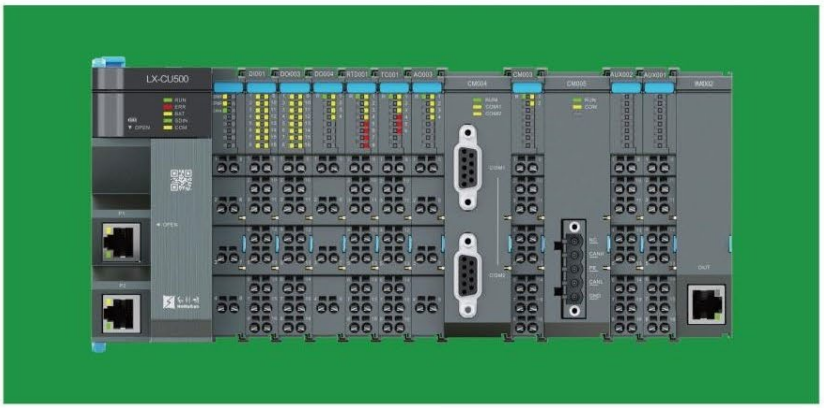


PLCopen Command Manual

LX Series Programmable Logic Controllers

Version: 2024.09

Copyright Notice

Address: Di Sheng Middle Road, No.2,
Economic-Technological Development Area, 100176, Beijing, China
Tel: +86 010-5898 1588
Consulting Hotline: 400-811-1999
Technology Request: +86 010-58981514
Fax: +86 010-5898 1558
Web: http://www.hollysys.com
Email: PLC@hollysys.com
Sina weibo: http://weibo.com/hollysysplc

Table of Contents

Chapter 1 Foreword

1.1 Purpose

This document will introduce the use of all hardware modules in the LX system, including product introduction, module components, technical parameters, indicators, wiring instructions, diagnostic information, etc. Please use it in conjunction with other supporting product materials to help you have a more comprehensive understanding of how to use the modules.

1.2 Object

This manual is intended for use by engineers or related professional technicians with expertise in automation and control systems.

1.3 Target Products

This manual is applicable to the following products:

- LX series PLC products

1.4 Statement of Use

- The contents of this manual have been tested according to regulations, and the diagrams are consistent with the software and hardware products described. However, errors are inevitable, and complete consistency cannot be guaranteed. If you have any suggestions for improving or correcting the contents of this manual, please let us know in time and we will make corrections in the next version.
- Due to product iteration and updates, the relevant parameters and information in this manual may change. If you have any questions, please consult the technical support staff.
- Since the devices described in this manual can be used in a variety of ways, the user and the person responsible for the use of the devices must ensure that each method is admissible. HollySys will not be legally responsible for any direct or indirect losses resulting from the use or misuse of these devices.
- Due to uncertainties in the actual application, HollySys assumes no responsibility for the direct use of the data provided in this manual.

1.5 Safety Precautions

1.5.1 Safety Statement

1. Please read and comply with these safety precautions before using this product.
2. To ensure personal and device safety, please strictly abide by the safety precautions on the product label and in the manual when using this product.
3. The words "Note", "Caution", "Warning" and "Danger" in this manual do not represent all the safety precautions that shall be observed, but only serve as a supplement to all precautions.
4. This product shall be used in an environment that meets the design specifications. Otherwise, it may cause faults. Device damage caused by failure to comply with relevant regulations is not covered by the product quality guarantee.
5. HollySys shall not be responsible for any personal safety accidents or property losses caused by any operation that does not comply with the provisions of this manual or does not meet the requirements.

1.5.2 Warning

For your safety and to avoid property losses, please pay attention to the warnings in this manual. The following warnings are indicated according to the hazard level from high to low. When there are multiple hazard levels, only the highest level is prompted. If the highest level is a warning that may cause personal injury, there may also be a warning that may cause property loss.

Warnings on personal safety: Indicated by a warning triangle  .

Warnings on property loss: Without any warning triangle.

 Danger
It indicates that death or serious personal injury may be caused if operations are not carried out as specified.

 Warning
It indicates that death or serious personal injury may be caused if operations are not carried out as specified.

 Caution
It indicates that minor personal injuries may be caused if operations are not carried out as specified.

Note
It indicates that property losses may be caused if operations are not carried out as specified.

Chapter 2 Document Guide

2.1 Related Manuals

The available manuals for this product are displayed in the table below. Please choose and refer to them based on your specific usage requirement.

Manual Name	Purpose	Content
Communication Manual for LX Series Programmable Logic Controllers	Guide users to build the LX system communication network	Describe the communication networks supported by the LX system, including the introduction to communication protocols and function usage
Programming Manual for LX Series Programmable Logic Controllers	Learn about how to use the FA-AutoThink programming software	Describe the interface, functions, and related operations of FA-AutoThink programming software, including: Software interface introduction Project creation Programming Online commissioning function
Basic Instruction Manual for LX Series Programmable Logic Controllers	Learn about the use of basic instructions of the LX system	Describe the use of basic instructions, including: Instruction function Parameter Example
Motion Instruction Manual for LX Series Programmable Logic Controllers	Learn about the use of motion control instructions of the LX system	Describe the use of motion control instructions, including: Instruction function Parameter Example
Module Manual for LX Series Programmable Logic Controllers	Learn about the hardware modules of the LX system	Describes all hardware modules, including: Module function Hardware interface Wiring Indicator Technical parameters

2.2 Usage Convention

The following symbols are used in this manual:

-  Tip: This symbol indicates helpful tips or suggestions for using the product more efficiently.

2.3 How to Get the Manual

If you need the electronic PDF file of this manual, please visit the official website of HollySys

(https://cn.hollysys.com/other/download.html) to download the file.

Chapter 3 Manual Revision History

Manual Version	Revision Date	Revision Contents
V1.0	2024.09	Create

Chapter 4 Overview commands

4.1 List of commands

Command Library	Commands	Name	Function Description
Single-axis motion command	SMC_AxisInit	Axis initialization configuration	Used to initialize the axis configuration. With this function block, you can set the basic configuration required for axis operation, such as axis type, user unit conversion, limit switch channel, etc.
	MC_Power	Axis enable	Control servo enabling status command
	MC_Home	Homing	Single axis origin return command
	MC_Stop	Axis stop	The control axis stops at the specified deceleration, any motion control command being executed on this axis is suspended, and all motion buffer commands on this axis are canceled
	MC_MoveJog	Jog	Single-axis inching command in positive/negative direction
	MC_MoveAbsolute	Absolute positioning	Single-axis absolute positioning command: control axis moves to specified absolute position
	MC_MoveRelative	Relative positioning	Single-axis relative positioning command: The control axis moves to the specified position incrementally.
	MC_MoveVelocity	Speed control	Single-axis speed control command, simulating speed control in position control mode
	SMC_SyncMoveVelocity	Cyclic Synchronous Speed Control	Speed control in cyclic synchronous speed mode
	MC_TorqueControl	Torque control	Single-axis torque control command, which controls the axis with a ramp in the torque mode
	SMC_SetTorqueLimit	Axial torque limit setting	Set the limit value of servo positive/negative torque
	MC_SetPosition	Set the current position	Redefines the current position of the specified axis absolutely or relative
	MC_SetOverride	Motion override setting (override)	This command sets the speed, acceleration/deceleration and jerk scale factor, which can increase or decrease the axis running speed, acceleration/deceleration and jerk according to the set ratio.
	SMC_SetDynamicLimits	Setting of motion parameter limit value	This command sets the limit values of axis motion parameters.
	MC_ReadDigitalInput	Read Axis Digital Input	Read Digital Input of Servo Equipment
	MC_ReadActualPosition	Read shaft actual position	This command reads the actual axis position.
	MC_ReadActualVelocity	Read actual axis speed	Read the actual velocity of the axis
	MC_ReadActualTorque	Read actual shaft	Read the actual torque of shaft

		torque	
	MC_ReadStatus	Read axis status	Reading the status of the axis PLCopen status machine
	MC_ReadAxisInfo	Read axis information	Used to read information about the specified axis
	MC_ReadAxisError	Error reading axis	Read axis error message (not a function block error)
	SMC_ReadAxisLimitStatus	Read axis limit status	Used to read the positive and negative hard limit status and positive and negative soft limit status of the specified axis
	MC_Reset	Reset axis error	Reset axis error
	MC_TouchProbe	Axis position latching	This command is used to lock the position of the specified axis and latch channel when a specified probe event trigger is detected.
	MC_AbortTrigger	Stop position latch	This command stops the position latch operation initiated by the MC_TouchProbe command.
Synchronous motion command	SMC_CamIn	Electronic cam	This command realizes the electronic cam function
	MC_GearIn	Electronic gear	This command realizes the function of electronic gear
	SMC_MoveLink	Chasing shear/flying shear	This command realizes the action of flying shearing process.
Axis group motion command	MC_GroupEnable	Axis group enabling	Axis group enabling
	MC_GroupDisable	Axis group disconnection enabling	Disabling the axis group function
	MC_GroupStop	Axis group stop	Control the axis group to decelerate and stop, suspend any motion control command being executed on the axis group, and cancel all motion buffer commands on the combined axis of this axis
	SMC_MoveLinearAbsolute2D	2-axis absolute position linear interpolation	This command realizes the linear interpolation function of 2-axis relative position
	SMC_MoveLinearRelative2D	Linear interpolation of relative position of 2 axes	This command realizes the linear interpolation function of 2-axis absolute position
	SMC_MoveCircularRelative2D	Circular interpolation for relative position of axis 2	This command realizes the circular interpolation function of 2-axis relative position
	MC_ GroupSetOverride	Axis group motion override setting (overriding)	This command is used to set the speed, acceleration/deceleration and jerk scale factor of the combined axis in the axis group. The speed, acceleration/deceleration and jerk of the combined axis operation can be increased or decreased according to the set ratio
	SMC_GroupSetDynamicLimits	Motion parameter limit setting of axis group	This command sets the limit values of axis group motion parameters.
	SMC_GroupCornerSpeedLimits	Axis group corner velocity limit setting	This command sets the corner speed limit parameters during the movement of the axis group to reduce the motion impact at non-smooth corner joints.
	SMC_GroupCircularSpeedLimits	Arc speed limit setting of axis group	This command sets the arc speed limit parameters during the movement of the axis group to reduce the motion impact at the arc turning.

4.2 Description of commands

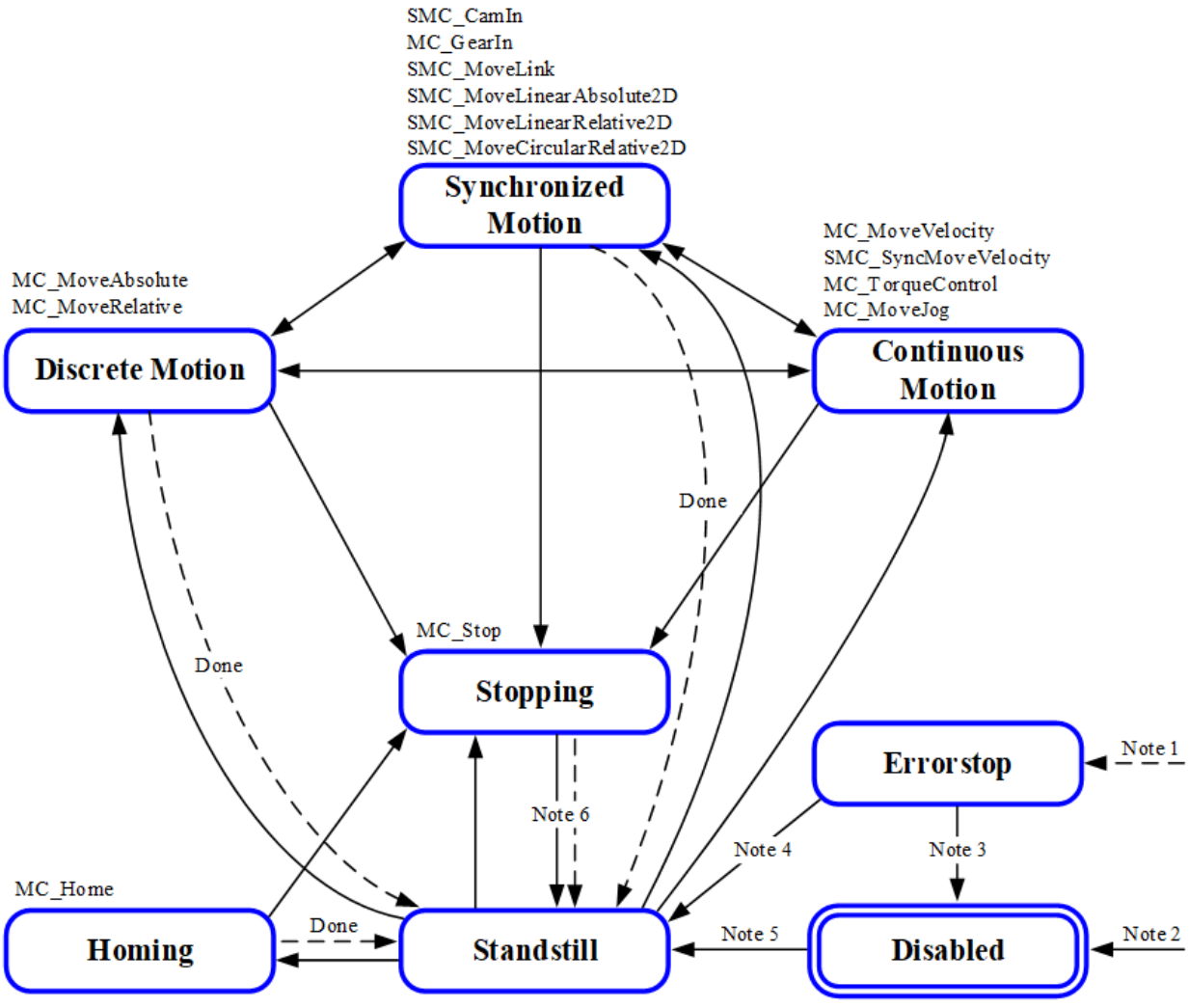
Before using the command, please understand the relevant information involved in the content of the command for better understanding and use.

Information Item	Description
Command/Function Block	Indicates the command name, for example: ADD
Name	It indicates the Chinese name of an command, such as: addition command.
FB/FUN	Indicates whether the command is a function block (FB) or a function (FUN). Function block type command: It can be called by program or FB. Functional command: It can be called by any one of program, FB and FUN.
Graphic Examples	Examples of command Application in LD Ladder Programming Language
Example of ST	Examples of command Application in ST Programming Language
Parameter	Describe the parameter information such as input, output and intermediate variables of commands.
Functions	Description of command functions
Examples	The application example of the command is illustrated by programming examples in LD and ST.
Precautions	Describe the precautions when using commands, including applications in special scenarios and occurrence of abnormal situations.
Reference	Other content information to be referenced when using this command

Chapter 5 Single Axis Motion Instructions

5.1 PLCopen Status Machine

The PLCopen single Axis status machine is as follows:



PLCopen single Axis status machine

The detailed Description of the state machine is as follows:

Axis status definition

Axis Status	Status Function Description
Disabled	Axis disable status (initial status)
ErrorStop	Fault shutdown status
Standstill	Enable status
Homing	Origin return status
Stopping	Stop status
Discrete Motion	Discrete motion
Continuous Motion	Continuous motion status
Synchronized Motion	Synchronous motion status

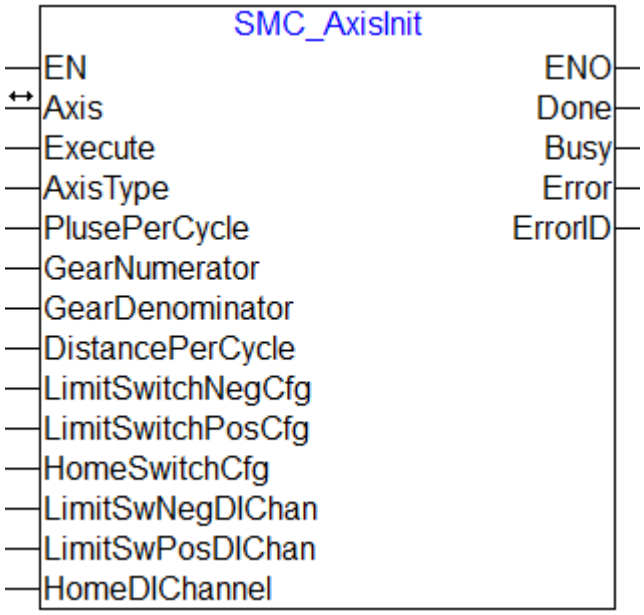
Axis Status Transition Conditions

Transformation	Axis Status Transition Conditions
Note 1	This state is entered immediately when the fault detection logic of the axis detects a fault.
Note 2	When the axis is fault free and MC_Power.Enable = FALSE
Note 3	When MC_Reset is called to reset the axis fault and MC_Power.Status = FALSE
Note 4	When MC_Reset is called to reset an axis fault AND MC_Power.Enable = TRUE and MC_Power.Status = TRUE
Note 5	When MC_Power.Enable = TRUE and MC_Power.Status = TRUE
Note 6	When MC_Stop.Done = TRUE and MC_Stop.Execute = FALSE

- The axis PLCopen status can be viewed through the axis parameter AxisState, and obtained through the command MC_ReadStatus.

5.2 SMC_AxisInit (Axis Configuration)

It is used to initialize the axis configuration. With this function block, you can set the basic configuration required for axis operation, such as axis type, user unit conversion, limit switch channel, etc.



SMC_AxisInit

5.2.1 Parameter

Parameter Type	Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	AxisType	Shaft type	No	-	-1:Unused 0:Virtual 50:EtherCAT_Position 51:EtherCAT_Speed 52:EtherCAT_Torque 53:EtherCAT_Encoder	EMAXISTYPE
	PlusePerCycle	Number of pulses per revolution of motor/encoder	No	-	Positive value	LREAL
	GearNumerator	Reduction ratio numerator (Revolutions of motor input shaft)	YES	1	Positive value	UDINT
	GearDenominator	Reduction ratio denominator (Number of retarder output shaft revolutions)	YES	1	Positive value	UDINT
	DistancePerCycle	When the reduction ratio is 1:1, it indicates the movement amount of the table for one rotation of the motor/encoder When the reduction ratio is not 1:1, it indicates the movement amount of the table for one revolution of the reducer output shaft	No	-	Positive value	LREAL
	LimitSwitchNegCfg	Negative Limit Switch Property Configuration	YES	-1	-1: Not used 0: Use drive body DI	INT

					(normally open) 1: Use the drive body DI (normally closed) 10: Use universal DI (normally open) 20: use universal DI (normally closed)	
	LimitSwitchPosCfg	Positive Limit Switch Property Configuration	YES	-1	-1: Not used 0: Use drive body DI (normally open) 1: Use the drive body DI (normally closed) 10: Use universal DI (normally open) 20: use universal DI (normally closed)	INT
	HomeSwitchCfg	Attribute configuration of origin switch	YES	0	-1: Not used 0: Use drive body DI (normally open) 10: Use universal DI (normally open)	INT
	LimitSwNegDIChan	Negative limit switch channel number When the driver body DI is used, the channel number is the bit of the driver input PIN in the Digital Input Object Dictionary (0x60FD), and 0x60FD must be included in the PDO; When using general DI, the channel number is the bit offset mapped by DI point in area I. For example, the bit offset of %IX100.1 in area I is 100×8+1.	YES	0	Non-negative value	UDINT
	LimitSwPosDIChan	Positive limit switch channel number The calculation method is the same as above	YES	1	Non-negative value	UDINT
	HomeDIChannel	Origin switch channel number The calculation method is the same as above	YES	2	Non-negative value	UDINT
OUTPUT	Done	Completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.2.2 Functions

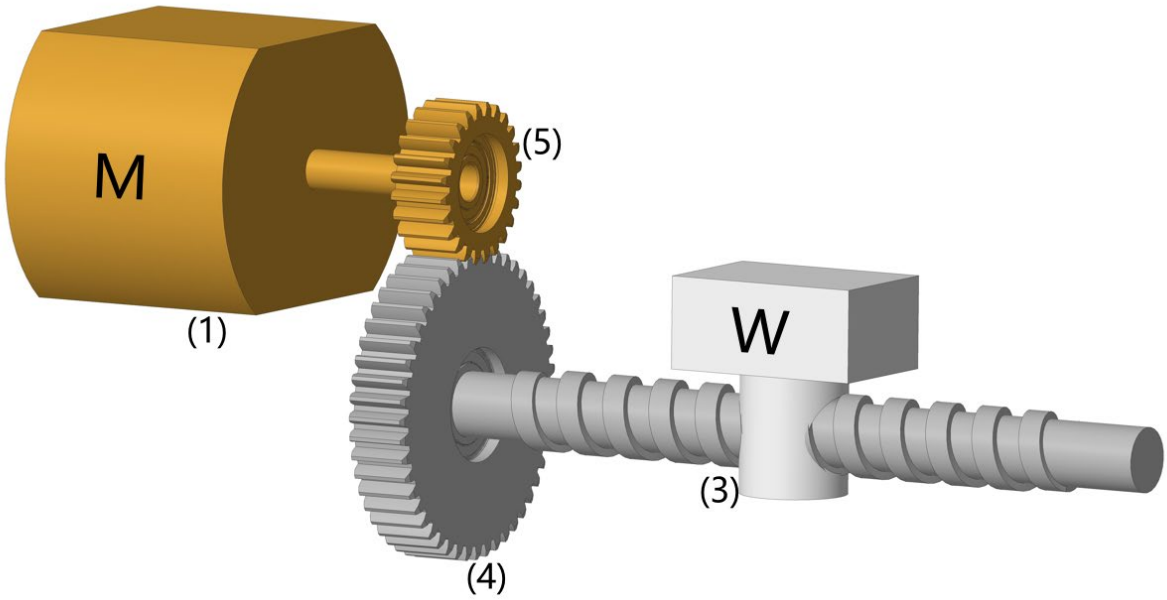
Set the basic parameters required for axis operation. When the rising edge of Execute triggers the functional block to initialize and configure axis-related parameters, when the output pin Done is TRUE, it indicates that the function block is successfully called to configure axis-related parameters. If there is an error in the configuration process, Error is TRUE, and ErrorID indicates the cause of the error. The classification of parameters is described as follows:

- Axis type setting

  The axis type of the specified axis can be set by setting the parameter AxisType to the axis type supported by Controller model. The axis types supported by LX series Controller are as follows:

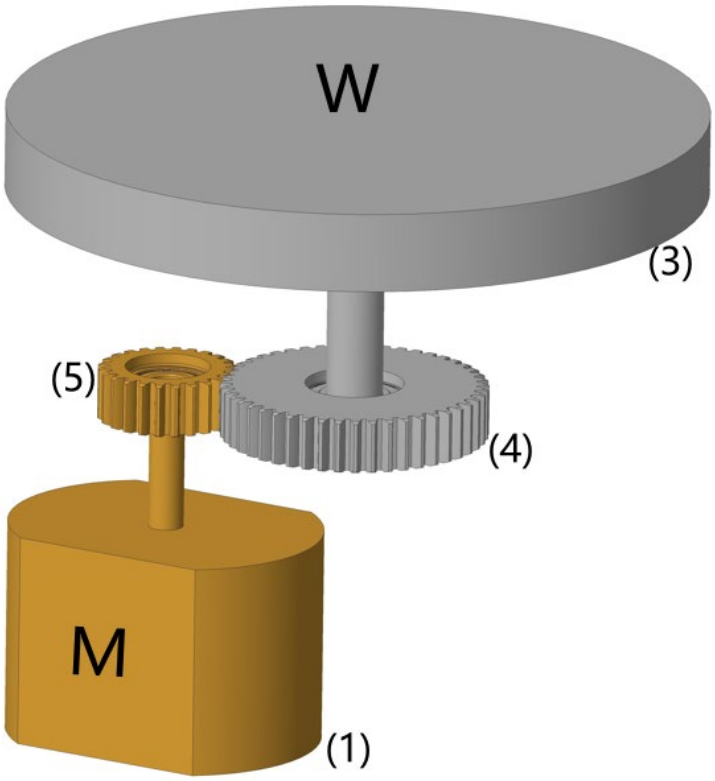
  - Unused (-1): The axis is not used.
  - Virtual axis (0): virtual axis
  - EtherCAT_Position(50): EtherCAT bus connection axis, position control
  - EtherCAT_Speed(51): EtherCAT bus connecting axis, speed control
  - EtherCAT_Torque(52): EtherCAT bus connecting shaft, torque control
  - EtherCAT_Encoder(53): EtherCAT bus connection axis, encoder count

- Attention shall be paid to different shaft types:
  - Axes of type EtherCAT_Encoder do not accept motion control commands and only track encoder feedback values.
  - When there are commands running in the axis command buffer of the axis, the set axis type does not take effect.
  - After the axis type is successfully set, the system will automatically modify the axis parameter RepMode according to different axis types. When the user sets the axis type as virtual axis, the system will set RepMode=2; when the user sets the axis type as other axis types, the system will set RepMode=0.
- Unit Conversion Factor Setting
  - Set the number of pulses (1) for the motor/encoder to rotate one circle through input parameter PlusePerCycle, unit: pulse/rev;
  - Set the reduction ratio numerator through input parameter GearNumerator, i.e. the number of teeth in (4) in the figure below;
  - Set the reduction ratio denominator through input parameter GearDenominator, i.e. the number of teeth in (5) in the following figure;
  - When the transmission is not used, the input parameter DistancePerCycle represents the movement amount (2) of the table for one revolution of the motor and encoder, in user unit/rev; when the transmission is used, the input parameter DistancePerCycle can be used to set the movement amount (3) of the table for one revolution of the reducer output shaft, in user unit/rev.
  - The user unit can be defined according to the user's needs, such as pulse number, millimeter and degree.
  - When the output is linear displacement, the schematic diagram is as follows:



M: MOTOR/ENCODER, W:TABLE

- When the output is rotational displacement, the schematic diagram is as follows:



M: MOTOR/ENCODER, W:TABLE

This command calculates the conversion relationship between the table movement amount and the number of motor command pulses according to the above setting parameters, and sets the axis unit conversion factor Units parameter. The finally calculated unit conversion factor can be viewed through the axis parameter Units. The conversion relationship is as follows:

$$\text{Units} = \frac{\text{PlusePerCycle} * \text{GearNumerator}}{\text{DistancePerCycle} * \text{GearDenominator}}$$

When no speed change device is used between the motor/encoder and the table, set the reduction ratio denominator GearDenominator equal to 1 and the reduction ratio numerator GearNumerator equal to 1. The conversion relationship is as follows:

$$\text{Pulse} = \frac{(1)\text{PlusePerCycle[LREAL]}}{(2)\text{DistancePerCycle[LREAL]}} * \text{Table Movement Amount (unit)}$$

When using a variable speed device between the motor/encoder and the table, modify GearNumerator and GearDenominator according to the actual situation. The conversion relationship is as follows:

$$\text{Pulse} = \frac{(1)\text{PlusePerCycle[LREAL]} * (4)\text{GearNumerator[UDINT]}}{(3)\text{DistancePerCycle[LREAL]} * (5)\text{GearDenominator[UDINT]}} * \text{Table Movement Amount (unit)}$$

- Positive and negative limit switch setting

  Set the positive and negative limit switch attribute through the parameter LimitSwitchPosCfg/LimitSwitchNegCfg. The Default Value is not used, indicating that the hard limit function corresponding to the axis is invalid.

  The parameter LimitSwNegDIChan/LimitSwPosDIChan is used to set the channel number of positive and negative limit, and the meaning of the channel number is related to the property setting of the positive and negative limit switch.

  When the attribute is set to "Use drive body DI", it indicates that the limit channel corresponding to the axis uses the DI channel of the drive body, and the corresponding channel number is the bit of Object Dictionary Digital inputs (0x60FD). The PDO mapping of the drive must contain the object dictionary 0x60FD, otherwise an error will be reported. For example: When using the drive body DI, set the channel number to 1, which means that bit 1 of 0x60FD is used as the limit switch channel.

  When the attribute is set to "Use universal DI", it means that the Controller general DI channel is used as the limit channel, and the corresponding channel number is the bit offset mapped by the DI point in area I. For example, when the Controller general DI is used, the mapping address of the DI used is %IX4.1, and its bit offset in area I is 4*8+1=33, i.e. the channel number shall be set to 33.

  If the positive and negative limit switch attribute is set to normally open, it indicates that it is valid when the input logic of the limit channel is 1, and if it is set to normally closed, it indicates that it is valid when the input logic of the limit channel is 0.

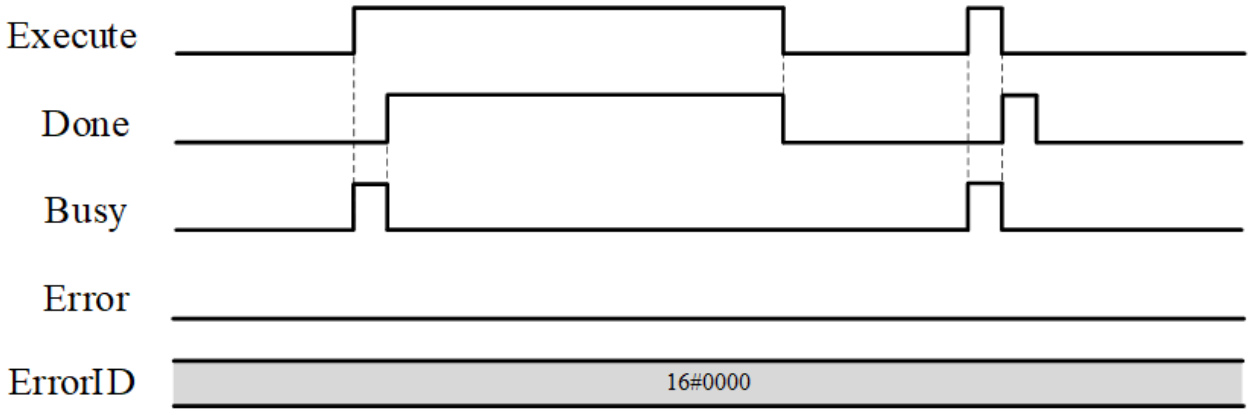
  If the attribute values of positive and negative limit are both set as drive body DI or general DI, the input value of channel number cannot be set as the same, otherwise the function block reports an error. In addition, if one of the property values for positive and negative limit is set to drive body DI, the other is set to general DI. If the channel number set to the general DI property corresponds to the mapping address range of the drive's Digtial Inputs in area I, and corresponds to the same bit as the channel number set to the drive body's DI property, a conflict will also occur between them and an error will be reported.

- Origin switch setting

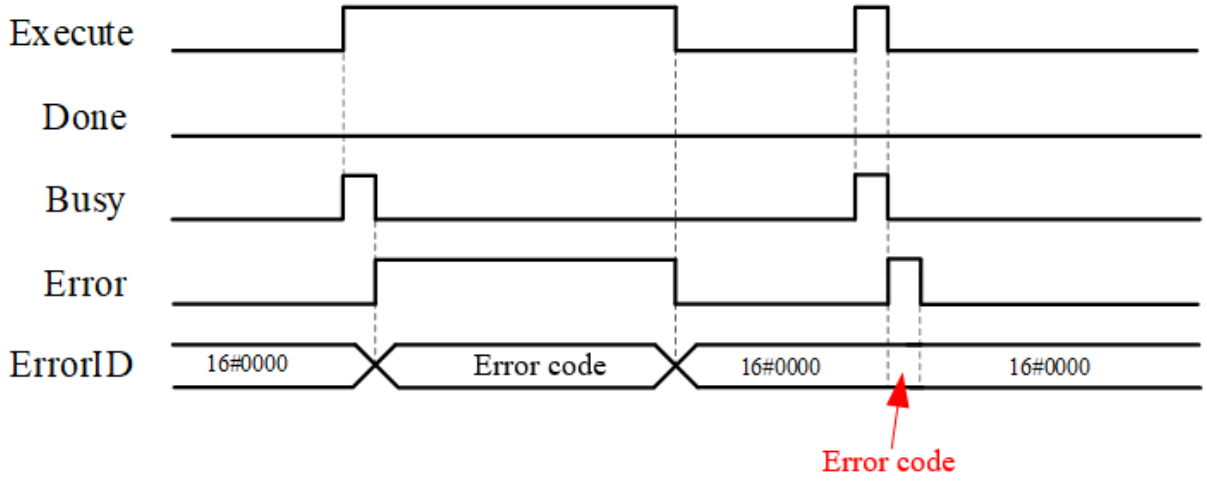
  HomeSwitchCfg and HomeDIChannel can be used to set the origin switch attribute and the origin switch channel value. The parameter meanings are the same as those of positive and negative limit switch settings, but the origin switch channel attribute cannot be set to normally closed, that is, the channel input logic cannot be reversed.

- Function block timing diagram

  (1) The function block executes normally without error. Execute the timing diagram that maintains high level. If the Execute has only one cycle of high level, the Done signal also remains for only one cycle.



(2) Error is reported during the execution of function block, and Execute keeps the timing chart of high level. If the Execute has a high level for only one cycle, the Error signal and the ErrorID are also maintained for only one cycle.



5.2.3 Examples

The example uses the SMC_AxisInit function block to initialize the axis type, user unit-related parameters, limit switch and origin switch logic for the specified axis as well as initialize the channel numbers of the limit switch and origin switch. The steps are as follows:

(1) Set the axis type to EtherCAT_Position;

(2) Set the unit conversion factor so that the motor makes one revolution (8388608 pulses) to drive the workbench to move 10000 user units.

(3) Set the positive and negative limit switch attributes and origin switch attribute to 10-use universal DI (normally open);

(4) Set the channel number of negative limit switch to 32, corresponding area I address of channel number to %IX4.0 (4*8=32), positive limit switch channel number to 33(4), corresponding area I

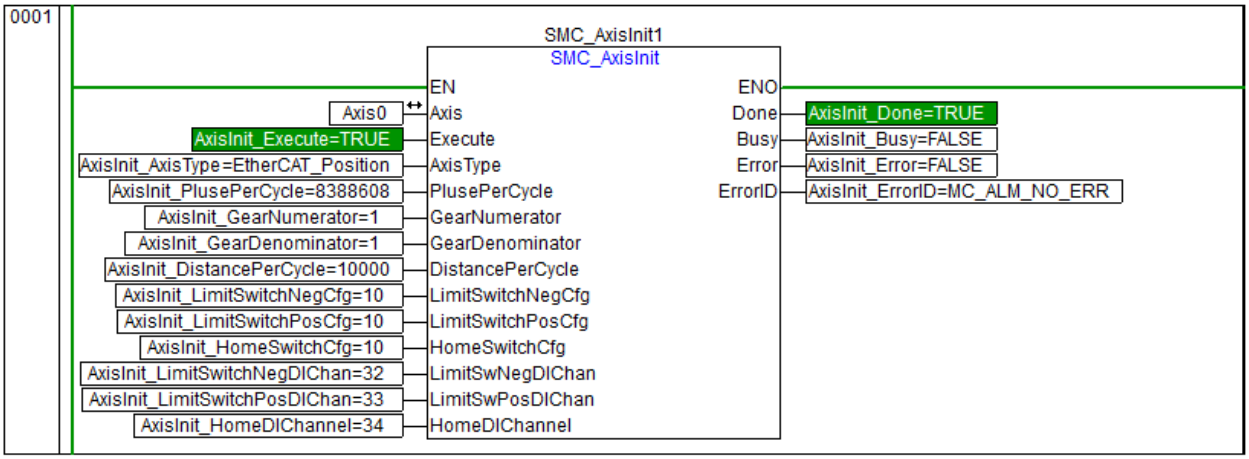
address of channel number to %IX4.1, original point switch channel number to 34, and corresponding area I address of channel number to %IX4.2.

After initialization by using the SMC_AxisInit function block, axis enable and other command control are performed.

■ Variable

Sr.No	Variable Name	Variable Type	Initial Value	Variable Description
1	SMC_AxisInit1	SMC_AxisInit	-	command Instance Definition
2	AxisInit_Execute	BOOL	FALSE	Rising edge enabling
3	AxisInit_AxisType	EMAXISTYPE	EtherCAT_Position	Shaft type
4	AxisInit_PlusePerCycle	LREAL	8388608	Number of pulses per servo cycle
5	AxisInit_GearNumerator	UDINT	1	Gear Ratio Numerator
6	AxisInit_GearDenominator	UDINT	1	Gear Ratio Denominator
7	AxisInit_DistancePerCycle	LREAL	10000	Distance per cycle
8	AxisInit_LimitSwitchNegCfg	INT	10	Negative Limit Switch Configuration
9	AxisInit_LimitSwitchPosCfg	INT	10	Positive Limit Switch Configuration
10	AxisInit_HomeSwitchCfg	INT	10	Origin switch configuration
11	AxisInit_LimitSwitchNegDIChan	UDINT	32	Negative limit switch DI channel
12	AxisInit_LimitSwitchPosDIChan	UDINT	33	Positive limit switch DI channel
13	AxisInit_HomeDIChannel	UDINT	34	Origin DI channel
14	AxisInit_Done	BOOL	FALSE	Initialization completion flag
15	AxisInit_Busy	BOOL	FALSE	Initialization busy flag
16	AxisInit_Error	BOOL	FALSE	Initialization error flag
17	AxisInit_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	Initialization error ID

■ LD Implementation Procedure



- After the initialization is completed, other command operations can be performed, such as MC_Power.

■ ST Implementation Procedure

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
```

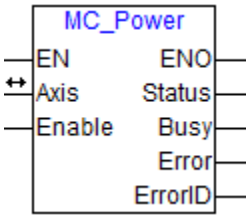
```
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

5.2.4 Precautions

- When setting parameters of this function block, the internal configuration sequence is to configure PlusePerCycle, GearNumerator, GearDenominator and DistancePerCycle first, then LimitSwitchNegCfg, LimitSwNegDIChan, LimitSwitchPosCfg and LimitSwPosDIChan. HomeSwitchCfg, HomeDIChannel, and finally configure AxisType. If the previous parameter setting is incorrect, the subsequent parameters will not be configured, and the successfully configured parameters will not be affected.
- This command can be reused for the same axis. When this command is reused for different tasks, errors may occur due to cross setting. Please avoid such use.
- Before using other commands, call the SMC_AxisInit command for axis initialization and determine that the SMC_AxisInit command Done pin is TRUE before performing other operations; otherwise, errors may be caused.

5.3 MC_Power (Enable Axis)

Control the servo enable status command.



5.3.1 Parameter

Parameter Type	Title	Description	Empty or not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Enable	Enable	No	FALSE	TRUE/FALSE	BOOL
OUTPUT	Status	Axis enable mark	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.3.2 Functions

This command enables the axis setting. The detailed description of this command is as follows:

(1) This command is valid at a high level. When Enable is set to TRUE, the specified axis can enter the enabling state, and Status (enable) becomes TRUE to realize motion control of the axis.

(2) When Enable is set to FALSE, the axis can be disabled. After disabling, the axis will not receive action commands and cannot realize axis control. In addition, the axis will also be abnormal when executing action commands on the axis after enabling.

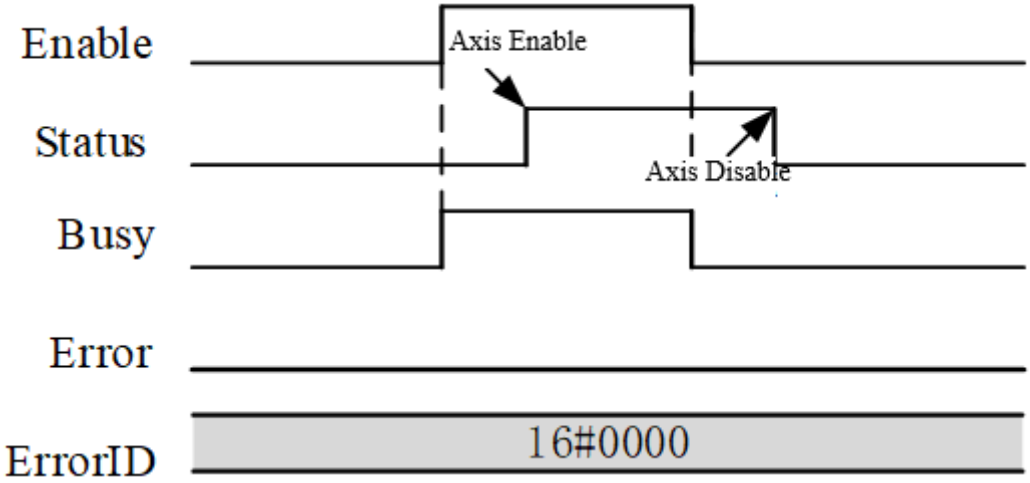
(3) When this command input is an "Enable type" command, the motion command will not be restarted. In principle, only one MC_Power (enable) module can be set for each axis.

(4) If Enable is set to FALSE, Busy will become FALSE, and Status will become FALSE when enabling is turned off. The status of the axis is shown as "enabling".

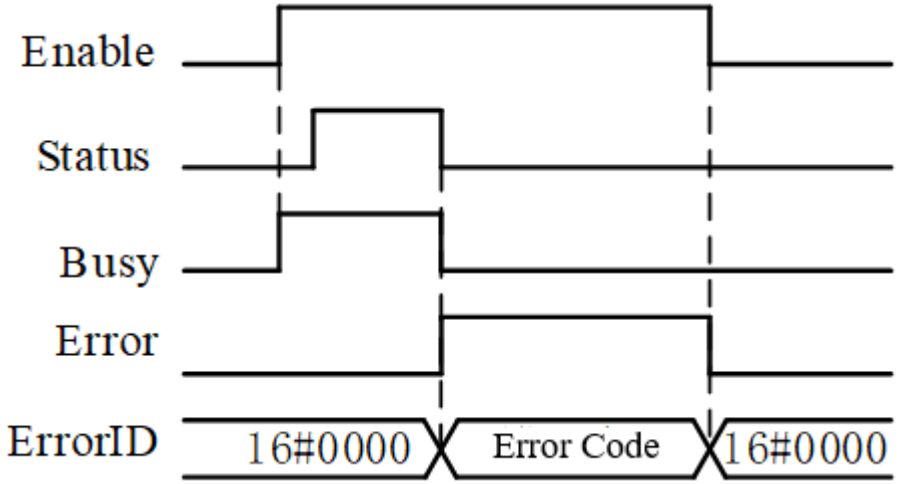
■ Function block timing diagram

When Enable is set to TRUE, the command Busy (in execution) becomes TRUE.

(1) The correct timing sequence for command execution is as follows:



(2) The Timing diagram in case of abnormality is as follows:



5.3.3 Examples

Create an MC_Power example program to set the axis enable.

Definition of Primary Variables

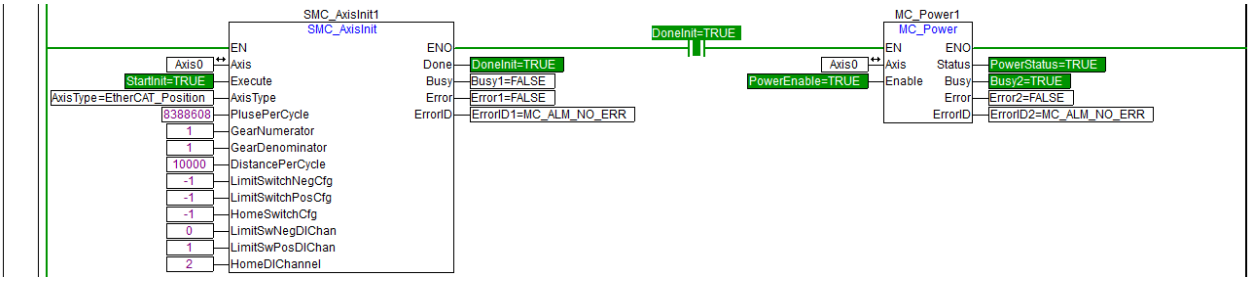
Title	Data Type	Initial Value	Description
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	Become TRUE at the start of initialization
DoneInit	BOOL	FALSE	Change to TRUE upon completion of initialization
PowerEnable	BOOL	FALSE	Change to TRUE at the beginning of enable
PowerStatus	BOOL	FALSE	Change to TRUE when enabled
StartReset	BOOL	FALSE	Change to TRUE at the start of reset

■ LD diagram program

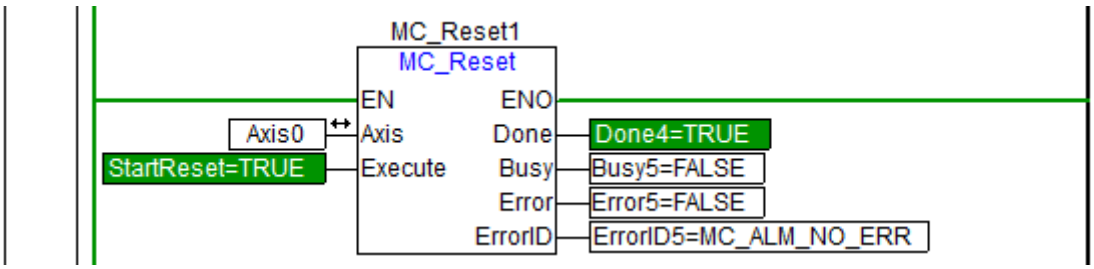
Use the ladder diagram to establish a program. The example program is as follows:

(1) First, initialize the axis by configuring its type and various parameters. After the initialization is

complete, enable the axis. When the servo is in the ON state, the axis can perform motion control.



(2) When an axis error occurs, the MC_Reset function block can be called for reset processing.



■ ST language program

(1) Initialize the axis.

```
(The * axis is initialized*)
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

(2) Initialize the axis.

```
(*Enable the axis*)
IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
```

```
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

(3) Reset processing can be carried out in case of error.

```
MC_Reset1(
Execute := StartReset,
Axis := Axis0,
Done=>Done4,
Busy=>Busy5,
Error=>Error5,
ErrorID=>ErrorID5);
```

5.3.4 Precautions

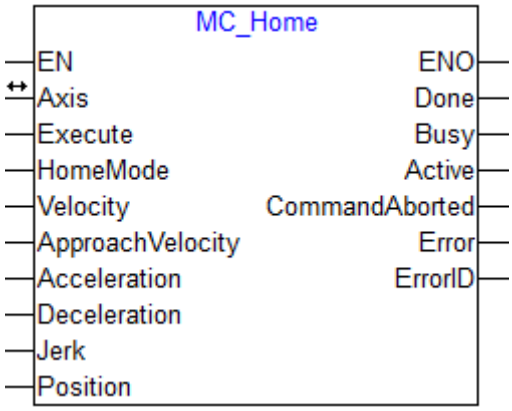
- Because the function block works regardless of whether its Enable is TRUE or FALSE, a trigger module should be added before calling Mc_Power.
- In some working environments such as vertical axis, disabling the enabling may cause sharp changes in the axis position. Therefore, be careful when calling this function block.

5.3.5 Refererence

- If an error occurs, refer to Function Block Error Codes.

5.4 MC_Home (Home)

Single axis homing commands.



5.4.1 Parameter

Parameter Type	Title	Description	Empty or not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	HomeMode	Origin homing mode	No	-	Support the following homing modes in CIA402 protocol: 1/2,3/5,7/11,17/18,19/21,23/27,33/34,35	SMC_HOME_MODE
	Velocity	High-speed search speed	No	-	Positive value	LREAL
	ApproachVelocity	Low-speed zero searching speed	No	-	Positive value	LREAL
	Acceleration	Acceleration	No	-	Positive value	LREAL
	Deceleration	Deceleration	No	-	Positive value	LREAL
	Jerk	0: Trapezoidal acceleration and deceleration Positive: S-shaped acceleration and deceleration acceleration	YES	0	0/positive	LREAL
	Position	After the homing is completed, the zero signal position is defined as Position.	YES	0	Positive/negative/0	LREAL
OUTPUT	Done	Origin reversion completed	YES	FALSE	TRUE/FALSE	BOOL

	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Active	In control, TRUE when the command is executed	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.4.2 Functions

This command is used for the homing function of a single axis. The origin of the specified axis is determined by detecting the origin switch signal or Z-phase signal of the servo motor. The homing mode is designated by the parameter HomeMode, which follows 15 regression modes in CIA402 protocol: 1/2, 3/5, 7/11, 17/18, 19/21, 23/27, 33/34 and 35.

Before using this command, call the SMC_AxisInit command to set necessary positive and negative limit switch and origin switch parameters. The origin switch signal can be set as servo driver body DI or Controller general DI. For specific configuration methods, please refer to the section of SMC_AxisInit.

This command can be used normally only after the specified axis is switched to the enabled state by using the MC_Power command and the supported axis type is set. This command supports the origin return function of the virtual axis, but does not support the origin return mode related to Z signal when it is set as a virtual axis.

The rising edge of the command is valid. When the rising edge of the command is executed, the input parameters such as locked axis name, origin return mode, speed and acceleration/deceleration are invalid when Execute=TRUE or Execute=FALSE is held. When the MC_Home command is run, the axis status is in Homing. The command input parameter HomeMode is used to set the homing mode, Velocity is used to set the speed of high-speed zero searching, and ApproachVelocity is used to set the speed of low-speed zero searching. The set value of ApproachVelocity should be less than that of Velocity. Acceleration and Deceleration are used to set acceleration and deceleration in the process of homing starting. Jerk is used to configure whether to enable the S-shaped acceleration and deceleration function, and Position is used to define the user's origin position after the homing is completed. After this function block is successfully executed, Busy = TRUE; after the Home command is sent and activated, Active = TRUE; after the normal execution of origin returning is completed, Done = TRUE. When this command is valid, the axis is in Homing state. The MC_Stop command that can make the axis in Stopping state can interrupt the MC_Home command. CommandAboarted=TRUE after the command is interrupted. After finding the origin by returning to the original point, you can check the completion status of homing through the IsHomed pin of the MC_ReadAxisInfo command.

When the same MC_Home command function block is enabled repeatedly, the new Home command will be bufferd, and the command function block will lose monitoring of the previous command and monitor the execution of the newly delivered command instead.

When this command is executed, the soft limit function of the Controller and the hard limit function related to the origin return mode are invalid, and the hard limit function not used in the specified origin return mode is valid.

■ Description of homing Mode

Description of the homing mode

Mode	Homing Direction	Signals Used for Homing	Final Position of Origin	Description
1	Negative	N-OT Z	Z	Negative homing, the deceleration point is the negative limit switch, and the origin is the motor Z signal. The falling edge of the negative limit must be encountered before encountering the Z signal
2	Positive	P-OT Z	Z	Positive homing, the deceleration point is the positive limit switch, and the origin is the motor Z signal. The falling edge of the forward limit must be encountered before encountering the Z signal
3	Positive	HMSW Z	Z	Positive homing, the deceleration point is the origin switch, and the origin is the motor Z signal. Before encountering the Z signal, you must first encounter the falling edge on the same side of the original switch
5	Negative	HMSW Z	Z	Negative homing, the deceleration point is the origin switch, and the origin is the motor Z signal. The falling edge on the same side of the original switch must be encountered before encountering the Z signal
7	Positive	HMSW P-OT Z	Z	Positive homing, the deceleration point is the origin switch and the origin is the motor Z signal. Before encountering the Z signal, the falling edge on the same side of the original switch must be encountered (missing the original switch, meeting the positive limit, high-speed negative direction)
11	Negative	HMSW N-OT Z	Z	Negative homing. The deceleration point is the origin switch, and the origin is the motor Z signal. Before encountering the Z signal, the falling edge on the same side of the original switch must be encountered (missing the original switch, meeting the negative limit, high-speed negative direction)
17	Negative	N-OT	N-OT↓	Negative homing, the deceleration point is the negative limit switch, and the home is the falling edge of the negative limit
18	Positive	P-OT	P-OT↓	Positive homing, the deceleration point is the positive limit switch, and the home is the falling edge of the forward limit.
19	Positive	HMSW	HMSW↓	Positive homing, the deceleration point is the origin switch, and the home is the falling edge of the origin switch
21	Negative	HMSW	HMSW↓	Negative homing, the deceleration point is the origin switch, and the home is the falling edge of the origin switch
23	Positive	HMSW P-OT	HMSW↓	Positive homing, the deceleration point is the origin switch, and the home is the falling edge of the origin switch (missing the origin switch, encountering the forward limit, high-speed negative direction)
27	Negative	HMSW N-OT	HMSW↓	Negative homing, the deceleration point is the origin switch, and the home is the falling edge of the origin switch (missing the origin switch, encountering the negative limit, high-speed negative direction)
33	Negative	Z	Z	Negative homing, with the origin being motor Z signal
34	Positive	Z	Z	Positive homing, the origin is motor Z signal
35	N/A	N/A	Current Location	Take the current position as the origin

● Note: Z indicates the Z pulse signal of servo motor, N-OT indicates the negative limit switch signal, P-OT indicates the positive limit switch signal, HMSW indicates the origin switch signal, and ↓ indicates the falling edge of the switch signal.

(1) Mode 1, looking for negative limit switch and Z signal

Origin: motor Z signal

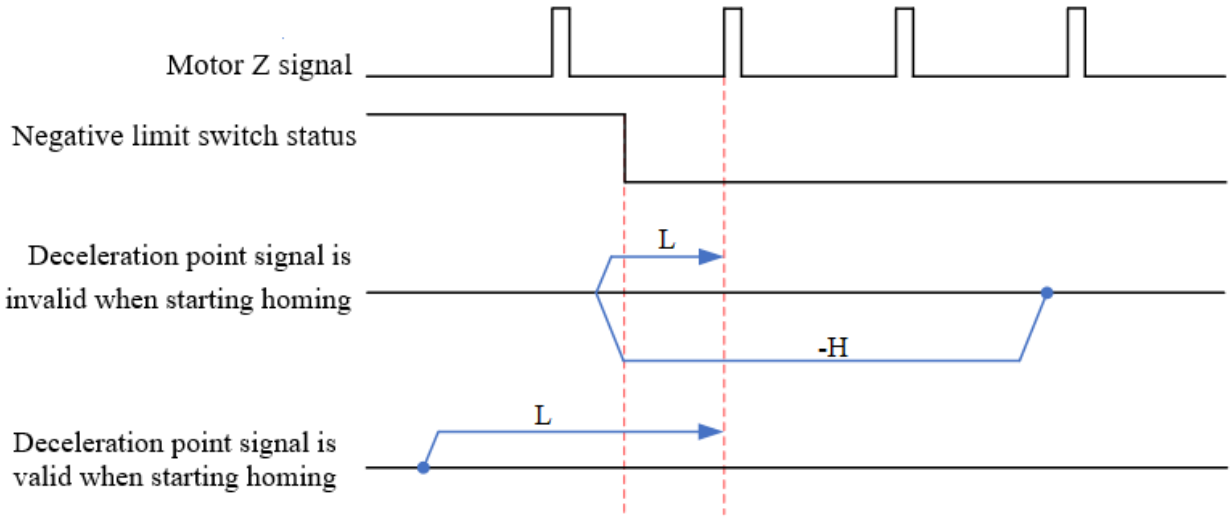
Deceleration point: negative limit switch

Motion track 1: If the negative limit switch status is invalid when starting homing, it moves towards the negative direction at high speed. After encountering the rising edge of the negative limit switch, it decelerates and stops, and then changes to low speed to move towards the positive direction. After encountering the falling edge of the negative limit switch when moving forward at low speed, continue to find the nearest Z-signal position as the origin at low speed.

Motion track 2: If the negative limit switch state is valid when home starting, it moves forward at low speed. After encountering the falling edge of the Negative Limit Switch in the Positive direction, continue to find the nearest Z-Signal position as Home at Low Speed in the Positive direction.

In this mode, the return-to-origin process will be aborted when the positive limit switch is in a valid status, i.e. the command will be interrupted.

The homing process for mode 1 is shown in the figure below:



- Note: "H" in the figure represents high-speed search velocity, "L" represents low-speed zero search velocity (ApproachVelocity), and "-" represents reverse motion, the same below.

(2) Mode 2: Looking for positive limit switch and Z signal.

Origin: motor Z signal

Deceleration point: positive limit switch

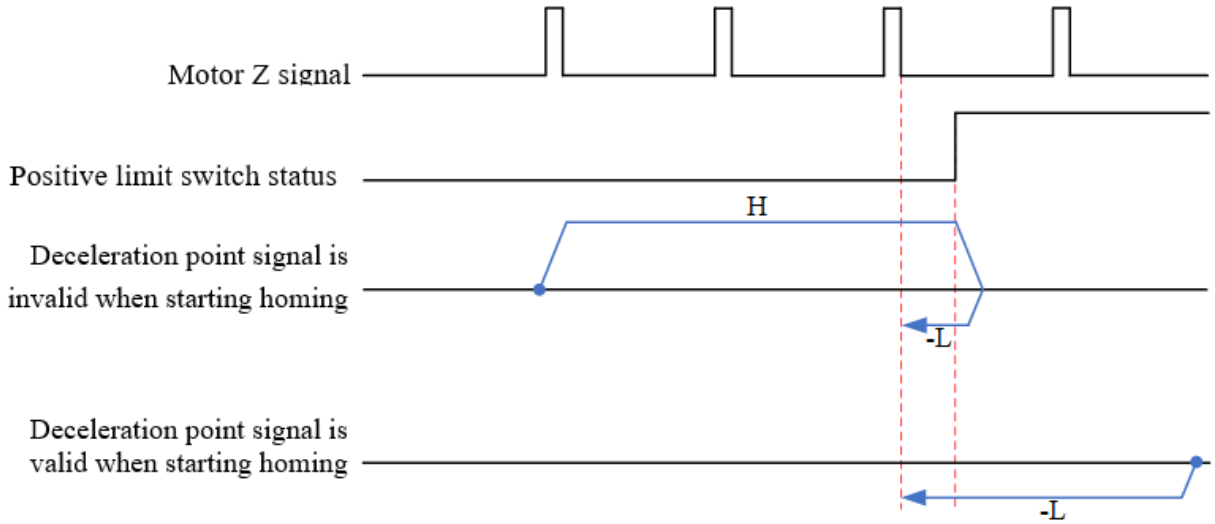
Motion track 1: If the positive limit switch state is invalid when homing starting, it moves towards the positive direction at high speed. After encountering the rising edge of the positive limit switch, it decelerates and stops, and then changes to low speed to move towards the negative direction. After encountering the falling edge of the positive limit switch while travelling negative at low speed, continue to find the nearest Z-signal position in negative at low speed as origin.

Motion track 2: If the positive limit switch state is valid when home starting, it moves towards the

negative direction at low speed. After encountering the falling edge of the positive limit switch in the negative direction, continue to find the nearest Z-signal position as origin at low speed in the negative direction.

In this mode, if the negative limit switch is in a valid state, the process of returning to the original point will be suspended, that is, the command will be interrupted.

The homing process for mode 2 is shown in the figure below:



(3) Mode 3: Seek the falling edge position and Z signal of the origin switch when moving towards the negative direction.

Origin: motor Z signal

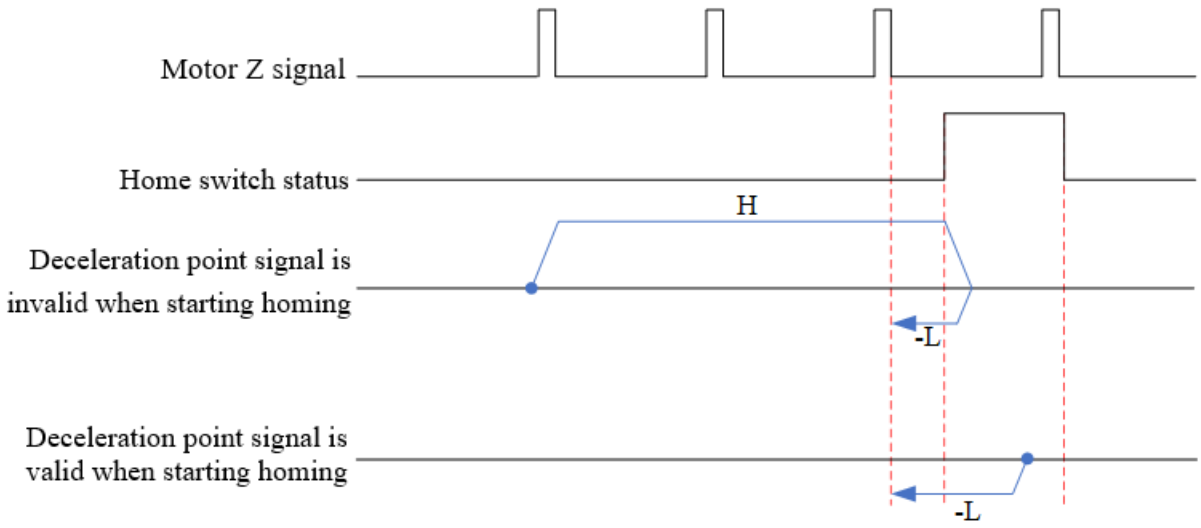
Deceleration point: origin switch

Motion track 1: If the origin switch is invalid when homing starting, it will move forward at high speed. When homing in the positive direction, decelerate and stop after encountering the rising edge of the origin switch, and then change to low speed to move in the negative direction. After encountering the falling edge of the origin switch during low-speed negative movement, continue to find the nearest Z signal position in the negative direction at low speed as the origin.

Motion track 2: If the origin switch is effective when homing starting, it moves towards the negative direction at a low speed. After encountering the falling edge of the origin switch during negative movement, continue to run at low speed towards the negative direction and find the nearest Z signal position as the origin.

In this mode, the return-to-origin process will be aborted regardless of whether the positive or negative limit switch is in valid state, i.e. the command will be interrupted.

The homing process for mode 3 is shown in the figure below:



(4) Mode 5, seek for the falling edge position and Z signal of the origin switch when moving forward.

Origin: motor Z signal

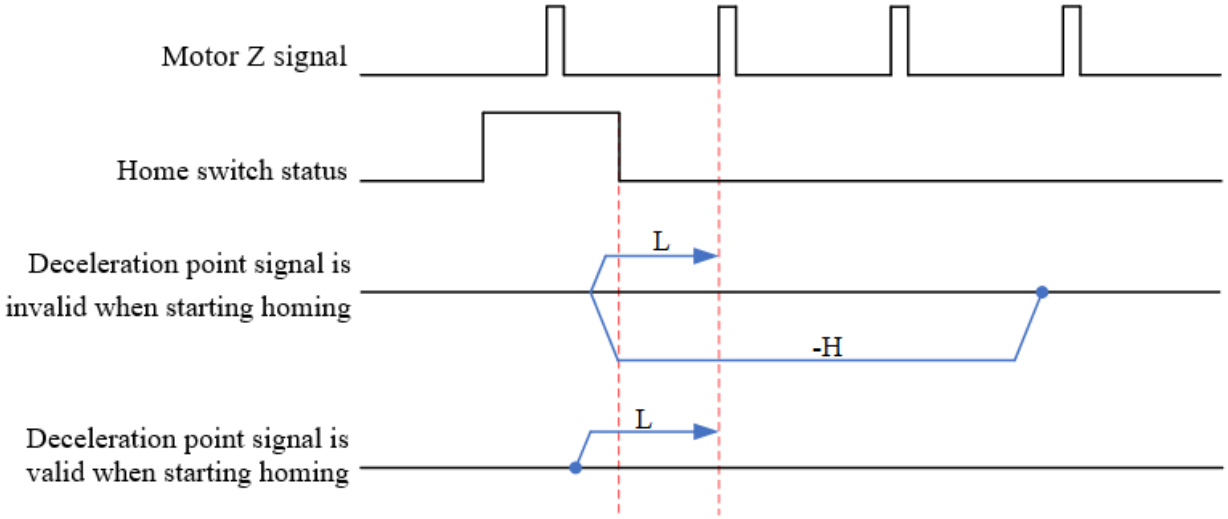
Deceleration point: origin switch

Motion track 1: If the origin switch is invalid when homing starting, it will move towards negative direction at high speed. When homing in the negative direction, decelerate and stop after encountering the rising edge of the origin switch, and then move to the positive direction at a low speed. After encountering the falling edge of the origin switch during low-speed forward motion, continue to find the nearest Z signal position in the forward direction at low speed as the origin.

Motion track 2: If the origin switch is valid when homing starting, it moves forward at a low speed. After encountering the falling edge of the origin switch during forward movement, continue to run at low speed towards the forward direction and find the nearest Z signal position as the origin.

In this mode, the return-to-origin process will be aborted regardless of whether the positive or negative limit switch is in valid state, i.e. the command will be interrupted.

The homing process of mode 5 is shown in the following figure:



(5) Mode 7, seeking for the falling edge position and Z signal of the origin switch when moving towards the negative direction, and automatically reverse in case of positive limit.

Origin: motor Z signal

Deceleration point: origin switch

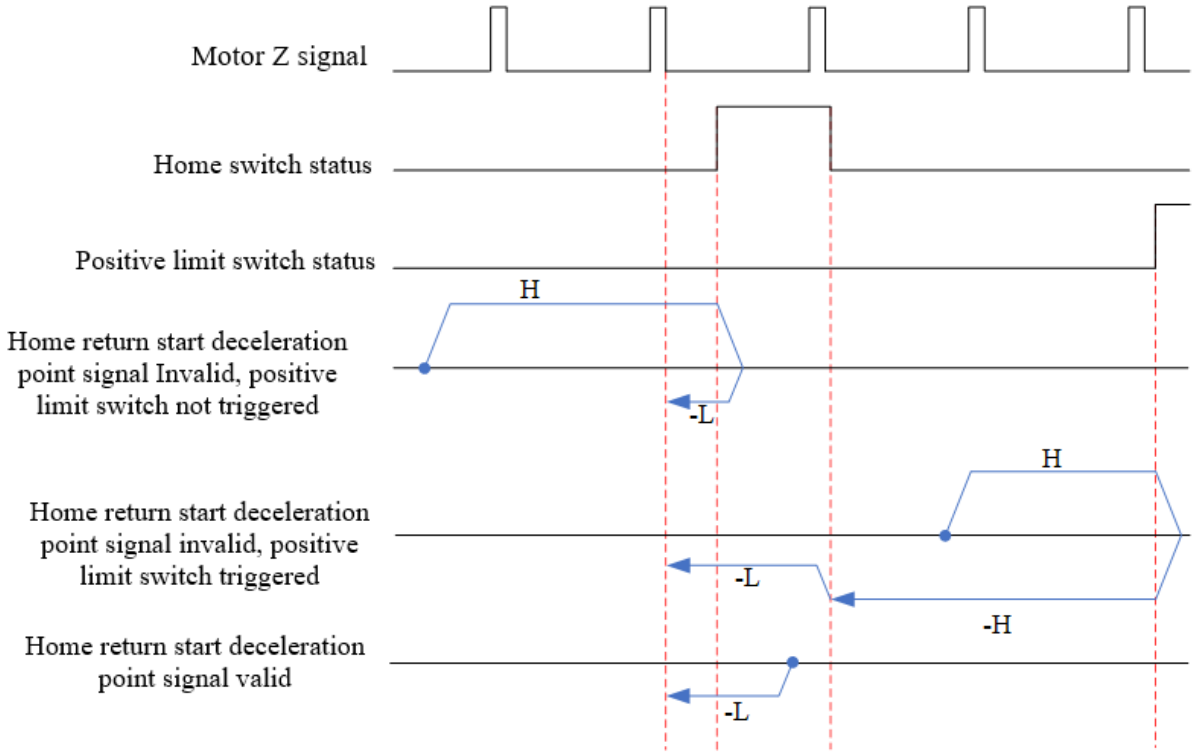
Motion track 1: If the origin switch state is invalid when homing starting, it moves at high velocity towards the positive direction. When moving in the positive direction, it decelerates and stops after encountering the rising edge of the origin switch, and then changes to low speed to move towards the negative direction. After encountering the falling edge of the origin switch during low-speed negative movement, continue to find the nearest Z signal position in the negative direction at low speed as the origin.

Motion track 2: If the origin switch state is invalid when homing starting, it moves towards the positive direction at high speed. When moving in the positive direction, it decelerates and stops when encountering the valid state of the positive limit switch, and then moves towards the negative direction at high speed. When homing in the negative direction, it starts to decelerate to low speed after encountering the rising edge of the origin switch, and then continues to move towards the negative direction at low speed. After encountering the falling edge of the origin switch during low-speed negative movement, continue to find the nearest Z signal position in the negative direction at low speed as the origin.

Motion track 3: If the origin switch state is valid when homing starting, it moves towards negative direction at low speed. After encountering the falling edge of the origin switch during negative movement, continue to find the nearest Z signal position in the negative direction at low speed as the origin.

In this mode, when the forward movement first encounters the valid state of the positive limit switch, it automatically reverses; When the negative limit switch is in the valid state, the return to origin process will be suspended, that is, the command will be interrupted.

The homing process of mode 7 is shown in the following figure:



(6) Mode 11, seek for the falling edge position and Z signal of the origin switch when moving forward, and automatically reverse in case of negative limit.

Origin: motor Z signal

Deceleration point: origin switch

Motion track 1: If the origin switch state is invalid when homing starting, it moves at high speed towards the negative direction. When homing in the negative direction, it decelerates and stops after encountering the rising edge of the origin switch, and then changes to low speed to move towards the positive direction. After encountering the falling edge of the origin switch when moving forward at low speed, continue to find the nearest Z signal position in the forward direction at low speed as the origin.

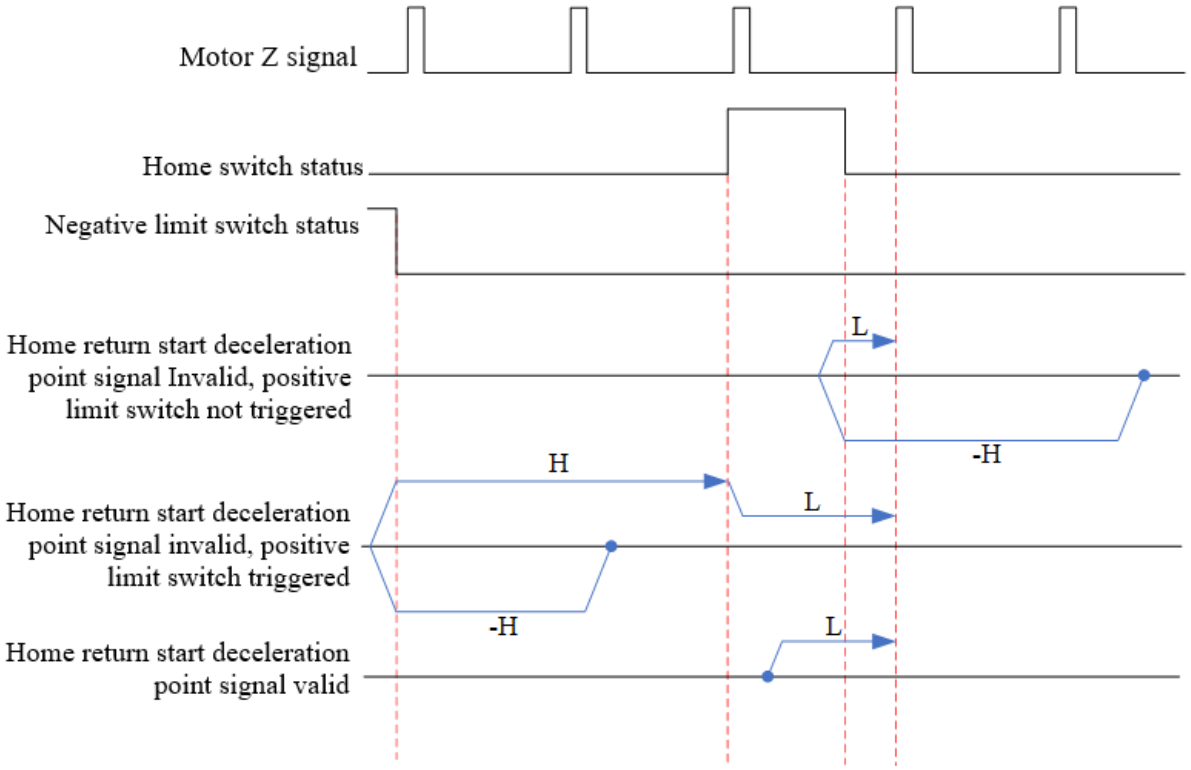
Motion track 2: If the origin switch state is invalid when homing starting, it moves towards the negative direction at high speed. When homing in the negative direction, it decelerates and stops when encountering the effective state of the negative limit switch, and then moves towards the positive direction at high speed. Start deceleration to low speed after encountering the rising edge of the origin switch during forward movement, and then continue moving towards the forward direction at low speed. After encountering the falling edge of the origin switch during low-speed forward motion, continue to find the nearest Z signal position in the forward direction at low speed as the origin.

Motion track 3: If the origin switch state is valid when homing starting, it moves forward at low speed. After encountering the falling edge of the origin switch during forward movement, continue to find the nearest Z signal position in the forward direction at low speed as the origin.

In this mode, when the movement towards the negative direction first encounters the valid state

of the negative limit switch, it automatically reverses; When encountering the valid state of the positive limit switch, the return-to-origin process is aborted, that is, the command is interrupted.

The homing process of mode 11 is shown in the following figure:



(7) Mode 17, Looking for Negative Limit Switch.

Origin: negative limit switch

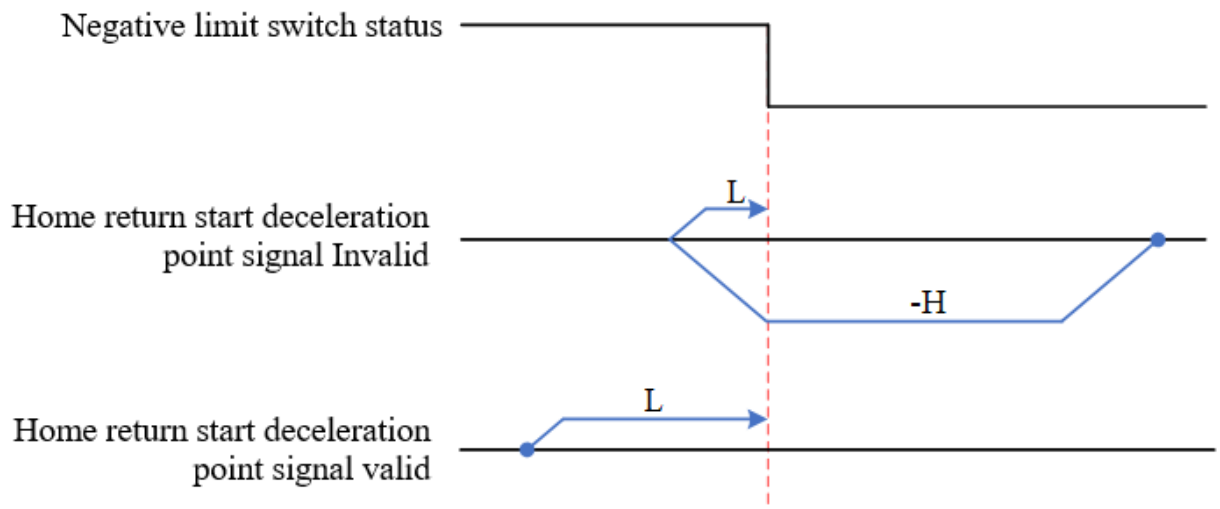
Deceleration point: negative limit switch

Motion track 1: If the negative limit switch state is invalid when starting back to zero, it moves towards the negative direction at high speed. After encountering the rising edge of the negative limit switch, it decelerates and stops, and then changes to low speed to move towards the positive direction. During the low-speed forward movement, find the position where the falling edge of the negative limit switch is located as the origin.

Motion track 2: If the negative limit switch state is valid when home starting, it moves forward at low speed. During the low-speed forward movement, find the position where the falling edge of the negative limit switch is located as the origin.

In this mode, the return-to-origin process will be aborted when the positive limit switch is in a valid state, i.e. the command will be interrupted.

The homing process of mode 17 is shown in the following figure:



(8) Mode 18: Seek for positive limit switch.

Origin: positive limit switch

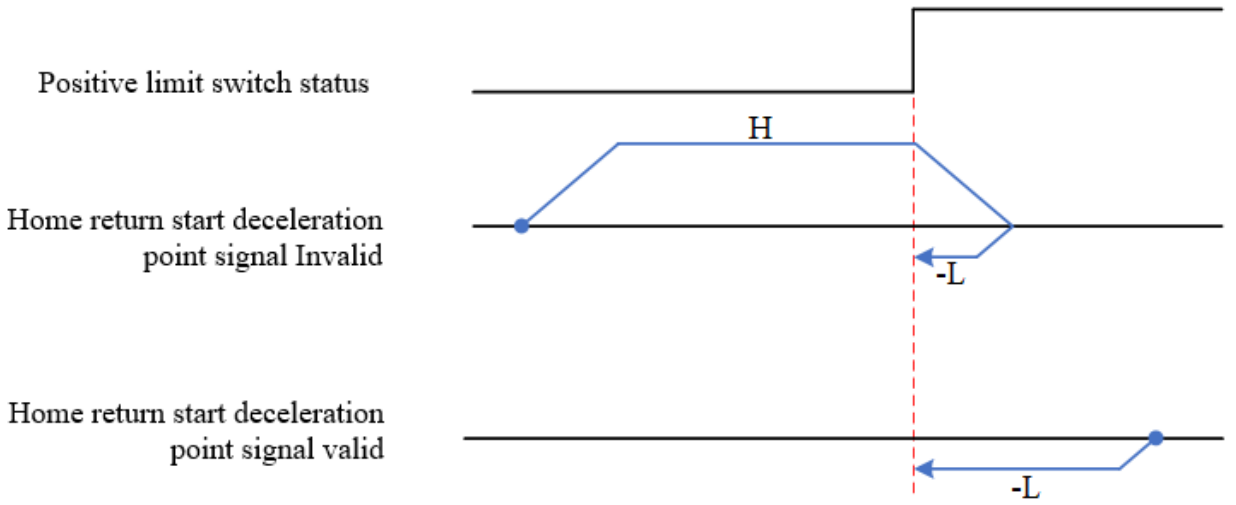
Deceleration point: positive limit switch

Motion track 1: If the positive limit switch state is invalid when starting back to zero, it moves towards the positive direction at high speed. After encountering the rising edge of the positive limit switch, it decelerates and stops, and then changes to low speed to move towards the negative direction. During the low-speed negative movement, align the position where the falling edge of the limit switch is located as the origin.

Motion track 2: If the positive limit switch state is valid when home starting, it moves towards the negative direction at low speed. During the low-speed negative movement, align the position where the falling edge of the limit switch is located as the origin.

In this mode, if the negative limit switch is in a valid state, the process of returning to the original point will be suspended, that is, the command will be interrupted.

The homing process of mode 18 is shown in the following figure:



(9) Mode 19: Seek for the falling edge position of the origin switch when homing in the negative

direction.

Origin: origin switch

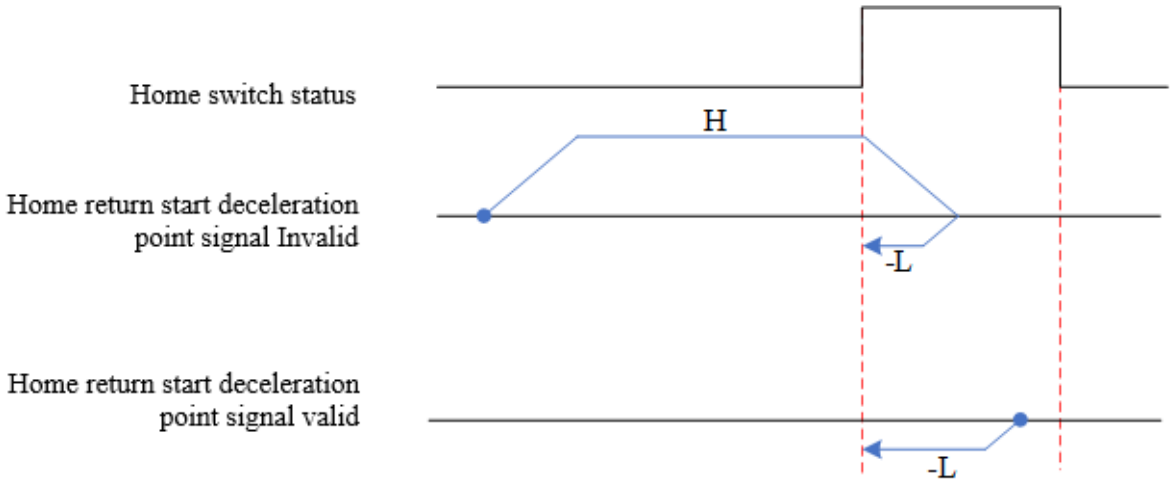
Deceleration point: origin switch

Motion track 1: If the origin switch is invalid when homing starting, it will move forward at high speed. When homing in the positive direction, decelerate and stop after encountering the rising edge of the origin switch, and then change to low speed to move in the negative direction. During low-speed movement towards the negative direction, find the position where the falling edge of the home switch is located as the origin.

Motion track 2: If the origin switch is effective when homing starting, it moves towards the negative direction at a low speed. During low-speed movement towards the negative direction, find the position where the falling edge of the home switch is located as the origin.

In this mode, the return-to-origin process will be aborted regardless of whether the positive or negative limit switch is in valid state, i.e. the command will be interrupted.

The homing process of mode 19 is shown in the figure below:



(10) Pattern 21, searching for the falling edge position of the home switch when moving forward.

Origin: origin switch

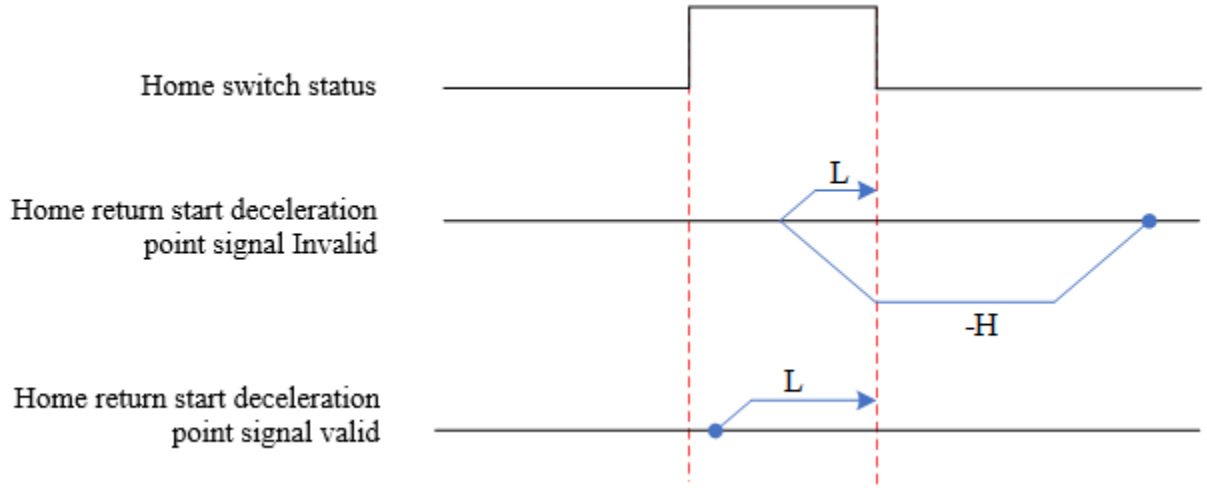
Deceleration point: origin switch

Motion track 1: If the origin switch is invalid when homing starting, it will move towards negative direction at high speed. When homing in the negative direction, decelerate and stop after encountering the rising edge of the origin switch, and then move to the positive direction at a low speed. In the process of low-speed forward movement, find the position where the falling edge of the home switch is located as the origin.

Motion track 2: If the origin switch is effective when homing starting, it moves forward at a low speed. In the process of low-speed forward movement, take the position where the falling edge of the origin finding switch is located as the origin.

In this mode, the return-to-origin process will be aborted regardless of whether the positive or negative limit switch is in valid state, i.e. the command will be interrupted.

The homing process of Pattern 21 is shown in the following figure:



(11) Mode 23: Seek for the falling edge position of the origin switch when it moves towards the negative direction, and automatically reverse in case of positive limit.

Origin: origin switch

Deceleration point: origin switch

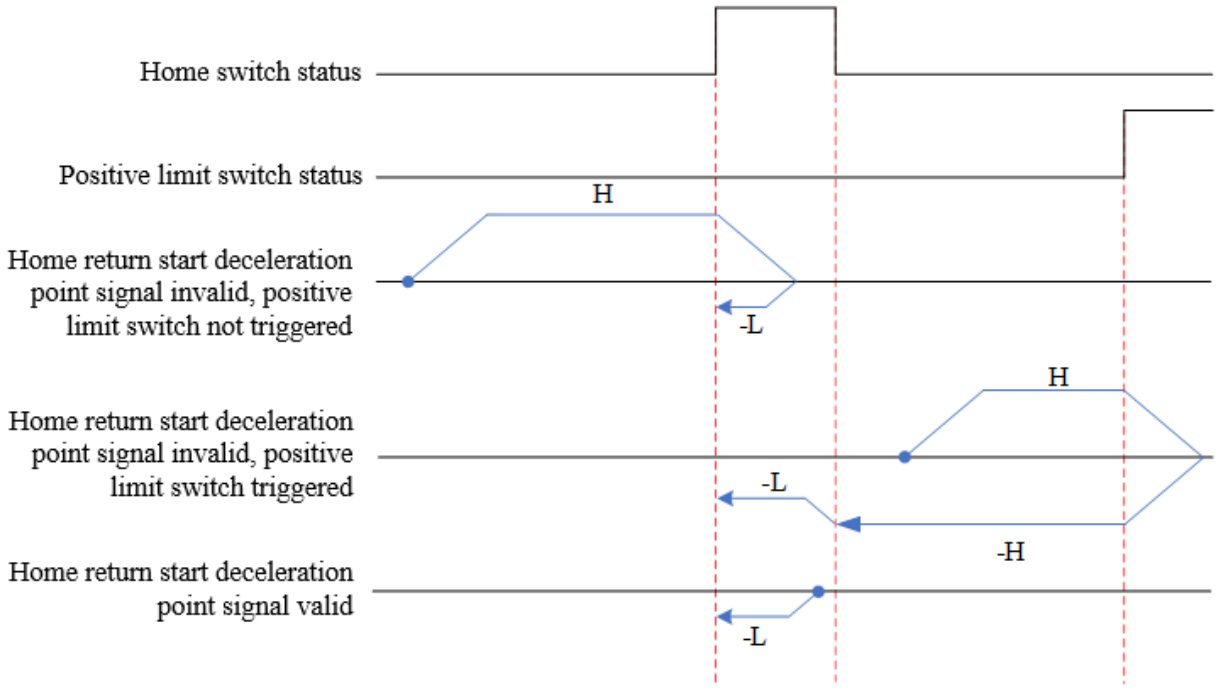
Motion track 1: If the origin switch state is invalid when homing starting, it moves at high speed towards the positive direction. When moving in the positive direction, it decelerates and stops after encountering the rising edge of the origin switch, and then changes to low speed to move towards the negative direction. During low-speed movement towards the negative direction, find the position where the falling edge of the home switch is located as the origin.

Motion track 2: If the origin switch state is invalid when homing starting, it moves towards the positive direction at high speed. When moving in the positive direction, it decelerates and stops when encountering the valid state of the positive limit switch, and then moves towards the negative direction at high speed. When homing in the negative direction, it starts to decelerate to low speed after encountering the rising edge of the origin switch, and then continues to move towards the negative direction at low speed. During low-speed movement towards the negative direction, find the position where the falling edge of the home switch is located as the origin.

Motion track 3: If the origin switch state is valid when homing starting, it moves towards negative direction at low speed. During low-speed movement towards the negative direction, find the position where the falling edge of the home switch is located as the origin.

In this mode, when the positive movement first encounters the valid state of the positive limit switch, it automatically reverses; When the negative limit switch is in the valid state, the return to origin process will be suspended, that is, the command will be interrupted.

The homing process of Pattern 23 is shown in the following figure:



(12) Mode 27: Seek for the falling edge position of the origin switch when moving in positive direction, and automatically reverse in case of negative limit.

Origin: origin switch

Deceleration point: origin switch

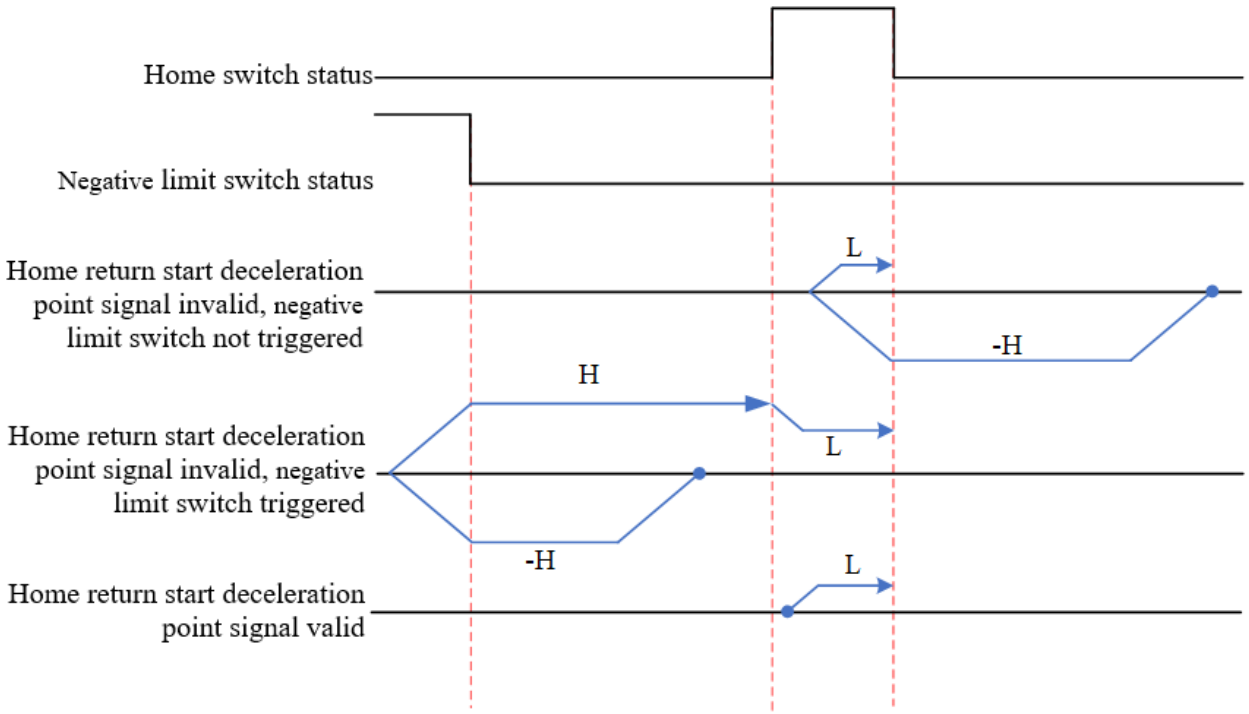
Motion track 1: If the origin switch state is invalid when homing starting, it moves at high speed towards the negative direction. When homing in the negative direction, it decelerates and stops after encountering the rising edge of the origin switch, and then changes to low speed to move towards the positive direction. In the process of low-speed forward movement, find the position where the falling edge of the home switch is located as the origin.

Motion track 2: If the origin switch state is invalid when homing starting, it moves towards the negative direction at high speed. When homing in the negative direction, it decelerates and stops when encountering the effective state of the negative limit switch, and then moves towards the positive direction at high speed. Start deceleration to low speed after encountering the rising edge of the origin switch during forward movement, and then continue moving towards the forward direction at low speed. In the process of low-speed forward movement, find the position where the falling edge of the home switch is located as the origin.

Motion track 3: If the origin switch state is valid when homing starting, it moves forward at low speed. In the process of low-speed forward movement, find the position where the falling edge of the home switch is located as the origin.

In this mode, when the movement towards the negative direction first encounters the valid state of the negative limit switch, it automatically reverses; When encountering the valid state of the positive limit switch, the return-to-origin process is aborted, that is, the command is interrupted.

The homing process of Pattern 27 is shown in the following figure:



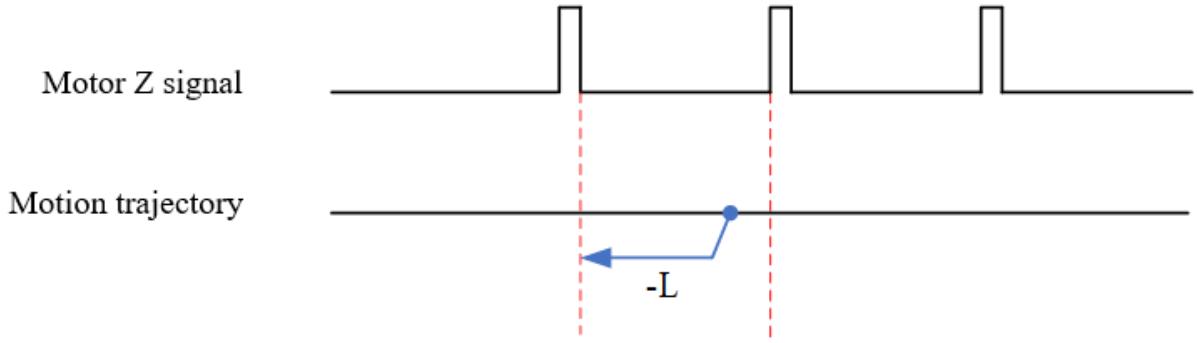
(13) Mode 33: Seek for the Z signal closest to when homing in the negative direction.

Origin: motor Z signal

Motion track: After homing starting, it moves towards the negative direction at low speed, and finds the nearest Z signal position as the origin.

In this mode, the return-to-origin process will be aborted regardless of whether the positive or negative limit switch is in valid state, i.e. the command will be interrupted.

The homing process of Patterns 33 is shown in the following figure:



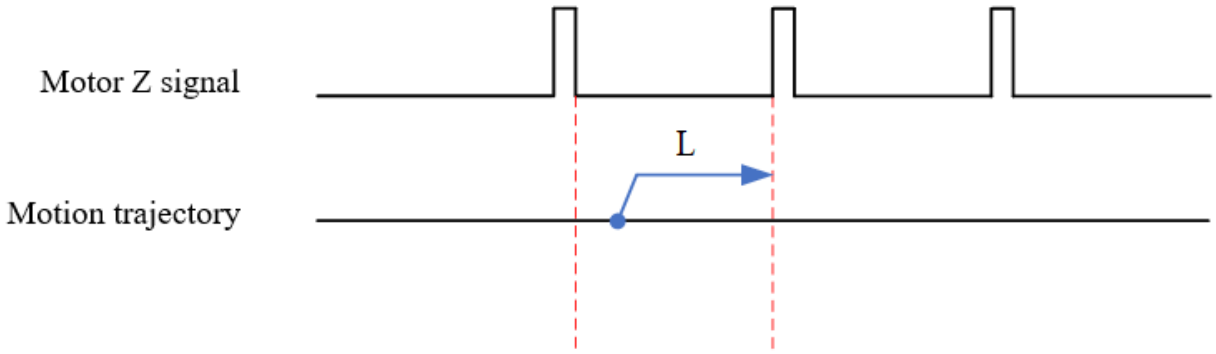
(14) Mode 34, searching for the nearest Z signal when homing in the positive direction.

Origin: motor Z signal

Motion track: After homing starting, it moves towards the forward direction at low speed, and finds the nearest Z signal position as the origin.

In this mode, the return-to-origin process will be aborted regardless of whether the positive or negative limit switch is in valid state, i.e. the command will be interrupted.

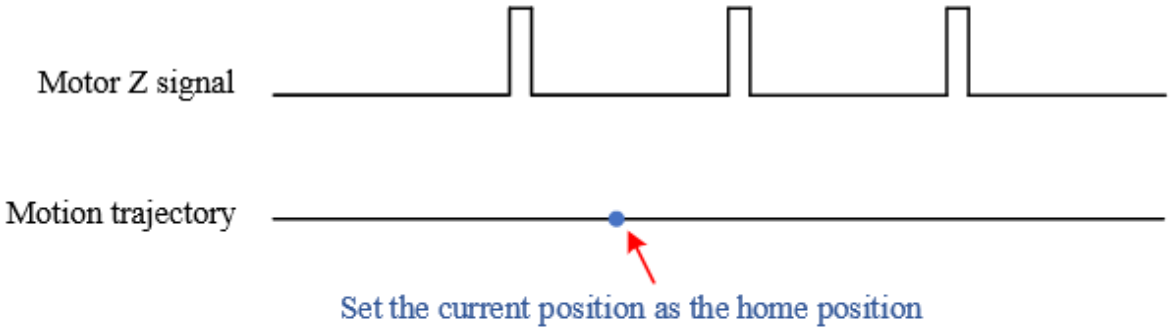
The homing process of Pattern 34 is shown in the following figure:



(15) Mode 35, with the current position as the origin.

Motion track: Use the starting position during homing as the origin for the homing process.

The homing process of Pattern 35 is shown in the following figure:

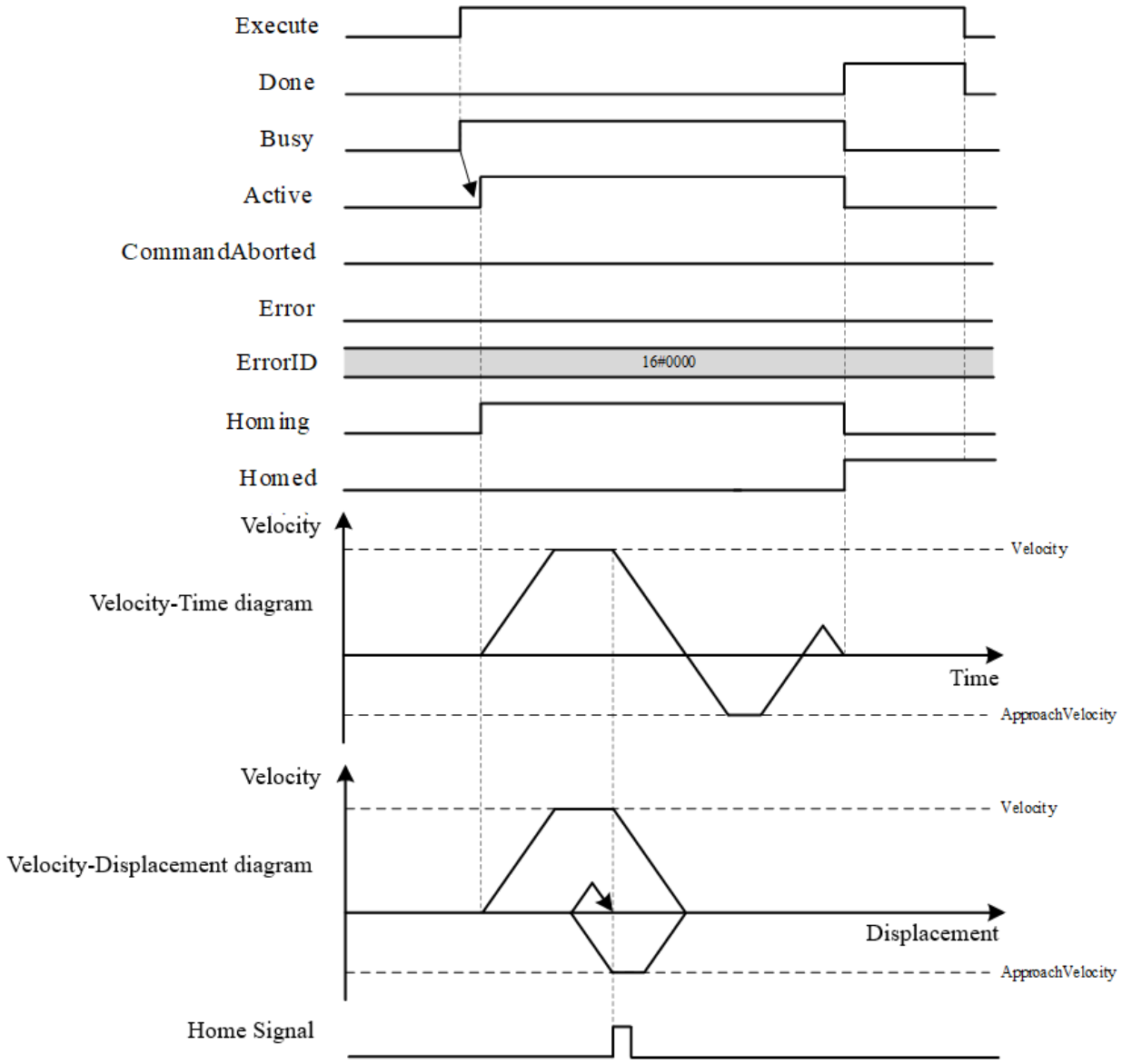


■ Function block timing diagram

(1) Timing diagram of starting the origin return and executing the origin return action normally.

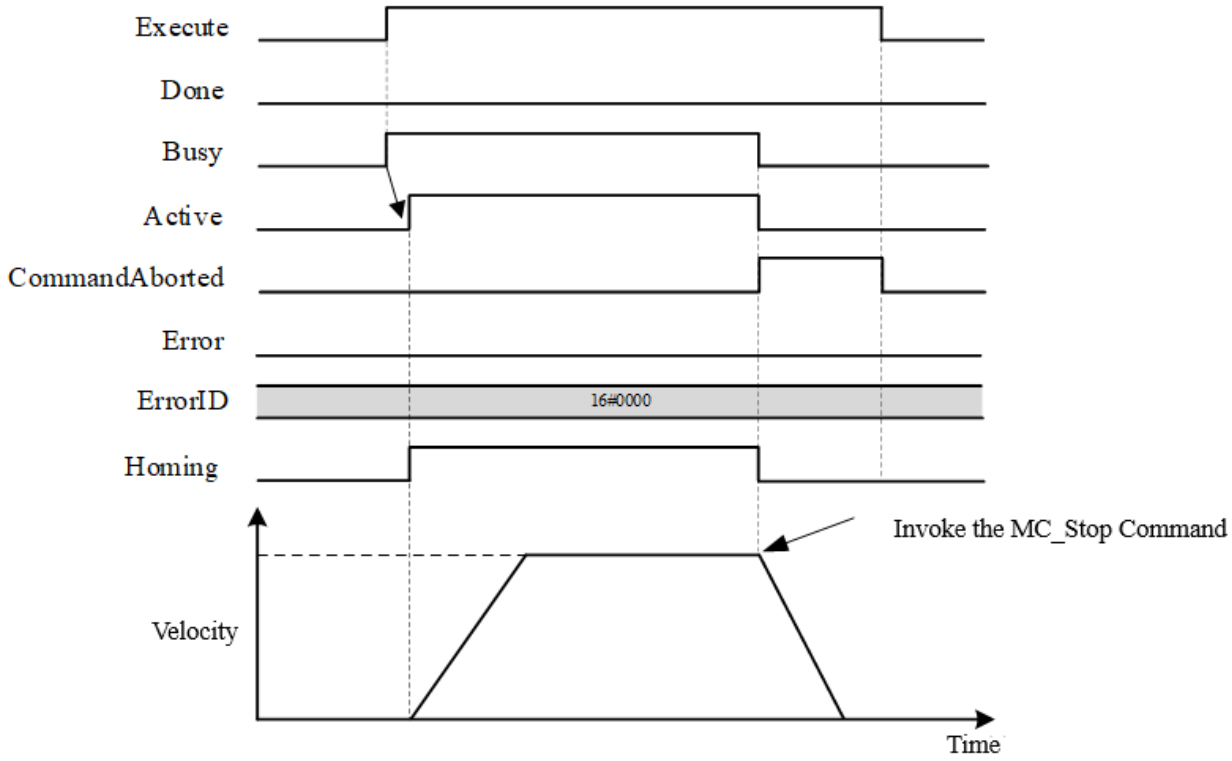
The axis status is Homing during the origin return process.

The following figure takes the homing mode 19 as an example to illustrate the return process.

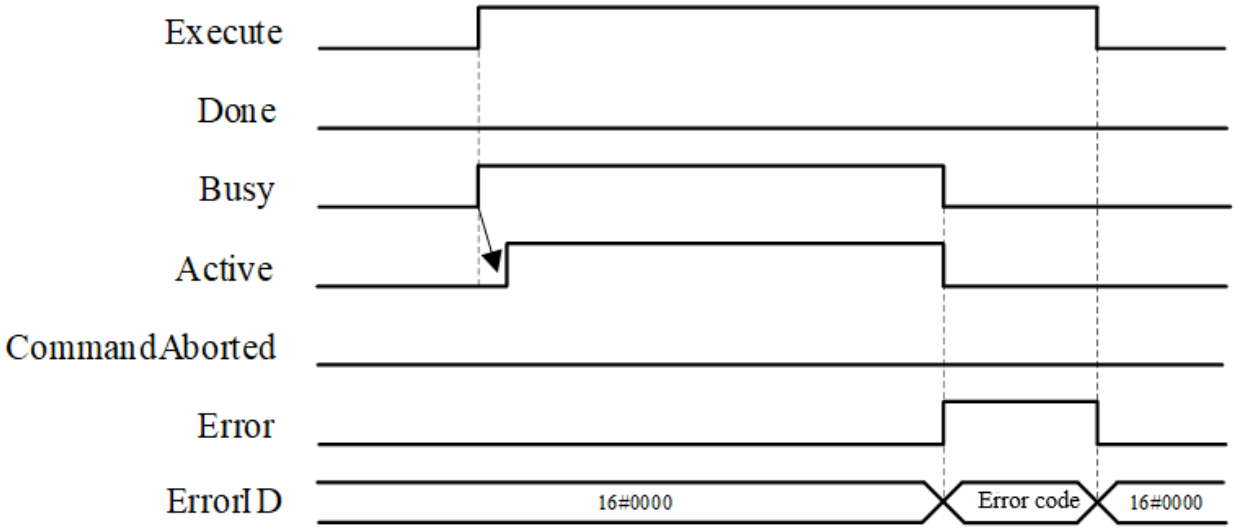


(2) The Timing diagram after the origin return action is interrupted by calling MC_Stop command in the process of homing.

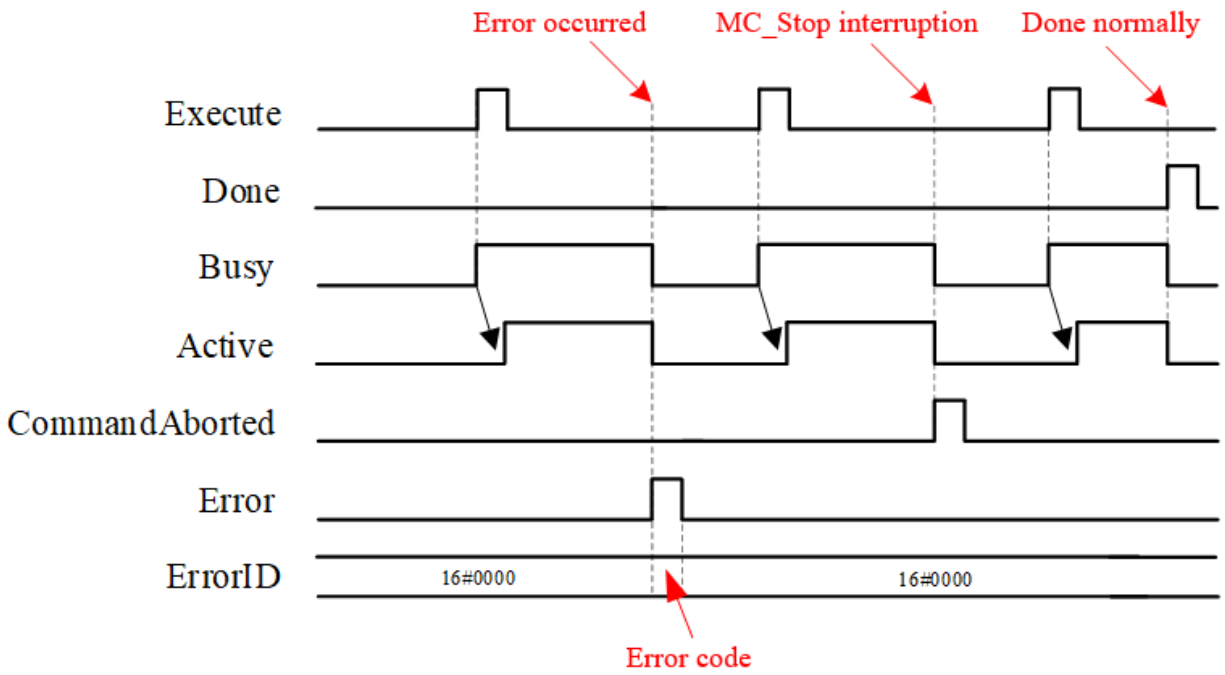
When MC_Stop is called to interrupt, the deceleration set by the input parameter of MC_Stop will be used.



(3) Timing diagram of failure in the homing process.

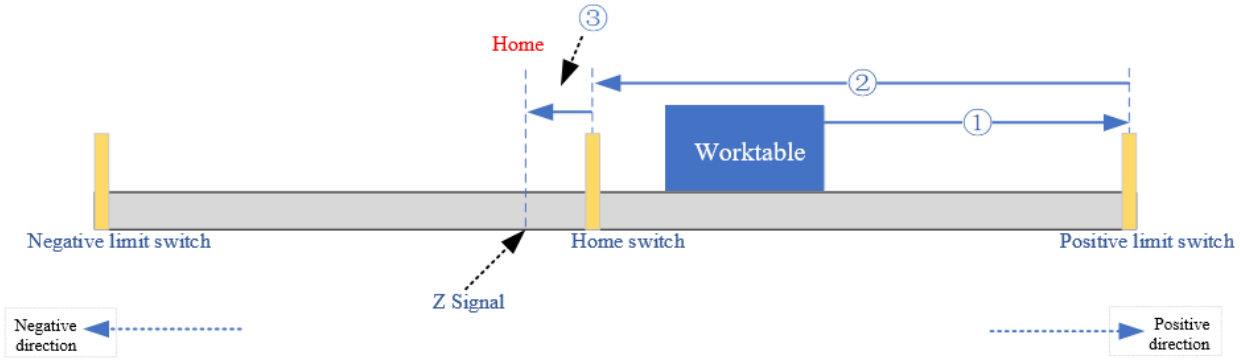


(4) Timing diagram of error, interruption by MC_Stop and normal execution completion during the operation of functional block when Execute is maintained for a cycle.



5.4.3 Examples

In this example process, the workbench must perform an origin return step before commencing any operations. The origin position is defined as the first Z signal in the negative direction of the origin switch. Only after successfully locating the origin can subsequent process steps be executed.



Based on the above requirements, the MC_Home command is used for home operation, and home mode 7 should be selected as the home mode. Taking the current stop position of the table as an example, when using the home mode 7, the table first moves forward at Velocity high speed. After encountering the positive limit switch, it decelerates and finds the origin switch reversely at Velocity high speed. After finding the origin switch, it then finds the nearest Z signal as the origin at ApproachVelocity low speed.

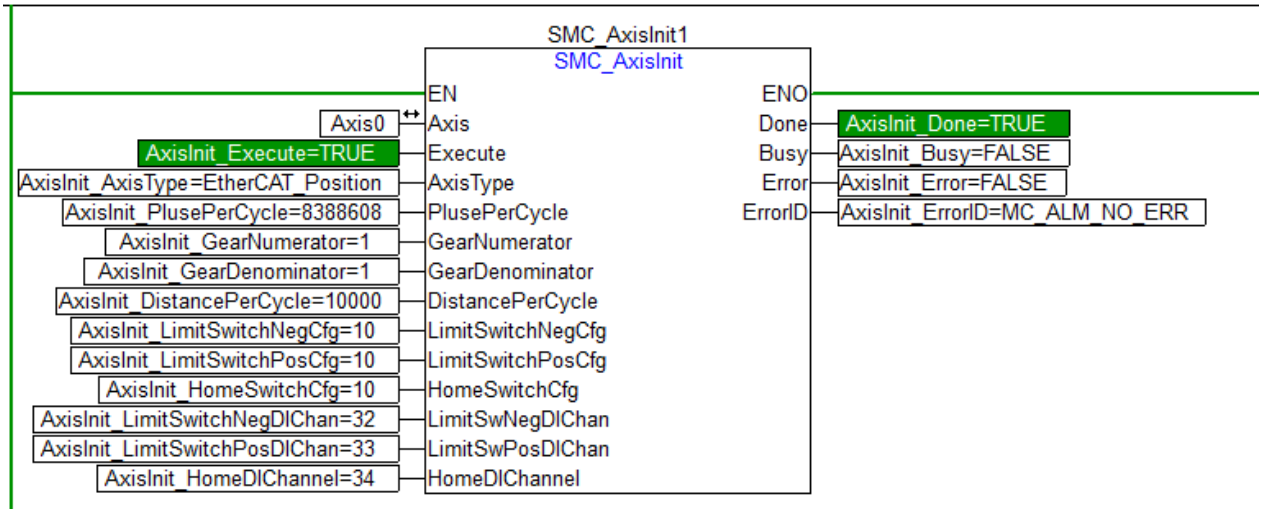
Variable

Sr.No	Variable Name	Variable Type	Initial Value	Variable Description
1	MC_Home	MC_HOME	-	Home command

2	Home_En	BOOL	FALSE	Home enable mark
3	Home_HomeMode	SMC_HOME_MODE	MC_HM1_NEG_LIMIT_Z	Home mode
4	Home_Vel	LREAL	10000	Home speed for finding deceleration point at high speed
5	Home_Vel2	LREAL	5000	Home low-speed origin finding speed
6	Home_Acc	LREAL	10000	Home acceleration
7	Home_Dec	LREAL	10000	Homing deceleration
8	Home_Jerk	LREAL	0	Home jerk
9	Home_Pos	LREAL	0	Zero position after homing
10	Home_Done	BOOL	FALSE	Home completion flag
11	Home_Busy	BOOL	FALSE	Home busy status flag
12	Home_Active	BOOL	FALSE	Home command operation mark
13	Home_Abort	BOOL	FALSE	Home abort flag
14	Home_Error	BOOL	FALSE	Home error flag
15	Home_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	Home error code
16	MC_MoveAbsolute	MC_MOVEABSOLUTE	-	Absolute position positioning command
17	MoveAbs_En	BOOL	FALSE	Absolute position positioning enable mark
18	MoveAbs_Pos	LREAL	20000	Absolute position locating target position
19	MoveAbs_Vel	LREAL	10000	Absolute position positioning speed
20	MoveAbs_Acc	LREAL	100000000	Absolute position positioning acceleration
21	MoveAbs_Dec	LREAL	100000000	Absolute position positioning deceleration
22	MoveAbs_Dir	MC_DIRECTION	mcPositiveDirection	Absolute position positioning direction
23	MoveAbs_Jerk	LREAL	0	Absolute position positioning jerk
24	MoveAbs_Buff	MC_BUFFER_MODE	Aborting	Absolute position positioning buffer mode
25	MoveAbs_Done	BOOL	FALSE	Absolute position positioning completion flag
26	MoveAbs_Busy	BOOL	FALSE	Absolute position positioning busy status flag
27	MoveAbs_Active	BOOL	FALSE	Absolute position positioning command active flag
28	MoveAbs_Abort	BOOL	FALSE	Absolute position positioning abort flag
29	MoveAbs_Error	BOOL	FALSE	Absolute position positioning error flag
30	MoveAbs_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	Absolute position positioning error codes

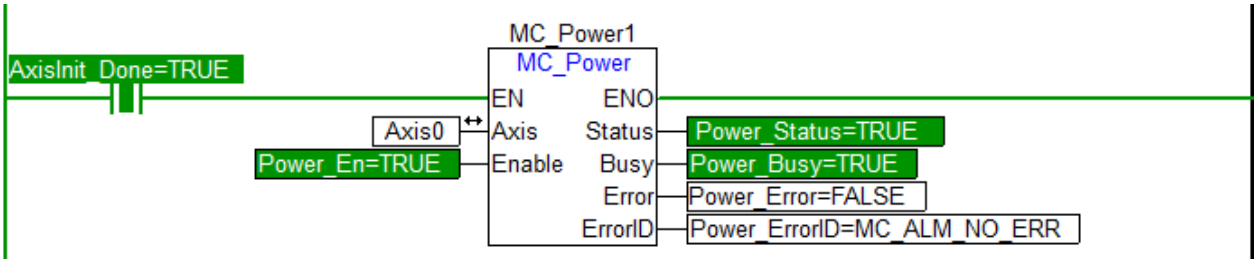
■ LD program implementation

(1) First, call the SMC_AxisInit command to initialize the axis parameters, set the axis type as EtherCAT_Poaition, and set the user unit, origin switch signal and positive/negative limit switch signal. The commands are as follows:

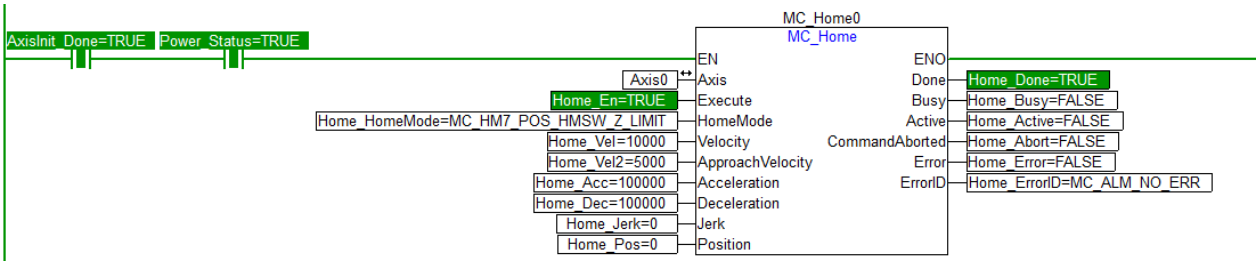


(2) Before calling the MC_Home command, call the MC_Power command to enable the

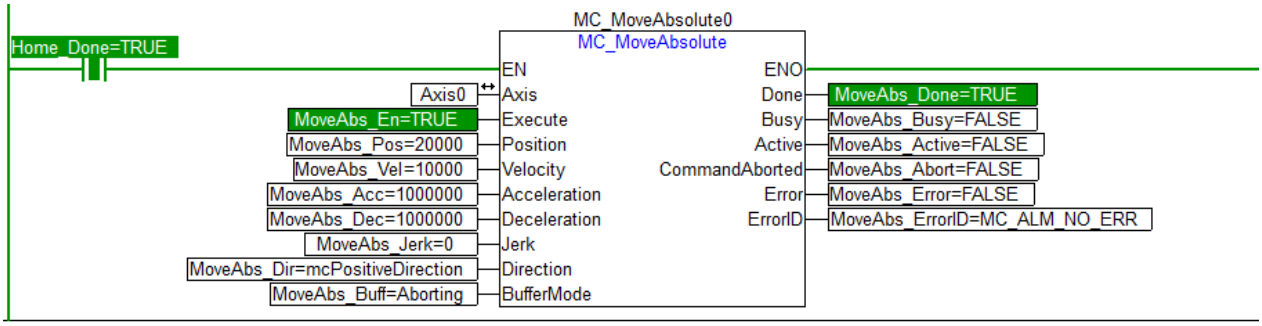
corresponding axis.



(3) Call the MC_Home command for homing operation, and set the homing mode to 7.



(4) After the homing operation is completed, perform other process flows, such as absolute position positioning with the origin as the starting point.



■ ST program implementation

(1) First, call the SMC_AxisInit command to initialize the axis parameters, set the axis type as EtherCAT_Poaition, and set the user unit, origin switch signal and positive/negative limit switch signal. The commands are as follows:

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
```

```
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

(2) Before calling the MC_Home command, call the MC_Power command to enable the corresponding axis.

```
IF TRUE = AxisInit_Done THEN
   Power_En := TRUE;
ELSE
   Power_En := FALSE;
END_IF;
MC_Power1(
Enable := Power_En,
Axis := Axis0,
Status=>Power_Status,
Busy=>Power_Busy,
Error=>Power_Error,
ErrorID=>Power_ErrorID);
```

(3) Call the MC_Home command for homing operation, and set the homing mode to 7.

```
IF TRUE = Power_Status THEN
   MC_Home0(
   Execute := Home_En,
   HomeMode := Home_HomeMode,
   Velocity := Home_Vel,
   ApproachVelocity := Home_Vel2,
   Acceleration := Home_Acc,
   Deceleration := Home_Dec,
   Jerk := Home_Jerk,
   Position := Home_Pos,
   Axis := Axis0,
   Done=>Home_Done,
   Busy=>Home_Busy,
   Active=>Home_Active,
   CommandAborted=>Home_Abort,
   Error=>Home_Error,
   ErrorID=>Home_ErrorID);
END_IF
```

(4) After the homing is completed, perform other process flows, such as absolute position positioning with the origin as the starting point.

```
IF TRUE=Home_Done THEN
   MC_MoveAbsolute0(
```

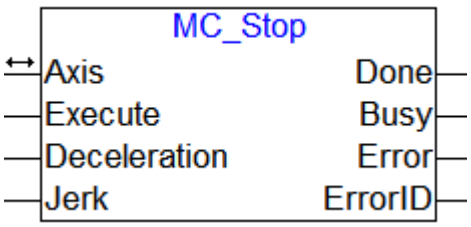
```
    Execute := MoveAbs_En,
    Position := MoveAbs_Pos,
    Velocity := MoveAbs_Vel,
    Acceleration := MoveAbs_Acc,
    Deceleration := MoveAbs_Dec,
    Jerk := MoveAbs_Jerk,
    Direction := MoveAbs_Dir,
    BufferMode := MoveAbs_Buff,
    Axis := Axis0,
    Done=>MoveAbs_Done,
    Busy=>MoveAbs_Busy,
    Active=>MoveAbs_Active,
    CommandAborted=>MoveAbs_Abort,
    Error=>MoveAbs_Error,
    ErrorID=>MoveAbs_ErrorID);
END_IF
```

5.4.4 Precautions

When this command is set to the homing mode related to Z signal, if the probe capture channel 1 has been occupied, the command reports an error. The MC_TouchProbe and HMC_GantryInit commands will occupy the probe capture channel. During use, the probe capture channel 1 is used for Z signal capture in a staggered manner.

5.5 MC_Stop (Axis Stop)

The control axis stops at the specified deceleration, any motion control command being executed on this axis is suspended, and all motion buffer commands on this axis are canceled.



5.5.1 Parameter

Parameter Type	Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL

	Deceleration	Deceleration	No	-	Positive value	LREAL
	Jerk	Jerk 0: Trapezoidal acceleration and deceleration Positive: S-shaped acceleration and deceleration	YES	0	0/positive	LREAL
OUTPUT	Done	Stop completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.5.2 Functions

This command controls the axis to decelerate and stop from the current speed according to the specified deceleration, and suspends any motion control command being executed on the axis.

■ command function details

(1) Deceleration parameter setting

The deceleration and jerk at the time of deceleration stop are specified by Deceleration and Jerk , respectively, of the input variables.

(2) Axis status when running this command

When the command rising edge triggers the pin to remain at high level, and the axis is always in Stopping state, the axis cannot deliver motion command.

