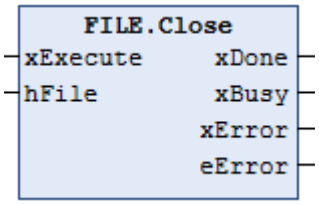
Info.

- You cannot use full size characters and the following symbols in a file name: [\], [/], [:], [*], [?], ["], [<], [>], [|].

11.9.2 FILE.Close (Close File)

This is a function block that closes a file.

■ Icon



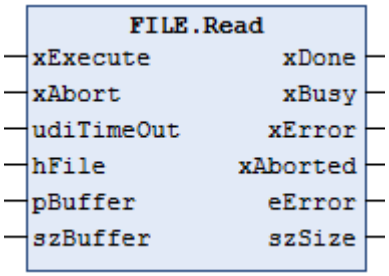
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	hFile	FILE.CAA.HANDLE	Handle of a file to be closed Specifies the handle output by FILE.Open.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".

11.9.3 FILE.Read (Read File)

This is a function block that reads data from the file opened.

■ Icon



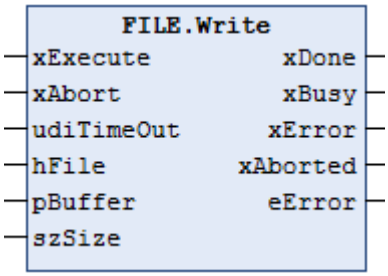
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	xAbort	BOOL	TRUE: Stops execution and resets all outputs.
	udiTimeOut	UDINT	Timeout time until the execution is stopped (μs)
	hFile	FILE.CAA.HANDLE	Handle of a file Specifies the handle output by FILE.Open.
	pBuffer	FILE.CAA.PVOID	Pointer to the data buffer to be read Gets a pointer by the ADR operator.
	szBuffer	FILE.CAA.SIZE	Size of the data buffer to be read Gets a pointer by the SIZEOF operator.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	xAborted	BOOL	TRUE: Execution is stopped by the user.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".
	szSize	FILE.CAA.SIZE	Size of the read data buffer

11.9.4 FILE.Write (Write File)

This is a function block that writes data to the file opened.

■ Icon



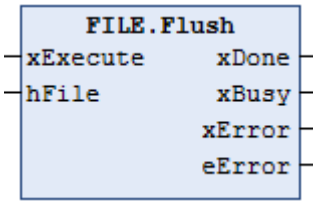
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	xAbort	BOOL	TRUE: Stops execution and resets all outputs.
	udiTimeOut	UDINT	Timeout time until the execution is stopped (μs)
	hFile	FILE.CAA.HANDLE	Handle of a file Specifies the handle output by FILE.Open.
	pBuffer	FILE.CAA.PVOID	Pointer to the data buffer to be written Gets a pointer by the ADR operator.
	szSize	FILE.CAA.SIZE	Size of the data buffer to be written Gets a pointer by the SIZEOF operator.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	xAborted	BOOL	TRUE: Execution is stopped by the user.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".

11.9.5 FILE.Flush (Flush File)

This is a function block that flushes buffer contents to a file.

■ Icon



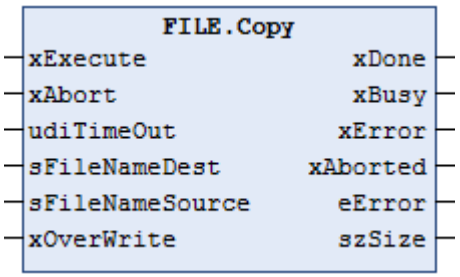
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	hFile	FILE.CAA.HANDLE	Handle of a file Specifies the handle output by FILE.Open.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".

11.9.6 FILE.Copy (Copy File)

This is a function block that copies a file.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	xAbort	BOOL	TRUE: Stops execution and resets all outputs.
	udiTimeOut	UDINT	Timeout time until the execution is stopped (μs)
	sFileNameDest	FILE.CAA.FILENAME	Copy destination file name
	sFileNameSource	FILE.CAA.FILENAME	Copy source file name
	xOverWrite	BOOL	TRUE: Copies to overwrite an existing file. FALSE: Outputs an error without copying to overwrite. If FALSE is specified in a case where there is an existing file, copy is not executed. No error is output.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	xAborted	BOOL	TRUE: Execution is stopped by the user.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".
	szSize	FILE.CAA.SIZE	Size of the copied file

Info.

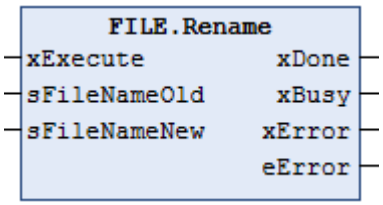
- You cannot use full size characters and the following symbols in a file name: [\], [/], [:], [*], [?], [”], [<], [>], [|].

11.9.7 FILE.Rename (Rename File)

This is a function block that changes a file name.

It is not possible to change the directory name of a directory that is currently open. Close it using the DirClose function block.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	sFileNameOld	FILE.CAA.FILENAME	File name before change
	sFileNameOld	FILE.CAA.FILENAME	File name after change
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".

Info.

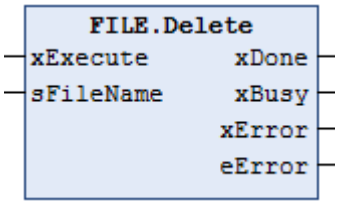
- You cannot use full size characters and the following symbols in a file name: [\], [/], [:], [*], [?], [”], [<], [>], [|].

11.9.8 FILE.Delete (Delete File)

This is a function block that deletes a file.

It is not possible to delete a file that is currently open. Close it using the Close function block.

■ Icon



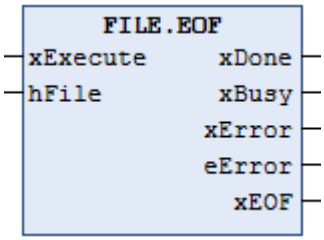
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	sFileName	FILE.CAA.FILENAME	File to be deleted
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".

11.9.9 FILE.EOF (End of File)

This is a function block that determines whether the current offset of a file is EOF (End Of File) or not. It can be used only when the OPEN mode is set to MREAD/MREADPLUS.

■ Icon



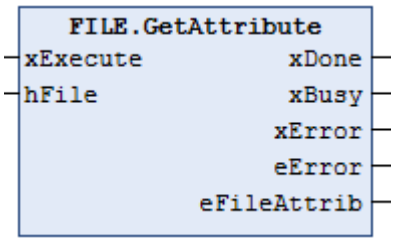
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	hFile	FILE.CAA.FILENAME	Handle of a file Specifies the handle output by FILE.Open.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".
	xEOF	BOOL	File: The current offset is EOF.

11.9.10 FILE.GetAttribute (Get File Attribute)

This is a function block that gets file attributes.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	hFile	FILE.CAA.FILENAME	Handle of a file Specifies the handle output by FILE.Open.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".
	eFileAttrib	FILE.ATTRIB	TRUE: The current offset is EOF. FALSE: The current offset is not EOF.

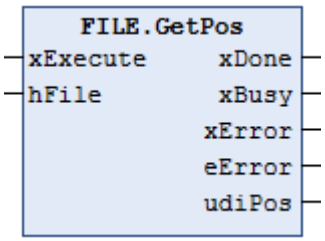
■ FILE.ATTRIB (Enumeration type)

Name	Value	Description
ARCHIVE	0	Archive file
HIDDEN	1	Hidden file
NORMAL	2	File without any other attributes
READONLY	3	Read only

11.9.11 FILE.GetPos (Get File Offset)

This is a function block that gets the current offset of a file.

■ Icon



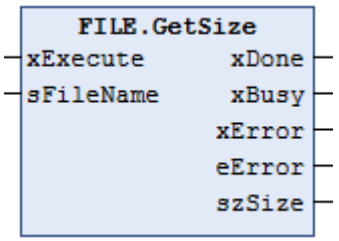
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	hFile	FILE.CAA.FILENAME	Handle of a file Specifies the handle output by FILE.Open.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".
	udiPos	__UXINT	The current offset (byte) is output.

11.9.12 FILE.GetSize (Get File Size)

This is a function block that gets the file size.

■ Icon



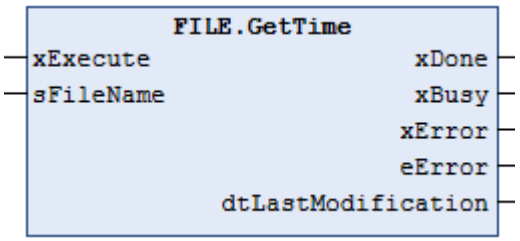
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	sFileName	FILE.CAA.FILENAME	File from which to get the file size
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".
	szSize	FILE.CAA.SIZE	The file size (byte) is output.

11.9.13 FILE.GetTime (Get File Update Time)

This is a function block that gets the update time of a file.

■ Icon



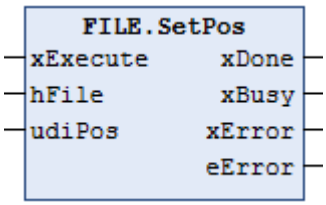
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	sFileName	FILE.CAA.FILENAME	File from which to get the file update time
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".
	dtLastModification	DATE_AND_TIME	The last update date and time is output. Example: DATE_AND_TIME#2020-01-11-15:12:30

11.9.14 FILE.SetPos (Set File Offset)

This is a function block that sets the offset of a file.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	hFile	FILE.CAA.HANDLE	Handle of a file Specifies the handle output by FILE.Open.
	udiPos	__UXINT	Offset to be set (byte)
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".

11.9.15 FILE.ERROR (Error ID)

This is an enumeration type error ID that is output when a function block for file operation is executed. It is used to output an error in a file operation or directory operation of the SD card.

■ FILE.ERROR (Enumeration type)

Name	Value	Description
NO_ERROR	0	Normal end
FIRST_ERROR	5100	First library specific error
TIME_OUT	5101	Timeout
ABORT	5102	Aborts processing by xAbort.
HANDLE_INVALID	5103	Invalid handle
NOT_EXIST	5104	No file or directory exists.
EXIST	5105	A file or directory already exists
NO_MORE_ENTRIES	5106	There are no other entries.
NOT_EMPTY	5107	The file or directory is not empty.
READ_ONLY_CAA	5108	The file or directory is write protected.
WRONG_PARAMETER	5109	Wrong parameter
ERROR_UNKNOWN	5110	Unknown error
WRITE_INCOMPLETE	5111	Not all the data is written.
FILE_NOT_IMPLEMENTED	5112	The function is not implemented.
ASM_CREATEJOB_FAILED	5113	Failed to create an AsyncManager job.
FILE_OPERATION_DENIED	5114	No access due to ForceFilePath / ForcelecFilePath
FIRST_MF	5150	First error unique to the manufacturer
LAST_ERROR	5199	insert manuf. specific errors here Last library specific error

11.9.16 Program example:SD CardFile write processing

SampleDir/SampleFile.txt is created in the SD card and specified data is written to the file.

■ File write processing sequence

The file write processing sequence is shown below.

- File open processing (overwrite mode, insert mode)
  - Overwrite mode: For existing files, the contents of the file are cleared.
  - Insert mode: For existing files, the contents of the file are not cleared.
  - When write is executed, data is written following the end of the previous data.
- File write processing
- File close processing

■ Operation example

- In this example, operations are performed by setting the value of uiProcess to 1, 2, and 2 in this order.
- The contents of SampleFile.txt which is output are as follows:

  NEW_DATA
  ADD_DATA1
  ADD_DATA2

- **Declaration section**

```
PROGRAM FileWrite

VAR
    bResult         : BOOL := FALSE;        // Processing result (FALSE=normal , TRUE=abnormal)
    eOpenResult     : FILE.ERROR;           // File open result
    eWriteResult    : FILE.ERROR;           // File write result
    eCloseResult    : FILE.ERROR;           // File close result
    sFileName       : STRING[80] := 'SampleDir/SampleFile.txt';
                                            // The name of the file to read
    sWriteData      : STRING(256);          // Data to write
    uiAddCnt        : UINT;                 // Addition write count

    uiProcess       : UINT := 0;            // Processing number(1=overwrite mode , 2=write-once mode)
    hFileHandle     : FILE.CAA.HANDLE;      // File handle
    iOpen           : FILE.Open;            // File open Function Block instance
    iWrite          : FILE.Write;           // File read Function Block instance
    iClose          : FILE.Close;           // File Close Instance of FunctionBlock
END_VAR
```

Implementation section

```
CASE uiProcess OF
    1:  // File open processing started (overwrite)
        iOpen(  xExecute := TRUE ,                  // File open processing started
                eFileMode := FILE.MODE.MWRITE ,     // Overwrite mode
                sFileName := sFileName );           // file name
        sWriteData := 'NEW_DATA';                   // Data settings to write
        uiAddCnt := 0;                              // Initialize the number of additional writes
        uiProcess := 3;                             // Transition to waiting for file open processing completion
    2:  // File open processing started (additional writing)
        iOpen(  xExecute := TRUE ,                  // File open processing started
                eFileMode := FILE.MODE.MAPPD ,      // Addendum mode
                sFileName := sFileName );           // file name
        uiAddCnt := uiAddCnt + 1;                   // Addendum write count update
        sWriteData := CONCAT(',ADD_DATA',TO_STRING(uiAddCnt));
                                                    // Data settings to write
        uiProcess := 3;                             // Transition to waiting for file open processing completion
    3:  // Waiting for file open processing to complete
        iOpen();                                    // Instance data update
        IF iOpen.xDone = TRUE THEN                  // File open completed normally
            hFileHandle := iOpen.hFile;             // Get file handle
            eOpenResult := iOpen.eError;            // File open processing result acquisition
            uiProcess := 10;                        // Transition to file write processing
            iOpen( xExecute := FALSE );             // File open processing finished
        ELSIF iOpen.xError = TRUE THEN              // File open processing error occurred
            eOpenResult := iOpen.eError;            // File open processing result acquisition
            uiProcess := 90;                        // Transition to end processing
            bResult := TRUE;                        // Abnormality
            iOpen( xExecute := FALSE );             // File open processing finished
        END_IF
    10: // File writing process started
        iWrite( xExecute := TRUE ,                  // File read processing started
                hFile := hFileHandle ,              // File handle obtained by opening a file
                pBuffer := ADR(sWriteData) ,        // Write data
                szSize := INT_TO_UDINT(LEN(STR:=sWriteData)) );
                                                    // Write data size (for character string)
        uiProcess := 11;                            // Transition to waiting for file write processing completion
    11: // Waiting for file write processing to complete
        iWrite();                                   // Instance data update
        IF iWrite.xDone = TRUE  THEN                // File writing completed normally
            eWriteResult := iWrite.eError;          // File write processing result acquisition
            uiProcess := 20;                        // Transition to file closing process
            iWrite( xExecute := FALSE );            // File writing process completed
        ELSIF iWrite.xError = TRUE THEN             // File write processing error occurred
            eWriteResult := iWrite.eError;          // File write processing result acquisition
            uiProcess := 20;                        // Transition to file close processing (to release the handle)
            bResult := TRUE;                        // Abnormality
            iWrite( xExecute := FALSE );            // File writing process completed
        END_IF
    20: // File close processing started
        iClose( xExecute := TRUE ,                  // File close processing started
                hFile := hFileHandle );             // File handle obtained by opening a file
        uiProcess := 21;                            // Transition to waiting for file close processing completion
    21: // Waiting for file close processing to complete
        iClose();                                   // Instance data update
        IF iClose.xDone = TRUE  THEN                // File close processing completed
            eCloseResult := iClose.eError;          // File close processing result acquisition
            uiProcess := 90;                        // Transition to end processing
            iClose( xExecute := FALSE );            // File close processing completed
        ELSIF iClose.xError = TRUE THEN             // File close processing error occurred
            eCloseResult := iClose.eError;          // File close processing result acquisition
            uiProcess := 90;                        // Transition to end processing
            bResult := TRUE;                        // Abnormality
            iClose( xExecute := FALSE );            // File close processing completed
        END_IF
    90: // End processing
        uiProcess := 0;
END_CASE;
```

11.9.17 Program example:SD CardFile read processing

Data in SampleDir/SampleFile.txt in the SD card is read into the buffer.

The effective range of data read into the buffer is judged from the data size information that is output after read processing.

■ File read processing sequence

The file read processing sequence is shown below.

- File open processing (read mode)
- File read processing
- File close processing

■ Explanation of variables

uiProcess :

Executes processing when the variable is set to 1 (read mode).

■ Operation example

- In this example, operations are performed according to the following contents of SampleFile.txt.

  NEW_DATA
  ADD_DATA1
  ADD_DATA2

- Read data and data size are as below.

  Data (STRING type): 'NEW_DATARNADD_DATA1RNADD_DATA2RNADD_DATA3'
  Data size: 41

- **Declaration section**

```
PROGRAM FileRead

VAR
    bResult         : BOOL := FALSE;        // Processing result (FALSE=normal , TRUE=abnormal)
    eOpenResult     : FILE.ERROR;           // File open result
    eReadResult     : FILE.ERROR;           // File read result
    eCloseResult    : FILE.ERROR;           // File close result
    sFileName       : STRING[80] := 'SampleDir/SampleFile.txt';
                                            // The name of the file to read
    szReadSize      : FILE.CAA.SIZE;        // Read data size
    sReadData       : STRING[256];          // Read data

    uiProcess       : UINT := 0;            // Process number(1=read mode)
    hFileHandle     : FILE.CAA.HANDLE;      // File handle
    iOpen           : FILE.Open;            // File open Function Block instance
    iRead           : FILE.Read;            // File read Function Block instance
    iClose          : FILE.Close;           // File Close Instance of FunctionBlock
END_VAR
```

Implementation section

```
CASE uiProcess OF
    1:  // File open processing started
        iOpen(  xExecute := TRUE ,                  // File open processing started
                eFileMode := FILE.MODE.MREAD ,      // Read mode
                sFileName := sFileName );           // file name
        uiProcess := 2;
    2:  // Waiting for file open processing to complete
        iOpen();                                    // Instance data update
        IF iOpen.xDone = TRUE THEN                  // File open completed normally
            hFileHandle := iOpen.hFile;             // Get file handle
            eOpenResult := iOpen.eError;            // File open processing result acquisition
            uiProcess := 10;                        // Transition to file read processing
            iOpen( xExecute := FALSE );             // File open processing finished
        ELSIF iOpen.xError = TRUE THEN              // File open processing error occurred
            eOpenResult := iOpen.eError;            // File open processing result acquisition
            uiProcess := 90;                        // Transition to end processing
            bResult := TRUE;                        // Abnormality
            iOpen( xExecute := FALSE );             // File open processing finished
        END_IF
    10: // File read processing started
        iRead(  xExecute := TRUE ,                  // File read processing started
                hFile := hFileHandle ,              // File handle obtained by opening a file
                pBuffer := ADR(sReadData) ,         // Data storage buffer address
                szBuffer := SIZEOF(sReadData) );    // Data storage buffer size
        uiProcess := 11;
    11: // Waiting for file read processing to complete
        iRead();                                    // Instance data update
        IF iRead.xDone = TRUE THEN                  // File read normal end
            eReadResult := iRead.eError;            // File read processing result acquisition
            szReadSize := iRead.szSize;             // Get read size
            uiProcess := 20;                        // Transition to file read processing
            iRead( xExecute := FALSE );             // File read processing completed
        ELSIF iRead.xError = TRUE THEN              // File read processing error occurred
            eReadResult := iRead.eError;            // File read processing result acquisition
            szReadSize := 0;                        // Read size initialization
            uiProcess := 20;                        // Transition to file close processing (to release the handle)
            bResult := TRUE;                        // Abnormality
            iRead( xExecute := FALSE );             // File read processing completed
        END_IF
    20: // File close processing started
        iClose( xExecute := TRUE ,                  // File close processing started
                hFile := hFileHandle );             // File handle obtained by opening a file
        uiProcess := 21;                            // Transition to waiting for file close processing completion
    21: // Waiting for file close processing to complete
        iClose();                                   // Instance data update
        IF iClose.xDone = TRUE  THEN                // File close processing completed
            eCloseResult := iClose.eError;          // File close processing result acquisition
            uiProcess := 90;                        // Transition to end processing
            iClose( xExecute := FALSE );            // File close processing completed
        ELSIF iClose.xError = TRUE THEN             // File close processing error occurred
            eCloseResult := iClose.eError;          // File close processing result acquisition
            uiProcess := 90;                        // Transition to end processing
            bResult := TRUE;                        // Abnormality
            iClose( xExecute := FALSE );            // File close processing completed
        END_IF
    90: // End processing
        uiProcess := 0;
END_CASE
```

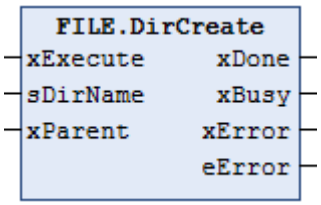
11.10 SD Card Operation (Directory Operation)

Directories in the SD card inserted in the SD memory card slot can be operated.

11.10.1 FILE.DirCreate (Create Directory)

This is a function block that creates a directory. An error occurs if there already exists a sub-directory.

■ **Icon**



■ **Parameter**

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	sDirName	FILE.CAA.FILENAME	Specifies a new directory name with an absolute path or relative path.
	xParent	BOOL	TRUE: Automatically creates a non-existing sub-directory. FALSE: An error occurs if there already exists a sub-directory.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".

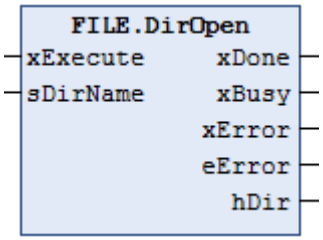
Info.

- You cannot use full size characters and the following symbols in a directory name: [\], [/], [:], [*], [?], [”], [<], [>], [|].

11.10.2 FILE.DirOpen (Open Directory)

This is a function block that opens a directory.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	sDirName	FILE.CAA.FILENAME	Specifies a directory name with an absolute path or relative path.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".
	hDir	FILE.CAA.HANDLE	Handle of the FILE.CAA.HANDLE directory

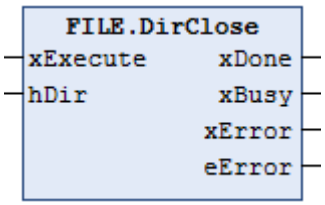
Info.

- You cannot use full size characters and the following symbols in a directory name: [\], [/], [:], [*], [?], [”], [<], [>], [|].

11.10.3 FILE.DirClose (Close Directory)

This is a function block that closes a directory.

■ Icon



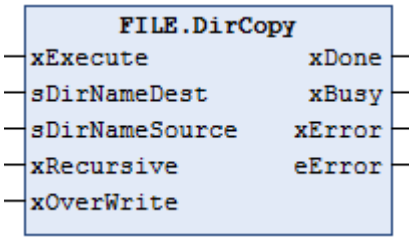
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	hFile	FILE.CAA.FILENAME	Handle of the directory to be closed Specifies the handle output by FILE.Open.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".

11.10.4 FILE.DirCopy (Copy Directory)

This is a function block that copies a directory.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	sDirNameDest	FILE.CAA.FILENAME	Directory name of the copy destination
	sDirNameSource	FILE.CAA.FILENAME	Directory of the copy source
	xRecursive	BOOL	TRUE: Copies the sub-directory and files.
	xOverWrite	BOOL	TRUE: Copies to overwrite an existing file.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".

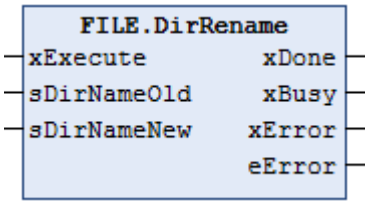
Info.

- You cannot use full size characters and the following symbols in a directory name: [\], [/], [:], [*], [?], [”], [<], [>], [|].

11.10.5 FILE.DirRename (Rename Directory)

This is a function block that renames a directory name. It is not possible to change the directory name of a directory that is currently open. Close it using the DirClose function block.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	sDirNameOld	FILE.CAA.FILENAME	Directory name before change
	sDirNameNew	FILE.CAA.FILENAME	Directory name after change
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".

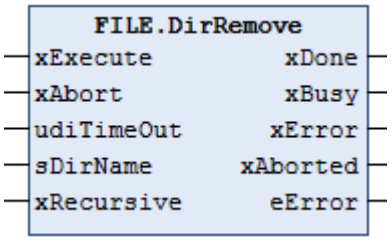
Info.

- You cannot use full size characters and the following symbols in a directory name: [\], [/], [:], [*], [?], [”], [<], [>], [|].

11.10.6 FILE.DirRemove (Delete Directory)

This is a function block that deletes a directory. It is not possible to delete a directory that is currently open. Close it using the DirClose function block.

■ Icon



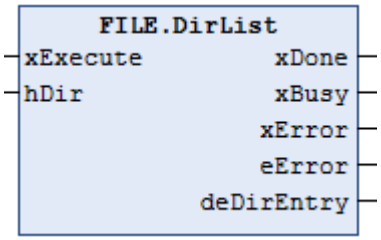
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	xAbort	BOOL	TRUE: Stops execution and resets all outputs.
	udiTimeOut	UDINT	Timeout time until the execution is stopped (μs)
	sDirName	FILE.CAA.FILENAME	Specifies a directory name with an absolute path or relative path.
	xRecursive	BOOL	TRUE: Deletes the sub-directory and all files. FALSE: Deletes only when the directory is empty. An error occurs if the directory is not empty.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	xAborted	BOOL	TRUE: Execution is stopped by the user.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".

11.10.7 FILE.DirList (Directory List)

This is a function block that outputs a list of directories and files inside the directory.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
	hDir	FILE.CAA.HANDLE	Directory from which to output a list Specifies the handle output by FILE.Open.
Output	xDone	BOOL	TRUE: Execution is completed.
	xBusy	BOOL	TRUE: Execution of the FB is not completed.
	xError	BOOL	TRUE: An error has occurred within the FB.
	eError	FILE.ERROR	An error ID is output. Refer to "11.9.15 FILE.ERROR (Error ID)".
	deDirEntry	FILE.FILE_DIR_ENTRY	Files and directories are output.

■ FILE_DIR_ENTRY (Structure)

Member	Type	Description
sEntry	FILE.CAA.FILENAME	Directory or file name
szSize	FILE.CAA.SIZE	File size
xDirectory	BOOL	TRUE: Directory FALSE: File
xExclusive	BOOL	TRUE: Exclusive access mode FALSE: Multiple access mode
dtLastModification	DATE_AND_TIME	Last update date and time.

11.11 SD Card Operation (CSV File Operation)

CSV files in the SD card inserted in the SD memory card slot can be operated (reading, writing).

11.11.1 Overview of CSV File Reading

With the GM1 controller, through use of function blocks, data can be read from a CSV file on the SD card. A procedure for reading data from 'ReadData.csv' in the 'Sample' folder on the SD card will be described.

- Folder structure

  SD card

  |--Sample

  | |--ReadData.csv (target CSV file to be read)

- File contents

  1,2,3,4,5,6,7,8,9,10\r\n

  11,12,13,14,15,16,17,18,19,20\r\n

  21,22,23,24,25,26,27,28,29,30\r\n

Procedure

1. Specify a target CSV file to be read.

   By executing CSV.CSVReaderInit, specify the name of a CSV file from which data is read, as well as a data separator.

2. Read data from the CSV file.

   There are three types of methods used to read data.

   - Read all data by batch.

     Execute CSV.ReadAll to read all data from the CSV file in array form by a single run of execution.
     Execution result

     asElement[0..29] =
     ['1','2','3','4','5','6','7','8','9','10','11','12',...,'22','23','24','25','26','27','28','29','30']

   - Read data element by element.

     Execute CSV.NextElement to read only one element in order from the beginning of the data at every run of execution.
     It is necessary to execute CSV.NextElement 30 times to read all data from a CSV file like this example.

     Execution result

     1st run of execution: sElement = '1'

     2nd run of execution: sElement = '2'

     ...

     29th run of execution: sElement = '29'

     30th run of execution: sElement = '30'

   - Read data line by line.

     Execute CSV.NextLine to read only one line in order from the beginning of the data at every run of execution.

Execution result
1st run of execution: sLine = ‘1,2,3,4,5,6,7,8,9,10’
2nd run of execution: sLine = ‘11,12,13,14,15,16,17,18,19,20’
3rd run of execution: sLine = ‘21,22,23,24,25,26,27,28,29,30’

Info.

- For an example of the process for reading all data by batch, refer to "11.11.15 Example of Process for Reading All Data from CSV File".
- For an example of the process for changing a target CSV file to be read, refer to "11.11.16 Example of Process for Reading Data from Multiple CSV Files".

11.11.2 CSV.CSVReaderInit (Specify Target CSV File To Be Read)

This is a function block that specifies a CSV file from which data is read. Separator settings can also be configured.

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	xExecute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	szFileName	STRING	''	Specifies the file name with an absolute path.(Note 1)
	sLineSeparator	STRING(2)	'RN'	Specifies a line separator.(Note 2) (Note 3) • 'RN': CR+LF • '$R': CR • '$N': LF
	sColumnSeparator	STRING(1)	';'	Specifies a column separator.(Note 2)
	xCloseFile	BOOL	FALSE	TRUE : Close the file Please confirm xDone is TRUE and change it to FALSE.
Output	xDone	BOOL	FALSE	TRUE: Execution of the FB is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.

Scope	Name	Type	Default value	Description
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	CSV.CSV_ERROR	NO_ERROR	An error ID is output. Refer to"11.11.6 CSV.CSV_ERROR (Reading Error Code)".
	hFile	SysFile.RTS_IEC_HANDLE	16#00000000	Handle of a file

(Note 1) Add the CSV extension ('.csv') at the end of the file name and specify up to 80 characters.

(Note 2) Set the argument to a line or column separator used in the CSV file. If the setting is not correct, data cannot be read.

(Note 3) 'RN' is handled as two characters, and '$R' and '$N' are each handled as one character.

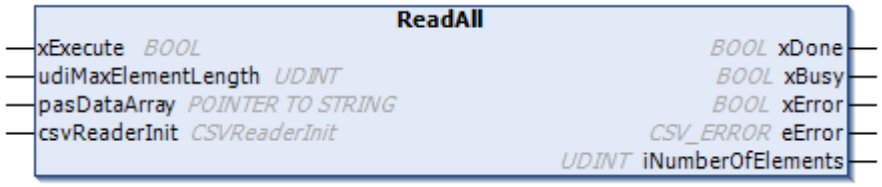
Info.

- You cannot use full size characters and the following symbols in a file name: [\], [/], [:], [*], [?], ["], [<], [>], [|].
- To use ReadAll, NextElement, or NextLine, execute CSVReaderinit in advance.
- If you specify a CSV file in which a line separator is not used at the end of data, the data cannot be properly read by ReadAll, NextElement, or NextLine.
- If you specify a CSV file in which several types of line or column separators are written in data, the data cannot be properly read by ReadAll, NextElement, or NextLine.

11.11.3 CSV.ReadAll (Read All File Data by Batch)

This is a function block that reads all data from a CSV file. All data can be read according to separators by a single run of execution.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	xExecute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	udiMaxElementLength	UDINT	80	Specify the maximum length of one element.(Note 1)
	pasDataArray	POINTER TO ARRAY OF STRING	-	Pointer to the buffer that stores read data(Note 2)
Output	xDone	BOOL	FALSE	TRUE: Execution of the FB is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	CSV.CSV_ERROR	NO_ERROR	An error ID is output. Refer to "11.11.6 CSV.CSV_ERROR (Reading Error Code)".
	iNumberOfElements	UDINT	0	The number of read elements
Input / output	csvReaderInit	CSVReaderInit	-	Reference to CSVReaderInit

(Note 1) Set the argument to a value that is greater than the maximum number of characters in each read element and less than or equal to 32000.

If the setting is outside the range, data cannot be properly read.

(Note 2) Set the number of elements in the array to a value greater than or equal to the total number of elements of the data to be read.

Set the STRING type memory size to a value equal to the udiMaxElementLength value.

Info.

