Smarter. Greener. Together.

Industrial Automation Headquarters

Delta Electronics, Inc.
Taoyuan Technology Center
No.18, Xinglong Rd., Taoyuan District,
Taoyuan City 33068, Taiwan
TEL: 886-3-362-6301 / FAX: 886-3-371-6301

Asia

Delta Electronics (Shanghai) Co., Ltd.
No.182 Minyu Rd., Pudong Shanghai, P.R.C.
Post code : 201209
TEL: 86-21-6872-3988 / FAX: 86-21-6872-3996
Customer Service: 400-820-9595

Delta Electronics (Japan), Inc.
Tokyo Office
Industrial Automation Sales Department
2-1-14 Shibadaimon, Minato-ku
Tokyo, Japan 105-0012
TEL: 81-3-5733-1155 / FAX: 81-3-5733-1255

Delta Electronics (Korea), Inc.
Seoul Office
1511, 219, Gasan Digital 1-Ro., Geumcheon-gu,
Seoul, 08501 South Korea
TEL: 82-2-515-5305 / FAX: 82-2-515-5302

Delta Energy Systems (Singapore) Pte Ltd.
4 Kaki Bukit Avenue 1, #05-04, Singapore 417939
TEL: 65-6747-5155 / FAX: 65-6744-9228

Delta Electronics (India) Pvt. Ltd.
Plot No.43, Sector 35, HSIIDC Gurgaon,
PIN 122001, Haryana, India
TEL: 91-124-4874900 / FAX : 91-124-4874945

Delta Electronics (Thailand) PCL.
909 Soi 9, Moo 4, Bangpoo Industrial Estate (E.P.Z),
Pattana 1 Rd., T.Phraksa, A.Muang,
Samutprakarn 10280, Thailand
TEL: 66-2709-2800 / FAX : 662-709-2827

Delta Electronics (Australia) Pty Ltd.
Unit 20-21/45 Normanby Rd., Notting Hill Vic 3168, Australia
TEL: 61-3-9543-3720

Americas

Delta Electronics (Americas) Ltd.
Raleigh Office
P.O. Box 12173, 5101 Davis Drive,
Research Triangle Park, NC 27709, U.S.A.
TEL: 1-919-767-3813 / FAX: 1-919-767-3969

Delta Electronics Brazil
São Paulo Sales Office
Rua Itapeva, 26 - 3°, andar Edificio Itapeva,
One - Bela Vista 01332-000 - São Paulo - SP - Brazil
TEL: 55-12-3932-2300 / FAX: 55-12-3932-237

Delta Electronics International Mexico S.A. de C.V.
Mexico Office
Gustavo Baz No. 309 Edificio E PB 103
Colonia La Loma, CP 54060
Tlalnepantla, Estado de México
TEL: 52-55-3603-9200

EMEA

Headquarters: Delta Electronics (Netherlands) B.V.
Sales: Sales.IA.EMEA@deltaww.com
Marketing: Marketing.IA.EMEA@deltaww.com
Technical Support: iatechnicalsupport@deltaww.com
Customer Support: Customer-Support@deltaww.com
Service: Service.IA.emea@deltaww.com
TEL: +31(0)40 800 3900

BENELUX: Delta Electronics (Netherlands) B.V.
De Witbogt 20, 5652 AG Eindhoven, The Netherlands
Mail: Sales.IA.Benelux@deltaww.com
TEL: +31(0)40 800 3900

DACH: Delta Electronics (Netherlands) B.V.
Coesterweg 45, D-59494 Soest, Germany
Mail: Sales.IA.DACH@deltaww.com
TEL: +49(0)2921 987 0

France: Delta Electronics (France) S.A.
ZI du bois Challand 2, 15 rue des Pyrénées,
Lisses, 91090 Evry Cedex, France
Mail: Sales.IA.FR@deltaww.com
TEL: +33(0)1 69 77 82 60

Iberia: Delta Electronics Solutions (Spain) S.L.U
Ctra. De Villaverde a Vallecas, 265 1º Dcha Ed.
Hormigueras – P.I. de Vallecas 28031 Madrid
TEL: +34(0)91 223 74 20

Carrer Llacuna 166, 08018 Barcelona, Spain
Mail: Sales.IA.Iberia@deltaww.com

Italy: Delta Electronics (Italy) S.r.l.
Via Meda 2–22060 Novedrate(CO)
Piazza Grazioli 18 00186 Roma Italy
Mail: Sales.IA.Italy@deltaww.com
TEL: +39 039 8900365

Russia: Delta Energy System LLC
Vereyskaya Plaza II, office 112 Vereyskaya str.
17 121357 Moscow Russia
Mail: Sales.IA.RU@deltaww.com
TEL: +7 495 644 3240

Turkey: Delta Greentech Elektronik San. Ltd. Sti. (Turkey)
Şerifali Mah. Hendem Cad. Kule Sok. No:16-A
34775 Ümraniye – İstanbul
Mail: Sales.IA.Turkey@deltaww.com
TEL: + 90 216 499 9910

GCC: Delta Energy Systems AG (Dubai BR)
P.O. Box 185668, Gate 7, 3rd Floor, Hamarain Centre
Dubai, United Arab Emirates
Mail: Sales.IA.MEA@deltaww.com
TEL: +971(0)4 2690148

Egypt + North Africa: Delta Electronics
Unit 318, 3rd Floor, Trivium Business Complex, North 90 street,
New Cairo, Cairo, Egypt
Mail: Sales.IA.MEA@deltaww.com

*We reserve the right to change the information in this catalogue without prior notice.

AX Series Motion Controller Instructions Manual

AX Series Motion Controller Instructions Manual

www.deltaww.com

Smarter. Greener. Together.

Version	Revision	Date
1st	The first version was published.	03/31/2020
2nd	1. Updated Sections 2.1.1 ~ 2.1.18, 2.2.1 ~ 2.2.16, 2.3.1.1 ~ 2.3.1.9, 2.3.1.11, 2.3.2.1 ~ 2.3.2.2, and 2.3.2.5 ~ 2.3.2.22: Added supported product AX-364E 2. Updated Sections 2.3.1.1, 2.3.1.3 ~ 2.3.1.8, 2.3.2.1 ~ 2.3.2.2, 2.3.2.5 ~ 2.3.2.11, and 2.3.2.15 ~ 2.3.2.16: Updated ErrorID 3. Added Sections 2.3.1.10, 2.3.1.12, 2.3.1.13, and 2.3.2.3 ~ 2.3.2.4 4. Updated Section 2.3.1.11: Updated bInVelocity and bone 5. Updated Section 2.3.2.6: Updated GroupErrorID 6. Updated Section 2.3.2.12: Updated dwErrorID 7. Updated Sections 2.3.3.1 ~ 2.3.3.16: Removed supported product AX-8 and added supported product AX-364E 8. Updated Appendix A1: Added DMC_ImmediateStop_P, DMC_GroupInterrupt, _GroupContinue, DMC_GroupReadParameter, and DMC_GroupWriteParameter 9. Updated Appendix A2: Updated DMC_GROUP_TRANSITION_MODE Added DMC_GROUP_PARAMETER 10. Updated Appendix A.3.1 11. Updated Appendix A.3.2	02/28/2021
3rd	1. Updated Section 2.3.2.22: Updated Timing to Take Effect of bWindowOnly, lrFirstPosition, and lrLastPosition 2. Updated Section 2.3.3: Added positioning axis speed range introduction 3. Updated Section 2.3.1.1: Updated the programming example figure 4. Updated Section 2.3.3.1: Updated the bStatus status timing	07/30/2021
4th	1. Updated Section 2.3.1.1: Updated the image of the DMC_TorqueControl motion behavior. 2. Updated Section 2.3.1.2: Updated Note 1 of Inputs. 3. Added Section 2.3.1.14 ~ 2.3.1.18. 4. Added Section 2.3.2.23 ~ 2.3.2.26. 5. Added Section 2.4.1: Added Servo Press introduction. 6. Updated Section A.2 & A.3: Added Data Type description and error code description.	06/01/2022
5th	1. Added Section 1.3: Added the overview of Delta motion instructions library 2. Updated Section 2.3.1.12: Added lrDeceleration and lrJerk descriptions 3. Updated Section 2.3.2.15: Added and modified function block parameters 4. Updated Section 2.3.2.21: Added descriptions of Harmonic2_Direct and Harmonic2_Inverse curves 5. Updated Section 2.3.2.24: Added descriptions of single-axis position 6. Added Section 2.3.2.27: Added DMC_GroupPower 7. Added Section 2.3.4: Added tension control 8. Updated Appendix A.1, A.2, and A.3: Added instructions, Data Types, and error codes	09/15/2022
6th	1. Update Section 2.1.1: Update function description 2. Update Section 2.3.1 and 2.3.2: Added version compatibility error troubleshooting 3. Update Section 2.3.1.1: Update function description	03/31/2023

Version	Revision	Date
	4. Update Section 2.3.4.1~2.3.4.3: Added library supported version 5. Add Section 2.3.2.29: Added DMC_GetCamSlaveData	

TOC

TOC 1
Preface 1
P.1 Introduction 1
P.1.1 Applicable Products 1
P.1.2 Associated Manuals 1
Chapter 1 Introduction to Motion Control 2
1.1 Introductions of Motion Control 3
1.1.1 Basic Knowledge of Motion Control Instructions 3
1.2 Categories of Motion Control Instructions 6
1.3 Overview of Delta Motion Instructions Library 7
Chapter 2 Motion Control Instruction 8
2.1 Motion Control Instructions 9
2.1.1 MC_Home 9
2.1.2 MC_Stop 13
2.1.3 MC_Halt 16
2.1.4 MC_MoveAbsolute 20
2.1.5 MC_MoveRelative 26
2.1.6 MC_MoveAdditive 30
2.1.7 MC_MoveSuperImposed 34
2.1.8 MC_CamIn 38
2.1.9 MC_CamOut 55
2.1.10 MC_MoveVelocity 59
2.1.11 MC_PositionProfile 64
2.1.12 MC_VelocityProfile 68
2.1.13 MC_AccelerationProfile 72
2.1.14 MC_Jog 76
2.1.15 MC_GearIn 79
2.1.16 MC_GearOut 83
2.1.17 MC_GearInPos 88
2.1.18 MC_Phasing 94
2.2 Administrative Motion Control Instructions 99
2.2.1 MC_Power 99
2.2.2 MC_SetPosition 104
2.2.3 MC_ReadParameter 109
2.2.4 MC_WriteParameter 112
2.2.5 MC_ReadBoolParameter 115
2.2.6 MC_WriteBoolParameter 118
2.2.7 MC_ReadActualPosition 121
2.2.8 MC_ReadActualVelocity 124
2.2.9 MC_ReadActualTorque 127
2.2.10 MC_Reset 128
2.2.11 MC_ReadStatus 131

2.2.12 MC_ReadAxisError 136
2.2.13 MC_CamTableSelect 139
2.2.14 MC_TouchProbe 143
2.2.15 MC_AbortTrigger 149
2.2.16 MC_DigitalCamSwitch 153
2.2.17 SMC_BacklashCompensation 160
2.3 Delta Motion Control Instructions 166
2.3.1 Motion Control Instructions 166
2.3.2 Administrative Instructions 279
2.3.3 Positioning Axis Instructions 401
2.3.4 Tension Control 455
2.4 Industry-Specific Instructions 479
2.4.1 Servo Press Instruction 479
Appendix A 516
A.1 Instruction Tables and Indexes 516
A.1.1 By Function 516
A.1.2 By Model 522
A.1.3 By Letter 524
A.2 Data Type: Enumeration and Structure 526
A.3 Error Codes and Troubleshooting 531
A.3.1 For Synchronous Axes 531
A.3.2 For Positioning Axis 549
A.4 Explanation of DMC_Home_P 553

Preface

P.1 Introduction

Thank you for purchasing the AX series Motion Controller with our advanced motion control system. Delta's AX series motion controller based on CODESYS integrates the control function of PLCs and motion controllers into one platform.

This manual introduces PLCOpen standard motion control instructions and Delta-defined instructions including single-axis, multi-axes instructions, and motion control applications.

Ensure that you fully understand the configuration and operations of the AX series motion control system and use the AX series Motion Controller CPU correctly.

P.1.1 Applicable Products

This manual relates to the following products

- **AX-3 series/AX-8 series**

P.1.2 Associated Manuals

The related manuals of the AX Motion Controller series are composed of the following.

1. **DIADesigner-AX User Manual**
   Contents include the use of DIADesigner-AX, the programming languages (ladder diagrams, sequential function charts, function block diagrams, and structured texts), the concept of POUs and Task, and the operation of motion control programming.

2. **AX-3 Series Operational Manual**
   It introduces basic knowledge of motion control structure, software/hardware setup, quick start of Software operations, devices to be used, motion control operations, troubleshooting, Input/ Output modules, modules of temperature measurement, etc.

3. **AX-8 Series Operational Manual**
   It introduces basic knowledge of motion control structure, software/hardware setup, quick start of Software operations, devices to be used, motion control operations, troubleshooting, Input/ Output modules, modules of temperature measurement, etc.

Chapter 1 Introduction to Motion Control

1.1 Introductions of Motion Control

This manual introduces the elements of motion control programming, including devices, symbols, and motion control instructions.

Motion control instructions are defined as function blocks (FB) and are used in the program for various control purposes. The motion control (MC) instructions are developed based on the specifications of PLCopen* motion control function blocks. In addition to the PLCopen-based instructions, Codesys also provides Delta-defined function blocks for users to achieve complete motion control applications.

This section overviews the motion control instructions for both PLCopen-based and Delta-defined function blocks. PLCopen defines the program and function block interfaces to achieve a standardized motion control programming environment for the languages specified in IEC61131-3. Using PLCopen-based instructions and Delta-defined instructions reduces training and support costs.

Before using the instructions, be sure that you sufficiently understand the devices, symbols and function of instructions.

You can also refer to the Appendix for a quick reference of the motion control instruction list and error codes.

***Note:**

PLCopen is an organization promoting industrial control based on IEC61131-3, an international standard widely adopted for PLC programming. For more information regarding PLCopen, check the official website at: http: //www.plcopen.org/

1.1.1 Basic Knowledge of Motion Control Instructions

Using motion control instructions requires the basic knowledge of motion control defined in the specifications of PLCopen motion control function blocks. This section provides an overview of these specifications.

- **Name of Motion Control Instructions**

  PLCopen-based motion control instructions begin with “MC_”, while Delta-defined function block instructions begin with “DFB_”.

Type	Description
MC_	PLCopen-based motion control instructions
DMC_	Delta-defined function block instructions*

  ***Note**: Delta-defined function block instructions (DFB) include Delta-defined motion control function blocks and other administrative/non-administrative function blocks applicable for AH Motion series CPUs. Therefore, you can look up a function block (FB) in this manual.

- **Types of Motion Control Related Instructions**

  Different categories of motion control instructions are divided by functions, such as single-axis motion instructions. Refer to Ch2 Motion Control Instructions for more details.

- **Execution of a Function Block**

  Function block instructions generally include two types of inputs for execution: Execute and Enable. When the instruction is run or enabled, the function block outputs can indicate the status. The basic outputs include Busy, Done, CommandAborted, and Error. For detailed information on inputs and outputs of each function block, refer to Ch2 Motion Control Instructions.

- Error Handling

  Information regarding error codes, indicators, and troubleshooting is in Appendix A for quick reference.

- Re-execution of a Function Block

  Re-execution of a function block refers to triggering Execute again after resetting it. You can change the input values and trigger Execute again while the function block is during operation (in busy status). Such output status will remain unchanged (in busy status), which also means the previously run instruction will be aborted by Aborting under the buffer mode.

- Multi-execution of Multiple Motion Control Instructions

Multi-execution of motion control instructions means that multiple instructions on the same axis are run in the same task execution period. The pattern of multi-execution is defined by the input variable BufferMode, specified to blend the two motions. Therefore, the instruction at the back will determine the behavior of the previous instruction according to BufferMode. Refer to *AX-3 Series Operation Manual* for more details.

- Buffer Modes

  Some motion instructions have an input called BufferMode.You can execute a different instruction instance during axis motion when the values for BufferMode are specified. This input decides whether the instruction runs immediately (non-buffered mode) or waits till the current motion instruction sets its status outputs.

  (Done/InVelocity/InPosition, etc.)

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction. When the instruction is run;

- The selected buffer mode is valid if the previous instruction is running.
- The selected buffer mode is invalid if the axis is in a Standstill state.
- The following Buffer Modes are supported.

Buffer Mode	Function
0: Aborting	Aborts the ongoing motion. The next instruction takes effect immediately.
1: Buffered	Automatically runs the next instruction after the ongoing motion is completed.
2: BlendingLow	The lower target velocity is the transit velocity between the current and the buffered instructions. (The transit velocity is the velocity that the current instruction uses as the transit point.)
3: BlendingPrevious	Takes the target velocity of the current instruction as the transit velocity.
4: BlendingNext	Takes the target velocity of the buffered instruction as the transit velocity.
5: BlendingHigh	Takes the higher target velocity as the transit velocity between the current instruction and the buffered instruction.

Refer to *AX-3 Series Operation Manual* for more details on buffer mode.

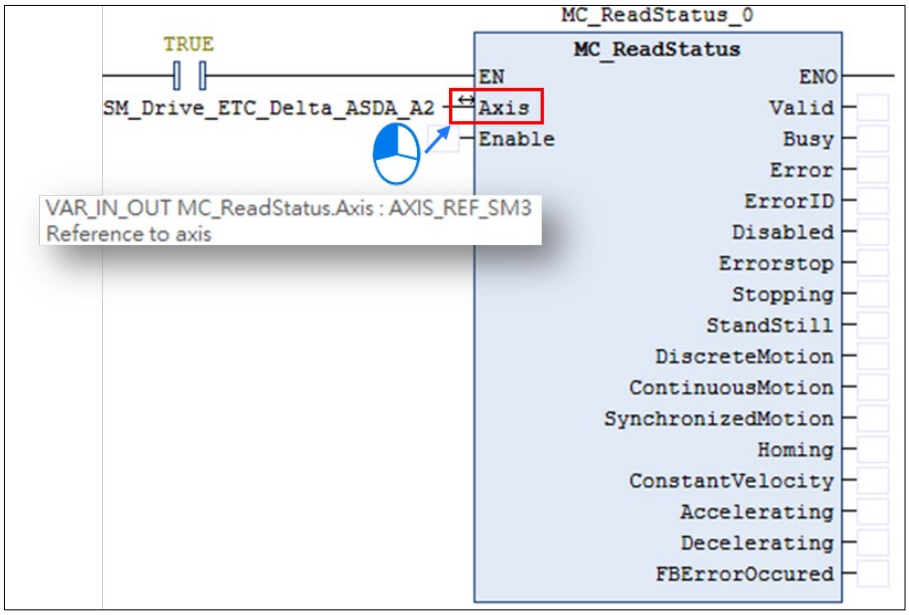
- **Structure Applicable for Motithe on Control**

  In PLCopen technical standard, the information and parameters required for configuring motion control on an axis are defined in a Structure.

  For AX Motion CPUs, a Structure is a Data Type applicable to group the data elements together, which is easier for users to specify proper parameters.

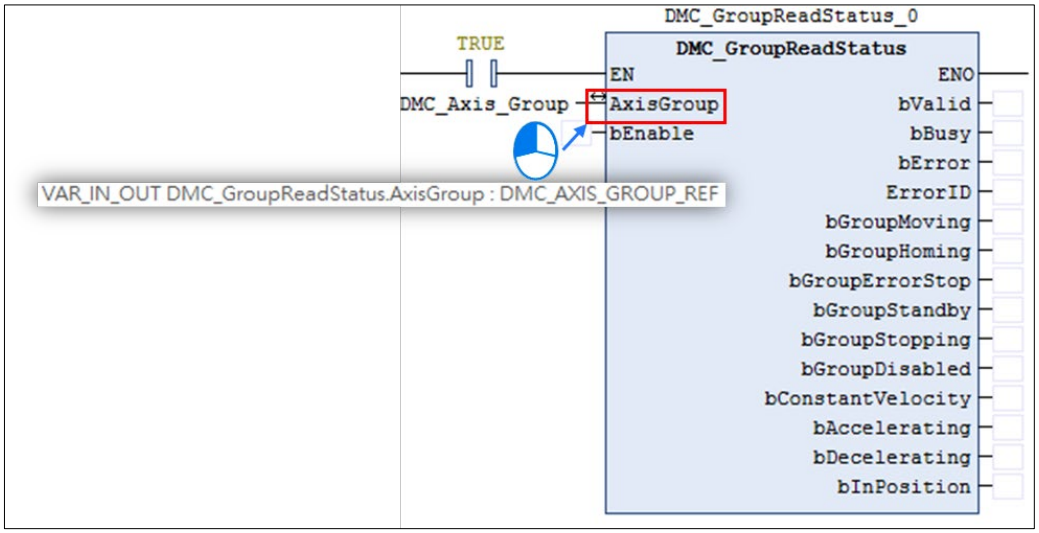
  For AX Motion CPUs, the applicable Structure is as below:

■ **Single-axis Function Block**



For single-axis FB, the applicable Structure is AXIS_REF_SM3.

■ **Function Block for Axis Group**



For AxisGroup FB, the applicable Structure is DMC_AXIS_GROUP_REF.

Note: For more details, refer to Appendix A.2 Data Types: Enumeration and Structure.

1.2 Categories of Motion Control Instructions

For AX Motion CPUs, motion control instructions are divided into two categories based on PLCOpen.

Categories	Type	Function Group	Description
Single-axis motion control instructions	Motion	Positioning on a single axis	"SMC": Motion instructions "MC_": PLCopen motion control instructions "DMC_": Delta-defined motion control instructions
		Velocity control on a single axis	
		Torque control on a single axis	
		Synchronized control on a single axis	
	Administrative	Administrative functions on a single axis	
Multi-axis motion control instructions Motion modules	Motion	Multi-axis coordinated control	Performing coordinated movement of an axis group
	Administrative	Administrative functions on multi-axis	Controlling, monitoring, or resetting axis group status

1.3 Overview of Delta Motion Instructions Library

This section describes related settings of Delta's motion instructions library DL_MotionControl and DL_MotionControlLight and Codesys Softmotion library in AX series motion control CPUs.

Delta's motion instructions DL_MotionControl and DL_MotionControlLight are derived from Codesys Softmotion. They provide users with convenient instructions for different occasions. The following tables list the version compatibility of Delta's motion instructions library and Codesys Softmotion instruction library:

Delta Motion Library	SM3_Basic V4.6.1.0	SM3_Basic V4.10.0.0
DL_MotionControl V1.1 and earlier versions	○	
DL_MotionControl V1.2 and later versions		○

Delta Motion Library	SML_Basic V4.5.1.0	SML_Basic V4.10.0.0
DL_MotionControlLight V1.1 and earlier versions	○	
DL_MotionControlLight V1.2 and earlier versions		○

Chapter 2 Motion Control Instruction

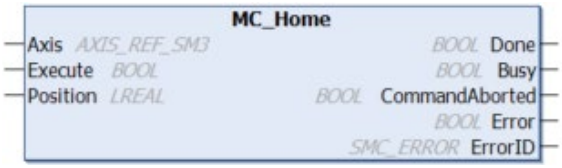
2.1 Motion Control Instructions

Motion control instructions are generally used to control motors to perform specific movement after the specified instruction being run. The function blocks used in this chapter are from the library “SM3_Basic” and able to operate synchronously with drives. As a result, synchronous axis type should be selected in axis settings. For more details about configuration related to synchronous axes, refer to section 7.4 in *AX-3 Series Operational Manual*.

2.1.1 MC_Home

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_Home controls the axis to perform the homing operation.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Home		MC_Home_instance(Axis : =, Execute: =, Position: =, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
Position	Specifies the set position. (Unit: user unit)	LREAL	Negative, positive, or 0 (0)	When *Execute* turns to True and *Busy* is False.

- **Outputs**

Name	Function	Data Type	Output range (Default Value)
Done	True when homing is completed.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)

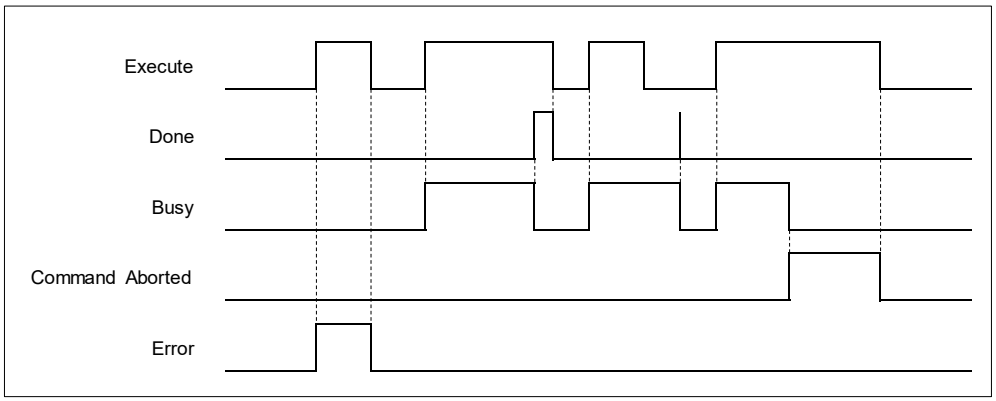
Name	Function	Data Type	Output range (Default Value)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the homing is completed.	• When *Execute* turns from True to False. • If *Execute* is False and *Done* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* turns to True. • When *Error* turns to True. • When *CommandAborted* turns to True.
CommandAborted	• When this instruction is aborted by another instruction. • When this instruction is aborted by MC_Stop.	• When *Execute* changes to False. • If *Execute* is False and *CommandAborted* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Error	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	• When Error Code is cleared.
ErrorID		

■ **Timing Diagram**



- **Inputs/Outputs**

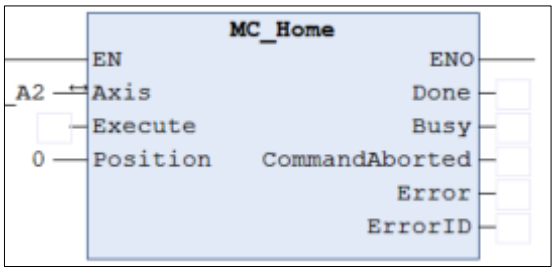
Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* is rising edge triggered and *Busy* is False.

***Note**: AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

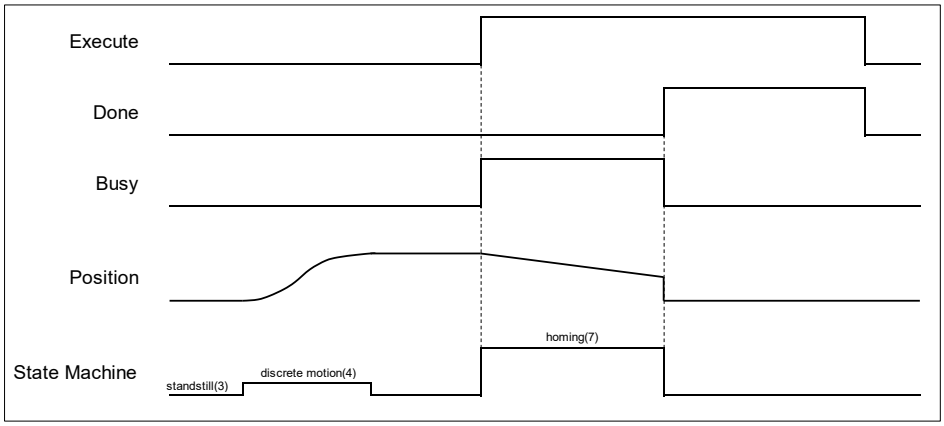
- **Function**
  - When *Execute* changes to True, the homing operation starts to be performed on the axis specified in Axis.
  - Position is to specify the set position for homing.
  - In case the MC_Home command is interrupted by MC_Stop and xWaitForHaltWhenStopInterruptsHome is TRUE, MC_Stop has to wait till the driver reaches velocity zero before setting to Done. Instead, if xWaitForHaltWhenStopInterruptsHome is FALSE, *Done* will shift to true once MC_Stop interrupts MC_Home.
  - When the MC_Home is running, after power off, the Home function block status will change to Error. For SoftMotion V4.10.0.0 and later, when the MC_Home is running, after power off, the Home function block status will change to Abort.

- **Troubleshooting**
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**
  - Example 1: Explain how to perform homing by using MC_Home and setting homing mode on the parameter configuration page with DIAdesigner-AX.



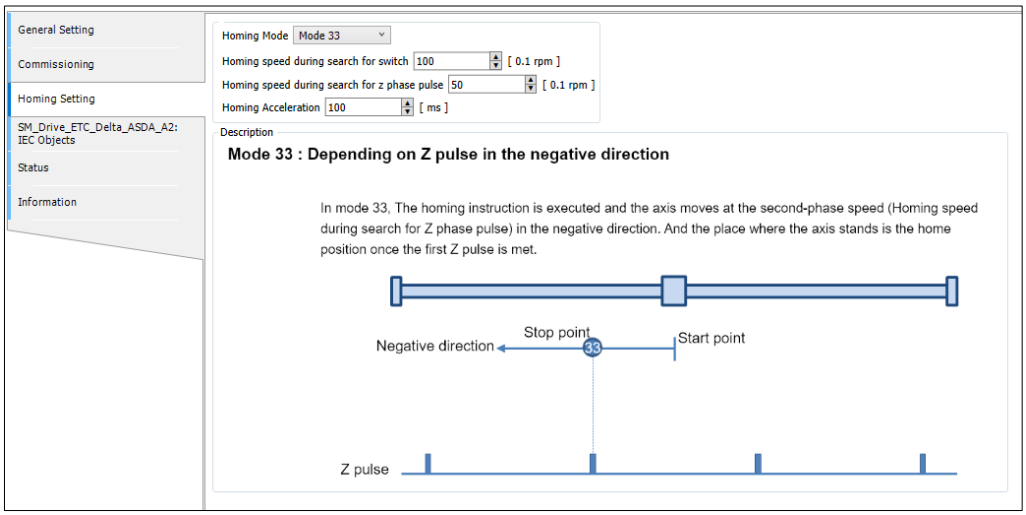
  - **Timing Diagram**



- Use MC_Home with the following parameters:

Homing mode	Speed during search for switch	Speed during search for zero	Homing acceleration
33	100	50	100

- The above parameters can be configured on the Homing Setting page:

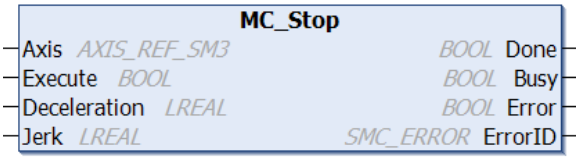


After MC_Home is run, the axis will move in the negative direction till finds the Z phase pulse. And the place where the axis stands is the home position once the first Z pulse is met.

2.1.2 MC_Stop

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_Stop decelerates an axis to a stop.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Stop		MC_Stop_instance(Axis : =, Execute : =, Deceleration : =, Jerk : =, Done =>, Busy =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
Deceleration	Deceleration rate (Unit: user unit/s2)*	LREAL	Positive or 0 (0)	When *Execute* is triggered to run, the rate will be updated.
Jerk	Jerk value (Unit: user unit/s3)*	LREAL	Positive or 0 (0)	When *Execute* is triggered to run, the value will be updated.

- **Outputs**

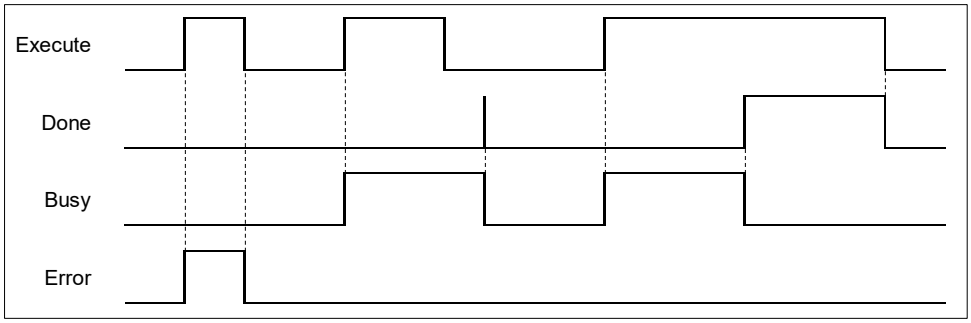
Name	Function	Data Type	Output Range (Default Value)
Done	True when zero velocity is reached.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

  - **Output Updating Time**

Name	Timing for shifting to True	Timing for shifting to False
Done	• True when the axis decelerates to a stop and reaches zero velocity.	• When *Execute* turns from True to False. • If *Execute* is False and *CommandAborted* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Busy	• True when *Execute* turns to True.	• When *Done* turns to True. • When *Error* turns to True.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

■ **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* is triggered to be True and *Busy* is False.

***Note**: AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

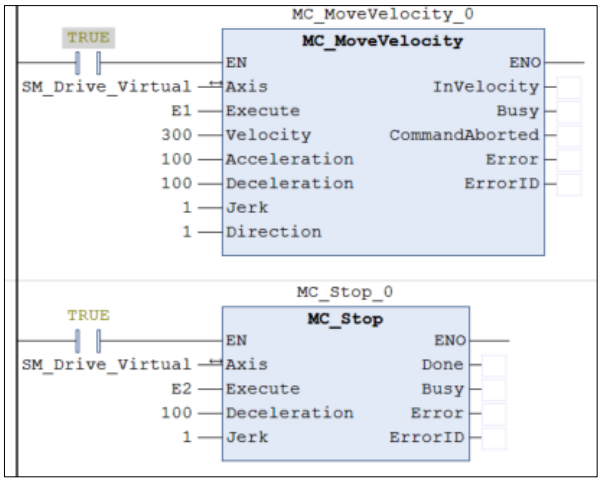
■ You can specify the deceleration rate when decelerating the moving axis to a stop. Also, State Machine will be stopping.
■ When MC_Power is False during deceleration, the motor is in Free Run state.
■ The Done output is set to True when axis has reached velocity zero. At the same time, the input Execute changes to False, while State Machine in stopping state changes to standstill.

- **Troubleshooting**

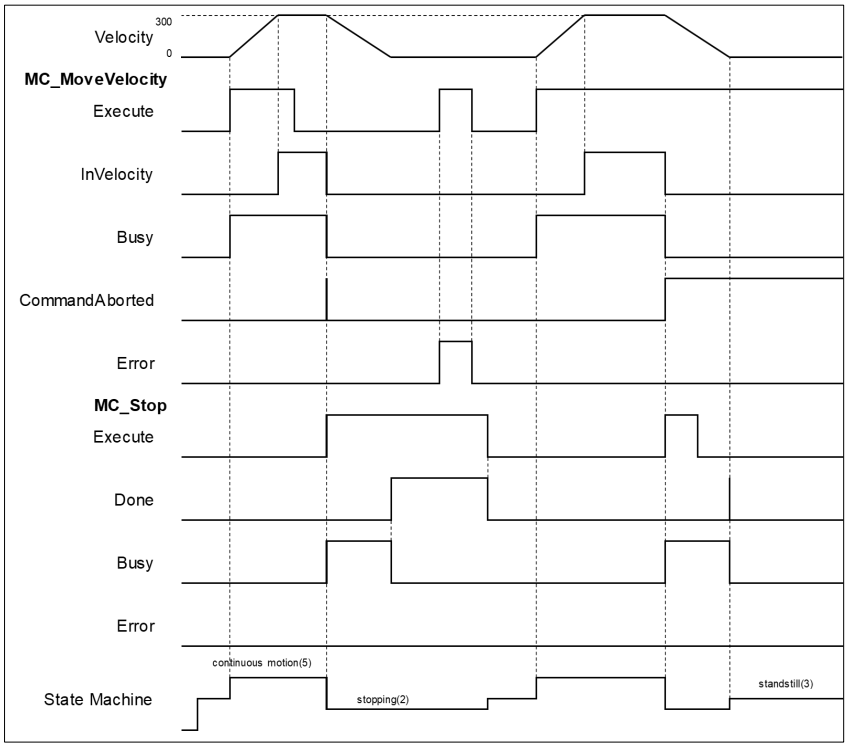
■ If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

■ The example below shows the behavior and position tracking when MC_Stop is run after MC_MoveVelocity.



- **Timing Diagram**



- When *Execute* of MC_Stop changes to True, it triggers *CommandAboted* of MC_MoveVelocity at the same time and the motion controller starts to decelerate the axis to a stop. The Axis state is moved to the “Stopping”.
- When the axis reaches zero velocity, the *Done* output will change to True. *Execute* is still True so the axis state remains in the state “Stopping”. After the stop is finished and *Execute* is False, the axis will change to Standstill.
- In case MC_MoveVelocity runs again while the axis state is “Stopping”, an error will be reported. (Error Code: SMC_AXIS_NOT_READY_FOR_MOTION).

2.1.3 MC_Halt

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_Halt stops the axis motion in a controlled way.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Halt		MC_Halt_instance(Axis : =, Execute : =, Deceleration : =, Jerk : =, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
Deceleration	Deceleration rate. (Unit: user unit/s2)	LREAL	Positive number or 0 (0)	When *Execute* turns to True, the rate will be updated.
Jerk	Jerk value. (Unit: user unit/s3)	LREAL	Positive number or 0 (0)	When *Execute* is triggered to be True, the value will be updated.

- **Outputs**

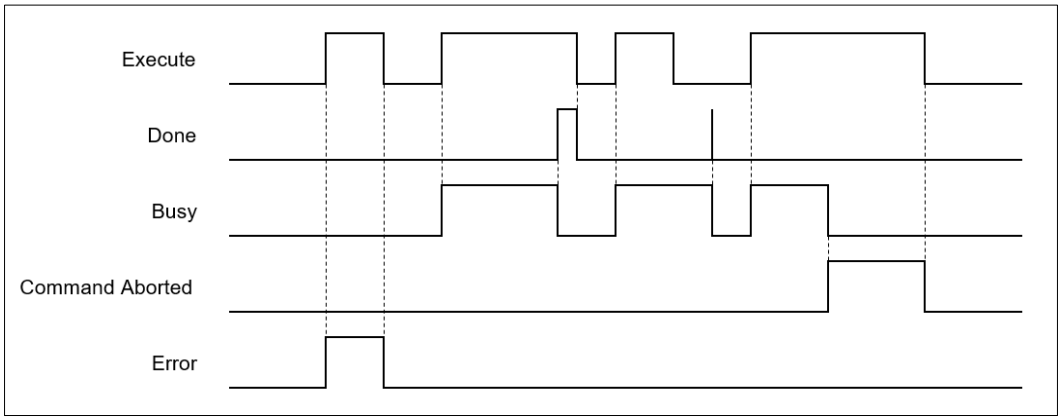
Name	Function	Data Type	Output Range (Default Value)
Done	True when zero velocity is reached.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

■ **Output Updating Time**

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the axis decelerates to a stop and reaches zero velocity.	• When *Execute* turns from True to False. • If *Execute* is False and *CommandAborted* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Busy	• When *Execute* turns to True.	• When *Done* turns to True. • When *Error* turns to True. • When *CommandAborted* turns to True.
CommandAborted	• When this instruction is aborted because of other function blocks.	• When *Execute* turns from True to False. • If *Execute* is False and *CommandAborted* turns to True, it will be True for only one period and immediately shift to False.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

■ **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* is triggered to be True and *Busy* is False

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

■ Any next motion command can be run when MC_halt is in Standstill mode (opposite to MC_Stop, which cannot be interrupted by other motion FBs.).