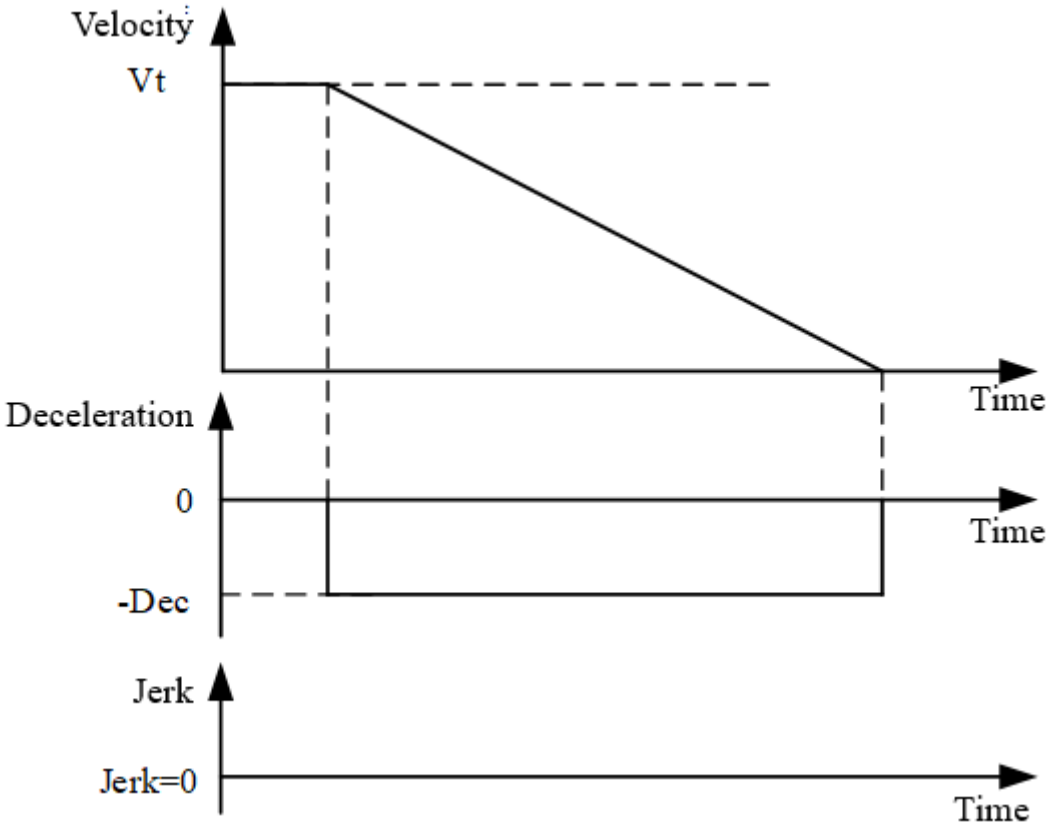
(3) Axis latch

The rising edge of this command is triggered. When it is triggered, the axis ID of the input and output parameter Axis, the input parameter Deceleration and the Jerk value are latched. During the execution of the command, the parameters entered by the modified command are invalid until this command is triggered repeatedly.

(4) Acceleration and deceleration curve

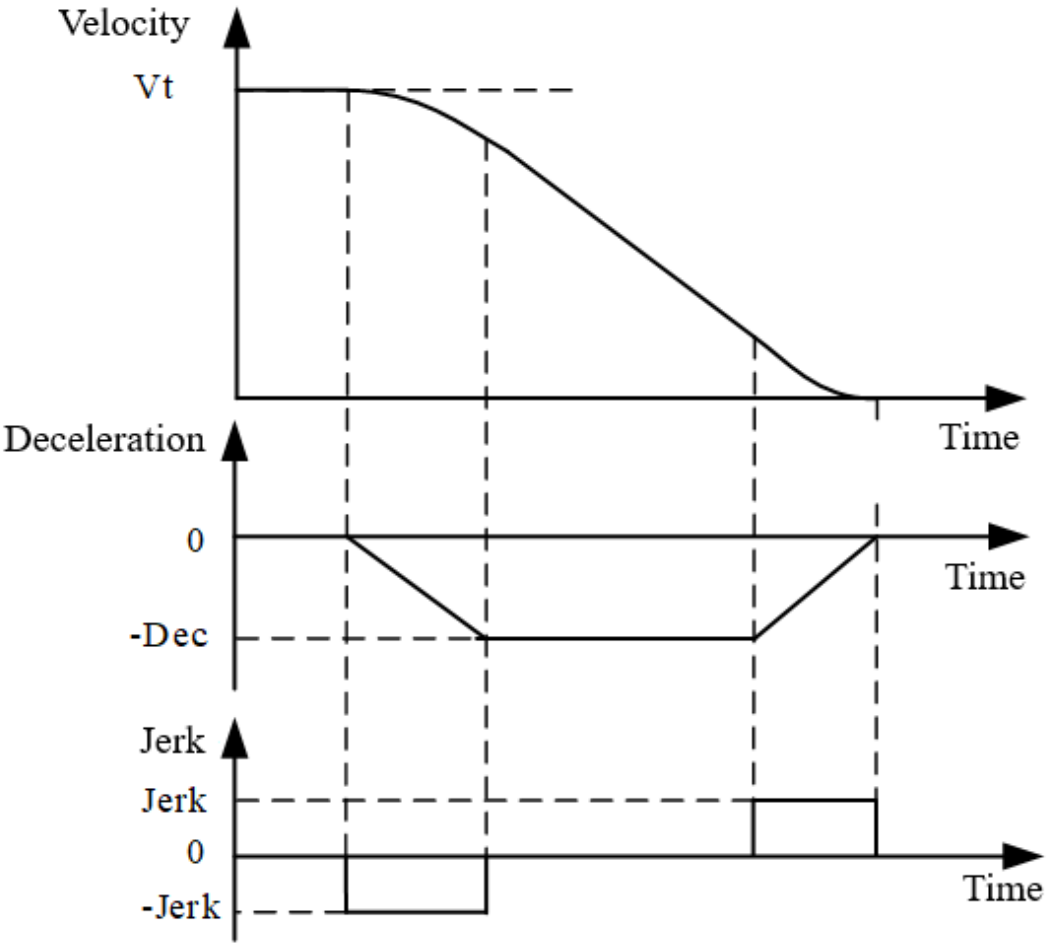
If the input parameter Jerk=0, the deceleration curve is a trapezoidal acceleration/deceleration curve; if the input parameter Jerk>0, the deceleration curve is an S-shaped acceleration/deceleration curve.

□ When Jerk=0, the deceleration curve is a trapezoidal acceleration/deceleration curve as follows:



Vt: Velocity at which deceleration starts, Dec: set value of deceleration, Jerk: set value of jerk

- When Jerk>0, the command value of speed is generated by deceleration. The deceleration curve is an S-shaped acceleration/deceleration curve as follows:



Vt: Velocity at the start of deceleration, Dec: Deceleration setting value, Jerk: Jerk setting value

■ Axis types supported by this command:

Virtual axis (0): virtual axis

EtherCAT_Position(50): EtherCAT bus connection axis, position control

EtherCAT_Speed(51): EtherCAT bus connection axis, speed control

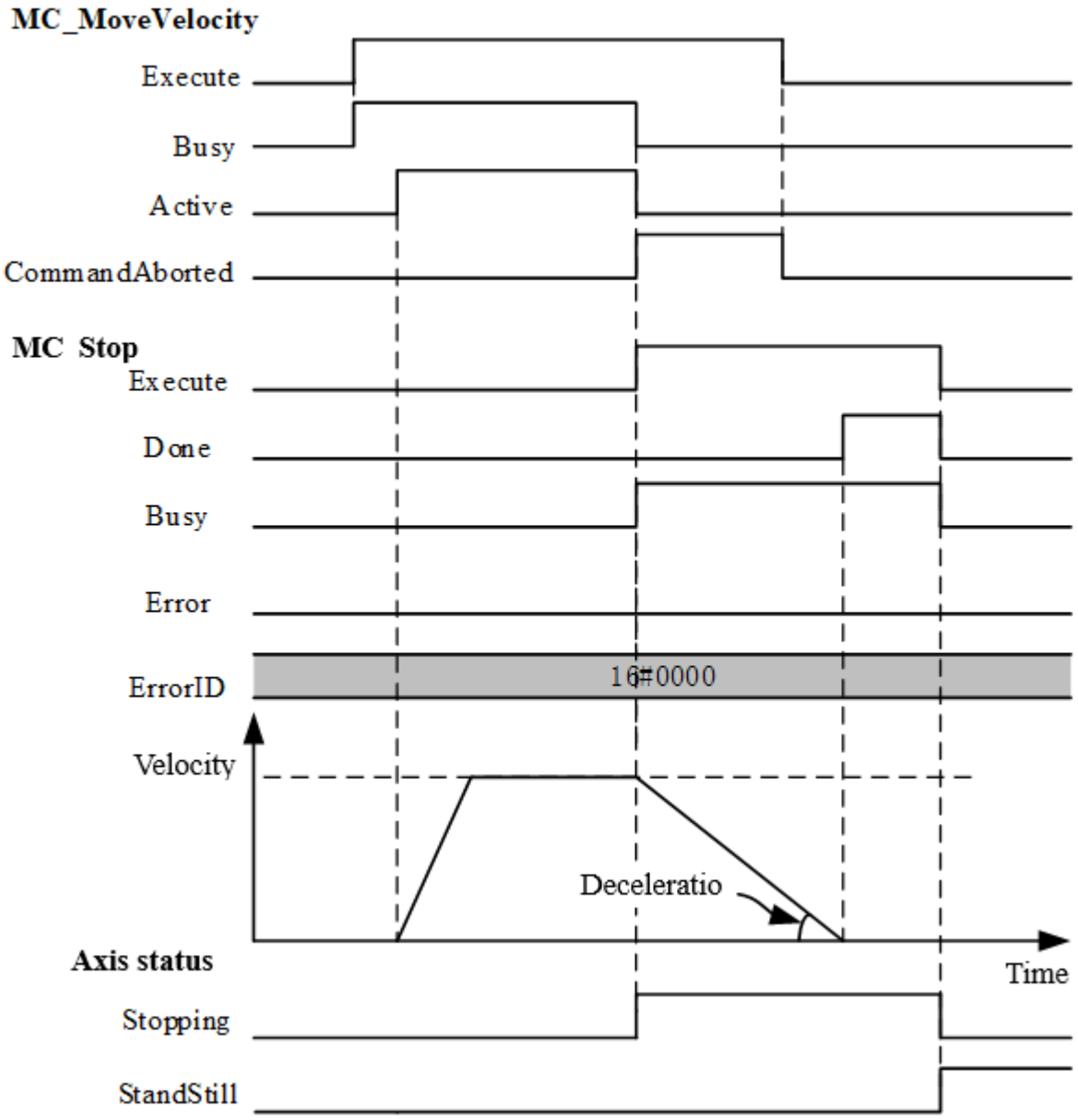
EtherCAT_Torque(52): EtherCAT bus connecting shaft, torque control

■ Timing diagram

(1) Normal timing diagram

On the rising edge of Execute input, if there is no abnormality, the Busy pin is set to TRUE. When the command is executed, the axis is in Stopping status and its motion decelerates and stops. Set to TRUE when Deceleration Done is complete. If Execute is kept at high level, the Busy and Done pins remain TRUE and the axis remains in Stopping status; if Execute is set at low level, the Busy and Done pins are set to FALSE and the axis switches to StandStill status.

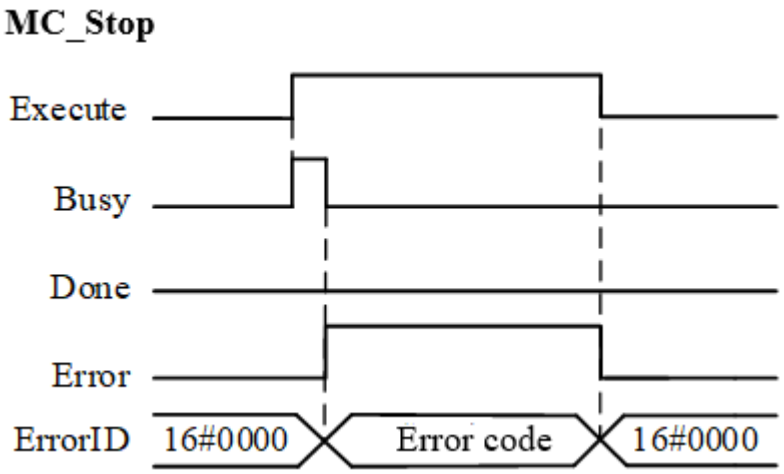
The correct sequence sequence for command execution is as follows:



(2) Pin timing when error occurs

When this command is executed, if the command input parameter is illegal, the axis is in Disabled status or ErrorStop status and other abnormalities occur, the call of this command will fail. At this time, the command output pin Error=True, and the error code ErrorID reports an abnormal error value. Please refer to the appendix Description of error codes to understand the cause of the error.

When the command input pin Execute is low, all output pins are restored to their Default Values.

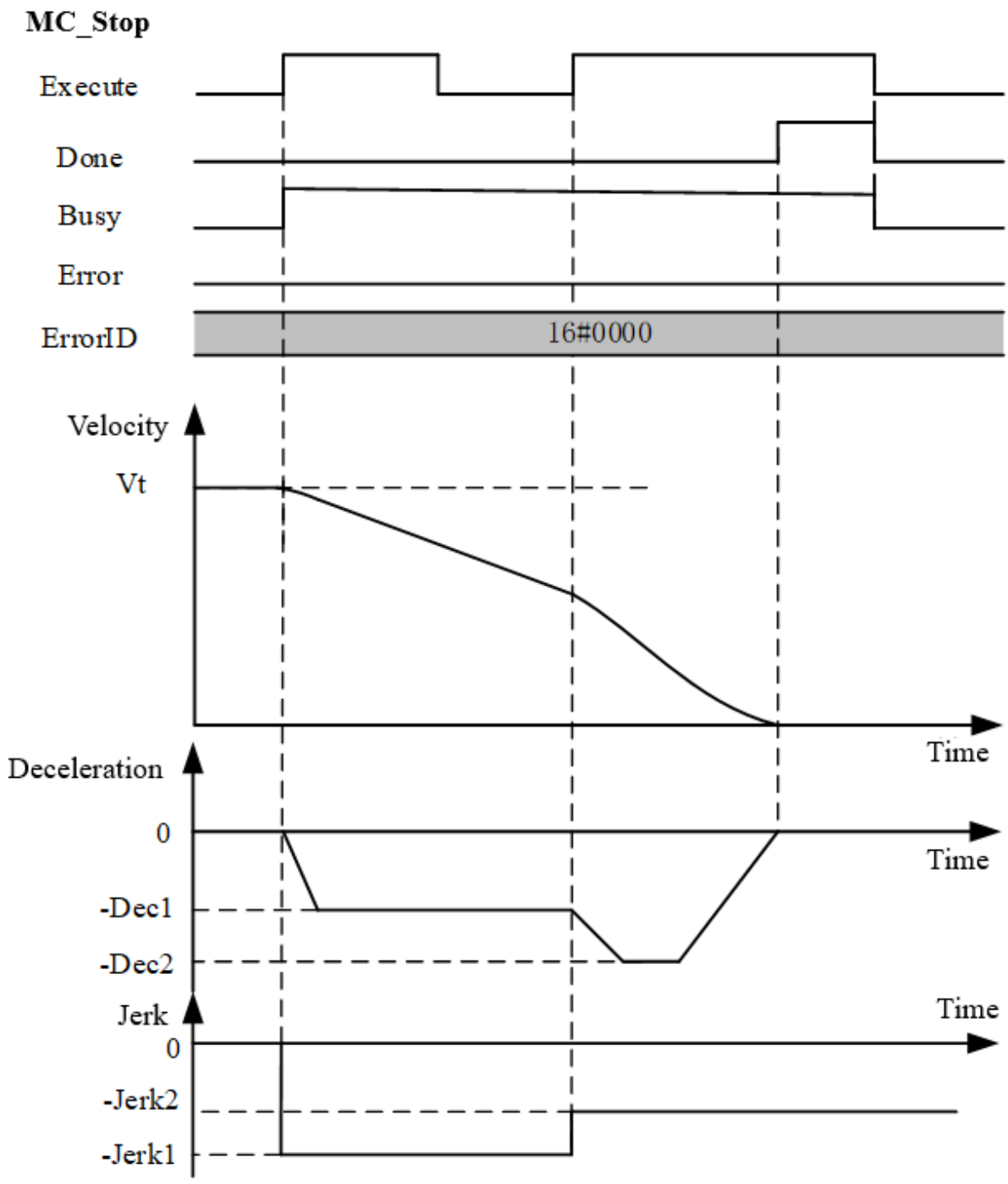


When the command error is solved, it is necessary to trigger a rising edge on the Execute input to initiate the MC_Stop function block.

(3) Repeat delivery command

When multiple MC_Stop functional blocks are triggered repeatedly, if there is a change in Deceleration and Jerk, the deceleration-stop control will be performed with the deceleration and jerk of the last delivery command, and the command will change the deceleration and jerk.

The following figure shows the timing diagram for a repeat delivery command.



Vt: Velocity at the start of deceleration, Dec1: Deceleration during the first command execution, Dec2: Deceleration during the second command execution, Jerk1: Jerk during the first command execution, Dec2: Jerk during the second command execution

5.5.3 Examples

In this example, Axis0 is set as the command axis and the Variable Type is AXIS_REF. The MC_MoveVelocity motion command is stopped with the MC_Stop command, which reads the current axis

status.

■ LD program implementation

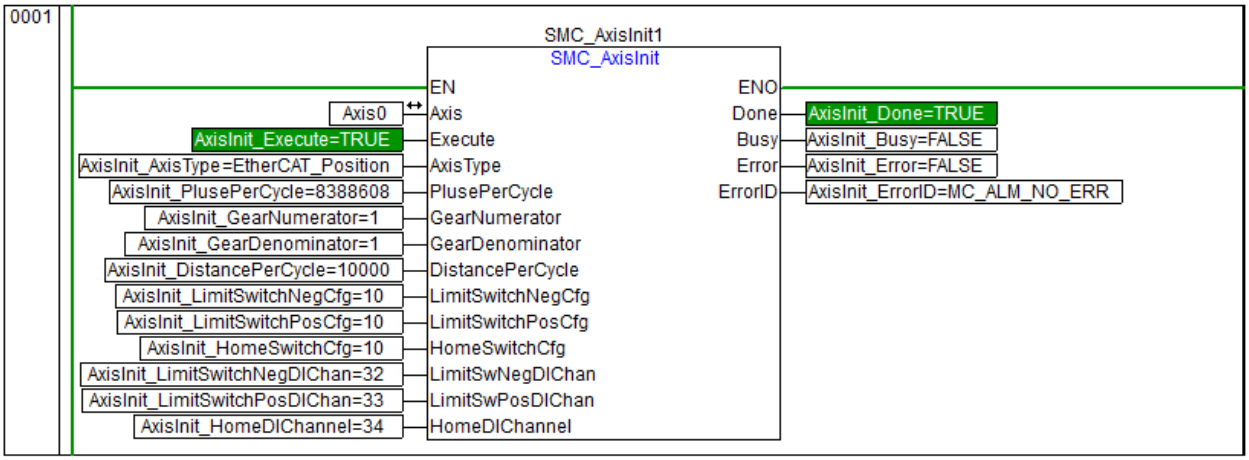
Use the ladder diagram to establish a program, input parameters Deceleration 10000, Jerk 0, trapezoidal acceleration and deceleration.

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
Axis0	AXIS_REF	-	Axis variable for axis 0
Power1_Status	BOOL	FALSE	Variable for which the axis enable is successful. This variable becomes TRUE when the axis is enabled successfully.
MV1_Active	BOOL	FALSE	Variable when the axis speed control command acts. This variable becomes TRUE when the axis is controlling motion.
MV1_Comd	BOOL	FALSE	Variable whose speed control command is interrupted. This variable becomes TRUE and the speed command is interrupted.
Stop1_Exe	BOOL	FALSE	Trigger variable for the rising edge of an axis stop command.
Stop1_Done	BOOL	FALSE	Variable at which the axis was successfully stopped. Done = TRUE, axis stop is finished.
Switch1	BOOL	FALSE	Axis ready state variable. After axis initialization, Switch to TRUE and the axis is ready.

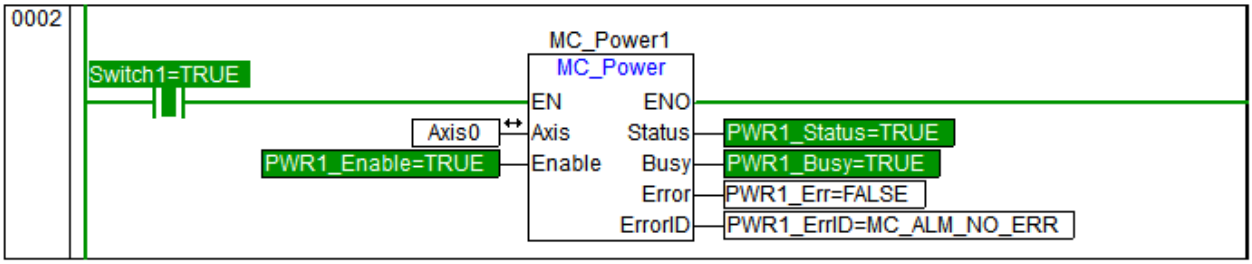
(1) Axis initialization

Call the SMC_AxisInit command to complete the initial configuration of Axis0, and set the axis type as 50:EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to the command Description section of SMC_AxisInit.



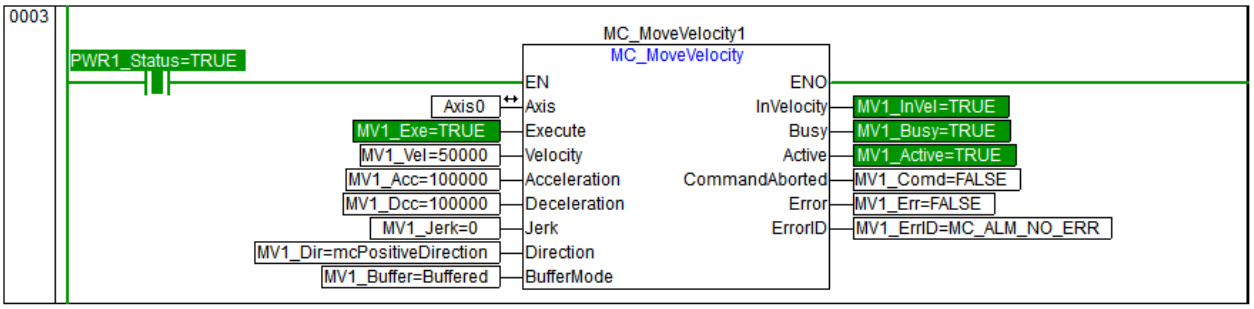
(2) Axis enable

In section 0002, the axis initialization is completed, the status variable Switch1 of axis ready becomes TRUE, and MC_Power is called to enable the axis.



(3) Axis delivery motion command

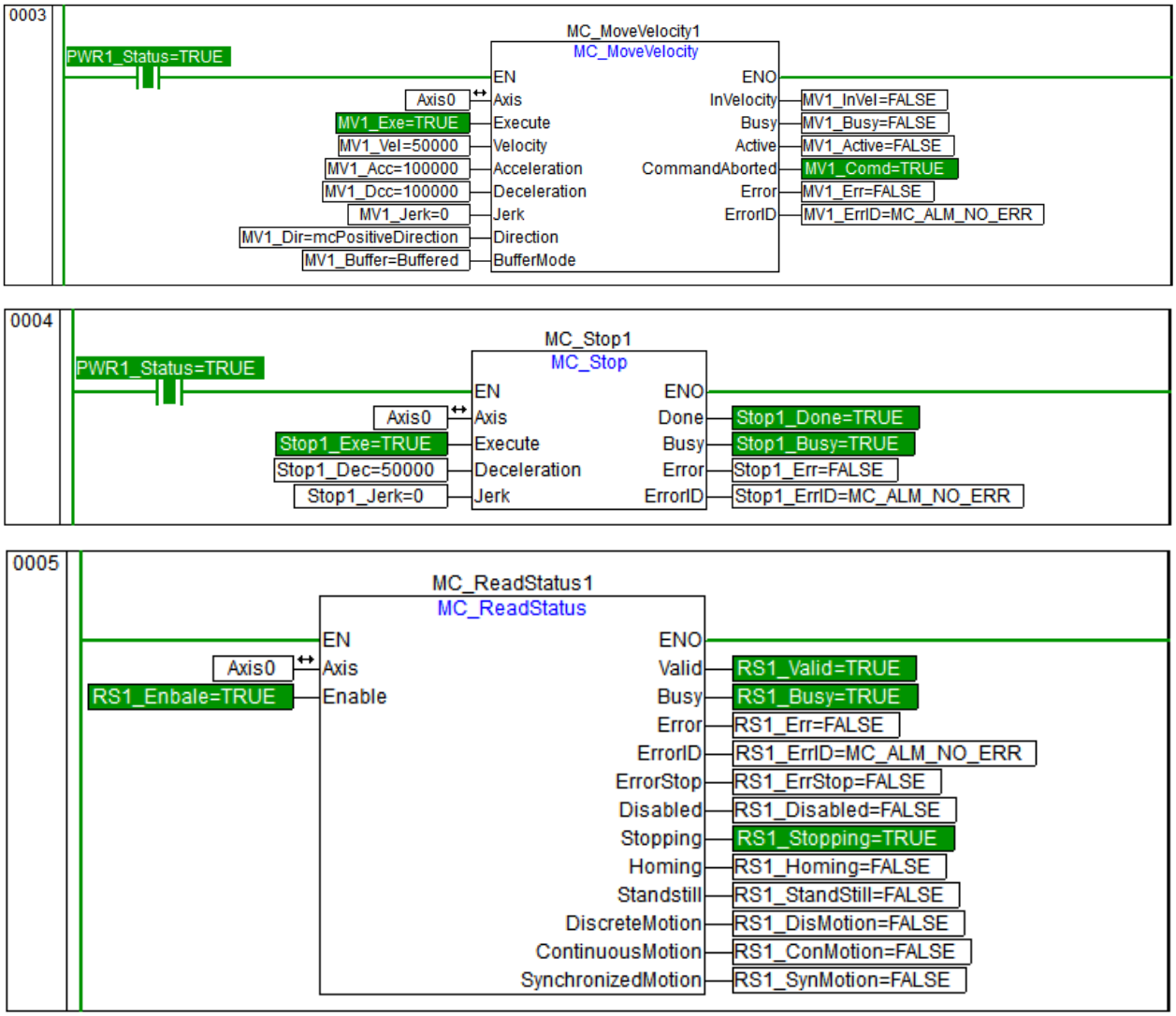
In section 0003, after the axis is enabled successfully, call MC_MoveVelocity to control the axis motion.



(4) Axis delivery deceleration stop command

In section 0004, MC_Stop is called to interrupt the MC_MoveLocity motion and the control axis decelerates to stop. CommandAborted pin of MC_MoveLocity becomes TRUE.

When the Execute pin of MC_Stop is held high, the status of the axis read in section 0005 remains Stopping.



■ ST language

Use ST language to set up the program, input parameters Deceleration 10000, Jerk 0, trapezoidal acceleration and deceleration.

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
Axis0	AXIS_REF	-	Axis variable for axis 0
Power1_Status	BOOL	FALSE	Variable for which the axis enable is successful. This variable becomes TRUE when the axis is enabled successfully.
MV1_Active	BOOL	FALSE	Variable when the axis speed control command acts. This variable becomes TRUE when the axis is controlling motion.
Stop1_Exe	BOOL	FALSE	Trigger variable for the rising edge of an axis stop command.
Stop1_Done	BOOL	FALSE	Variable at which the axis was successfully stopped. Done = TRUE, axis stop is finished.
Switch1	BOOL	FALSE	Axis ready state variable. After axis initialization, Switch to TRUE and the axis is ready.
InitFlag	BOOL	FALSE	If this variable is TRUE, the initialization parameter setting is completed.

(1) Axis initialization

Initial configuration of Axis0 with axis type set to 50:EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to its command Description section.

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

(2) Stop command input parameter initialization.

```
IF FALSE=InitFlag THEN
Stop1_Dec:=10000; (* Set the deceleration parameter of MC_Stop command *)
Stop1_Jerk:= 0; (*set Jerk parameter of MC_Stop command *)
InitFlag:=TRUE;
END_IF
```

(3) Axis enable

After the axis initialization is completed and the axis ready status Switch1 is TRUE, call MC_Power to complete the axis enable.

```
IF Switch1 THEN
 MC_Power1(
 Enable := PWR1_Enable,
 Axis := Axis0,
 Status=> PWR1_Status,
 Busy=> PWR1_Busy,
 Error=> PWR1_Err,
 ErrorID=> PWR1_ErrID);
END_IF
```

(4) Delivery motion command

Call MC_MoveVelocity to deliver motion command to Axis0.

```
IF  PWR1_Status THEN
   MC_MoveVelocity1(
```

```
    Execute := MV1_ Exe,
    Velocity := MV1_ Vel,
    Acceleration := MV1_ Acc,
    Deceleration := MV1_ Dcc,
    Jerk := MV1_ Jerk,
    Direction := MV1_ Dir,
    BufferMode := MV1_ Buffer,
    Axis := Axis0,
    InVelocity=> MV1_ InVel,
    Busy=> MV1_ Busy,
    Active=> MV1_ Active,
    CommandAborted=> MV1_ Comd,
    Error=> MV1_ Err,
    ErrorID=> MV1_ Error);
END_IF
```

(5) Run the MC_Stop command

Call the MC_Stop command to control the axis to decelerate and stop MC_MoveVelocity movement.

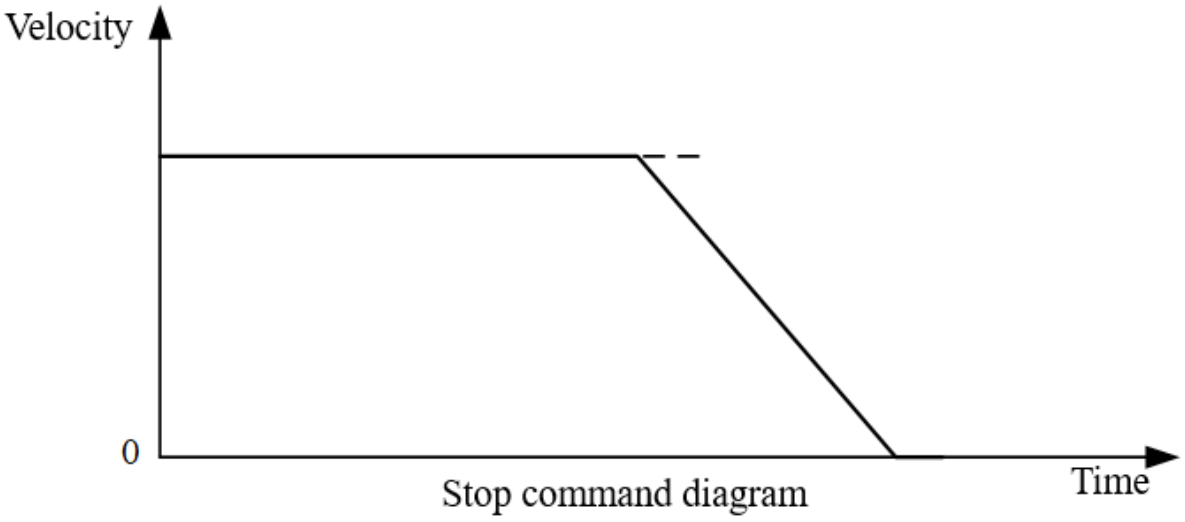
```
IF  PWR1_Status THEN
    MC_Stop1(
    Execute := Stop1_Exe,
    Deceleration := Stop1_Dec,
    Jerk := Stop1_Jerk,
    Axis := Axis0,
    Done=> Stop1_Done,
    Busy=> Stop1_Busy,
    Error=> Stop1_Err,
    ErrorID=> Stop1_ErrID);
END_IF
```

(6) Run MC_ReadStatus

```
Call MC_ReadStatus to view the axis status.
MC_ReadStatus1(
Enable := RS1_Enbale,
Axis := Axis0,
Valid=> RS1_Valid,
Busy=> RS1_Busy,
Error=> RS1_Err,
ErrorID=> RS1_ErrID,
ErrorStop=> RS1_ErrStop,
Disabled=> RS1_Disabled,
Stopping=> RS1_Stopping,
Homing=> RS1_Homing,
Standstill=> RS1_StandStill,
DiscreteMotion=> RS1_DisMotion,
```

```
ContinuousMotion=> RS1_ConMotion,
SynchronizedMotion=> RS1_SynMotion);
```

The schematic diagram of MC_MoveVelocity motion command deceleration stop is as follows:



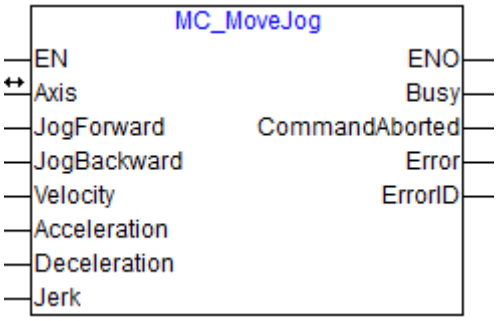
Stop command diagram

5.5.4 Precautions

- The MC_Stop command is successfully executed. If the input parameter Execute keeps high level, the axis remains in Stopping status and motion command delivery is prohibited.
- When the input parameter Execute is at low level, the stop function will be released and the axis will switch to StandStill status. Motion commands can be delivered normally.

5.6 MC_MoveJog (JOG)

Single-axis inching command in positive/negative direction



5.6.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	JogForward	Positive enable	No	-	TRUE/FALSE	BOOL
	JogBackward	Negative enable	No	-	TRUE/FALSE	BOOL
	Velocity	Target speed	No	-	Non-negative	LREAL
	Acceleration	Acceleration	No	-	Positive value	LREAL
	Deceleration	Deceleration	No	-	Positive value	LREAL
	Jerk	0: Trapezoidal acceleration and deceleration Positive: S-shaped acceleration and deceleration acceleration	YES	0	0/positive	LREAL
OUTPUT	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.6.2 Functions

This command can carry out positive and negative inching motion according to the set speed, acceleration and deceleration, jerk and other parameters. The detailed Description of this command is as follows:

(1) High level triggering of this command is valid, and the axis moves according to the set parameters. During execution, the modified latched parameters are invalid until high level takes effect after being triggered again. When JogForward (positive enable) is set as TRUE, the axis moves in positive direction; when JogBackward (negative enable) is set as TRUE, the axis moves in negative direction.

(2) When JogForward and JogBackward are both TRUE, the axis moves in positive direction.

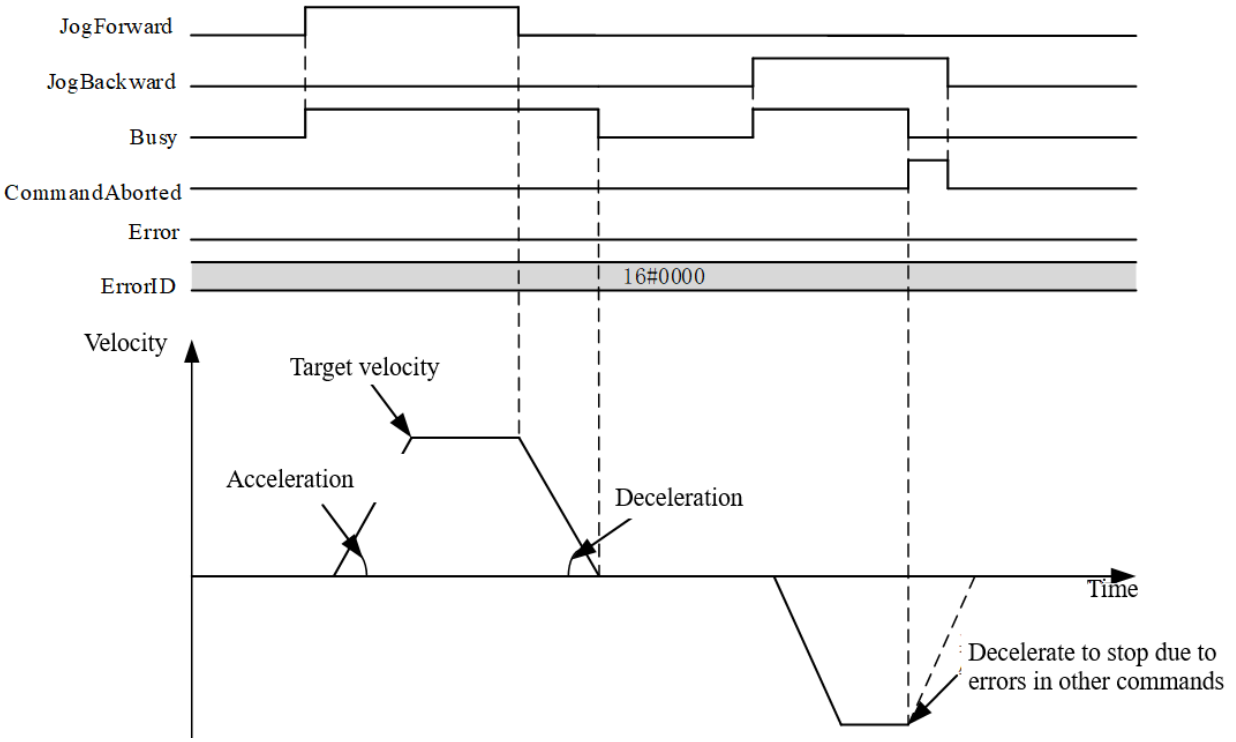
(3) When JogForward is changed from TRUE to FALSE, the axis decelerates and stops according to the set deceleration. During deceleration, since the axis does not stop, Busy (busy flag) will not change to FALSE, which is also the case in negative direction.

- Restart and multiple start motion commands
  - When restarting in the same direction, set JogForward or JogBackward to FALSE and decelerate. If JogForward or JogBackward is set to TRUE again, the axis will accelerate during deceleration and move with new motion parameters again.
  - When restarting the command in different directions, if JogForward is set to TRUE and JogBackward is set to TRUE when moving in the positive direction, the direction will be reversed at this time for reverse movement. The function block now moves with the input variable when JogBackward (Negative Enable) is set to TRUE. Similarly, when JogBackward is set to TRUE, the same motion will occur if JogForward is set to TRUE during negative rotation.
  - Description of JogForward (positive enable) and JogBackward (negative enable) pins: When JogForward is set to TRUE and then JogBackward is set to TRUE for negative rotation, JogBackward is set to FALSE. At this time, the axis stops decelerating and does not move forward just because JogForward is TRUE. When positive rotation is required, re-trigger the high level.
- Function block timing diagram

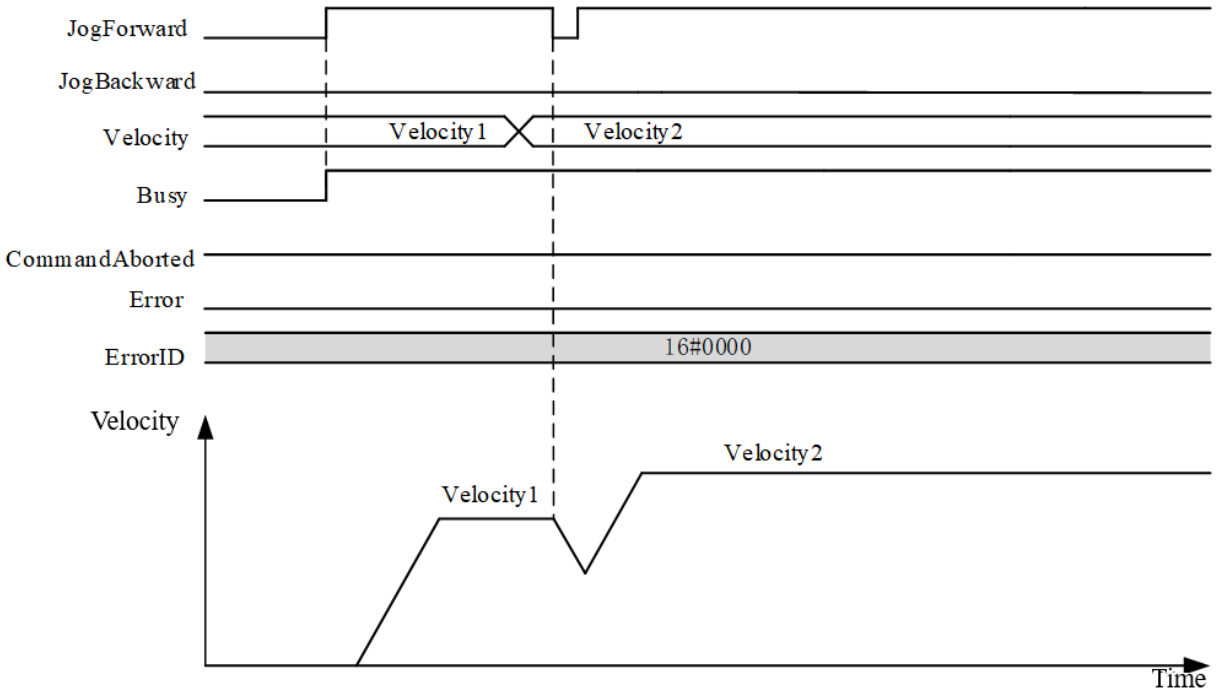
  Busy flag becomes TRUE when the function block starts JogForward or JogBackward is TRUE.

  At the same time when the falling edge of JogForward or JogBackward starts decelerating and stops the axis, Busy becomes FALSE. When this command is stopped by other commands, CommandAborted becomes TRUE and Busy becomes FALSE.

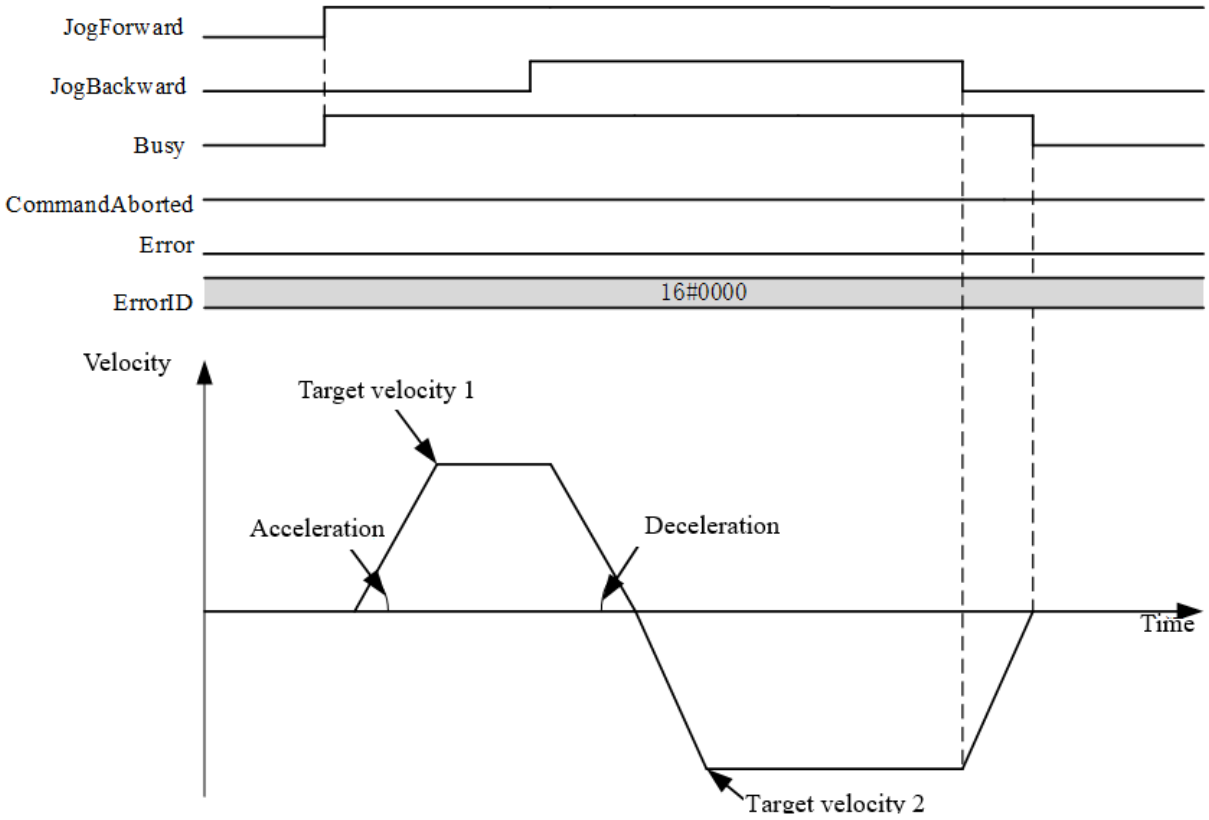
(1) Normal pin timing



(2) Timing diagram of restart in the same direction



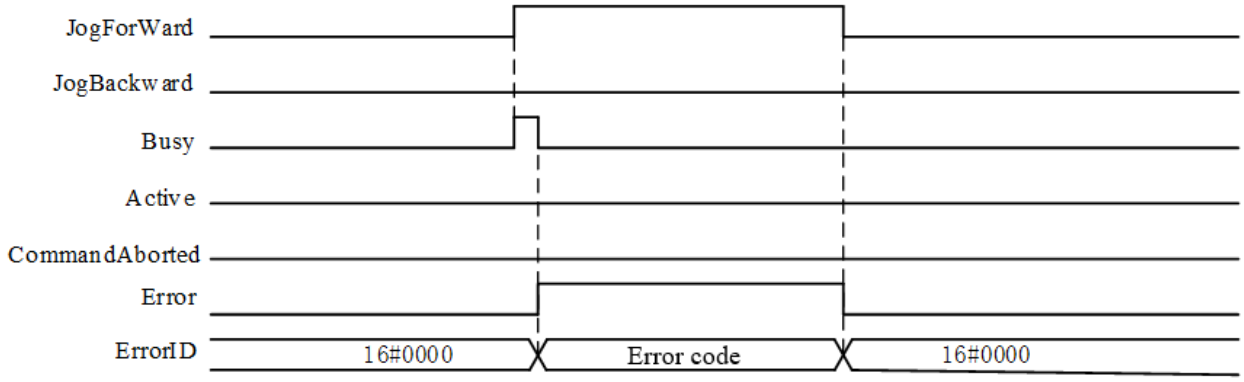
(3) Timing diagram of restart in different directions



(4) Error timing diagram

When an exception occurs during command execution, Error becomes TRUE and the axis stops moving.

The value of ErrorID can be viewed to understand the specific cause of the error.



5.6.3 Examples

Set up MC_MoveJog sample program to run simulated velocity motion command.

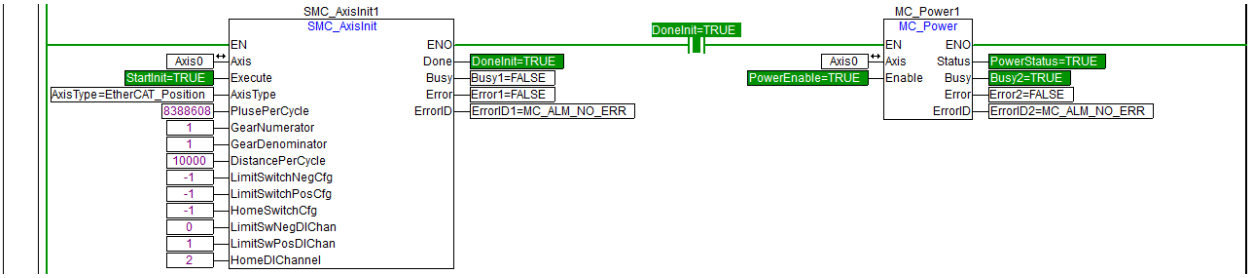
Definition of Primary Variable

Name	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	Change to TRUE at the start of initialization
DoneInit	BOOL	FALSE	Change to TRUE upon completion of initialization
PowerEnable	BOOL	FALSE	Change to TRUE at the beginning of enabling
PowerStatus	BOOL	FALSE	Change to TRUE when enabled
FwMoveJog	BOOL	FALSE	TRUE at start forward command
BwMoveJog	BOOL	FALSE	TRUE at start of negative command
BusyMoveJog	BOOL	FALSE	Commanded delivery complete becomes TRUE
ComMoveJog	BOOL	FALSE	Become TRUE when the command is interrupted
ErrorMoveJog	BOOL	FALSE	Change to TRUE in case of command error
StartReset	BOOL	FALSE	Change to TRUE at the start of reset

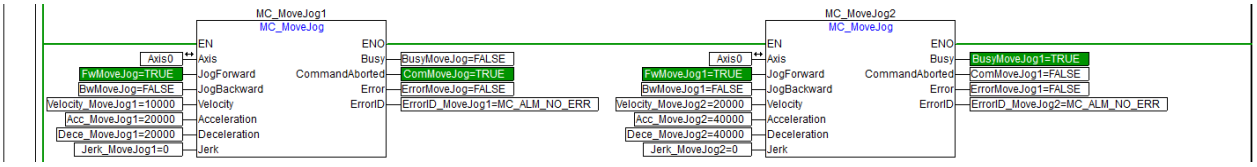
- LD program implementation

Use the ladder diagram to establish a program. The input parameters are Velocity 10000, Acceleration and Deceleration 20000, Jerk 0, and the positive rising edge triggers the control axis for inching motion. The example program is as follows:

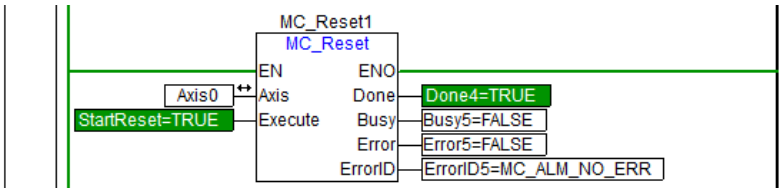
(1) Set the axis type to EtherCAT_Position.



(2) After the axis is enabled, you can enter specified motion parameters in the input variables of the function block to send velocity motion commands and repeat commands. It can be seen that when repeated sending occurs, the next function block will interrupt the previous one.



(3) When there is an error in the axis, the MC_Reset function block can be called for reset processing.



■ ST language program

(1) Initialize and enable the axis

```
(The * axis is initialized*)
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
```

```
ErrorID=>ErrorID1);
(*Enable the axis*)
IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

(2) High level triggers to start positive/negative inching movement, and the control axis moves.

```
(*Start speed control command*)
MC_MoveJog1(
JogForward := FwMoveJog,
JogBackward := BwMoveJog,
Velocity := 10000,
Acceleration := 20000,
Deceleration := 20000,
Jerk := 0,
Axis := Axis0,
Busy=>BusyMoveJog,
CommandAborted=>ComMoveJog,
Error=>ErrorMoveJog,
ErrorID=>ErrorID_MoveJog1);
```

(3) Repeatedly trigger the high level, and control the axis to move.

```
(* Repeated start speed control command *)
MC_MoveJog2(
JogForward := FwMoveJog1,
JogBackward := BwMoveJog1,
Velocity := 20000,
Acceleration := 40000,
Deceleration := 40000,
Jerk := 0,
Axis := Axis0,
Busy=>BusyMoveJog1,
CommandAborted=>ComMoveJog1,
Error=>ErrorMoveJog1,
ErrorID=>ErrorID_MoveJog2);
```

(4) Reset processing can be carried out in case of error.

```
MC_Reset1(
Execute := Exe5,
Axis := Axis0,
Done=>Done4,
```

```
Busy=>Busy5,
Error=>Error5,
ErrorID=>ErrorID5);
```

5.6.4 Precautions

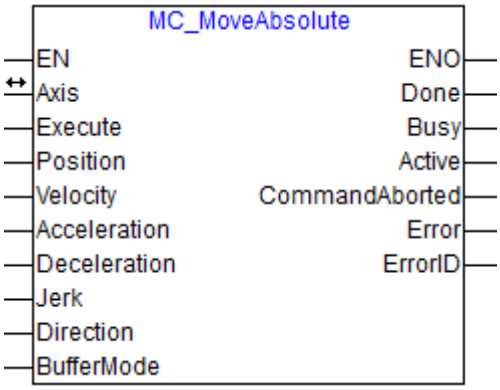
- It is not allowed to input 0 value for acceleration and deceleration in the input variables.
- When the positive enabling and negative enabling are both TRUE, the positive enabling takes precedence, so it is necessary to determine the motion direction before use.

5.6.5 Reference

- If an error occurs, refer to the appendix Function Block Error Codes.
- For information about initialization and enabling, please refer to the section of Initialization and Enabling.

5.7 MC_MoveAbsolute (Absolute Positioning)

Single-axis absolute positioning command. The control axis moves to the specified absolute position.



5.7.1 Parameter

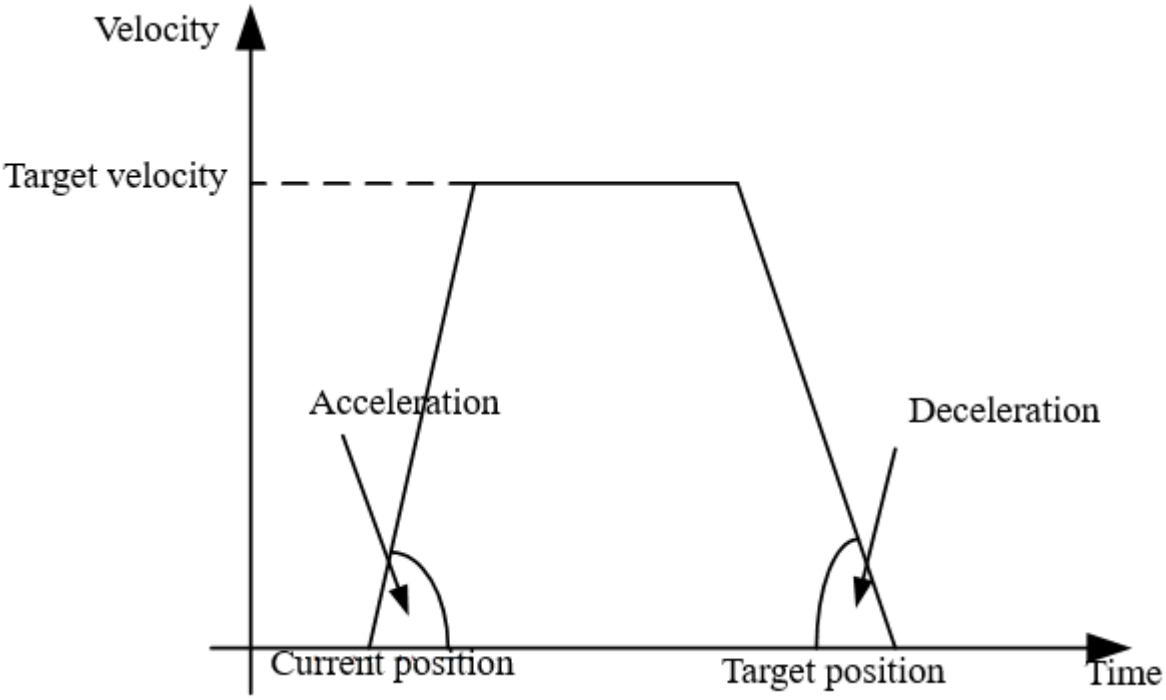
Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	Position	Target position (absolute)	No	-	Positive/negative/0	LREAL
	Velocity	Target speed	No	-	Non-negative	LREAL
	Acceleration	Acceleration	No	-	Positive value	LREAL
	Deceleration	Deceleration	No	-	Positive value	LREAL

	Jerk	0: Trapezoidal acceleration and deceleration Positive: S-shaped acceleration and deceleration acceleration	YES	0	0/positive	LREAL
	Direction	Direction of motion, valid only in cycle mode. 0: Forward 1: Shortest distance 2: Negative 3: Current motion direction 4: The direction is not specified, and it is automatically calculated within the current cyclic stroke range	YES	4	0: mcPositiveDirection 1: mcShortestWay 2: mcNegativeDirection 3: mcCurrentDirection 4: mcNoDirection	MC_DIRECTION
	BufferMode	Buffer mode 0: Interrupt 1: buffer 2: low-speed fusion between adjacent commands 3: Speed fusion of the previous command 4: Speed fusion of the following command 5: High-speed fusion between adjacent commands	YES	0	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	MC_BUFFER_MODE
OUTPUT	Done	Positioning movement completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Active	In control, TRUE when the command is executed	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.7.2 Functions

This command can carry out absolute position movement according to the set speed, acceleration and deceleration, jerk and other parameters. The detailed description of this command is as follows:

- The rising edge triggering of this command is valid. At the rising edge of the Execute input, specify Position (absolute position), Velocity (target speed), Acceleration, Deceleration and Jerk in the input variables to start absolute positioning action. During execution, The modification of latched parameters is invalid until the rising edge triggers this command again. The action example of absolute positioning is shown in the following figure.

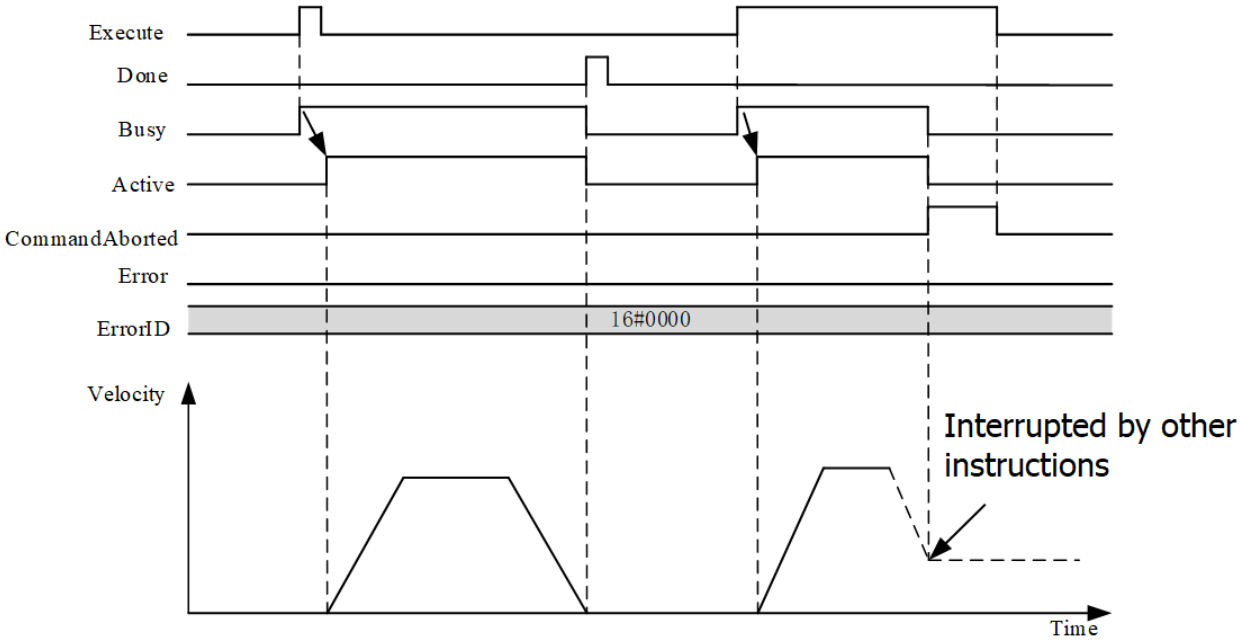


- Direction (direction selection) introduction
  - When Direction is mcPositiveDirection, the direction of motion is positive.
  - When the Direction is mcShortestWay, the shortest distance will be calculated according to the forward and backward movement distances calculated by the algorithm. If they are the same, the direction is positive.
  - When Direction is mcNegativeDirection, the direction of motion is negative.
  - When Direction is set to mcCurrentDirection, the command will act along the command direction of the previous action. If the direction of the previous action is positive, the current motion direction is positive; otherwise, it is negative.
  - When Direction is set to mcNoDirection, the command determines the motion direction based on the values of current position and target position. If the current position is smaller than the target position, the movement direction is positive; if the current position is larger than the target position, the movement direction is positive.
- BufferMode
  - Absolute positioning commands support BufferMode (buffer mode selection), BufferMode supports 0: Aborting, 1: Buffered, 2: BlendingLow, 3: BlendingPrevious, 4: BlendingNext, and 5: BlendingHigh.
  - When sending this command, if BufferMode is 0: Aborting, it will interrupt the currently executing command. If BufferMode is 1: Buffered, from the cycle when the currently executing command ends normally, the buffered command starts automatically. If BufferMode is 2: BlendingLow, it blends at a low speed between adjacent commands. If BufferMode is 3: BlendingPrevious, it blends at the speed of the previous command. If BufferMode is 4: BlendingNext, it blends at the

speed of the next command. If BufferMode is 5: BlendingHigh, it blends at a high speed between adjacent commands.

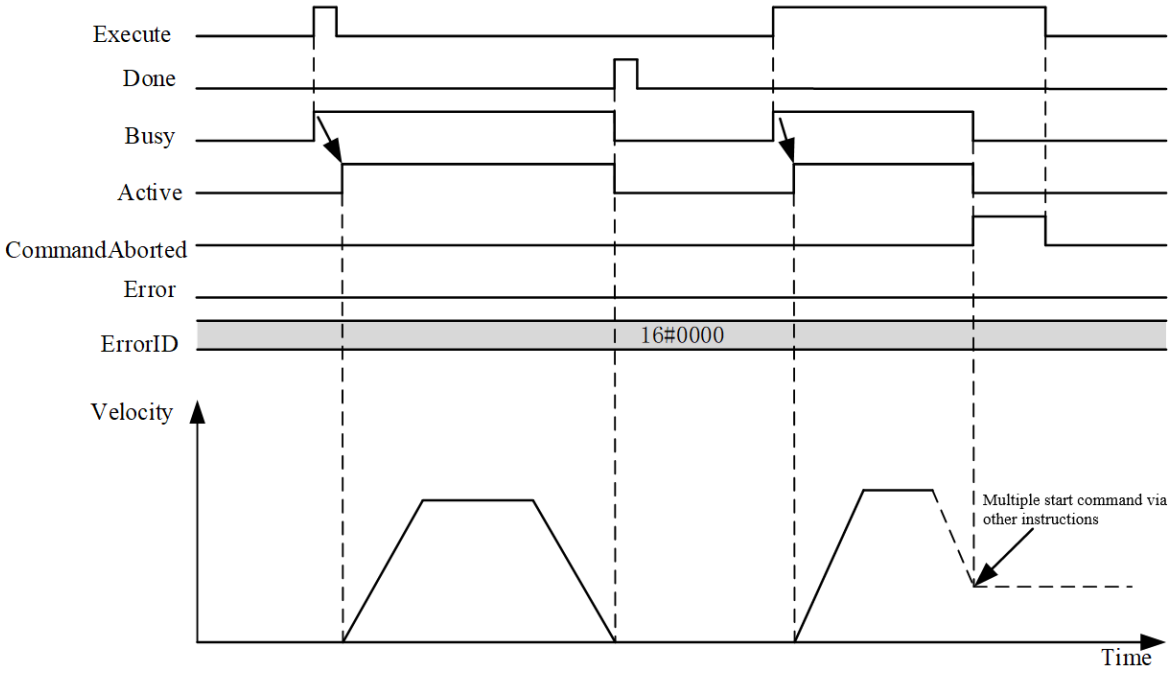
- For details on Buffer Mode, please refer to the Description of Buffer Mode in the appendix section.

- Repeated startup motion command
  - The connection mode of this instruction with the previous one when this instruction is repeatedly activated is specified by BufferMode (Buffer Mode Selection), which can be chosen to interrupt, buffer, or merge.
  - When BufferMode is set to Aborting, the action of this command can be changed by setting Execute as TRUE restart command again. The commanded Position (Target Position), Velocity (Target Speed), Acceleration (Acceleration), Deceleration (Deceleration) and Jerk (Jerk) can be changed. Example actions are shown in the figure below.



- Function block timing diagram

(1) The correct timing sequence for command execution is as follows:



(2) Timing diagram in case of abnormality

When an exception occurs during the execution of this command, Error becomes FALSE and the axis stops.

The cause of the exception can be understood by viewing the output value of ErrorID.

5.7.3 Examples

Create an MC_MoveAbsolute example program to run absolute positioning commands.

Definition of Primary Variable

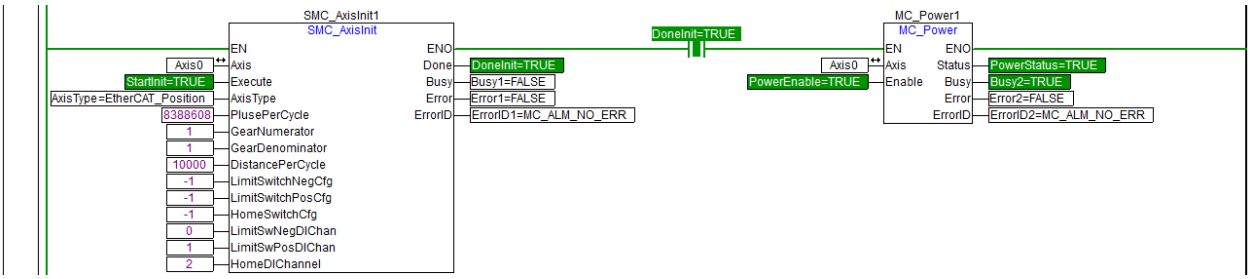
Variable Name	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	Change to TRUE at the start of initialization
DoneInit	BOOL	FALSE	Change to TRUE upon completion of initialization
PowerEnable	BOOL	FALSE	Change to TRUE at the beginning of enabling
PowerStatus	BOOL	FALSE	Change to TRUE when enabled
StartMoveAbs	BOOL	FALSE	TRUE when absolute positioning command is started
BusyMoveAbs	BOOL	FALSE	Absolute positioning command delivery completed changes to TRUE
ActiveMoveAbs	BOOL	FALSE	Change to TRUE when the command is running
DoneMoveAbs	BOOL	FALSE	Change to TRUE when the absolute positioning command is completed
ComMoveAbs	BOOL	FALSE	Become TRUE when the command is interrupted
ErrorMoveAbs	BOOL	FALSE	Change to TRUE in case of command error
StartReset	BOOL	FALSE	Change to TRUE at the start of reset

■ LD program implementation

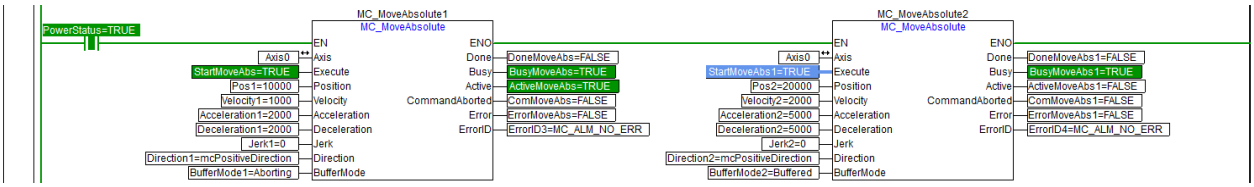
Use the ladder diagram to establish a program, input the parameters Position 10000, Velocity 1000, Acceleration and Deceleration 2000, Jerk 0, Direction for positive running, BufferMode parameter

Aborting, and the rising edge triggers the control axis to move. The example procedure is as follows:

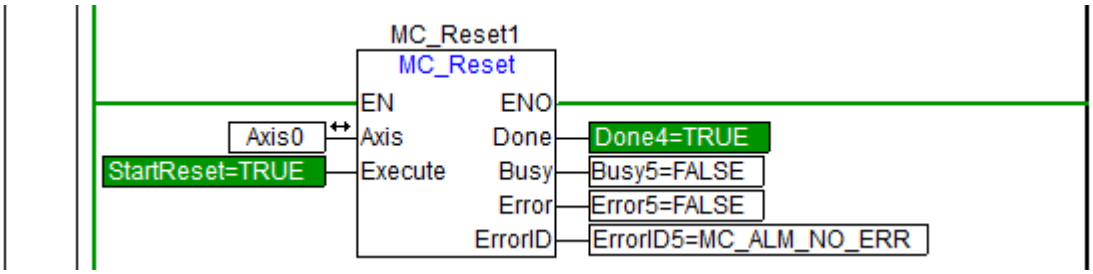
(1) First, initialize the axis and set its type and various parameters. After the initialization is completed, enable the axis. When it enters the ON state of servo enable, motion control can be realized for the axis.



(2) After the axis is enabled, specified motion parameters can be entered in the input variables of the functional block to send absolute positioning motion commands and repeat commands.



(3) When there is an error in the axis, the MC_Reset function block can be called for reset processing.



■ ST language program

(1) Initialize and enable the axis

```
(The * axis is initialized*)
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
```

```
HomeDIChannel := 2,
Axis := Axis0,
Done=>Done1,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
(*Enable the axis*)
IF DoneInit THEN
  MC_Power1(
  Enable := PowerEnable,
  Axis := Axis0,
  Status=> PowerStatus,
  Busy=>Busy2,
  Error=>Error2,
  ErrorID=>ErrorID2);
END_IF
```

(2) The rising edge triggers the MC_MoveAbsolute command or this command is executed repeatedly to control axis motion.

```
IF PowerStatus THEN
(*Start absolute positioning command*)
MC_MoveAbsolute1(
Execute := StartMoveAbs,
Position := 10000,
Velocity := 1000,
Acceleration := 2000,
Deceleration := 2000,
Jerk := 0,
Direction := 4,
BufferMode := 0,
Axis := Axis0,
Done=> DoneMoveAbs,
Busy=> BusyMoveAbs,
Active=> ActiveMoveAbs,
CommandAborted=> ComMoveAbs,
Error=> ErrorMoveAbs,
ErrorID=>ErrorID3);
(* Repeated start of absolute positioning command *)
MC_MoveAbsolute2(
Execute := StartMoveAbs1,
Position := 50000,
Velocity := 2000,
Acceleration := 5000,
Deceleration := 5000,
Jerk := 0,
Direction := 4,
```

```
BufferMode :=0,
Axis := Axis0,
Done=> DoneMoveAbs1,
Busy=> BusyMoveAbs1,
Active=> ActiveMoveAbs1,
CommandAborted=> ComMoveAbs1,
Error=> ErrorMoveAbs1,
ErrorID=>ErrorID4);
END_IF
```

(3) In case of an error, call the MC_Reset function block to reset the axis.

```
IF ErrorMoveAbs THEN
(*Reset processing can be performed in case of error *)
  MC_Reset1(
  Execute := StartReset,
  Axis := Axis0,
  Done=>Done4,
  Busy=>Busy5,
  Error=>Error5,
  ErrorID=>ErrorID5);
END_IF
```

5.7.4 Precautions

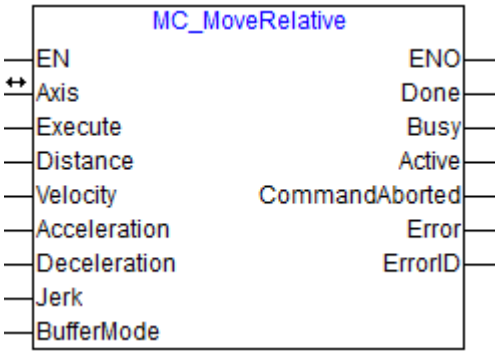
- When an absolute positioning command is used, the target position shall be entered between the positive limit and negative limit.
- When Direction is set to mcCurrentDirection, the command will act in the same direction as that of the previous action. Therefore, the motion direction should be confirmed according to the previous command before use.

5.7.5 Reference

- In case of any fault, please refer to the Description of error codes for function in the appendix.
- For information about initialization and enabling, please refer to the section initialization and enabling.
- BufferMode (buffer mode selection)Refer to the appendix Buffer Mode Description.

5.8 MC_MoveRelative (Relative Positioning)

Single-axis relative positioning command: The control axis moves to the specified position incrementally.



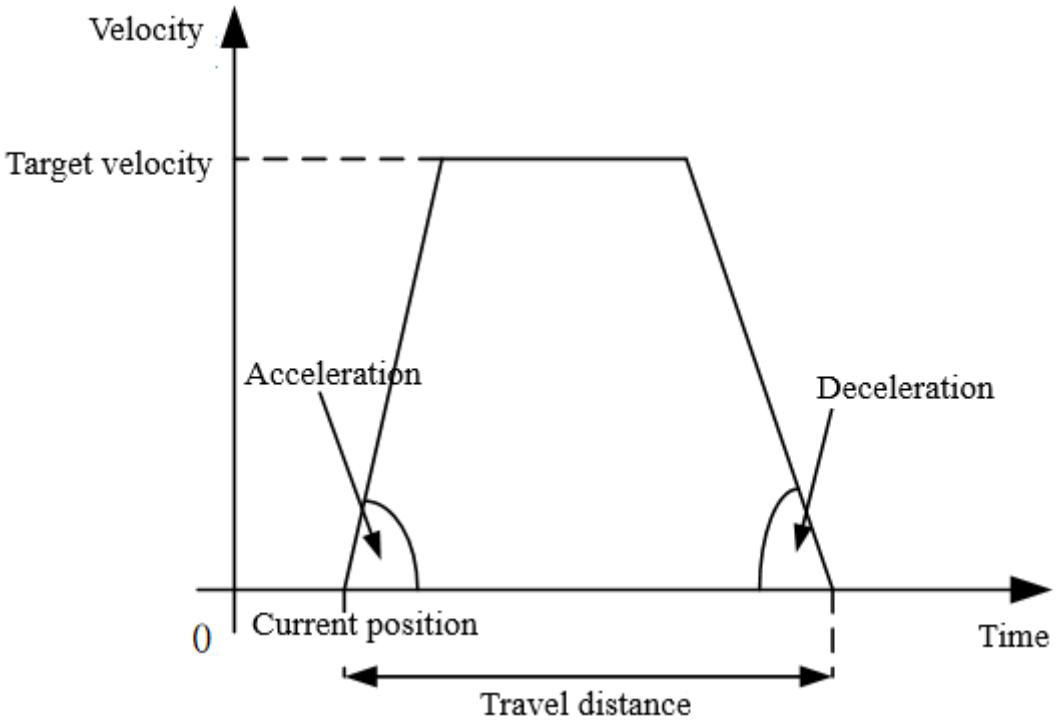
5.8.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	Distance	Target position (increments)	No	-	Non-zero value	LREAL
	Velocity	Target speed	No	-	Non-negative	LREAL
	Acceleration	Acceleration	No	-	Positive value	LREAL
	Deceleration	Deceleration	No	-	Positive value	LREAL
	Jerk	0: Trapezoidal acceleration and deceleration Positive: S-shaped acceleration and deceleration acceleration	YES	0	0/positive	LREAL
	BufferMode	Buffer mode 0: Interrupt 1: Buffer 2: Low-speed fusion between adjacent commands 3: Speed fusion of the previous command 4: Speed fusion of the following command 5: High-speed fusion between adjacent commands	YES	0	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	MC_BUFFER_MODE
OUTPUT	Done	Positioning movement completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Active	In control, TRUE when the command is executed	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.8.2 Functions

This command can carry out relative positioning motion according to the set speed, acceleration and deceleration, jerk and other parameters. The control axis moves to the specified position in an incremental manner. Details of this command are as follows:

- The rising edge triggering of this command is valid. At the rising edge of the Execute input, specify Distance, Velocity, Acceleration, Deceleration and Jerk in the input variables to start absolute positioning action. During execution, The modification of latched parameters is invalid until the rising edge triggers this command again. The action example of absolute positioning is shown in the following figure.



- In this command, when Distance (movement distance) is set to a positive value, the command will rotate in the positive direction; when it is set to a negative value, the command will rotate in the negative direction.
- BufferMode

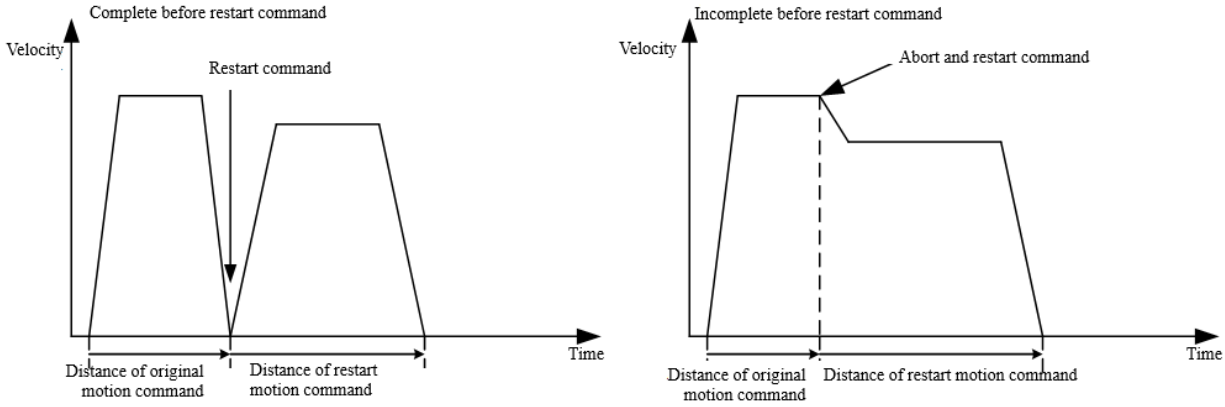
  The absolute positioning command supports BufferMode (buffer mode selection). BufferMode supports 0: Aborting, 1: Buffered, 2: BlendingLow, 3: BlendingPrevious, 4: BlendingNext, and 5: BlendingHigh.

  When sending this command, if BufferMode is 0: Aborting, it will interrupt the currently executing command. If BufferMode is 1: Buffered, from the cycle when the current executing command normally ends, the buffered command will automatically start. If BufferMode is 2: BlendingLow, it will blend at a low speed between adjacent commands. If BufferMode is 3: BlendingPrevious, it will blend at the speed of the previous command. If BufferMode is 4: BlendingNext, it will blend at the speed of the

next command. If BufferMode is 5: BlendingHigh, it will blend at a high speed between adjacent commands.

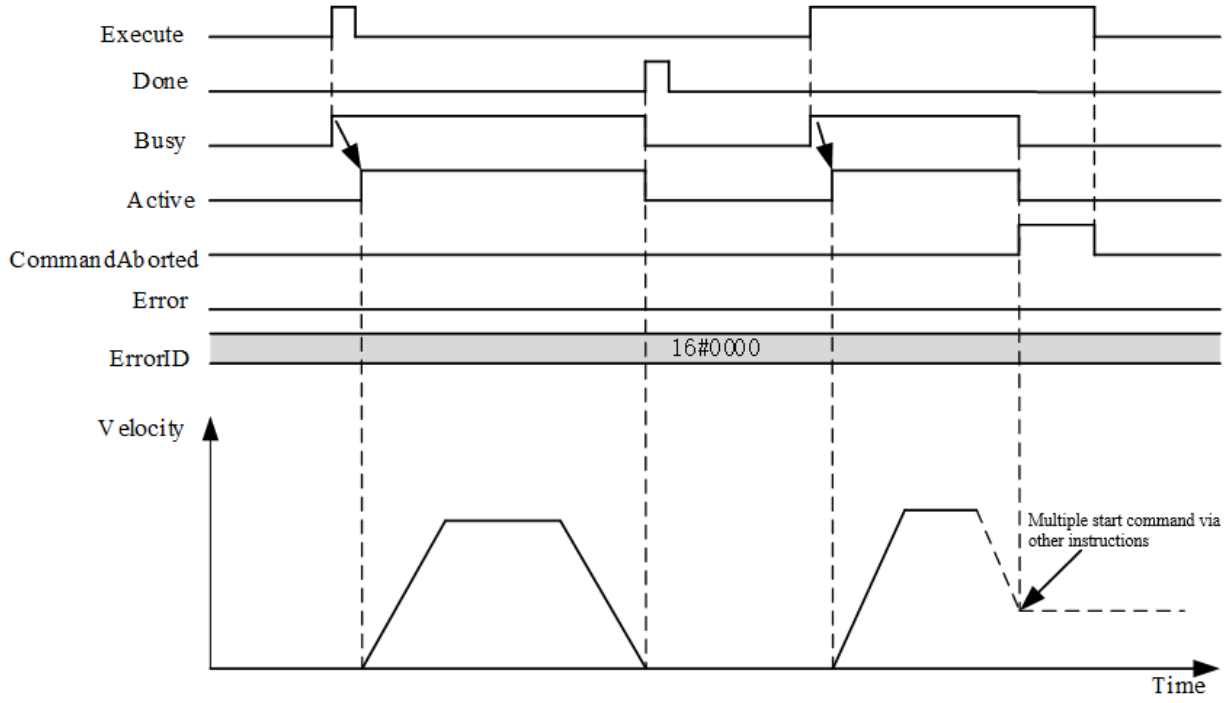
For detailed information on buffer modes, please refer to the buffer mode description in the appendix section.

- Repeated start motion command
  - The connection mode between this command and the previous command during repeated startup is specified by BufferMode (buffer mode selection), with options of interruption, buffering and fusion.
  - When BufferMode is set to Aborting, the action of this command can be changed by setting Execute as TRUE restart command again. Distance (target position), Velocity (target speed), Acceleration, Deceleration and Jerk of the command can be changed. Example actions are shown in the figure below.



- Function block timing diagram

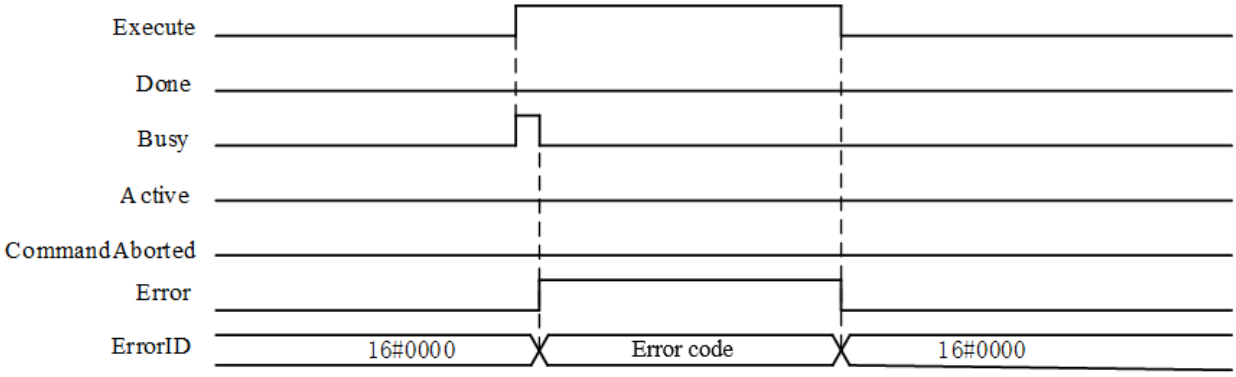
(1) The correct timing sequence for command execution is as follows:



(2) Timing diagram in case of abnormality

When an exception occurs during the execution of this command, Error turns FALSE and the axis stops.

The cause of the exception can be understood by viewing the output value of ErrorID.



5.8.3 Examples

Set up MC_MoveRelative sample program to run relative positioning.

Definition of Primary Variable

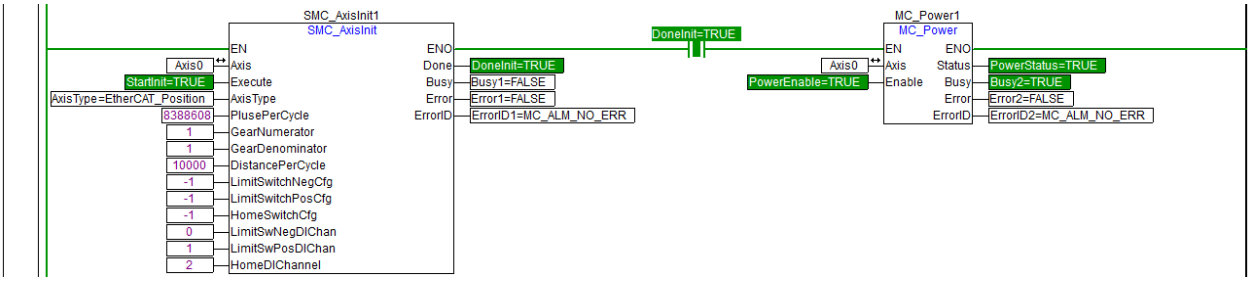
Variable Name	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	Change to TRUE at the start of initialization
DoneInit	BOOL	FALSE	Change to TRUE upon completion of initialization

PowerEnable	BOOL	FALSE	Change to TRUE at the beginning of enabling
PowerStatus	BOOL	FALSE	Change to TRUE when enabled
StartMoveRel	BOOL	FALSE	TRUE when relative positioning command is started
BusyMoveRel	BOOL	FALSE	Relative positioning command delivery completed changes to TRUE
ActiveMoveRel	BOOL	FALSE	Change to TRUE when the command is running
DoneMoveRel	BOOL	FALSE	Change to TRUE when the relative positioning command is completed
ComMoveRel	BOOL	FALSE	Change to TRUE when the command is interrupted
ErrorMoveRel	BOOL	FALSE	Change to TRUE in case of command error
StartReset	BOOL	FALSE	Change to TRUE at the start of reset

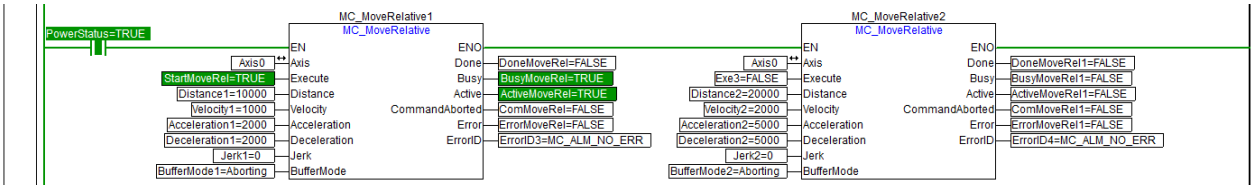
■ LD program implementation

Use the ladder diagram to establish a program, input parameters Distance 10000, Velocity 1000, Acceleration and Deceleration 2000, Jerk 0, Direction for forward running, BufferMode parameter Aborting, the rising edge triggers the control axis to move. The example procedure is as follows:

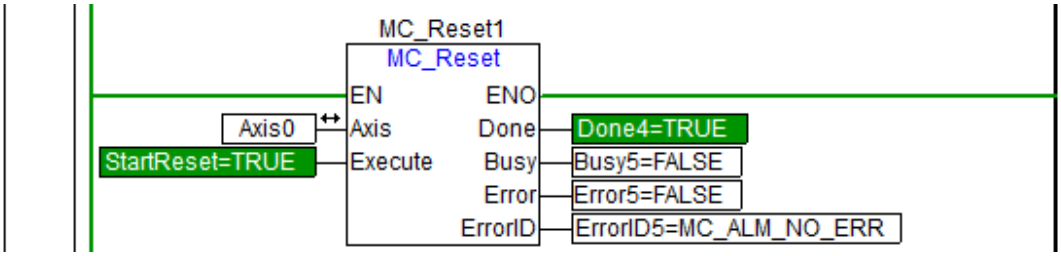
(1) First, initialize the axis and set its type and various parameters. After the initialization is completed, enable the axis. When it enters the ON state of servo enabling, motion control can be realized for the axis.



(2) After the axis is enabled, specified motion parameters can be entered in the input variables of the functional block to send relative motion commands and repeat commands.



(3) When there is an error in the axis, the MC_Reset function block can be called for reset processing.



■ ST language program

(1) Initialize and enable the axis

```
(The * axis is initialized*)
```

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
(*Enable the axis*)
IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

(2) Relative motion command triggered by rising edge

```
(* Start relative motion command *)
IF PowerStatus THEN
 MC_MoveRelative1(
 Execute := StartMoveRel,
 Distance := 10000,
 Velocity := 1000,
 Acceleration := 2000,
 Deceleration := 2000,
 Jerk := 0,
 BufferMode := 0,
 Axis := Axis0,
 Done=> DoneMoveRel,
 Busy=> BusyMoveRel,
 Active=> ActiveMoveRel,
 CommandAborted=> ComMoveRel,
 Error=> ErrorMoveRel,
```

```
ErrorID=>ErrorID3);
```

(3) Repeated start of relative motion command

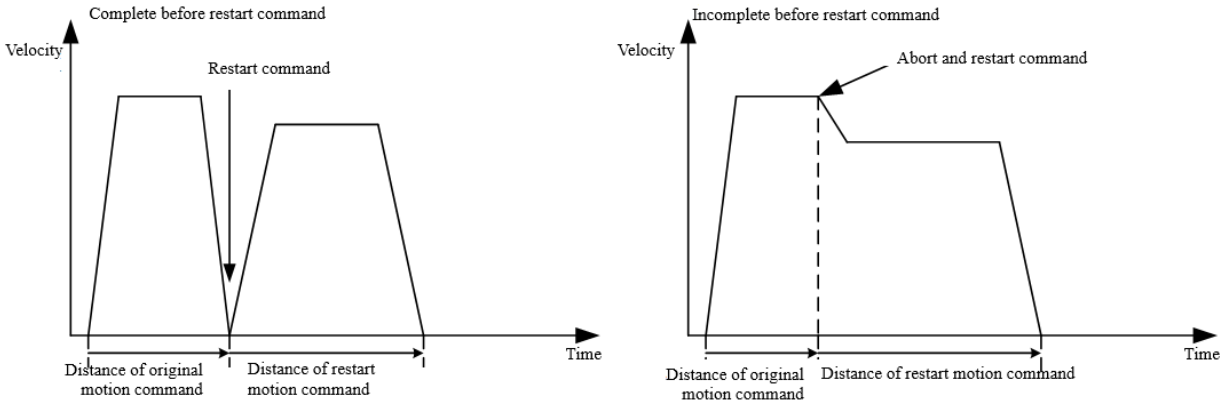
```
(* Repeated start motion command *)
 MC_MoveRelative2(
 Execute := StartMoveRel1,
 Distance := 20000,
 Velocity := 2000,
 Acceleration := 5000,
 Deceleration := 5000,
 Jerk := 0,
 BufferMode := 0,
 Axis := Axis0,
 Done=> DoneMoveRel1,
 Busy=> BusyMoveRel1,
 Active=> ActiveMoveRel1,
 CommandAborted=> ComMoveRel1,
 Error=> ErrorMoveRel1,
 ErrorID=>ErrorID4);
END_IF
```

(4) Reset in case of error

```
(*Reset processing can be performed in case of error *)
MC_Reset1(
Execute := StartReset,
Axis := Axis0,
Done=>Done4,
Busy=>Busy5,
Error=>Error5,
ErrorID=>ErrorID5);
```

5.8.4 Precautions

- When entering the parameter Distance (movement distance), it is not allowed to enter a value of 0, otherwise an error will be reported.
- If the positioning is completed before the functional block module is restarted, the position at the time of restarting shall be the relative position of the reference for positioning.
- If the positioning is not completed before the function block module is restarted, when the restart of the function block interrupts the restart, the position at the time of command execution will be taken as the reference.

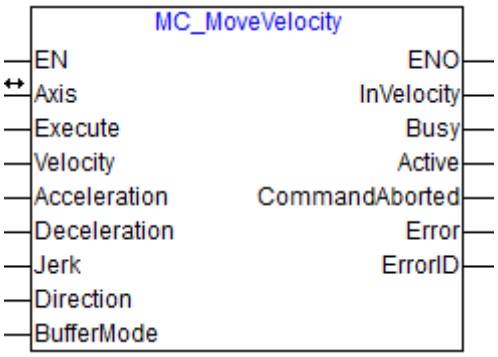


5.8.5 Reference

- If an error occurs, refer to the appendix Function Block Error Codes.
- Refer to the MC_MoveAbsolute function block for its use.
- BufferMode (buffer mode selection)Refer to the appendix Description of Buffer Mode.

5.9 MC_MoveVelocity (Velocity Control)

Single-axis speed control command, simulating speed control in position control mode.



5.9.1 Parameter

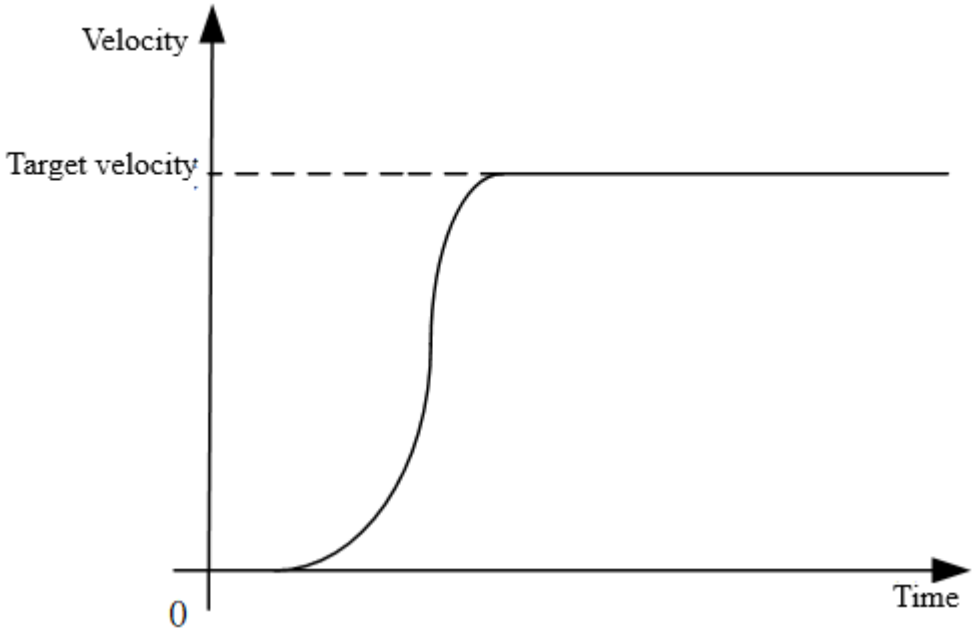
Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	Velocity	Target speed	No	-	Non-negative	LREAL
	Acceleration	Acceleration	No	-	Positive value	LREAL
	Deceleration	Deceleration	No	-	Positive value	LREAL
	Jerk	0: Trapezoidal acceleration and	YES	0	0/positive	LREAL

		deceleration Positive: S-shaped acceleration and deceleration acceleration				
	Direction	Direction of motion, valid only in cycle mode. 0: Forward 2: Negative 3: Current motion direction	YES	0	0: mcPositiveDirection 2: mcNegativeDirection 3: mcCurrentDirection	MC_DIRECTION
	BufferMode	Buffer mode 0: Interrupt 1: buffer 2: low-speed fusion between adjacent commands 3: Speed fusion of the previous command 4: Speed fusion of the following command 5: High-speed fusion between adjacent commands	YES	0	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	MC_BUFFER_MODE
OUTPUT	InVelocity	Target speed arrival	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Active	In control, TRUE when the command is executed	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.9.2 Functions

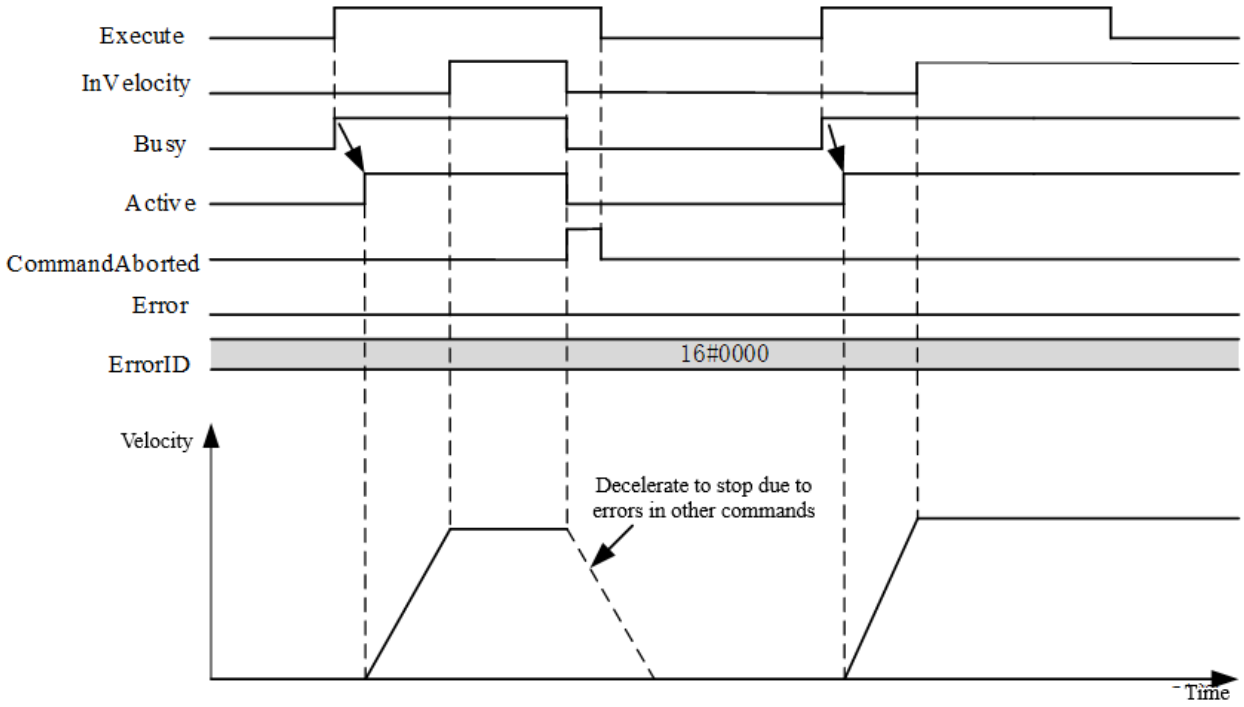
This command can simulate speed control motion according to the set parameters such as speed, acceleration and deceleration, jerk, etc. The detailed description of this command is as follows:

- The rising edge triggering of this command is valid. At the rising edge of the Execute (start) input, Velocity, Acceleration, Deceleration and Jerk are specified in the input variables to start the absolute positioning action. During execution, the modification of latched parameters is invalid until another rising edge triggers this command. Examples of command actions are as follows.



- Direction (direction selection) introduction
  - When Direction is mcPositiveDirection, the direction of motion is positive.
  - When Direction is mcNegativeDirection, the direction of motion is negative.
  - When Direction is set to mcCurrentDirection, the command will act along the command direction of the previous action. If the direction of the previous action is positive, the current motion direction is positive; otherwise, it is negative.
- Repeated startup motion command
  - Change the input variable in the positioning operation, set BufferMode to Aborting (interrupt), and then set Execute to TRUE again to change the action of this command.
  - When the BufferMode is set to Aborting for repeated startup commands, parameters such as Velocity (target speed), Acceleration, Deceleration and Direction of the command can be changed.
  - The action of this command during repeated startup is specified by BufferMode (buffer mode selection). Interruption, buffering and fusion can be selected for repeated startup of this command.
- InVelocity (reaching the target speed) is an output indicating that the same velocity is reached for starting this command and restarting motion command. Therefore, even if the speed is changed by using MC_SetOverride after InVelocity becomes TRUE, InVelocity will not become FALSE. When the speed is changed using MC_SetOverride before InVelocity (reach target speed) becomes TRUE, if the change target speed has been reached, InVelocity (reach target speed) turns TRUE.
- Function block timing diagram

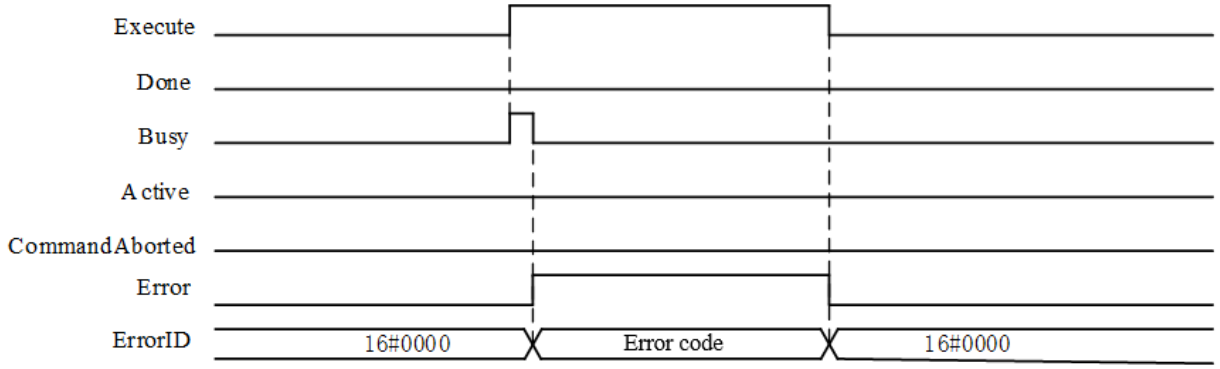
(1) The correct timing sequence for command execution is as follows:



(2) Error Timing diagram

When an error occurs during the execution of this command, Error becomes TRUE and the axis stops.

The output value of ErrorID can be viewed to understand the cause of the exception.



5.9.3 Examples

Set up MC_MoveVelocity example program to run simulated velocity motion command.

Definition of Primary Variable

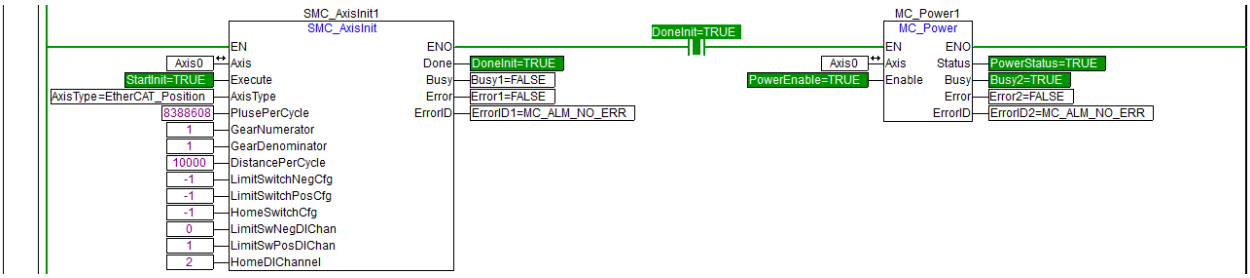
Variable Name	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	Change to TRUE at the start of initialization
DoneInit	BOOL	FALSE	Change to TRUE upon completion of initialization
PowerEnable	BOOL	FALSE	Change to TRUE at the beginning of enabling

PowerStatus	BOOL	FALSE	Change to TRUE when enabled
StartMoveVel	BOOL	FALSE	TRUE on start command
BusyMoveVel	BOOL	FALSE	Change to TRUE when delivery command is completed
ActiveMoveVel	BOOL	FALSE	Change to TRUE when the command is running
InVelMoveVel	BOOL	FALSE	Change TRUE when the commanded speed is reached
ComMoveVel	BOOL	FALSE	Change TRUE when the command is interrupted
ErrorMoveVel	BOOL	FALSE	Change to TRUE in case of command error
StartReset	BOOL	FALSE	Change to TRUE at the start of reset

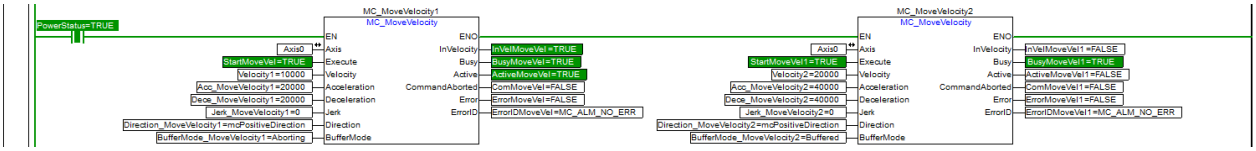
■ LD program implementation

A program is established using a ladder diagram. The input parameters are Velocity 10000, Acceleration and Deceleration 20000, Jerk 0, Direction for forward operation, BufferMode parameter Aborting, and the rising edge triggers the control axis to perform velocity motion. The example program is as follows:

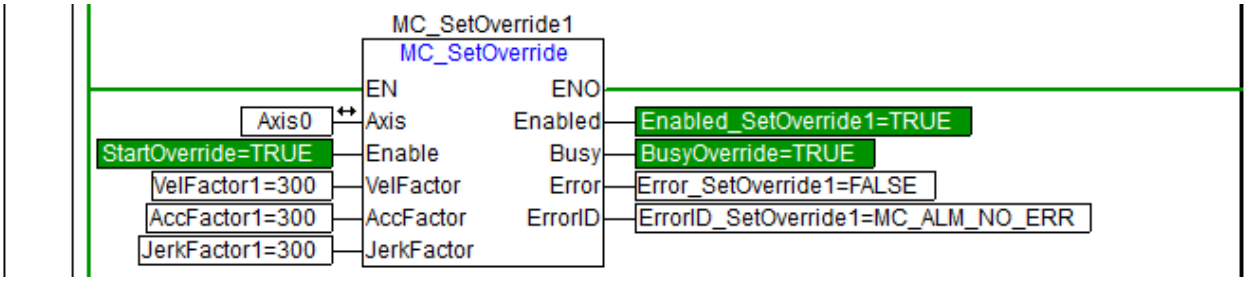
(1) First, initialize the axis and set its type and various parameters. After the initialization is completed, enable the axis. When it enters the ON state of servo enabling, motion control can be realized for the axis.



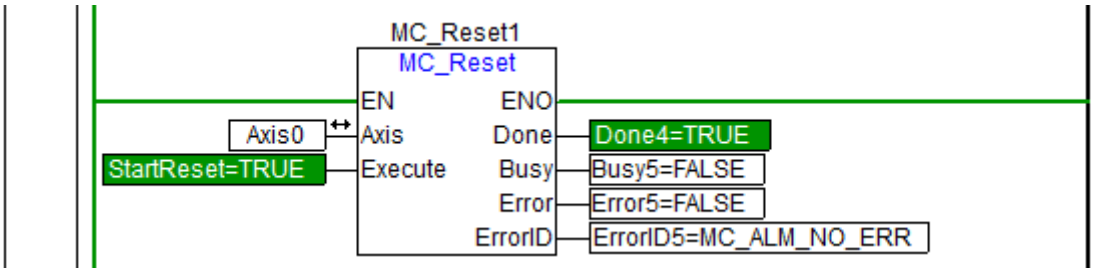
(2) After the axis is enabled, specified motion parameters can be entered in the input variables of the functional block to send velocity motion command and repeat command.



(3) Use the MC_SetOverride command to set override and check the state change of InVelocity pin.



(4) When an axis error occurs, the MC_Reset function block can be called for reset processing.



■ ST language program

(1) Initialize and enable the axis

```
(The * axis is initialized*)
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
(*Enable the axis*)
IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

(2) The rising edge triggers the MC_MoveAbsolute command or this command is executed repeatedly to control axis motion.

```
IF PowerStatus THEN
(*Start absolute positioning command*)
MC_MoveAbsolute1(
Execute := StartMoveAbs,
```

```
Position := 10000,
Velocity := 1000,
Acceleration := 2000,
Deceleration := 2000,
Jerk := 0,
Direction := 4,
BufferMode := 0,
Axis := Axis0,
Done=> DoneMoveAbs,
Busy=> BusyMoveAbs,
Active=> ActiveMoveAbs,
CommandAborted=> ComMoveAbs,
Error=> ErrorMoveAbs,
ErrorID=>ErrorID3);
(* Repeated start of absolute positioning command *)
MC_MoveAbsolute2(
Execute := StartMoveAbs1,
Position := 50000,
Velocity := 2000,
Acceleration := 5000,
Deceleration := 5000,
Jerk := 0,
Direction := 4,
BufferMode :=0,
Axis := Axis0,
Done=> DoneMoveAbs1,
Busy=> BusyMoveAbs1,
Active=> ActiveMoveAbs1,
CommandAborted=> ComMoveAbs1,
Error=> ErrorMoveAbs1,
ErrorID=>ErrorID4);
END_IF
```

(3) In case of an error, call the MC_Reset function block to reset the axis.

```
IF ErrorMoveAbs THEN
(*Reset processing can be performed in case of error *)
  MC_Reset1(
  Execute := StartReset,
  Axis := Axis0,
  Done=>Done4,
  Busy=>Busy5,
  Error=>Error5,
  ErrorID=>ErrorID5);
END_IF
```

5.9.4 Precautions

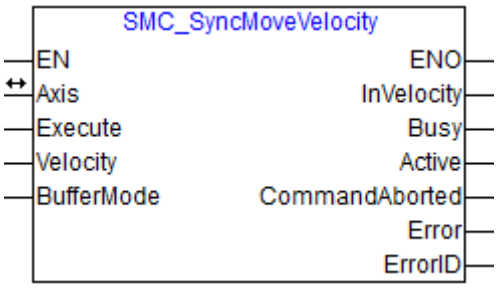
- When an absolute positioning command is used, the target position shall be entered between the positive limit and negative limit.
- When Direction is set to mcCurrentDirection, the command will act in the same direction as that of the previous action. Therefore, the motion direction should be confirmed according to the previous command before use.

5.9.5 Reference

- In case of any fault, please refer to the Description of Error Codes for function in the appendix.
- For information about initialization and enabling, please refer to the Initialization and Enable Chapter.
- BufferMode (buffer mode selection)Refer to the Appendix Description of Buffer Mode.

5.10 SMC_SyncMoveVelocity (Cyclic Synchronous Speed Control)

Speed control in cyclic synchronous speed mode.



5.10.1 Parameter

Parameter Description

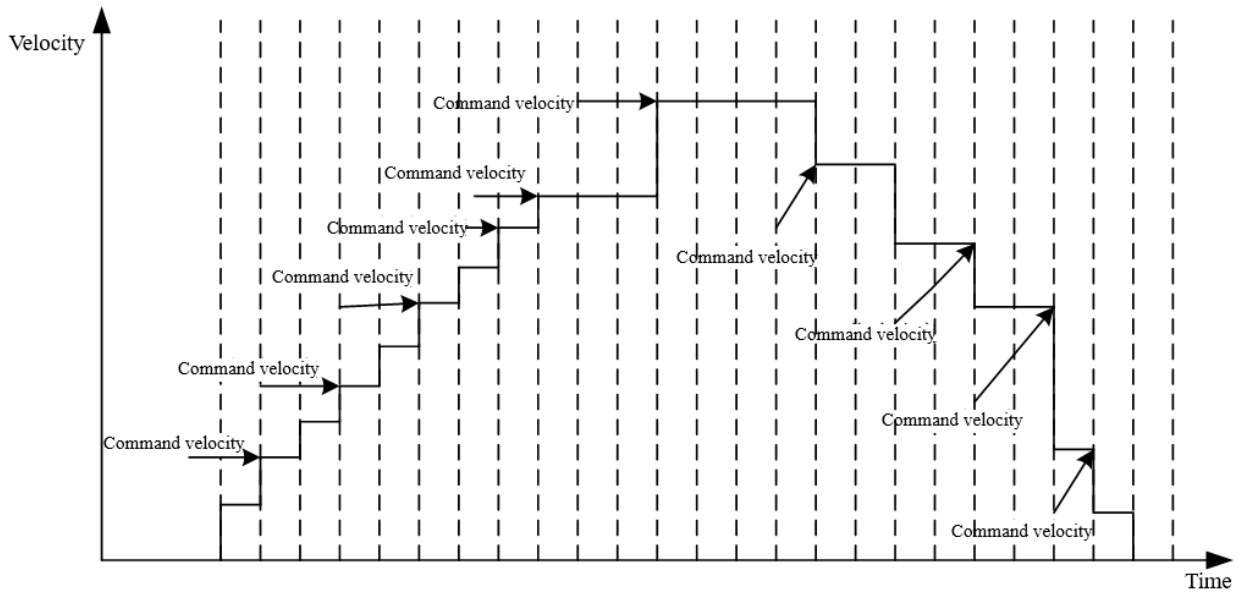
Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	Velocity	Target speed	No	-	Positive/negative/0	LREAL
	BufferMode	Buffer mode 0: Interrupt 1: buffer	YES	0	0: Aborting 1: Buffered	MC_BUFFER_MODE
OUTPUT	InVelocity	Target speed arrival	YES	FALSE	TRUE/FALSE	BOOL

	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Active	In control, TRUE when the command is executed	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.10.2 Functions

This command outputs the target speed given by the user program to the servo driver in the cyclic synchronous speed mode according to the task cycle. The details of this command are as follows:

- When Execute (rising edge trigger) is TRUE, the command triggers to switch the servo drive into speed control mode. When there is this command in the task, regardless of the priority of the task, it will reach the target speed in the next cycle. The schematic diagram of the command action is as follows.

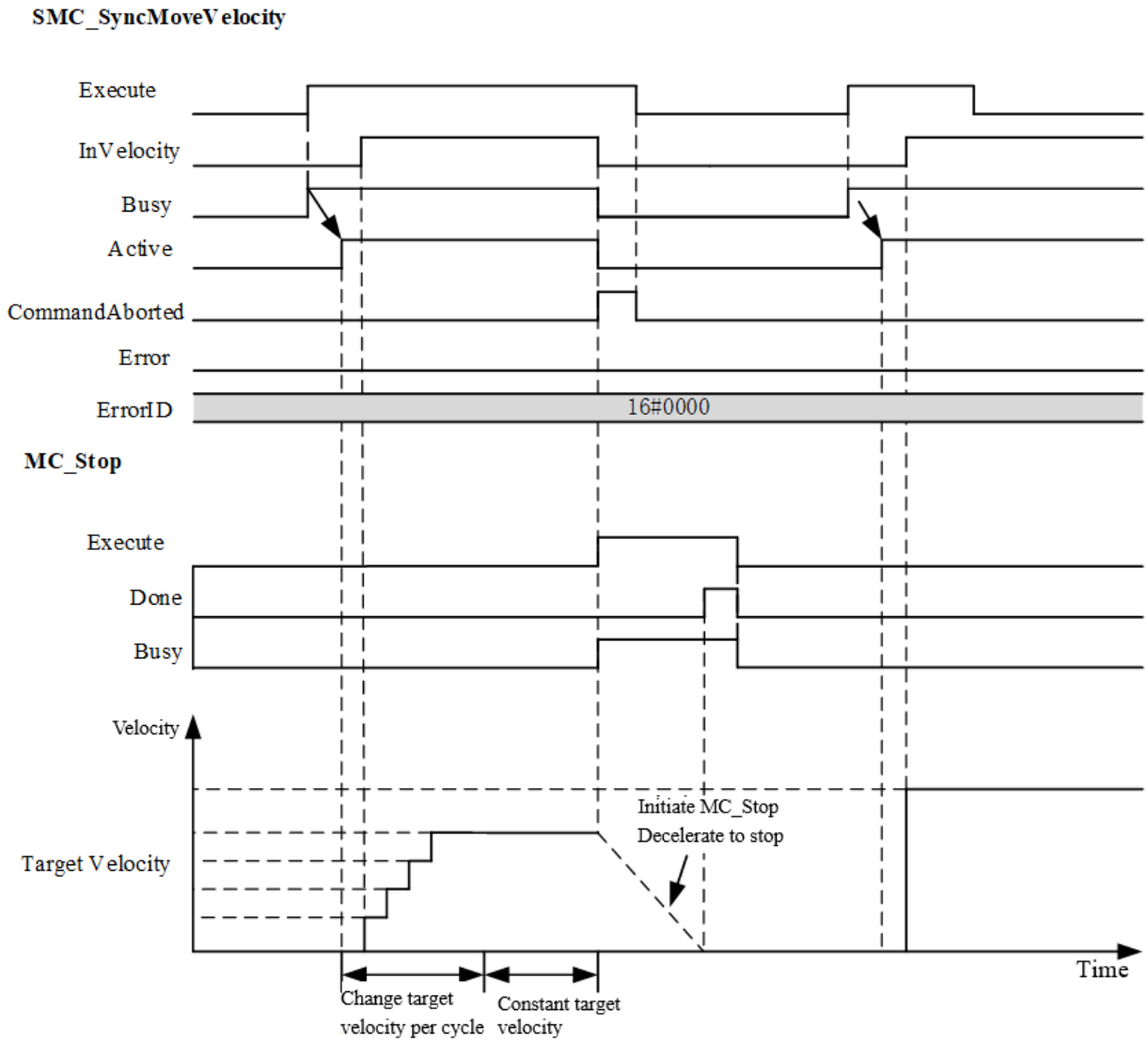


- When using this command, the following object data need to be mapped in the basic setting of axis: control word (6040 HEX), status word (6041 HEX), control mode (6060 HEX), control mode display (6061 HEX), target speed (60FF HEX) and actual velocity (606C HEX).
- When the input target speed is positive, it moves in the positive direction; when a negative number is input, it moves in the negative direction. The function block does not move when a speed of 0 is entered.
- This command supports real-time change of target speed. The user can modify the set target speed through the function block program in each cycle. When the input target speed is different from that of the previous cycle, the new target speed will be used. If it is the same as that of the previous cycle,

the target speed of the previous cycle will be used for movement.

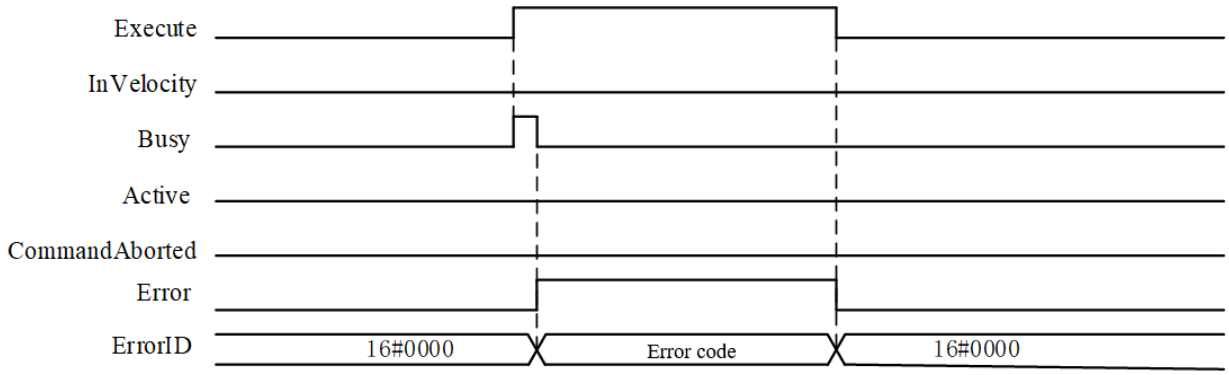
- This command supports interruption and buffering in BufferMode (buffer mode selection). It can interrupt other motion commands or be interrupted by other motion commands. When this command is restarted, you can choose buffering or interruption. To end this command, use the MC_Stop command to stop.
- Function block timing diagram

(1) Timing diagram under normal conditions



(2) Abnormal timing diagram

When an error occurs during the execution of this command, Error turns to TRUE, other pins turn to FALSE, the axis stops moving, and ErrorID outputs relevant error codes. The timing diagram for this error event is as follows.



5.10.3 Examples

Set up SMC_SyncMoveVelocity sample program to run cyclic synchronization velocity command.

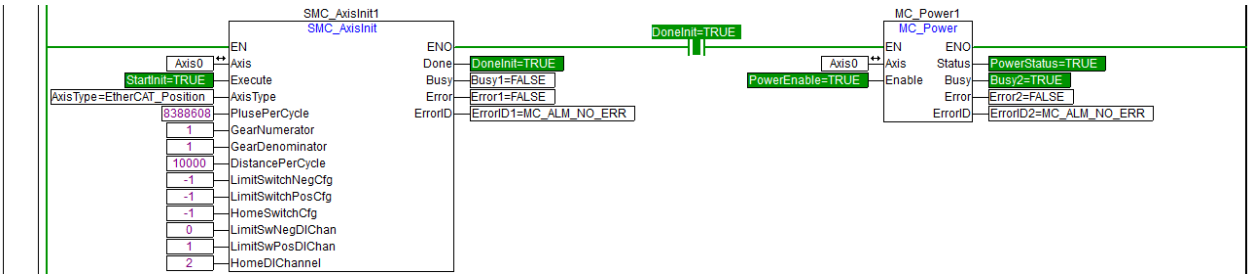
Definition of Primary Variable

Variable Name	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	Change to TRUE at the start of initialization
DoneInit	BOOL	FALSE	Change to TRUE upon completion of initialization
PowerEnable	BOOL	FALSE	Change to TRUE at the beginning of enabling
PowerStatus	BOOL	FALSE	Change to TRUE when enabled
StartSyncMove	BOOL	FALSE	TRUE when Cyclic Synchronous Velocity Command is initiated
BusySyncMove	BOOL	FALSE	Change to TRUE when delivery command is completed
ActiveSyncMove	BOOL	FALSE	Change to TRUE when the command is running
InVelocitySyncMove	BOOL	FALSE	Change to TRUE when the speed is reached
ComSyncMove	BOOL	FALSE	Become TRUE when the command is interrupted
ErrorSyncMove	BOOL	FALSE	Change to TRUE in case of command error
StartReset	BOOL	FALSE	Change to TRUE at the start of reset

■ LD program implementation

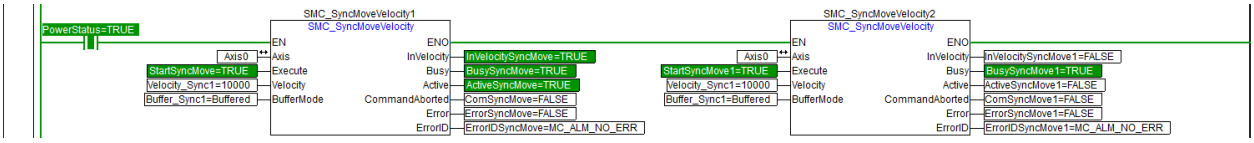
A program is established by using a ladder diagram, with the input parameter Velocity of 10000 and BufferMode parameter Aborting. The rising edge triggers the control axis to perform periodic synchronous velocity motion. The example program is as follows:

(1) After ensuring that the object parameter mapping is completed, first initialize the axis and set its type and various parameters. After the axis initialization is completed, enable the axis. When it enters the ON state of servo enabling, the axis can realize motion control.

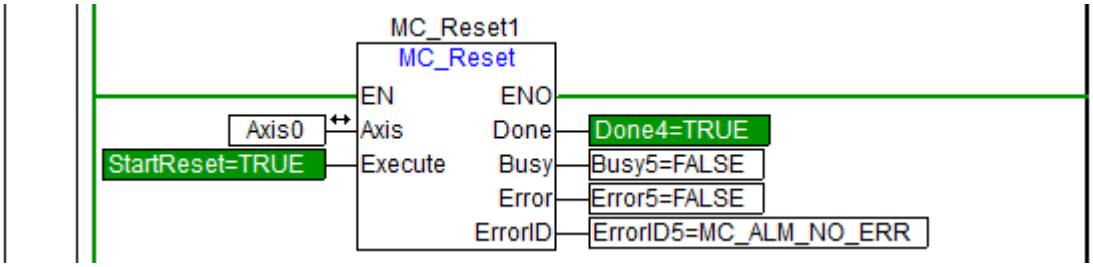


(2) After the axis is enabled, specified motion parameters can be entered in the input variables of

the functional block to send synchronous speed command and repeat command.



(3) When there is an error in the axis, the MC_Reset function block can be called for reset processing.



■ ST language program

(1) Initialize the axis

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

(2) Enable the axis

```
IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
```

```
END_IF
```

(3) Start cyclic synchronous speed command

```
IF PowerStatus THEN
SMC_SyncMoveVelocity1(
Execute := StartSyncMove,
Velocity := 10000,
BufferMode := Buffer_Sync1,
Axis := Axis0,
InVelocity=>InVelocitySyncMove,
Busy=>BusySyncMove,
Active=>ActiveSyncMove,
CommandAborted=>ComSyncMove,
Error=>ErrorSyncMove,
ErrorID=>ErrorIDSyncMove);
```

(4) Repeated start cycle synchronous speed command

```
  SMC_SyncMoveVelocity2(
  Execute := StartSyncMove1,
  Velocity := 10000,
  BufferMode := Buffer_Sync1,
  Axis := Axis0,
  InVelocity=>InVelocitySyncMove1,
  Busy=>BusySyncMove1,
  Active=>ActiveSyncMove1,
  CommandAborted=>ComSyncMove1,
  Error=>ErrorSyncMove1,
  ErrorID=>ErrorIDSyncMove1);
END_IF
```

(5) Reset processing can be carried out in case of error

```
MC_Reset1(
Execute := StartReset,
Axis := Axis0,
Done=>Done4,
Busy=>Busy5,
Error=>Error5,
ErrorID=>ErrorID5);
```

5.10.4 Precautions

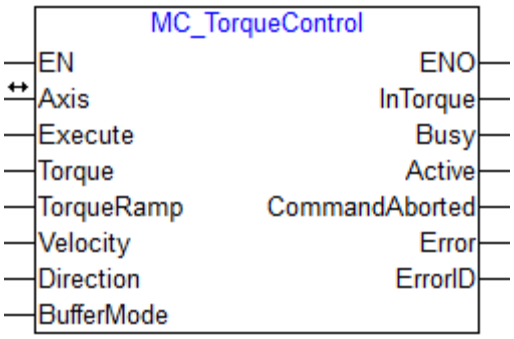
- Virtual axis and encoder axis are not supported in this command, and the axis type needs to be EtherCAT bus axis type.
- The override MC_SetOverride command is not valid for this command.
- The speed of this command fluctuates greatly, so be careful when using it in different scenarios.