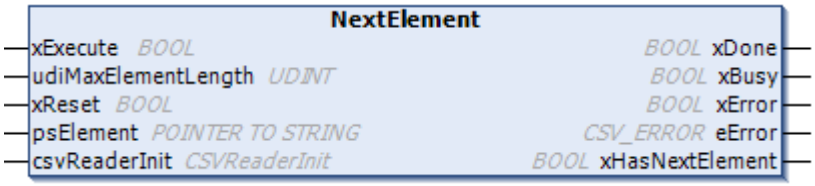
- To use ReadAll, execute CSVReaderInit in advance.
- Set the input argument csvReaderInit to CSVReaderInit, which is executed to specify the target CSV file to be read.
- The maximum number of characters of one element that can be read by ReadAll is 32000.

  Data cannot be properly read if one element exceeds 32000 characters, and thus do not use such data.
- If you specify 0 (NULL) for the pointer （pasDataArray）, the function does not operate properly, and thus do not specify so.

11.11.4 CSV.NextElement (Read One Element)

This is a function block that reads one element from a CSV file. Data can be read one element by one element in order from the beginning of the data at every run of execution.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	xExecute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	udiMaxElementLength	UDINT	255	Specify the maximum length of one element.(Note 1)
	xReset	BOOL	FALSE	While it is TRUE, the read position is reset to the beginning of data.
	psElement	POINTER TO STRING	-	Pointer to the buffer that stores read data(Note 2)
Output	xDone	BOOL	FALSE	TRUE: Execution of the FB is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	CSV.CSV_ERROR	NO_ERROR	An error ID is output. Refer to "11.11.6 CSV.CSV_ERROR (Reading Error Code)".
	xHasNextElement	BOOL	FALSE	TRUE: One element to be read next is present. FALSE: One element to be read next is not present.
Input / output	csvReaderInit	CSVReaderInit	-	Reference to CSVReaderInit

(Note 1) Set the argument to a value that is greater than the maximum number of characters in one read element and less than or equal to 32000.

If the setting is outside the range, data cannot be properly read.

(Note 2) Set the STRING type memory size to a value equal to the udiMaxElementLength value.

Info.

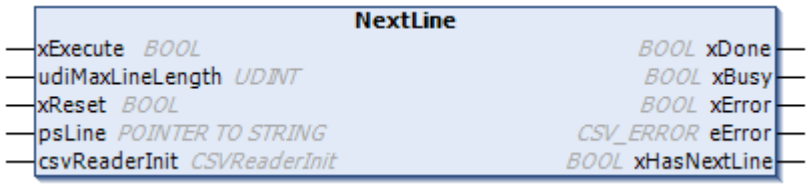
- To execute NextElement, execute CSVReaderInit in advance.
- Set the input argument csvReaderInit to CSVReaderInit, which is executed to specify the target CSV file to be read.
- The maximum number of characters that can be read for a single element using NextElement is 32,000 characters.

  If a single element exceeds 32,000 characters, it cannot be read correctly using NextElement. Please refrain from using it in such cases.
- If you specify 0 (NULL) for the pointer (psElement), the function does not operate properly, and thus do not specify so.

11.11.5 CSV.NextLine (Read One Line)

This is a function block that reads one line from a CSV file. Data can be read one line by one line in order from the beginning of the data at every run of execution.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	xExecute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	udiMaxLineLength	UDINT	255	Specify the maximum length of one line.(Note 1)
	xReset	BOOL	FALSE	While it is TRUE, the read position is reset to the beginning of data.
	psLine	POINTER TO STRING	-	Pointer to the buffer that stores read data(Note 2)
Output	xDone	BOOL	FALSE	TRUE: Execution of the FB is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	CSV.CSV_ERROR	NO_ERROR	An error ID is output. Refer to "11.11.6 CSV.CSV_ERROR (Reading Error Code)".
	xHasNextLine	BOOL	FALSE	TRUE: One line to be read next is present.

Scope	Name	Type	Default value	Description
				FALSE: One line to be read next is not present.
Input / output	csvReaderInit	CSVReaderInit	-	Reference to CSVReaderInit

(Note 1) Set the argument to a value that is greater than the maximum number of characters in one read line and less than or equal to 32000.

If the setting is outside the range, data cannot be properly read.

(Note 2) Set the STRING type memory size to a value equal to the udiMaxLineLength value.

Info.

- To execute NextLine, execute CSVReaderInit in advance.
- Set the input argument csvReaderInit to CSVReaderInit, which is executed to specify the target CSV file to be read.
- The maximum number of characters of one line that can be read by NextLine is 32000.

  Data cannot be properly read if one line exceeds 32000 characters, and thus do not use such data.
- If you specify 0 (NULL) for the pointer (psLine), the function does not operate properly, and thus do not specify so.

11.11.6 CSV.CSV_ERROR (Reading Error Code)

This is an enumeration type error code that is output when the function block for reading a CSV file is executed.

■ CSV.CSV_ERROR (Enumeration type)

Name	Value	Description
NO_ERROR	0	No error
TIME_OUT	2	Timeout
CANNOT_SET_POSITION	10	The delimiter position could not be set because the column/row delimiter character is either the same string or NULL.
END_OF_BUFFER	12	The maximum limit of the internal buffer size has been reached.
INVALID_HANDLE	17	Invalid handle
MAXIMUM_ELEMENT_SIZE_EXCEEDED	18	The read data has exceeded the maximum data length.
INVALID_POINTER	19	Invalid pointer

11.11.7 Overview of CSV File Writing

With the GM1 controller, through use of function blocks, data can be written to a CSV file on the SD card.

A procedure for adding data to ‘WriteData_Init1.csv’ and ‘WriteData_Init2.csv’ on a SD card, as well as writing and saving data to ‘WriteData_New.csv’, a new CSV file, will be described. ‘WriteData_Init1.csv’ is present in ‘Sample1’, an existing folder, and ‘WriteData_Init2.csv’ is present in ‘Sample2’, another existing folder.

- Folder structure

  SD card
  |--Sample1 (existing folder)
  | |--WriteData_Init1.csv (existing CSV file specified in step 1))
  | |--WriteData_New.csv (new CSV file saved in step 4)
  |--Sample2 (existing folder)
  | |--WriteData_Init2.csv (existing CSV file specified in step 6))

1 2 Procedure

1. Specify a target CSV file to write.
   By executing CSV.Init, specify the name of a CSV file to which data is written, as well as the name of a directory where the file is located, and a data separator.

2. Add the data to be written to an internal buffer.
   Before the data is written to the CSV file, it is necessary to add the data to an internal storage area (internal buffer) once. The added data remains added to the internal buffer until CSV.WriteFile is executed in step 3.

- Adding the data to the internal buffer

  The CSV.Add'Type' function block by which data can be added includes 20 types (refer to "11.11.9 CSV.Add'Type' (Add Data to Internal Buffer)" for supported types). Execute CSV.Add'Type' to add specified values to the internal buffer. A column separator specified in step 1 is automatically added at every run of execution.
- Adding a line feed code to the internal buffer

  Execute CSV.NewLine to add a line separator specified in step 1 to the internal buffer.

3. Write to the CSV file.

Execute CSV.WriteFile to write the data added to the internal buffer (the data added in step 2) to the CSV file.

4. Change the target to write to a new CSV file.

Execute CSV.NewFile to change the target to write to a new CSV file. Perform steps 2 and 3 in the similar way to write the data to the new CSV file and save it in the directory specified by CSV.Init in step 1.

At this time, if the file name is set to the name of an existing CSV file, the new CSV file is saved by overwriting the existing CSV file.

5. Add the data to the end of data in an existing CSV file.

After setting the file name to the name of the existing CSV file and executing CSV.Init, perform steps 2 and 3 to add the data to the specified CSV file.

6. Change the directory

By changing a folder path and then executing CSV.Init, you can change the directory ('Sample1') specified in step 1 to another directory ('Sample2'). Perform steps 2 and 3 in the similar way to add the data to the specified CSV file.

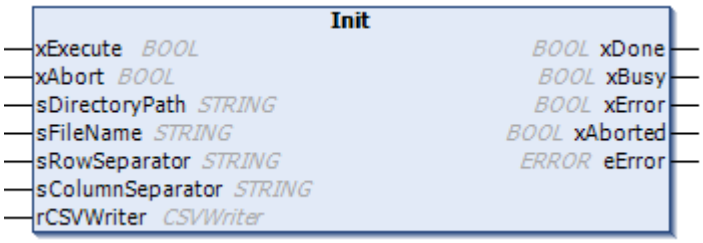
Info.

- If the file name specified by CSV.Init is set to NULL, an error occurs. However, when you specify NULL for the file name specified by CSV.NewFile, an error does not occur and a number is added to the name of the CSV file that is the target to write before the execution of CSV.NewFile. For details, refer to "11.11.12 CSV.NewFile (Change Target To Write to New CSV File)".
- If you specify a directory path to the file name specified by either of CSV.Init and CSV.NewFile, a new folder is created in the directory specified by CSV.Init, and a new CSV file is created/saved in the new folder.
- To change the target CSV file to write by the execution of CSV.Init or CSV.NewFile, execute CSV.WriteFile beforehand. For details, refer to "11.11.11 CSV.WriteFile (Write, Save Data to CSV File)".
- For an example of the process for writing data to a CSV file, refer to "11.11.17 Example of Process for Writing Log Data to CSV File".

11.11.8 CSV.Init (Specify Target CSV File To Write)

This is a function block that specifies a CSV file to which data is written. If the specified CSV file does not exist, a new file is created. Separator settings can also be configured.Execute the function block by UserTask.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	xExecute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	sDirectoryPath	STRING	''	Specifies a directory name.(Note 1)
	sFileName	STRING	''	Specifies a file name with a relative path.(Note 2)
	sRowSeparator	STRING	'RN'	Specifies a line separator.(Note 3) ● 'RN': CR+LF ● '$R': CR ● '$N': LF
	sColumnSeparator	STRING	';'	Specifies a column separator.(Note 4)
	xAbort	BOOL	FALSE	TRUE: Execution of the FB is aborted.
Output	xDone	BOOL	FALSE	TRUE: Execution of the FB is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	CSV.ERROR	NO_ERROR	An error ID is output. Refer to "11.11.14 CSV.ERROR (Writing Error Code)".
	xAborted	BOOL	FALSE	TRUE: Execution is aborted by the user's xAbort input.
Input / output	rCSVWriter	CSVWriter	-	"11.11.13 CSV.CSVWriter"

(Note 1) The directory name must be up to 80 characters.
To specify the root directory, specify '.' or './'.

(Note 2) Set the file name by specifying a path relative to the directory specified in the input argument sDirectoryPath. Add the CSV extension ('.csv') at the end of the file name and specify up to 80 characters.

If the specified directory path does not exist, a directory is created. A new CSV file is created in the directory.

(Note 3) 'RN' is handled as two characters, and '$R' and '$N' are each handled as one character.

(Note 4) To read the CSV file using CSVReaderInit, set the parameter to one character.

Info.

- You cannot use full size characters and the following symbols in directory and file names: [\], [/], [:], [*], [?], ["], [<], [>], [|].
- To use WriteFile, Add'Type', NewLine or NewFile, execute CSV.Init in advance.
- To specify another CSV file, execute WriteFile and then execute CSV.Init again.

11.11.9 CSV.Add'Type' (Add Data to Internal Buffer)

This is a function block that adds input data to an internal buffer. The table below shows supported data types.

■ Add'Type' categories

Category	Type	Default value
Truth	BOOL	FALSE
Integer	BYTE/WORD/DWORD/LWORD/SINT/USINT/INT/UINT/DINT/ UDINT/LINT/ULINT	0
Floating-point number	REAL（Please do not use LREAL.）	0
Character string	STRING	''
Time	TIME/LTIME/DATE/TIME_OF_DAY/DATE_AND_TIME	Minimum value of each data type

■ Icon (description of only AddSTRING)

■ Parameter

Scope	Name	Type	Default value	Description
Input	xExecute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	**Input(Note 1)	Refer to the table above.	Refer to the table above.	Specifies data added to the internal buffer.
	xAbort	BOOL	FALSE	TRUE: Execution of the FB is aborted.
Output	xDone	BOOL	FALSE	TRUE: Execution of the FB is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	CSV.ERROR	NO_ERROR	An error ID is output. Refer to"11.11.14 CSV.ERROR (Writing Error Code)".
	xAborted	BOOL	FALSE	TRUE: Execution is aborted by the user's xAbort input.
Input / output	rCSVWriter	CSVWriter	-	"11.11.13 CSV.CSVWriter"

(Note 1) **: A prefix that denotes a data type is added to the beginning of the argument.

Info.

- This FB absolutely requires arguments of CSVWriter and thus can be used only after the execution of CSV.Init.
- Up to 5000 characters can be added to the internal buffer.

  The FB does not properly operate if the added characters exceeds 5000 characters, and thus do not use such data.

11.11.10 CSV.NewLine (Add Line Separator to Internal Buffer)

This is a function block that adds a line separator to an internal buffer. The line separator is specified in the input argument sRowSeparator of CSV.Init.

■ Icon

■ Parameter

Scope	Name	Type	Default value	Description
Input	xExecute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	xAbort	BOOL	FALSE	TRUE: Execution of the FB is aborted.
Output	xDone	BOOL	FALSE	TRUE: Execution of the FB is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	CSV.ERROR	NO_ERROR	An error ID is output. Refer to "11.11.14 CSV.ERROR (Writing Error Code)".
	xAborted	BOOL	FALSE	TRUE: Execution is aborted by the user's xAbort input.
Input / output	rCSVWriter	CSVWriter	-	"11.11.13 CSV.CSVWriter"

Info.

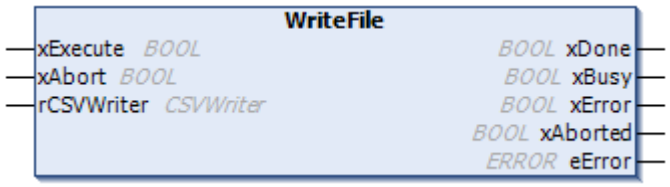
- This FB absolutely requires arguments of CSVWriter and thus can be used only after the execution of CSV.Init.
- If you write data to another CSV file by CSV.Init or NewFile, add a line feed code by executing NewLine before WriteFile. If no line feed is added, the data cannot be properly read from the CSV file through reading.
- Up to 5000 characters can be added to the internal buffer.

  The FB does not properly operate if the added characters exceeds 5000 characters, and thus do not use such data.

11.11.11 CSV.WriteFile (Write, Save Data to CSV File)

This is a function block that writes data added to an internal buffer to a CSV file and saves the data. With the execution of WriteFile, the data added to the internal buffer is written to the CSV file and thus is reset.Execute the function block by UserTask.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	xExecute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	xAbort	BOOL	FALSE	TRUE: Execution of the FB is aborted.
Output	xDone	BOOL	FALSE	TRUE: Execution of the FB is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	CSV.ERROR	NO_ERROR	An error ID is output. Refer to "11.11.14 CSV.ERROR (Writing Error Code)".
	xAborted	BOOL	FALSE	TRUE: Execution is aborted by the user's xAbort input.
Input / output	rCSVWriter	CSVWriter	-	"11.11.13 CSV.CSVWriter"

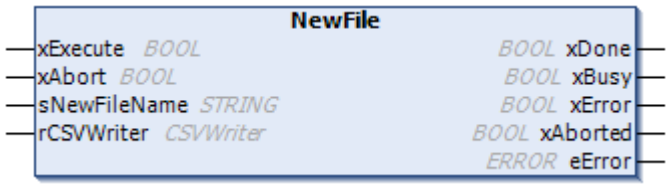
Info.

- If you write data to another CSV file by CSV.Init or NewFile, reset the internal buffer by executing WriteFile before executing CSV.Init or NewFile. If the internal buffer is not reset, the data cannot be properly written to the specified CSV file.
- If SD card write protection is enabled, an error occurs, preventing data from being written and a new file from being created.
- If there is no free space on the SD card, an error occurs and a new blank file is created.
- If the number of files saved to the directory on the SD card has reached the upper limit, creating a new file in the directory results in improper operation, and thus do not use it.

11.11.12 CSV.NewFile (Change Target To Write to New CSV File)

This is a function block that changes the target to write to a new CSV file.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	xExecute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	sNewFileName	STRING	''	Specifies a file name with a relative path.(Note 1)
	xAbort	BOOL	FALSE	TRUE: Execution of the FB is aborted.
Output	xDone	BOOL	FALSE	TRUE: Execution of the FB is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	CSV.ERROR	NO_ERROR	An error ID is output. Refer to "11.11.14 CSV.ERROR (Writing Error Code)".
	xAborted	BOOL	FALSE	TRUE: Execution is aborted by the user's xAbort input.
Input / output	rCSVWriter	CSVWriter	-	"11.11.13 CSV.CSVWriter"

(Note 1) Please set the relative path from the directory specified in the input argument sDirectoryPath of CSV.Init. Ensure that the length of the relative path, including the directory name specified in the input argument sDirectoryPath, the path separator ('\'), and the CSV extension ('.csv'), is set to be less than 80 characters.

If the specified directory path does not exist, a directory will be created. Within it, the CSV file will be saved.

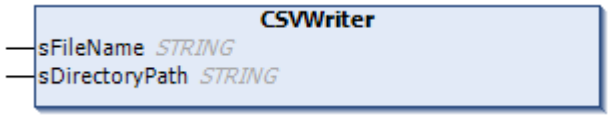
Info.

- This FB absolutely requires arguments of CSVWriter and thus can be used only after the execution of CSV.Init.
- Full-width characters and some symbols (\ / : * ? " < > |) cannot be used in the file name.
  If a NULL or unusable symbol is used in the file name, an error will occur and the value 21 will be output to eError.
- A new CSV file is not created only through the execution of NewFile, but is created through the execution of WriteFile.
- If the file name is set to the name of an existing CSV file, the new CSV file is saved by overwriting the existing CSV file.

11.11.13 CSV.CSVWriter

This is a function block used in the input / output parameters of CSV.Init, Add'Type', NewLine, NewFile, and WriteFile. Use this FB with the default values left unchanged.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	sFileName	STRING	'dataFileSeq.csv'	-
	sDirectoryPath	STRING	'C:/temp'	-

11.11.14 CSV.ERROR (Writing Error Code)

This is an enumeration type error code that is output when the function block for writing a CSV file is executed.

■ CSV.ERROR (Enumeration type)

Name	Value	Description
NO_ERROR	0	No error
CANNOT_OPEN_FILE	7	A file cannot be created.
CANNOT_WRITE_DATA	9	Data cannot be written to the file.
END_OF_BUFFER	10	The number has reached the upper limit on the internal buffer size.
CANNOT_GET_FILE_SIZE	14	The file cannot be found.

11.11.15 Example of Process for Reading All Data from CSV File

An example program for specifying a CSV file on the SD card and reading all data from the CSV file will be described below.

- Implementation Example

  Reading all data from 'Data.csv', which is in the root directory on the SD card, by batch.
  - Folder structure

    SD card

    |--Data.csv
  - File contents

    Apple,4,600\r\n

    Orange,2,250\r\n

  When execution of the program is completed, all the data is read to the asElement array.

  asElement[0..5]=['Apple','4','600','Orange','2','250']
- Description of process

  When the case number (iStep) is changed to 1, a reading process is executed. Details of the process for each case number in the implementation section are as described below.

  1. Execute CSV.CSVReaderInit to specify a CSV file from which data is read.
  2. Execute ReadAll to read all data.

     When reading of all the data is completed, the variable xFinish goes TRUE.
- Declaration section

```
VAR
// Start,Finish Flag
iStep                   : INT := 0;         //Change to 1:Start
xFinish                 : BOOL := FALSE;    //TRUE:Finish

// FB instance
CSVReaderInit_0         : CSV.CSVReaderInit;
ReadAll_0               : CSV.ReadAll;

// Variables
asElement               : ARRAY [0..999] OF STRING(255);    //output
sFileName               : STRING := 'Data.csv';
sColumnSep              : STRING := ',';
sLineSep                : STRING := '$R$N';
xExecuteCSVReaderInit   : BOOL := FALSE;
xExecuteReadAll         : BOOL := FALSE;
END_VAR
```

- Implementation section

```
//FunctionBlock
CSVReaderInit_0( xExecute           := xExecuteCSVReaderInit,
                 szFileName         := sFileName,
                 sColumnSeparator   := sColumnSep,
                 sLineSeparator     := sLineSep );

ReadAll_0( xExecute             := xExecuteReadAll,
           udiMaxElementLength  := 255,
```

```
            pasDataArray          := ADR(asElement),
            csvReaderInit         := CSVReaderInit_0 );

CASE iStep OF
    1: // Open CSVFile
        xFinish                   := FALSE;
        xExecuteCSVReaderInit     := TRUE;
        IF ( CSVReaderInit_0.xDone = TRUE ) THEN
            xExecuteCSVReaderInit   := FALSE;
            iStep                   := 2;
        END_IF

    2: // Read All Data
        xExecuteReadAll := TRUE;
        IF ( ReadAll_0.xDone = TRUE ) THEN
            xExecuteReadAll := FALSE;
            xFinish         := TRUE;
            iStep           := 0;
        END_IF
END_CASE
```

11.11.16 Example of Process for Reading Data from Multiple CSV Files

The following is an example of a program that reads all the data of two CSV files on an SD card one element at a time, starting with the first.

- Implementation Example

  Reading all data from ‘Data.csv’ and ‘Data1.csv’ in the root directory on the SD card one element by one element.

  - Folder structure

    SD card

    |--Data.csv

    |--Data1.csv

  - File contents

    Contents of Data.csv

    Apple,4,600\r\n

    Orange,2,250\r\n

    Contents of Data1.csv

    Cherry,6,300\r\n

    Tomato,8,500\r\n

  When execution of the program is completed, all the data is read to the asElement array.

  asElement[0..11]=['Apple','4','600','Orange','2','250','Cherry','6','300','Tomato','8','500']

- Processing content

  When the case number (byStep) is changed to 1, a process is executed. Details of the process for each case number in the implementation section are as described below.

  1. Execute CSVReaderInit to specify the CSV file (sFileName) from which data is read first.
  2. Execute NextElement to read one element.
  3. When there is another element that can be read, return to the case number 2.

     When all elements are read, proceed to the case number 4.

4. Execute CSVReaderInit to specify the CSV file (sNewFileName) from which data is read second.
5. Execute NextElement to read one element.
6. If there are readable elements, it will return to case number 5.
   If all elements have been read, the processing will be completed and xFinish will become TRUE.

- Declaration section

```
VAR
// Start,Finish Flag
iStep                   : INT := 0;         //Change to 1:Start
xFinish                 : BOOL := FALSE;    //TRUE:Finish

// FB instance
CSVReaderInit_0         : CSV.CSVReaderInit;
NextElement_0           : CSV.NextElement;

// Variables
asElement               : ARRAY [0..999] OF STRING(255);    //output
sFileName               : STRING := 'Data.csv'; // First Read File
sNewFileName            : STRING := 'Data1.csv'; // Second Read File
sColumnSep              : STRING := ',';
sLineSep                : STRING := '$R$N';
xExecuteCSVReaderInit   : BOOL := FALSE;
xExecuteNextElement     : BOOL := FALSE;
iCount_item             : INT := 0;
END_VAR
```

- Implementation section

```
//FunctionBlock
CSVReaderInit_0( xExecute           := xExecuteCSVReaderInit,
                 szFileName         := sFileName,
                 sColumnSeparator   := sColumnSep,
                 sLineSeparator     := sLineSep );

NextElement_0( xExecute             := xExecuteNextElement,
               udiMaxElementLength  := 255,
               psElement            := ADR(asElement[iCount_item]),
               csvReaderInit        := CSVReaderInit_0 );

CASE iStep OF
    1: // Open First CSVFile
        xFinish                 := FALSE;
        xExecuteCSVReaderInit   := TRUE;
        IF ( CSVReaderInit_0.xDone = TRUE ) THEN
            xExecuteCSVReaderInit   := FALSE;
            iStep                   := 2;
        END_IF

    2: // Read the First CSVFile
        xExecuteNextElement := TRUE;
        IF ( NextElement_0.xDone = TRUE ) THEN
            iCount_item         := iCount_item + 1;
            xExecuteNextElement := FALSE;
```

```
            iStep               := 3;
        END_IF

    3: // Repeat until all Elements are read (First CSVFile)
        IF ( NextElement_0.xHasNextElement = TRUE ) THEN
            iStep    := 2;
        ELSE
            iStep    := 4;
        END_IF

    4: // Open Second CSVFile
        sFileName               := sNewFileName;
        xExecuteCSVReaderInit   := TRUE;
        IF ( CSVReaderInit_0.xDone = TRUE ) THEN
            xExecuteCSVReaderInit   := FALSE;
            iStep                   := 5;
        END_IF

    5: // Read the Second CSVFile
        xExecuteNextElement := TRUE;
        IF ( NextElement_0.xDone = TRUE ) THEN
            iCount_item         := iCount_item + 1;
            xExecuteNextElement := FALSE;
            iStep               := 6;
        END_IF

    6: // Repeat until all Elements are read (Second CSVFile)
        IF ( NextElement_0.xHasNextElement = TRUE ) THEN
            iStep               := 5;
        ELSE
            xFinish             := TRUE;
            iStep               := 0;
        END_IF
END_CASE
```

11.11.17 Example of Process for Writing Log Data to CSV File

An example program for writing log data to a CSV file on the SD card will be described below.

The example shows a program used to output a log of parameters of an axis (Axis1) in operation. When log out is performed one time, 10 elements, “log number, local time, parameter 1, parameter 2,..., parameter 8”, are output as a log. An interval at which a log is output and the number of outputs are set by the variable tSample_interval and the variable iNumber_of_Times, respectively. The name of a CSV file to which logs are output is ‘LogData_**.csv’ (**: The number of execution times).

- Implementation Example

  The program is executed at a total of two times, and logs are output to two CSV files (‘LogData_1.csv’ and ‘LogData_2.csv’). For instance, 10 logs are output at one-second intervals each time.

  - Folder structure

    SD card

    |--LogData_1.csv (CSV file created at first execution)

    |--LogData_2.csv (CSV file created at second execution)

  When the program execution is completed two times in total, data is written to ‘LogData_1.csv’ and ‘LogData_2.csv’ as shown below.

  Contents of LogData_1.csv

  1,DT#2000-01-01-01:01:01,10,1,0,0,10,1,0,0\r\n

  2,DT#2000-01-01-01:01:02,20,5,0,0,20,5,0,0\r\n

  ...

  10,DT#2000-01-01-01:01:10,100,5,0,0,100,5,0,0\r\n

  Contents of LogData_2.csv

  1,DT#2000-01-01-01:02:01,110,5,0,0,110,5,0,0\r\n

  2,DT#2000-01-01-01:02:02,120,5,0,0,120,5,0,0\r\n

  ...

  10,DT#2000-01-01-01:02:10,200,1,0,0,200,1,0,0\r\n

- Description of process

  When the case number (iStep) is changed to 1, a writing process is executed. Details of the process for each case number in the implementation section are as described below.

  1. For first execution, execute CSV.Init to create a new CSV file.

     For execution at second and succeeding times, execute NewFile to create a new CSV file.
  2. Execute AddSTRING to add data output as a log to the internal buffer.
  3. Continually return to the case number 2 until all the specified elements are added.

     When all the specified elements are added, proceed to the case number 4.
  4. Execute NewLine to add a line feed code to the internal buffer.
  5. Execute WriteFile to write the data added to the internal buffer to the CSV file.
  6. Continually return to the case number 2 until the specified number of logs are written.

     When the specified number of logs are written, proceed to the case number 7.
  7. At the completion of log data writing, xFinish = TRUE.

- Declaration section

```
VAR
// Start,Finish Flag
iStep               : INT := 0;         //Change to 1:Start
xFinish             : BOOL := FALSE;    //TRUE:Finish

// FB instance
BLINK_0             : BLINK;
R_TRIG_0            : R_TRIG;
CTU_0,CTU_1         : CTU;
CSVWriter_0         : CSV.CSVWriter;
Init_0              : CSV.Init;
AddSTRING_0         : CSV.AddSTRING;
NewLine_0           : CSV.NewLine;
WriteFile_0         : CSV.WriteFile;
NewFile_0           : CSV.NewFile;

// Variables
sFileName           : STRING := 'LogData';  // Base of FileName
sDirectoryPath      : STRING := '.';        // Directory path
sLineSep            : STRING := '$R$N';
sColumnSep          : STRING := ',';
uliTimeZone         : LINT := 9;    // Time zone at UTC
iNumber_of_Elem     : INT := 10;    // The number of Write Data
iNumber_of_Times    : INT := 10;    // The number of times to get the data
tSample_interval    : TIME := T#1S; // Time Interval (1 second or more)

xCTURESET_0         : BOOL := FALSE;
xCTURESET_1         : BOOL := FALSE;
xExecuteInit        : BOOL := FALSE;
xExecuteSTRING      : BOOL := FALSE;
xExecuteNewLine     : BOOL := FALSE;
xExecuteWriteFile   : BOOL := FALSE;
xExecuteNewFile     : BOOL := FALSE;
xStart              : BOOL := FALSE;
iNumber_of_xStart   : INT := 0;
aData               : ARRAY [0..9] OF STRING;
iCount_Data         : INT := 0;
sSerialNumber       : STRING;
sCreateFileName     : STRING;
END_VAR
```

- Implementation section

```
//Stored Data
IF ( R_TRIG_0.Q = TRUE ) AND ( iCount_Data < iNumber_of_Times ) THEN
    aData[0] := TO_STRING(iCount_Data + 1);

    aData[1] := TO_STRING(TO_DT(( GetDateTime() + uliTimeZone*60*60*1000 )
/ 1000));
    aData[2] := TO_STRING(Axis1.fSetPosition);
    aData[3] := TO_STRING(Axis1.fSetVelocity);
    aData[4] := TO_STRING(Axis1.fSetAcceleration);
    aData[5] := TO_STRING(Axis1.fSetJerk);
    aData[6] := TO_STRING(Axis1.fActPosition);
    aData[7] := TO_STRING(Axis1.fActVelocity);
    aData[8] := TO_STRING(Axis1.fActAcceleration);
```

```
    aData[9] := TO_STRING(Axis1.fActJerk);
    iCount_Data := iCount_Data + 1;
END_IF

//Create FileName
sSerialNumber   := CONCAT( STR1 := '_' ,
                           STR2 := TO_STRING( iNumber_of_xStart + 1) );

sCreateFileName := CONCAT( STR1 := sFileName ,
                           STR2 := CONCAT( STR1 := sSerialNumber ,
                                           STR2 := '.csv') );

//FunctionBlock
BLINK_0( ENABLE     := xStart ,
         TIMEHIGH   := T#1MS ,
         TIMELOW    := tSample_interval - T#1MS );

R_TRIG_0( CLK   := BLINK_0.OUT );

CTU_0( CU       := AddSTRING_0.xDone ,
       RESET    := xCTURESET_0 ,
       PV       := TO_WORD(iNumber_of_Elem));

CTU_1( CU       := WriteFile_0.xDone ,
       RESET    := xCTURESET_1 ,
       PV       := TO_WORD(iNumber_of_Times));

Init_0( xExecute        := xExecuteInit ,
        sDirectoryPath  := sDirectoryPath ,
        sFileName       := sCreateFileName ,
        sRowSeparator   := sLineSep ,
        sColumnSeparator:= sColumnSep ,
        rCSVWriter      := CSVWriter_0 );

AddSTRING_0( xExecute   := xExecuteSTRING ,
             sInput     := aData[CTU_0.CV] ,
             rCSVWriter := CSVWriter_0 );

NewLine_0( xExecute     := xExecuteNewLine ,
           rCSVWriter   := CSVWriter_0 );

WriteFile_0( xExecute   := xExecuteWriteFile ,
             rCSVWriter := CSVWriter_0 );

NewFile_0( xExecute     := xExecuteNewFile ,
           sNewFileName := sCreateFileName ,
           rCSVWriter   := CSVWriter_0 );

CASE iStep OF
    1: // Init or NewFile
        xStart  := TRUE;
        xFinish := FALSE;
        IF ( iNumber_of_xStart = 0 ) THEN
            xExecuteInit    := TRUE;
        ELSIF ( iNumber_of_xStart > 0 ) THEN
            xExecuteNewFile := TRUE;
```

```
        END_IF
        IF ( Init_0.xDone = TRUE ) OR ( NewFile_0.xDone = TRUE ) THEN
            iStep   :=2;
        END_IF

    2: // AddSTRING
        IF ( CTU_1.CV < iCount_Data ) THEN
            xExecuteSTRING  := TRUE;
            IF ( AddSTRING_0.xDone = TRUE ) THEN
                xExecuteSTRING  := FALSE;
                iStep           := 3;
            END_IF
        END_IF

    3: // Repeat until the specified number is added to the buffer
        IF ( CTU_0.Q = TRUE ) THEN
            xCTURESET_0 := TRUE;
            iStep       := 4;
        ELSE
            iStep       := 2;
        END_IF

    4: // NewLine
        xExecuteNewLine := TRUE;
        IF ( NewLine_0.xDone = TRUE ) THEN
            xExecuteNewLine := FALSE;
            iStep           := 5;
        END_IF

    5: // WriteFile
        xExecuteWriteFile   := TRUE;
        IF ( WriteFile_0.xDone = TRUE ) THEN
            xCTURESET_0         := FALSE;
            xExecuteWriteFile   := FALSE;
            iStep               := 6;
        END_IF

    6: // Repeat until the specified number is written
        IF ( CTU_1.Q = TRUE ) THEN
            xCTURESET_1 := TRUE;
            iStep       := 7;
        ELSE
            iStep       := 2;
        END_IF

    7: // Finish
        iNumber_of_xStart   := iNumber_of_xStart + 1;
        xStart              := FALSE;
        xCTURESET_1         := FALSE;
        iCount_Data         := 0;
        xExecuteInit        := FALSE;
        xExecuteNewFile     := FALSE;
        xFinish             := TRUE;
        iStep               := 0;
END_CASE
```

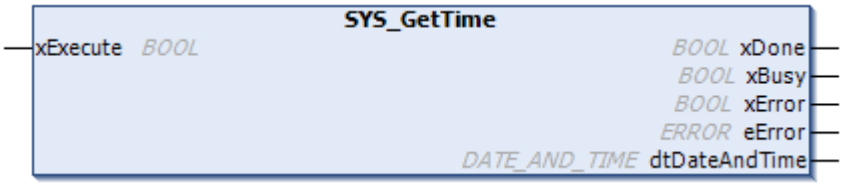
11.12 Clock Setting

This section describes function blocks that are used to set the clock of the GM1 Controller. Enter a function block name by using the RTCLK (namespace).