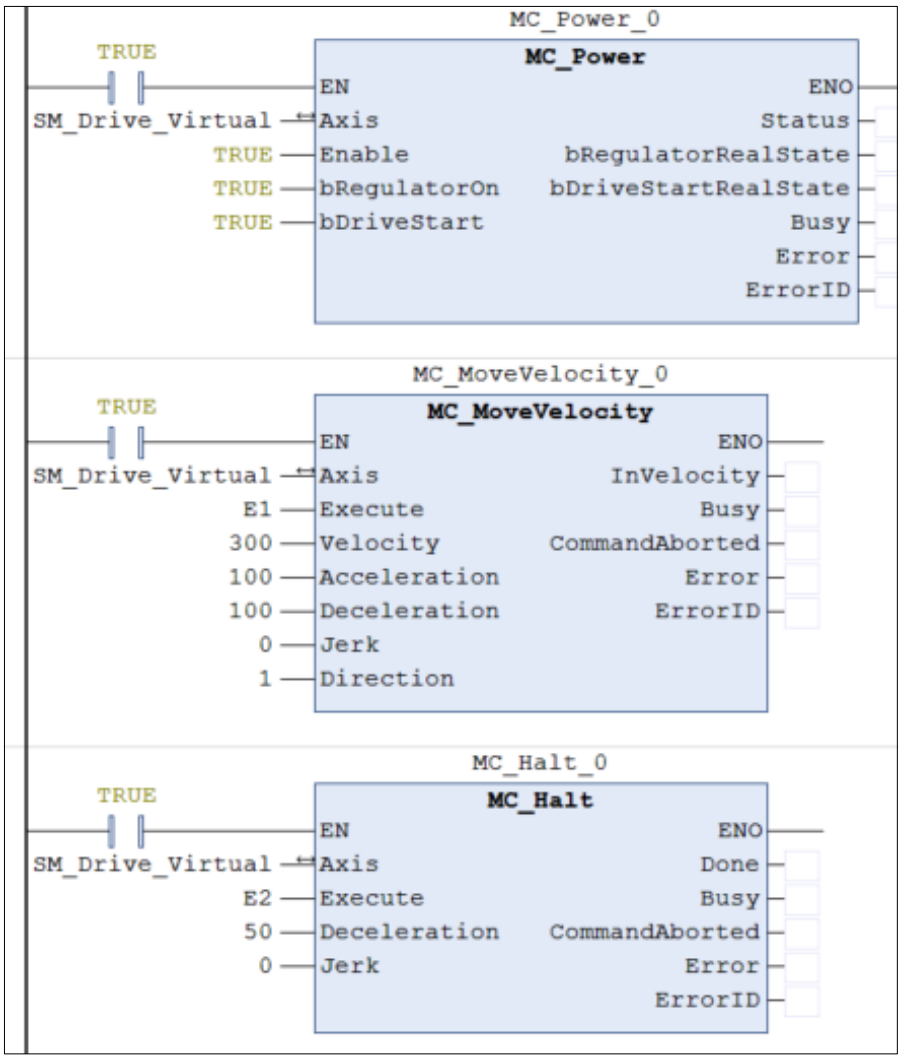
■ When MC_Halt is run, the axis will enter the discrete_motion state. Once the axis reaches zero, the axis state will transfer to Standstill.

- **Troubleshooting**

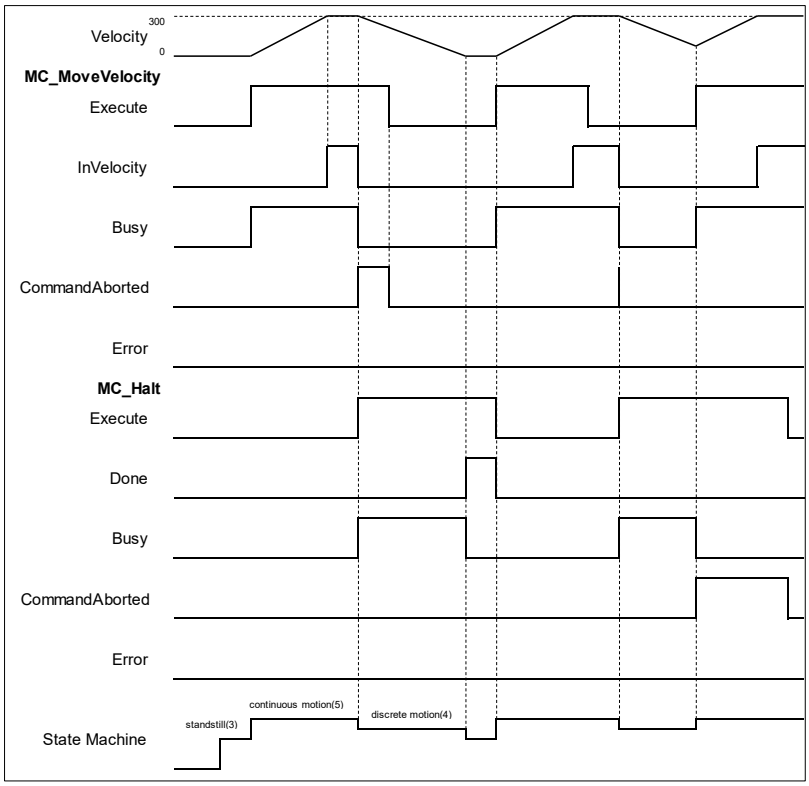
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

  - The example below shows the behavior and position tracking when MC_Halt is run after MC_MoveVelocity.
  - The MC_Halt stops MC_MoveVelocity if there is no another instruction run before the axis enters “Standstill” state.l”.
  - If MC_MoveVelocity runs again during the deceleration, it will abort MC_Halt immediately and accelerate again without entering “Standstill” state. This re-execution behavior is allowed for MC_Halt but not allowed iMC_Stop.



  - **Timing Diagram**

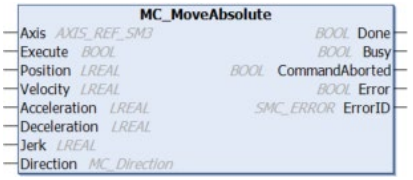


- When *Execute* of MC_Halt changes to True, it triggers *CommandAborted* of MC_MoveVelocity at the same time, and the motion controller starts to decelerate the axis to a stop. The Axis state changes to the “DiscreteMotion”.
- When the axis reaches zero velocity, *Done* will change to True. The axis state will change to “Standstill”.
- When MC_Halt is not decelerating the axis to zero velocity and Execution is True, the Execute input of MC_MoveVelocity will change to True again and stop MC_Halt. Which CommandAboted will change to True with the axis state transferred from discrete_motion to continuous_motion.

2.1.4 MC_MoveAbsolute

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_MoveAbsolute controls the axis to move to the specified absolute target position at a specified behavior.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_MoveAbsolute		MC_MoveAbsolute_instance(Axis : =, Execute : =, Position : =, Velocity : =, Acceleration : =, Deceleration : =, Jerk : =, Direction : =, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
Position	Absolute target position (Unit: user unit)	LREAL	Negative, positive or 0 (0)	When *Execute* turns to True and *Busy* is False.
Velocity	Target velocity (Unit: user unit/s)	LREAL	Positive or 0 (0)	When *Execute* turns to True and *Busy* is False.
Acceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive or 0 (0)	When *Execute* turns to True and *Busy* is False.
Deceleration	Deceleration rate (Unit: user unit/s^2)	LREAL	Positive or 0 (0)	When *Execute* turns to True and *Busy* is False.
Jerk	Jerk value (Unit: user unit/s^3)	LREAL	Positive or 0 (0)	When *Execute* turns to True and *Busy* is False.
Direction	Rotation direction	MC_Direction*	3: fastest 2: current 1: positive 0: shortest -1: negative (shortest)	When *Execute* turns to True and *Busy* is False.

***Note:** MC_Direction: Enumeration (Enum)

- **Outputs**

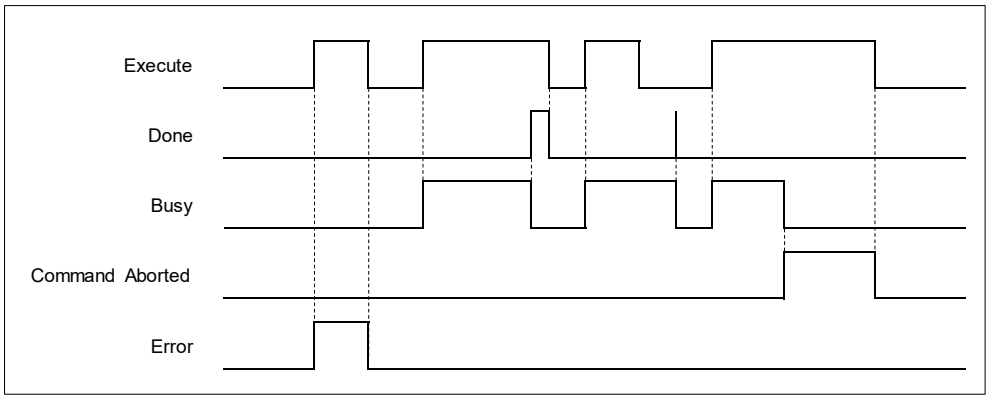
Name	Function	Data Type	Output Range (Default Value)
Done	True when absolute target position is reached.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the axis is being controlled	BOOL	True/False (False)
Error	True if an error occurs	BOOL	True/False (False)
ErrorID	Record the error code when the error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• When the absolute positioning is completed.	• When *Execute* turns to False • If *Execute* is False and *CommandAborted* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* turns to True. • When *Error* turns to True. • When *CommandAborted* turns to True.
CommandAborted	• When this instruction is aborted by another function block. • When this instruction is aborted because of the execution of MC_Stop instruction.	• When *Execute* turns to False. • If *Execute* is False and *CommandAborted* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

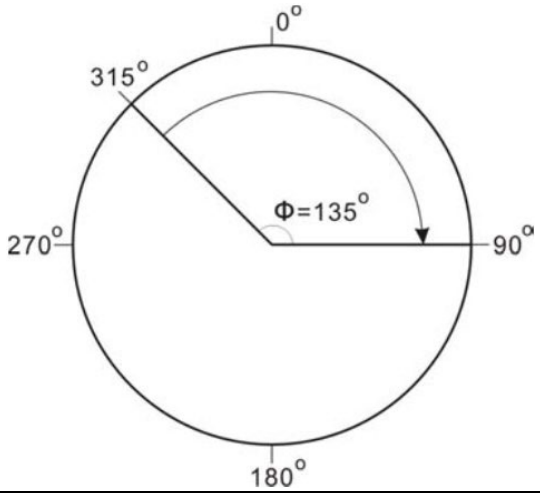
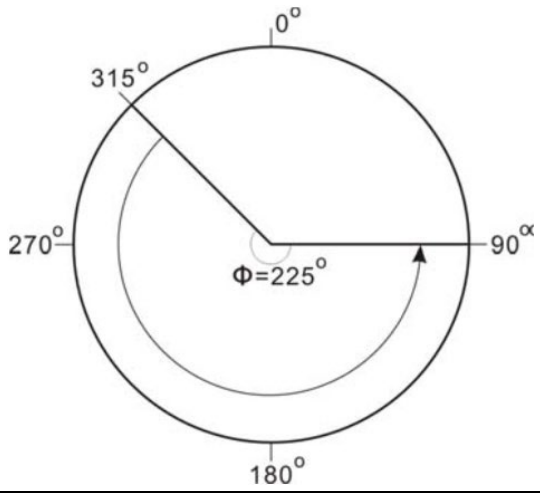
Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* is triggered to be True and Busy is False.

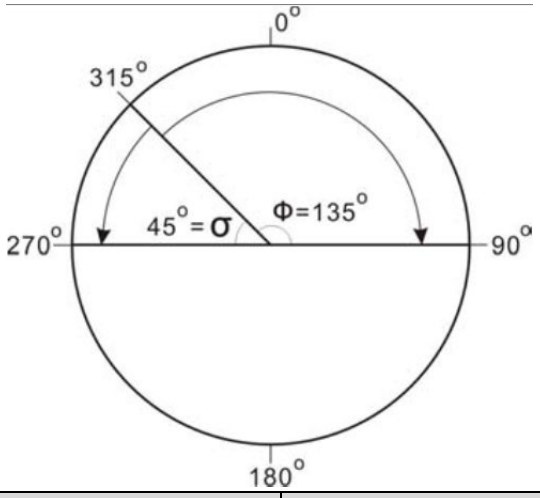
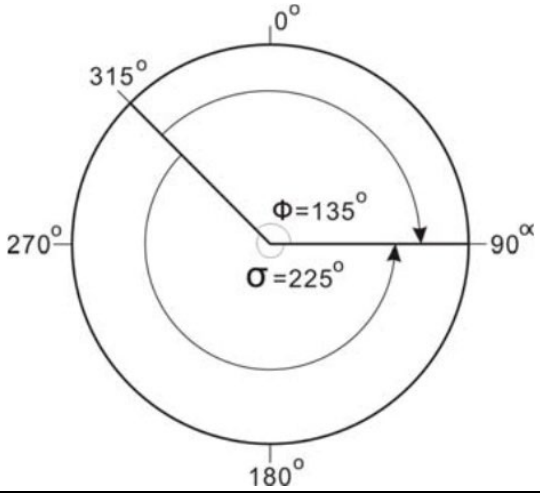
***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

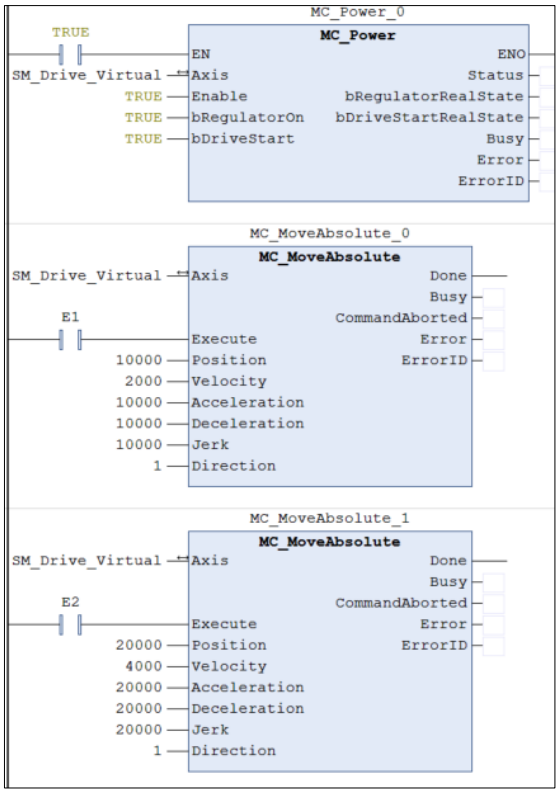
  - **Direction**

    - Direction is used to define the rotation of servo axis and is effective only for modulo/rotary axis.
    - When the direction value is different, the motion direction and the travel distance of the rotary axis will be different as follows. Suppose the output unit of the physical device is “degree”, the motion direction of the rotary axis is illustrated as follows:

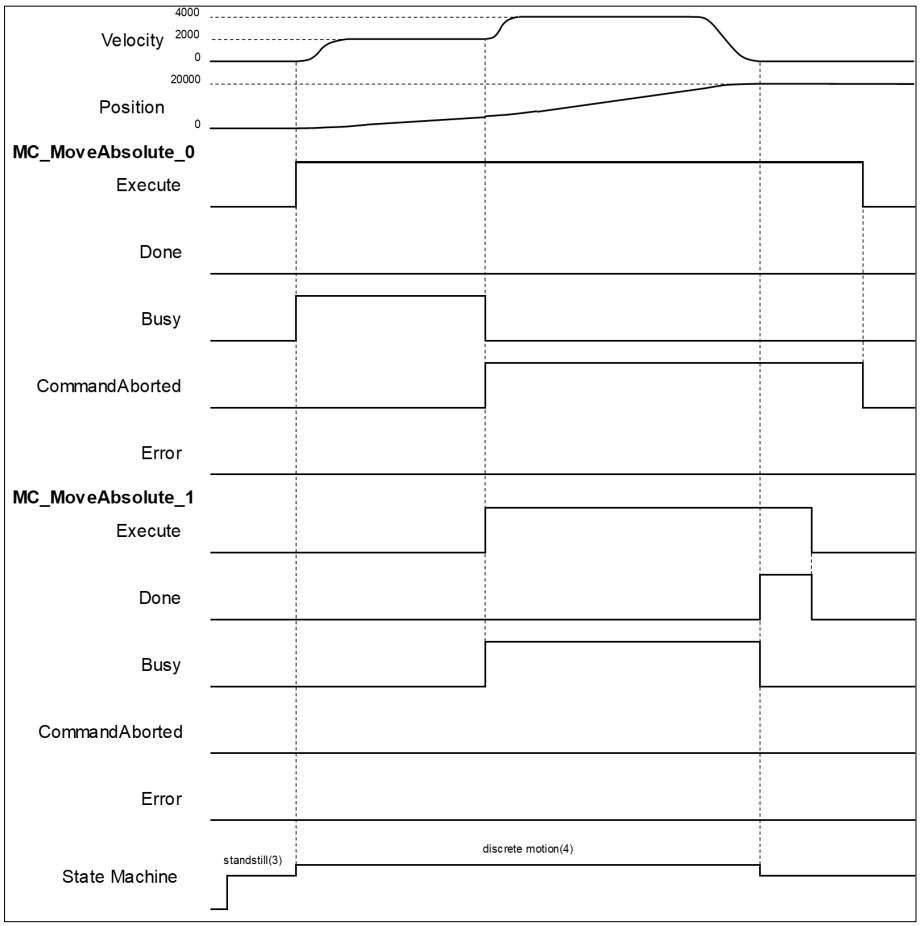
Direction: 1 (Positive direction) **Current position: 315°** **Target position: 90°** **Movement angle: 135°**	**Direction: -1 (Negative direction)** **Current position: 315°** **Target position: 90°** **Movement angle: 225°**
	

Direction: 0(shortest), 3(fastest) **Current position: 315°** **Target position: 90°** **Movement angle: 135°**	**Direction: 0(shortest), 3(fastest)** **Current position: 315°** **Target position: 270°** **Movement angle: 45°**
	
Direction: 2(current) **Rotary axis status: Moving in the negative direction before the function block is run.** **Current position: 315°** **Target position: 90°** **Movement angle: 225°**	**Direction: 2(current)** **Rotary axis status: motionless or moving in the positive direction before the function block is run.** **Current position: 315°** **Target position: 90°** **Movement angle: 135°**
	

- **Troubleshooting**
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.
- **Example**
  - The example below describes the behavior of 2 MC_MoveAbsolute instructions which are connected with each other.



- **Timing Diagram**



- If Execute is True when MC_MoveAbsolute_0 block is running, the axis will move towards the target position. Once the Execute input of MC_MoveAbsolute_1 changes to True, the execution of MC_MoveAbsolute_0 block will be aborted, which CommandAborted turns True. The final position will be 20,000.
- When MC_MoveAbsolute_1 block is run, the axis will move towards the absolute target position according to MC_MoveAbsolute_1 parameters.
- When axis reaches the absolute position 20000 set by MC_MoveAbsolute_1, the Done input of MC_MoveAbsolute_1 will turn True as Busy changing to False.
- In case *Execute* of MC_MoveAbsolute_1 switches to False, the Done output will also change to False state.

2.1.5 MC_MoveRelative

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_MoveRelative controls the axis to move a specified relative distance with a specified behavior.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_MoveRelative		MC_MoveRelative_instance(Axis : =, Execute : =, Distance : =, Velocity : =, Acceleration : =, Deceleration : =, Jerk : =, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
Distance	Relative distance to be moved. (Unit: user unit)	LREAL	Negative, positive or 0(0)	When *Execute* turns to True and *Busy* is False.
Velocity	Target velocity. (Unit: user unit/s)	LREAL	Positive or 0(0)	When *Execute* turns to True and *Busy* is False.
Acceleration	Acceleration rate. (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Deceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	True when relative distance is completed.	BOOL	True/False (False)

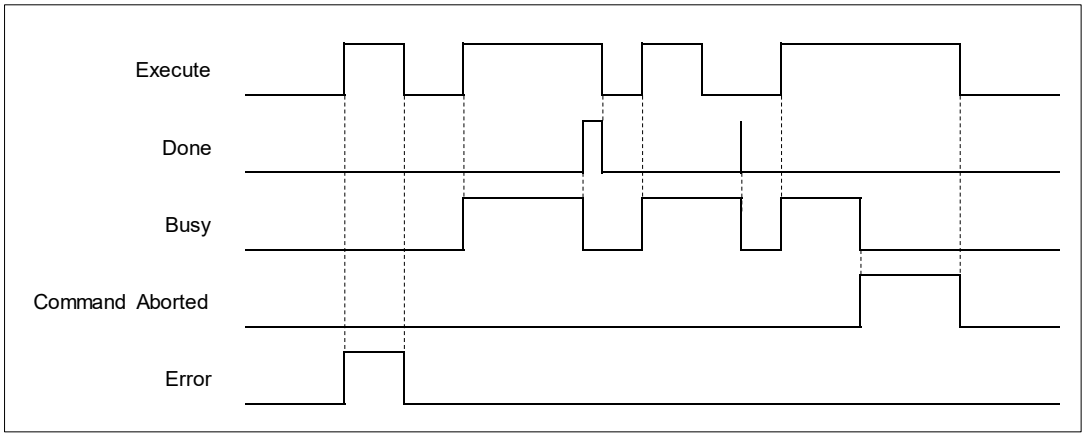
Name	Function	Data Type	Output Range (Default Value)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• When the relative positioning is completed.	• When *Execute* turns from True to False. • If *Execute* is False and *Done* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* changes to True. • When *Error* changes to True. • When *CommandAborted* turns to True.
CommandAborted	• When this instruction is interrupted by another function block. • When this instruction is interrupted because of the execution of MC_Stop instruction.	• When *Execute* changes to False. • If *Execute* is False and *CommandAborted* turns to True, it will be True for only one period and immediately shift to False.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and Busy is False.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

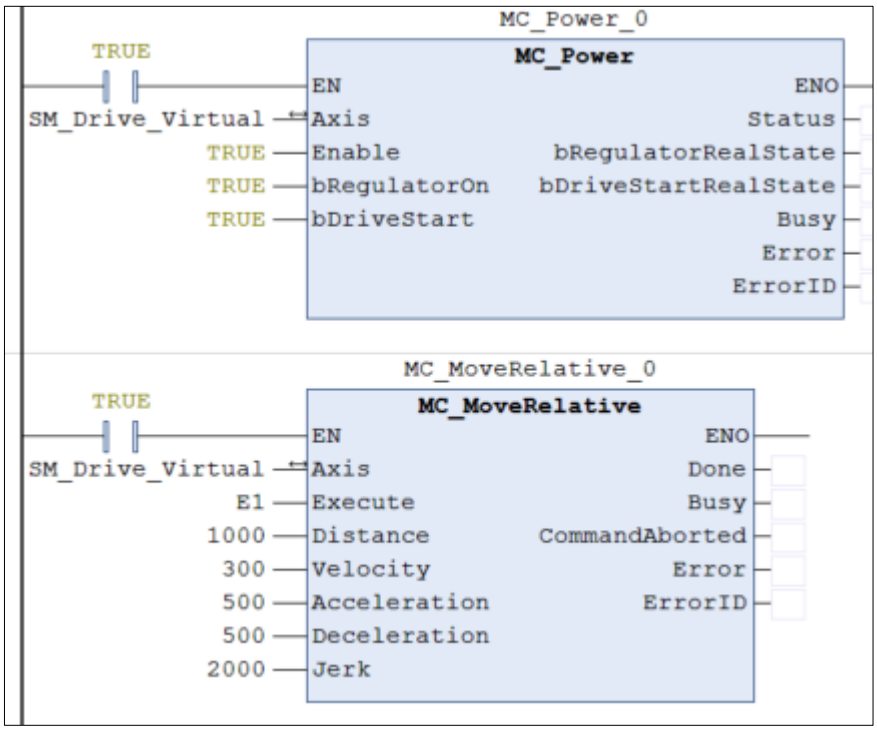
  The instruction performs relative positioning with specified target velocity (Velocity), acceleration rate (Acceleration), deceleration rate (Deceleration) and Jerk value (Jerk) when execute changes to True.

- **Troubleshooting**

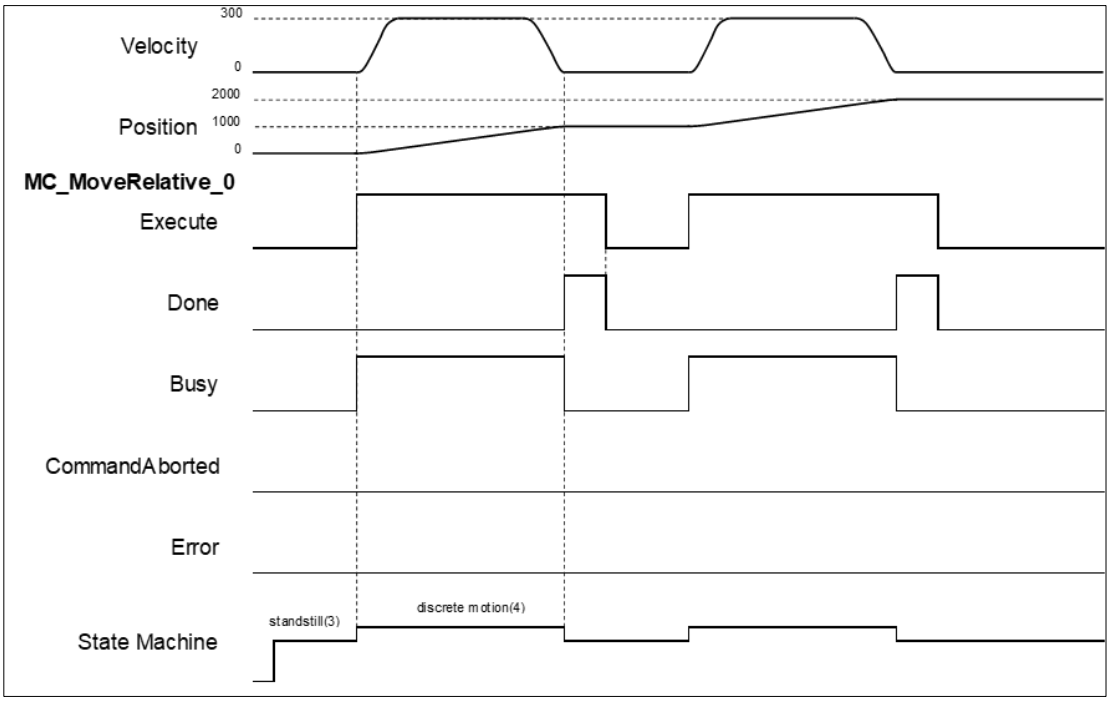
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

  - The example below describes the behavior of the MC_MoveRelative instruction.



- **Timing Diagram**



- When *Execute* changes to True, MC_MoveRelative drives the axis to the target position. During movement, *Busy* is True in the state of Discrete motion.
- When the axis moved the specified relative distance (1,000), *Done* changes to True, and *Busy* changes to False.
- When *Execute* changes to False, *Done* changes to False too.
- When *Execute* changes to True again, the instruction will be run again to drive the axis to the target position and reach the position of 2,000.

2.1.6 MC_MoveAdditive

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_MoveAdditive controls the axis to move an additional distance at a given speed and acceleration.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_MoveAdditive		MC_MoveAdditive_instance(Axis : =, Execute : =, Distance : =, Velocity : =, Acceleration : =, Deceleration : =, Jerk : =, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
Distance	Relative distance to be moved. (Unit: user unit)	LREAL	Negative, positive or 0(0)	When *Execute* turns to True and *Busy* is False.
Velocity	Target velocity. (Unit: user unit/s)	LREAL	Positive or 0(0)	When *Execute* turns to True and *Busy* is False.
Acceleration	Acceleration rate. (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Deceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	True when additive distance is completed.	BOOL	True/False (False)

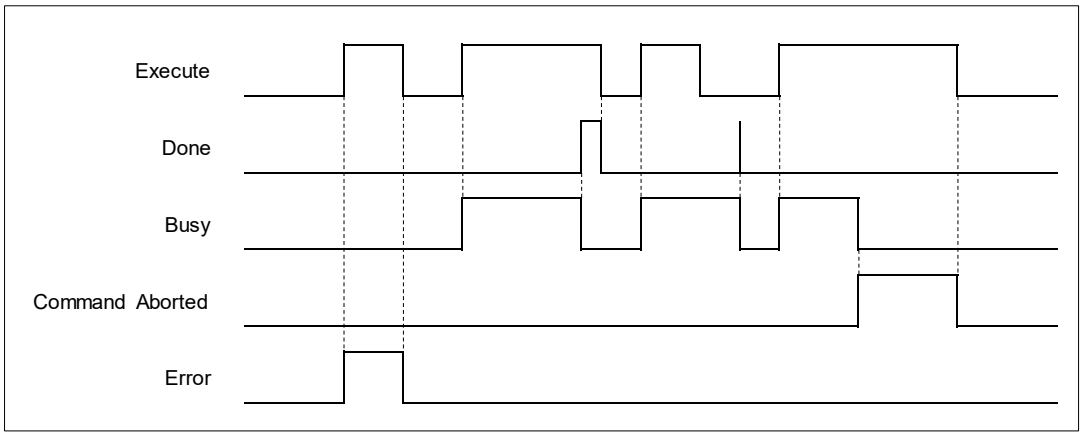
Name	Function	Data Type	Output Range (Default Value)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Timing**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• True when the additive positioning is completed.	• When *Execute* changes to False. • If *Execute* is False and *CommandAborted* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Busy	• True when *Execute* changes to True.	• When *Done* changes to True. • When *Error* changes to True. • When *CommandAborted* changes to • True.
CommandAborted	• When this instruction is aborted because of the execution of MC_Stop instruction.	• When *Execute* changes to False. • If *Execute* is False and *CommandAborted* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and Busy is False.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

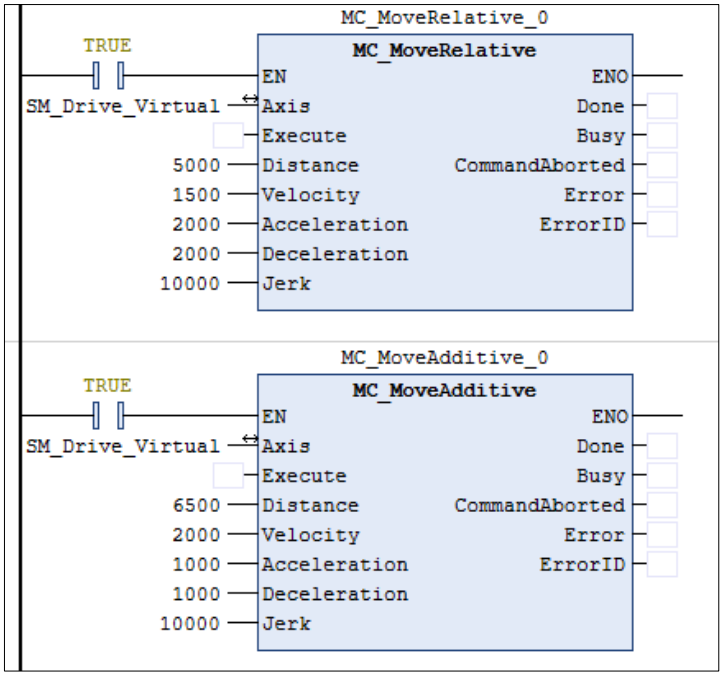
  - MC_MoveAdditive runs the instruction based on user-defined parameters to drive the specific axis to move an additional distance.
  - When MC_MoveAdditive runs alone, the behavior will be identical to a MC_MoveRelative.
  - In case the previous instruction is on-going, an additional distance will be added again to run MC_MoveAdditive instruction.

- **Troubleshooting**

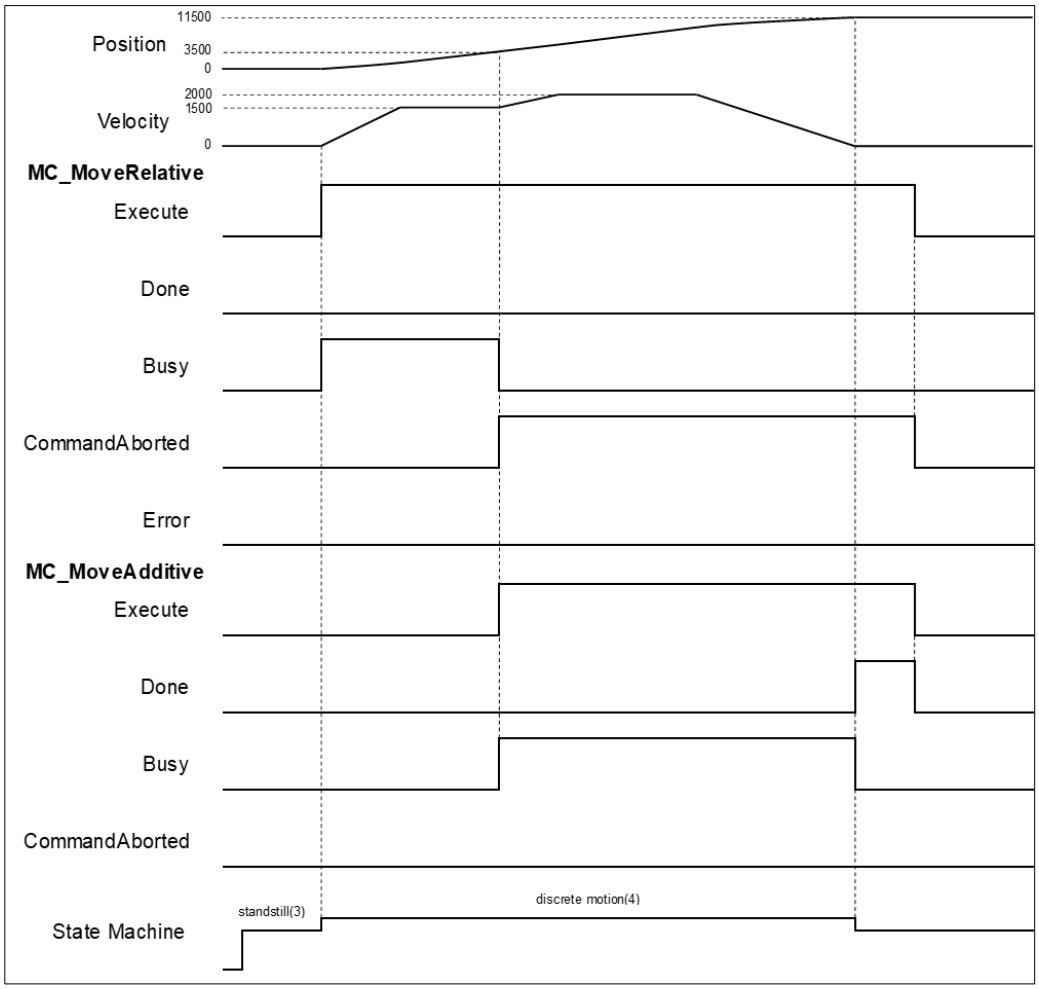
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

  - The example below describes the behavior of MC_MoveRelative and MoveAdditive instructions which are run in a series.



- **Timing Diagram**

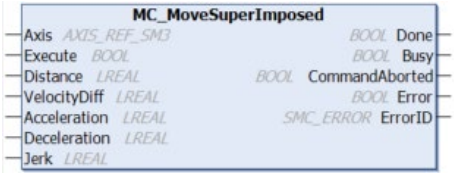


- When *Execute* changes to True, MC_MoveRelative drives the axis to the target position. After *Execute* changes to True at the position 3500, the MC_MoveRelative instruction will be aborted and *CommandAborted* changes to True. At the same time, the axis remains in Discrete motion state.
- Meanwhile, the MC_MoveAdditive instruction is run and adds a relative distance of 6,500 to the previous target position 5,000, and the new target position 11,500.
- When the axis reaches 13,500, Done changes to True.

2.1.7 MC_MoveSuperImposed

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_MoveSuperimposed controls the axis to move a relative superimposed distance at a specified behavior while the axis is moving.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_MoveSuperImposed		MC_MoveSuperImposed _instance(Axis : =, Execute : =, Distance : =, VelocityDiff : =, Acceleration : =, Deceleration : =, Jerk : =, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
Distance	Additional relative distance to be moved. (Unit: user unit)	LREAL	Negative, positive or 0 (0)	When *Execute* turns to True and *Busy* is False.
VelocityDiff	Additional target velocity (Unit: user unit/s)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Acceleration	Additional acceleration rate (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Deceleration	Additional deceleration rate (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Jerk	Additional jerk value (Unit: user unit/s^3)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)

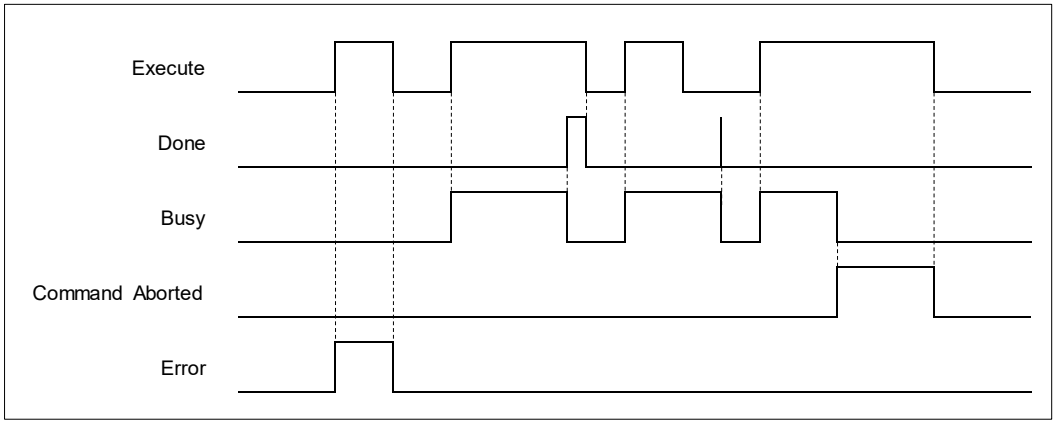
Done	True when the superimposed movement is completed.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• When the superimposed distance is completed.	• When *Execute* turns from True to False. • If Execute is False and Done turns to True, Done will be True for only one scan cycle and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* changes to True. • When *Error* changes to True • When *Commandaborted* turns to True
CommandAborted	• When one instruction is aborted by another instruction with the Buffer Mode set to Aborting. • When this instruction is aborted because of the execution of MC_Stop instruction.	• When *Execute* changes to False. • If *Execute* is False and *CommandAborted* turns to True, it will be True for only one period and immediately shift to False.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and Busy is False.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

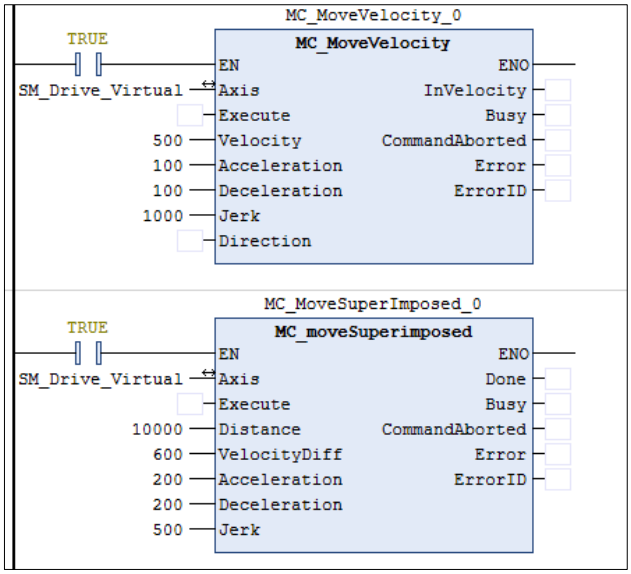
  - The input values of VelocityDiff, Distance, *Acceleration*, *Deceleration* and *Jerk* are superimposed on the on-going motion of the previous instruction.
  - If executing MC_MoveSuperImposed block in Standstill state, the function will be identical to MC_MoveRelative.
  - MC_MoveSuperImposed can be aborted by other function blocks.
  - An error will occur when MC_MoveSuperImposed is repeatedly run on the same axis.
  - If changing the input values during the execution of MC_MoveSuperImposed or re-execute the function block before the instruction finished, the axis will react according to the new superimposed values and instruction, which are the sum of the previous instruction and MC_MoveSuperimposed instruction. When the superimposed distance is reached, the axis will resume the operation of the previous instruction until the superimposed total distance is reached.
  - MC_MoveSuperimposed and the function block, which is previously run, will be interrupted if a new function block has started while MC_MoveSuperimposed is superimposed on other function blocks.