5.10.5 Refererence

- If an error occurs, refer to the appendix Function Block Error Codes.
- For information about initialization and enabling, please refer to the section of Initialization and Enabling.
- BufferMode (buffer mode selection) refer to the appendix Buffer Mode Description.

5.11 MC_TorqueControl (Torque Control)

Single-axis torque control command, which controls the axis in torque mode with a ramp.



5.11.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	Torque	Target torque Unit: 1% of rated torque	YES	300	0 ~ 1000	LREAL
	TorqueRamp	Torque change rate Unit: 1% rated torque/s	No	-	Positive value	LREAL
	Velocity	Maximum speed	No	-	Non-negative	LREAL
	Direction	Direction of motion 0: Forward 2: Negative	YES	0	0: mcPositiveDirection 2: mcNegativeDirection	MC_DIRECTION
	BufferMode	Buffer mode 0: Interrupt 1: Buffer	YES	0	0: Aborting 1: Buffered	MC_BUFFER_MODE
OUTPUT	InTorque	Target torque reached The output is valid when the set torque reaches the target torque and the absolute value of the difference between the feedback torque and the	YES	OFF	TRUE/FALSE	BOOL

		target torque is within 5%.				
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Active	In control, TRUE when the command is executed	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.11.2 Functions

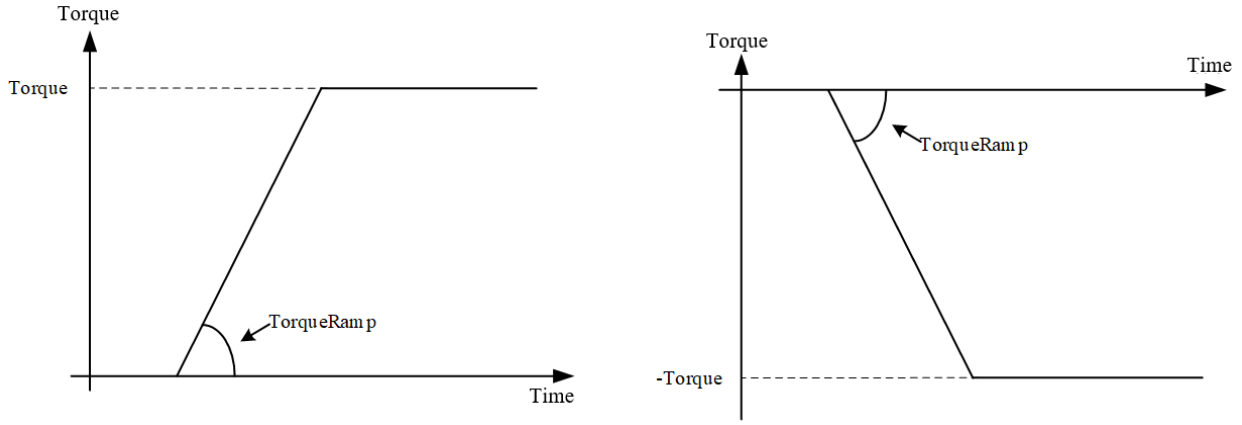
This command is used to realize the torque control function, which only supports EtherCAT bus servo axis and does not support virtual axis. This command uses the synchronous torque mode (CST) of servo drive to realize the torque control function, and sets the servo control mode to CST mode when the command is run.

The rising edge of this command is enabled. When the rising edge of Execute is enabled, the input parameters of Torque, TorqueRamp, Velocity and Direction are latched. When the Execute is TRUE, the modification of the input parameters is invalid. When the command operates normally, the axis is in Continuous Motion state and performs torque motion. The command output parameter Busy and Active are TRUE. When the set torque reaches the target torque and the absolute value of the difference between the feedback torque and the target torque is within 5%, the InTorque pin output is valid. When the command is executed, it can be interrupted by MC_Stop. After being interrupted, the CommandAboarted pin output is valid.

■ Description of Input Parameters

(1) **Torque:** Target torque, in the unit of 1% rated torque. When the set value exceeds the 2nd decimal place, the second digit shall be rounded off and the latter shall be directly discarded. For example, if the set torque value is 10.26435, the actual effective value is 10.3. The actual torque of the driver is limited by the positive and negative torque limits. When using it, pay attention to setting a reasonable maximum positive and negative torque limit value.

(2) **TorqueRamp:** Torque ramp, in the unit of 1% rated torque/s, indicating the change rate of target torque. When an command is executed, it specifies the inclination from the current torque to the output target torque.



TorqueRamp Diagram

(3) **Velocity:** The speed limit parameter is used to restrict the maximum speed during torque control of the drive, measured in pulse unit/s. The object dictionary for driver speed limiting is 0x607F. If 0x607F is mapped in the PDO parameters, the command writes the parameter Velocity setting value into 0x607F through PDO when the Execute rising edge occurs; if 0x607F is not mapped in the PDO parameters, the command writes the parameter Velocity setting value into 0x607F through SDO when the Execute rising edge occurs. Multi-axis scene object dictionary 0x607F needs to be offset by 0x800, please refer to the relevant manual of the drive for details.

When using this command for torque control, the following two conditions must be met to use Velocity as a maximum speed limit:

- The servo can limit the maximum speed of the motor through object dictionary 0x607F;
- Servo Object Dictionary 0x607F has pulse units instead of speed units.

The acceleration in torque control depends on the servo internal control.

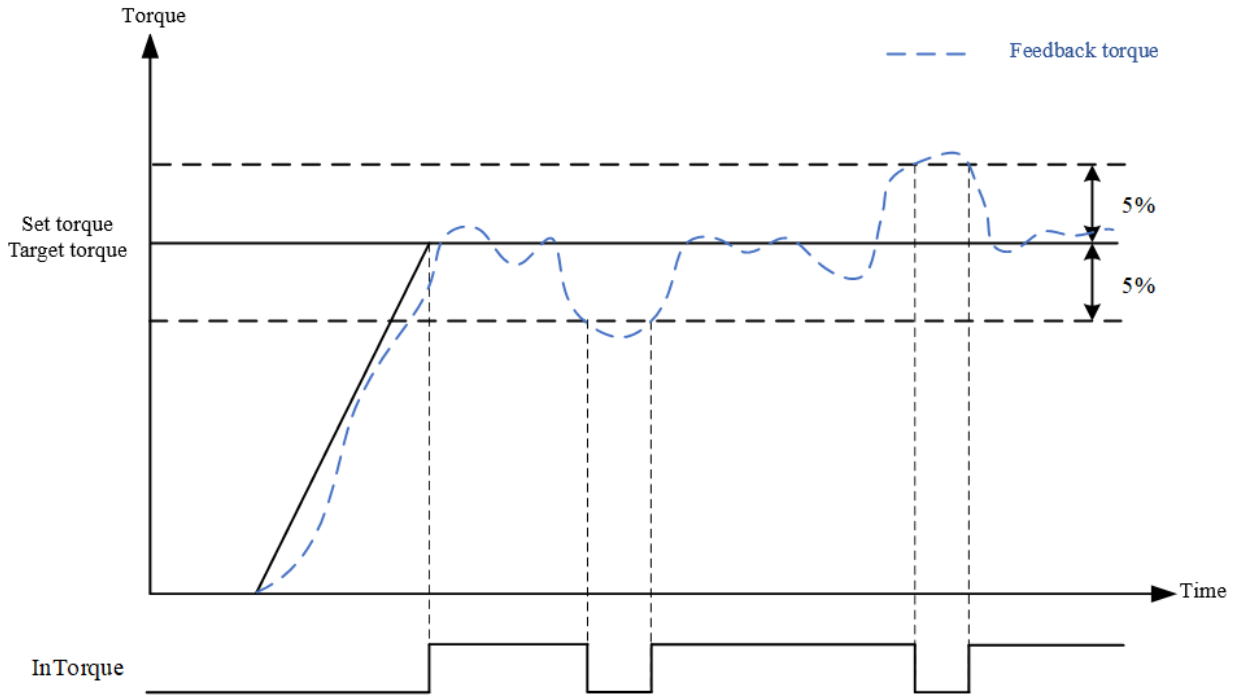
(4) **Direction:** Specify the output direction of target torque. To output in the positive direction of an axis, specify the positive direction; to output in the negative direction of an axis, specify the negative direction.

(5) **BufferMode:** Specify the connection mode between the previous axis action and this action. This command supports the following two modes:

- Interrupt: Immediately stop the current command and switch to this command.
- Buffer: The bufferd command can only be executed after the currently executing command is completed.

■ Output Parameter Description

InTorque: Target torque reached sign; the output is valid when the set torque reaches the target torque and the absolute value of the difference between the feedback torque and the target torque is within 5%. The schematic diagram is as follows:



InTorque Diagram

- Object Dictionary Description

  Torque control requires the drive to map the following object dictionary (offset 0x800 is required for multi-axis scene object dictionary):

  - Controlword (6040 hex)
  - Statusword (6041 hex)
  - Mode of Operation (6060 hex)
  - Mode of Operation Display (6061 hex)
  - Target Torque (6071 hex)
  - Torque Actual Value (6077 hex)
  - Velocity Actual Value (606C hex)

- Stop control in torque control mode

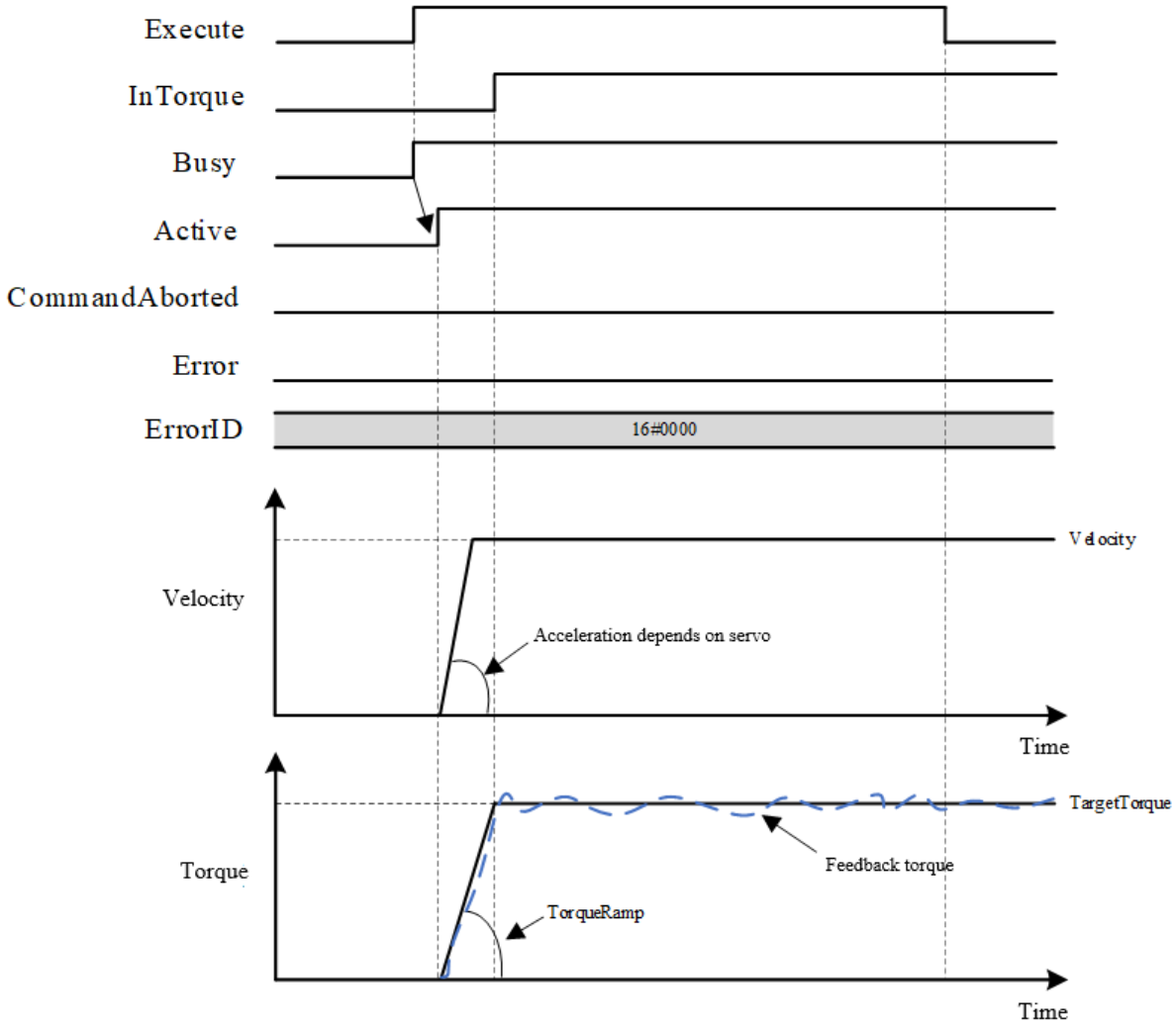
  During this command, the MC_Stop command can be called to stop the torque movement. When stopping, it decelerates the target torque from the current actual torque to 0 according to the set value of parameter TorqueRamp, and then cancels the torque command. After stopping, switch the drive control mode to the control mode before torque command movement.

  When the MC_Stop command is called to interrupt the MC_TorqueControl command, the Axis status switches from ContinuousMotion to Stopping. When the MC_TorqueControl command is interrupted, its output pin CommandAbort is valid.

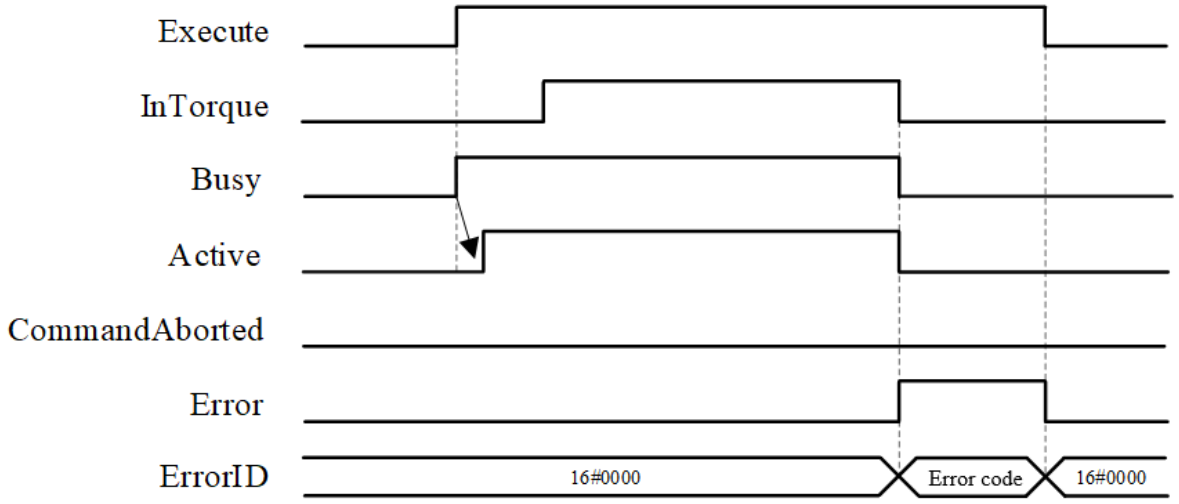
- Function block timing diagram

  (1) Timing diagram with normal execution of functional block and no error (actual torque can reach

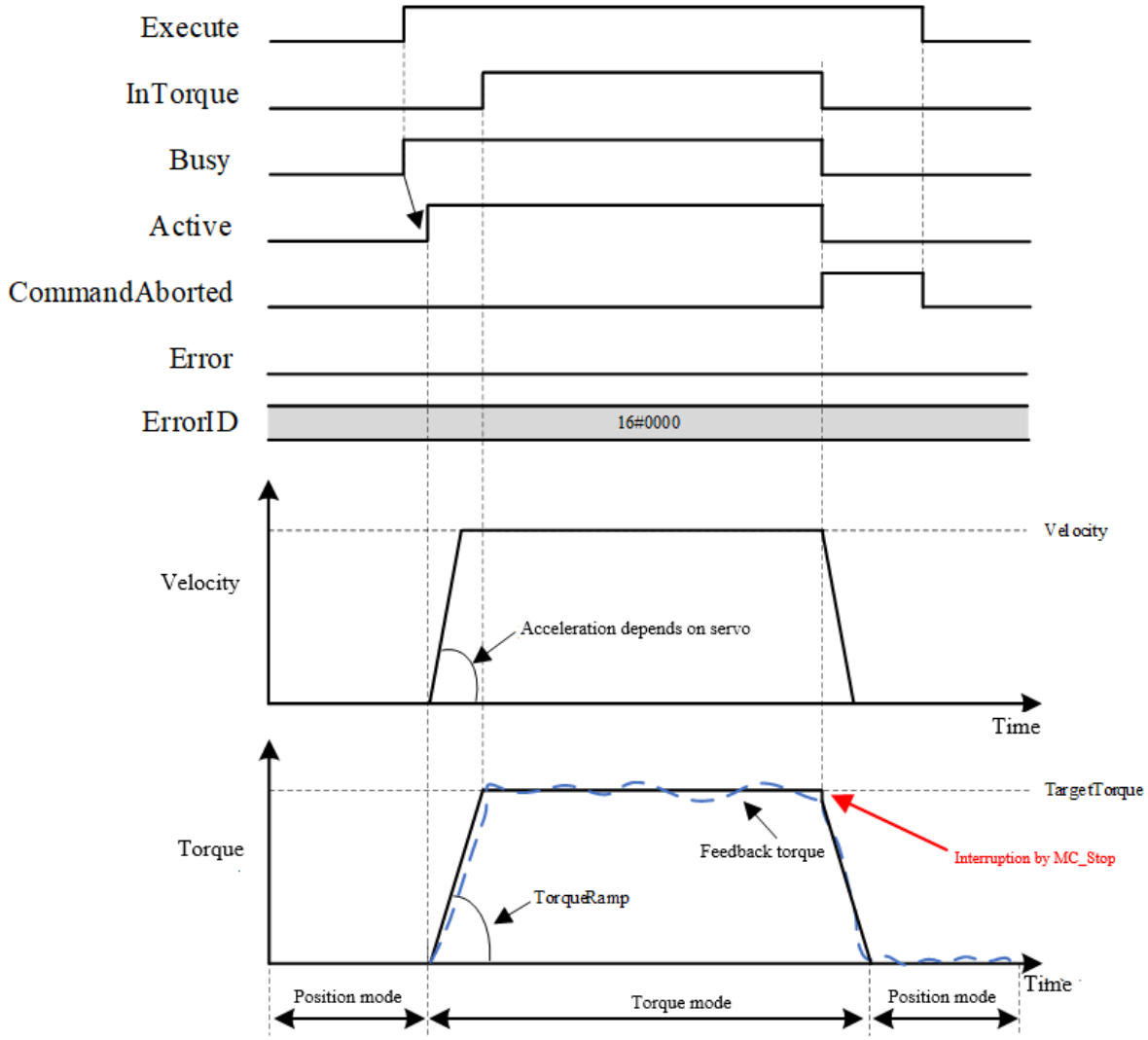
the given torque).



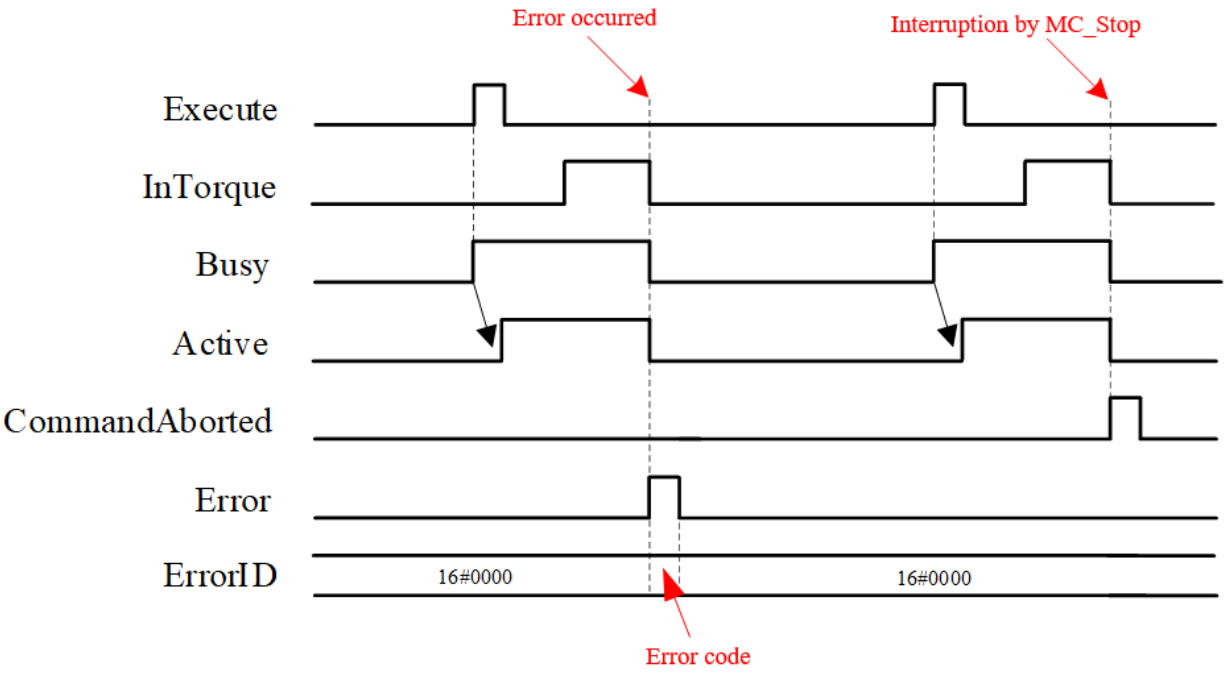
(2) Timing diagram of driver error reporting during function block execution.



(3) Timing diagram of MC_Stop interruption command during the operation of functional block (assuming that position control mode is before torque motion control).



(4) Execute the timing diagram of errors and interruption by MC_Stop command during the operation of functional block when it is maintained for a cycle.



5.11.3 Examples

This example uses the MC_TorqueControl function block for torque motion control, reads the actual torque value and finally calls the MC_Stop command to stop the torque motion command.

The servo needs to map the required PDO during torque motion control, otherwise an error will be reported.

Before torque motion control, use SMC_AxisInit to initialize axis-related parameters and call MC_Power to enable the axis.

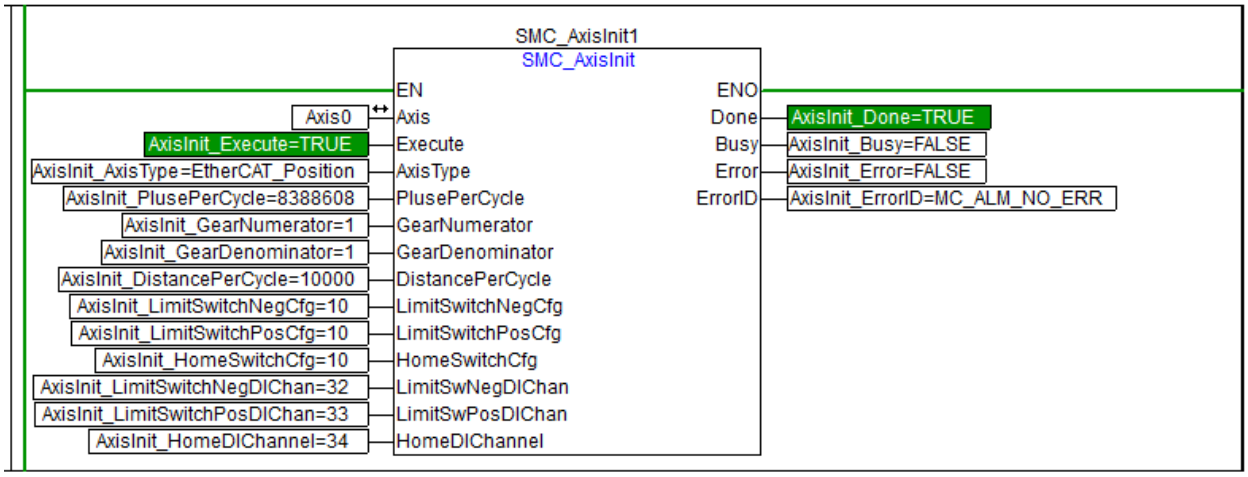
■ Variable

No.	Variable Name	Variable Type	Initialization Value	Variable Description
1	MC_TorqueControl0	MC_TORQUECONTROL	-	Definition of Torque Control Function Block Instance
2	TorqueControl_En	BOOL	FALSE	Torque control enable signal
3	TorqueControl_Torque	LREAL	0	Target torque value
4	TorqueControl_TorqueRamp	LREAL	0	Torque Ramp
5	TorqueControl_Vel	LRE Stop Function Block AL	0	Torque control maximum speed limit
6	TorqueControl_Dir	MC_DIRECTION	moPositiveDirection	Direction of motion
7	TorqueControl_Buffer	MC_BUFFER_MODE	Aborting	Buffer mode
8	TorqueControl_InTorque	BOOL	FALSE	Torque reaches the flag position
9	TorqueControl_Busy	BOOL	FALSE	Torque control execution flag bit
10	TorqueControl_Active	BOOL	FALSE	Torque control activation flag bit
11	TorqueControl_Abort	BOOL	FALSE	Torque control suspension flag bit
12	TorqueControl_Error	BOOL	FALSE	Torque control error flag bit
13	TorqueControl_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	Error code
14	MC_ReadActualTorque1	MC_READACTUALTORQUE	-	Actual Torque Reading Function Block Instance Definition
15	ReadTorque_En	BOOL	FALSE	Actual torque reading enable signal

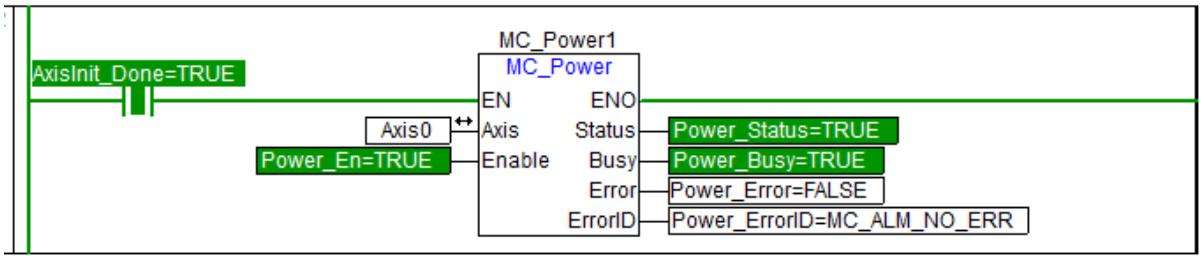
16	ReadTorque_Valid	BOOL	FALSE	Actual torque data valid flag bit
17	ReadTorque_Busy	BOOL	FALSE	Actual torque reading flag being executed
18	ReadTorque_Error	BOOL	FALSE	Actual torque reading error flag bit
19	ReadTorque_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	Actual torque reading error code
20	ReadTorque_Torque	LREAL	0	Actual torque value read
21	MC_Stop1	MC_STOP	-	Stop Function Block Instance Definition
22	Stop_En	BOOL	FALSE	Stop function block enable signal
23	Stop_Dec	LREAL	0	Stop function block deceleration
24	Stop_Jerk	LREAL	0	Stop function block jerk
25	Stop_Done	BOOL	FALSE	Stop completion flag bit
26	Stop_Busy	BOOL	FALSE	Stopping flag bit
27	Stop_Error	BOOL	FALSE	Error flag bit
28	Stop_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	Error code

■ LD program implementation

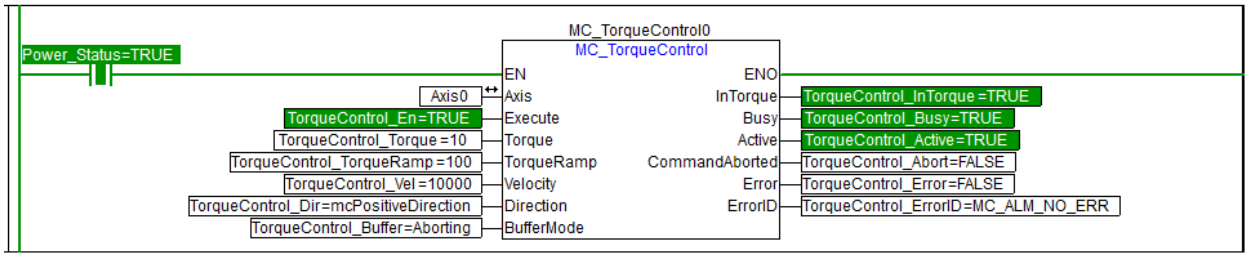
(1) Call the SMC_AxisInit command to initialize axis parameters, which shall be set according to actual servo conditions.

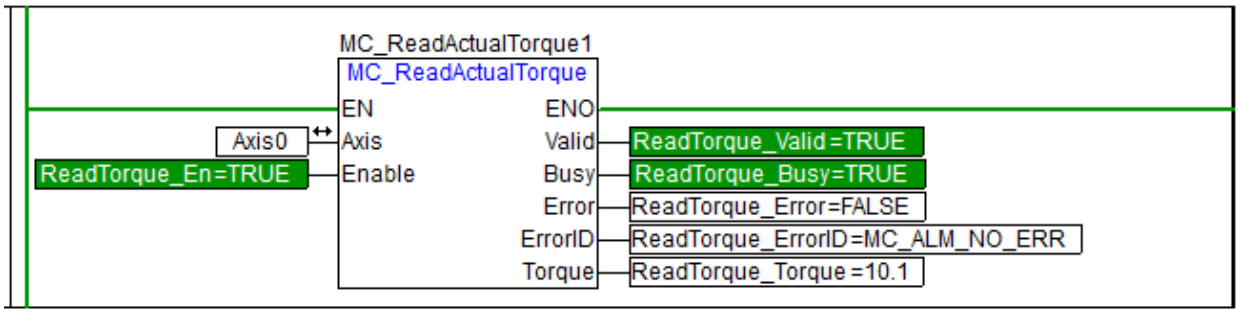


(2) After the axis initialization is completed, call MC_Power command to enable the axis.

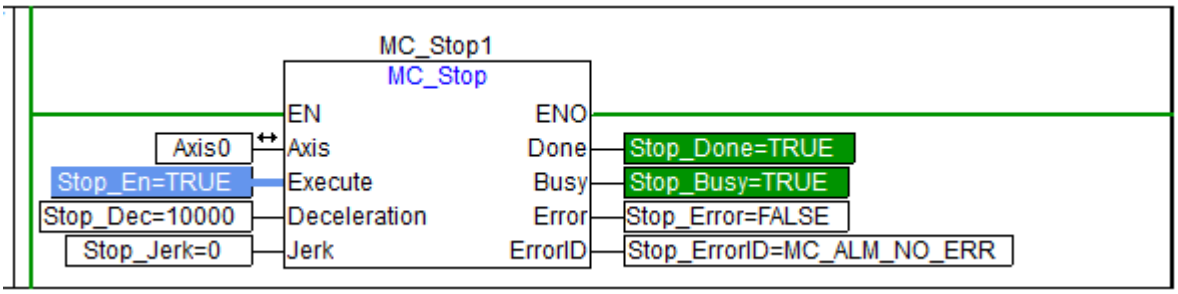


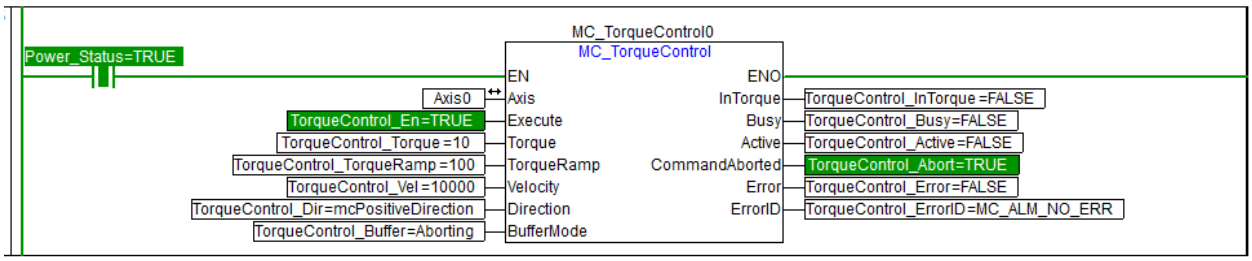
(3) After the axis is enabled successfully, call MC_TorqueControl command to control the torque motion of the axis. The current torque value of the servo motor can be obtained through MC_ReadActualTorque.





(4) When it is necessary to stop the torque motion control, call MC_Stop to interrupt the torque control command. After interruption, the MC_TorqueControl CommandAborted command will be valid.





■ ST program implementation

(1) Call the SMC_AxisInit command to initialize axis parameters, which shall be set according to actual servo conditions.

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
```

```
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

(2) After the axis initialization is completed, call MC_Power command to enable the axis.

```
IF TRUE = AxisInit_Done THEN
   Power_En := TRUE;
ELSE
   Power_En := FALSE;
END_IF;
MC_Power1(
Enable := Power_En,
Axis := Axis0,
Status=>Power_Status,
Busy=>Power_Busy,
Error=>Power_Error,
ErrorID=>Power_ErrorID);
```

(3) After the axis is enabled successfully, call MC_TorqueControl command to control the torque motion of the axis. The current torque value of the servo motor can be obtained through MC_ReadActualTorque.

```
IF TRUE = Power_Status THEN
   MC_TorqueControl0(
   Execute := TorqueControl_En,
   Torque := TorqueControl_Torque,
   TorqueRamp := TorqueControl_TorqueRamp,
   Velocity := TorqueControl_Vel,
   Direction := TorqueControl_Dir,
   BufferMode := TorqueControl_Buffer,
   Axis := Axis0,
   InTorque=>TorqueControl_InTorque,
   Busy=>TorqueControl_Busy,
   Active=>TorqueControl_Active,
   CommandAborted=>TorqueControl_Abort,
   Error=>TorqueControl_Error,
   ErrorID=>TorqueControl_ErrorID);
END_IF
MC_ReadActualTorque1(
Enable := ReadTorque_En,
```

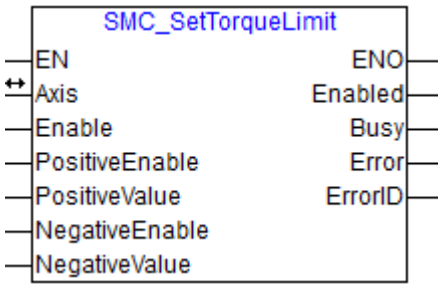
```
Axis := Axis0,
Valid=>ReadTorque_Valid,
Busy=>ReadTorque_Busy,
Error=>ReadTorque_Error,
ErrorID=>ReadTorque_ErrorID,
Torque=>ReadTorque_Torque);
```

(4) When it is necessary to stop the torque motion control, call MC_Stop to interrupt the torque control command. After interruption, the MC_TorqueControl CommandAborted command will be valid.

```
MC_Stop1(
Execute := Stop_En,
Deceleration := Stop_Dec,
Jerk := Stop_Jerk,
Axis := Axis0,
Done=>Stop_Done,
Busy=>Stop_Busy,
Error=>Stop_Error,
ErrorID=>Stop_ErrorID);
```

5.12 SMC_SetTorqueLimit (Set Axis Torque Limit)

Set the limit value of servo positive/negative torque.



5.12.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Enable	Enable	No	-	TRUE/FALSE	BOOL
	PositiveEnable	Positive limiting enabled	No	-	TRUE/FALSE	BOOL
	PositiveValue	Maximum torque in forward direction Unit: 1% rated torque	YES	300	0 ~ 1000	LREAL
	NegativeEnable	Negative limiting enable	No	-	TRUE/FALSE	BOOL
	NegativeValue	Maximum torque in negative direction	YES	300	0 ~ 1000	LREAL

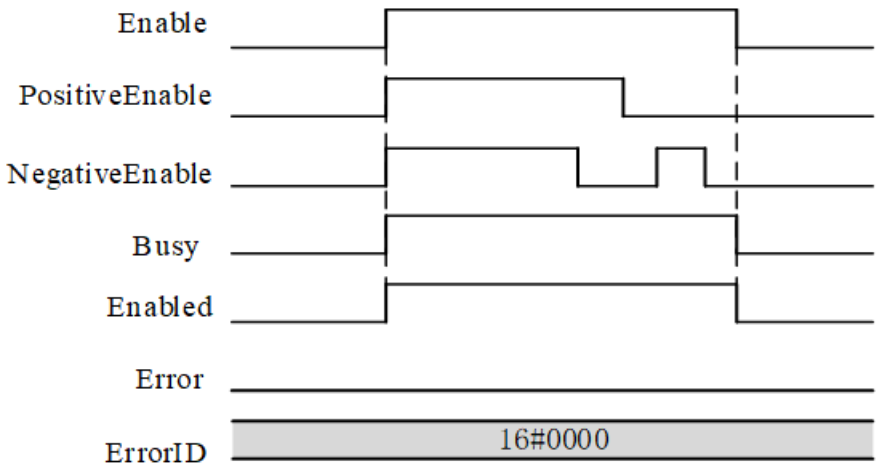
		Unit: 1% rated torque				
OUTPUT	Enabled	Setting succeeded	No	-	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.12.2 Functions

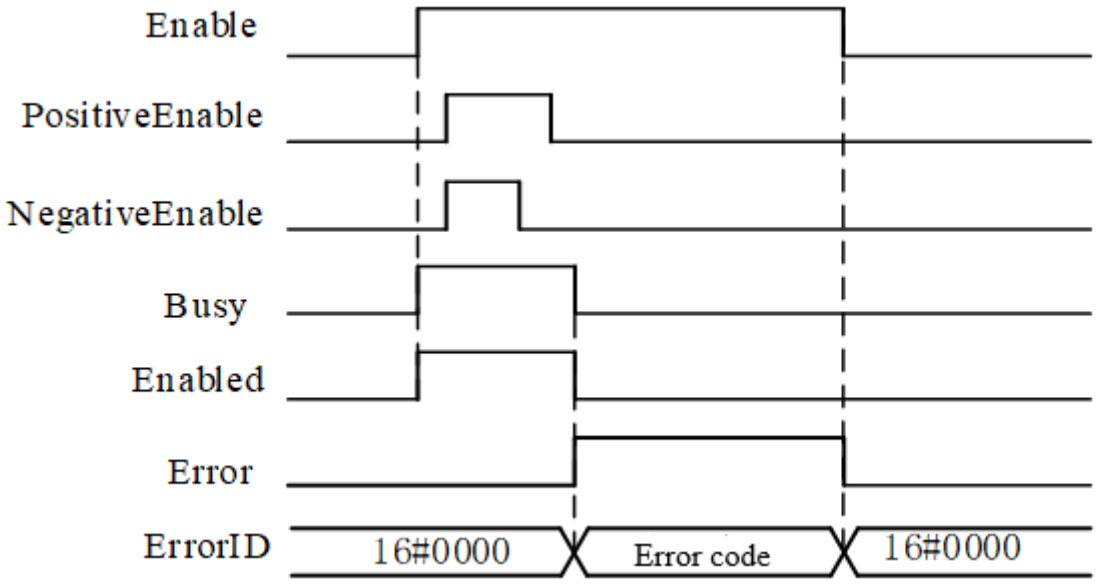
This command can set the positive and negative torque limit values for the servo driver according to CIA402 protocol. The details of this command are as follows.

- When Enable is TRUE, if PositiveEnable is set to TRUE, the positive maximum torque value will be limited by PositiveValue; If NegativeEnable is set to TRUE, the maximum negative torque value will be limited by a NegativeValue.
- When Enable is set to FALSE, the limit value will not be set and the torque will be limited by using the previously set limit value. When PositiveEnable or NegativeEnable is set to FALSE, the positive and negative limit values will not be set and the previously set value will be maintained for limitation.
- The unit of the set value is 1% of rated torque. When the set value exceeds the second decimal place, the second decimal place shall be rounded off.
- The input of this command is Enable type. When Enable is TRUE, the functional block keeps running and there is no restart trigger for the command.
- The illegality check of positive and negative limit values will only be carried out when PositiveEnable or NegativeEnable is set to TRUE.
- Function block timing diagram

(1) Normal timing diagram



(2) Error timing diagram



5.12.3 Examples

Set up SMC_SetTorqueLimit sample program to set the limit value of servo positive/negative torque.

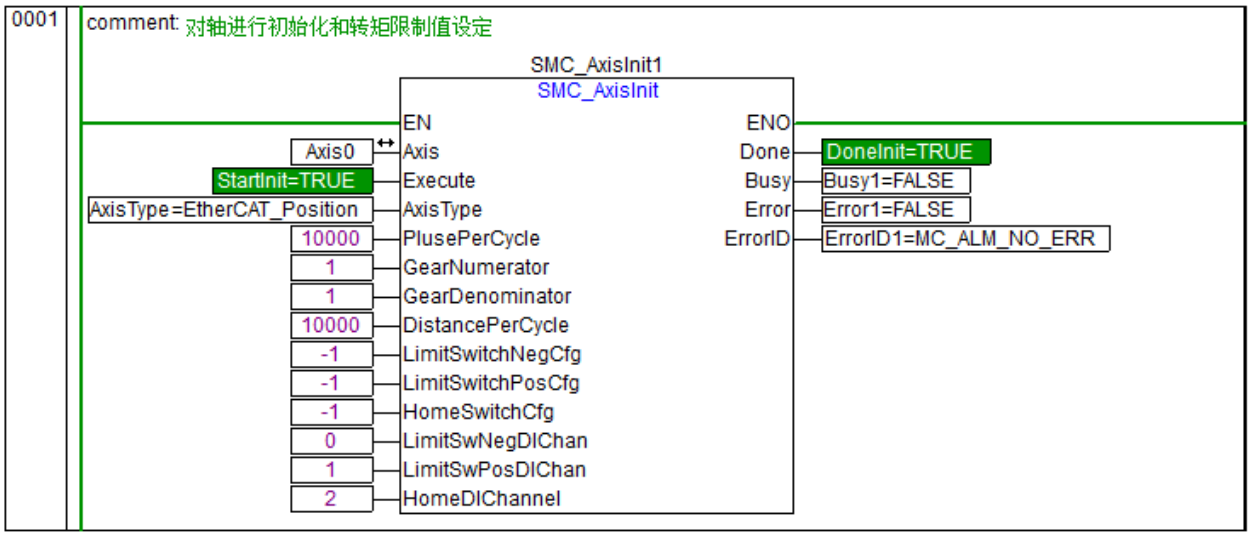
Primary variables

Variable Name	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	becomes TRUE at the start of initialization
DoneInit	BOOL	FALSE	Changes to TRUE upon completion of initialization
SetTorqEnable	BOOL	FALSE	Changes to TRUE at the beginning of enabling
PosSetTorqEnable	BOOL	FALSE	Changes to TRUE when forward enable is set
NegSetTorqEnable	BOOL	FALSE	TRUE when negative enabling is set
EnabledSetTor	BOOL	FALSE	Changed to TRUE when setting is successful
BusySetTor	BOOL	FALSE	Changes to TRUE at start setting
ErrorSetTor	BOOL	FALSE	Changes to TRUE when an error occurs

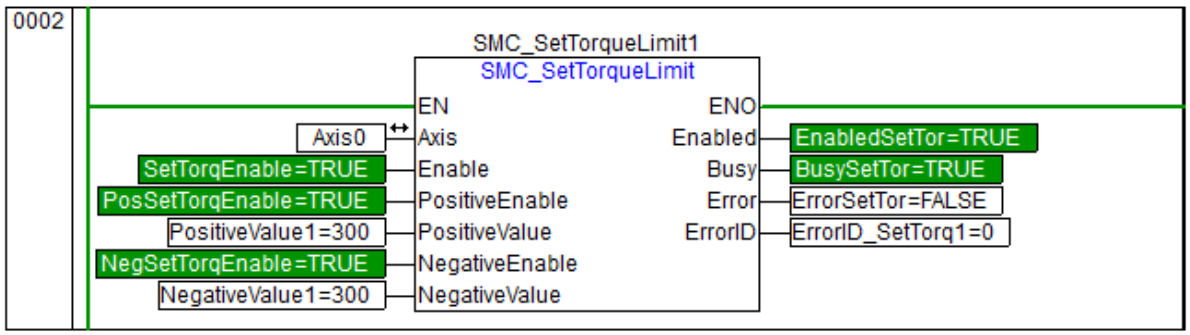
- LD program implementation

The ladder diagram is used to build the program, and the input parameters PositiveValue and NegativeValue are both 300. Set SMC_SetTorqueLimit Enable to TRUE, sample procedure is as follows:

(1) Before setting the axis torque, initialize the axis first.



(2) Set the limit value of servo positive/negative torque according to the set value



■ ST language program

(1) Initialize the setting of axis number and axis type.

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

(2) Torque limit setting for the axis

```
SMC_SetTorqueLimit1(
Enable := SetTorqEnable,
PositiveEnable := PosSetTorqEnable,
PositiveValue := 300,
NegativeEnable := NegSetTorqEnable,
NegativeValue := 300,
Axis := Axis0,
Enabled=>EnabledSetTor,
Busy=>BusySetTor,
Error=>ErrorSetTor,
ErrorID=>ErrorID_SetTorq1);
```

5.12.4 Precautions

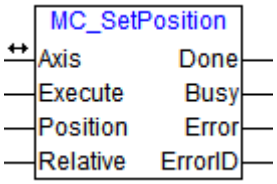
- Virtual and encoder axes are not supported by this function block.
- Due to differences in servo motors, the Default Value of some servo internal torque limits is 350 or 300. When the set value exceeds this value, an error will be reported, and the maximum value inside the servo is its Default Value.

5.12.5 Reference

- If an error occurs, please refer to the appendix Function Block Error Code Description.
- Refer to the chapter of MC_TorqueControl command for use after setting the limit value.

5.13 MC_SetPosition (Set Current Position)

Redefines the current position of a specified axis, either absolutely or relative.



5.13.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF

INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	Position	LOCATION	No	-	Positive/negative/0	LREAL
	Relative	Mode selection FALSE: Absolute mode, defining the Position as the current position TRUE: Relative mode, increasing the value corresponding to the current position	YES	FALSE	TRUE/FALSE	BOOL
OUTPUT	Done	Completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.13.2 Functions

This command redefines the current position of the specified axis in an absolute or relative manner.

- command function details

(1) Input parameter lock

The rising edge triggering of this command is valid. At the rising edge of the Execute input, the axis ID, Position and Relative value of the input/output parameter Axis are latched. During command execution, the modified latched parameters are invalid until another rising edge triggers this command.

(2) Position setting mode

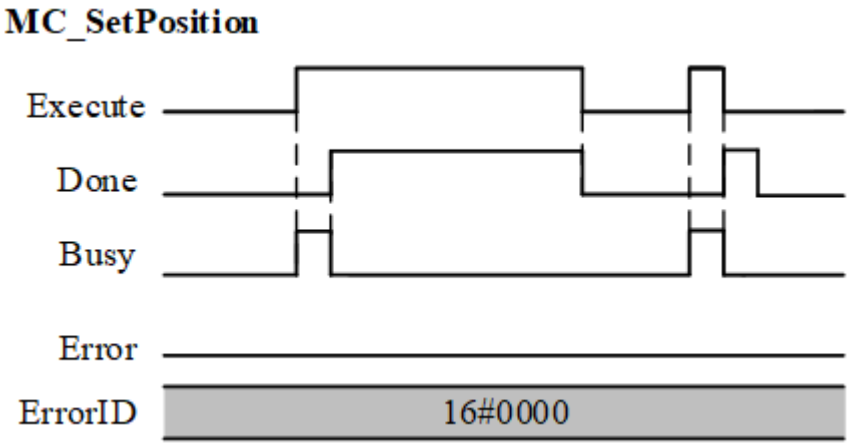
When the input parameter Relative=FALSE, this command redefines the current position of the specified axis in an absolute way. After executing this command, Position is defined as the current position of the axis;

When the input parameter Relative=TRUE, this command redefines the current position of the specified axis in a relative manner. After executing this command, add Position to the current position of the axis as the current position of the axis.

- Timing diagram

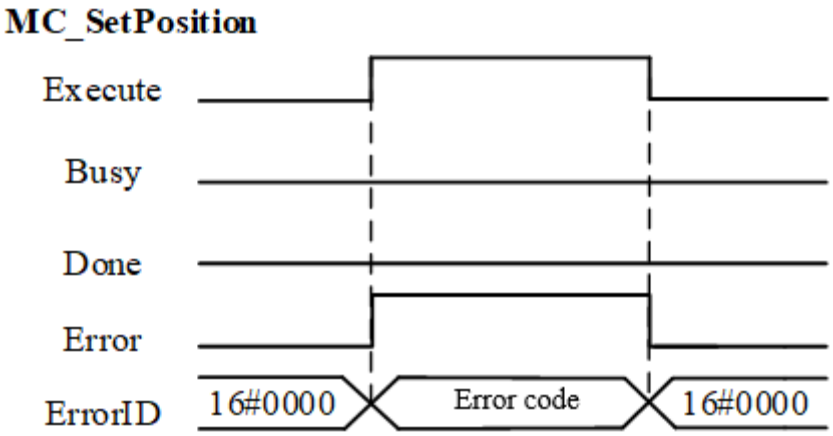
(1) Normal timing diagram

On the rising edge of the Execute input, if there is no abnormality, the Busy pin is set to TRUE. After the axis position is successfully set, the Done pin is set to TRUE. At this time, the Busy pin is set to FALSE, and the Done pin will be kept for at least one cycle.



(2) Error pin timing diagram

When this command is executed, if the parameter and axis type are illegal (the axis type is Unused), the call of this command will fail. At this time, the command output pin Error=True, and the error code ErrorID reports an abnormal error value. Refer to the appendix Description of Error Codes to obtain the cause of the error. When the command input pin Execute is at a low level, all output pins are restored to their Default Values.



- When the abnormal situation of the command is solved, it is necessary to trigger Execute again on the rising edge and start the MC_SetPosition functional block to complete the current position setting of the axis.

5.13.3 Examples

Create an MC_SetPosition example program to set the current position of the axis.

■ LD program implementation

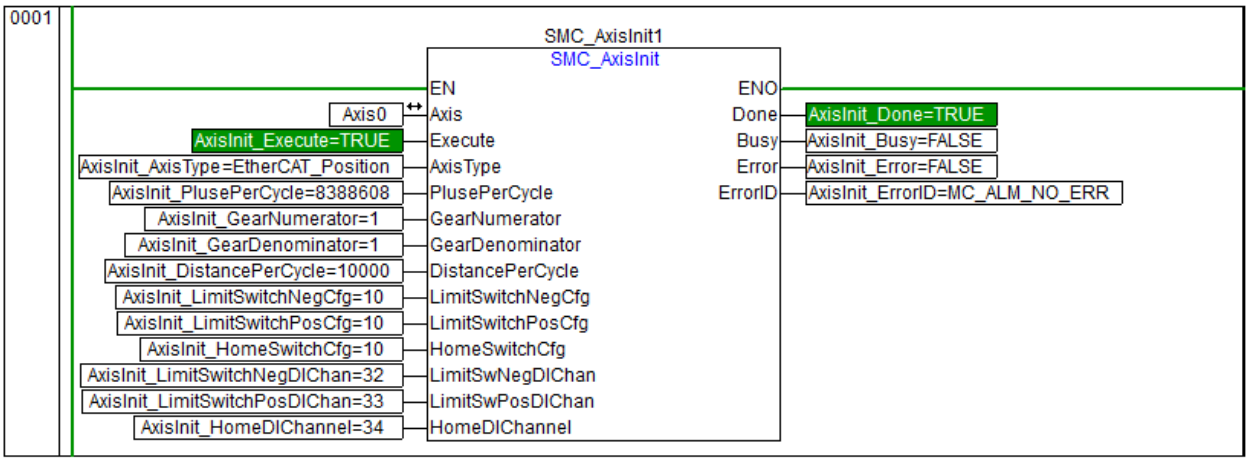
Using a ladder diagram, set the Position parameter to 0 and Relative to FALSE, defining the current axis position as 0 in absolute mode.

Description of Primary Variables

Variable Vame	Variable Type	Initial Value	Notes
Axis0	AXIS_REF	-	Axis variable for axis 0
SetPos_Done	BOOL	FALSE	Variable when the current position of the axis is successfully set. This variable becomes TRUE when the current position of the axis is successfully set.
Switch1	BOOL	FALSE	Axis ready state variable. After axis initialization, Switch to TRUE and the axis is ready.

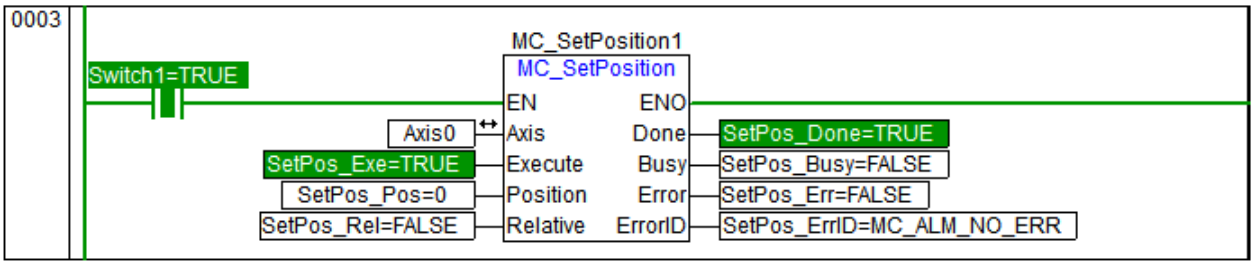
(1) Axis initialization

Set Axis0.AxisID = 0, axis type to 50:EtherCAT_Position, and call the axis initialization SMC_AxisInit command to complete the axis initialization configuration. For the SMC_AxisInit command, please refer to its Command Description Chapter.



(2) Execute the MC_SetPosition command

Call the MC_SetPosition command to set the current position of the axis. After initializing and configuring Axis0, and once the axis is ready (Switch1=TRUE), set the current position of the axis.



■ ST language

Use ST language to set up a program, input the parameter Position as 10000 and Relative as TRUE, define the current axis position in relative mode, and add 10000 on the basis of the current axis position as the current axis position.

(1) Primary variables

The primary variables of ST language program are the same as those of LD language program.

(2) Axis initialization

Set Axis0.AxisID = 0, axis type to 50:EtherCAT_Position, and call the axis initialization SMC_AxisInit command to complete the axis initialization configuration. For the SMC_AxisInit command, please refer to Command Description Chapter.

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

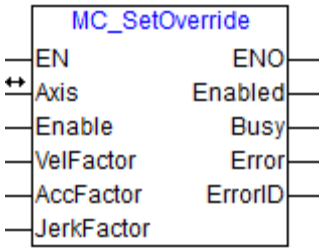
(3) Execute the MC_SetPosition command

Redefines the current position of the specified axis in an absolute manner.

```
Relative humidity: = TRUE; (* Mode selection parameter setting *)
Position: =10000; (*Position parameter setting *)
IF  Switch1 THEN
   MC_SetPosition1(
   Execute := Exe_SetPos,
   Position := Position,
   Relative := Relative,
   Axis := Axis0,
   Done=>Done_Pos,
   Busy=>Busy_Pos,
   Error=>Error_Pos,
   ErrorID=>ErrorID_Pos);
END_IF
```

5.14 MC_SetOverride

This command sets the speed, acceleration/deceleration and jerk scale factor, which can increase or decrease the axis running speed, acceleration/deceleration and jerk according to the set ratio.



5.14.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Enable	Enable	No	-	TRUE/FALSE	BOOL
	VelFactor	is the speed scaling factor in 1%	YES	100	0 ~ 500	LREAL
	AccFactor	Acceleration and deceleration scale factor, in 1%	YES	100	0 ~ 500 (non-zero)	LREAL
	JerkFactor	is the jerk scale factor in 1%	YES	100	0 ~ 500 (non-zero)	LREAL
OUTPUT	Enabled	Setting succeeded	No	-	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.14.2 Functions

This command can increase or decrease the speed, acceleration and deceleration of axis operation according to the set scale factor. The details of this command are as follows:

- This command overrides the target speed of the axis and can change the target speed during the movement of the axis. The following commands can be modified:
  - MC_MoveJog
  - MC_MoveRelative
  - MC_MoveAbsolute
  - MC_MoveVelocity
- The unit of the override scale factor is [%].

  Modified target speed = currently executed speed × speed scale factor;

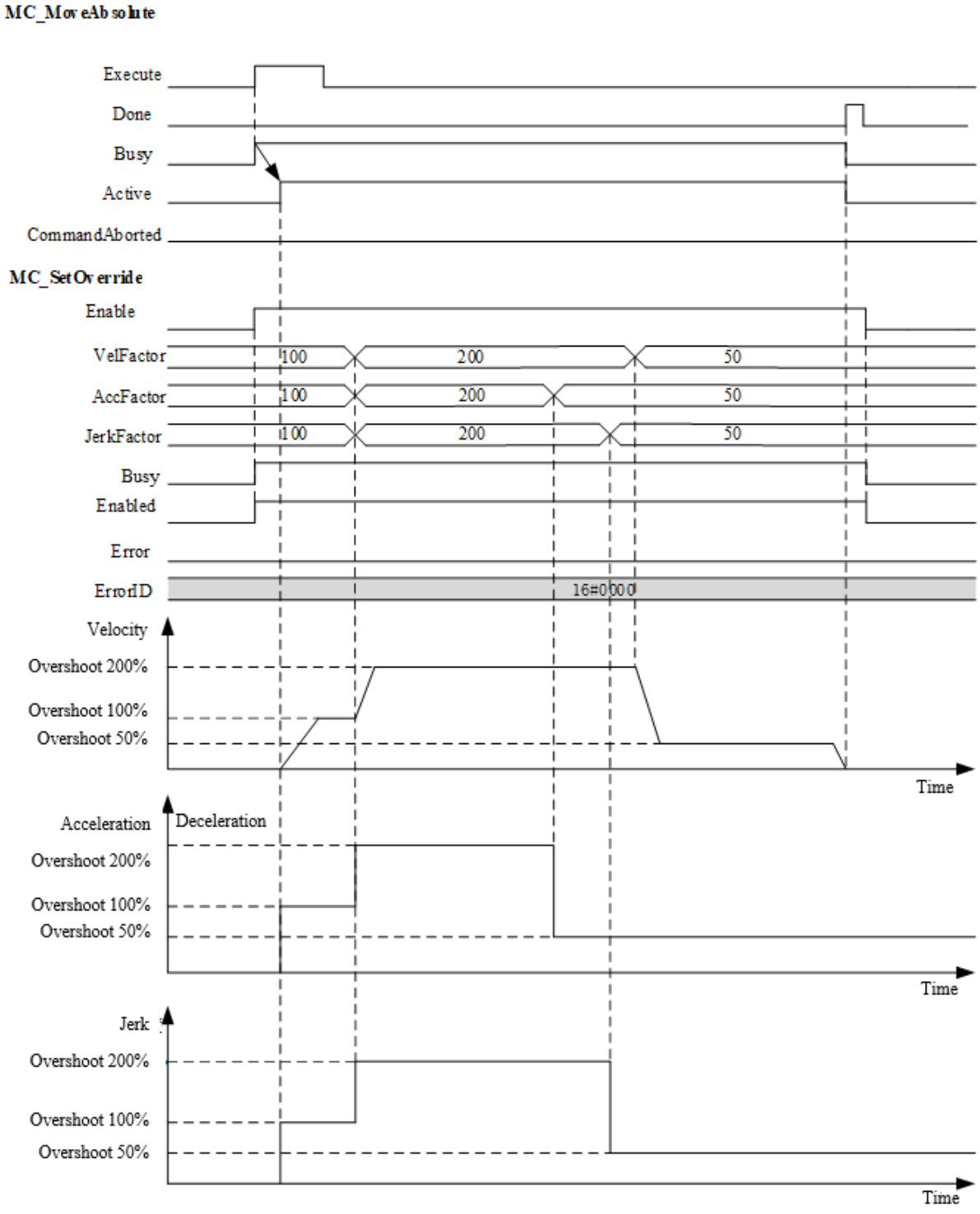
  Modified acceleration/deceleration = current acceleration/deceleration × scale factor of acceleration/deceleration;

  Modified jerk = current jerk × jerk scale factor.
- When the target speed after override is greater than the maximum speed, the axis runs at the maximum speed, and acceleration/deceleration and jerk are also consistent.

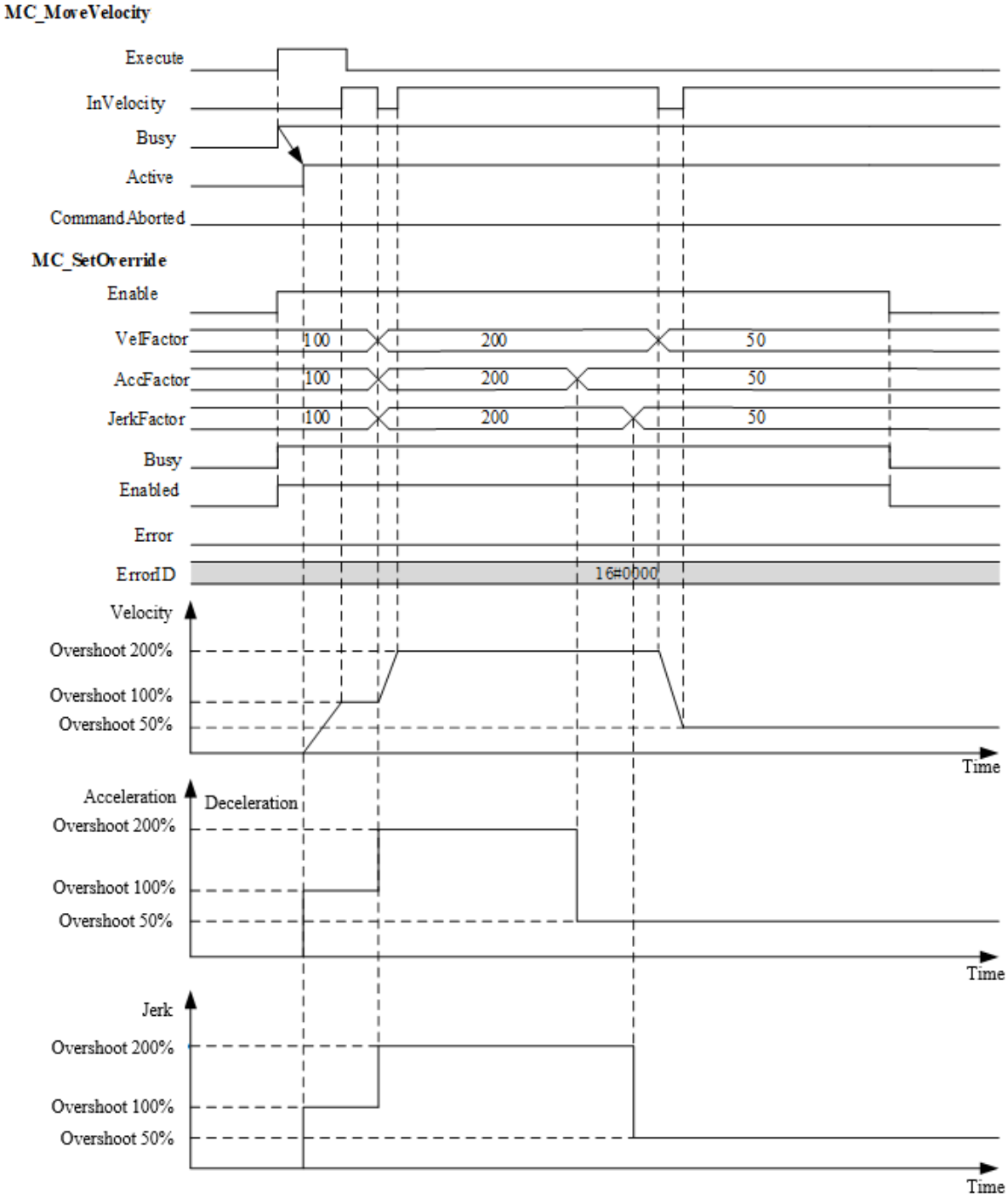
- When Enable is TRUE, the function block continues to set the override of scale factor. When Enable is FALSE, the axis keeps the value of override scale factor set last time for motion.
- Function block timing diagram

(1) Override the MC_MoveAbsolute

When Enable is set to TRUE, Busy and Enabled immediately turns TRUE; when Enable is set to FALSE, Busy and Enabled immediately turns FALSE. The timing diagram of using override command in MC_MoveAbsolute is as follows.

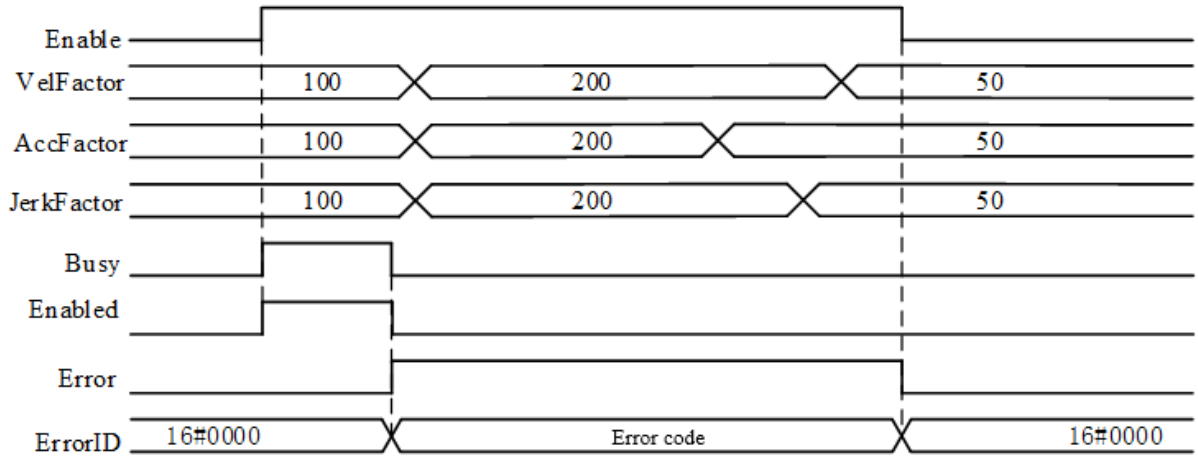


(2) Perform override on the MC_MoveVelocity command. When InVelocity becomes TRUE, use override to modify the target speed, and then InVelocity will become FALSE; when the new target speed is reached, InVelocity will become TRUE. When Enable of MC_SetOverride is set to FALSE, the axis moves with the override scaling factor value set last time. The command timing diagram is as follows.



(3) Error Timing diagram

When an error occurs during the execution of this command, Error becomes TRUE and Enabled and Busy become FALSE. After the error is cleared, Error becomes FALSE and Enabled and Busy becomes TRUE. The timing diagram is shown below.



■ Restart and multiple startup commands

The input of this command is Enable type. When Enable is TRUE, the function block keeps running. When there is an MC_SetOverride command being executed and other MC_SetOverride commands are started, the later-executed command will be processed first.

5.14.3 Examples

Create MC_SetOverride example program to set the override value of axis motion parameters.

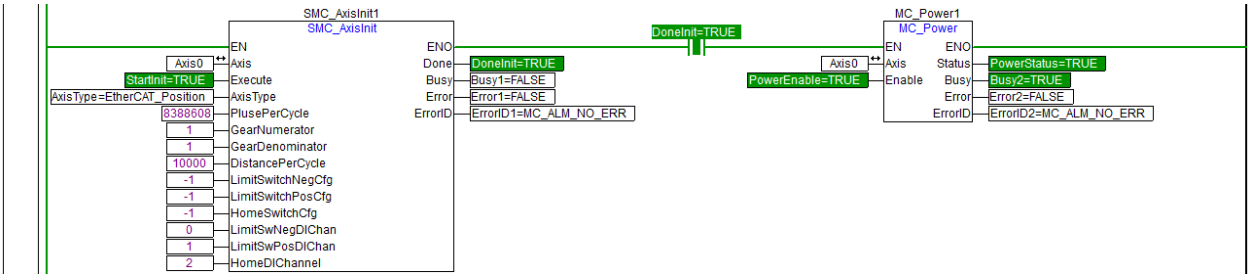
■ Primary variables

Variable Name	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	Change to TRUE at the start of initialization
DoneInit	BOOL	FALSE	Change to TRUE upon completion of initialization
PowerEnable	BOOL	FALSE	Change to TRUE at the beginning of enabling
PowerStatus	BOOL	FALSE	Change to TRUE when enabled
StartMoveVel	BOOL	FALSE	TRUE at start of motion command
InVelMoveVel	BOOL	FALSE	Change to TRUE when the commanded speed is reached
StartOverride	BOOL	FALSE	Change to TRUE at the start of override execution
BusyOverride	BOOL	FALSE	Change to TRUE at override
EnabledOverride	BOOL	FALSE	Change to TRUE when override is successful
ErrorOverride	BOOL	FALSE	Change to TRUE in case of command error

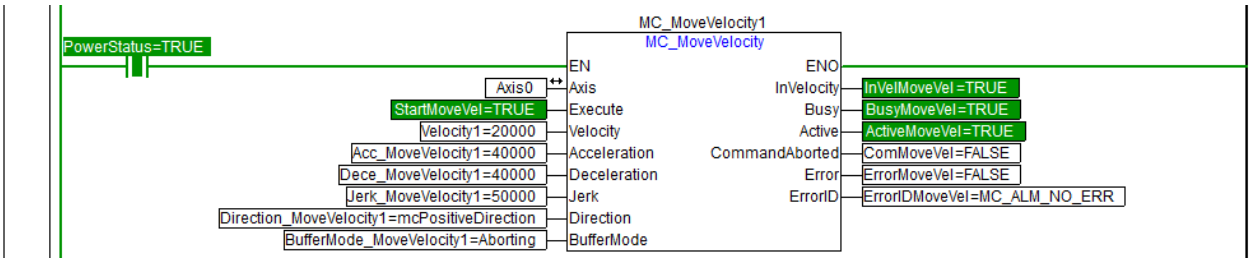
■ LD program implementation

Use the ladder diagram to build a program, and input parameters VelFactor, AccFactor and JerkFactor are all 300. Execute the MC_MoveVelocity command, set the MC_MoveVelocity as 20000, and control the axis to move. Set Enable of MC_SetOverride to TRUE, and the speed, acceleration/deceleration and jerk of the axis are all values after override. The example program is as follows:

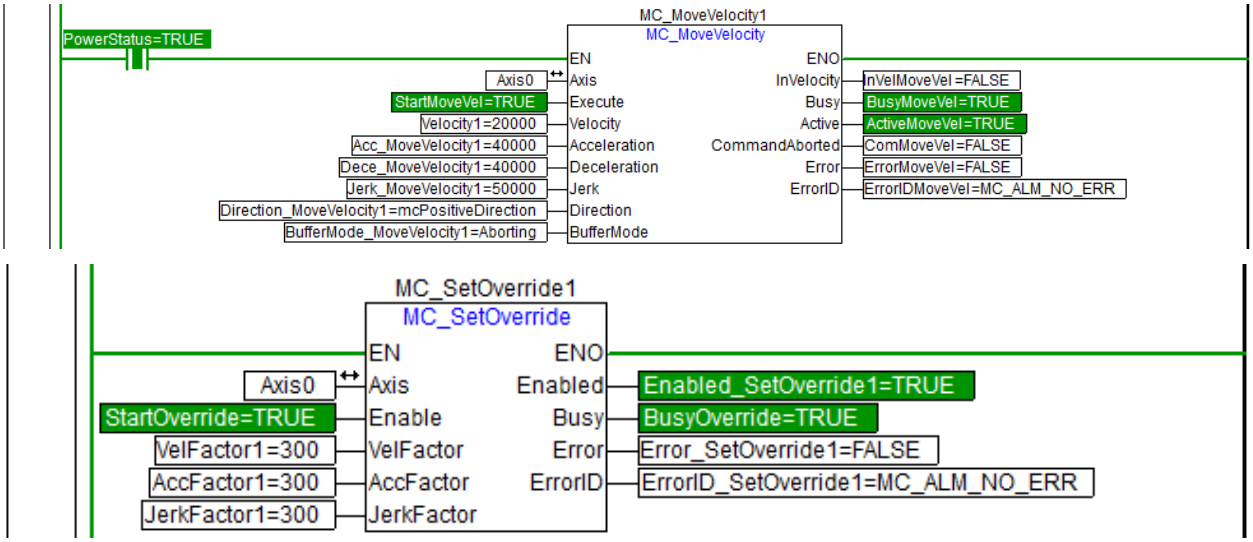
(1) Initialize and enable the axis. Set the axis ID to 0 and the axis type to EtherCAT_Position. After the axis initialization is completed, enable the axis.



(2) Call the MC_MoveVelocity instance to output the status of the pin variable. The InVelocity pin is TRUE, and the actual velocity instructed by MC_MoveVelocity reaches the set command velocity 20000.



(3) Keep the Enable pin of MC_SetOverride at high level and set the override parameter. It can be seen that after setting the parameter, the output pin variable InVelocity pin of the MC_MoveVelocity instance becomes FALSE for a period of time, and then becomes TRUE again when the speed is reached.



■ ST language program

(1) Initialize the axis number and axis type settings, and enable the axis after initialization.

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
```

```
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
(*Enable the axis*)
IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

(2) Trigger the MC_MoveVelocity command on a rising edge to control the axis motion.

```
MC_MoveVelocity1(
Execute := StartMoveVel,
Velocity := 20000,
Acceleration := 40000,
Deceleration := 40000,
Jerk := 50000,
Direction := 0,
BufferMode := 0,
Axis := Axis0,
InVelocity=> InVelMoveVel,
Busy=> BusyMoveVel,
Active=> ActiveMoveVel,
CommandAborted=> ComMoveVel,
Error=> ErrorMoveVel,
ErrorID=> ErrorIDMoveVel);
```

(3) Keep MC_SetOverride at high level to configure the motion override settings.

```
MC_SetOverride1(
Enable := StartOverride,
VelFactor := 300,
AccFactor := 300,
JerkFactor := 300,
```

```
Axis := Axis0,
Enabled=>EnabledOverride,
Busy=>BusyOverride,
Error=>ErrorOverride,
ErrorID=>ErrorID);
```

5.14.4 Precautions

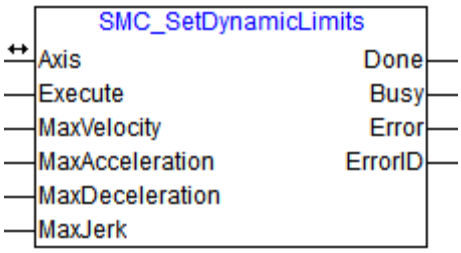
- When configuring override settings with this command, if the specified speed exceeds the axis's maximum speed, the axis will operate at its maximum speed.
- This command has no effect on the encoder axis.

5.14.5 Reference

- For axis initialization and enable, refer to the commands section of SMC_AxisInit and MC_Power .
- For error codes, please refer to the appendix Error Codes of Functional Block.

5.15 SMC_SetDynamicLimits (Set Axis Dynamic Parameter Limits)

This command sets the limit values of axis motion parameters.



5.15.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	MaxVelocity	Maximum speed	YES	1E100	Positive value/0	LREAL
	MaxAcceleration	Maximum acceleration	YES	1E100	Positive value	LREAL
	MaxDeceleration	Maximum deceleration	YES	1E100	Positive value	LREAL
	MaxJerk	Maximum jerk	YES	1E100	Positive value	LREAL
OUTPUT	Done	Completed	YES	FALSE	TRUE/FALSE	BOOL

	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.15.2 Functions

This command sets the limit values of axis motion parameters.

- command function details

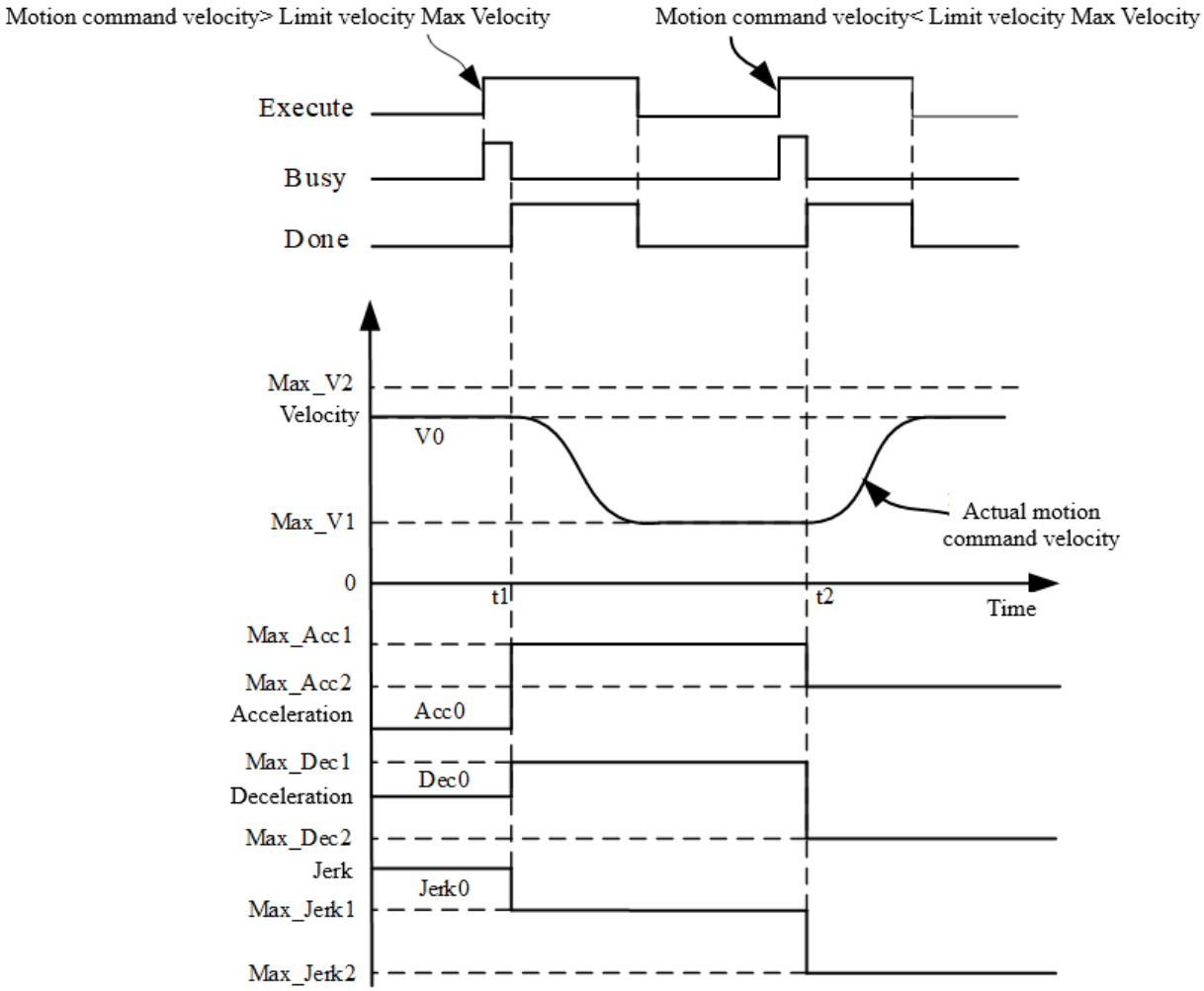
(1) Input parameter lock

The rising edge triggering of this command is valid. At the rising edge of the Execute input, the axis ID of the input/output parameter Axis and the values of the input parameters MaxVelocity, MaxAcceleration, MaxDeceleration and MaxJerk are latched. During command execution, the modified latched parameters are invalid until another rising edge triggers this command.

(2) Functional details

After setting the limit value of the axis motion parameter, the motion parameter of the axis motion command takes the minimum value of its own motion parameter and the set limit parameter. As shown in the figure below, execute this command twice, and set different values of MaxVelocity, MaxAcceleration, MaxDeceleration and MaxJerk each time.

Set the maximum motion parameters as shown in the following figure:



In the above figure, V0 is a command speed of the motion command, Acc0 is a command acceleration of the motion command, Dec0 is a command deceleration of the motion command, and Jerk0 is a command jerk of the motion command.

At time t1, the maximum speed limit MaxVelocity (Max_V1 value in the figure) is successfully set. If the maximum speed limit MaxVelocity is less than the axis motion command speed V0, the actual maximum speed of the axis motion command is the limit value MaxVelocity, and the axis motion command decelerates. Meanwhile, Max_Acc1, Max_Dec1 and Max_Jerk1 are the set limit values of other motion parameters respectively.

At time t2, the maximum speed limit MaxVelocity is successfully set again. MaxVelocity=Max_V2. MaxVelocity is greater than the speed V0 of axis motion command. The axis motion command takes the self-set command speed as the actual running speed, and the axis motion command accelerates. Meanwhile, Max_Acc2, Max_Dec2 and Max_Jerk2 are the set limit values of other motion parameters respectively.

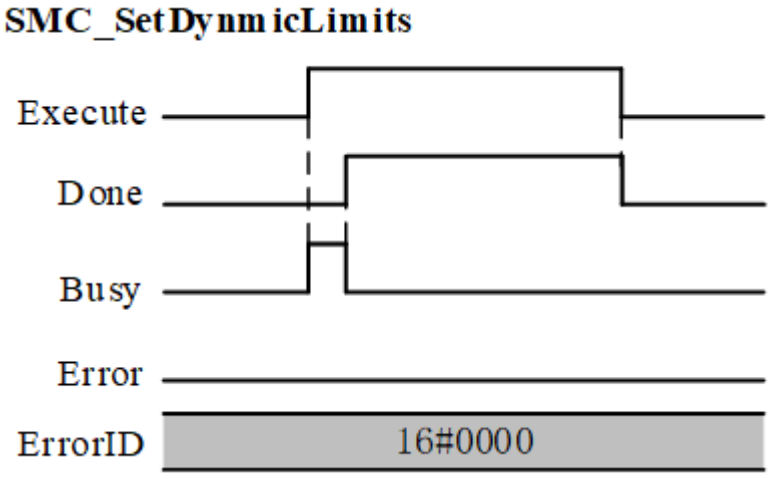
- Timing diagram

(1) Normal timing diagram

At the rising edge of Execute input, if no errors occur, the Busy pin will be set to TRUE. After the

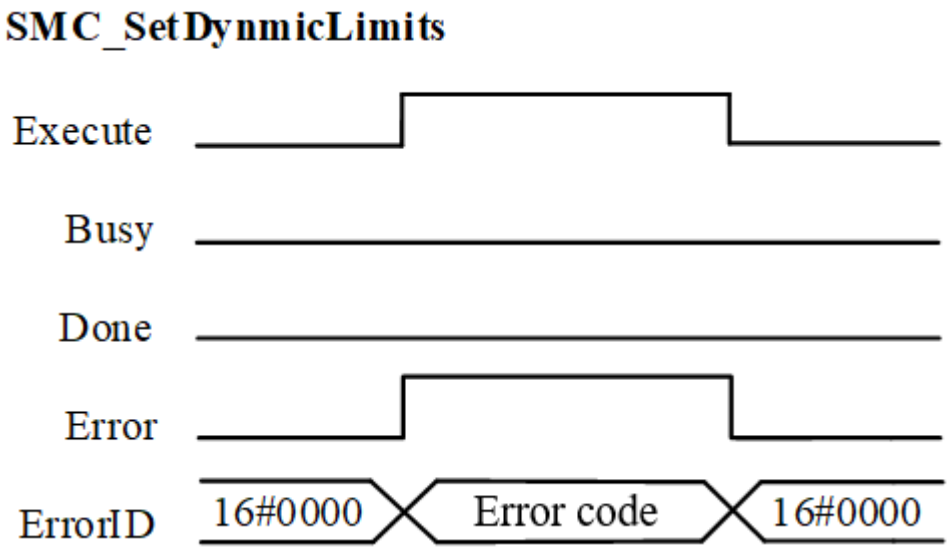
parameter setting is completed, the Done will be set to TRUE and Busy will be set to FALSE. If Execute keeps high level, the Done pin will remain true; if Execute is set low level, the Done pin will be set to FALSE.

The correct timing sequence for command execution is as follows:



(2) Error pin timing diagram

When executing this command, if the parameter is illegal and the axis type is illegal (the axis type is Unused), it will fail to call this command. The command output pin Error=True, and the error code ErrorID reports an abnormal error value. Please refer to the appendix Error Code Description to obtain the cause of the error. When the command input pin Execute goes low, all output pins revert to their Default Values.



- When the abnormal situation of the command is solved, it is necessary to trigger Execute again on the rising edge and start the SMC_SetDynamicLimits functional block to complete the setting of axis motion parameter limit.

5.15.3 Examples

Set up SMC_SetDynamicLimits example program to set the limit values of axis motion parameters.

The input parameter of SMC_SetDynamicLimits command MaxVelocity is 20000, and MaxAcceleration, MaxDeceleration and MaxJerk are all 50000. Execute the command MC_Movelocity, and set the speed to 50000 with both acceleration/deceleration and jerk of 100000. The control axis moves. Execute the SMC_SetDynamicLimits command, and the axis motion is limited by speed. The actual running speed is 20000, and both acceleration and deceleration are limited to 50000. The example procedure is as follows:

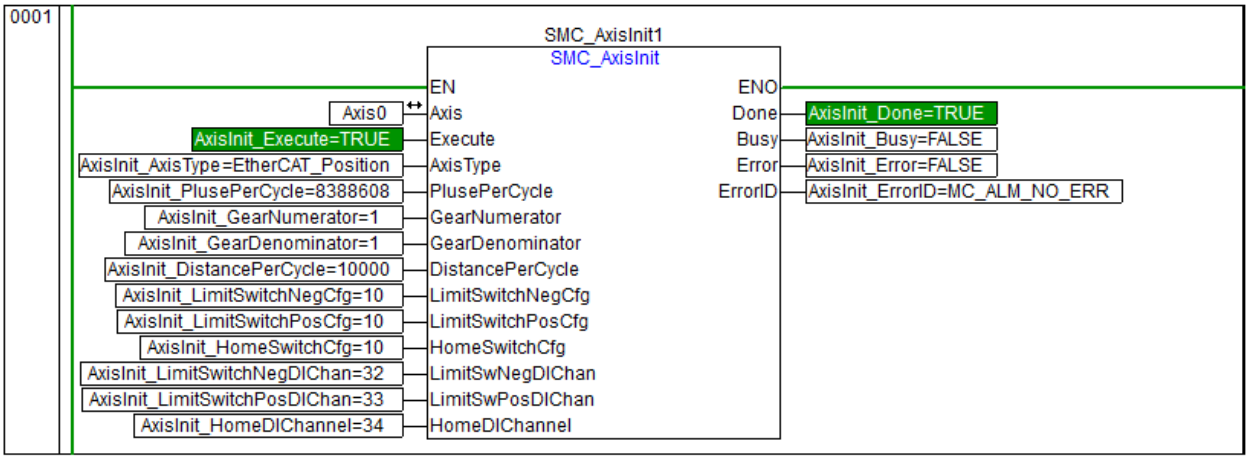
- LD program implementation

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
Axis0	AXIS_REF	-	Axis variable for axis 0
Power1_Status	BOOL	FALSE	Variable for which the axis enable is successful. This variable becomes TRUE when the axis is enabled successfully.
MV1_Vel	LREAL	-	Velocity variable of the input shaft speed control command. The commanded speed that controls the motion of the axis.
MV1_Active	BOOL	FALSE	Variable when the axis speed control command acts. This variable becomes TRUE when the axis is controlling motion.
MV1_InVel	BOOL	FALSE	Identification variable for the axis speed control command to reach the target speed. This variable becomes TRUE and the speed commanded reaches the target speed.
Switch1	BOOL	FALSE	Axis ready state variable. After axis initialization, Switch to TRUE and the axis is ready.

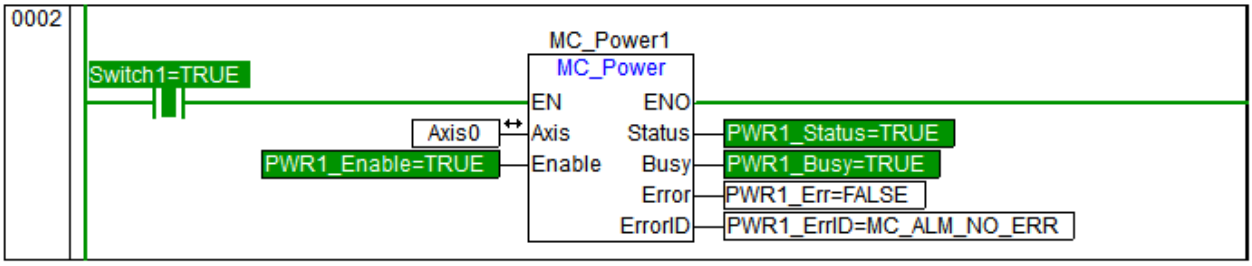
(1) Axis initialization

The AxisID of Axis0 is set to 0, the axis type is set to 50:EtherCAT_Position, and the axis initialization SMC_AxisInit command is called to complete the initial configuration of Axis0. For the usage of the command shown in SMC_AxisInit, please refer to its Introduction Chapter.



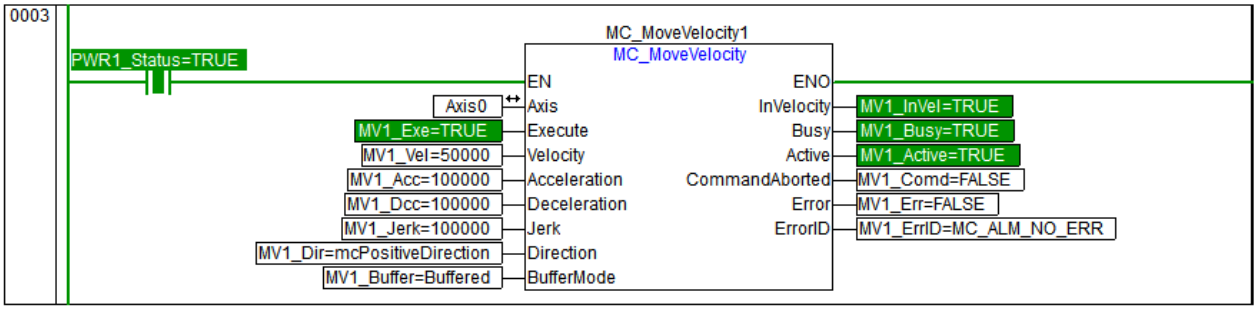
(2) Axis enable

After the axis initialization is completed and the axis ready state Switch1 is TRUE, call MC_Power to complete the axis enable.



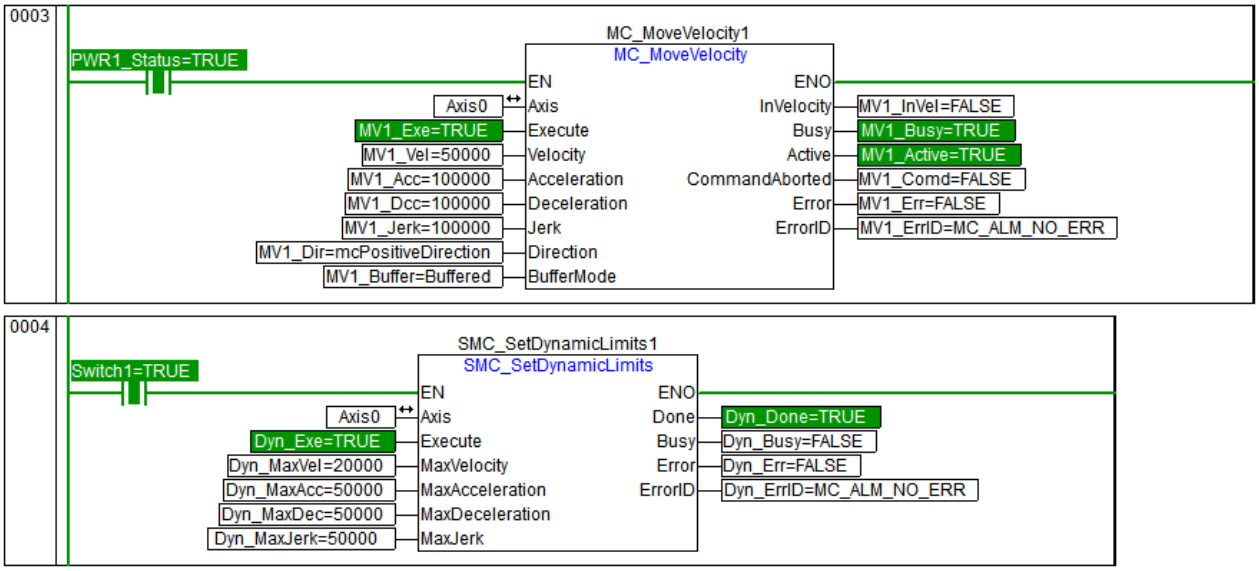
(3) Axis delivery motion command

In section 0003, after the axis is enabled successfully, the rising edge triggers the MC_MoveVelocity motion command to control the axis motion. The state of the pin variable is output through the MC_MoveVelocity instance. The InVelocity pin is TRUE, and the actual velocity of the MC_MoveVelocity command reaches the set command velocity MV1_Vel= 50000.



(4) Set motion parameter limits

In section 0004, trigger the Execute pin of SMC_SetDynamicLimits1 to complete the setting of motion parameter limit. In section 0003, the output pin variable InVelocity pin of the MC_MoveVelocity instance is FALSE, which can intuitively show that the speed of the MC_MoveVelocity command is limited, and the actual velocity of this command does not reach the set command velocity.



■ ST language

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
Axis0	AXIS_REF	-	Axis variable for axis 0
Power1_Status	BOOL	FALSE	Variable for which the axis enable is successful. This variable becomes TRUE when the axis is enabled successfully.
MV1_Vel	LREAL	-	Velocity variable of the input shaft speed control command. The commanded speed that controls the motion of the axis.
MV1_Active	BOOL	FALSE	Variable when the axis speed control command acts. This variable becomes TRUE when the axis is controlling motion.
MV1_InVel	BOOL	FALSE	Identification variable for the axis speed control command to reach the target speed. This variable becomes TRUE and the speed commanded reaches the target speed.
Switch1	BOOL	FALSE	Axis ready state variable. After axis initialization, Switch to TRUE and the axis is ready.
InitFlag	BOOL	FALSE	If this variable is TRUE, the initialization parameter setting is completed.

(1) Axis initialization

Call the SMC_AxisInit command to complete the initial configuration of Axis0. Set the axis type to 50: EtherCAT_Position. For the SMC_AxisInit commands, please refer to its Introduction Chapter.

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
```

```
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

(2) Initialize motion limit parameters

```
IF  FALSE = InitFlag THEN
MaxVel :=20000; (*speed limit *)
MaxAcc:=50000; (* acceleration limit *)
MaxDec:=50000; (*deceleration limit *)
MaxJerk:=50000; (*jerk limit*)
MV1_Vel:=50000; (*motion command speed*)
MV1_Acc:=100000; (*motion command speed*)
MV1_Dec:=100000; (*motion command speed*)
MV1_Jerk: =100000; (*motion command speed*)
   InitFlag:=TRUE;
END_IF
```

(3) Axis enable

After the axis initialization is completed and the axis ready state Switch1 is TRUE, call MC_Power to complete the axis enable.

```
IF  Switch1 THEN
   MC_Power1(
   Enable := PWR1_Enable,
   Axis := Axis0,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
   Error=> PWR1_Err
   ErrorID=> PWR1_ErrID);
END_IF
```

(4) Delivery motion command

After the axis is enabled successfully, the rising edge triggers the MC_MoveVelocity motion command to control the axis motion.

```
IF  PWR1_Status THEN
   MC_MoveVelocity1(
   Execute := MV1_ Exe,
   Velocity := MV1_ Vel,
   Acceleration := MV1_ Acc,
   Deceleration := MV1_ Dcc,
   Jerk := MV1_ Jerk,
   Direction := MV1_ Dir,
   BufferMode := MV1_ Buffer,
```

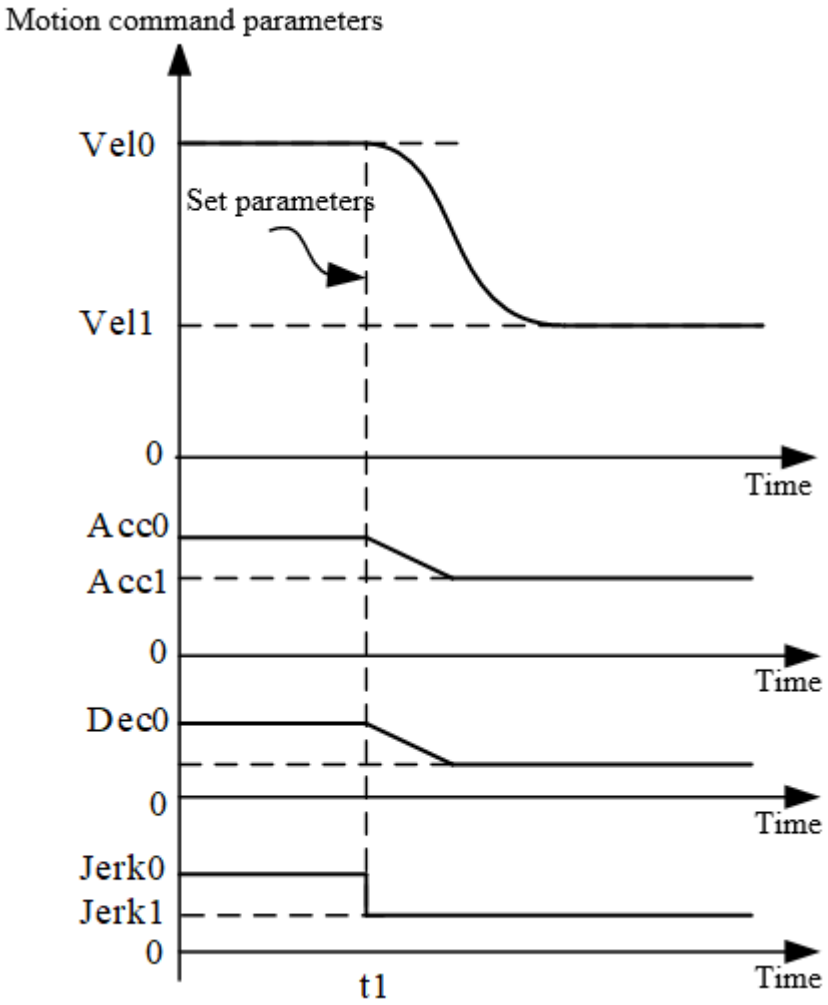
```
    Axis := Axis0,
    InVelocity=> MV1_ InVel,
    Busy=> MV1_ Busy,
    Active=> MV1_ Active,
    CommandAborted=> MV1_ Comd,
    Error=> MV1_ Err,
    ErrorID=> MV1_ Error);
END_IF
```

(5) Motion parameter limit setting

The rising edge triggers the SMC_SetDynamicLimits command to complete the setting of motion parameter limits.

```
IF  Switch1 THEN
    SMC_SetDynamicLimits1(
    Execute := Dyn_Exe,
    MaxVelocity := Dyn_MaxVel,
    MaxAcceleration := Dyn_MaxAcc,
    MaxDeceleration := Dyn_ MaxDec,
    MaxJerk := Dyn_MaxJerk,
    Axis := Axis0,
    Done=> Dyn_Done,
    Busy=> Dyn_Busy,
    Error=> Dyn_Err,
    ErrorID=> Dyn_ErrID);
END_IF
```

Observe the motion parameter curve of MC_MoveVelocity motion command. In the period from 0 to t1, the commanded motion parameters are not limited. Observe the velocity curve of motion command, and the actual running speed of motion command reaches the commanded speed set by MC_MoveVelocity; After t1, the maximum speed set by SMC_SetDynamicLimits command is lower than the command speed of MC_MoveVelocity motion command. The schematic diagram of axis motion parameter being limited is as follows:



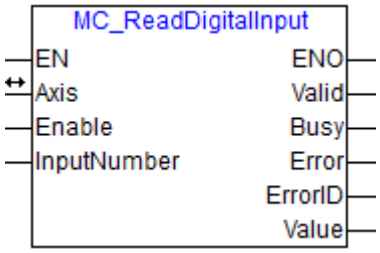
- Vel0, Acc0, Dec0 and Jerk0 are the motion speed, acceleration, deceleration and jerk of the motion command respectively.
- Vel1, Acc1, Dec1 and Jerk1 are motion speed, acceleration, deceleration and jerk after setting limit parameters respectively

5.15.4 Precautions

- When this command is called, only MaxVelocity can be set to 0, and the remaining input motion limit parameters need to be positive.
- After the maximum parameter is successfully set by this command, the falling edge cannot cancel the maximum parameter limit. The user needs to call this command again to set the parameter limit value.

5.16 MC_ReadDigitalInput (Read Axis Digital Input)

Read the Digital Input of the servo equipment.



5.16.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Enable	Enable	No	-	TRUE/FALSE	BOOL
	InputNumber	Input code, indicating the bit number in Digtal inputs object dictionary	No	-	0 ~ 31	INT
OUTPUT	Valid	Output valid	No	-	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR
	Value	Reading results	No	-	TRUE/FALSE	BOOL

5.16.2 Functions

This command follows the CIA402 standard protocol and reads the Digital Input of servo equipment. The detailed description of this command is as follows:

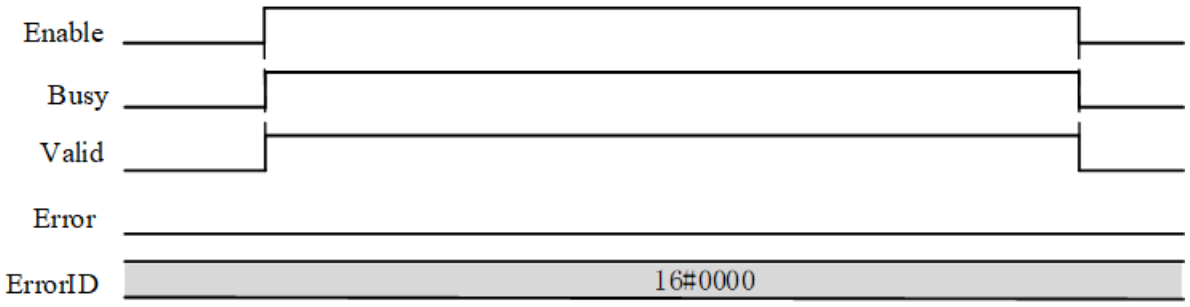
(1) This command is valid at a high level. After the axis initialization is completed, input the Input number of the function block to read Digital Input.

(2) When Enable is TRUE, the function block continuously reads Digital Input, and when Enable is FALSE, the function block pin returns to the Default Value.

■ Function block timing diagram

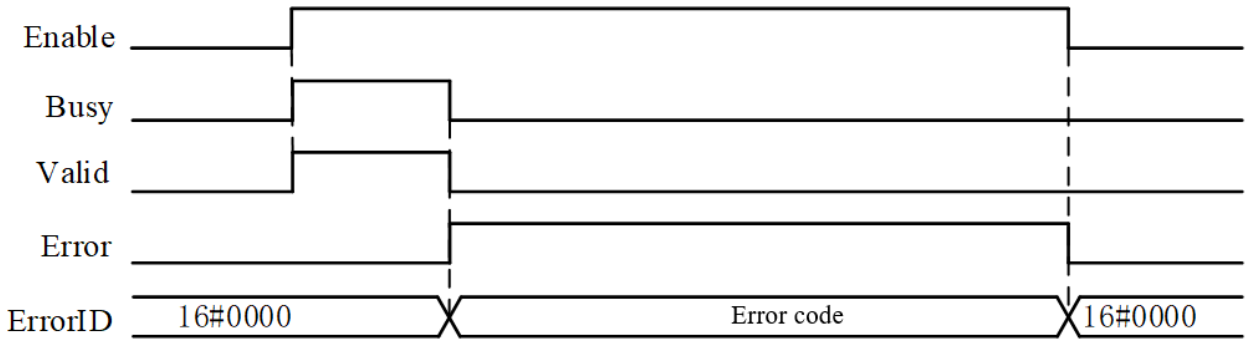
When Enable is TRUE, Busy and Valid are immediately TRUE. If Digital Input is valid, Value (read result) is TRUE. When Enable is FALSE, the function block pin returns to its Default Value.

(1) Normal timing diagram



(2) Error timing diagram

When an error occurs, Error is TRUE and the other pins are FALSE.



5.16.3 Examples

Set up MC_ReadDigitalInput sample program to read the Digital Input value of servo equipment.

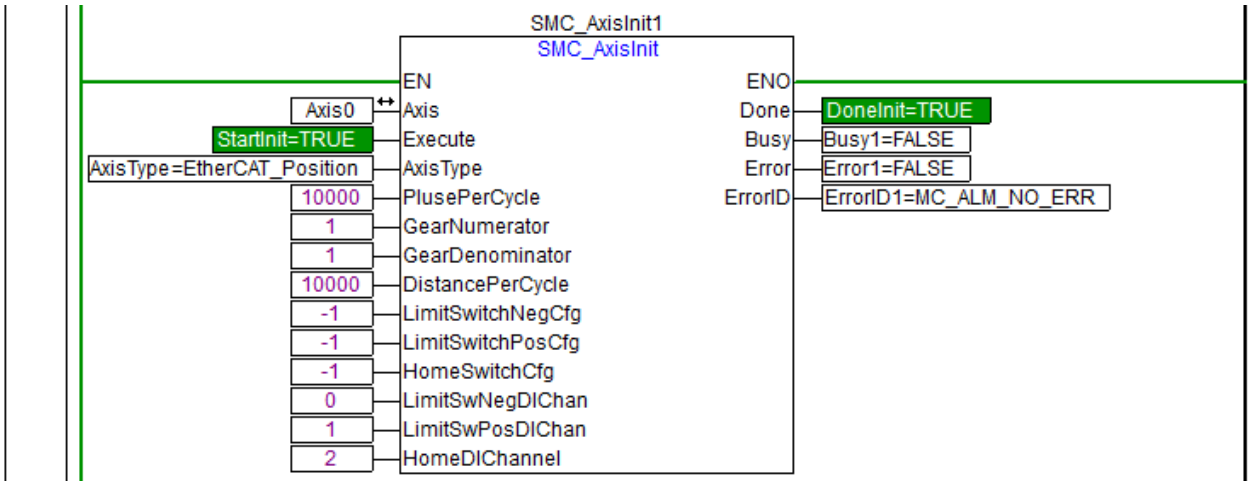
Description of Primary Variables

Variable Name	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	becomes TRUE at the start of initialization
DoneInit	BOOL	FALSE	Changes to TRUE upon completion of initialization
ReadInputEnable	BOOL	FALSE	Changes to TRUE at the beginning of reading
ValidReadInput	BOOL	FALSE	Changes to TRUE when reading is successful
BusyReadInput	BOOL	FALSE	Changes to TRUE on reading
ValueReadInput	BOOL	FALSE	Changes to TRUE after reading the value
ErrorReadInput	BOOL	FALSE	Changes to TRUE when an error occurs

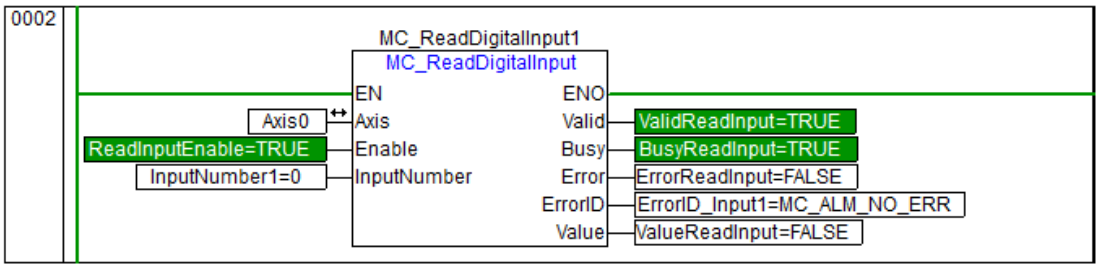
■ LD program implementation

Use the ladder diagram to build a program and input the corresponding object dictionary number. Set Enable of MC_ReadDigitalInput to TRUE. The sample procedure is as follows:

(1) Before reading the Digital Input of the axis, initialize the axis first.



(2) After initialization, enter the corresponding object dictionary digits to read Digital Input.



■ ST language program

(1) Initialize the axis number and axis type settings

```
ST language program
Initialize the axis number and axis type settings
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

(2) Read Axis Digital Input

```
MC_ReadDigitalInput1(
Enable := ReadInputEnable,
InputNumber := 0,
Axis := Axis0,
Valid=>ValidReadInput,
Busy=>BusyReadInput,
Error=>ErrorReadInput,
ErrorID=>ErrorID_Input1,
Value=>ValueReadInput);
```

5.16.4 Precautions

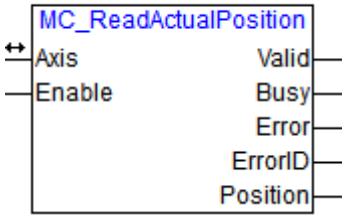
- This command does not support virtual axes, but supports EtherCAT bus axes.
- Before using this command, it is necessary to map the PDO (0x60FD) corresponding to Digtial Inputs.

5.16.5 Reference

- Refer to Initialization Chapter for details about the initialization.
- If an error occurs, please refer to the appendix Introduction of Function Block Error Code Description.

5.17 MC_ReadActualPosition (Read Axis Actual Position)

This command reads the actual position of the axis.



5.17.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Enable	Enable	No	-	TRUE/FALSE	BOOL

OUTPUT	Valid	Output valid	No	-	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR
	Position	Reading results	No	-	Positive/negative/0	LREAL

5.17.2 Functions

This command reads the actual position of the axis.

■ Command function details

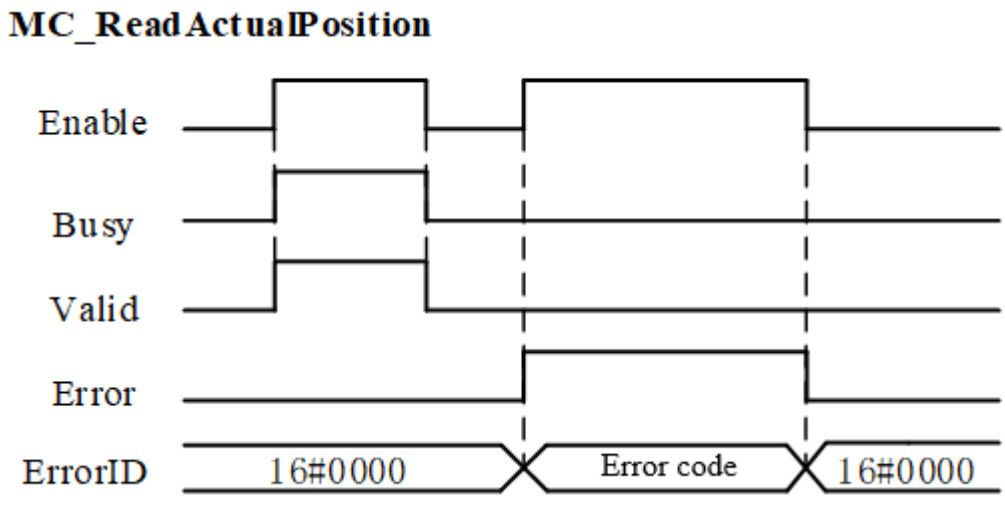
(1) This command is valid at high level. When the input parameters are legal and there is no abnormality in the axis, set Enable as high level TRUE, and set the output parameter Valid to high level, which is valid. The command continuously reads the actual position of the axis.

(2) The axis types supported by this command are:

- □ Virtual axis (0): Virtual axis
- □ EtherCAT_Position(50): EtherCAT bus connecting shaft, position control
- □ EtherCAT_Speed(51): EtherCAT bus connecting shaft, speed control
- □ EtherCAT_Torque(52): EtherCAT bus connecting shaft, torque control

■ Timing diagram

The following is the normal and error timing diagram during command execution.



● When the command enable pin is at high level and there is no abnormal error, the Busy and Valid pins are immediately set to TRUE to read out the actual position of the axis.

● When an abnormal error occurs during the execution of a command, the command output pins Busy and Valid are set to FLASE, the output result Position remains the current value, Error is set to True, and

the error code ErrorID reports an abnormal error value. Please refer to the appendix Error Code Description to obtain the cause of the error. The command input Enable goes low, so the output pin reverts to its Default Value.

5.17.3 Examples

Run the command MC_Movelocity to keep the control axis moving, and call MC_ReadActualPosition to read the actual position of the axis.

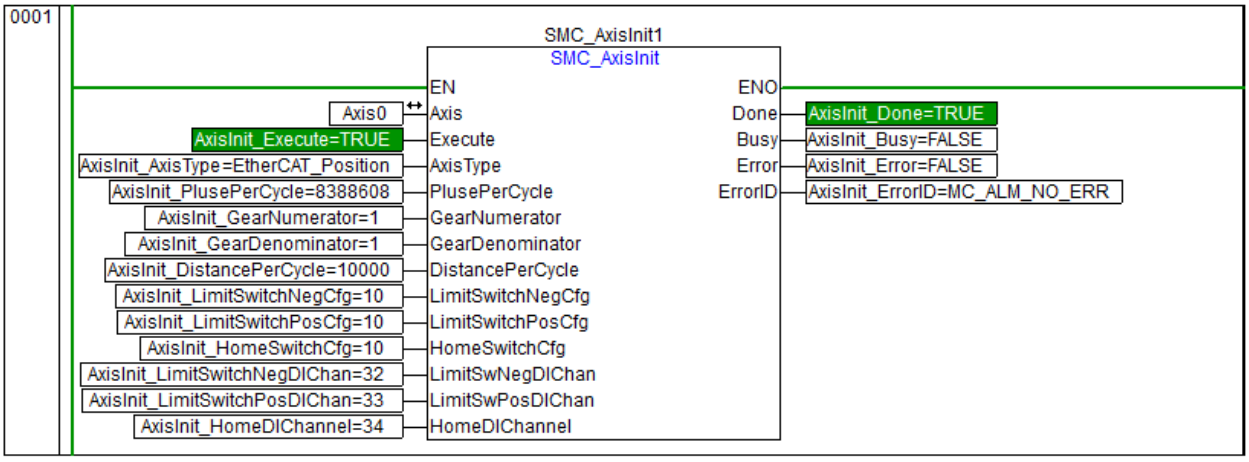
■ LD program implementation

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
Axis0	AXIS_REF	-	Axis variable for axis 0
Power1_Status	BOOL	FALSE	Variable for which the axis enable is successful. This variable becomes TRUE when the axis is enabled successfully.
ReadActPos_Valid	BOOL	FALSE	The axis reads a variable when the actual position is active. This variable becomes TRUE when the axis position is successfully read.
ReadActPos_Pos	LREAL	-	The axis reads a variable of the actual position.
Switch1	BOOL	FALSE	The state variable of axis readiness. If Switch is TRUE, the axis is ready.

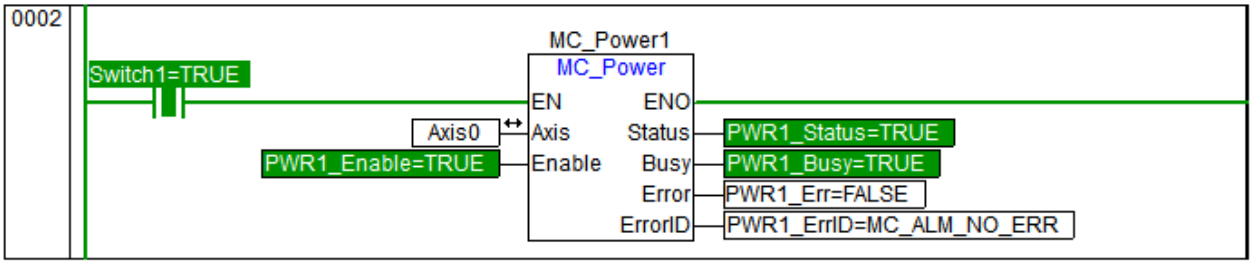
(1) Axis initialization

The AxisID of Axis0 is set to 0, the axis type is set to 50:EtherCAT_Position, and the axis initialization SMC_AxisInit command is called to complete the initial configuration of Axis0. For the usage of SMC_AxisInit command, please refer to its introduction chapter.

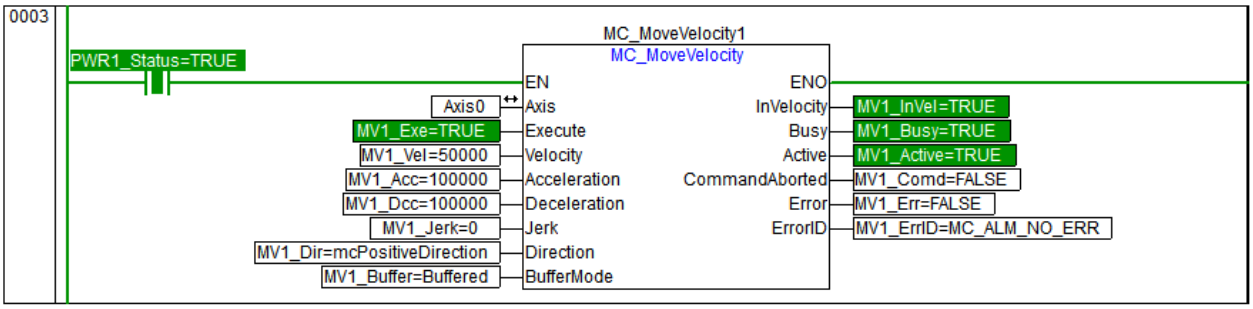


(2) Axis enable

After the axis initialization is completed and the axis ready state Switch1 is TRUE, call MC_Power to complete the axis enable.

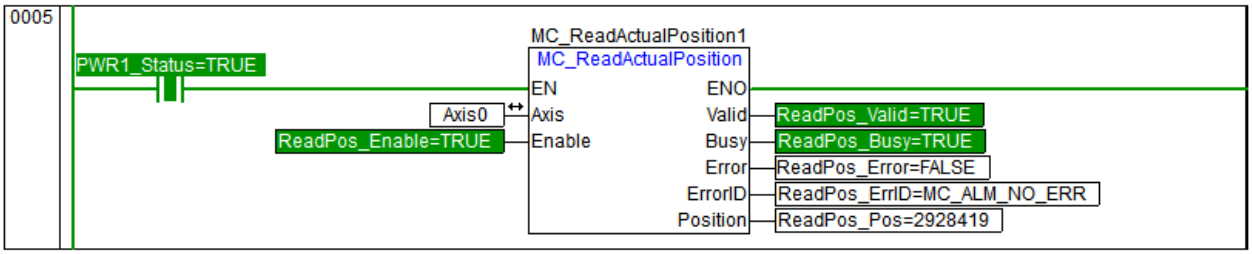


(3) In section 0003, after the axis is enabled successfully, the rising edge triggers the MC_MoveVelocity motion command to control the axis motion.



(4) Execute the reading axis position command

In section 0005, execute the MC_ReadActualPosition command to start reading the actual position of the axis. ReadPos_Valid is TRUE, indicating that the reading position is valid, and ReadPos_Pos is the read actual position of the axis.



■ ST language program

(1) The primary variables of ST language program are the same as those of LD language program.

(2) Set the axis number of Axis0

```
Axis0.AxisID:=0;
```

(3) Axis initialization

Call SMC_AxisInit command to complete the initial configuration of Axis0, and set the axis type as 50:EtherCAT_Position. For the usage of SMC_AxisInit commands, please refer to its Description section.

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
```

```
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

(4) Axis enable

After the axis initialization is completed and the axis ready state Switch1 is TRUE, call MC_Power to complete the axis enable.

```
IF  Switch1 THEN
   MC_Power1(
   Enable := PWR1_Enable,
   Axis := Axis0,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
   Error=> PWR1_Err
   ErrorID=> PWR1_ErrID);
END_IF
```

(5) After the axis is enabled successfully, execute the MC_MoveVelocity motion command to control the axis motion.