11.12.1 SYS_GetTime (Get Time)

This is a function block (FB) that gets the current local time

■ **Icon**



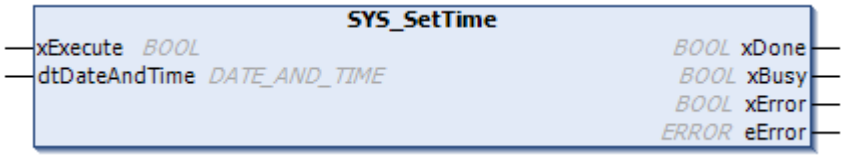
■ **Parameter**

Scope	Name	Type	Description
Input	xExecute	BOOL	TRUE: Active FALSE: Stop
Output	xDone	BOOL	TRUE: The function block is normally ended.
	xBusy	BOOL	TRUE: The function block is active.
	xError	BOOL	TRUE: An error has occurred.
	eError	ERROR	Details of error contents
	dtDateAndTime	DT	Current local time

11.12.2 SYS_SetTime (Set Time)

This is a function block (FB) that sets the current local time.

■ Icon



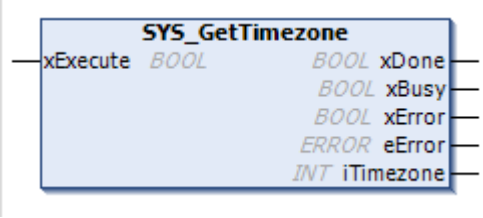
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	TRUE: Active FALSE: Stop
	dtDateAndTime	DT	Current time to be set
Output	xDone	BOOL	TRUE: The function block is normally ended.
	xBusy	BOOL	TRUE: The function block is active.
	xError	BOOL	TRUE: An error has occurred.
	eError	ERROR	Details of error contents

11.12.3 SYS_GetTimezone (Get Time Zone Information)

This is a function block (FB) that gets the time zone information.

■ Icon



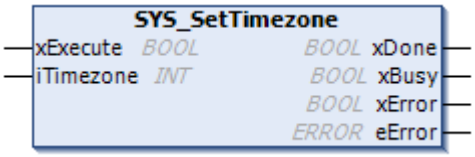
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	TRUE: Active FALSE: Stop
Output	xDone	BOOL	TRUE: The function block is normally ended.
	xBusy	BOOL	TRUE: The function block is active.
	xError	BOOL	TRUE: An error has occurred.
	eError	ERROR	Details of error contents
	iTimezone	INT	Time zone information (Offset from UTC)

11.12.4 SYS_SetTimezone (Set Time Zone Information)

This is a function block (FB) that sets the time zone information.

■ Icon



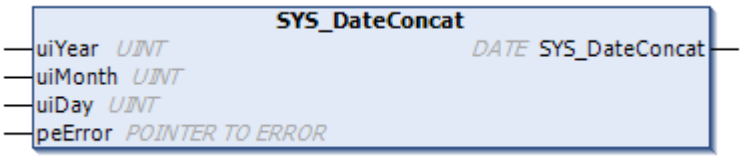
■ Parameter

Scope	Name	Type	Description
Input	xExecute	BOOL	TRUE: Active FALSE: Stop
	iTimezone	INT	Time zone information (Offset from UTC)
Output	xDone	BOOL	TRUE: The function block is normally ended.
	xBusy	BOOL	TRUE: The function block is active.
	xError	BOOL	TRUE: An error has occurred.
	eError	ERROR	Details of error contents

11.12.5 SYS_DateConcat (Convert from UINT Type to DATE Type)

This is a function (FUN) that converts a UINT type date to a DATE type.

■ Icon



■ Parameter

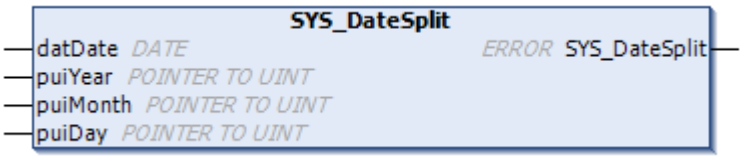
Scope	Name	Type	Description
Input	uiYear	UINT	Year: 1970 to 2099
	uiMonth	UINT	Month: 1 to 12
	uiDay	UINT	Day: 1 to 31
	peError	POINTER TO ERROR	Pointer to the error information storage location

Scope	Name	Type	Description
Output	SYS_DateConcat	DATE	Return value: Returns DT#1970-01-01 if the input value is invalid.

11.12.6 SYS_DateSplit (Convert from DATE Type to UINT Type)

This is a function (FUN) that converts a DATE type date to a UINT type.

■ Icon



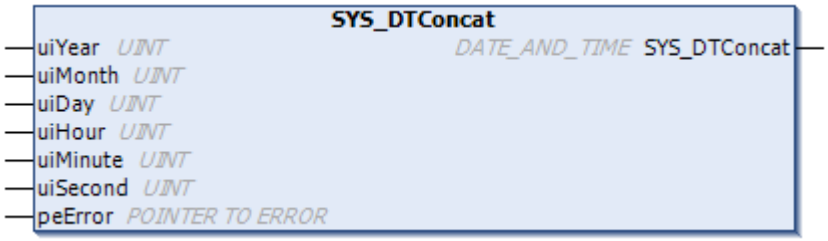
■ Parameter

Scope	Name	Type	Description
Input	datDate	DATE	Date data
	puiYear	POINTER TO UINT	Pointer to the year data storage location: 1970 to 2099
	puiMonth	POINTER TO UINT	Pointer to the month data storage location: 1 to 12
	puiDay	POINTER TO UINT	Pointer to the day data storage location: 1 to 31
Output	SYS_DateSplit	ERROR	Return value: Error information

11.12.7 SYS_DTConcat (Convert from UINT Type to DT Type)

This is a function (FUN) that converts a UINT type date and time to a DT type.

■ Icon



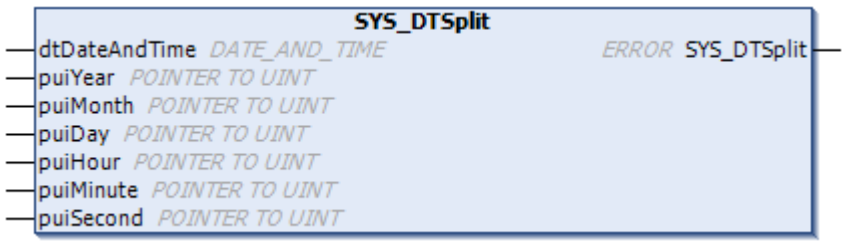
■ Parameter

Type	Name	Type	Description
Input	uiYear	UINT	Year: 1970 to 2099
	uiMonth	UINT	Month: 1 to 12
	uiDay	UINT	Day: 1 to 31
	uiHour	UINT	Hour: 0 to 23
	uiMinute	UINT	Minute: 0 to 59
	uiSecond	UINT	Second: 0 to 59
	peError	POINTER TO ERROR	Pointer to the error information
Output	SYS_DTConcat	DT	Return value: Returns DT#1970-01-01-00:00:00 if the input value is invalid.

11.12.8 SYS_DTSplit (Convert from UINT Type to DT Type)

This is a function (FUN) that converts a UINT type date and time to a DT type.

■ Icon



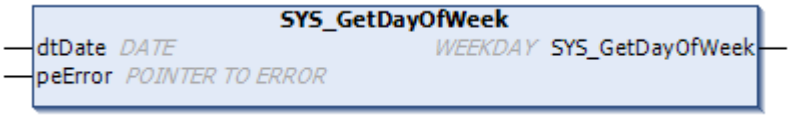
■ Parameter

Scope	Name	Type	Description
Input	dtDateAndTime	DT	Date and time data
	uiYear	POINTER TO UINT	Pointer to the year data storage location: 1970 to 2099
	uiMonth	POINTER TO UINT	Pointer to the month data storage location: 1 to 12
	uiDay	POINTER TO UINT	Pointer to the day data storage location: 1 to 31
	uiHour	POINTER TO UINT	Pointer to the hour data storage location: 0 to 23
	uiMinute	POINTER TO UINT	Pointer to the minute data storage location: 0 to 59
	uiSecond	POINTER TO UINT	Pointer to the second data storage location: 0 to 59
Output	SYS_DTSplit	ERROR	Return value: Error information

11.12.9 SYS_GetDayOfWeek (Get Day of the Week)

This is a function (FUN) that gets the day of the week from the DATE type date.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	dtDate	DATE	Date data
	peError	POINTER TO ERROR	Pointer to the error information
Output	SYS_GetDayOfWeek	Panasonic_GM_System.WEEKDAY	Return value: Day of the week

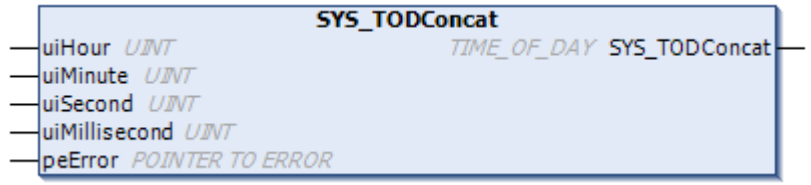
RTCLK.WEEKDAY (Day of the week)

Name	Value	Description
SUNDAY	16#00	Sunday
MONDAY	16#01	Monday
TUESDAY	16#02	Tuesday
WEDNESDAY	16#03	Wednesday
THURSDAY	16#04	Thursday
FRIDAY	16#05	Friday
SATURDAY	16#06	Saturday

11.12.10 SYS_TODConcat (Convert from UINT Type to TOD Type)

This is a function (FUN) that converts a UINT type time with milliseconds to a TOD type.

■ Icon



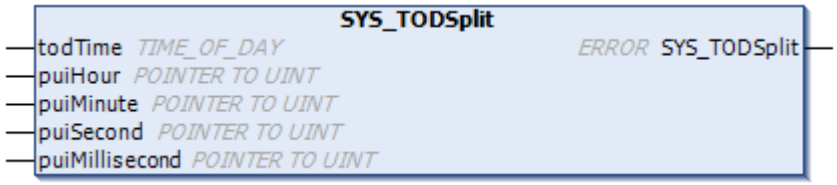
■ Parameter

Scope	Name	Type	Description
Input	uiHour	UINT	Hour: 0 to 23
	uiMinute	UINT	Minute: 0 to 59
	uiSecond	UINT	Second: 0 to 59
	uiMillisecond	UINT	Millisecond: 0 to 999
	peError	POINTER TO ERROR	Pointer to the error information
Output	SYS_TODConcat	TOD	Return value Returns TOD#00:00:00 if the input value is invalid.

11.12.11 SYS_TODSplit (Convert from TOD Type to UINT Type)

This is a function (FUN) that converts a TOD type time with milliseconds to a UINT type.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	todTime	TOD	Time (hour, minute, second) with millisecond data
	puiHour	POINTER TO UINT	Pointer to the hour data storage location: 0 to 23
	puiMinute	POINTER TO UINT	Pointer to the minute data storage location: 0 to 59
	puiSecond	POINTER TO UINT	Pointer to the second data storage location: 0 to 59
	puiMillisecond	POINTER TO UINT	Pointer to the millisecond data storage location: 0 to 999
Output	SYS_TODSplit	ERROR	Return value: Error information

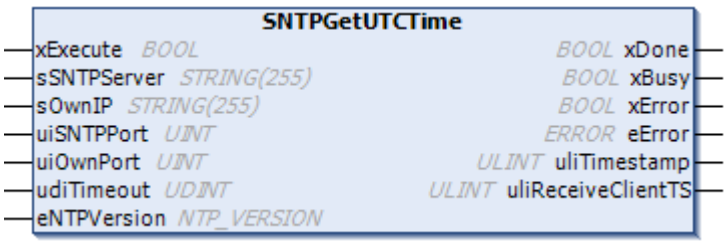
11.12.12 ERROR (Clock Instruction Error Code)

Name	Value	Description
NO_ERROR	0	No error
FIRST_ERROR	5700	First error unique to the library
TIME_OUT	5751	The time limit is exceeded.
NOT_AVAILABLE	5752	Not available.
INPUT_VALID	5753	Invalid input value
DTU_ERROR_UNKNOWN	5754	Unknown error
DTU_WRONG_PARAMETER	5755	Wrong parameter
DTU_TZI_NOT_SET	5756	The time zone information has not been initialized.
FIRST_MF	5770	First error unique to the manufacturer
LAST_ERROR	5799	Last error unique to the library

11.12.13 SNTP.SNTPGetUTCTime (Get SNTP Time)

This is a function block used to communicate with the SNTP server and get the current server time and Main Unit time. The gotten time is information about time in units of milliseconds that has elapsed since 00:00:00 on January 1 in 1970.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	xExecute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	sSNTPServer(Note 2)	STRING(255)	''	Specifies the IP address of the SNTP server.
	sOwnIP(Note 2)	STRING(255)	'0.0.0.0'	Specifies the IP address of the GM1 controller.(Note 1)
	uiSNTPPort(Note 2)	UINT	0	Specifies the port number of the SNTP server.
	uiOwnPort(Note 2)	UINT	0	Specifies the port number of the GM1 controller. Set it to 123.

Scope	Name	Type	Default value	Description
	udiTimeout	UDINT	1000	Timeout (Unit: ms)
	eNTPVersion	NTP_VERSION	V3	Specifies the NTP version.
Output	xDone	BOOL	FALSE	TRUE: Execution of the FB is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	SNTP.ERROR	NO_ERROR	An error ID is output. Refer to "11.12.14 SNTP.ERROR (SNTP Error Code)".
	uliTimestamp	ULINT	0	The current time of the SNTP server (unit: ms)
	uliReceiveClientTS	ULINT	0	The current time of the Main Unit (unit: ms)

(Note 1) Specify the IP address of the LAN port connected to the server.

(Note 2) A change made to any of the sSNTPServer, sOwnIP, uiSNTPPort, and uiOwnPort values after execution of the FB of the same instance does not take effect.

■ NTP_VERSION (Enumeration type)

Name	Value	Description
V3	0	Version 3
V4	1	Version 4

Info.

- Values for uliTimestamp and uliReceiveClientTS are acquired according to Coordinated Universal Time (UTC). To change the Main Unit time, conversion to a local time is necessary.
- SNTPGetUTCTime is designed to only get time. To update time, refer to "11.12.15 Example of SNTP Time Synchronization".

11.12.14 SNTP.ERROR (SNTP Error Code)

This is an enumeration type error code that is output when an SNTP function block is executed.

■ SNTP.ERROR (Enumeration type)

Name	Value	Description
NO_ERROR	0	No error
TIME_OUT	1	Timeout
INIT_ERROR	2	Initialization failed.
SEND_ERROR	3	Failed to send the request to the server.
SERVER_REFUSED_REQUEST	6	The server refused the request.
INVALID_LICENSE	7	Invalid license

11.12.15 Example of SNTP Time Synchronization

An example program for communicating with the SNTP server and synchronizing the gotten server time and the GM1 controller time will be described below.

- Description of process

  When the case number (byStep) is changed to 1, a process for time synchronization is executed. Before execution of the process, specify values for the variable sSendSrv_ip (IP address of the SNTP server) and variable sClient_ip (IP address of the GM1 controller). Details of the process for each case number in the implementation section are as described below.

  1. Execute SYS_GetTimeZone to get information on the time zone for the GM1 controller.
  2. Execute SNTPGetUTCTime to communicate with the SNTP server and get the current server time. After that, execute SYS_SetTime to set the GM1 controller time to the current local time.
  3. When time synchronizaion is completed, the variable xFinish goes TRUE.

- Declaration section

```
VAR
// Start SNTP function
byStep              : BYTE := 0;                //Process No
xFinish             : BOOL := FALSE;

// Setting Variables for SNTP_Client
sSendSrv_ip         : STRING := '192.168.1.100';//SNTP Server IP
sClient_ip          : STRING := '192.168.1.5';  //SNTP_Client(GM1) IP
uiSendSrv_port      : UINT   := 123;            //SNTP_Server port
uiClient_port       : UINT   := 123;            //SNTP_Client(GM1) port
udiTimeout          : UDINT  := 1000;           //Timeout[ms]

// FB instance
GetTimeZone_0       : SYS_GetTimezone;
SNTPGetUTCTime_0    : SNTP.SNTPGetUTCTime;
SetTime_0           : SYS_SetTime;

// Variables
xTimeZone_exe       : BOOL := FALSE;            //TRUE : TimeZone exe
xSNTPGetUTCTime_exe : BOOL := FALSE;            //TRUE : SNTPGetUTC exe
xSetTime_exe        : BOOL := FALSE;            //TRUE : SetTime exe
liTimeZone          : LINT;                     //GM1 TimeZone[min]
uliGETTime          : ULINT;                    //UTC[ms]

END_VAR
```

- Implementation section

```
//FunctionBlock
GetTimeZone_0( xExecute     := xTimeZone_exe,
               iTimezone    => liTimeZone );

SNTPGetUTCTime_0( xExecute      := xSNTPGetUTCTime_exe,
                  sSNTPServer   := sSendSrv_ip,
                  sOwnIP        := sClient_ip,
                  uiSNTPPort    := uiSendSrv_port,
```

```
                  uiOwnPort     := uiClient_port,
                  udiTimeout    := udiTimeout,
                  eNTPVersion   := SNTP.V3,
                  uliTimestamp  => uliGETTime );

SetTime_0( xExecute      := xSetTime_exe,
           dtDateAndTime := TO_DT((uliGETTime + (liTimeZone*60*1000))/1000)
);

(* Calculation of Set Time

liTimeZone*60*1000
 ...Convert Timezone information from hours to milliseconds

uliGETTime + (liTimeZone*60*1000)
 ...Calculation of Local Time

(uliGETTime + (liTimeZone*60*1000))/1000)
 ...Convert Local Time from milliseconds to Seconds

*)

CASE byStep OF
    1:  // Get GM1 Time zone information
        xFinish         := FALSE;
        xTimeZone_exe   := TRUE;
        IF ( GetTimeZone_0.xDone = TRUE ) THEN
            byStep  := 2;
        END_IF

    2:  // Update GM1 Time
        xSNTPGetUTCTime_exe := TRUE;
        IF ( SNTPGetUTCTime_0.xDone = TRUE ) THEN
            xSetTime_exe    := TRUE;
            IF ( SetTime_0.xDone = TRUE ) THEN
                byStep := 3;
            END_IF
        END_IF

    3:  //Finish
        xTimeZone_exe       := FALSE;
        xSNTPGetUTCTime_exe := FALSE;
        xSetTime_exe        := FALSE;
        xFinish             := TRUE;
        byStep              := 0;
END_CASE
```

11.13 System Data

11.13.1 SYS_GetSystemError (Get System Error)

This is a function block that gets the information of a system error that has occurred in the GM1 Controller.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	None		
Output	Error	BOOL	TRUE: An error has occurred.
	ErrorId	UDINT	Error ID of the error that has occurred

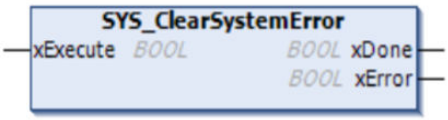
Info.

- For the error IDs, refer to the *GM1 Controller RTEX User's Manual (Operation Edition)* or *GM1 Controller EtherCAT User's Manual (Operation Edition)*.

11.13.2 SYS_ClearSystemError (Clear System Error)

This is a function block that clears a system error that has occurred in the GM1 Controller.

■ Icon



■ Parameter

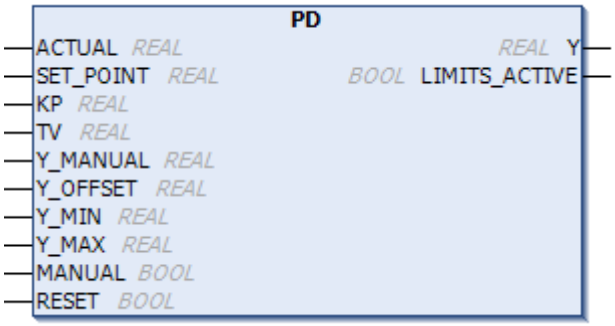
Scope	Name	Type	Description
Input	xExecute	BOOL	Starts execution at the rising edge.
Output	xDone	BOOL	TRUE: Processing is completed.
	xError	BOOL	TRUE: An error has occurred within the FB.

11.14 PID Control

11.14.1 PD (PD Control)

This is a function block (FB) that performs PD control. P control can be performed when TV is set to 0.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	ACTUAL	REAL	-	Current value
	SET POINT	REAL	-	Target value
	KP	REAL	-	Proportionality constant P
	TV	REAL	-	Derivative time D (unit: s)
	Y_MANUAL	REAL	-	A value output to output value (Y) when MANUAL = TRUE is set.
	Y_OFFSET	REAL	-	An offset value of output value (Y)
	Y_MIN	REAL	-	A lower limit value of output value (Y)
	Y_MAX	REAL	-	An upper limit value of output value (Y)
	MANUAL	BOOL	-	TRUE: The value set in Y_MANUAL is output to output value (Y).
	RESET	BOOL	-	TRUE: Output value (Y) is reset. Output value (Y) is set to the offset value (Y_OFFSET) to reset the integral portion.
Output	Y	REAL	-	Output value
	LIMITS_ACTIVE	BOOL	FALSE	TRUE: Output value (Y) is outside the range defined by Y_MIN/Y_MAX.

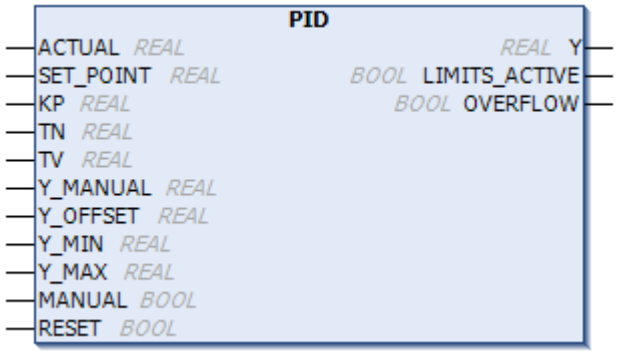
11.14.2 PID (PID Control)

This is a function block (FB) that performs PID control. Cycle time is automatically measured and PID operation is executed. PI control can be performed when TV is set to 0.

Info.

- Cycle time is a time passed while the FB is called twice.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	ACTUAL	REAL	-	Current value
	SET POINT	REAL	-	Target value
	KP	REAL	-	Proportionality constant P
	TN	REAL	-	Integral time I (unit: s)
	TV	REAL	-	Derivative time D (unit: s)
	Y_MANUAL	REAL	-	A value output to output value (Y) when MANUAL = TRUE is set.
	Y_OFFSET	REAL	-	An offset value of output value (Y)
	Y_MIN	REAL	-	A lower limit value of output value (Y)
	Y_MAX	REAL	-	An upper limit value of output value (Y)
	MANUAL	BOOL	-	TRUE: The value set in Y_MANUAL is output to output value (Y).
	RESET	BOOL	-	TRUE: Output value (Y) is reset. Output value (Y) is set to the offset value (Y_OFFSET) to reset the integral portion.
Output	Y	REAL	-	Output value
	LIMITS_ACTIVE	BOOL	FALSE	TRUE: Output value (Y) is outside the range defined by Y_MIN/Y_MAX.

Scope	Name	Type	Default	Description
	OVERFLOW	BOOL	-	TRUE: Integer portion overflow

Info.

- The maximum preciseness is 1 ms and thus the precision decreases when the operation is executed in a short cycle time.

  For instance, if the cycle time is 1 ms, PID control measures the cycle time as 2 ms or 0 ms in some cases.

  If the operation is executed in a short cycle time, use of PID_FIXCYCLE is recommended.

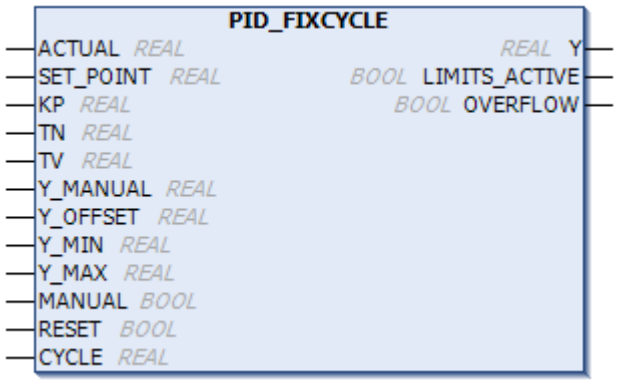
11.14.3 PID_FIXCYCLE [PID Control (Any Cycle Time)]

This is a function block (FB) that performs PID control. Cycle time can be manually set. PID operation is executed over the set cycle time. Except for the manually set cycle time, this FB operates in the same way as PID function blocks.

Info.

- Cycle time is a time passed while the FB is called twice.

■ Icon



■ Parameter

Scope	Name	Type	Default	Description
Input	ACTUAL	REAL	-	Current value
	SET POINT	REAL	-	Target value
	KP	REAL	-	Proportionality constant P
	TN	REAL	-	Integral time I (unit: s)
	TV	REAL	-	Derivative time D (unit: s)

Scope	Name	Type	Default	Description
	Y_MANUAL	REAL	-	A value output to output value (Y) when MANUAL = TRUE is set.
	Y_OFFSET	REAL	-	An offset value of output value (Y)
	Y_MIN	REAL	-	A lower limit value of output value (Y)
	Y_MAX	REAL	-	An upper limit value of output value (Y)
	MANUAL	BOOL	-	TRUE: The value set in Y_MANUAL is output to output value (Y).
	RESET	BOOL	-	TRUE: Output value (Y) is reset. Output value (Y) is set to the offset value (Y_OFFSET) to reset the integral portion.
	CYCLE	REAL	-	A time passed while the FB is called twice
Output	Y	REAL	-	Output value
	LIMITS_ACTIVE	BOOL	FALSE	TRUE: Output value (Y) is outside the range defined by Y_MIN/Y_MAX.
	OVERFLOW	BOOL	-	TRUE: Integer portion overflow

11.15 Recipe function

Variables in each recipe definition added to the Recipe Manager can be manipulated as recipes with the Recipe Method command.

The recipe method is affected by the “Storage” and “General” tabs.

If these are not configured correctly, the function will not behave properly.

■ Recipe method command description method

Each recipe method command is a method belonging to the function block RecipeManCommands.

Therefore, an item that sets RecipeManCommands as an instance must be written at the beginning of the method.

- Declaration section

```
RecipeManCommands_0   : RecipeManCommands;
```

- Implement section

```
output := RecipeManCommands_0.CreateRecipe(input1 , input2);
```

■ Terminology

- Recipe (GM Programmer tools)

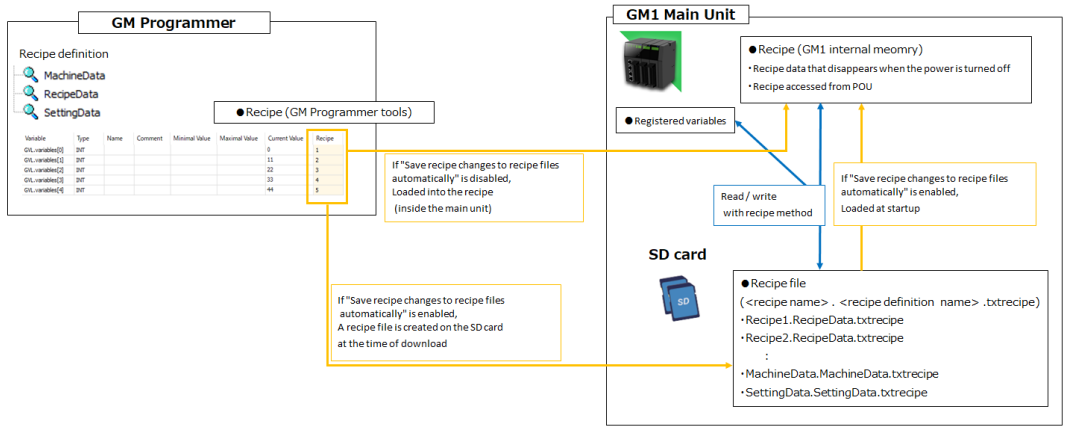
  A common file format for recipes handled by the GM Programmer recipe feature. The format of the standard recipe file is <Recipe Name>.<Recipe Definition Name>.<Recipe Extension>. The format for automatic saving in Recipe Manager is also the recipe file format.The contents follow the settings of the “Storage” tab.

- Recipe File

  The file format of recipes handled by the GM Programmer recipe function.The recipe file format is <Recipe Name<.<Recipe Extension<. By setting <Recipe Name< to <Recipe Name<.<Recipe Definition Name<, you can also handle recipe files.

- Recipe (GM1 internal meomry)

  Refers to a recipe that was created at runtime in the GM1 Main Unit.This recipe is not automatically saved and is deleted by resetting or turning on and off. If “Automatically save changes” in the Recipe Manager is enabled, a recipe file is created base on the Recipe(in SD card) when the user logs in to the GM1 Main Unit or when ReloadRecipes is run. In recipe commands, “recipe” refers mainly to the Recipe (GM1 internal meomry). The format of the recipe file is <recipe name>.<Recipe definition name>.<Recipe extension>.



Info.

- Recipes created in the tool cannot be rewritten directly from POU.
- Basically runtime recipes are manipulated by POU.
- If you want to handle the contents of a recipe created in the tool in POU, it is done via a recipe file or a runtime recipe saved automatically in the SD card.

■ Recipe file character limit

In GM Programmer tool, Recipe files has a limited number of characters. If set outside of these ranges, it may not work correctly.

The character limit for each item is as follows:

Setting item	Limit value
Recipe Name	Specify up to 35 characters.
Recipe definition name	Specify up to 35 characters.
Extension	Please enter no more than 10 characters, including periods.
File path name	Please enter no more than 176 characters, including the “\” symbol.

11.15.1 CreateRecipe (Create Recipe)

A method that creates a new recipe in the specified recipe definition from the "current value" and saves it as a recipe file. In addition, it is not possible to overwrite an existing recipe or specify the same name. An error occurs.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	RecipeDefinitionName	STRING	Recipe definition name for the recipe to be created (Note 1)
	RecipeName	STRING	The name of the recipe to be created (Note 2)
Output	CreateRecipe	DWORD	Output runtime ReturnValues

(Note 1) Be sure to specify the target recipe definition before operating.