- **Troubleshooting**

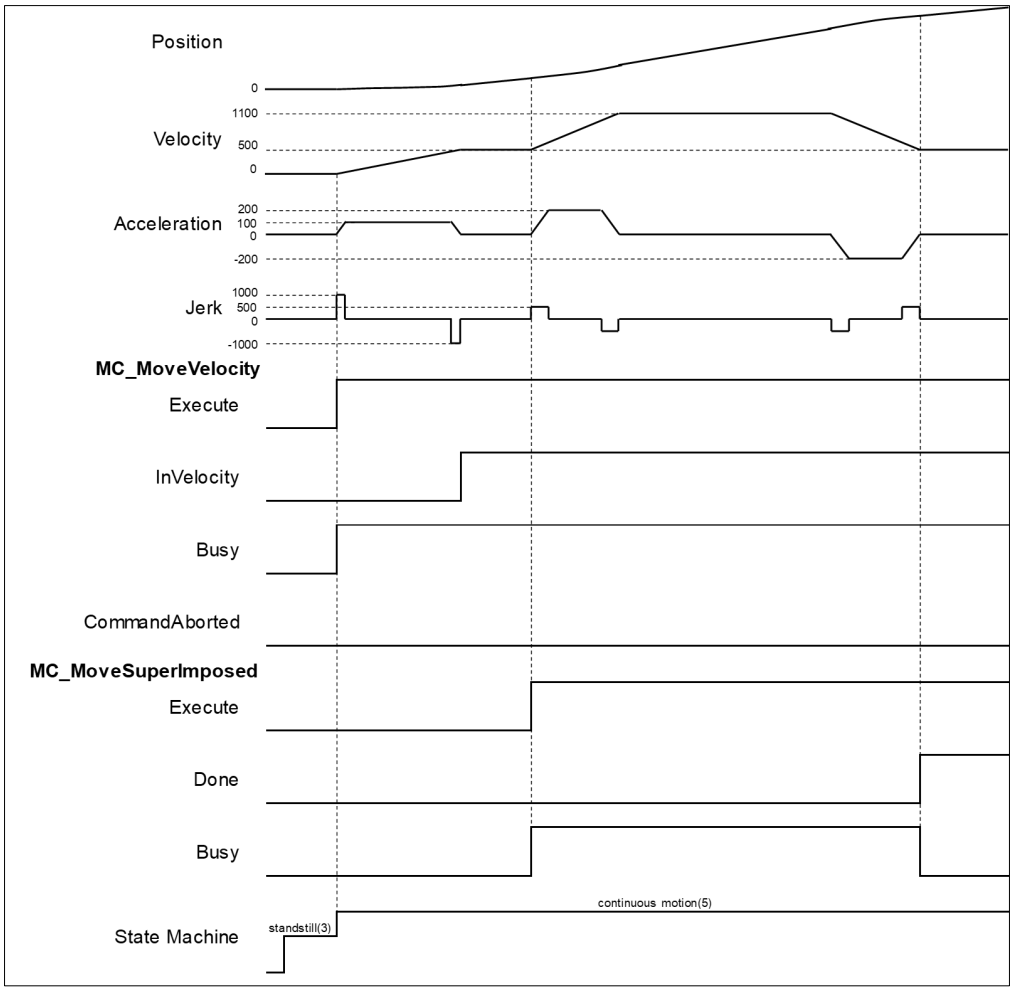
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

  - The example below describes the behavior of MC_MoveSuperImposed applied to MC_MoveVelocity.



  - **Timing Diagram**

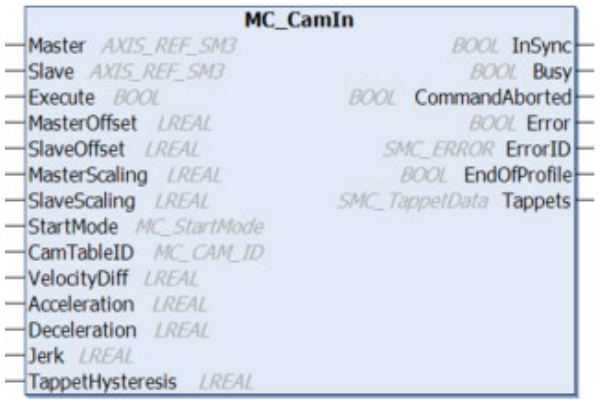


- When *Execute* of MC_MoveVelocity changes to True, the specific axis starts to move towards the target velocity (500) at the constant speed.
- When *Execute* of MC_MoveSuperImposed changes to True, the MC_MoveSuperImposed instruction starts and applies the additional values (velocity, distance, acceleration, deceleration and jerk) to the axis and the axis performs a superimposed motion path. Since VelocityDiff is set as 600 and the target superimposed distance is far enough, the velocity will be superimposed to 1100(500 + 600).
- When the execution of MC_MoveSuperImposed has finished, *Done* will turn True and MC_MoveVelocity will keep going.

2.1.8 MC_CamIn

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_CamIn performs cam operation.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_CamIn		MC_CamIn_instance(Master : =, Slave : =, Execute : =, MasterCompensation : =, SlaveCompensation : =, MasterScaling : =, SlaveScalling : =, StarMode : =, CamTableID : =, VelocityDiff : =, Acceleration : =, Deceleration : =, Jerk : =, TappetHysteresis : =, InSync =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>, EndOfProfile =>, Tappets =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
MasterCompensation	Turns the position of the master axis by the specified Compensation value. (Unit: user unit)	LREAL	Negative, positive or 0 (0)	When *Execute* turns to True and *Busy* is False.
SlaveCompensation	Turns the displacement of the slave axis by the specified Compensation value. (Unit: user unit)	LREAL	Negative, positive or 0 (0)	When *Execute* turns to True and *Busy* is False.
MasterScaling	Scales the master axis up and down with the specified factor.	LREAL	Negative, positive or 0 (0)	When *Execute* turns to True and *Busy* is False.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
SlaveScaling	Scales the slave axis up and down with the specified factor.	LREAL	Negative, positive or 0 (0)	When *Execute* turns to True and *Busy* is False.
StartMode	Specifies the engagement behavior of the slave axis	MC_StartMode	0: absolute 1: relative 2: ramp_in 3: ramp_in_pos 4: ramp_in_neg (absolute)	When *Execute* turns to True and *Busy* is False.
CamTableID	Cam table identifier, which is from output of CamTableSelect.	MC_ CAM_ID	MC_CAM_ID*	When *Execute* turns to True and *Busy* is False.
VelocityDiff	Maximum velocity difference under ramp_in mode. (Unit: user unit/s)	LREAL	Positive or 0(0)	When *Execute* turns to True and *Busy* is False.
Acceleration	The acceleration rate under ramp_in mode. (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Deceleration	The deceleration rate under ramp_in mode. (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Jerk	The jerk value under ramp_in mode. (Unit: user unit/s^3)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
TappetHysteresis	The hysteresis rate of tappet.	LREAL	Positive or 0(0)	When *Execute* turns to True and *Busy* is False.

***Note:** MC_CAM_ID(Struct): Cam table variables, from output of MC_CAMTableSelect, are input to MC_CamIn.

Name	Function	Data Type	Setting Value (Default Value)
pCT	Internal information stored in the cam table	POINTER TO BYTE	Positive or 0(0)
Periodic	Periodic mode	BOOL	True/False (True)
MasterAbsolute	MasterAbsolute mode	BOOL	True/False (True)
SlaveAbsolute	SlaveAbsolute mode	BOOL	True/False (True)
StartMaster	The start master axis position in the cam table	LREAL	Negative, positive or 0 (0)
EndMaster	The end master axis position in the cam table	LREAL	Negative, positive or 0 (0)
StartSlave	The start slave axis position in the cam table	LREAL	Negative, positive or 0 (0)

Name	Function	Data Type	Setting Value (Default Value)
EndSlave	The end slave axis position in the cam table	LREAL	Negative, positive, or 0(0)
byCompatibilityMode	Compatibility mode	BYTE	Positive or 0(0)

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
InSync	True when the specified master/slave cam operation is synchronized.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when this instruction is aborted.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*1	SMC_ERROR (SMC_NO_ERROR)
EndOfProfile	True when the end point of the cam profile is completed.	BOOL	True/False (False)
Tappets	Can be used with the function block of SMC_GetTappetValue.	SMC_TappetData*2	SMC_TappetData

***Note:**

1. SMC_ERROR: Enumeration (Enum)
2. SMC_TappetData: Structure(Struct)

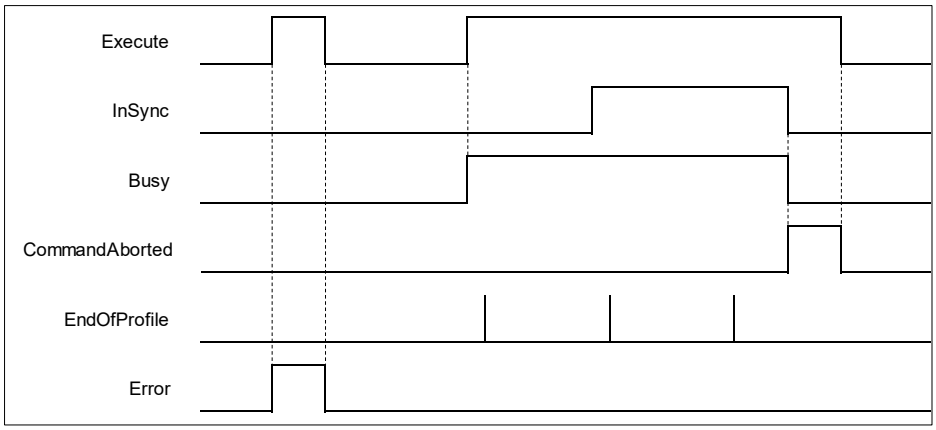
Name	Function	Data Type	Output Range (Default Value)
ctt	Tappet action active when axis passes tappets in the specified direction (positive or negative).	SMC_CAMTAPPETTYPE	0: TAPPET_pos (Pass in positive direction) 1: TAPPET_all (No specific direction) 2: TAPPET_neg (Pass in negative direction) (TAPPET_pos)
cta	The action activated when axis passes tappets.	SMC_CAMTAPPETACTION	0: TAPPETACTION_on (Switch ON) 1: TAPPETACTION_off (Switch OFF) 2: TAPPETACTION_inv (Inverts) 3: TAPPETACTION_time (Switches on after a delay for a certain

Name	Function	Data Type	Output Range (Default Value)
			time period.) (TAPPETACTION_on)
dwDelay	Specify the delay time for switching ON under TAPPETACTION_time mode.	DWORD	Positive or 0(0)
dwDuration	Specify the time duration for which the tappet is switched to ON under TAPPETACTION_time mode.	DWORD	Positive or 0(0)
iGroupID	Track ID of tappets	INT	Positive, negative, or 0 (0)
x	Master position where tappet is switched.	LREAL	Positive, negative, or 0 (0)
dwActive	Internal variable	DWORD	Positive or 0(0)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
InSync	• When the synchronization between the master and slave axis is completed.	• When *Execute* is False. • If *Execute* is False and *Done* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Busy	• When an instruction is being run.	• When *CommandAborted* is True. • When Error is True.
CommandAborted	• When MC_CamOut is run. • When one instruction is aborted by another instruction. • When a function block instruction is aborted by MC_Stop.	• When *Execute* is False. • If *Execute* is False and *CommandAborted* turns to True, *CommandAborted* will be True for only one period and immediately shift to False.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* is False. (Error codes are cleared.)
ErrorID		
EndOfProfile	• Cyclic end of the cam profile	• Shift to True for only one cycle and immediately shift to False if MC_CamTableSelect Periodic is 1 (cycle). • *Execute* shifts to False if MC_CamTableSelect Periodic is 0 (none-cycle).

- **Timing Diagram**



- When *Execute* turns from FALSE to TRUE and Busy is TRUE, *InSync* turns from False to True as soon as the synchronization between master and slave axis is completed. When coming to the end of CAM cycle, *EndOfProfile* turns from FALSE to TRUE for only one period, then switch back to FALSE. Once the meshing of master and slave axis is deactivated, such as executing MC_CamOut, *CommandAborted turns from* FALSE to TRUE, while both *InSync* and *Busy* turns from TRUE to FALSE. Then, *CommandAborted* will shift from TRUE to FALSE as well as *Execute*.

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Master	Specifies the master axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and *Busy* is False.
Slave	Specifies the slave axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and *Busy* is False.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

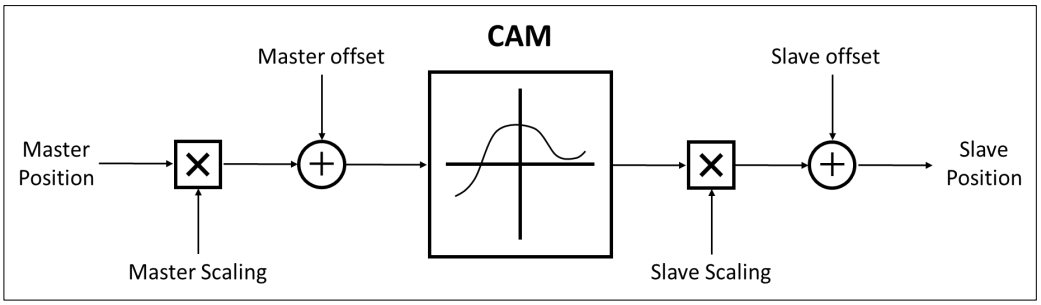
  - **Relationship between master axis position and slave axis position**

    - The cam relationship which is planned in the software is the position relationship between the master axis and slave axis. The “position” mentioned here is the cam phase of the master axis / slave axis instead of the actual axis position. If the cam relationship which is planned is seen as the function CAM as below, the input of the function CAM is the master axis cam phase and the output is the slave axis cam phase. The formula is shown as below.

      y = CAM(x)

      x: The master axis cam phase

      y: The slave axis cam phase
    - The cam phase comes from the axis positions and there is a conversion between them. The conversion between the axis position and cam phase is related with the *MasterAbsolute*, *SlaveAbsolute*, *MasterCompensation*, *SlaveCompensation*, *MasterScaling*, and *SlaveScaling*.
    - The slave axis follows the master axis to make the synchronous cam motion by using the MC_CamIn instruction. In the synchronous cam motion, the corresponding relationship between the master axis position and slave axis position is based on the pre-planned cam relationship (the cam curve or cam table). The process in which the slave axis position is calculated through the master axis position is illustrated as follows.



- The following formula is generated from the figure above:
  Position_Slave=SlaveScaling×CAM(MasterScaling×MasterPosition+MasterCompensation)+ SlaveCompensation
  It can be seen that when master axis is in absolute mode, master position is the remainder of the current master position divided by modulo; When master axis is in relative mode, master position is the start point position (usually 0) of master axis in the corresponding cam curve.

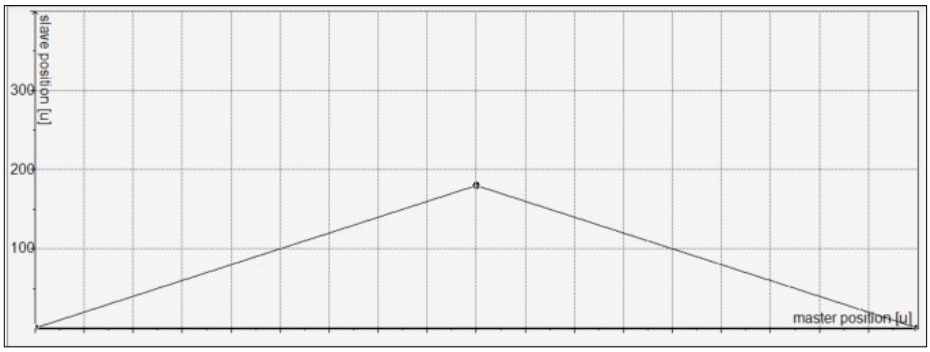
■ **Relation between StartMode and MasterAbsolute/ SlaveAbsolute of CamTableSelect**

- Absolute mode (StartMode = 0): The slave current position is not involved in the Cam calculation as the cam synchronization starts, However, a jump can be caused if the current position of slave axis and its start position from the Cam are not the same
- Relative mode (StartMode = 1): Cam changes according to the current position of slave axis, which the slave axis position will be added to the slave current position for calculation. If the slave setpoint position according to the cam is not at the start point 0, a jump may occur.
- Ramp mode (StartMode = 2, 3, 4): Add a motion curve for compensation according to *VelocityDif, Acceleration, Deceleration, and Jerk,* so as to prevent a jump in cam when meshing starts.

MC_CamTableSelect.MasterAbsolute	Master axis
absolute	Absolute mode
relative	Relative mode

MC_CamIn.StartMode	MC_CamTableSelect.SlaveAbsolute	Slave axis
absolute	True	Absolute mode
absolute	False	Relative mode
relative	True	Relative mode
relative	False	Relative mode
ramp_in	True	Ramp in Absolute mode
ramp_in	False	Ramp in Relative mode
ramp_in_pos	True	Ramp in positive Absolute mode
ramp_in_pos	False	Ramp in positive Relative mode
ramp_in_neg	True	Ramp in negative Absolute mode
ramp_in_neg	False	Ramp in negative Relative mode

- Cam table

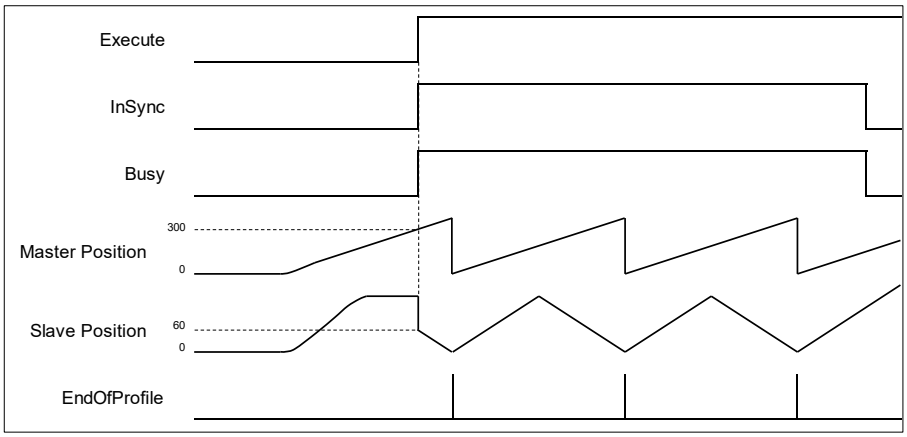


1. Absolute mode on master axis (MasterAbsolute = true)

   Absolute mode on slave axis (SlaveAbsolute = true)

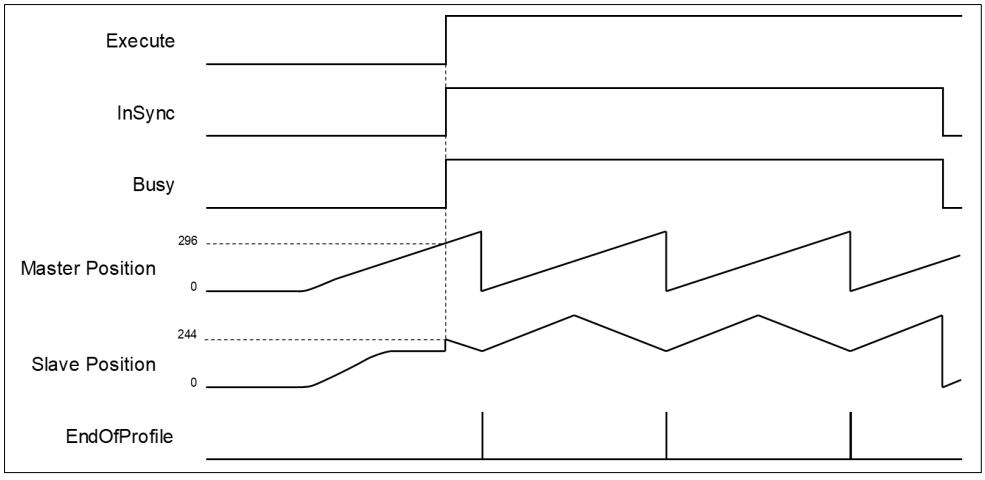
1.1 Absolute mode (StartMode = 0)

Cam master and slave axis meshing position: for master axis, is the current position. For the slave axis, follows the cam table.



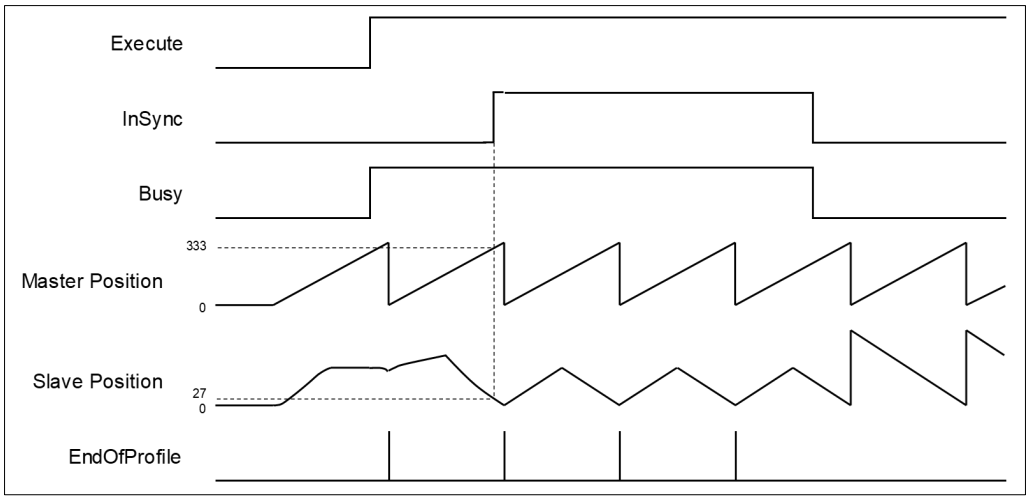
1.2 Absolute mode (StartMode = 1)

Cam master and slave axis meshing position: for master axis, is the current position. For the slave axis, follows the cam table plus slave current position (180+64 = 244). In addition, a jump will occur if the start point of master axis is not same as the start position on cam table.



1.3 Ramp in mode (StartMode=2)

Cam master and slave axes meshing position, respectively, are master current position and the slave position added with a motion curve for compensation, which is configured via VelocituDiff, Acceleration and Deceleration settings, for the purpose of preventing a jump while ramping in.



1.4 Ramp in positive, Ramp in negative (StartMode = 3, 4)

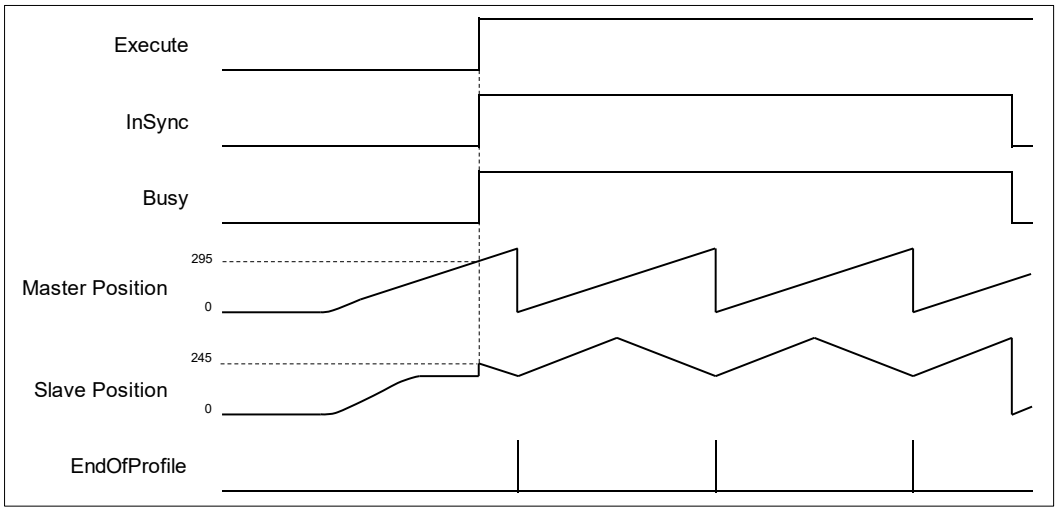
With a rotary/modulo slave axis, ramp_in_pos compensates only in the positive direction and ramp_in_neg in the negative direction. For linear slaves, the compensation direction is generated automatically with ramp_in_pos, ramp_in_neg, and ramp_in mode, which also means these three modes are under the same running condition.

2. Absolute mode on master axis (MasterAbsolute = true),

Relative mode on slave axis (SlaveAbsolute = false)

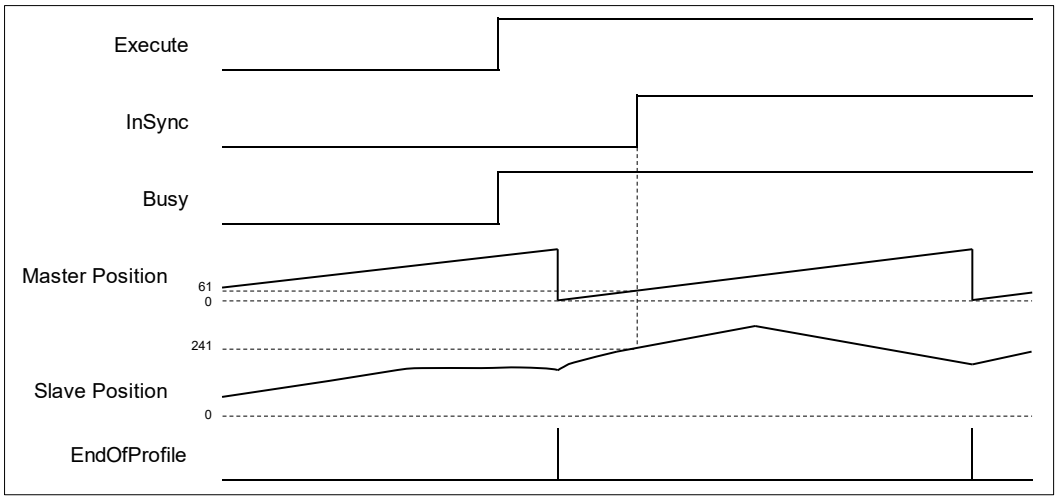
2.1 Absolute/ Relative mode (StartMode = 0, 1)

The defined positions of master and slave axis when cam is engaged, respectively, are master current position and the slave position from the cam table added on the slave current position (180+65 = 245). In addition, a jump will occur if the start point of master axis is not same as the start position on cam table.



2.2 Ramp in mode (StartMode = 2)

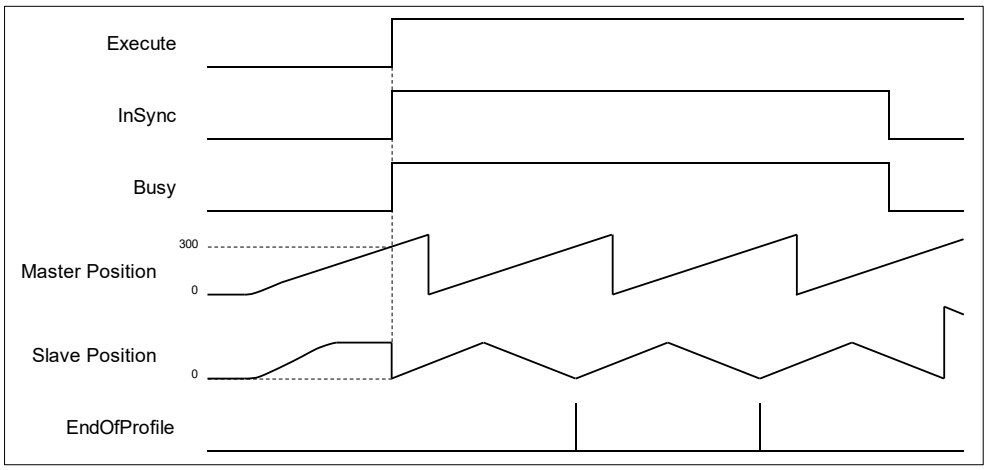
The defined positions of master and slave axis when cam is engaged, respectively, are master current position and the slave position added with a motion curve for compensation, which is configured via VelocituDiff, Acceleration and Deceleration settings, for the purpose of preventing a jump while ramping in. The slave meshing position will be the position on the cam table plus slave current position (61 + 180 = 241).



3. Master absolute mode (MasterAbsolute = false)/ Slave absolute mode (SlaveAbsolute = true)

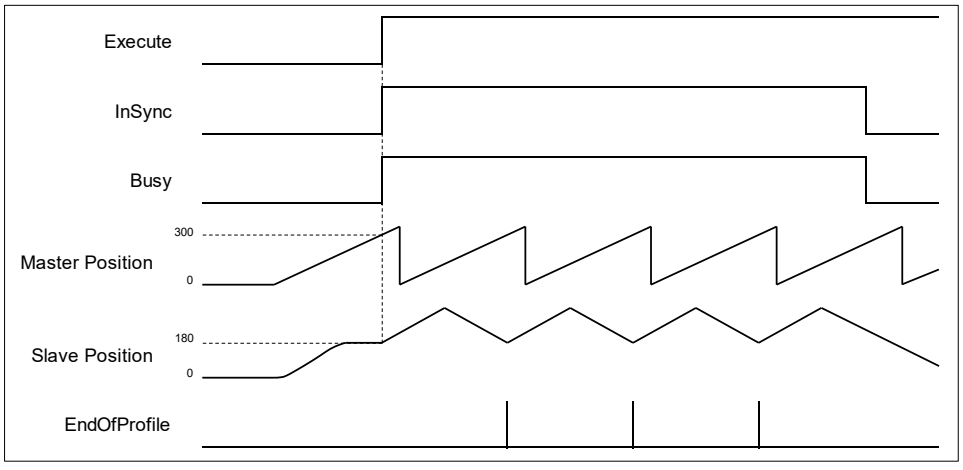
3.1 Absolute mode (StartMode = 0)

Master-slave meshing position: The current position of master axis will be the start position as well as the zero position on the cam table. The corresponding position of the slave axis on the cam table should also be zero, while the slave position (meshing) is zero under absolute mode.



3.2 Relative mode (StartMode = 1)

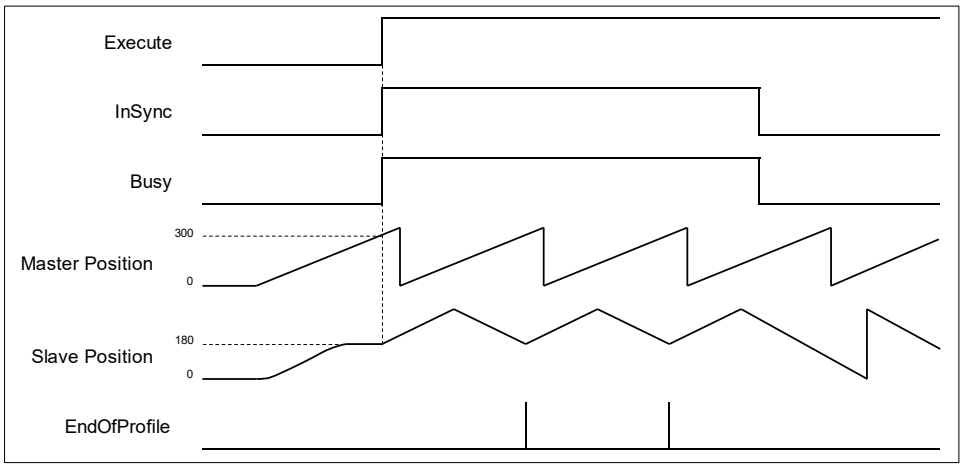
Master-slave meshing position: The current position of master axis will be the start position as well as the zero position on the cam table. The corresponding position of the slave axis on the cam table should also be zero, while the slave position (meshing) under relative mode should be the sum of corresponding position and slave current position (0+180=180).



4. Master relative mode (MasterAbsolute = false)/ Slave relative mode (SlaveAbsolute = false)

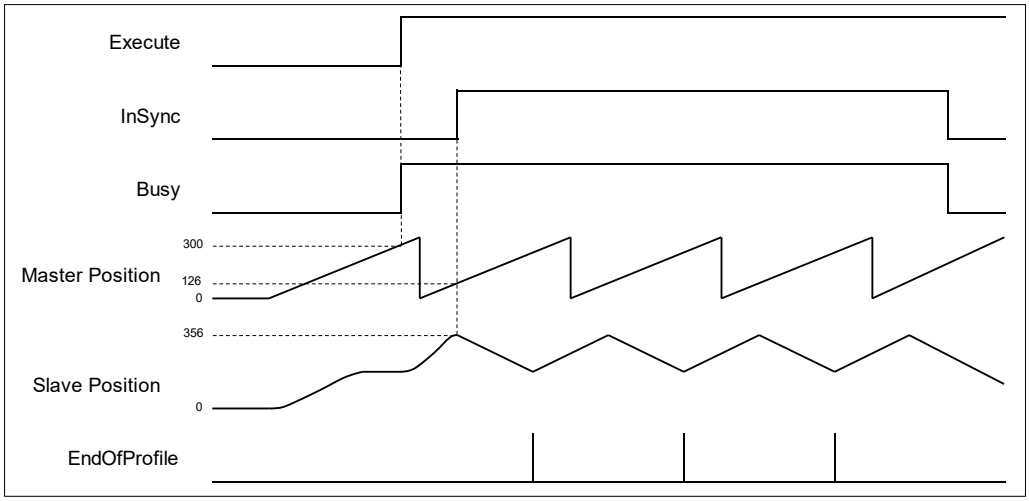
4.1 Absolute/ Relative mode (StartMode = 0, 1)

Master-slave meshing position: The current position of master axis will be the start position as well as the zero position on the cam table. The corresponding position of the slave axis on the cam table should also be zero, while the slave position (meshing) under relative mode should be the sum of corresponding position and slave current position (0+180=180).



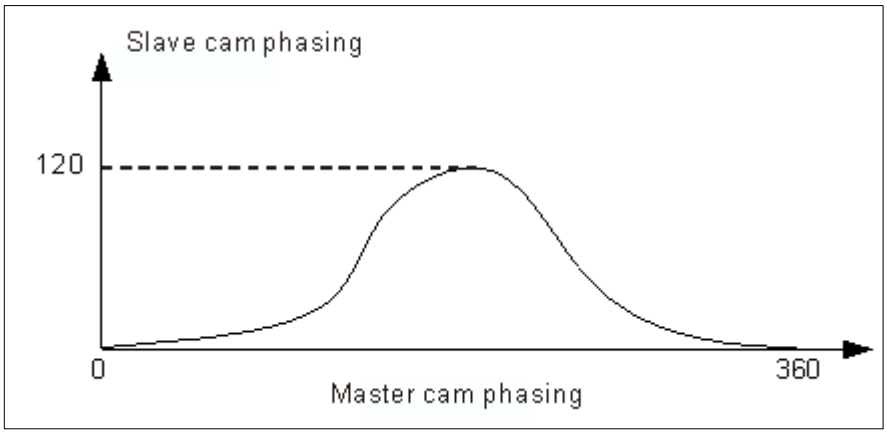
4.2 Ramp in mode (StartMode = 2)

Master-slave meshing position: The current position of master axis will be the start position as well as the zero position on the cam table. A compensating curve is added to the slave position according to the settings of VelocityDiff, Acceleration and Deceleration to avoid jumps.

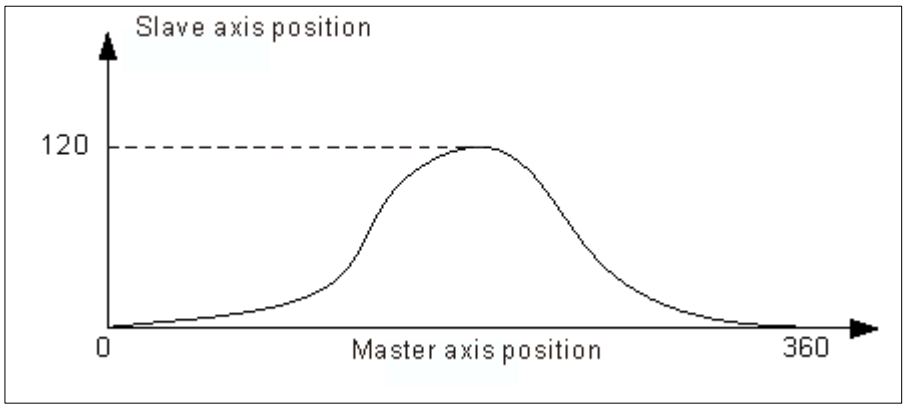


- **Compensations and scaling (MasterCompensation/MasterScaling/SlaveCompensation/Slavescaling)**

  - Position compensations and scaling can be performed by modifying the parameters based on the preplanned cam curve. For example, you can specify the scaling factor to adjust phase and compensation between master and slave in cam table with only one cam curve needed for a processed product, which comes in multiple sizes, so as to switch between different sizes of the product during production. In addition, Compensations and scaling factors of master and slave axis can be configured respectively.
  - Compensations and scaling between master and slave axis determine the actual operation for cam profile, which is demonstrated in the following example. The preplanned cam profile curve is shown below.

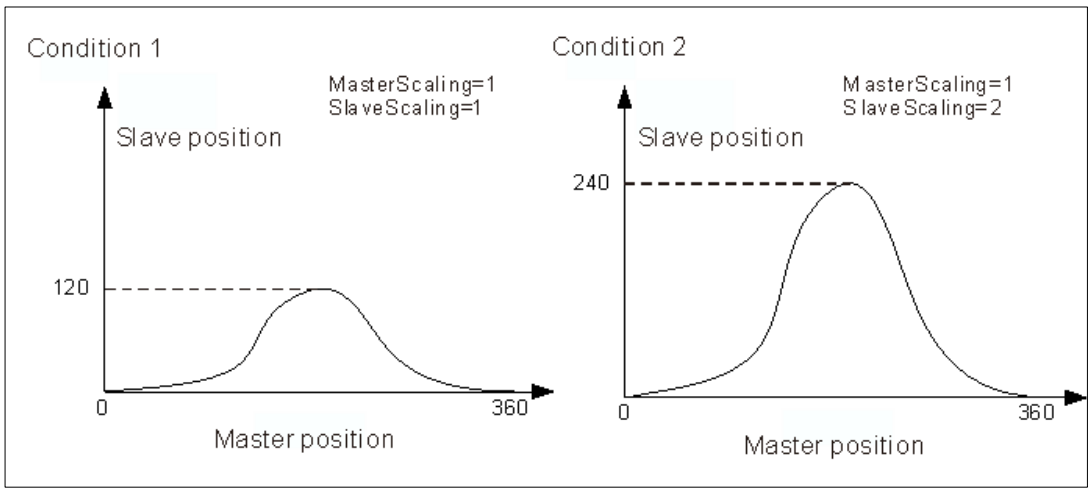


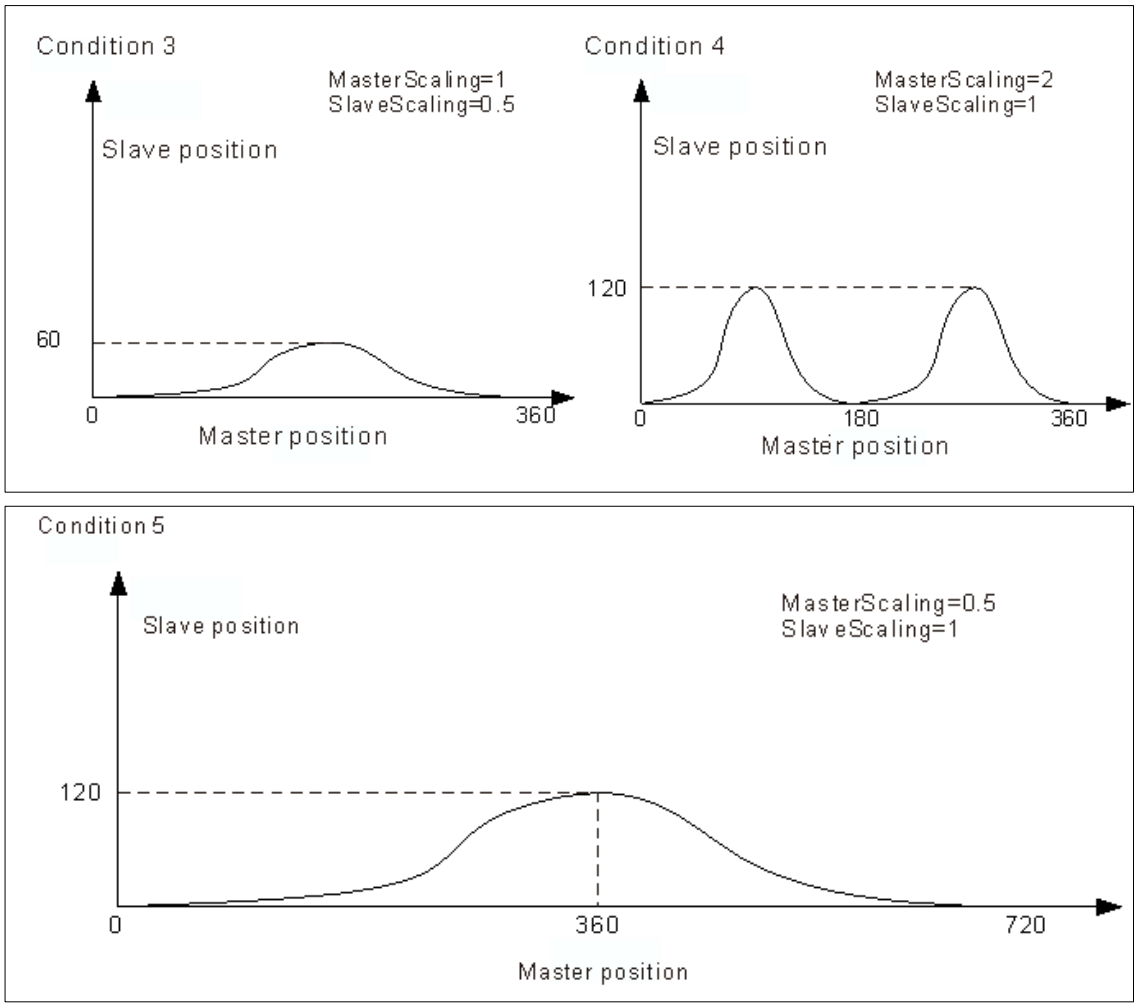
- If the master and slave axis are under absolute mode, the start position of master and slave axis will be zero while performing meshing action. Without any compensation and scaling (default setting), the relationship between the actual positions of master and slave axis are shown below.