```
IF  PWR1_Status THEN
   MC_MoveVelocity1(
   Execute := MV1_ Exe,
   Velocity := MV1_ Vel,
   Acceleration := MV1_ Acc,
   Deceleration := MV1_ Dcc,
   Jerk := MV1_ Jerk,
   Direction := MV1_ Dir,
   BufferMode := MV1_ Buffer,
   Axis := Axis0,
   InVelocity=> MV1_ InVel,
   Busy=> MV1_ Busy,
   Active=> MV1_ Active,
   CommandAborted=> MV1_ Comd,
   Error=> MV1_ Err,
   ErrorID=> MV1_ Error);
END_IF
```

(6) Execute the MC_ReadActualPosition command to read the actual axis position.

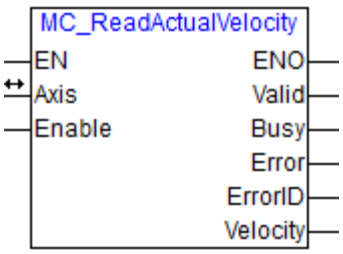
```
IF  PWR1_Status THEN
   MC_ReadActualPosition1(
   Enable := ReadPos_Enable,
   Axis := Axis0,
   Valid=> ReadPos_Valid,
   Busy=> ReadPos_Busy,
   Error=> ReadPos_Err,
   ErrorID=> ReadPos_ErrID,
   Position=> ReadPos_Pos);
END_IF
```

5.17.4 Precautions

When an error occurs during the execution of commands, the output result Position remains at the current value.

5.18 MC_ReadActualVelocity (Read Axis Actual Speed)

Read the actual velocity of the axis.



5.18.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Enable	Enable	No	-	TRUE/FALSE	BOOL
OUTPUT	Valid	Output valid	No	-	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR
	Velocity	Reading results	No	-	Positive/negative/0	LREAL

5.18.2 Functions

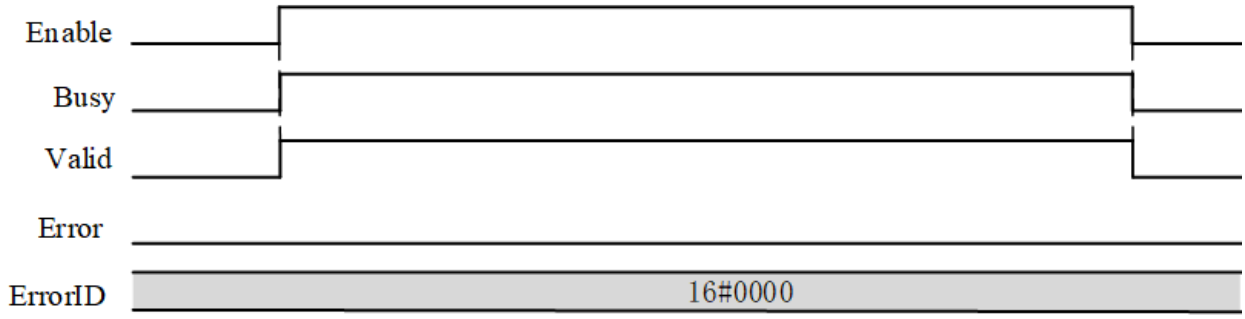
This command reads the actual velocity of the axis. The details of this command are as follows:

(1) This command is valid at high level. After the axis initialization is completed, the actual velocity can be read directly.

(2) When Enable is TRUE, the function block continuously reads the actual velocity, and when Enable is FALSE, the function block pin returns to the Default Value.

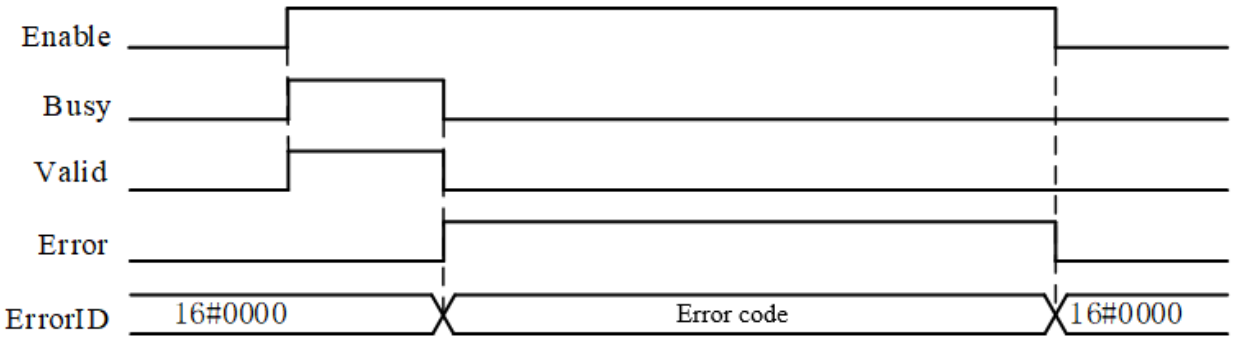
■ Function block timing diagram

(1) Normal timing diagram



(2) Error timing diagram

When an error occurs, Error is TRUE and the other pins are FALSE.



5.18.3 Examples

Set up MC_ReadActualVelocity sample program to read the actual velocity of the axis.

Description of Primary Variables

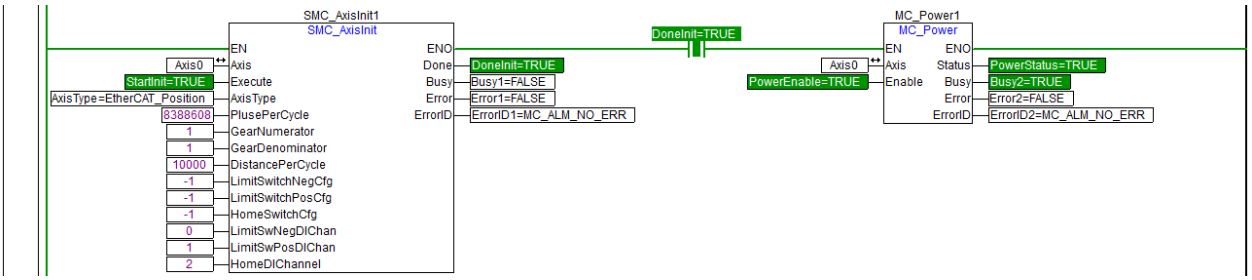
Title	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	Change to TRUE at the start of initialization
DoneInit	BOOL	FALSE	Change to TRUE upon completion of initialization

ReadVelEnable	BOOL	FALSE	Change to TRUE at the beginning of reading
ValidReadVel	BOOL	FALSE	Change to TRUE when reading is successful
BusyReadVel	BOOL	FALSE	Change to TRUE on reading
ValueReadVel	LREAL		Actual velocity read
ErrorReadVel	BOOL	FALSE	Change to TRUE when an error occurs

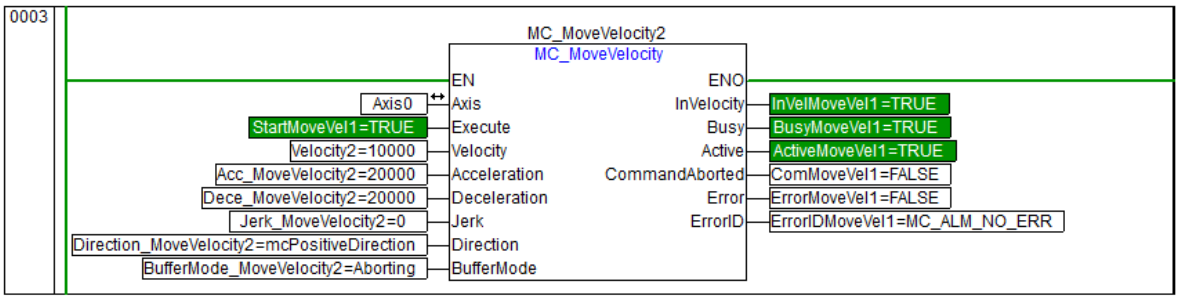
■ LD program implementation

Use the ladder diagram to build a program and input corresponding object dictionary changes. To set MC_ReadActualVelocity Enable to TRUE, the sample procedure is as follows:

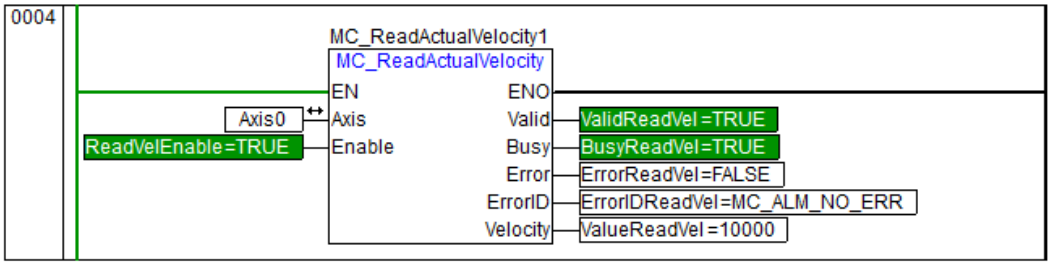
(1) First, initialize the axis and set its type and various parameters. After the initialization is completed, enable the axis. When it enters the ON state of servo enabling, motion control can be realized for the axis.



(2) Run the MC_MoveVelocity command, input Velocity as 10000 and control the axis to move.



(3) Call the MC_ReadActualVelocity command to read its velocity, and you can see the actual velocity value.



■ ST language program

(1) Initialize the axis number and axis type settings

```
SMC_AxisInit1(
```

```
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
(*Enable the axis*)
IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

(2) Input motion related parameters and start the speed control command

```
(*Start speed control command*)
MC_MoveVelocity1(
Execute := StartMoveVel,
Velocity := 10000,
Acceleration := 20000,
Deceleration := 20000,
Jerk := 0,
Direction := 0,
BufferMode := 0,
Axis := Axis0,
InVelocity=>InVelMoveVel,
Busy=>BusyMoveVel,
Active=>ActiveMoveVel,
CommandAborted=>ComMoveVel,
Error=>ErrorMoveVel,
ErrorID=>ErrorIDMoveVel);
```

(3) Read the actual velocity of the axis

```
MC_ReadActualVelocity1(
Enable := ReadVelEnable,
Axis := Axis0,
Valid=>ValidReadVel,
Busy=>BusyReadVel,
Error=>ErrorReadVel,
ErrorID=>ErrorIDReadVel,
Velocity=>ValueReadVel);
```

5.18.4 Precautions

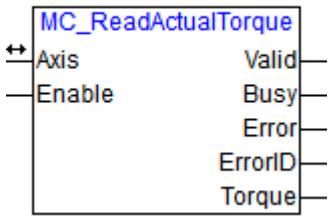
- Over-limit axis number error will be reported.
- This command supports virtual axis and Ethercat bus axis.

5.18.5 Reference

- For MC_MoveVelocity and other related motion commands, please refer to the chapter of MC_MoveVelocity and Other Motion Commands.
- For detailed errors related to ErrorID, refer to the appendix Function Block Error Codes.

5.19 MC_ReadActualTorque (Read Axis Actual Torque)

This command reads the torque value of a servo device conforming to the CIA402 standard.



5.19.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Enable	Enable	No	-	TRUE/FALSE	BOOL
OUTPUT	Valid	Output valid	No	-	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL

	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR
	Torque	Reading results Unit: 1% rated torque	No	-	Positive/negative/0	LREAL

5.19.2 Functions

This command reads the torque value of servo equipment conforming to CIA402 standard.

■ Command function details

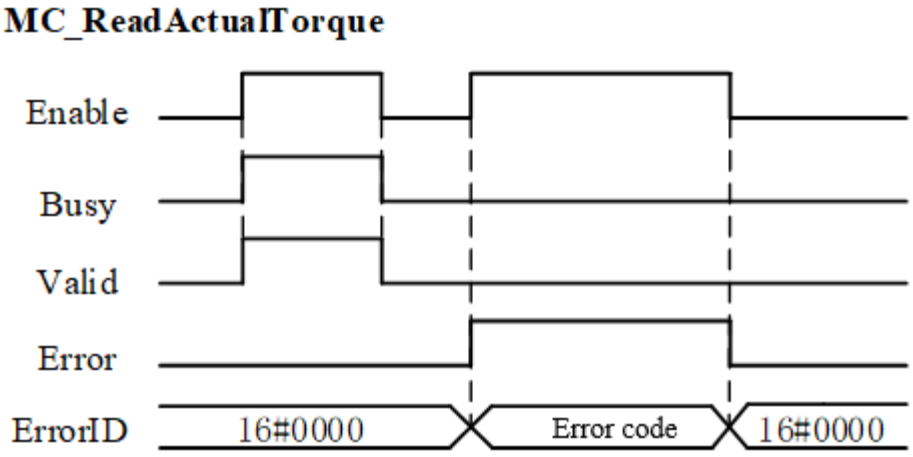
(1) This command is valid at high level. When the input parameters are legal and there is no error in the axis, set Enable as high level TRUE, set the output parameter Valid to high level, which is valid. The command continuously reads the torque of the servo equipment, and the result read by the command is 1% of the rated torque.

(2) The axis types supported by this command are:

- □ EtherCAT_Position(50): EtherCAT bus connecting shaft, position control
- □ EtherCAT_Speed(51): EtherCAT bus connecting shaft, speed control
- □ EtherCAT_Torque(52): EtherCAT bus connecting shaft, torque control

■ Timing diagram

The following is the normal and error timing diagram during command execution.



- When the command enable pin is at high level and there is no abnormal error, the Busy and Valid pins are immediately set to TRUE, and Torque reads the torque value of servo equipment.
- When an abnormal error occurs during the execution of a command, the command output pins Busy and Valid are set to FLASE, the output result Torque is 0, Error is set to True, and the error code ErrorID reports an abnormal error value. Please refer to the appendix Error Code Description to obtain the cause of the error. The command input Enable goes low, so the output pin reverts to its Default Value.

5.19.3 Examples

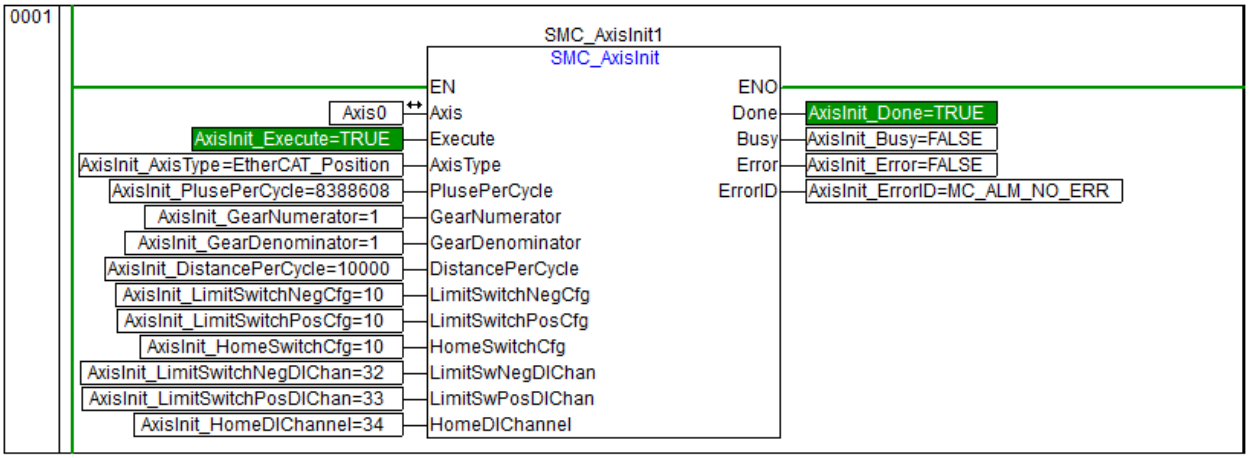
■ LD program implementation

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
Axis0	AXIS_REF	-	Axis variable for axis 0
Power1_Status	BOOL	FALSE	Variable for which the axis enable is successful. This variable becomes TRUE when the axis is enabled successfully.
ReadTor_Valid	BOOL	FALSE	The axis reads the variable when the actual torque is active. This variable becomes TRUE when the shaft actual torque is successfully read.
ReadTor_Tor	LREAL	-	The axis reads the variable of the actual torque. The output is the actual torque of the read shaft in 1% rated torque.
Switch1	BOOL	FALSE	The state variable of axis readiness. If Switch is TRUE, the axis is ready.

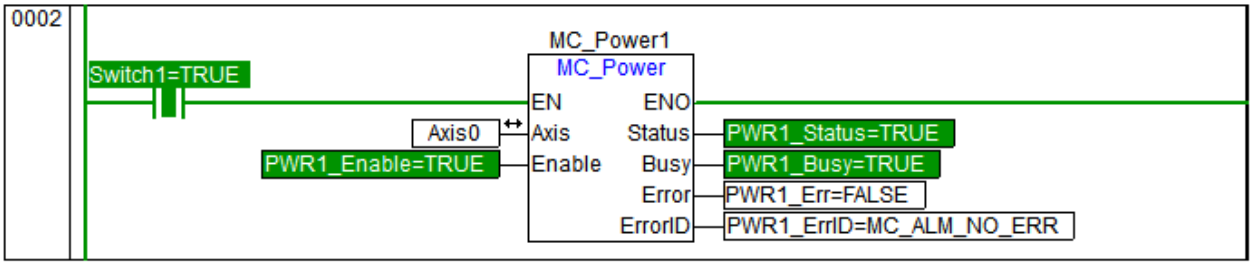
(1) Axis initialization

The AxisID of Axis0 is set to 0, the axis type is set to 50:EtherCAT_Position, and the axis initialization SMC_AxisInit command is called to complete the initial configuration of Axis0. For the SMC_AxisInit command, please refer to its Introduction Chapter.

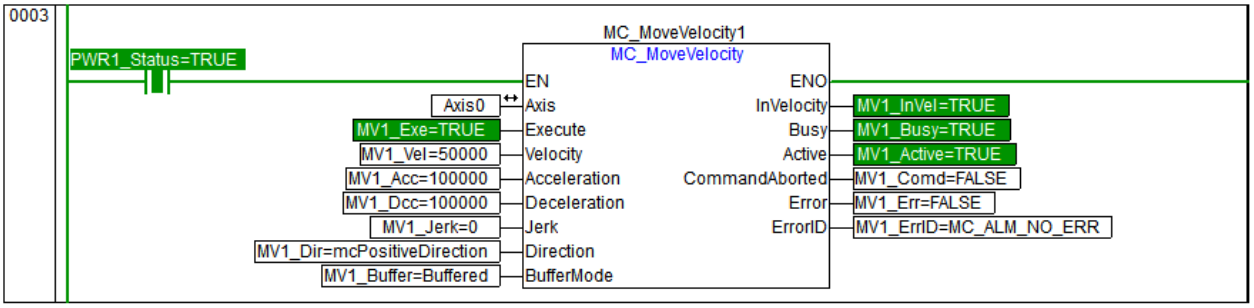


(2) Axis enable

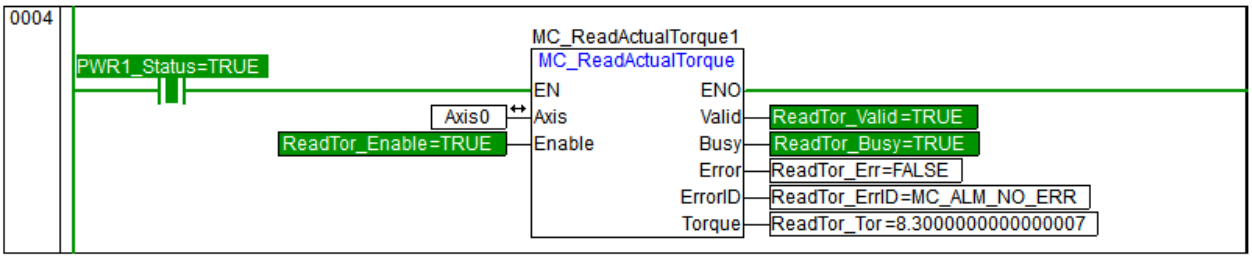
After the axis initialization is completed and the axis ready state Switch1 is TRUE, call MC_Power to complete the axis enable.



(3) In section 0003, after the axis is enabled successfully, the rising edge triggers the MC_MoveVelocity motion command to control the axis motion.



(4) In section 0004, the high level triggers the Enable pin of MC_ReadActualTorque to start reading the actual torque of the axis. ReadTor_Valid is TRUE, the reading torque is valid, and ReadPos_Tor is 1% of rated torque.



■ ST language program

The primary variables of ST language program are the same as those of LD language program.

(1) Set the axis number of Axis0

Axis0.AxisID:=0;

(2) Axis initialization

Call SMC_AxisInit command to complete the initial configuration of Axis0, and set the axis type as 50:EtherCAT_Position. For the SMC_AxisInit commands, please refer to its introduction.

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
```

```
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

(3) Axis enable

After the axis initialization is completed and the axis ready state Switch1 is TRUE, call MC_Power to complete the axis enabling.

```
IF  Switch1 THEN
   MC_Power1(
   Enable := PWR1_Enable,
   Axis := Axis0,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
   Error=> PWR1_Err
   ErrorID=> PWR1_ErrID);
END_IF
```

(4) After the axis is enabled successfully, the rising edge triggers the MC_MoveVelocity motion command to control the axis motion.

```
IF  PWR1_Status THEN
   MC_MoveVelocity1(
   Execute := MV1_ Exe,
   Velocity := MV1_ Vel,
   Acceleration := MV1_ Acc,
   Deceleration := MV1_ Dcc,
   Jerk := MV1_ Jerk,
   Direction := MV1_ Dir,
   BufferMode := MV1_ Buffer,
   Axis := Axis0,
   InVelocity=> MV1_ InVel,
   Busy=> MV1_ Busy,
   Active=> MV1_ Active,
   CommandAborted=> MV1_ Comd,
   Error=> MV1_ Err,
   ErrorID=> MV1_ Error);
END_IF
```

(5) The high level triggers the MC_ReadActualTorque command to read the actual torque of the axis.

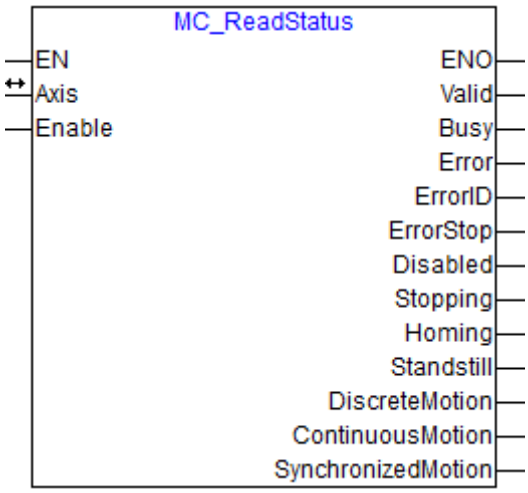
```
IF  MC_Power1. Status THEN
   MC_ReadActualTorque1(
   Enable := ReadTor_Enable,
   Axis := Axis0,
   Valid=> ReadTor_Valid,
   Busy=> ReadTor_Busy,
   Error=> ReadTor_Err,
   ErrorID=> ReadTor_ErrID,
   Torque=> ReadTor);
END_IF
```

5.19.4 Precautions

- The servo equipment calling the command must be a servo equipment conforming to CIA402 standard.
- The command reads the PDO (0x60FD) corresponding to the feedback torque (16#6077) of servo equipment, so the slave station of servo equipment needs to be configured with relevant PDO.

5.20 MC_ReadStatus (Read Axis Status)

Read the status of the axis PLCopen status machine.



5.20.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type

IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Enable	Enable	No	-	TRUE/FALSE	BOOL
OUTPUT	Valid	Output valid	No	-	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR
	ErrorStop	When TRUE, the axis is in ErrorStop state	YES	FALSE	TRUE/FALSE	BOOL
	Disabled	Axis Disabled when TRUE	YES	FALSE	TRUE/FALSE	BOOL
	Stopping	When TRUE, the axis is in Stopping state	YES	FALSE	TRUE/FALSE	BOOL
	Homing	When TRUE, the axis is in Homing state	YES	FALSE	TRUE/FALSE	BOOL
	Standstill	Standstill when TRUE	YES	FALSE	TRUE/FALSE	BOOL
	DiscreteMotion	Indicate that the axis is in DiscreteMotion when TRUE	YES	FALSE	TRUE/FALSE	BOOL
	ContinuousMotion	Indicate that the axis is in ContinuousMotion when TRUE	YES	FALSE	TRUE/FALSE	BOOL
	SynchronizedMotion	Indicate that the axis is in SynchronizedMotion when TRUE	YES	FALSE	TRUE/FALSE	BOOL

5.20.2 Functions

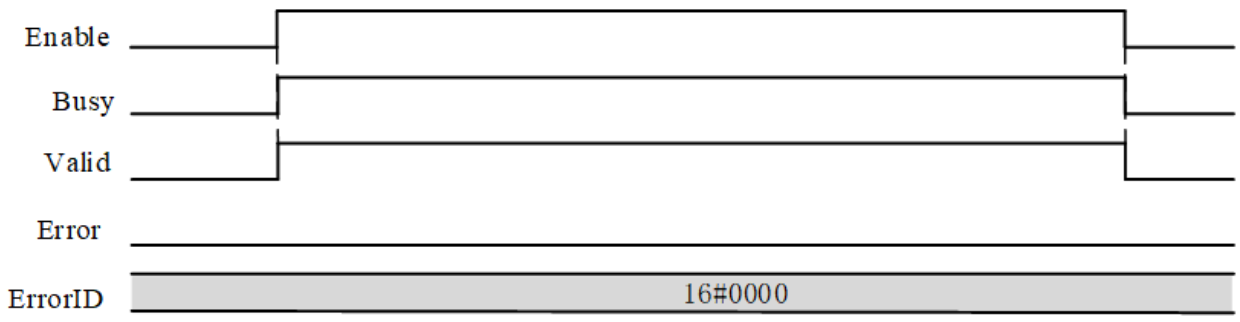
This command reads the status of the axis PLCopen status machine. The detailed Description of this command is as follows:

(1) This command is valid at high level. After the axis initialization is completed, the state of the axis PLCopen status machine can be read directly.

(2) When Enable is TRUE, the function block continuously reads the status of the axis PLCopen status machine. When Enable is FALSE, the function block pin returns to the Default Value.

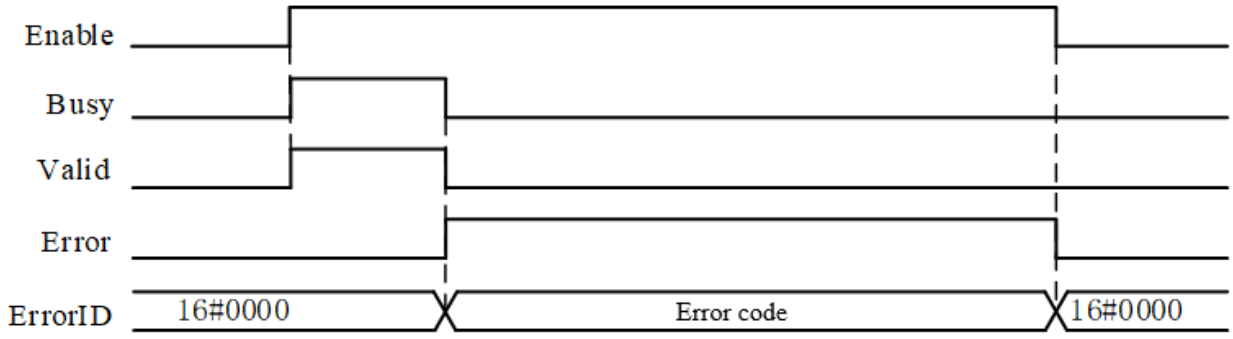
■ Function block timing diagram

(1) Normal timing diagram



(2) Error timing diagram

When an error occurs, Error is TRUE and the other pins are FALSE.



5.20.3 Examples

Set up an MC_ReadStatus example program to read the status of the axis PLCopen status machine.

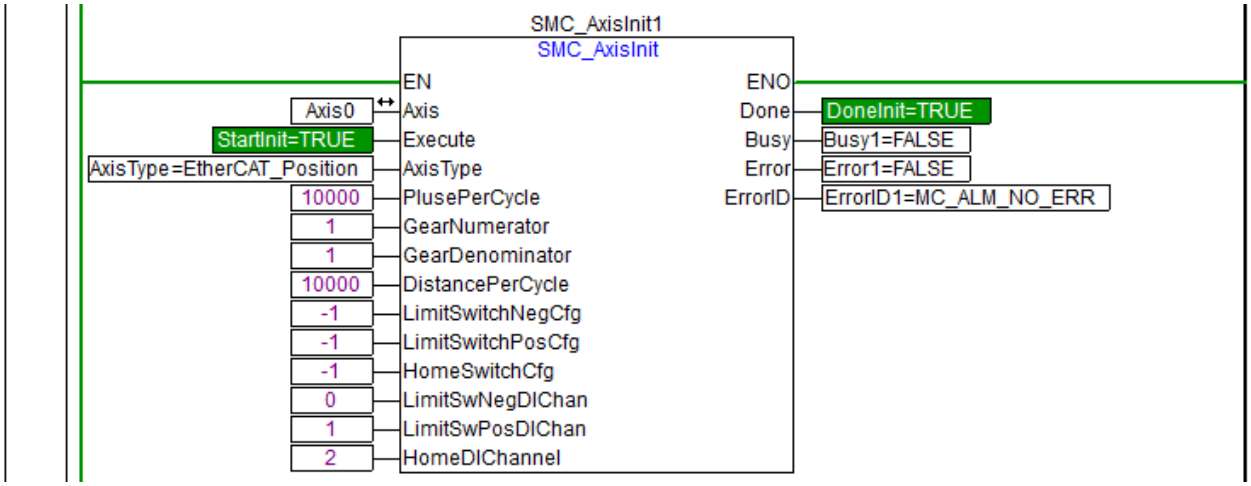
- Primary variables

Variable Name	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	Change to TRUE at the start of initialization
DoneInit	BOOL	FALSE	Change to TRUE upon completion of initialization
ReadStatusEnable	BOOL	FALSE	Change to TRUE at the beginning of reading
ValidReadStatus	BOOL	FALSE	Change to TRUE when reading is successful
BusyReadStatus	BOOL	FALSE	Change to TRUE on reading
ErrorReadStatus	BOOL	FALSE	Change to TRUE when an error occurs

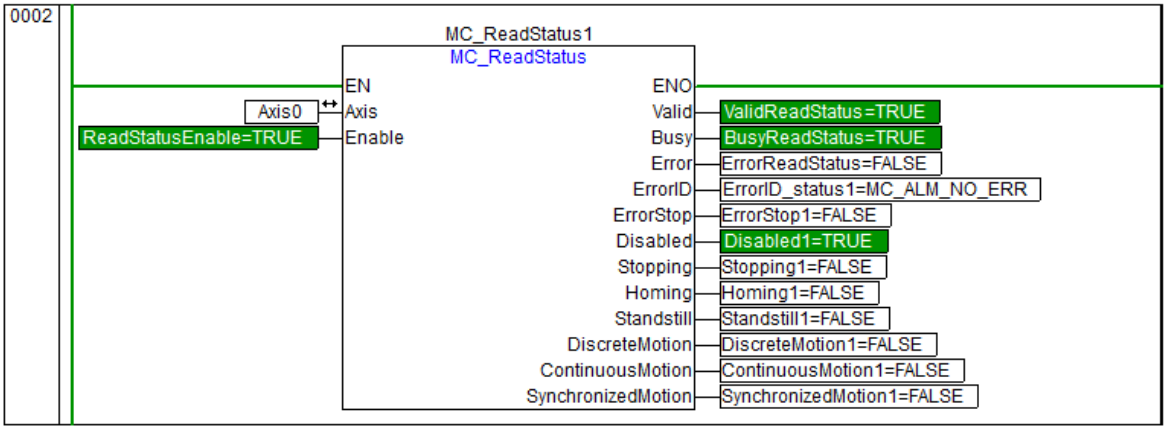
- LD program implementation

Use the ladder diagram to form a program. To set MC_ ReadStatus Enable to TRUE, the sample procedure is as follows:

(1) Before reading the status of PLCopen status machine of axis, initialize the axis first.



(2) After initialization, the status of axis PLCopen status machine can be read directly.



■ ST language program

(1) Initialize the axis number and axis type settings

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

(2) Read the status of axis PLCopen status machine

```
MC_ReadStatus1(
Enable := ReadStatusEnable,
Axis := Axis0,
Valid=>ValidReadStatus,
Busy=>BusyReadStatus,
Error=>ErrorReadStatus,
ErrorID=>ErrorID_status1,
ErrorStop=>ErrorStop1,
Disabled=>Disabled1,
Stopping=>Stopping1,
Homing=>Homing1,
Standstill=>Standstill1,
```

```
DiscreteMotion=>DiscreteMotion1,
ContinuousMotion=>ContinuousMotion1,
SynchronizedMotion=>SynchronizedMotion1);
```

5.20.4 Precautions

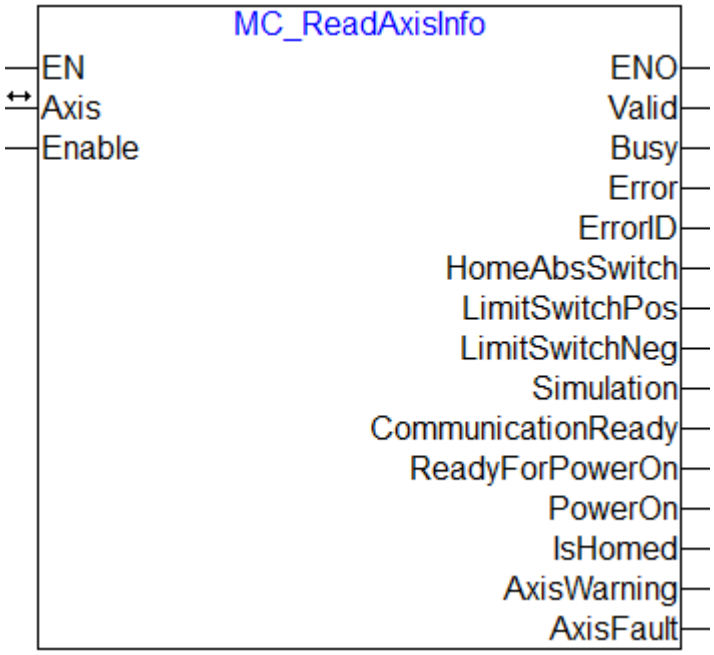
This command is of Enable type. When the Enable is at high level, the function block continuously monitors the Axis status without restart or triggering multiple starts.

5.20.5 Reference

If there is an error, please refer to the appendix Function Block Error Codes.

5.21 MC_ReadAxisInfo (Read Axis Info)

It is used to read information about the specified axis.



5.21.1 Parameter

Parameter Type	Variable Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Enable	Enable	No	-	TRUE/FALSE	BOOL
OUTPUT	Valid	Output valid	No	-	TRUE/FALSE	BOOL

	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR
	HomeAbsSwitch	Origin switch status	YES	FALSE	TRUE/FALSE	BOOL
	LimitSwitchPos	Positive Limit Switch Status	YES	FALSE	TRUE/FALSE	BOOL
	LimitSwitchNeg	Negative Limit Switch Status	YES	FALSE	TRUE/FALSE	BOOL
	Simulation	Axis in simulation state (retained parameter, FALSE)	YES	FALSE	TRUE/FALSE	BOOL
	CommunicationReady	Axis communication status is normal	YES	FALSE	TRUE/FALSE	BOOL
	ReadyForPowerOn	Axis ready enable	YES	FALSE	TRUE/FALSE	BOOL
	PowerOn	Axis enabling	YES	FALSE	TRUE/FALSE	BOOL
	IsHomed	Homing completed or not	YES	FALSE	TRUE/FALSE	BOOL
	AxisWarning	Axis warning	YES	FALSE	TRUE/FALSE	BOOL
	AxisFault	Axis failure	YES	FALSE	TRUE/FALSE	BOOL

5.21.2 Functions

This command reads axis-related information when Enable is TRUE. When the output pin Valid is TRUE, the read axis information state is valid; otherwise it is invalid. If an error occurs during reading, the output pin Error is set to TRUE and ErrorID displays the cause of the error. When Enable is FALSE, all function block output pins are invalid.

- Description of output shaft information
  - The output pin HomeAbsSwitch indicates the status of the origin return switch channel. TRUE means valid, and FALSE means invalid;
  - The output pins LimitSwitchPos and LimitSwitchNeg indicate the status of positive and negative hard limit switch channels. TRUE indicates valid, while FALSE indicates invalid;
  - The output pin Simulation indicates whether the current axis is in simulation mode. TRUE indicates that the axis is in simulation mode, and FALSE indicates non-simulation mode. The simulation function is not supported temporarily, so pin Simulation is always FALSE;
  - The output pin CommunicationReady indicates whether the communication state of the current axis is normal. If the current axis is an EtherCAT bus axis and the communication is normal, the CommunicationReady output is TRUE; otherwise, it is FALSE;
  - The output pin ReadyForPowerOn indicates whether the current axis is ready for enabling. When the current axis has no error and the axis is not enabled, the ReadyForPowerOn output is TRUE; otherwise, it is FALSE;
  - The output pin PowerOn indicates whether the current axis is enabled or not. When the current axis has no error and the axis is enabled, the PowerOn output is TRUE; otherwise, it is FALSE;
  - The output pin IsHomed indicates whether the current axis has completed homing. TRUE indicates that the current axis has completed homing, and FALSE indicates that the current axis has not completed homing;
  - The output pin AxisWarning indicates whether a warning occurs on the current axis. If the current

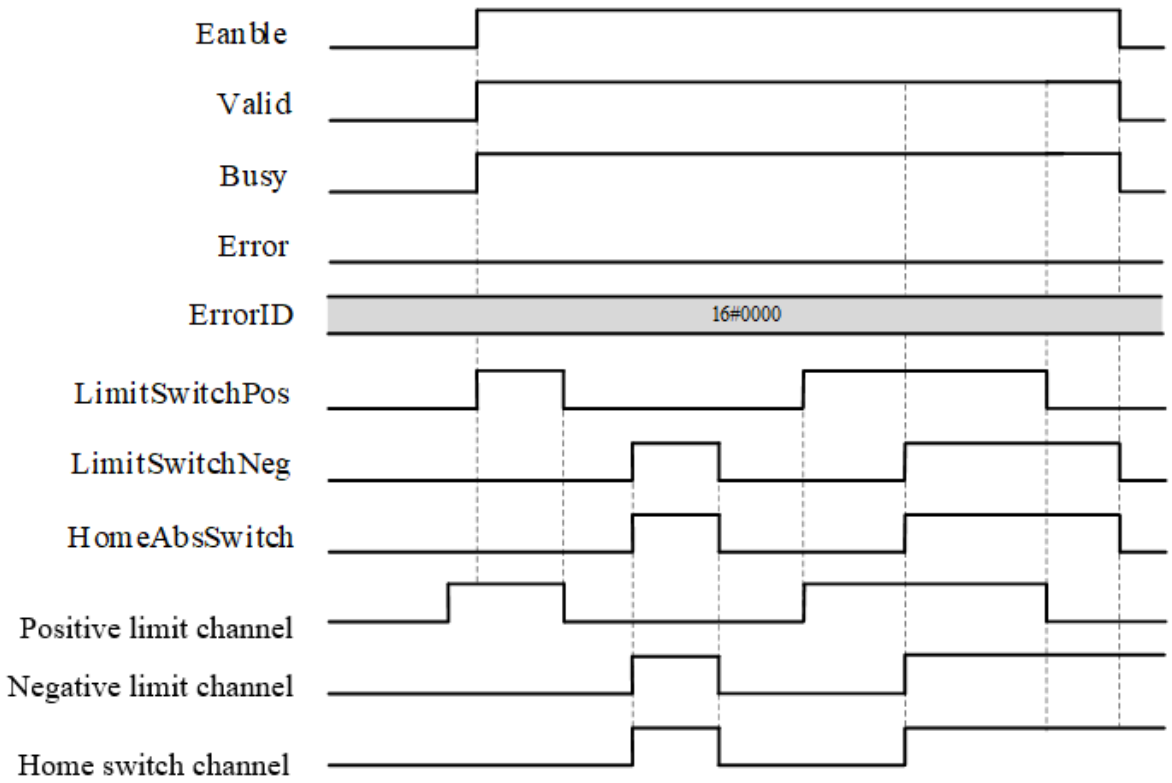
axis is an EtherCAT axis and a warning is reported, AxisWarning outputs TRUE; otherwise, it is FALSE;

- The output pin AxisFault indicates whether the current axis is faulty or not. TRUE indicates that the current axis is faulty, and FALSE indicates that the current axis is fault-free.

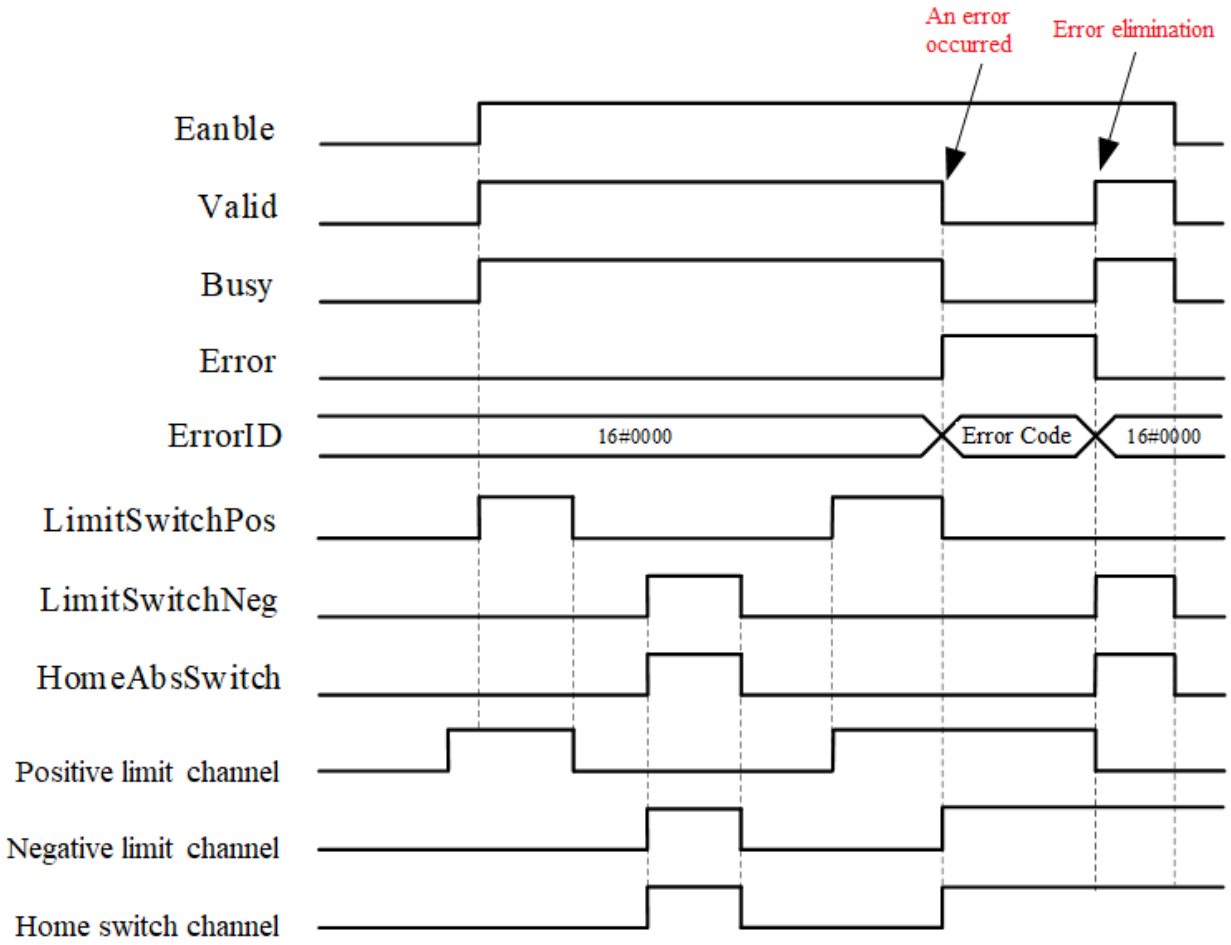
The above output pins are not forced to be mutually exclusive, and the output status is displayed according to the actual axis status information.

- Function block timing diagram

(1) Timing diagram of positive and negative hard limit status, origin switch status, and other axis information status Timing diagrams are similar.



(2) Timing diagram of recovery after an error occurs when obtaining the positive and negative hard limit status and the lower limit switch status.



5.21.3 Examples

The example uses the function block MC_ReadAxisInfo to get the axis information status of Axis0. EXAMPLE Call MC_Power to enable Axis0 by determining whether there is an error in Axis0 and whether Axis0 is ready for PowerOn.

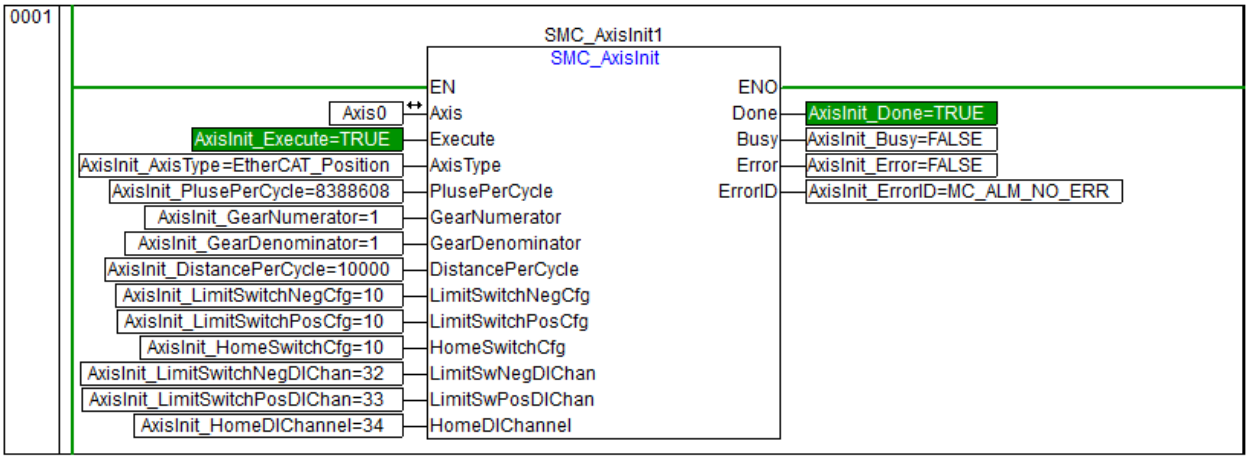
■ Variable

No.	Variable Name	Variable Type	Initial Value	Variable Description
1	MC_ReadAxisInfo	MC_READAXISINFO		Command for reading axis information
2	ReadAxisInfo_En	BOOL	FALSE	Enabled, valid for rising edge
3	ReadAxisInfo_Valid	BOOL	FALSE	Valid flag of reading axis information
4	ReadAxisInfo_Busy	BOOL	FALSE	Busy flag for reading axis information
5	ReadAxisInfo_Error	BOOL	FALSE	Error flag for reading axis information
6	ReadAxisInfo_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	Axis information reading error ID
7	ReadAxisInfo_HomeAbsSwitch	BOOL	FALSE	Absolute origin switch status
8	ReadAxisInfo_LimitSwitchPos	BOOL	FALSE	Positive Limit Switch Status
9	ReadAxisInfo_LimitSwitchNeg	BOOL	FALSE	Negative Limit Switch Status
10	ReadAxisInfo_Simulation	BOOL	FALSE	Simulation mode flag
11	ReadAxisInfo_ComRdy	BOOL	FALSE	Communication Ready Flag
12	ReadAxisInfo_RdyPowerON	BOOL	FALSE	Axis enable ready flag

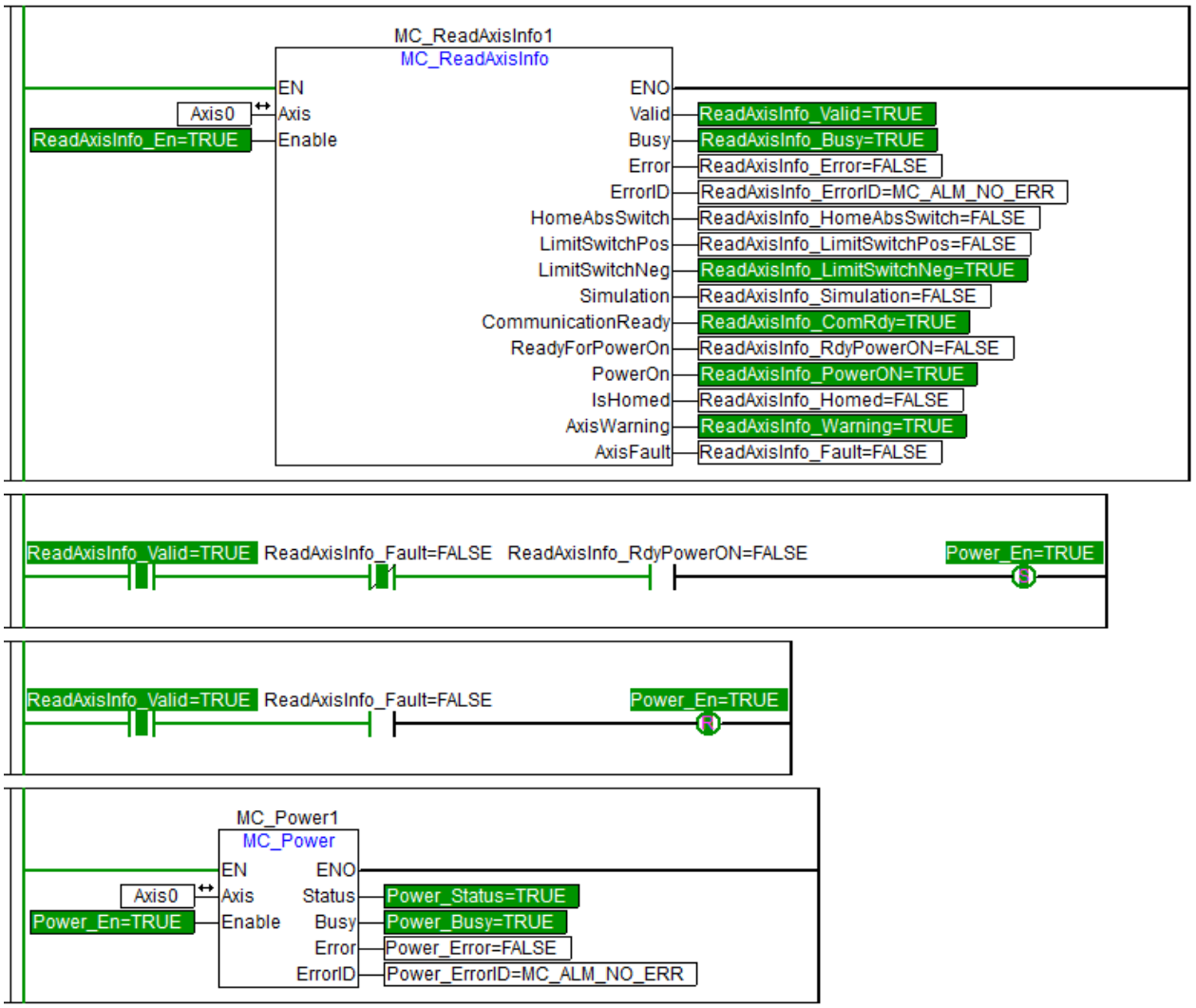
13	ReadAxisInfo_PowerON	BOOL	FALSE	Axis enabling status flag
14	ReadAxisInfo_Homed	BOOL	FALSE	Axis origin return completion flag
15	ReadAxisInfo_Warning	BOOL	FALSE	Warning sign for reading axis information
16	ReadAxisInfo_Fault	BOOL	FALSE	Fault flag for reading axis information
17	MC_Power1	MC_POWER		Axis enable command
18	Power_En	BOOL	FALSE	Axis enabled, level valid
19	Power_Status	BOOL	FALSE	Axis enabling status
20	Power_Busy	BOOL	FALSE	Busy sign of axis enabling command
21	Power_Error	BOOL	FALSE	Axis enable command error flag
22	Power_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	Axis enable command error ID

■ LD program implementation

(1) First, call the SMC_AxisInit command to initialize axis-related information.



(2) Then call the MC_ReadAxisInfo command to obtain axis information. After the axis has no error and is ready for enabling, call the MC_Power command to enable the axis.



■ ST program implementation

(1) First, call the MC_AxisInit command to initialize axis-related information.

```
MC_ReadAxisInfo1(
Enable := ReadAxisInfo_En,
Axis := Axis0,
Valid=>ReadAxisInfo_Valid,
Busy=>ReadAxisInfo_Busy,
Error=>ReadAxisInfo_Error,
ErrorID=>ReadAxisInfo_ErrorID,
HomeAbsSwitch=>ReadAxisInfo_HomeAbsSwitch,
LimitSwitchPos=>ReadAxisInfo_LimitSwitchPos,
LimitSwitchNeg=>ReadAxisInfo_LimitSwitchNeg,
Simulation=>ReadAxisInfo_Simulation,
CommunicationReady=>ReadAxisInfo_ComRdy,
ReadyForPowerOn=>ReadAxisInfo_RdyPowerON,
PowerOn=>ReadAxisInfo_PowerON,
IsHomed=>ReadAxisInfo_Homed,
```

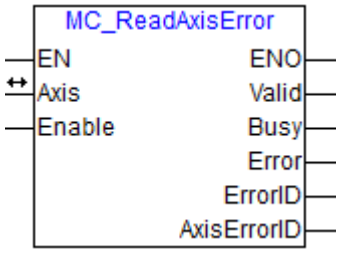
```
AxisWarning=>ReadAxisInfo_Warning,
AxisFault=>ReadAxisInfo_Fault);
```

(2) Then call the MC_ReadAxisInfo command to obtain axis information. After the axis has no error and is ready for enabling, call the MC_Power command to enable the axis.

```
IF TRUE = ReadAxisInfo_Valid AND FALSE = ReadAxisInfo_Fault AND
TRUE = ReadAxisInfo_RdyPowerON THEN
   Power_En := TRUE;
ELSIF TRUE = ReadAxisInfo_Valid AND TRUE =  ReadAxisInfo_Fault THEN
   Power_En := FALSE;
END_IF;
MC_Power1(
Enable := Power_En,
Axis := Axis0,
Status=>Power_Status,
Busy=>Power_Busy,
Error=>Power_Error,
ErrorID=>Power_ErrorID);
```

5.22 MC_ReadAxisError (Read Axis Error)

Read axis error messages (not function block errors).



5.22.1 Parameter

Parameter Type	Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Enable	Enable	No	-	TRUE/FALSE	BOOL
OUTPUT	Valid	Output valid	No	-	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR
	AxisErrorID	Axis error code	YES	0	-	DWORD

5.22.2 Functions

This command reads the axis error information. The details of this command are as follows:

(1) This command is valid for high level. After the axis initialization, the axis error information can be read directly. If there is no error in the read axis, Valid is TRUE and AxisErrorID is 0; if an error occurs in the axis, AxisErrorID outputs the corresponding axis error code.

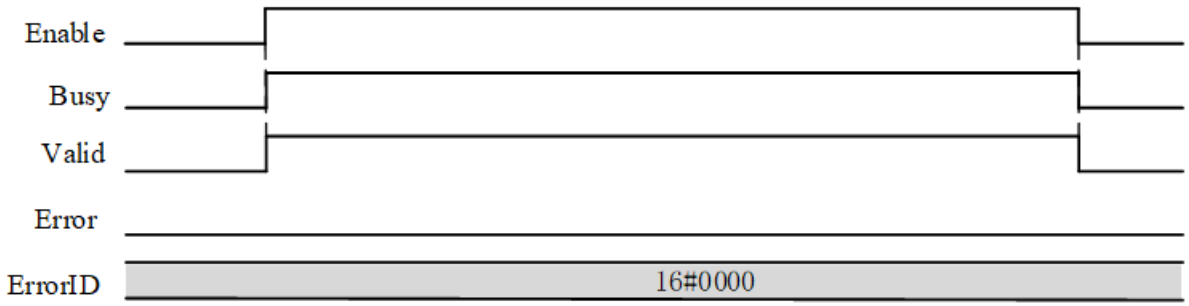
(2) When Enable is TRUE, the function block continuously reads the axis error information; when Enable is FALSE, the function block pin returns to the Default Value.

(3) Axis error description:

Error Code	Error Type	Error Description
100001	Axis error	Axis error
100002	Slave Station Offline	Slave Station Offline Error
100003	Reading failed	Failed to read error code
200000+	System error	System error
0-65535	Axis error code	Refer to the axis manual, and read the error code of axis 603F object dictionary

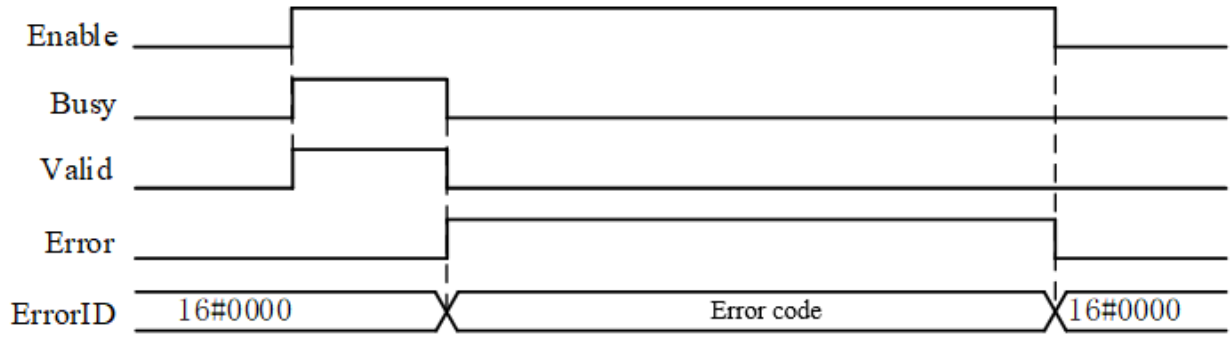
■ Function block timing diagram

(1) Normal timing diagram



(2) Error timing diagram

When an error occurs, Error is TRUE and the other pins are FALSE.



5.22.3 Examples

Set up MC_ ReadAxisError sample program to read axis error information.

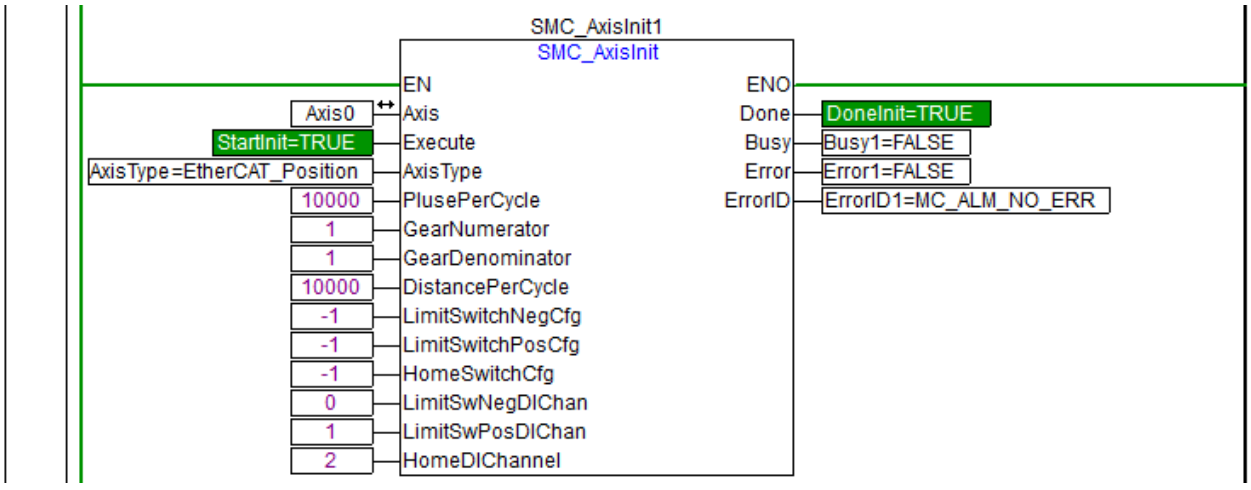
- Primary variables

Name	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	Change to TRUE at the start of initialization
DoneInit	BOOL	FALSE	Change to TRUE upon completion of initialization
ReadAxisErrEnable	BOOL	FALSE	Change to TRUE at the beginning of reading
ValidReadAxisErr	BOOL	FALSE	Change to TRUE when reading is successful
BusyReadAxisErr	BOOL	FALSE	Change to TRUE on reading
ErrorReadAxisErr	BOOL	FALSE	Change to TRUE when an error occurs
AxisErrorId	DWORD	0	Read result

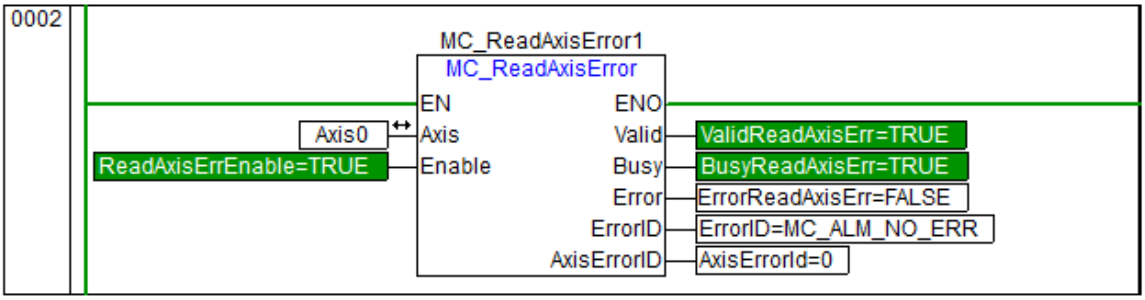
- LD program implementation

Use the ladder diagram to form a program. To set MC_ ReadAxisError Enable to TRUE, the sample procedure is as follows:

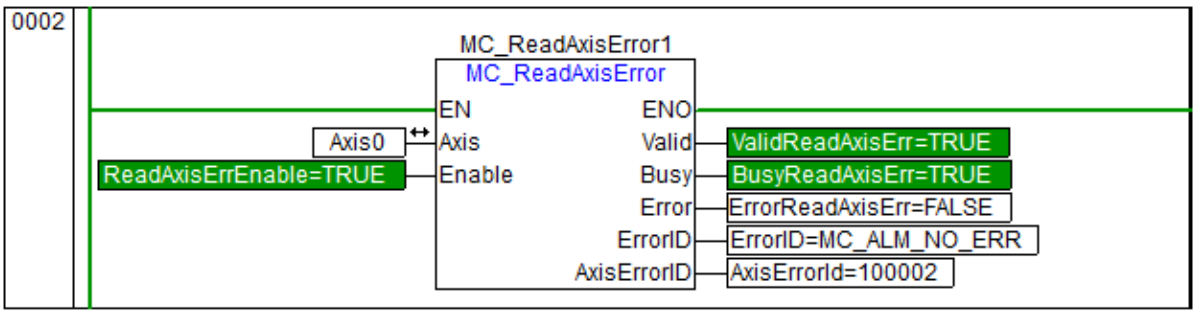
(1) Before reading the axis error information, initialize the axis first.



(2) After initialization, the axis error information can be read directly.



(3) Make Slave Station offline error and read the error.



■ ST language program

(1) Initialize the axis number and axis type settings

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

(2) Read axis error

```
MC_ReadAxisError1(
Enable := ReadAxisErrEnable,
Axis := Axis0,
Valid=>ValidReadAxisErr,
```

```
Busy=>BusyReadAxisErr,
Error=>ErrorReadAxisErr ,
ErrorID=>ErrorID,
AxisErrorID=>AxisErrorId);
```

5.22.4 Precautions

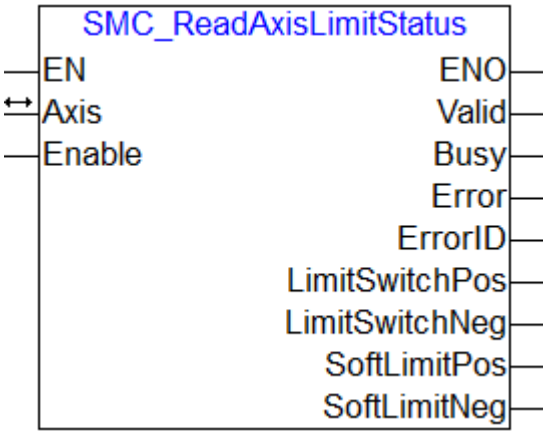
This command is of the Enable type, and there is no restart or multiple start.

5.22.5 Reference

Refer to the appendix Description of Functional Module Error Codes for error information of functional modules.

5.23 SMC_ReadAxisLimitStatus (Read Axis Limit Status)

It is used to read the positive and negative hard limit status and positive and negative soft limit status of the specified axis.



5.23.1 Parameter

Parameter Type	Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Enable	Enable	No	-	TRUE/FALSE	BOOL
OUTPUT	Valid	Output valid	No	-	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL

	ErrorID	Command error code	YES	0	-	SMC_ERROR
	LimitSwitchPos	Positive hard limit active	YES	FALSE	TRUE/FALSE	BOOL
	LimitSwitchNeg	Negative hard limit active	YES	FALSE	TRUE/FALSE	BOOL
	SoftLimitPos	Positive soft limit active	YES	FALSE	TRUE/FALSE	BOOL
	SoftLimitNeg	Negative soft limit active	YES	FALSE	TRUE/FALSE	BOOL

5.23.2 Functions

When Enable=TRUE, this command will read the current hardware limit status and software limit status of the axis, including positive and negative hard limit statuss and positive and negative soft limit statuss. The read limit status is valid only when the output pin is Valid. When Enable is FALSE, all function block output pins are invalid.

If the set hard limit attribute is normally open, when the specified limit channel is TRUE, the corresponding limit status is valid; otherwise, it is invalid. If the set hard limit attribute is normally closed, when the specified limit channel is FALSE, the corresponding limit status is valid; otherwise, it is invalid.

The soft limit status is related to the axial parameter FSLimit and RSLimit. When the actual position MPos is greater than or equal to the FSLimit limit value, the positive soft limit status is TRUE, otherwise it is FALSE; when the actual position MPos is less than or equal to the RSLimit limit value, the negative soft limit status is TRUE, otherwise it is FALSE.

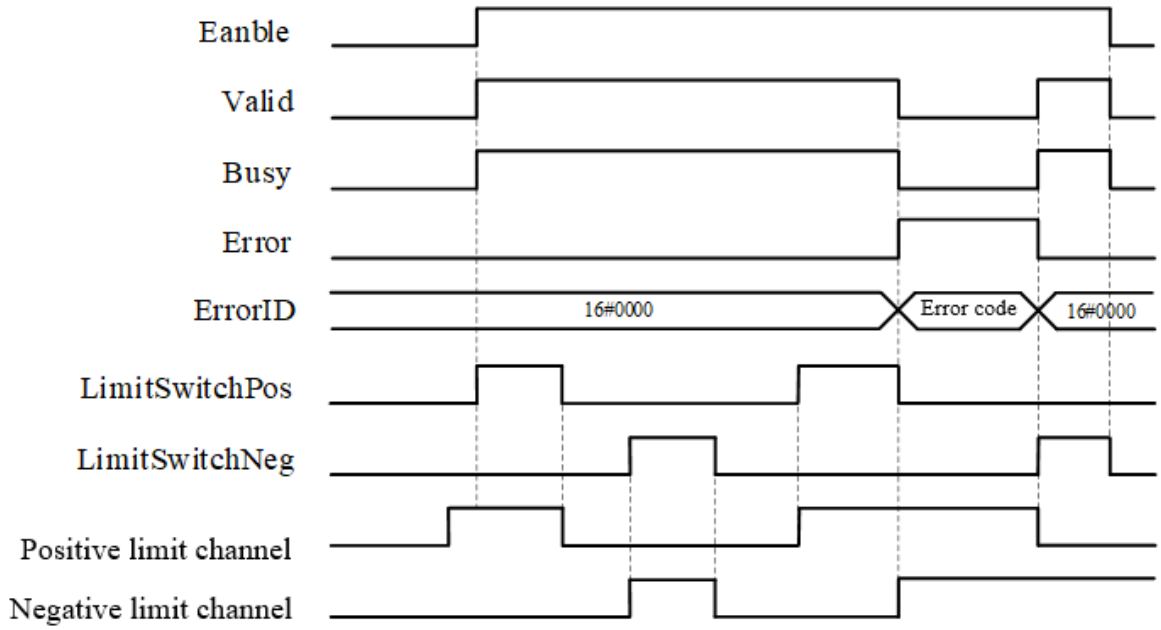
If the positive and negative hard limit input states are valid at the same time, the command reports an error.

Multiple commands of the same axis can run simultaneously, and the status results read by each command are consistent.

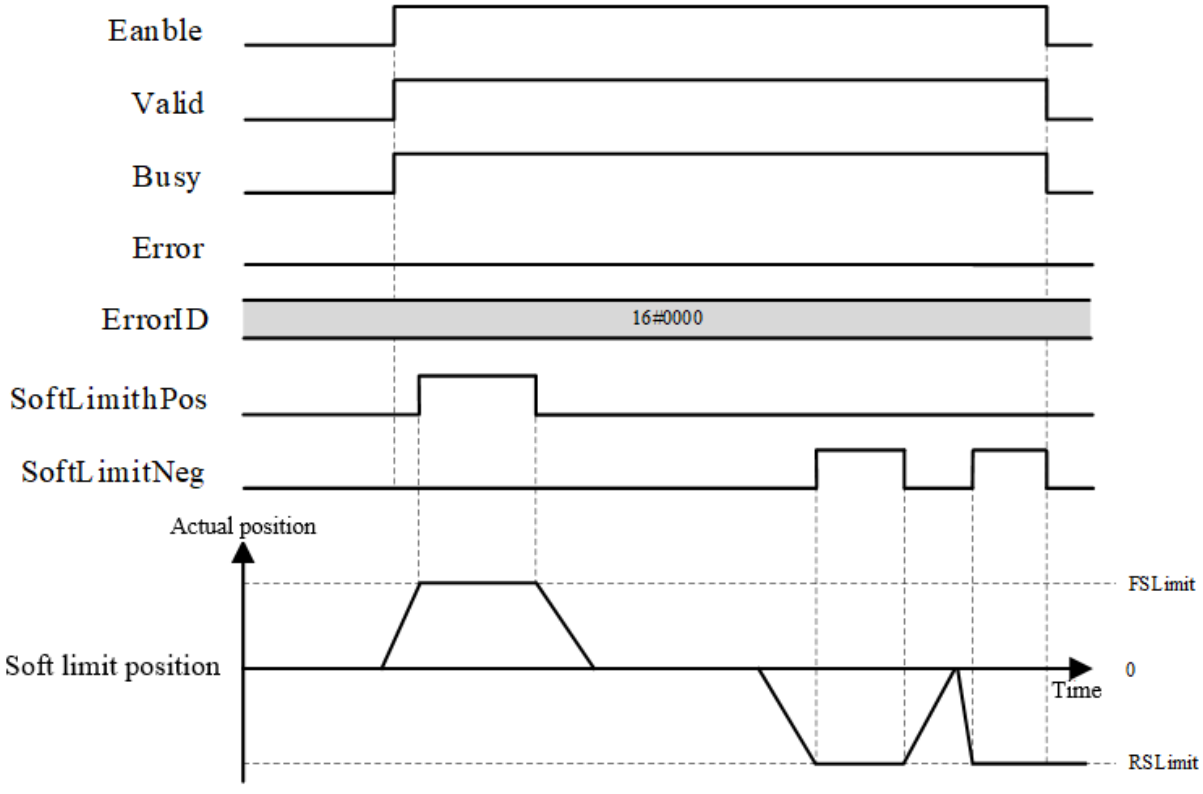
- Function block timing diagram

(1) Timing diagram of hardware positive limit and negative limit status.

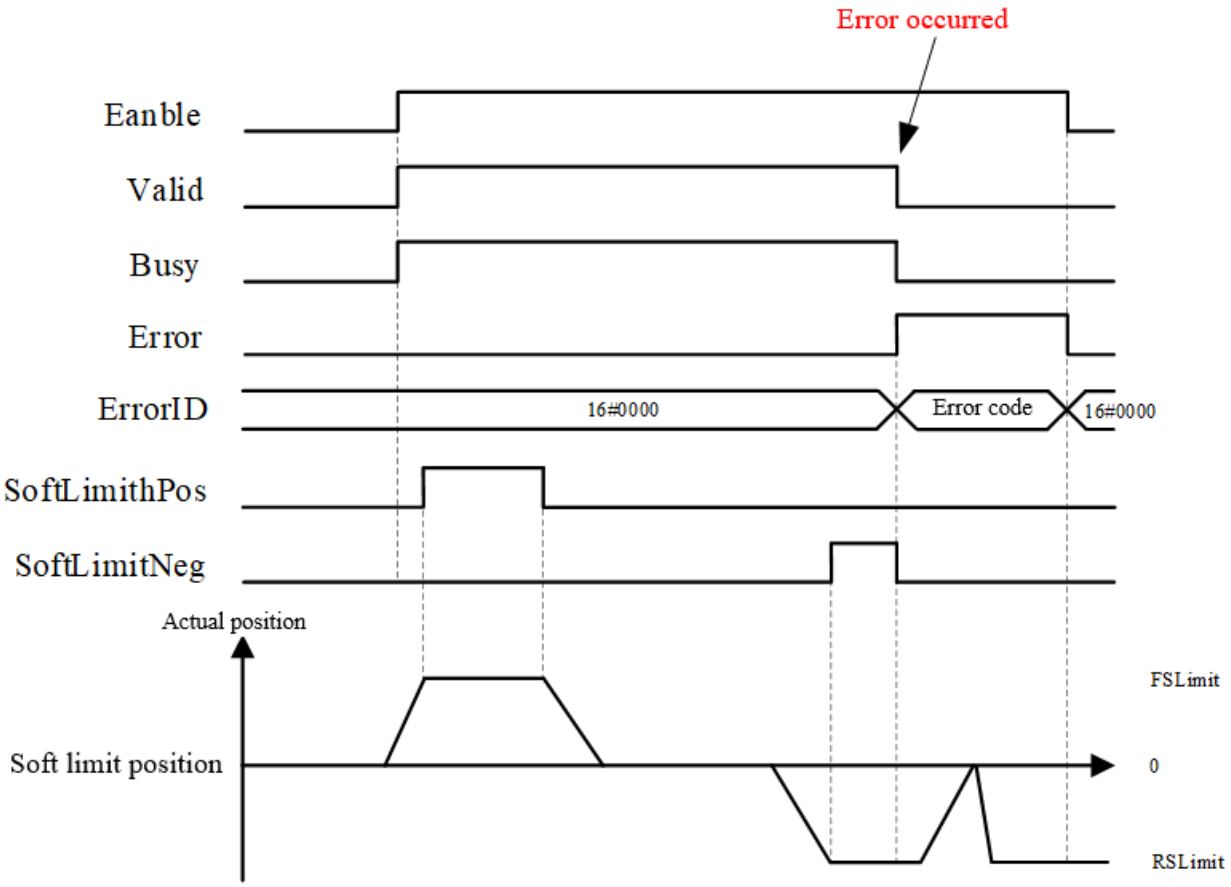
When the positive and negative hard limit switches are valid at the same time, this command reports an error.



(2) Timing diagram of software positive limit and negative limit status.



(3) Obtain the timing diagram when errors occur in positive and negative limit status of software.



5.23.3 Examples

This example uses the SMC_AxisInit functional block to illustrate the initialized positive and negative hard limit attribute configuration and channel configuration results.

The function block SMC_ReadAxisLimitStatus is used to obtain the positive and negative hard limit channel status and positive and negative soft limit status of Axis0.

The function block SMC_ReadAxisLimitStatus output pin LimitSwitchNeg is TRUE when the negative limit switch channel (%IX4.0) logic is active.

■ Variable

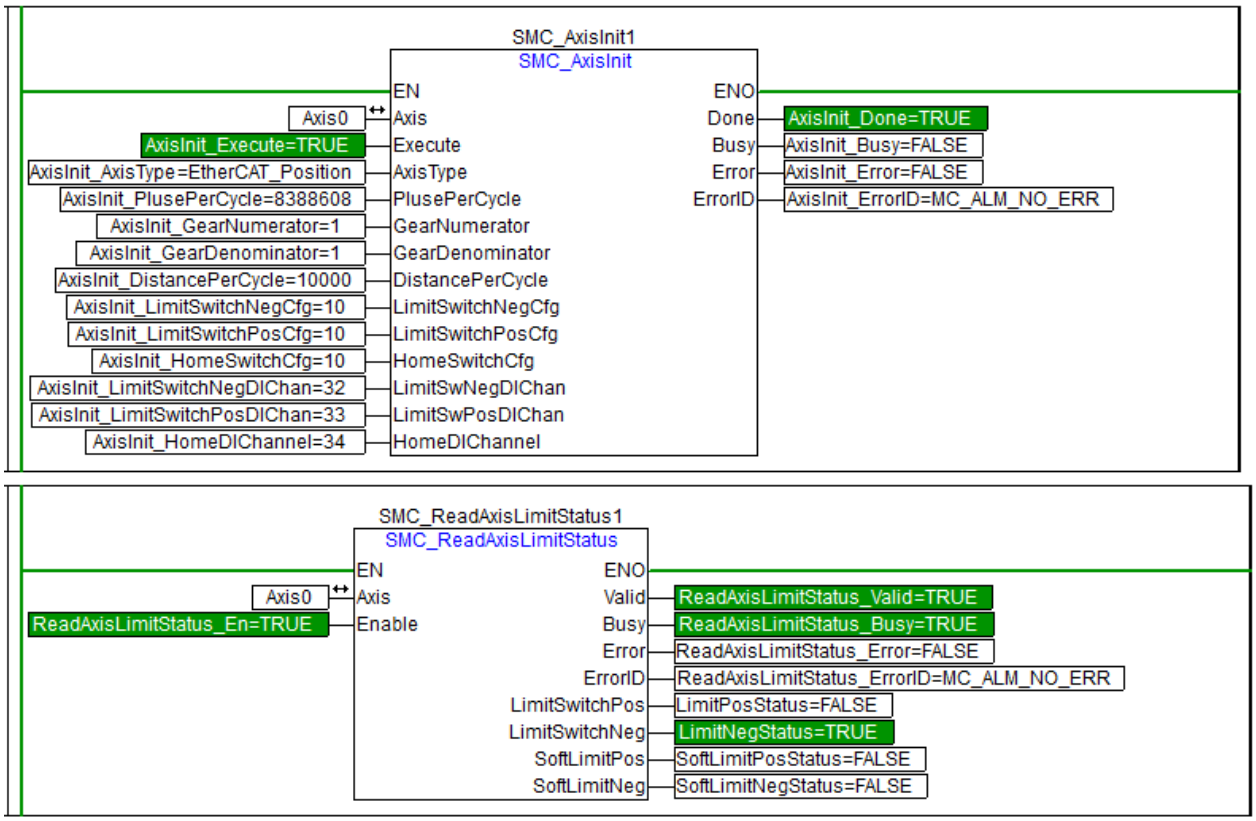
No.	Variable Name	Variable Type	Initial Value	Variable Description
1	SMC_ReadAxisLimitStatus1	SMC_ReadAxisLimitStatus		Read axis limit status command
2	ReadAxisLimitStatus_En	BOOL	FALSE	Enable reading axis limit command
3	ReadAxisLimitStatus_Valid	BOOL	FALSE	Valid flag of reading axis limit status
4	ReadAxisLimitStatus_Busy	BOOL	FALSE	Read axis limit status busy flag
5	ReadAxisLimitStatus_Error	BOOL	FALSE	Error flag for reading axis limit status
6	ReadAxisLimitStatus_ErrorID	SMC_ERROR	MC_ALM_NO_ERROR	Read axis limit status error ID
7	LimitPosStatus	BOOL	FALSE	Positive limit status
8	LimitNegStatus	BOOL	FALSE	Negative limit status
9	SoftLimitPosStatus	BOOL	FALSE	Positive soft limit status

10	SoftLimitNegStatus	BOOL	FALSE	Negative soft limit status

■ LD program implementation

First call the SMC_AxisInit command to initialize axis-related information, and then call the SMC_ReadAxisLimitStatus command to obtain limit status information.

Trigger the negative hard limit status to be valid, and check whether it is TRUE.



■ ST program implementation

(1) Call the SMC_AxisInit command to initialize axis-related information.

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
```

```
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

(2) Then call the command SMC_ReadAxisLimitStatus to obtain limit status information.

```
SMC_ReadAxisLimitStatus1(
Enable := ReadAxisLimitStatus_En,
Axis := Axis0,
Valid=>ReadAxisLimitStatus_Valid,
Busy=>ReadAxisLimitStatus_Busy,
Error=>ReadAxisLimitStatus_Error,
ErrorID=>ReadAxisLimitStatus_ErrorID,
LimitSwitchPos=>LimitPosStatus,
LimitSwitchNeg=>LimitNegStatus,
SoftLimitPos=>SoftLimitPosStatus,
SoftLimitNeg=>SoftLimitNegStatus);
```

5.23.4 Precautions

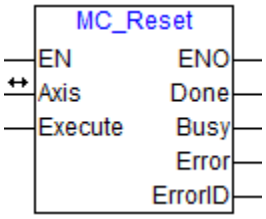
- If the positive and negative hard limit status are valid at the same time, the positive soft limit setting value is less than or equal to the negative soft limit setting value, or the soft limit status and the hard limit status are valid at the same time, an error will be reported by the function block.
- When the travel mode is set to cyclic travel (RepMode = 1 or 2), the limit status is invalid and has no practical significance.

5.23.5 Reference

Command SMC_AxisInit.

5.24 MC_Reset (Reset Axis Error)

Reset axis error.



5.24.1 Parameter

Parameter Type	Title	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
OUTPUT	Done	Completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.24.2 Functions

Axis errors can be reset by this command, which is described in detail as follows:

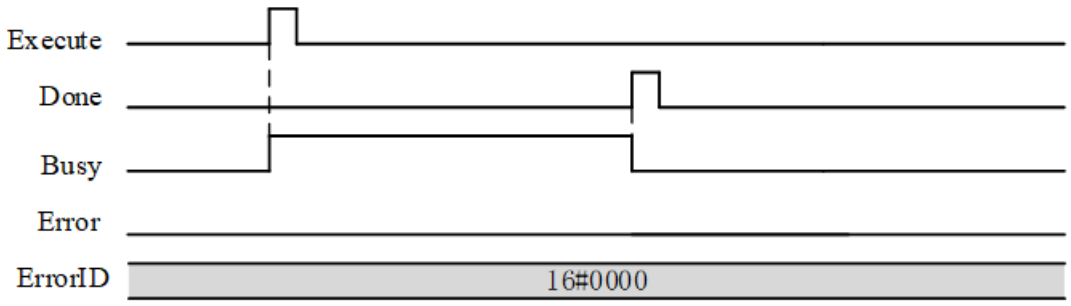
(1) This command can reset the faults of Ethercat bus axis and virtual axis.

(2) This command is valid when triggered by the rising edge of Execute, and Busy (busy flag) is TRUE during reset. After reset, the Done pin is TRUE, other pins are FALSE, Error after reset fails is TRUE, and ErrorID outputs corresponding error code.

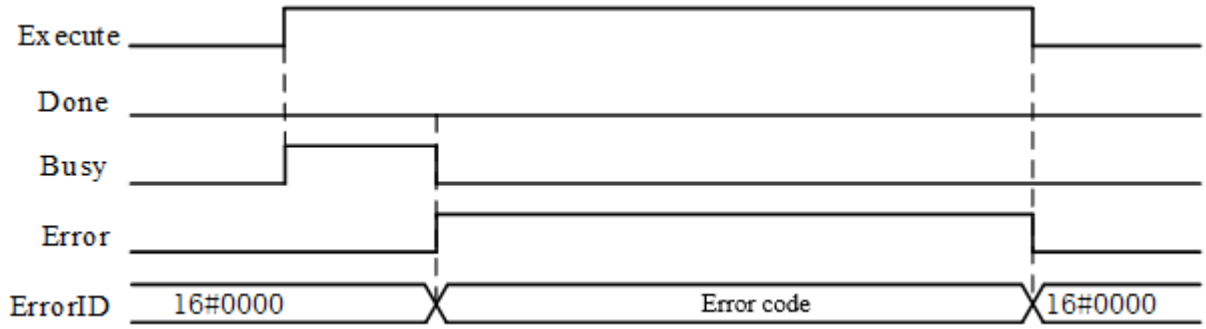
(3) After a successful reset, if the driver is enabled, the axis PLCopen status machine will enter the StandStill state; otherwise, it will enter the Disabled state.

■ Function block timing diagram

(1) Timing diagram of axis failure but successful reset.



(2) Timing diagram of axis reset failure.



5.24.3 Examples

Create an MC_ Reset example program to reset the axis error.

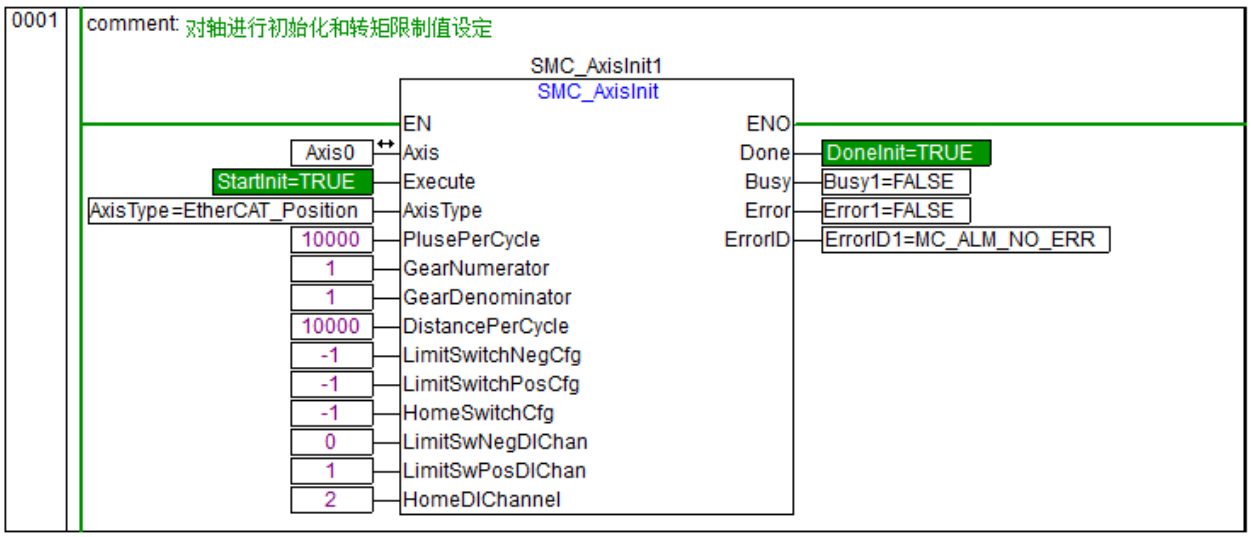
- Primary variables

Name	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0
StartInit	BOOL	FALSE	Change to TRUE at the start of initialization
DoneInit	BOOL	FALSE	Change to TRUE upon completion of initialization
ResetExe	BOOL	FALSE	Change to TRUE at the start of reset
DoneReset	BOOL	FALSE	Change to TRUE when reset is successful
BusyReset	BOOL	FALSE	Change to TRUE at reset
ErrorReset	BOOL	FALSE	Change to TRUE on reset failure

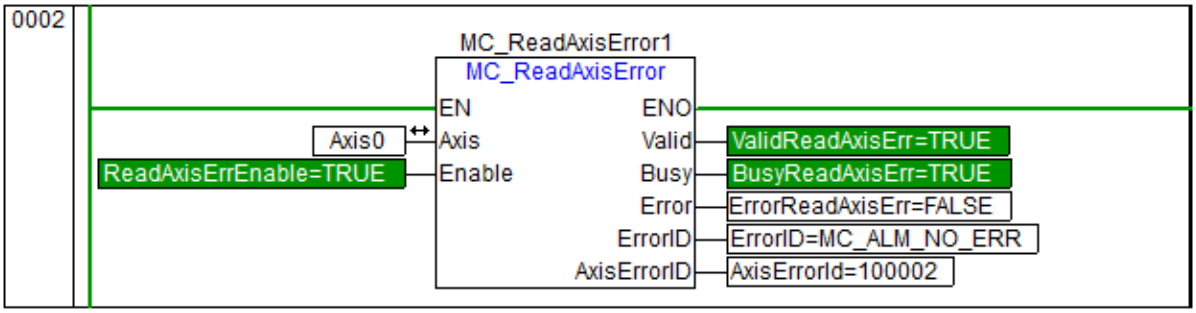
- LD program implementation

Use the ladder diagram to form a program. Set MC_ Reset Execute (rising edge trigger) to TRUE, sample procedure is as follows:

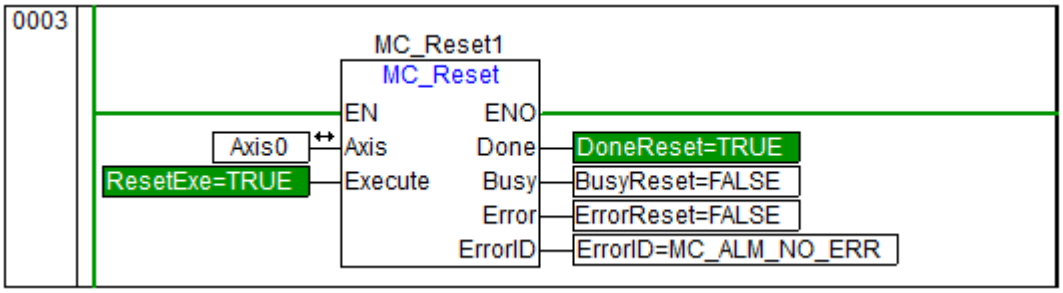
(1) First, initialize the axis



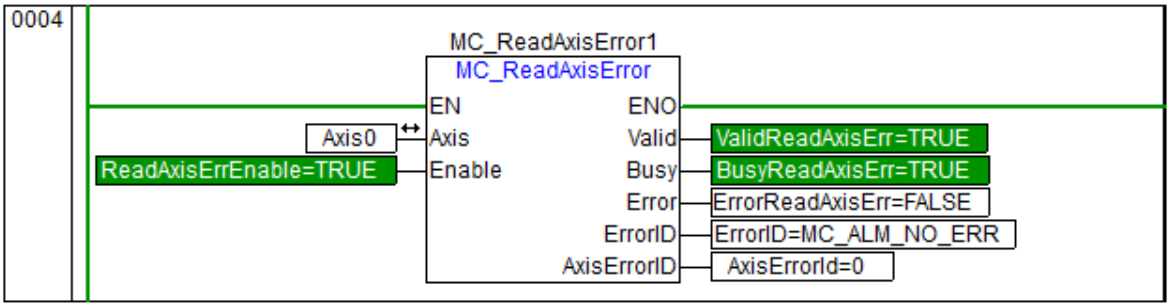
(2) Servo offline error



(3) Reset with MC_Reset



(4) When reading the axis error, it indicates that the error has been successfully reset.



■ ST language program

(1) Initialize the axis number and axis type settings

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
```

```
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

(2) Call the MC_Reset command for reset

```
MC_Reset1(
Execute := ResetExe,
Axis := Axis0,
Done=>DoneReset,
Busy=>BusyReset,
Error=>ErrorReset,
ErrorID=>ErrorID);
```

5.24.4 Precautions

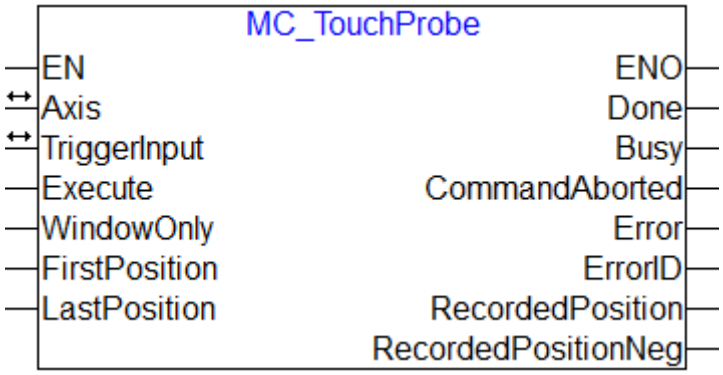
- When there is a non-resettable error, the axis error cannot be reset by calling this function block.

5.24.5 Reference

- For relevant reset error information, refer to the appendix Function Block Error Codes.

5.25 MC_TouchProbe(Enable External Latch)

This command latches the position of the specified axis and latched channel when a specified probe event trigger is detected.



5.25.1 Parameter

Parameter Description

Parameter	Name	Description	Empty	Default	Scope	Data Type

Type			or Not	Value		
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF
	TriggerInput	Latching configuration parameters	No	-	See data structure MC_TRIGGER_REF	MC_TRIGGER_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	WindowOnly	FALSE: Disable the window function. TRUE: Enable the window function, and only the current position is [FirstPosition, LastPosition] the probe signal is detected.	YES	FALSE	TRUE/FALSE	BOOL
	FirstPosition	Probe window start position	YES	0	Positive/negative/0	LREAL
	LastPosition	Probe Window End Position	YES	0	Positive/negative/0	LREAL
OUTPUT	Done	Latching completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR
	RecordedPosition	Rising edge latch position	YES	-	Positive/negative/0	LREAL
	RecordedPositionNeg	Falling Edge Latch Position	YES	-	Positive/negative/0	LREAL

MC_TRIGGER_REF Data Structure

Member	Description	Default Value	Scope	Data Type
LatchMode	Latch mode: 0: Drive mode	0	0	USINT
LatchID	Latch channel in drive mode: 0:Latch 1 1:Latch 2	0	0/1	USINT
DriveInput	Trigger signal in drive mode: 0: Drive IO terminal 1: Z phase of encoder	0	0/1	UDINT
TriggerEdge	Trigger Edge Type: 0: Rising edge 1: falling edge 2: Double-edge	0	0/1/2	USINT
ContinuousTrigger	Triggered continuously or not	FALSE	TRUE/FALSE	BOOL

5.25.2 Functions

This command implements the probe capture function for EtherCAT servo or encoder axis (ECI002) in accordance with CIA402.

This command does not support imaginary axis mode. When the command is running, if the state of the corresponding axis jumps to ErrorStop, this functional block reports an error and releases the latch channel occupied by the command.

Functions supported by this directive:

- Support probe capture of two latched channels;
- Support driver DI terminal trigger and Z-phase signal as the trigger source, which needs to be supported by servo driver;
- Supporting rising edge, falling edge and double-edge trigger types, which needs to be supported by the servo driver;
- Supporting continuous trigger mode;
- Supports window position capture mode.

Using this command, you need to map the following object dictionaries into PDO:

- Touch probe function (60B8 hex, required);
- Touch probe status (60B9 hex, required);
- Touch probe pos 1 pos value (60BA hex, required for probe 1 rising edge mode);
- Touch probe pos1 neg value (60BB hex, required for probe 1 falling edge mode);
- Touch probe pos 2 pos value (60BC hex, required for rising edge mode of probe 2);
- Touch probe pos2 neg value (60BD hex, required for probe 2 falling edge mode).

In the case of a multi-axis servo drive, it is necessary to map the offset object dictionary according to the actual situation.

When this command is on the rising edge of Excute, check whether TriggerInput parameter and window parameter are within the effective range. If yes, latch TriggerInput parameter and window parameter records. The parameters cannot be changed when Excute is in other states. If an error occurs during checking, the function block will report an error and output pin Error will be set to TRUE. After the input parameter check is passed, the command enters the latch state and the output pin Busy is set to TRUE. When the command detects that the probe input specified by LatchID is valid and meets the probe detection conditions, the function block will latch the current position of the axis and refresh the latched position to the output pin RecordedPosition or RecordedPositionNeg as required. At the same time, the function block output pin is Done TRUE. The output pins Busy, Done and Error are forced to be mutually exclusive.

This command can detect the rising and falling edges of the probe signal independently or simultaneously. When only rising (falling) edges are detected, the command writes the position value detected by the rising (falling) edge into the RecordedPosition(RecordedPositionNeg) parameter. At this time, a detection cycle is completed and the Done pin is set. If rising and falling edges are detected at the same time, after the rising edge of the command Excute is valid, the position will be written into the RecordedPosition pin immediately after the command detects the rising edge of the probe, and the position will be written into the RecordedPositionNeg pin immediately after the falling edge is detected. At this point, it is a complete detection cycle and the Done signal is output. There is no requirement for the input sequence of rising and falling edges.

The command can be configured as a single shot or continuous shot. When it is configured as single trigger, if the Done signal output is valid, it means that the single position capture is completed and the command execution ends; when it is configured as continuous trigger mode, after the position capture is completed, the Done outputs a valid signal, resets again after an IEC scanning cycle, and the command automatically starts to detect new probe input signals. In the continuous firing mode, if the input frequency of the probe signal is greater than the frequency of the IEC scan cycle, this command will lose part of the probe signal.

■ Window Mode Description

When the input parameter WindowOnly is FALSE, the window detection function is invalid. As long as the probe input signal is valid, the position of the corresponding axis can be latched.

When the input parameter WindowOnly is TRUE, the window detection function is effective. The probe capture command now checks the probe signal and latches the position only if the current position of the axis is within the window.

(1) Limited stroke

Effective window range: FirstPosition ≤ Window Range ≤ LastPosition, when the input parameter FirstPosition > LastPosition, the function block reports an error.

(2) Cyclic stroke

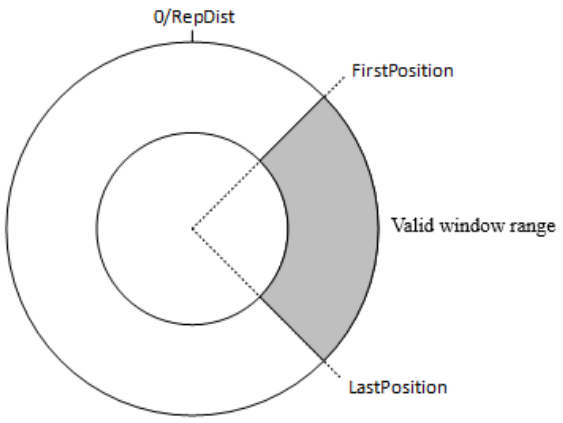
Both FirstPosition ≤ LastPosition and FirstPosition > LastPosition during cyclic travel can be configured according to actual usage.

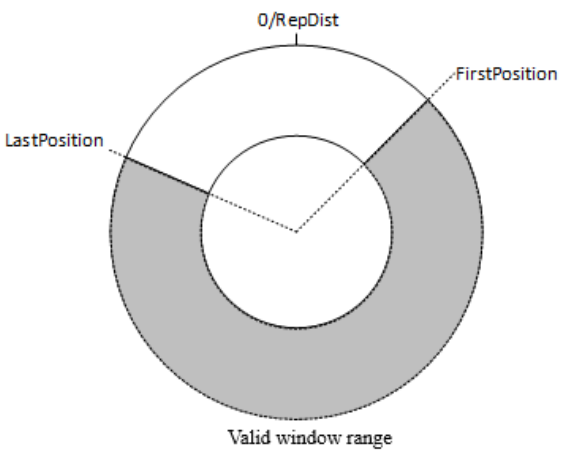
When FirstPosition > LastPosition, the setpoint will span the upper and lower limit positions of the cyclic stroke.

The function block reports an error when the set values FirstPosition and LastPosition exceed the upper and lower limits of the cyclic stroke.

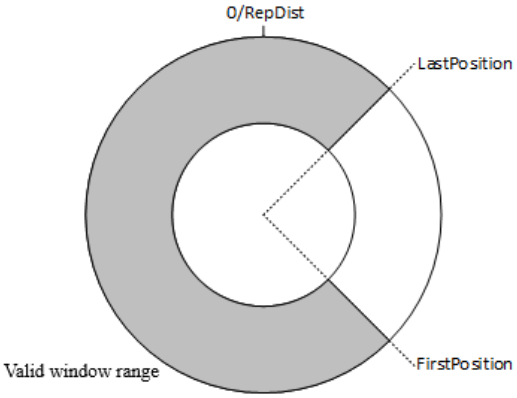
□ The effective range in cyclic stroke mode 1 is as follows, and the cyclic stroke range is [0, RepDist]:

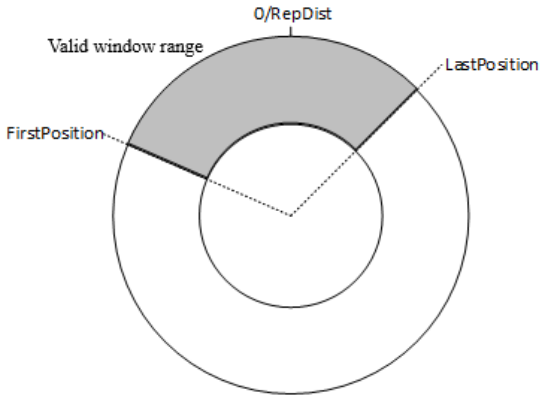
When FirstPosition ≤ LastPosition, as shown in the following figure:





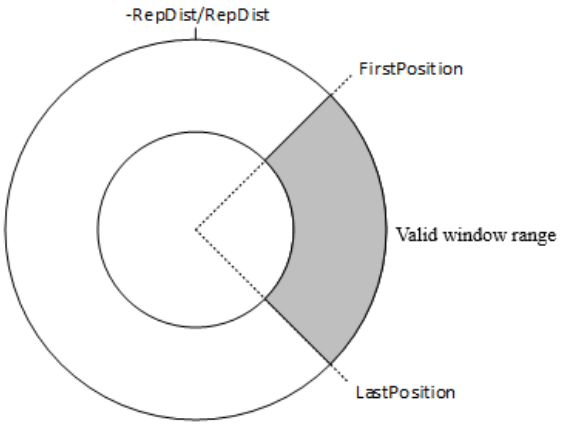
When FirstPosition > LastPosition, as shown in the following figure:

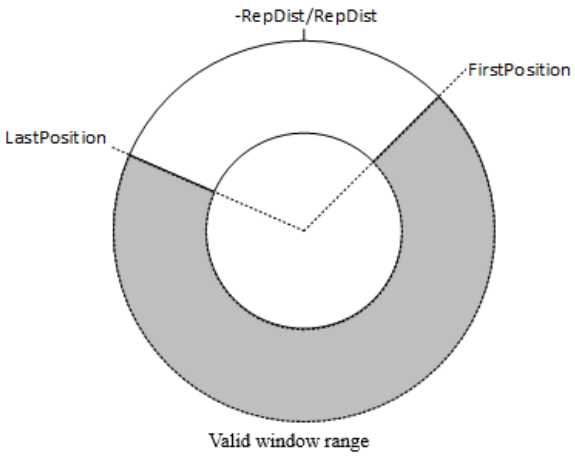




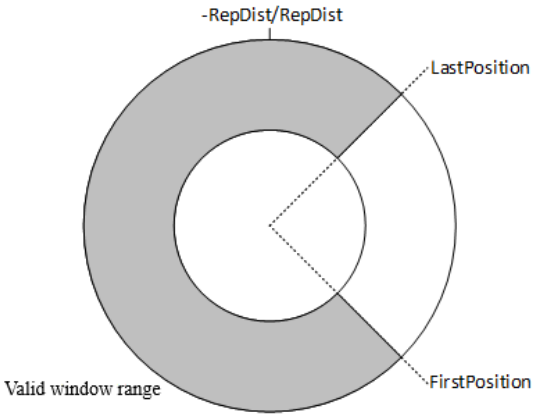
□ The effective range of cyclic stroke mode 2 is shown in the following figure, and the cyclic stroke range is [-RepDist, RepDist]:

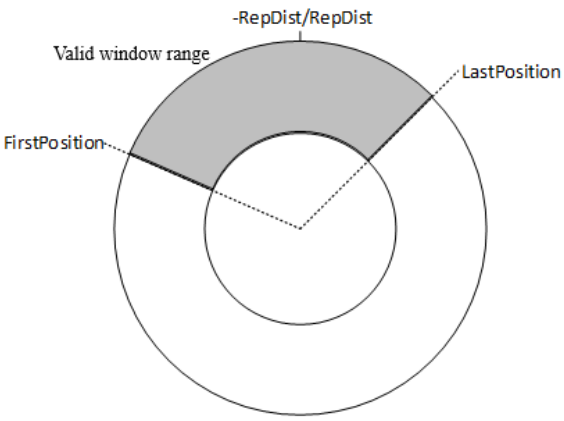
When FirstPosition ≤ LastPosition, as shown in the following figure:





When FirstPosition > LastPosition, as shown in the following figure:





■ Description of interruption

This command supports simultaneous detection of probe 1 and probe 2 on the same axis. If two

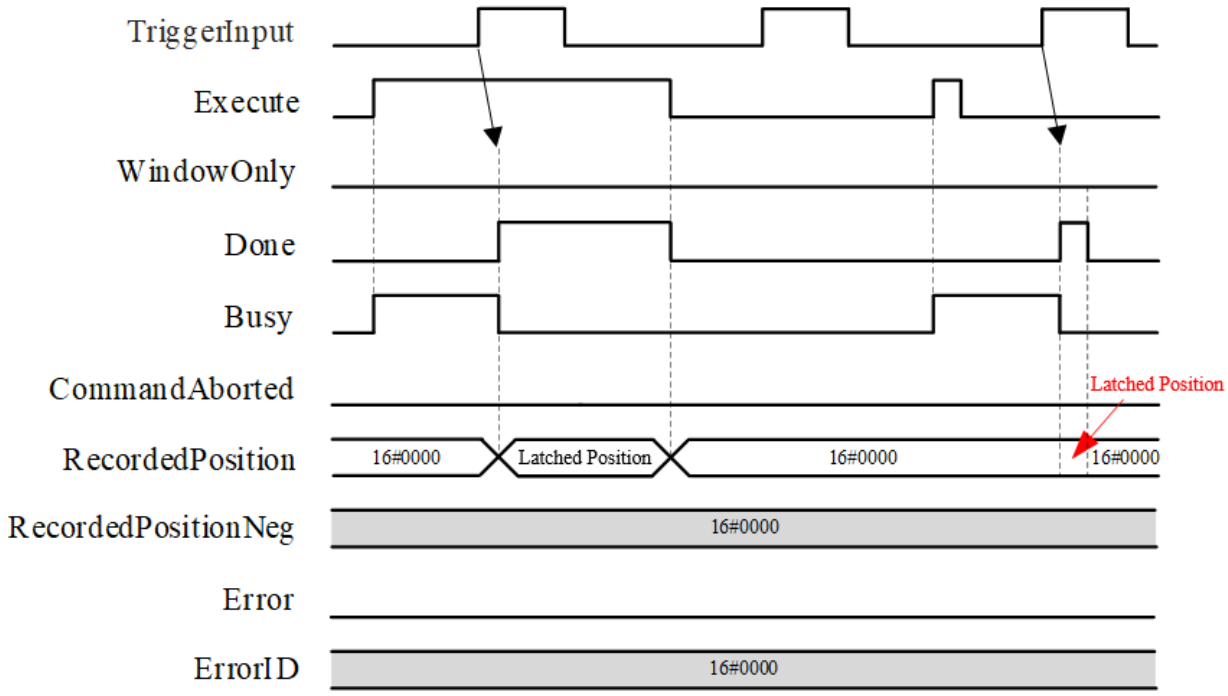
probe commands are defined in the program and the probe IDs are different, the two probe commands will operate independently without affecting each other. However, if the probe IDs are the same, an error will be reported for the probe command to be executed later. If only one probe command is defined in the program, and the rising edge is triggering this command again while it is running, the function block will report an error and abort the running probe command to release the probe ID occupied by the command. If a probe command is called repeatedly, the function block will report an error upon enabling and will not execute the command. This is not a valid use case, and repeating the same function block instance will trigger a warning during configuration compilation.

During normal operation, the MC_TouchProbe command can be aborted through the MC_AbortTrigger command and the occupied probe can be released. When the MC_AbortTrigge output pin Done is TRUE, it indicates that the interruption is successful. At this time, the MC_TouchProb command output pin CommandAborted will be set.

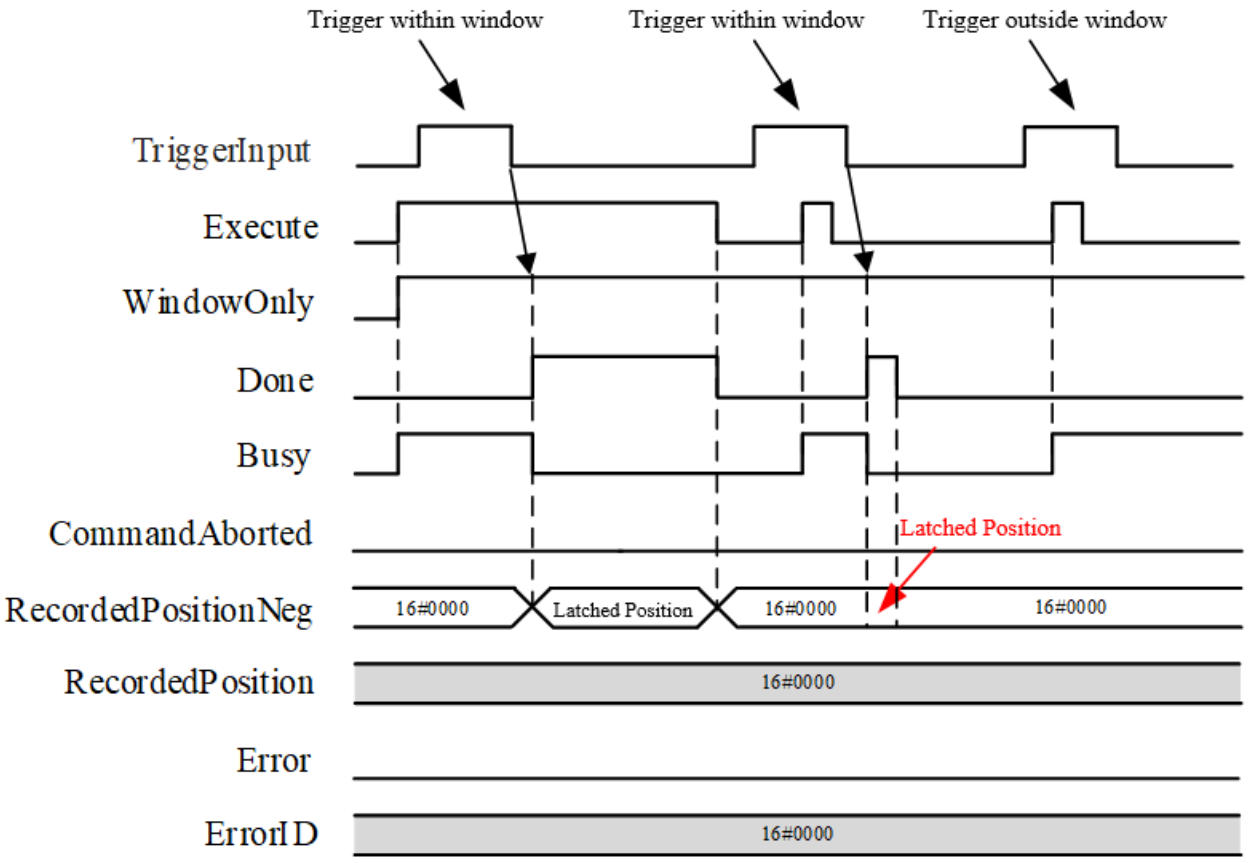
When MC_Home and HMC_GantryInit commands are used, if the homing mode is configured as Z-phase signal correlation, probe 1 will be used to capture Z-phase signals. At this time, the channel of probe 1 of the axis will be occupied, and other commands cannot be used or interrupted. If Probe 1 is already occupied when initiating Z signal-related homing with MC_Home or HMC_GantryInit, an error will be reported.

■ Function block timing diagram

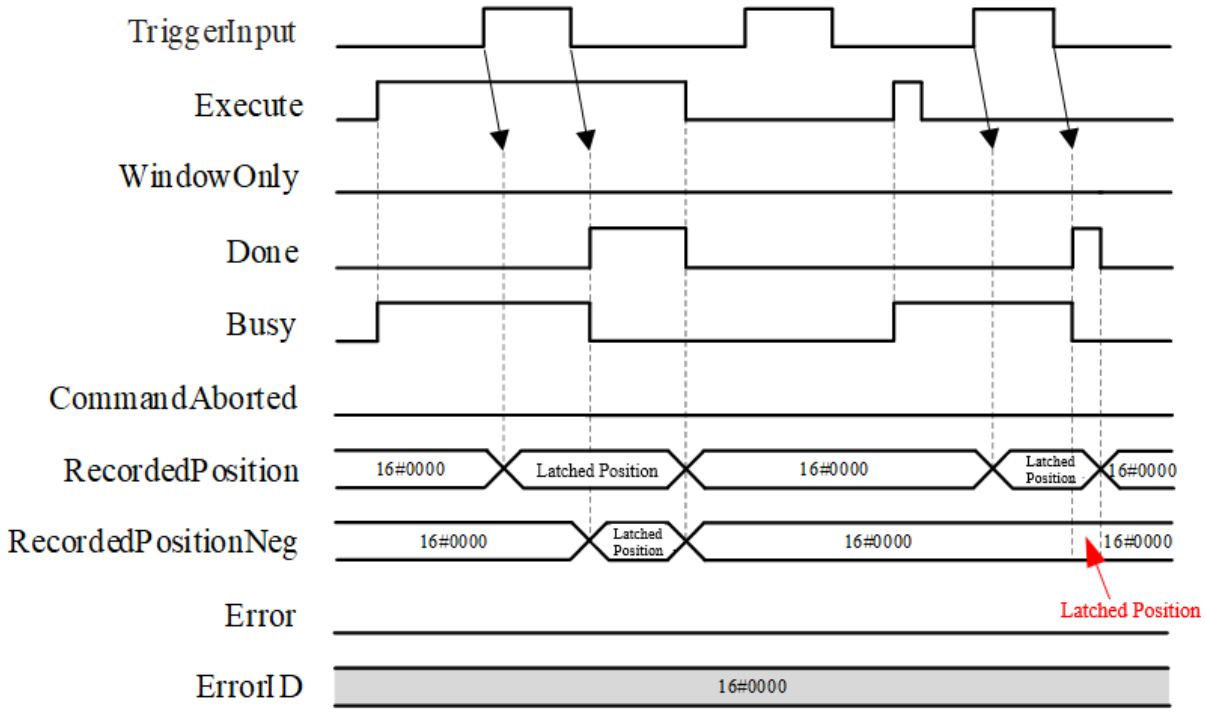
(1) The rising edge of probe 1 is valid, the DI terminal is triggered in single-trigger mode, and the window function is invalid.



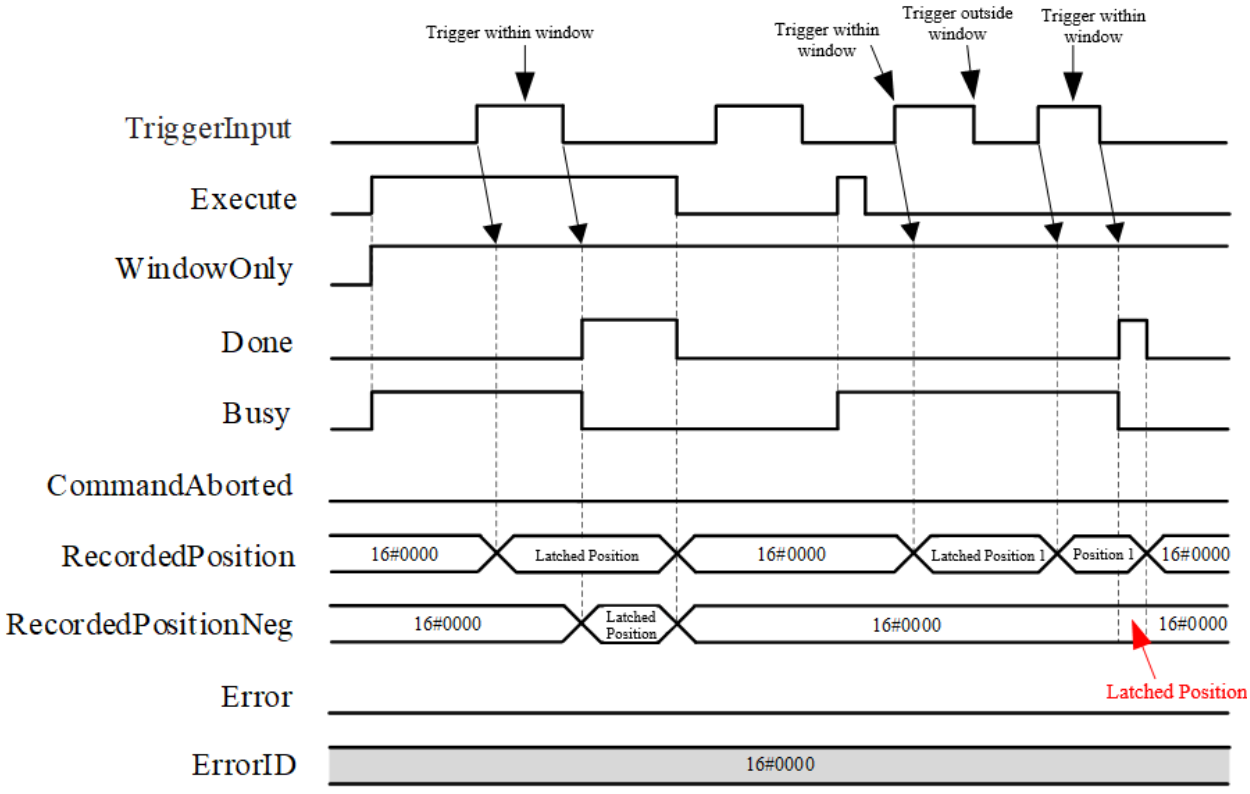
(2) The falling edge of the probe 1 is valid, and the DI terminal is triggered in single-trigger mode. The window function is valid.



(3) The rising and falling edges of probe 1 are valid, the DI terminal is triggered in single-trigger mode, and the window function is invalid.

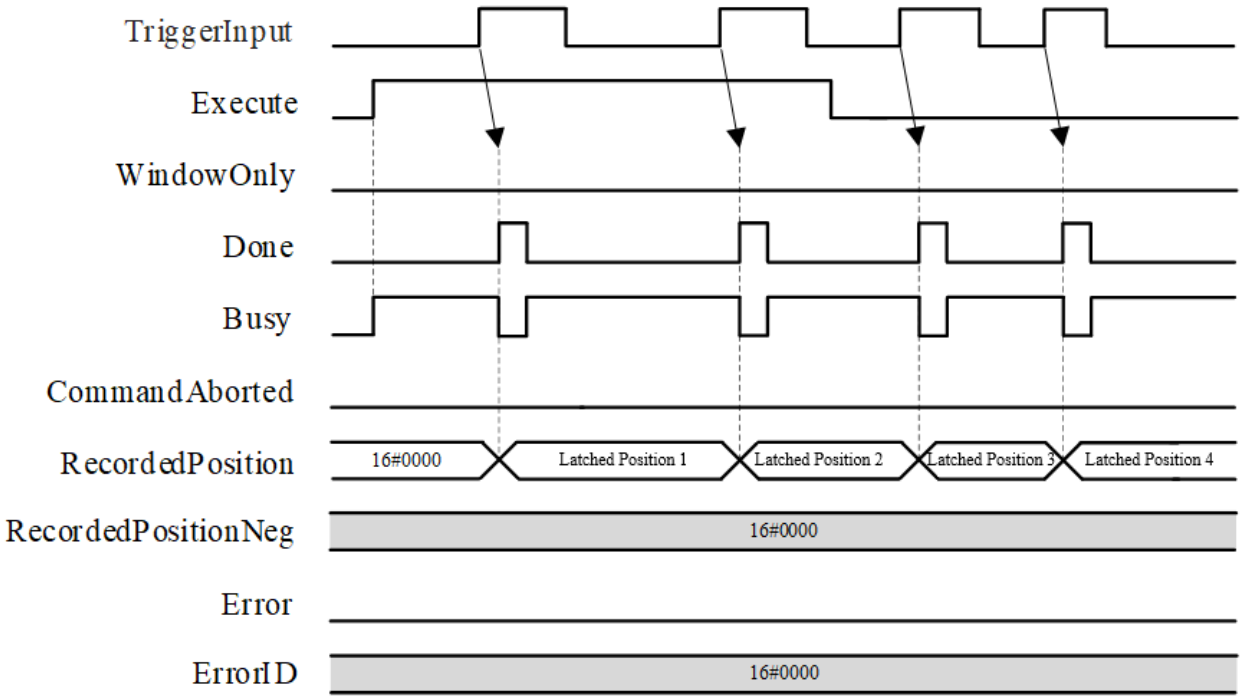


(4) The rising and falling edges of probe 1 are valid, the DI terminal is triggered in single-trigger mode, and the window function is valid.

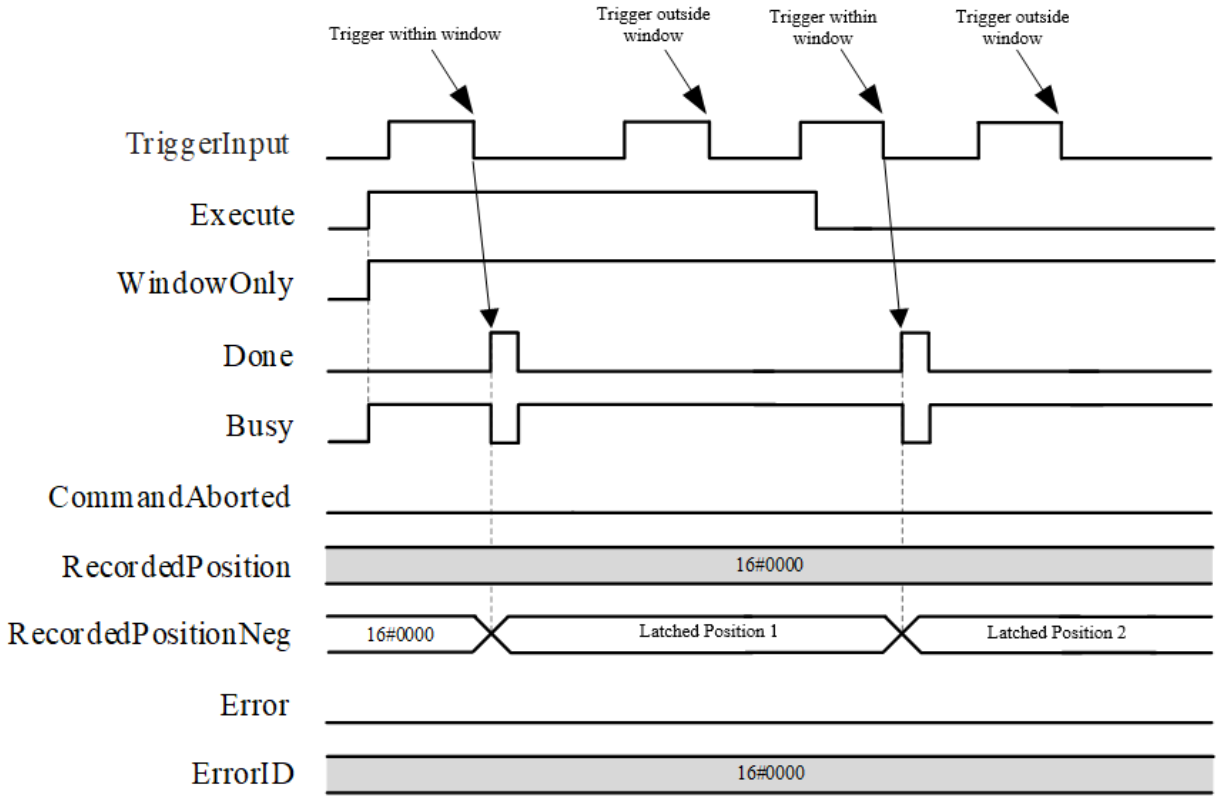


(5) The rising edge of probe 1 is valid, the DI terminal is triggered in continuous trigger mode, and

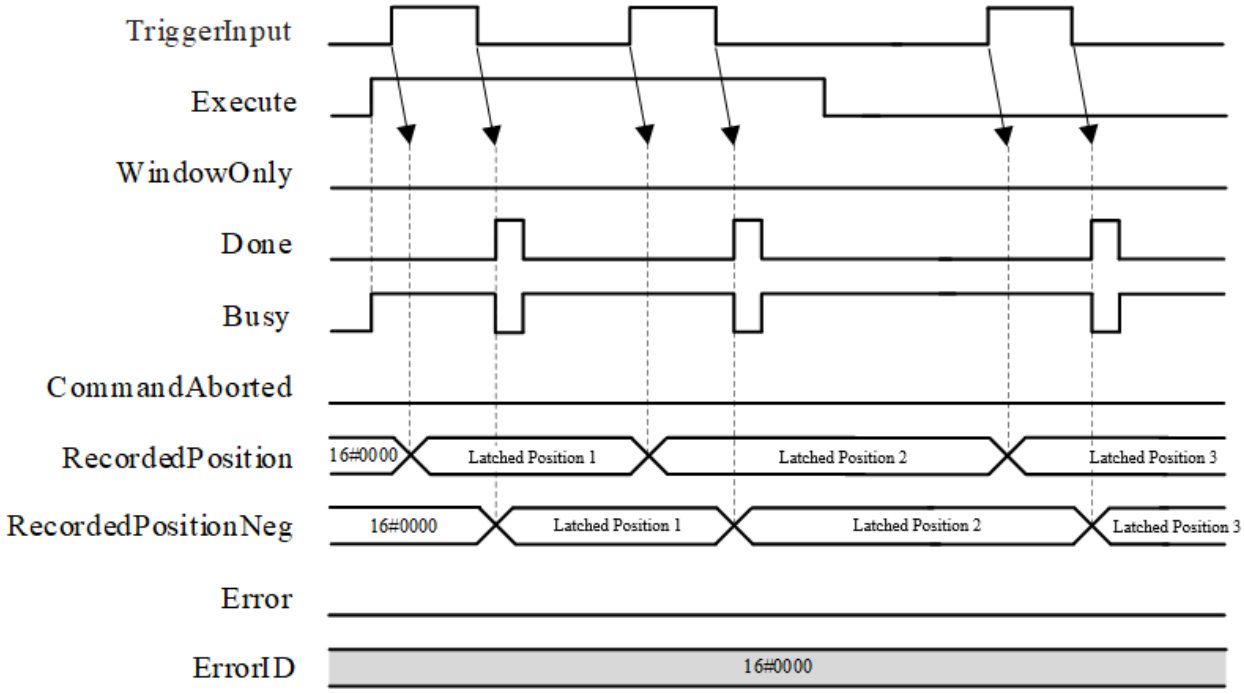
the window function is invalid.