(Note 2) Be sure to specify the recipe name to be created before operating.

■ Program example

A program that reads the “current value” of the recipe definition corresponding to the input variable input1 and creates a recipe with the recipe name of the input variable input2 and a recipe file.

- Declaration section

```
RecipeManCommands_0    : RecipeManCommands;
input1                 : STRING := 'RecipeDefName';
input2                 : STRING := 'RecipeName';
```

- Implement section

```
output := RecipeManCommands_0.CreateRecipe(input1 , input2);
```

■ Recipe Manager settings

The settings on the General tab of the Recipe Manager affect the following.

Setting item		Overview
Recipe management within PLC	-	If enabled, the RecipeManCommands method will be executable.
Save Recipe	Save recipe changes to recipe files automatically	If enabled, recipes created by the CreateRecipe method will automatically be saved to the SD card as recipe files.If this setting is disabled, recipes are created at runtime only and are not stored in recipe files.
Load Recipe	Load only exact matches in the variable list	The CreateRecipe method is not affected.
	Load matching variables by the variable name	The CreateRecipe method is not affected.
Write Recipe	Limit variables to minimum/maximum when the recipe value is out of range	The CreateRecipe method is not affected.
	Do not write to a variable if the recipe value is outside the minimum/maximum range	The CreateRecipe method is not affected.
Read Recipe	Check the recipe changes	The CreateRecipe method is not affected.

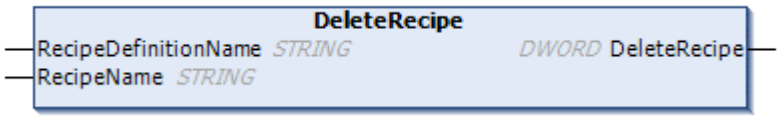
Info.

- If the SD card is not inserted into the GM1 Main Unit, a recipe file is not created, but no error occurs.
- When SD card write protection is enabled, a recipe file is not created, but no error occurs.
- If there is no free space on the SD card, no error occurs, but an empty recipe file is created.
- Since no error occurs in the above three cases, if you want to create a recipe file, check the size of the recipe file (FILE.GetSize) after it is generated.

11.15.2 DeleteRecipe (Delete Recipe)

This is a method to delete a recipe for the specified recipe definition. If a corresponding recipe file exists, the file is also deleted.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	RecipeDefinitionName	STRING	Recipe definition name for the recipe to be deleted (Note 1)
	RecipeName	STRING	The name of the recipe to be deleted (Note 2)
Output	DeleteRecipe	DWORD	Output runtime ReturnValues

(Note 1) Be sure to specify the target recipe definition before operating.

(Note 2) Be sure to specify the recipe name to be deleted before operating.

■ Program example

This is a program that deletes the recipes present in the recipe definition that correspond to the input variables input1 and input2.

- Declaration section

```
RecipeManCommands_0   : RecipeManCommands;
input1                : STRING := 'RecipeDefName';
input2                : STRING := 'RecipeName';
```

- Implement section

```
output := RecipeManCommands_0.DeleteRecipe(input1 , input2);
```

■ Recipe Manager settings

The settings on the General tab of the Recipe Manager affect the following.

Setting item		Overview
Recipe management within PLC	-	If enabled, the RecipeManCommands method will be executable.
Save Recipe	Save recipe changes to recipe files automatically	If this setting is disabled, DeleteRecipe will not delete recipe files.
Load Recipe	Load only exact matches in the variable list	The DeleteRecipe method is not affected.

Setting item		Overview
	Load matching variables by the variable name	The DeleteRecipe method is not affected.
Write Recipe	Limit variables to minimum/maximum when the recipe value is out of range	The DeleteRecipe method is not affected.
	Do not write to a variable if the recipe value is outside the minimum/maximum range	The DeleteRecipe method is not affected.
Read Recipe	Check the recipe changes	The DeleteRecipe method is not affected.

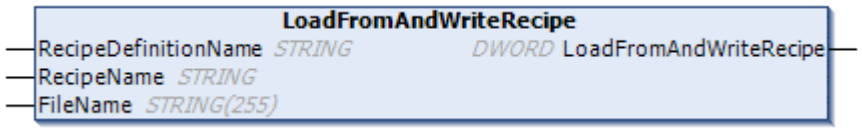
Info.

- If an SD card is not inserted into the GM1 Main Unit, the recipe file is not deleted and an error occurs.
- When SD card write protection is enabled, the recipe file is not deleted and an error occurs.

11.15.3 LoadFromAndWriteRecipe (Load and Write Recipe File)

Load the recipe from the specified recipe file. This is a method that then writes the recipe value to the corresponding recipe and “current value”. If the item “Variable” is not listed in the recipe file to be loaded, it will not be loaded correctly.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	RecipeDefinitionName	STRING	Recipe definition name for the recipe to be loaded (Note 1)
	RecipeName	STRING	The name of the recipe to be loaded (Note 2)
	FileName	STRING(255)	Name of the recipe file to be loaded (Note 3)
Output	LoadFromAndWriteRecipe	DWORD	Output runtime ReturnValues

(Note 1) Be sure to specify the target recipe definition before operating.

(Note 2) Be sure to specify the recipe name to be loaded before operating.

(Note 3) Be sure to specify the target recipe file before operating.

■ Program example

Load the recipe file corresponding to the input variable input3 for the recipe present in the recipe definition corresponding to the input variables input1 and input2. This is a program that writes that value to the “current value” of the recipe and recipe definition.

- Declaration section

```
RecipeManCommands_0   : RecipeManCommands;
input1                : STRING := 'RecipeDefName';
input2                : STRING := 'RecipeName';
input3                : STRING := 'RecipeName.RecipeDefName.txtrecipe';
```

- Implement section

```
output := RecipeManCommands_0.LoadFromAndWriteRecipe(input1 , input2 , inpu
t3);
```

■ Recipe Manager settings

The settings on the General tab of the Recipe Manager affect the following.

Setting item		Overview
Recipe management within PLC	-	If enabled, the RecipeManCommands method will be executable.
Save Recipe	Save recipe changes to recipe files automatically	If this setting is disabled, the recipe must be executed after it has been created in runtime.
Load Recipe	Load only exact matches in the variable list	The LoadFromAndWriteRecipe method can only be executed if the contents of the recipe file to be loaded match all of the settings on the Storage tab and the variable names in the recipe definition.
	Load matching variables by the variable name	When the LoadFromAndWriteRecipe method is executed, only variables that match the variable names in the recipe file to be loaded and the variable names in the recipe definition are written.
Write Recipe (Note 1)	Limit variables to minimum/maximum when the recipe value is out of range	When writing a recipe value to the “current value” in the recipe definition, if the value is outside the minimum/ maximum value range specified in the recipe definition, the minimum or maximum value set in the recipe definition will be written instead.
	Do not write to a variable if the recipe value is outside the minimum/maximum range	When writing a recipe value to the “current value” in the recipe definition, if the value is outside the minimum/ maximum value range specified in the recipe definition, the value is not written and the “current value” is preserved.
Read Recipe	Check the recipe changes	The LoadFromAndWriteRecipe method is not affected.

(Note 1) When writing recipe values to “current value” using this function, be sure to set the maximum and minimum values for all variables that can be set. It will not work correctly if you only set it partially. If you want to write all variables regardless of the maximum and minimum values, do not set the maximum and minimum values for all variables.

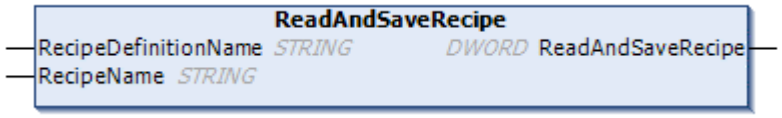
Info.

- If an SD card is not inserted into the GM1 Main Unit, an error occurs.

11.15.4 ReadAndSaveRecipe (Recipe File Overwrite Save)

This is a method that loads the “current value” into the recipe and then saves the recipe to a recipe file. If a recipe file with the same name already exists, it is overwritten and saved. To execute the ReadAndSaveRecipe method, a recipe must be created, for example, by CreateRecipe.

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	RecipeDefinitionName	STRING	Recipe definition name for the recipe to be read (Note 1)
	RecipeName	STRING	The name of the recipe to be saved (Note 2)
Output	ReadAndSaveRecipe	DWORD	Output runtime ReturnValues

(Note 1) If there is no free space on the SD card, no error occurs, but an empty recipe file is created.

(Note 2) Be sure to specify the target recipe name before operating.

■ Program example

This program reads the current value of the recipe definition corresponding to the input variable input1 into the recipe of input variable input2 and saves it in a recipe file.

- Declaration section

```
RecipeManCommands_0   : RecipeManCommands;
input1                : STRING := 'RecipeDefName';
input2                : STRING := 'RecipeName';
```

- Implement section

```
output := RecipeManCommands_0.ReadAndSaveRecipe(input1 , input2);
```

■ Recipe Manager settings

The settings on the General tab of the Recipe Manager affect the following.

Setting item		Overview
Recipe management within PLC	-	If enabled, the RecipeManCommands method will be executable.
Save Recipe	Save recipe changes to recipe files automatically	If this setting is disabled, the behavior will vary depending on the “Check for recipe changes” setting. ● “Check for recipe changes” is disabled: A recipe file is created on the SD card.

Setting item		Overview
		● “Check for recipe changes” is enabled: A recipe file is created on the SD card only when it is determined that the overwrite save has occurred due to a change in the recipe value. If this setting is enabled, follow the “Check for recipe changes” setting.
Load Recipe	Load only exact matches in the variable list	The ReadAndSaveRecipe method is not affected.
	Load matching variables by the variable name	The ReadAndSaveRecipe method is not affected.
Write Recipe	Limit variables to minimum/maximum when the recipe value is out of range	The ReadAndSaveRecipe method is not affected.
	Do not write to a variable if the recipe value is outside the minimum/maximum range	The ReadAndSaveRecipe method is not affected.
Read Recipe	Check the recipe changes	If you enable this setting, the ReadAndSaveRecipe method first loads the “current value” into the recipe when it runs.

Info.

- If an SD card is not inserted into the GM1 Main Unit, the recipe file is not overwritten and saved and an error occurs.
- When SD card write protection is enabled, the recipe file is not overwritten and saved and an error occurs.
- If there is no free space on the SD card, no error occurs, but an empty recipe file is created.

11.15.5 prvCompareRecipe (Compare Recipes)

This is a method to compare the “current value” of the recipe definition with the specified recipe (GM1 internal meomry). It can be executed when the “Check for recipe changes” setting in the Recipe Manager is enabled. Since the prvCompareRecipe method belongs to Private, an error occurs when the function block RecipeManCommands is executed before operation.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	RecipeDefinitionName	STRING	Recipe definition name to be compared (Note 1)

Scope	Name	Type	Description
	RecipeName	STRING	The name of the recipe to be compared (Note 2)
Output	prvCompareRecipe	BOOL	If the recipe matches the “current value” of the recipe definition, TRUE is returned

(Note 1) Be sure to specify the target recipe definition before operating.

(Note 2) Be sure to specify the recipe name to be compared before operating.

■ Program example

This is a program that compares the value of the recipe of input variable input2 present in the recipe definition corresponding to input variable input1 with the “current value”.

- Declaration section

```
RecipeManCommands_0   : RecipeManCommands;
input1                : STRING := 'RecipeDefName';
input2                : STRING := 'RecipeName';
```

- Implement section

```
output := RecipeManCommands_0.ReloadRecipes(input1);
output := RecipeManCommands_0.prvCompareRecipe(input1 , input2);
```

■ Recipe Manager settings

The settings on the General tab of the Recipe Manager affect the following.

Setting item		Overview
Recipe management within PLC	-	If enabled, the RecipeManCommands method will be executable.
Save Recipe	Save recipe changes to recipe files automatically	If this setting is disabled, the recipe must be executed after it has been created in runtime.
Load Recipe	Load only exact matches in the variable list	Verify that the settings in the Storage tab and the variable names in the recipe definition exactly match the contents of the recipe file to be compared.
	Load matching variables by the variable name	
Write Recipe	Limit variables to minimum/maximum when the recipe value is out of range	The prvCompareRecipe method is not affected.
	Do not write to a variable if the recipe value is outside the minimum/maximum range	The prvCompareRecipe method is not affected.
Read Recipe	Check the recipe changes	If enabled, the prvCompareRecipe method becomes executable.

Info.

- Please execute ReloadRecipes (reload of the recipe file in the SD card) and reflect it in the Recipe(GM1 internal meomry).If another method is executed before this method, an error will occur.

11.15.6 ReloadRecipes (Load Recipe File in SD Card)

This is a method to load the recipe file saved in the SD card for the specified recipe definition. The loaded recipe file is saved as recipe(GM1 internal meomry). This is necessary to access the recipe file in the card correctly, such as when the SD card is inserted and disconnected during GM1 operation.

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	RecipeDefinitionName	STRING	Recipe definition name to be reloaded (Note 1)
Output	ReloadRecipes	DWORD	Output runtime ReturnValues

(Note 1) Be sure to specify the target recipe definition before operating.

■ Program example

This is a program that reads the recipe file in the SD card that belongs to the recipe definition corresponding to the input variable input1.

- Declaration section

```
RecipeManCommands_0   : RecipeManCommands;
input1                : STRING := 'RecipeDefName';
```

- Implement section

```
output := RecipeManCommands_0.ReloadRecipes(input1);
```

■ Recipe Manager settings

The settings on the General tab of the Recipe Manager affect the following.

Setting item		Overview
Recipe management within PLC	-	If enabled, the RecipeManCommands method will be executable.
Save Recipe	Save recipe changes to recipe files automatically	If this setting is disabled, ReloadRecipes will not load the recipe file.
Load Recipe	Load only exact matches in the variable list	The ReloadRecipes method is not affected.
	Load matching variables by the variable name	The ReloadRecipes method is not affected.

Setting item		Overview
Write Recipe	Limit variables to minimum/maximum when the recipe value is out of range	The ReloadRecipes method is not affected.
	Do not write to a variable if the recipe value is outside the minimum/maximum range	The ReloadRecipes method is not affected.
Read Recipe	Check the recipe changes	The ReloadRecipes method is not affected.

Info.

- If there is no data on the SD card inserted into GM1, or if the SD card is not inserted, no error occurs. Run GetRecipeCount, for example, to verify that the recipe file in the card was loaded correctly.

11.15.7 GetRecipeCount (Count Recipes)

This is a method to count the number of recipes belonging to the specified recipe definition.The count target is the recipes (GM1 internal meomry).

■ Icon

■ Parameter

Scope	Name	Type	Description
Input	RecipeDefinitionName	STRING	Recipe definition name that counts the number of recipes (Note 1)
Output	GetRecipeCount	INT	Output the number of recipes present in the recipe definition

(Note 1) Be sure to specify the target recipe definition before operating.

■ Program example

This is a program that counts the number of recipes present in the recipe definition corresponding to the input variable input1.

- Declaration section

```
RecipeManCommands_0   : RecipeManCommands;
input1                : STRING := 'RecipeDefName';
```

- Implement section

```
output := RecipeManCommands_0.GetRecipeCount(input1);
```

■ Recipe Manager settings

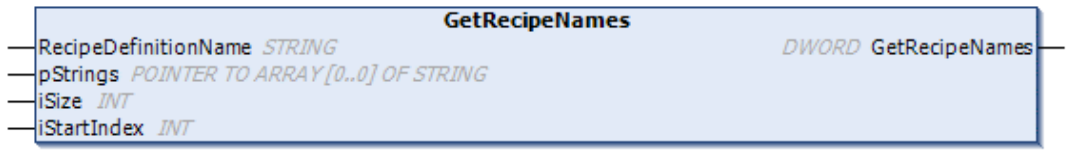
The settings on the General tab of the Recipe Manager affect the following.

Setting item		Overview
Recipe management within PLC	-	If enabled, the RecipeManCommands method will be executable.
Save Recipe	Save recipe changes to recipe files automatically	The GetRecipeCount method is not affected.
Load Recipe	Load only exact matches in the variable list	The GetRecipeCount method is not affected.
	Load matching variables by the variable name	The GetRecipeCount method is not affected.
Write Recipe	Limit variables to minimum/maximum when the recipe value is out of range	The GetRecipeCount method is not affected.
	Do not write to a variable if the recipe value is outside the minimum/maximum range	The GetRecipeCount method is not affected.
Read Recipe	Check the recipe changes	The GetRecipeCount method is not affected.

11.15.8 GetRecipeNames (Get Recipe Names)

This is a method to acquire the name of the recipe that belongs to the specified recipe definition. The count target is the recipes (GM1 internal meomry).

■ Icon



■ Parameter

Scope	Name	Type	Description
Input	RecipeDefinitionName	STRING	Recipe definition name that acquires the recipe name (Note 1)
	pStrings	POINTER TO ARRAY OF STRING	Pointer to store the acquired recipe name (Note 2)
	iSize	INT	Number of recipes to acquire (Note 3) (Note 4)
	iStartIndex	INT	Index value of the recipe to acquire. (Note 4) Index value for recipe starting with 0.
Output	GetRecipeNames	DWORD	Output runtime ReturnValues

(Note 1) Be sure to specify the target recipe definition before operating.

(Note 2) The pointer array must be STRING (80) type. If you specify another type, the correct recipe name will not be acquired.

(Note 3) Do not set a value greater than the number of arrays set for pStrings. Failure to set the appropriate value may result in an unexpected error.

(Note 4) Be sure to set a value greater than or equal to 0 before operating. If you set a negative number, it will not work.

■ Program example

This is a program that acquires the recipe name from the input variable input4 +1th recipe for the number of input variables input3 for the recipe present in the recipe definition corresponding to the input variable input1.

- Declaration section

```
RecipeManCommands_0   : RecipeManCommands;
input1                : STRING := 'RecipeDefName';
input2                : ARRAY [0..9] OF STRING;
input3                : INT:=10;
input4                : INT:=0;
```

- Implement section

```
output := RecipeManCommands_0.GetRecipeNames(input1,ADR(input2),input3,inpu
t4);
```

■ Recipe Manager settings

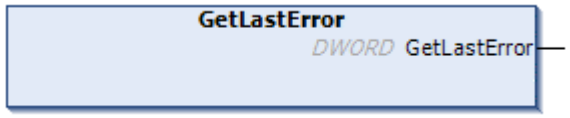
The settings on the General tab of the Recipe Manager affect the following.

Setting item		Overview
Recipe management within PLC	-	If enabled, the RecipeManCommands method will be executable.
Save Recipe	Save recipe changes to recipe files automatically	The GetRecipeNames method is not affected.
Load Recipe	Load only exact matches in the variable list	The GetRecipeNames method is not affected.
	Load matching variables by the variable name	The GetRecipeNames method is not affected.
Write Recipe	Limit variables to minimum/maximum when the recipe value is out of range	The GetRecipeNames method is not affected.
	Do not write to a variable if the recipe value is outside the minimum/maximum range	The GetRecipeNames method is not affected.
Read Recipe	Check the recipe changes	The GetRecipeNames method is not affected.

11.15.9 GetLastError (Get Last ReturnValues)

This is a method that acquires the ReturnValues value from the last action when using the recipe method.

■ Icon



■ Parameter

Scope	Name	Type	Description
Output	GetLastError	DWORD	Output the last output ReturnValues

■ ReturnValues (GVL)

Name	Number (DWORD)	Description
ERR_OK	16#0	Operation successful
ERR_RECIPE_FILE_NOT_FOUND	16#4000	Recipe file not found
ERR_RECIPE_MISMATCH	16#4001	The recipe file contents do not match the current recipe. Occurs only if the storage type is a string and the variable name in the file does not match the variable name in the recipe definition. The recipe file will not be loaded at this time.
ERR_RECIPE_SAVE_ERR	16#4002	The save operation failed due to the following reasons: ● The file cannot be created or opened because the SD card is full ● The configured file path does not exist ● The configured file extension is not allowed at runtime
ERR_RECIPE_NOT_FOUND	16#4003	The specified recipe does not exist
ERR_RECIPE_DEFINITION_NOT_FOUND	16#4004	The specified recipe definition does not exist
ERR_RECIPE_ALREADY_EXIST	16#4005	The specified recipe already exists in the recipe definition
ERR_NO_RECIPE_MANAGER_SET	16#4006	“Recipe Management in PLC” is not enabled in the Recipe Manager
ERR_RECIPE_NOMEMORY	16#4008	The recipe definition does not have enough free memory to create a new recipe: ● More than 50 recipes were created in the recipe definition

■ Program example

This is a program that acquires the ReturnValues value obtained from the last recipe method operation executed.

- Declaration section

```
RecipeManCommands_0   : RecipeManCommands;
```

- Implement section

```
output := RecipeManCommands_0.GetLastError();
```

■ ■Recipe Manager settings

The settings on the General tab of the Recipe Manager affect the following.

Setting item		Overview
Recipe management within PLC	-	If enabled, the RecipeManCommands method will be executable.
Save Recipe	Save recipe changes to recipe files automatically	The GetLastError method is not affected.
Load Recipe	Load only exact matches in the variable list	The GetLastError method is not affected.
	Load matching variables by the variable name	The GetLastError method is not affected.
Write Recipe	Limit variables to minimum/maximum when the recipe value is out of range	The GetLastError method is not affected.
	Do not write to a variable if the recipe value is outside the minimum/maximum range	The GetLastError method is not affected.
Read Recipe	Check the recipe changes	The GetLastError method is not affected.

11.15.10 GetLastInfo (Get Last InfoValues)

This is a method that acquires the InfoValues value obtained from the last action when using the recipe method. It may be acquired automatically at the start of GM1 operation.

■ Icon

■ Parameter

Scope	Name	Type	Description
Output	GetLastInfo	InfoValues	Outputs the final output InfoValues (Note 1)

(Note 1) If multiple InfoValues occur simultaneously, the sum of the UDINT values for each Info is output.

■ InfoValues (Enumeration type)

Occurs when the Recipe Manager load recipe setting is “Load variables that match the variable name”

Name	Number (UDINT)	Description
NO_INFO	16#0	No Info occurrence
INFO_RECIPE_MANAGER_NOT_ALL_VARIABLES_FOUND	16#1	When loading a recipe file, some variables in “Variables” in the recipe definition are not in the recipe file
INFO_RECIPE_MANAGER_OTHER_VARIABLES_FOUND	16#2	When loading a recipe file, some variables in the recipe file are not in “variables” in the recipe definition
INFO_RECIPE_MANAGER_ONE_OR_MORE_VARIABLES_FOUND	16#3	When the recipe file was loaded, one or more values were written to “current value” in the recipe definition
INFO_RECIPE_MANAGER_ALL_VARIABLES_FOUND	16#4	When the recipe file was loaded, all values were written to “current value” in the recipe definition

■ Program example

This is a program that acquires the InfoValues value obtained from the last recipe method operation executed.

- Declaration section

```
RecipeManCommands_0   : RecipeManCommands;
```

- Implement section

```
output := RecipeManCommands_0.GetLastInfo();
```

■ Recipe Manager settings

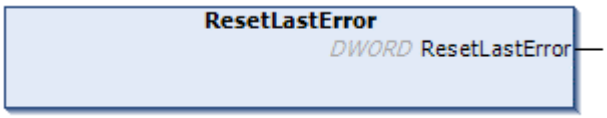
The settings on the General tab of the Recipe Manager affect the following.

Setting item		Overview
Recipe management within PLC	-	If enabled, the RecipeManCommands method will be executable.
Save Recipe	Save recipe changes to recipe files automatically	The GetLastInfo method is not affected.
Load Recipe	Load only exact matches in the variable list	The GetLastInfo method is not affected.
	Load matching variables by the variable name	The GetLastInfo method is not affected.
Write Recipe	Limit variables to minimum/maximum when the recipe value is out of range	The GetLastInfo method is not affected.
	Do not write to a variable if the recipe value is outside the minimum/maximum range	The GetLastInfo method is not affected.
Read Recipe	Check the recipe changes	The GetLastInfo method is not affected.

11.15.11 ResetLastError (GetLastError Reset)

This is a method to reset the value of GetLastError. It is used in conjunction with GetLastError.

■ Icon



■ Parameter

Scope	Name	Type	Description
Output	ResetLastError	DWORD	Output runtime ReturnValues

■ Program example

This is a program that resets the GetLastError value.

- Declaration section

```
RecipeManCommands_0   : RecipeManCommands;
```

- Implement section

```
output := RecipeManCommands_0.ResetLastError();
```

■ Recipe Manager settings

The settings on the General tab of the Recipe Manager affect the following.

Setting item		Overview
Recipe management within PLC	-	If enabled, the RecipeManCommands method will be executable.
Save Recipe	Save recipe changes to recipe files automatically	The ResetLastError method is not affected.
Load Recipe	Load only exact matches in the variable list	The ResetLastError method is not affected.
	Load matching variables by the variable name	The ResetLastError method is not affected.
Write Recipe	Limit variables to minimum/maximum when the recipe value is out of range	The ResetLastError method is not affected.
	Do not write to a variable if the recipe value is outside the minimum/maximum range	The ResetLastError method is not affected.
Read Recipe	Check the recipe changes	The ResetLastError method is not affected.

11.15.12 ResetLastInfo (GetLastInfo Reset)

This is a method to reset the value of GetLastInfo. It is used in conjunction with GetLastInfo.

■ Icon

■ Parameter

Scope	Name	Type	Description
Output	ResetLastInfo	DWORD	Output runtime ReturnValues

■ Program example

This is a program that resets the GetLastInfo value obtained when using the Recipe Method command.

- Declaration section

```
RecipeManCommands_0   : RecipeManCommands;
```

- Implement section

```
output := RecipeManCommands_0.ResetLastInfo();
```

■ Recipe Manager settings

The settings on the General tab of the Recipe Manager affect the following.

Setting item		Overview
Recipe management within PLC	-	If enabled, the RecipeManCommands method will be executable.
Save Recipe	Save recipe changes to recipe files automatically	The ResetLastInfo method is not affected.
Load Recipe	Load only exact matches in the variable list	The ResetLastInfo method is not affected.
	Load matching variables by the variable name	The ResetLastInfo method is not affected.
Write Recipe	Limit variables to minimum/maximum when the recipe value is out of range	The ResetLastInfo method is not affected.
	Do not write to a variable if the recipe value is outside the minimum/maximum range	The ResetLastInfo method is not affected.
Read Recipe	Check the recipe changes	The ResetLastInfo method is not affected.

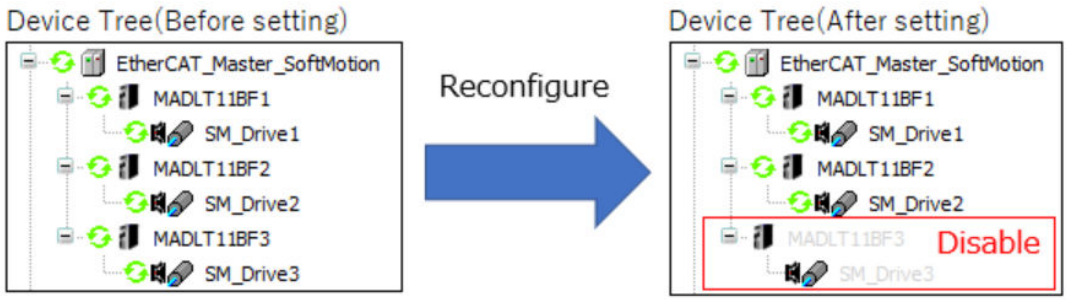
11.16 Enable/Disable Devices

By enabling/disabling a specified device, you can manage multiple system configurations in a shared project.

11.16.1 Overview of Device Enable/Disable Settings

With the GM1 controller, through use of function blocks, the enable/disable settings on specified devices can be changed. If a specified device is set to disable, the specified device is handled as being nonexistent in the system.

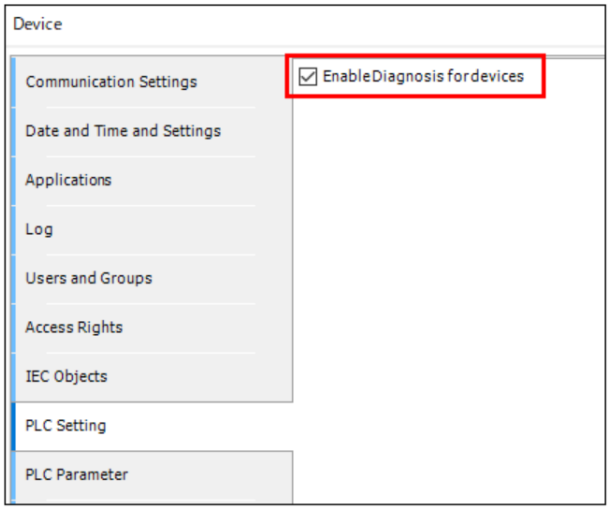
You can change the enable/disable setting on a slave device under EtherCAT_Master_SoftMotion in the Device tree.



Procedure

1. Configure enable/disable settings on slave devices.
   Using INode.Enable, configure the enable/disable setting on each slave.
2. Reconfigure the EtherCAT master.
   Execute Reconfigure to reconfigure the slave devices.
3. Reinitialize a servo amplifier out of the EtherCAT slave.
   Execute SMC3_ReinitDrive to reinitialize the slave axis that is set to enable.

To perform this operation, it is necessary to select “Enable diagnosis for device” checkbox on the “PLC Settings” tab in the Device object.



Info.

- It is possible to disable a device by selecting the device from the Device tree in the project and right-clicking it. However, in this case, the disable setting cannot be changed to enable by the program process described above. If you configure an enable/disable setting on a device through a program process, always set the device to the enable setting in the project.
- If two or more servo amplifiers are connected, avoid combined use of automatic number assignment and user-definable numbering in the StationAlias setting.
- For a program example, refer to "11.16.5 Sample Example: Changing EtherCAT Slave Enable/Disable Setting".

11.16.2 INode.Enable (Enable/Disable Setting)

This property is used for enabling/disabling a device. After a change is made to the property, the change takes effect by executing Reconfigure (reconfiguration). Write like DeviceName.Enable.

■ Parameter

Scope	Name	Type	Default value	Description
PROP	INode.Enable	BOOL	FALSE	TRUE: Enable FALSE: Disable

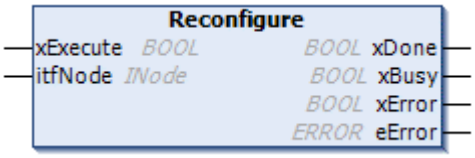
Info.

- To use this property, it is necessary to select “Enable diagnosis for device” checkbox on the “PLC Settings” tab in the Device object.

11.16.3 Reconfigure (Reconfigure Devices)

This function block reads information such as enable/disable settings on the corresponding device and all subdevices and restarts communication after the devices are reconfigured. After the execution of Reconfigure, all changed enable/disable settings on the slave devices and parameter changes take effect.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	xExecute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	itfNode	INode	-	Specifies the name of the node to be reconfigured. Example: EtherCAT_Master_SoftMotion
Output	xDone	BOOL	FALSE	TRUE: Execution of the FB is completed.
	xBusy	BOOL	FALSE	TRUE: FB is in progress.
	xError	BOOL	FALSE	TRUE: An error has occurred within the FB.
	eError	DED.ERROR	NO_ERROR	An error ID is output. "11.16.4 DED.ERROR (Error Code)"

Info.

- To use this FB, it is necessary to select “Enable diagnosis for device” checkbox on the “PLC Settings” tab in the Device object.

11.16.4 DED.ERROR (Error Code)

This is a list of error codes for device diagnosis.