- When the position compensations and scaling are not set as default, impacts on the relationship between the actual positions of master and slave axis are shown below.

1. When MasteCompensation= 0, the impact of MasterScaling and SlaveScaling on the actual cam profile.





Condition 1: When MasterScaling and SlaveScaling are set to 1 with no Compensations, the actual cam profile will be same as preplanned.

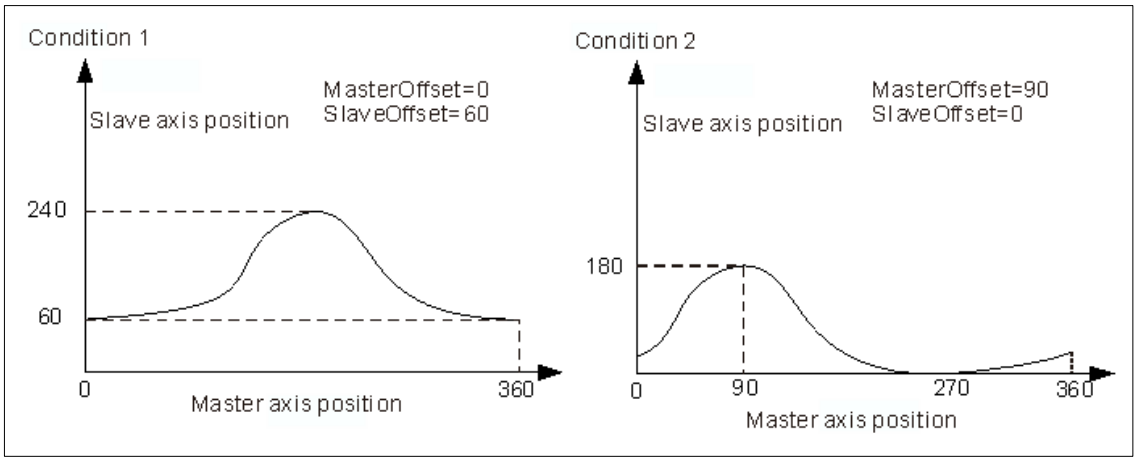
Condition 2: When MasterScaling=1 and SlaveScaling=2 with no Compensations, the slave position will be two times more than the preplanned.

Condition 3: When MasterScaling=1 and SlaveScaling=0.5 with no Compensations, the slave position will be half of the preplanned.

Condition 4: When MasterScaling=2 and SlaveScaling=1 with no Compensations, the position of master axis is doubled compared with the preplanned position relative to the slave axis. From the angle of cam phasing, the master phasing is twice the pre-planned phasing, which the cam master cycle changes from 360 to 180 and the cam slave phasing remains unchanged.

Condition 5: When MasterScaling=0.5 and SlaveScaling=1 with no Compensations, the position of master axis will be half of the preplanned position relative to the slave axis. From the angle of cam phasing, the master phasing is half the pre-planned phasing, which the cam master cycle changes from 360 to 720 and the cam slave phasing remains unchanged.

2. When MasteScaling = 0, the impact of MasterCompensation and SlaveCompensation on the actual operation of cam profile.



Condition 1: When MasterScaling=1, SlaveScaling=1, MasterCompensation=0 and SlaveCompensation=60, the slave position relative to the master position will be added with 60 based on the preplanned position. For example, the master position 180 corresponds to the slave position 180 in a planned cam relationship which the corresponding slave axis position should be 240(240=180+60) during the actual execution.

Condition 2: When MasterScaling=1, SlaveScaling=1, MasterCompensation=90 and SlaveCompensation=0, the master position relative to the slave position will be added with 90 based on the preplanned position. For example, the master position 180 corresponds to the slave position 180 in a planned cam relationship, which the master axis position 90 should correspond to the slave axis position 180(180=90+90) during the actual execution.

- **Period mode**

  - Use Periodic of MC_CamTableSelect to control the period mode. Under non-periodic mode, EndOfProfile remains to be TRUE after executing for one period. Meanwhile, the slave axis stops moving, but still in sync. The status of slave axis stays under synchronized_motion.
  - At the same time, *Execute* changes to False, while OutputsInSync, Busy and EndOfProfile of MC_CamIn remains to be TRUE.

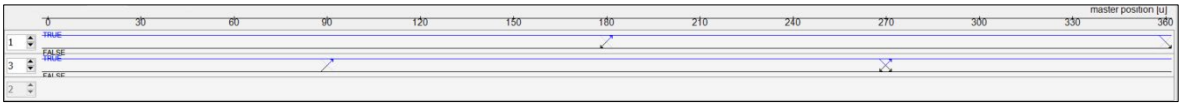
- **Tappet table***

  - Use Tappet table to set tappets in Cam and read the status of tappets with SMC_GetTappetValue, which can also be modified according to the settings in Tappet table and the direction when CAM master passing the tappets.

	Track ID	X	positive pass	negative pass
	1			
		180	switch ON	switch OFF
		360	switch OFF	none
	3			
		90	switch ON	none
		270	invert	switch OFF

***Note:** At the same position, a maximum of three tappets can be set on the Tappet table.

  - You can configure several tappets for each track ID on the Tappet table, then view the relationship between tappets and the master axis. While moving the points on Tappets page, the setting parameters on Tappet table page will be changed simultaneously.



- **TappetHysteresis**

- Set hysteresis intervals of tappet to avoid vibration in axes and encoders, which may lead to wrong switching actions. The specified axis position must exceed the interval so the next action will be run. The unit for Hysteresis is user-defined.

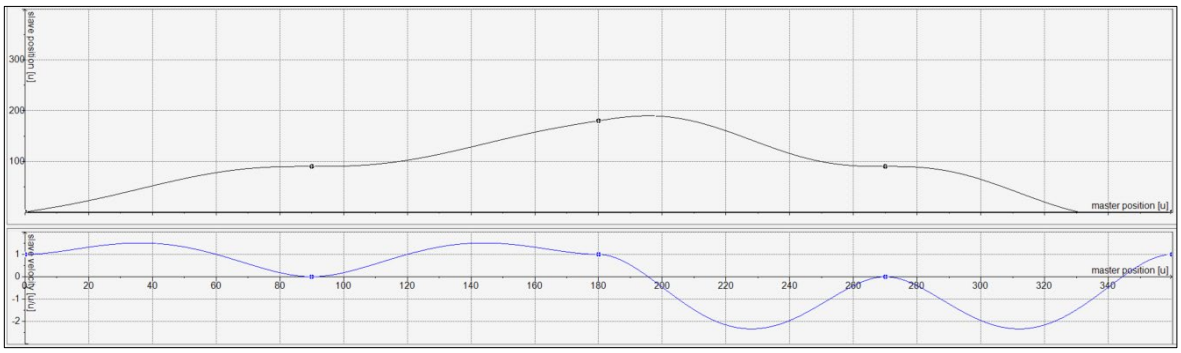
- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

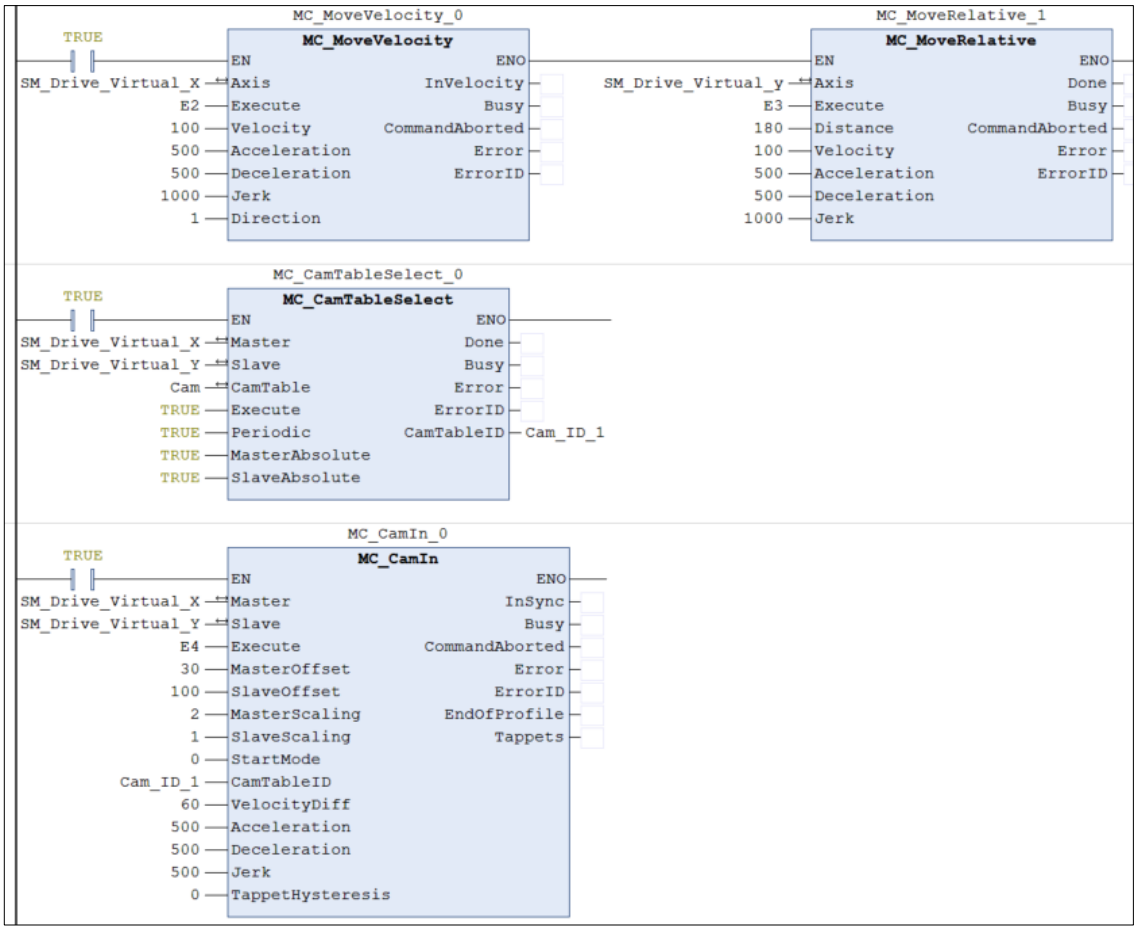
- **Example**

  - **Example1:** Demonstrate the execution result after cam parameters relating to MC_CamInbeing being configured. Both the master and slave axis are rotary axes in this example.

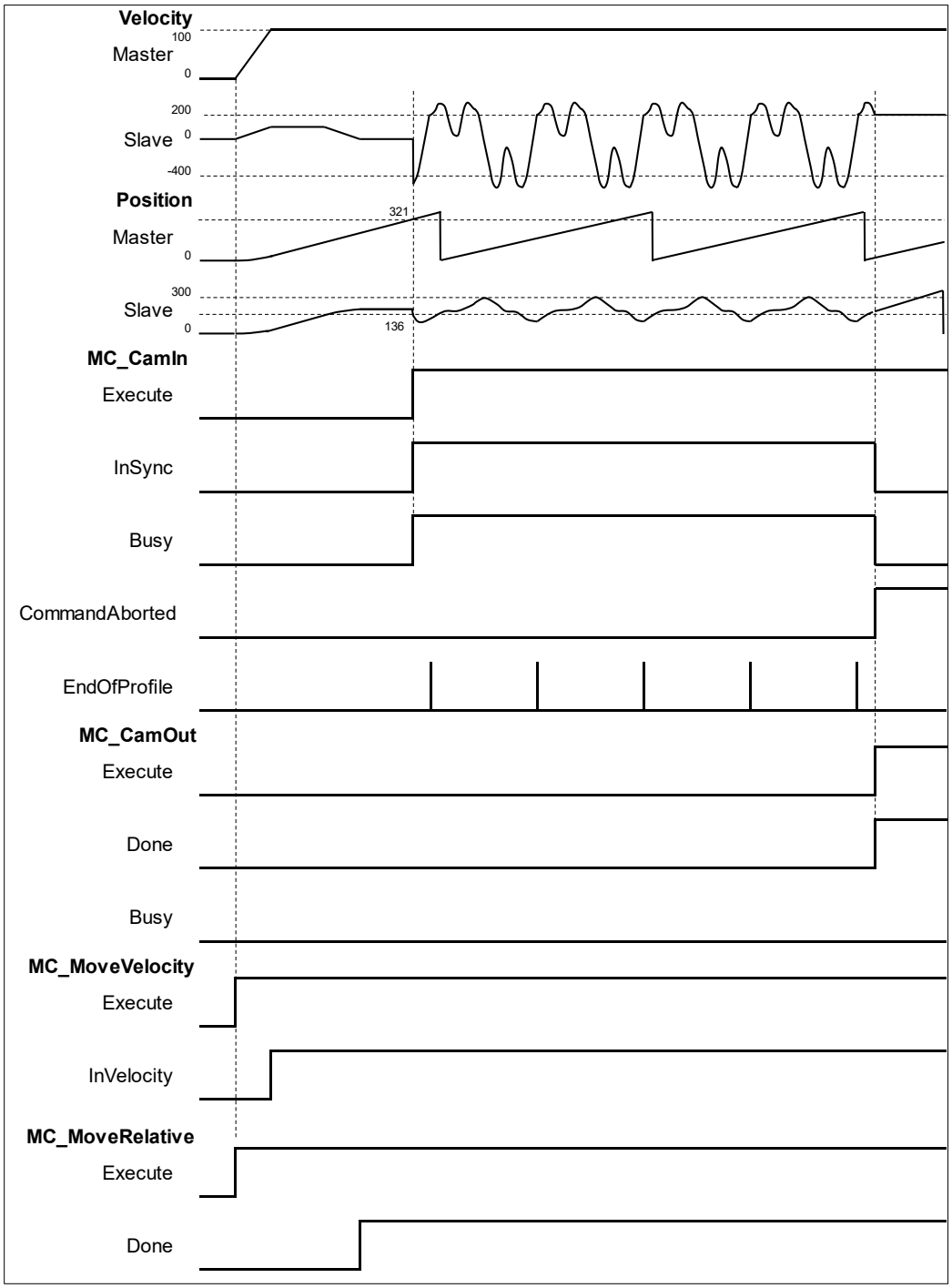
    - Cam curve planning:



	X	Y	V	A	J	Segment Type	min(Position)	max(Position)	max(|Velocity|)	max(|Acceleration|)
	0	0	1	0	0					
						Poly5	0	90	1.51200000000...	0.0437803772552189...
	90	90	0	0	0					
						Poly5	90	180	1.51200000000...	0.0437803772552188...
	180	180	1	0	0					
						Poly5	90	189.8427604...	2.33748148148...	0.10754458161865568
	270	90	0	0	0					
						Poly5	-9.84276047...	90	2.33748148148...	0.10754458161865568
	360	0	1	0	0					



- **Timing Diagram**



- The calculation for axis position and meshing position on cam coordinate:

Position_Slave = SlaveScaling×CAM (MasterScaling×MasterPosition + MasterCompensation) + SlaveCompensation

Slave meshing position = 1 × CAM (2 × 321(master position while executing CamIn) + 30) + 100

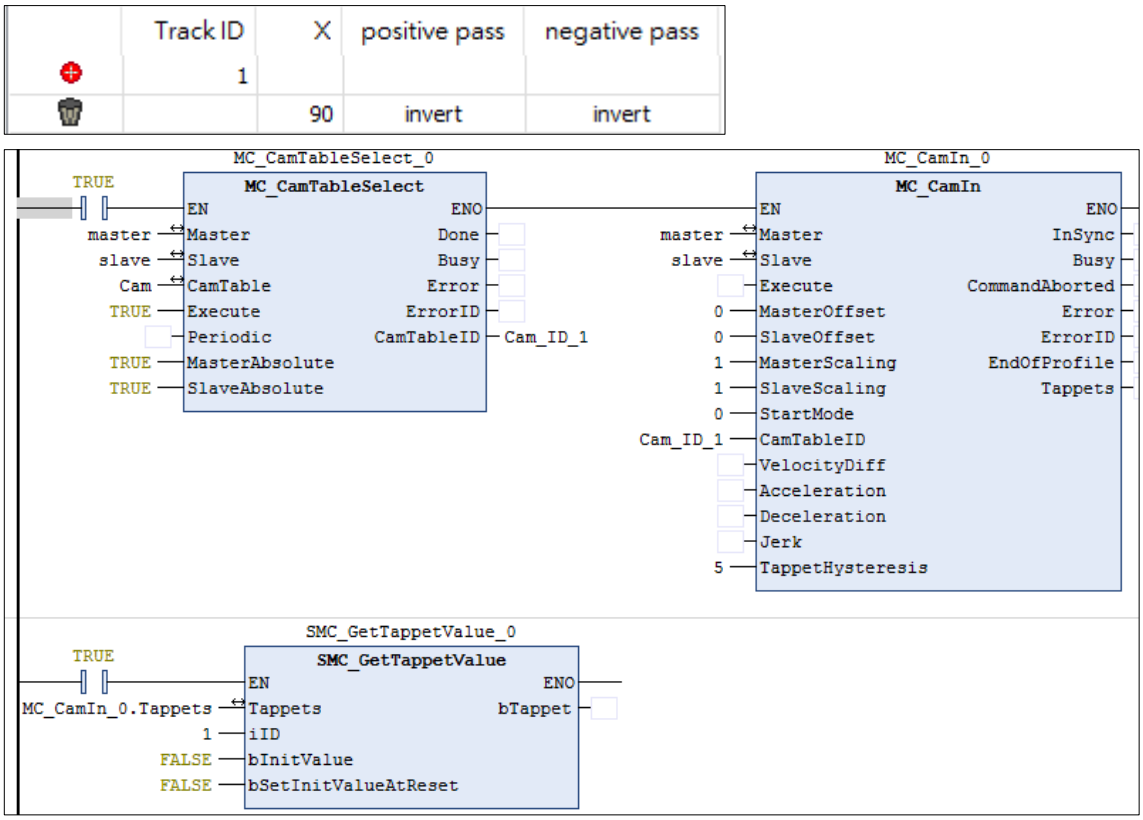
= 1 × CAM (672) + 100 = 1 × CAM (312) + 100

= 36 + 100

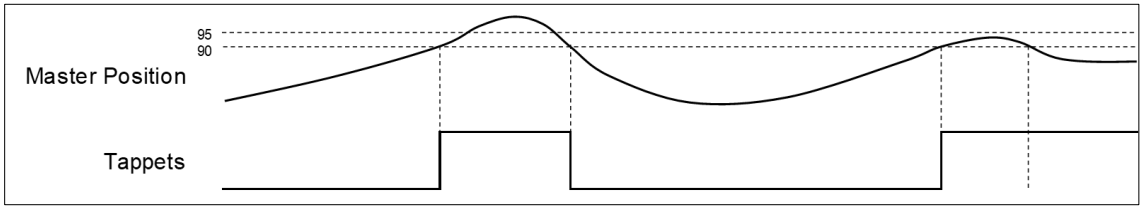
= 136

When meshing is completed, the master position will be at 321 and the slave position is at 136.

- **Example 2:** The operation of tappet after configuring *TappetHysteresis* as the following example demonstrates.
  - Tappets



- **Timing Diagram**



1. The tappets switch to ON when the master axis passes position 90. Master keeps moving forward until its position exceeds the hysteresis interval and the axis performs reversing. Then the master axis passes position 90 again and exceeds the hysteresis interval, which will make tappets switch to OFF.
2. The tappets switch to ON when the master axis passes position 90. The master axis keeps moving forward and performs reversing without exceeding the hysteresis interval. Therefore, the tappets will not switch to OFF when the master position passes 90 once again.

2.1.9 MC_CamOut

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_CamOut deactivates the meshing between master and slave axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_CamOut		MC_CamOut_instance(Slave : =, Execute : =, Done =>, Busy =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when the demeshing task between master and slave is completed.	BOOL	True/False (False)
Busy	TRUE when the instruction is run.	BOOL	True/False (False)
Error	TRUE when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

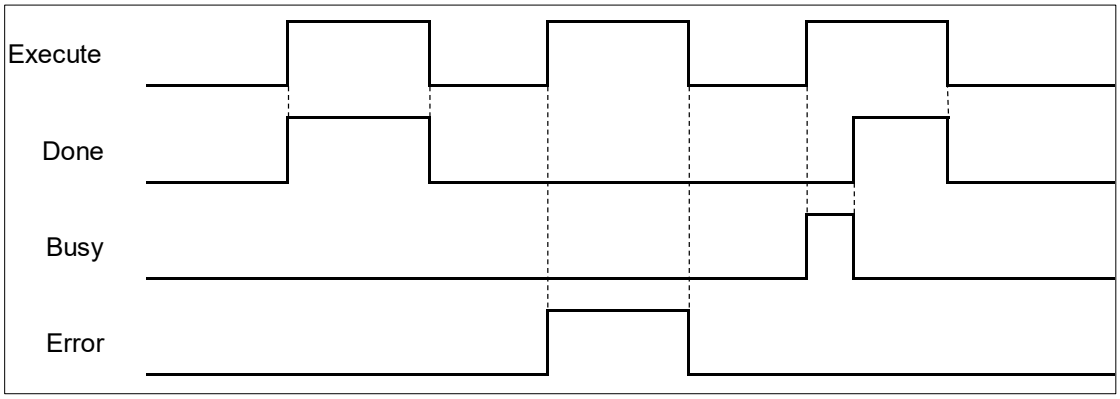
***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• When the instruction CamOut is completed.	• When *Execute* is False. • If *Execute* is False and *Done* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Busy	• When an instruction is being run.	• When *Error* and *Done* are True.

Name	Timing for Shifting to True	Timing for Shifting to False
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* is False. (Error codes are cleared.)
ErrorID		

■ **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Slave	Specifies the slave axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and Busy is False.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

■ When the slave axis is decoupled from the master axis by MC_CamOut, it moves with the sustained velocity and the slave state is under ContinuousMotion. (irrelevant to the velocity of slave axis)

■ If the synchronization between master and slave axis is not established while executing MC_CamOut. An error of SMC_AXIS_NOT_READY_FOR_MOTION(34) will be reported.

■ The axis state still remains continuous_motion, even though the slave axis is desynchronized at standstill with velocity 0.

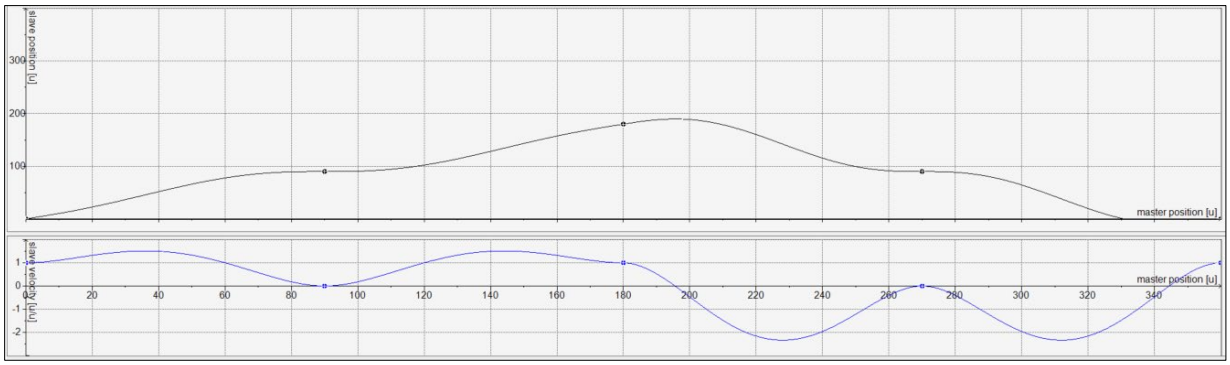
- **Troubleshooting**

■ If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to *ErrorID* (Error Code) to address the problem.

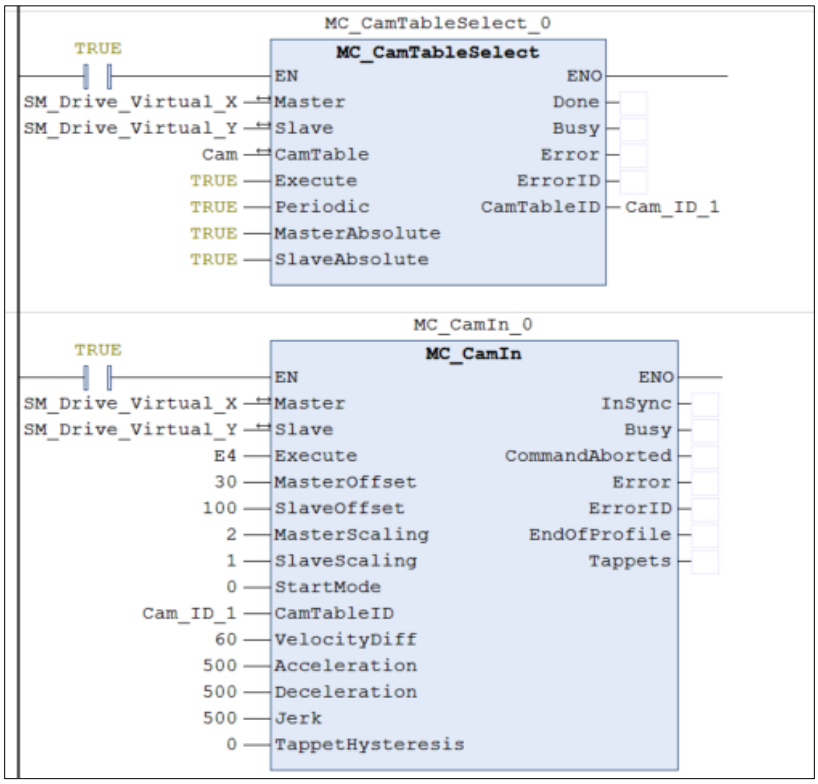
- **Example**

■ The following example gives the operation result of MC_CamOut. The master and slave axis in this example are rotary axes.

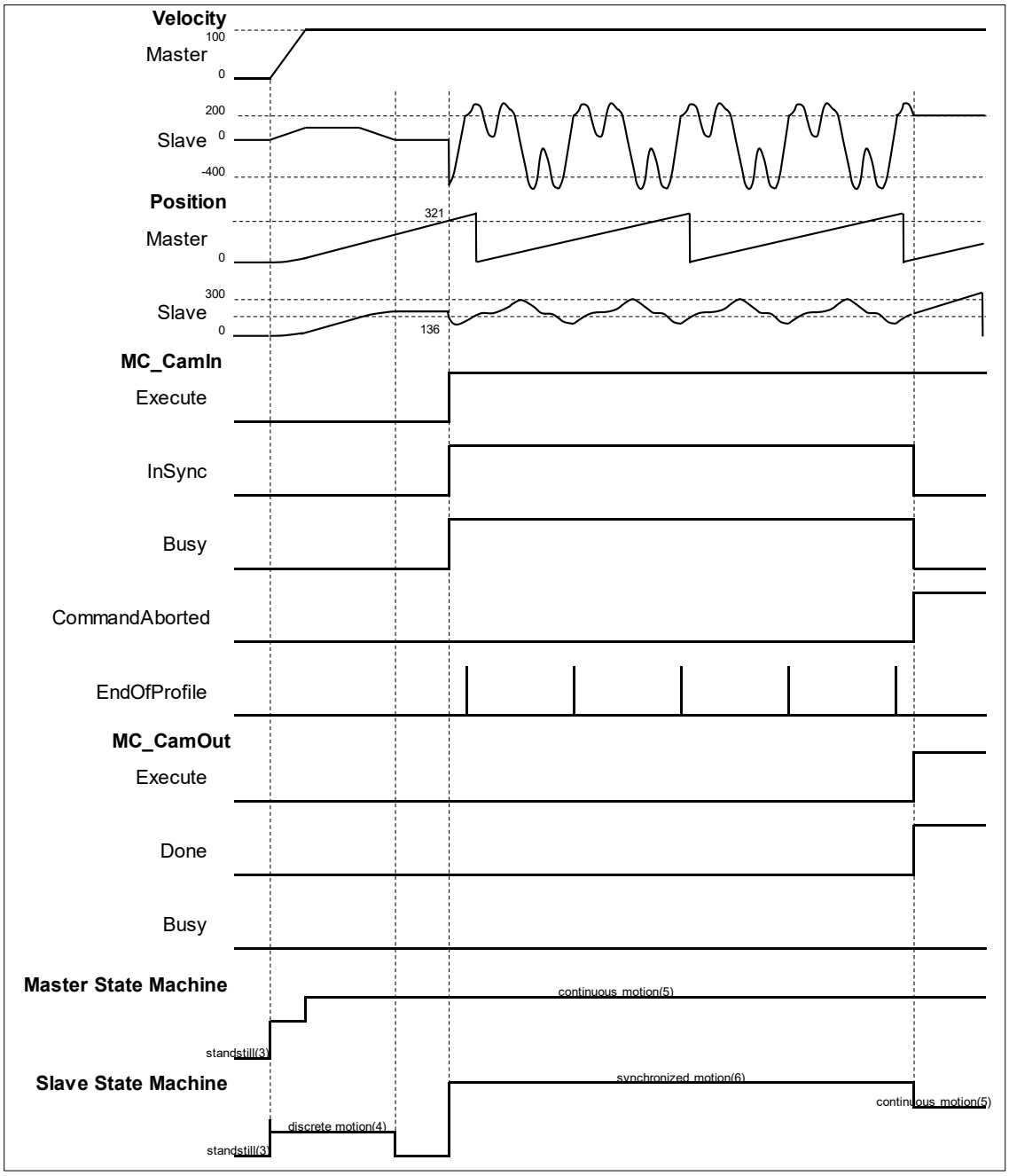
■ Cam curve planning



	X	Y	V	A	J	Segment Type	min(Position)	max(Position)	max(|Velocity|)	max(|Acceleration|)
	0	0	1	0	0					
						Poly5	0	90	1.51200000000...	0.0437803772552189...
	90	90	0	0	0					
						Poly5	90	180	1.51200000000...	0.0437803772552188...
	180	180	1	0	0					
						Poly5	90	189.8427604...	2.33748148148...	0.10754458161865568
	270	90	0	0	0					
						Poly5	-9.84276047...	90	2.33748148148...	0.10754458161865568
	360	0	1	0	0					



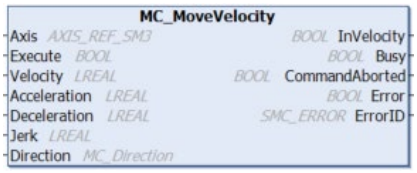
- Timing Diagram



- The master-slave meshing is deactivated when MC_CamOut is run. At the same time, MC_CamIn is aborted and a falling edge is detected at *CommandAborted.*
- The slave axis continues to move at the current speed after being decoupled from the master axis, while the axis state changes to continuous_motion.

2.1.10 MC_MoveVelocity

- **Supported Devices**: AX-308E, AX-8, AX-364E

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_MoveVelocity		MC_MoveVelocity_instance (Axis : =, Execute : =, Velocity : =, Acceleration : =, Deceleration : =, Jerk : =, Direction : =, InVelocity =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
Velocity	Target velocity. (Unit: user unit/s)	LREAL	Positive or 0(0)	When *Execute* turns to True and *Busy* is False.
Acceleration	Acceleration rate. (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Deceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Direction	Specifies the direction for servo motor rotation.	MC_ Direction*	3: fastest 2: current 1: positive 0: shortest -1: negative (current)	When *Execute* turns to True and *Busy* is False.

***Note:** MC_Direction: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
InVelocity	When the specified target	BOOL	True/False (False)

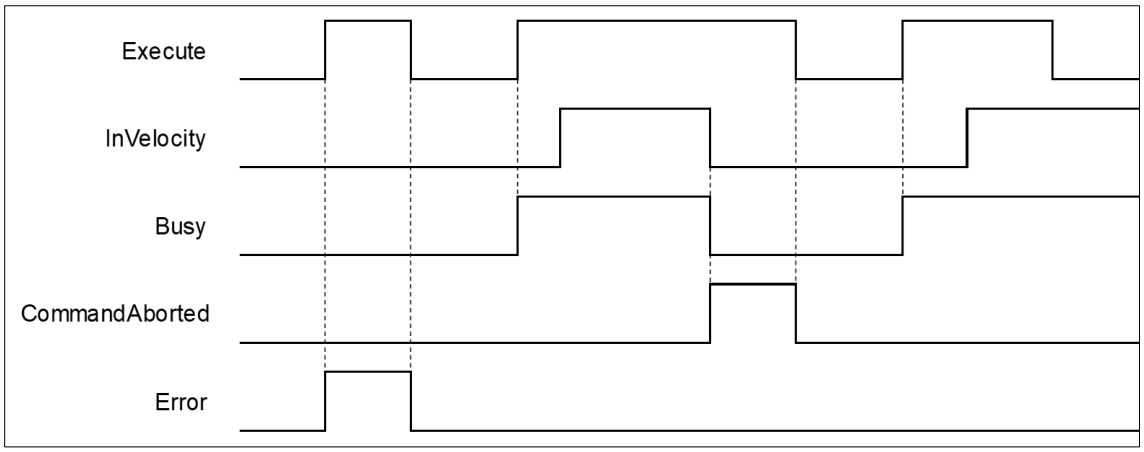
Name	Function	Data Type	Output Range (Default Value)
	velocity is reached.		
Busy	When Execute turns to True.	BOOL	True/False (False)
CommandAborted	True when this instruction is aborted	BOOL	True/False (False)
Error	True when an error occurs	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

■ **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
InVelocity	• True when the specified target velocity is reached.	• When CommandAborted turns to True • When CommandAborted turns to True and the target velocity is changed.
Busy	• When Execute turns to True.	• When Error turns to True. • When CommandAborted turns to True.
CommandAborted	• When this instruction is aborted by another instruction. • When this instruction is aborted because of the execution of MC_Stop instruction.	• When Execute changes to False. • If Execute is False and CommandAborted turns to True, it will be True for only one period and immediately shift to False.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When Execute turns from True to False. (Error Code is cleared)
ErrorID		

■ **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect

			(Default Value)	
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When Execute turns to True and Busy is False.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

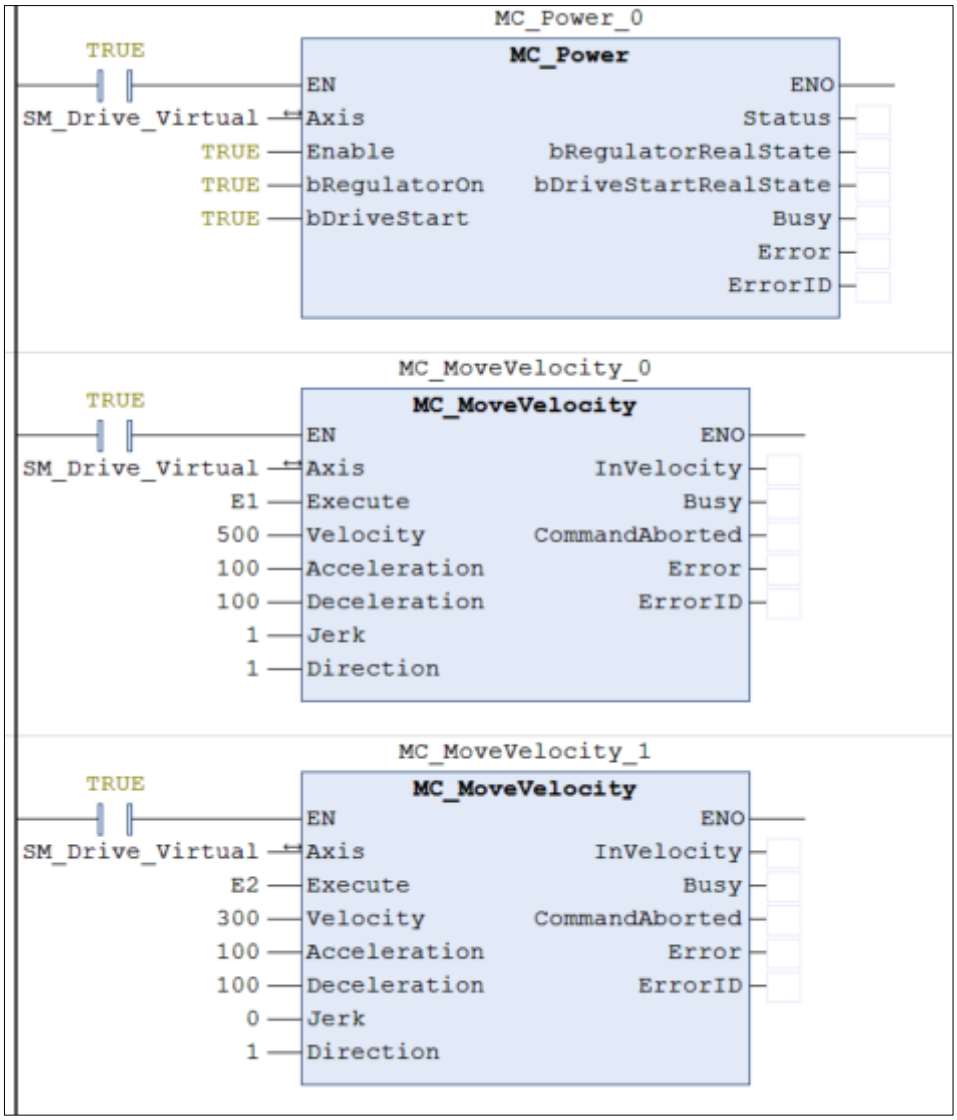
  - The instruction performs speed control with specified target velocity (Velocity), acceleration rate (Acceleration), deceleration rate (Deceleration) and Jerk value (Jerk) when execute changes to True.
  - Users can execute another motion instruction to abort the ongoing motion of MC_MoveVelocity.
  - When aborted by other instructions, OutputsInVelocity will be False and Outputs*CommandAborted is True.*
  - When *Execute* of MC_MoveVelocity changes to True, the axis starts to move at the target velocity. Even if *Execute* turns False, the execution of the function block will not be affected.
  - When ExecuteInputs of MC_MoveVelocity is retriggered and a new target velocity is specified, the axis will change the velocity to the requested velocity.
  - In case the Execute pin changes to False after the function block is run, *InVelocity* of MC_MoveVelocity will turn True when the target velocity is reached. *InVelocity* will remain as True, until being aborted by other instructions.
  - *InVelocity* remains as True when MC_MoveVelocity reaches the target velocity. Even if the velocity being changed by MC_MoveSuperimposed, the movement of *InVelocity* will not be affected.