(6) The falling edge of probe 1 is valid, the DI terminal is triggered in continuous trigger mode, and the window function is valid.

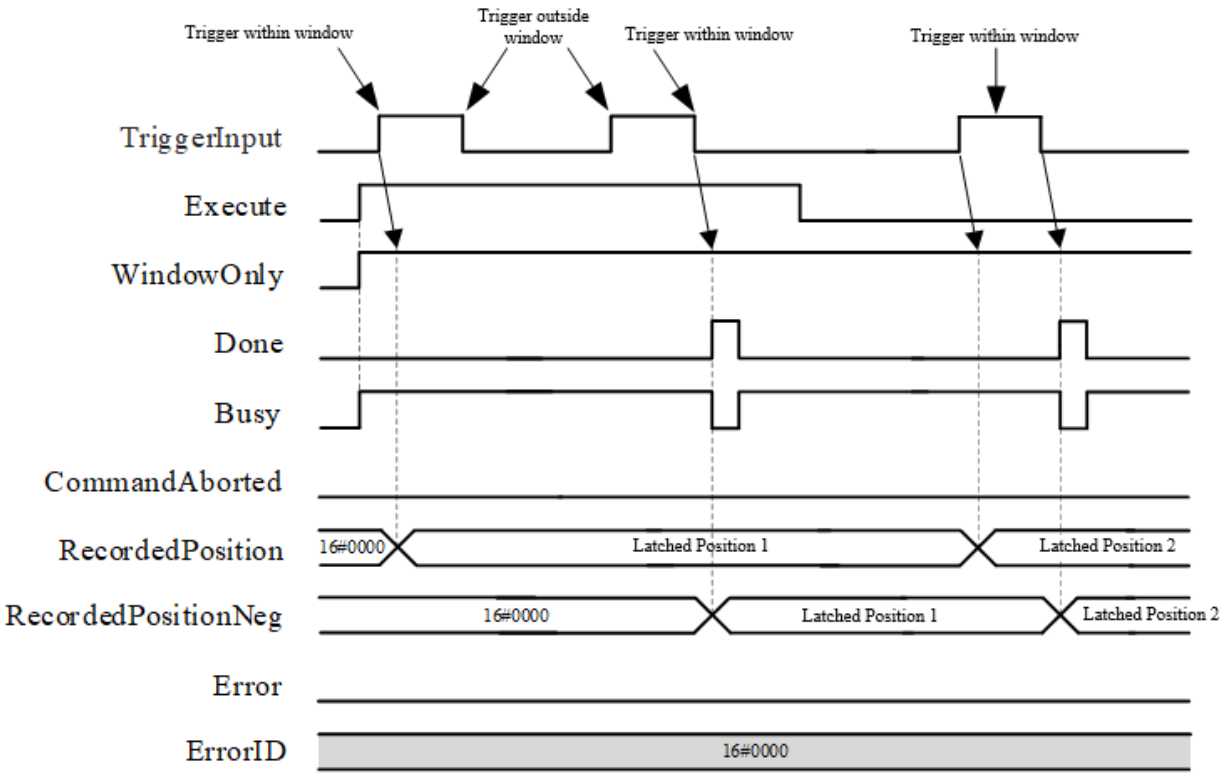


(7) The rising and falling edges of probe 1 are valid, the DI terminal is triggered in continuous trigger mode, and the window function is invalid.

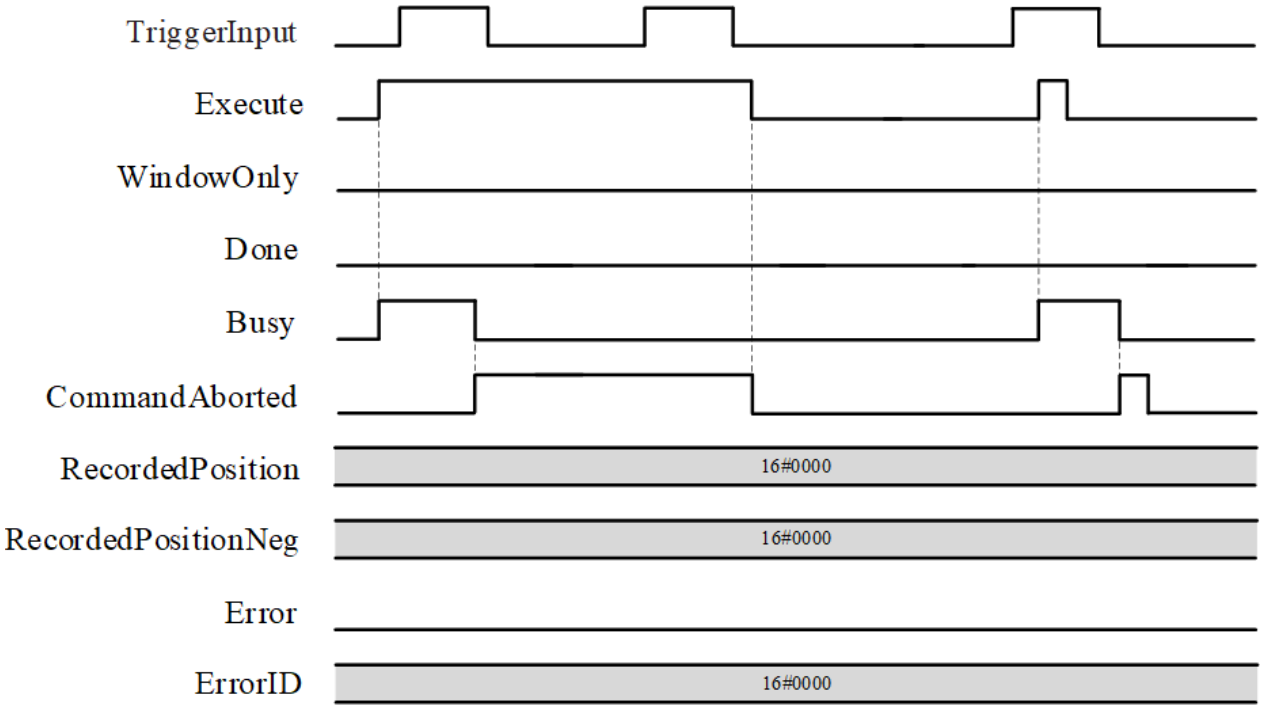


(8) The rising and falling edges of probe 1 are valid, the DI terminal is triggered in continuous

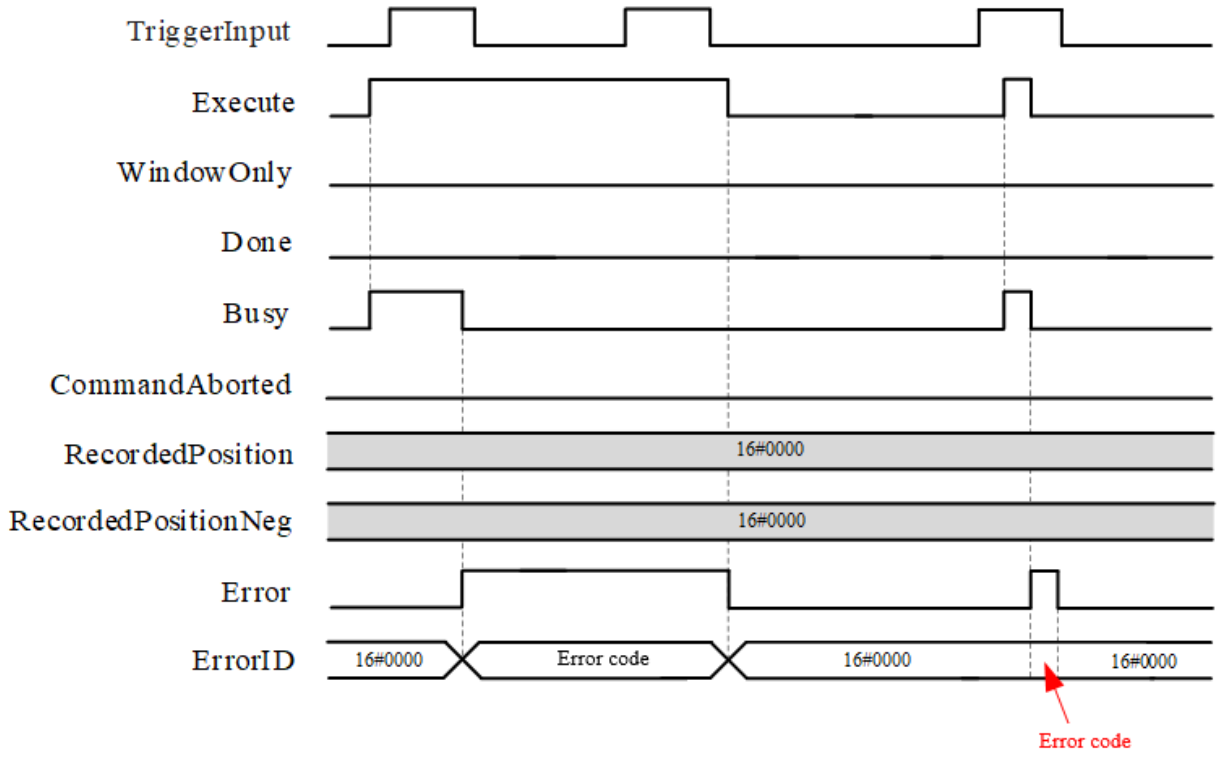
trigger mode, and the window function is valid.



(9) The rising edge of probe 1 is valid, the DI terminal is triggered, and the window function is invalid, which is interrupted by MC_AbortTrigger command.



(10) The rising edge of the probe 1 is valid, the DI terminal is triggered, the window function is invalid, and the command fails.



5.25.3 Examples

This example uses the function block MC_TouchProbe to grab EtherCAT servo probe signals and record positions.

The trigger conditions are set as follows:

- Use probe 1;
- Triggered by rising edge;
- The trigger signal is the driver IO terminal;
- Continuous triggering;
- No window trigger is used.

Before using the probe capture function, you need to use the SMC_AxisInit command for axis type setting and user unit configuration, and also configure the relevant PDO mapping.

The DI function needs to be configured on the servo driver side as the trigger signal of Probe 1. Please refer to the servo related manual for details.

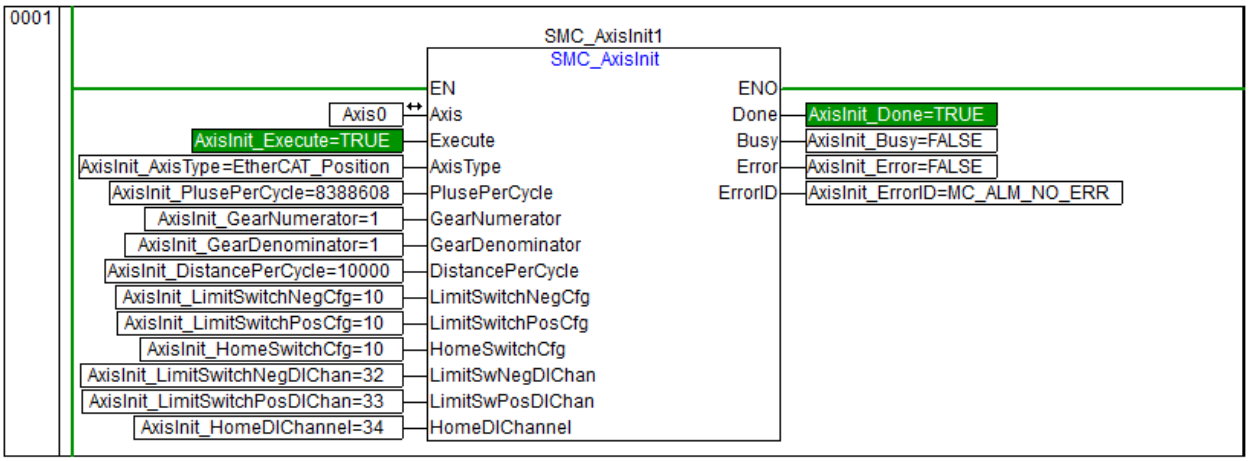
- Variable

No.	Variable Name	Variable Type	Initial Value	Variable Description
1	MC_TouchProbe0	MC_TOUCHPROBE	FALSE	Probe capture command
2	TriggerInput0	MC_TRIGGER_REF	FALSE	Probe Trigger Input Definitions
3	TouchProbe_En	BOOL	FALSE	Enable probe capture, and the rising edge is valid.

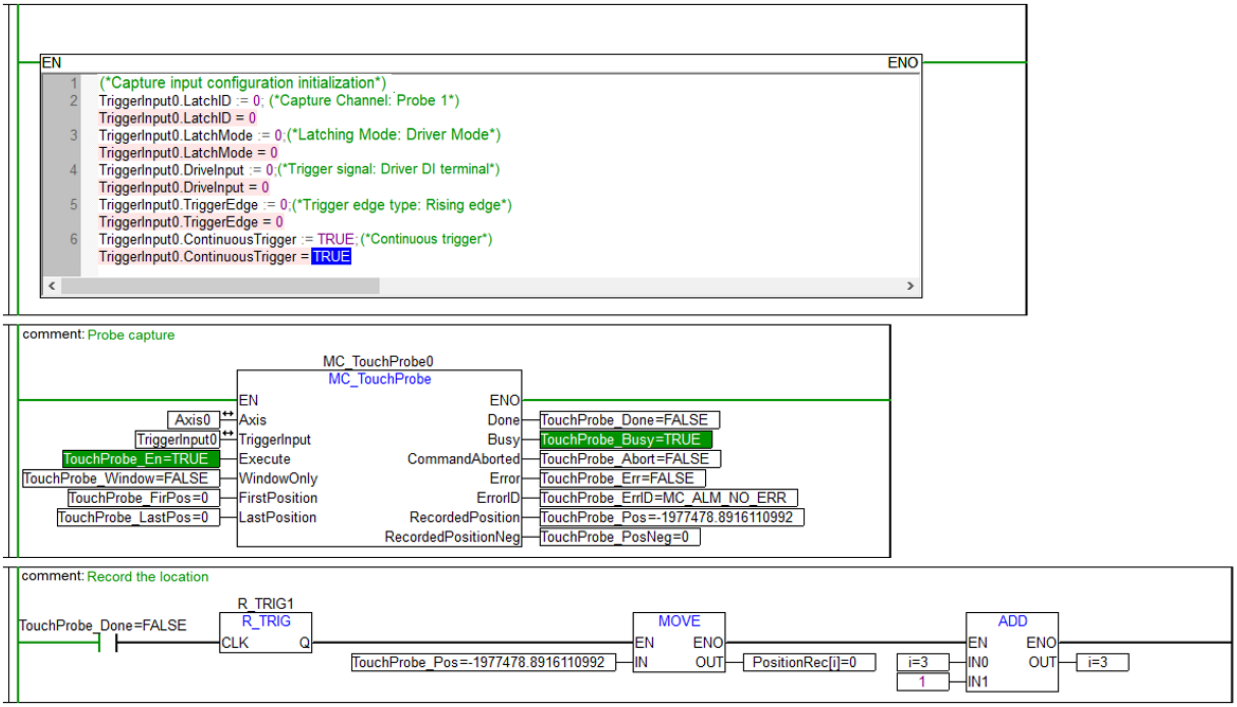
4	TouchProbe_Window	BOOL	FALSE	Enable window capture
5	TouchProbe_FirPos	LREAL	0	Window mode start position
6	TouchProbe_LastPos	LREAL	0	Window mode end position
7	TouchProbe_Done	BOOL	FALSE	Completion flag of probe capture command
8	TouchProbe_Busy	BOOL	FALSE	Probe capture command busy flag
9	TouchProbe_Abort	BOOL	FALSE	Flag of probe capture command interrupted
10	TouchProbe_Err	BOOL	FALSE	Probe capture command error flag
11	TouchProbe_ErrID	SMC_ERROR	MC_ALM.	Probe capture command error ID
12	TouchProbe_Pos	LREAL	0	Probe capture rising edge position
13	TouchProbe_PosNeg	LREAL	0	Probe capture commanded falling edge position
14	R_TRIG1	R_TRIG	FALSE	Rising edge trigger
15	PositionRec	ARRAY[0..10] OF LREAL		Position record array
16	i	INT	0	Variable

■ LD program implementation

(1) Call the SMC_AxisInit command to initialize axis-related information.



(2) After initializing TriggerInput0 structure, call MC_TouchProbe to capture probe position and record the captured position.



■ ST program implementation

(1) First call the MC_AxisInit command to initialize axis-related information

```
MC_ReadAxisInfo1(
Enable := ReadAxisInfo_En,
Axis := Axis0,
Valid=>ReadAxisInfo_Valid,
Busy=>ReadAxisInfo_Busy,
Error=>ReadAxisInfo_Error,
ErrorID=>ReadAxisInfo_ErrorID,
HomeAbsSwitch=>ReadAxisInfo_HomeAbsSwitch,
LimitSwitchPos=>ReadAxisInfo_LimitSwitchPos,
LimitSwitchNeg=>ReadAxisInfo_LimitSwitchNeg,
Simulation=>ReadAxisInfo_Simulation,
CommunicationReady=>ReadAxisInfo_ComRdy,
ReadyForPowerOn=>ReadAxisInfo_RdyPowerON,
PowerOn=>ReadAxisInfo_PowerON,
IsHomed=>ReadAxisInfo_Homed,
AxisWarning=>ReadAxisInfo_Warning,
AxisFault=>ReadAxisInfo_Fault);
```

(2) Capture input configuration initialization

```
TriggerInput0.LatchID := 0; (*Capture channel: probe 1 *)
TriggerInput0.LatchMode := 0; (*LatchMode:DriveMode*)
TriggerInput0.DriveInput := 0; (*Trigger Signal:DI Terminal of Drive *)
TriggerInput0.TriggerEdge := 0; (*Edge type: rising edge *)
TriggerInput0.ContinuousTrigger := TRUE;(*Continuous triggering*)
```

(3) Call MC_TouchProbe to capture probe position

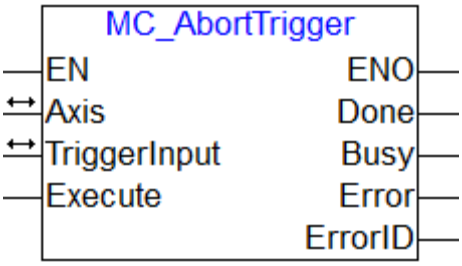
```
(*MC_TouchProbe commands*)
MC_TouchProbe0(
   Execute := TouchProbe_En,
   WindowOnly := TouchProbe_Window,
   FirstPosition := TouchProbe_FirPos,
   LastPosition := TouchProbe_LastPos,
   Axis := Axis0,
   TriggerInput := TriggerInput0,
   Done=>TouchProbe_Done,
   Busy=>TouchProbe_Busy,
   CommandAborted=>TouchProbe_Abort,
   Error=>TouchProbe_Err,
   ErrorID=>TouchProbe_ErrID,
   RecordedPosition=>TouchProbe_Pos,
   RecordedPositionNeg=>TouchProbe_PosNeg);
```

(4) Record the capture position.

```
R_TRIG1(CLK:=TouchProbe_Done);
(*Record Location*)
IF R_TRIG1.Q = TRUE THEN
   PositionRec[i] := TouchProbe_Pos;
   i:=i+1;
END_IF;
```

5.26 MC_AbortTrigger (Disable External Latch)

This command stops the position latch initiated by the MC_TouchProbe command.



5.26.1 Parameter

Parameter Description

Parameter Type	Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Axis	Axis name	No	-	-	AXIS_REF

	TriggerInput	Latching configuration parameters	No	-	See data structure MC_TRIGGER_REF	MC_TRIGGER_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
OUTPUT	Done	Completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

5.26.2 Functions

This command aborts position latch operations initiated by MC_TouchProbe. Non-EtherCAT axes are not supported.

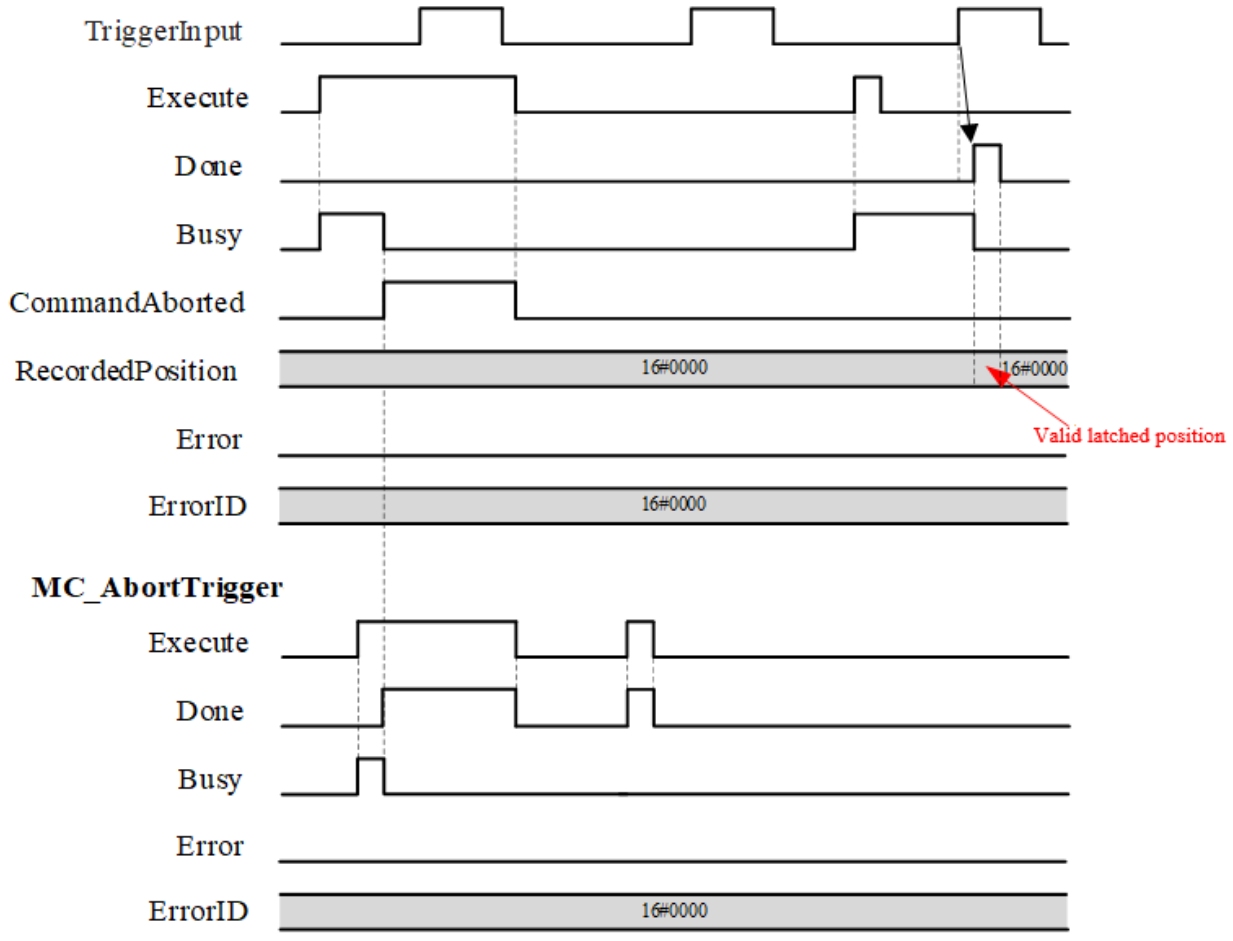
When this command is executed, only the position latch operation corresponding to the specified Axis and LatchID will be matched. If other configuration parameters are different from the input parameter of this command, the function of stopping position latch by this command will not be affected.

This command can only stop the position locking operation initiated by MC_TouchProbe, but cannot stop the internal position locking operation initiated by MC_Home and HMC_GantryInit commands. An error is reported in this command.

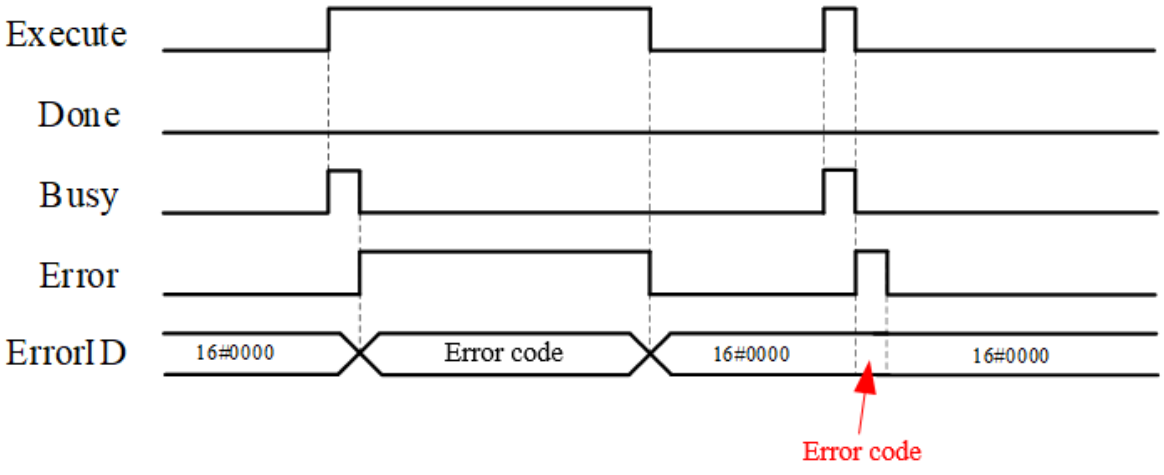
When the command is executed, if the position latch of the specified latch channel has been completed or the specified latch channel does not perform the position latch operation, the command will be directly and normally completed.

- Function block timing diagram

(1) The rising edge of touch probe 1 is valid, the DI terminal is triggered, and the window function is invalid in single-trigger mode. Use MC_AbortTrigger to stop the position capture of touch probe 1. At this time, the timing diagram is called by two commands in the same task.



(2) Timing diagram of failure during the execution of MC_AbortTrigger command.



5.26.3 Examples

In this example, the MC_AbortTrigger command is used to abort the position capture operation

initiated by the MC_TouchProbe command. After interruption, the CommandAborted pin of the MC_TouchProbe command is set to TRUE, and the Done pin of the MC_AbortTrigger command is set to TRUE.

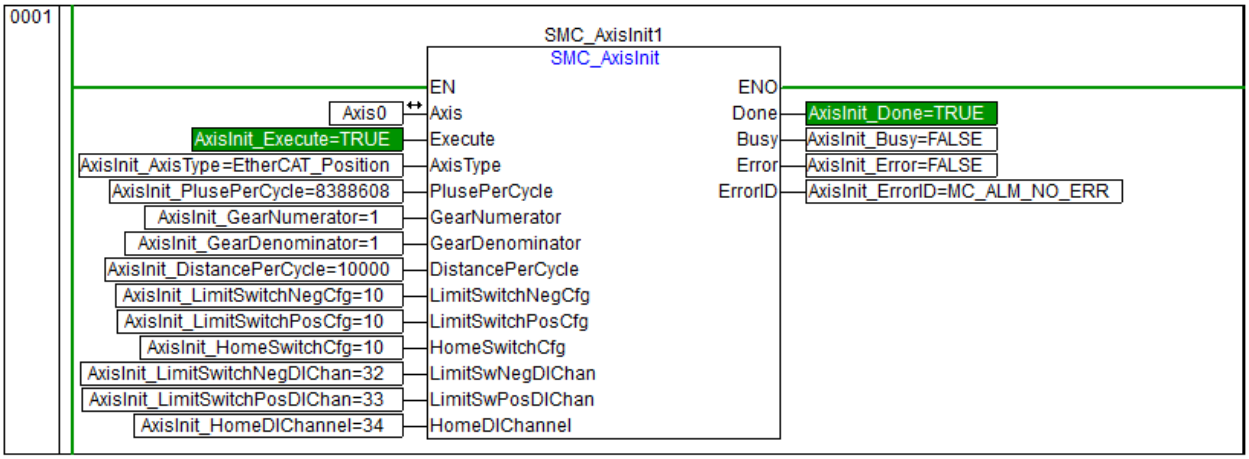
The example Description of MC_TouchProbe is consistent with that in the chapter of MC_TouchProbe command, so it will not be repeated in this chapter.

■ Variable

No.	Variable Name	Variable Type	Initial Value	Variable Description
1	MC_AbortTrigger0	MC_ABORTTRIGGER		Abort probe trigger command
2	AbortTrigger_En	BOOL	FALSE	Enabling mark of stop triggering command
3	AbortTrigger_Done	BOOL	FALSE	Completion flag of abort trigger command
4	AbortTrigger_Busy	BOOL	FALSE	Busy flag for abort triggering command
5	AbortTrigger_Error	BOOL	FALSE	Error flag for abort trigger command
6	AbortTrigger_Error	MC_ERROR	MC_ALM_NO_ERROR	Error ID of abort trigger command

■ LD program implementation

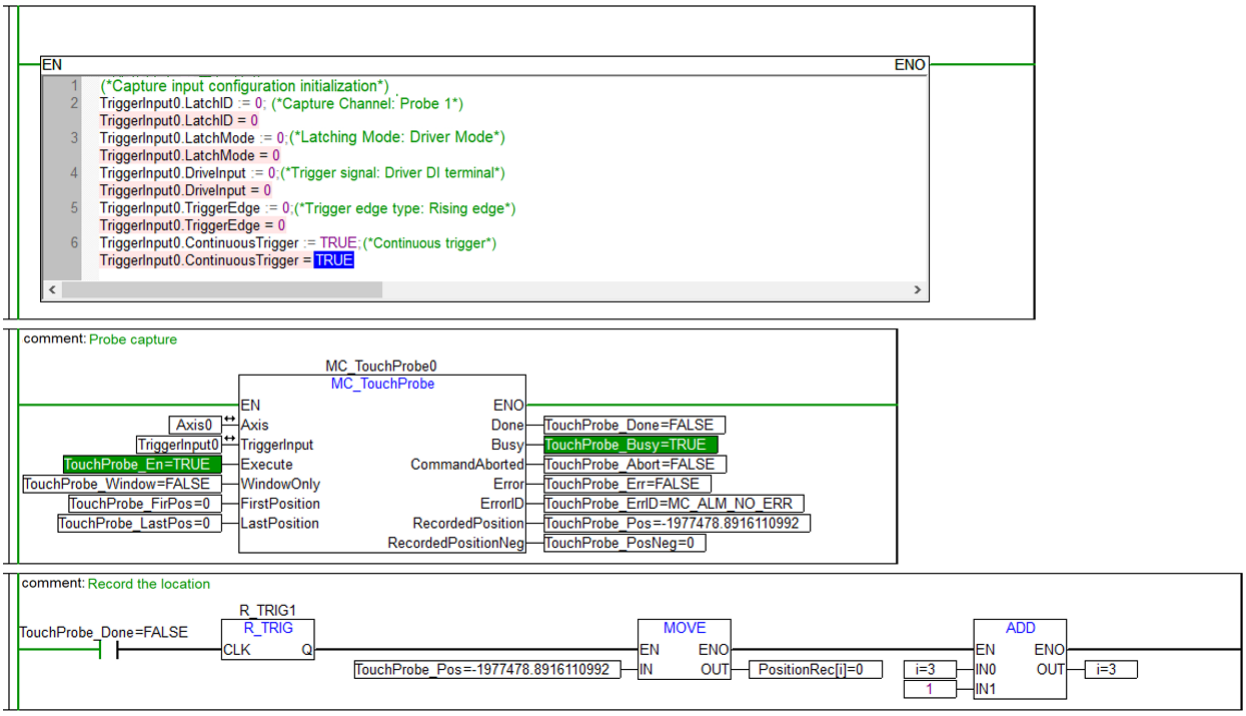
(1) First, call the SMC_AxisInit command to initialize axis-related information.

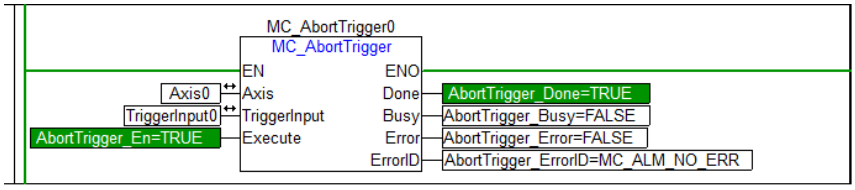


(2) After initializing the TriggerInput0 structure, call MC_TouchProbe to capture probe position.

A call to the MC_AbortTrigger command interrupts probe capture while the MC_TouchProbe command is in the Busy status.

After interruption, the MC_AbortTrigger command Done is TRUE and the MC_TouchProbe command CommandAborted is TRUE.





■ ST program implementation

(1) Call the MC_AxisInit command to initialize axis-related information.

```
MC_ReadAxisInfo1(
Enable := ReadAxisInfo_En,
Axis := Axis0,
Valid=>ReadAxisInfo_Valid,
Busy=>ReadAxisInfo_Busy,
Error=>ReadAxisInfo_Error,
ErrorID=>ReadAxisInfo_ErrorID,
HomeAbsSwitch=>ReadAxisInfo_HomeAbsSwitch,
LimitSwitchPos=>ReadAxisInfo_LimitSwitchPos,
LimitSwitchNeg=>ReadAxisInfo_LimitSwitchNeg,
Simulation=>ReadAxisInfo_Simulation,
CommunicationReady=>ReadAxisInfo_ComRdy,
ReadyForPowerOn=>ReadAxisInfo_RdyPowerON,
PowerOn=>ReadAxisInfo_PowerON,
IsHomed=>ReadAxisInfo_Homed,
AxisWarning=>ReadAxisInfo_Warning,
AxisFault=>ReadAxisInfo_Fault);
```

(2) Capture input configuration initialization.

```
TriggerInput0.LatchID := 0; (*Capture channel: probe 1 *)
TriggerInput0.LatchMode := 0; (*LatchMode:DriveMode*)
TriggerInput0.DriveInput := 0; (*Trigger Signal:DI Terminal of Drive *)
TriggerInput0.TriggerEdge := 0; (*Edge type: rising edge *)
TriggerInput0.ContinuousTrigger := TRUE;(* Continuous trigger*)
```

(3) Call MC_TouchProbe to capture the probe position.

```
(*MC_TouchProbe commands*)
MC_TouchProbe0(
   Execute := TouchProbe_En,
   WindowOnly := TouchProbe_Window,
   FirstPosition := TouchProbe_FirPos,
   LastPosition := TouchProbe_LastPos,
   Axis := Axis0,
   TriggerInput := TriggerInput0,
   Done=>TouchProbe_Done,
   Busy=>TouchProbe_Busy,
   CommandAborted=>TouchProbe_Abort,
   Error=>TouchProbe_Err,
   ErrorID=>TouchProbe_ErrID,
   RecordedPosition=>TouchProbe_Pos,
   RecordedPositionNeg=>TouchProbe_PosNeg);
```

(4) Record the capture position.

```
R_TRIG1(CLK:=TouchProbe_Done);
(*Record Location*)
IF R_TRIG1.Q = TRUE THEN
   PositionRec[i] := TouchProbe_Pos;
   i:=i+1;
END_IF;
```

(5) When the MC_TouchProbe command is in Busy status, call the MC_AbortTrigger command to interrupt probe capture.

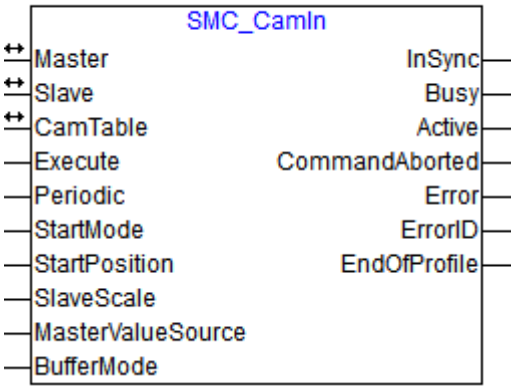
After interruption, the MC_AbortTrigger command Done is TRUE and the MC_TouchProbe command CommandAborted is TRUE.

```
(*MC_AbortTrigger Command*)
MC_AbortTrigger0(
   Execute := AbortTrigger_En,
   Axis := Axis0,
   TriggerInput := TriggerInput0,
   Done=>AbortTrigger_Done,
   Busy=>AbortTrigger_Busy,
   Error=>AbortTrigger_Error,
   ErrorID=>AbortTrigger_ErrorID);
```

Chapter 6 Synchronous Motion Instructions

6.1 SMC_CamIn (Electronic CAM)

This command realizes the electronic cam function.



6.1.1 Parameter

Command Parameters

Parameter Type	Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Master	Master Axis name	No	-	-	AXIS_REF
	Slave	Slave Axis Name	No	-	-	AXIS_REF
	CamTable	Cam gauge	No	-	See data structure SMC_CAM_TABLE	SMC_CAM_TABLE
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	Periodic	FALSE: Single mode TRUE: Circulation mode	YES	FALSE	TRUE/FALSE	BOOL
	StartMode	Startup mode 0: Absolute position start (triggered by passing through the absolute position in forward direction) 2: Start immediately 3: DI trigger startup	YES	0	0:mcAbsolute 10:mcImmediately 11:mcDITrigger	MC_CAMIN_STARTMODE
	StartPosition	When the absolute position is started, it is the starting position;	YES	0	Absolute position start: positive/negative/0; DI trigger start:	LREAL

		When DI trigger is started, it is the bit offset mapped by DI point in Area I. For example, the bit offset of %IX100.1 in area I is 100×8+1.			0 ~ (field I byte capacity * 8)-1	
	SlaveScale	Cam follower position data scale factor	YES	1	Positive value	LREAL
	MasterValueSource	Master Position Source 1: Synchronize with master feedback position	YES	1	1: mcActualValue	MC_SOURCE
	BufferMode	Buffer mode 0: Interrupt 1: buffer	YES	1	0: Aborting 1: Buffered	MC_BUFFER_MODE
OUTPUT	InSync	Synchronizing	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Active	In control, TRUE when the command is executed	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR
	EndOfProfile	Cam Cycle Complete	YES	FALSE	TRUE/FALSE	BOOL

SMC_CAM_TABLE Data Structure

Member	Description	Default Value	Scope	Data Type
PointerToPos	A pointer to an array of cam positions (the position array is a two-dimensional array, the first dimension being the master position and the second dimension being the slave axis position)	0	-	POINTER TO ARRAY OF LREAL
PointsNum	Number of positions	0	3 or more	UDINT
CurveType	Cam curve interpolation mode 1: straight line 3:3rd order curve	1	1:smcStraightLine 3:smcPolynomic3	USINT

The SMC_CAM_TABLE data structures include:

(1) The array of cam positions is a two-dimensional array ArrayPos[2][PointsNum]. The cam position array is completed by selecting discrete coordinate points on the target motion track (two-dimensional) of master and slave axes. PointerToPos is the pointer to this two-dimensional array.

(2) The number of SMC_CAM_TABLE data structures should be greater than or equal to 3, and the number of positions should strictly correspond to the length of the cam position array.

(3) The cam curve interpolation mode supports linear interpolation and cubic curve interpolation.

- Note: In single shot mode, after the cam is activated: for immediate start and DI triggered start,

the commanded left boundary is at the position of cam start; For absolute position start, the commanded left boundary is at the set absolute position; Commanded right boundary = left boundary + length of cam travel section. The order is cancelled when the master moves negatively across the left boundary or positively across or reaches the right boundary.

6.1.2 Functions

This command realizes the electronic cam function.

- Command function description

(1) This command can make the master and slave shaft synchronize cam actions according to the cam position array.

(2) The rising edge triggering of this command is valid. At the rising edge of the Execute input, the input parameter is legal. When there is no abnormality in the axis, the value of the input parameter will be latched. During the execution of the command, modifying the latched parameter is invalid until another rising edge triggers this command.

(3) The cam position array in the SMC_CAM_TABLE data structure needs to be compiled before running this command.

(4) The cam position array is compiled according to the discrete coordinate points on the target motion track (two-dimensional) of the master and slave shafts selected, and the cam approaches the target motion track through the linkage of the master and slave shafts.

(5) Master position source. The master position source MasterValueSource can only be set to 1, and the slave axis command position is synchronized with the feedback position of the master.

(6) The slave camshaft supports the following shaft types:

- Virtual axis (0): virtual axis
- EtherCAT_Position(50): EtherCAT bus connecting shaft, position control
- EtherCAT_Speed(51): EtherCAT bus connecting shaft, speed control
- EtherCAT_Torque(52): EtherCAT bus connecting shaft, torque control

- Description of Cam Position Array

(1) Position relationship between master and slave camshaft position array:

The master and slave camshaft positions are pre-stored in the master and slave shaft position array. The first dimension of the two-dimensional cam position array is the master position, and the second dimension is the slave position, which are in a one-to-one correspondence.

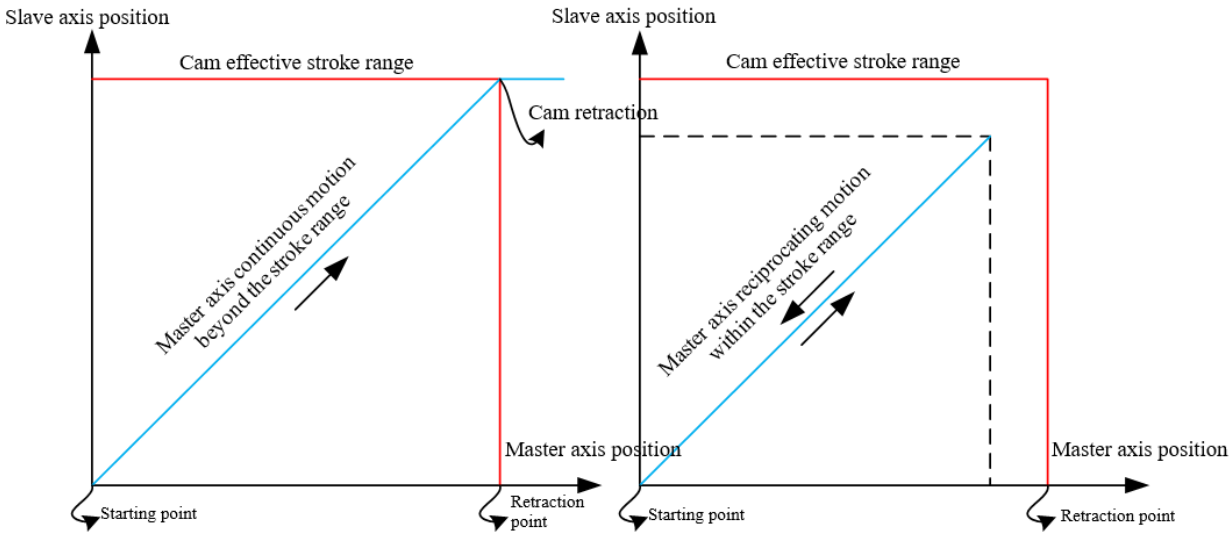
(2) Cam stroke

The stroke length of a master camshaft is the last element value in the array of master camshaft positions minus the first element value.

In single mode, after the cam is activated: for immediate start and DI trigger start, the left boundary of the command is set to the position at the time of cam activation; For absolute

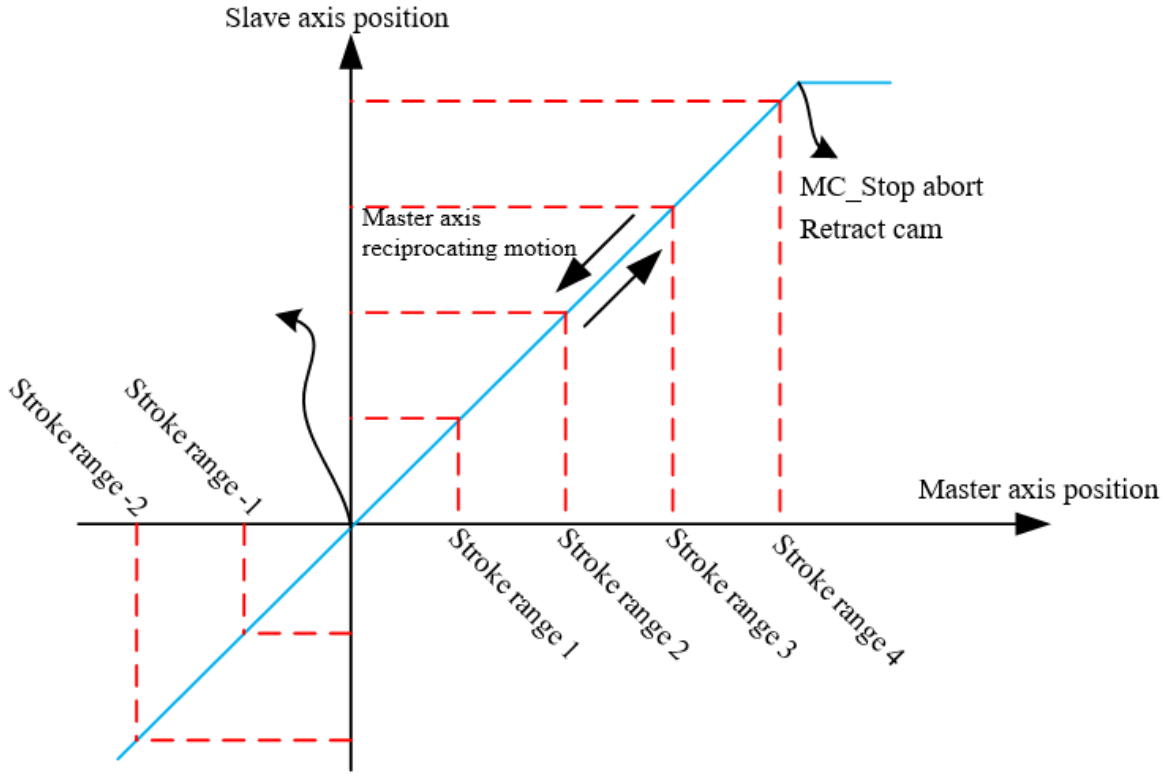
position start, the left boundary is set to the specified absolute position; The right boundary of the command= left boundary + length of cam stroke section. The command will be canceled if the master moves in the negative direction across the left boundary or in the positive direction across or reaches the right boundary.

When the single master camshaft is within the effective stroke range, the master can perform unidirectional and reciprocating motion. When the master reciprocates within the effective stroke range, the slave shaft will also perform reciprocating linkage action. The following figure shows the motion track of master and slave shafts started immediately by a single cam.



Cyclic immediate start of cam-synchronized master and slave axis motion trajectory

For recirculating cams, when the master camshaft position exceeds the stroke length of a master camshaft, it will enter the next cam stroke. At this time, the slave shaft will take the slave shaft position at the end of the previous cam stroke as the starting point, and make incremental displacement calculation on this basis. Repeat the linkage action of the previous cam stroke in turn. The master camshaft can also do unidirectional and reciprocating motion. If it is necessary to cancel the circulating cam, MC_Stop command can be executed to stop and cancel the cam action. For detailed use method, please refer to the Description of MC_Stop command. The following figure shows the motion track of master and slave shafts immediately started by the circulating cam.



Cyclic immediate start of cam-synchronized Master and slave axis motion trajectory

(3) Cam activation position

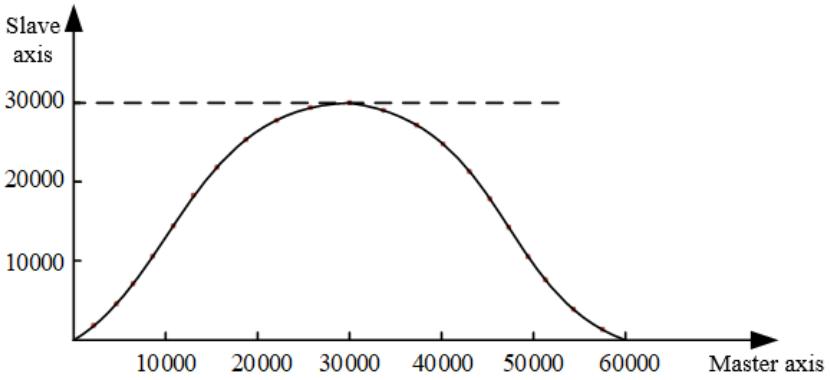
After the command activation, wait for the master to reach the StartPosition.

When the master passes through StartPosition, the starting point of the cam position array is executed and the cam command output variable InSync (in cam synchronization) becomes TRUE.

The master and slave positions of a cam position array are specified in relative quantities from the starting point. Interpolate the two cam position data according to the set cam curve interpolation method, and calculate the slave axis position corresponding to the master position, as shown in the following figure.

Cam table

Master axis	Slave axis
0	0
10000	12000
20000	21000
30000	30000
40000	25000
50000	11000
60000	0



(4) Abnormal cam position array

An exception is detected triggering this command when the specified cam position array does not exist in the Controller.

(5) Use of cam position array

The same array of cam positions can be specified for several axes. Note: During the execution of this command, when the user rewrites the position of the master in the cam position array to non-incremental data, an error exception will occur when running this command.

(6) Cam activation

This command can be activated in any state of StandStill, position control, speed control and synchronous control.

The cam start mode includes 0: fixed position start, 10: immediate start and 11: DI trigger start.

□ Fixed position activation

Cam activation mode is 0: start at fixed position. After the start command, wait for the master camshaft Mpos to cross the cam start position StartPosition from the negative direction and execute the starting point of the cam table. The output variable InSync becomes TRUE.

A cam position array specifies the positions of the master and slave in relative terms, i.e., from the absolute position of each axis at the beginning of the cam position array. For example, in the cyclic stroke mode with a master movement stroke of [0 to 360), when StartPosition is 100. °As shown below, the absolute position of a master is the value of the master position from the cam position array plus StartPosition, and the absolute position of a slave axis is the value of the slave axis displacement from the cam position array plus the slave absolute position at the beginning of the cam position array.

Cam position array

Master axis	Slave axis
0	0
60	1000
120	1500
180	2000
240	1500
300	1000
360	0

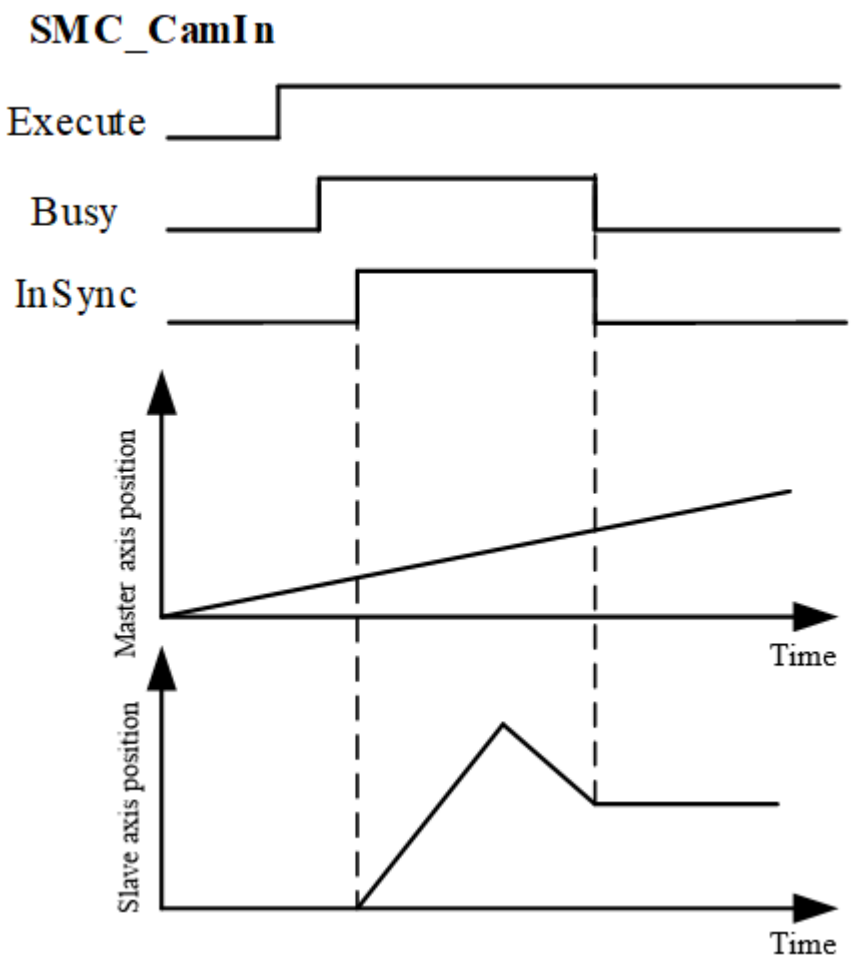
StartPosition=100

Absolute positions of master and slave axes

Master axis	Slave axis
100	0+ Absolute position of the slave axis at the start of the cam position array
160	1000+Absolute position of the slave axis at the start of the cam position array
220	1500+Absolute position of the slave axis at the start of the cam position array
280	2000+Absolute position of the slave axis at the start of the cam position array
340	1500+Absolute position of the slave axis at the start of the cam position array
40	1000+Absolute position of the slave axis at the start of the cam position array
100	0+Absolute position of the slave axis at the start of the cam position array

- Immediate start

The cam start mode is 10: immediate start. The cam action starts immediately after the start command. At this time, StartPosition has no meaning.



- DI trigger start

The cam start mode is 11:DI triggered start. After the cam command is issued, when a set DI signal is 1, the cam will be started. The starting position is the Mpos position of the master when

the program detects that DI is 1. StartPosition is the bit offset mapped by DI point in Area I. For example, the bit offset of %IX100.1 in area I is 100×8+1, i.e. StartPosition=801.

When DI triggers start, the value of StartPosition shall be less than the capacity of Run Controller I area.

(7) Periodic mode

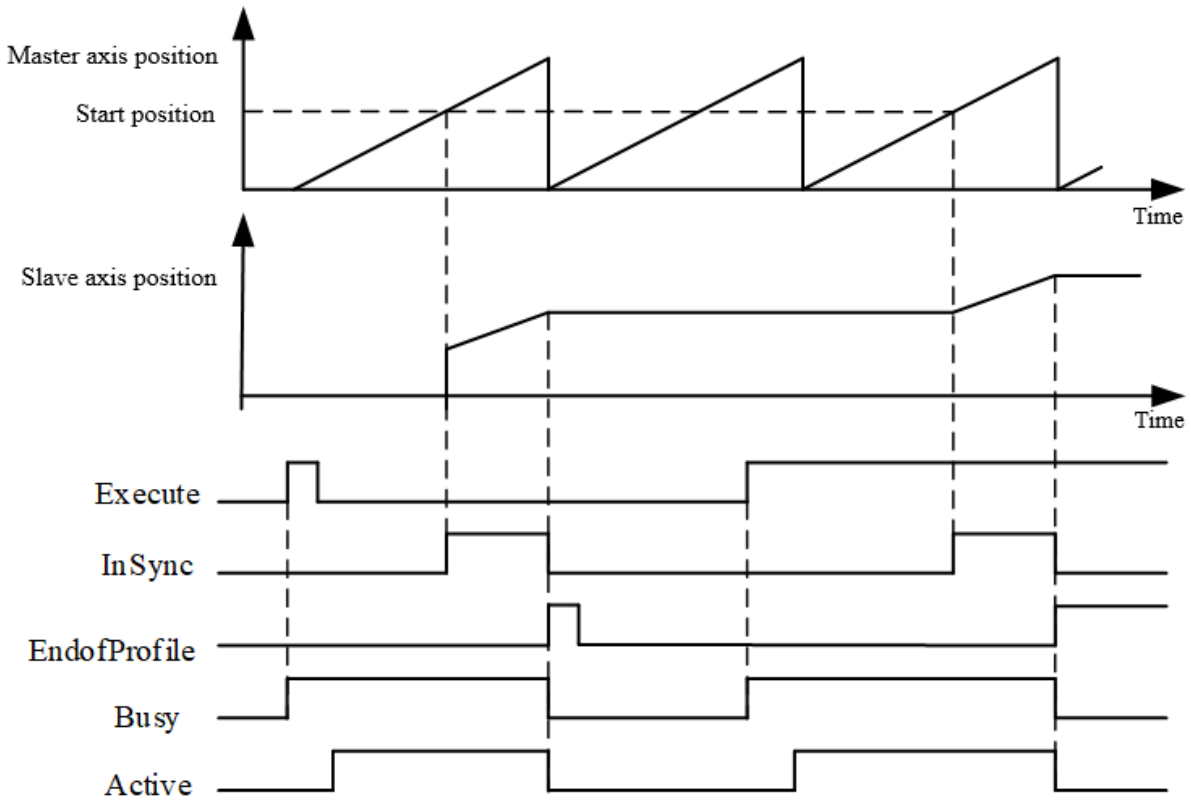
According to the boundary range of master position, cams are divided into single-time cam and cyclic cam, which are determined by input parameter Periodic. Single mode, Periodic=FALSE: The command is cancelled after the master moves across the left boundary in negative direction or crosses the right boundary in positive direction.

Cyclic cam, Periodic=TRUE. When the master camshaft position exceeds the stroke length of a master camshaft, it will enter the next cam stroke and continue to run in cycles.

□ Single cam

After the cam is activated, if the position of the master camshaft is within the effective stroke range, the cam can maintain normal linkage operation. When the master exceeds the effective stroke range, the cam command will be automatically cancelled. After the cam is started and the triggering conditions of the cam are met, the cam command InSync pin is set to TRUE, and the axis starts moving. After a single cam command is completed, the EndofProfile pin is set to TRUE and kept for at least one cycle, and the Busy and Active pins become FALSE. After encountering the falling edge of Execute, all output pins return to Default Values.

The following figure shows the pin timing at fixed position startup in single mode:



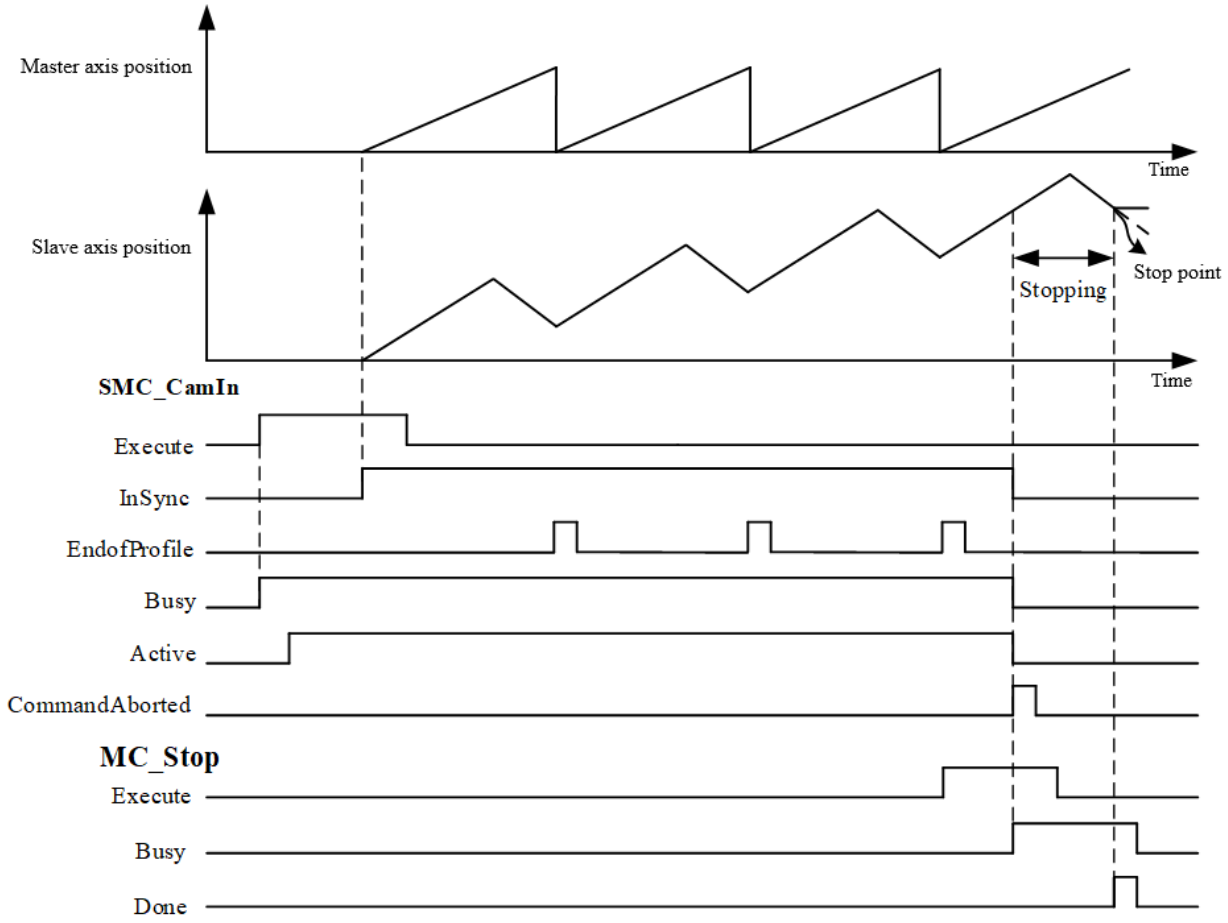
□ Recirculating cam

The recycling master camshaft position is not limited by a boundary range, and the cam function remains active after start if no stop withdraw command is executed.

The slave camshaft performs corresponding linkage action with the master. When a cam stroke of the master ends, it enters the next cam stroke. Incremental calculation is performed from the end point of the previous cam stroke of the master as the starting point to execute periodic linkage action. At the end of each cam stroke, the output pin EndofProfile is set high for one cycle.

When the Stop Cycle Camshaft command is cancelled with the MC_Stop command, the cam output pin CommandAborted is now set to TRUE and the remaining output pins are reset. The slave decelerates and stops along the cam motion track.

The following figure shows the timing of the cam command pin in cycle mode and immediate start mode. The MC_Stop command is used to cancel the stop cycle cam command.



(8) Handling of repeated triggering of function block

□ The same cam command function block is triggered repeatedly

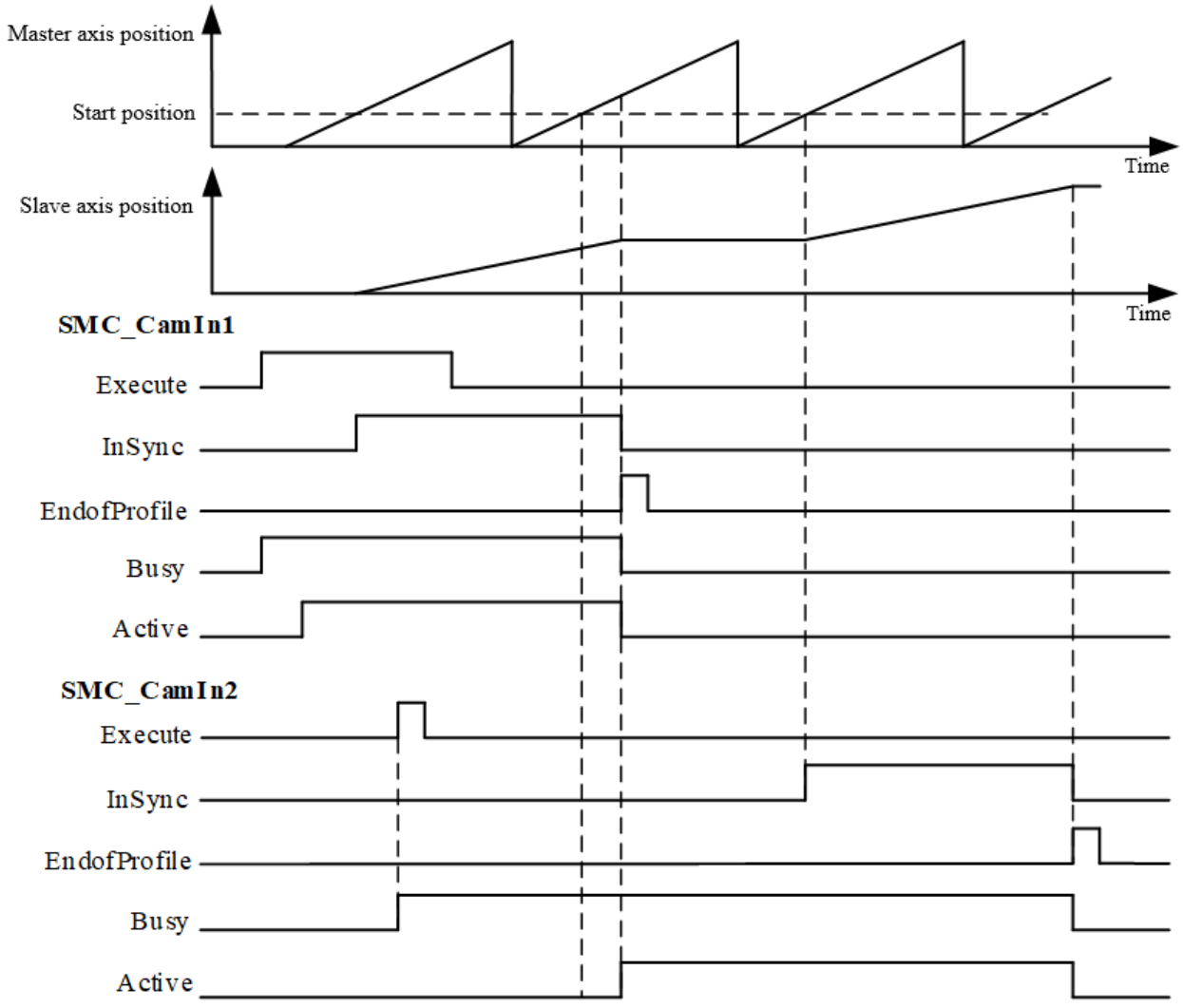
For the same cam command function block in operation, if the execution is triggered repeatedly, there are:

(a) If the BufferMode of the command is 0: Aborting, interrupt the currently executing command;

(b) If the BufferMode of the command is 1: Buffered, the command will be put into the motion buffer. The status monitored by the subsequent function block will be the running state of the next command, and the running state of the previous command will be out of supervision.

□ Buffer cam command waiting to run

When a cam command is already executing, the subsequent cam command will wait for the previous command to complete before it begins execution. In the example below, during execution of a SMC_CamIn1 (single cam, fixed position start) command, the start SMC_CamIn2 command is triggered repeatedly. At this time, the EndOfProfile pin of SMC_CamIn1 becomes TRUE, the Active pin of SMC_CamIn2 becomes True, the next StartPosition (cam start position) of is reached, InSync (synchronization in progress) becomes TRUE, and the cam operation begins.



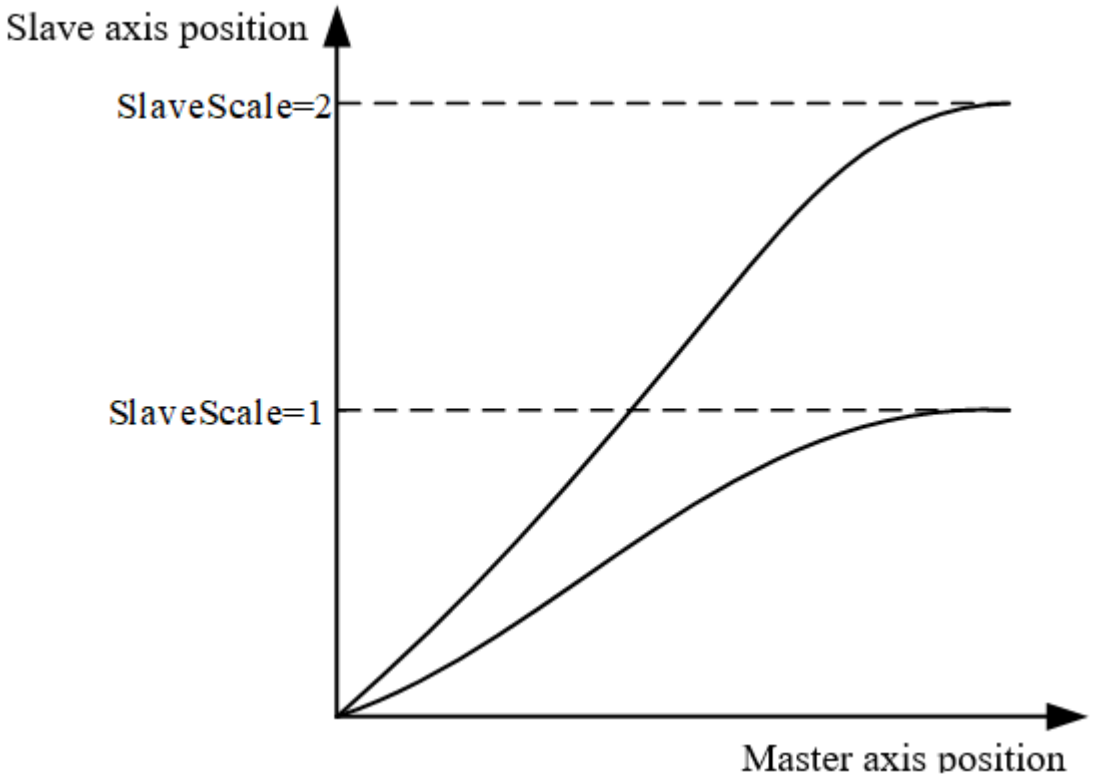
(9) Buffer mode selection

Specify the connection mode between the previous axis action and this axis action.

This command only supports 0: Aborting and 1: Buffered. When this commandn is sent, if BufferMode is 0: Aborting, the currently executing command will be immediately aborted and switched to this command. If BufferMode is 1:Buffered, the buffered command will be automatically started from the cycle when the currently executing command ends normally.

(10) Slave camshaft position data scale factor (scaling factor)

According to the relationship between master phase and slave axis displacement of the specified cam array, modifying the value of scale factor SlaveScale of cam and slave axis position data can increase or decrease the slave axis displacement by a specified ratio (SlaveScale).



Slave camshaft position data scale factor (SlaveScale)

(11) EndofProfile

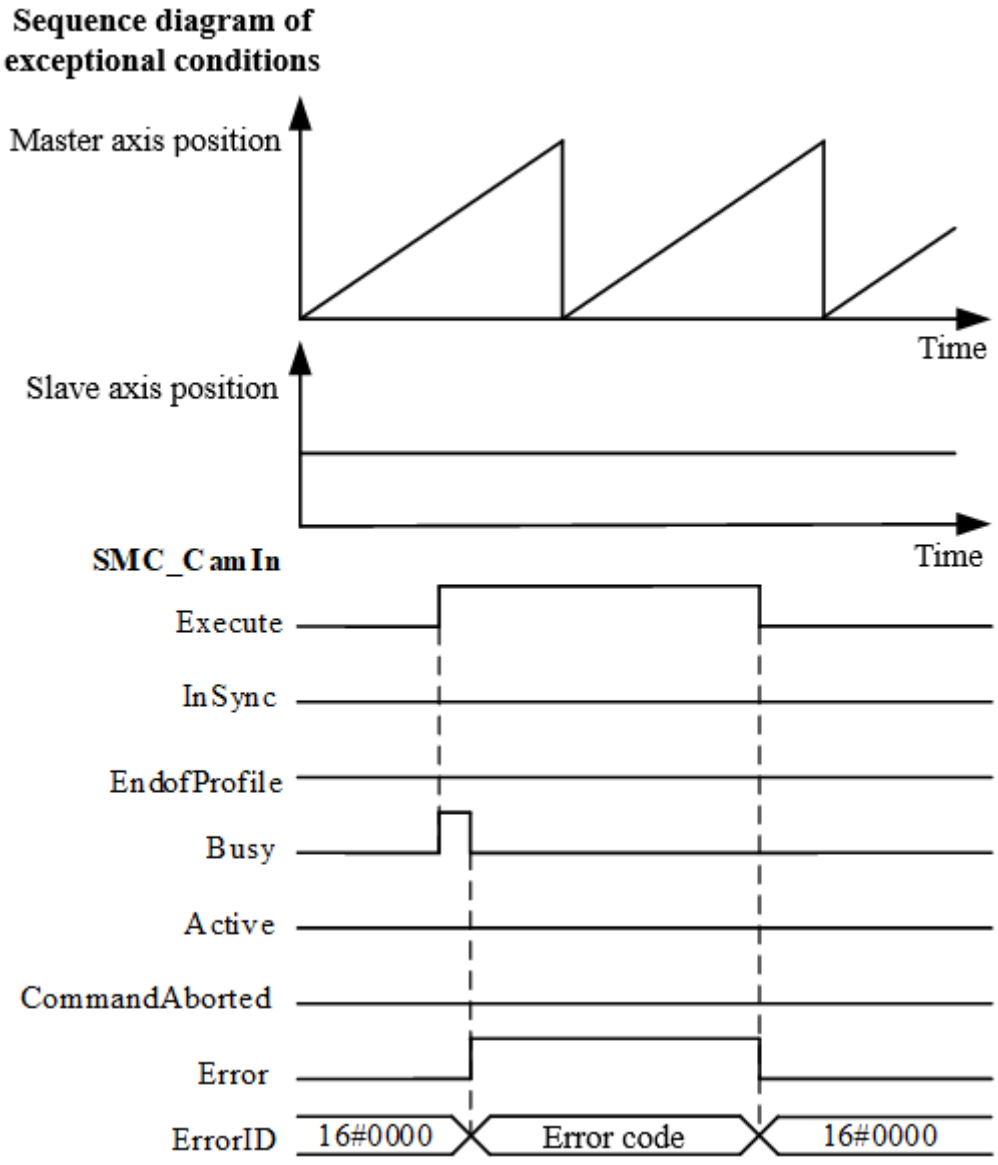
When the cam command is in single mode, the cam cycle is completed and the command output pin EndofProfile is set to TRUE and held until the input pin Execute becomes FALSE.

When the cam command is in cycle mode, the first cam stroke is completed, and the command output pin EndofProfile is set to TRUE for one cycle. The command enters the next cam stroke, and EndofProfile becomes FALSE until the cam stroke is completed. The EndofProfile pin is set to TRUE for one cycle, and EndofProfile is updated with the continuous change of the cyclic cam. The stop cycle cam command can be cancelled with the MC_Stop command.

Details of the EndofProfile pin timing are shown in the above timing diagram.

(12) Abnormal

When the cam command is activated, if an error occurs, the Error pin will be set to True, and the error code ErrorID will report an error value. Please refer to the appendix Error Code Description to obtain the cause of the error. When the command input pin Execute goes low, the output pin returns to its default value.



6.1.3 Example

In this example, MasterAxis is defined as the main axis, SlaveAxis as the slave axis, and the Variable Type is AXIS_REF.

The master executes the MC_MoveVelocity command, and the slave executes the MC_CamIn command. Retracting the cam command is stopped with MC_Stop command.

- LD program implementation

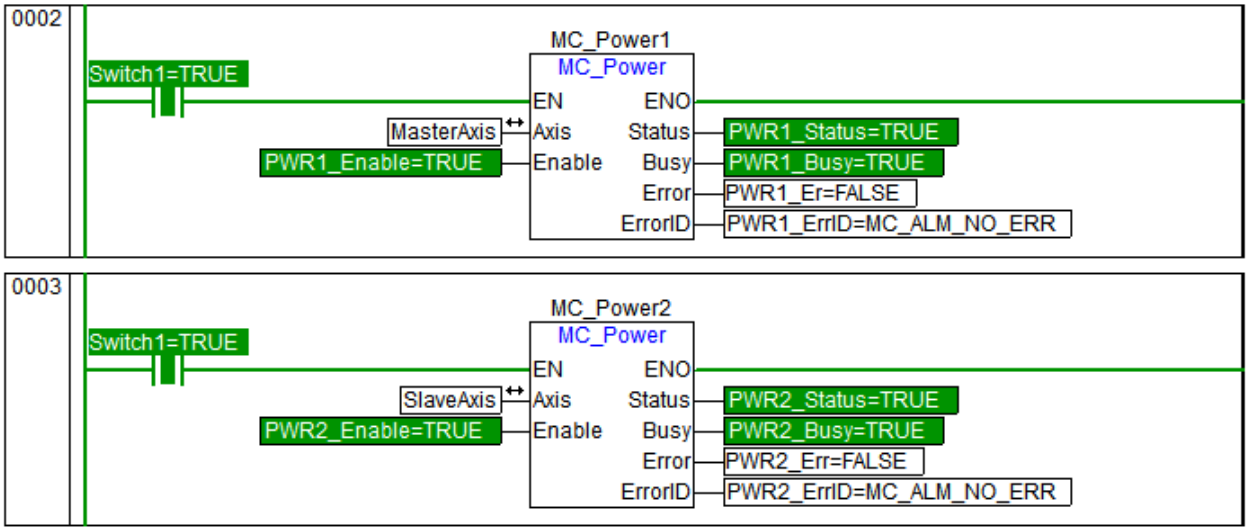
  In this example, the periodic mode is established and the cam command is started immediately. Set Periodic (Repeat Mode) = TRUE to repeat the action.

  (1) Primary variables of the example program

Variable Name	Variable Type	Initial Value	Notes
MasterAxis	AXIS_REF	-	Axis variables of the master
SlaveAxis	AXIS_REF	-	Axis variable of slave axis
Power1_Status	BOOL	FALSE	Variable for successful master enable. This variable becomes TRUE when the axis is enabled successfully.
Power2_Status	BOOL	FALSE	Variable that enabled the slave axis successfully. This variable becomes TRUE when the axis is enabled successfully.
MV_Exe	BOOL	FALSE	The variable triggered by the rising edge of the speed control command for an axis.
MV_InVel	BOOL	FALSE	Identification variable for the axis speed control command to reach the target speed. This variable becomes TRUE and the speed commanded reaches the target speed.
CamTable	SMC_CAM_TABLE	-	Cam Table Variables
ArrTablePos	ARRAY OF LREAL	-	An array of cam positions.
PointerToPos	POINTER TO ARRAY OF LREAL	-	A pointer variable to the array of cam positions, array position Pos[2][PosNum]. PosNum is the number of positions.
Cam_InSync	BOOL	FALSE	Synchronization variable of the cam command of the delivered slave axis. When the cam command is synchronized from the slave axis, this variable becomes TRUE.
Cam_EndProFile	BOOL	FALSE	Variable of completion of cam cycle instructed by the delivered slave axis cam; when the cam command cam cycle is completed, this variable becomes TRUE.
Switch1	BOOL	FALSE	Axis ready state variable. After axis initialization, Switch to TRUE and the axis is ready.

(2) Axis initialization. Set the AxisID of MasterAxis to 0, the AxisID of SlaveAxis to 1, and the axis type of both master and slave axes to 50:EtherCAT_Position. Call the SMC_AxisInit command to complete the initialization configuration of master and slave axes. For the SMC_AxisInit, command, please refer to its introduction.

(3) Axis enabling. In section 0002, after the master and slave axes are initialized and Switch1 is TRUE, execute the axis enabling command to complete the axis enabling.



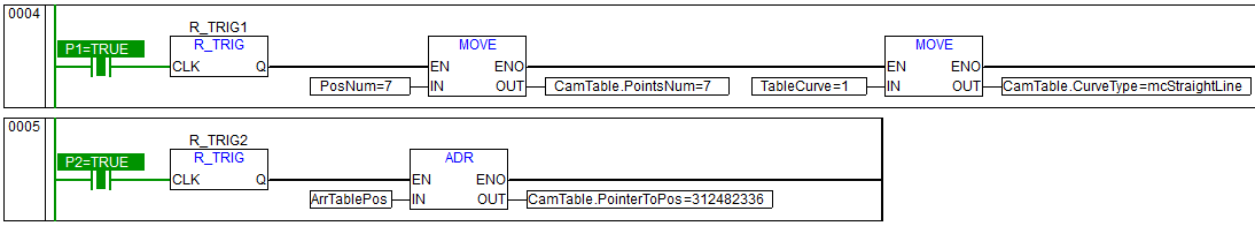
(4) Initialize the SMC_CAM_TABLE data structure. Section 0004 sets the number of cam position array positions and cam interpolation curve mode, and section 0005 takes addresses for cam

position arrays. The initialization of the cam data structure variable CamTable is completed.

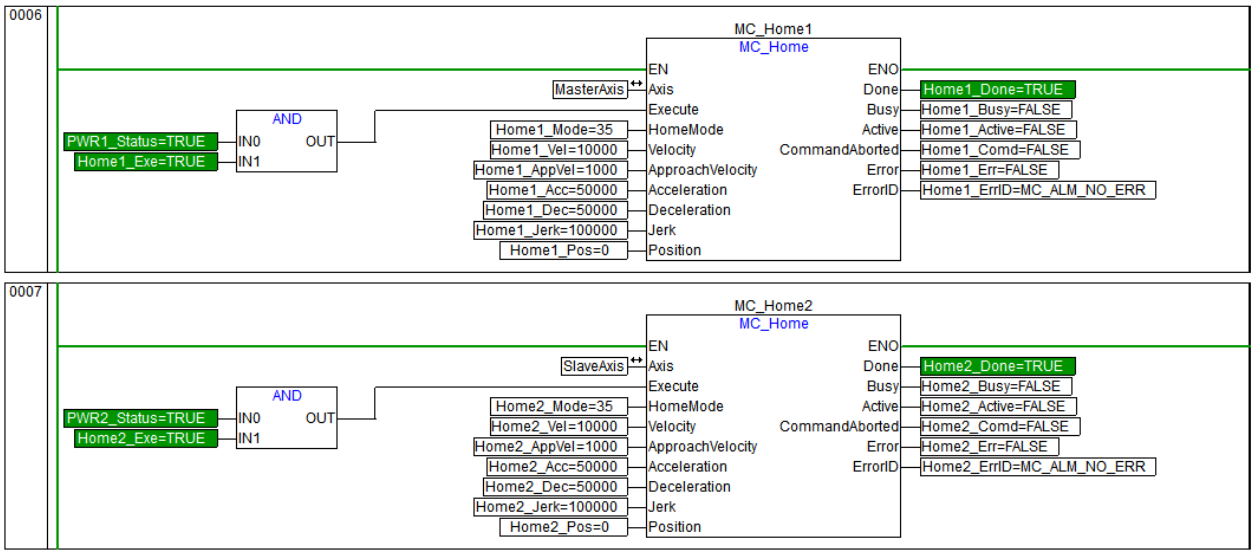
ArrTablePos [0][PosNum] ={0,60,120,180,240,300,360};

ArrTablePos [1][PosNum] ={0,1000, 2000,3000,2500,2000,1500};

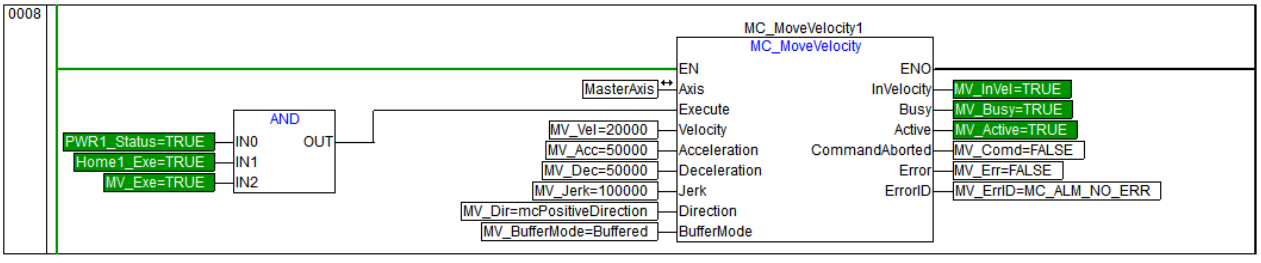
PosNum=7;



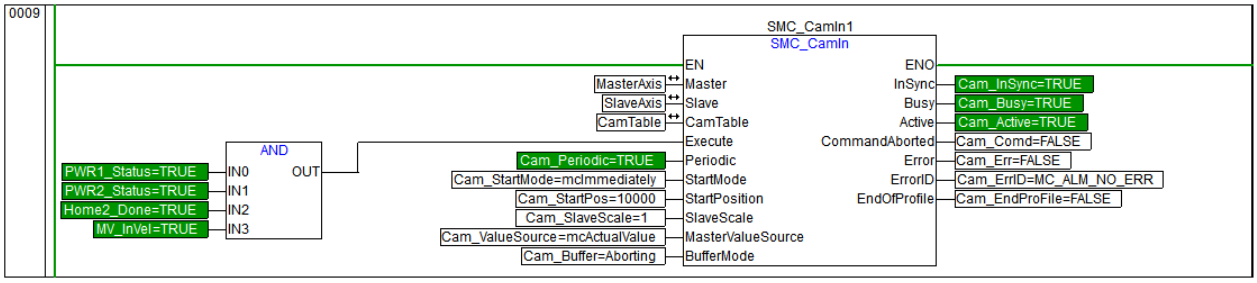
(5) The master and slave axes are enabled successfully. If the origin is not determined, execute the origin return command of the master and slave axes to determine the origin. See Sections 0006 and 0007 for origin return. For the usage of MC_Home command, see its introduction section.



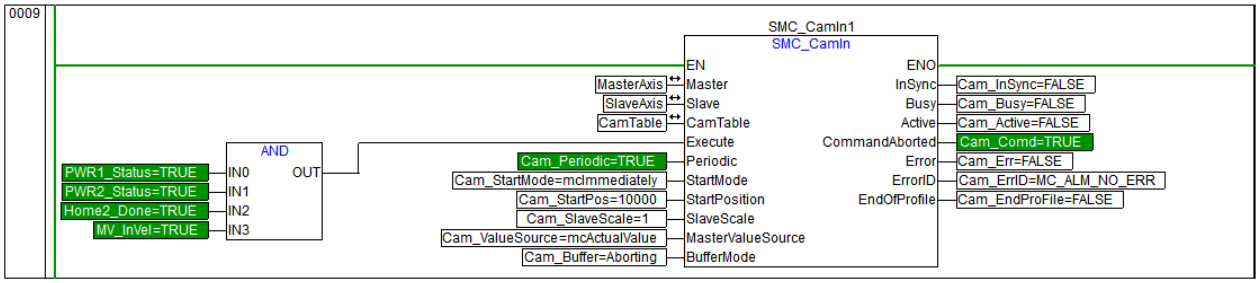
(6) See Section 0008. After the homing is completed, execute MC_MoveVelocity command. The master starts to move.

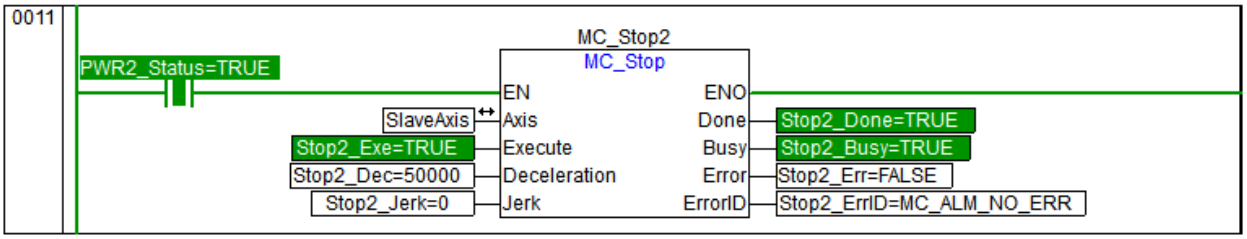


(7) See section 0009, when the MV_InVel pin of master speed command MC_MoveVelocity becomes TRUE, start the SMC_CamIn1 command of the slave axis. The cam command is cycle start immediately and the Cam_InSync pin is TRUE.

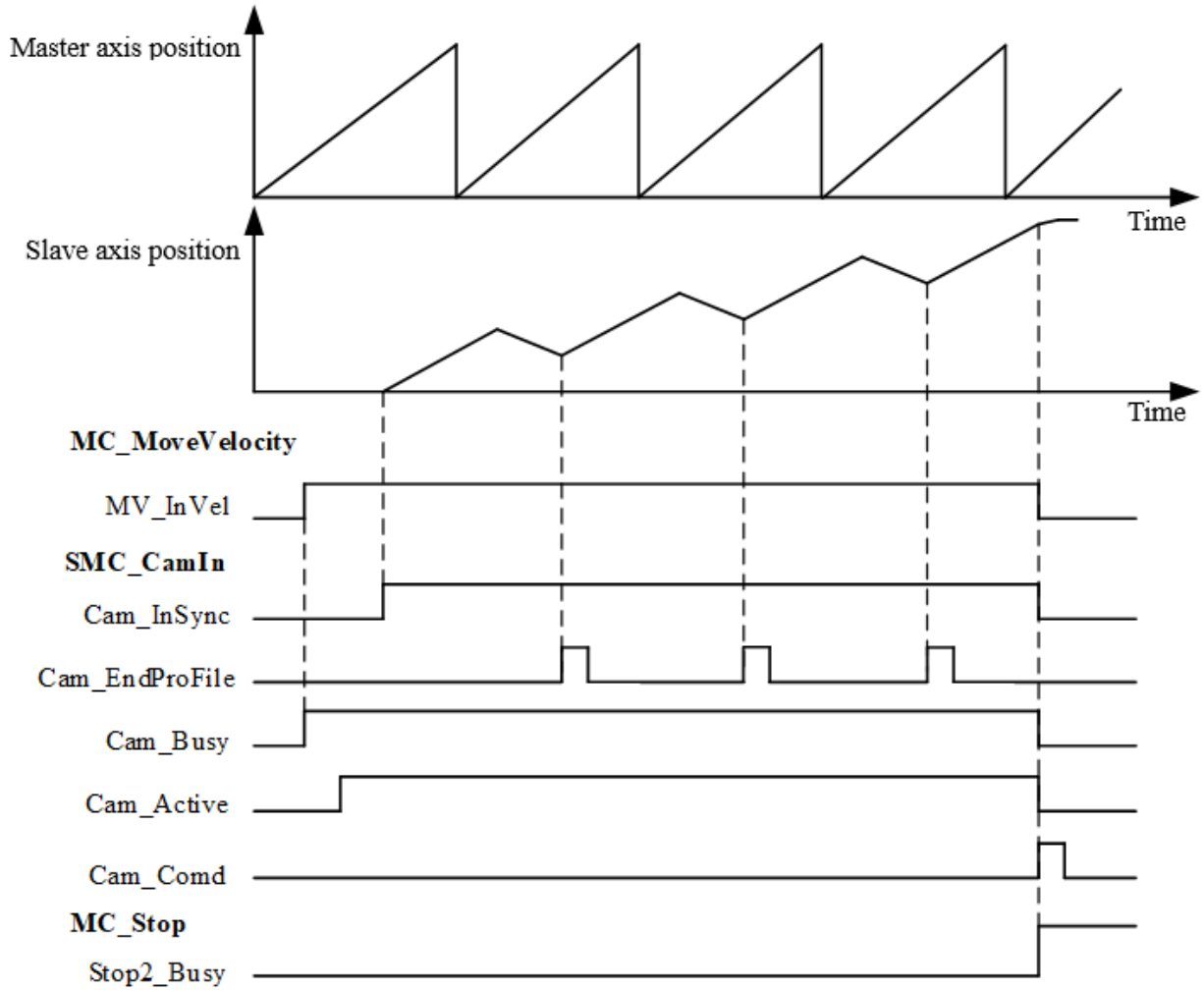


(8) For cycle cams, it is possible to override and stop the cam motion command from the axis using MC_Stop. In the following section 0011, the rising edge triggers the MC_Stop2 command, and the cam command of the slave axis is interrupted. The Cam_Comd pin of the cam becomes TRUE in section 0009.





The timing diagram of some variables in this example is as follows:



■ ST language program

Command is set to single mode, absolute position start and cam start fixed at 10000. The master position crosses the cam start position 10000 in the negative direction, and the cam action starts.

(1) Primary variables

Variable Name	Variable Type	Initial Value	Notes
MasterAxis	AXIS_REF	-	Axis variables of the master
SlaveAxis	AXIS_REF	-	Axis variable of slave axis
Power1_Status	BOOL	FALSE	Variable for successful master enable. This variable becomes TRUE when the axis is enabled successfully.
Power2_Status	BOOL	FALSE	Variable that enabled the slave axis successfully. This variable becomes TRUE when the axis is enabled successfully.
MV_Exe	BOOL	FALSE	Rising edge trigger variable of axis speed control command.
MV_InVel	BOOL	FALSE	Identification variable for the axis speed control command to reach the target speed. This variable becomes TRUE and the speed commanded reaches the target speed.
CamTable	SMC_CAM_TABLE	-	Cam Table Variables
ArrTablePos	ARRAY OF LREAL	-	An array of cam positions.
PointerToPos	POINTER TO ARRAY	-	A pointer variable to the array of cam positions, array position

	OF LREAL		Pos[2][PosNum]. PosNum is the number of positions.
Cam_InSync	BOOL	FALSE	Synchronization variable of cam command, which becomes TRUE when the cam command is synchronized from an axis.
Cam_EndProFile	BOOL	FALSE	Variable commanded by the cam to complete the cam cycle, which becomes TRUE when the cam commanded cam cycle is completed.
Switch1	BOOL	FALSE	Axis ready state variable. After axis initialization, Switch to TRUE and the axis is ready.
InitFlag	BOOL	FALSE	If this variable is TRUE, the initialization parameter setting is completed.

(2) Axis initialization

Set the AxisID of MasterAxis to 0 and that of SlaveAxis to 1. Call the SMC_AxisInit command to complete the initialization configuration of master and slave axes. The type of both master and slave axes is 50: EtherCAT_Position, and call the SMC_AxisInit command to complete the initialization configuration of master and slave axes.

(3) Axis enabling

After the initialization of the master and slave axes is completed, and the axis ready status Switch1 is TRUE, the high level triggers the axis enable command to complete the axis enable.

```
IF  Switch1 THEN
   MC_Power1(
   Enable := PWR1_Enable,
   Axis := MasterAxis,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
Error=> PWR1_Err,
ErrorID=> PWR1_ErrID);
MC_Power2(
   Enable := PWR2_Enable,
   Axis := SlaveAxis,
   Status=> PWR2_Status,
   Busy=> PWR2_Busy,
   Error=> PWR2_Err,
   ErrorID=> PWR2_ErrID);
END_IF
```

(4) Initialize SMC_CAM_TABLE data structure

Set the number of cam position array positions and the mode of cam interpolation curve, get addresses for the cam position array, and complete the initialization of cam data structure variable CamTable.

```
IF FALSE=InitFlag THEN
CamTable.PointsNum := PointsNum;(*Number of position array*)
ArrTablePos[0,0]:=0; (*initialization position cam array*)
ArrTablePos[0,1]:=10000; (*initialization position cam array*)
…
…
```

```
ArrTablePos[0, PointsNum-1]:=60000; (*initialization position cam array*)
ArrTablePos[1,0]:=0; (*initialization position cam array*)
ArrTablePos[1,1]:=5000; (*initialization position cam array*)
…
…
ArrTablePos[1, PointsNum-1]:=10000; (*initialization position cam array *)
CamTable.PointsNum := CurveType;(*cam curve interpolation method*)
CamTable.PointerToPos := ADR(ArrTablePos); (*cam position array take address*)
InitFlag:=TRUE;
END_IF
```

(5) Execute the origin return command of master and slave axis to determine the homing. For the MC_Home command, see its introduction chapter.

```
IF  PWR1_Status THEN
   MC_Home1(
   Execute := Home1_Exe,
   HomeMode := Home1_Mode,
   Velocity := Home1_Vel,
   ApproachVelocity := Home1_AppVel,
   Acceleration := Home1_Acc,
   Deceleration := Home1_Dec,
   Jerk := Home1_Jerk,
   Position := Home1_Pos,
   Axis := MasterAxis,
   Done=> Home1_Done,
   Busy=> Home1_Busy,
   Active=> Home1_Active,
   CommandAborted=> Home1_Comd,
   Error=> Home1_Err,
   ErrorID=> Home1_ErrID);
   MC_Home2(
   Execute := Home2_Exe,
   HomeMode := Home2_Mode,
   Velocity := Home2_Vel,
   ApproachVelocity := Home2_AppVel,
   Acceleration := Home2_Acc,
   Deceleration := Home2_Dec,
   Jerk := Home2_Jerk,
   Position := Home2_Pos,
   Axis := SlaveAxis,
   Done=> Home2_Done,
   Busy=> Home2_Busy,
   Active=> Home2_Active,
   CommandAborted=> Home2_Comd,
   Error=> Home2_Err,
   ErrorID=> Home2_ErrID);
```

```
END_IF
```

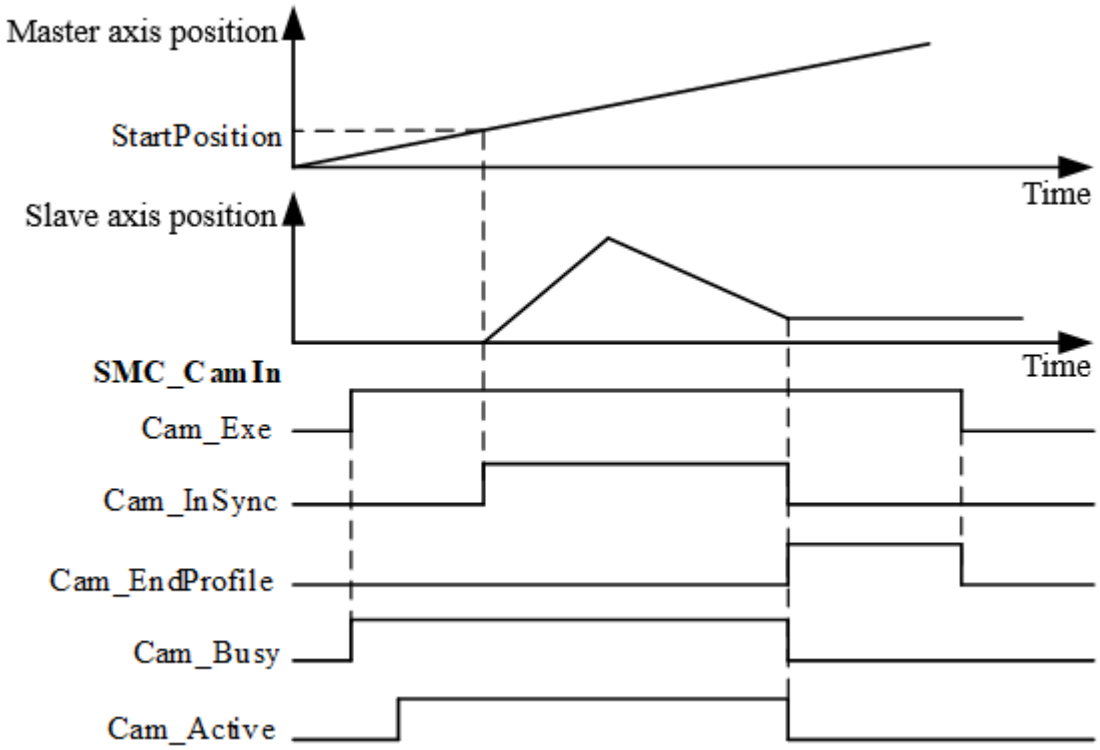
(6) Send cam command from the cam, start at a single fixed position, and when the cam changes from the negative direction to the positive direction of the fixed position, the cam acts.

```
IF  PWR1_Status AND PWR2_Status AND Home1_Done THEN
   SMC_CamIn1(
   Execute := Cam_Exe,
   Periodic := Cam_Periodic,
   StartMode := Cam_StartMode,
   StartPosition := Cam_StartPos,
   SlaveScale := Cam_SlaveScale,
   MasterValueSource := Cam_ValueSource,
   BufferMode := Cam_Buffer,
   Master := MasterAxis,
   Slave := SlaveAxis,
   CamTable := CamTable,
   InSync=> Cam_InSync,
   Busy=> Cam_Busy,
   Active=> Cam_Act,
   CommandAborted=> Cam_Comd,
   Error=> Cam_Err,
   ErrorID=> Cam_ErrID,
   EndOfProfile=> Cam_EndPro);
END_IF
```

(7) Delivery of master motion command

```
IF  PWR1_Status AND Home1_Done THEN
   MC_MoveVelocity1(
   Execute := MV_Exe,
   Velocity := MV_Vel,
   Acceleration := MV_Acc,
   Deceleration := MV_Dcc,
   Jerk := MV_Jerk,
   Direction := MV_Dir,
   BufferMode := MV_Buffer,
   Axis := Axis0,
   InVelocity=> MV_InVel,
   Busy=> MV_Busy,
   Active=> MV_Active,
   CommandAborted=> MV_Comm,
   Error=> MV_Err,
   ErrorID=> MV_Error);
END_IF
```

The figure below shows single mode, which immediately starts the cam's position profile and pin timing.



6.1.4 Precautions

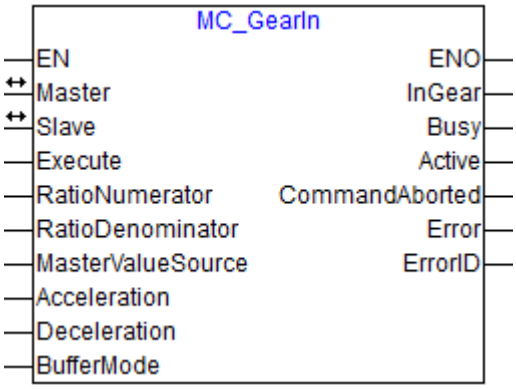
- The elements in the master position array must increase monotonically (the same number is not allowed), and the data can be positive or negative.
- During the execution of this command, when the user rewrites the position of the master in the cam position array to non-incremental data, an error exception will occur when running this command.
- The number of cam position array positions is greater than or equal to 3. The number of cam position array positions must strictly correspond to the length of the cam position array.
- When the cam is started in absolute position, the master passes through the absolute position in positive direction to trigger the cam command.
- When the cam is started by DI trigger, the input parameter StartPosition must be positive and its value must be less than the capacity of Run Controller I area.
- The reference starting point when the cam is started, the dynamic coordinate point used for calculation during operation and the reference coordinate point when withdrawing are all based on the Mpos value of the master camshaft. When the master is under position closed-loop control, since the master Mpos comes from an external detection device, in order to avoid accidental withdrawal of the cam caused by small disturbance of Mpos at the boundary of the effective stroke range of the cam, the withdrawal condition will also take into account the master Dpos value.
- For a single cam, when Mpos comes from an external detection device, in order to avoid abnormal withdrawal of the cam caused by small disturbance, it shall be ensured that there is a certain margin

between the normal working region of the cam and the boundary position of the cam (ensure that the disturbance of Mpos does not exceed the boundary position of the cam).

- When the slave axis exceeds the soft limit during cam action, the cam will be interrupted and the slave axis will stop running immediately.
- For a single cam, the slave axis follows the master motion within the stroke boundary. After the master moves across the left boundary in negative direction or crosses the right boundary in positive direction, the cam command is revoked.

6.2 MC_GearIn (Electronic Gear)

This command implements the electronic gear function.



6.2.1 Parameter

Parameter Type	Title	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Master	Master axisname	No	-	-	AXIS_REF
	Slave	Slave axis Name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	RatioNumerator	Gear ratio numerator	YES	1	Positive/negative/0	DINT
	RatioDenominator	Gear ratio denominator	YES	1	Positive value	UDINT
	MasterValueSource	Master position source 0: Synchronize with the master command position 1: Synchronize with master feedback position	YES	0	0: mcSetValue 1: mcActualValue	MC_SOURCE
	Acceleration	Acceleration	No	-	Positive value	LREAL
	Deceleration	Deceleration	No	-	Positive value	LREAL
	BufferMode	Buffer mode 0: Interrupt 1: Buffer	YES	0	0: Aborting 1: Buffered	MC_BUFFER_MODE
OUTPUT	InGear	Synchronizing	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL

	Active	In control, TRUE when the command is executed	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

6.2.2 Functions

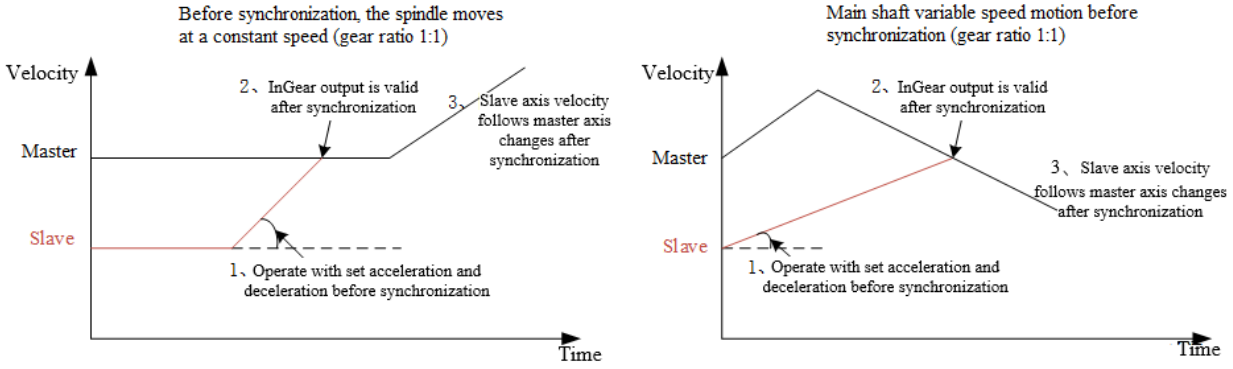
This command sets the gear ratio between the master and slave for electronic gear action. The slave axis is designated as the motion axis, and electronic gear action can be carried out according to the set parameters such as gear ratio numerator, gear ratio denominator, acceleration and deceleration. The detailed introduction of this command is as follows:

(1) The rising edge triggering of this command is effective, and the setting object is the slave shaft. The electronic gear acts according to the set position and feedback position of the master.

(2) After the gear action is started, the target speed of the slave shaft is the speed of the main shaft multiplied by the gear ratio, and the moving distance of the slave shaft is the moving distance of the main shaft multiplied by the gear ratio.

(3) When the gear ratio is positive, the slave axis moves in the same direction as the main shaft; when the gear ratio is negative, the slave axis moves in the opposite direction along the main shaft.

(4) Before reaching the target position, it is called "chasing", and after that, it is called "synchronizing". Before synchronization, the slave axis moves according to the set acceleration and deceleration. After synchronization, the slave changes completely with the master, as shown in the schematic diagram below.

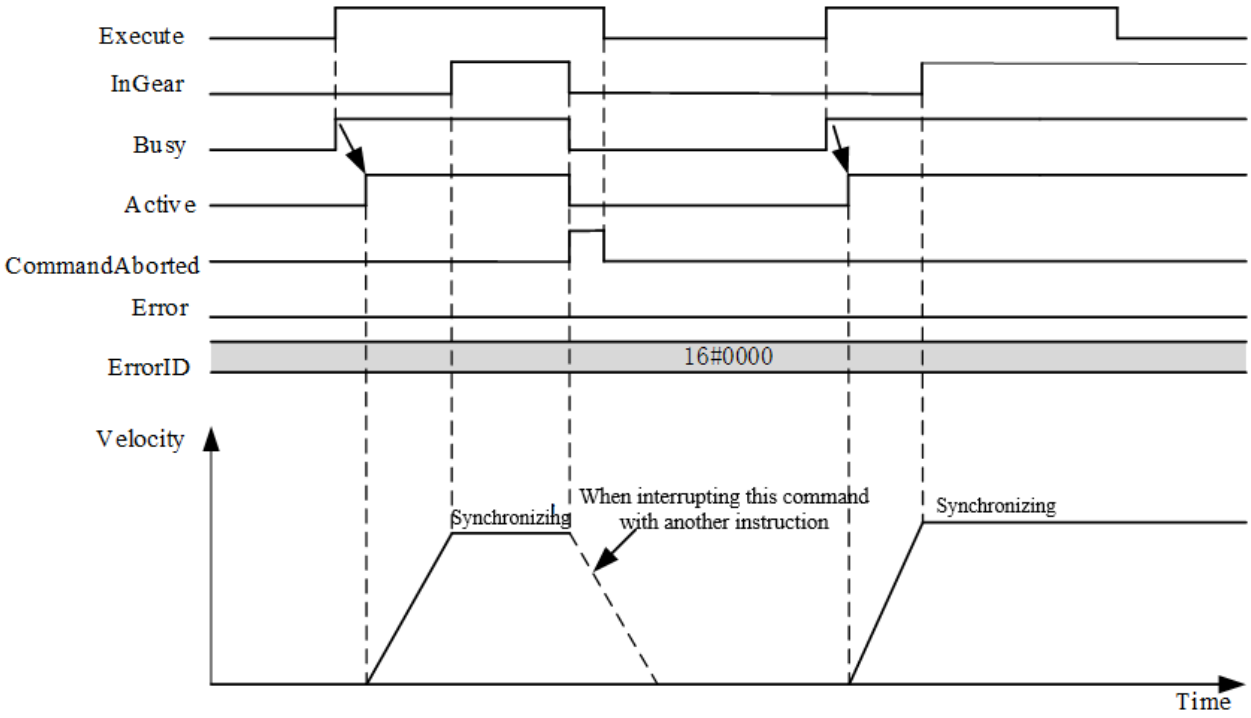


(5) The actions of this command during restart and multiple start are specified by BufferMode (buffer mode selection). Interruption and buffering can be selected for multiple start of this command. When interrupt is selected, the slave will recalculate the following action according to the gear ratio and master speed, and determine whether the InGear flag bit is set according to the calculation result.

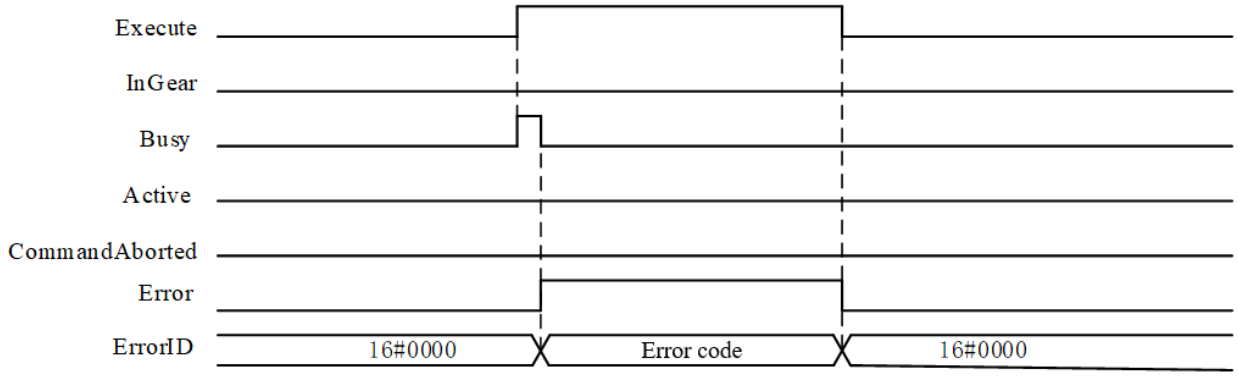
(6) To stop the electronic gear action halfway, use other commands to interrupt or use the MC_Stop command.

■ Timing diagram

(1) Normal timing diagram



(2) Error timing diagram



6.2.3 Examples

Set up MC_GearIn example program to run electronic gear action commands.

■ Variable

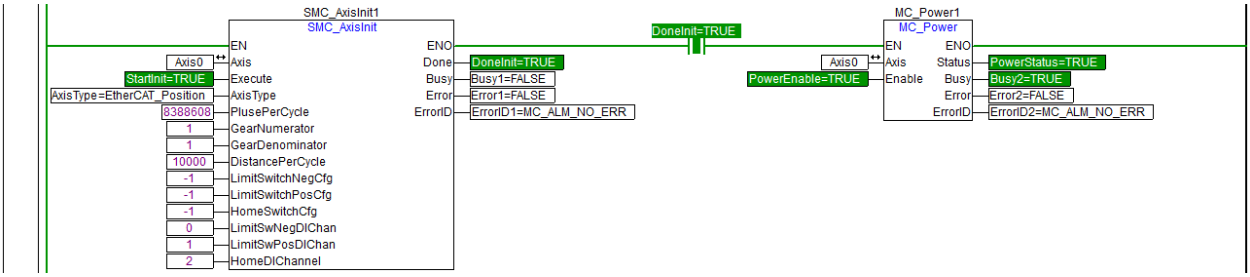
Name	Data Type	Initial Value	Notes
Axis0	AXIS_REF		Axis variable for axis 0

StartInit	BOOL	FALSE	Change to TRUE at the start of initialization
DoneInit	BOOL	FALSE	Change to TRUE upon completion of initialization
PowerEnable	BOOL	FALSE	Change to TRUE at the beginning of enabling
PowerStatus	BOOL	FALSE	Change to TRUE when enabled
StartGear	BOOL	FALSE	TRUE on start command
RatioNumerator	DINT	1	Gear Ratio Numerator
RatioDenominator	UDINT	1	Gear Ratio Denominator
MasterValueSource	MC_SOURCE	0	Master position source selection
InGear	BOOL	FALSE	Change to TRUE during command synchronization
BusyGear	BOOL	FALSE	Change to TRUE when the command is delivered successfully
ActiveGear	BOOL	FALSE	The command changes to TRUE during operation.
ComGear	BOOL	FALSE	Change to TRUE when the command is interrupted
ErrorGear	BOOL	FALSE	Change to TRUE in case of command error

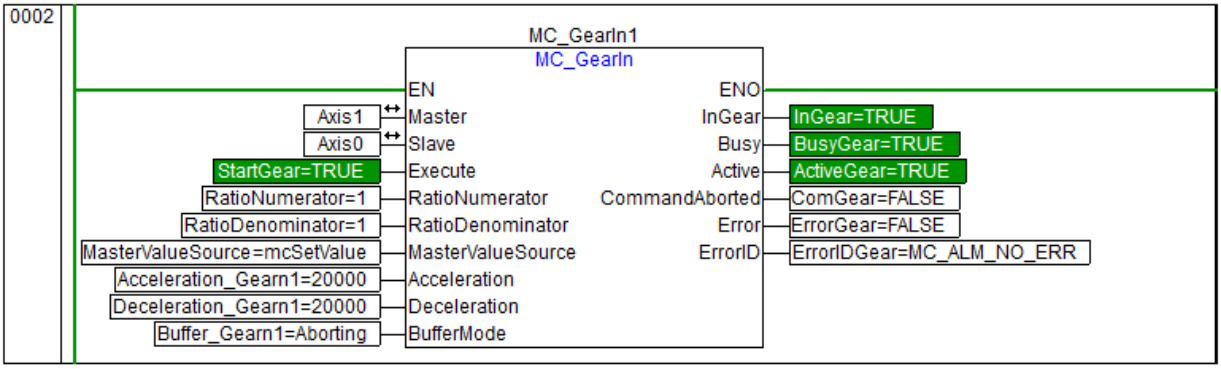
■ LD program implementation

Use the ladder diagram to build a program, determine that the master is axis 1 and the slave axis is axis 0, send an MC_MoveVelocity command to the master, input the master speed as 10000, input the gear ratio as 1:1, select the position source synchronized with the master command position, and trigger the rising edge to control the slave axis for electronic gear action. The example program is as follows:

(1) Initialize the slave axis, and initialize the type and various parameters of the axis. After the axis is initialized, enable the axis. When it enters the ON state of servo enabling, the axis can realize motion control.

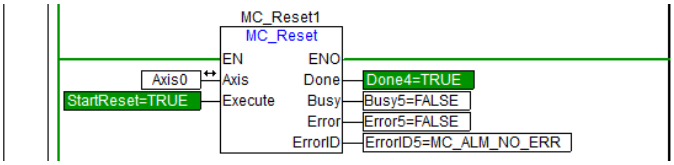


(2) Initiate the master to start speed motion, then input relevant motion parameters and gear ratio of the slave. The rising edge triggers the control electronic gear to move.



(3) When an error occurs to the axis, the MC_Reset function block can be called for reset

processing.