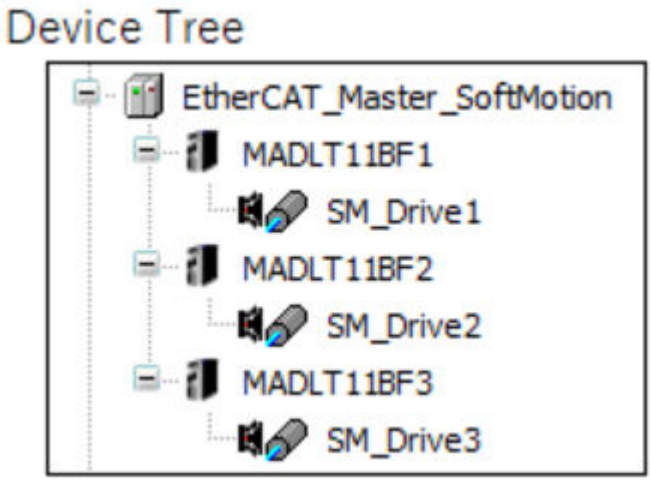
■ DED.ERROR (Enumeration type)

Name	Value	Description
NO_ERROR	0	No error
FIRST_ERROR	1300	First error unique to the library
TIME_OUT	1301	Timeout error

Name	Value	Description
ABORT	1302	Operation was aborted.
REF_INVALID	1303	The interface reference was invalid.
NOT_SUPPORTED	1304	The function is not supported.
ERROR_IO	1305	A general I/O configuration error occurred.
PARAM_INVALID	1306	Invalid parameter
NODE_NOT_EXISTING	1307	The specified node does not exist.
NO_MEMORY	1308	Dynamic memory allocation is disabled or system memory runs low.
ADR_NOT_FOUND	1309	The specified I/O address is invalid.
INST_NOT_FOUND	1310	There is no IDevice instance associated with the specified I/O address.
NO_DATA	1311	There is no data available.
OPERATION_INVALID	1312	Operation is not possible.
FIRST_MF	1350	First error unique to the manufacturer
LAST_ERROR	1399	Last error unique to the library

11.16.5 Sample Example: Changing EtherCAT Slave Enable/Disable Setting

This is a program coded to disable the "MADLT11BF3” device in the EtherCAT device configuration shown below. This program is useful if a project has a configuration made up of three real axes and the enable/disable setting on the third axis changes depending on the system.



To use this program, it is necessary to select “Enable diagnosis for device” checkbox on the “PLC Settings” tab in the Device object.

- Description of process

  When the case number (iStep) is set to 1, a process is executed. When the process is completed, the variable xFinish goes TRUE.

- Declaration section

```
VAR
// Change to 1 : To Start
iStep                   : INT    := 0;
// Finish Flag
xFinish                 : BOOL   := FALSE;
// FB instance
Reconfigure_0           : Reconfigure;
Reinit_1                : SMC3_ReinitDrive ;
Reinit_2                : SMC3_ReinitDrive ;
// Variables
xExecuteReconfigure     : BOOL := FALSE;
xExecuteReinit1         : BOOL := FALSE;
xExecuteReinit2         : BOOL := FALSE;

END_VAR
```

- Implementation section

```
//FunctionBlock
Reconfigure_0( xExecute := xExecuteReconfigure,
               itfNode  := EtherCAT_Master_SoftMotion);
Reinit_1( Axis     := SM_Drive1,
          bExecute := xExecuteReinit1);
Reinit_2( Axis     := SM_Drive2,
          bExecute := xExecuteReinit2);

CASE iStep OF
    1: // Device setting (Enable/Disable)
        MADLT11BF1.Enable   := TRUE;
        MADLT11BF2.Enable   := TRUE;
        MADLT11BF3.Enable   := FALSE;
        iStep               := 2;
    2: // Reconfigure
        xExecuteReconfigure := TRUE;
        IF ( Reconfigure.xDone = TRUE ) THEN
            xExecuteReconfigure := FALSE;
            iStep           := 3;
        END_IF
    3: // SMC3_ReinitDrive
        xExecuteReinit1      := TRUE;
        xExecuteReinit2      := TRUE;
        IF ( Reinit_1.bDone = TRUE ) AND ( Reinit_2.bDone = TRUE ) THEN
            xExecuteReinit1 := FALSE;
            xExecuteReinit2 := FALSE;
            xFinish         := TRUE;
            iStep           := 0;
        END_IF
END_CASE
```

11.17 Project Management Function

Project data can be backed up and restored via an SD memory card.

For instructions on how to execute the main operations of the project management feature, please refer to the Operation section.

11.17.1 What is Project Management Function?

The backup and restore functions for data of each type in the GM1 controller are called project management functions.

■ Backup function

This is a function used to collectively save data of every type in the GM1 controller to an SD memory card. File data to be saved is called a backup file. This function can be executed by any of two methods: operation by the controller and a function block in a program.

■ Restore function

This is a function used to transfer a backup file in an SD memory card to the GM1 controller. Data of every type in the GM1 controller will be replaced by a backup file. This function can be executed by any of two methods: operation by the controller and a function block in a program.

■ Backup file and restoration configuration file

When the backup function is performed, a backup file and a restoration configuration file are created in a predetermined directory on the SD memory card. These files have roles as shown below.

File name	Description	Backup	Restore
Backup file AUTOEXEC.GM1	Data of every type in the GM1 controller is stored in this file.	Creation	Reference
Restoration configuration file AUTOEXEC.INI	This file is used to specify data subject to restoration. This is text data and thus is editable.	Creation	Reference

■ Each type of settings data subject to backup and restoration

Target data	Backup	Restore
Source file	○	○
External file	○	Selectable(Note 1)
Bootstrap application	○	○
RETAIN variable	○	Selectable(Note 1)
Network setting	○	Selectable(Note 1)
Time zone	○	Selectable(Note 1)
Account (device user)	○	○
Certificates	×	×

(Note 1) Whether or not to specify it for a target of restoration can be selected.

■ Method for writing specifications in restoration configuration file

The restoration configuration file is a text file used to specify each type of settings data for a target of restoration when data is restored. The following is an example of specifications written in a restoration configuration file.

```
[AUTOEXEC_OPTION]
                Application_File=yes
                External_File=no
                Retain_File=no
                Network_File=no
                Timezone_File=no
```

Method for specifying each type of settings data

Writing	Description
Application_File	Source file, bootstrap application, and account (device user)
External_File	External file
Retain_File	RETAIN variable
Network_File	Network setting
Timezone_File	Time zone

Method for specifying data for a target of restoration.

Writing	Description
yes	Specify it for a target of restoration
no	Not specify it for a target of restoration

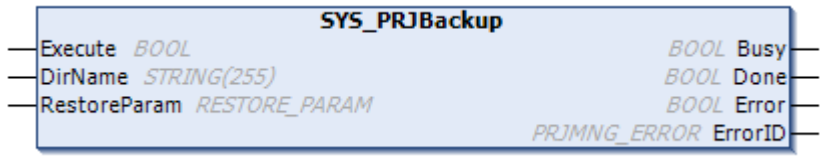
Info.

- Application_File cannot be excluded from the target of restoration. If it is excluded from the target of restoration, an error occurs.
- In writing, do not use spaces around the equals "=" sign. If there is any single- or double-width space around the equals "=" sign, the type of data is regarded as (no), being excluded from the target of restoration.

11.17.2 SYS_PRJBackup (Project Backup)

This is a function block that backs up project data on an SD memory card.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	Execute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	DirName	STRING(255)	-	A path to the folder in the SD memory card to which the “AUTOEXEC.GM1” backup file and the ”AUTOEXEC.INI” restoration configuration file are output.
	RestoreParam	RESTORE_PARAM	-	Parameter for the creation of the ”AUTOEXEC.INI” restoration configuration file Refer to"11.17.3 SYS_PRJRestore (Restore Project)".
Output	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Done	BOOL	FALSE	TRUE: Execution is completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	PRJMNG_ERROR	FALSE	Error ID output Refer to"11.17.4 PRJMNG_ERROR (Error Code)".

■ Description of functions

- In the folder specified by “DirName”, the “AUTOEXEC.GM1” backup file and the ”AUTOEXEC.INI” restoration configuration file are created. The restoration configuration file is created in accordance with details specified in “RestoreParam”.
- If the folder specified by “DirName” does not exist, a folder is created. Even a hierarchy folder is created.
- When the “AUTOEXEC.GM1” backup file and the ”AUTOEXEC.INI” restoration configuration file are present in the “DirName” folder, they are overwritten.
- The backup file and restoration configuration file that are renamed cannot be created.

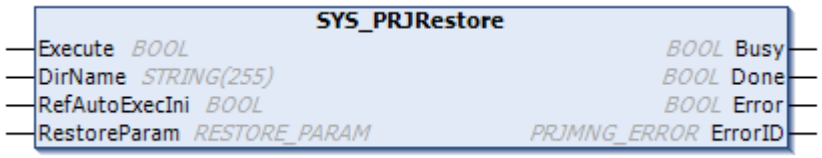
Info.

- In any of the following cases, the process is judged to be abnormal and the Error output goes TRUE.
  - A directory that is one level higher “..” or the root directory "//” is specified in “DirName”.
  - The SYS_PRJBackup command and SYSPRJ_Restore command are executed simultaneously.
  - The SD memory card is not in ready state, is write-protected, or is out of memory.
  - The number of created files or the number of directories has exceed the number permitted in the SD memory card.

11.17.3 SYS_PRJRestore (Restore Project)

This is a function block that restores project data from an SD memory card.

■ Icon



■ Parameter

Scope	Name	Type	Default value	Description
Input	Execute	BOOL	FALSE	At rising edge: Execution of the FB starts.
	DirName	STRING(255)	-	A path to the folder in the SD memory card in which the “AUTOEXEC.GM1” backup file to be referenced is stored.
	RefAutoExecIni	BOOL	TRUE	TRUE: Selects target data to be restored according to the ”AUTOEXEC.INI” restoration configuration file. FALSE: Selects target data to be restored according to the “RestoreParam” input.
	RestoreParam	RESTORE_PARAM	-	Parameter for selecting target data to be restored. Details will be described later.
Output	Busy	BOOL	FALSE	TRUE: Execution of the FB is not completed.
	Done	BOOL	FALSE	TRUE: Execution is completed.
	Error	BOOL	FALSE	TRUE: An error has occurred within the FB.
	ErrorID	PRJMNG_ERROR	FALSE	Error ID output Refer to"11.17.4 PRJMNG_ERROR (Error Code)".

■ Description of functions

The “AUTOEXEC.GM1” backup file in the folder specified by “DirName” is restored. The target data to be restored can be specified by any of two methods: through use of an FB argument or the ”AUTOEXEC.INI” restoration configuration file in the SD memory card.

1. To use the FB argument, specify the restoration target data by “RestoreParam” and set “RefAutoExecIni” to FALSE.
2. To use the ”AUTOEXEC.INI” restoration configuration file in the SD memory card, set “RefAutoExecIni” to TRUE. No specification is required for “RestoreParam”.

After restoration is completed, the GM1 controller automatically restarts.

Results of the restoration can be checked after the restart by"11.17.5 SYS_GetPRJRestoreResult (Project Restoration Results)".

■ RESTORE_PARAM (Structure)

This structure is used to specify restoration targets. It has the same specification items as in the ”AUTOEXEC.INI” restoration configuration file.

Member	Type	Default value	Description
ApplicationFile	BOOL	TRUE	Selects “source file”, “bootstrap application”, and “account (device user)” as restoration targets. TRUE: Restoration target FALSE: Not restoration target
ExternalFile	BOOL	FALSE	Selects “external file” as a restoration target. TRUE: Restoration target FALSE: Not restoration target
RetainFile	BOOL	FALSE	Selects “persistent variables” as a restoration target. TRUE: Restoration target FALSE: Not restoration target
NetworkFile	BOOL	FALSE	Selects “network settings ” as a restoration target. TRUE: Restoration target FALSE: Not restoration target
TimezoneFile	BOOL	FALSE	Selects “time zone” as a restoration target. TRUE: Restoration target FALSE: Not restoration target

Info.

- “Source file”, “bootstrap application”, and “account (device user)” cannot be excluded from restoration targets. If they are excluded from the target of restoration, an error occurs.
- A backup file name and a restoration configuration file name cannot be specified.
- In any of the following cases, the process is judged to be abnormal and the Error output goes TRUE.
  - The “DirName” folder does not exist.
  - The “AUTOEXEC.GM1” backup file is not present in the “DirName” folder.
  - “RefAutoExecIni” is set to TRUE ,and the ”AUTOEXEC.INI” restoration configuration file is not present in the “DirName” folder.
  - “RefAutoExecIni” is set to TRUE ,and “Application File=yes” is not found in the ”AUTOEXEC.INI” restoration configuration file.
  - “RefAutoExecIni” is set to FALSE, and “RestoreParam.ApplicationFile” is set to FALSE.
  - The SYS_PRJBackup command and SYS_PRJRestore command are executed simultaneously.
  - The SD memory card is not in ready state.

11.17.4 PRJMNG_ERROR (Error Code)

This is an enumeration type error code that is output when either of project backup and project restoration function blocks are executed.

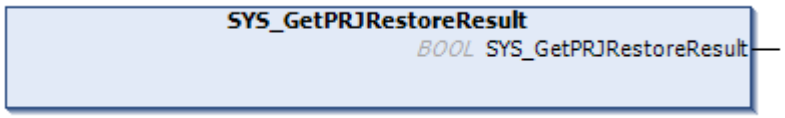
■ PRJMNG_ERROR (Enumeration type)

Name	Value	Description
NO_ERROR	0	No error
SYSTEM_ERROR	1	Internal error
NO_SD_CARD	2	SD card not inserted The SD memory card is not in ready state.
NOT_EXIST	3	The folder specified by “DirName” does not exist. A directory that is one level higher “..” is specified in “DirName”. The “AUTOEXEC.GM1” backup file is not present in the “DirName” folder. The ”AUTOEXEC.INI” restoration configuration file is not present in the “DirName” folder when “RefAutoExecIni” is set to TRUE.
MULTIPLE_EXEC	4	Multiple execution error occurs.
DIR_CREATE_FAIL	5	SD memory card not inserted The SD memory card is not in ready state. The SD memory card is write-protected. Failed to create the folder, which is specified by “DirName”, on the SD memory card. A directory that is one level higher “..” is specified in “DirName”. The root directory "//” is specified in “DirName”.
BACKUP_FAILED	6	Failed to create the backup file in the “DirName” folder on the SD memory card.
FILE_READ_FAILED	7	Error or failure in reading of the “AUTOEXEC.GM1” backup file
MODEL_VERSION_MISMATCH	8	The model version in the “AUTOEXEC.GM1” backup file does not match the model version of the GM1 controller.
INVALID_RESTORE_CONDITION	9	Application_File (“source file”, “bootstrap application”, and “account (device user)”) is excluded from the specified restoration target. Application_File in the ”AUTOEXEC.INI” restoration configuration file is excluded from the specified restoration target when “RefAutoExecIni” is set to TRUE. “RestoreParam.ApplicationFile” is set to FALSE when “RefAutoExecIni” is set to FALSE.
RESTORE_FAILED	10	Failed to restore project.
RETAIN_DATA_CORRUPTED	11	Failed to restore retain variables file

11.17.5 SYS_GetPRJRestoreResult (Project Restoration Results)

This function is used to get results of the execution of either of restore (controller operation) and SYS_PRJRestore (Restore Project).

■ Icon



■ Parameter

Scope	Name	Type	Description
Output	SYS_GetPRJRestoreResult	RESTORE_RESULT	This structure stores results of restoration. Details will be described later.

■ Description of functions

Results of the execution of either of restore (controller operation) and SYS_PRJRestore (Restore Project) take effect after the restart of the GM1 controller.

- Results of the execution will be kept until any of the actions below is performed.
  - Power OFF
  - Download project
  - Online change
  - Reset (Initialize PLC)
  - Device reset (Initialize PLC)
  - Reset Device by means of hard switching
- The following actions do not clear results of the restoration.
  - Execution of restore (controller operation)
  - Calling SYS_PRJRestore (Restore Project)

■ RESTORE_RESULT (Structure)

This structure stores results of restoration.

Member	Type	Default value	Description
Result	BOOL	FALSE	TRUE: Completion of restoration FALSE: Not completion of restoration
RestoreDateAndTime	DATE_AND_TIME	DT#1970-01-01-00:00:00	Date and time when restoration was completed

(MEMO)

12 Function Blocks for the Pulse Output Unit

12.1 Basic Configuration of Function Blocks for the Pulse Output Unit

This section describes the basic configuration of the function block.

12.1.1 Specifications of the Function Block

■ Common parameters

Listed below are the common arguments used in the GM1 Pulse Output Unit Function Blocks.

Scope	Parameter	Description
Input	UnitID	Please specify the connected Unit ID of the pulse output unit targeted by the function block execution within the range of 1 to 15. Please do not set a Unit ID that is not connected to a pulse output unit.
	AxisNo	Please specify the axis number targeted by the function block execution within the range of 1 to 4.
	Execute	This is a trigger that executes the function block. ● Execute the function block on the rising edge of Execute = TRUE. ● After the completion of the function block processing, changing Execute to FALSE will clear CommandAborted, Done, and Error. ● During the execution of the function block (when Busy = TRUE), if Execute becomes FALSE, the function block will continue to operate. At this time, when operation is finished, CommandAborted, Done or Error is held for one cycle.
	Enable	This enables an execution of a function block. ● The function block is executed on the rising edge of Enable = TRUE. ● If Enable becomes FALSE, the function block processing is stopped.
Output	Busy	During the execution of the function block, Busy = TRUE.(Note 1)
	Done	When the function block processing is complete, Done = TRUE.(Note 1)
	CommandAborted	If the function block processing is interrupted, CommandAborted = TRUE.(Note 1)
	Error	If an error occurs during the execution of the function block, Error = TRUE.
	ErrorID	Error information can be checked when an error occurs.
	Valid	When an output becomes valid, this output becomes TRUE.

(Note 1) After a function block is processed, one of the following parameters is set to TRUE: CommandAborted, Done, or Error.

■ Tasks

Either MotionTask or UserTask can be executed.
Use the same task for performing a process for the same axis.

■ Simulation mode

Simulation mode is not supported. Executing in simulation mode will result in a function block error (PG_NOT_SUPPORTED).

12.1.2 Notes for Executing the Function Block

■ Busy state

If the function block is busy (Busy = TRUE), call the function block at every cycle when executing a task.

Info.

- Exceptions, if any, will be described in the specifications for each function block.

■ Interruption of function block processing

- The following function blocks cannot be executed at the same time on the same axis: PG_Jog, PG_MoveRelative, PG_MoveAbsolute, PG_LatchPosition, PG_Home, and PG_Pulser

  The function block executed first takes precedence over the others.

  As for the function block that is executed later, CommandAborted is set to TRUE and processing is not started.
- PG_Stop takes precedence over any other control.

  If operation is stopped by executing PG_Stop, CommandAborted is set to TRUE for all other functions blocks and their controls are stopped
- If a stop operation (RUN → STOP) is applied in the middle of executing the function block, the pulse output unit stops outputting pulse signals.

  When operation is resumed, CommandAborted is set to TRUE for the function block being operated and its processing is interrupted.
- If an error occurs in a pulse output unit, the pulse output unit stops outputting pulse signals.

  For the function block that is being executed, the PG_AXIS_UNIT_ERROR is issued and its processing is interrupted.

■ Continuous execution of a function block

In a cycle where the Done output is set to TRUE, re-execute the function block (Execute = TRUE) to continue execution of the function block.

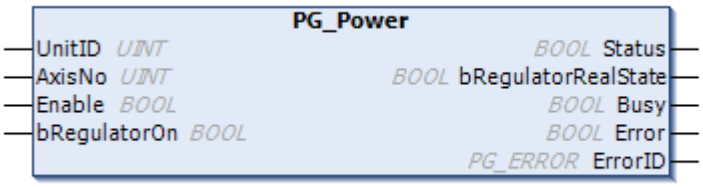
12.2 Function Blocks for the Pulse Output Unit

Various motion operations can be realized by using the function blocks for the pulse output unit. This section describes motion control function blocks for the single axis.

12.2.1 PG_Power

This is a function block (FB) that controls the servo ON/OFF of the axis of the pulse output unit.

■ Icon



■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	Enable	BOOL	FALSE	TRUE: Execution of the FB is enabled.
	bRegulatorOn	BOOL	FALSE	TRUE: Servo ON FALSE: Servo OFF
Output	Status	BOOL	FALSE	TRUE: The axis is operational FALSE: The axis cannot be executed.
	bRegulatorRealState	BOOL	FALSE	TRUE: Servo ON state FALSE: Servo OFF state
	Busy	BOOL	FALSE	TRUE: FB is in progress.
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	NO_ERROR	An error ID is output.

■ Notes for Executing the Function Block

- After executing PG_Power, check the servo ON state using the bRegulatorRealState output parameter.
- It is not necessary to call PG_Power at every cycle.

12.2.2 PG_Jog

This is a function block (FB) that causes the axis of the pulse output unit to keep traveling in a forward or backward direction.

■ Icon

■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	JogForward	BOOL	FALSE	TRUE: Travels in a forward direction.
	JogBackward	BOOL	FALSE	TRUE: Travels in a backward direction.
	Velocity	LREAL	0	Specifies the target speed (u/s).
	AccDec	LREAL	0	Specifies the acceleration / deceleration (u/s^2).
Output	Done	BOOL	FALSE	TRUE: FB operation is completed.
	Busy	BOOL	FALSE	TRUE: FB operation is in progress.
	CommandAborted	BOOL	FALSE	TRUE: FB operation is interrupted.
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	NO_ERROR	An error ID is output.

■ Operations when the function block is executed

- To start the operation, set either JogForward or JogBackward to TRUE depending on the direction you want to move.
- If both JogForward and JogBackward are set to TRUE, an error (PG_JOG_INVALID_REQUEST) will occur.

> - When executing again after the occurrence of the error, set both JogForward and JogBackward to FALSE once.

- When switching from JogForward to JogBackward (or from JogBackward to JogForward), the operation will switch after the current pulse output under control is completed.
- Velocity can be changed during operation (when Busy is TRUE).

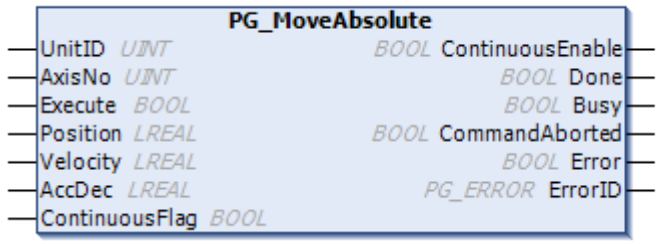
- The AccDec input cannot be changed during operation.

- When executing the function block, use the PG_Power function block in advance to set to the servo ON state.
- If the function block is executed in the servo OFF state, the PG_AXIS_NOT_READY_FOR_MOTION error is issued.

12.2.3 PG_MoveAbsolute

This is a function block (FB) that causes the axis of the pulse output unit to travel to a position specified as an absolute position.

■ Icon



■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Position	LREAL	0	Specifies the target position (u).
	Velocity	LREAL	0	Specifies the maximum velocity (u/s).
	AccDec	LREAL	0	Specifies the acceleration / deceleration (u/s^2).
	ContinuousFlag	BOOL	FALSE	FALSE: E-point control TRUE: P-point control
Output	ContinuousEnable	BOOL	FALSE	TRUE: Position, Velocity, and AccDec can be changed.
	Done	BOOL	FALSE	TRUE: FB operation is completed.
	Busy	BOOL	FALSE	TRUE: FB operation is in progress.
	CommandAborted	BOOL	FALSE	TRUE: FB operation is interrupted.
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	NO_ERROR	An error ID is output.

■ Operations when the function block is executed

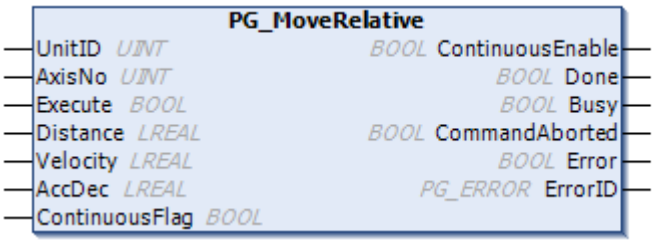
- The axis will perform an absolute move to the position specified by Position.
  (Coordinates: -2147483648 to 2147483647u)
- To execute E-point control, specify as follows
  - Please execute with ContinuousFlag = FALSE specified.
- To use the P-point control, specify as follows.
  - Please execute with ContinuousFlag = TRUE specified.
  - After the start of PG_MoveAbsolute execution, it becomes possible to overwrite Position, Velocity, and AccDec when ContinuousEnable = TRUE. Change the inputs triggered by the rising edge of Execute = TRUE.
  - If an overwrite is attempted when ContinuousEnable = FALSE, it will result in an error (PG_SET_VALUE_CHANGE_FAILED).
  - If no overwrite is performed, the execution of PG_MoveAbsolute will end (Done = TRUE) upon completion of the pulse output.

- When executing the function block, use the PG_Power function block in advance to set to the servo ON state.
- If the function block is executed in the servo OFF state, the PG_AXIS_NOT_READY_FOR_MOTION error is issued.

12.2.4 PG_MoveRelative

This is a function block (FB) that causes the axis of the pulse output unit to travel to a position specified as a relative position.

■ Icon



■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Distance	LREAL	0	Specifies the relative distance (u).
	Velocity	LREAL	0	Specifies the maximum velocity (u/s).

Scope	Parameter name	Type	Default	Description
	AccDec	LREAL	0	Specifies the acceleration / deceleration (u/s^2).
	ContinuousFlag	BOOL	FALSE	FALSE: E-point control TRUE: P-point control
Output	ContinuousEnable	BOOL	FALSE	TRUE: Distance, Velocity, and AccDec can be changed.
	Done	BOOL	FALSE	TRUE: FB operation is completed.
	Busy	BOOL	FALSE	TRUE: FB operation is in progress.
	CommandAborted	BOOL	FALSE	TRUE: FB operation is interrupted.
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	NO_ERROR	An error ID is output.

■ Operations when the function block is executed

- The axis will perform a relative move by the distance specified in Distance.
  (Coordinates: -2147483648 to 2147483647u)
- To execute E-point control, specify as follows
  - Please execute with ContinuousFlag = FALSE specified.
- To use the P-point control, specify as follows.
  - Please execute with ContinuousFlag = TRUE specified.
  - After starting the execution of PG_MoveRelative, it becomes possible to overwrite Distance, Velocity, and AccDec when ContinuousEnable = TRUE. Please change the inputs using the rising edge of Execute = TRUE as a trigger.
  - If an overwrite is attempted when ContinuousEnable = FALSE, it will result in an error (PG_SET_VALUE_CHANGE_FAILED).
  - If no overwrite is performed, upon completion of the pulse output, the execution of PG_MoveRelative will end (Done = TRUE).

- Please turn on the servo in advance using PG_Power before execution.
- Please turn on the servo in advance using PG_Power before execution.Running it with the servo off will result in an error (PG_AXIS_NOT_READY_FOR_MOTION).

12.2.5 PG_LatchPosition

This is a function block (FB) that moves the axis of the pulse output unit a specified relative distance from the position where an external signal is input.

■ Icon

■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Distance	LREAL	0	Specifies the relative distance (u).
	Velocity	LREAL	0	Specifies the maximum velocity (u/s).
	AccDec	LREAL	0	Specifies the acceleration / deceleration (u/s^2).
	PositioningStart	BOOL	FALSE	Positioning start input (for debugging)
Output	ContinuousEnable	BOOL	FALSE	TRUE: Start input
	Done	BOOL	FALSE	TRUE: FB operation is completed.
	Busy	BOOL	FALSE	TRUE: FB operation is in progress.
	CommandAborted	BOOL	FALSE	TRUE: FB operation is interrupted.
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	NO_ERROR	An error ID is output.

■ Operations when the function block is executed

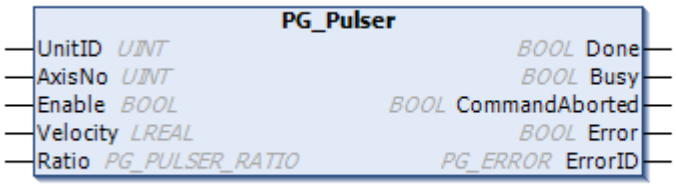
- The latch operation begins upon triggering with Execute = TRUE. It performs a relative move the distance of Distance from the starting point of the external signal input.
- PositioningStart functions as an alternative input to the external signal for initiating position control. It can be used for debugging purposes.

- Please turn on the servo in advance with PG_Power before execution. Running it with the servo off will result in an error (PG_AXIS_NOT_READY_FOR_MOTION).
- While in execution (Busy = TRUE), changing the input arguments or re-executing is not possible. Triggering Execute = TRUE will result in the error (PG_FB_IN_BUSY).

12.2.6 PG_Pulser

This is a function block (FB) that enables constant speed operation for the axes of the pulse output unit using external pulse input.

■ Icon



■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	Enable	BOOL	FALSE	TRUE: Enables pulser operation. FALSE: Disables pulser operation.
	Velocity	LREAL	0	Specifies the maximum velocity (u/s).
	Ratio	PG_PULSER_RATIO	RATIO_x1	Specifies the multiplication ratio between the pulser input and pulser output.
Output	Done	BOOL	FALSE	TRUE: FB operation is completed.
	Busy	BOOL	FALSE	TRUE: FB operation is in progress.
	CommandAborted	BOOL	FALSE	TRUE: FB operation is interrupted.
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	NO_ERROR	An error ID is output.

■ Operations when the function block is executed

- Please turn on the servo in advance with PG_Power before execution. Running it with the servo off will result in an error (PG_AXIS_NOT_READY_FOR_MOTION).

■ PG_PULSER_RATIO (Enumeration type)

For specifying the multiplication ratio between the pulser input and pulser output using PG_PULSER_RATIO, refer to the following table.

Definition	Description
RATIO_x1	x1
RATIO_x2	x2

Definition	Description
RATIO_x5	x5
RATIO_x10	x10
RATIO_x50	x50
RATIO_x100	x100
RATIO_x500	x500
RATIO_x1000	x1000

12.2.7 PG_Stop

This is a function block (FB) that causes the axis of the pulse output unit to make a forced stop or deceleration stop.

■ Icon

■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	Execute	BOOL	FALSE	Starts execution at the rising edge.
	StopMethod	BOOL	FALSE	TRUE: Emergency stop FALSE: Deceleration stop
Output	Done	BOOL	FALSE	TRUE: FB operation is completed.
	Busy	BOOL	FALSE	TRUE: FB operation is in progress.
	CommandAborted	BOOL	FALSE	TRUE: FB operation is interrupted.
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	NO_ERROR	An error ID is output.

■ Operations when the function block is executed

- Executing PG_Stop during the operation of the following function blocks will cause the function block in motion to interrupt its operation (CommandAborted = TRUE).
  - PG_Jog
  - PG_MoveRelative
  - PG_MoveAbsolute
  - PG_LatchPosition
  - PG_Pulser
  - PG_Home

  If you want to re-execute these function blocks, please set Execute of PG_Stop to FALSE before executing.
- The stop method can be specified using the StopMethod input.

  TRUE: Forced stop

  FALSE: Deceleration stop

- When decelerating to a stop, it will decelerate using the acceleration/deceleration rate set in the function block that is in operation.

- Please turn on the servo in advance with PG_Power before execution. Running it with the servo off will result in an error (PG_AXIS_NOT_READY_FOR_MOTION).
- While in execution (Busy = TRUE), changing the input arguments or re-executing is not possible. Triggering Execute = TRUE will result in the error (PG_FB_IN_BUSY).

12.2.8 PG_Home

This is a function block (FB) that performs home return of the pulse output unit.

■ Icon

■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	Execute	BOOL	FALSE	Starts execution at the rising edge.
	Velocity	LREAL	0	Specifies the maximum velocity (u/s).
	StartUpVelocity	LREAL	0	Specifies the startup velocity (u/s).
	AccDec	LREAL	0	Specifies the acceleration / deceleration (u/s^2).
Output	Done	BOOL	FALSE	TRUE: FB operation is completed.
	Busy	BOOL	FALSE	TRUE: FB operation is in progress.
	CommandAborted	BOOL	FALSE	TRUE: FB operation is interrupted.
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	NO_ERROR	An error ID is output.

■ Operations when the function block is executed

- For homing, the startup speed does not use the unit parameter settings but instead uses the value of StartUpVelocity.
- When reading the external signals for home position input or proximity to home position input, please use PG_ReadStatus.
- Please set the startup speed to 1u/s or higher. The value set as the startup speed will become the creep speed.
- Please set the maximum speed to 1u/s or higher. Setting it to 0 will cause an error during execution.