- **Troubleshooting**

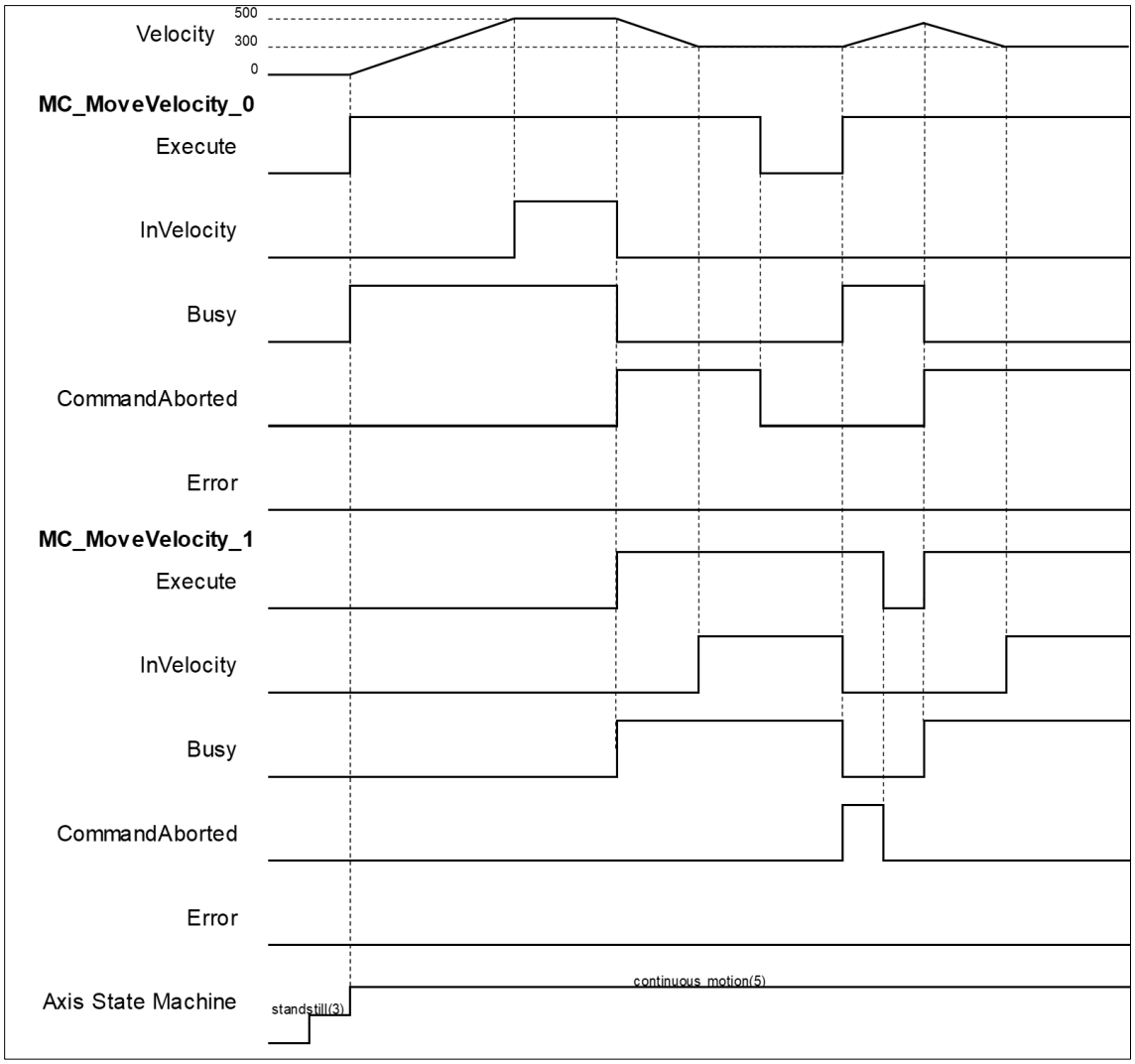
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

  - The example below describes the behavior of two MC_MoveVelocity.



- **Timing Diagram**



- When Execute changes to True, the first MC_MoveVelocity controls the axis to reach the specified target velocity 500. When it reaches 500, InVelocity changes to True.
- If Execute of MC_MoveVelocity_1 changes to True, InVelocity will change to False and CommanAborted will change to True while MC_MoveVelocity_0 is aborted.
- MC_MoveVelocity_1 will decelerate the axis to the velocity 300. When 300 is reached, InVelocity will change to True and remain in this status as long as the velocity is not changed.
- When Execute of MC_MoveVelocity_0 changes to False, CommanAborted will change to False.
- If MC_MoveVelocity_0 is restarted by Execute, which changes to True, the axis will abort MC_MoveVelocity_1 and accelerate toward the velocity 500.
- Before the axis reaches the target velocity of MC_MoveVelocity_0, Execute of MC_MoveVelocity_1 will again turn False to True and aborts MC_MoveVelocity_0. In this case, the axis decelerates again without reaching the target velocity.

2.1.11 MC_PositionProfile

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_PositionProfile is used to set time and position to plan motion profiles.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_PositionProfile		MC_PositionProfile_instance (Axis: =, TimePosition: =, Execute : =, ArraySize: =, PositionScale: =, Compensation: =, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
ArraySize	Number of motion profile arrays	INT	Positive or 0(0)	When *Execute* turns to True and *Busy* is False.
PositionScale	Overall scale factor in value.	LREAL	Negative, positive or 0(1)	When *Execute* turns to True and *Busy* is False.
Compensation	Overall profile Compensation in value (Unit: user unit/s)	LREAL	Negative, positive or 0(0)	When *Execute* turns to True and *Busy* is False.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	True when execution of path planning is finished.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
Error	True when an error occurs	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

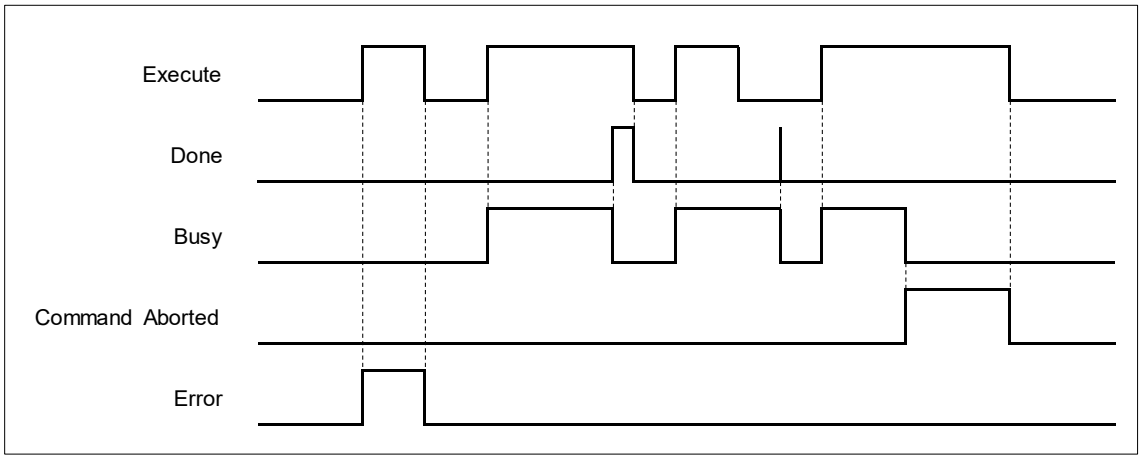
Name	Function	Data Type	Output Range (Default Value)
	Refer to Appendix for error code descriptions.		

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	When the execution of path planning is completed.	• When Execute turns from True to False. • If Execute is False and Done turns to True, Done will be True for only one scan cycle and immediately shift to False.
Busy	When Execute changes to True.	• When Done changes to True. • When Error changes to True. • When Commandaborted turns to True.
CommandAborted	When this instruction is aborted by another function block.	• When Execute changes to False. • If Execute is False and CommandAborted turns to True, it will be True for only one period and immediately shift to False.
Error	When an error occurs in the execution conditions or input values for the instruction.	When Execute turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	When Execute turns to True and Busy is False.
TimePosition	Time and position during execution.	MC_TP_REF*2	MC_TP_REF	When Execute turns to True and Busy is False.

***Note:**

1. AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

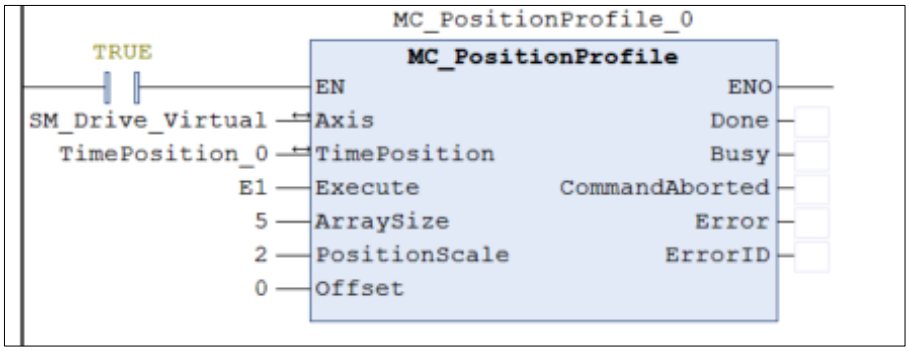
2. MC_TP_REF: Structure(STRUCT).

Name	Function	Data Type	Setting Value (Default Value)
Number_of_pairs	There's no need to set this variable, which has been replaced by InputsArraySize.	INT	-
IsAbsolute	Set the mode of position.	BOOL	True: Absolute mode False: Relative mode (True)
MC_TP_Array	Time and position data during execution of instruction.	ARRAY [1..100] OF SMC_TP	SMC_TP*

***Note:** SMC_TP: Structure(STRUCT).

Name	Function	Data Type	Setting Value (Default Value)
delta_time	Period of time between position points	TIME	Positive or 0(TIME#0ms)
position	Position of the position point	LREAL	Negative, positive or 0(0)

- **Function**
  - MC_PositionProfile carries out motion profile with time and position according to the user-defined data in TimePosition variables, the state is Discrete Motion during the movement.
  - MC_MoveSuperimposed will not be able to function while MC_PositionProfile is being used.
- **Troubleshooting**
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.
- **Example**
  - The example below describes the behavior of MC_PositionProfile instruction.

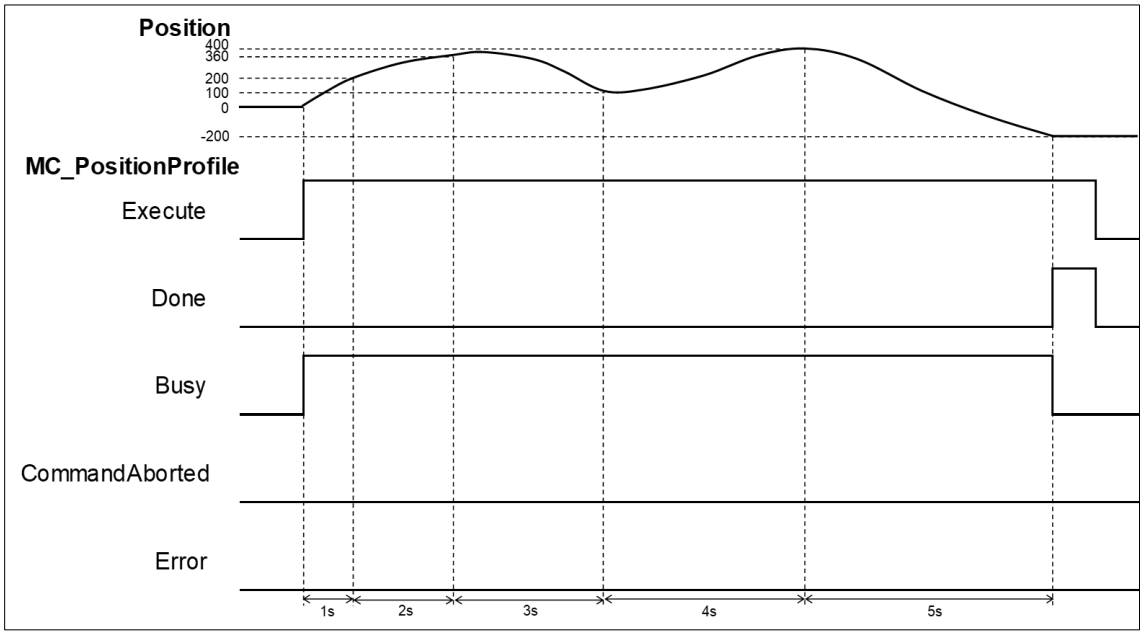


Configure for the curve of motion profile:

```
TimePosition_0.MC_TP_Array[1].delta_time := T#1S;
TimePosition_0.MC_TP_Array[2].delta_time := T#2S;
TimePosition_0.MC_TP_Array[3].delta_time := T#3S;
TimePosition_0.MC_TP_Array[4].delta_time := T#4S;
TimePosition_0.MC_TP_Array[5].delta_time := T#5S;


TimePosition_0.MC_TP_Array[1].position :=100;
TimePosition_0.MC_TP_Array[2].position :=180;
TimePosition_0.MC_TP_Array[3].position :=50;
TimePosition_0.MC_TP_Array[4].position :=200;
TimePosition_0.MC_TP_Array[5].position :=-100;
```

- **Timing Diagram**

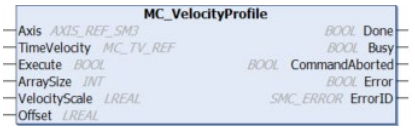


- When *Execute* of MC_PositionProfile is raised, the target axis moves along the curve, which is generated by the settings of delta_time and position in TimePosition.
- The Setting Value of *IsAbsolute* is True, MC_PositionProfile plans motion curves in the mode of Abosolue position.
- Motion curve of MC_PositionProfile is generated according to the time-position data in TimePosition. As a result of PositionScale=2, the position will be 200 after one second of execution, position 300 after two seconds and so on. When runs after 5 second, the position should be -200.

2.1.12MC_VelocityProfile

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_VelocityProfile is used to set time and velocity to plan motion profiles.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_VelocityProfile		MC_VelocityProfile_instance(Axis: =, TimeVelocity: =, Execute : =, ArraySize: =, VelocityScale: =, Compensation: =, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
ArraySize	Number of motion profile arrays	INT	Positive or 0(0)	When *Execute* turns to True and *Busy* is False.
VelocityScale	Overall scale factor in value.	LREAL	Negative, positive or 0(1)	When Execute turns to True and *Busy* is False.
Compensation	Overall profile Compensation in value (Unit: user unit/s)	LREAL	Negative, positive or 0(0)	When Execute turns to True and *Busy* is False.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	True when execution of path planning is finished.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

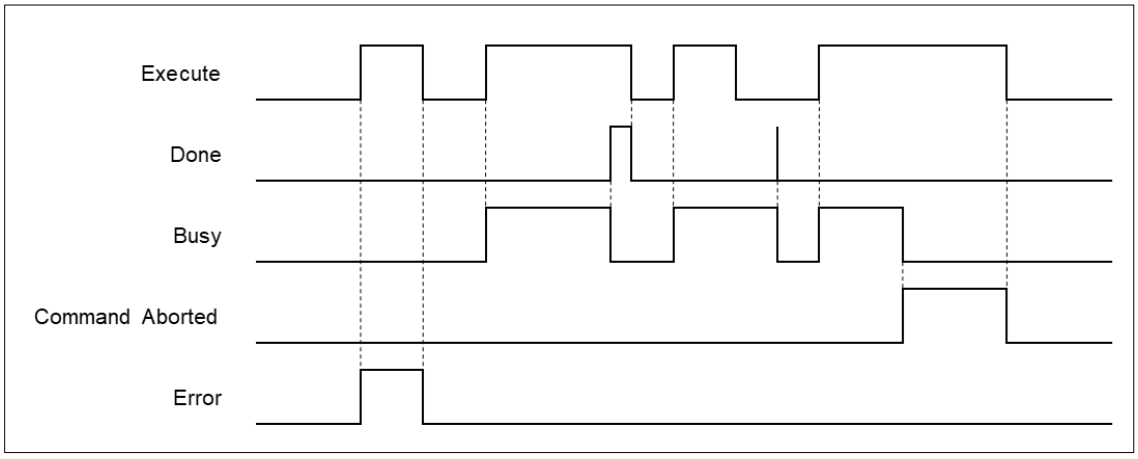
Name	Function	Data Type	Output Range (Default Value)
	error occurs. Refer to Appendix for error code descriptions.		

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	When the execution of path planning is completed.	• When Execute turns from True to False. • If Execute is False and Done turns to True, Done will be True for only one scan cycle and immediately shift to False.
Busy	When Execute changes to True.	• When Done changes to True. • When Error changes to True. • When Commandaborted turns to True.
CommandAborted	When this instruction is aborted by another function block.	• When Execute changes to False. • If Execute is False and CommandAborted turns to True, it will be True for only one period and immediately shift to False.
Error	When an error occurs in the execution conditions or input values for the instruction.	• When Execute turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	When Execute turns to True and Busy is False.
TimeVelocity	Time and velocity during execution.	MC_TV_REF*2	MC_TV_REF	When Execute turns to True and Busy is False.

***Note:**

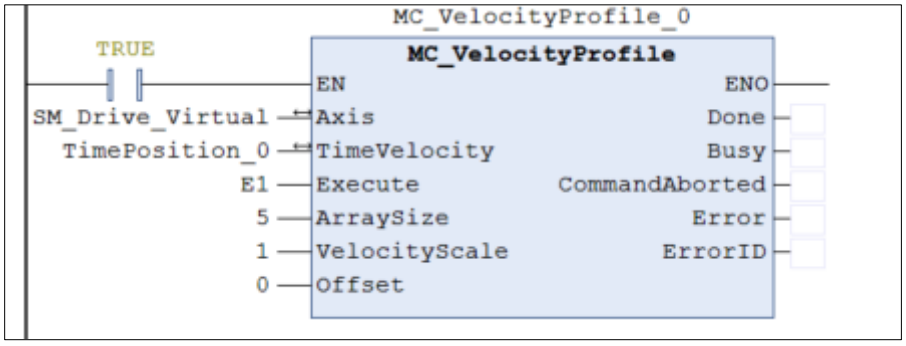
1. AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.
2. MC_TV_REF: Structure(STRUCT).

Name	Function	Data Type	Setting Value (Default Value)
Number_of_pairs	There's no need to set this variable, which has been replaced by InputsArraySize.	INT	-
IsAbsolute	Set the mode of velocity.	BOOL	True: Absolute mode False: Relative mode (True)
MC_TV_Array	Time and velocity data during execution of instruction.	ARRAY [1..100] OF SMC_TV	SMC_TV*

***Note:** SMC_TV: Structure(STRUCT).

Name	Function	Data Type	Setting Value (Default Value)
delta_time	Period of time between position points	TIME	Positive or 0(TIME#0ms)
velocity	Velocity of the position point	LREAL	Negative, positive or 0(0)

- **Function**
  - MC_VelocityProfile carries out motion profile with time and velocity according to the user-defined data in TimeVelocity variables, the state is Continuous Motion during the movement.
  - MC_VelocityProfile will not be able to function while MC_PositionProfile is being used.
- **Troubleshooting**
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.
- **Example**
  - The example below describes the behavior of MC_VelocityProfile.

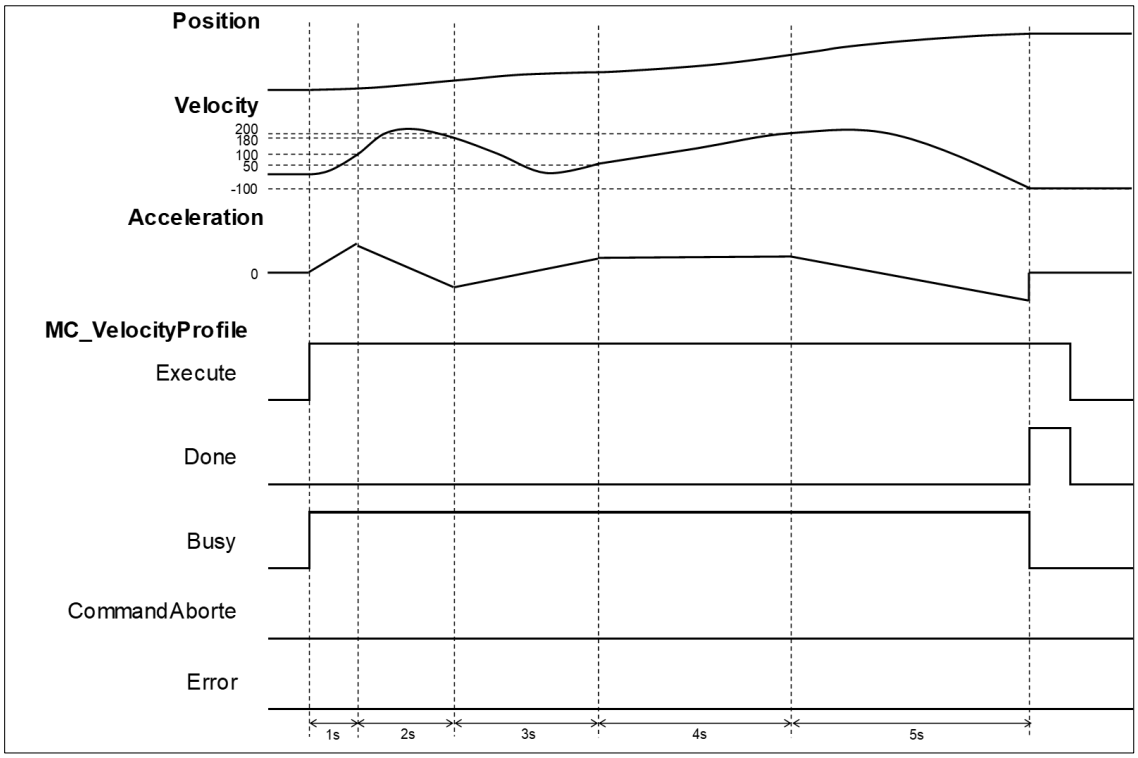


Configure for the curve of motion profile:

```
TimeVelocity_0.MC_TV_Array[1].delta_time := T#1S;
TimeVelocity_0.MC_TV_Array[2].delta_time := T#2S;
TimeVelocity_0.MC_TV_Array[3].delta_time := T#3S;
TimeVelocity_0.MC_TV_Array[4].delta_time := T#4S;
TimeVelocity_0.MC_TV_Array[5].delta_time := T#5S;


TimeVelocity_0.MC_TV_Array[1].velocity :=100;
TimeVelocity_0.MC_TV_Array[2].velocity :=180;
TimeVelocity_0.MC_TV_Array[3].velocity :=50;
TimeVelocity_0.MC_TV_Array[4].velocity :=200;
TimeVelocity_0.MC_TV_Array[5].velocity :=-100;
```

- **Timing Diagram**



- When Execute of MC_VelocityProfile is raised, the target axis moves along the curve, which is generated by the settings of delta_time and velocity in TimeVelocity.
- The Setting Value of IsAbsolute is True, MC_VelocityProfile plans motion curves in Abosolue mode.
- Since Motion curve of MC_VelocityProfile is generated according to the time-velocity data in TimeVelocity, the velocity will be 100 after one second of execution, position 180 after two seconds and so on. When runs after 5 second, the position should be -100.

2.1.13MC_AccelerationProfile

- **Supported Devices**: AX-308E, AX-8, AX-364E

Similar to MC_PositionProfile, MC_AccelerationProfile is used to set time and acceleration to plan motion profiles. However, its position points are defined by acceleration variables in MC_TV_REF.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_AccelerationProfile		MC_AccelerationProfile _instance (Axis: =, TimeAcceleration: =, Execute : =, ArraySize: =, AccelerationScale: =, Compensation: =, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
ArraySize	Number of motion profile arrays	INT	Negative, positive or 0(0)	When *Execute* turns to True and *Busy* is False.
AccelerationScale	Overall scale factor in value.	LREAL	Negative, positive or 0(1)	When *Execute* turns to True and *Busy* is False.
Compensation	Overall profile Compensation in value (Unit: user unit/s)	LREAL	Negative, positive or 0(0)	When *Execute* turns to True and *Busy* is False.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	True when execution of path planning is finished.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)

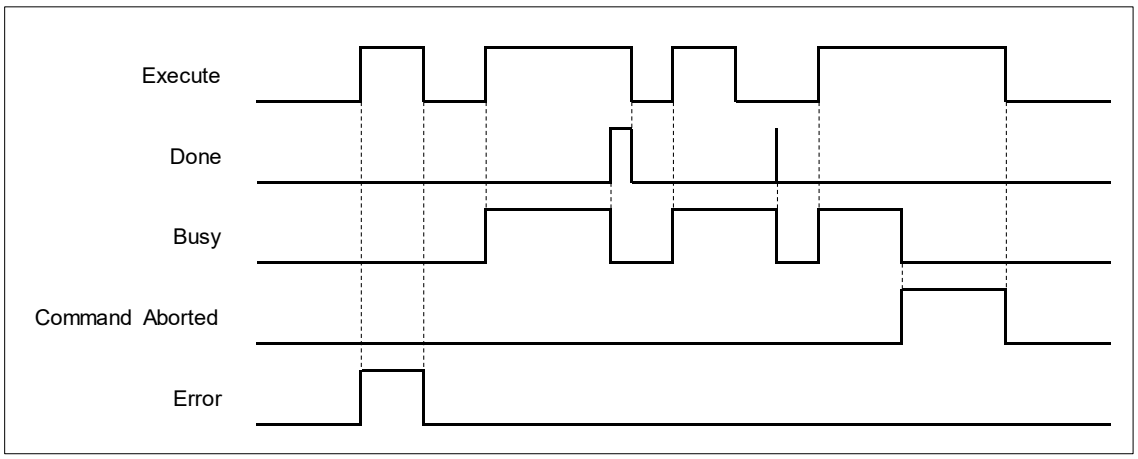
Name	Function	Data Type	Output Range (Default Value)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	When the execution of path planning is completed.	• When Execute turns from True to False. • If Execute is False and Done turns to True, Done will be True for only one scan cycle and immediately shift to False.
Busy	When Execute changes to True.	• When Done changes to True. • When Error changes to True. • When Commandaborted turns to True.
CommandAborted	When this instruction is aborted by another function block.	• When Execute changes to False. • If Execute is False and CommandAborted turns to True, it will be True for only one period and immediately shift to False.
Error	When an error occurs in the execution conditions or input values for the instruction.	When Execute turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the	AXIS_REF_SM3*1	AXIS_REF_SM3	When Execute turns to True and

	axis.			Busy is False.
TimeAcceleration	Time and acceleration during execution.	MC_TA_REF*2	MC_T_REF	When Execute turns to True and Busy is False.

***Note:**

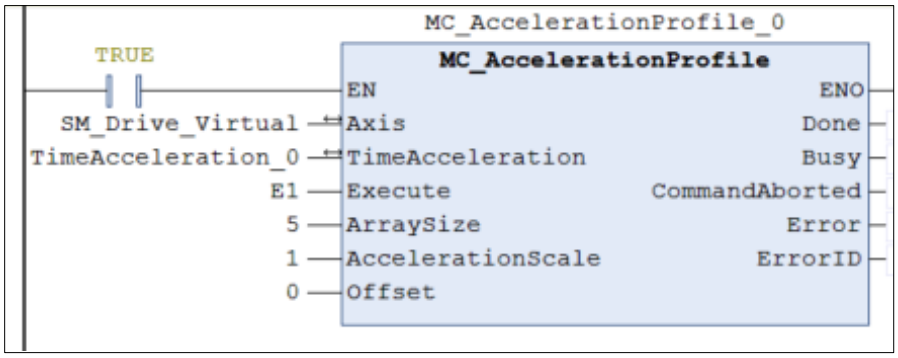
1. AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.
2. MC_TA_REF: Structure(STRUCT).

Name	Function	Data Type	Setting Value (Default Value)
Number_of_pairs	There's no need to set this variable, which has been replaced by InputsArraySize.	INT	-
IsAbsolute	Set the mode of acceleration.	BOOL	True: Absolute mode False: Relative mode (True)
MC_TA_Array	Time and acceleration data during execution of instruction.	ARRAY [1..100] OF SMC_TA	SMC_TA*

***Note:** SMC_TA: Structure(STRUCT).

Name	Function	Data Type	Setting Value (Default Value)
delta_time	Period of time between position points	TIME	Positive or 0(TIME#0ms)
acceleration	Acceleration of the position point	LREAL	Negative, positive or 0(0)

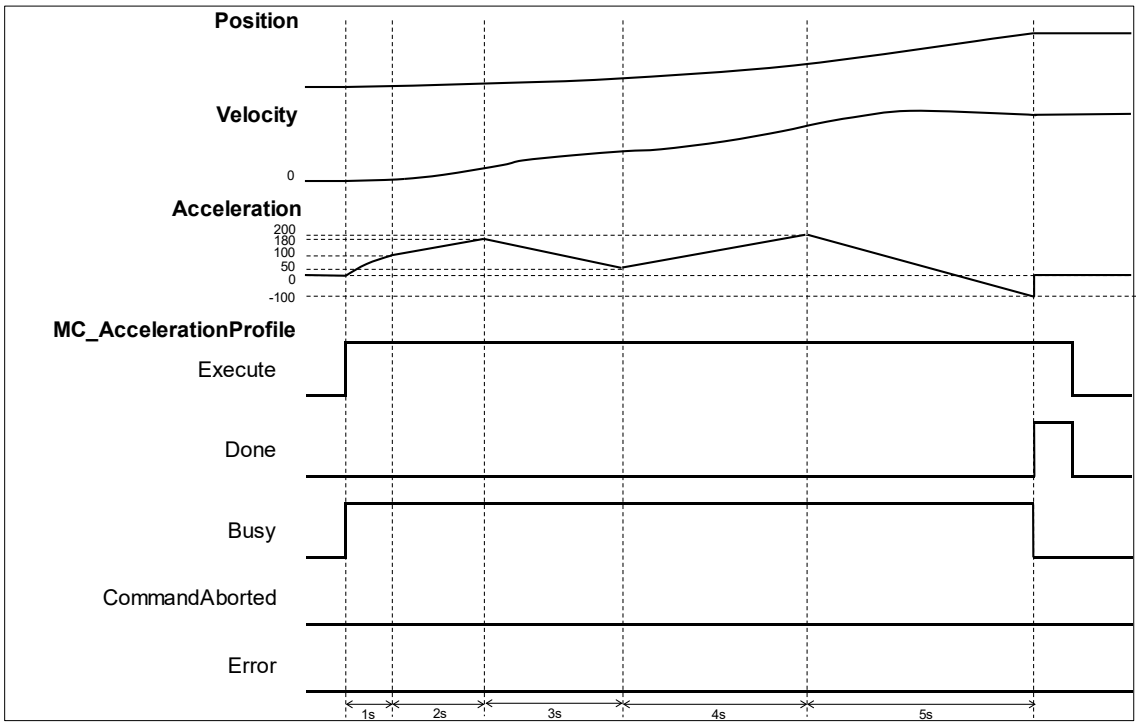
- **Function**
  - MC_AccelerationProfile carries out motion profile with time and acceleration according to the user-defined data in TimeAcceleration variables, the state is Continuous Motion during the movement.
  - MC_MoveSuperimposed cannot add a specific distance to an existing motion of MC_AccelerationProfile.
- **Troubleshooting**
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.
- **Example**
  - The example below describes the behavior of MC_AccelerationProfile.




```
TimeAcceleration_0.MC_TA_Array[1].delta_time := T#1S;
TimeAcceleration_0.MC_TA_Array[2].delta_time := T#2S;
TimeAcceleration_0.MC_TA_Array[3].delta_time := T#3S;
TimeAcceleration_0.MC_TA_Array[4].delta_time := T#4S;
TimeAcceleration_0.MC_TA_Array[5].delta_time := T#5S;


TimeAcceleration_0.MC_TA_Array[1].acceleration :=100;
TimeAcceleration_0.MC_TA_Array[2].acceleration:=180;
TimeAcceleration_0.MC_TA_Array[3].acceleration:=50;
TimeAcceleration_0.MC_TA_Array[4].acceleration:=200;
TimeAcceleration_0.MC_TA_Array[5].acceleration:=-100;
```

- **Timing Diagram**

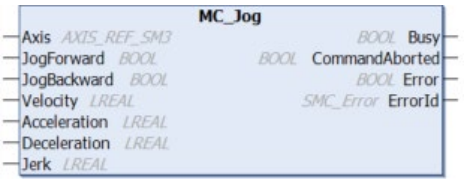


- When Execute of MC_AccelerationProfile is raised, the target axis moves along the curve, which is generated by the settings of delta_time and Acceleration in TimeAcceleration.
- The Setting Value of IsAbsolute is True, MC_AccelerationProfile plans motion curves in Abosolue mode.
- Since Motion curve of MC_AccelerationProfile is generated according to the time-acceleration data in TimeAcceleration, the velocity will be 100 after one second of execution, position 180 after two seconds and so on. When runs after 5 seconds, the position should be -100.

2.1.14MC_Jog

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_Jog enables an axis to be moved forward or backward.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Jog		MC_Jog_instance (Axis : =, JogForward: =, JogBackward: =, Velocity : =, Acceleration : =, Deceleration : =, Jerk : =, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
JogForward	JogForward changes from FALSE to TRUE.	BOOL	True/False (False)	-
JogBackward	JogBackward changes from FALSE to TRUE	BOOL	True/False (False)	-
Velocity	The target velocity (Unit: user unit/s)	LREAL	Positive or 0(0)	When *Execute* turns to True and *Busy* is False.
Acceleration	Acceleration rate. (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Deceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s3)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)

Name	Function	Data Type	Output Range (Default Value)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

* **Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Busy	True when JogForward or JogBackward turns to True.	• When Error turns to True. • When CommandAborted turns to True. • When status is standstill and there's no complementarity between the status of JogForward and JogBackward.
CommandAborted	True when the instruction is interrupted.	When JogForward and JogBackward shift to False.
Error	When an error occurs in the execution conditions or input values for the instruction.	When JogForward and JogBackward shift to False.(Error code is cleared)
ErrorID		

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When JogForward or JogBackward shift to True.

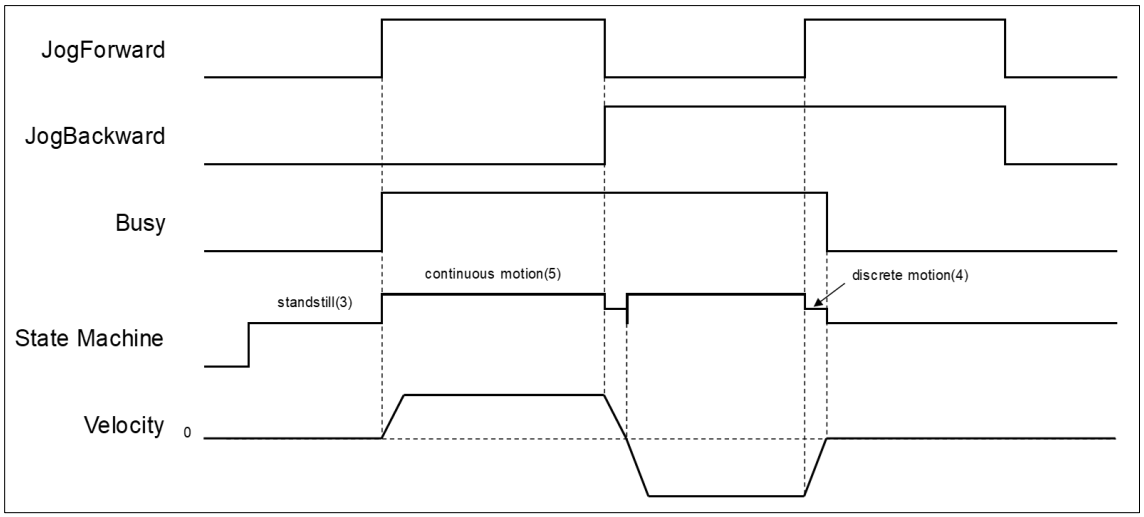
* **Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

  - MC_Jog starts a continuous movement with the specified Velocity, while *JogForward* and *JogBackward* determine on the direction of axis movement (negative or positive).

JogForward	JogBackward	Movement
False	False	Motionless
True	False	Jog in positive direction
False	True	Jog in negative direction
True	True	Motionless

  - When changing the direction, MC_Jog will reread the Input parameter settings to perform Jog operation.



- The following description is with reference to the above figure:

  - When *JogFoward* is True and *JogBackward* is False, the axis starts to move in positive direction, which the status is continuous_motion.
  - When *JogFoward* is False and *JogBackward* is True, the axis starts to move in negative direction, which the status is discrete_motion.
  - When decelerating to reach 0 velocity, the status will shift to Standstill, then the axis starts to accelerate in negative direction and change the status to continuous_motion; at the same time, Busy is still True.
  - When both *JogForward* and *JogBackward* are True, the axis will accelerate/ decelerate to reach 0 velocity, the status will be discrete_motion.
  - When both JogForward and JogBackward are True, the axis will stop moving and the status will be Standstill with Busy output changing from Busy to False.

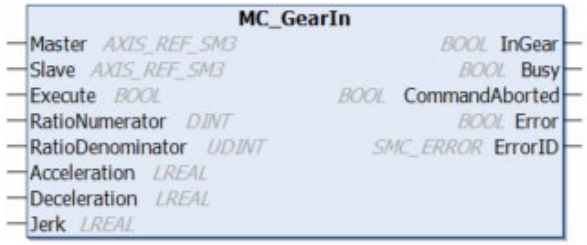
- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

2.1.15MC_GearIn

- **Supported Devices**: AX-308E, AX-8, AX-364E

The function block MC_GearIn activates a linear master-slave meshing.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_GearIn		MC_GearIn_instance (Master : =, Slave : =, Execute : =, RatioNumerator : =, RatioDenominator : =, Acceleration : =, Deceleration : =, Jerk : =, InGear =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
RatioNumerator	Gear ratio numerator* between master and slave axis.	DINT	Negative, positive or 0(1)	When *Execute* turns to True and *Busy* is False.
RatioDenominator	Gear ratio denominator* between master and slave axis.	UDINT	Positive(1)	When *Execute* turns to True and *Busy* is False.
Acceleration	Acceleration rate. (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Deceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.