■ ST language program

(1) Initialize the axis

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

(2) Enable the axis

```
IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

(3) Send the electronic gear command to carry out electronic gear action

```
MC_GearIn1(
Execute := StartGear,
RatioNumerator := RatioNumerator,
RatioDenominator := RatioDenominator,
MasterValueSource := MasterValueSource,
Acceleration := Acceleration_Gearn1,
```

```
Deceleration := Deceleration_Gearn1,
BufferMode := Buffer_Gearn1,
Master := Axis1,
Slave := Axis0,
InGear=>InGear,
Busy=>BusyGear,
Active=>ActiveGear,
CommandAborted=>ComGear,
Error=>ErrorGear,
ErrorID=>ErrorIDGear);
Reset in case of error
MC_Reset1(
Execute := StartReset,
Axis := Axis0,
Done=>Done4,
Busy=>Busy5,
Error=>Error5,
ErrorID=>ErrorID5);
```

6.2.4 Precautions

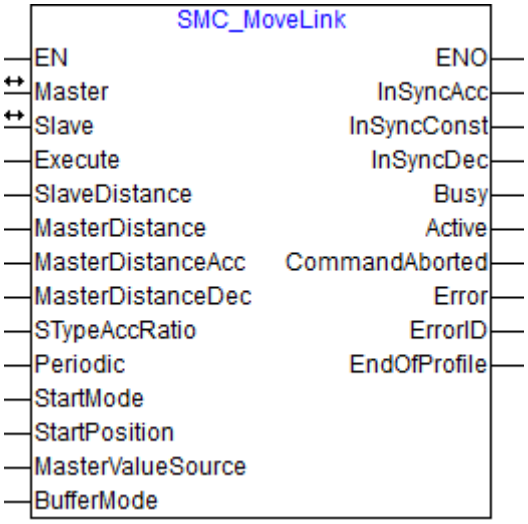
- The action of the electronic gear takes effect on the slave axis. The acceleration and deceleration in the electronic gear take effect from the acceleration or deceleration of the slave axis during catching up. If it is synchronized, the slave axis moves completely with the main shaft, and the acceleration and deceleration do not take effect.
- The main shaft and slave axis cannot be set to the same shaft, and the denominator of gear ratio cannot be 0.

6.2.5 Reference

- For the use of masters, see other single-axis motion chapters.
- In case of any error, please refer to the appendix Function Block Error Code Description.

6.3 SMC_MoveLink (Flying Shear)

This command implements the pursuit process action.



6.3.1 Parameter

Parameter Type	Title	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	Master	Master axis name	No	-	-	AXIS_REF
	Slave	Slave axis Name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	SlaveDistance	Slave axis motion distance	No	-	Positive/negative/0	LREAL
	MasterDistance	Master movement distance	No	-	Positive value	LREAL
	MasterDistanceAcc	Distance traveled by master in slave axis acceleration phase	No	-	Positive/0	LREAL
	MasterDistanceDec	Distance moved by the master during deceleration stage of slave axis	No	-	Positive/0	LREAL
	STypeAccRatio	S-shaped acceleration and deceleration percentage, in 1%	YES	0	0 ~ 100	LREAL
	Periodic	FALSE: Single mode TRUE: Circulation mode	YES	FALSE	TRUE/FALSE	BOOL
	StartMode	Startup mode 0: Absolute position start (triggered when passing through the absolute position point in forward direction) 10: Start immediately 11: DI trigger startup	YES	0	0:mcAbsolute 10:mcImmediately 11:mcDITrigger	MC_CAMIN_STARTMODE
	StartPosition	When the absolute position is started, it is the starting position;	YES	0	Absolute position start: positive/negative/0;	LREAL

		When DI trigger is started, it is the bit offset mapped by DI point in Area I. For example, the bit offset of %IX100.1 in area I is 100×8+1.			DI trigger start: 0 ~ (field I byte capacity * 8)-1	
	MasterValueSource	Master Position Source 1: Synchronize with Master feedback position	YES	1	1: mcActualValue	MC_SOURCE
	BufferMode	Buffer mode 0: Interrupt 1: Buffer	YES	1	0: Aborting 1: Buffered	MC_BUFFER_MODE
OUTPUT	InSyncAcc	The slave axis is in the acceleration synchronization section	YES	FALSE	TRUE/FALSE	BOOL
	InSyncConst	The slave axis is in the constant-speed synchronization section	YES	FALSE	TRUE/FALSE	BOOL
	InSyncDec	The slave is in the deceleration synchronous section	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Active	In control, TRUE when the command is executed	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Execution Aborting	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR
	EndOfProfile	End of period	YES	FALSE	TRUE/FALSE	BOOL

6.3.2 Functions

This command realizes the chasing shearing process.

■ Command function description

(1) This command realizes the synchronous shearing action of master and slave axis, which is a kind of electronic cam.

(2) Rising edge triggering is valid. On the rising edge of the Execute input, when the input parameters are legal and there is no abnormality in the axis, the value of the input parameter will be latched. During the execution of the command, modifying the latched parameter will be invalid until another rising edge triggers this command.

(3) To stop the axis in motion via this command, use the MC_Stop command.

(4) Flying shearing stroke. The stroke length of the master in the chasing shearing motion is the MasterDistance. If the starting point of the flying shearing motion master is Mpos0, then the effective stroke range for a single flying shearing is [Mpos0, Mpos0+ MasterDistance].

- Single mode. After MoveLink is started: For immediate start and DI triggered start, the command left boundary is at the position of cam start; for absolute position start, the command left boundary is at the set absolute position; command right boundary = left boundary + master stroke length. The order is cancelled when the master moves negatively across the left boundary or positively across or reaches the right boundary.
- For single shearing motion, when the master is within the effective stroke range, the master can perform unidirectional and reciprocating motion. When the master reciprocates within the effective stroke range, the slave shaft will also perform reciprocating linkage action.
- Cycle mode. When the master position crosses a master stroke length, it will enter the next flying shearing stroke. At this time, the slave axis will take the slave axis position at the end of the previous flying shearing stroke as the starting point, and make incremental displacement calculation on this basis. Repeat the linkage action of the previous flying shearing stroke in turn. To cancel the cyclic flying shear motion, execute the MC_Stop command. This will stop and cancel the flying shear operation.

The linkage motion track of the master and slave for motion track is consistent with that of the cam. For details of the motion track diagram, please refer to the linkage motion track diagram of the master and slave in the Chapter of Cam Command Description (see the introduction part of cam stroke in SMC_CamIn).

(5) The track distance of slave axjs is 0.

When the motion distance of slave axis is 0, after MoveLink is started: InSyncAcc outputs TRUE in the stage of slave axis acceleration; InSyncDec outputs TRUE in the stage of slave axis deceleration; InSyncConst outputs TRUE in other stages.

(6) The follow-up command slave axis supports the following axis types:

- Virtual axis (0): virtual axis
- EtherCAT_Position(50): EtherCAT bus connecting shaft, position control
- EtherCAT_Speed(51): EtherCAT bus connecting shaft, speed control
- EtherCAT_Torque(52): EtherCAT bus connecting shaft, torque control

■ Initiation mode of flying shearing

This command can be started in any state of StandStill, position control, speed control and synchronous control.

When the flying shearing motion is started, the starting point position of the master will be set according to different start modes. According to the set value of start mode StartMode, the flying shearing start mode is divided into 0: fixed position start, 10: immediate start and 11: DI start.

■ Fixed position start

The command start mode is 0: start at fixed position. After the start command, wait for the flying shearing command master Mpos to cross the cam starting position StartPosition from the negative direction. The command is triggered and the flying shearing motion starts.

■ Immediate start

The command start mode is 10: immediate start. After the start command, the flying shearing is started immediately. At this time, StartPosition has no meaning.

- DI trigger start

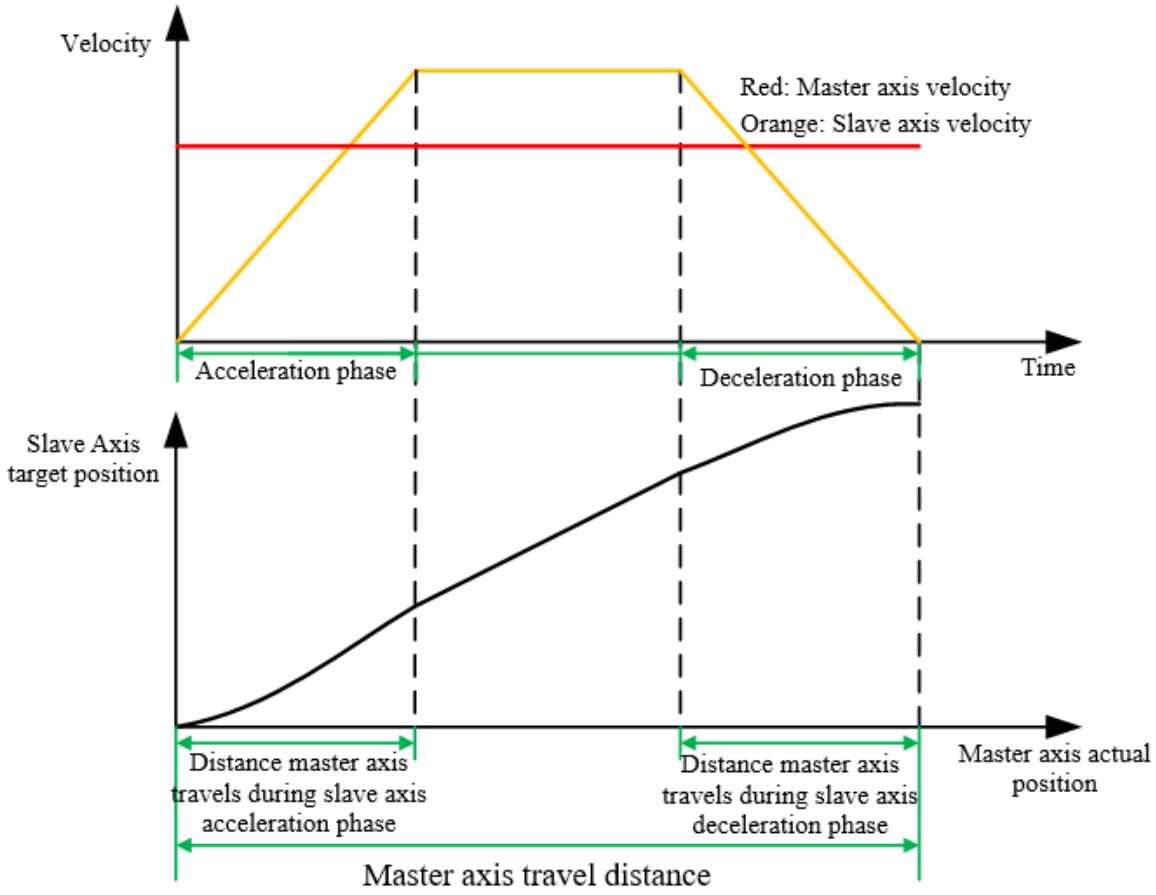
The command start mode is 11: DI trigger start. After issuing the flying shearing command, the command is triggered when the DI signal is set to 1, and the starting position is the Mpos position of the master when the program detects that DI is 1. StartPosition is the bit offset mapped by DI point in Area I. For example, the bit offset of %IX100.1 in area I is 100×8+1, i.e. StartPosition=801. When DI triggers start, the value of StartPosition shall be less than the capacity of Run Controller I area.

- Planning method of acceleration and deceleration stage

Acceleration and deceleration phase planning method. The command supports acceleration and deceleration in trapezoidal or S-shaped curves during the acceleration and deceleration stage. According to the set distance of acceleration and deceleration stage, acceleration and deceleration mode, trapezoidal or S-shaped acceleration and deceleration can be selected.

- Trapezoidal acceleration and deceleration (STypeAccRatio=0)

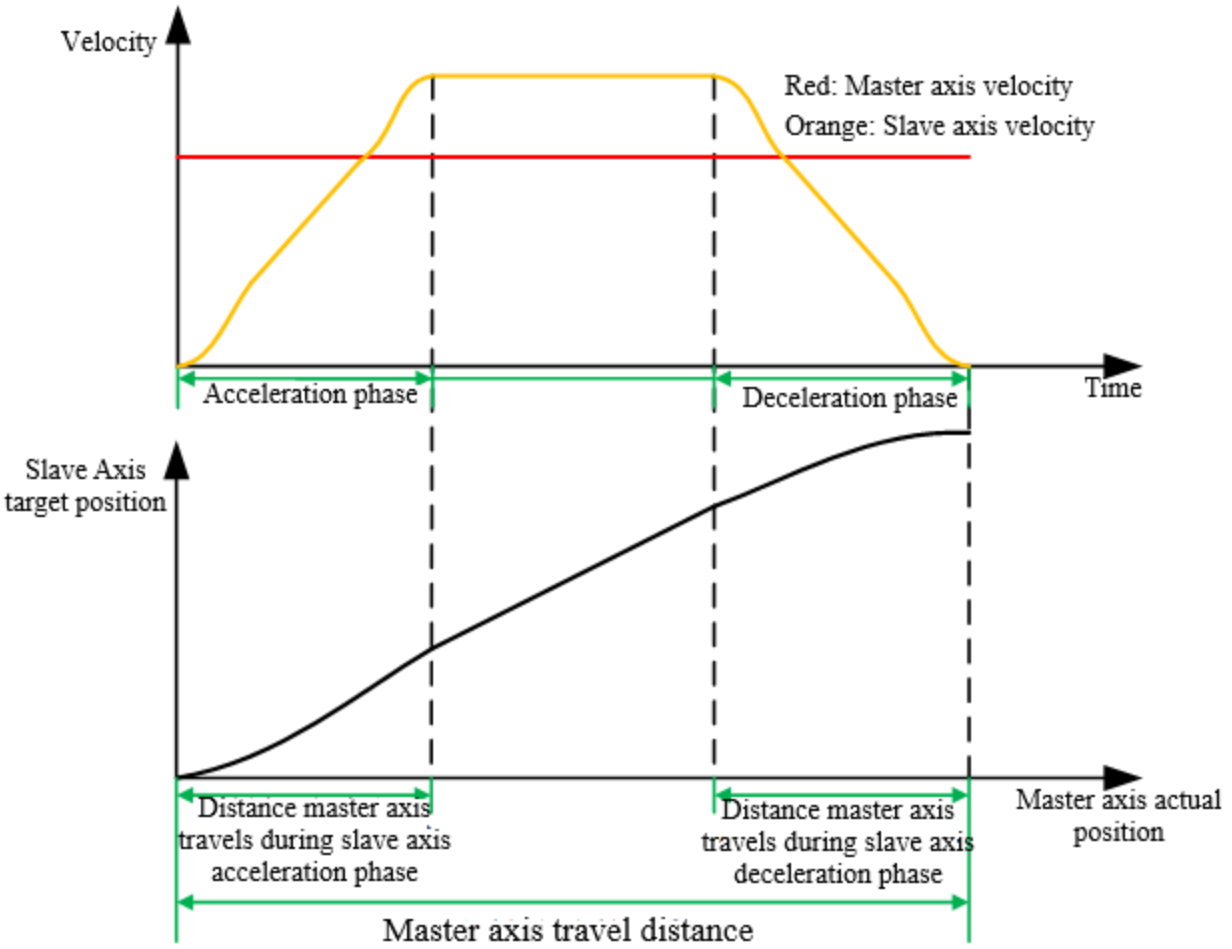
When STypeAccRatio = 0, the speed of the slave axis is planned according to the trapezoidal curve acceleration and deceleration. The target position of the slave axis in this cycle is calculated in real time based on the trapezoidal model and the actual position of the master. During trapezoidal acceleration and deceleration, taking the uniform motion of the master as an example, the speed and displacement speed curves of the master and slave are shown below:



■ S-curve acceleration and deceleration (STypeAccRatio in the range of (0,100])

When STypeAccRatio is in the range of (0,100], the speed of the slave axis is planned according to the S-shaped curve acceleration and deceleration, and the target position of the slave axis in this cycle is calculated in real time based on the S-shaped model and the actual position of the master. STypeAccRatio represents the proportion of S-shaped acceleration and deceleration in the whole acceleration/deceleration process. Taking the acceleration stage as an example, the ratio of the time spent in the acceleration stage to that in the whole acceleration stage is STypeAccRatio/100/2, and the ratio of the time spent in the constant acceleration stage to that in the whole acceleration stage is 1-STypeAccRatio/100. The ratio of the time spent in the deceleration phase to the total acceleration phase is STypeAccRatio/100/2.

Taking STypeAccRatio = 50/100 and the master moving at a constant speed as an example, the schematic diagram of speed and displacement curves of master and slave axis is as follows:



When STypeAccRatio is set to 100, there will be no constant acceleration/constant deceleration stage in the acceleration/deceleration stage. The ratio of acceleration and deceleration in the acceleration stage is 0.5 respectively, which is similar to that in the deceleration stage.

■ Periodic (cycle mode)

According to the boundary range of master position, the flying shearing motion can be divided into single and periodic motion. The flying shearing motion is determined by the input parameter Periodic. When Periodic = False, it is single motion ; when Periodic = TRUE, it is periodic motion.

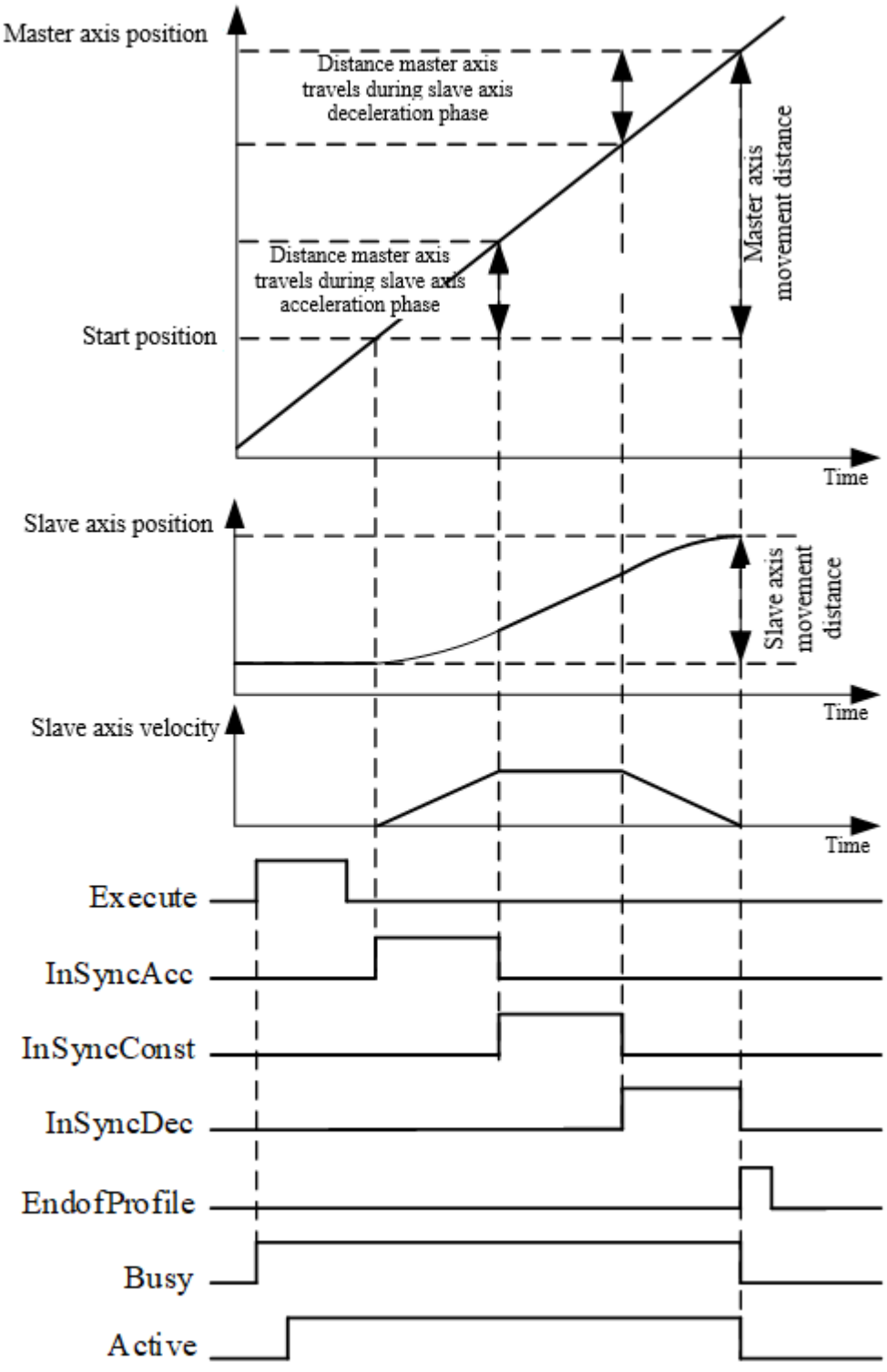
(1) Single flying shearing

After the flying shearing motion is started, if the master position is within the effective stroke range, the flying shearing motion can maintain normal linkage operation. When the master exceeds the effective stroke range, the command will be cancelled automatically.

After the flying shearing is started and the command triggering conditions are met, the command InSyncAcc pin is set to TRUE, and the slave axis moves synchronously with the master; When the slave enters the constant-speed synchronization section after completing the acceleration synchronization section, the InSyncConst pin is set to TRUE and the InSyncAcc pin is set to FALSE. When the slave shaft enters the deceleration synchronization section, the InSyncDec pin is set to TRUE and the InSyncConst and InSyncAcc pins are set to FALSE.

After the single command is completed, the EndofProfile pin is set to TRUE and kept for at least one cycle, and the Busy and Active pins become FALSE. After encountering the falling edge of Execute, all output pins are restored to Default Values.

The following figure shows the pin timing at fixed position startup in single mode:

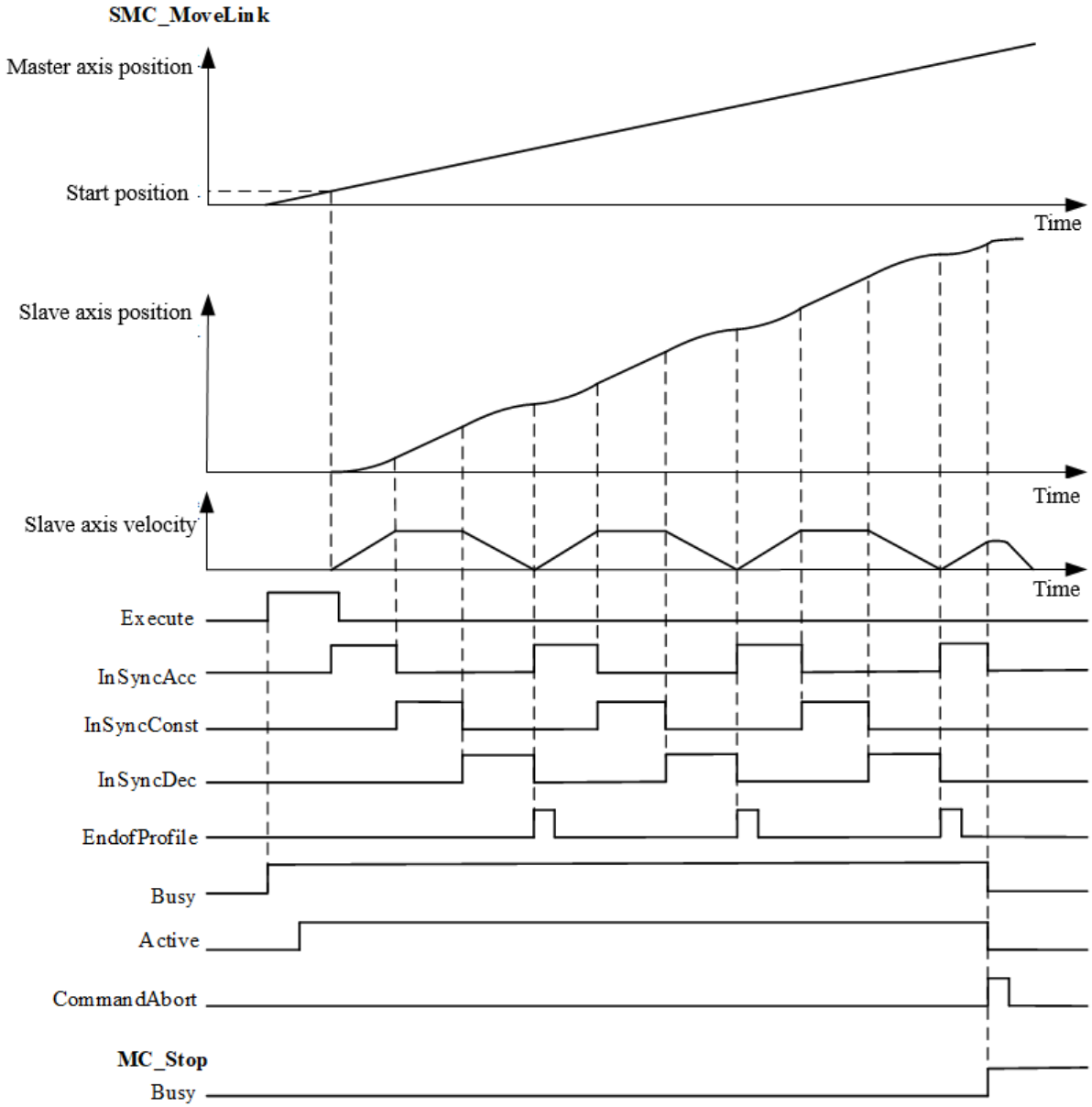


(2) Cyclic flying shearing

For cyclic trailing motion, there is no boundary range limit for master position. If the corresponding stop cancellation command is not executed after startup, the trailing function will be always valid.

The following figure shows the timing of cut command pins in cycle mode and fixed position startup mode. The slave axis performs corresponding linkage actions with the master. When the master finishes a flying shearing stroke, it enters the next stroke. Incremental calculation is performed from the end point of the previous flying shearing stroke as the starting point to execute periodic linkage actions. At the end of each clipping pass, the output pin EndofProfile is set high for one cycle.

When the MC_Stop command is used to cancel the abort command of cyclic flying shearing, the flying shearing command output pin CommandAborted is set as TRUE and other output pins are reset.



■ BufferMode (buffer mode selection)

Specify the connection mode between the previous axis action and this action.

This command only supports 0: Aborting and 1: Buffered.

When this command is sent, if BufferMode is 0: Aborting, the currently executing command will be immediately aborted and switched to this command. If BufferMode is 1:Buffered, the bufferd command will be automatically started from the cycle when the currently executing command ends normally.

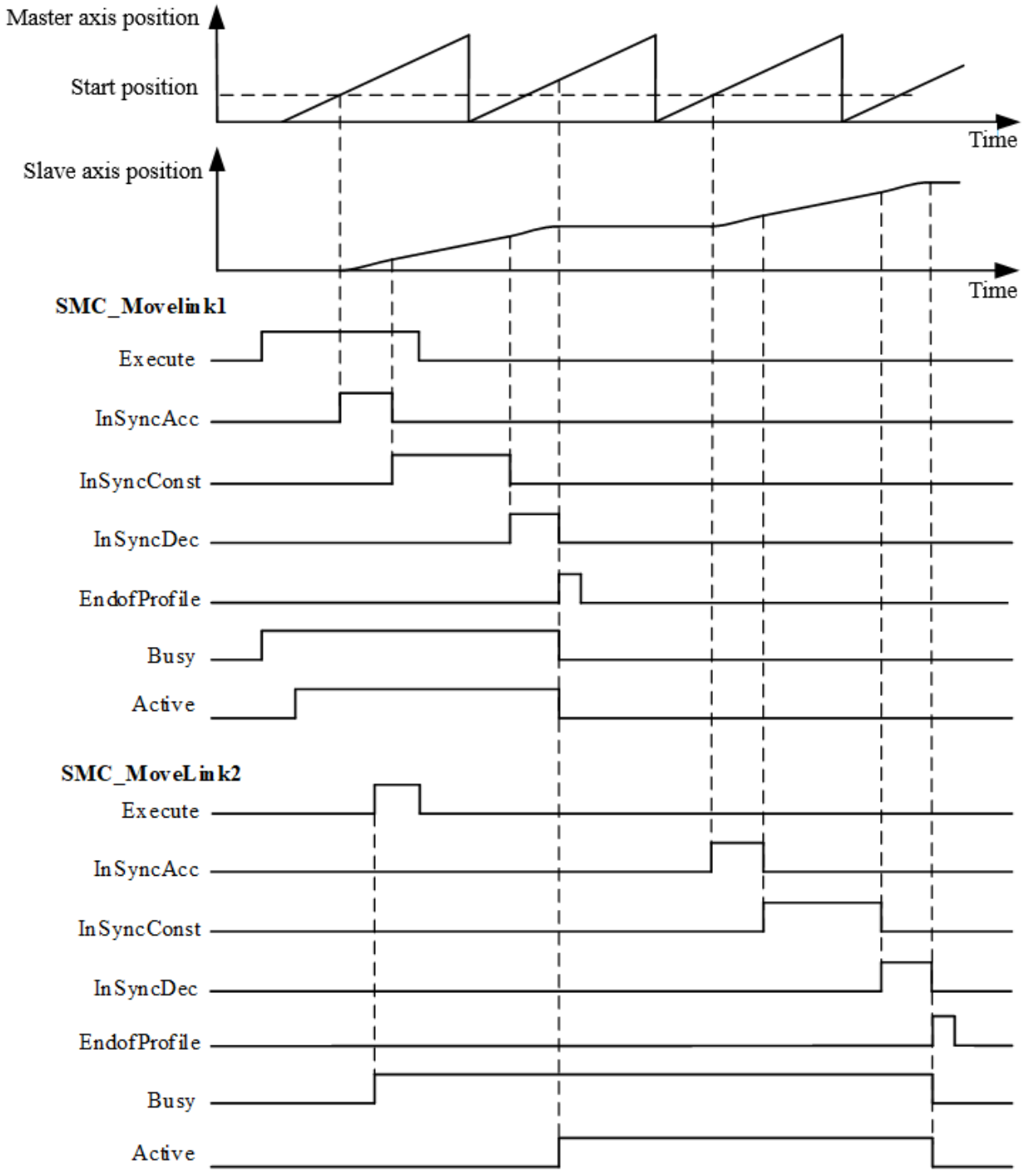
- Handling of repeated triggering of function block
  - For the same flying shearing motion command function block in operation, if it is triggered repeatedly for execution, there are:

    (a) If the BufferMode of the command is 0:Aborting, abort the current execution command;

    (b) If the BufferMode of the command is 1:Buffered, the command will be put into the motion buffer. The state monitored by the subsequent function block will be the running state of the next command, and the running state of the previous command will be out of supervision.

  - When the function block command is waiting for repeated triggering, during the execution of the previous flying shearing command, the next command starts to be executed after the end of the previous command.

    As shown in the following example, during the execution of SMC_MoveLink1 (single mode, fixed position start), a specified delay is applied before re-triggering the SMC_MoveLink2 command (single mode, fixed position start). Once EndOfProfile of SMC_MoveLink1 becomes TRUE and Active of SMC_MoveLink2 becomes TRUE, the cam starts its motion when the master axis reaches the StartPosition of the SMC_MoveLink2 command. At this point, InSyncAcc (synchronized acceleration) turns TRUE, initiating the flying shear action.



■ EndofProfile (End of periodic flying chasing)

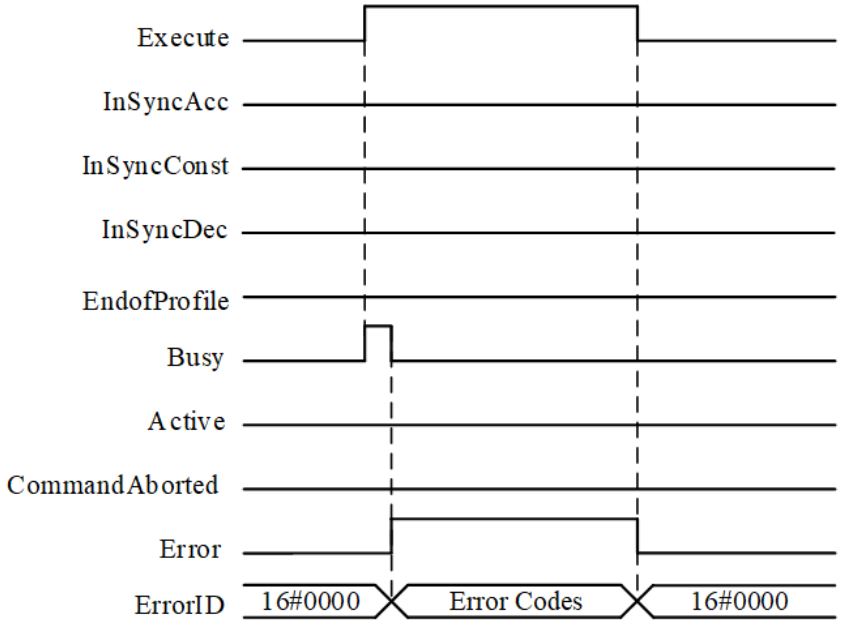
When the chase command is in single mode, the command cycle is completed, and the command output pin EndofProfile is set to TRUE and held until the input pin Execute becomes FALSE.

When the cut command is in periodic mode, the first cut stroke is completed, and the command output pin EndofProfile is set to TRUE for only one cycle. The command enters the next cut stroke, and EndofProfile becomes FALSE until the stroke is completed. The EndofProfile pin is set to TRUE for one cycle, and EndofProfile continues to change and update with the periodic cut action.

- Error

When starting the flying shearing motion command, if an error occurs, the Error pin is set to True, and the error code ErrorID reports an error value. Refer to the appendix Description of Error Codes to obtain the cause of the error. When the command input pin Execute goes low, the output pin returns to its Default Value.

SMC_MoveLink abnormal situation sequence diagram



6.3.3 Examples

In this example, MasterAxis is defined as the master axis, SlaveAxis as the slave axis, and the variable type is AXIS_REF.

The MasterAxis executes the MC_MoveVelocity command, and the SlaveAxis executes the MC_MoveLink command. Use the MC_Stop command to stop revoking the posthumous motion command.

- LD program implementation

This example sets up a periodic mode, immediately starting the flying shearing command.

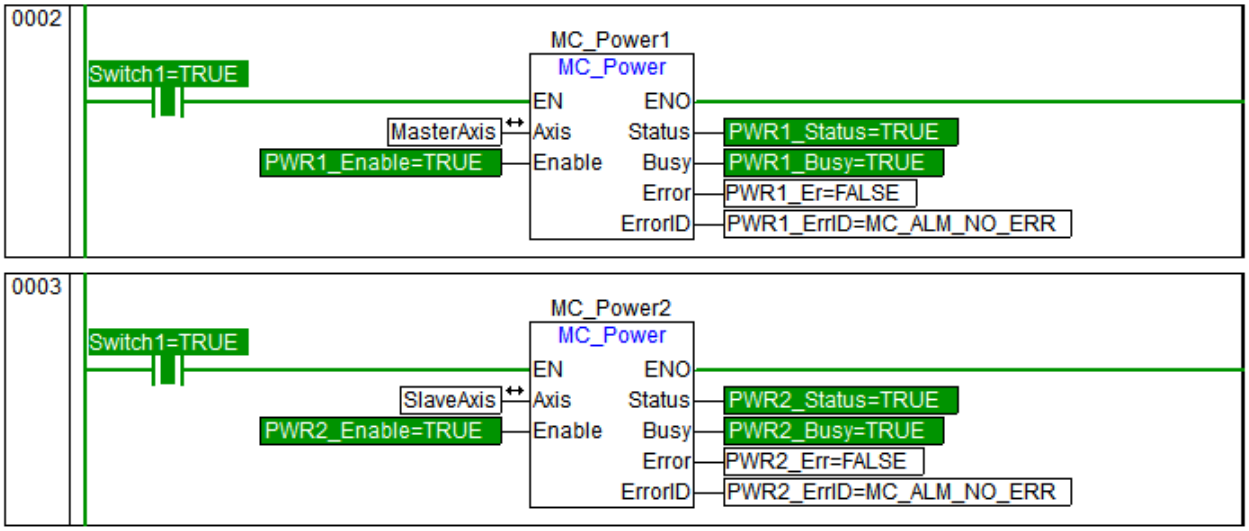
Example Program Primary Variables

Variable Name	Variable Type	Initial Value	Notes
MasterAxis	AXIS_REF	-	Axis variables of the master axis
SlaveAxis	AXIS_REF	-	Axis variables of the slave axis
Power1_Status	BOOL	FALSE	Variable for successful master enable. This variable becomes TRUE when the axis is enabled successfully.
Power2_Status	BOOL	FALSE	Variable for successful slave enable. This variable becomes TRUE when the axis is enabled successfully.

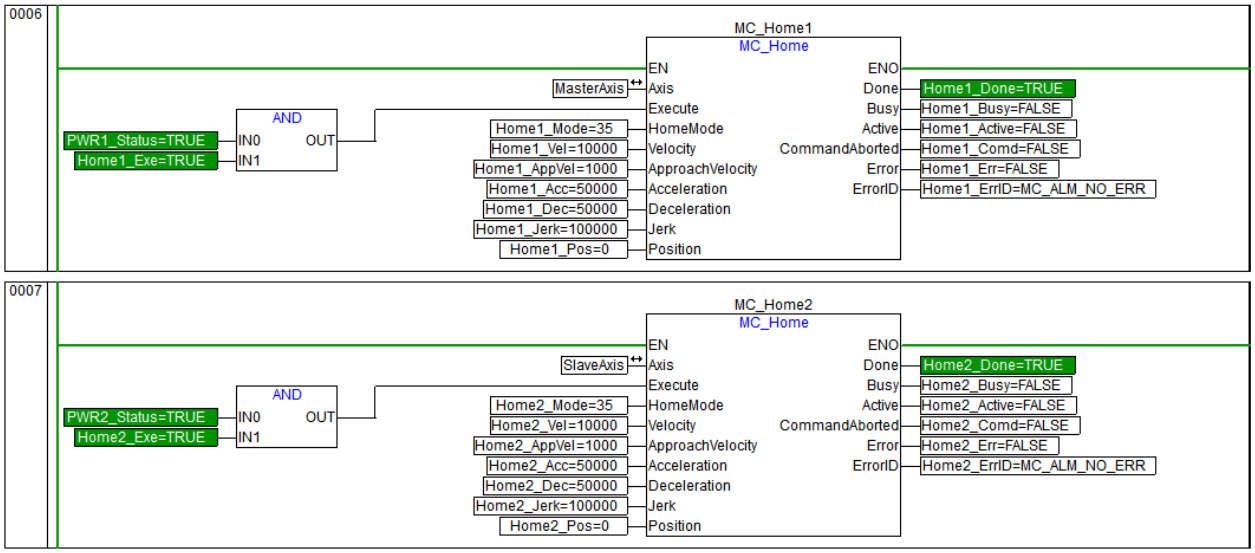
Switch1	BOOL	FALSE	The state variable of axis readiness. Switch to TRUE, indicating that the axis is ready.
MV_Exe	BOOL	FALSE	The variable triggered by the rising edge of the speed control command for an axis.
MV_InVel	BOOL	FALSE	Identification variable for the axis speed control command to reach the target speed. This variable becomes TRUE and the speed commanded reaches the target speed.
ML_SlaveDis	LREAL	-	Slave axis motion distance for delivering a flying shearing command from an axis.
ML_MasterDis	LREAL	-	Master axis motion distance for delivering a flying shearing command from an axis.
ML_DisAcc	LREAL	-	The master axis motion distance during the slave axis acceleration phase when issuing a flying shear command to the slave axis.
ML_DisDec	LREAL	-	The master axis motion distance during the slave axis deceleration phase when issuing a flying shear command to the slave axis.

(1) Axis initialization. Set the AxisID of MasterAxis to 0, the AxisID of SlaveAxis to 1, and the axis type of both master and slave axes to 50. Call the SMC_AxisInit command to complete the initialization configuration of master and slave axes. For the usage of SMC_AxisInit command, please refer to its introduction.

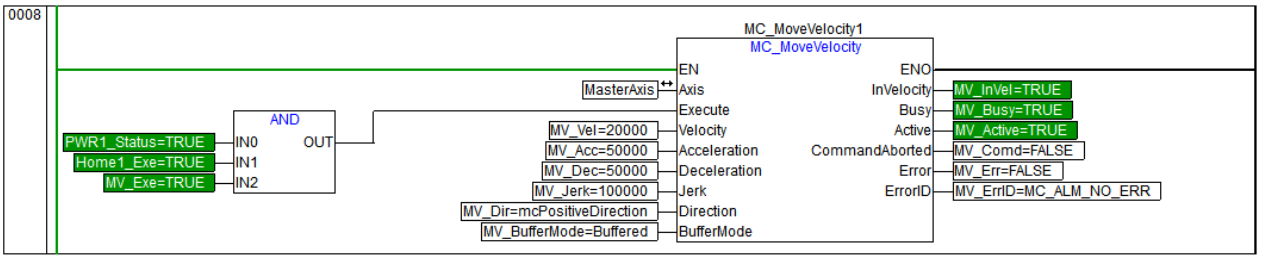
(2) Axis enable. In section 0002, after the master and slave axis are initialized and Switch1 is TRUE, execute the axis enable command to complete the axis enable.



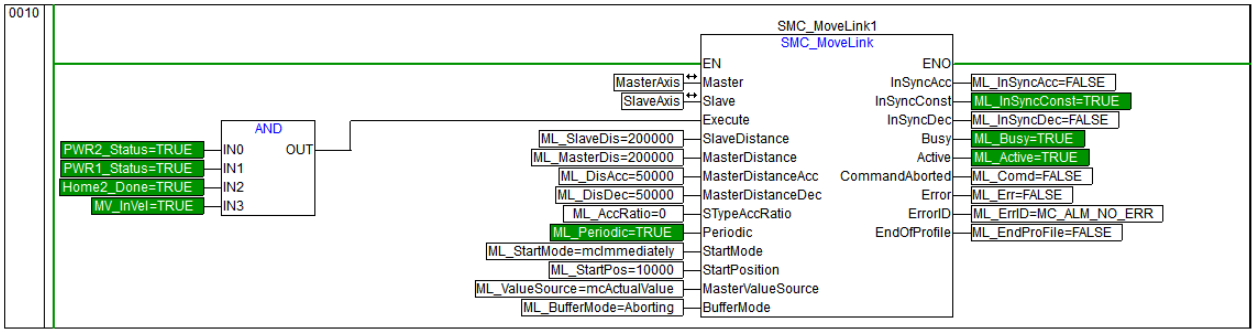
(3) The master and slave axis are enabled successfully. If the home position is not determined, execute the home command of the master and slave axes to determine the origin. See Sections 0006 and 0007 for homing details. For the usage of the MC_Home command, see its introduction section.



(4) See Section 0008. After the homing is completed, execute MC_MoveVelocity command. The master starts to move.

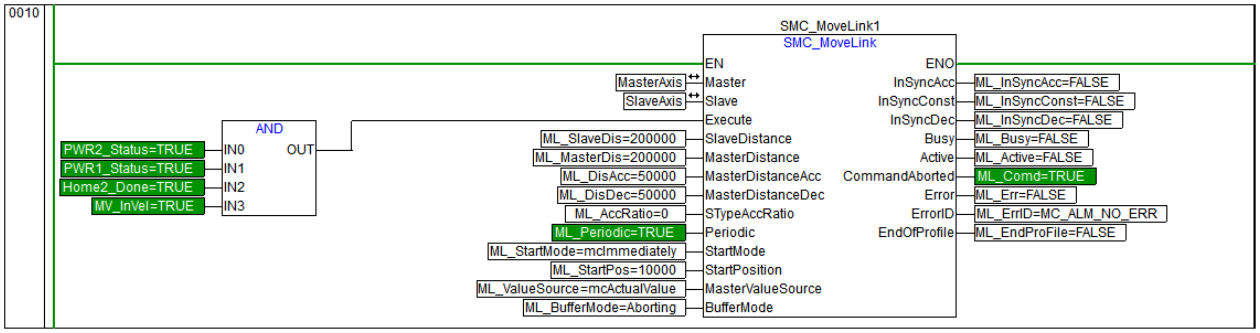


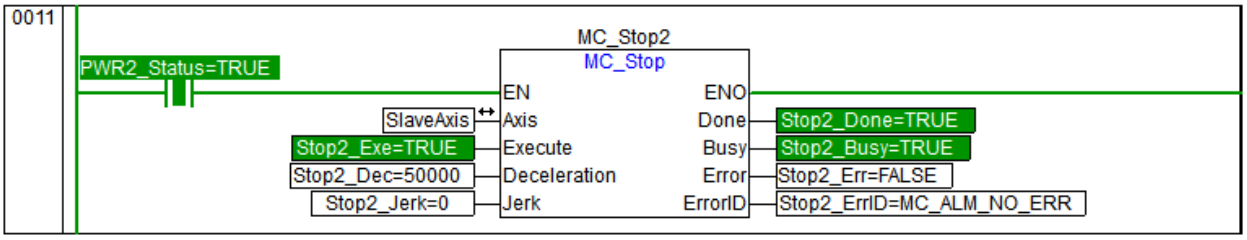
(5) See Section 0009. When the MV_InVel pin of master speed command MC_MoveVelocity becomes TRUE, start the SMC_MoveLink1 command of the slave axis. The command is to start the cycle immediately. The ML_InSyncAcc pin is TRUE, and the slave axis enters the acceleration synchronization section of flying shearing motion (ML_InSyncAcc=TRUE). The slave axis starts moving following the master, and then enters the uniform synchronization section of flying shearing motion (ML_InSyncConst =TRUE) and deceleration synchronization section of flying shearing motion (ML_InSyncDec =TRUE) in turn.



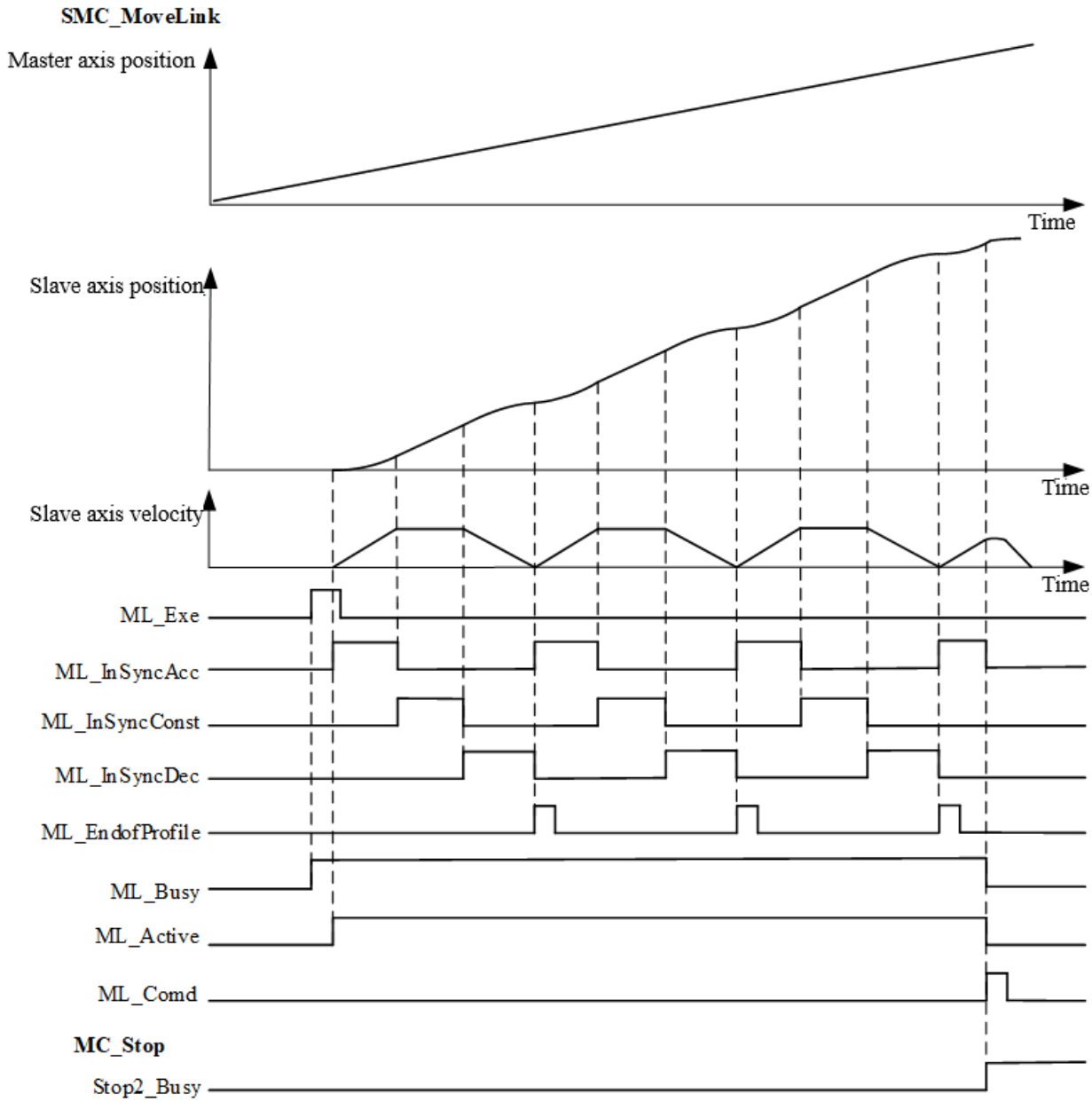
(6) For a cyclic flying shearing command, you can use MC_Stop to cancel and abort the flying shearing motion command of the slave axis. In the following section 0011, the MC_Stop2 is executed to abort the flying shearing command of the slave axis, and the ML_Comd of the trace

command becomes TRUE.





The timing diagram of this example is as follows:



■ ST language program.

Command is set to single mode, absolute position start and the fixed position of cam is at 100. The master position crosses the cam start position 100 in the negative direction, and the cam action begins.

Primary Variables

Variable Name	Variable Type	Initial Value	Notes
MasterAxis	AXIS_REF	-	Axis variables of the master axis
SlaveAxis	AXIS_REF	-	Axis variables of the slave axis
Power1_Status	BOOL	FALSE	Variable for successful master enable. This variable becomes TRUE when the axis is enabled successfully.
Power2_Status	BOOL	FALSE	Variable for successful slave enable. This variable becomes TRUE when the

			axis is enabled successfully.
MV_Exe	BOOL	FALSE	The variable triggered by the rising edge of the speed control command for an axis.
Switch1	BOOL	FALSE	The state variable of axis readiness. Switch to TRUE, indicating that the axis is ready.
InitFlag	BOOL	FALSE	If this variable is TRUE, the initialization parameter setting is completed.
ML_SlaveDis	LREAL	-	Slave axis motion distance for delivering a flying shearing command from an axis.
ML_MasterDis	LREAL	-	Master axis motion distance for delivering a flying shearing command from an axis.
ML_DisAcc	LREAL	-	The master axis motion distance during the slave axis acceleration phase when issuing a flying shear command to the slave axis.
ML_DisDec	LREAL	-	The master axis motion distance during the slave axis deceleration phase when issuing a flying shear command to the slave axis.

(1) Axis initialization

Set the AxisID of MasterAxis to 0 and that of SlaveAxis to 1. Call SMC_AxisInit to complete the initialization configuration of master and slave axes. The type of both master and slave axes is 50. Call SMC_AxisInit to complete the initialization configuration of master and slave axes. For the usage of SMC_AxisInit, please refer to its introduction.

(2) Initialization of flying shearing command parameters

```
IF FALSE = InitFlag THEN
  ML_SlaveDis:=200;
  ML_MasterDis:=200;
  ML_DisAcc:=10;
  ML_ DisDec:=10;
  ML_Periodic:=FALSE;
  ML_STartMode:= mcAbsolute;
  ML_StartPosition:=100;
  ML_Buffer:=Buffered;
  InitFlag:=TRUE;
END_IF
```

(3) Axis enable

After the master and slave axes are initialized and Switch1 is TRUE, execute the axis enable command to complete the axis enabling.

```
IF  Switch1 THEN
    MC_Power1(
    Enable := PWR1_Enable,
    Axis := MasterAxis,
    Status=> PWR1_Status,
    Busy=> PWR1_Busy,
Error=> PWR1_Err
ErrorID=> PWR1_ErrID)
MC_Power2(
    Enable := PWR2_Enable,
    Axis := SlaveAxis,
    Status=> PWR2_Status,
```

```
  Busy=> PWR2_Busy,
  Error=> PWR2_Err
  ErrorID=> PWR2_ErrID);
END_IF
```

(4) Execute the homing command of master and slave axes to determine the home. For the usage of MC_Home command, see its introduction chapter.

```
IF  PWR1_Status THEN
 MC_Home1(
 Execute := Home1_Exe,
 HomeMode := Home1_Mode,
 Velocity := Home1_Vel,
 ApproachVelocity := Home1_AppVel,
 Acceleration := Home1_Acc,
 Deceleration := Home1_Dec,
 Jerk := Home1_Jerk,
 Position := Home1_Pos,
 Axis := MasterAxis,
 Done=> Home1_Done,
 Busy=> Home1_Busy,
 Active=> Home1_Active,
 CommandAborted=> Home1_Comd,
 Error=> Home1_Err,
 ErrorID=> Home1_ErrID);
 MC_Home2(
 Execute := Home2_Exe,
 HomeMode := Home2_Mode,
 Velocity := Home2_Vel,
 ApproachVelocity := Home2_AppVel,
 Acceleration := Home2_Acc,
 Deceleration := Home2_Dec,
 Jerk := Home2_Jerk,
 Position := Home2_Pos,
 Axis := SlaveAxis,
 Done=> Home2_Done,
 Busy=> Home2_Busy,
 Active=> Home2_Active,
 CommandAborted=> Home2_Comd,
 Error=> Home2_Err,
 ErrorID=> Home2_ErrID);
END_IF
```

(5) The flying shearing command is sent from the axis, and a single fixed position starts. When the master moves from the negative direction of the fixed position to the positive direction, the flying shearing action starts.

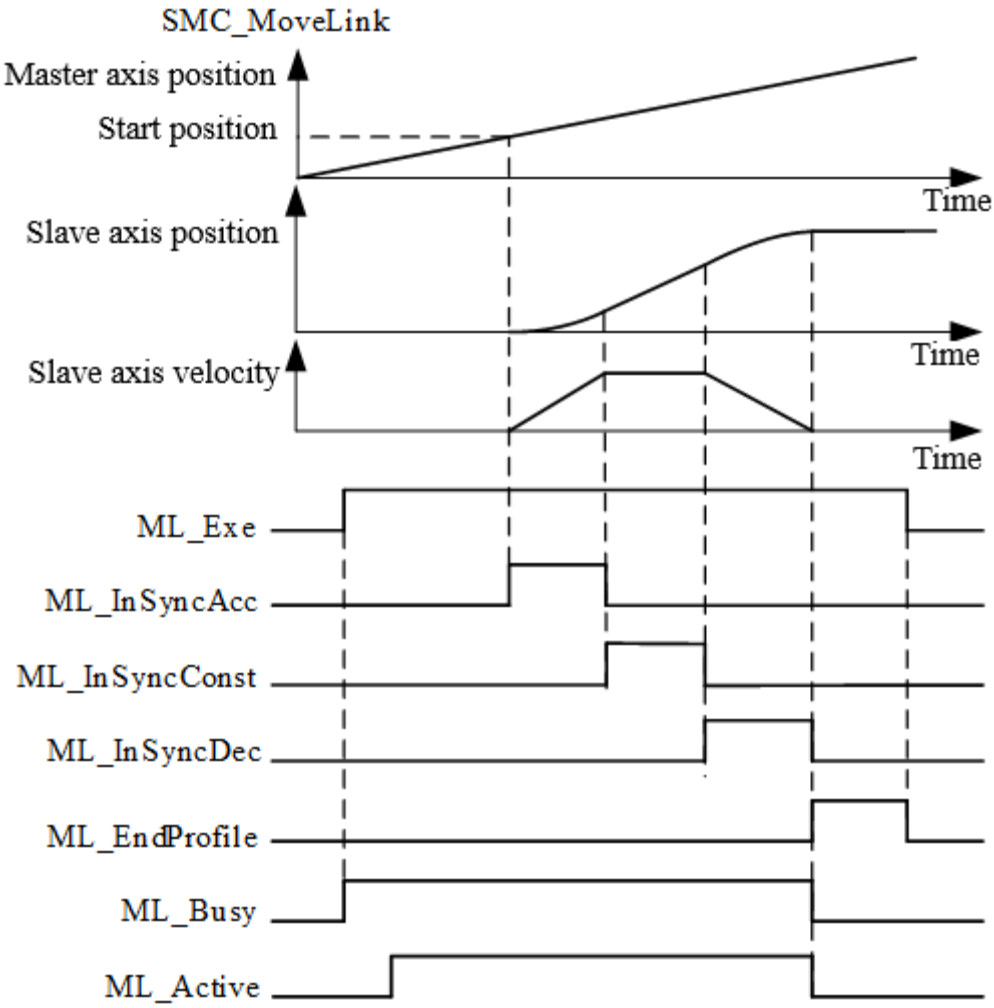
```
IF  PWR1_Status AND PWR2_Status AND Home1_Done THEN
 SMC_MoveLink1(
```

```
  Execute := ML_Exe,
  SlaveDistance := ML_SlaveDis,
  MasterDistance := ML_MasterDis,
  MasterDistanceAcc := ML_DisAcc,
  MasterDistanceDec := ML_ DisDec,
  STypeAccRatio := ML_AccRatio,
  Periodic := ML_Periodic,
  StartMode := ML_STartMode,
  StartPosition := ML_StartPosition,
  MasterValueSource := ML_ValueSource,
  BufferMode := ML_Buffer,
  Master := MasterAxis,
  Slave := SlaveAxis,
  InSyncAcc=> ML_InSyncAcc,
  InSyncConst=> ML_InSyncConst,
  InSyncDec=> ML_InSyncDec,
  Busy=> ML_Busy,
  Active=> ML_Act,
  CommandAborted=> ML_Comd,
  Error=> ML_Err,
  ErrorID=> ML_ErrID,
  EndOfProfile=> ML_EndPro);
END_IF
```

(6) Delivery of master motion command

```
IF  PWR1_Status AND Home1_Done THEN
  MC_MoveVelocity1(
  Execute := MV_Exe,
  Velocity := MV_Vel,
  Acceleration := MV_Acc,
  Deceleration := MV_Dcc,
  Jerk := MV_Jerk,
  Direction := MV_Dir,
  BufferMode := MV_Buffer,
  Axis := Axis0,
  InVelocity=> MV_InVel,
  Busy=> MV_Busy,
  Active=> MV_Active,
  CommandAborted=> MV_Comm,
  Error=> MV_Err,
  ErrorID=> MV_Error);
END_IF
```

The example master and slave axis position curve and command pin timing are shown in the following figure:



6.3.4 Precautions

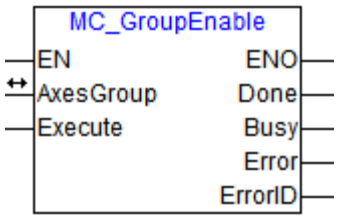
- The motion distance of the master needs to be positive.
- For a single trailing motion, the slave axis follows the master movement within the travel boundaries and is revoked after the master has crossed the left boundary in negative direction or the right boundary in positive direction.
- When flying shearing is started at absolute position, the master passes through the absolute position in forward direction to trigger cam command.
- When tracing is started by DI triggering, the input parameter StartPosition must be positive. The value of StartPosition should be less than the capacity of Operation Controller Zone I
- When SlaveDistance is 0, after MoveLink is started: InSyncAcc outputs TRUE in MasterDistanceAcc phase; InSyncDec outputs TRUE in MasterDistanceDec phase; InSyncConst outputs TRUE in other phases.
- When the master and slave axis are set to be on the same axis, the command reports an error.

- When the slave axis exceeds the soft limit during flying shearing, the command is interrupted and the slave axis aborts running immediately.
- When the counter mode of the master is rotation mode, please specify MasterDistance (master movement distance) as the value within 1 revolution of the ring counter of the master.

Chapter 7 Axis Group Motion Instruction

7.1 MC_GroupEnable (Enable Axes Group)

Axis group enable.



7.1.1 Parameter

■ Pin Parameters

Pin Parameter Description

Parameter Type	Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	AxesGroup	Axis group name	No	-	-	AXES_GROUP_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
OUTPUT	Done	Completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

■ Structural parameters of axis group

Currently, only two-axis interpolation is supported.

AXES_GROUP_REF data structure

Member	Meaning	Default Value	Scope	Data Type
AxisInterp_ID	Interpolation combined motion axis ID	255	0~63	USINT
AxisX_ID	Interpolated X-axis ID	255	0~63	USINT
AxisY_ID	Interpolated Y-axis ID	255	0~63	UDINT
ReservedPara1	Retained parameters	255	-	USINT
ReservedPara2	Retained parameters	255	-	USINT
ReservedPara3	Retained parameters	255	-	USINT
ReservedPara4	Retained parameters	255	-	USINT
ReservedPara5	Retained parameters	255	-	USINT
AxesGroupDisabled	Axis Group Disabled	TRUE	TRUE/FALSE	BOOL

- Users need to manually add axis group variables in AT project and enter legal combined axis ID, X-axis ID and Y-axis ID.

7.1.2 Functions

This command enables the axis group.

■ Command function description

(1) The MC_GroupEnable command changes the axis group specified by AxesGroup to Enable status.

(2) When defining an axis group, you need to customize three axes corresponding to the interpolation combined axis and two sub-axes in the axis group. The customized three axis IDs correspond to the three interpolated axis IDs in the axis group.

(3) The axis group variable AxesGroup type is AXES_GROUP_REF. After setting the axis group, call this command to enable the axis group. When AxesGroup.AxesGroupDisabled=FALSE and the axis group is enabled, the axis group can execute all multi-axis coordination commands.

(4) Execute the command of rising edge, and the axis parameters are checked to be normal. The MC_GroupEnable command sets the AxesGroupDisabled variable in the specified axis group to FALSE, and completes the axis group enable. AxesGroupDisabled variables can only be changed by MC_GroupEnable command and MC_GroupDisable command, and users are strictly prohibited from changing them manually.

(5) When the axis type of the combined axis is an unused axis, there is no need to call MC_Power to enable the axis of the combined axis. The axis group enable command automatically completes the enable of the combined axis. Therefore, only two sub-axes need to be enabled.

(6) If the combined axis is set as a virtual axis, call MC_Power to enable the axis, just like the sub-axis.

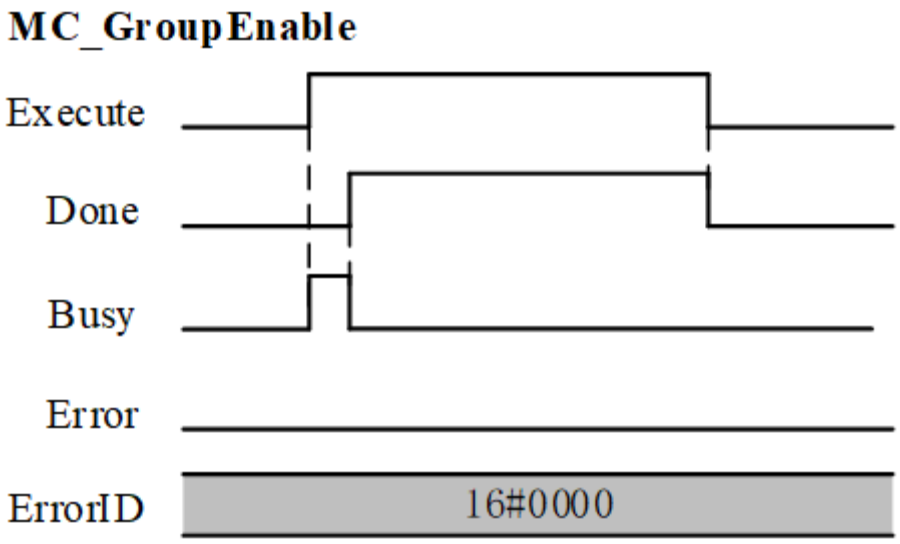
(7) The condition for invalidating the axis group is to execute the MC_GroupDisable (close the axis group function). After this command is successfully executed, the AxesGroupDisabled variable is set to TRUE and the axis group function is closed.

(8) Supported axis types. When this command is called, the axis types assigned to the two sub-axes in the axis group are EtherCat bus servo axis and Virtual: virtual axis. If other axis types are specified, an error will occur. The type of combined axis is virtual axis.

■ Timing diagram

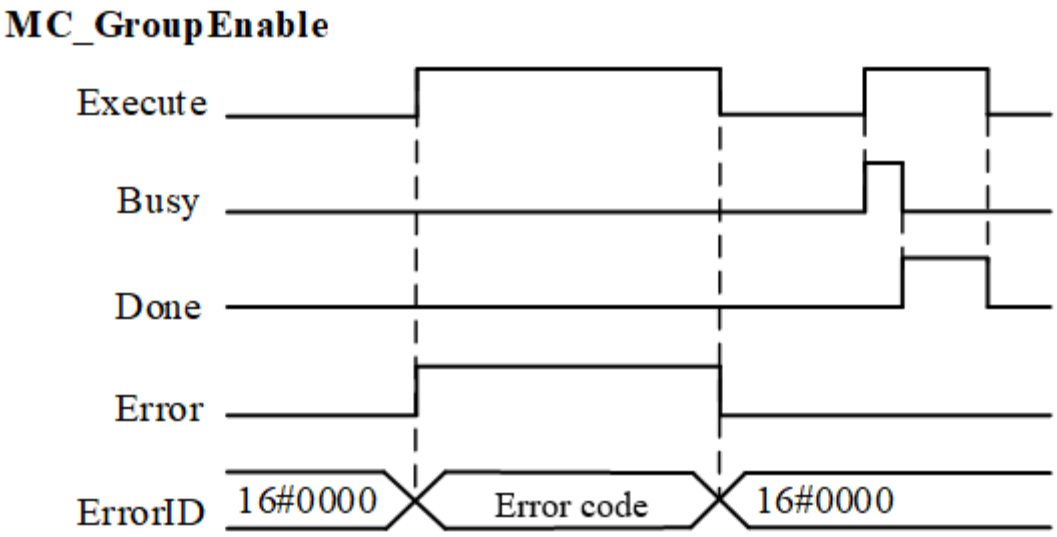
The following is a normal and error timing diagram when the MC_GroupEnable is executed.

(1) Normal timing diagram



(2) Error pin timing diagram

When this command is executed, there are abnormal conditions such as illegal axis ID and illegal axis type in the axis group, and this command fails to be called. At this time, the command output pin Error=True, and the error code ErrorID reports an abnormal error value. Please refer to the appendix Description of error codes to understand the cause of the error. When the command input pin Execute is low, all output pins are restored to their Default Values.



- After the error condition of the command is resolved, re-execute the MC_GroupEnable function block to complete the axis group enable.

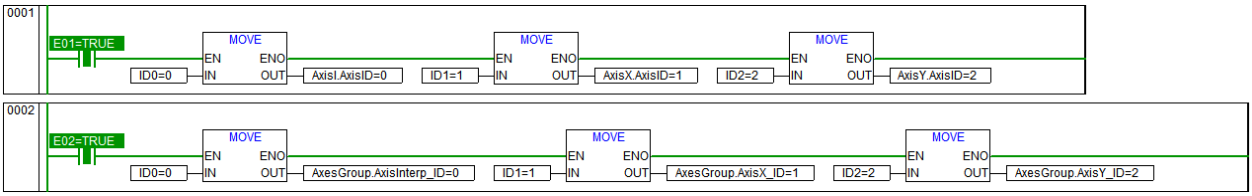
7.1.3 Examples

■ LD program implementation

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enabling operation.
Power0_Status	BOOL	FALSE	Variable for successful combination of axes enable. This variable becomes TRUE when the combined axis is enabled successfully.
Power1_Status	BOOL	FALSE	Variables whose sub-axis-axis X enable is successful. When the sub-axis X is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variables whose sub-axisY enable is successful. When the sub-axis Y is enabled successfully, this variable becomes TRUE.
GrpEnable1_Exe	BOOL	FALSE	Axis group enable rising edge trigger variable.

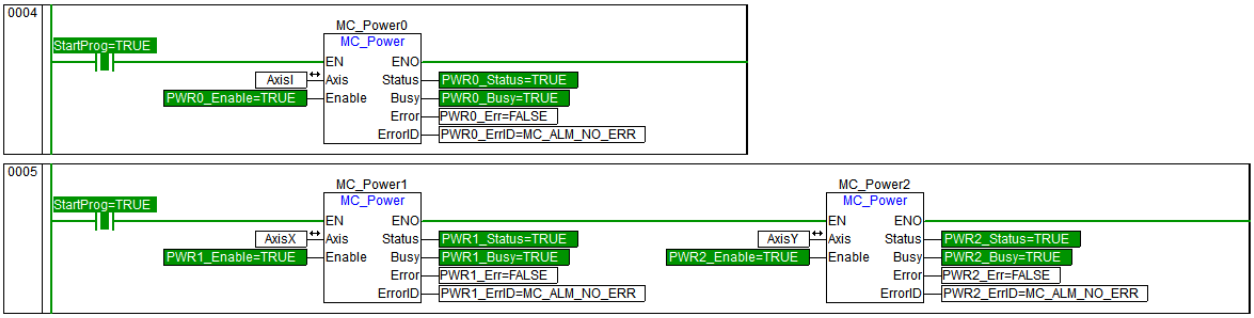
(1) Set the IDs of joint and sub-axes in the axis group. When defining an axis group, you need to define three axes in the axis group first, and match the IDs of the defined axis numbers of the three axes with those of the combined and sub-axes in the axis group. See sections 0001 and 0002 below.



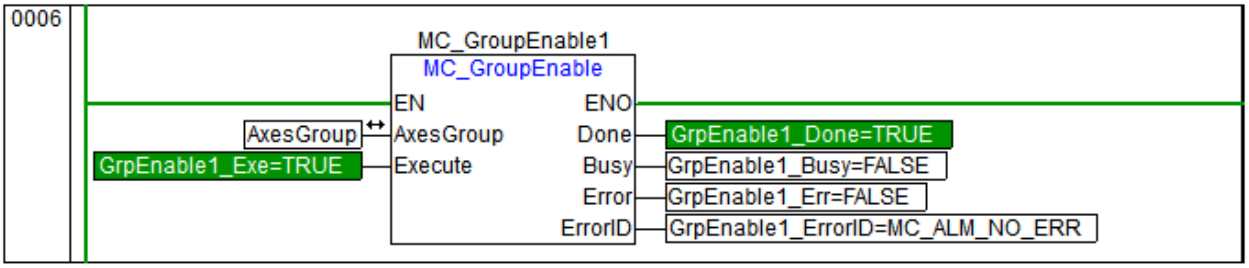
(2) Axis initialization. Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group respectively, and set the combined axis as 0: Virtual and the sub-axis axis type as 50: EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to their respective commands.

- Note: If the axis type for a combined axis is not explicitly set and remains as the default 'unused' state, the MC_GroupEnable command will automatically configure the combined axis as a virtual axis and enable it. In this example, the combined axis is defined as a virtual axis, and the MC_Power command must be invoked to complete the combined axis enable.

(3) Axis enable. After the initialization of each axis in the axis group is completed and the axis ready state StartProg is TRUE, execute the axis enable command to complete the axis enable. See Sections 0004 to 0005.



(4) Axis group enable. Call MC_GroupEnable to enable the axis group. If AxesGroupDisabled in the axis group variables is FALSE, the axis group is enabled successfully. See Section 0006.



■ ST language construction program

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enabling operation.
Power0_Status	BOOL	FALSE	Variable for successful combination of axis enable. This variable becomes TRUE when the combined axis is enabled successfully.
Power1_Status	BOOL	FALSE	Variables whose sub-axis X enable is successful. When the sub-axis X is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variables whose sub-axis Y enable is successful. When the sub-axis Y is enabled successfully, this variable becomes TRUE.
GrpEnable1_Exe	BOOL	FALSE	Axis group enable rising edge trigger variable.

(1) Set the IDs of combined axis and sub-axis in the axis group. Set the axis ID of the defined combined axis AxisI, sub-axes AxisX and AxisY.

```
IF Switch1 THEN
  AxisI.AxisID:=0;
  AxisX.AxisID:=1;
  AxisY.AxisID:=2;
  AxesGroup.AxisInterp_ID:=AxisI.AxisID;
  AxesGroup.AxisX_ID:=AxisX.AxisID;
  AxesGroup.AxisY_ID:=AxisY.AxisID;
END_IF
```

(2) Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group. Set the axis type of axis combination to 0: Virtual; set the axis types of axis group AxisX/AxisY to 50: EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to its command section.

(3) After the initialization of each axis in the axis group is completed and the ready state StartProg of the axis is TRUE, execute the axis enable command to complete the axis enable.

```
IF StartPro THEN
MC_Power0(*Combined axis enable*)
   Enable := PWR0_Enable,
   Axis := AxisI,
   Status=> PWR0_Status,
   Busy=> PWR0_Busy,
Error=> PWR0_Err,
ErrorID=> PWR0_ErrID);
MC_Power1((*Sub-axis AxisX Enable*)
   Enable := PWR1_Enable,
   Axis := AxisX,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
Error=> PWR1_Err,
ErrorID=> PWR1_ErrID);
MC_Power2 (*Sub-axis AxisY Enable*)
   Enable := PWR2_Enable,
   Axis := AxisY,
   Status=> PWR2_Status,
   Busy=> PWR2_Busy,
   Error=> PWR2_Err,
   ErrorID=> PWR2_ErrID);
END_IF
```

(4) Execute axis group enable

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
```

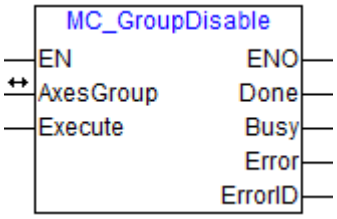
7.1.4 Precautions

- After the axis group is enabled, it is forbidden to manually modify the axis ID again;
- The AxesGroupDisabled state variable of the axis group is controlled by MC_GroupEnable/MC_GroupDisable commands and cannot be modified manually;

- After the axis group is enabled, it is forbidden to manually switch the axis type to an axis type not supported by the axis group.

7.2 MC_GroupDisable (Disable Axes Group)

Disable the axis group function.



7.2.1 Parameter

Parameter Type	Title	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	AxesGroup	Axis group name	No	-	-	AXES_GROUP_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
OUTPUT	Done	Completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

7.2.2 Functions

This order disables the axis group function.

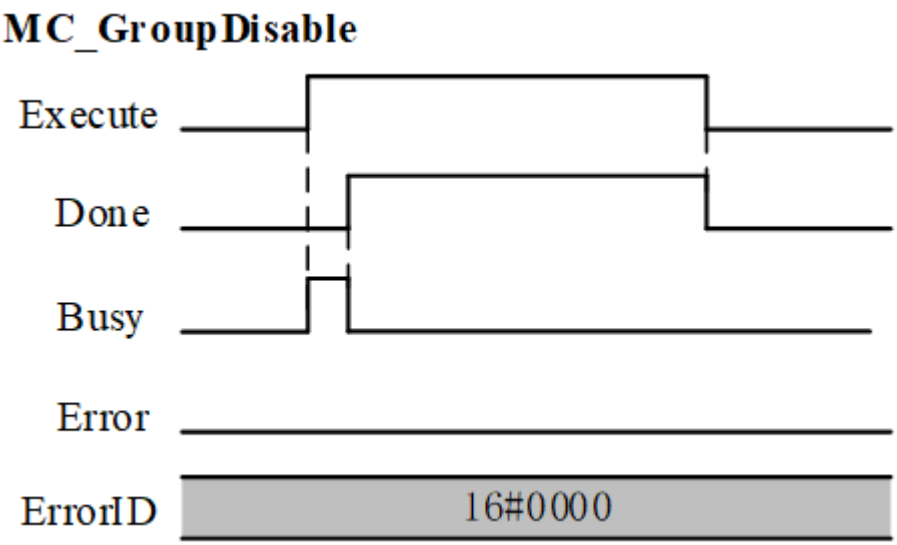
- command function Description

(1) Execute the command of rising edge, and the axis parameters are checked to be normal. The MC_GroupDisable command sets the AxesGroupDisabled variable in the specified axis group to TRUE, and disables the axis group function. Disable the axis group.

(2) After the command is successfully executed, the action of the axis group will be stopped (without affecting the operation of single axis and synchronous commands).

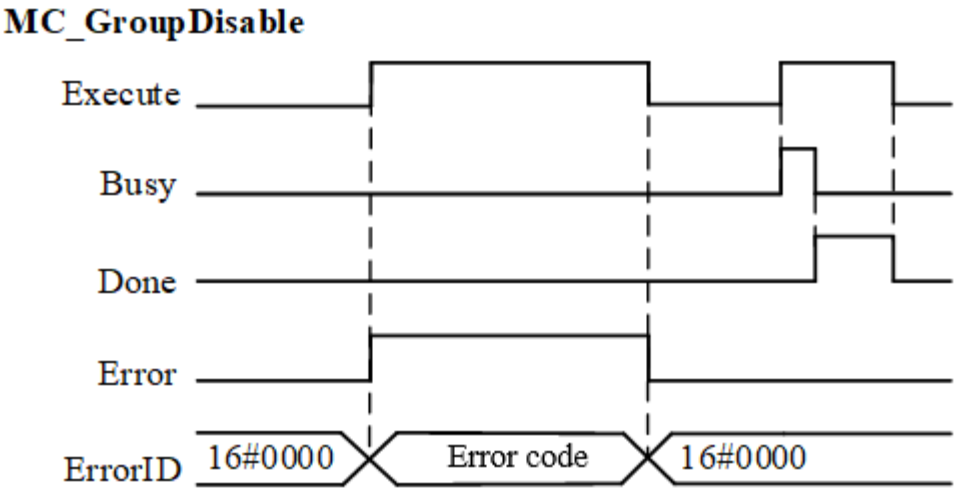
- Timing diagram

(1) Normal timing diagram



(2) Error pin timing diagram

When this command is executed, there are abnormal conditions such as illegal axis ID and illegal axis type in the axis group, and this command fails to be called. At this time, the command output pin Error=True, and the error code ErrorID reports an abnormal error value. Please refer to the appendix Description of Error Codes to understand the cause of the error. When the command input pin Execute is low, all output pins are restored to their Default Values.



- After the abnormal condition of the command is solved, re-execute the MC_GroupDisable function block to disable the axis group enable.

7.2.3 Examples

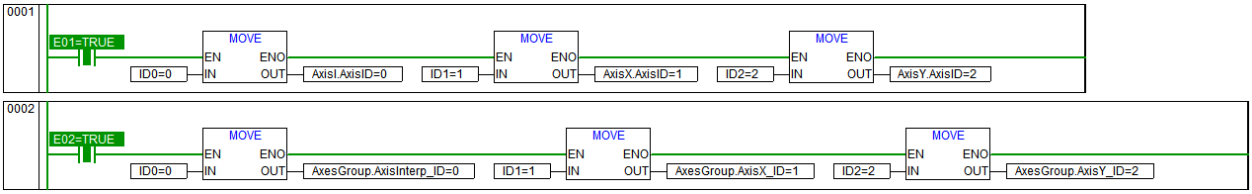
■ Variable

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enabling operation.
Power0_Status	BOOL	FALSE	Variable for successful combination of axes enabling. This variable becomes TRUE when the axis group is enabled successfully.
Power1_Status	BOOL	FALSE	Variables whose sub-axis X enable is successful. When the sub-axis X is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variables whose sub-axis Y enable is successful. When the sub-axis Y is enabled successfully, this variable becomes TRUE.
GrpDisable1_Exe	BOOL	FALSE	Axis group disconnection enable rising edge trigger variable.

■ LD diagram program

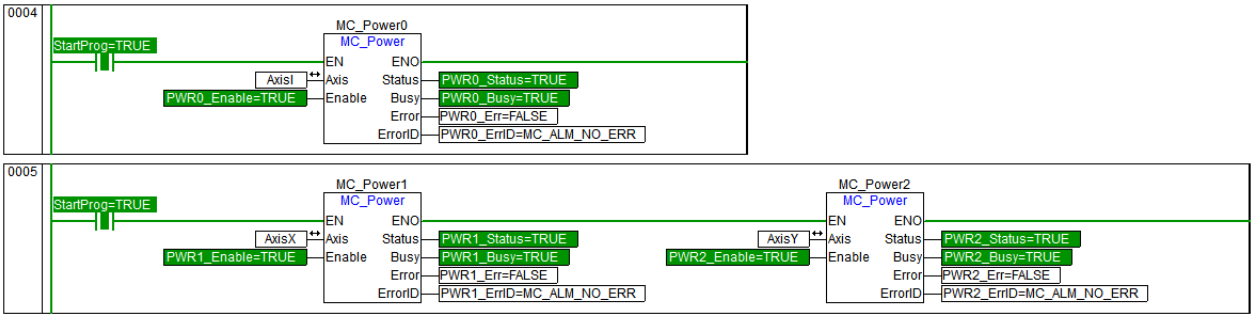
(1) Set the IDs of joint and sub-axes in the axis group. When defining an axis group, you need to define three axes in the axis group first, and match the IDs of the defined axis numbers of the three axes with those of the combined and sub-axes in the axis group. See sections 0001 and 0002 below.



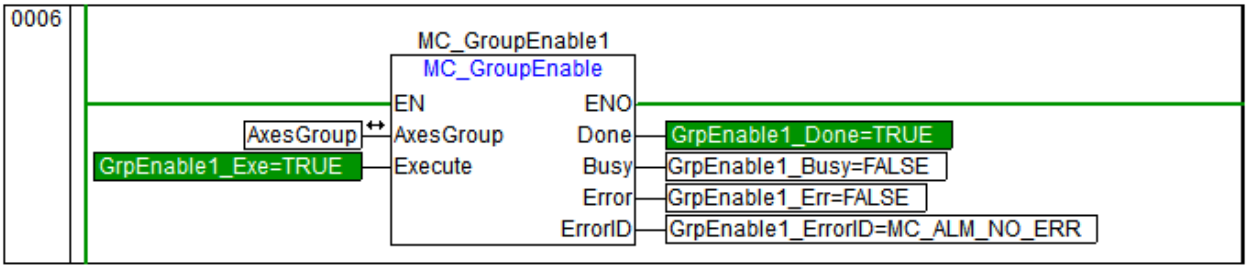
(2) Axis initialization. Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group respectively, and set the axis group as 0: Virtual and the Sub-axis axis type as 50: EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to their respective commands.

- Note: When the axis type of the axis group is not set and it is the default unused axis, the MC_GroupEnable command will automatically configure the axis type of the axis group as a virtual axis and enable the axis group by itself. In this example, the combined axis is set as a virtual axis, and MC_Power needs to be called to enable the axis group.

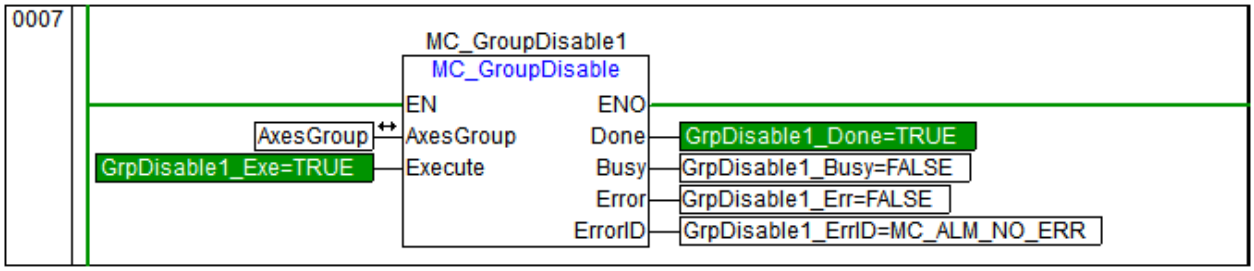
(3) Axis enable. After the initialization of each axis in the axis group is completed and the axis ready state StartProg is TRUE, execute the axis enabling command to complete the combined axis enable. See sections 0004 to 0005.



(4) Axis group enable. Call MC_GroupEnable to enable the axis group. Set AxesGroup.AxesGroupDisabled in the axis group variables to FALSE, and enable the axis group. See Section 0006.



(5) Disable axis group. Call MC_GroupDisable to disable the axis group. AxesGroupDisabled in the axis group variables changes to TRUE, and the axis group is enabled. See Section 0007.



■ ST language construction program

(1) Set the IDs of combined axis and sub-axes in the axis group. Set the axis ID of the defined combined axis AxisI, sub-axes AxisX and AxisY.

```
IF Switch1 THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
END_IF
```

(2) Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group. Set the axis type of axis combination to 0: Virtual; set the axis types of axis group AxisX/AxisY to 50: EtherCAT_Position. For the usage of SMC_AxisInit command, please refer

to its comand section.

(3) After the initialization of each axis in the axis group is completed and the ready state StartPro of the axis is TRUE, execute the axis enable command to complete the axis enable.

```
IF StartPro THEN
MC_Power0(*Combined axis enable*)
   Enable := PWR0_Enable,
   Axis := AxisI,
   Status=> PWR0_Status,
   Busy=> PWR0_Busy,
   Error=> PWR0_Err,
   ErrorID=> PWR0_ErrID);
MC_Power1(*Sub-axis AxisX Enable*)
   Enable := PWR1_Enable,
   Axis := AxisX,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
   Error=> PWR1_Err,
   ErrorID=> PWR1_ErrID);
MC_Power2 (*Sub-axis AxisY Enable*)
   Enable := PWR2_Enable,
   Axis := AxisY,
   Status=> PWR2_Status,
   Busy=> PWR2_Busy,
   Error=> PWR2_Err,
   ErrorID=> PWR2_ErrID);
END_IF
```

(4) Execute axis group enable

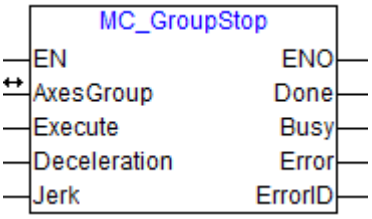
```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
```

(5) Execute MC_GroupDisable to disable the axis group function

```
MC_GroupDisable1(
Execute := GrpDisable_Exe,
AxesGroup := AxesGroup,
Done=> GrpDisable_Done,
Busy=> GrpDisable_Busy,
Error=> GrpDisable_Err,
ErrorID=> GrpDisable_ErrorID);
```

7.3 MC_GroupStop (AxesGroup Motion Stop)

Control the axis group to decelerate and stop, suspend any motion control command being executed on the axis group, and cancel all motion buffer command on the combined axis of axis group.



7.3.1 Parameter

Parameter Type	Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	AxesGroup	Axis group name	No	-	-	AXES_GROUP_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	Deceleration	Deceleration	No	-	Positive value	LREAL
	Jerk	Jerk 0: Trapezoidal acceleration and deceleration Positive: S-shaped acceleration and deceleration	YES	0	0/positive	LREAL
OUTPUT	Done	Stop complete	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

7.3.2 Functions

This command controls the axis group to decelerate and stop from the current speed according to the specified deceleration, and suspends any motion control command being executed on the combined axis in this axis group.

■ Command function details

(1) Deceleration parameter setting

The deceleration and jerk at the time of deceleration stop are specified by the input variables Deceleration (deceleration) and Jerk (jerk).

(2) Axis status when running this command

When the command rising edge triggers the pin to remain at high level, and the axis group is always in Stopping status, the axis group cannot send an interpolation motion command. When

calling MC_GroupStop, only the axis group are in Stopping status.

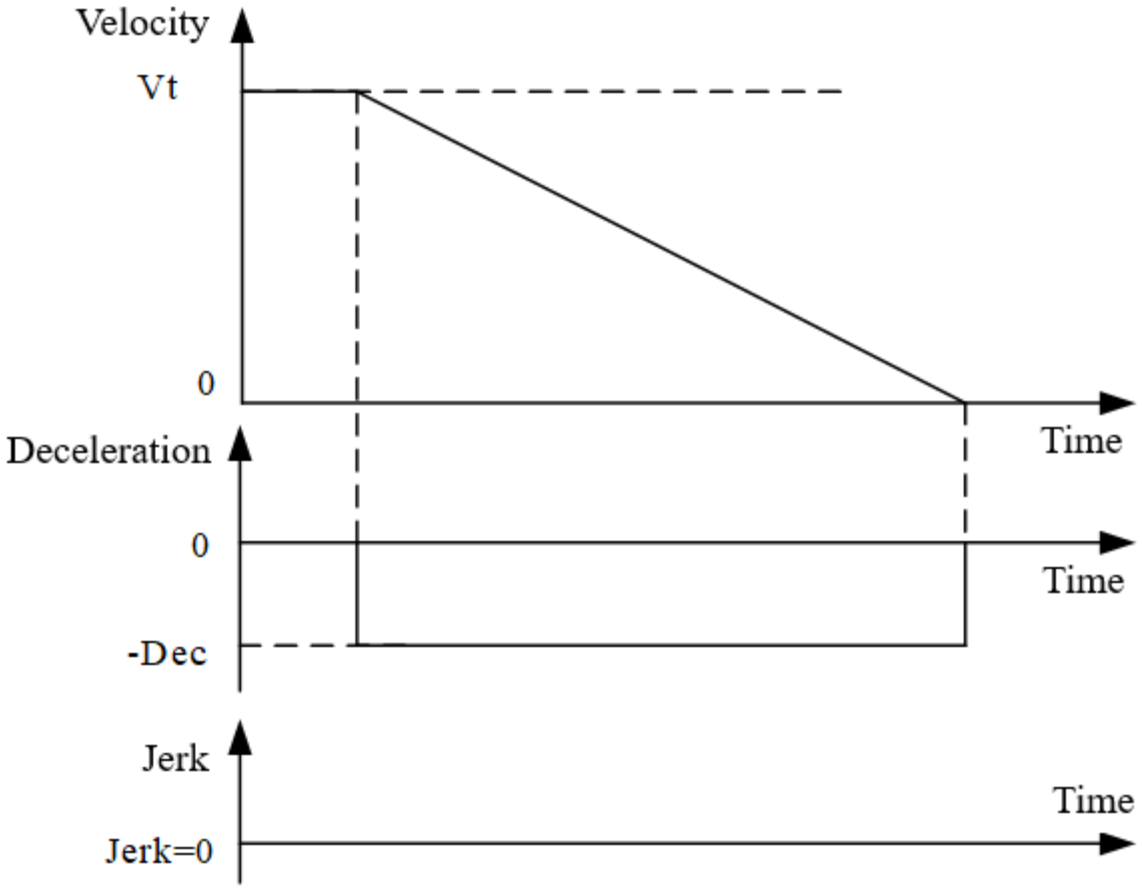
(3) Input parameter latch

When the rising edge of this command is triggered, each axis ID, input parameter Deceleration and Jerk value of the input and output parameter axis group will be latched. During the execution of the command, the modification of latched parameters will be invalid until this command is repeatedly triggered. In addition, after the AxesGroup is enabled, it is forbidden to modify parameters such as combined axis ID, sub-axis ID and axis type of AxesGroup.

(4) Acceleration and deceleration curve

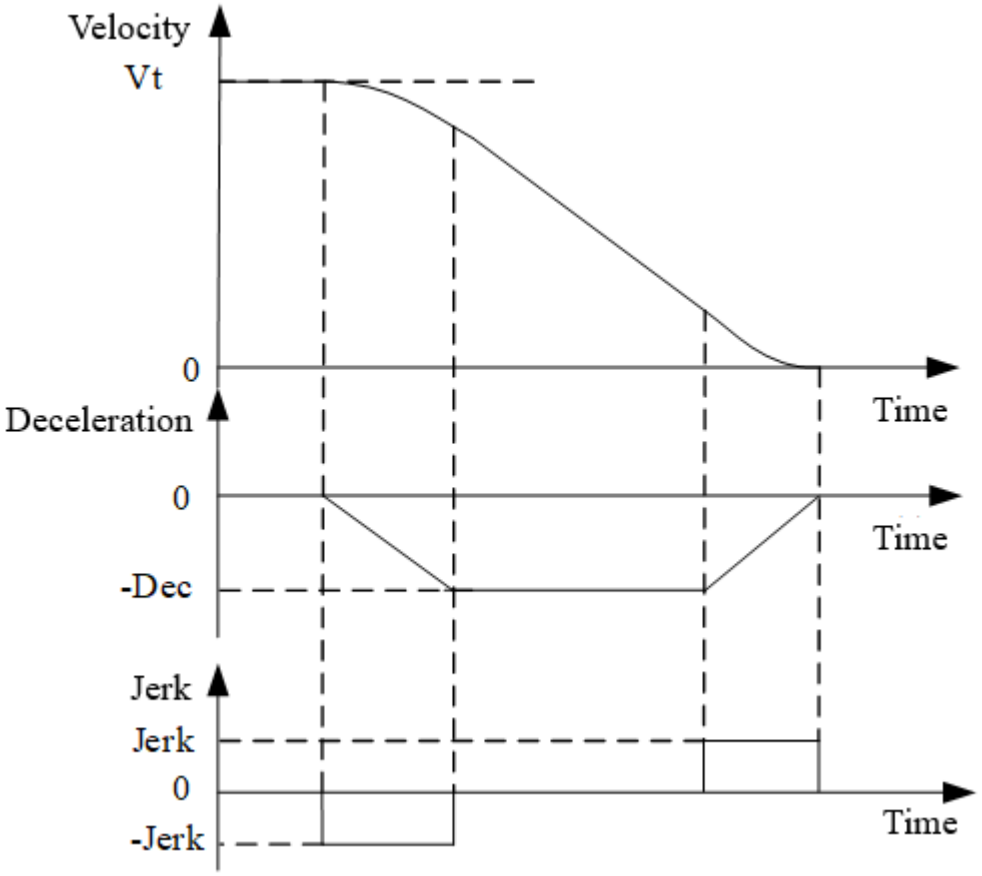
If the input parameter Jerk=0, the deceleration curve is a trapezoidal acceleration/deceleration curve; if the input parameter Jerk>0, the deceleration curve is an S-shaped acceleration/deceleration curve.

□ When Jerk=0, the motion deceleration curve of axis group is a trapezoidal acceleration and deceleration curve as follows:



Vt: speed when deceleration starts, Dec: set value of deceleration, Jerk: set value of jerk

□ When Jerk>0, the command value of speed is generated by deceleration. The axis group motion deceleration curve is an S-shaped acceleration and deceleration curve as follows:



Vt: speed when deceleration starts, Dec: set value of deceleration, Jerk: set value of jerk

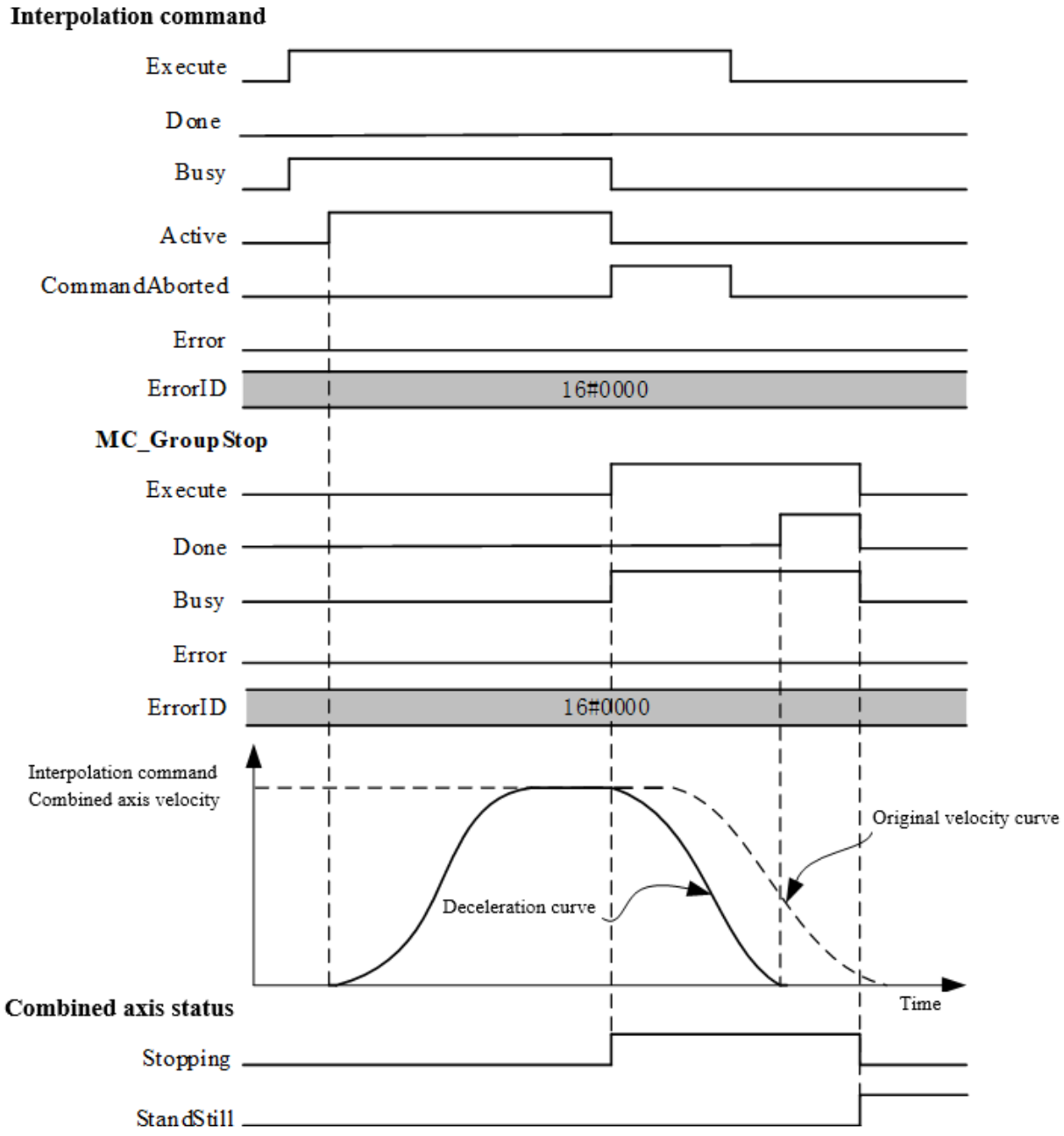
When the MC_GroupStop command is executed, it is possible to decelerate and stop the SMC_MoveLinearAbsolute (linear interpolation of absolute value of 2 axes), SMC_MoveLinearRelative (linear interpolation of relative value of 2 axes) and SMC_MoveCircular2D (circular interpolation of 2 axes) commands with the set deceleration parameters.

If the MC_GroupStop command is executed during the execution of the interpolation command, the axis group will decelerate and stop on the track of linear or circular interpolation. The interpolation command in the axis group is interrupted.

■ Timing diagram

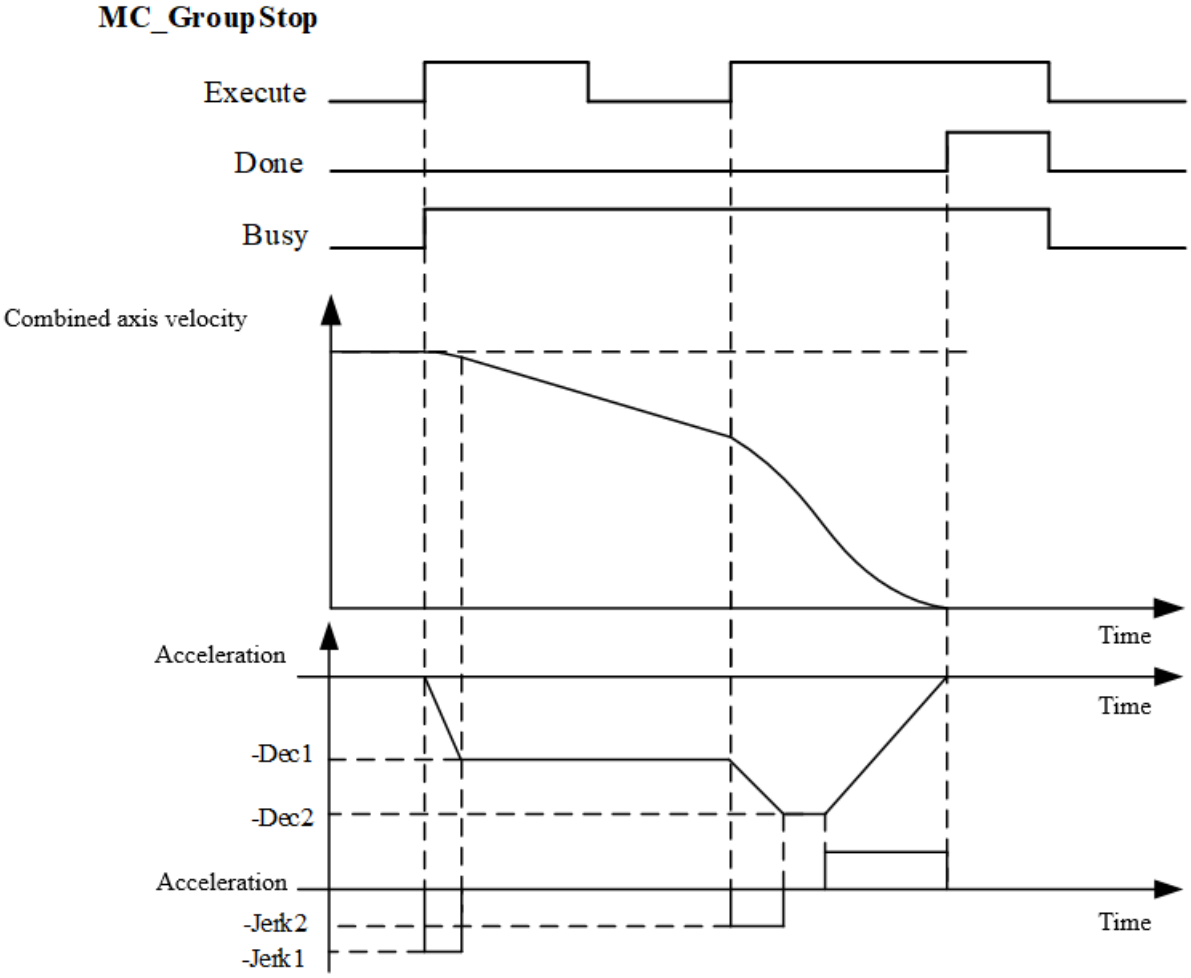
(1) Normal timing diagram

The rising edge triggering of this command is valid. Execute the rising edge of the input, if there is no error, the Busy pin is set to TRUE. When the command is executed, the axis group is in Stopping status, and the motion of the axis group decelerates and stops. After the deceleration of axis group motion is completed, set "Done" to TRUE. If Execute remains at high level, the Busy and Done pins remain TRUE, and the axis group is continuously in Stopping status; if Execute is set at low level, the Busy and Done pins are set to FALSE, and the combined axis is switched to StandStill status.



(2) Repeated delivery command pin timing diagram

When multiple MC_GroupStop functional blocks are triggered repeatedly, the deceleration (deceleration) of the last delivery command is used for deceleration and stop control, and the command changes the deceleration and jerk.

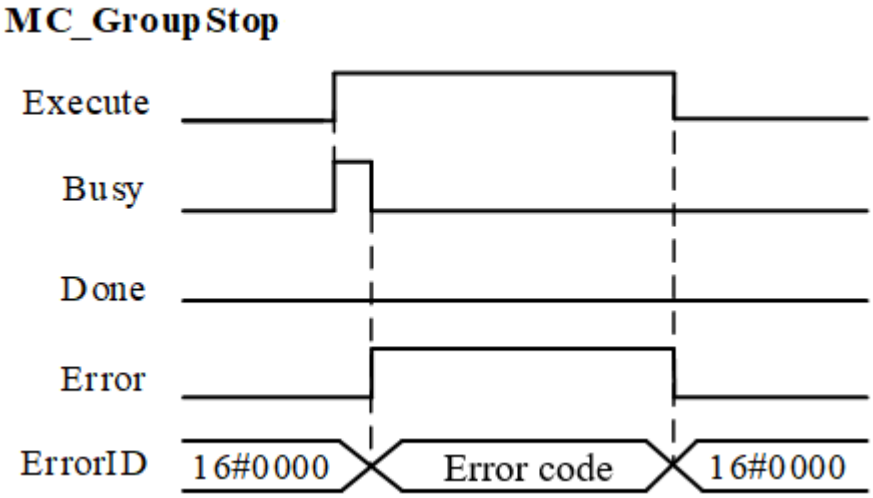


Dec1: deceleration commanded at first delivery, Dec2: deceleration at second delivery

Jerk 1: JERK of the first delivery command, Jerk 2: Jerk at the second delivery

(3) Error pin timing diagram

When this command is executed, if the command input parameters are illegal, the axis group is in Disabled status or ErrorStop status and other abnormalities occur, the call of this command will fail. At this time, the command output pin Error=True, and the error code ErrorID reports an error value. Please refer to the appendix Description of error codes to understand the cause of the error. When the command input pin Execute is low, all output pins are restored to their Default Values.



- When the error command is solved, it is necessary to trigger Execute again on the rising edge and start the MC_GroupStop function block to execute the stop action of the control axis group according to the specified deceleration.

7.3.3 Examples

In this example, the MC_GroupStop command is used to stop the linear interpolation motion command for relative position of 2 axes, and the MC_ReadStatus command is used to read the axis status of the axis group.

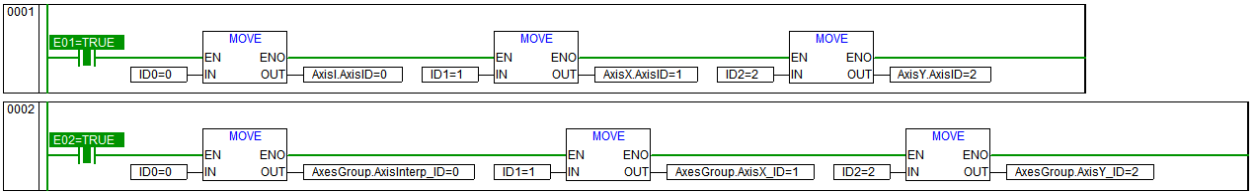
■ LD diagram program

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enabling operation.
Power0_Status	BOOL	FALSE	Variable for successful combination of axes enabling. This variable becomes TRUE when the combined axis is enabled successfully.
Power1_Status	BOOL	FALSE	Variables whose Sub-axis-axis X enable is successful. When the Sub-axis axis X is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variables whose Sub-axis-axis Y enable is successful. When the Sub-axis axis Y is enabled successfully, this variable becomes TRUE.
GrpEnableStatus	BOOL	TRUE	When this variable is TRUE, the axis group is enabled.
GrpEnable1_Exe	BOOL	FALSE	Axis group enable rising edge trigger variable.
Rel2D1_Exe	BOOL	FALSE	The rising-edge trigger variable of the relative interpolation command sent by the axis group.

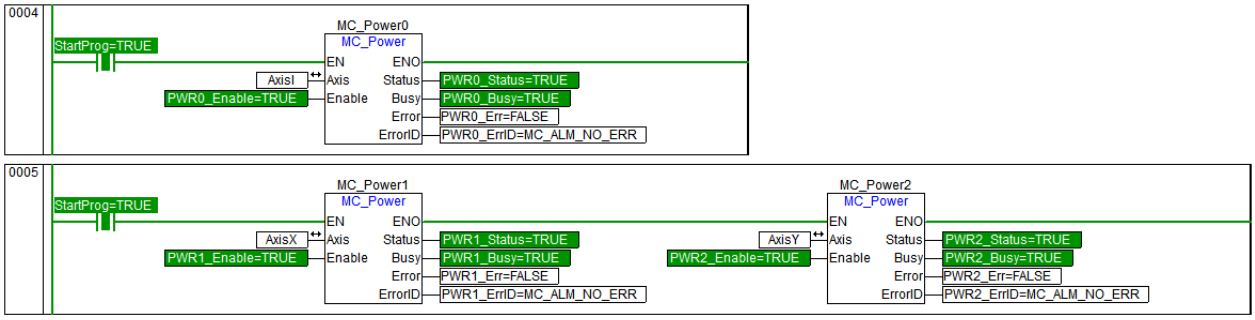
Rel2D1_ Comdabort	BOOL	FALSE	Interrupt pin variable for relative interpolation commands projected by the axis group.
GrpStop1_Busy	BOOL	FALSE	Busy variable for a stop command delivered by the axis group.
GrpStop1_Don3	BOOL	FALSE	Done variable for a stop command delivered by the axis group. This variable becomes TRUE when the shaft Combined stop is completed.
Stopping	BOOL	FALSE	Axis status readout command Stopping pin variable for axis combined axes. This variable becomes TRUE when the joint starts to decelerate and stop.

(1) Set the IDs of combined axes and sub-axes in the axis group. When defining an axis group, you need to define three axes in the axis group first, and match the IDs of the defined axis numbers of the three axes with those of the combined and sub-axes in the axis group. See sections 0001 and 0002 below.

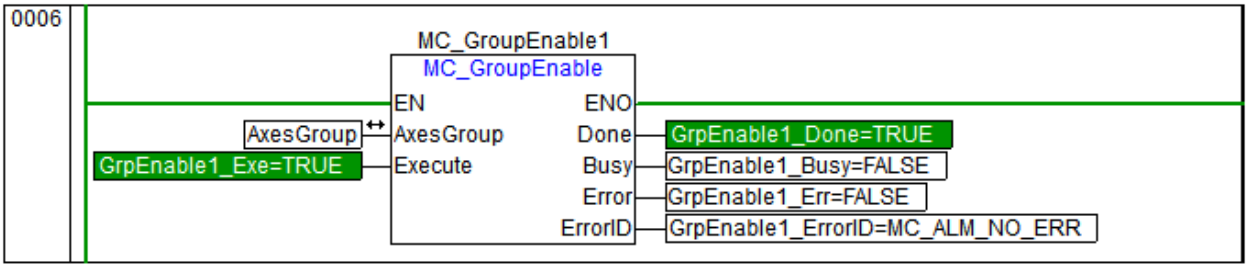


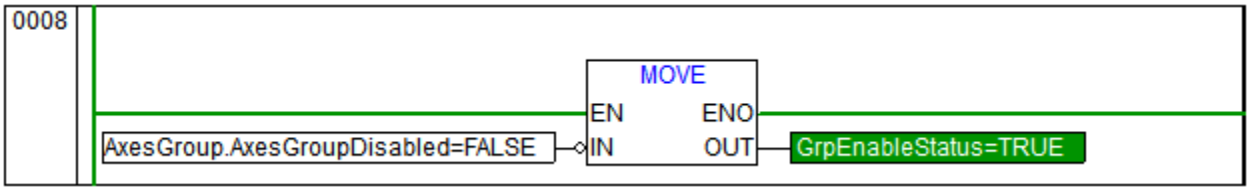
(2) Axis initialization. Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group respectively, and set the combined axis as 0: Virtual and the Sub-axis axis type as 50: EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to their respective commands.

(3) Axis enable. After each axis in the axis group is ready (StartProg =TRUE), call MC_Power to enable the axes of the combined axis and two sub-axes of the axis group. See sections 0004 to 0005.

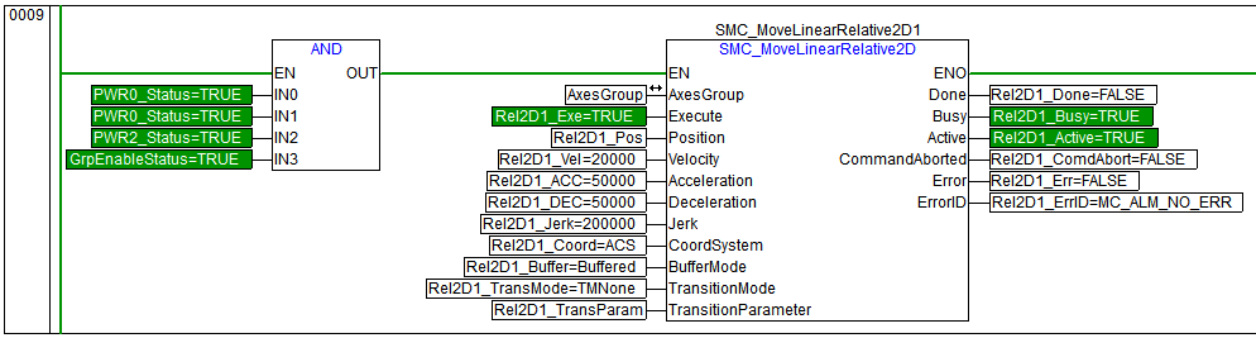


(4) Axis group enable. Call MC_GroupEnable to enable the axis group. If AxesGroupDisabled in the axis group variables is FALSE, the axis group is enabled successfully. See Section 0006. After the axis group is enabled successfully, reverse AxesGroupDisabled and assign it to GrpEnableStatus variable.



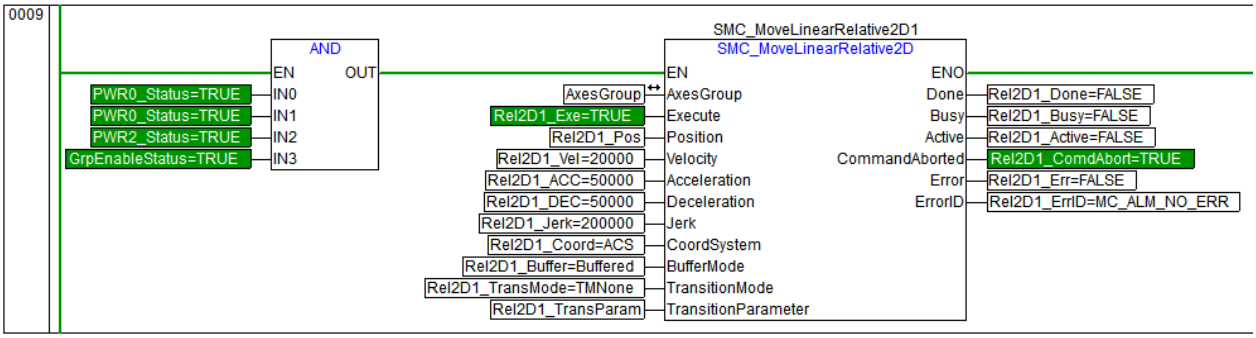


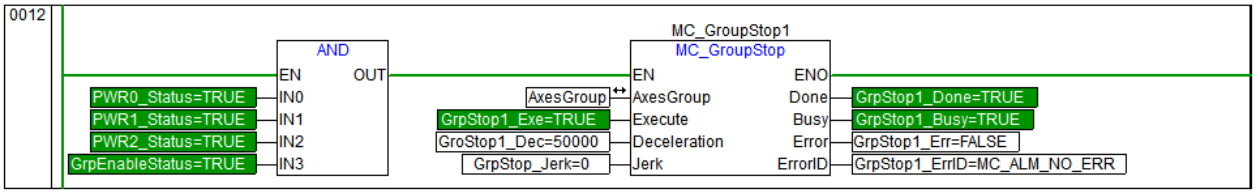
(5) In Section 0009, after each axis in the axis group and the axis group are enabled, trigger the relative position interpolation motion command to control the action of the axis group. For details about how to use the SMC_MoveLinearRelative2D command, see its description document.

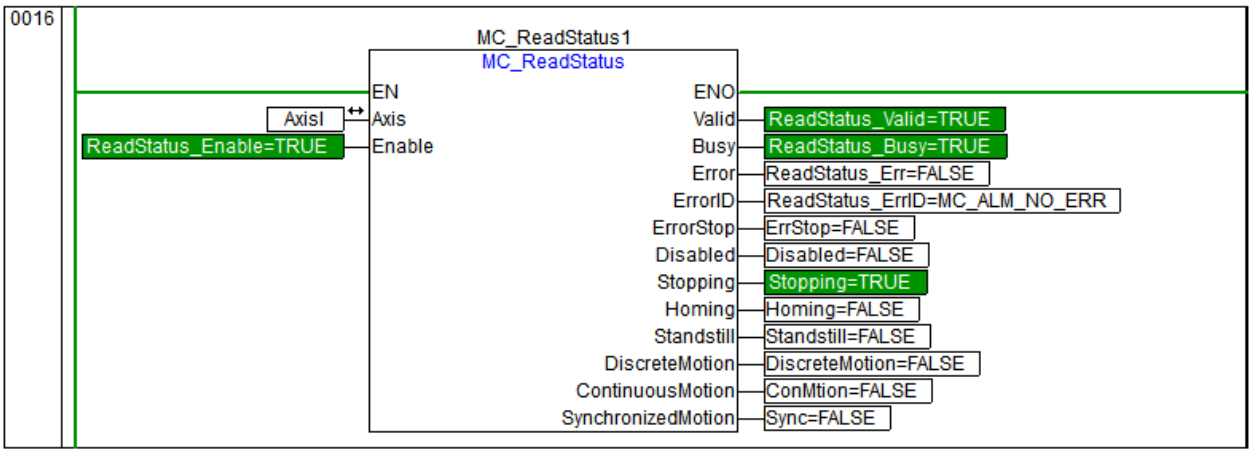


(6) Section 0012 calls MC_GroupStop command to stop the interpolation motion of axis group, with deceleration set at 50000 and Jerk at 0. The trapezoidal curve is used for deceleration.

The output pin CommandAborted of the interpolation motion command SMC_MoveLinearRelative2D1 is set to TRUE when the MC_GroupStop command controls the deceleration stop of the axis group, and the interpolation command is interrupted as described in Section 0009 below. At this point, look at the output pin of MC_ReadStatus1 in section 0016 to determine the status of the combined axis of axis group.







■ ST language

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enable.
InitFlag	BOOL	FALSE	If the variable is TRUE, initialization is complete.
Power0_Status	BOOL	FALSE	Variable for successful combination of axes enabling. This variable becomes TRUE when the combined axis is enabled successfully.
Power1_Status	BOOL	FALSE	Variables whose Sub-axis-axis X enable is successful. When the sub-axis X is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variables whose Sub-axis-axis Y enable is successful. When the sub-axis Y is enabled successfully, this variable becomes TRUE.

GrpEnableStatus	BOOL	TRUE	When this variable is TRUE, the axis group is enabled.
Rel2D1_Exe	BOOL	FALSE	The rising edge trigger variable for interpolation commands sent by the axis group.
GrpStop_Exe	BOOL	FALSE	The rising edge trigger variable for aborting commands sent by the axis group.

(1) Set the IDs of combined and sub-axes in the axis group. Set the axis ID of the defined combined axis AxisI, sub-axes AxisX and AxisY.

```
IF FALSE=InitFlag THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
   GrpStop_Dec:=50000;
   GrpStop_Jerk:=0;
   InitFlag:=TRUE;
END_IF
```

(2) Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group. Set the axis group type of the combined axis to 0: Virtual; set the axis types of axis group AxisX/AxisY to 50: EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to its command section.

(3) Execute the MC_Power command to enable the three axes in the axis group.

```
IF StartPro THEN
MC_Power0(*Combined axis enable*)
   Enable := PWR0_Enable,
   Axis := AxisI,
   Status=> PWR0_Status,
   Busy=> PWR0_Busy,
   Error=> PWR0_Err,
   ErrorID=> PWR0_ErrID);
MC_Power1((*Sub-axis AxisX Enable*)
   Enable := PWR1_Enable,
   Axis := AxisX,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
   Error=> PWR1_Err,
   ErrorID=> PWR1_ErrID);
MC_Power2 (*Sub-axis AxisY Enable*)
   Enable := PWR2_Enable,
   Axis := AxisY,
   Status=> PWR2_Status,
   Busy=> PWR2_Busy,
   Error=> PWR2_Err,
   ErrorID=> PWR2_ErrID);
```

```
END_IF
```

(4) Execute axis group enable

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
GrpEnableStatus:= AxesGroup. AxesGroupDisabled;
```

(5) After the axis and axis group enable function is operated, the rising edge triggers the axis group interpolation command. For the usage of SMC_MoveLinearRelative2D command, please refer to its command section.