- Please turn on the servo in advance with PG_Power before execution. Running it with the servo off will result in an error (PG_AXIS_NOT_READY_FOR_MOTION).
- While in execution (Busy = TRUE), changing the input arguments or re-executing is not possible. Triggering Execute = TRUE will result in the error (PG_FB_IN_BUSY).

12.2.9 PG_SetPosition

This is a function block (FB) that sets the elapsed value and the feedback counter of the pulse output unit to desired values.

■ Icon

■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	Execute	BOOL	FALSE	Starts execution at the rising edge.
	PositionType	PG_POSITION_TYPE	ABSOLUTE	Specify the target settings. ABSOLUTE: Elapsed value FEEDBACK: Feedback counter BOTH: Elapsed value and feedback counter
	AbsoluteCounter	LREAL	0	Set value (Elapsed value)
	FeedbackCounter	LREAL	0	Set value (Feedback counter)
Output	Done	BOOL	FALSE	TRUE: FB operation is completed.
	Busy	BOOL	FALSE	TRUE: FB operation is in progress.
	CommandAborted	BOOL	FALSE	TRUE: FB operation is interrupted.
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	NO_ERROR	An error ID is output.

■ Operations when the function block is executed

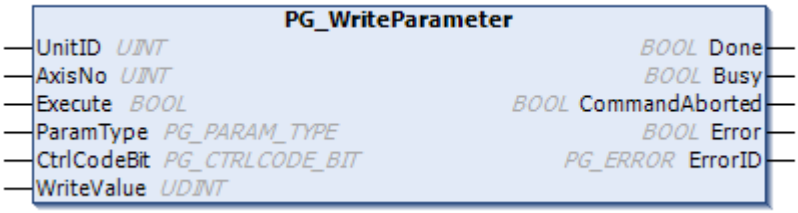
- If PositionType = BOTH is specified, you can set both the elapsed value and the feedback counter simultaneously.

- Please execute while the axis is stopped. If executed while the axis is in motion, it will result in an error (PG_AXIS_IN_DRIVEN).

12.2.10 PG_WriteParameter

This is a function block (FB) that writes the parameters to the pulse output unit.

■ Icon



■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	Execute	BOOL	FALSE	Starts execution at the rising edge.
	ParamType	PG_PARAM_TYPE	0	Specify the target parameter by combining PG_PARAM_TYPE and CtrlCodeBit.
	CtrlCodeBit	PG_CTRLCODE_BIT	0	
	WriteValue	UDINT	0	Write value
Output	Done	BOOL	FALSE	TRUE: FB operation is completed.
	Busy	BOOL	FALSE	TRUE: FB operation is in progress.
	CommandAborted	BOOL	FALSE	TRUE: FB operation is interrupted.
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	NO_ERROR	An error ID is output.

■ Operations when the function block is executed

- Upon startup of GM1, the values set in the "Pulse_4Axes Parameters" of GM Programmer are applied.
- Parameter changes made with PG_WriteParameter are not saved to the main unit.
  If you wish to save parameters to the main unit, please use the retained data.

■ How to Specify "Pulse_4Axes Parameters"

Specify the "Pulse_4Axes Parameters"by combining the argument ParamType and the argument CtrlCodeBit as shown in the following table.

If ParamType = CONTROL_CODE

- Specify the parameter to be written using the argument CtrlCodeBit.
- The value to be written can be specified using PG_CTRLCODE_VALUE (enumeration) or by setting the UDINT value in the table as the argument WriteValue.

- Using CtrlCodeBit = ALL_BIT, you can set the bit-wise OR of each parameter to the argument WriteValue, allowing you to write to all parameters in bulk.

Pulse_4Axes Parameter	CtrlCodeBit	WriteValue	UDINT value	Description
DirectionOfRotationInput	PULSE_INPUT_ROTATION_DIRECTION	PULSEIN_DIRECTION_FORWARD	16#00000000	Forward
		PULSEIN_DIRECTION_REVERSE	16#00010000	Reverse
Count	PULSE_INPUT_COUNT	PULSEIN_COUNT_ENABLE	16#00000000	Enable
		PULSEIN_COUNT_DISABLE	16#00020000	Cleared
PulseInputMode	PULSE_INPUT_MODE	PULSEIN_MODE_2PHASE	16#00000000	2-phase input
		PULSEIN_MODE_DIRECTION_DISTINCTION	16#00040000	Direction distinction input
		PULSEIN_MODE_INDIVIDUAL	16#00080000	Individual input
PulseInputCountMultiplication	PULSE_INPUT_MULTIPLICATION	PULSEIN_MULTIPLICATION_x1	16#00000000	x1
		PULSEIN_MULTIPLICATION_x2	16#00100000	x2
		PULSEIN_MULTIPLICATION_x4	16#00200000	x4
DirectionOfRotationOutput	PULSE_OUTPUT_ROTATION_DIRECTION	PULSEOUT_DIRECTION_FORWARD	16#00000000	Forward
		PULSEOUT_DIRECTION_REVERSE	16#01000000	Reverse
Pulse output mode	PULSE_OUTPUT_MODE	PULSEOUT_MODE_PULSESIGN	16#00000000	Pulse/Sign
		PULSEOUT_MODE_CWCCW	16#02000000	CW/CCW
PulseOutputFrequencyDivisionMode	PULSE_OUTPUT_DIVIDED_MODE	PULSEOUT_DIVIDED_BY1	16#00000000	Divided by 1
		PULSEOUT_DIVIDED_BY2	16#10000000	Divided by 2
		PULSEOUT_DIVIDED_BY4	16#20000000	Divided by 4
		PULSEOUT_DIVIDED_BY8	16#30000000	Divided by 8
		PULSEOUT_DIVIDED_BY16	16#40000000	Divided by 16
		PULSEOUT_DIVIDED_BY32	16#50000000	Divided by 32
		PULSEOUT_DIVIDED_BY64	16#60000000	Divided by 64
		PULSEOUT_DIVIDED_BY128	16#70000000	Divided by 128
DeviationCounterClearTime	DEVIATION_COUNTER_CLEAR_TIME	DEVIATION_COUNTER_CLEAR_TIME_1ms	16#00000000	1ms
		DEVIATION_COUNTER_CLEAR_TIME_10ms	16#80000000	10 ms
AccelerationDecelerationMethod	PULSE_OUT_ACC_DEC	PULSEOUT_ACC_DEC_LINEAR	16#00000000	Linear Acceleration / Deceleration
		PULSEOUT_ACC_DEC_SSHAPED	16#00000002	S Acceleration / Deceleration

Pulse_4Axes Parameter	CtrlCodeBit	WriteValue	UDINT value	Description
OriginReturnDirection	HOMING_DIRECTION	HOME_DIRECTION_NEGATIVE	16#00000000	(-) Direction of the elapsed value
		HOME_DIRECTION_POSITIVE	16#00000004	(+) Direction of the elapsed value
StartUpTime	STARTUP_TIME	STARTUP_TIME_20us	16#00000000	0.02ms
		STARTUP_TIME_5us	16#00000008	0.005ms
		STARTUP_TIME_1us	16#00000800	0.001ms
OriginInputLogic	HOME_INPUT_LOGIC	HOME_INPUT_NORMAL_CLOSE	16#00000000	NC contact
		HOME_INPUT_NORMAL_OPEN	16#00000010	NO contact
OriginNeighborhoodLogic	NEARHOME_INPUT_LOGIC	NEARHOME_INPUT_NORMAL_OPEN	16#00000000	NO contact
		NEARHOME_INPUT_NORMAL_CLOSE	16#00000020	NC contact
OriginSearch	HOME_SEARCH	HOME_SEARCH_DISABLE	16#00000000	Disable
		HOME_SEARCH_ENABLE	16#00000040	Enable
LimitInputLogic	LIMIT_INPUT_LOGIC	LIMIT_INPUT_NORMAL_CLOSE	16#00000000	NC contact
		LIMIT_INPUT_NORMAL_OPEN	16#00000080	NO contact
SShapedPattern	S_ACC_DEC	S_ACC_DEC_SIN_CURVE	16#00000000	Sin curve
		S_ACC_DEC_THIRD_CURVE	16#00003000	Third curve
Write all ParamTypes at once	ALL_BIT	-	-	Writes to all parameters

Example of settings

- Specify with PG_CTRLCODE_VALUE (enumeration).
  Startup time = 0.005ms (STARTUP_TIME_5us)

```
PG_WriteParameter_0(
    UnitID:=1,
    AxisNo:=1,
    Execute:=bExe_Wpara,
    ParamType:=CONTROL_CODE,
    CtrlCodeBit:=STARTUP_TIME,
    WriteValue:=STARTUP_TIME_5us,
);
```

- Specify with a UDINT value.
  Startup time = 0.005ms (16#00000008)

```
PG_WriteParameter_0(
    UnitID:=1,
    AxisNo:=1,
```

```
    Execute:=bExe_Wpara,
    ParamType:=CONTROL_CODE,
    CtrlCodeBit:=STARTUP_TIME,
    WriteValue:=16#00000008,
);
```

- Use ALL_BIT to set both the acceleration/deceleration mode and the startup time in bulk.
  Acceleration/Deceleration method = S-curve acceleration/deceleration (16#00000002),
  Startup time = 0.005ms (16#00000008)
  (In this example, all other parameters are set to a UDINT value of 0.)

```
PG_WriteParameter_0(
    UnitID:=1,
    AxisNo:=1,
    Execute:=bExe_Wpara,
    ParamType:=CONTROL_CODE,
    CtrlCodeBit:=ALL_BIT,
    WriteValue:=16#00000009,
);
```

If ParamType ≠ CONTROL_CODE

- Specify the parameter to be written within the argument ParamType.
- Specify the value to be written as a UDINT type in the argument WriteValue.

Pulse_4Axes Parameter	ParamType	CtrlCodeBit	WriteValue
PulseInputAorB SignalInCnst	PULSE_INPUT_SIGNAL_INCNST	-	0: Not InCnst (No input time constant) 1: 0.1 us 2: 0.5 us 3: 1.0 us 4: 2.0 us 5: 10.0 us
Home input Input time constant	HOME_INPUT_INCNST	-	0: Not InCnst (No input time constant) 1: 10 us 2: 100 us

12.2.11 PG_ReadParameter

This is a function block (FB) that reads the parameters of the pulse output unit.

■ Icon

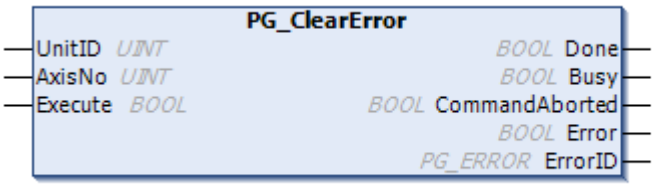
■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	Execute	BOOL	FALSE	Starts execution at the rising edge.
	ParamType	PG_PARAM_TYPE	0	Specify the target parameter by combining PG_PARAM_TYPE and CtrlCodeBit.
	CtrlCodeBit	PG_CTRLCODE_BIT	0	
Output	ReadValue	UDINT	0	Read value
	Done	BOOL	FALSE	TRUE: FB operation is completed.
	Busy	BOOL	FALSE	TRUE: FB operation is in progress.
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	NO_ERROR	An error ID is output.

12.2.12 PG_ClearError

This is a function block (FB) that clears the limit error or the set value error of the pulse output unit.

■ Icon



■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	Execute	BOOL	FALSE	Starts execution at the rising edge.
Output	Done	BOOL	FALSE	TRUE: FB operation is completed.
	Busy	BOOL	FALSE	TRUE: FB operation is in progress.
	CommandAborted	BOOL	FALSE	TRUE: FB operation is interrupted.
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	0	An error ID is output.

12.2.13 PG_ReadStatus

This is a function block (FB) that reads the status of the pulse output unit.

■ Icon

■ Parameter

Scope	Parameter name	Type	Default	Description
Input	UnitID	UINT	-	Specifies the unit ID.
	AxisNo	UINT	-	Specifies the axis No.
	Enable	BOOL	FALSE	TRUE: Execution of the FB is enabled.
Output	AbsoluteCounter	LREAL	0	Read value (Elapsed value)
	FeedbackCounter	LREAL	0	Read value (Feedback counter)
	InputFlags	PG_InputFlags	0	Read value content (Input flag)
	Valid	BOOL	FALSE	TRUE:
	Error	BOOL	FALSE	TRUE: An error has occurred in the FB.
	ErrorID	PG_ERROR	NO_ERROR	An error ID is output.

■ PG_InputFlags

For the contents of PG_InputFlags, refer to the following table.

Parameter	Name	Description
PulseOutputBusy	Pulse output busy	TRUE: Pulse output is in progress
PulseOutputDone	Pulse output done	TRUE: Pulse output is complete
AccelerationZone	Acceleration zone	TRUE: Axis is accelerating
ConstantSpeedZone	Constant speed zone	TRUE: Axis is operating at constant speed
DecelerationZone	Deceleration zone	TRUE: Axis is decelerating
RotationDirection	Rotation direction	Rotation direction monitor TRUE: Rotating in the direction of increasing elapsed value
HomeInput	Home input	Home position input signal monitor TRUE: Home position input is active
NearHomeInput	Near home input	Close to home position input signal monitor TRUE: Close to home position input is active

Parameter	Name	Description
HomingDone	Home return done	TRUE: Homing complete
OutputStopError	Output stop error	TRUE: Error occurred in the pulse output unit, output stopped
SetValueChangeConfirmation	Set value change	TRUE: Possible to overwrite set values during P-point control
OverPositiveLimitInput	Limit (+) input	Limit (+) input signal monitoring contact TRUE: Limit (+) input is active
OverNegativeLimitInput	Limit (-) input	Limit (-) input signal monitoring contact TRUE: Limit (-) input is active
TimingInputMonitor	Timing input monitor	Position control start input (timing input) monitoring contact TRUE: Position control start input is active
SetValueError	Set value error	RUE: Setting value error occurred
LimitError	Limit error	TRUE: Limit input was received during operation or at startup
ServoOnOutputState	Servo ON output status	TRUE: Servo ON state

12.3 Error Codes

This section describes errors that are output in function blocks for pulse output unit and their contents. These errors are defined in PG_ERROR.

12.3.1 Error Check Method

With a function block that has the Error and ErrorID output parameters, it is possible to check whether an error has occurred.

When the Error output becomes TRUE, its error content is output to the ErrorID.

■ Error occurrence example

In the following example, PG_NOT_SUPPORTED has occurred in the PG_Power function block.

Error = TRUE (An error has occurred.)

ErrorID = PG_NOT_SUPPORTED (The function block not supported in the simulation mode was executed.)

Declaration section

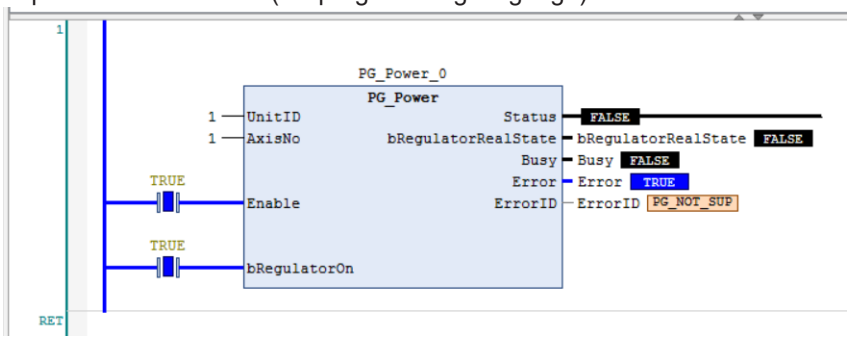
Device.Application.MC_PRG

Expression	Type	Value
PG_Power_0	PG_Power	
bRegulatorRealState	BOOL	FALSE
Busy	BOOL	FALSE
Error	BOOL	TRUE
ErrorID	PG_ERROR	PG_NOT_SUPPORTED

Implementation section (ST programming language)

```
PG_Power_0(UnitID:=1,
           AxisNo:=1,
           Enable:=TRUE,
           bRegulatorOn:=TRUE,
           Status=>Status,
           bRegulatorRealState=>bRegulatorRealState,
           Busy=>Busy,
           Error=>Error,
           ErrorID=>ErrorID);RETURN
```

Implementation section (LD programming language)



12.3.2 PG_ERROR

For the content of PG_ERROR output in each function block, refer to the following table.

Error name	Description
PG_NO_ERROR	Normal (no error)
PG_INVALID_UNIT	The specified unit ID or axis No. is invalid.
PG_SYSTEM_ERROR	This is an internal error in the GM1 Controller.
PG_NOT_SUPPORTED	The function block not supported in the simulation mode was executed.
PG_AXIS_NOT_READY_FOR_MOTION	The axis is in the servo OFF state.
PG_AXIS_UNIT_ERROR	The output of the pulse output unit has made an error stop.
PG_AXIS_SET_VALUE_ERROR	A set value error has occurred in the pulse output unit.
PG_AXIS_LIMIT_ERROR	A limit error has occurred in the pulse output unit.
PG_AXIS_IN_STOP	The function block could not be executed because stop processing was being executed by PG_Stop.
PG_AXIS_IN_RESET	The function block could not be executed because axis information was being changed by PG_ClearError and PG_SetPosition.
PG_AXIS_IN_DRIVEN	The function block could not be executed because the axis was moving.
PG_CHANGED_DURING_OPERATION	The unit ID or axis No. was changed during operation.
PG_INVALID_TARGET_VALUE	Abnormal Information (position, speed, acceleration, or deceleration) was Input.
PG_JOG_INVALID_REQUEST	Both JogForward and JogBackward of JOG were simultaneously set to TRUE.
PG_SET_VALUE_CHANGE_FAILED	Failed to change command information.
PG_FB_IN_BUSY	Using the timing when Execute is set to TRUE as the trigger is invalid for the function block being executed (Busy = TRUE).
PG_ERROR_CLEAR_FAILED	Failed to clear the error in the pulse output unit.
PG_PARAMETER_WRITE_FAILED	Failed to write the parameter.

(MEMO)

13 Reference Information

13.1 Motion Errors (SMC_ERROR Type)

This section describes errors that are output in motion control instructions and their contents. Motion control errors are defined in SMC_ERROR.

13.1.1 Error Check Method

This section describes errors that are output in motion control instructions and their contents.

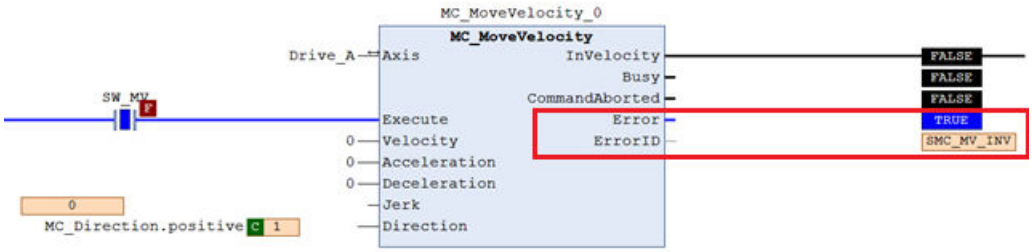
Motion control errors are defined in SMC_ERROR.

■ Error check method

With a function block that has an output Error and output ErrorID, it is possible to check that an error has occurred.

The following shows an example of an error that has occurred when the MC_MoveVelocity function block is executed.

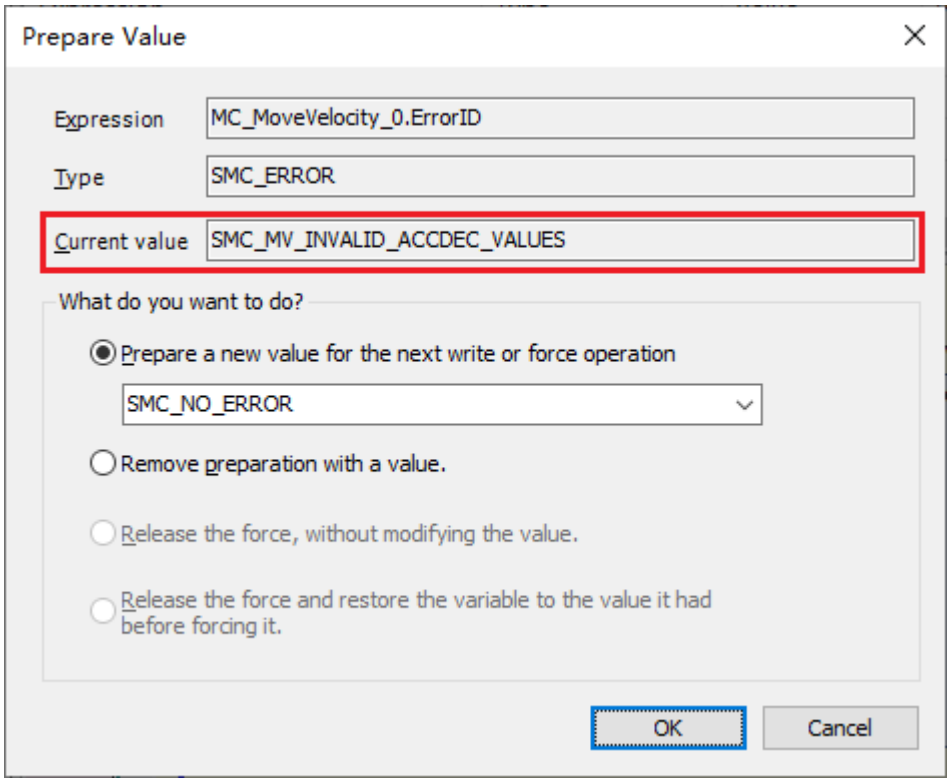
“TRUE” is output to the output Error and an error is output to the output ErrorID.



The error name can be checked by double-clicking the output ErrorID.

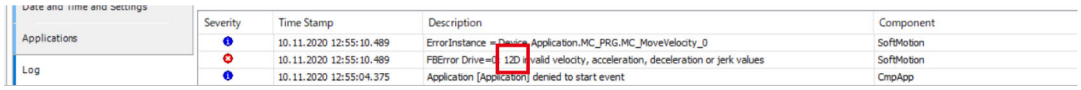
An error name defined in the enumeration type SMC_ERROR is displayed in the "Current value" field in the "Presetting values" dialog box.

Double-clicking SMC_MV_INV in the above execution example displays the following dialog box where error name "SMC_MV_INVALID_ACCDEC_VALUES" can be checked.



When an error occurs, the value of the error that has occurred (SMC_ERROR) is also recorded in hexadecimal number on the "Log" screen of the device editor.

The following example shows a record when an error ("SMC_MV_INVALID_ACCDEC_VALUES") with an error value of 12D (301 in decimal) has occurred.



13.1.2 SMC_ERROR

Error name	Value	Description
SMC_NO_ERROR	0	No error
SMC_DI_GENERAL_COMMUNICATION_ERROR	1	Communication error Communication disconnection or another communication problem occurred.
SMC_DI_AXIS_ERROR	2	Axis error Amplifier alarm or another axis problem occurred.
SMC_DI_FIELDBUS_LOST_SYNCRONICITY	3	The fieldbus lost synchronicity.
SMC_DI_SWLIMITS_EXCEEDED	10	The software limit has been exceeded.
SMC_DI_HWLIMITS_EXCEEDED	11	The hardware end switch is active.

Error name	Value	Description
SMC_DI_LINEAR_AXIS_OUTOFRANGE	12	An overflow occurred in the linear axis.
SMC_DI_HALT_OR_QUICKSTOP_NOT_SUPPORTED	13	The drive state is set to Halt or the Quickstop is unsupported.
SMC_DI_VOLTAGE_DISABLED	14	No power is supplied to the drive.
SMC_DI_IRREGULAR_ACTPOSITION	15	This error is not used.
SMC_DI_POSITIONLAGERROR	16	Position lag error The difference between the commanded position and actual position has exceeded the specified limit when position lag monitoring is active.
SMC_DI_HOMING_ERROR	17	Home return error
SMC_REGULATOR_OR_START_NOT_SET	20	Either the controller is disabled or the brake has been applied. Servo OFF or another similar problem occurred during axis movement.
SMC_WRONG_CONTROLLER_MODE	21	The executed function block is set to unsupported controller mode (SMC_SetControllerMode).
SMC_INVALID_ACTION_FOR_LOGICAL	25	Invalid operation was performed on the logical axis.
SMC_FB_WASNT_CALLED_DURING_MOTION	30	The function block was not called on the POU while the motion instruction was being executed ("Busy"). The operation was stopped while the motion instruction was being executed ("Busy").
SMC_AXIS_IS_NO_AXIS_REF	31	The type of the AXIS_REF type variable is different.
SMC_AXIS_REF_CHANGED_DURING_OPERATION	32	The AXIS_REF variable was changed during operation.
SMC_FB_ACTIVE_AXIS_DISABLED	33	The axis became disabled (MC_Power.bRegulatorOn) during movement.
SMC_AXIS_NOT_READY_FOR_MOTION	34	The axis cannot execute a motion instruction (an attempt was made to execute a motion instruction with MC_Stop enabled, for example).
SMC_AXIS_ERROR_DURING_MOTION	35	An error (such as amplifier alarm) occurred during motion operation.
SMC_VD_MAX_VELOCITY_EXCEEDED	40	The maximum velocity (fMaxVelocity) was exceeded.
SMC_VD_MAX_ACCELERATION_EXCEEDED	41	The maximum acceleration (fMaxAccleration) was exceeded.
SMC_VD_MAX_DECELERATION_EXCEEDED	42	The maximum deceleration (fMaxDeceleration) was exceeded.
SMC_3SH_INVALID_VELACC_VALUES	50	Either an invalid velocity or acceleration was specified.
SMC_3SH_MODE_NEEDS_HWLIMIT	51	For safety reasons, a request is made to invoke the mode in which the end switch is used.
SMC_FRC_NO_FREE_HANDLE	60	There is no file open handle.

Error name	Value	Description
SMC_MAC_INITIALIZATION_FAILED	65	SMC_MultiAcyclicCommunicator initialization failed.
SMC_MAC_INVALID_TASK_HANDLE	66	There is an invalid handle for the axis.
SMC_MAC_TOO_MANY_TASKS	67	There are too many tasks that use an axis generating SDO.
SMC_MAC_ATOMIC_ADD_FAILED	68	An attempt to add Atomic failed.
SMC_SDO_INVALID_DATALENGTH	69	An invalid data length (> 4) occurred due to SDO reading.
SMC_SCM_NOT_SUPPORTED	70	An invalid controller mode was set for SMC_SetControllerMode.
SMC_SCM_AXIS_IN_WRONG_STATE	71	The controller mode cannot be changed in the current axis state (an attempt was made to execute SMC_SetControllerMode with MC_Stop enabled, for example).
SMC_SCM_INTERRUPTED	72	SMC_SetControllerMode was interrupted by MC_Stop or ErrorStop.
SMC_ST_WRONG_CONTROLLER_MODE	75	The motion instruction was executed in an incorrect controller mode.
SMC_RAG_ERROR_DURING_STARTUP	80	An error occurred when the axis group was started up.
SMC_RAG_ERROR_AXIS_NOT_INITIALIZED	81	The axis is not in the specified state.
SMC_PP_WRONG_AXIS_TYPE	85	The function block does not support virtual axes or logical axes.
SMC_PP_NUMBER_OF_ABSOLUTE_BITS_INVALID	86	The number of bits is invalid (between 8 and 32 bits).
SMC_CGR_ZERO_VALUES	90	An invalid value was specified.
SMC_CGR_DRIVE_POWERED	91	A gear parameter was changed while the drive was in operation.
SMC_CGR_INVALID_POSPERIOD	92	An invalid position (0 or less, or half or more than the bus bandwidth) was specified.
SMC_CGR_POSPERIOD_NOT_INTEGRAL	93	The modulo period is not an integer.
SMC_P_FTASKCYCLE_EMPTY	110	There is no information in one cycle time. (fTaskCycle = 0)
SMC_R_NO_ERROR_TO_RESET	120	There is no error to be reset (MC_Reset was executed when there was no function block error, for example).
SMC_R_DRIVE_DOESNT_ANSWER	121	There is no response to an error reset.
SMC_R_ERROR_NOT_RESETTABLE	122	An error reset cannot be executed.
SMC_R_DRIVE_DOESNT_ANSWER_IN_TIME	123	Communication with the axis is not working.
SMC_R_CANNOT_RESET_COMMUNICATION_ERROR	124	A reset cannot be executed due to a communication error.
SMC_RP_PARAM_UNKNOWN	130	The parameter number is undefined.
SMC_RP_REQUESTING_ERROR	131	An error occurred in communication with the drive.
SMC_RP_DRIVE_PARAMETER_NOT_MAPPED	132	Parameters are not assigned to the drive.

Error name	Value	Description
SMC_RP_PARAM_CONVERSION_ERROR	133	Conversion of drive parameter values failed. Soft motion parameters are undefined.
SMC_WP_PARAM_INVALID	140	The parameter number is undefined or write operations are inhibited.
SMC_WP_SENDING_ERROR	141	Refer to the error number for WriteDriveParameter.
SMC_WP_DRIVE_PARAMETER_NOT_MAPPED	142	Parameters are undefined for the drive.
SMC_WP_PARAM_CONVERSION_ERROR	143	Conversion of drive parameter values failed. Soft motion parameters are undefined.
SMC_H_AXIS_WASNT_STANDSTILL	170	The axis is not in a standstill state.
SMC_H_AXIS_DIDNT_START_HOMING	171	An error occurred when home return was started.
SMC_H_AXIS_DIDNT_ANSWER	172	An error occurred when home return was started.
SMC_H_ERROR_WHEN_STOPPING	173	An error occurred after the axis stopped in home return mode. It is possible that deceleration was not set.
SMC_H_AXIS_IN_ERRORSTOP	174	The drive is in the ErrorStop state. Home return cannot be executed.
SMC_MS_UNKNOWN_STOPPING_ERROR	180	Undefined error
SMC_MS_INVALID_ACCDEC_VALUES	181	Either an invalid velocity or acceleration was specified.
SMC_MS_DIRECTION_NOT_APPLICABLE	182	"shortest" cannot be applied to the direction.
SMC_MS_AXIS_IN_ERRORSTOP	183	Because the drive is in the ErrorStop state, stop operation cannot be executed with MC_Stop.
SMC_BLOCKING_MC_STOP_WASNT_CALLED	184	MC_Stop (Execute=TRUE) blocks the axis. MC_Stop (Execute=FALSE) must be executed.
SMC_UNKNOWN_TASK_INTERVAL	200	The task interval of the bus task is undefined.
SMC_MA_INVALID_VELACC_VALUES	201	Either an invalid velocity or acceleration was specified.
SMC_MA_INVALID_DIRECTION	202	There is an error in the specified direction ("Direction").
SMC_MR_INVALID_VELACC_VALUES	226	Either an invalid velocity or acceleration was specified.
SMC_MR_INVALID_DIRECTION	227	There is an error in the specified direction ("Direction").
SMC_MAD_INVALID_VELACC_VALUES	251	Either an invalid velocity or acceleration was specified.
SMC_MAD_INVALID_DIRECTION	252	There is an error in the specified direction ("Direction").
SMC_MSI_INVALID_VELACC_VALUES	276	Either an invalid velocity or acceleration was specified.
SMC_MSI_INVALID_DIRECTION	277	There is an error in the specified direction ("Direction").
SMC_MSI_INVALID_EXECUTION_ORDER	278	The same instance of MC_MoveSuperImposed was called more than once in a single cycle.