***Note:** A negative gear ratio will make the master and slave axis move in an opposite direction.

- **Outputs**

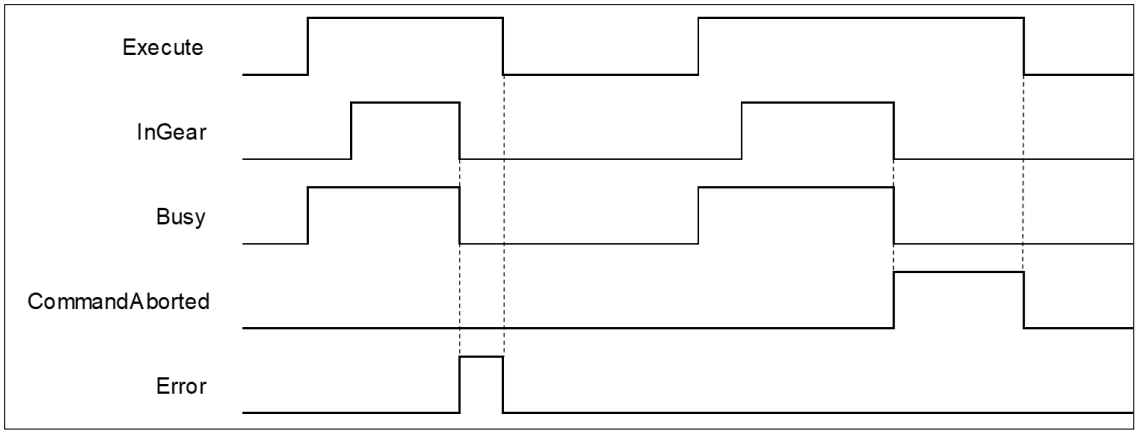
Name	Function	Data Type	Output Range (Default Value)
InGear	True if the meshing was successful.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
InGear	• When the slave axis reaches the target velocity and the meshing is successful.	• When *Error* turns to True. • When *CommandAborted* turns to True. • When the gear ratio changes.
Busy	• When the meshing is processed.	• When *Error* turns to True. • When *CommandAborted* turns to True.
CommandAborted	• When MC_GearOut is run. • True when the instruction is aborted by another function block. • When this instruction is aborted because of the execution of MC_Stop instruction.	• When *Execute* changes to False. • If *Execute* is False and *CommandAborted* turns to True, *CommandAborted* will be True for only one scan cycle and immediately shift to False.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



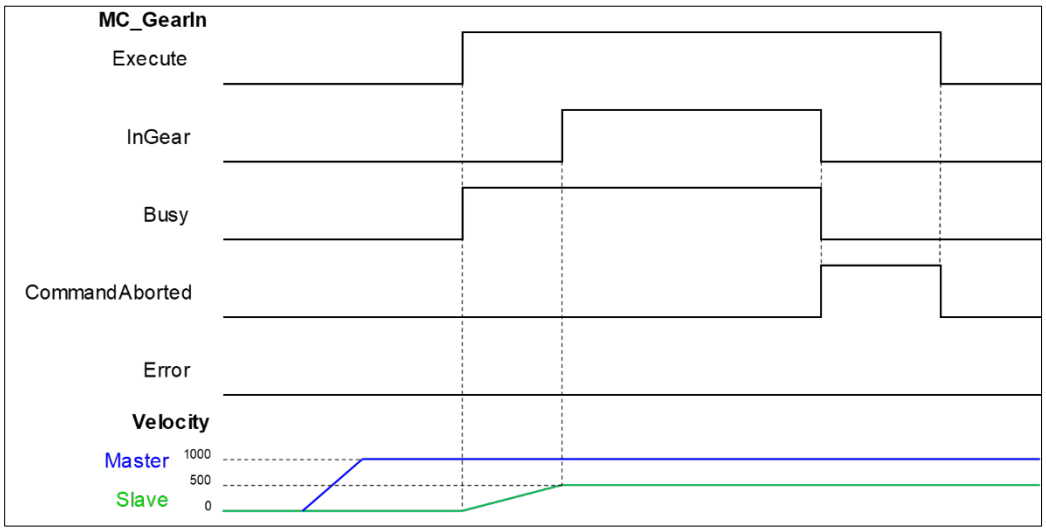
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Master	Specifies the master axis number.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and Busy is False.
Slave	Specifies the slave axis number.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and Busy is False.

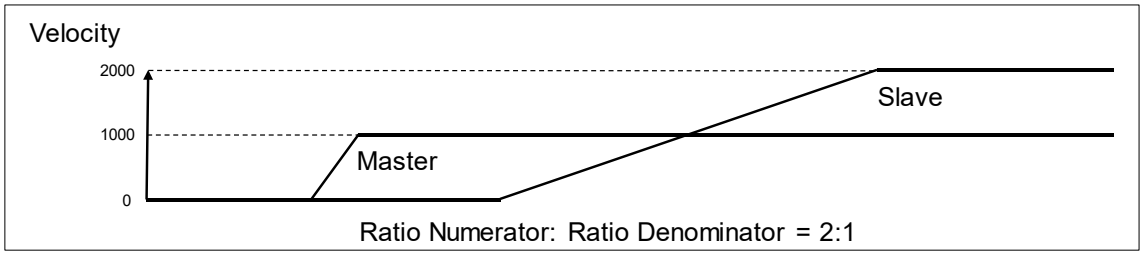
***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

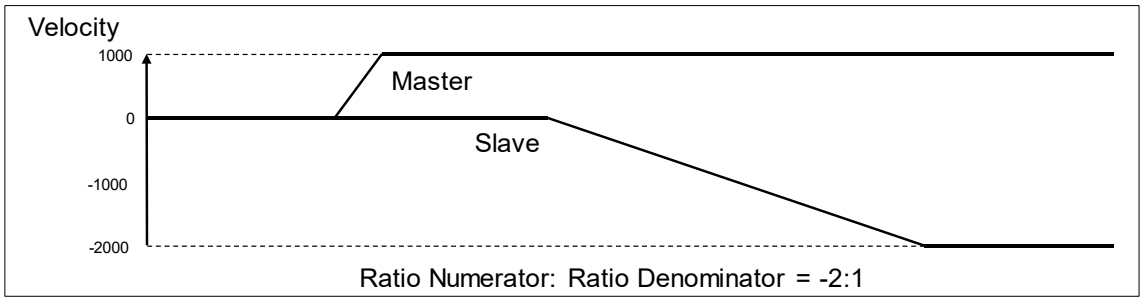
  - In MC_GearIn, slave axis will follow master axis to move at the Set Position.
  - When Execute is True, The target speed of slave axis is the velocity of master axis times gear ratio (Velocity * RatioNumerator / RatioDenominator)



  - After the gear relation is established, slave axis will follow master axis to move at the given proportional relationship to accomplish the synchronized control of master and slave axis. Master and slave axis could be real or virtual axis or the external encoder master axis.
    - RatioNumerator, RatioDenominator
      - When the value of gear ratio is positive, the master and slave axis move in same direction.



      - When the value of gear ratio is negative, the master and slave axis move in opposite direction.



- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

2.1.16MC_GearOut

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_GearOut disconnects the gear relation (velocity) between master and slave axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_GearOut		MC_GearOut_instance (Slave : =, Execute : =, Done =>, Busy =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	True when the gear disconnection is completed.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

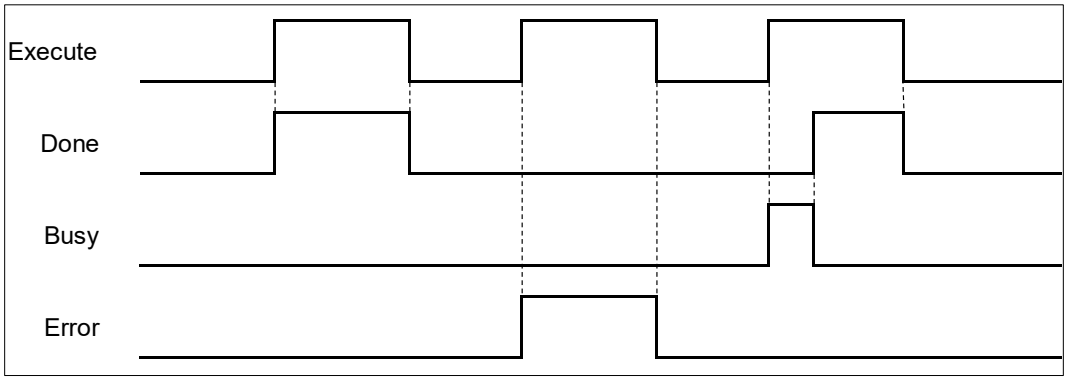
***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• When the gear disconnection is completed.	• When *Execute* changes to False. • If *Execute* is False and *Done* turns to True, *Done* will be True for only one scan cycle and immediately shift to False. • When *Error* turns to True.

Name	Timing for Shifting to True	Timing for Shifting to False
Busy	• When *Execute* changes to True.	• When *Done* turns to True. • When *Error* turns to True.
Error	• When an error occurs in the execution conditions or input values for the instruction. (Error Code is recorded)	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**

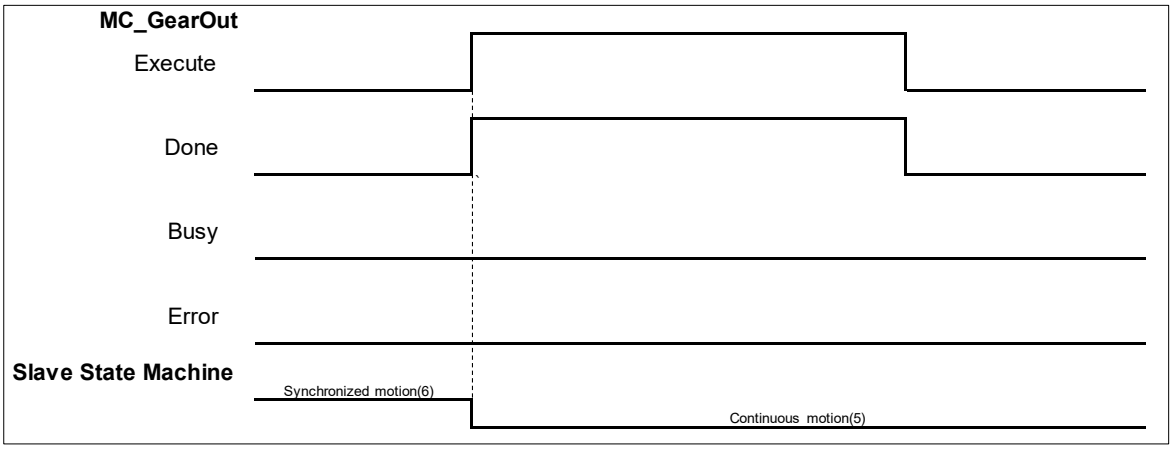


- **Inputs/Outputs**

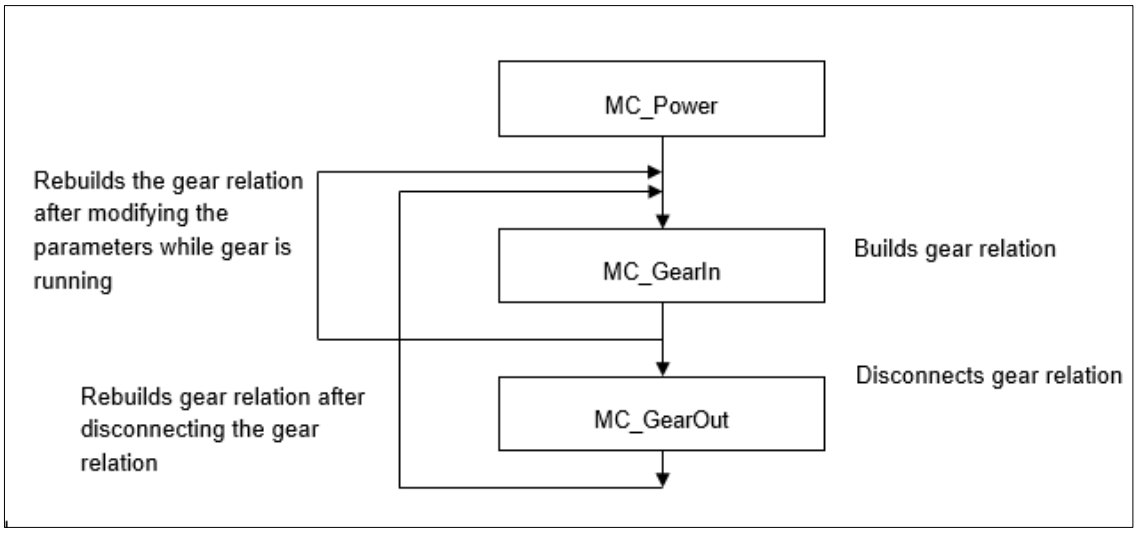
Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Slave	Specify the slave axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and *Busy* is False.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**
  - After the gear relationship is disconnected, the slave axis will keep moving at the speed where the gear is disconnected. The axis will be in ContinuousMotion (it has nothing to do with the axis velocity).



  - When the slave axis is out of sync and the velocity is zero, the status will be continuous_motion and remain unchanged.
  - The sequence for execution of the instructions related to electronic gear.

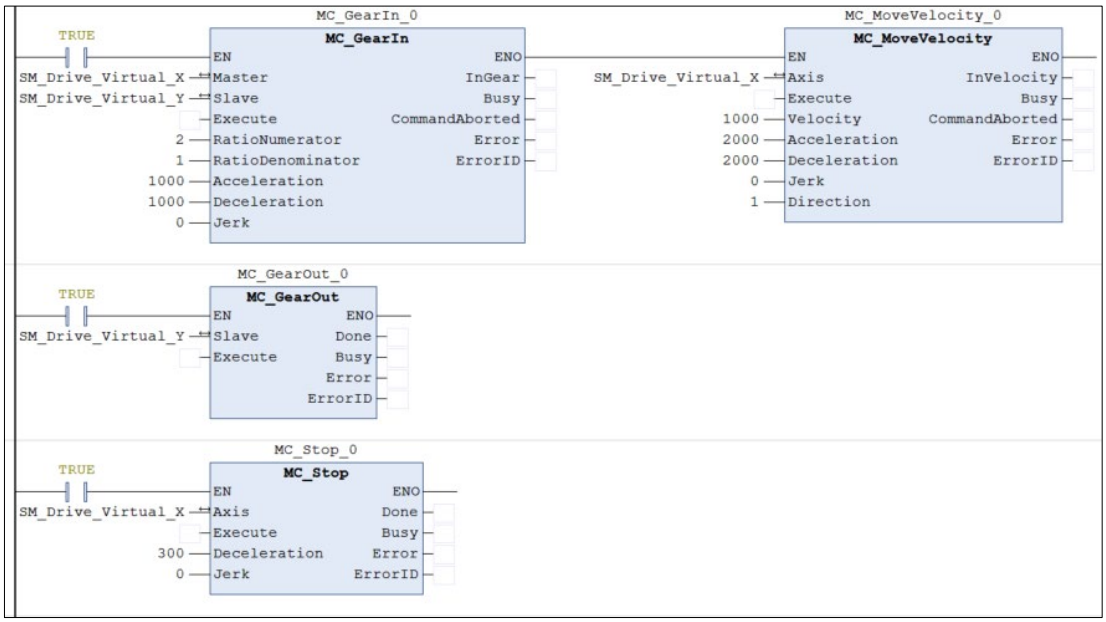


- **Troubleshooting**

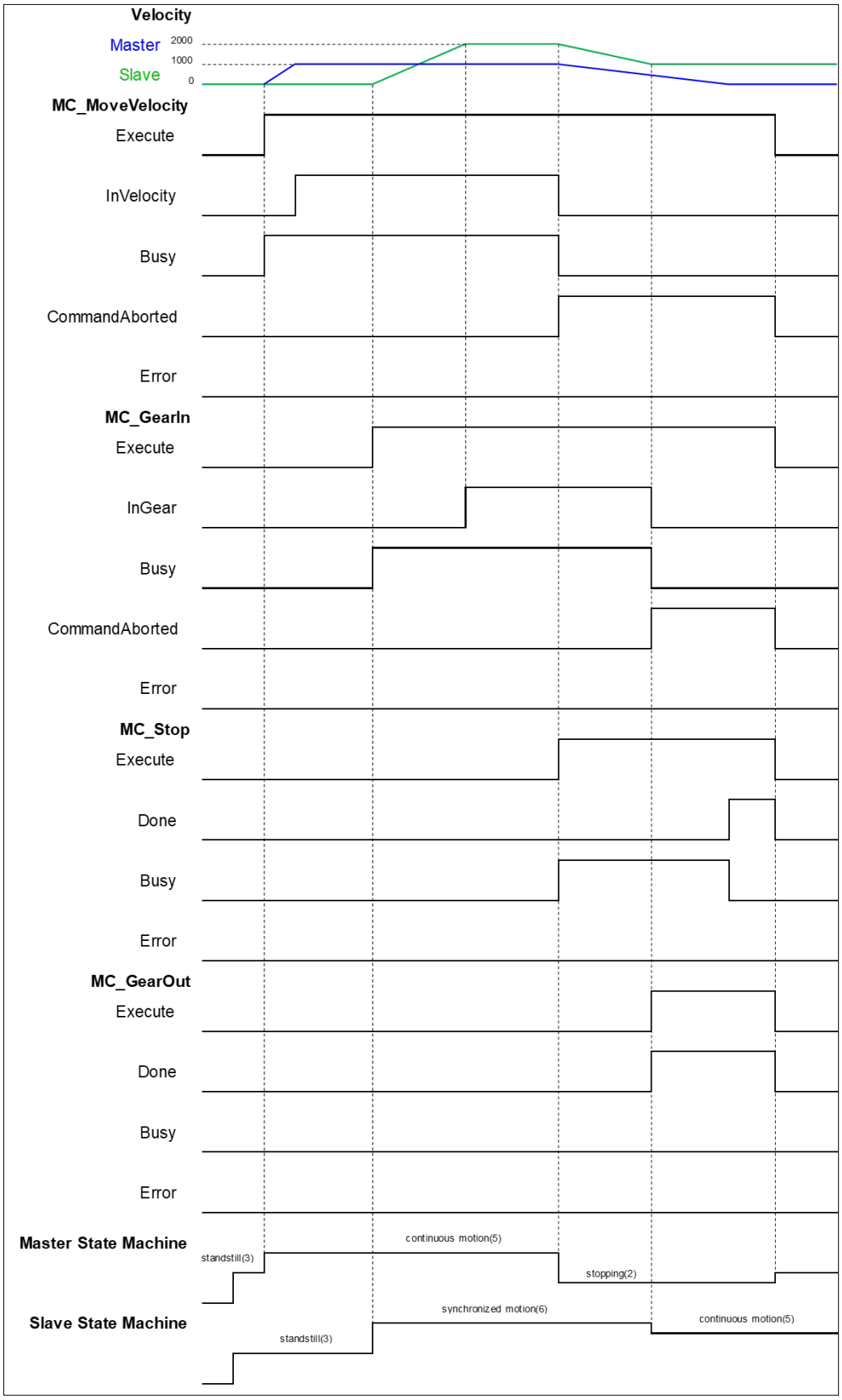
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

  - The following example describes the corresponding motion state throughout the gear operation via gear-related instructions.



  - **Timing Diagram**



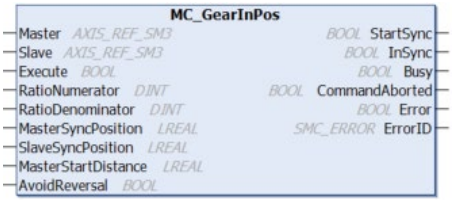
- When Execute of MC_MoveVelocity changes to True, master axis starts to move.
  - When M1 Execute of MC_GearIn changes to True, the slave axis starts to catch the master axis.

- If the velocity of salve axis reaches doubled master axis' velocity (RatioNumerator: RatioDenominator = 2: 1), *InGear* of MC_GearIn will change to True. After the master axis is synchronized with slave axis, the state of slave axis turns to Synchronized Motion.
- When Execute of MC_Stop changes to True, the master axis starts to decelerate. At the same time, the slave axis also decelerates based on the gear ratio.
- In the process of the MC_Stop execution, when Execute of MC_GearOut changes to True, master and slave axis will be under asynchronous status and slave axis will keep moving at the speed (Continuous Motion state) when the gear relation is decoupled.

2.1.17 MC_GearInPos

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_GearInPos establish a gear synchronization relationship between the master axis and the slave axis at the specified location.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_GearInPos		MC_GearInPos_instance (Master : =, Slave : =, Execute : =, RatioNumerator : =, RatioDenominator : =, MasterSyncPosition : =, SlaveSyncPosition : =, MasterStartDistance : =, AvoidReversal : =, StartSync =>, InSync =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
RatioNumerator	Gear ratio numerator* between master and slave axis.	DINT	Negative, positive or 0(1)	When *Execute* turns to True and *Busy* is False.
RatioDenominator	Gear ratio denominato* between master and slave axis.	UDINT	Positive or 0(1)	When *Execute* turns to True and *Busy* is False.
MasterSyncPosition	Master Position at which the axes are synchronized.	LREAL	Negative, positive or 0(0)	When *Execute* turns to True and *Busy* is False.
SlaveSyncPosition	Slave Position at which the axes are synchronized.	LREAL	Negative, positive or 0(0)	When *Execute* turns to True and *Busy* is False.
MasterStartDistance	Master Distance for synchronization procedure.	LREAL	Negative, positive or 0(0)	When *Execute* turns to True and *Busy* is False.
AvoidReversal	Reversal is not allowed.	BOOL	True/False (False)	When *Execute* turns to True and *Busy* is False.

***Note:** A negative gear ratio will make the master and slave axis move in an opposite direction.

- **Outputs**

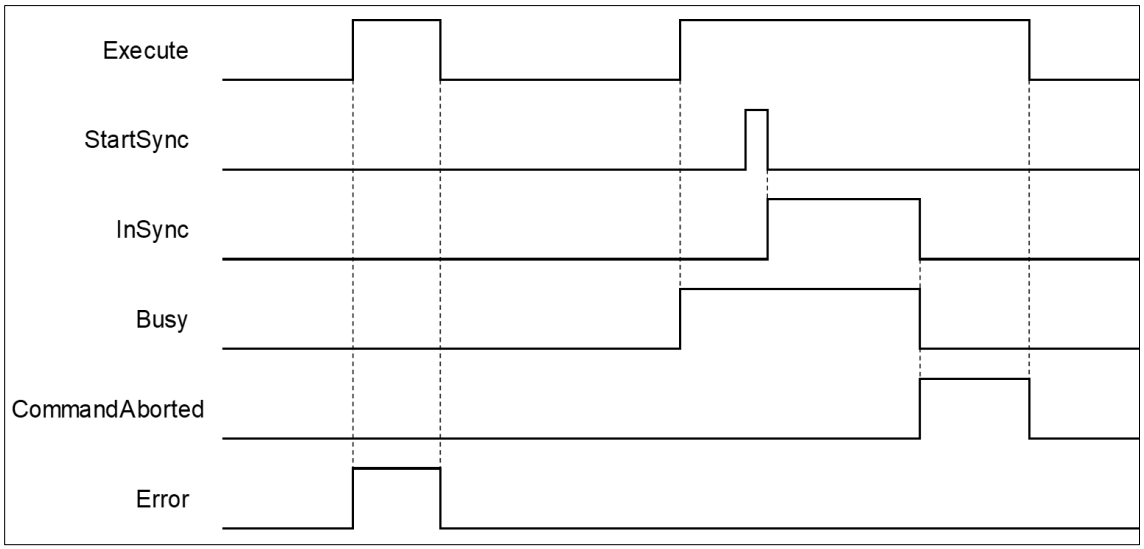
Name	Function	Data Type	Output Range (Default Value)
StartSync	True when the synchronization starts.	BOOL	True/False (False)
InSync	True when the synchronization is ongoing.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
StartSync	• When master axis travels to the start position.	• When *InSync* turns to True. • When *Error* turns to True.
InSync	• When the synchronization between master and slave axis is completed.	• When *CommandAborted* turns to True. • When *Error* turns to True.
Busy	• After the synchronization begins.	• When *CommandAborted* turns to True. • When *Error* turns to True.
CommandAborted	• When MC_GearOut is run. • True when the instruction is aborted by another function block. • When this instruction is aborted because of the execution of MC_Stop instruction.	• When *Execute* changes to False. • If *Execute* is False and *CommandAborted* turns to True, *CommandAborted* will be True for only one scan cycle and immediately shift to False.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Master	Specifies the master axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and Busy is False.
Slave	Specifies the slave axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and Busy is False.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

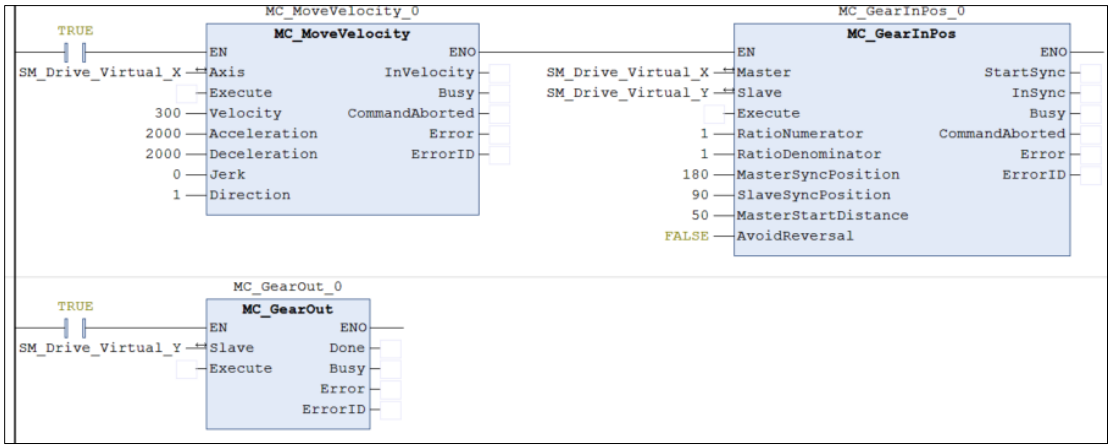
  - Position where the master axis runs StartSync = MasterSyncPosition – MasterStartDistance.
  - It's a must to ensure an appropriate parameter settings of sync position if both master and slave axis are working under Finite mode. Supposed the master and slave axis moving in the positive direction, if the master axis position missed the StartSync position, the gear will not be able to run normally. Therefore, it is suggested to set the master and slave axis operating under Modulo mode.
  - During the progress of synchronization between the master and slave axis, MC_GearInPos begins to plan the motion path of slave axis automatically with gear ratio based on parameters of the position where master axis runs StartSync, MasterSyncPosition and SlaveSyncPosition. After synchronizing complete, slave axis will start to move by following master axis.
  - When MasterStartDistance = 0 or being negative, CAM motion will be completed immediately.

- **Troubleshooting**

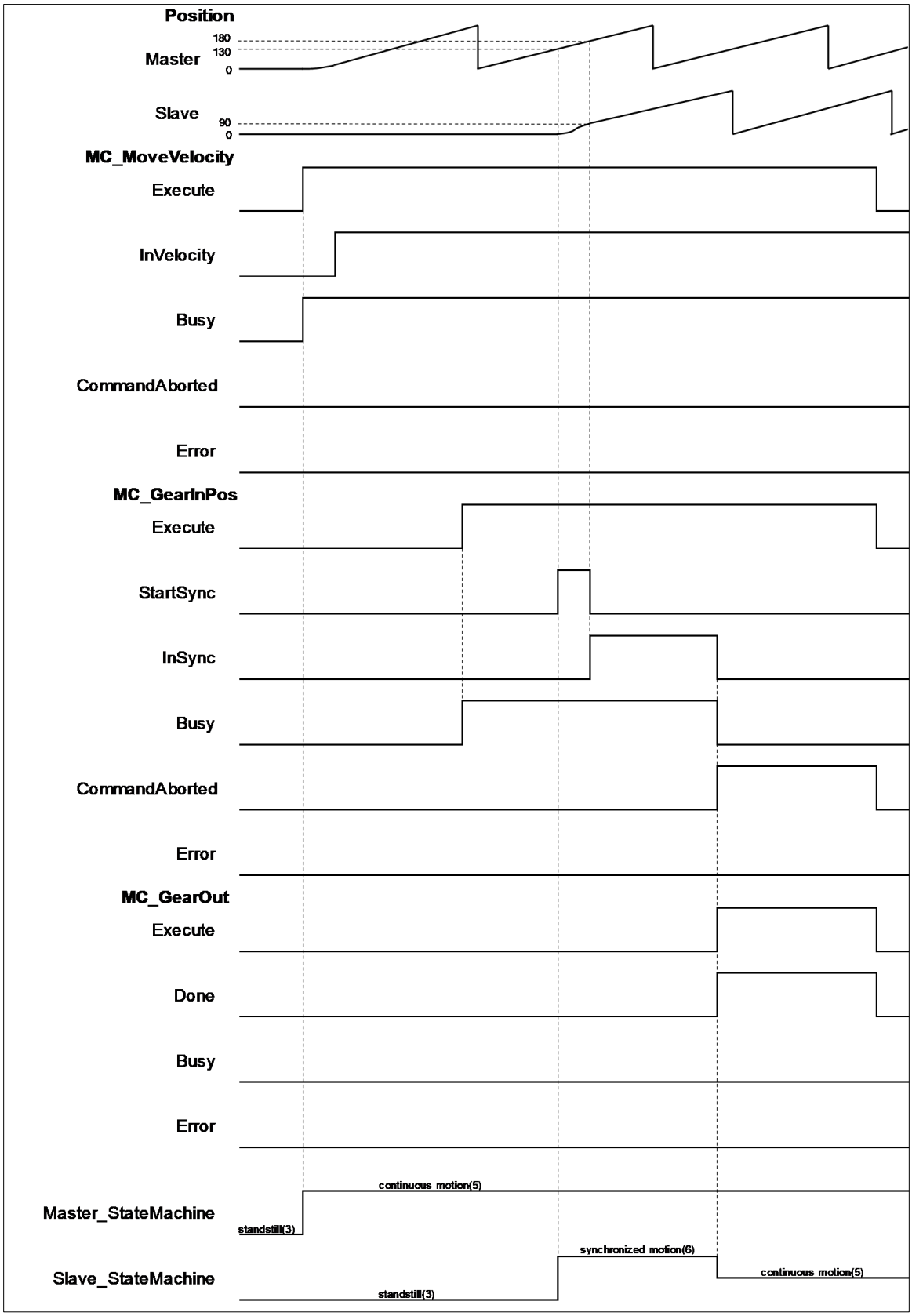
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

  - The following example describes the corresponding motion state throughout the gear operation via MC_GearInPos-related instructions.



- **Timing Diagram**



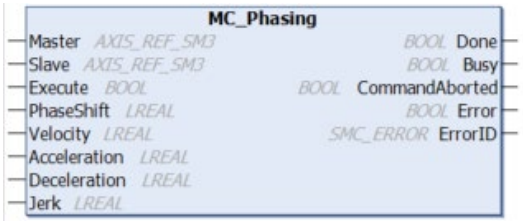
- When *Execute of* MC_MoveVelocity turns to True, the master axis starts to move.
- When *Execute of* MC_GearInPos turns to True, it's waiting for master axis to reach StartSync position.
- When StartSync position is reached, Outputs*StartSync* of MC_GearInPos turns to True. At the same time, a motion curve is planned for slave axis to move, which the axis will enter Synchronized Motion state.

- When both master and slave axis reach the synchronization position, Outputs*InSync* of MC_GearInPos turns to True and Outputs*StartSync* changes to False.
- When *Execute* of MC_GearOut turns to True, the master and slave axis move asynchronously, entering Continuous Motion state.

2.1.18 MC_Phasing

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_Phasing specifies the phase shift value between the master and slave axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Phasing		MC_Phasing_instance (Master : =, Slave : =, Execute : =, PhaseShift : =, Velocity : =, Acceleration : =, Deceleration : =, Jerk : =, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
PhaseShift	Phase shift amount between master and slave axis*	LREAL	Negative, positive or 0(0)	When *Execute* turns to True and *Busy* is False.
Velocity	The max velocity of the phase shift amount (Unit: user unit/s)	LREAL	Positive or 0(0)	When *Execute* turns to True and *Busy* is False.
Acceleration	The max acceleration of the phase shift amount (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Deceleration	The max deceleration of the phase shift amount (Unit: user unit/s^2)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.
Jerk	The max Jerk value of the phase shift amount (Unit: user unit/s^3)	LREAL	Positive(0)	When *Execute* turns to True and *Busy* is False.

* **Note:** In case of positive values, the slave axis is behind the master axis. Conversely, the slave axis is ahead of the master axis when the value is negative.

Outputs

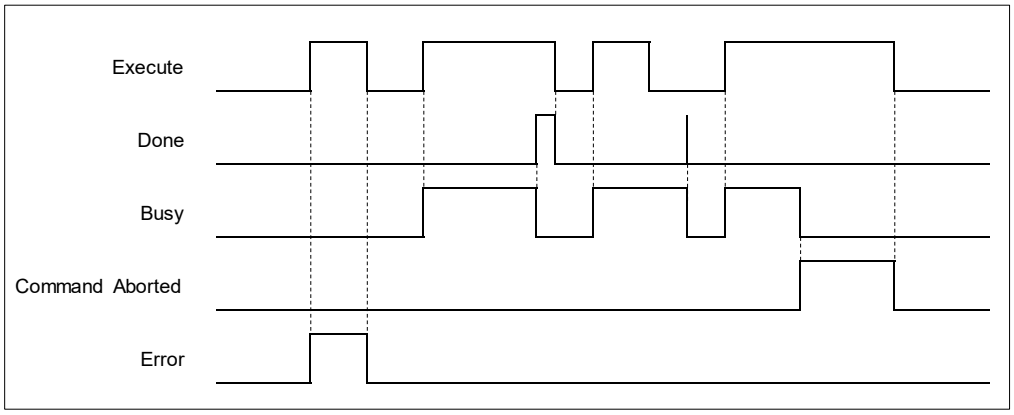
Name	Function	Data Type	Output Range (Default Value)
Done	True when phasing operation is completed.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
CommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

* **Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• When the phasing operation is completed.	• When *Execute* changes to False. • If *Execute* is False and *Done* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Busy	• When the phasing operation is run.	• When *Error* turns to True. • When *CommandAborted* turns to True.
CommandAborted	• When this instruction is aborted by another instruction with the Buffer Mode set to Aborting. • When this instruction is aborted because of the execution of MC_Stop instruction.	• When *Execute* changes to False. • If *Execute* is False and *CommandAborted* turns to True, it will be True for only one period and immediately shift to False.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



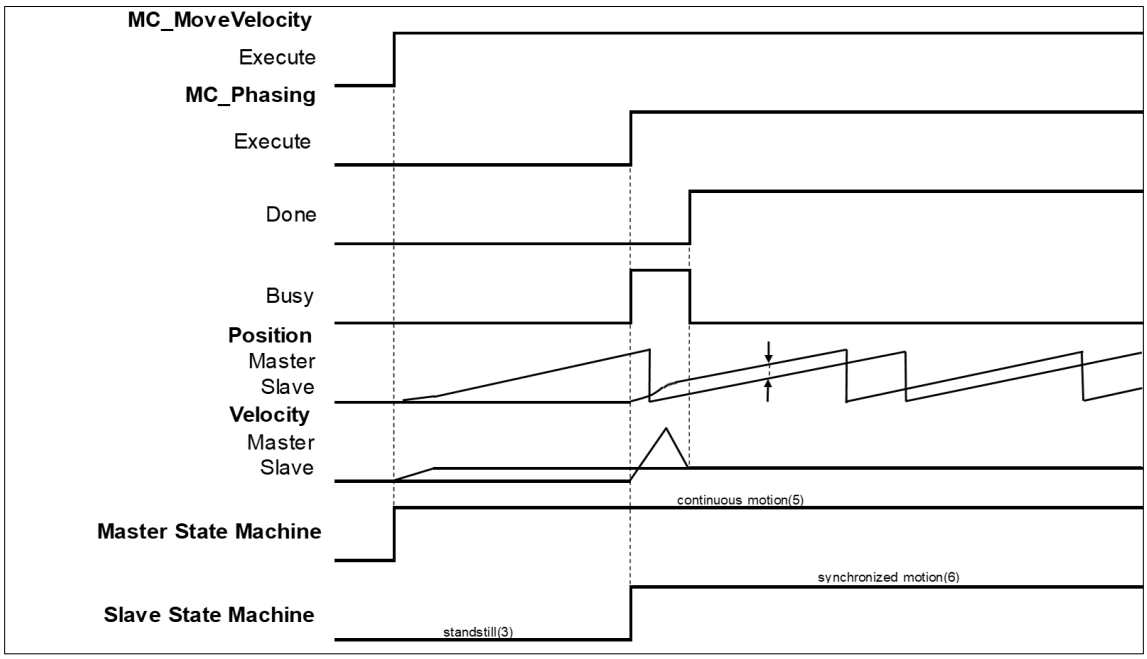
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Master	Specifies the master axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When Execute turns to True and Busy is False.
Slave	Specifies the slave axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When Execute turns to True and Busy is False.

* **Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

  - When *Execute* turns to True and the master-slave axis relation is established, the slave axis will shift the phase by planning a smooth curve. In case of a positive PhaseShift value, the slave axis is behind the master axis. Conversely, the slave axis is ahead of the master axis when the value is negative.
  - The position of master axis remains unchanged while MC_Phasing acts on the slave axis



  - MC_Phasing can be used when the state is not under Synchronized motion.

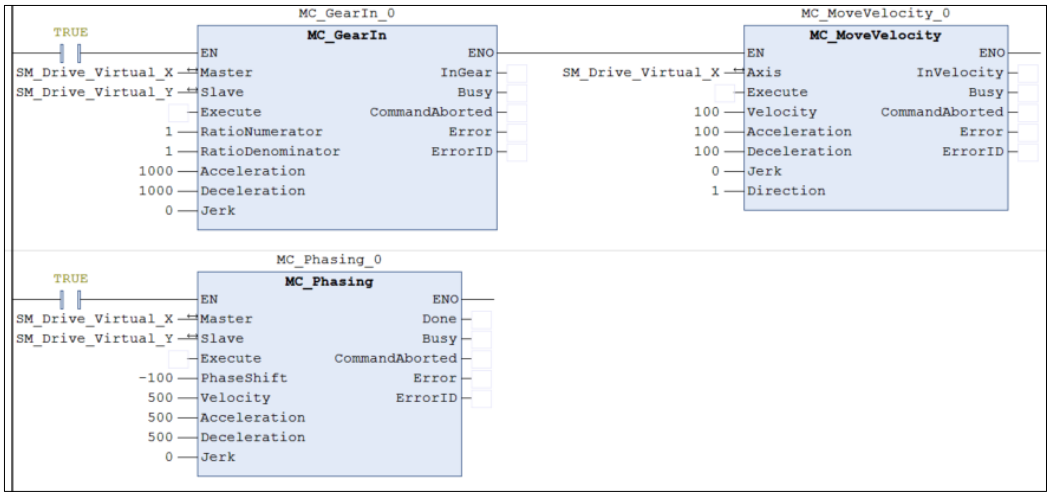
    - When MC_Phasing is run, the state of slave axis will remain as Synchronized motion.
    - When runs MC_Phasing before establishing gear relationship between the master and slave axis, the slave axis will be directly synchronized with the master axis and both move based on the gear ratio which is 1: 1.
    - When the slave axis runs MC_Phasing, it can be aborted by other single-axis function blocks and the synchronous relationship will be disconnected.

- **Troubleshooting**

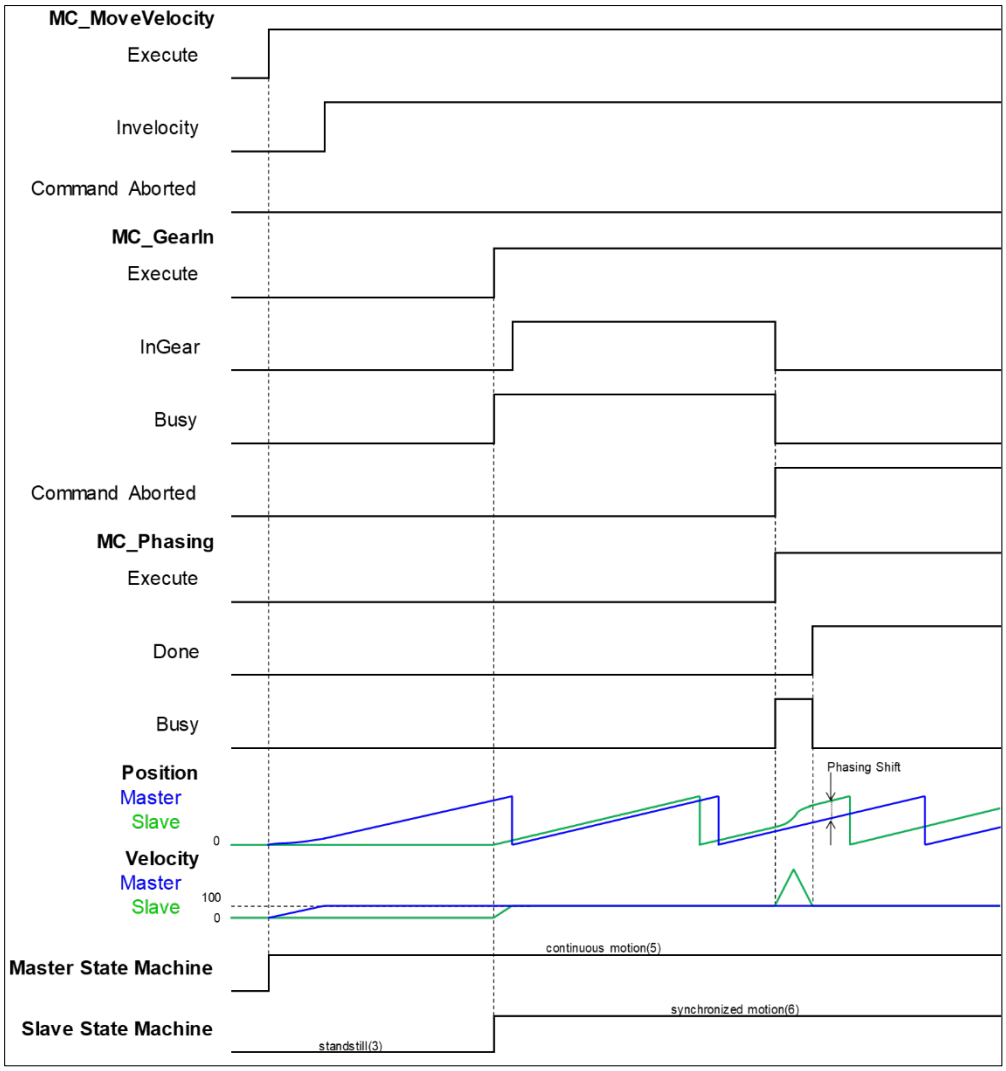
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

- The following example describes the corresponding motion state and path throughout the gear operation.