```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus THEN
   SMC_MoveLinearRelative2D1(
   Execute := Rel2D1_Exe,
   Position := Rel2D1_Pos,
   Velocity := Rel2D1_Vel,
   Acceleration := Rel2D1_ACC,
   Deceleration := Rel2D1_DEC,
   Jerk := Rel2D1_Jerk,
   CoordSystem := Rel2D1_Coord,
   BufferMode := Rel2D1_Buffer,
   TransitionMode := Rel2D1_Trans,
   TransitionParameter := Rel2D1_Param,
   AxesGroup := AxesGroup,
   Done=> Rel2D1_Done,
   Busy=> Rel2D1_Busy,
   Active=> Rel2D1_Active,
   CommandAborted=> Rel2D1_ComdAbort,
   Error=> Rel2D1_Err,
   ErrorID=> Rel2D1_ErrID);
END_IF
```

(6) Call MC_GroupStop command to stop the interpolation motion of axis group, set the deceleration as 50000 and Jerk as 0, and slow down in trapezoidal curve. When the MC_GroupStop command controls the deceleration stop of the axis group, the output pin CommandAborted of the interpolation motion command SMC_MoveLinearRelative2D1 is set to TRUE and the interpolation command is interrupted. At this time, the output pin of MC_ReadStatus1 determines the status of the combined axis.

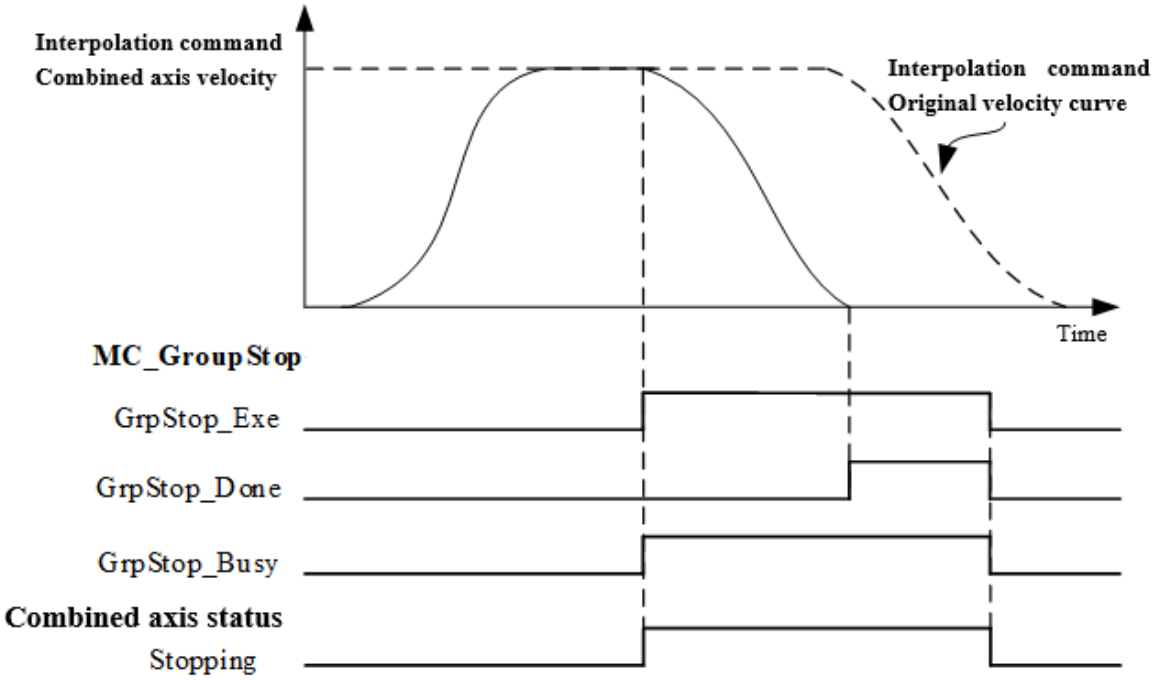
```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus THEN
   MC_GroupStop1(
   Execute := GrpStop_Exe,
   Deceleration := GrpStop_Dec,
   Jerk := GrpStop_Jerk,
```

```
   AxesGroup := AxesGroup,
   Done=> GrpStop_Done,
   Busy=> GrpStop_Busy,
   Error=> GrpStop_Err,
   ErrorID=> GrpStop_ErrID);
END_IF
```

(7) Read the status of combined axis

```
MC_ReadStatus1(
Enable := ReadStatus Enable,
Axis := AxisI,
Valid=> ReadStatus_Valid,
Busy=> ReadStatus_Busy,
Error=> ReadStatus_Err,
ErrorID=> ReadStatus ErrID,
ErrorStop=>ErrStop,
Disabled=>Disabled,
Stopping=>Stopping,
Homing=>Homing,
Standstill=>Standstill,
DiscreteMotion=>DiscreteMotion,
ContinuousMotion=>ConMtion,
SynchronizedMotion=>Sync);
```

The following figure shows that the axis group interpolation command is interrupted by MC_GroupStop.

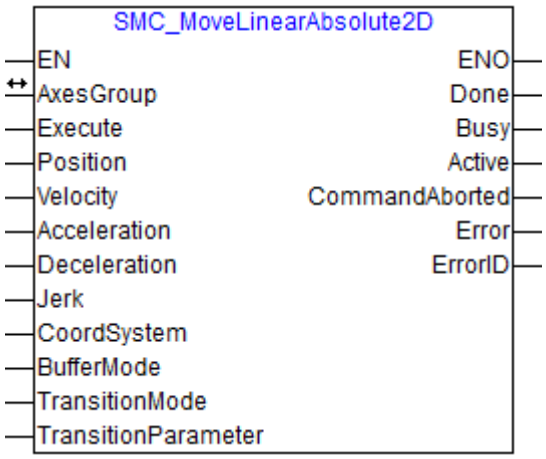


7.3.4 Precautions

- When MC_GroupStop is called, the axis group needs to be enabled.
- When MC_GroupStop is called, after the axis group motion is successfully stopped, if the input pin Execute of MC_GroupStop is at high level, the combined axis of axis group will remain in Stopping state. At this time, the interpolation command fails to be sent. When the input pin Execute is low level, the combined axis switches to StandStill status and can normally send interpolation commands.

7.4 SMC_MoveLinearAbsolute2D (Absolute Linear Interpolation on Two Axes)

This command realizes the linear interpolation of 2 axes absolute position.



7.4.1 Parameter

Parameter type	TITLE	DESCRIPTION	Empty or not	Default value	SCOPE	Data type
IN_OUT	AxesGroup	Axis group name	No	-	-	AXES_GROUP_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	Position	Target position (absolute)	No	-	Positive/negative/0	ARRAY[0..1] OF LREAL
	Velocity	Target speed	No	-	Non-negative	LREAL
	Acceleration	Acceleration	No	-	Positive value	LREAL
	Deceleration	Deceleration	No	-	Positive value	LREAL
	Jerk	Jerk 0: Trapezoidal acceleration and deceleration Positive: S-shaped acceleration and deceleration	YES	0	0/positive	LREAL
	CoordSystem	Coordinate system 0: Axis coordinate system	YES	0	0: ACS	MC_COORD_SYSTEM
	BufferMode	Buffer mode 1: Cache 2: low-speed fusion between adjacent commands 3: Speed fusion of the previous command 4: Speed fusion of the following command 5: High-speed fusion between adjacent commands	YES	1	1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	MC_BUFFER_MODE
	TransitionMode	Transition Curve Selection 0: no transition curve inserted	YES	0	0:TMNone	MC_TRANSITION_MODE

	TransitionParameter	Retained parameters	YES	0	-	ARRAY[0..1] OF LREAL
OUTPUT	Done	Interpolation motion finished	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Active	In control, TRUE when the command is executed	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

7.4.2 Functions

This command realizes the linear interpolation function of 2 axes absolute position.

- Function Description

(1) This command realizes 2-axis linear interpolation, and the target position is based on absolute position command.

(2) Target location

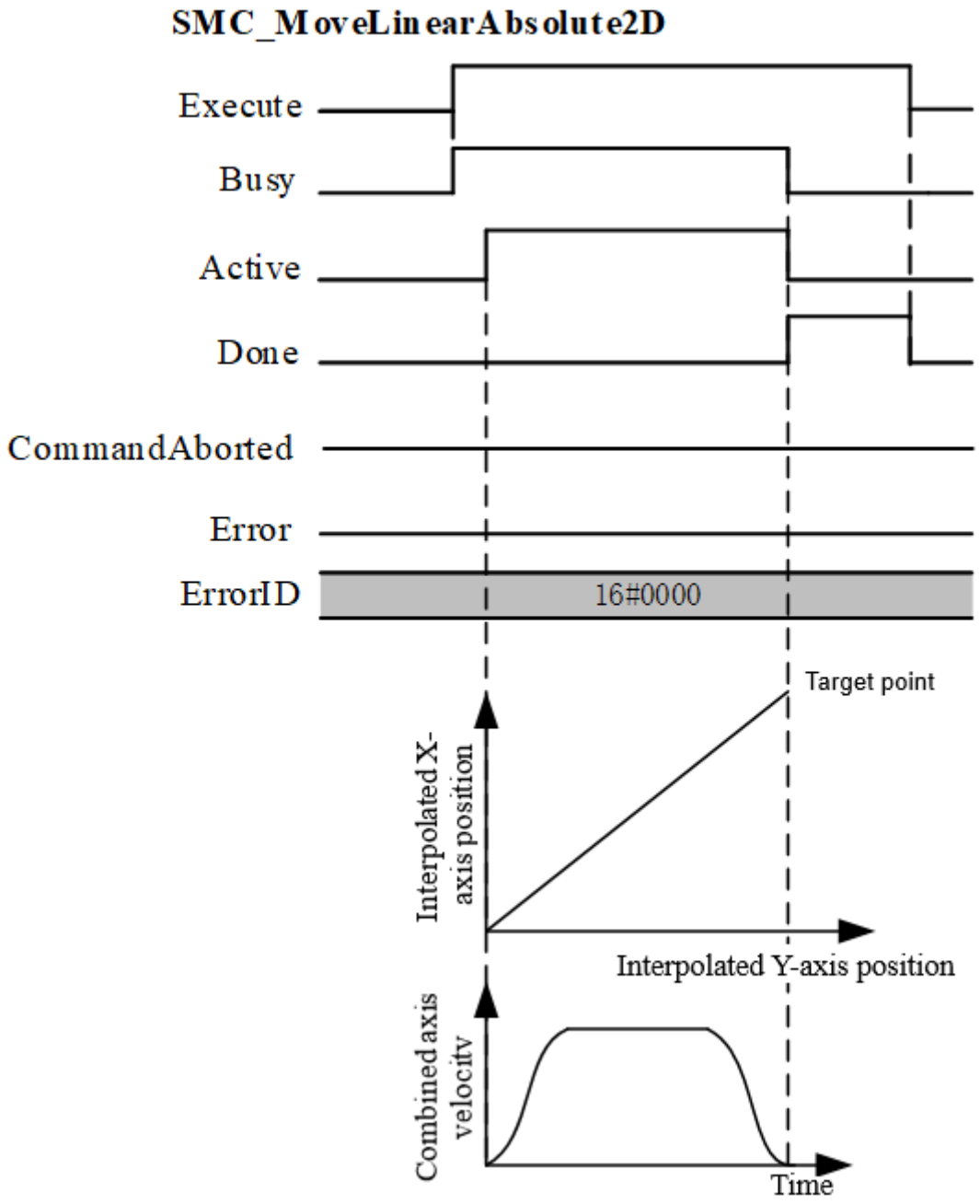
The target position of this command is absolute. With the start time of this command as the starting point and the absolute position of 2 axes set by the command input parameter as the ending point, the connecting line between the starting point and the ending point is the motion track of the command.

(3) Other functions are the same as those of SMC_MoveLinearRelative2D command.

- timing diagram

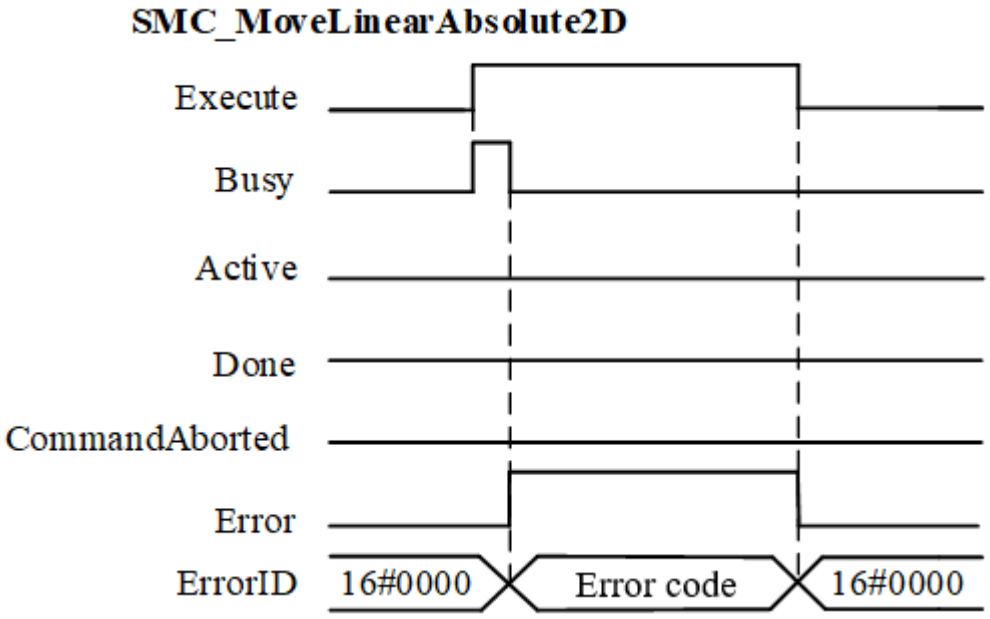
The following is a normal and error sequence diagram when the absolute position interpolation command is executed.

(1) Normal sequence diagram



(2) Error pin timing diagram

During the execution of this command, if there are errors such as illegal parameters, 0: Unused (unused axis) for an axis in the axis group, and illegal ID of each axis in the axis group, the call of this command fails, with command output pin Error=True, and the error code ErrorID reports an error value. Please refer to the appendix Error Code Description to obtain the cause of the error. When the command input pin Execute goes low, all output pins revert to their default values.



- After the error situation of the command is solved, it is necessary to trigger Execute again on the rising edge to send the axis group interpolation command and carry out the absolute position interpolation motion of the axis group.

7.4.3 Examples

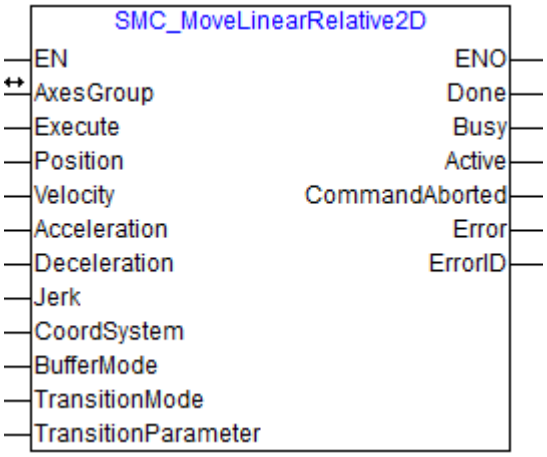
The SMC_MoveLinearAbsolute2D example refers to the SMC_MoveLinearRelative2D command. The only difference between the two commands is that the target position is an absolute position or a relative position.

7.4.4 Precautions

Same as precautions for the SMC_MoveLinearRelative2D command.

7.5 SMC_MoveLinearRelative2D (Relative Linear Interpolation on Two Axes)

This command realizes the linear interpolation function of 2-axis relative position.



7.5.1 Parameter

Parameter Type	Title	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	AxesGroup	Axis group name	No	-	-	AXES_GROUP_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	Position	Target Location (Relative)	No	-	Positive/negative/0	ARRAY[0..1] OF LREAL
	Velocity	Target speed	No	-	Non-negative	LREAL
	Acceleration	Acceleration	No	-	Positive value	LREAL
	Deceleration	Deceleration	No	-	Positive value	LREAL
	Jerk	Jerk 0: Trapezoidal acceleration and deceleration Positive: S-shaped acceleration and deceleration	YES	0	0/positive	LREAL
	CoordSystem	Coordinate system 0: Axis coordinate system	YES	0	0: ACS	MC_COORD_SYSTEM
	BufferMode	Buffer mode 1: Buffer 2: low-speed fusion between adjacent commands 3: Speed fusion of the previous command 4: Speed fusion of the following command 5: High-speed fusion between adjacent commands	YES	1	1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	MC_BUFFER_MODE
	TransitionMode	Transition Curve Selection 0: no transition curve inserted	YES	0	0:TMNone	MC_TRANSITION_MODE

	TransitionParameter	Retained parameters	YES	0	-	ARRAY[0..1] OF LREAL
OUTPUT	Done	Interpolation motion finished	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Active	In control, TRUE when the command is executed	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

7.5.2 Functions

This command realizes the linear interpolation function of 2-axis relative position.

- Interpolation steps

(1) Determine the interpolated axis group. The AxesGroup variable type is AXES_GROUP_REF.

(2) Specify the axis number in the axis group variable, corresponding to the axis number of the combined axis and sub-axis.

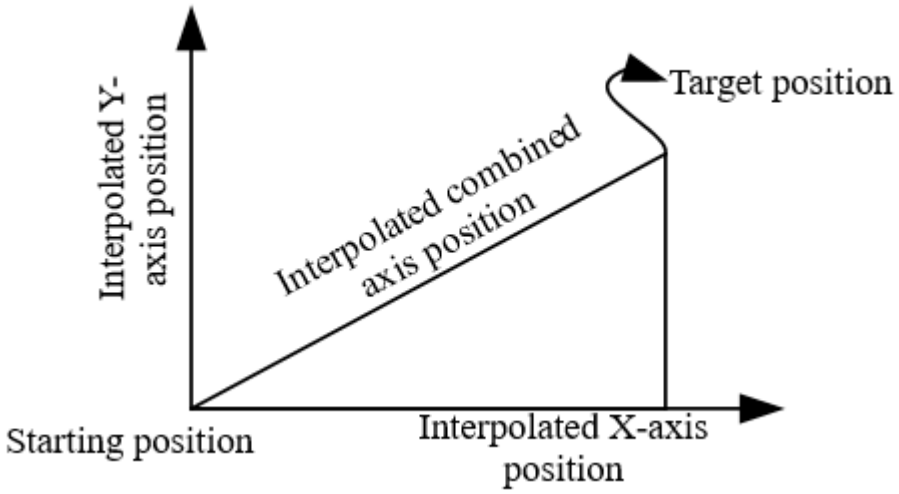
(3) Complete the enable of each axis and axis group.

(4) Call the relative position linear interpolation command to execute the linear interpolation action.

- Basic functions
  - Target location

With the start time of this command as the starting point and the relative position of the two sub-axes set by the command input parameters as the endpoint, the connecting line between the starting point and the ending point is the motion track of the command.

The target position of this command is incremental motion according to the endpoint of the previous motion, meaning each step operates in incremental mode. The target position of the interpolation combined axis is based on the position at the time when the command starts to be executed, following the length of the hypotenuse of a right triangle, with the incremental positions of the two individual axes.



- Input parameter latching

  The rising edge triggering of this command is valid. When executing the rising edge of input, the command latches parameters such as the combined axes and the ID of the two sub-axes from the AxesGroup input/output parameters, , and the input/output parameters—Position, Velocity, Acceleration, Deceleration, and Jerk. During command execution, the modified latched parameters are invalid until another rising edge triggers this command.

- Motion parameter setting

  The joint-axis motion speed of interpolation motion is set by the input parameter Velocity of this command, the acceleration and deceleration are respectively set by the input parameter Acceleration and Deceleration, and the jerk is set by the input parameter Jerk.

  The Sub-axis-axis motion speed is calculated by real-time correlation of the combined axis command speed.

- Limit

  When the interpolation sub-axis encounters a limit, the maximum value of the sub-axis deceleration calculated by the combined axis decomposition and its own fast deceleration is selected for optimal protection stop.

- Handling of repeated triggering of function block

  When a function block command is waiting for repeated triggering, the repeatedly triggered command will be put into the motion buffer. The monitoring state of the function block is the running state of the next command, and the running state of the previous command will not be monitored. During the execution of the previous interpolation command, the next interpolation command starts to be executed after the end of the previous one.

- BufferMode (buffer mode selection)

  Relative position linear interpolation command supports BufferMode (buffer mode selection), which supports 1: Buffered, 2: BlendingLow, 3: BlendingPrevious, 4: BlendingNext and 5: BlendingHigh.

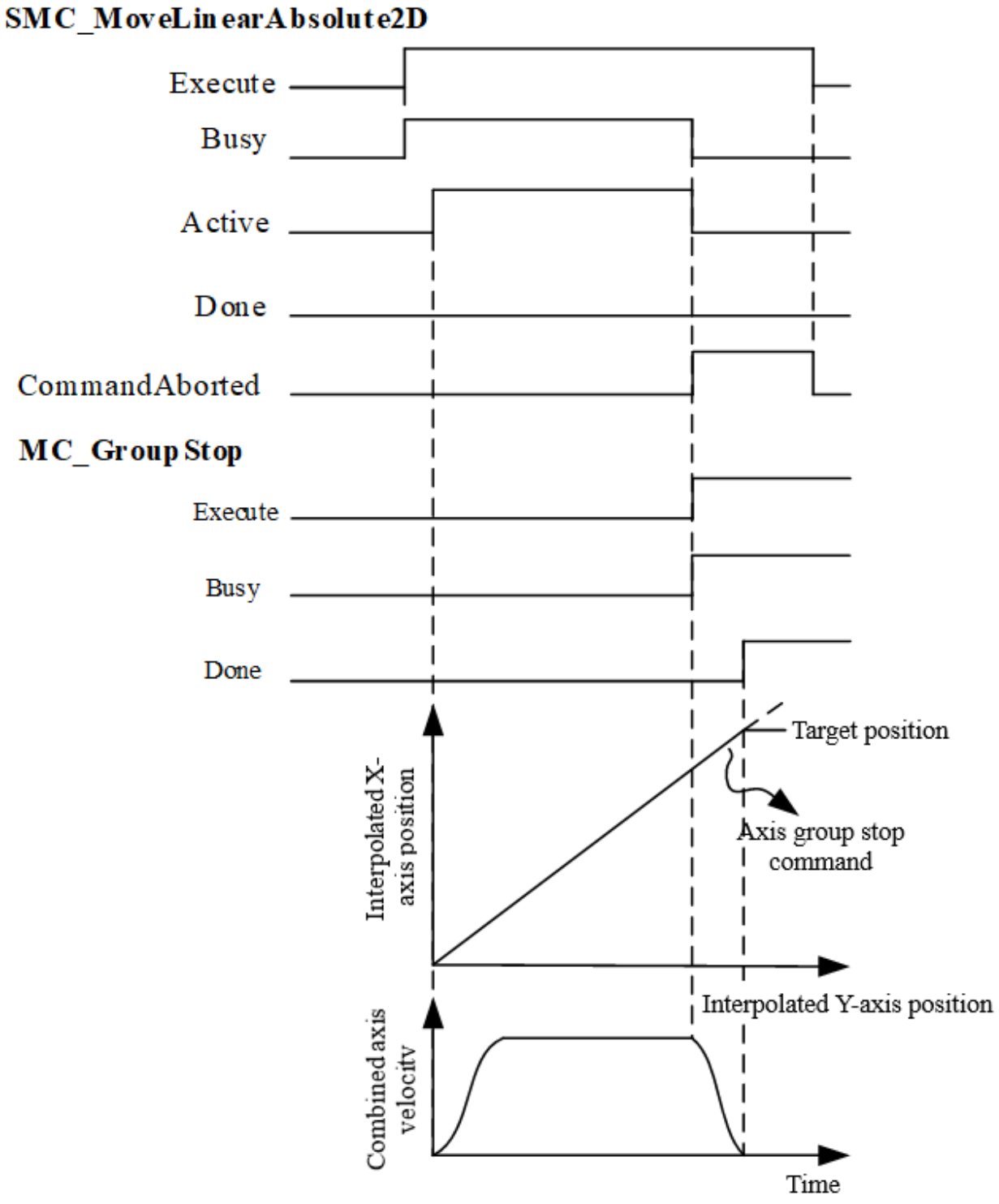
  When this command is sent, if BufferMode is 1:Buffered, the bufferd command will be automatically

started from the cycle when the currently executing command ends normally. If BufferMode is 2:BlendingLow, the adjacent commands are blended at a low speed. If BufferMode is 3: BlendingPrevious, the speed of the previous command will be blended. If BufferMode is 4: BlendingNext, the speed of the following command will be blended. If BufferMode is 5: BlendingHigh, it indicates high-speed fusion between adjacent commands.

For details about the buffer mode of interpolation commands, please refer to Buffer Mode Description in the appendix.

- Command interrupt

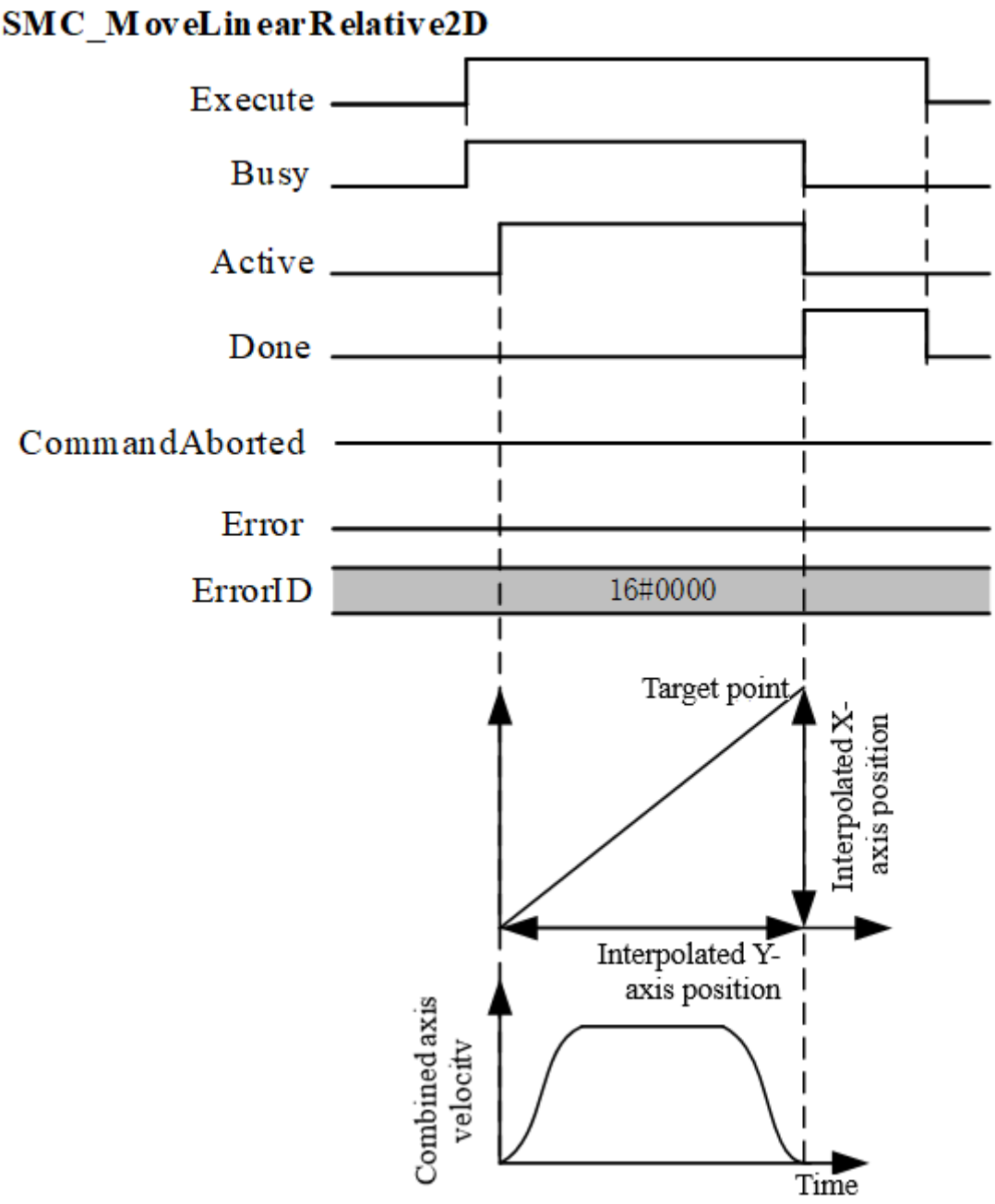
When executing the axis group interpolation command, you can use the MC_GroupStop command to interrupt the interpolation command. Each axis of the axis group decelerates and stops along the interpolation motion track.



- CoordSystem
  - Specify the coordinate system for linear interpolation of relative position.
  - Use the axis coordinate system (ACS).
- Timing diagram

The following is a normal and error timing diagram when the relative position interpolation command is executed.

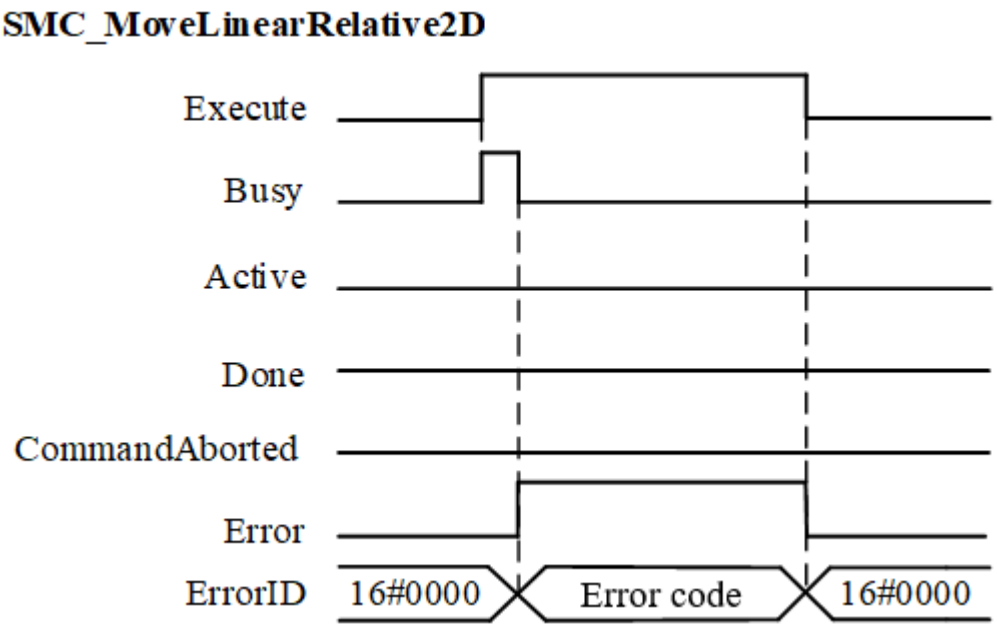
(1) Normal timing diagram



(2) Error pin timing diagram

When this command is executed, if there are abnormalities such as illegal parameters, the axis type of an axis in the axis group being 0: Unused (unused axis), and illegal ID of each axis in the axis group, the call of this command fails, with command output pin Error=True, and the error code ErrorID reports an abnormal error value. Please refer to the appendix Error Code Description to obtain the cause of the error. When the command input pin Execute goes low, all

output pins revert to their Default Values.



- When the abnormal situation of the command is solved, it is necessary to trigger Execute again on the rising edge to send the interpolation command of the axis group and carry out the interpolation action of the relative position of the axis group.

7.5.3 Examples

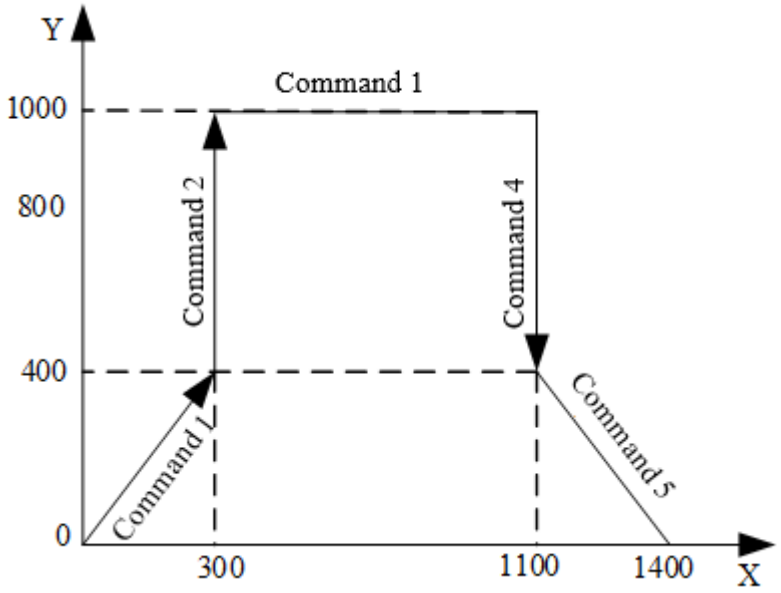
- Action example
  - Action introduction

    In the following example, 5 linear interpolation commands for relative positions are executed and will move to the target position after executing the interpolation command.

    Set the BufferMode selection of interpolation commands as Buffered to buffer multiple motion commands.

    In this example, when the output Active of linear interpolation command 1 becomes TRUE, the remaining interpolation commands 2-5 are executed.

  - Action diagram



Sequentially execute the linear interpolation command 1 to (300, 400) position, the linear interpolation command 2 to (300, 1000) position, the linear interpolation command 3 to (1100, 1000) position, and the linear interpolation command 4 to (1100, 400) position. Execute the linear interpolation command 3 to (1400, 0) position.

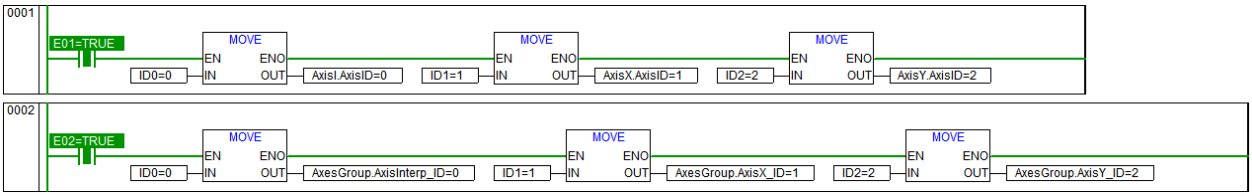
- LD diagram program

Set the speed of each interpolation command to 100, acceleration and deceleration to 200, acceleration to 5000, and buffer mode to Buffered.

Description of Primary Variables

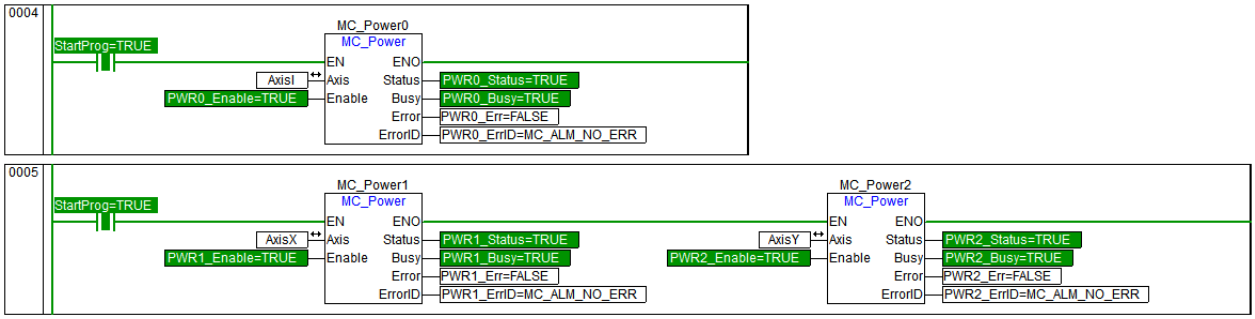
Variable name	Variable type	Initial value	Notes
AxesGroup	AXES_GROUP_REF	-	axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, axes in the axis group can be enabled with ready status.
Power0_Status	BOOL	FALSE	Variables that enable the axis successfully. When the axis closing is enabled successfully, this variable becomes TRUE.
Power1_Status	BOOL	FALSE	Variables that enable the success of sub-axis X. When the sub-axis X is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variables that enable the success of sub-axis Y. When the sub-axis Y is enabled successfully, this variable becomes TRUE.
GrpEnableStatus	BOOL	TRUE	When this variable is TRUE, the axis group is enabled.
GrpEnable1_Exe	BOOL	FALSE	axis group enable rising edge trigger variable.
AxisHomed	BOOL	FALSE	If this variable is TRUE, each axis in the group returns to its homing.
Rel2D1_Exe	BOOL	FALSE	The rising edge trigger variable of the relative interpolation command delivered by the axis group.
Rel2D1_Active	BOOL	FALSE	Variable of the Active pin of the relative interpolation command delivered by the axis group. When this variable becomes TRUE, the axis group executes the interpolation command.

(1) Set the axis IDs of combined and sub axes in the axis group. To define an axis group, firstly, you need to define three axes in the axis group and match the IDs of the defined three axes with those of combined and sub-axes. See Sections 0001 and 0002 below.

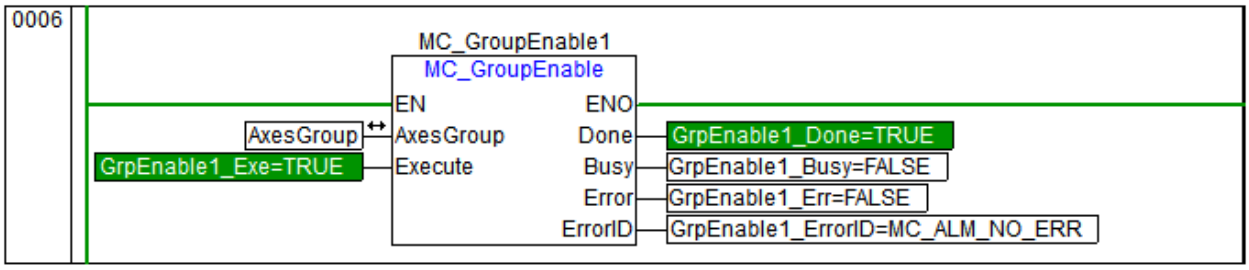


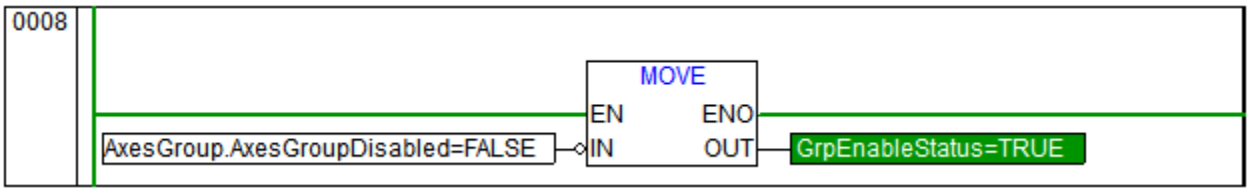
(2) Axis initialization. Call the SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group respectively, and set the combined axis as 0: Virtual and split-axis type as 50: EtherCAT_Position. The usage of SMC_AxisInit command is described in the respective Command Chapter.

(3) Axis enable. After each axis in the axis group is ready (StartProg =TRUE), call MC_Power to enable the combined axes and two sub-axes of the axis group. See Sections 0004 ~ 0005.



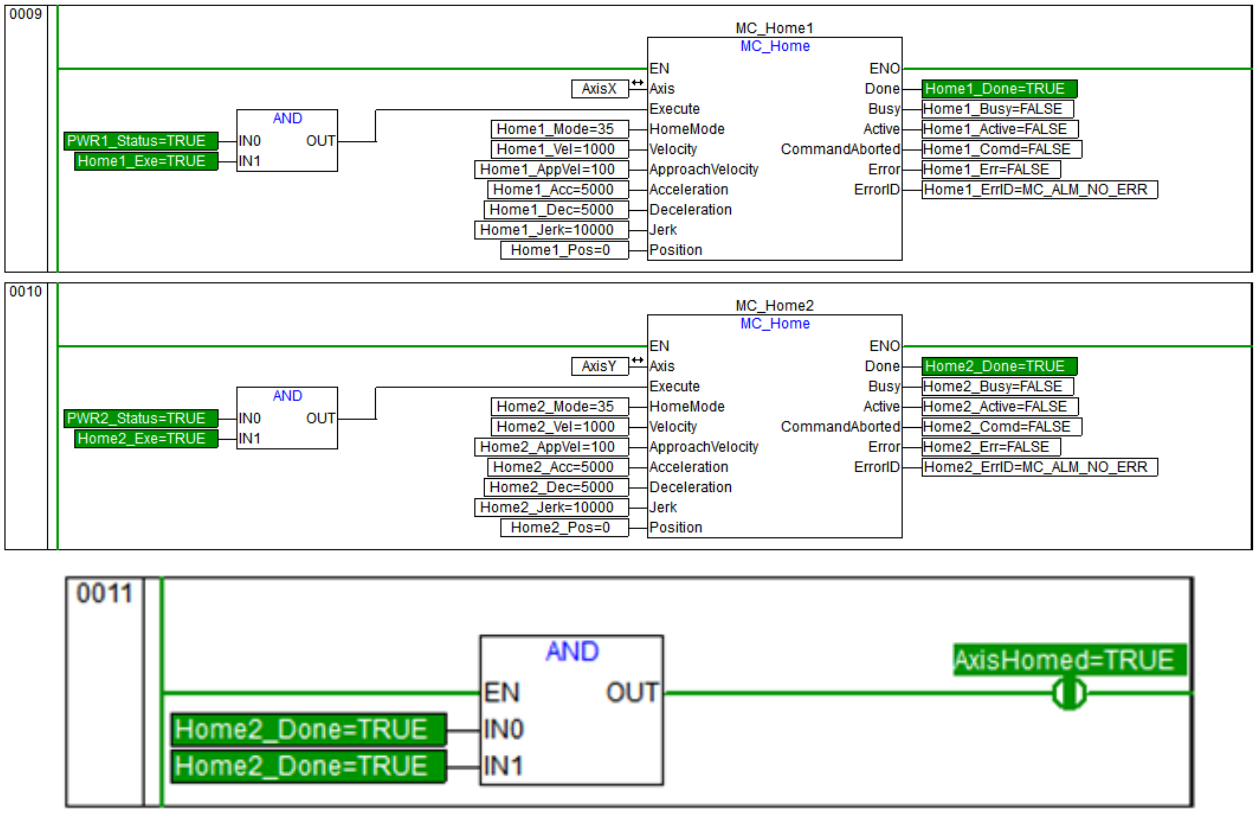
(4) Axis group enable. Call MC_GroupEnable to enable the axis group. AxesGroupDisabled is set to FALSE in the axis group variable. See Section 0006. After the AxesGroupDisabled is enabled, inversely assign AxesGroupDisabled to the GrpEnableStatus variable.



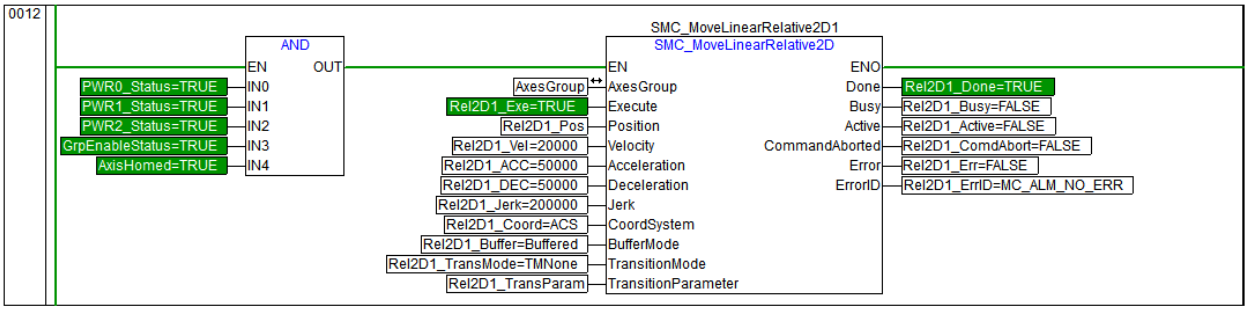


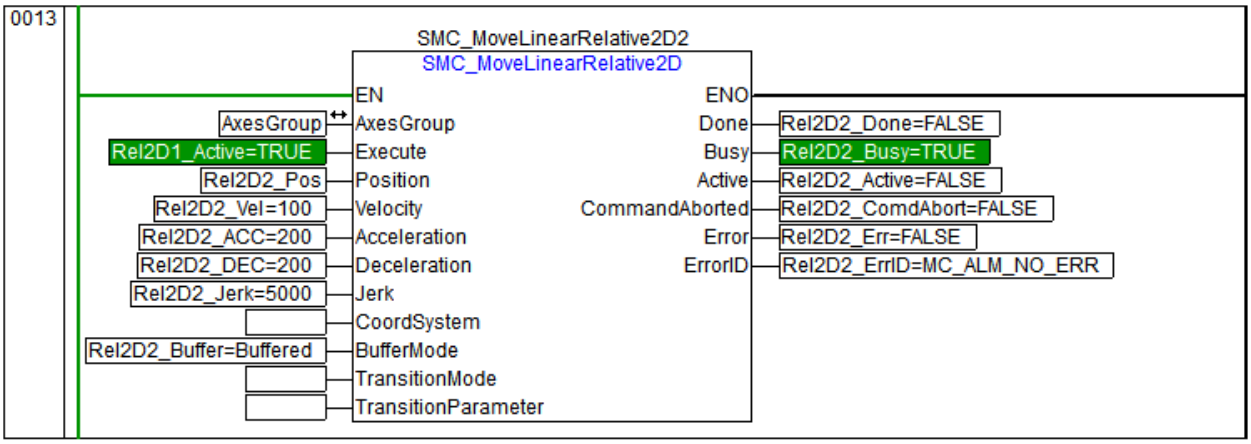
(5) After the two sub-axes in the axis group enable, if the homing is not determined, this executes

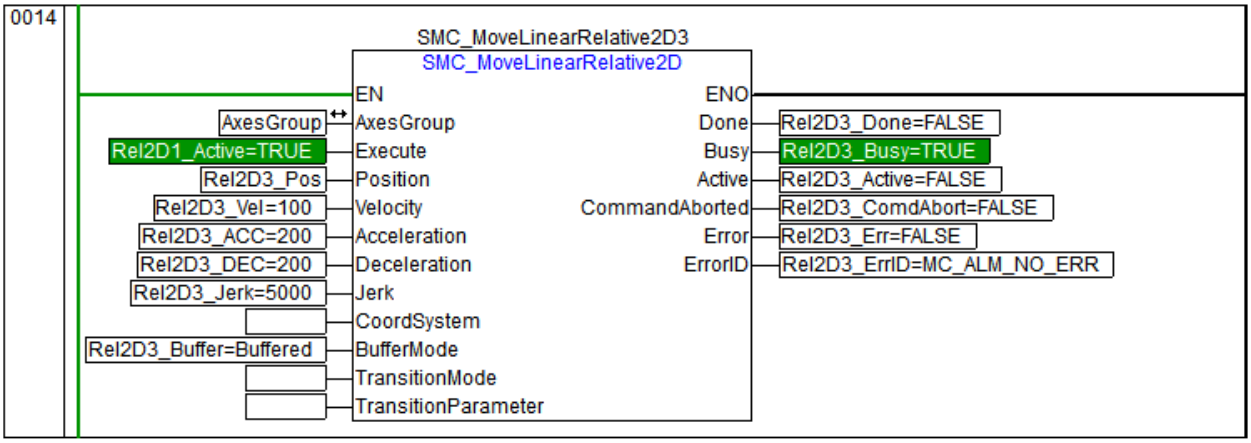
the homing operation. After the homing is completed, set AxisHomed to TRUE. See Section 0009-0011.

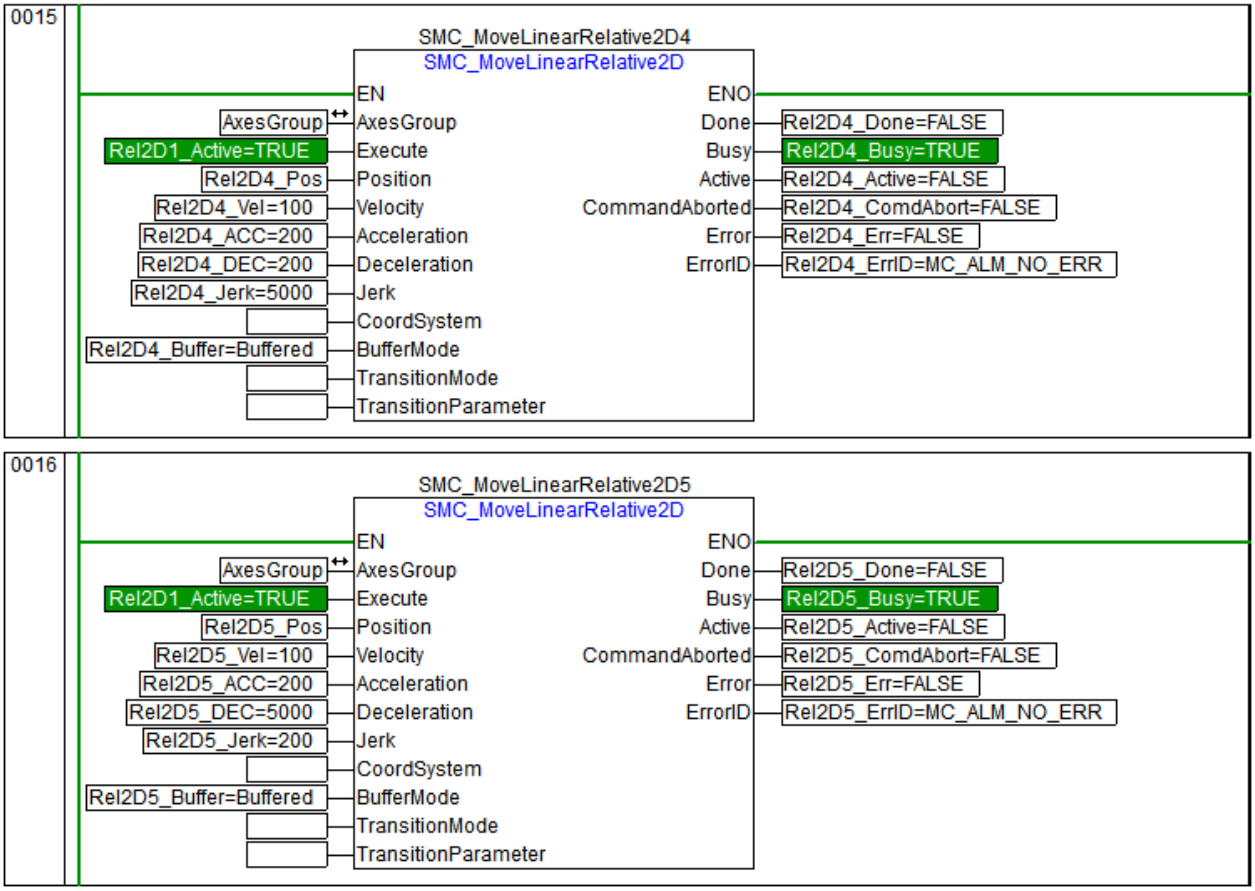


(6) In Section 0011, after the axis homing is completed, the interpolation motion command begin to execute. Execute command 1, and deliver commands 2-5 in sequence when Rel2D1_Active of command 1 is TRUE. See section 0012-0016 below.


0012
AND
EN
OUT
PWR0_Status=TRUE
PWR1_Status=TRUE
PWR2_Status=TRUE
GrpEnableStatus=TRUE
AxisHomed=TRUE
IN0
IN1
IN2
IN3
IN4
SMC_MoveLinearRelative2D1
SMC_MoveLinearRelative2D
AxesGroup
Rel2D1_Exe=TRUE
Rel2D1_Pos
Rel2D1_Vel=20000
Rel2D1_ACC=50000
Rel2D1_DEC=50000
Rel2D1_Jerk=200000
Rel2D1_Coord=ACS
Rel2D1_Buffer=Buffered
Rel2D1_TransMode=TMNone
Rel2D1_TransParam
EN
AxesGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
Rel2D1_Done=TRUE
Rel2D1_Busy=FALSE
Rel2D1_Active=FALSE
Rel2D1_ComdAbort=FALSE
Rel2D1_Err=FALSE
Rel2D1_ErrID=MC_ALM_NO_ERR


0013
SMC_MoveLinearRelative2D2
SMC_MoveLinearRelative2D
AxesGroup
Rel2D1_Active=TRUE
Rel2D2_Pos
Rel2D2_Vel=100
Rel2D2_ACC=200
Rel2D2_DEC=200
Rel2D2_Jerk=5000
Rel2D2_Buffer=Buffered
EN
AxesGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
Rel2D2_Done=FALSE
Rel2D2_Busy=TRUE
Rel2D2_Active=FALSE
Rel2D2_ComdAbort=FALSE
Rel2D2_Err=FALSE
Rel2D2_ErrID=MC_ALM_NO_ERR


0014
SMC_MoveLinearRelative2D3
SMC_MoveLinearRelative2D
AxesGroup
Rel2D1_Active=TRUE
Rel2D3_Pos
Rel2D3_Vel=100
Rel2D3_ACC=200
Rel2D3_DEC=200
Rel2D3_Jerk=5000
Rel2D3_Buffer=Buffered
EN
AxesGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
Rel2D3_Done=FALSE
Rel2D3_Busy=TRUE
Rel2D3_Active=FALSE
Rel2D3_ComdAbort=FALSE
Rel2D3_Err=FALSE
Rel2D3_ErrID=MC_ALM_NO_ERR



■ ST language program

Description of Primary Variables

Variable name	Variable type	Initial value	Notes
AxesGroup	AXES_GROUP_REF	-	Axie group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, axes in the axis group can be enabled with ready status.
Power0_Status	BOOL	FALSE	Variables that enable the axis successfully. When the axis closing is enabled successfully, this variable becomes TRUE.
Power1_Status	BOOL	FALSE	Variables that enable the success of sub-axis X. When the sub-axis X is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variables that enable the success of sub-axis Y. When the sub-axis Y is enabled successfully, this variable becomes TRUE.
GrpEnableStatus	BOOL	TRUE	When this variable is TRUE, the axis group is enabled.
GrpEnable1_Exe	BOOL	FALSE	axis group enable rising edge trigger variable.
AxisHomed	BOOL	FALSE	If this variable is TRUE, each axis in the group returns to homing.
InitFlag	BOOL	FALSE	If this variable is TRUE, the initialization parameter setting is completed.
Rel2D1_Exe	BOOL	FALSE	The rising edge trigger variable of the relative interpolation command delivered by the axis group.
Rel2D1_Active	BOOL	FALSE	Variable of the Active pin of the relative interpolation command

			delivered by the axis group. When this variable becomes TRUE, the axis group executes the interpolation command.

(1) Set the axis IDs of combined and split axes in the axis group. Set the defined axis IDs of Axis I, Axis X and Axis Y.

```
IF Switch1 THEN
AxisI.AxisID:=0;
AxisX.AxisID:=1;
AxisY.AxisID:=2;
AxesGroup.AxisInterp_ID:=AxisI.AxisID;
AxesGroup.AxisX_ID:=AxisX.AxisID;
AxesGroup.AxisY_ID:=AxisY.AxisID;
END_IF
```

(2) Set command motion parameters.

```
IF FALSE = InitFlag THEN
(* Parameter setting of relative position linear interpolation command 1. Default values are used for input parameters that have not been set.
Rel2D1_Pos[0]:=300;
Rel2D1_Pos[1]:=400;
Rel2D1_Vel:=100;
Rel2D1_ACC:=200;
Rel2D1_DEC:=200;
Rel2D1_Jerk:=5000;
Rel2D1_Buffer:=Buffered;
(* Parameter setting of relative position linear interpolation command 2. Default values are used for input parameters that have not been set.
Rel2D2_Pos[0]:= 0;
Rel2D2_Pos[1]:=400;
Rel2D2_Vel:=100;
Rel2D2_ACC:=200;
Rel2D2_DEC:=200;
Rel2D2_Jerk:=5000;
Rel2D2_Buffer:=Buffered;
(* Parameter setting of relative position linear interpolation command 3. Default values are used for input parameters that have not been set.
Rel2D3_Pos[0]:=300;
Rel2D3_Pos[1]:=400;
Rel2D3_Vel:=100;
Rel2D3_ACC:=200;
Rel2D3_DEC:=200;
Rel2D3_Jerk:=5000;
Rel2D3_Buffer:=Buffered;
(* Parameter setting of relative position linear interpolation command 4. Default values are used for input parameters that have not been set.
Rel2D4_Pos[0]:= 0;
Rel2D4_Pos[1]:=400;
```

```
Rel2D4_Vel:=100;
Rel2D4_ACC:=200;
Rel2D4_DEC:=200;
Rel2D4_Jerk:=5000;
Rel2D4_Buffer:=Buffered;
(* Parameter setting of linear interpolation command 5 for relative position. Default values are used for input parameters that have not been set.
Rel2D5_Pos[0]:=300;
Rel2D5_Pos[1]:=400;
Rel2D5_Vel:=100;
Rel2D5_ACC:=200;
Rel2D5_DEC:=200;
Rel2D5_Jerk:=5000;
Rel2D5_Buffer:=Buffered;
InitFlag:=TRUE;
END_IF
```

(3) Call the SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group. The axis type of the axis combination is set to 0: Virtual; the axis types of AxisX/AxisY are both set to 50: EtherCAT_Position. The SMC_AxisInit command is described in the commands section.

(4) Execute the MC_Power command to complete the axis enable of three axes in the axis group.

```
IF StartPro THEN
MC_Power0 (*Combined axis enable *)
Enable := PWR0_Enable,
Axis := AxisI,
Status=> PWR0_Status,
Busy=> PWR0_Busy,
Error=> PWR0_Err,
ErrorID=> PWR0_ErrID);
MC_Power1 (*Sub-axis AxisX enable *)
Enable := PWR1_Enable,
Axis := AxisX,
Status=> PWR1_Status,
Busy=> PWR1_Busy,
Error=> PWR1_Err,
ErrorID=> PWR1_ErrID);
MC_Power2 (*Sub-axis AxisY Enable *)
Enable := PWR2_Enable,
Axis := AxisY,
Status=> PWR2_Status,
Busy=> PWR2_Busy,
Error=> PWR2_Err,
ErrorID=> PWR2_ErrID);
END_IF
```

(5) Enable axis group

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
GrpEnableStatus:= AxesGroup. AxesGroupDisabled;
```

(6) Execute the homing command of the split axis in the axis group to determine the home. See its command section for how to use the MC_Home command.

```
IF PWR1_Status THEN
MC_Home1(
Execute := Home1_Exe,
HomeMode := Home1_Mode,
Velocity := Home1_Vel,
ApproachVelocity := Home1_AppVel,
Acceleration := Home1_Acc,
Deceleration := Home1_Dec,
Jerk := Home1_Jerk,
Position := Home1_Pos,
Axis := AxisX,
Done=> Home1_Done,
Busy=> Home1_Busy,
Active=> Home1_Active,
CommandAborted=> Home1_Comd,
Error=> Home1_Err,
ErrorID=> Home1_ErrID);
MC_Home2(
Execute := Home2_Exe,
HomeMode := Home2_Mode,
Velocity := Home2_Vel,
ApproachVelocity := Home2_AppVel,
Acceleration := Home2_Acc,
Deceleration := Home2_Dec,
Jerk := Home2_Jerk,
Position := Home2_Pos,
Axis := AxisY,
Done=> Home2_Done,
Busy=> Home2_Busy,
Active=> Home2_Active,
CommandAborted=> Home2_Comd,
Error=> Home2_Err,
ErrorID=> Home2_ErrID);
END_IF
```

```
IF Home1_Done AND Home2_Done THEN
AxisHomed:=TRUE;
END_IF
```

(7) After the axis homing is completed, start executing the interpolation motion command. Execute command 1, and deliver commands 2-5 in sequence when Rel2D1_Active of command 1 is TRUE.

```
IF PWR0_Status AND PWR1_Status AND PWR2_Status AND GrpEnableStatus AND AxisHomed THEN
SMC_MoveLinearRelative2D1 (*interpolation command 1*)
Execute := Rel2D1_Exe,
Position := Rel2D1_Pos,
Velocity := Rel2D1_Vel,
Acceleration := Rel2D1_ACC,
Deceleration := Rel2D1_DEC,
Jerk := Rel2D1_Jerk,
CoordSystem := Rel2D1_Coord,
BufferMode := Rel2D1_Buffer,
TransitionMode := Rel2D1_Trans,
TransitionParameter := Rel2D1_Param,
AxesGroup := AxesGroup,
Done=> Rel2D1_Done,
Busy=> Rel2D1_Busy,
Active=> Rel2D1_Active,
CommandAborted=> Rel2D1_ComdAbort,
Error=> Rel2D1_Err,
ErrorID=> Rel2D1_ErrID);
SMC_MoveLinearRelative2D2 (*interpolation command 2*)
Execute := Rel2D1_Active,
Position := Rel2D2_Pos,
Velocity := Rel2D2_Vel,
Acceleration := Rel2D2_ACC,
Deceleration := Rel2D2_DEC,
Jerk := Rel2D2_Jerk,
CoordSystem := Rel2D2_Coord,
BufferMode := Rel2D2_Buffer,
TransitionMode := Rel2D2_Trans,
TransitionParameter := Rel2D2_Param,
AxesGroup := AxesGroup,
Done=> Rel2D2_Done,
Busy=> Rel2D2_Busy,
Active=> Rel2D2_Active,
CommandAborted=> Rel2D2_ComdAbort,
Error=> Rel2D2_Err,
ErrorID=> Rel2D2_ErrID);
SMC_MoveLinearRelative2D3 (*interpolation command 3*)
Execute := Rel2D1_Active,
```

```
Position := Rel2D3_Pos,
Velocity := Rel2D3_Vel,
Acceleration := Rel2D3_ACC,
Deceleration := Rel2D3_DEC,
Jerk := Rel2D3_Jerk,
CoordSystem := Rel2D3_Coord,
BufferMode := Rel2D3_Buffer,
TransitionMode := Rel2D3_Trans,
TransitionParameter := Rel2D3_Param,
AxesGroup := AxesGroup,
Done=> Rel2D3_Done,
Busy=> Rel2D3_Busy,
Active=> Rel2D3_Active,
CommandAborted=> Rel2D3_ComdAbort,
Error=> Rel2D3_Err,
ErrorID=> Rel2D3_ErrID);
SMC_MoveLinearRelative2D4 (*interpolation command 4*)
Execute := Rel2D1_Active,
Position := Rel2D4_Pos,
Velocity := Rel2D4_Vel,
Acceleration := Rel2D4_ACC,
Deceleration := Rel2D4_DEC,
Jerk := Rel2D4_Jerk,
CoordSystem := Rel2D4_Coord,
BufferMode := Rel2D4_Buffer,
TransitionMode := Rel2D4_Trans,
TransitionParameter := Rel2D4_Param,
AxesGroup := AxesGroup,
Done=> Rel2D4_Done,
Busy=> Rel2D4_Busy,
Active=> Rel2D4_Active,
CommandAborted=> Rel2D4_ComdAbort,
Error=> Rel2D4_Err,
ErrorID=> Rel2D4_ErrID);
SMC_MoveLinearRelative2D5 (*interpolation command 5*)
Execute := Rel2D1_Active,
Position := Rel2D5_Pos,
Velocity := Rel2D5_Vel,
Acceleration := Rel2D5_ACC,
Deceleration := Rel2D5_DEC,
Jerk := Rel2D5_Jerk,
CoordSystem := Rel2D5_Coord,
BufferMode := Rel2D5_Buffer,
TransitionMode := Rel2D5_Trans,
TransitionParameter := Rel2D5_Param,
```

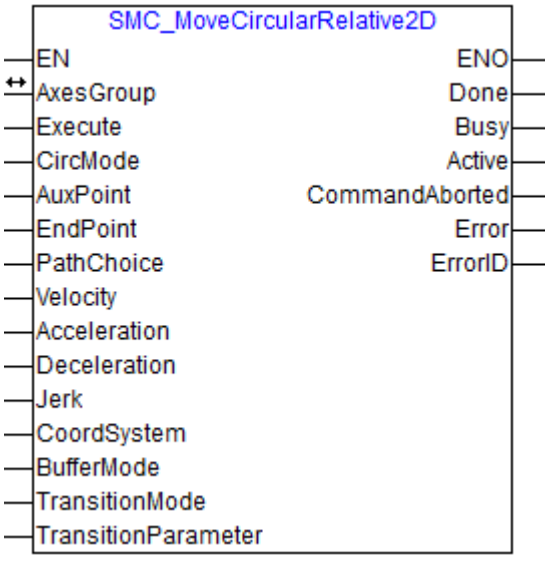
```
AxesGroup := AxesGroup,
Done=> Rel2D5_Done,
Busy=> Rel2D5_Busy,
Active=> Rel2D5_Active,
CommandAborted=> Rel2D5_ComdAbort,
Error=> Rel2D5_Err,
ErrorID=> Rel2D5_ErrID);
END_IF
```

7.5.4 Precautions

- When calling this command, the input parameter Positon cannot be set to 0.
- This command BufferMode does not support 0: Aborting.

7.6 SMC_MoveCircularRelative2D (Relative Circular Interpolation on Two Axes)

This command realizes 2 axes realtive position circular interpolation.



7.6.1 Parameter

Parameter Type	Title	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	AxesGroup	Axis group name	No	-	-	AXES_GROUP_REF
INPUT	Execute	Rising edge	No	-	TRUE/FALSE	BOOL

		trigger				
	CircMode	Mode selection 0: Three o'clock 1: center/end point	YES	0	0:BORDER 1:CENTER	MC_CIRC_MODE
	AuxPoint	When CircMode is BORDER, AuxPoint is the intermediate point position; When CircMode is CENTER, AuxPoint is the position of center point.	No	-	Positive/negative/0	ARRAY[0..1] OF LREAL
	EndPoint	End position	No	-	Positive/negative/0	ARRAY[0..1] OF LREAL
	PathChoice	Direction of motion (valid when CircMode is CENTER) 0: clockwise 1: counterclockwise	YES	0	0:CLOCKWISE 1:COUNTERCLOCKWISE	MC_CIRC_MODE
	Velocity	Target speed	No	-	Non-negative	LREAL
	Acceleration	Acceleration	No	-	Positive value	LREAL
	Deceleration	Deceleration	No	-	Positive value	LREAL
	Jerk	Jerk 0: Trapezoidal acceleration and deceleration Positive: S-shaped acceleration and deceleration	YES	0	0/positive	LREAL
	CoordSystem	Coordinate system 0: Axis coordinate system	YES	0	0: ACS	MC_COORD_SYSTEM
	BufferMode	Buffer mode 1: Buffer 2: low-speed fusion between adjacent commands 3: Speed fusion of the previous command 4: Speed fusion of the following command 5: High-speed fusion between adjacent commands	YES	1	1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	MC_BUFFER_MODE
	TransitionMode	Transition Curve Selection 0: no transition curve inserted	YES	0	0:TMNone	MC_TRANSITION_MODE
	TransitionParameter	Retained	YES	0	-	ARRAY[0..1] OF LREAL

		parameters				
OUTPUT	Done	Interpolation motion finished	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Active	In control, TRUE when the command is executed	YES	FALSE	TRUE/FALSE	BOOL
	CommandAborted	Termination of Execution	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

7.6.2 Functions

This command realizes 2-axis absolute position linear interpolation.

- Interpolation steps

(1) Determine the interpolated axis group. The AxesGroup variable type is AXES_GROUP_REF.

(2) Specify the axis number in the axis group variable, corresponding to the axis number of the combined axis and sub-axis.

(3) Complete the enable of each axis and axis group. Then the circular interpolation command can be used.

- Basic functions
  - Arc command position

    The 2-axis circular interpolation is executed with the 2D plane, and the target position of the interpolated combined axis increases the arc length of the command track based on the position at the time when the command starts to be executed.

    The position specified by the circular interpolation command is based on the designation of "relative value positioning", that is, the command positions AuxPoint and EndPoint of the circular interpolation command are calculated incrementally with the starting time position of the command as the home.

  - Input parameter latch

    The rising edge triggering of this command is valid. When executing the rising edge of the input, latch the IDs of the combined axis and two sub-axes of the input and output parameters AxesGroup, the input parameters Position, Velocity, Acceleration, Deceleration, BufferMode and other parameter values. During the execution of the command, modifying the latched parameters is invalid until another rising edge triggers this command.

  - Motion parameter setting

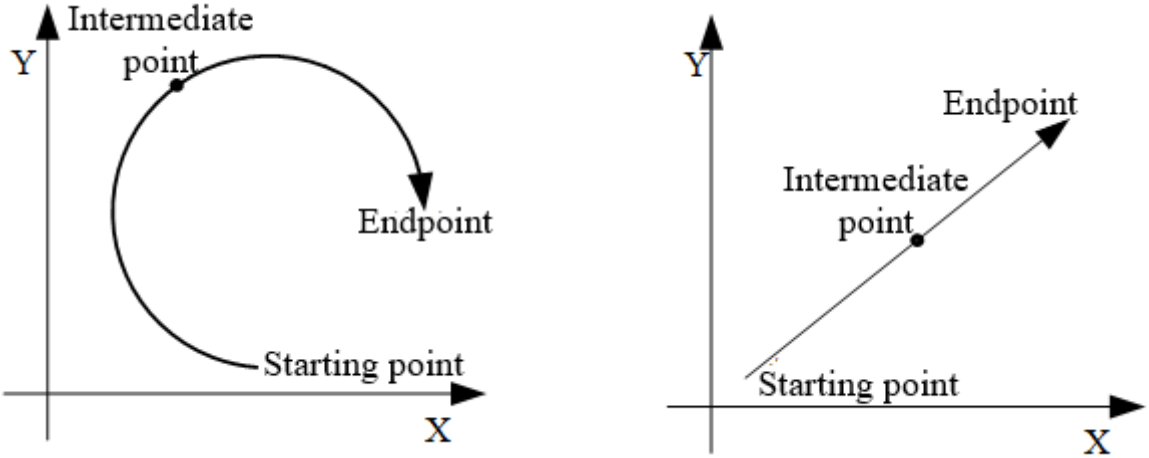
    The interpolation velocity, acceleration, deceleration and jerk of circular interpolation are specified by the command input parameters Velocity, Acceleration, Deceleration and Jerk.

■ Circular interpolation mode (CircMode)

There are two circular interpolation modes: three-point mode and center/endpoint mode, which can be specified by CircMode. When CircMode is BORDER, AuxPoint is the middle point; when CircMode is CENTER, AuxPoint is the center point.

(a) Three-point mode (CircMode=BORDER)

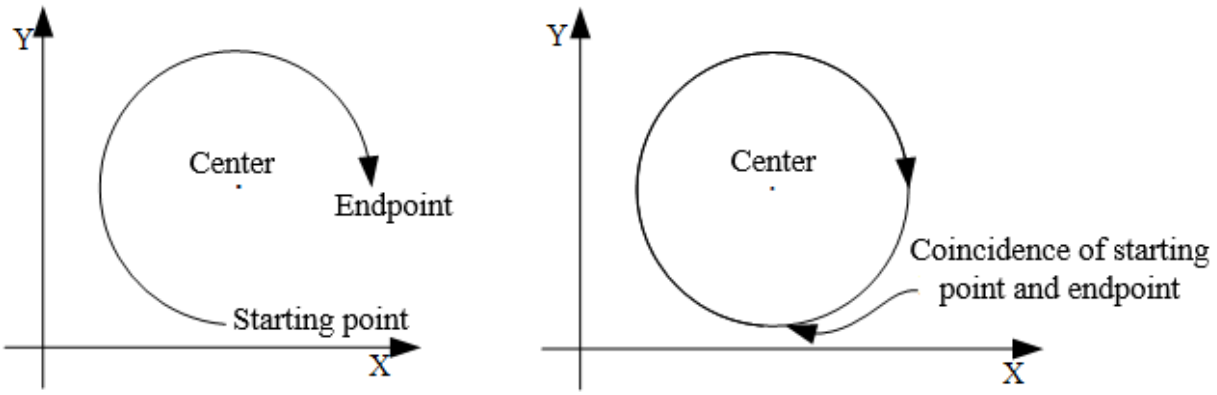
Starting from the current position, perform circular interpolation that connects the middle point AuxPoint(X, Y) and the endpoint EndPoint(X, Y). Linear interpolation from the starting point to the endpoint is performed when thoes points are on the same line, or the middle point and endpoint are at the same point, and the starting point and middle point are at the same point.



(b) Circle mode/endpoint mode (CircMode=CENTER)

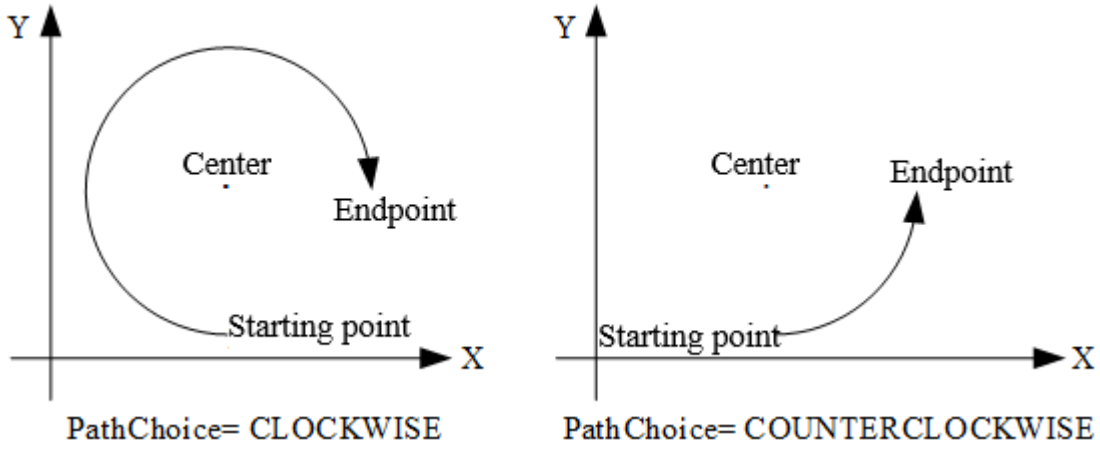
Starting from the current position, perform circular interpolation by connecting the EndPoint(X, Y) with the arc specified by the center position AuxPoint(X, Y). The direction of rotation of the circular arc is specified with PathChoice. Draw a full circle when the starting point and endpoint are the same.

When the radius from the specified center position to the starting point is different from that to the end point, circular interpolation will be performed by using the radius from the specified center position to the starting point.



- Motion direction of circular interpolation (PathChoice)

  This function is effective when CircMode is CENTER, and the direction of motion PathChoice is 0: CLOCKWISE and 1: COUNTERCLOCKWISE. The following figure shows the interpolation trajectory where AuxPoint and EndPoint of two interpolated circular arcs are consistent, but the direction of circular interpolation motion is opposite.



- Handling of repeated triggering of function block

  When a function block command is waiting for repeated triggering, the repeatedly triggered command will be put into the motion buffer. The monitoring status of the function block is the running state of the next command, and the running state of the previous command will not be monitored. During the execution of the previous interpolation command, the next interpolation command starts to be executed after the end of the previous one.

- BufferMode (buffer mode selection)

  The 2-axis circular interpolation command supports BufferMode, including 1: Buffered, 2: BlendingLow, 3: BlendingPrevious, 4: BlendingNext and 5: BlendingHigh.

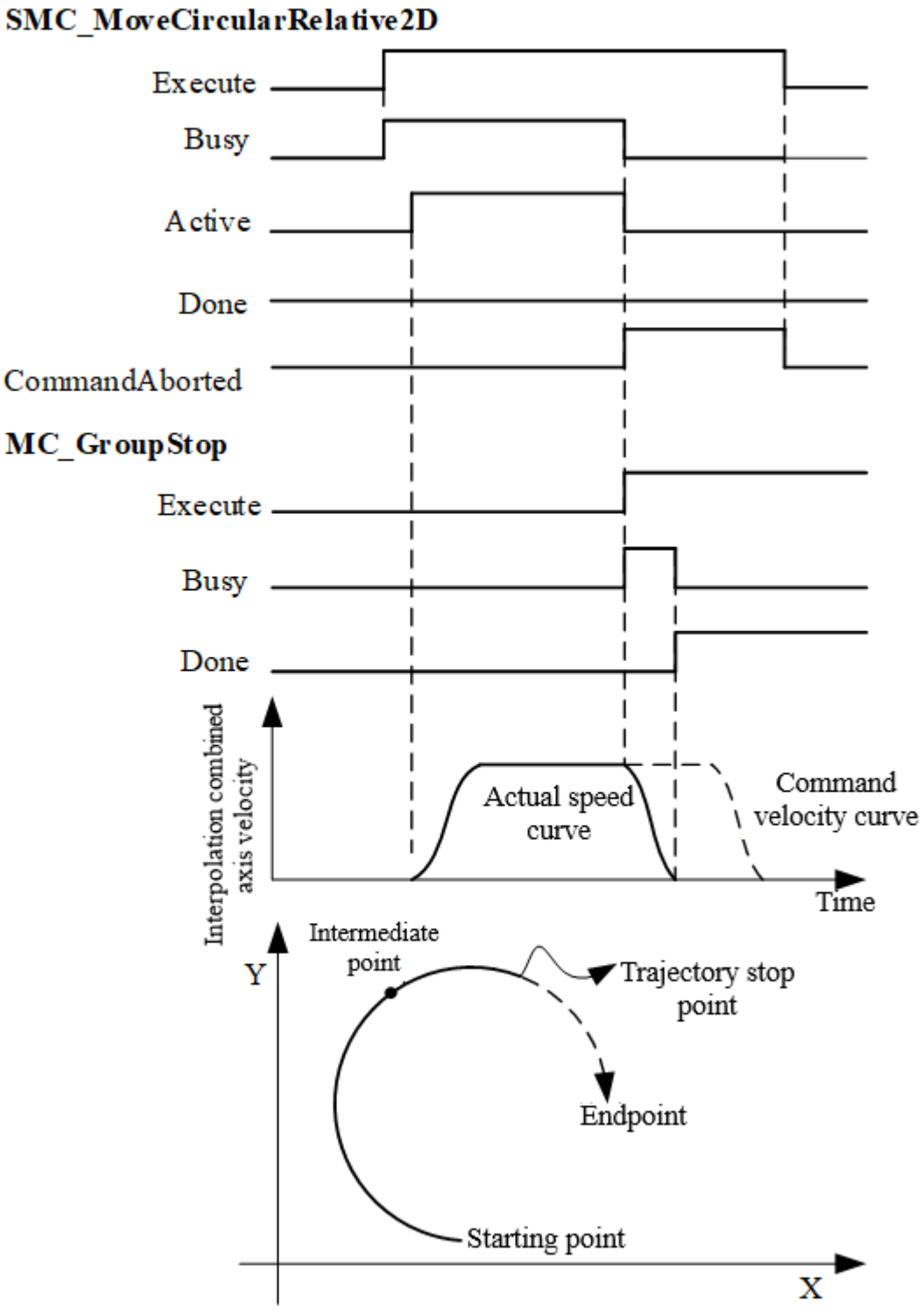
  When this command is delivered, if BufferMode is 1:Buffered, the bufferd command will be automatically started from the cycle when the currently executing command ends normally. If BufferMode is 2:BlendingLow, the adjacent commands are blended at a low speed. If BufferMode is 3: BlendingPrevious, the speed of the previous command will be blended. If BufferMode is 4:

BlendingNext, the speed of the following command will be blended. If BufferMode is 5: BlendingHigh, it indicates high-speed fusion between adjacent commands.

For details about the buffer mode of interpolation commands, please refer to Buffer Mode Description in the appendix.

- Command stop

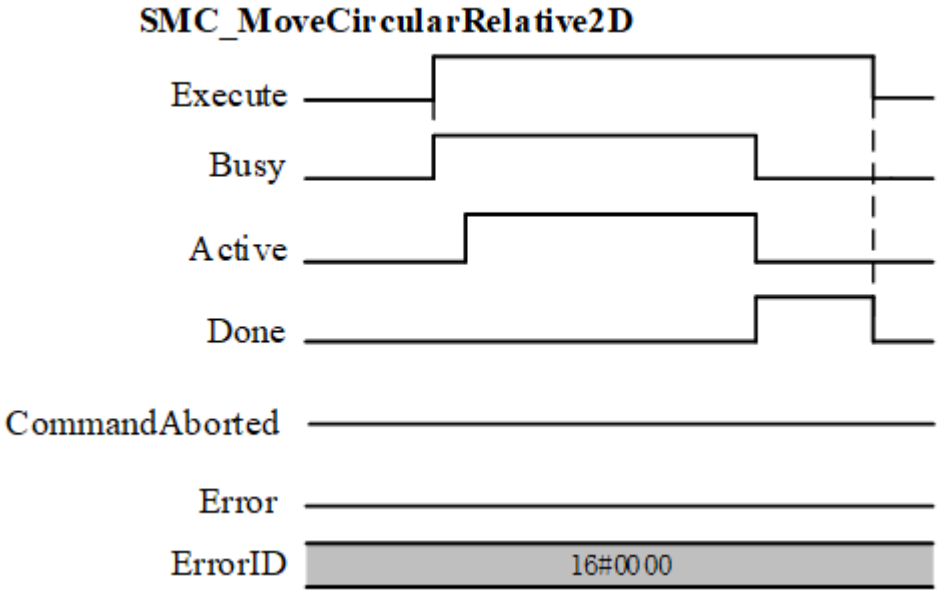
The MC_GroupStop command can be used to interrupt the interpolation command during the execution of the axis group circular interpolation command. Each axis of the axis group decelerates and stops along the arc interpolation motion track.



- CoordSystem
  - Specify the coordinate system for circular interpolation.
  - Use the axis coordinate system (ACS).
- Timing diagram

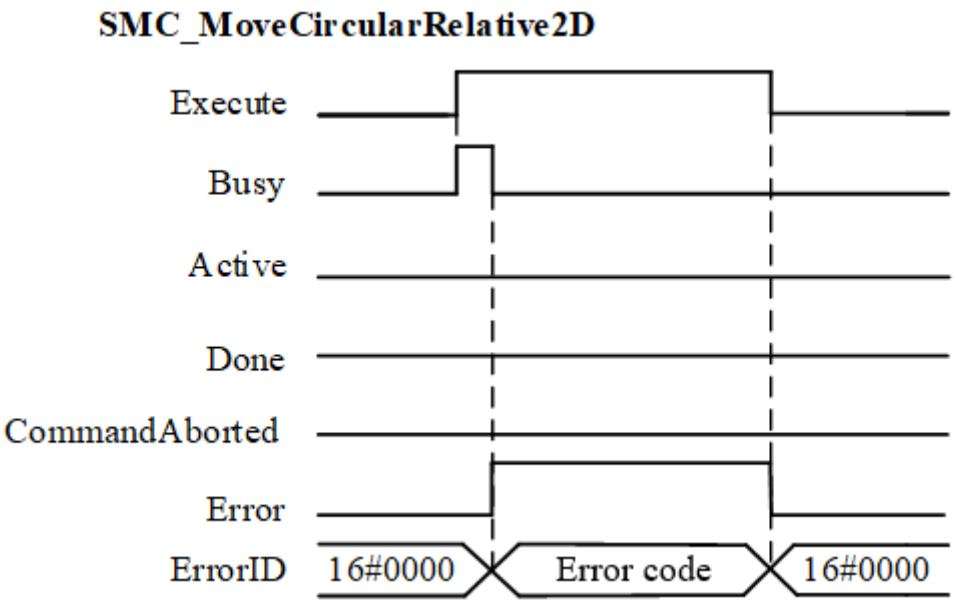
The following is a normal and error timing diagram when the circular interpolation command is executed.

(1) Normal timing diagram



(2) Error pin timing diagram

During the execution of this command, if there are errors such as illegal parameters, 0: Unused (unused axis) for an axis in the axis group, and illegal ID of each axis in the axis group, the call of this command fails, with command output pin Error=True, and the error code ErrorID reports an abnormal error value. Please refer to the appendix Error Code Description to obtain the cause of the error. When the command input pin Execute goes low, all output pins revert to their Default Values.



- After the error of the command is solved, it is necessary to trigger Execute again on the rising edge to send the axis group interpolation command and carry out the absolute position interpolation motion of the axis group.

7.6.3 Examples

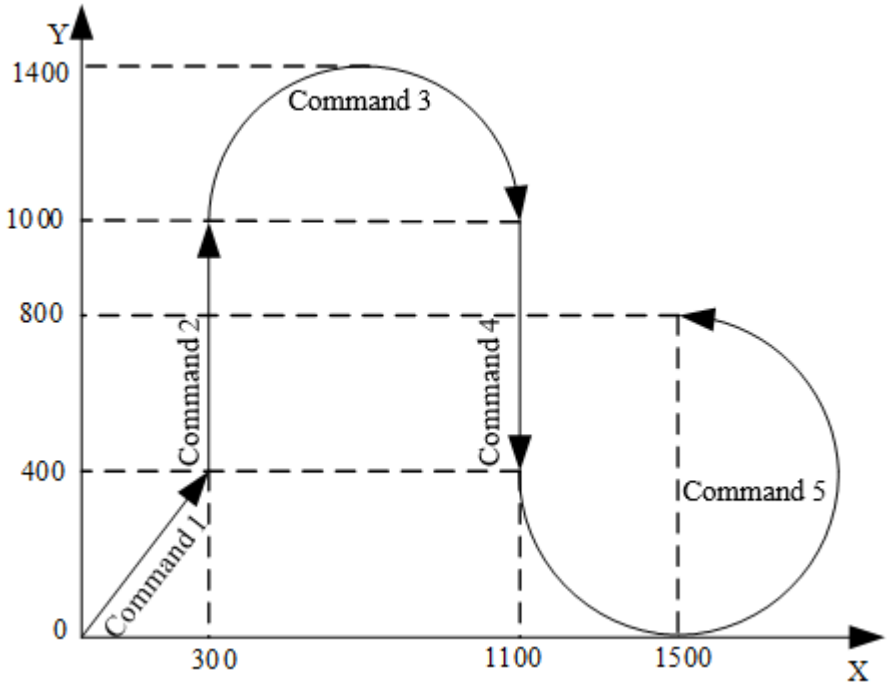
- Action example
  - Action introduction

    The following example includes a linear interpolation command and a circular interpolation command. After the interpolation command is executed, it will move to the target position.

    Set BufferMode of the interpolation command as Buffered to buffer multiple motion commands.

    In this example, when the output Active of the linear interpolation command 1 becomes TRUE, the remaining interpolation commands 2-5 are executed.

  - Action diagram



The linear interpolation command 1 is executed to the position (300, 400), the linear interpolation command 2 is executed to the position (300, 1000), the circular interpolation command 3 is executed to the position (1100, 1000), the linear interpolation command 4 is executed to the position (1100, 400), and the circular interpolation command 5 is executed to the position (1500, 800) in sequence.

- LD diagram program

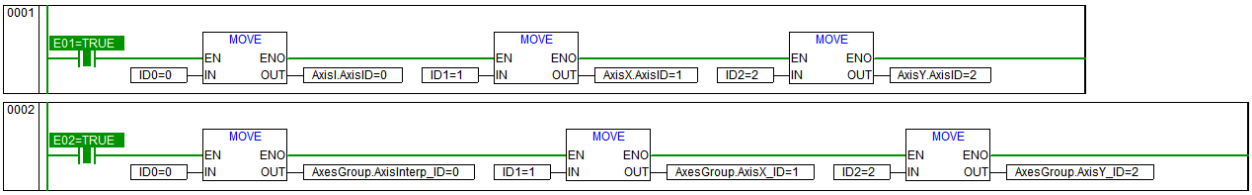
The speed of each interpolation command is set to 100, the acceleration and deceleration are all 200, the jerk is 5000, and the buffer mode is Buffered.

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enabling operation.
Power1_Status	BOOL	FALSE	Variable that interpolates the sub-axis X-axis in the axis group and enables successfully. When the X-axis is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variable whose interpolation sub-axis and Y-axis are enabled successfully in the axis group. When the Y axis is enabled successfully, this variable becomes TRUE.
Power0_Status	BOOL	FALSE	Variable for successful interpolation of combined axis enable in the axis group. When the interpolation combined axis is enabled successfully, this variable becomes TRUE.
GrpEnableStatus	BOOL	TRUE	When this variable is TRUE, the axis group is enabled.
GrpEnable1_Exe	BOOL	FALSE	Axis group enable rising edge trigger variable.

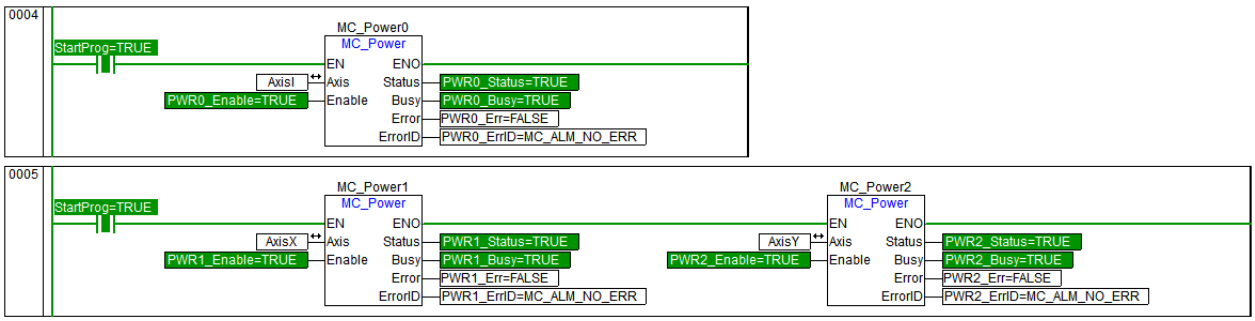
AxisHomed	BOOL	FALSE	If this variable is TRUE, the return to the homing of each axis in the axis group is completed.
Rel2D1_Exe	BOOL	FALSE	The rising edge trigger variable of the circular interpolation command sent by the axis group.
Rel2D1_Active	BOOL	FALSE	The variable of the Active pin of the circular interpolation command sent by the axis group. When this variable becomes TRUE, the axis group executes the interpolation command.

(1) Set the IDs of combined axis and sub-axis in the axis group. When defining an axis group, you need to define three axes in the axis group first, and match the IDs of the defined axis numbers of the three axes with those of the combined axis and sub-axis in the axis group. See sections 0001 and 0002 below.

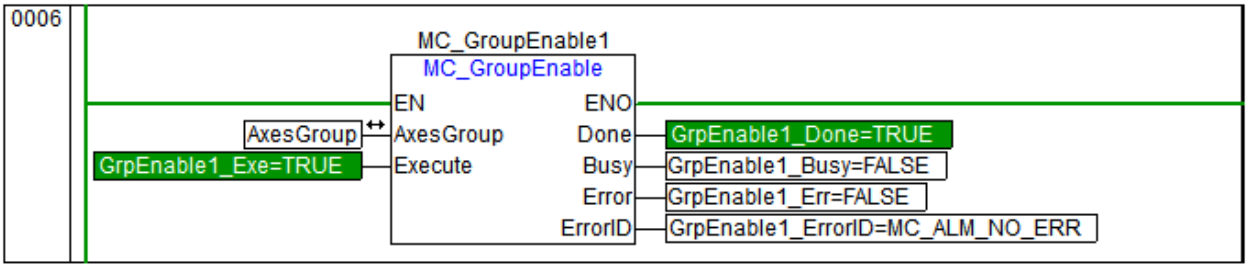


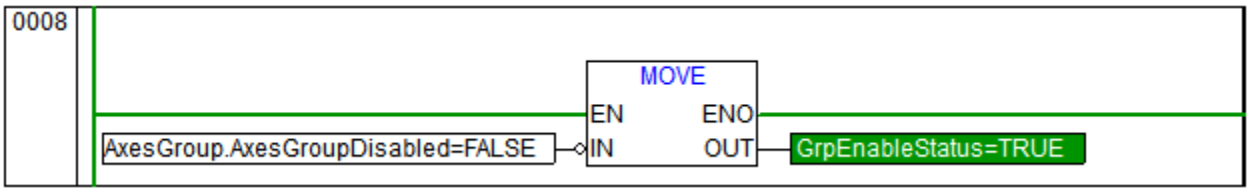
(2) Axis initialization. Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group respectively, and set the combined axis as 0: Virtual, and the sub-axis type as 50: EtherCAT_Position. Please refer to relevant sections for the usage of SMC_AxisInit command.

(3) Axis enable. After each axis in the axis group is ready (StartProg =TRUE), call MC_Power to enable the axes of the combined axis and two sub-axes of the axis group. See Sections 0004 to 0005.

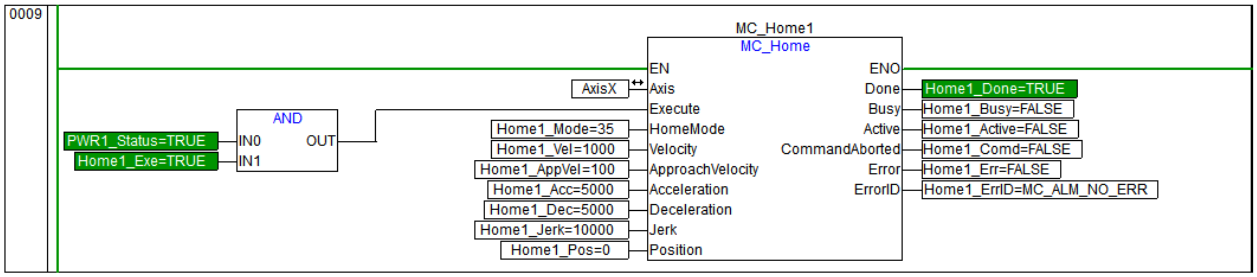


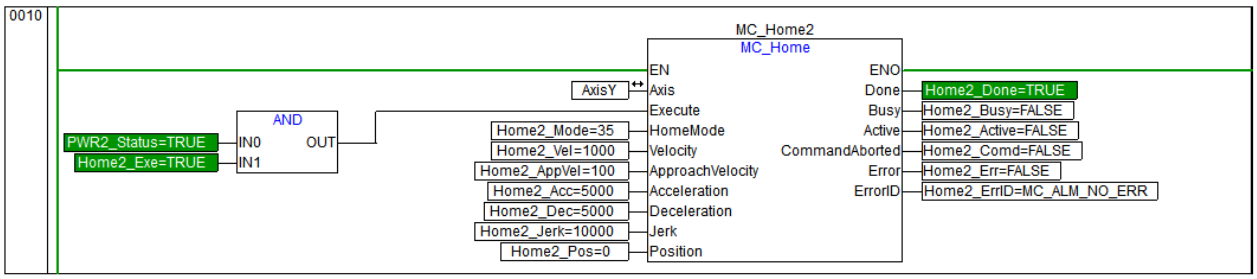
(4) Axis group enable. Call MC_GroupEnable to enable the axis group. If AxesGroupDisabled in the axis group variables is FALSE, the axis group is enabled successfully. See Section 0006. After the axis group is enabled successfully, invert AxesGroupDisabled and assign it to GrpEnableStatus variable, see Section 0008.

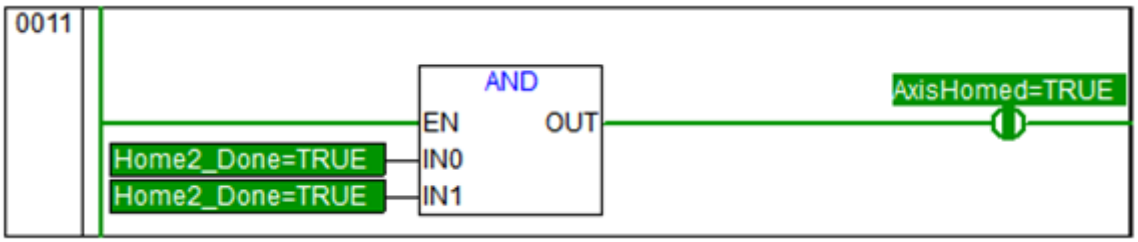




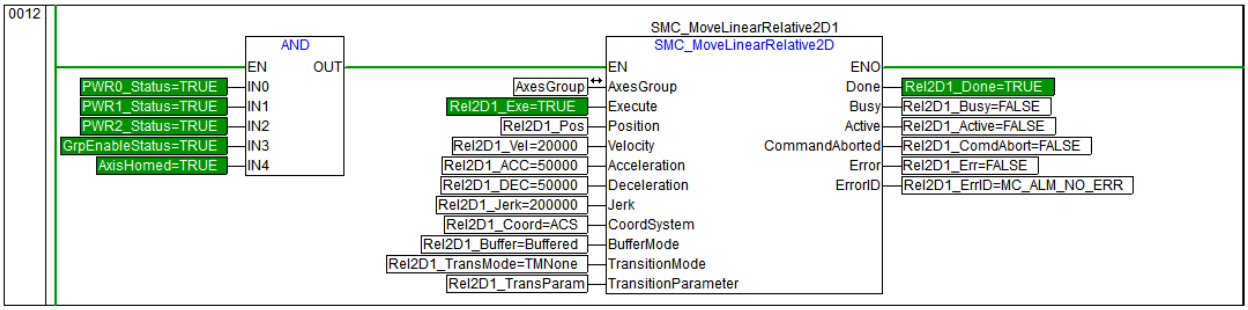
(5) After the two sub-axes in the axis group are enabled, if the home is not determined, perform homing. When homing is complete, the homing completed variable AxisHomed is set to TRUE. See Section 0009-0011.

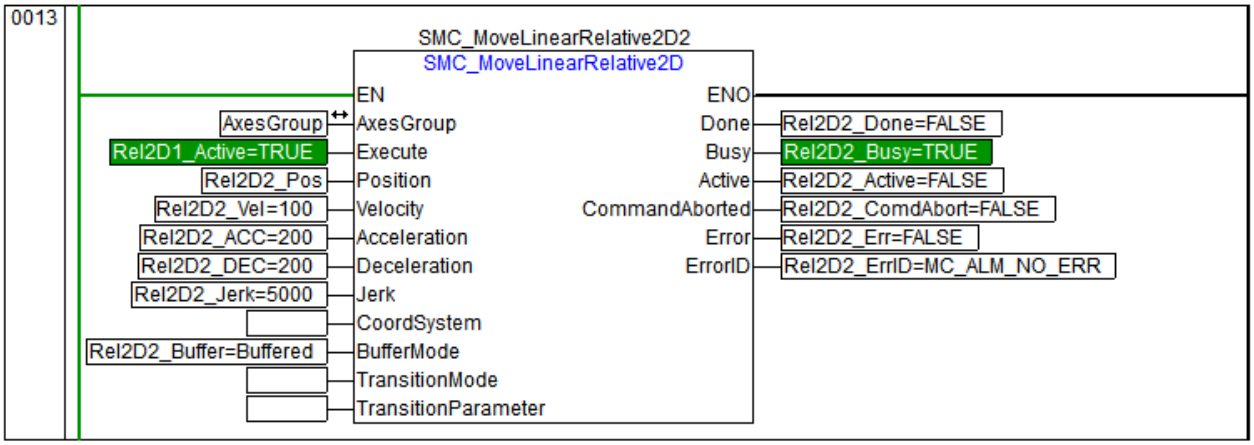


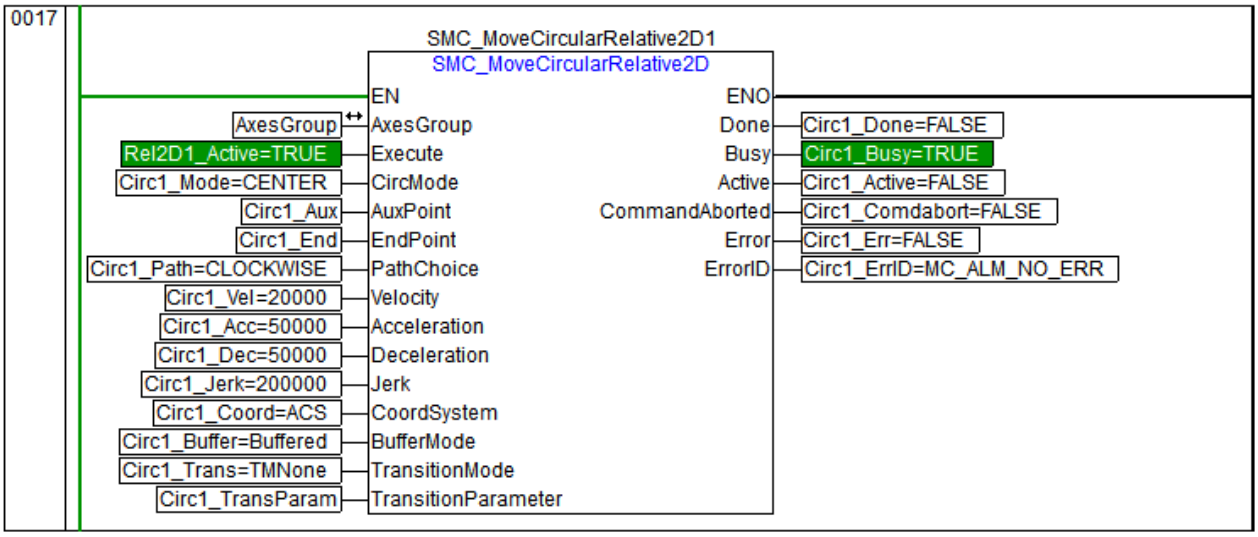


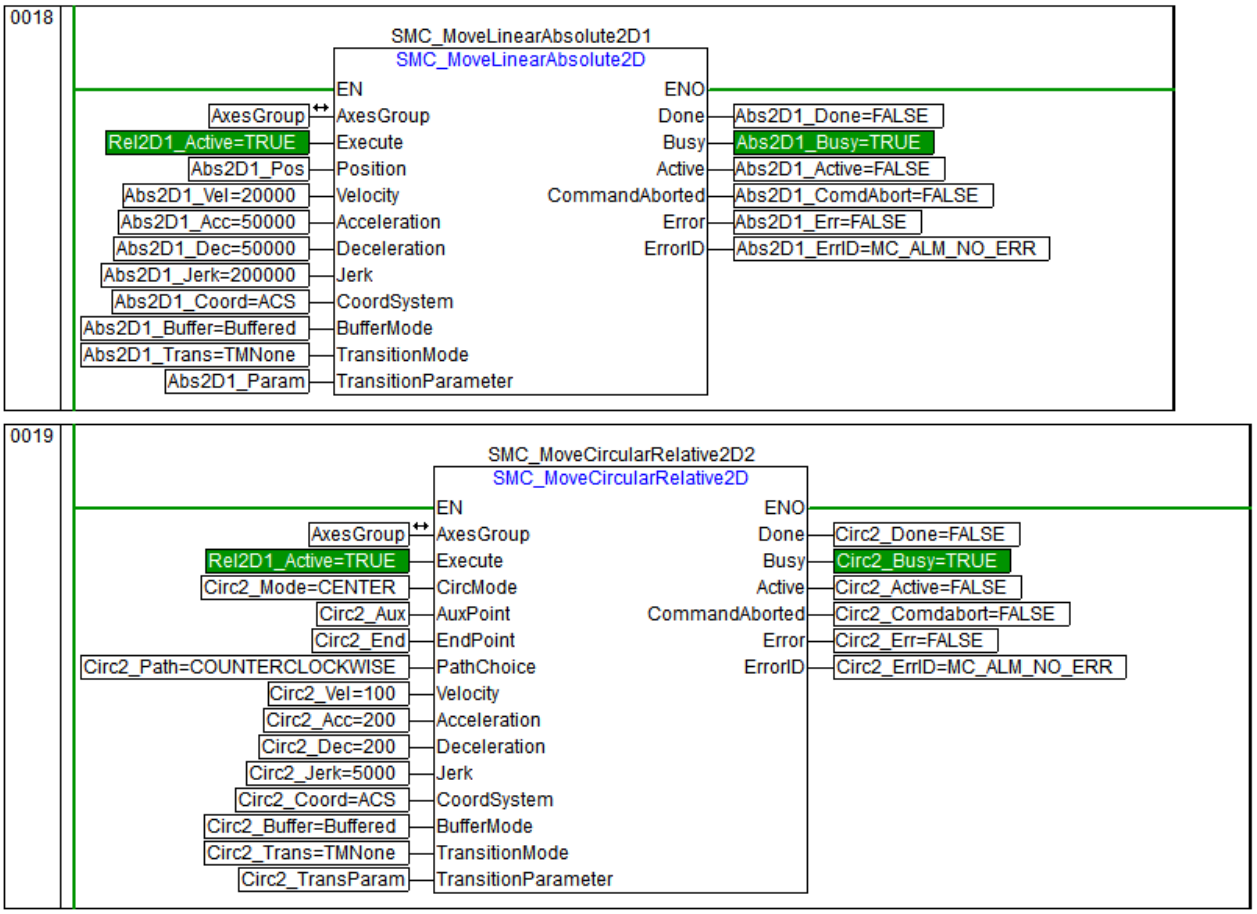


(6) In section 0011, after the homing is completed, the interpolation motion commands are executed. Execute command 1 and when Rel2D1_Active of command 1 is TRUE, commands 2-5 are sent sequentially. See section 0012-0013/0017-0019 below.


0012
AND
EN
OUT
PWR0_Status=TRUE
PWR1_Status=TRUE
PWR2_Status=TRUE
GrpEnableStatus=TRUE
AxisHomed=TRUE
IN0
IN1
IN2
IN3
IN4
SMC_MoveLinearRelative2D1
SMC_MoveLinearRelative2D
EN
ENO
AxesGroup
Rel2D1_Exe=TRUE
Rel2D1_Pos
Rel2D1_Vel=20000
Rel2D1_ACC=50000
Rel2D1_DEC=50000
Rel2D1_Jerk=200000
Rel2D1_Coord=ACS
Rel2D1_Buffer=Buffered
Rel2D1_TransMode=TMNone
Rel2D1_TransParam
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
Rel2D1_Done=TRUE
Rel2D1_Busy=FALSE
Rel2D1_Active=FALSE
Rel2D1_ComdAbort=FALSE
Rel2D1_Err=FALSE
Rel2D1_ErrID=MC_ALM_NO_ERR


0013
SMC_MoveLinearRelative2D2
SMC_MoveLinearRelative2D
EN
ENO
AxesGroup
Rel2D1_Active=TRUE
Rel2D2_Pos
Rel2D2_Vel=100
Rel2D2_ACC=200
Rel2D2_DEC=200
Rel2D2_Jerk=5000
Rel2D2_Buffer=Buffered
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
Rel2D2_Done=FALSE
Rel2D2_Busy=TRUE
Rel2D2_Active=FALSE
Rel2D2_ComdAbort=FALSE
Rel2D2_Err=FALSE
Rel2D2_ErrID=MC_ALM_NO_ERR


0017
SMC_MoveCircularRelative2D1
SMC_MoveCircularRelative2D
EN
ENO
AxesGroup
Rel2D1_Active=TRUE
Circ1_Mode=CENTER
Circ1_Aux
Circ1_End
Circ1_Path=CLOCKWISE
Circ1_Vel=20000
Circ1_Acc=50000
Circ1_Dec=50000
Circ1_Jerk=200000
Circ1_Coord=ACS
Circ1_Buffer=Buffered
Circ1_Trans=TMNone
Circ1_TransParam
Execute
CircMode
AuxPoint
EndPoint
PathChoice
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
Circ1_Done=FALSE
Circ1_Busy=TRUE
Circ1_Active=FALSE
Circ1_Comdabort=FALSE
Circ1_Err=FALSE
Circ1_ErrID=MC_ALM_NO_ERR



■ ST language program

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enabling operation.
Power1_Status	BOOL	FALSE	Variable that interpolates the sub-axis X-axis in the axis group and enables successfully. When the X-axis is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variable whose interpolation sub-axis and Y-axis are enabled successfully in the axis group. When the Y axis is enabled successfully, this variable becomes TRUE.
Power0_Status	BOOL	FALSE	Variable for successful interpolation of combined axis enable in the axis group. When the interpolation combined axis is enabled successfully, this variable becomes TRUE.
GrpEnableStatus	BOOL	TRUE	When this variable is TRUE, the axis group is enabled.
GrpEnable1_Exe	BOOL	FALSE	Axis group enable rising edge trigger variable.
AxisHomed	BOOL	FALSE	If this variable is TRUE, the homing of each axis in the axis group is completed.
InitFlag	BOOL	FALSE	If this variable is TRUE, the initialization parameter setting is completed.

Rel2D1_Exe	BOOL	FALSE	The rising-edge trigger variable of interpolation command 1 sent by the axis group.
Rel2D1_Active	BOOL	FALSE	The variable of the Active pin of interpolation command 1 sent by the axis group. When this variable becomes TRUE, the axis group executes the interpolation command.

(1) Set the IDs of joint and sub-axes in the axis group. Set the axis ID of the defined combined axis AxisI, sub-axes AxisX and AxisY.

```
IF Switch1 THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
END_IF
```

(2) Set command motion parameters

```
IF FALSE = InitFlag THEN
(* Parameter setting of linear interpolation command 1 for relative position. Default Values for unset input parameters *)
   Rel2D1_Pos[0]:=300;
   Rel2D1_Pos[1]:=400;
   Rel2D1_Vel:=100;
   Rel2D1_ACC:=200;
   Rel2D1_DEC:=200;
   Rel2D1_Jerk:=5000;
   Rel2D1_Buffer:=Buffered;
(* Parameter setting of relative position linear interpolation command 2. Default Values for unset input parameters *)
   Rel2D2_Pos[0]:= 0;
   Rel2D2_Pos[1]:=400;
   Rel2D2_Vel:=100;
   Rel2D2_ACC:=200;
   Rel2D2_DEC:=200;
   Rel2D2_Jerk:=5000;
   Rel2D2_Buffer:=Buffered;
(* Parameter setting of circular interpolation command 3. Default Values for unset input parameters *)
   Circ1_Mode:= BORDER;
   Circ1_Aux [0] :=400;
   Circ1_Aux [1] :=400;
   Circ1_End [0] :=800;
   Circ1_End [1] :=0;
   Circ1_Path:= CLOCKWISE;
   Circ1_Vel:=100;
   Circ1_ACC:=200;
   Circ1_DEC:=200;
   Circ1_Jerk:=5000;
```

```
    Circ1_Buffer:=Buffered;
(* Parameter setting of absolute position linear interpolation command 4. Default Values for unset input
parameters *)
    Abs2D1_Pos[0]:= 1100;
    Abs2D1_Pos[1]:=400;
    Abs2D1_Vel:=100;
    Abs2D1_ACC:=200;
    Abs2D1_DEC:=200;
    Abs2D1_Jerk:=5000;
    AbsD1_Buffer:=Buffered;
(* Parameter setting of circular interpolation command 5. Default Values for unset input parameters *)
    Circ1_Mode:= CENTER;
    Circ1_Aux [0] :=400;
    Circ1_Aux [1] := 0;
    Circ1_End [0] :=400;
    Circ1_End [1] :=400;
    Circ1_Path:= COUNTERCLOCKWISE;
    Circ1_Vel:=100;
    Circ1_ACC:=200;
    Circ1_DEC:=200;
    Circ1_Jerk:=5000;
    Circ1_Buffer:=Buffered;
    InitFlag:=TRUE;
END_IF
```

(3) Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group. Set the axis type of axis combination to 0: Virtual; set the axis types of axis group AxisX/AxisY to 50: EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to its command chapter.

(4) Execute the MC_Power command to enable the three axes in the axis group.

```
IF StartPro THEN
MC_Power0(*Combined axis enable*)
    Enable := PWR0_Enable,
    Axis := AxisI,
    Status=> PWR0_Status,
    Busy=> PWR0_Busy,
    Error=> PWR0_Err,
    ErrorID=> PWR0_ErrID);
MC_Power1((*Sub-axis AxisX Enable*)
    Enable := PWR1_Enable,
    Axis := AxisX,
    Status=> PWR1_Status,
    Busy=> PWR1_Busy,
    Error=> PWR1_Err,
    ErrorID=> PWR1_ErrID);
MC_Power2 (*Sub-axis AxisY Enable*)
```

```
   Enable := PWR2_Enable,
   Axis := AxisY,
   Status=> PWR2_Status,
   Busy=> PWR2_Busy,
   Error=> PWR2_Err,
   ErrorID=> PWR2_ErrID);
END_IF
```

(5) Execute axis group enable

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
GrpEnableStatus:= AxesGroup. AxesGroupDisabled;
```

(6) Execute the homing command of sub-axis in the axis group to determine the home. For the usage MC_Home command, see its introduction chapter.

```
IF  PWR1_Status THEN
   MC_Home1(
   Execute := Home1_Exe,
   HomeMode := Home1_Mode,
   Velocity := Home1_Vel,
   ApproachVelocity := Home1_AppVel,
   Acceleration := Home1_Acc,
   Deceleration := Home1_Dec,
   Jerk := Home1_Jerk,
   Position := Home1_Pos,
   Axis := AxisX,
   Done=> Home1_Done,
   Busy=> Home1_Busy,
   Active=> Home1_Active,
   CommandAborted=> Home1_Comd,
   Error=> Home1_Err,
   ErrorID=> Home1_ErrID);
   MC_Home2(
   Execute := Home2_Exe,
   HomeMode := Home2_Mode,
   Velocity := Home2_Vel,
   ApproachVelocity := Home2_AppVel,
   Acceleration := Home2_Acc,
   Deceleration := Home2_Dec,
   Jerk := Home2_Jerk,
   Position := Home2_Pos,
   Axis := AxisY,
```

```
    Done=> Home2_Done,
    Busy=> Home2_Busy,
    Active=> Home2_Active,
    CommandAborted=> Home2_Comd,
    Error=> Home2_Err,
    ErrorID=> Home2_ErrID);
END_IF
IF  Home1_Done  AND Home2_Done THEN
    AxisHomed:=TRUE;
END_IF
```

(7) After the axis homing is completed, start executing the interpolation motion command. Execute command 1. When Rel2D1_Active of command 1 is TRUE, commands 2-5 are sent sequentially.

```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus AND AxisHomed THEN
SMC_MoveLinearRelative2D1(*interpolation command 1*)
    Execute := Rel2D1_Exe,
    Position := Rel2D1_Pos,
    Velocity := Rel2D1_Vel,
    Acceleration := Rel2D1_ACC,
    Deceleration := Rel2D1_DEC,
    Jerk := Rel2D1_Jerk,
    CoordSystem := Rel2D1_Coord,
    BufferMode := Rel2D1_Buffer,
    TransitionMode := Rel2D1_Trans,
    TransitionParameter := Rel2D1_Param,
    AxesGroup := AxesGroup,
    Done=> Rel2D1_Done,
    Busy=> Rel2D1_Busy,
    Active=> Rel2D1_Active,
    CommandAborted=> Rel2D1_ComdAbort,
    Error=> Rel2D1_Err,
    ErrorID=> Rel2D1_ErrID);
SMC_MoveLinearRelative2D2(*interpolation command 2*)
    Execute := Rel2D1_Active,
    Position := Rel2D2_Pos,
    Velocity := Rel2D2_Vel,
    Acceleration := Rel2D2_ACC,
    Deceleration := Rel2D2_DEC,
    Jerk := Rel2D2_Jerk,
    CoordSystem := Rel2D2_Coord,
    BufferMode := Rel2D2_Buffer,
    TransitionMode := Rel2D2_Trans,
    TransitionParameter := Rel2D2_Param,
    AxesGroup := AxesGroup,
    Done=> Rel2D2_Done,
    Busy=> Rel2D2_Busy,
```

```
    Active=> Rel2D2_Active,
    CommandAborted=> Rel2D2_ComdAbort,
    Error=> Rel2D2_Err,
    ErrorID=> Rel2D2_ErrID);
SMC_MoveCircularRelative2D1(*interpolation command 3*)
    Execute := Rel2D1_Active,
    CircMode := Circ1_Mode ,
    AuxPoint := Circ1_Aux,
    EndPoint := Circ1_End,
    PathChoice:= Circ1_Path,
    Velocity := Circ1_Vel,
    Acceleration := Circ1_ACC,
    Deceleration := Circ1_DEC,
    Jerk := Circ1_Jerk,
    CoordSystem := Circ1_Coord,
    BufferMode := Circ1_Buffer,
    TransitionMode := Circ1_ Trans,
    TransitionParameter := Circ1_Param,
    AxesGroup := AxesGroup,
    Done=> Circ1_Done,
    Busy=> Circ1_Busy,
    Active=> Circ1_Active,
    CommandAborted=> Circ1_Abort,
    Error=> Circ1_Err,
    ErrorID=> Circ1_ErrID);
SMC_MoveLinearAbsolute2D1(*interpolation command 4*)
    Execute := Rel2D1_Active,
    Position := Abs2D1_Pos,
    Velocity := Abs2D1_Vel,
    Acceleration := Abs2D1_ACC,
    Deceleration := Abs2D1_DEC,
    Jerk := Abs2D1_Jerk,
    CoordSystem := Abs2D1_Coord,
    BufferMode := Abs2D1_Buffer,
    TransitionMode := Abs2D1_Trans,
    TransitionParameter := Abs2D1_Param,
    AxesGroup := AxesGroup,
    Done=> Abs2D1_Done,
    Busy=> Abs2D1_Busy,
    Active=> Abs2D1_Active,
    CommandAborted=> Abs2D1_ComdAbort,
    Error=> Abs2D1_Err,
    ErrorID=> Abs2D1_ErrID);
SMC_MoveCircularRelative2D2(*interpolation command 5*)
    Execute := Rel2D1_Active,
```

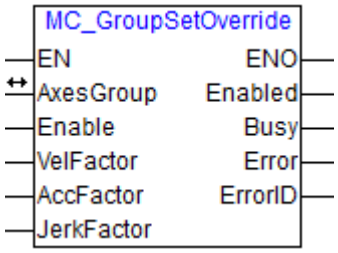
```
    CircMode := Circ2_Mode ,
    AuxPoint := Circ2_Aux,
    EndPoint := Circ2_End,
    PathChoice:= Circ2_Path,
    Velocity := Circ2_Vel,
    Acceleration := Circ2_ACC,
    Deceleration := Circ2_DEC,
    Jerk := Circ2_Jerk,
    CoordSystem := Circ2_Coord,
    BufferMode := Circ2_Buffer,
    TransitionMode := Circ2_ Trans,
    TransitionParameter := Circ2_Param,
    AxesGroup := AxesGroup,
    Done=> Circ2_Done,
    Busy=> Circ2_Busy,
    Active=> Circ2_Active,
    CommandAborted=> Circ2_Abort,
    Error=> Circ2_Err,
    ErrorID=> Circ2_ErrID);
END_IF
```

7.6.4 Precautions

- When the starting point and endpoint are the same, an exception will occur that the starting point and end point of circular interpolation are the same.
- BufferMode (buffer mode selection) in this command does not support 0: Aborting.

7.7 MC_GroupSetOverride (Set Axes Group Override Factors)

This command sets the speed, acceleration/deceleration and jerk scale factor of the combined axis in the axis group, which can increase or decrease the running speed, acceleration/deceleration and jerk of the combined axis according to the set ratio.



7.7.1 Parameter

Parameter Type	Title	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	AxesGroup	Axis group name	No	-	-	AXES_GROUP_REF
INPUT	Enable	Enable	No	-	TRUE/FALSE	BOOL
	VelFactor	is the speed scaling factor in 1%	YES	100	0 ~ 500	LREAL
	AccFactor	Acceleration and deceleration scale factor, in 1%	YES	100	0 ~ 500 (non-zero)	LREAL
	JerkFactor	is the jerk scale factor in 1%	YES	100	0 ~ 500 (non-zero)	LREAL
OUTPUT	Enabled	Setting succeeded	No	-	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

7.7.2 Functions

This command can increase or decrease the running speed, acceleration and deceleration of the axis group according to the set scale factor. The details of this command are as follows:

(1) This command overrides the target speed of the axis group, and can change the target speed during the movement of the axis group. The commands that can be changed are as follows:

- □ SMC_MoveLinearAbsolute2D
- □ SMC_MoveCircularRelative2D
- □ SMC_MoveLinearRelative2D

(2) The unit of override scaling factor is [%].

- □ Target speed after change = currently executed speed × speed scale factor;
- □ Changed acceleration/deceleration = current acceleration/deceleration × scale factor of acceleration/deceleration;
- □ Changed jerk = current jerk × jerk scale factor.

(3) When the target speed after override is greater than the maximum speed, the running speed of the axis group is the maximum speed. The same applies to acceleration/deceleration and jerk.

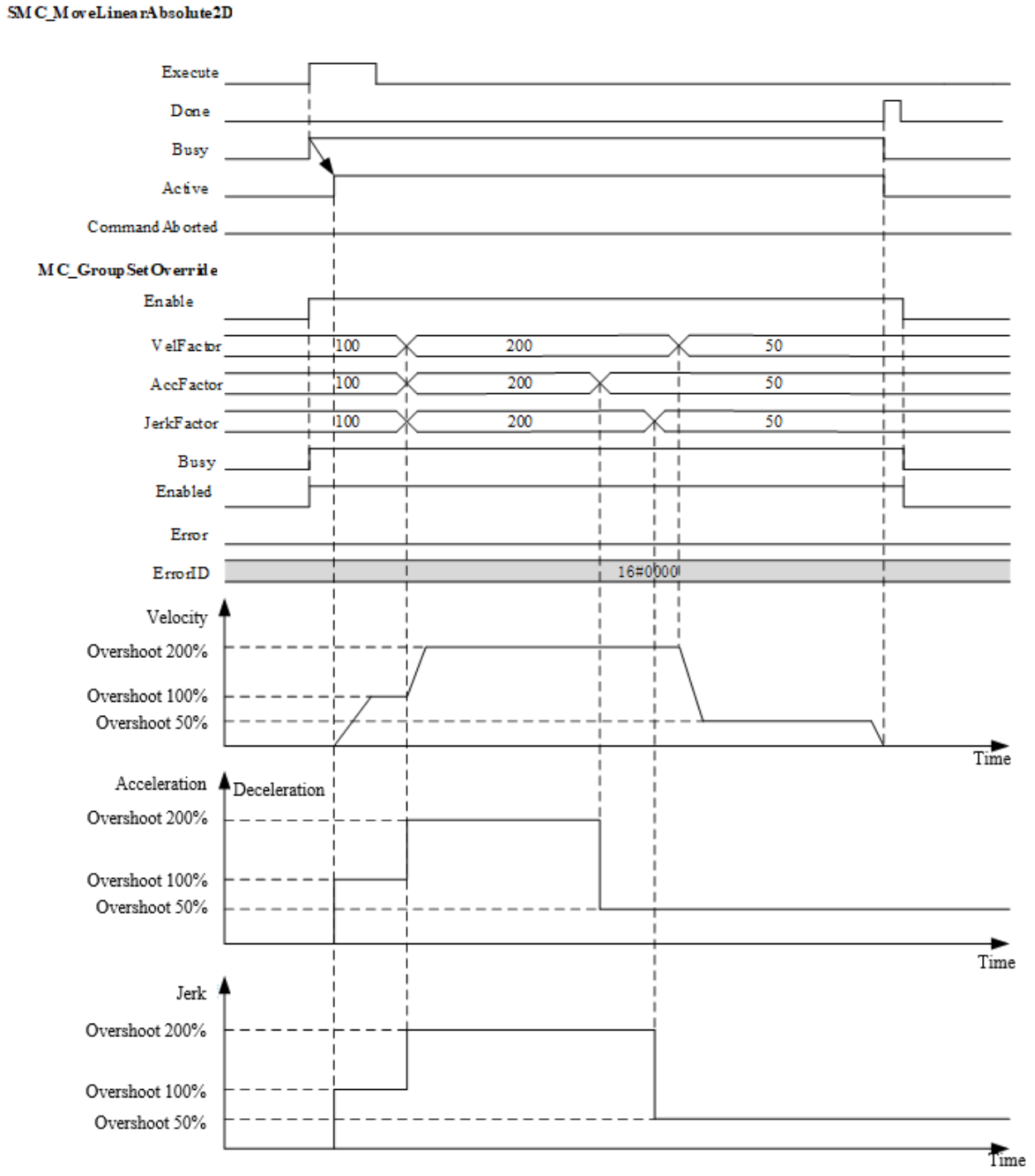
(4) When Enable is TRUE, the function block continues to set the override of scale factor. When

Enable is FALSE, the axis group keeps the override scale factor value set last time for motion.

■ Function block timing diagram

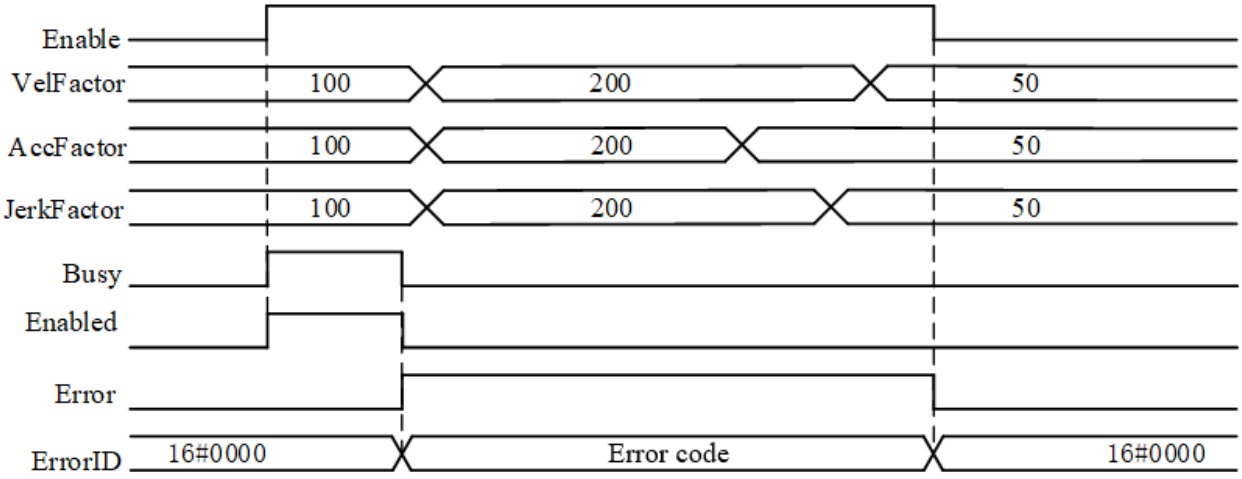
(1) Override setting of SMC_ MoveLinearAbsolute2D (2-axis absolute position interpolation command)

When Enable is TRUE, Busy and Enabled are immediately TRUE; when Enable is FALSE, Busy and Enabled are immediately FALSE. The timing diagram that uses override command in SMC_MoveLinearAbsolute2D (axis 2 absolute position interpolation command) is as follows.



(2) Error timing diagram

When an exception occurs during the execution of this command, Error becomes TRUE and Enabled and Busy become FALSE. After the error is cleared, Error becomes FALSE and Enabled and Busy become TRUE. The timing diagram is shown below.



(3) Restart and multiple start commands

The input of this command is Enable type. When Enable is TRUE, the function block runs all the time. When there is an MC_GroupSetOverride command being executed and other MC_GroupSetOverride commands are started, the later-executed command will be processed first.

7.7.3 Examples

Set up MC_GroupSetOverride example program to set the override value of axis group motion parameters.

Description of Primary Variables

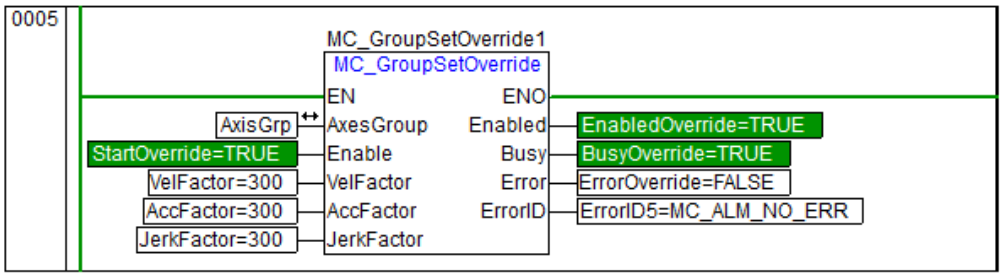
Title	Data Type	Initial Value	Notes
AxisGrp	AXES_GROUP_REF		Variable name of axis group
StartOverride	BOOL	FALSE	Change to TRUE at the start of override execution
BusyOverride	BOOL	FALSE	Change to TRUE at override
EnabledOverride	BOOL	FALSE	Change to TRUE when override is successful
ErrorOverride	BOOL	FALSE	Change to TRUE in case of command error

■ LD diagram program

The ladder diagram is used to build the program, and the input parameters VelFactor, AccFactor and JerkFactor are all 300. Execute the SMC_MoveLinearAbsolute2D command to control the axis group for motion. Set Enable of MC_GroupSetOverride to TRUE, and the speed, acceleration/deceleration and jerk of the axis group are all values after overriding. The example program is as follows:

(1) Initialize and enable the axis group. Set the IDs and types of X, Y and combined axes respectively. Set the combined axis type as a virtual axis. Then enable the axis group. After enabling, send SMC_MoveLinearAbsolute2D command to the axis group. The rising edge triggers the control axis group to run with 2-axis absolute position interpolation command. For specific usage, please refer to the section of SMC_MoveLinearAbsolute2D command.

(2) Keep the Enable pin of MC_GroupSetOverride at high level to set the override parameters.



- ST language program

(1) Initialize and enable the axis group. Set the IDs and types of X, Y and combined axes respectively. Set the combined axis type as a virtual axis. Then enable the axis group. After enabling, send SMC_MoveLinearAbsolute2D command to the axis group. The rising edge triggers the control axis group to run with 2-axis absolute position interpolation command. For specific usage, please refer to the section of SMC_MoveLinearAbsolute2D command.

(2) Keep MC_GroupSetOverride at high level to set the override of motion parameters.