Error name	Value	Description
SMC_LOGICAL_NO_REAL_AXIS	300	Unused error
SMC_MV_INVALID_ACCDEC_VALUES	301	Either an invalid velocity or acceleration was specified.
SMC_MV_DIRECTION_NOT_APPLICABLE	302	"shortest" or "fastest" cannot be applied to the direction ("Direction").
SMC_PP_ARRAYSIZE	325	There is an error in the specified array size.
SMC_PP_STEP0MS	326	The step time is 0s.
SMC_VP_ARRAYSIZE	350	There is an error in the specified array size.
SMC_VP_STEP0MS	351	The step time is 0s.
SMC_AP_ARRAYSIZE	375	There is an error in the specified array size.
SMC_AP_STEP0MS	376	The step time is 0s.
SMC_TP_TRIGGEROCCUPIED	400	The trigger is already enabled.
SMC_TP_COULDNT_SET_WINDOW	401	The drive does not support the window function.
SMC_TP_COMM_ERROR	402	Communication error
SMC_AT_TRIGGERNOTOCCUPIED	410	The trigger is already disabled.
SMC_MCR_INVALID_VELACC_VALUES	426	Either an invalid velocity or acceleration was specified.
SMC_MCR_INVALID_DIRECTION	427	An invalid direction was specified.
SMC_MCA_INVALID_VELACC_VALUES	451	Either an invalid velocity or acceleration was specified.
SMC_MCA_INVALID_DIRECTION	452	An invalid direction was specified.
SMC_MCA_DIRECTION_NOT_APPLICABLE	453	"fastest" cannot be applied to the direction ("Direction").
SMC_SDL_INVALID_AXIS_STATE	475	Function block "SMC_ChangeDynamicLimits" was executed in a state other than "Standstill" or "Power_off".
SMC_SDL_INVALID_VELACC_VALUES	476	An invalid velocity, acceleration, deceleration, or jerk was specified.
SMC_CR_NO_TAPPETS_IN_CAM	600	The cam is not equipped with a tappet.
SMC_CR_TOO_MANY_TAPPETS	601	The tappet group ID exceeds MAX_NUM_TAPPETS.
SMC_CR_MORE_THAN_32_ACCESSES	602	There are 32 or more accesses to one cam.
SMC_CI_NO_CAM_SELECTED	625	No cam is selected. It is possible that the correct cam table is not set in the CamTableID parameter of MC_CamIn.
SMC_CI_MASTER_OUT_OF_SCALE	626	The current commanded position on the master axis is outside the range of the cam table.
SMC_CI_RAMPIN_NEEDS_VELACC_VALUES	627	A velocity and acceleration must be specified when StartMode is set to ramp_in.
SMC_CI_SCALING_INCORRECT	628	The scaling variables (fEditor, TableMasterMin, and Max) are incorrect.
SMC_CI_TOO_MANY_TAPPETS_PER_CYCLE	629	The number of tappet outputs is too many to be enabled in one cycle.

Error name	Value	Description
SMC_CB_NOT_IMPLEMENTED	640	The function block for cam format is not implemented.
SMC_GI_RATIO_DENOM	675	RatioDenominator (denominator of the gear ratio) is set to 0.
SMC_GI_INVALID_ACC	676	The value specified in "Acceleration" is invalid.
SMC_GI_INVALID_DEC	677	The value specified in "Deceleration" is invalid.
SMC_GI_MASTER_REGULATOR_CHANGED	678	The Enable / Disable state of the master axis was changed without permission.
SMC_GI_INVALID_JERK	679	The value specified in "Jerk" is invalid.
SMC_PH_INVALID_VELACCDEC	725	The specified velocity, acceleration, or deceleration were invalid.
SMC_PH_ROTARYAXIS_PERIOD0	726	fPositionPeriod for the rotation axis is set to 0.
SMC_NO_CAM_REF_TYPE	750	The cam type is not an MC_CAM_REF structure.
SMC_CAM_TABLE_DOES_NOT_COVER_MASTER_SCALE	751	The master axis area (xStart and xEnd) of the cam table is outside the curve data range.
SMC_CAM_TABLE_EMPTY_MASTER_RANGE	752	The cam data table is empty.
SMC_CAM_TABLE_INVALID_MASTER_MINMAX	753	The maximum value and minimum value of the master axis in the cam data are invalid.
SMC_CAM_TABLE_INVALID_SLAVE_MINMAX	754	The maximum value and minimum value of the slave axis in the cam data are invalid.
SMC_GIP_MASTER_DIRECTION_CHANGE	775	The rotation direction of the master axis was changed while the salve axis was connected.
SMC_GIP_SLAVE_REVERSAL_CANNOT_BE_AVOIDED	776	The AvoidReversal input is set, but reverse rotation of the slave axis cannot be avoided.
SMC_GIP_AVOID_REVERSAL_FOR_FINITE_AXIS	777	The AvoidReversal input cannot be set for the finite slave axis.
SMC_BC_BL_TOO_BIG	800	The fBacklash gear backlash is too large (larger than position period/2).
SMC_QPROF_DIVERGES	825	Internal error: Failed in calculating the secondary path
SMC_QPROF_INVALID_PARAMETER	826	Internal error: Failed in calculating the secondary path
SMC_QPROF_NO_RESULT	827	Internal error: Failed in calculating the secondary path
SMC_QPROF_INVALID_NEW_LBD	828	Internal error: Failed in calculating the secondary path
SMC_QPROF_BAD_NEGOTIATION	829	Internal error: Failed in calculating the secondary path
SMC_QPROF_INVALID_INTERVAL	830	Internal error: Failed in calculating the secondary path
SMC_QPROF_NOT_ENOUGH_PHASES	831	Internal error: Failed in calculating the secondary path
SMC_TG_INTERNAL_ERROR	832	Internal error: Failed in calculating the secondary path

Error name	Value	Description
SMC_SRT_NOT_STANDSTILL_OR_POWEROFF	850	Execution is possible only in the standstill or power_off state.
SMC_SRT_INVALID_RAMPTYPE	851	The value specified in RampType is invalid.
SMC_SMT_NOT_STANDSTILL_OR_POWEROFF	852	Execution is possible only in the standstill or power_off state.
SMC_SMT_INVALID_MOVEMENTTYPE_OR_POSITIONPERIOD	853	The value specified in MovementType or PositionPeriod is invalid.
SMC_SMT_AXIS_NOT_VIRTUAL	854	The function block is valid only for the virtual axis.
SMC_NO_LICENSE	1000	License error
SMC_INT_VEL_ZERO	1001	Because Velocity is set to 0, path processing cannot be performed.
SMC_INT_NO_STOP_AT_END	1002	The final velocity of the path is other than 0.
SMC_INT_DATA_UNDERRUN	1003	GEOINFO-List was processed by DataIn, but the end of the list has not been reached.
SMC_INT_VEL_NONZERO_AT_STOP	1004	The velocity at the time of stoppage is greater than 0.
SMC_INT_TOO_MANY_RECURSIONS	1005	There are too many recursions of SMC_Interpolator.
SMC_INT_NO_CHECKVELOCITIES	1006	SMC_CheckVelocities is not called by SMC_OUTQUEUE.
SMC_INT_PATH_EXCEEDED	1007	Internal error or calculation error
SMC_INT_VEL_ACC_DEC_ZERO	1008	The specified velocity and acceleration / deceleration are 0 or less.
SMC_INT_DWIPOTIME_ZERO	1009	The motion task was called when dwIpoTime = 0.
SMC_INT_JERK_NONPOSITIVE	1010	A negative value was set for "Jerk".
SMC_INT_QPROF_DIVERGES	1011	Internal error The calculation algorithm is incorrect.
SMC_INT_INVLALID_VELOCITY_MODE	1012	The specified velocity mode is invalid.
SMC_INT_TOO_MANY_AXES_INTERPOLATED	1013	The maximum allowable number of axes for interpolation has been exceeded.
SMC_INT_DEGENERATE_SEGMENT	1014	
SMC_INT2DIR_BUFFER_TOO_SMALL	1050	
SMC_INT2DIR_PATH_FITS_NOT_IN_QUEUE	1051	
SMC_XINT_INVALID_DIRECTION	1070	
SMC_XINT_NOINTERSECTION	1071	
SMC_WAR_INT_OUTQUEUE_TOO_SMALL	1080	
SMC_WAR_END_VELOCITIES_INCORRECT	1081	The specified final velocity is incorrect.
SMC_CV_ACC_DEC_VEL_NONPOSITIVE	1100	Negative values are specified for the velocity and acceleration/deceleration.
SMC_CA_INVALID_ACCDEC_VALUES	1120	Negative values are specified for fGapVelocity, fGapAcceleration, and fGapDeceleration.
SMC_DEC_ACC_TOO_LITTLE	1200	The specified acceleration is unacceptable.

Error name	Value	Description
SMC_DEC_RET_TOO_LITTLE	1201	The specified deceleration is unacceptable.
SMC_DEC_OUTQUEUE_RAN_EMPTY	1202	Data underrun The queue was read, but it was empty.
SMC_DEC_JUMP_TO_UNKNOWN_LINE	1203	Because the line number is unknown, the cursor cannot jump to the line.
SMC_DEC_INVALID_SYNTAX	1204	The syntax is invalid.
SMC_DEC_3DMODE_OBJECT_NOT_SUPPORTED	1205	The object is not supported in 3D mode.
SMC_DEC_NEGATIVE_PERIOD	1206	A negative value is specified for the period during which an additional axis is disabled.
SMC_DEC_DIMENSIONS_EXCLUSIVE_AU	1207	Both axis A and axis U are not always interpolated. PA and PU are mutually exclusive.
SMC_DEC_DIMENSIONS_EXCLUSIVE_BV	1208	Both axis B and axis V are not always interpolated. PB and PV are mutually exclusive.
SMC_DEC_DIMENSIONS_EXCLUSIVE_CW	1209	Both axis C and axis W are not always interpolated. PC and PW are mutually exclusive.
SMC_IPR_TOO_SMALL__BUFFER	1259	The buffer size specified for OutOueue is too small.
SMC_GCV_BUFFER_TOO_SMALL	1300	
SMC_GCV_BUFFER_WRONG_TYPE	1301	
SMC_GCV_UNKNOWN_IPO_LINE	1302	
SMC_NO_CNC_REF_TYPE	1500	
SMC_NO_OUTQUEUE_TYPE	1501	The specified pointer is not SMC_OUTQUEUE.
SMC_GEOINFO_BUFFER_MISALIGNED	1502	The buffer segments aligned by four-byte boundaries are not used by pbyBuffer.
SMC_3D_MODE_NOT_SUPPORTED	1600	The FB functions only with 2D paths.

13.2 RTEX communication error

13.2.1 RTEX Error ID

■ WARNING_CODE (Union type)

Member	Type	Description
uiWarningCode	UINT	Warning code
tWarningCodeMember	ALARM_WARNING_CODES	Main code (warning number) and sub-code (0) of the warning code

■ List of RTEX Error IDs

Category	Error_Code / Sub_Error_Code	Cause
Command header related	0011h	● Mismatched node address (MAC-ID)
	0012h	● C/R bit is 1 despite of command. ● Sub_Chk is 0 in 32-byte mode.
Command code, control mode related	0021h	● Cyclic command is not defined.
	0022h	● Non-cyclic command is not defined (when cyclic command is normal). ● Combination error of control mode and non-cyclic command. ● Subcommand is undefined in 32-byte mode.
	002Eh	● Combination of communication cycle, semi-closed/full-closed, 16 / 32 byte mode, and control mode is not correct. ● Control mode has been changed in less than 2 ms. ● Control mode has been changed during profile position latch positioning / profile home return (Type_Code = 12h, 13h, 31h, 32h, 33h, 34h, 36h) operation. ● Control mode has been changed during execution of non-cyclic command (Busy = 1). ● Run the home return command (□4h) Type_Code = 1□h / 2□h during the velocity control (CV) / torque control (CT). ● Control mode has been changed to the velocity control during the 2 degrees of freedom control (synchronous) mode. ● Control mode has been changed to the torque control during the 2 degrees of freedom control (standard / synchronous) mode. ● Control mode has been changed during the retracting operation.
Argument related	0031h	● Type_Code / Sub_Type_Code is not defined.
	0032h	● Non-cyclic data / sub-command data other than Type_Code / Sub_Type_Code is out of setup range.
	0033h	● Cyclic data (Command_Data1) is out of setup range.
	0034h	● Feed forward data (Command_Data3, Sub_Command_Data2 / 3) is out of setup range.
Not executable 1 (general)	0041h	● Write access is attempted to read only media.
	0042h	● Alarm clear command is executed while an alarm that cannot be cleared has occurred and no warning was issued.

Category	Error_Code / Sub_Error_Code	Cause
	0043h	● External scale error clear command is executed when not in full-closed control mode or when no external scale error is detected.
	0045h	● In servo on state, reset command is executed in attribute C parameter validation mode.
	0046h	● After deceleration and stop according to the drive inhibit input (POT / NOT), direction command POT / NOT is applied. ● During deceleration according to the drive inhibit input (POT / NOT), a profile operation (except Type_Code = 31h, 32h, 33h, 34h, and 36h) is started.
Not executable 2 (related to home return)	0051h	● Multi-turn clearing of the home return command was executed while the encoder was in the incremental mode. ● Multi-turn clearing of the home return command was executed even when the single-turn absolute function was effective.
	0052h	● During cyclic position control (CP) (* including full-closed control) in absolute mode, Type_Code = 1□h of the home return command (□4h) has been executed. ● During profile position control (PP) (* including full-closed control) in absolute mode, profile home return has been executed.
	0053h	● During cyclic position control (CP) (* including full-closed control) in absolute mode, actual position set / command position set (Type_Code = 21h, 22h) of the home return command (□4h) have been executed.
	0055h	● Multi-turn clearing of the home return command is executed while in the full-closed control mode.
	0056h	● Multi-turn clearing of the home return command is executed while in the servo-on condition.
	0057h	● Type_Code = 1□h of the home return command is executed while in the servo-off state.
	0058h	● While the external input is not assigned to the latch correction terminal, Type_Code is executed by using the external input as a trigger. ● Started the latch mode with a stop function operated by the amplifier output signal as the trigger signal when Pr7.111 “Trigger signal assignment setting for the latch mode with a stop function” was set to 0 “Disabled”.
	0059h	● Executed the home return command (□4h) while the profile position latch positioning / profile home return (Type_Code = 12h, 13g, 31h, 32h, 33h, 34h, 36h) was operated. ● During profile positioning / profile continuous revolution (Type_Code = 10h, 11h, 20h), initialization mode (Type_Code = 1□h, 31h) of home return command (□4h) has been executed.
	005Ah	● Z phase is set to latch trigger signal despite absolute external scale.
	005Bh	● Received the following commands in the virtual full-closed control mode. • Home return command (□4h) • Profile position latch absolute positioning (12h) of the profile command (17h) • Profile position latch absolute positioning (13h) of the profile command (17h) • Profile home return (31h to 34h, 36h) of the profile command (17h) • Config command

Category	Error_Code / Sub_Error_Code	Cause
		● Received a command to change to the virtual full-closed control mode under the following conditions. • While initialization mode of home return command (□4h) was operated, latch mode was operated, or latch mode with stop function was operated • Changed to a command other than command code (□4h) after starting home return command (Type_Code: 51h to 53h) • During a period from starting the latch to detecting the latch after starting home return command (Type_Code: 51h to 53h) • While profile position latch absolute positioning (12h) of the profile command (17h) was operated • While profile position latch absolute positioning (13h) of the profile command (17h) was operated • While profile home return (31h to 34h, 36h) of the profile command (17h) was operated • After starting profile command (12h, 13h, 31h to 34h, 36h), during the period from when a change was made to a command other than command code (17h) until the latch or home was detected • While Config command was executed
	005Fh	● Latch mode with stop function (Type_Code = F1h) was used in a setting other than the cyclic position control (CP). ● Latch mode with stop function (Type_Code = F1h) was used in a setting other than the communication cycle of 0.5 ms/command update cycle of 1.0 ms ● Latch mode with stop function (Type_Code = F1h) was used in a setting other than the electronic gear ratio of less than 1.
Not executable 3 (Related to hardware factor)	0061h	● EEPROM writing is not permitted because of under voltage of the control power.
Not executable 4 (in process)	0101h	● Not permitted to be accepted because the previous command is in process.
	0102h	● Command is not permitted to be accepted because the servo driver is accessing to the encoder now.
	0103h	● Command is not permitted to be accepted because the servo driver is accessing to the external scale now.
	0104h	● Type_Code has been changed while operating under profile position control (PP).
	0105h	● During execution of the PANATERM command (test run operation, FFT, Z phase search, pin assignment setting, or fit gain), received the RTEX command (reset command, home return command, or parameter command).
Not executable 5 (access inhibited)	0201h	● Command is not permitted to be accepted because parameter writing or writing to EEPROM is inhibited now. ● Write parameter command or write EEPROM command is issued while bit 0 of Pr7.23 RTEX function expansion setup 2 is set at 1.

13.2.2 Alarm Codes

■ ALARM_CODE (Union type)

Member	Type	Description
uiAlarmCode	UINT	Alarm code
tAlarmCodeMember	ALARM_WARNING_CODES	Main code and sub-code of the alarm code

■ ALARM_WARNING_CODES (Structure)

Member	Type	Description
byMainCode	BYTE	Main alarm number code
bySubCode	BYTE	Sub alarm number code

■ List of alarm codes

Error No.		Alarm name	Attribute		
Main	Sub		History	Can be cleared	Emergency stop(Note 6)
11	0	Control power supply undervoltage protection		○	
12	0	Over-voltage protection	○	○	
13	0	Main power supply undervoltage protection (Insufficient voltage between P and N)		○	○
	1	Main power supply undervoltage protection (AC interception detection)		○	○
14	0	Over-current protection	○		
	1	IPM error protection	○		
15	0	Overheat protection	○		○
	1	Encoder overheat error protection	○		○
16	0	Overload protection	○	○(Note 1)	
	1	Torque saturation error protection	○	○	
18	0	Regenerative overload protection	○		○
	1	Regenerative transistor error protection	○		
21	0	Encoder communication line breakage fault protection	○		
	1	Encoder communication error protection	○		
23	0	Encoder communication data error protection	○		
24	0	Position deviation excess protection	○	○	○
	1	Speed deviation excess protection	○	○	○
25	0	Hybrid deviation excess protection	○		○
26	0	Overspeed protection	○	○	○
	1	2nd overspeed protection	○	○	

Error No.		Alarm name	Attribute		
Main	Sub		History	Can be cleared	Emergency stop(Note 6)
27	1	Absolute clearing protection	○		
	4	Command error protection	○		○
	5	Command generation error protection	○		○
	6	Operation command contention protection	○	○	
	7	Position information initialization error protection	○		
28	0	Pulse regeneration limit protection	○	○	○
29	1	Counter overflow protection 1	○		
	2	Counter overflow protection 2	○		
31	0	Safety function error protection 1	○		
	2	Safety function error protection 2	○		
33	0	Input duplicated allocation error-1 protection	○		
	1	Input duplicated allocation error-2 protection	○		
	2	Input function number error-1 protection	○		
	3	Input function number error-2 protection	○		
	4	Output function number error-1 protection	○		
	5	Output function number error-2 protection	○		
	8	Latch input allocation error protection	○		
34	0	Motor operable range setting error protection	○	○	
	1	One revolution absolute operable range error protection	○	○	
36	0 to 1	EEPROM parameter error protection			
37	0 to 2	EEPROM check code error protection			
38	0	Over-travel inhibit input protection 1		○	
	1	Over-travel inhibit input protection 2		○	
	2	Over-travel inhibit input protection 3	○		
40	0	Absolute system failure protection	○	○(Note 2)	
41	0	Absolute counter limit excess protection	○		
42	0	Absolute overspeed protection	○	○(Note 2)	
44	0	Single-turn counter error protection	○		
45	0	Multi-turn counter error protection	○		
47	0	Absolute status error protection	○		
50	0	External scale wiring error protection	○		
	1	External scale communication error protection	○		
	2	External scale communication data error protection	○		
51	0	External scale ST error protection 0	○		
	1	External scale ST error protection 1	○		

Error No.		Alarm name	Attribute		
Main	Sub		History	Can be cleared	Emergency stop(Note 6)
	2	External scale ST error protection 2	○		
	3	External scale ST error protection 3	○		
	4	External scale ST error protection 4	○		
	5	External scale ST error protection 5	○		
55	0	Phase-A wiring error protection	○		
	1	Phase-B wiring error protection	○		
	2	Phase-Z wiring error protection	○		
70	0	Phase U current detector error protection	○		
	1	Phase W current detector error protection	○		
72	0	Thermal relay error protection	○		
80	3	PLL incomplete error protection	○	○	
82	0	RTEX node addressing error protection	○		
83	0	RTEX continuous communication error protection 1	○	○	○
	1	RTEX continuous communication error protection 2	○	○	○
84	0	RTEX Communication timeout error protection	○	○	○
	3	RTEX communication synchronization error protection	○		
	5	RTEX communication cycle error protection	○	○	○
85	0	Retracting operation completion (I/O)(Note 7)	○	(Note 8)	○
	2	Retracting operation error(Note 7)	○	(Note 8)	○
86	0	RTEX cyclic data error protection 1	○	○	○
	1	RTEX cyclic data error protection 2	○	○	○
	2	RTEX update counter error protection	○		○
87	0	Forced alarm input protection		○	○
	1	Retracting operation completion (I/O)(Note 7)	○	(Note 8)	○
	3	Retracting operation error(Note 7)	○	(Note 8)	○
90	2	RTEX multi-axis synchronization establishment error protection	○		
91	1	RTEX command error protection	○	○	
	3	RTEX command error protection 2	○	○	
92	0	Encoder data restoration error protection	○		
	1	External scale data restoration error protection	○		
	3	Multi-turn data upper-limit value mismatch error protection	○		
93	0	Parameter setting error protection 1	○		
	2	Parameter setting error protection 2	○		

Error No.		Alarm name	Attribute		
Main	Sub		History	Can be cleared	Emergency stop(Note 6)
	3	External scale connection error protection	○		
	5	Parameter setting error protection 4	○		
	8	Parameter setting error protection 6	○		
94	2	Home return error protection	○	○	
	3	Home return error protection 2	○	○	
95	0 to 4	Motor automatic recognition error protection			
96	2	Control unit error protection 1	○		
	3	Control unit error protection 2	○		
	4	Control unit error protection 3	○		
	5	Control unit error protection 4	○		
	6	Control unit error protection 5	○		
	7	Control unit error protection 6	○		
98	1	RTEX hardware error protection 1	○		
	2	RTEX hardware error protection 2	○		
	3	RTEX hardware error protection 3	○		
Other numbers		Other error protections	-	-	-

(Note 1) When Err 16.0 “Over-load protection” occurs, it can be cleared approx. 10 seconds after it occurs. The alarm clear command is received as is and clearing process takes place after it is ready to be cleared.

(Note 2) When Err 40.0 “Absolute system failure protection” or Err 42.0 ”Absolute overspeed protection” occurs, the error cannot be cleared until absolute clear is performed.

(Note 3) When an alarm that cannot be cleared occurs, cycle the control power supply after removing the cause of the error or use RTEX software reset command to clear the alarm.

(Note 4) When an alarm that can be cleared occurs, use RTEX communication or USB communication (setup support software) to clear the alarm. Always clear the alarm while all axes are stopped and after securing safety.

(Note 5) If the internal control circuit of the servo amplifier malfunctions due to excessive noise etc., the display will be as shown below.

In such a case, immediately turn OFF the power.

(Note 6) Emergency stop refers to an alarm that is triggered if Pr 5.10 “Sequence at alarm” is set to 4 to 7 and that causes an immediate stop. For details, refer to the instruction manual and other technical references for the servo amplifier.

(Note 7) The alarm generated during retracting operation is switched by Pr 6.86 “Retreat operation alarm setup” bit 15.

Example: When bit 15 = 0, Err 85.0 and Err 85.2 will occur (A5N compatible specification).

When bit 15 = 1, Err 87.1 and Err 87.3 will occur (A6B compatible specification).

(Note 8) Whether alarm can be cleared or not is determined by the setting (bit 0 or 2) of Pr 6.86.

Bit 0: Err 85.0 / Err 87.1 (Retracting operation completion (I/O)) alarm clear attribute

Bit 2: Err 85.2 / Err 87.3 (Retracting operation error) alarm clear attribute; For either case, 0: Alarm clear invalid, 1: Alarm clear valid

13.2.3 Warning Codes

■ WARNING_CODE (Union type)

Member	Type	Description
uiWarningCode	UINT	Warning code
tWarningCodeMember	ALARM_WARNING_CODES	Main code and sub-code of the warning code

■ ALARM_WARNING_CODES (Structure)

Member	Type	Description
byMainCode	BYTE	Main alarm number code
bySubCode	BYTE	Sub alarm number code

■ General warnings

Warning No. (hexadecimal)	Warning name	Description	Warning latch	Output setting	Warning mask
			Pr6.27 (Note 1)	Pr4.40 / Pr4.41 (Note 2)	Pr6.38 / Pr6.39 Corresponding bit (Note 3)
A0	Overload warning Warning	Load factor is 85% or more of the protection level.	○	1	Pr6.38 bit7
A1	Over-regeneration warning	Regenerative load factor has exceeded 85% of the protection level.	○	2	Pr6.38 bit5
A2	Battery warning (Note 4)	Battery voltage is 3.2 V or less.	Latch fixed	3	Pr6.38 bit0
A3	Fan warning	Fan has stopped for 1 second.	○	4	Pr6.38 bit6
A4	Encoder communication warning	The number of successive encoder communication errors has exceeded the specified value.	○	5	Pr6.38 bit4
A5	Encoder overheat warning	The encoder temperature exceeds the specified value.	○	6	Pr6.38 bit3
A6	Oscillation detection warning	Oscillation state was detected.	○	7	Pr6.38 bit13

Warning No. (hexadecimal)	Warning name	Description	Warning latch	Output setting	Warning mask
			Pr6.27 (Note 1)	Pr4.40 / Pr4.41 (Note 2)	Pr6.38 / Pr6.39 Corresponding bit (Note 3)
A7	Lifetime detection warning	The remaining life expectancy of a capacitor or a fan dropped below the specified value.	Latch fixed	8	Pr6.38 bit2
A8	External scale error warning	The external scale detected a warning.	○	9	Pr6.38 bit8
A9	External scale communication warning	The number of successive external scale communication errors has exceeded the specified value.	○	10	Pr6.38 bit14
AC	Deterioration diagnosis warning (Note 6)	Load characteristic estimated value or torque command value at a constant velocity has exceeded the set range.	○	22	Pr6.39 bit7

■ Extended warning

Warning No. (hexadecimal)	Warning name	Description	Warning latch	Output setting	Warning mask
			Pr6.27 (Note 1)	Pr4.40 / Pr4.41 (Note 2)	Pr6.38 / Pr6.39 Corresponding bit (Note 3)
C0	RTEX continuous communication error warning	The number of successive errors (CRC error) detected when reading the received data sent to the local node The number of successive errors (CRC error) has exceeded the value set by Pr 7.26 “RTEX continuous communication error warning setup”.	○	11	Pr6.38 bit9
C1	RTEX accumulated communication error warning	The number of successive errors (CRC error) detected when reading the received data sent to the local node has exceeded the value set by Pr 7.27 “RTEX accumulated communication error warning setup”.	Latch fixed	12	Pr6.38 bit10
C2	RTEX_ Update_Counter error warning	The Update_Counter was not updated properly because the data accumulated exceeded the count value set by Pr 7.28 “RTEX_Update_Counter error warning setup”.	Latch fixed	13	Pr6.38 bit11
C3	Main power OFF warning	When Pr 7.14 “Main power OFF warning detection time” was set to 10 to 1999, instantaneous power failure	○	14	Pr6.38 bit12

Warning No. (hexadecimal)	Warning name	Description	Warning latch	Output setting	Warning mask
			Pr6.27 (Note 1)	Pr4.40 / Pr4.41 (Note 2)	Pr6.38 / Pr6.39 Corresponding bit (Note 3)
		that occurred between L1 and L3 exceeded the time set by Pr 7.14.			
D2	PANATERM command execution warning	While bit 0 of Pr 7.99 “RTEX function enhancement setting 6” was set to 1 and RTEX communication was established, an operation command (test run, FFT, etc.) was executed by the setup support software “PANATERM”.	○	30	Pr6.39 bit8

(Note 1) The symbol “○” marked in the “Warning latch” column indicates that it is possible to switch the mode between non-latch mode (latch for 1 second) and latch mode by using Pr 6.27 ”Warning latch state setup”. Only latch mode is available for the battery warning and the lifetime detection warning.

(Note 2) Select the warning output signal 1 (WARN 1) or warning output signal 2 (WARN 2) through Pr 4.40 “Warning output select 1” or Pr 4.41 “Warning output select 2”. When the set value is 0, all warnings are ORed before being output. Do not use any settings other than the settings shown in the above table.

(Note 3) Each warning detection can be disabled by Pr 6.38 “Warning mask setting” or Pr 6.39 “Warning mask setting 2”. The corresponding bits are shown in the table. Set the bit to 1 to disable the warning detection. For extended warning, warning detection can be disabled by parameter settings.

Also note that bit arrangements of these masks are different from those of general-purpose type MINAS-A6 series.

(Note 4) When the single-turn absolute function is enabled, a battery alarm is not detected.

(Note 5) Warning can be cleared by alarm clear. If warning cause is not resolved yet, the warning is cleared once, but a warning is issued again.

(Note 6) If bit 1 of Pr 6.97 “Function enhancement setup 3” is set to 0, it is disabled.