- **Timing Diagram**



  - Execute MC_MoveVelocity to make the master axis run at a constant speed, then execute MC_GearIn to establish gear relationship between the master and slave axis.
  - When *Execute* of MC_Phasing turns to True, the relationship between the two axes will be disconnected. MC_Phasing turns the phase of the slave axis by the specified PhaseShift value.

- When the slave axis reaches the specified value, Done of MC_Phasing turns to True and the Busy output is reset.

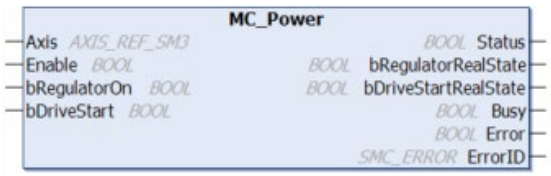
2.2 Administrative Motion Control Instructions

Administrative motion control instructions refer to the actions of configuring corresponding settings and retrieving related information made for drivers, which will not cause actual displacement of motors. The function blocks used in this chapter are from the library “SM3_Basic” and can operate synchronously with drives. As a result, synchronous axis type should be selected in axis settings. For more details about configuration related to synchronous axes, refer to section 7.4 in AX-3 Series Operational Manual.

2.2.1 MC_Power

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_Power enables or disables the specific axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Power		MC_Power_instance(Axis : =, Enable : =, bRegulatorOn: =, bDriveStart : =, Status =>, bRegulatorRealState =>, bDriveStartRealState =>, Busy =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
bRegulatorOn	Enables the power	BOOL	True/False (False)	Only when *Enable*=True.
bDriveStart	Controls the QuickStop mechanism.	BOOL	True/False (False)	Only when *Enable*=True.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Status	The specific axis is ready to be moved by the function blocks.	BOOL	True/False (False)
bRegulatorRealState	The power is turned ON.	BOOL	True/False (False)
bDriveStartRealState	Quick stop function is applicable on the device.	BOOL	True/False (False)

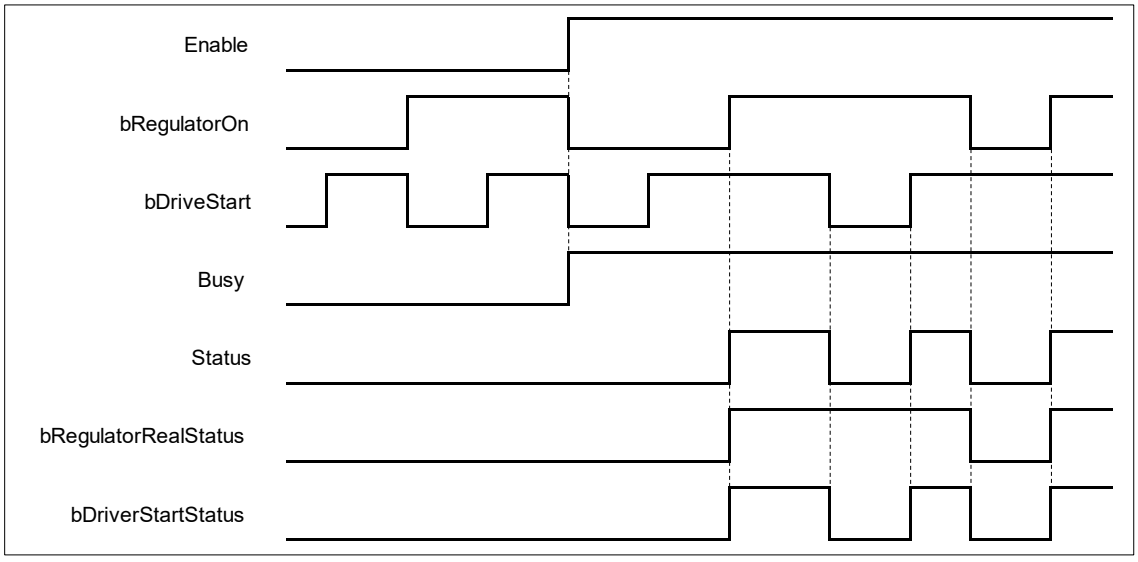
Name	Function	Data Type	Output Range (Default Value)
Busy	Function block is operating.	BOOL	True/False (False)
Error	Errors occur in function block.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Status	• When Enable is true upon detecting rising edge for both *bRegulatorRealState* and *bDriveStartRealState*.	• When *Enable* is true, *bRegulatorRealState* or *bDriveStartRealState* shifts to False. • When *Error* turns to True.
bRegulatorRealState	• When *Enable* is true upon detection of rising edge for *bRegulatorRealState*.	• When *Enable* is true, bRegulatorRealState shifts to False. • When *Error* turns to True.
bDriveStartRealState	• When *Enable* is true, both *bRegulatorRealState* and *bDriveStartRealState* are True.	• When *Enable* is true, *bRegulatorRealState* or *bDriveStartRealState* shifts to False. • When Error turns to True.
Busy	• When *Enable* turns to true	• When *Enable* turns to False. • When *Error* turns to True.
Error	• When errors occur in the execution conditions or the input values.	• When errors are cleared.
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

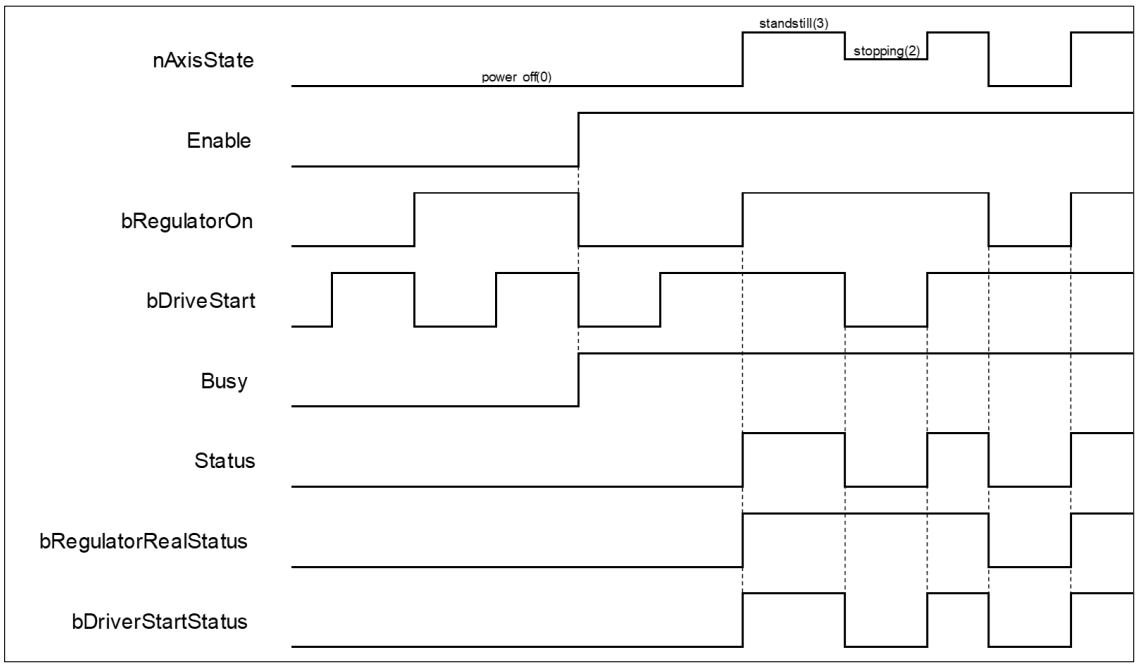
Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Enable* turns to True.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

  - When Enable is True, bRegulatorOn and bDriveStart are effective.
  - When Enable, bRegulatorOn and bDriveStart are True. Status turns to True and nAxisState turns to Standstill.
  - When Enable and bRegulatorOn are True, set DriveStart to be False which nAxisState(axis status) turns to Stopping.
  - When Enable and bDriveStart are True, set RegulatorOn to be False which nAxisState(axis status) directly turns to Disabled.

  - Timing Diagram

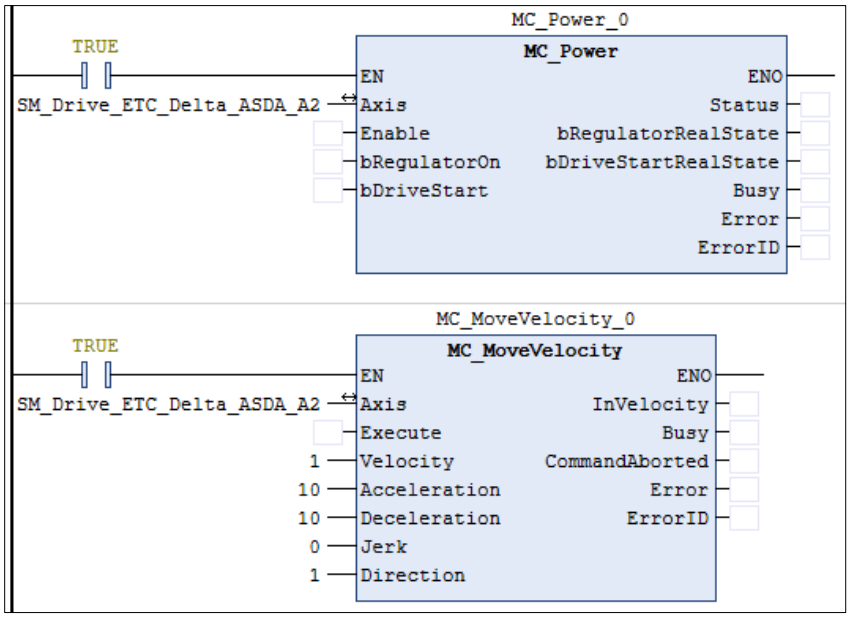


- **Troubleshooting**

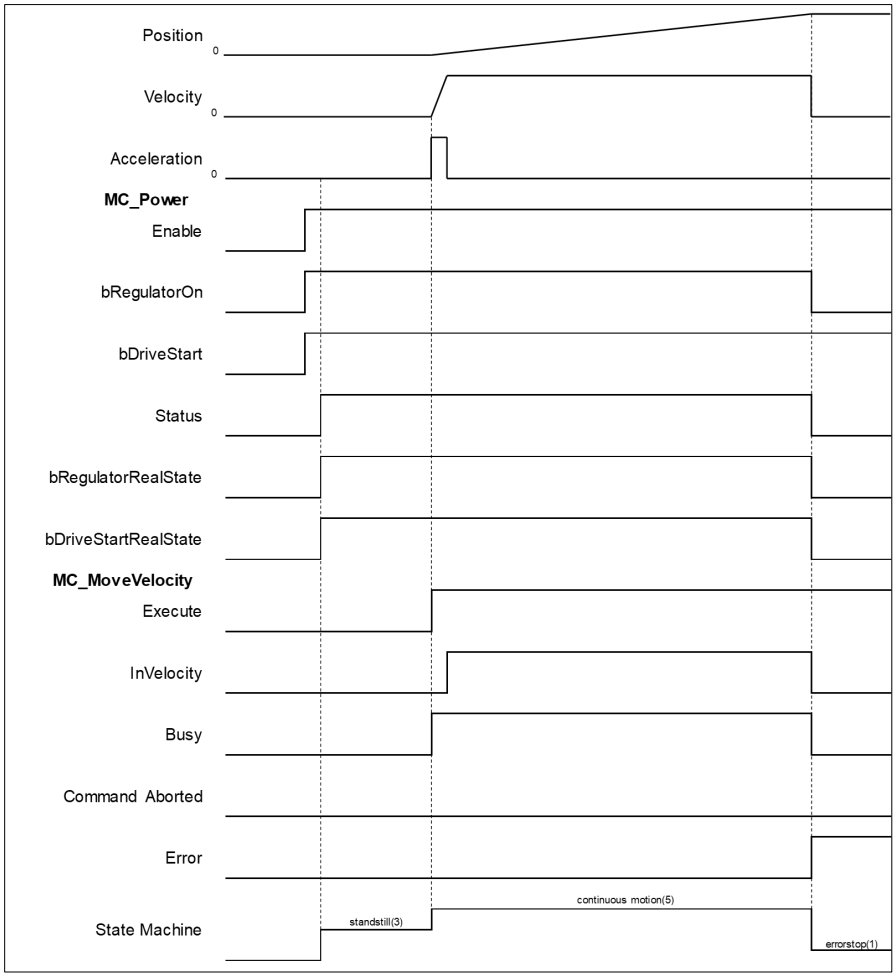
  - If an error occurs during the execution of the instruction or the axis is in an Errorstop state, *Error* turns to True and the axis will be decelerated to a stop. You can refer to ErrorID (Error Code) to address the problem.

- **Example:**

  - Example1: The following example explains the movement of a moving axis when *bRegulatorOn* of MC_Power turns to False.



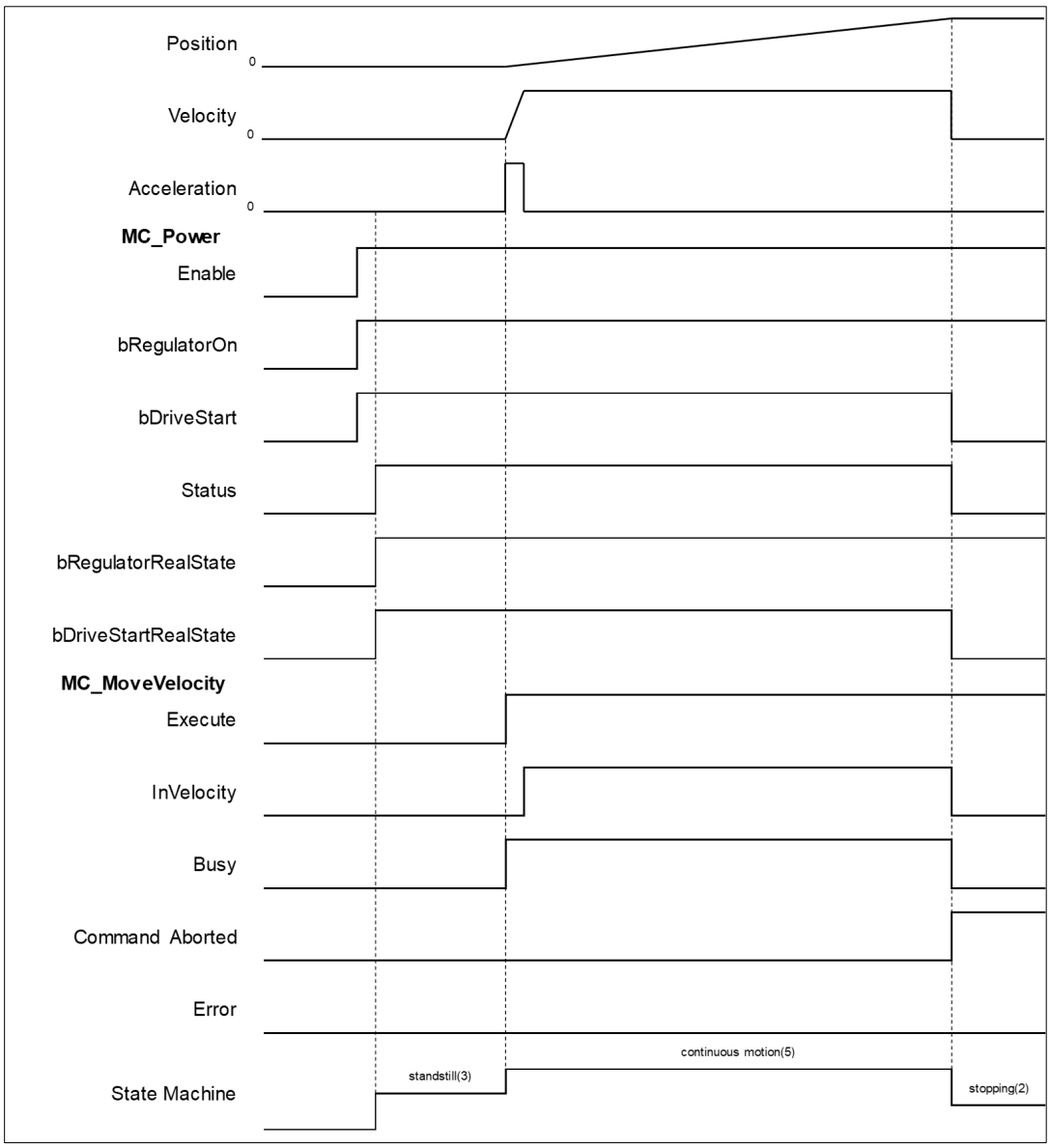
■ **Timing Diagram**



- Shift *bRegulatorOn* of MC_Power from True to False while the axis is moving to stop the axis immediately.
- At the same time, an error of SMC_REGULATOR_OR_START_NOT_SET(20) occurs in MC_MoveVelocity and the axis state directly switches from continuous_motion to errorstop.

- Example 2: Continue with example 1. The following example explains the movement of a moving axis when *bDriveStart* of MC_Powers turns to False.

- **Timing Diagram**



- Shift *bDriveStart* of MC_Power from True to False while the axis is moving. Such action will stop the axis immediately.
- *CommandAbort* turns to True and aborts MC_MoveVelocity, while the axis status switches from continuous_motion to stopping.

2.2.2 MC_SetPosition

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_SetPositionn changes the current position by shifting the coordinates of an axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_SetPosition		MC_SetPosition_instance(Axis : =, Execute : =, Position : =, Mode : =, Done =>, Busy =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (True)	-
Position	Axis position (User-defined unit)	LREAL	Positive, negative, or 0 (0)	When *Execute* turns to True and Busy is False.
Mode	Specify relative position (True) or absolute position(False)	BOOL	True/False (False)	When *Execute* turns to True and Busy is False.

- **Outputs**

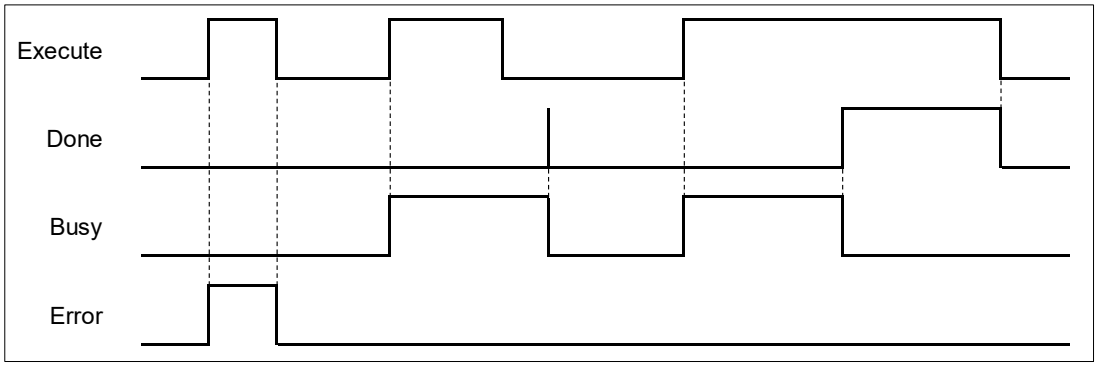
Name	Function	Data Type	Output Range (Default Value)
Done	True when finishes coordinate modification.	BOOL	True/False (False)
Busy	True when the instruction is running.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

  - **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• When the modification to coordinate is completed.	• When *Execute* turns from True to False. • If Execute is False and Done turns to True, Done will be True for only one scan cycle and immediately shift to False.
Busy	• When *Execute* is triggered to be True.	• When *Done* turns to True. • When *Error* turns to True.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and Busy is False.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

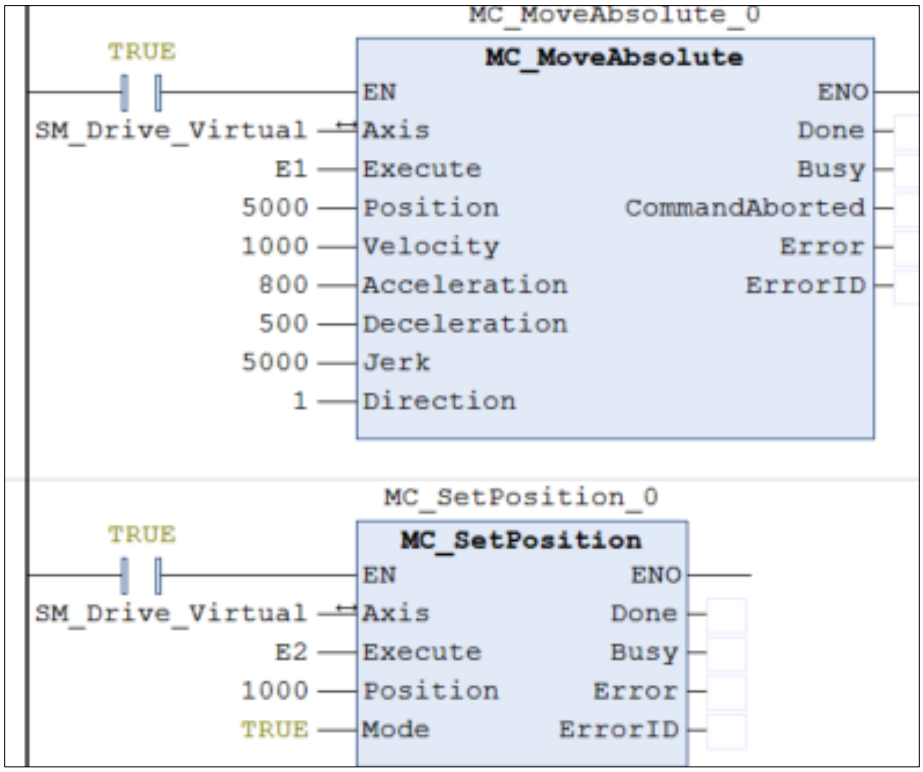
  - When configuring the parameters of axis position via MC_Position, there's no displacement made by the axis itself but instead, only the coordinate system moves.
  - To avoid possible position jumps occurring to the slave axis, you should avoid running MC_SetPosition to the synchronized master axis, or discontinuous jumps in velocity will exist in the slave axis.
  - The value of Position will be added directly to the current position under the relative mode, which will be the new location coordinate. For the absolute position, the value of Position will be set to the current location coordinate.

- **Troubleshooting**

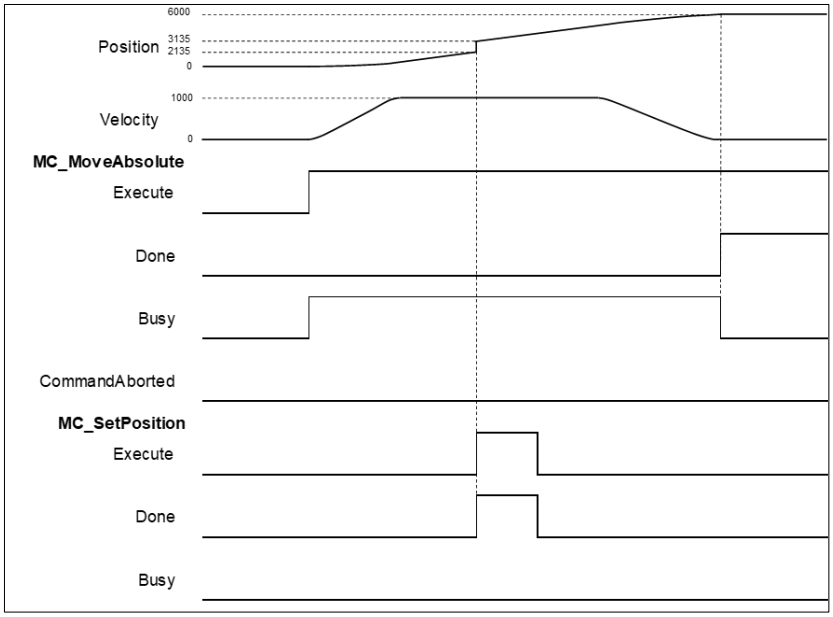
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

  - Example1: Demonstrate the condition of executing MC_MoveAbsolute while using MC_SetPosition to change the coordinate system under relative mode.

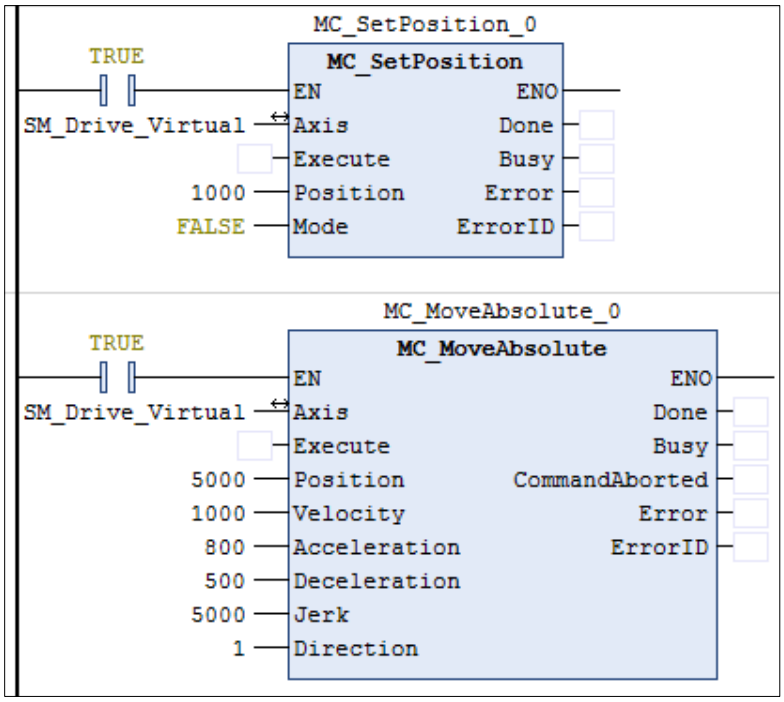


- **Timing Diagram**

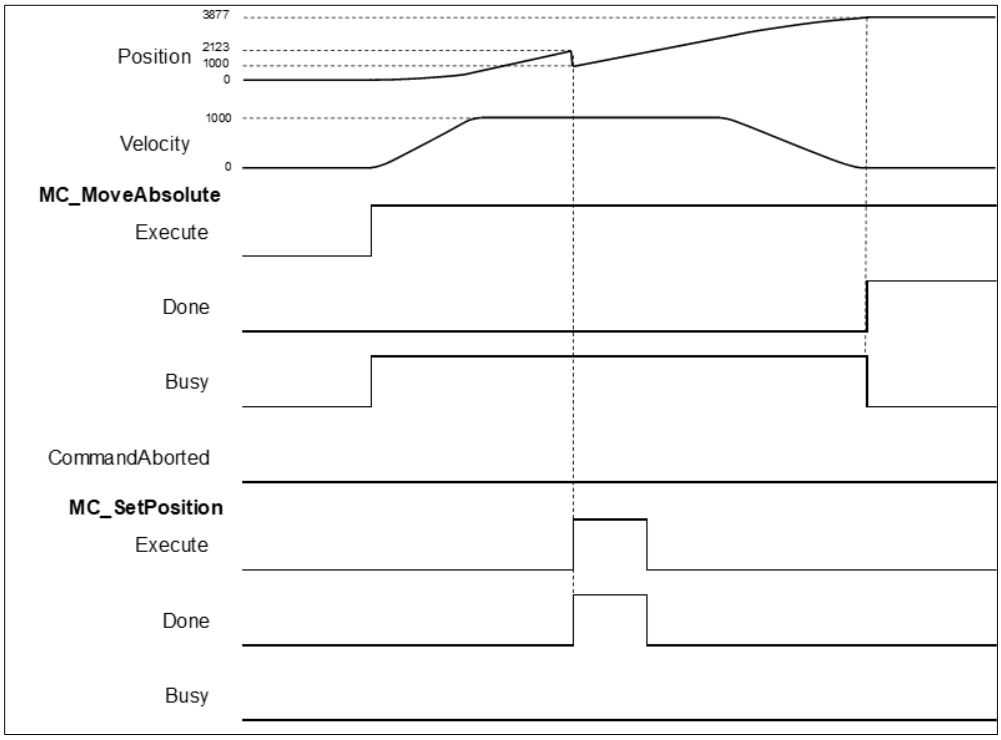


  - After MC_MoveAbsolute is run, use MC_SetPosition to define a new coordinate system under relative mode.
  - When detecting a rising edge of Execute of MC_SetPosition, the new location of the axis will be 3135 after adjusting the coordinate system, in which the axis was originally located at 2135.
  - Done is True when the execution of MC_MoveAbsolute is completed and the current position will be at 6000. At this time, MC_MoveAbsolute still moves to 5000 on the old coordinate system, which will become 6000 after the coordinate adjustment.

- Example 2: Demonstrate the condition of executing MC_MoveAbsolute while using MC_SetPosition to change the coordinate system under absolute mode.

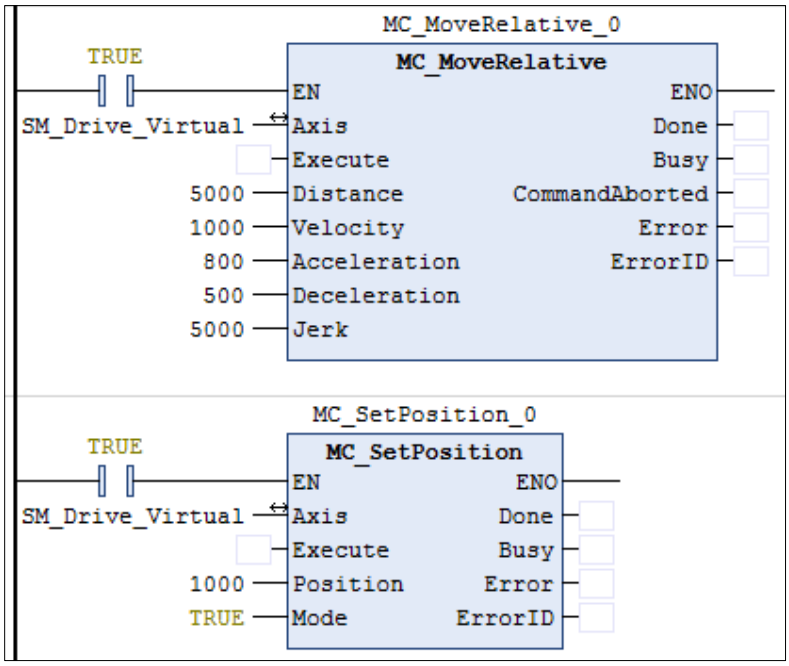


- **Timing Diagram**

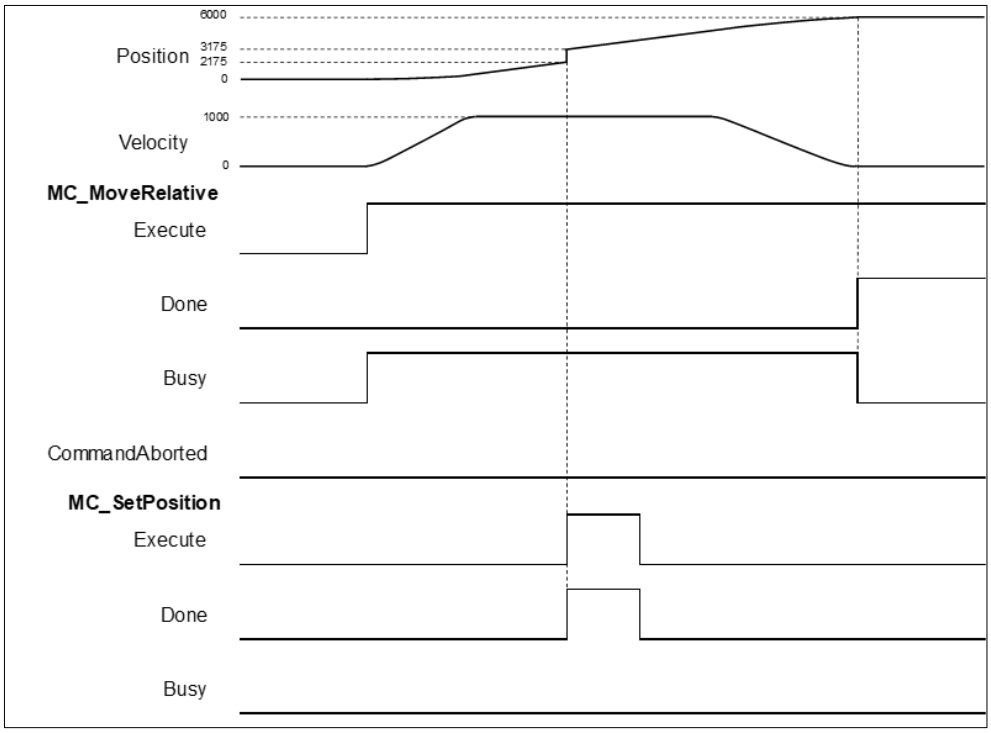


  - After MC_MoveAbsolute is run, use MC_SetPosition to define a new coordinate system under absolute mode.
  - When detecting a rising edge of Execute of MC_SetPosition, the new location of the axis will be 1000 after adjusting the coordinate system by moving a distance of 1123. The axis was originally located at 2123.
  - Done is True when the execution of MC_MoveAbsolute is completed and the current position will be at 3877(5000 - 1123). At this time, MC_MoveAbsolute still moves to 5000 on the old coordinate system, which will become 3877 after the coordinate adjustment.

- Example 3: Demonstrate the condition of executing MC_MoveRelative while using MC_SetPosition to change the coordinate system under relative mode.



- Timing Diagram

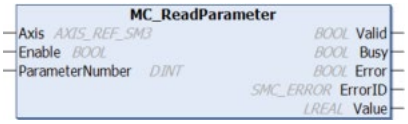


- After MC_MoveRelative is run, use MC_SetPosition to define a new coordinate system under relative mode.
- When detecting a rising edge of Execute of MC_SetPosition, the new location of the axis will be 3175 after adjusting the coordinate system, which the axis was originally located at 2175.
- Done is True when the execution of MC_MoveAbsolute is completed and the current position will be at 6000. At this time, MC_MoveAbsolute still moves to 5000 on the old coordinate system, which will become 6000 after the coordinate adjustment.

2.2.3 MC_ReadParameter

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_ReadParameter reads a value of a specific axis parameter.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_ReadParameter		MC_ReadParameter_instance(Axis : =, Enable : =, ParameterNumber : =, Valid =>, Busy =>, Error =>, ErrorID =>, Value =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	Run the function block.	BOOL	True/False (False)	-
ParameterNumber	Number of the parameter to be read.	DINT	Positive, negative, or 0 (0)	When *Enable* is detected to be rising edge.

- **Outputs**

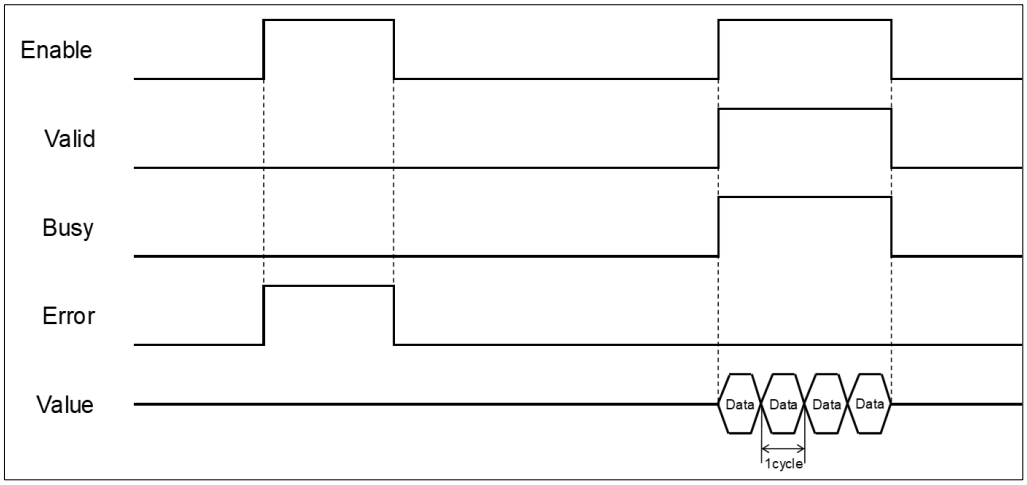
Name	Function	Data Type	Output Range (Default Value)
Valid	True when the parameter to be read exists and can be further processed.	BOOL	True/False (False)
Busy	True when the function block is being run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	When a command error occurs, record the error code. For the detailed description of the error code, refer to the manual's Appendix.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
Value	Value of the parameter to read.	LREAL*2	Positive, negative, or 0 (0)

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Valid	• When *Enable* is triggered to True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.
Busy	• When *Enable* is triggered to True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.
Error	• When an error occurs in executing conditions or input values.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		
Value	• When *Valid* is True and there're ongoing updates.	• When *Valid* is False and stops updating.

- **Timing Diagram**



***Note:** Data = parameter's value. One cycle = one task period

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Enable* is True.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

  - How to use MC_ReadParameter to read the parameter values of the desired object with its parameter number in EtherCAT Object Dictionary:
    - Use SHL instruction to move the data length of the desired object to the left for 24 bits.
    - Use SHL instruction to move the index of the desired object to the left for 8 bits.
    - The input ParameterNumber must contain the data length of the index and the subindex. Refer to the following formula:

      ParameterNumber: = - DWORD_TO_DINT(SHL(TO_DWORD(data length of object dictionary), 24) + SHL(TO_DWORD(index of object dictionary), 8) + object sub-index);
  - To read axis parameters, you'll need to enter the parameter number of AXIS_REF_SM3(FB) into the ParameterNumber input.

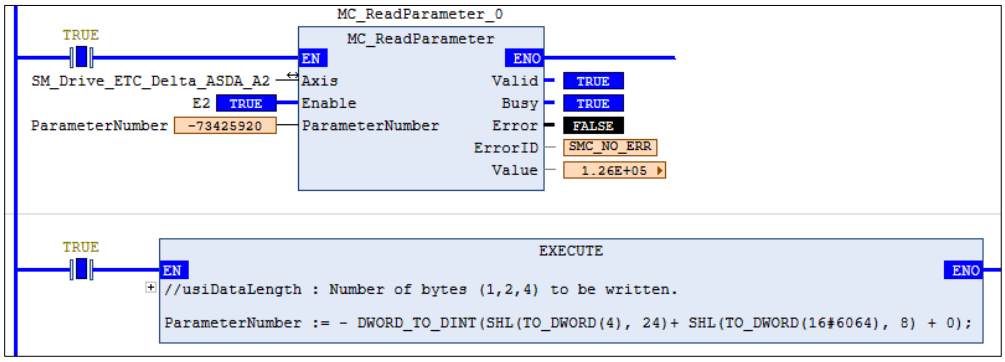
- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error

Code) to address the problem.

- **Example**

  - This example demonstrates how to use MC_ReadParameter to read the value of object 0x6064(current position returned by motor) in the drive.