```
MC_GroupSetOverride1(
Enable := StartOverride,
VelFactor := 300,
AccFactor := 300,
JerkFactor := 300,
Axis := Axis0,
Enabled=>EnabledOverride,
Busy=>BusyOverride,
Error=>ErrorOverride,
ErrorID=>ErrorID);
```

7.7.4 Precautions

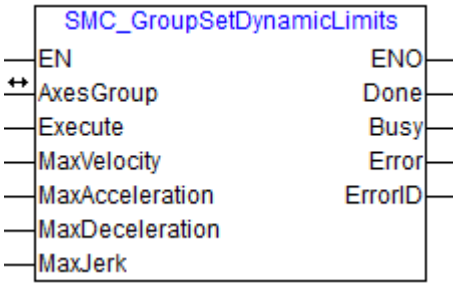
- When using this command for override setting, if the maximum speed of the axis group is exceeded, the maximum speed will be taken as the running speed.
- This command has no effect on the encoder axis.
- For a command that this command is invalid, Enabled will remain TRUE even if the "Enabled" of the command is set to TRUE.
- This command is only valid for the combined axis in the axis group.

7.7.5 Reference

- For axis group motion commands, refer to the section of SMC_MoveLinearAbsolute2D command.
- For error codes, please refer to the appendix Functional Block Error Codes.

7.8 SMC_GroupSetDynamicLimits (Set Axes Group Dynamic Parameter Limits)

This command sets the limit values of the axis group motion parameters.



7.8.1 Parameter

Parameter Type	Name	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	AxesGroup	Axis group name	No	-	-	AXIS_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	MaxVelocity	Maximum speed	YES	1E100	Positive/0	LREAL
	MaxAcceleration	Maximum acceleration	YES	1E100	Positive value	LREAL
	MaxDeceleration	Maximum deceleration	YES	1E100	Positive value	LREAL
	MaxJerk	Maximum jerk	YES	1E100	Positive value	LREAL
OUTPUT	Done	Completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

7.8.2 Functions

This command sets the limit values of axis group motion parameters.

■ Command function details

(1) Input parameter latch

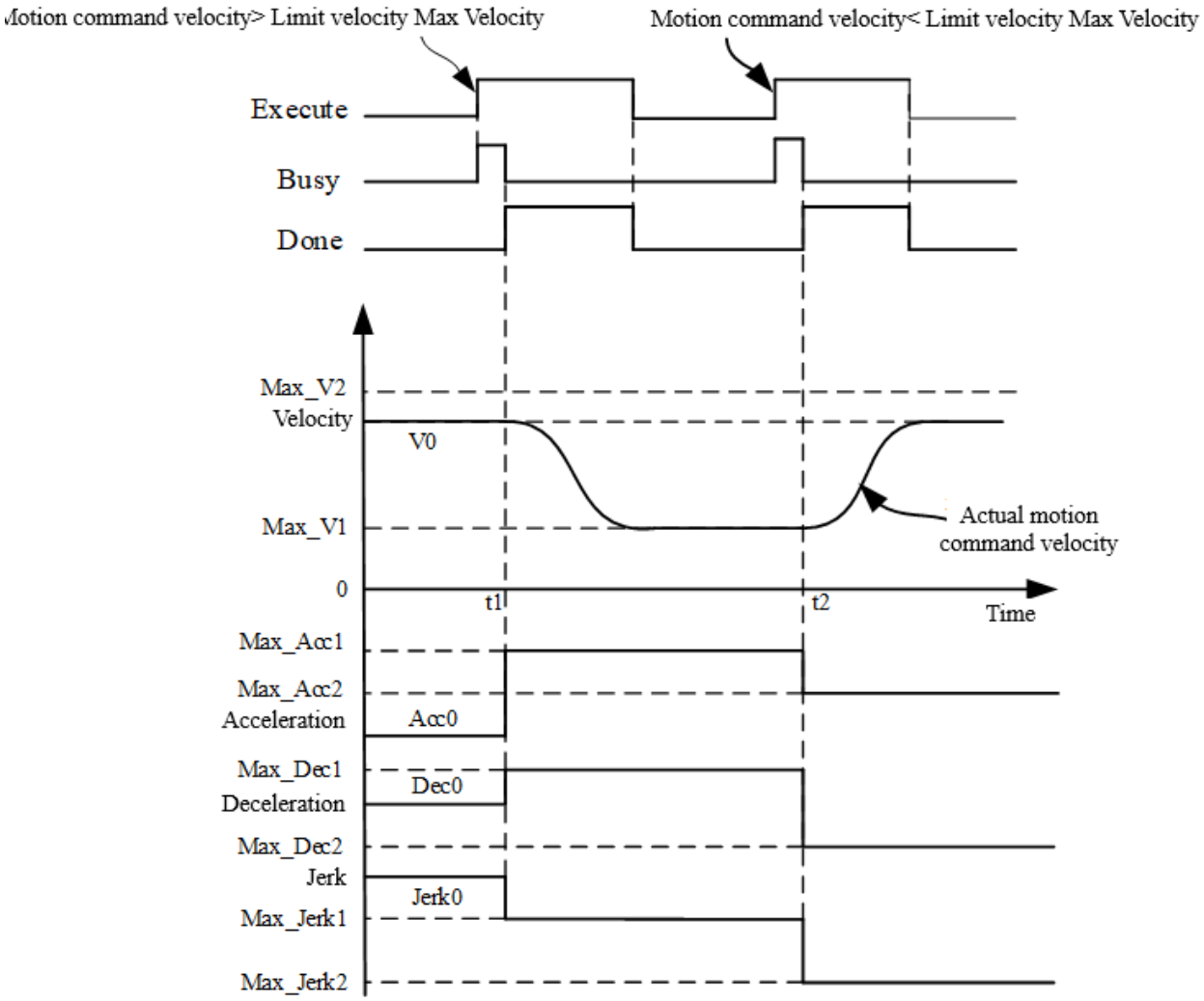
The rising edge triggering of this command is valid. When executing the rising edge of input, latch the IDs of the combined axis and two sub-axes of the input/output parameter AxesGroup, as well as the values of input parameters MaxVelocity, MaxAcceleration, MaxDeceleration and MaxJerk. During command execution, modifying the latched parameters is invalid until another rising edge triggers this command.

(2) Functional details

After the limit values of axis group motion parameters are set, the motion parameter of the axis

group motion command takes the minimum value of its own motion parameter and the set limit parameter.

The command is triggered twice by the rising edge, and different values of MaxVelocity, MaxAcceleration, MaxDeceleration and MaxJerk are set for each trigger. Set the maximum motion parameters as shown in the following figure:



In the above figure, V0 is a command speed of the motion command, Acc0 is a command acceleration of the motion command, Dec0 is a command deceleration of the motion command, and Jerk0 is a command jerk of the motion command.

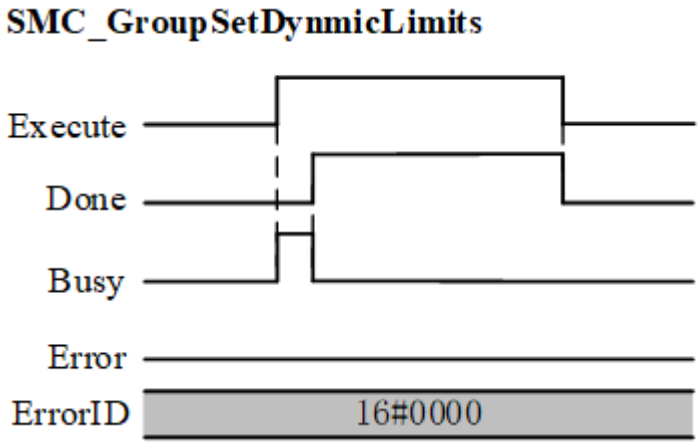
At time t1, the maximum speed MaxVelocity is successfully set (MaxVelocity= Max_V1). MaxVelocity is less than the combined axis command velocity V0 of the axis group motion command. The maximum speed during the running of the axis group motion command is MaxVelocity, and the axis group motion command slows down. Meanwhile, Max_Acc1, Max_Dec1 and Max_Jerk1 are the set limit values of other motion parameters respectively.

At time t2, the maximum speed MaxVelocity is successfully set again, and MaxVelocity=Max_V2.

MaxVelocity is greater than the speed V0 of the axis group motion command. The axis group motion command takes the self-set command speed as the actual running speed, and the axis group motion command increases. Meanwhile, Max_Acc2, Max_Dec2 and Max_Jerk2 are the set limit values of other motion parameters respectively.

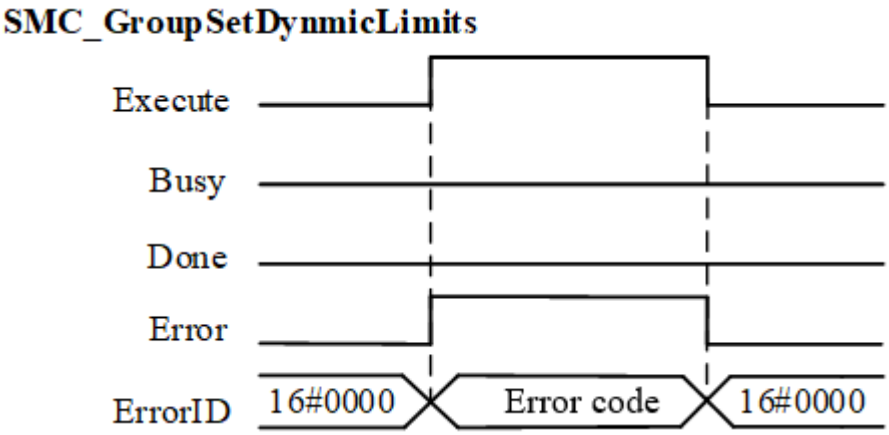
■ Timing diagram

(1) Normal timing diagram



(2) Error pin timing diagram

When this command is executed, if errors such as illegal parameters and illegal axis parameters in the axis group occur, the call of this command fails, with command output pin Error=True and error code ErrorID reporting an abnormal error value. Please refer to the appendix Description of error codes to obtain the cause of the error. When the command input pin Execute goes low, all output pins revert to their Default Values.



- When the abnormal situation of the command is solved, it is necessary to trigger Execute again on the rising edge and start the SMC_SetGroupDynamicLimits functional block to complete the setting of axis motion parameter limit.

7.8.3 Examples

Set up SMC_GroupSetDynamicLimits sample program to set the limit values of axis group motion parameters.

- Action example

(1) Execute the interpolation motion command SMC_MoveLinearRelative2D. The speed in this command is set as 20000, with acceleration and deceleration of 20,000 and jerk of 500,000 to control the movement of axis group.

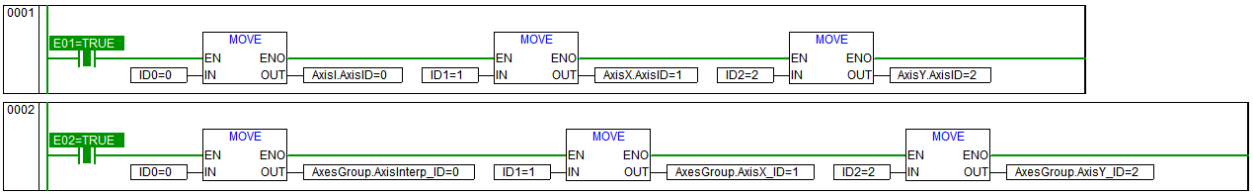
(2) Execute the SMC_GroupSetDynamicLimits command and set the motion parameter limit MaxVelocity to 10000, MaxAcceleration and MaxDeceleration to 20000, and MaxJerk to 50000.

- LD diagram program

Examples of Primary Variables

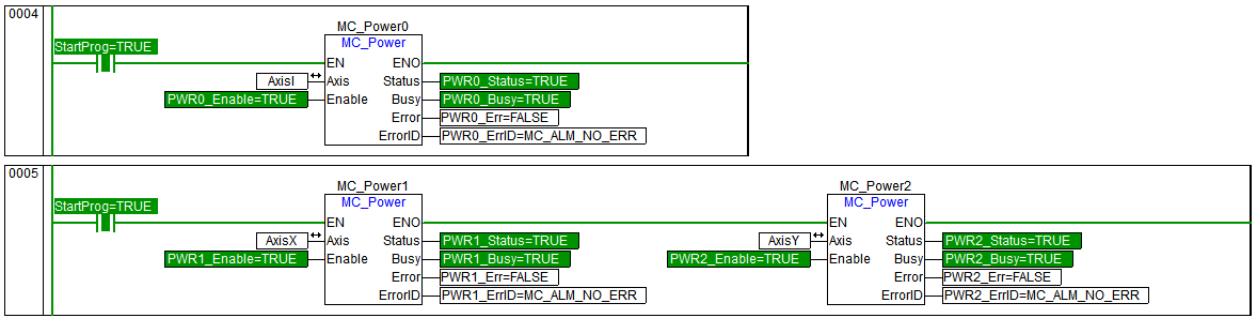
Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enabling operation.
Power1_Status	BOOL	FALSE	Variables whose Sub-axis-axis X enable is successful. When the Sub-axis X is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variables whose Sub-axis-axis Y enable is successful. When the Sub-axis Y is enabled successfully, this variable becomes TRUE.
Power0_Status	BOOL	FALSE	Variable for successful combination of axes enabling. This variable becomes TRUE when the combined axis is enabled successfully.
GrpEnableStatus	BOOL	TRUE	When this variable is TRUE, the axis group is enabled.
GrpEnable1_Exe	BOOL	FALSE	Axis group enable rising edge trigger variable.
AxisHomed	BOOL	FALSE	If this variable is TRUE, the return to the origin of each axis in the axis group is completed.
Rel2D1_Exe	BOOL	FALSE	Rising-edge trigger variable of the relative interpolation command delivered.
Rel2D1_Vel	BOOL	FALSE	Variable corresponding to the input speed of relative interpolation command delivered.
Dyn_Exe	BOOL	FALSE	Rising edge trigger variable of axis group motion parameter limit command.

(1) Set the IDs of joint and sub-axes in the axis group. When defining an axis group, you need to define three axes in the axis group first, and match the IDs of the defined axis numbers of the three axes with those of the combined and sub-axes in the axis group. See sections 0001 and 0002 below.

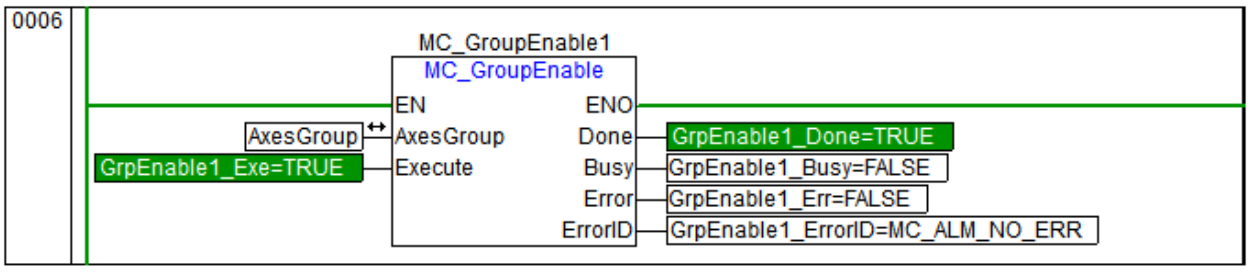


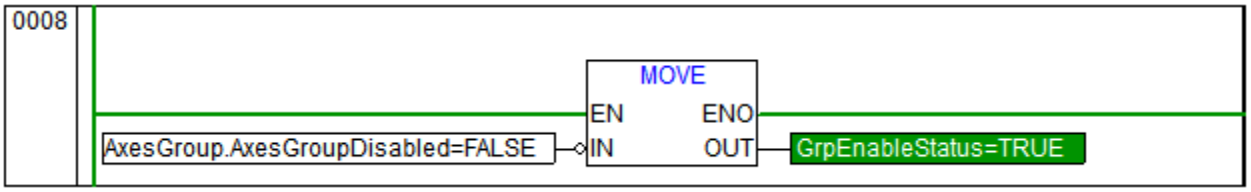
(2) Axis initialization. Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group respectively, and set the combined axis as 0: Virtual and the Sub-axis type as 50: EtherCAT_Position. Please refer to relevant chapters for the usage of SMC_AxisInit command.

(3) Axis enable. After each axis in the axis group is ready (StartProg =TRUE), call MC_Power to enable the axes of the combined axis and two sub-axes of the axis group. See Sections 0004 to 0005.

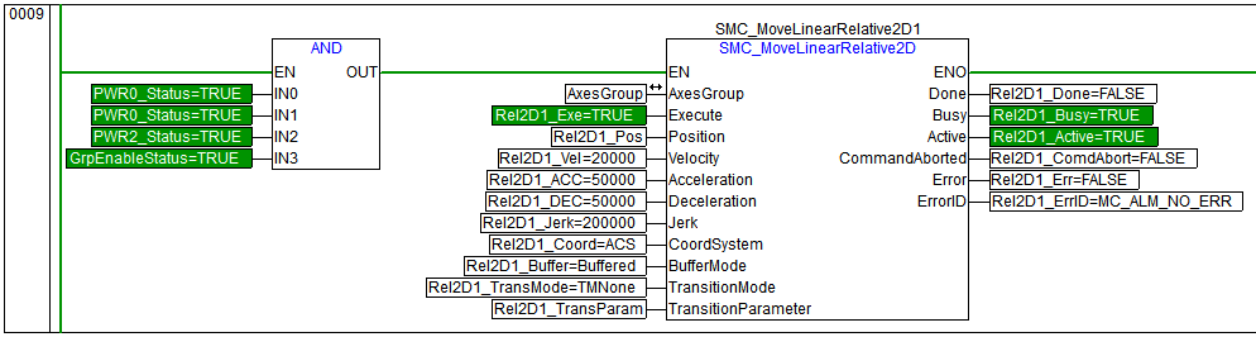


(4) Axis group enable. Call MC_GroupEnable to enable the axis group. If AxesGroupDisabled in the axis group variables is FALSE, the axis group is enabled successfully. See Section 0006. After the axis group is enabled successfully, reverse AxesGroupDisabled and assign it to GrpEnableStatus variable.

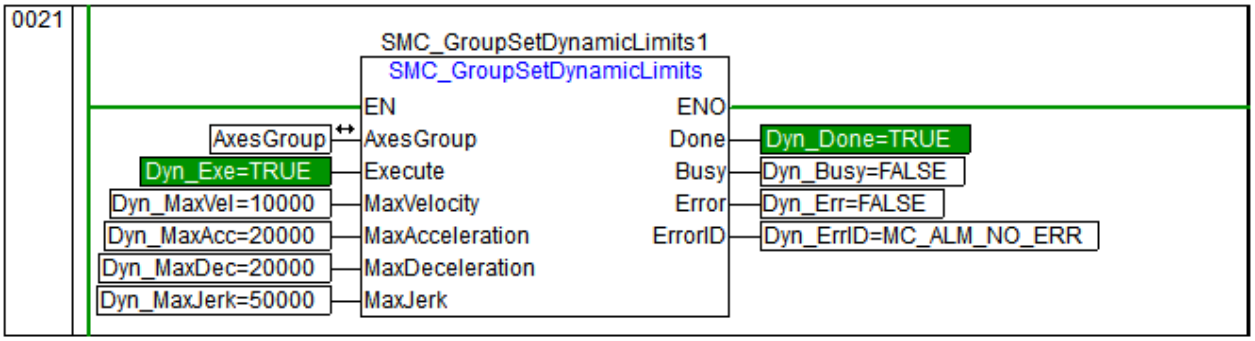




(5) In Section 0009, after each axis in the axis group and the axis group are enabled, trigger the relative position interpolation motion command to control the movement of the axis group. Please refer to relevant chapters for the usage of SMC_MoveLinearRelative2D command.



(6) In section 0021, execute the SMC_GroupSetDynamicLimits command to set the limit value of axis group motion parameters.



■ ST language program

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enabling operation.
Power0_Status	BOOL	FALSE	Variables whose Sub-axis-axis X enable is successful. When the Sub-axis X is enabled successfully, this variable becomes TRUE.
Power1_Status	BOOL	FALSE	Variables whose Sub-axis-axis Y enable is successful. When the Sub-axis Y is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variable for successful combination of axes enabling. This variable becomes TRUE when the combined axis is enabled successfully.
GrpEnableStatus	BOOL	TRUE	When this variable is TRUE, the axis group is enabled.
GrpEnable1_Exe	BOOL	FALSE	Axis group enable rising edge trigger variable.
AxisHomed	BOOL	FALSE	If this variable is TRUE, the return to the origin of each axis in the axis group is completed.
Rel2D1_Exe	BOOL	FALSE	Rising-edge trigger variable of the relative interpolation command delivered.
Rel2D1_Vel	BOOL	FALSE	Variable corresponding to the input speed of relative interpolation command delivered. .
Dyn_Exe	BOOL	FALSE	Rising edge trigger variable of motion parameter limit command.
InitFlag	BOOL	FALSE	If this variable is TRUE, the initialization parameter setting is completed.

(1) Set the IDs of combined axis and sub-axes in the axis group. Set the axis ID of the defined combined axis AxisI, sub-axes AxisX and AxisY.

```
IF FALSE=InitFlag THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
Dyn_MaxVel:=10000; (*speed limit *)
Dyn_MaxAcc:=20000; (* acceleration limit *)
Dyn_MaxDec:=20000; (*deceleration limit *)
Dyn_MaxJerk:=50000; (*jerk limit *)
   InitFlag:=TRUE;
END_IF
```

(2) Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group. Set the axis type of axis combination to 0: Virtual; set the axis types of axis group AxisX/AxisY to 50: EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to its command chapter.

(3) Execute the MC_Power command to enable the three axes in the axis group.

```
IF StartPro THEN
MC_Power0(*Combined axis enable*)
   Enable := PWR0_Enable,
   Axis := AxisI,
   Status=> PWR0_Status,
   Busy=> PWR0_Busy,
   Error=> PWR0_Err,
   ErrorID=> PWR0_ErrID);
MC_Power1((*Sub-axis AxisX enable*)
   Enable := PWR1_Enable,
   Axis := AxisX,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
   Error=> PWR1_Err,
   ErrorID=> PWR1_ErrID);
MC_Power2 (*Sub-axis AxisY enable*)
   Enable := PWR2_Enable,
   Axis := AxisY,
   Status=> PWR2_Status,
   Busy=> PWR2_Busy,
   Error=> PWR2_Err,
   ErrorID=> PWR2_ErrID);
END_IF
```

(4) Execute axis group enable.

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
GrpEnableStatus:= AxesGroup. AxesGroupDisabled;
```

(5) After the axis and axis group enable is completed, execute the interpolation command of axis group.

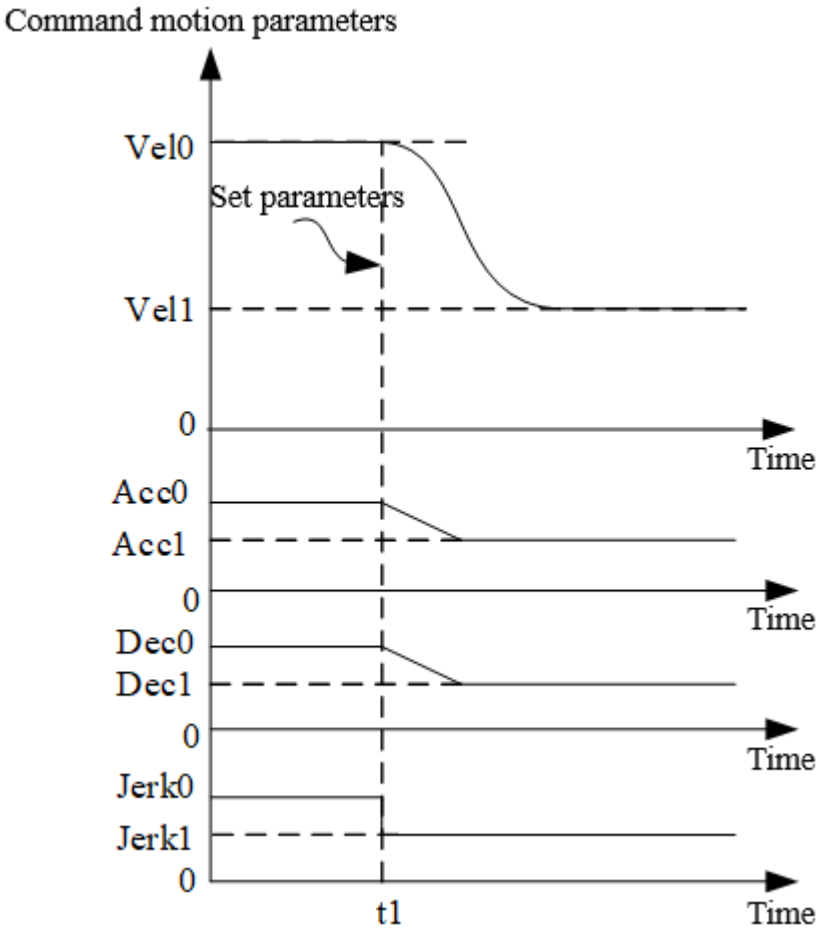
```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus THEN
   SMC_MoveLinearRelative2D1(
   Execute := Rel2D1_Exe,
   Position := Rel2D1_Pos,
   Velocity := Rel2D1_Vel,
   Acceleration := Rel2D1_ACC,
   Deceleration := Rel2D1_DEC,
   Jerk := Rel2D1_Jerk,
   CoordSystem := Rel2D1_Coord,
   BufferMode := Rel2D1_Buffer,
   TransitionMode := Rel2D1_Trans,
   TransitionParameter := Rel2D1_Param,
   AxesGroup := AxesGroup,
   Done=> Rel2D1_Done,
   Busy=> Rel2D1_Busy,
   Active=> Rel2D1_Active,
   CommandAborted=> Rel2D1_ComdAbort,
   Error=> Rel2D1_Err,
   ErrorID=> Rel2D1_ErrID);
END_IF
```

(6) Execute the SMC_GroupSetDynamicLimits command to complete the motion parameter limit setting.

```
SMC_SetGroupDynamicLimits1(
Execute := Dyn_Exe,
MaxVelocity := Dyn_MaxVel,
MaxAcceleration := Dyn_MaxAcc,
MaxDeceleration := Dyn_MaxDec,
MaxJerk := Dyn_MaxJerk,
AxesGroup := AxesGroup,
Done=> Dyn_Done,
Busy=> Dyn_Busy,
Error=> Dyn_Err,
ErrorID=> Dyn_ErrID);
```

Observe the motion parameter curve of the interpolation motion command. During 0~t1, the motion

parameters of the interpolation command are not limited, and the actual running speed of the interpolation command reaches the set command speed; after t1, the motion parameter limit set by the SMC_GroupSetDynamicLimits command takes effect, and the interpolation command speed is decelerated to the speed limit value. The diagram of limited interpolation closing axis motion parameters is as follows:



Vel0, Acc0, Dec0 and Jerk0 are the command speed, acceleration, deceleration and jerk of the interpolation command respectively.

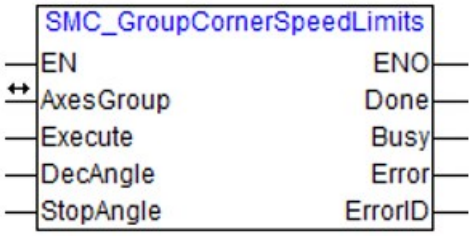
Vel1, Acc1, Dec1 and Jerk1 are respectively the speed, acceleration, deceleration and jerk after setting the limit parameters of interpolation command.

7.8.4 Precautions

- When this command is called, only MaxVelocity can be set to 0, and the remaining input motion limit parameters need to be positive.
- After the maximum parameter is successfully set by this command, the falling edge cannot cancel the maximum parameter limit. The user needs to call this command again to set the parameter limit value.

7.9 SMC_GroupCornerSpeedLimits (Set Axes Group Corner Speed Limits)

This command sets the corner speed limit parameters during the movement of the axis group to reduce the motion impact at non-smooth corner joints.



7.9.1 Parameter

Parameter Type	Title	Description	Empty or Not	Default Value	Scope	Data Type
IN_OUT	AxesGroup	Axis group name	No	-	-	AXES_GROUP_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	DecAngle	Critical deceleration angle in degrees	YES	30	0 ~ 30 (≤StopAngle)	LREAL
	StopAngle	Critical stopping angle in degrees	YES	90	0 ~ 90 (not 0)	LREAL
OUTPUT	Done	Completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

7.9.2 Functions

This command sets the corner speed limit parameters during the movement of the axis group to reduce the motion impact at non-smooth corner joints.

■ Command function details

(1) Input parameter latch

The rising edge triggering of this command is valid. When executing the rising edge of input, latch the IDs of the combined axis and two sub-axes of the input/output parameter AxesGroup, as well as the values of the input parameters DecAngle and StopAngle. During command execution, modifying the latched parameters is invalid until another rising edge triggers this command.

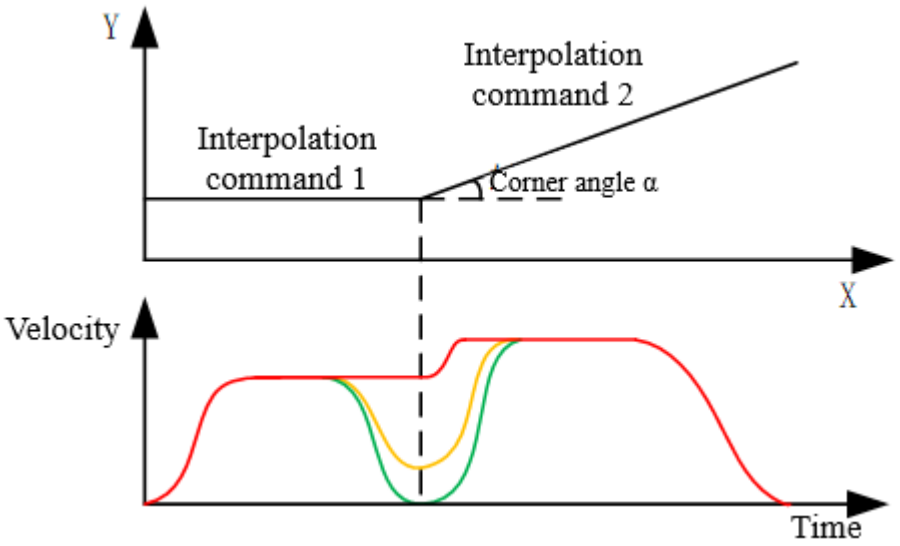
(2) Transfer velocity limit parameters

The critical deceleration angle DecAngle can limit the velocity at the junction of axis group motion commands, and the critical stop angle DecAngle limits the velocity at the junction of axis group motion commands to 0. The critical deceleration angle DecAngle is less than or equal to the critical stop angle DecAngle.

(3) Function description

Set the corner velocity limit parameters during the movement of the axis group, and the velocity at the joint of the interpolation motion command of the axis group is limited to reduce the motion impact at the non-smooth corner joint.

As shown in the following figure, for the speed curve at the junction of two interpolation commands, the speed of the first interpolation command is taken as the velocity at the junction of the two commands.



In the above figure, interpolation motion command 2 is set to be fused at the velocity of interpolation command 1, i.e. the starting point speed of the second interpolation command is set as the command speed of interpolation command 1.

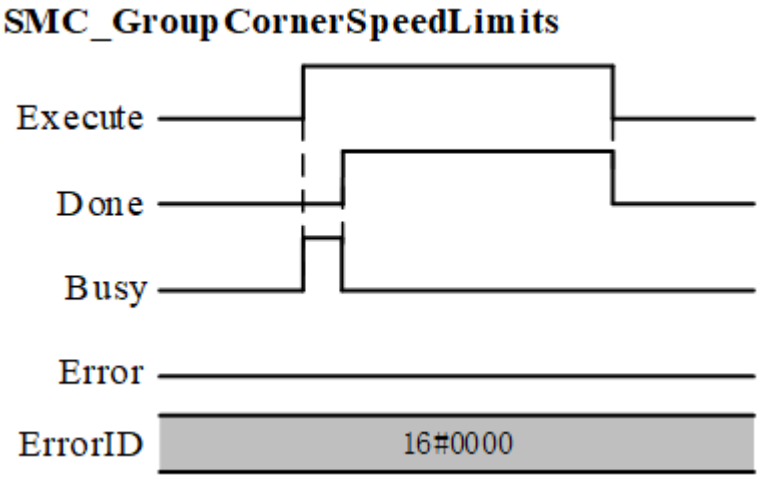
If the set critical deceleration angle and critical stop angle are greater than the turning angle α between the two commands, the connecting speed between the two commands is not limited, and the velocity curve is shown as the red speed curve;

If the set critical deceleration angle is less than the turning angle α between the two commands, and the critical stop angle is greater than the turning angle α between the two commands, the connection velocity between the two commands is limited, and the velocity curve is shown as the orange velocity curve;

If the set critical deceleration angle and critical stop angle are both smaller than the turning angle α between the two commands, the connection velocity between the two commands is limited, and the velocity at the connection point is 0. The green velocity curve is shown in the velocity curve.

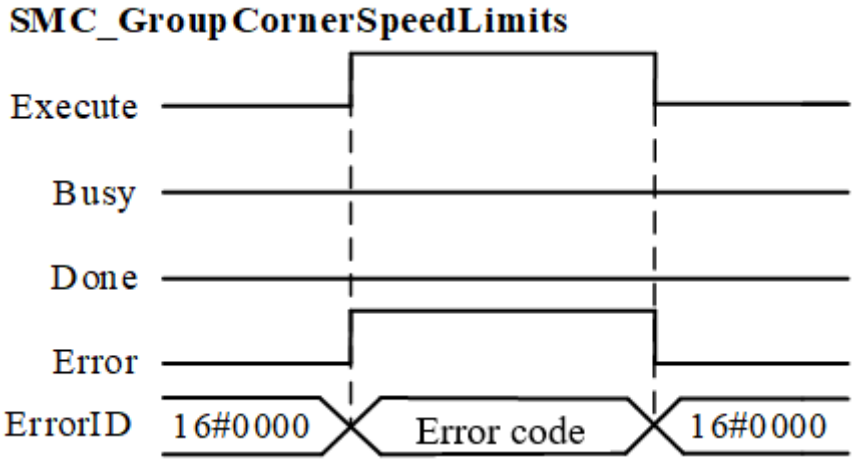
■ Timing diagram

(1) Normal timing diagram



(2) Error pin timing sequence

When this command is executed, if errors such as illegal parameters and illegal axis parameters in the axis group occur, the call of this command fails, with command output pin Error=True and error code ErrorID reporting an abnormal error value. Please refer to the appendix Description of error codes to obtain the cause of the error. When the command input pin Execute goes low, all output pins revert to their Default Values.



- When the abnormal situation of the command is solved, it is necessary to trigger Execute again on the rising edge and start the SMC_SetGroupCornerSpeedLimits function block to complete the setting of corner speed limit parameters of the axis group.

7.9.3 Examples

■ Action example

■ Action introduction

Execute SMC_GroupCornerSpeedLimits command to set the corner speed limit parameters of

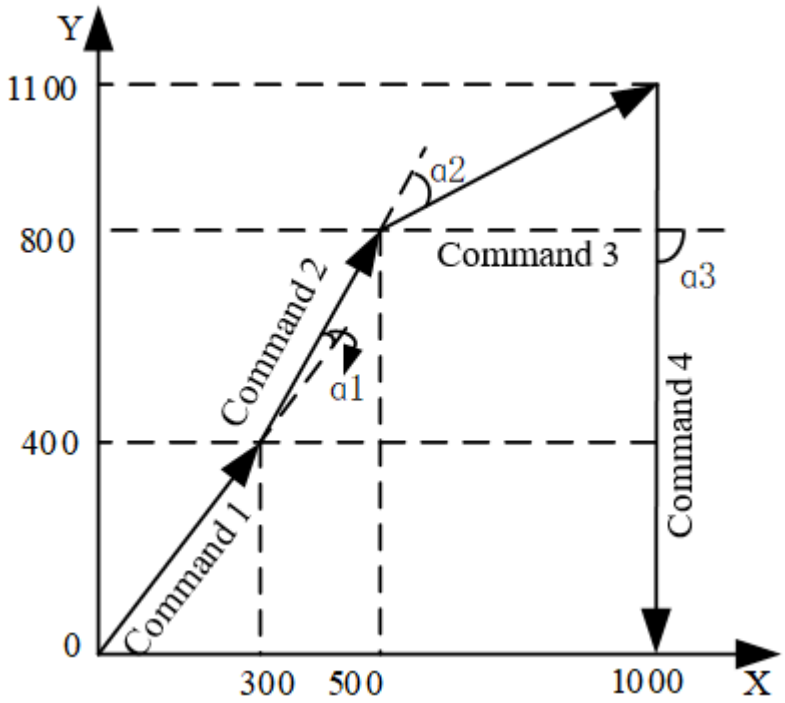
the axis group.

The following example executes 4 linear interpolation commands.

In this example, the buffer mode BufferMode of the linear interpolation command 1 is set to Buffered. The buffer modes of the remaining three interpolation commands are 3: velocity fusion of the previous command, 4: velocity fusion of the next command and 5: high-velocity fusion between adjacent commands.

In this example, when the output Active of the linear interpolation command 1 becomes TRUE, the remaining interpolation commands 2-4 are executed.

■ Schematic Diagram of Action



The linear interpolation command 1 reaches the position (300, 400), the linear interpolation command 2 reaches the position (500, 800), the linear interpolation command 3 reaches the position (1000, 1100), and the linear interpolation command 4 reaches the position (1000, 0) in this order.

The break angle α1 of interpolation command 1 and 2 is about 10°, the break angle of interpolation command 2 and 3 is 32°, and the break angle of interpolation command 3 and 4 is 121°.

Run SMC_GroupCornerSpeedLimits to set the corner velocity limit parameters of the axis group. Set the deceleration angle DecAngle to 20° and the stop angle StopAngle to 60°.

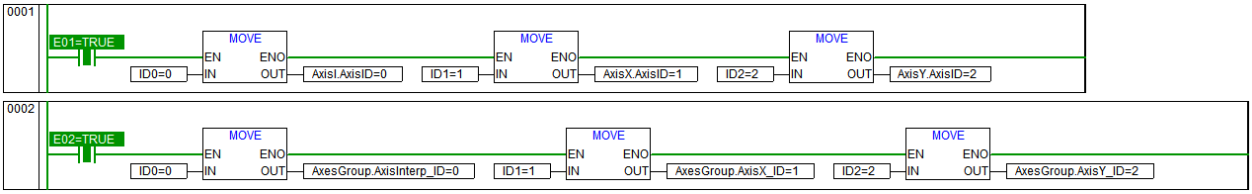
■ LD diagram program

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group

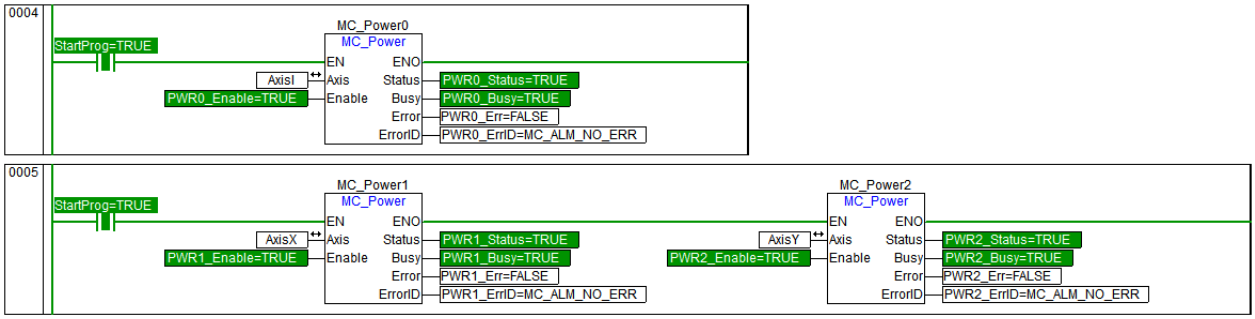
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enabling operation.
Power1_Status	BOOL	FALSE	Variables whose Sub-axis-axis X enable is successful. When the Sub-axis X is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variables whose Sub-axis-axis Y enable is successful. When the Sub-axis Y is enabled successfully, this variable becomes TRUE.
Power0_Status	BOOL	FALSE	Variable for successful combination of axes enabling. This variable turns to TRUE when the combined axis is enabled successfully.
GrpEnableStatus	BOOL	TRUE	When this variable is TRUE, the axis group is enabled.
AxisHomed	BOOL	FALSE	Variable for axis homing completion. If this variable is TRUE, the return to the home each axis in the axis group is completed.
Rel2D1_Active	BOOL	FALSE	The variable sent by the axis group relative to the Active pin of interpolation command 1. When this variable becomes TRUE, the axis group executes the interpolation command.

(1) Set the IDs of joint and sub-axes in the axis group. When defining an axis group, you need to define three axes in the axis group first, and match the IDs of the defined axis numbers of the three axes with those of the combined and sub-axes in the axis group. See sections 0001 and 0002 below.

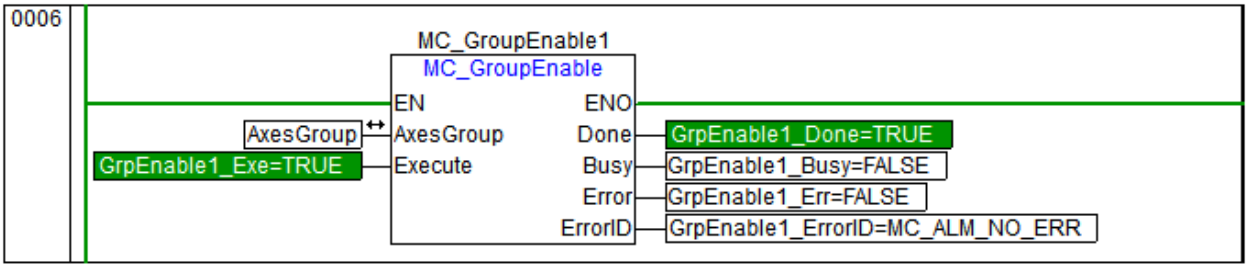


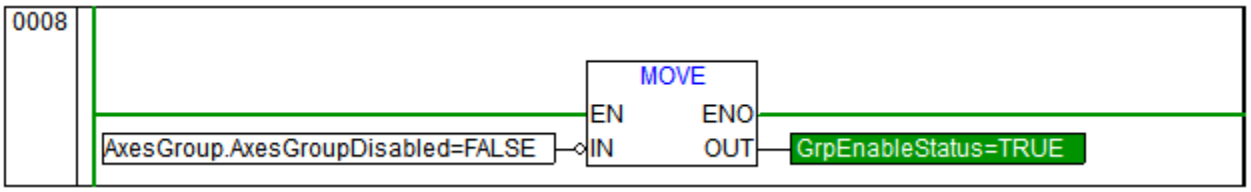
(2) Axis initialization. Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group respectively, and set the combined axis as 0: Virtual and the Sub-axis axis type as 50: EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to its sections.

(3) Axis enable. After each axis in the axis group is ready (StartProg =TRUE), call MC_Power to enable the axes of the combined axis and two sub-axes of the axis group. See Sections 0004 to 0005.

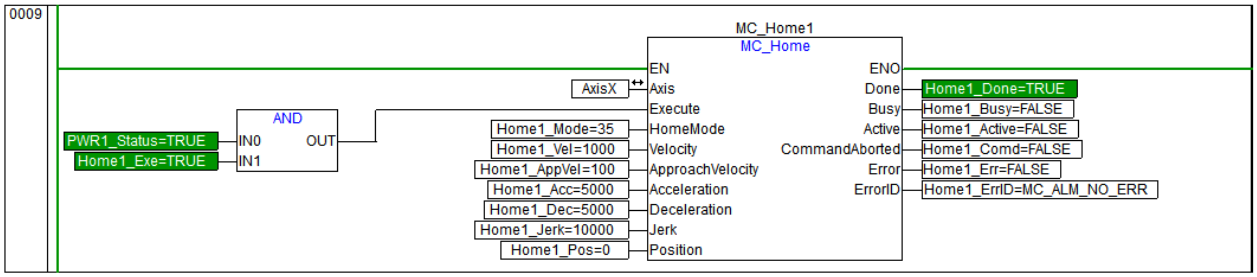


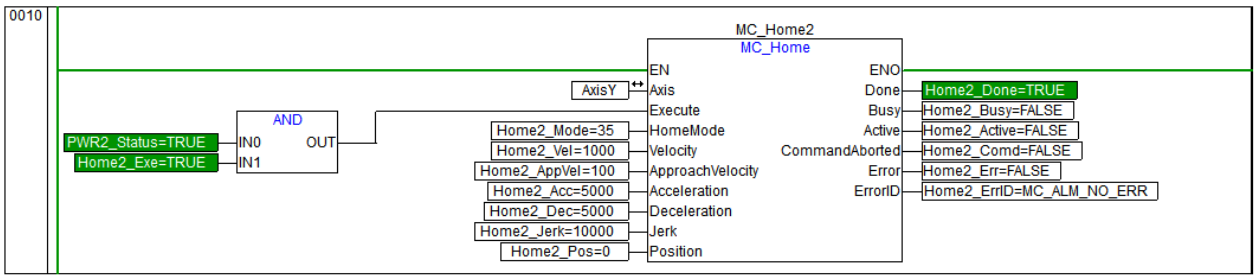
(4) Axis group enable. Call MC_GroupEnable to enable the axis group. If AxesGroupDisabled in the axis group variables is FALSE, the axis group is enabled successfully. See Section 0006. After the axis group is enabled successfully, invert AxesGroupDisabled and assign it to GrpEnableStatus variable, see Section 0008.

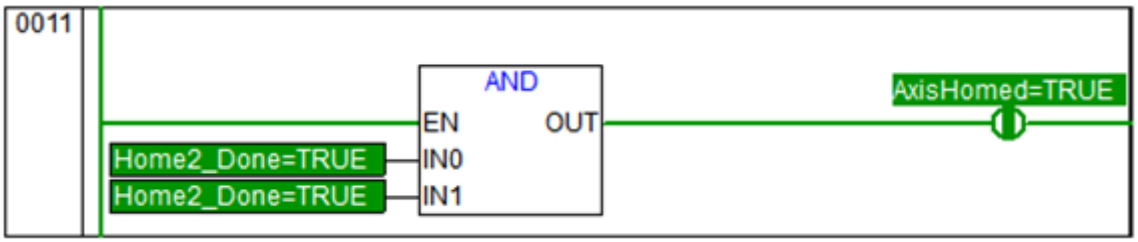




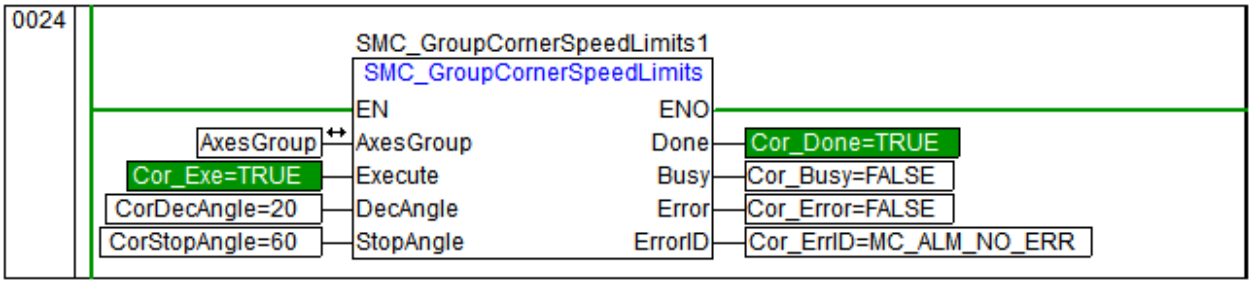
(5) After the two sub-axes in the axis group are enabled, if the home is not determined, perform homing action. When homing is complete, the homing completed variable AxisHomed is set to TRUE. See Section 0009-0011.



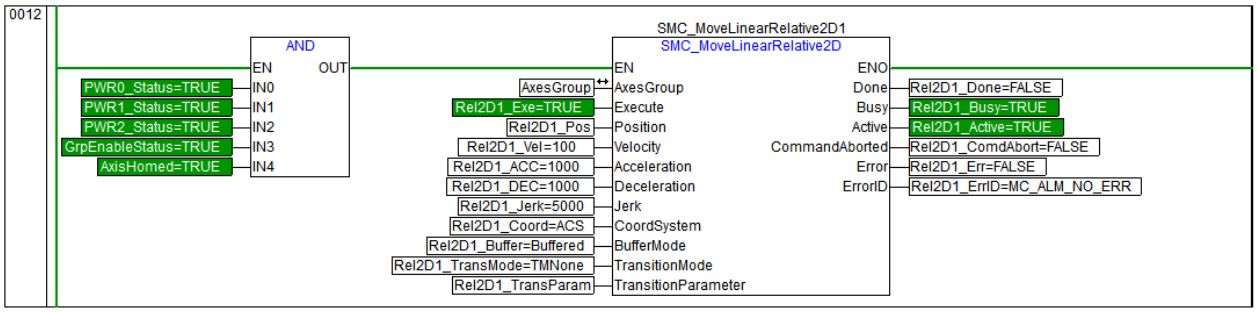


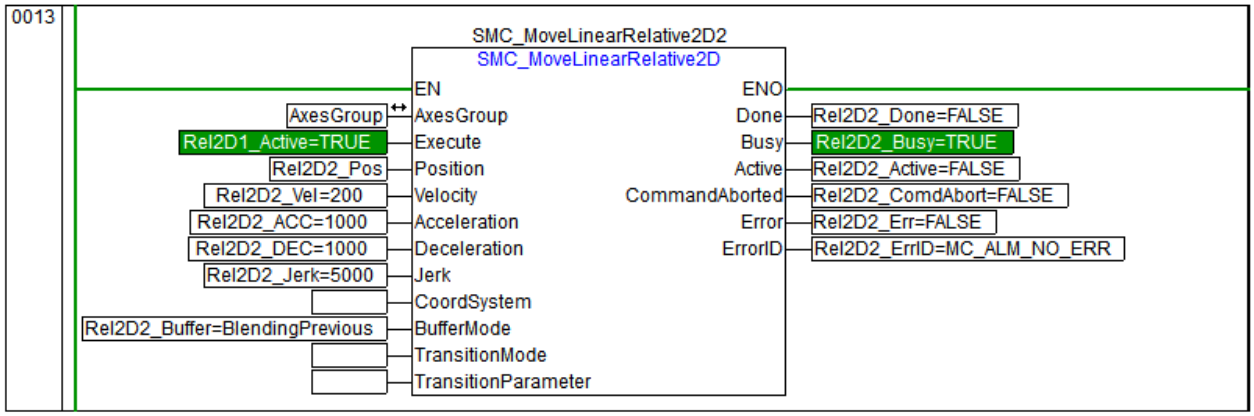


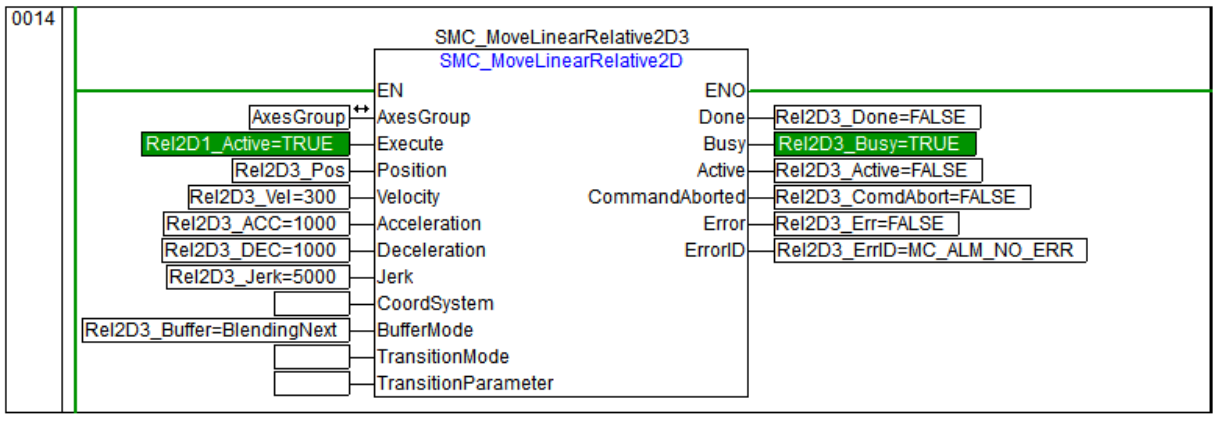
(6) In section 0024, execute the SMC_GroupCornerSpeedLimits command to set the corner velocity limit parameters.

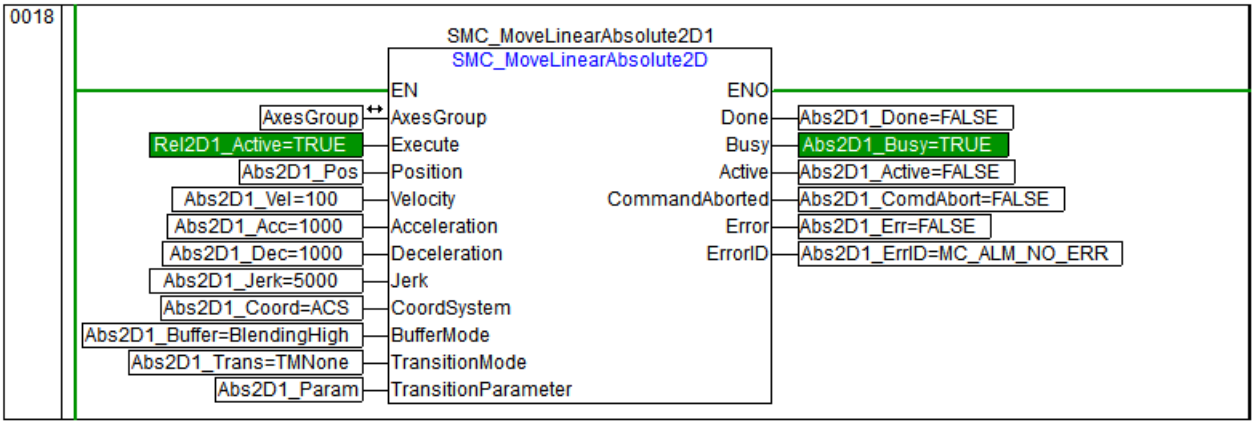


(7) In section 0011, after the axis homing is completed, start executing the interpolation motion command. Execute command 1. When Rel2D1_Active of command 1 is TRUE, commands 2-4 are sent in sequence. See paragraphs 0012-0014 and 0018 below.









■ ST language program

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enabling operation.
Power0_Status	BOOL	FALSE	Variable for successful combination of axes enabling. This variable becomes TRUE when the combined axis is enabled successfully.
Power1_Status	BOOL	FALSE	Variables whose Sub-axis-axis X enable is successful. When the Sub-axis X is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variables whose Sub-axis-axis Y enable is successful. When the Sub-axis Y is enabled successfully, this variable becomes TRUE.
GrpEnableStatus	BOOL	TRUE	When this variable is TRUE, the axis group is enabled.
AxisHomed	BOOL	FALSE	Variable for axis homing completion. If this variable is TRUE, the homing of each axis in the axis group is completed.
InitFlag	BOOL	FALSE	If this variable is TRUE, the initialization parameter setting is completed.
Rel2D1_Exe	BOOL	FALSE	Rising-edge trigger variable of relative interpolation command 1 delivered.
Rel2D1_Active	BOOL	FALSE	The variable sent relative to the Active pin of interpolation command 1. When this variable becomes TRUE, the axis group executes the interpolation command.

(1) Set the IDs of combined axis and sub-axes in the axis group. Set the axis ID of the defined combined axis AxisI, sub-axes AxisX and AxisY.

```
IF Switch1 THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
END_IF
```

(2) Set command motion parameters

```
IF FALSE = InitFlag THEN
(* Parameter setting of linear interpolation command 1 for relative position. Default Values for unset input parameters *)
   Rel2D1_DEC:=1000;,
   Rel2D1_Jerk:=5000;,
   Rel2D1_Buffer:=Buffered;
(* Parameter setting of relative position linear interpolation command 2. Default Values for unset input parameters *)
   Rel2D2_Pos[0]:= 200;
   Rel2D2_Pos[1]:=400;
   Rel2D2_Vel:=200;
   Rel2D2_ACC:=1000;,
   Rel2D2_DEC:=1000;,
   Rel2D2_Jerk:=5000;,
   Rel2D2_Buffer:= BlendingPrevious;
(* Parameter setting of linear interpolation command 3 for relative position. Default Values for unset input parameters *)
   Rel2D2_Pos[0]:= 500;
   Rel2D2_Pos[1]:=300;
   Rel2D2_Vel:=300;
   Rel2D2_ACC:=1000;,
   Rel2D2_DEC:=1000;,
   Rel2D2_Jerk:=5000;,
   Rel2D2_Buffer:= BlendingNext;
(* Parameter setting of absolute position linear interpolation command 4. Default Values for unset input parameters *)
   Abs2D1_Pos[0]:= 1000;
   Abs2D1_Pos[1]:= 0;
   Abs2D1_Vel:=100;
   Abs2D1_ACC:=1000;,
   Abs2D1_DEC:=1000;,
   Abs2D1_Jerk:=5000;,
   AbsD1_Buffer:= BlendingHigh;
(*corner velocity limit parameter setting *)
   CorDecAngle:=20;
   CorStopAngle:=60;
InitFlag:=TRUE; (* Parameter initialization flag *)
END_IF
```

(3) Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group. Set the axis type of axis combination to 0: Virtual; set the axis types of axis group AxisX/AxisY to 50: EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to its command section.

(4) Execute the MC_Power command to enable the three axes in the axis group.

```
IF StartPro THEN
MC_Power0(*Combined axis enable*)
```

```
    Enable := PWR0_Enable,
    Axis := AxisI,
    Status=> PWR0_Status,
    Busy=> PWR0_Busy,
    Error=> PWR0_Err,
    ErrorID=> PWR0_ErrID);
MC_Power1((*Sub-axis AxisX enable*)
    Enable := PWR1_Enable,
    Axis := AxisX,
    Status=> PWR1_Status,
    Busy=> PWR1_Busy,
    Error=> PWR1_Err,
    ErrorID=> PWR1_ErrID);
MC_Power2 (*Sub-axis AxisY enable*)
    Enable := PWR2_Enable,
    Axis := AxisY,
    Status=> PWR2_Status,
    Busy=> PWR2_Busy,
    Error=> PWR2_Err,
    ErrorID=> PWR2_ErrID);
END_IF
```

(5) Execute axis group enable.

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
GrpEnableStatus:= AxesGroup. AxesGroupDisabled;
```

(6) Execute the homing command of sub-axis in the axis group to determine the home. See its introduction section for the usage of MC_Home command.

```
IF  PWR1_Status THEN
    MC_Home1(
    Execute := Home1_Exe,
    HomeMode := Home1_Mode,
    Velocity := Home1_Vel,
    ApproachVelocity := Home1_AppVel,
    Acceleration := Home1_Acc,
    Deceleration := Home1_Dec,
    Jerk := Home1_Jerk,
    Position := Home1_Pos,
    Axis := AxisX,
    Done=> Home1_Done,
    Busy=> Home1_Busy,
```

```
  Active=> Home1_Active,
  CommandAborted=> Home1_Comd,
  Error=> Home1_Err,
  ErrorID=> Home1_ErrID);
  MC_Home2(
  Execute := Home2_Exe,
  HomeMode := Home2_Mode,
  Velocity := Home2_Vel,
  ApproachVelocity := Home2_AppVel,
  Acceleration := Home2_Acc,
  Deceleration := Home2_Dec,
  Jerk := Home2_Jerk,
  Position := Home2_Pos,
  Axis := AxisY,
  Done=> Home2_Done,
  Busy=> Home2_Busy,
  Active=> Home2_Active,
  CommandAborted=> Home2_Comd,
  Error=> Home2_Err,
  ErrorID=> Home2_ErrID);
END_IF
IF  Home1_Done  AND Home2_Done THEN
  AxisHomed:=TRUE;
END_IF
```

(7) Execute the corner velocity limit parameter setting command.

```
SMC_GroupCornerSpeedLimits1(
Execute := Cor_Exe,
DecAngle := Cor_DecAngle,
StopAngle := Cor_StopAngle,
AxesGroup := AxesGroup,
Done=> Cor_Done,
Busy=> Cor_Busy,
Error=> Cor_Err,
ErrorID=> Cor_ErrID);
```

(8) After the axis homing, execute the interpolation motion command. Execute command 1. When Rel2D1_Active of command 1 is TRUE, commands 2-4 are sent in sequence.

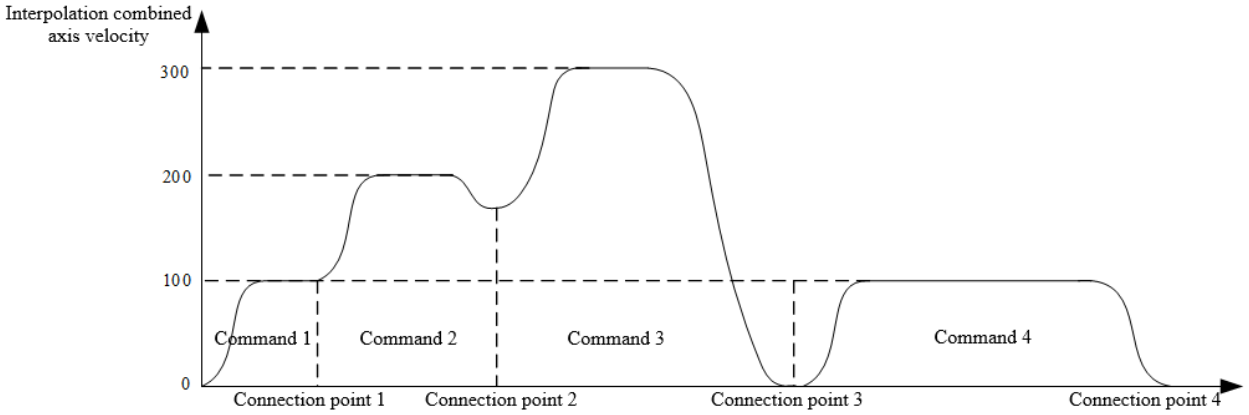
```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus AND AxisHomed THEN
SMC_MoveLinearRelative2D1(*interpolation command 1*)
  Execute := Rel2D1_Exe,
  Position := Rel2D1_Pos,
  Velocity := Rel2D1_Vel,
  Acceleration := Rel2D1_ACC,
  Deceleration := Rel2D1_DEC,
  Jerk := Rel2D1_Jerk,
  CoordSystem := Rel2D1_Coord,
```

```
    BufferMode := Rel2D1_Buffer,
    TransitionMode := Rel2D1_Trans,
    TransitionParameter := Rel2D1_Param,
    AxesGroup := AxesGroup,
    Done=> Rel2D1_Done,
    Busy=> Rel2D1_Busy,
    Active=> Rel2D1_Active,
    CommandAborted=> Rel2D1_ComdAbort,
    Error=> Rel2D1_Err,
    ErrorID=> Rel2D1_ErrID);
SMC_MoveLinearRelative2D2(*interpolation command 2*)
    Execute := Rel2D1_Active,
    Position := Rel2D2_Pos,
    Velocity := Rel2D2_Vel,
    Acceleration := Rel2D2_ACC,
    Deceleration := Rel2D2_DEC,
    Jerk := Rel2D2_Jerk,
    CoordSystem := Rel2D2_Coord,
    BufferMode := Rel2D2_Buffer,
    TransitionMode := Rel2D2_Trans,
    TransitionParameter := Rel2D2_Param,
    AxesGroup := AxesGroup,
    Done=> Rel2D2_Done,
    Busy=> Rel2D2_Busy,
    Active=> Rel2D2_Active,
    CommandAborted=> Rel2D2_ComdAbort,
    Error=> Rel2D2_Err,
    ErrorID=> Rel2D2_ErrID);
SMC_MoveLinearRelative2D3(*interpolation command 3*)
    Execute := Rel2D1_Active,
    Position := Rel2D3_Pos,
    Velocity := Rel2D3_Vel,
    Acceleration := Rel2D3_ACC,
    Deceleration := Rel2D3_DEC,
    Jerk := Rel2D3_Jerk,
    CoordSystem := Rel2D3_Coord,
    BufferMode := Rel2D3_Buffer,
    TransitionMode := Rel2D3_Trans,
    TransitionParameter := Rel2D3_Param,
    AxesGroup := AxesGroup,
    Done=> Rel2D3_Done,
    Busy=> Rel2D3_Busy,
    Active=> Rel2D3_Active,
    CommandAborted=> Rel2D3_ComdAbort,
    Error=> Rel2D3_Err,
```

```
    ErrorID=> Rel2D3_ErrID);
SMC_MoveLinearAbsolute2D1(*interpolation command 4*)
    Execute := Rel2D1_Active,
    Position := Abs2D1_Pos,
    Velocity := Abs2D1_Vel,
    Acceleration := Abs2D1_ACC,
    Deceleration := Abs2D1_DEC,
    Jerk := Abs2D1_Jerk,
    CoordSystem := Abs2D1_Coord,
    BufferMode := Abs2D1_Buffer,
    TransitionMode := Abs2D1_Trans,
    TransitionParameter := Abs2D1_Param,
    AxesGroup := AxesGroup,
    Done=> Abs2D1_Done,
    Busy=> Abs2D1_Busy,
    Active=> Abs2D1_Active,
    CommandAborted=> Abs2D1_ComdAbort,
    Error=> Abs2D1_Err,
    ErrorID=> Abs2D1_ErrID);
END_IF
```

The set corner deceleration angle is 20° and the stop angle is 60°. The rotation angles of interpolation command 1 and command 2 are less than the deceleration angle, and the velocity at the junction of the two commanded rotation angles (see connecting point 1 in the figure below) is not limited; The rotation angles of interpolation command 2 and command 3 are greater than the deceleration angle and less than the stop angle, and the speed at the junction of the two commanded rotation angles (see connecting point 2 in the figure below) is decelerated; The rotation angles of interpolation command 3 and command 4 are greater than the stop angle, and the speed at the junction of the two commanded rotation angles (see the connecting point 3 in the figure below) is reduced to 0;

The combined-axis speed curve of the interpolation motion command is as follows:

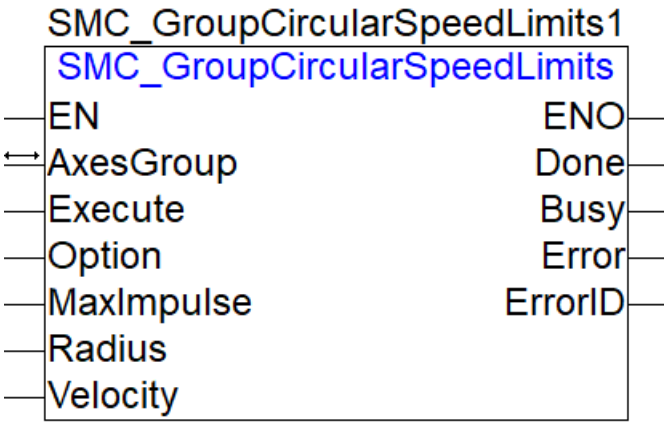


7.9.4 Precautions

- When this command is called, the deceleration angle must be less than or equal to the stop angle.
- After the corner velocity limit parameter is successfully set by this command, the falling edge cannot cancel the parameter limit. The user needs to call the command again to set parameter limits.

7.10 SMC_GroupCircularSpeedLimits (Set Axes Group Circular Speed Limits)

This command sets the arc speed limit parameters during the movement of the axis group to reduce the motion impact at the arc turning.



7.10.1 Parameter

Parameter Type	Title	Description	Empty or Not	Default Value	Scope	Data Type

IN_OUT	AxesGroup	Axis group name	No	-	-	AXES_GROUP_REF
INPUT	Execute	Rising edge trigger	No	-	TRUE/FALSE	BOOL
	Option	Parameter setting mode: 0: Speed limit by setting the maximum impact value at arc turning; 1: The speed is limited by setting the maximum speed of the specified arc radius.	YES	0	0/1	USINT
	MaxImpulse	Valid when Option is 0: Maximum impact value, in the same unit as jerk	YES	1E100	Positive value	LREAL
	Radius	Valid when Option is 1: Specified radius of circular arc	YES	1	Positive value	LREAL
	Velocity	Valid when Option is 1: Maximum interpolation speed for specified arc radius	YES	1E30	Positive value	LREAL
OUTPUT	Done	Completed	YES	FALSE	TRUE/FALSE	BOOL
	Busy	Busy flag	YES	FALSE	TRUE/FALSE	BOOL
	Error	Command error flag	YES	FALSE	TRUE/FALSE	BOOL
	ErrorID	Command error code	YES	0	-	SMC_ERROR

7.10.2 Functions

Set the arc speed limit parameters during the movement of the axis group to reduce the motion impact at the arc turning.

■ command function details

(1) Input parameter latch

The rising edge triggering of this command is valid. When executing the rising edge of input, latch the IDs of the combined axis and two sub-axes of the input/output parameter AxesGroup, input parameters Option, MaxImpulse, Radius and Velocity. During command execution, modifying the latched parameters is invalid until another rising edge triggers this command.

(2) Parameter Description

□ Option: arc speed limit parameter setting mode. This command supports 2 parameter setting modes. When Option is 0, the speed is limited by setting the maximum impact value at the arc turning; when Option is 1, the speed is limited by setting the maximum speed of the specified arc radius.

□ When the parameter setting mode Option is 0, the MaxImpulse parameter is valid and is the set maximum impact value. The commanded arc radius Radius and the maximum interpolation Velocity of the commanded arc radius are invalid.

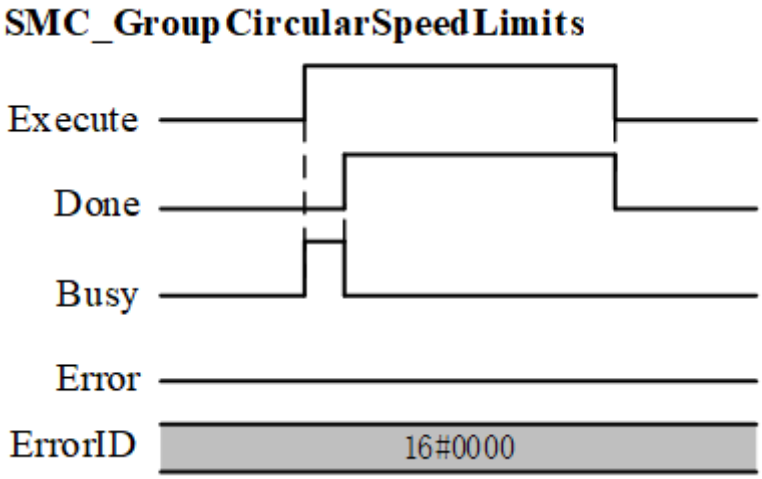
□ When the parameter setting mode Option is 1, the commanded arc radius Radius and the maximum interpolation velocity of the commanded arc radius Velocity are valid, but the maximum impact value MaxImpulse parameter is valid.

□ MaxImpulse: maximum impact value, in the same unit as jerk. This parameter takes effect when Option is 0.

- When the parameter setting method Option is 1, recalculate MaxImpulse according to the maximum interpolation velocity Velocity and curvature of the specified arc radius. MaxImpulse =Curvature² * dSpeed³. Curvature=1/Radius.
- The speed limit parameters entered are all positive values.

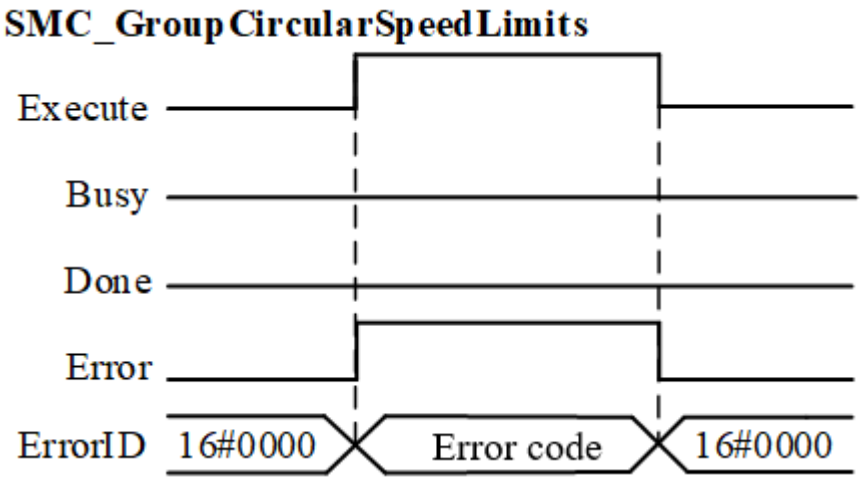
■ Timing diagram

(1) Normal timing diagram



(2) Error pin timing

During the execution of this command, if there are error such as illegal parameter, 0: Unused (unused axis) for sub-axis and illegal ID of each axis in the axis group, the call of this command fails, with command output pin Error=True and error code ErrorID reporting an error value. Please refer to the appendix Error Code Description to obtain the cause of the error. When the command input pin Execute goes low, all output pins revert to their Default Values.



- When the abnormal situation of the command is solved, it is necessary to trigger Execute again on the rising edge and start the SMC_GroupCircularLimits functional block to complete the setting of arc

speed limit parameters.

7.10.3 Examples

- Action example

(1) Execute the circular interpolation command SMC_MoveCircularRelative2D. The speed in this command is set as 5000, the acceleration and deceleration are both 50000, and the jerk is 500000 to control the movement of axis group.

(2) Execute the SMC_GroupCircularSpeedLimits command to set arc speed limit parameters.

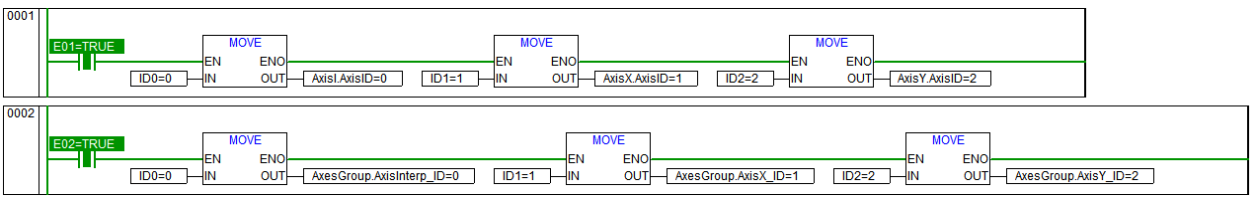
- LD diagram program

Set the input parameter Option of arc speed limit command to 0, and set MaxImpulse to 10000 considering the impact of motion impact at arc turning on equipment safety. At this time, the jerk of the circular interpolation command in the axis group is limited, and the maximum jerk is 10000.

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enabling operation.
Power1_Status	BOOL	FALSE	Variables whose Sub-axis-axis X enable is successful. When the Sub-axis X is enabled successfully, this variable becomes TRUE.
Power2_Status	BOOL	FALSE	Variables whose Sub-axis-axis Y enable is successful. When the Sub-axis Y is enabled successfully, this variable becomes TRUE.
Power0_Status	BOOL	FALSE	Variable for successful combination of axes enabling. This variable turns TRUE when the combined axis is enabled successfully.
GrpEnableStatus	BOOL	TRUE	When this variable is TRUE, the axis group is enabled.
GrpEnable1_Exe	BOOL	FALSE	Axis group enable rising edge trigger variable.
Circ2D1_Exe	BOOL	FALSE	Rising edge trigger variable of circular interpolation command.
CirSpeed_Exe	BOOL	FALSE	Rising edge trigger variable of arc speed limit parameter command.

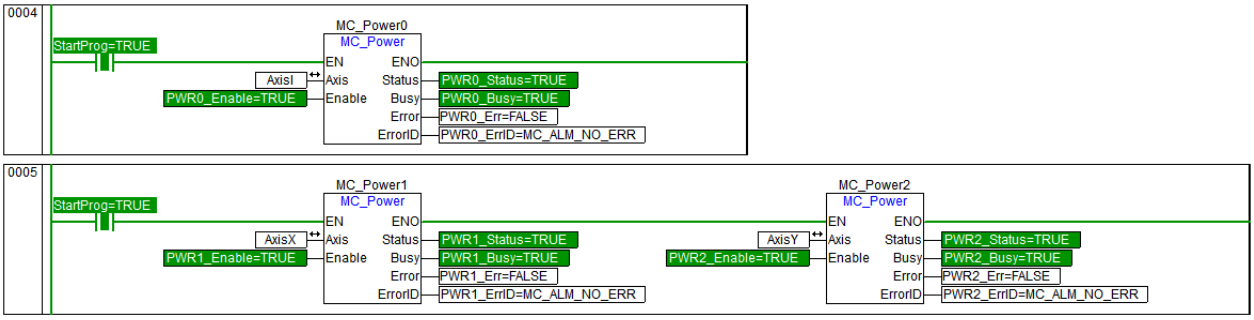
(1) Set the IDs of joint and sub-axes in the axis group. When defining an axis group, you need to define three axes in the axis group first, and match the IDs of the defined axis numbers of the three axes with those of the combined and sub-axes in the axis group. See sections 0001 and 0002 below.



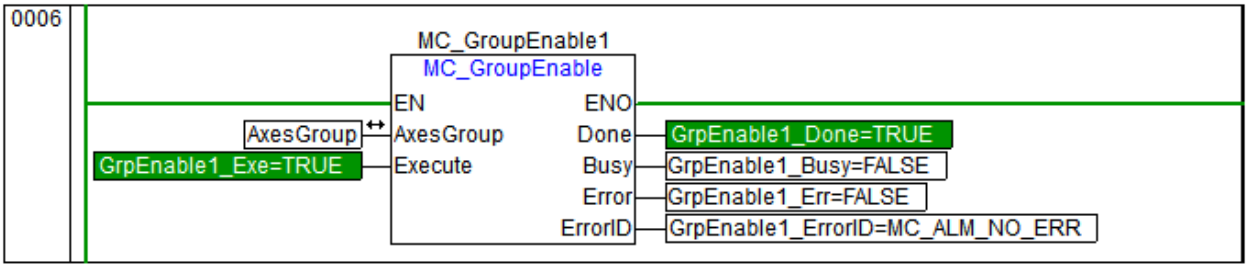
(2) Axis initialization. Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group respectively, and set the combined axis as 0: Virtual and the Sub-

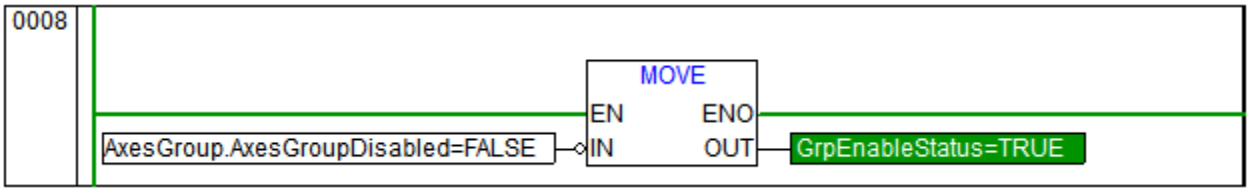
axis axis type as 50: EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to its chapters.

(3) Axis enable. After each axis in the axis group is ready (StartProg =TRUE), call MC_Power to enable the axes of the combined axis and two sub-axes of the axis group. See Sections 0004 to 0005.

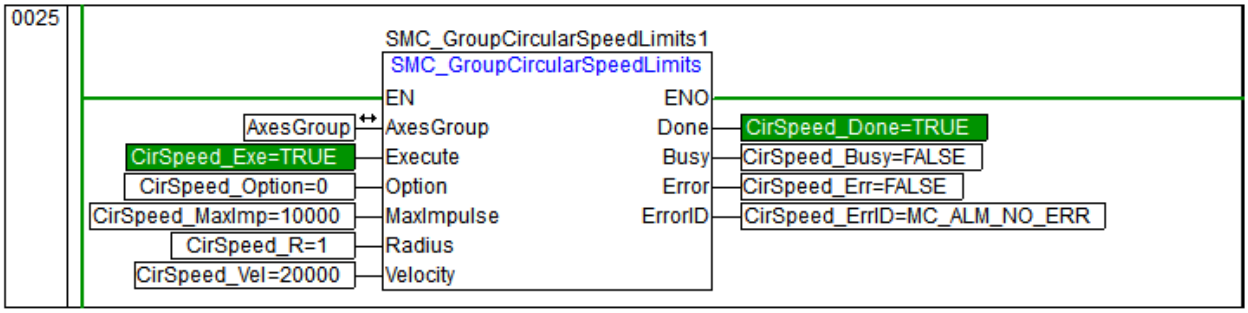


(4) Call MC_GroupEnable to enable the axis group. If AxesGroupDisabled in the axis group variables is FALSE, the axis group is enabled successfully. See Section 0006 and 0008. After the axis group is enabled successfully, reverse AxesGroupDisabled and assign it to GrpEnableStatus variable.

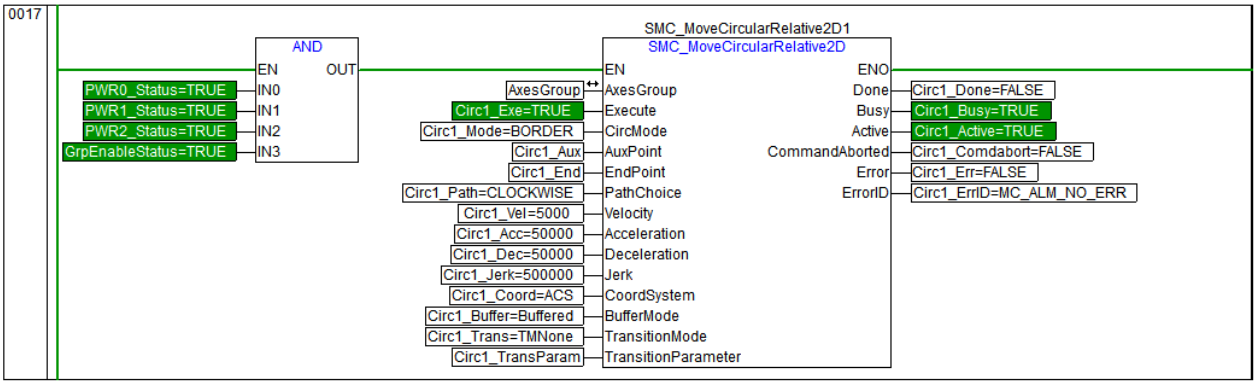




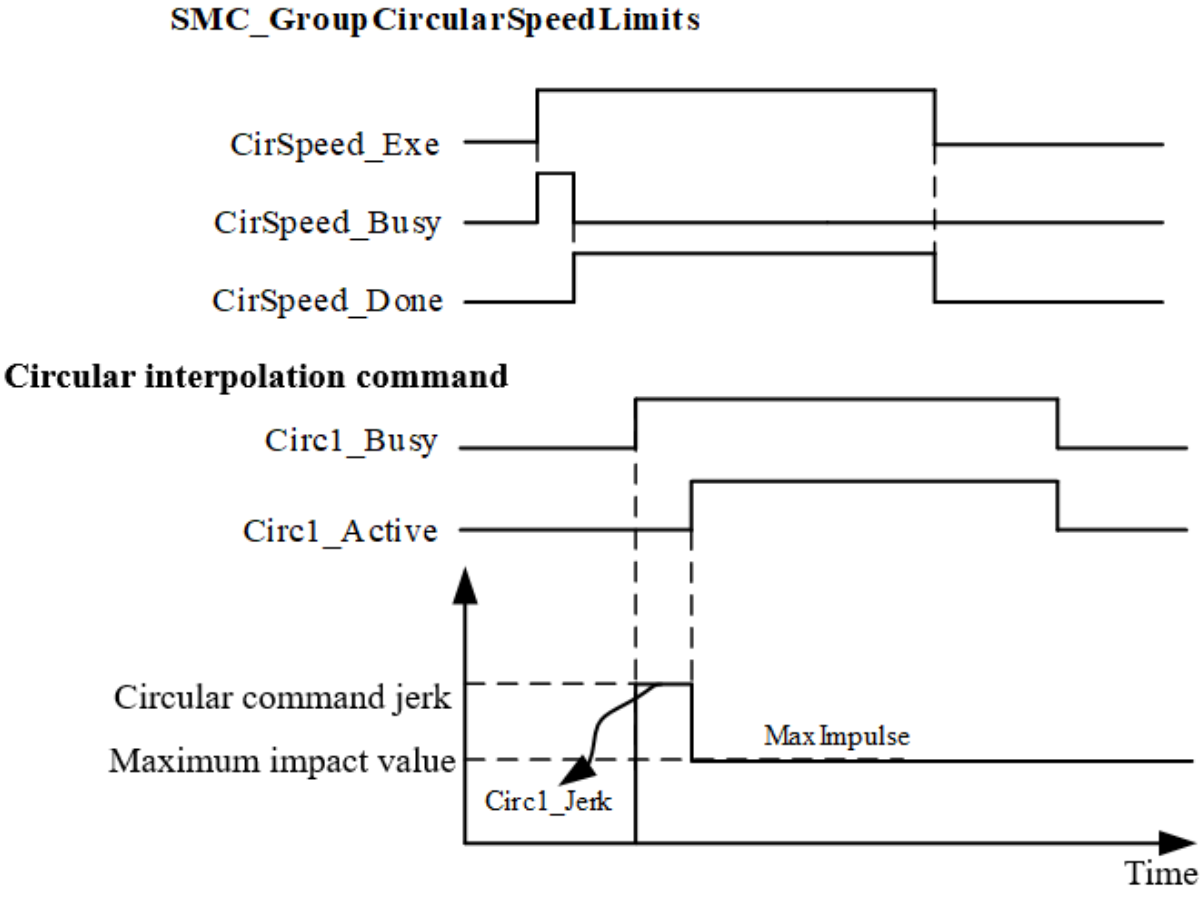
(5) In section 0025, execute the SMC_GroupCircularSpeedLimits command to set the limit value of the arc speed limit parameter of the axis group. The actual jerk of the arc interpolation command is the set MaxImpulse (10000).



(6) In section 0017, execute the circular interpolation command.



After the parameters are set, the jerk parameter of circular interpolation command is limited as follows:



■ ST language

Execute the SMC_GroupCircularSpeedLimits command, set the input parameter Option to 1, the arc radius to 1000, and the maximum interpolation velocity of the arc to 1000. Calculate MaxImpulse = (1/1000)2 * 10003 = 1000.

Execute the circular interpolation command SMC_MoveCircularRelative2D. In this command, the arc radius is 1000, the command speed is set to 5000, both acceleration and deceleration are 50000, and jerk is 500000. The speed and jerk of the axis group interpolation motion are limited. The speed of the axis combination axis is 1000, and the jerk is the limit value 1000 (MaxImpulse).

Description of Primary Variables

Variable Name	Variable Type	Initial Value	Notes
AxesGroup	AXES_GROUP_REF	-	Axis group name
AxisI	AXIS_REF	-	Axis variables of combined axis in axis group
AxisX	AXIS_REF	-	Axis variables of sub-axis X in axis group
AxisY	AXIS_REF	-	Axis variables of sub-axis Y in axis group
StartProg	BOOL	FALSE	If this variable is TRUE, the axes in the axis group are ready to perform the axis enabling operation.
Power0_Status	BOOL	FALSE	Variables whose sub-axis X enable is successful. When the sub-axis X is enabled successfully, this variable becomes TRUE.
Power1_Status	BOOL	FALSE	Variables whose sub-axis Y enable is successful. When the sub-axis Y is enabled successfully, this variable becomes TRUE.

Power2_Status	BOOL	FALSE	Variable for successful combination of axes enabling. This variable turns TRUE when the combined axis is enabled successfully.
InitFlag	BOOL	FALSE	If this variable is TRUE, the initialization parameter setting is completed.
GrpEnableStatus	BOOL	TRUE	When this variable is TRUE, the axis group is enabled.
GrpEnable1_Exe	BOOL	FALSE	Axis group enable rising edge trigger variable.
Circ2D1_Exe	BOOL	FALSE	Rising edge trigger variable of circular interpolation command.
CirSpeed_Exe	BOOL	FALSE	Rising edge trigger variable of arc speed limit parameter command.

(1) Set the IDs of joint and sub-axes in the axis group. Set the axis ID of the defined combined axis AxisI, sub-axes AxisX and AxisY.

```
IF Switch1 THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
END_IF
```

(2) Set command motion parameters

```
IF FALSE = InitFlag THEN
(* Parameter setting of circular interpolation command 3. Default Values for unset input parameters *)
   Circ1_Mode:= BORDER;
   Circ1_Aux [0] :=1000;
   Circ1_Aux [1] :=1000;
   Circ1_End [0] :=2000;
   Circ1_End [1] :=0;
   Circ1_Path:= CLOCKWISE;
   Circ1_Vel:=2000;
   Circ1_ACC:=5000;
   Circ1_DEC:=5000;
   Circ1_Jerk:=50000;
   Circ1_Buffer:=Buffered;
   CirSpeed_Option:=1;
   CirSpeed_R:=1000;
   CirSpeed_Vel:=1000;
   InitFlag:=TRUE;
END_IF
```

(3) Call SMC_AxisInit command to complete the axis initialization configuration of three axes in the axis group. Set the axis type of axis combination to 0: Virtual; set the axis types of axis group AxisX/AxisY to 50: EtherCAT_Position. For the usage of SMC_AxisInit command, please refer to its command section.

(4) Execute the MC_Power command to enable the three axes in the axis group.

```
IF StartPro THEN
MC_Power0(*Combined axis enable*)
   Enable := PWR0_Enable,
```

```
   Axis := AxisI,
   Status=> PWR0_Status,
   Busy=> PWR0_Busy,
   Error=> PWR0_Err,
   ErrorID=> PWR0_ErrID);
MC_Power1((*Sub-axis AxisX enable*)
   Enable := PWR1_Enable,
   Axis := AxisX,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
   Error=> PWR1_Err,
   ErrorID=> PWR1_ErrID);
MC_Power2 (*Sub-axis AxisY enable*)
   Enable := PWR2_Enable,
   Axis := AxisY,
   Status=> PWR2_Status,
   Busy=> PWR2_Busy,
   Error=> PWR2_Err,
   ErrorID=> PWR2_ErrID);
END_IF
```

(5) Execute axis group enable

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
GrpEnableStatus:= AxesGroup. AxesGroupDisabled;
```

(6) Execute interpolation motion command

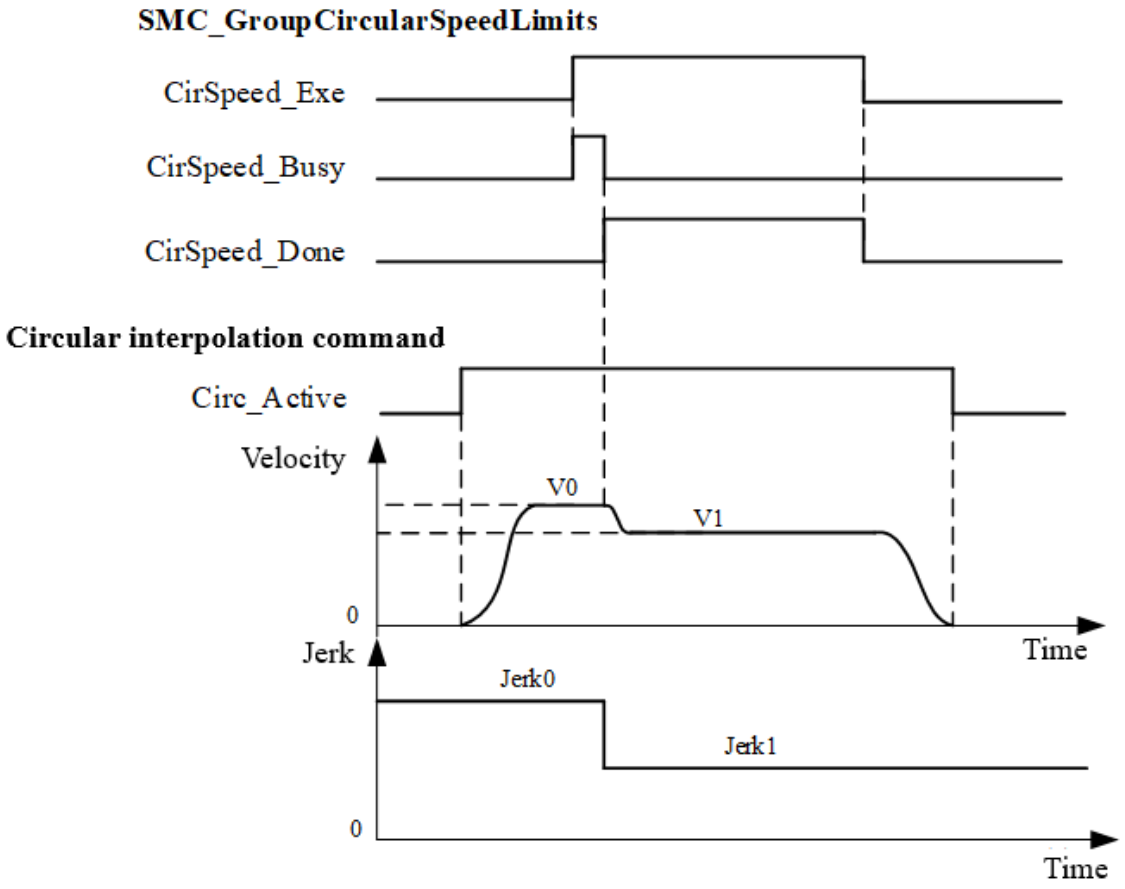
```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus THEN
SMC_MoveCircularRelative2D1(*interpolation command3*)
   Execute := Rel2D1_Active,
   CircMode := Circ1_Mode ,
   AuxPoint := Circ1_Aux,
   EndPoint := Circ1_End,
   PathChoice:= Circ1_Path,
   Velocity := Circ1_Vel,
   Acceleration := Circ1_ACC,
   Deceleration := Circ1_DEC,
   Jerk := Circ1_Jerk,
   CoordSystem := Circ1_Coord,
   BufferMode := Circ1_Buffer,
   TransitionMode := Circ1_ Trans,
```

```
TransitionParameter := Circ1_Param,
AxesGroup := AxesGroup,
Done=> Circ1_Done,
Busy=> Circ1_Busy,
Active=> Circ1_Active,
CommandAborted=> Circ1_Abort,
Error=> Circ1_Err,
ErrorID=> Circ1_ErrID);
```

(7) Execute the SMC_GroupCircularSpeedLimits command to set the limit value of the arc speed limit parameter of the axis group.

```
SMC_GroupCircularSpeedLimits1(
Execute := CirSpeed_Exe,
Option := CirSpeed_Option,
MaxImpulse := CirSpeed_MaxImp,
Radius := CirSpeed_R,
Velocity := CirSpeed_Vel,
AxesGroup := AxesGroup,
Done=> CirSpeed_Done
Busy=> CirSpeed_Busy,
Error=> CirSpeed_Err,
ErrorID=> CirSpeed_ErrID);
```

After the arc speed limit parameter is set, the interpolation command motion parameters are limited. The interpolated command speed and jerk curves are as follows:



7.10.4 Precautions

- When this command is called, the input parameters MaxImpulse, Radius and Velocity are positive.
- After the maximum parameter is successfully set by this command, the falling edge cannot cancel the parameter limit. The user needs to call this command again to set the parameter limit value.

Chapter 8 APPENDICES

8.1 Error Code Description

Function Block Error Code Description

Error code	Error name	Error Description
0	MC_ALM_NO_ERR	No error
2000	MC_ERR_AXIS_TYPE_UNUSED	The axis type is unused/there are axes in the axis group that are not used
2001	MC_ERR_AXISID_INVALID	Illegal axis ID/illegal axis ID in the axis group
2002	MC_ERR_AXISID_IS_REPEATED	Same axis number/same axis number in the axis group
2003	MC_ERR_AXIS_TYPE_NOT_SUPPORTED	Unsupported axis type/axis in the axis group is an unsupported axis type
2004	MC_ERR_SLAVEID_INVALID	Illegal Slave StationID/illegal Slave StationID in axis group
2005	MC_ERR_SLAVE_OFFLINE	Slave Station Offline/Slave StationID Offline in Axis Group
2006	MC_ERR_SLAVE_FAULT	Slave Station Fault / Slave Station ID fault in axis group
2007	MC_ERR_SYSTEM_ERR	System error occurred
2008	MC_ERR_AXIS_ERR	Error in axis/error in axis group
2009	MC_ERR_AXIS_MOTION_IS_REJECTED_IN_STOPPING	Reject motion command in Stopping state of axis
2010	MC_ERR_AXIS_STATE_ERROR_STOP	Axis error stop / Error stop of an axis in the axis group
2011	MC_ERR_AXIS_DISABLE	Axis not enabled/axes in the axis group not enabled
2012	MC_ERR_PDO_SETTING_MISSING	PDO not configured
2013	MC_ERR_AXIS_MOTION_BUFFER_FULL	Axis motion buffer full
2014	MC_ERR_VELOCITY_INVALID	Illegal target speed parameter
2015	MC_ERR_ACCELERATION_INVALID	Illegal acceleration parameter
2016	MC_ERR_DECELERATION_INVALID	Illegal deceleration parameter
2017	MC_ERR_JERK_INVALIDE	Illegal jerk parameter
2018	MC_ERR_BUFFERMODE_INVALID	BufferMode parameter illegal
2019	MC_ERR_DIRECTION_INVALID	Illegal direction parameter
2020	MC_ERR_SDO_READ_ERR	EtherCAT SDO read error
2021	MC_ERR_SDO_WRITE_ERR	EtherCAT SDO write error
2022	MC_ERR_AXIS_NOT_ASSOCIATED_WITH_PHYSICAL_DEVICE	The axis is not associated with an actual physical device
2200	MC_ERR_AXES_GROUP_DISABLED	axis group not enabled
2201	MC_ERR_RELATIVE_INTERPOLATION_DIST0	Parameter check when the movement distance of relative interpolation command is 0
2202	MC_ERR_COORD_SYSTEM_INVALID	Illegal coordinate system

2203	MC_ERR_TRANSITION_MODE_INVALID	Illegal transition curve selection parameter
2204	MC_ERR_AXES_GROUP_ENABLED_FAILURE	axis group enable failed
2950	MC_ERR_PDO_UNMAPPED_FOR_STATUS_WORD	Bus driver PDO not mapped Statusword
2951	MC_ERR_PDO_UNMAPPED_FOR_ACTUAL_POS	Bus driver PDO is not mapped to Position actual value
2952	MC_ERR_PDO_UNMAPPED_FOR_TOUCH_PROBE_STATUS	Touch probe status not mapped by bus driver PDO
2953	MC_ERR_PDO_UNMAPPED_FOR_TOUCH_PROBE_POS_POS1	Touch probe pos1 pos value is not mapped by bus driver PDO
2954	MC_ERR_PDO_UNMAPPED_FOR_TOUCH_PROBE_NEG_POS1	Touch probe pos1 neg value is not mapped by bus driver PDO
2955	MC_ERR_PDO_UNMAPPED_FOR_TOUCH_PROBE_POS_POS2	Touch probe pos2 pos value is not mapped by bus driver PDO
2956	MC_ERR_PDO_UNMAPPED_FOR_TOUCH_PROBE_NEG_POS2	Touch probe pos2 neg value is not mapped by bus driver PDO
2957	MC_ERR_PDO_UNMAPPED_FOR_DIGITAL_INPUTS	Bus driver PDO not mapped to Digital inputs
2958	MC_ERR_PDO_UNMAPPED_FOR_OPERATE_MODE_DISPLAY	Bus Driver PDO Unmapped Modes of operation display
2959	MC_ERR_PDO_UNMAPPED_FOR_ACTUAL_VELOCITY	Bus driver PDO is not mapped to Velocity actual value
2960	MC_ERR_PDO_UNMAPPED_FOR_ACTUAL_TORQUE	Bus driver PDO is not mapped to Torque actual value
2961	MC_ERR_PDO_UNMAPPED_FOR_ERROR_CODE	Error code not mapped by bus driver PDO
2962	MC_ERR_PDO_UNMAPPED_FOR_CONTROL_WORD	Bus Driver PDO not mapped Controlword
2963	MC_ERR_PDO_UNMAPPED_FOR_OPERATE_MODE	Modes of operation for bus driver PDO unmapped
2964	MC_ERR_PDO_UNMAPPED_FOR_TARGET_POS	Bus driver PDO not mapped to Target position
2965	MC_ERR_PDO_UNMAPPED_FOR_TARGET_VELOCITY	Bus driver PDO is not mapped to Target velocity
2966	MC_ERR_PDO_UNMAPPED_FOR_TARGET_TORQUE	Bus driver PDO is not mapped to Target torque
2967	MC_ERR_PDO_UNMAPPED_FOR_TOUCH_PROBE_FUNC	Touch probe function not mapped by bus driver PDO
2968	MC_ERR_PDO_UNMAPPED_FOR_DIGITAL_OUTPUTS	Bus driver PDO not mapped to Digital outputs
2969	MC_ERR_PDO_UNMAPPED_FOR_MAX_PROFILE_VELOCITY	Bus driver PDO not mapped Max profile velocity
2970	MC_ERR_PDO_UNMAPPED_FOR_MAX_MOTOR_SPEED	Max motor speed not mapped by bus driver PDO
2971	MC_ERR_PDO_UNMAPPED_FOR_MAX_TORQUE	Bus driver PDO not mapped Max torque
2972	MC_ERR_PDO_UNMAPPED_FOR_POS_TORQUE_LIMIT	Bus driver PDO not mapped Positive torque limit value
2973	MC_ERR_PDO_UNMAPPED_FOR_NEG_TORQUE_LIMIT	Bus driver PDO not mapped Negative torque limit value
3000	MC_ERR_AXIS_INIT_UNITS_PARA_CONFIG_ERR	Error in setting relevant parameters of user unit
3001	MC_ERR_AXIS_INIT_POS_LIMIT_SWITCH_CFG_ERR	Forward limit switch input parameter configuration error
3002	MC_ERR_AXIS_INIT_NEG_LIMIT_SWITCH_CFG_ERR	Negative limit switch input parameter configuration error
3003	MC_ERR_AXIS_INIT_HOME_IN_SWITCH_CFG_ERR	Origin return switch input parameter configuration error
3004	MC_ERR_AXIS_INIT_HMSW_SWITCH_CHANNEL_NUM_ERR	Hardware switch channel number configuration error
3005	MC_ERR_AXIS_INIT_HMSW_SWITCH_CHANNEL	Hardware limit switch type error

	_TYPE_ERR	
3006	MC_ERR_AXIS_INIT_POS_LIMIT_SWITCH_NUM_ERR	Positive limit switch channel number error
3007	MC_ERR_AXIS_INIT_NEG_LIMIT_SWITCH_NUM_ERR	Negative limit switch channel number error
3008	MC_ERR_AXIS_INIT_HOME_IN_SWITCH_NUM_ERR	Origin return switch channel number error
3009	MC_ERR_AXIS_INIT_SET_USER_UNITS_ERR	Error in setting user unit
3010	MC_ERR_AXIS_INIT_AXIS_TYPE_CFG_TIMEOUT	Axis type configuration timeout
3011	MC_ERR_AXIS_INIT_SET_AXIS_TYPE_REPEAT	Repeated setting of the same axis type
3012	MC_ERR_AXIS_INIT_SET_AXIS_TYPE_TIMING_ERR	When setting the axis type, the axis already has a motion control command
3013	MC_ERR_AXIS_INIT_SET_OPERATION_MODE_ERR	Failed to set axis operation mode
3014	MC_ERR_AXIS_INIT_NOT_ECAT_ENCODER_AXIS_TYPE	Current axis is not of EtherCAT_Encoder axis type
3015	MC_ERR_NEG_LIM_CHAN_AND_POS_LIM_CHAN_REUSED	Repeated positive and negative hard limit channel number configuration
3016	MC_ERR_DRIVER_CHAN_CONFLICT_WITH_COMM_DI_CHAN	The actual address converted from the DI channel number of the driver body conflicts with the physical DI channel address
3017	MC_ERR_NEG_LIM_CHAN_CONFLICT_WITH_POS_LIM	The set positive limit channel number conflicts with the negative limit channel number
3018	MC_ERR_POS_LIM_CHAN_CONFLICT_WITH_NEG_LIM	The set negative limit channel number conflicts with the positive limit channel number
3030	MC_ERR_AXIS_POWER_MOD_DISP_TIMEOUT	Axis type configuration timeout
3040	MC_ERR_HOME_APPROACH_VELOCITY_INVALID	Homing APPROACH VELOCITY SETTING ILLEGAL
3041	MC_ERR_HOME_CAPTURE_CHL_INVALID	Illegal origin return capture channel number
3042	MC_ERR_HOME_CAPTURE_CHL_BUSY	OR capture channel occupied
3043	MC_ERR_HOME_MODE_Z_NOT_SUPPORTED	Homing does not support Z-phase mode
3044	MC_ERR_HOME_HOME_MODE_INVALID	Invalid regression mode of input parameter origin
3045	MC_ERR_HOME_AXIS_TYPE_FOR_Z_ERR	Z-phase capture mode is configured, but it is not EtherCAT servo axis
3046	MC_ERR_HOME_HOME_CHL_CONFIG_ERR	Origin switch channel configuration error
3047	MC_ERR_HOME_POT_CHL_CONFIG_ERR	Positive Limit Switch Channel Configuration Error
3048	MC_ERR_HOME_NOT_CHL_CONFIG_ERR	Negative limit switch channel configuration error
3049	MC_ERR_HOME_ALL_DI_CONFIG_ERR	Advanced Origin REDI Channel Configuration Error
3090	MC_ERR_MOVE_RELATIVE_DIST_INVALID	Illegal distance of single-axis relative motion
3100	MC_ERR_MCMD_NOT_SYNC_MOVE_VELOCITY	The current command is not SyncMoveVelocity command
3101	MC_ERR_SYNC_MOVE_VELOCITY_CMD_DATA_NULL	SyncMoveVelocity command data is null
3120	MC_ERR_SET_JERKM_MODE_INVALID	Acceleration and deceleration type setting error
3130	MC_ERR_TORQUE_CONTROL_TORQUE_RAMP_INVALID	Torque control function block input reference torque change rate check invalid
3131	MC_ERR_TORQUE_CONTROL_TORQUE_INVALID	Torque control function block input reference target torque check invalid
3132	MC_ERR_TORQUE_CONTROL_VELOCITY_INVALID	Torque control function block input parameter speed check invalid
3133	MC_ERR_TORQUE_CONTROL_MAX_VELOCITY_CONFIG_ERR	Torque Control Function Block Maximum Speed Configuration Error
3160	MC_ERR_SET_OVERRIDE_VEL_FACTOR_INVALID	Speed scale factor parameter error
3161	MC_ERR_SET_OVERRIDE_ACC_FACTOR_INVAL	Acceleration and deceleration scale factor

	ID	parameter error
3162	MC_ERR_SET_OVERRIDE_JERK_FACTOR_INVALID	Jerk scale factor parameter error
3170	MC_ERR_MAX_VELOCITY_INVALID	Illegal maximum speed limit parameter
3171	MC_ERR_MAX_ACCELERATION_INVALID	Illegal maximum limit acceleration parameter
3172	MC_ERR_MAX_DECELERATION_INVALID	Illegal maximum limiting deceleration parameter
3173	MC_ERR_MAX_JERK_INVALID	Illegal maximum limit jerk parameter
3180	MC_ERR_READ_DIGITAL_INPUT_NUM_INVALID	Illegal parameter for reading Digital Input Number
3190	MC_ERR_READ_LIMIT_STATUS_SOFT_LIMIT_PARA_ERR	Error in relevant parameters of soft limit
3191	MC_ERR_READ_LIMIT_STATUS_EACH_STATUS_CONFLICT	Limit state conflict error
3210	MC_ERR_RESET_CMD_BACK_INVALID	Reset function return error
3211	MC_ERR_RESET_AXIS_TIME_OUT_ERR	Axis reset error timeout failed
3212	MC_ERR_RESET_AXIS_WAIT_TIME_OUT_ERR	Waiting for reset timeout
3220	MC_ERR_SET_TORQUE_LIMIT_POS_VALUE_INVALID	Illegal positive limit value
3221	MC_ERR_SET_TORQUE_LIMIT_NEG_VALUE_INVALID	Illegal negative limit value
3222	MC_ERR_SET_TORQUE_LIMIT_SLAVE_INVALID	Slave Station No. illegal
3223	MC_ERR_SDO_WRITE_POS_TORQUE_LIMITERR	Error in writing forward limit value SDO
3224	MC_ERR_SDO_WRITE_NEG_TORQUE_LIMITERR	Write negative limit value SDO error
3225	MC_ERR_SDO_WRITE_POS_TORQUE_LIMITOVERRUN	Over-limit of writing forward limit value
3226	MC_ERR_SDO_WRITE_NEG_TORQUE_LIMITOVERRUN	Over-limit of writing negative limit value
3230	MC_ERR_TOUCH_PROBE_LATCH_MODE_INVALID	Invalid input parameter lock mode configuration of position lock function block
3231	MC_ERR_TOUCH_PROBE_LATCH_ID_INVALID	Invalid input parameter latch ID configuration of position latch function block
3232	MC_ERR_TOUCH_PROBE_TRIGGER_SIGNAL_INVALID	Invalid triggering signal configuration of input parameter latch of position latch functional block
3233	MC_ERR_TOUCH_PROBE_TRIGGER_EDGE_INVALID	Invalid input parameter latch triggering edge configuration of position latch function block
3234	MC_ERR_TOUCH_PROBE_WINDOW_CONFIG_ERR_1	Window Configuration Error 1, First Position Set Greater Than Last Position With Limited Stroke
3235	MC_ERR_TOUCH_PROBE_WINDOW_CONFIG_ERR_2	Window configuration error 2, unreasonable window position setting during cyclic stroke 1
3236	MC_ERR_TOUCH_PROBE_WINDOW_CONFIG_ERR_3	Window configuration error 3, unreasonable window position setting during cyclic stroke 2
3237	MC_ERR_TOUCH_PROBE_IS_REUSED	Latch channel reuse error
3238	MC_ERR_TOUCH_PROBE_SET_USED_ERR	An error occurred when setting the channel in use
3239	MC_ERR_TOUCH_PROBE_SYSTEM_ERR	System internal error occurred in position acquisition
3240	MC_ERR_TOUCH_PROBE_PARA_COMBINE_ERR	Configuration parameter combination error
3241	MC_ERR_TOUCH_PROBE_CHANNEL_INNER_USED	The channel is used internally or the channel use status is unknown
3242	MC_ERR_TOUCH_PROBE_FB_IS_USING	The function block is being called
3243	MC_ERR_TOUCH_PROBE_PDO_CONFIG_CHECK_ERR	PDO mapping check error during position latch configuration
5000	MC_ERR_GEAR_RATIO_DENOMINATOR_INVALID	Gear ratio denominator is 0

5001	MC_ERR_GEAR_MASTER_SOURCE_INVALID	Illegal master position source parameter
5040	MC_ERR_CAM_POINTER_TO_ARRARYPOS_ISNULL	Pointer to cam position array is null
5041	MC_ERR_CAM_TABLE_POSNUM_INVALID	Illegal number of positions in cam position array (less than 3)
5042	MC_ERR_CAM_MASTER_SOURCE_INVALID	Illegal electronic camshaft position source parameter
5043	MC_ERR_CAM_SLAVESCALE_INVALID	Electronic camshaft slave shaft position data scale factor illegal
5044	MC_ERR_CAM_DI_STARTPOSITION_INVALID	Electronic cam DI triggering starting position illegal
5045	MC_ERR_CAM_START_MODE_INVALID	Electronic cam start mode error
5046	MC_ERR_CAM_SLAVE_STANDIN_POSITION_LIMIT1	The slave shaft is in the limit status when the electronic cam starts
5047	MC_ERR_CAM_ARRARYPOS_ISNOT_MONOINCREASE	cam array is not a monotonically increasing array
5048	MC_ERR_CAM_MASTER_POS_MONODECREASE	The master position data changes to decreasing
5049	MC_ERR_CAM_STARTMODE_INVALID	Cam Enable Mode Error
5050	MC_ERR_CAM_CAPTURE_CHL_OVERLIMIT	Cam trap channel exceeds the upper limit
5051	MC_ERR_CAM_DI_CHL_OVERLIMIT	Cam DI channel exceeds the upper limit
5052	MC_ERR_CAM_CAMTABLE_CURVETYPE_INVALID	Illegal cam curve interpolation mode
5053	MC_ERR_CAM_TABLE_CHANGED_ERR_IN_RUNNING	The cam table is tampered with illegal values during operation
5100	MC_ERR_MOVELINK_MASTER_DIST_INVALID	Illegal flying shearing master movement distance parameter
5101	MC_ERR_MOVELINK_MASTER_ACCDIST_INVALID	Illegal distance parameter of master movement in slave axis acceleration stage
5102	MC_ERR_MOVELINK_MASTER_DECDIST_INVALID	Illegal distance parameter of master movement during trailing trimming slave axis deceleration stage
5103	MC_ERR_MOVELINK_S_RATIO_INVALID	Illegal flying shearing S-shaped acceleration/deceleration ratio parameter
5104	MC_ERR_MOVELINK_START_MODE_INVALID	Cut start mode error
5105	MC_ERR_MOVELINK_MASTER_SOURCE_INVALID	Illegal flying shearing master position source parameter
5106	MC_ERR_MOVELINK_DI_STARTPOPITION_INVALID	Illegal starting position triggered by clipping DI
5107	MC_ERR_MOVELINK_SLAVE_STANDIN_POS_LIMIT	The slave axis is in the limit status when the flying shearing movement starts.
5108	MC_ERR_MOVELINK_START_MODE_INVALID1	Trimming start mode error (internal)
5109	MC_ERR_MOVELINK_CAPTURE_CHL_OVERLIMIT	The pursuit capture channel exceeds the upper limit
5110	MC_ERR_MOVELINK_DI_CHL_OVERLIMIT	Trail DI channel exceeds the upper limit
5111	MC_ERR_MOVELINK_S_CURVE_DIST_INVALID	Illegal acceleration/deceleration distance of flying shearing S-shaped curve
5112	MC_ERR_MOVELINK_DIST_PARAM_INVALID	Illegal setting of flying shearing distance parameter
5113	MC_ERR_MOVELINK_MASTER_ADD_DIST_INVALID	Illegal completion and remaining distance of flying shear superimposed motion master
5150	MC_ERR_GANTRY_AXIS_NUM_ERR	The number of gantry synchronous shaft groups shall be greater than or equal to 2 and less than or equal to 8
5151	MC_ERR_GANTRY_AXISARRY_OR_DICHLARRT_IS_NULL	Input parameter axis number array or DI channel array in gantry synchronization correlation function is empty
5152	MC_ERR_GANTRY_AXIS_TYPES_ARE_DIFFERE	Shaft types in the gantry synchronous shaft

	NT	group are different
5153	MC_ERR_GANTRY_AXIS_TYPE_SET_ERR	Gantry synchronizing shaft type error
5154	MC_ERR_GANTRY_INIT_INPUT_DICHL_REPEATED	Repetition of the origin switch channel used in gantry synchronization initialization
5155	MC_ERR_GANTRY_INIT_HOME_MODE_SET_ERR	Gantry synchronous initialization origin return mode setting error, regression mode not supported 33, 34, 35
5156	MC_ERR_GANTRY_INIT_AXIS_POS_FE_OVERRIDE	The inter-axis deviation is too large after the gantry synchronous initialization is completed.
5157	MC_ERR_GANTRY_INIT_INPUT_PARA_POSERR_SET_ERR	The input parameter dPosErr for gantry synchronization initialization is less than 0
5158	MC_ERR_GANTRY_AXIS_NOT_REPMODE_0	Gantry synchronizing shaft travel mode is not limited
5159	MC_ERR_GANTRY_INIT_NOT_ECAT_AXIS_FOR_Z	Z-phase capture is used in gantry synchronization initialization but the axis is not an EtherCAT bus axis
6010	MC_ERR_CIRC_INTERP_MODE_INVALID	Unsupported circular interpolation mode
6011	MC_ERR_CIRC_MODE_INVALID	Illegal arc mode
6012	MC_ERR_CIRC_PATHCHOICE_INVALID	Illegal circular motion direction
6020	MC_ERR_CIRC_SPEEDLIMITS_OPTION_INVALID	Illegal setting mode value of arc speed limit parameter
6021	MC_ERR_CIRC_SPEEDLIMITS_MAXIMPULSE_INVALID	Illegal maximum impact value of arc speed limit
6022	MC_ERR_CIRC_SPEEDLIMITS_RADIUS_INVALID	Illegal specified arc radius for arc speed limit
6023	MC_ERR_CIRC_SPEEDLIMITS_VELOCITY_INVALID	Arc speed limitThe maximum interpolation speed of the specified arc radius is illegal
6030	MC_ERR_CORNER_SPEEDLIMITS_DECANGLE_INVALID	Illegal deceleration angle parameter of steering corner velocity limit
6031	MC_ERR_CORNER_SPEEDLIMITS_STOPANGLE_INVALID	Illegal corner velocity limit stop angle parameter

8.2 Description of Buffer Mode

8.2.1 Buffer mode

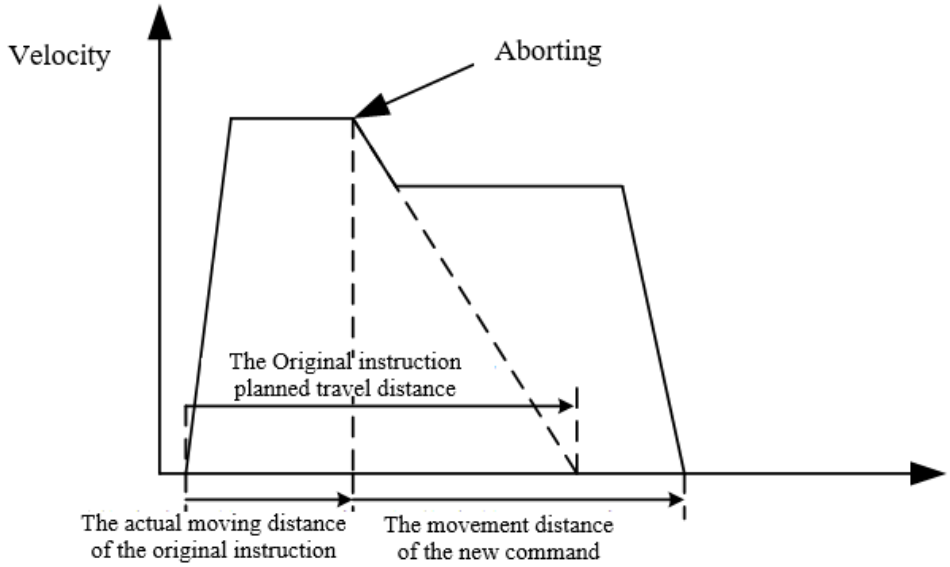
Define the variable type of buffer mode as MC_BUFFER_MODE.

Enumerated Values	MC_BUFFER_MODE	Description
0	Aborting	Interrupt. Interrupt the current motion command and execute it immediately.
1	Buffered	buffer. After the current motion command is executed, this motion command will be executed.
2	BlendingLow	with low-speed fusion between adjacent commands.
3	BlendingPrevious	Speed fusion of the previous command.
4	BlendingNext	Velocity fusion for the next command.
5	BlendingHigh	with high-speed fusion between adjacent commands.

8.2.1.1 Interrupt

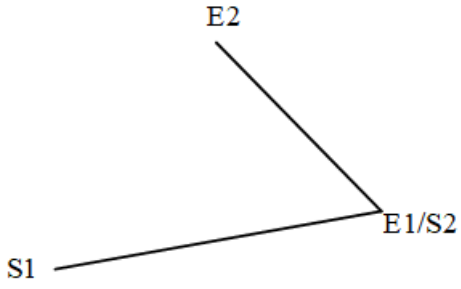
Only single-axis motion commands support the interrupt function; interpolation commands do not support this function.

If the cache mode of the triggered single-axis motion command is set to Aborting (Interrupt), then this command can change the action of the currently executing command. The currently executing motion command is interrupted. At this point, the axis performs the triggered motion command, and parameters such as Position (target position), Velocity (target velocity), Acceleration (acceleration), Deceleration (deceleration), and Jerk (jerk) are changed. An example of the action is shown in the following figure.



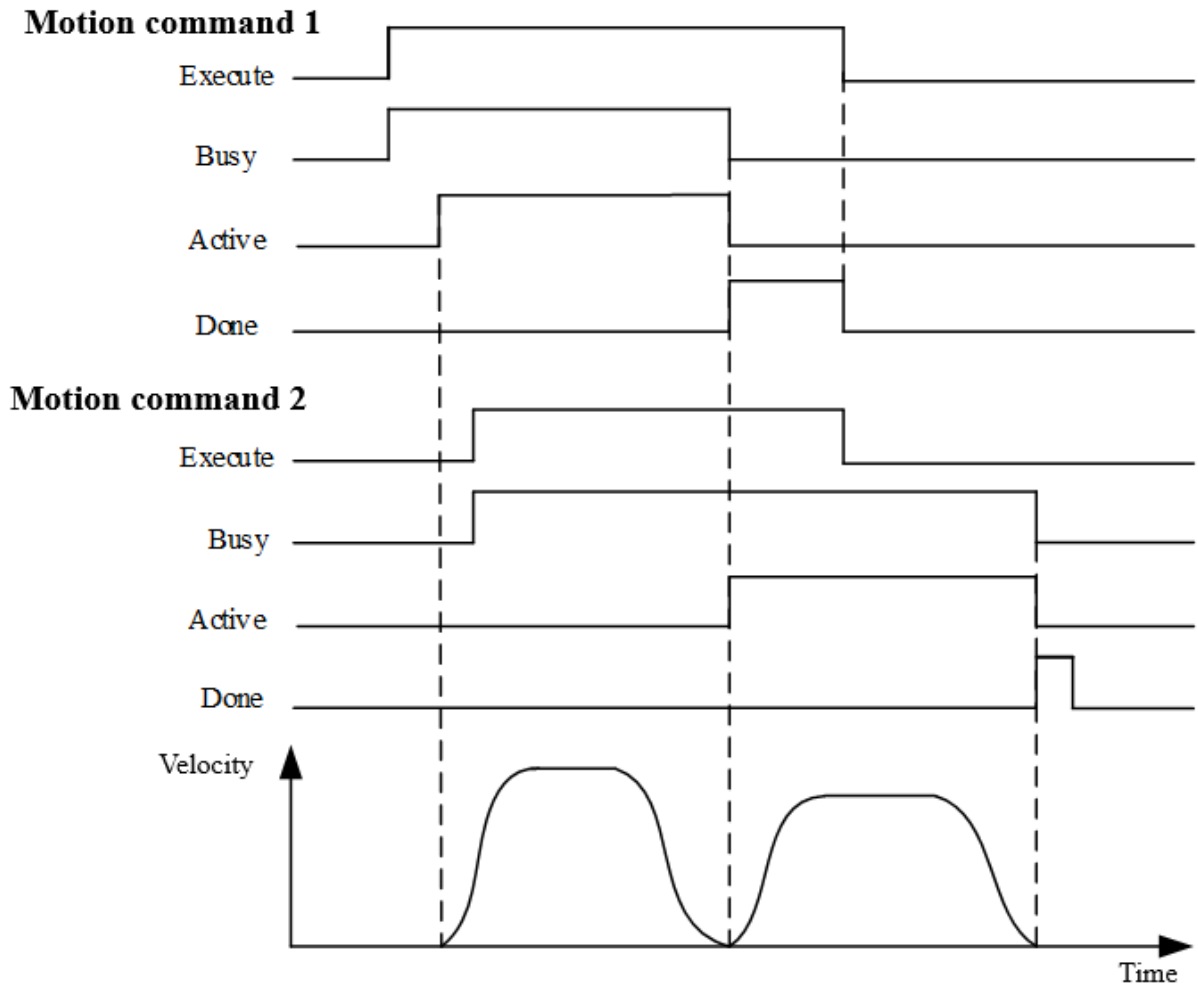
8.2.1.2 Buffer

After the current moving command is executed, execute the next command. The following figures are diagrams of motion in buffer mode:



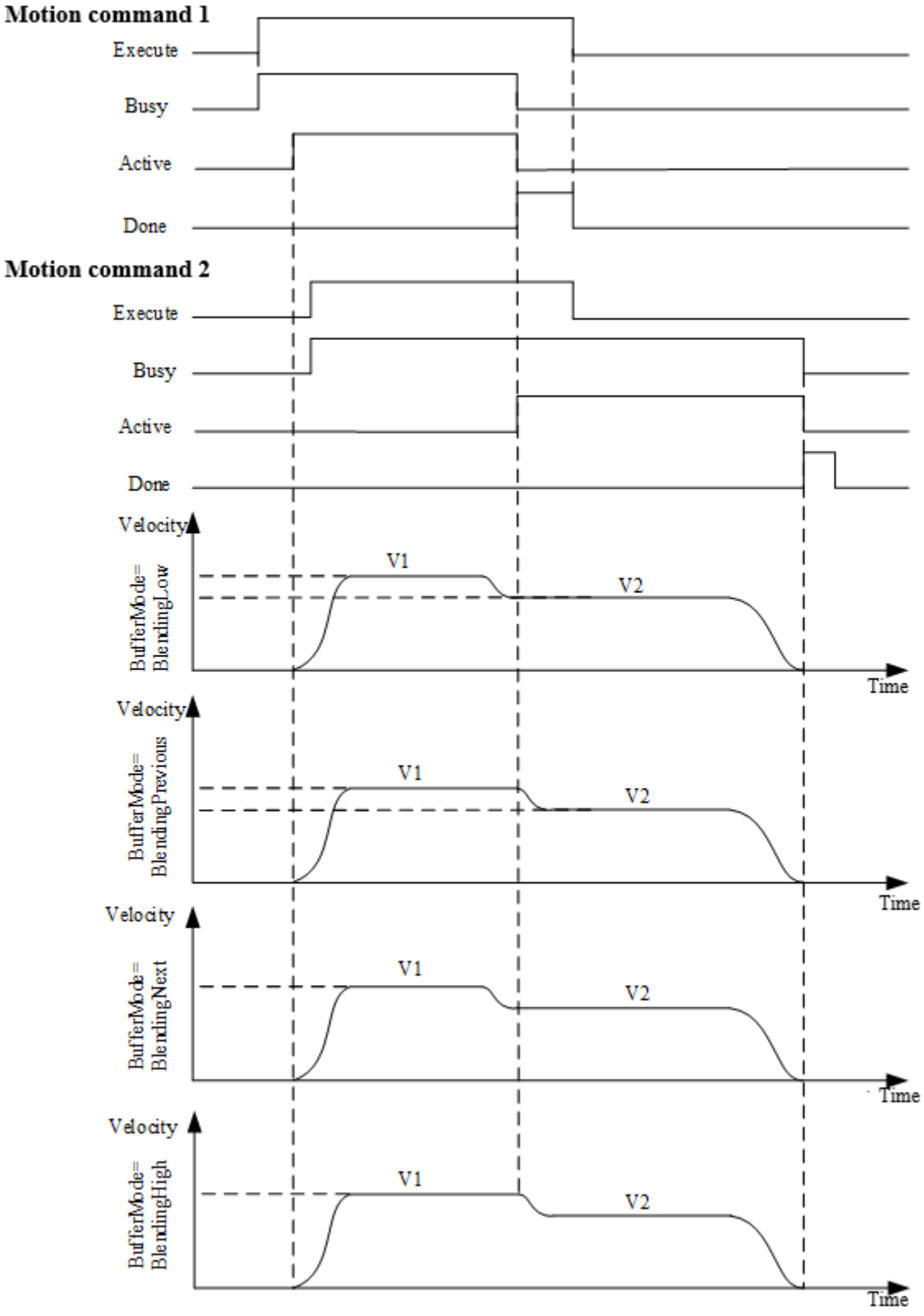
- S1: Ttarting position of previous motion command
- E1: Target position of the previous motion command

- S2: starting position of the next motion command
- E2: Target position of the next motion command



8.2.1.3 Fusion

The previous and next commands are speed fused in the set fusion mode, and the specific speed fusion mode is determined by the fusion mode.


Motion command 1
Execute
Busy
Active
Done
Motion command 2
Execute
Busy
Active
Done
Velocity
BufferMode=
BlendingLow
V1
V2
Time
Velocity
BufferMode=
BlendingPrevious
V1
V2
Time
Velocity
BufferMode=
BlendingNext
V1
V2
Time
Velocity
BufferMode=
BlendingHigh
V1
V2
Time

Beijing HollySys Intelligent Technologies Co., Ltd..

Di Sheng Middle Road, No.2

Economic-Technological Development Area

100176 Beijing, China

Tel: 010-5898 1588

Hotline: 400-811-1999

Fax: 010-5898 1558

http://www.hollysys.com