13.3 List of AMP Parameters

13.3.1 Class 0: Basic Setting

Class	No.	Parameter name	Unit	Setting range
0	00	Rotational direction setup	-	0 to 1
	01	Control mode setup	-	0 to 6
	02	Real-time auto-gain tuning setup	-	0 to 6
	03	Selection of machine stiffness at real-time auto-gain tuning	-	0 to 31
	04	Inertia ratio	%	0 to 10000
	08	Command pulse counts per one motor revolution	pulse	0 to 2^{23}
	09	Numerator of electronic gear	-	0 to 2^{30}
	10	Denominator of electronic gear	-	1 to 2^{30}
	11	Number of output pulses per motor rotation	pulse/r	1 to 2097152
	12	Reversal of pulse output logic / output source selection	-	0 to 3
	13	1st torque limit	%	0 to 500
	14	Position deviation excess setup	Command unit	0 to 2^{30}
	15	Absolute encoder setup	-	0 to 4
	16	External regenerative resistor setup	-	0 to 3
	17	Load factor of external regenerative resistor selection	-	0 to 4

13.3.2 Class 1: Gain Adjustment

Class	No.	Parameter name	Unit	Setting range
1	00	1st gain of position loop	0.1/s	0 to 30000
	01	1st gain of velocity loop	0.1 Hz	1 to 32767
	02	1st time constant of velocity loop integration	0.1 ms	1 to 10000
	03	1st filter of speed detection	-	0 to 5
	04	1st time constant of torque filter	0.01 ms	0 to 2500
	05	2nd gain of position loop	0.1/s	0 to 30000
	06	2nd gain of velocity loop	0.1 Hz	1 to 32767
	07	2nd time constant of velocity loop integration	0.1 ms	1 to 10000
	08	2nd filter of speed detection	-	0 to 5

Class	No.	Parameter name	Unit	Setting range
	09	2nd time constant of torque filter	0.01 ms	0 to 2500
	10	Velocity feed forward gain	0.1%	0 to 4000
	11	Velocity feed forward gain	0.1%	0 to 4000
	12	Velocity feed forward gain	0.01 ms	0 to 6400
	13	Torque feed forward filter	0.01 ms	0 to 6400
	14	2nd gain setup	-	0 to 1
	15	Mode of position control switching	-	0 to 10
	16	Delay time of position control switching	0.1 ms	0 to 10000
	17	Level of position control switching	-	0 to 20000
	18	Hysteresis at position control switching	-	0 to 20000
	19	Position gain switching time	0.1 ms	0 to 10000
	20	Mode of velocity control switching	-	0 to 5
	21	Delay time of velocity control switching	0.1 ms	0 to 10000
	22	Level of velocity control switching	-	0 to 20000
	23	Hysteresis at velocity control switching	-	0 to 20000
	24	Mode of torque control switching	-	0 to 3
	25	Delay time of torque control switching	0.1 ms	0 to 10000
	26	Level of torque control switching	-	0 to 20000
	27	Hysteresis at torque control switching	-	0 to 20000

13.3.3 Class 2: Vibration Suppression Function

Class	No.	Parameter name	Unit	Setting range
2	00	Adaptive filter mode setup	-	0 to 6
	01	1st notch frequency	Hz	50 to 5000
	02	1st notch width selection	-	0 to 20
	03	1st notch depth selection	-	0 to 99
	04	2nd notch frequency	Hz	50 to 5000
	05	2nd notch width selection	-	0 to 20
	06	2nd notch depth selection	-	0 to 99
	07	3rd notch frequency	Hz	50 to 5000
	08	3rd notch width selection	-	0 to 20
	09	3rd notch depth selection	-	0 to 99

Class	No.	Parameter name	Unit	Setting range
	10	4th notch frequency	Hz	50 to 5000
	11	4th notch width selection	-	0 to 20
	12	4th notch depth selection	-	0 to 99
	13	Selection of damping filter switching	-	0 to 6
	14	1st damping frequency	0.1 Hz	0 to 3000
	15	1st damping filter setup	0.1 Hz	0 to 1500
	16	2nd damping frequency	0.1 Hz	0 to 3000
	17	2nd damping filter setup	0.1 Hz	0 to 1500
	18	3rd damping frequency	0.1 Hz	0 to 3000
	19	3rd damping filter setup	0.1 Hz	0 to 1500
	20	4th damping frequency	0.1 Hz	0 to 3000
	21	4th damping filter setup	0.1 Hz	0 to 1500
	22	Command smoothing filter	0.1 ms	0 to 10000
	23	Command FIR filter	0.1 ms	0 to 10000
	24	5th notch frequency	Hz	50 to 5000
	25	5th notch width selection	-	0 to 20
	26	5th notch depth selection	-	0 to 99
	27	1st vibration control width setting	-	0 to 1000
	28	2nd vibration control width setting	-	0 to 1000
	29	3rd vibration control width setting	-	0 to 1000
	30	4th vibration control width setting	-	0 to 1000

13.3.4 Class 3: Speed, Torque Control, Full-closed Control

Class	No.	Parameter name	Unit	Setting range
3	12	Acceleration time setting	ms/(1000 r/min)	0 to 10000
	13	Deceleration time setting	ms/(1000 r/min)	0 to 10000
	14	Sigmoid acceleration/ deceleration time setup	ms	0 to 10000
	17	Speed limit selection	-	0 to 1
	21	Speed limit value 1	r/min	0 to 20000
	22	Speed limit value 2	r/min	0 to 20000
	23	External scale selection	-	0 to 6
	24	External scale numerator of division	-	0 to 2^{23}
	25	External scale denominator of division	-	1 to 2^{23}

Class	No.	Parameter name	Unit	Setting range
	26	External scale reversal of direction	-	0 to 3
	27	External scale Z phase disconnection detection disable	-	0 to 1
	28	Hybrid deviation excess protection	Command unit	1 to 2^{27}
	29	Hybrid deviation clear setting	Rotation	0 to 100
	32	External scale movement judgment threshold at virtual full-closed control mode	External scale unit	0 to 65534

13.3.5 Class 4: I/O Monitor Setting

Class	No.	Parameter name	Unit	Setting range
4	00	SI1 input selection	-	0 to 00FFFFFFh
	01	SI2 input selection	-	0 to 00FFFFFFh
	02	SI3 input selection	-	0 to 00FFFFFFh
	03	SI4 input selection	-	0 to 00FFFFFFh
	04	SI5 input selection	-	0 to 00FFFFFFh
	05	SI6 input selection	-	0 to 00FFFFFFh
	06	SI7 input selection	-	0 to 00FFFFFFh
	07	SI8 input selection	-	0 to 00FFFFFFh
	10	SO1 output selection	-	0 to 00FFFFFFh
	11	SO2 output selection	-	0 to 00FFFFFFh
	12	SO3 output selection	-	0 to 00FFFFFFh
	16	Type of analog monitor 1	-	0 to 28
	17	Analog monitor 1 output gain	-	0 to 214748364
	18	Type of analog monitor 2	-	0 to 28
	19	Analog monitor 2 output gain	-	0 to 214748364
	21	Analog monitor output setup	-	0 to 2
	31	Positioning complete range	Command unit	0 to 2097152
	32	Positioning complete (In-position) output setup	-	0 to 10
	33	INP hold time	ms	0 to 30000
	34	Zero-speed	r/min	10 to 20000
	35	Speed coincidence range	r/min	10 to 20000
	36	At-speed (Speed arrival)	r/min	10 to 20000

Class	No.	Parameter name	Unit	Setting range
	37	Mechanical brake action at stalling setup	ms	0 to 10000
	38	Mechanical brake action at running setup	ms	0 to 32000
	39	Brake release speed setup	r/min	30 to 3000
	40	Selection of alarm output 1	-	0 to 40
	41	Selection of alarm output 2	-	0 to 40
	42	2nd Positioning complete (In-position) range	Command unit	0 to 2097152
	44	Position compare output pulse width setting	0.1 ms	0 to 32767
	45	Position compare output polarity select	-	0 to 7
	47	Pulse output select	-	0 to 1
	48	Position compare value 1	Command unit	-2147483648 to 2147483647
	49	Position compare value 2	Command unit	-2147483648 to 2147483647
	50	Position compare value 3	Command unit	-2147483648 to 2147483647
	51	Position compare value 4	Command unit	-2147483648 to 2147483647
	52	Position compare value 5	Command unit	-2147483648 to 2147483647
	53	Position compare value 6	Command unit	-2147483648 to 2147483647
	54	Position compare value 7	Command unit	-2147483648 to 2147483647
	55	Position compare value 8	Command unit	-2147483648 to 2147483647
	56	Position compare output delay compensation amount	0.1 us	-32768 to 32767
	57	Position compare output assignment setting	-	-2147483648 to 2147483647

13.3.6 Class 5: Enhancing Setting

Class	No.	Parameter name	Unit	Setting range
5	03	Denominator of pulse output division	-	0 to 8388608
	04	Over-travel inhibit input setup	-	0 to 2
	05	Sequence at over-travel inhibit	-	0 to 2

Class	No.	Parameter name	Unit	Setting range
	06	Sequence at Servo-Off	-	0 to 9
	07	Sequence at main power OFF	-	0 to 9
	08	LV trip selection at main power OFF	-	0 to 3
	09	Detection time of main power off	ms	20 to 2000"13.3.9 Class 8: Special Setting 3"
	10	Sequence at alarm	-	0 to 7
	11	Torque setup for emergency stop	%	0 to 500
	12	Over-load level setup	%	0 to 500
	13	Over-speed level setup	r/min	0 to 20000
	14	Motor working range setup	0.1 revolution	0 to 1000
	15	Control input signal read setting	-	0 to 3
	20	Position setup unit select	-	0 to 1
	21	Selection of torque limit	-	0 to 4
	22	2nd torque limit	%	0 to 500
	23	Torque limit switching setup 1	ms/100 %	0 to 4000
	24	Torque limit switching setup 2	ms/100 %	0 to 4000
	25	Positive direction torque limit	%	0 to 500
	26	Negative direction torque limit	%	0 to 500
	31	USB axis address	-	0 to 127
	33	Pulse regenerative output limit setup	-	0 to 1
	45	Quadrant projection positive direction compensation value	0.1%	-1000 to 1000
	46	Quadrant projection negative direction compensation value	0.1%	-1000 to 1000
	47	Quadrant projection compensation delay time	ms	0 to 1000
	48	Quadrant projection compensation filter setting L	0.01 ms	0 to 6400
	49	Quadrant projection compensation filter setting H	0.1 ms	0 to 10000
	56	Slow stop deceleration time setting	ms/(1000 r/min)	0 to 10000

Class	No.	Parameter name	Unit	Setting range
	57	Slow stop S-shape acceleration and deceleration setting	ms	0 to 1000
	66	Deterioration diagnosis convergence judgment time	0.1 s	0 to 10000
	67	Deterioration diagnosis inertia ratio upper limit	%	0 to 10000
	68	Deterioration diagnosis inertia ratio lower limit	%	0 to 10000
	69	Deterioration diagnosis unbalanced load upper limit	0.1%	-1000 to 1000
	70	Deterioration diagnosis unbalanced load lower limit	0.1%	-1000 to 1000
	71	Deterioration diagnosis dynamic friction upper limit	0.1%	-1000 to 1000
	72	Deterioration diagnosis dynamic friction lower limit	0.1%	-1000 to 1000
	73	Deterioration diagnosis viscous friction upper limit	0.1%/ (10000 r/min)	0 to 10000
	74	Deterioration diagnosis viscous friction lower limit	0.1%/ (10000 r/min)	0 to 10000
	75	Deterioration diagnosis velocity setting	r/min	-20000 to 20000
	76	Deterioration diagnosis torque average time	ms	0 to 10000
	77	Deterioration diagnosis torque upper limit	0.1%	-1000 to 1000
	78	Deterioration diagnosis torque lower limit	0.1%	-1000 to 1000

(Note 1) When using this setup value at a value smaller than the default value, confirm that it matches the user's power supply environment.

13.3.7 Class 6: Special Setting 1

Class	No.	Parameter name	Unit	Setting range
6	02	Speed deviation excess setup	r/min	0 to 20000
	05	Position control 3rd gain effective time	0.1 ms	0 to 10000
	06	Position control 3rd gain scale factor	%	50 to 1000
	07	Additional value to torque command	%	-100 to 100

Class	No.	Parameter name	Unit	Setting range
	08	Torque compensation value in positive direction	%	-100 to 100
	09	Torque compensation value in negative direction	%	-100 to 100
	10	Function expansion setup	-	-32768 to 32767
	11	Current response setup	%	10 to 100
	14	Immediate stop time at the time of alarming	ms	1000
	15	2nd over-speed level setup	r/min	0 to 20000
	18	Power turn-on wait time	0.1 s	0 to 100
	22	A, B phase external scale pulse output method selection	-	0 to 1
	23	Load fluctuation correction gain	%	-100 to 100
	24	Load fluctuation correction filter	0.01 ms	10 to 2500
	27	Alarm latch time selection	-	0 to 3
	31	Real time auto tuning estimation speed	-	0 to 3
	32	Real time auto tuning custom setup	-	-32768 to 32767
	34	Hybrid vibration suppression gain	-	0 to 30000
	35	Hybrid vibration suppression filter	0.1/s	0 to 32000
	36	Dynamic brake operation input setup	0.01 ms	0 to 1
	37	Oscillation detecting level	-	0 to 1000
	38	Alarm mask setup	0.1%	-32768 to 32767
	39	Alarm mask setup 2	-	-32768 to 32767
	41	1st damping depth	-	0 to 1000
	42	Two-stage torque filter time constant	-	0 to 2500
	43	Two-stage torque filter damping term	0.01 ms	0 to 1000
	47	Function expansion settings 2	-32768 to 32767	-32768 to 32767
	48	Adjustment filter	0 to 2000	0 to 2000
	49	Command response filter / adjustment filter damping term setting	0 to 99	0 to 99
	50	Viscous friction compensation gain	0.1%/ (10000 r/min)	0 to 10000
	51	Immediate stop completion wait time	ms	0 to 10000
	57	Torque saturation error protection detection time	ms	0 to 5000
	60	2nd damping depth	-	0 to 1000
	61	1st resonance frequency	0.1 Hz	0 to 3000

Class	No.	Parameter name	Unit	Setting range
	62	1st resonance damping ratio	-	0 to 1000
	63	1st anti-resonance frequency	0.1 Hz	0 to 3000
	64	1st anti-resonance damping ratio	-	0 to 1000
	65	1st response frequency	0.1 Hz	0 to 3000
	66	2nd resonance frequency	0.1 Hz	0 to 3000
	67	2nd resonance damping ratio	-	0 to 1000
	68	2nd anti-resonance frequency	0.1 Hz	0 to 3000
	69	2nd anti-resonance damping ratio	-	0 to 1000
	70	2nd response frequency	0.1 Hz	0 to 3000
	71	3rd damping depth	-	0 to 1000
	72	4th damping depth	-	0 to 1000
	73	Load estimation filter	0.01 ms	0 to 2500
	74	Torque compensation frequency 1	0.1 Hz	0 to 5000
	75	Torque compensation frequency 2	0.1 Hz	0 to 5000
	76	Load estimation count	-	0 to 8
	85	Retracting operation condition setting	-	-32768 to 32767
	86	Retracting operation alarm setting	-	-32768 to 32767
	88	Absolute multi-rotation data upper limit	-	0 to 65534
	97	Function expansion setting 3	-	-2147483648 to 2147483647
	98	Function expansion setting 4	-	-2147483648 to 2147483647

13.3.8 Class 7: Special Setting 2

Class	No.	Parameter name	Unit	Setting range
7	00	Display on LED	-	0 to 32767
	01	Display time setup upon power-up	100 ms	-1 to 1000
	03	Output setup during torque limit	-	0 to 1
	09	Correction time of latch delay 1	25 ns	-2000 to 2000
	10	Soft limit function	-	0 to 3
	11	Positive side software limit value	Command unit	-1073741823 to 1073741823
	12	Negative side software limit value	Command unit	-1073741823 to 1073741823
	13	Absolute home position offset	Command unit	-1073741823 to 1073741823

Class	No.	Parameter name	Unit	Setting range
	14	Main power OFF warning detection time	ms	0 to 2000
	15	Positioning adjacent range	Command unit	0 to 1073741823
	16	Torque saturation error protection frequency	No. of times	0 to 30000
	20	RTEX communication cycle setup	-	-1 to 12
	21	RTEX command updating cycle ratio setting	-	1 to 2
	22	RTEX function extended setup 1	-	-32768 to 32767
	23	RTEX function extended setup 2	-	-32768 to 32767
	24	RTEX function extended setup 3	-	-32768 to 32767
	25	RTEX speed unit setup	-	0 to 1
	26	RTEX continuous error warning setup	No. of times	0 to 32767
	27	RTEX accumulated error warning setup	No. of times	0 to 32767
	28	RTEX_Update_Counter error warning setup	No. of times	0 to 32767
	29	RTEX monitor select 1	-	0 to 32767
	30	RTEX monitor select 2	-	0 to 32767
	31	RTEX monitor select 3	-	0 to 32767
	32	RTEX monitor select 4	-	0 to 32767
	33	RTEX monitor select 5	-	0 to 32767
	34	RTEX monitor select 6	-	0 to 32767
	35	RTEX command setting 1	-	0 to 2
	36	RTEX command setting 2	-	0 to 2
	37	RTEX command setting 3	-	0 to 2
	38	RTEX_Update_Counter error protection setup	No. of times	0 to 32767
	41	RTEX function extended setup 5	-	-32768 to 32767
	78	Signal reading setting for latch trigger with stop function	-	0 to 3
	91	RTEX communication cycle extended setup	ns	0 to 2000000
	92	Correction time of latch delay 2	25 ns	-2000 to 2000
	93	Home return limit speed	r/min	0 to 20000
	95	Number of RTEX continuous communication error protection 1 detections	No. of times	0 to 17
	96	Number of RTEX continuous communication error protection 2 detections	No. of times	0 to 17

Class	No.	Parameter name	Unit	Setting range
	97	Number of RTEX communication timeout error protection detections	No. of times	0 to 17
	98	Number of RTEX cyclic data error protection 1 / 2 detections	No. of times	0 to 17
	99	RTEX function extended setup 6	-	-32768 to 32767
	108	RTEX communication synchronization setup	-	0 to 7
	110	RTEX function extended setup 7	-	-2147483648 to 2147483647
	111	Trigger signal allocation setting of latch mode with stop function	-	0 to 64
	112	Selection of RTEX communication status flag	-	0 to 1

13.3.9 Class 8: Special Setting 3

Class	No.	Parameter name	Unit	Setting range
8	01	Profile linear acceleration constant	10000 Command unit/s^2	1 to 429496
	04	Profile linear deceleration constant	10000 Command unit/s^2	1 to 429496
	10	Amount of travel after profile position latch detection	Command unit	-1073741823 to 1073741823
	12	Profile home return position mode setup	-	0 to 1
	13	Profile home return velocity 1	Command unit/s or r/min	0 to 2147483647
	14	Profile home return velocity 2	Command unit/s or r/min	0 to 2147483647
	17	Relative movement of retracting operation	Command unit	-2147483648 to 2147483647
	18	Retracting operation speed	Command unit/s or r/min	0 to 2147483647

13.4 Monitor Commands

These commands are specified with RTEX_ReadAmpData (amplifier monitor).

<table>
<tr><th colspan="2">Type_Code
(Note 1) (Note 3)</th><th colspan="2" rowspan="2">Name</th><th rowspan="2">Index
(Note 2)</th><th rowspan="2">Unit</th><th rowspan="2">Description</th></tr>
<tr><th>A4N compatible</th><th>Standard</th></tr>
<tr><td>101h</td><td>01h</td><td>Position deviation</td><td>PERR</td><td>0
(1,2)</td><td>Command unit</td><td><In position control mode>
Position deviation
<In full-closed control mode>
External scale deviation
* The computation method (reference) of position deviation and external scale deviation is set in bit 14 of Pr 7.23 “Command position deviation output switching”.
<table><tr><th>Pr7.23 bit14</th><th>Computation method of positional deviation</th></tr><tr><td>0</td><td>Deviation from the command after filtering</td></tr><tr><td>1</td><td>Deviation from the command before filtering</td></tr></table><In speed / torque control mode>
Undefined
Note: Although the same data is returned whether Index is 1 or 2, use Index = 0.</td></tr>
<tr><td>102h</td><td>02h</td><td>Encoder resolution</td><td>-</td><td>0</td><td>pulse/r</td><td>Encoder resolution of the motor connected</td></tr>
<tr><td>104h</td><td>04h</td><td>Command position
(after filtering)</td><td>MPOS</td><td>0</td><td>Command unit</td><td>Command position (after filtering)</td></tr>
<tr><td>105h</td><td>05h</td><td>Actual speed</td><td>ASPD</td><td>0</td><td>Set the unit through Pr 7.25.</td><td>Motor actual speed
* Set the unit through Pr 7.25 “RTEX speed unit setup”.
<table><tr><th>Pr7.25</th><th>Unit</th></tr><tr><td>0</td><td>[r/min]</td></tr><tr><td>1</td><td>[Command unit/s]</td></tr></table></td></tr>
<tr><td>106h</td><td>06h</td><td>Internal command torque</td><td>TRQ</td><td>0</td><td>0.1%</td><td>Command torque to motor</td></tr>
<tr><td>-</td><td>07h</td><td>Actual position</td><td>APOS</td><td>0</td><td>Command unit</td><td>Motor actual position
* Position of the external scale in full-closed mode</td></tr>
<tr><td>-</td><td>08h</td><td>Internal command position (before filtering)</td><td>IPOS</td><td>0</td><td>Command unit</td><td>Internal command position before filtering</td></tr>
</table>

Type_Code (Note 1) (Note 3)		Name		Index (Note 2)	Unit	Description
A4N compatible	Standard					
-	09h	Latch position 1	LPOS1	0	Command unit	Motor actual position latched in CH1
-	0Ah	Latch position 2	LPOS2	0	Command unit	Motor actual position latched in CH2
-	0Ch	Command velocity (after filtering)	MSPD	0	Set the unit through Pr 7.25.	Command velocity after filtering * Set the unit through Pr 7.25 “RTEX speed unit setup”. Pr7.25 / Unit: 0 / [r/min]; 1 / [Command unit/s] * The value is undefined in torque control mode.
-	0Dh	External scale position (Note 4)	EXPOS	0	Pulse (External scale)	External scale position

(Note 1) When a Type_Code error occurs, command error (0031h) will be returned.

Manufacturer will use a Type_Code not listed above.

When a Type_Code used by the manufacturer is set, undefined value will be returned in place of command error (0031h).

(Note 2) When an Index error occurs, command error (0032h) will be returned.

(Note 3) A4N compatible: Type_Code compatible with A4N series can be used, but only with main commands.

Standard: Type_Code newly created for A5N and A5N series and can be used with both main commands and subcommands. When using with main commands, set leftmost 4 bits to 0.

* Although the product supports the A4N compatible Typer_Code to maintain compatibility, basically use the standard Type_Code.

(Note 4) The version before the function extended version 1 is not supported.

Type_Code		Name		Index	Unit	Description
A4N compatible	Standard					
111h	11h	Regenerative load ratio	-	0	% (Note 2)	Ratio of the regenerative overload protection to the alarm occurrence level
112h	12h	Overload ratio	-	0	0.1%	Ratio of the actual load to the rated motor load
-	21h	Logical input signal	-	0	-	Logic level of input signal
-	22h	Logical output signal	-	0	-	Logic level of output signal
-	23h	Logical input signal	-	0	-	Logic level of input signal (expansion portion)

Type_Code		Name		Index	Unit	Description
A4N compatible	Standard					
		(expansion portion)				
-	24h	Logical output signal (expansion portion)	-	0	-	Logic level of output signal (expansion portion)
-	25h	Physical input signal	-	0	-	Physical level of input signal
-	26h	Physical output signal	-	0	-	Physical level of output signal
131h	31h	Inertia ratio	-	0	%	The ratio of load inertia to the motor's rotor inertia (equivalent of value in Pr 0.04) Inertia ratio = (load inertia / rotor inertia) × 100
132h	32h	Automatic motor recognition	-	0	-	0: Invalid 1: Valid
133h	33h	Cause of no revolution	-	0	-	The number which shows the cause that the motor is not running.
134h	34h	Warning flags	-	0	-	The flag that shows the state of the waring currently occurring. * The corresponding bit is set to 1 to activate the flag (showing warning status).
-	37h	Multiple alarm occurrences /Warning information(Note 1)	-	Refer to Section 6-9-6.	-	Information of all the alarms or warnings currently occurring
201h	41h	Mechanical angle (Single turn data)	-	0	pulse	The mechanical angle (one revolution data of an absolute encoder) of the motor * The polarity is fixed and data increases at CCW rotation. One revolution data = 0 to (Encoder resolution - 1) .
202h	42h	Electrical angle	-	0	0.7031°	Motor electrical angle * The polarity is fixed and data increases at CCW rotation. Electrical angle = 0 to 1FF [Hex]
-	43h	Multi-turn data	-	0	Turn	Multi-turn data of the absolute encoder * In the incremental mode (Pr 0.15 = 1), multi-turn data becomes an indefinite value.
-	44h	Encoder status(Note 1)	-	0	-	The status of the encoder
-	47h	Encoder pulse	-	0	pulse	The sum of encoder feedback pulses

Type_Code		Name		Index	Unit	Description
A4N compatible	Standard					
		sum(Note 1)				
-	48h	External scale pulse sum(Note 1)	-	0	Pulse (External scale)	The sum of external scale feedback pulses
-	49h	External scale absolute position(Note 1)	-	0	Pulse (External scale)	The absolute position of the external scale
-	61h	Power on cumulative time	-	-	30 min	Cumulative on-time of control power to the servo amplifier * Because the power ON time is recored in unit of 30 minutes, a turn-on period shorter than 30 minutes is not recorded in the cumulative on-time. not recorded in the cumulative on-time.

(Note 1) The version before the function extended version 1 is not supported.

(Note 2) Be careful that the unit is different from that used for A4N and A5N. (A4N, A5N: [0.1%], A6N: [%])

* With the function extended version 3 or higher, the unit can be changed through bit 7 of Pr 7.99.

Pr7.99 bit7 0: [%], 1: [0.1%]

Type_Code		Name		Index	Unit	Description
A4N compatible	Standard					
-	62h	Servo amplifier temperature	-	-	°C	Temperature inside the servo amplifier
-	63h	Encoder temperature	-	-	°C	Temperature inside the encoder * Applicable only to 23-bit encoder. 0 for unsupported encoder.
-	64h	Number of inrush resistance relay operations	-	-	Cycle	Operating cycles of inrush current suppression resistor relay * Saturation will occur at maximum value of 40000000h. * Because the power ON time is recored in unit of 30 minutes, a turn-on period shorter than 30 minutes is not recorded in the cumulative cycles.
-	65h	No. of dynamic brake operations	-	-	Cycle	Number of operations of dynamic brake relay * Saturation will occur at maximum value of 40000000h. * Because the power ON time is recored in unit of 30 minutes, a turn-on period shorter than 30 minutes is not recorded in the cumulative time.
-	66h	Fan operating time	-	-	30 min	Operating time of cooling fan * Because the power ON time is recored in unit of 30 minutes, a turn-on period

Type_Code		Name		Index	Unit	Description
A4N compatible	Standard					
						shorter than 30 minutes is not recorded in the cumulative time. * 0 when no fan is installed.
-	67h	Fan life expectancy	-	-	0.1%	Percent of fan life expectancy * Because the power ON time is recored in unit of 30 minutes, a turn-on period shorter than 30 minutes is not recorded in the cumulative time. * 0 when no fan is installed.
-	68h	Capacitor life expectancy	-	-	0.1%	Percent of life expectancy of main power source capacitor * Because the power ON time is recored in unit of 30 minutes, a turn-on period shorter than 30 minutes is not recorded in the cumulative time.
-	69h	Voltage across PN	-	-	V	Main power source PN voltage
-	6Ch	Consumed power of motor (Note 1)	-	-	W	Momentary power consumption of the motor
-	6Dh	Motor power consumption (Note 1)	-	-	Wh	Power consumption of the motor
-	6Eh	Cumulative motor power consumption (Note 1)	-	-	Wh	Cumulative value of motor power consumption
401h	71h	RTEX Cumulative communication errors	-	0	Cycle	Cumulative number of RTEX communication errors * Saturation will occur at maximum value of FFFFh. The count will be cleared upon restarting of servo amplifier or resetting of control power source.
-	77h	RTEX UpdateCounter cumulative error count(Note 1)	-	0	Cycle	Cumulative number of communication errors of RTEX UpdateCounter * Saturation will occur at maximum value of 7FFFh. The count will be cleared upon restarting of servo amplifier or resetting of control power source.
-	78h	RTEX communication Cumulative RTEX communication timeout errors(Note 1)	-	0	Cycle	Cumulative number of RTEX communication data reception interruption errors * Saturation will occur at maximum value of FFFFh. The count will be cleared upon restarting of servo amplifier or resetting of control power source.

Type_Code		Name		Index	Unit	Description
A4N compatible	Standard					
411h	81h	Encoder cumulative communication errors	-	0	Cycle	Cumulative number of communication errors between encoders * Saturation will occur at maximum value of FFFFh. The count will be cleared upon restarting of servo amplifier or resetting of control power source.

(Note 1) The version before the function extended version 1 is not supported.

Type_Code		Name		Index	Unit	Description
A4N compatible	Standard					
413h	83h	External scale cumulative communication errors(Note 1)	-	0	Cycle	Cumulative number of communication errors between external scales * Saturation will occur at maximum value of FFFFh. The count will be cleared upon restarting of servo amplifier or resetting of control power source.
-	84h	External scale abnormal communication data errors (Note 1)	-	0	Cycle	Cumulative number of communication data errors in communication between external scales * Saturation will occur at maximum value of FFFFh. The count will be cleared upon restarting of servo amplifier or resetting of control power source.
-	85h	For manufacturer's use	-	-	-	-
-	86h	Hybrid position deviation (Note 1)	-	-	Command unit	Tolerance between encoder position and external scale position
-	87h	External scale data(Note 1) (Leftmost 24 bits)	-	0	Pulse (External scale)	Rightmost 24 bits of external scale data
-	88h	External scale data(Note 1) (Rightmost 24 bits)	-	0	Pulse (External scale)	<Virtual full-close control mode function disabled> Leftmost 24 bits of external scale data is output. <Virtual full-close control mode function enabled> ● When an AB-phase output type scale is connected, position data (16 bits) is output that is set to 0 when the power is turned ON. Note that it is not affected by Pr 3.26 Reversal of direction.

Type_Code		Name		Index	Unit	Description
A4N compatible	Standard					
						● When a serial incremental scale is connected, position data (24 bits) of the serial incremental scale is output. Note that the data output is position data affected by Pr 3.26 Reversal of direction.
-	89h	External scale status(Note 1)	-	0	-	Status of external scale
-	A1h	Velocity control command (Note 1)	-	0	Command unit/s	Velocity control command
-	A5h	Internal position command speed(Note 1)	-	0	Command unit/s	Internal position command speed
-	A6h	Speed deviation(Note 3)	-	0	Command unit/s	Speed deviation
-	A8h	Positive direction torque limit value(Note 1)	-	0	0.05%	Positive direction torque limit value
-	A9h	Negative direction torque limit (Note 1)	-	0	0.05%	Negative direction torque limit value
-	AAh	Speed limit value (Note 1)	-	0	Command unit/s	Speed limit value
-	ABh	Gain switching flag (Note 1)	-	0	-	Gain switching flag
-	B1h	Deterioration diagnosis state (Note 1)	-	0	-	Deterioration diagnosis state
-	B2h	Deterioration diagnosis torque average time (Note 1)	-	0	0.1% (Note 2)	Deterioration diagnosis torque command average time
-	B3h	Deterioration diagnosis torque command standard value (Note 3)	-	0	0.1%	Deterioration diagnosis torque command standard value
-	B4h	Deterioration diagnosis inertia ratio estimate(Note 1)	-	0	%	Deterioration diagnosis inertia ratio estimate

(Note 1) The version before the function extended version 1 is not supported.

(Note 2) Be careful that the unit is different from the one of the data displayed on the setup support software (PANATERM).

(Note 3) The version before the function extended version 2 is not supported.

Type_Code		Name		Index	Unit	Description
A4N compatible	Standard					
-	B5h	Deterioration diagnosis unbalanced load estimate (Note 1)	-	0	0.1% *2)	Deterioration diagnosis unbalanced load estimate
-	B6h	Deterioration diagnosis unbalanced load estimate (Note 1)	-	0	0.1% *2)	Deterioration diagnosis unbalanced load estimate
-	B7h	Deterioration diagnosis unbalanced load estimate (Note 1)	-	0	0.1%/(10000 r/min) *2)	Deterioration diagnosis unbalanced load estimate
-	FAh	Monitor flag (Note 1)	-	0	-	Various flag information of the servo amplifier The contents of Monitor_Data, the response data, are as follows.

Byte	bit	Description
12, 20	7 to 0	For manufacturer's use
13, 21	7 to 0	For manufacturer's use
14, 22	7 to 6	For manufacturer's use
	5	Semi-closed / full-closed selection state 0: Semi-closed 1: Full-closed
	4	Incremental / absolute selection state 0: Incremental mode 1: Absolute mode
	3 to 0	For manufacturer's use
15, 23	7 to 0	For manufacturer's use

(Note 1) The version before the function extended version 2 is not supported.

(MEMO)

Revision History

The manual code is shown at the bottom of the cover page.

Date of issue	Manual No.	Revision details
February 2021	WUME-GM1PGR-01	1st edition
August 2021	WUME-GM1PGR-02	2nd Edition Added instructions related to EtherCAT. Changed PMC_ReadLatchPosition parameters. Added instructions related to PID control. Added instructions related to the GM1 Pulse Output Unit.
March 2022	WUME-GM1PGR-03	3rd Edition Clerical corrections Added function block argument for pulse output unit ● Added switching of P-point / E-point control for positioning control ● Added specification of creep speed to home return
April 2022	WUME-GM1PGR-04	4th Edition ● Changed the company name.
June 2022	WUME-GM1PGR-05	5th Edition ● Added new instructions related to CNC control. ● Added instructions related to motor setting changes. ● Added instructions related to recipe manager functions. ● Added description of ladder instruction execution box. ● Clerical corrections
August 2023	WUME-GM1PGR-06	6th Edition ● Added a description of instructions on single axis control. ● Added new direct commands and buffer mode to single axis control. ● Added new instructions related to cam synchronous control. ● Added new instructions related to CNC control. ● Added new function for clearing system error.
November 2023	WUME-GM1PGR-07	7th Edition ● Made changes associated with provision for RTEX maximum 32 axes. ● Added new instructions related to EtherCAT communication control. ● Added new instructions related to EtherCAT slave enable/disable setting. ● Added new instructions related to MQTT client. ● Added new instructions related to DNS client. ● Added new instructions related to SNTP client. ● Added new instructions related to CSV file operation. ● Added new instructions related to project management functions. ● Added descriptions on the axis structure.
April 2024	WUME-GM1PGR-08	8th Edition ● Updated description of instructions related to COM port (general-purpose communication).

Date of issue	Manual No.	Revision details
		● Updated description of instructions related to LAN port (MQTT, general-purpose communication). ● Updated description of instructions related to SD card operation (CSV file operation, file operation). ● Updated description of instructions related to function blocks for the pulse output unit. ● Updated description of instructions related to motion control function blocks (single axis control, synchronous control, CNC control). ● Clerical corrections

(MEMO)

Please contact .....

Industrial Device Business Division,
Panasonic Industry Co., Ltd.
7-1-1 Morofuku, Daito City, Osaka, 574-0044, Japan
industrial.panasonic.com/ac/e/

WUME-GM1PGR-08