  - Input the data length, index, and subindex of the object to the above formula and you'll get the ParameterNumber, which should be entered into the ParameterNumber input. After that, whenever the FB MC_ReadParameter is run, it will visit the object dictionary specified by the drive and return the values.
  - The following figure shows the information related to object parameters 0x6064.

Object 6064_h: Position actual value

INDEX	6064_h
Name	Position actual value
Object Code	VAR
Data Type	INTEGER32
Access	RO
PDO Mapping	Yes
Value Range	INTEGER32
Default Value	0
Comment	單位：PUU

2.2.4 MC_WriteParameter

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_WriteParameter writes a value to a specific parameter.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_WriteParameter		MC_WriteParameter_instance(Axis : =, Execute : =, ParameterNumber : =, Value : =, Done =>, Busy =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	Run the function block.	BOOL	True/False (False)	-
ParameterNumber	Number of the parameter to be written.	DINT	Positive, negative, or 0 (0)	When *Execute* turns to True and *Busy* is False.
Value	Value to be written to the parameter.	LREAL	Positive, negative, or 0 (0)	When *Execute* turns to True and *Busy* is False.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	True if the value is written successfully.	BOOL	True/False (False)
Busy	True when the function block is being run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

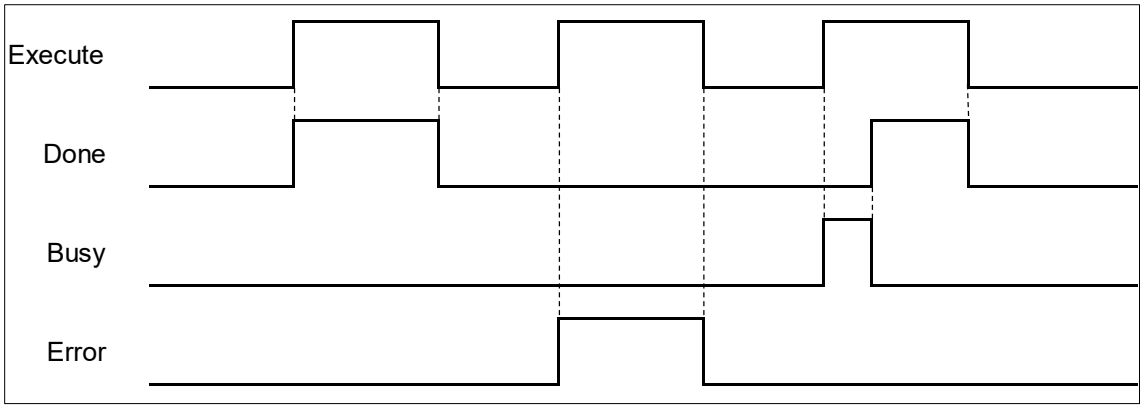
***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False

Done	• When the value is written successfully.	• When *Execute* turns from True to False.
Busy	• When *Execute* is triggered to be True. • When the value is being written to the parameter.	• When *Done* turns to True. • When *Error* turns to True.
Error ErrorID	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and *Busy* is False.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

  - How to use MC_ReadParameter to read the parameter values of the desired object with its parameter number in EtherCAT Object Dictionary:
    - Use SHL instruction to move the data length of the desired object to the left for 24 bits.
    - Use SHL instruction to move the index of the desired object to the left for 8 bits.
    - The input ParameterNumber must contains the data length of the index and the subindex. Refer to the following formula:

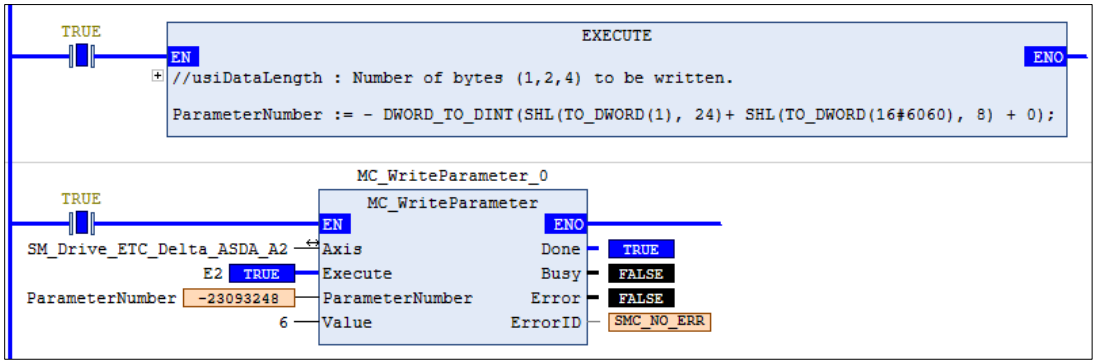
    ParameterNumber: = - DWORD_TO_DINT(SHL(TO_DWORD (data length of object dictionary), 24) + SHL(TO_DWORD(index of object dictionary), 8) + object sub-index);
  - To write the value to the parameter, you'll need to input the parameter number of AXIS_REF_SM3(FB) to ParameterNumber.
  - Write parameter values to the input fSetPosition by using MC_WriteParameter while the axis is moving. The value of fSetPosition is changed for only one task cycle time in EtherCAT, then fSetPosition resumes its original planned motion curve to move.

- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

  - This example demonstrates how to use MC_WriteParameter to write the value to the object 0x6060(operation mode) in the drive.



  - Input the data length, index, and subindex of the object to the above formula and you'll get the ParameterNumber, which should be entered into the ParameterNumber input. After the value is written to MC_WriteParameter successfully, the control mode of the drive will change to 6.

  - The following figure shows the information related to the parameters of object 0x6060

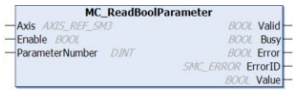
Object 6060_h: Modes of operation

INDEX	6060_h
Name	Modes of operation
Object Code	VAR
Data Type	INTEGER8
Access	RW
PDO Mapping	Yes
Value Range	INTEGER8
Default Value	0
Comment	0: Reserved

2.2.5 MC_ReadBoolParameter

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_ReadBoolParameter reads the value of a specific Boolean parameter.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_ReadBoolParameter		MC_ReadBoolParameter_instance(Axis : =, Enable : =, ParameterNumber : =, Valid =>, Busy =>, Error =>, ErrorID =>, Value =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	Run the function block.	BOOL	True/False (False)	-
ParameterNumber	Number of the specific Boolean parameter	DINT	Positive, negative, or 0 (0)	When *Enable* turns from False to True.

- **Outputs**

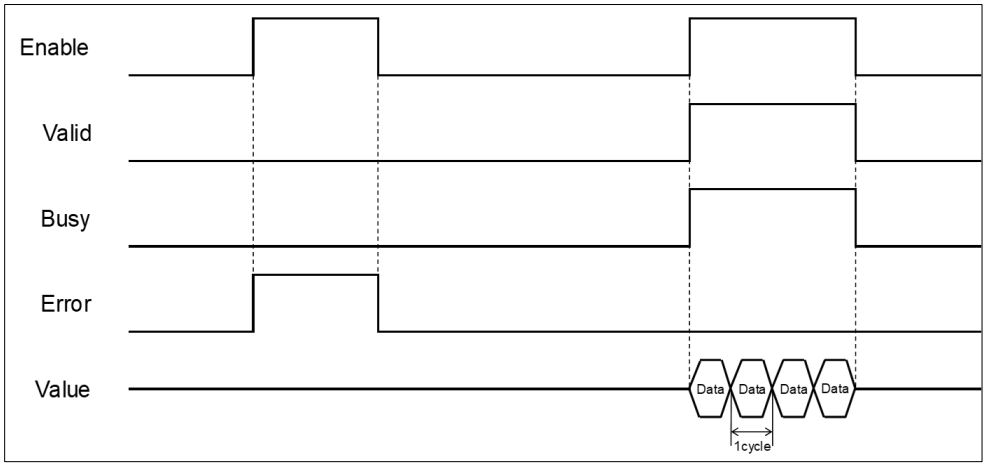
Name	Function	Data Type	Output Range (Default Value)
Valid	True when the parameter to read exists and can be further processed.	BOOL	True/False (False)
Busy	True when the function block is being run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Error codes.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
Value	Value of the parameter to read.	BOOL	True/False (False)

***Note:** SMC_ERROR: Enumeration (Enum)

  - **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Valid	• When *Enable* is triggered to True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.
Busy	• When *Enable* is triggered to True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.
Error	• When an error occurs in executing conditions or input values.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		
Value	• When *Valid* is True and there're ongoing updates.	• When *Valid* is False and stop updating.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Enable* turns to True.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

  - How to use MC_ReadBoolParameter to read the parameter values of desired object with its parameter number in EtherCAT Object Dictionary:
    - Use SHL instruction to move the data length of desired object to the left for 24 bits.
    - Use SHL instruction to move the index of desired object to the left for 8 bits.
    - The input ParameterNumber must contains the data length the index and the subindex. Refer to the following formula:
      ParameterNumber: = - DWORD_TO_DINT (SHL (TO_DWORD (data length of object dictionary), 24) + SHL (TO_DWORD (index of object dictionary), 8) + object sub-index);
  - For operation example, refer to the example in MC_ReadParameter.
  - To read axis parameters, you will need to enter the parameter number of AXIS_REF_SM3 (FB) to ParameterNumber input.

- **Troubleshooting**

- If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

2.2.6 MC_WriteBoolParameter

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_WriteBoolParameter writes a Boolean value to a specific parameter.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_WriteBool Parameter		MC_WriteBoolParameter_instance(Axis : =, Execute : =, ParameterNumber : =, Value : =, Done =>, Busy =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	Run the function block.	BOOL	True/False (False)	-
ParameterNumber	Number of the parameter to be written.	DINT	Positive, negative, or 0 (0)	When *Execute* turns to True and Busy is False.
Value	Boolean value to be written to the parameter.	BOOL	True/False (False)	When *Execute* turns to True and Busy is False.

- **Outputs**

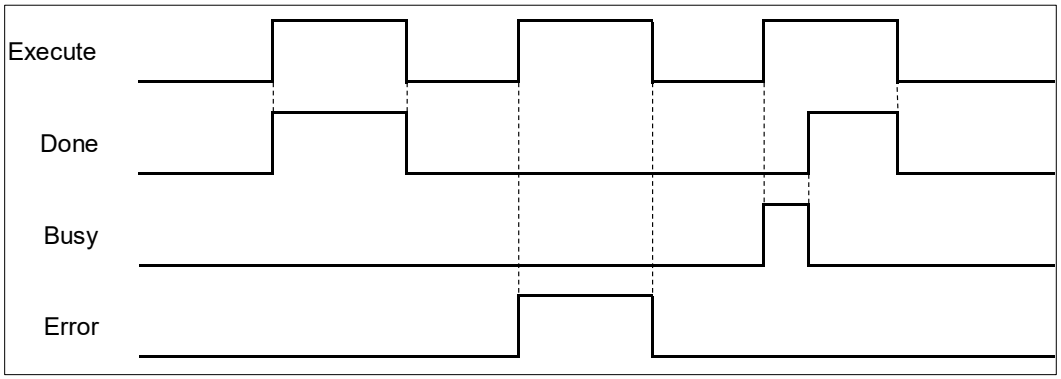
Name	Function	Data Type	Output Range (Default Value)
Done	True if the Boolean value is written successfully.	BOOL	True/False (False)
Busy	True when the function block is being run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

■ **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• When the value is written successfully.	• When *Execute* turns from True to False.
Busy	• When *Execute* is triggered to be True. • When the value is being written to the parameter.	• When *Done* turns to True. • When *Error* turns to True.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

■ **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and Busy is False.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**
  - How to use MC_WriteBoolParameter to write the parameter value to the desired object with its parameter number in EtherCAT Object Dictionary:
    - Use SHL instruction to move the data length of the desired object to the left for 24 bits.
    - Use SHL instruction to move the index of the desired object to the left for 8 bits.

    The input ParameterNumber must contains the data length of the index and the subindex. Refer to the following formula:

    ParameterNumber: = - DWORD_TO_DINT(SHL(TO_DWORD(data length of object dictionary), 24) + SHL(TO_DWORD(index of object dictionary), 8) + object sub-index);
  - To write the value to the parameter, you'll need to enter the parameter number of AXIS_REF_SM3(FB) into ParameterNumber input.
  - For operation example, refer to the example in MC_WriteParameter.

- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code)

to address the problem.

2.2.7 MC_ReadActualPosition

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_ReadActualPosition reads the current axis position.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_ReadActualPosition		MC_ReadActualPosition_instance(Axis : =, Enable : =, Valid =>, Busy =>, Error =>, ErrorID =>, Position =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	Run the function block.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Valid	True when the parameter to read exists and can be further processed.	BOOL	True/False (False)
Busy	True when the function block is being run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
Position	The current position of axis	LREAL	Positive, negative, or 0 (0)

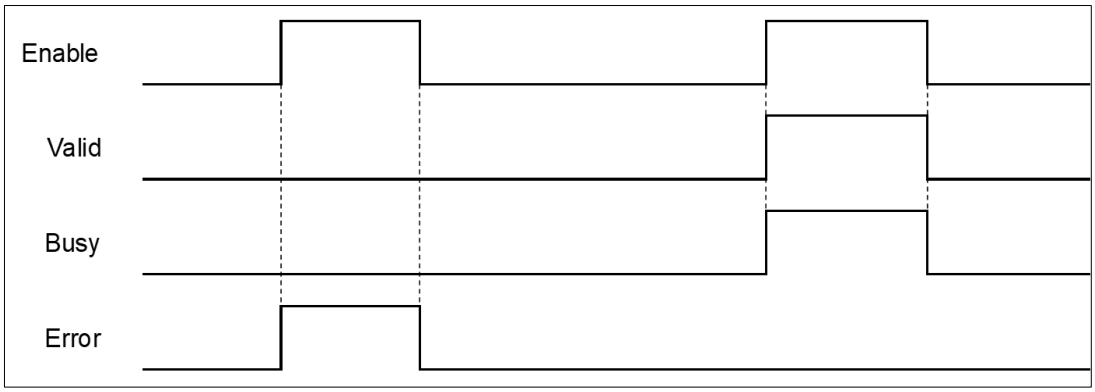
***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Valid	• When *Enable* is triggered to True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.
Busy	• When *Enable* is triggered to True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.

Name	Timing for Shifting to True	Timing for Shifting to False
Error	• When an error occurs in executing conditions or input values.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		
Position	• When *Valid* is True and there're ongoing updates.	• When *Valid* is False and stop updating.

- **Timing Diagram**



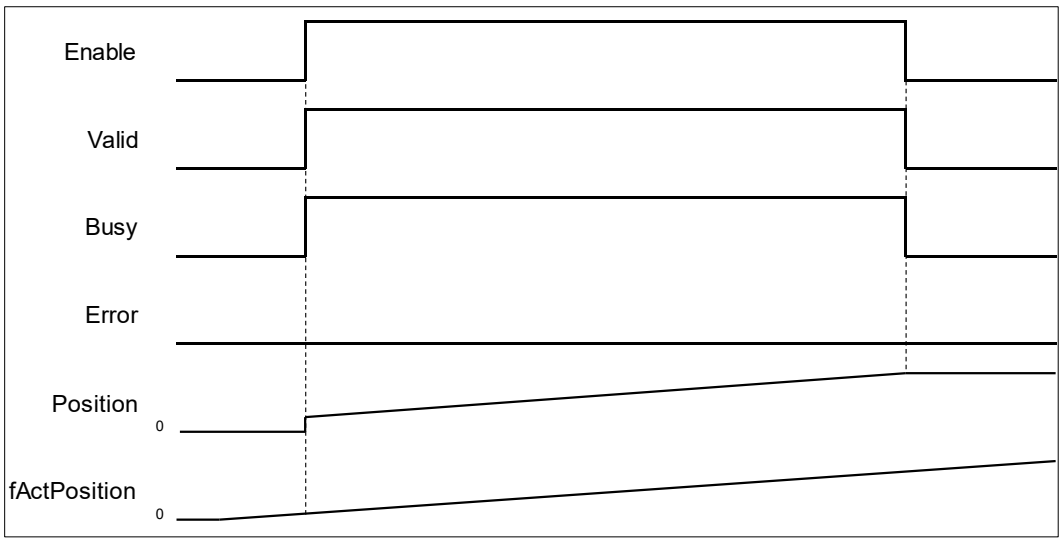
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Enable* turns to True.

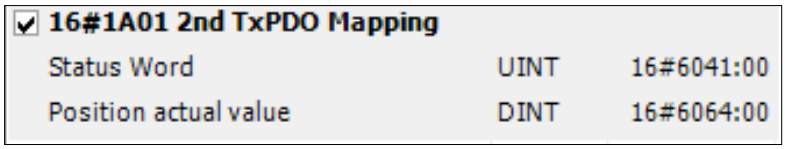
***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

  - The value read from Position of MC_ReadActualPosition is value of fActPosition in AXIS_REF_SM3.



  - While using MC_ReadActualPosition, OD 0x6064(Actual position) must be mapping to TxPDO to read the actual position of the servo. If not, the values read by the function block will be 0.



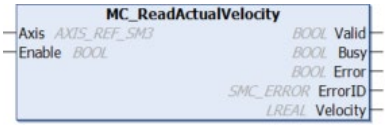
- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

2.2.8 MC_ReadActualVelocity

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_ReadActualVelocity reads the actual axis velocity value.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_ReadActual Velocity		MC_ReadActualVelocity_instance(Axis : =, Enable : =, Valid =>, Busy =>, Error =>, ErrorID =>, Velocity =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	Run the function block.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Valid	True when the parameter to read exists and can be further processed.	BOOL	True/False (False)
Busy	True when the function block is being run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
Velocity	The current velocity of axis.	LREAL	Positive, negative, or 0 (0)

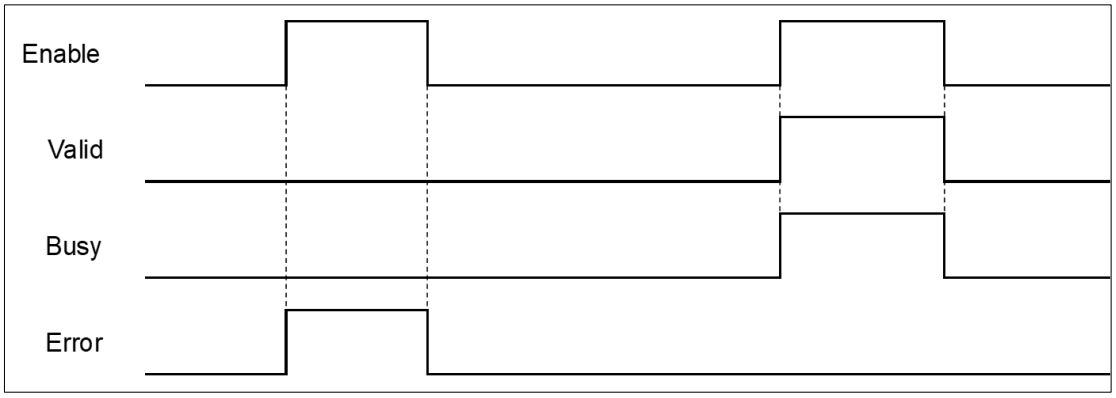
***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Valid	• When *Enable* is triggered to True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.

Name	Timing for Shifting to True	Timing for Shifting to False
Busy	• When *Enable* is triggered to True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.
Error	• When an error occurs in executing conditions or input values.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		
Velocity	• When *Valid* is True and there're ongoing updates.	• When *Valid* is False and stop updating.

- **Timing Diagram**



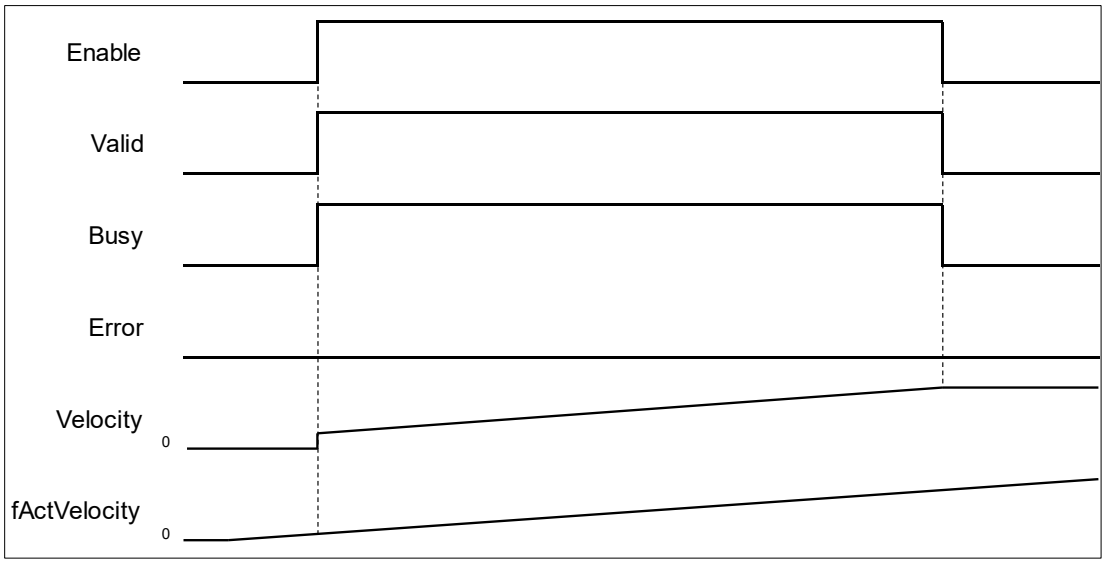
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When Enable turns to True.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

  - The velocity value read by MC_ReadActualVelocity is the value of fActVelocity in AXIS_REF_SM3.



- While using MC_ReadActualVelocity, OD 0x606C (Actual velocity) must be mapping to TxPDO so as to read the actual velocity of the servo.

16#1A02 3rd TxPDO Mapping		
Status Word	UINT	16#6041:00
Position actual value	DINT	16#6064:00
Velocity actual value	DINT	16#606C:00

- If 0x606C is not mapping to TxPDO, the actual velocity of the servo will be calculated based on OD 0x6064(Actual position).

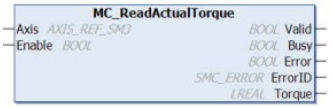
- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

2.2.9 MC_ReadActualTorque

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_ReadActualTorque reads the actual torque value of axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_ReadActualTorque		MC_ReadActualTorque_instance(Axis : =, Enable : =, Valid =>, Busy =>, Error =>, ErrorID =>, Torque =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	Run the function block.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Valid	True when the parameter to read exists and can be further processed.	BOOL	True/False (False)
Busy	True when the function block is being run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
Torque	The actual torque of axis.	LREAL	Positive or 0(0)

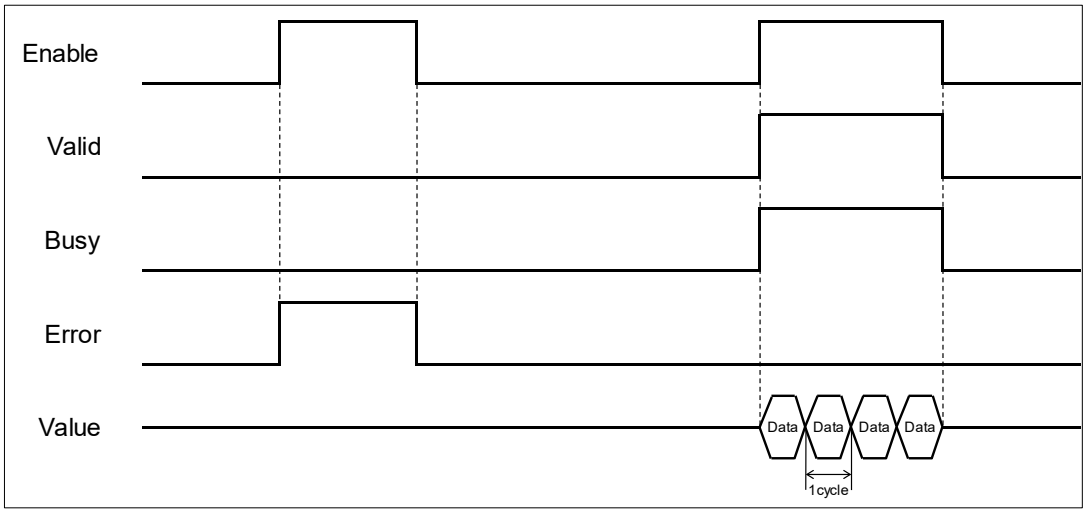
***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Valid	• When *Enable* is triggered to True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.

Name	Timing for Shifting to True	Timing for Shifting to False
Busy	• When *Enable* is triggered to True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.
Error	• When an error occurs in executing conditions or input values.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		
Torque	• When *Valid* is True and there're ongoing updates.	• When *Valid* is False and stop updating.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Enable* turns to True.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

  - The torque value read by MC_ReadActualTorque is the value of fActTorque in AXIS_REF_SM3.
  - While using MC_ReadActualTorque, OD 0x6077 (Torque actual value) must be mapping to TxPDO so as to read the actual torque of the servo.

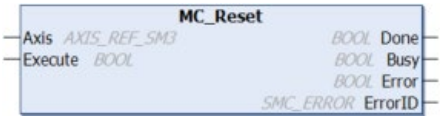
- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

2.2.10 MC_Reset

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_Reset clears axis-related errors so that the error memory is available for new error messages.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Reset		MC_Reset_instance(Axis : =, Execute : =, Done =>, Busy =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-

- **Outputs**

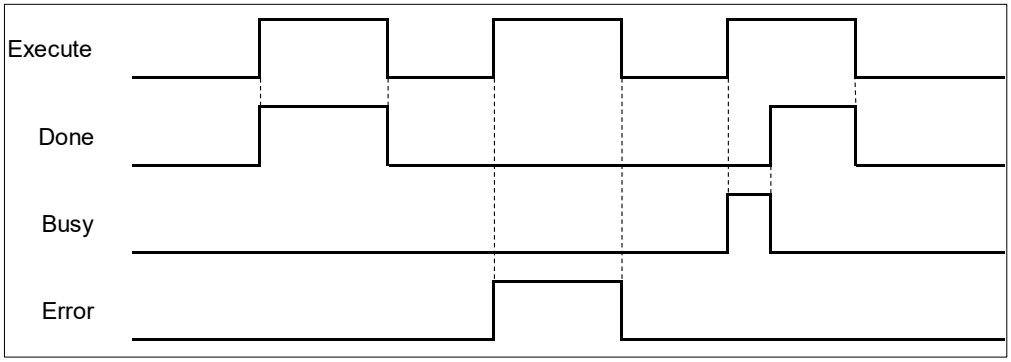
Name	Function	Data Type	Output Range (Default Value)
Done	Errors are cleared and the status changes to Standstill or Disabled.	BOOL	True/False (False)
Busy	True when the instruction is triggered to run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• When complete clearing axis-related errors.	• When *Execute* turns from True to False. • If Execute is False and Done turns to True, Done will be True for only one scan cycle and immediately shift to False.
Busy	• When *Execute* is triggered to be True.	• When *Done* turns to True. • When *Error* turns to True.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



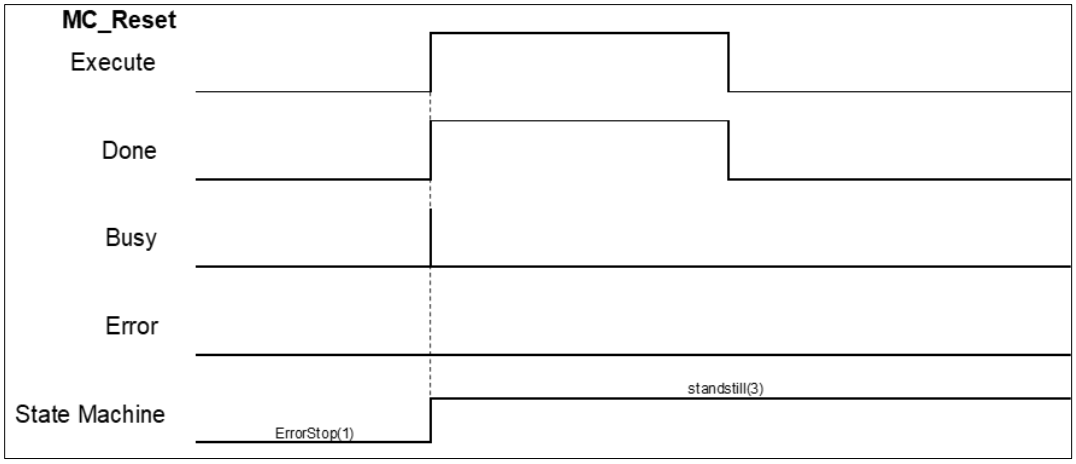
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and *Busy* is False.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

  - The function block MC_Reset can change the error status of axis back to normal. When Enable of MC_Power is True, the axis status changes from Errorstop to Standstill. When Enable of MC_Power is False, the axis status changes from Errorstop to Disabled.



  - After errors being reported by the servo controller, users can use MC_Reset to clear them and then the axis state will return to Standstill.
  - If not able to use MC_Reset to clear the axis errors, such as communication error, SMC_R_ERROR_NOT_RESETTABLE 122 (Error could not be reset.) will be reported by MC_Reset.

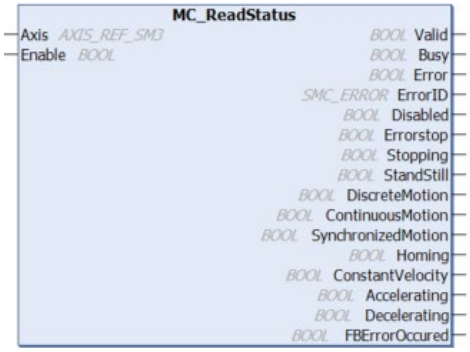
- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

2.2.11 MC_ReadStatus

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_ReadStatus reads the status of the specified axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_ReadStatus		MC_ReadStatus_instance(Axis : =, Enable : =, Valid =>, Busy =>, Error =>, ErrorID =>, Disabled=>, Errorstop=>, Stopping=>, StandStill=>, DiscreteMotion=>, ContinuousMotion=>, SynchronizedMotion=>, Homing=>, ConstantVelocity=>, Accelerating=>, Decelerating=>, FBErrorOccured=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	Run the function block.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Valid	True when the parameter to read exists and can be further processed.	BOOL	True/False (False)
Busy	True when the function block is being run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error	SMC_ERROR*1	SMC_ERROR (SMC_NO_ERROR)

Name	Function	Data Type	Output Range (Default Value)
	code descriptions.		
Disabled	refer to SMC_AXIS_STATE*2 for axis state descriptions.	BOOL	True/False (False)
Errorstop		BOOL	True/False (False)
Stopping	refer to SMC_AXIS_STATE*2 for axis state descriptions.	BOOL	True/False (False)
StandStill		BOOL	True/False (False)
DiscreteMotion		BOOL	True/False (False)
ContinuousMotion		BOOL	True/False (False)
SynchronizedMotion		BOOL	True/False (False)
Homing		BOOL	True/False (False)
ConstantVelocity	True when the axis moves at a constant speed.	BOOL	True/False (False)
Accelerating	True when the axis accelerates.	BOOL	True/False (False)
Decelerating	True when the axis decelerates.	BOOL	True/False (False)
FBErrorOccured	True when an error occurs.	BOOL	True/False (False)

***Note:**
1. SMC_ERROR: Enumeration (Enum)
2. SMC_AXIS_STATE: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Valid	• When *Enable* is triggered to be True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.
Busy	• When *Enable* is triggered to True.	• When *Enable* turns from True to False. • When *Error* is rising edge.
Error	• When an error occurs in execution conditions or input values.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		
Disabled	• When the axis is in Disabled state.	• When the axis is not in Disabled state.
Errorstop	• When the axis is in Errorstop state.	• When the axis is not in Errorstop state.
Stopping	• When the axis is in Stopping state.	• When the axis is not in Stopping state.
StandStill	• When the axis is in StandStill state.	• When the axis is not in StandStill state.
DiscreteMotion	• When the axis is in Discrete Motion state.	• When the axis is not in Discrete Motion state.
ContinuousMotion	• When the axis is in Continuous Motion state.	• When the axis is not in Continuous Motion state.
SynchronizedMotion	• When the axis is in Synchronized Motion state.	• When the axis is not in Synchronized Motion state.
Homing	• When the axis is in Homing state.	• When the axis is not in Homing state.

Name	Timing for Shifting to True	Timing for Shifting to False
ConstantVelocity	• When the axis moves at a constant speed.	• When the axis moves at a non-constant speed.
Accelerating	• When the axis moves with acceleration.	• When the axis moves without acceleration.
Decelerating	• When the axis moves with deceleration.	• When the axis moves without deceleration.
FBErrorOccured	• When errors exist.	• When errors are cleared.

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Enable* turns to True.

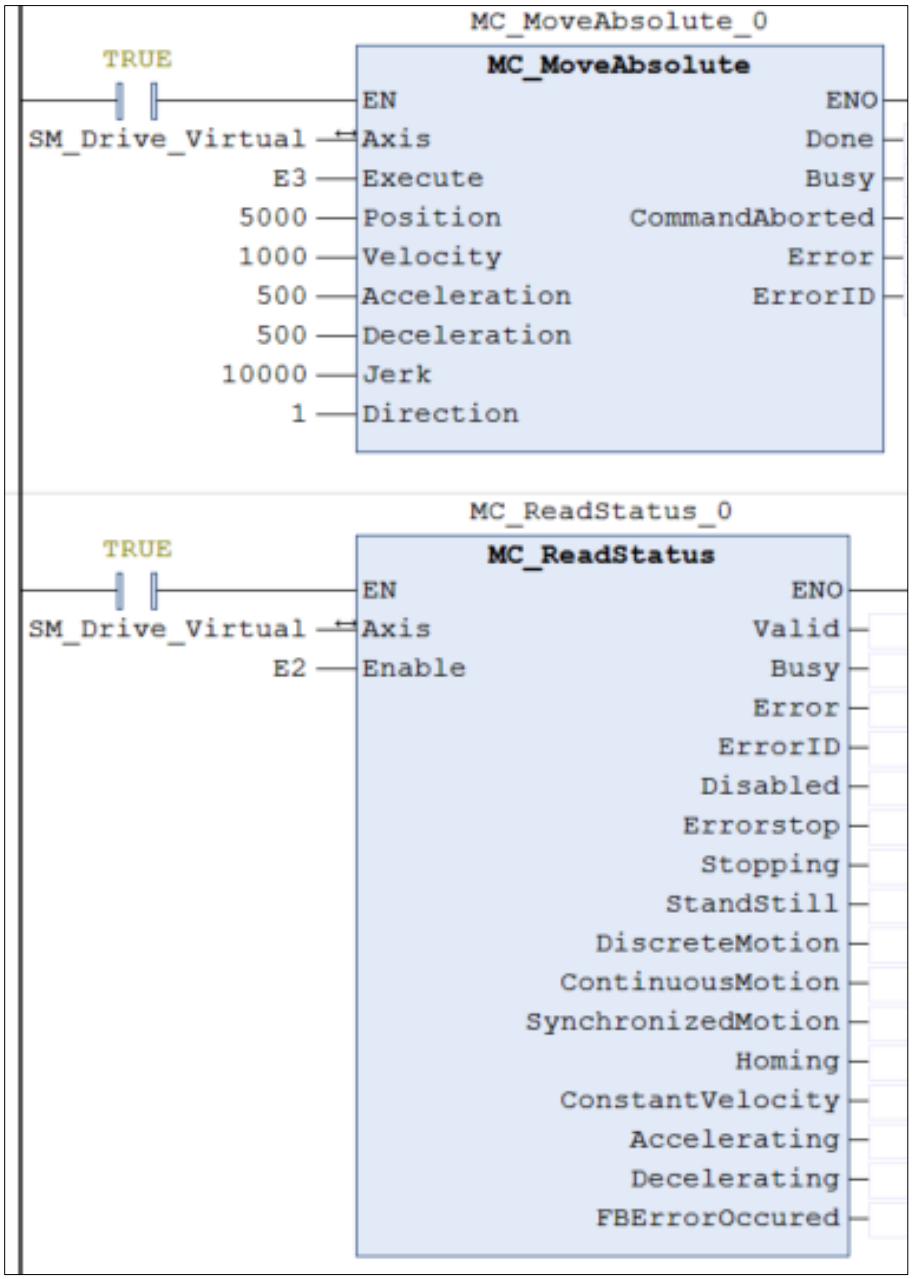
***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Troubleshooting**

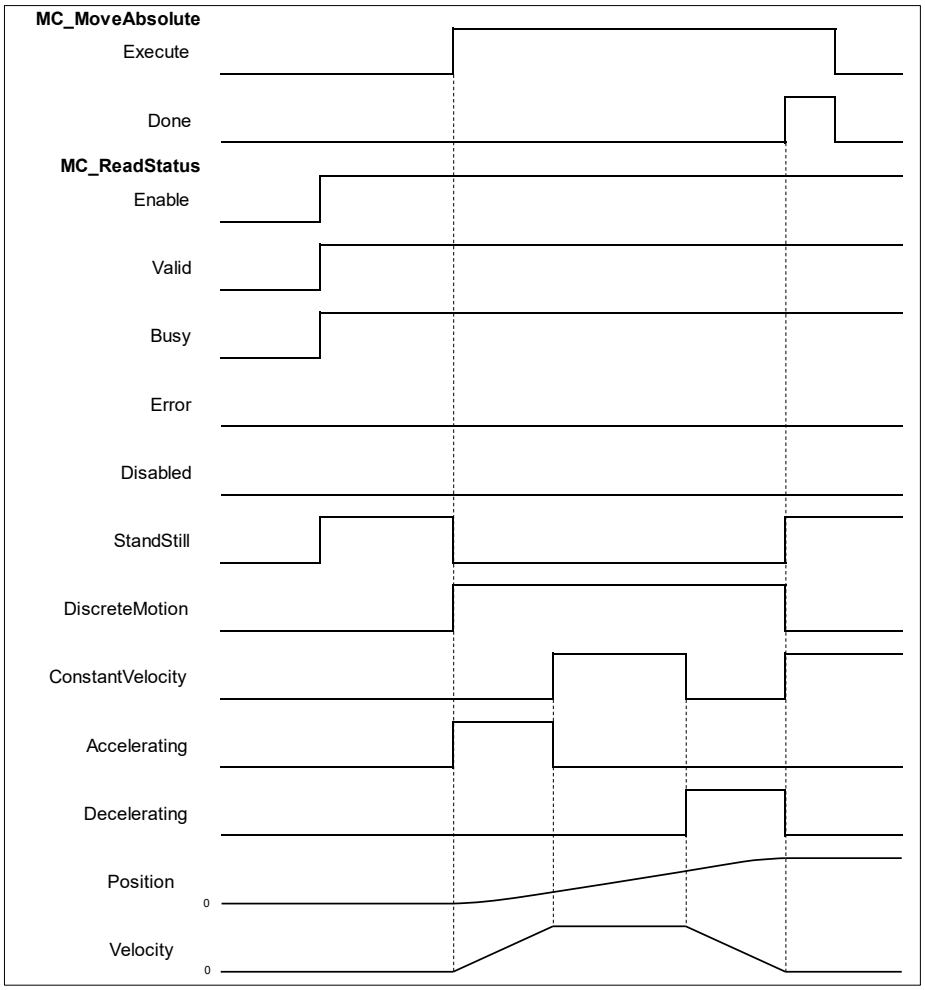
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

  - This example demonstrates using MC_ReadStatus to read axis status while executing MC_MoveAbsolute.



- **Timing Diagram**

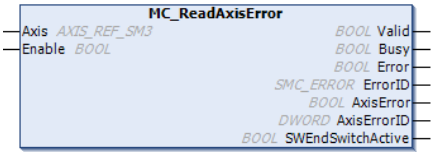


- After MC_MoveAbsolute being run, axis state turns from Standstill to Discrete_motion. At the same time, the axis begins to accelerate and Outputs*Accelerating* turns to True.
- When axis velocity reaches the setting in MC_MoveAbsolute, the axis moves at constant speed. Meanwhile, the output of ConstantVelocity turns to True and Outputs*Accelerating* turns to False. Upon moving close to the target position, the axis starts decelerating, which *Decelerating* turns to True and *ConstantVelocity* turns to False.
- Done of MC_MoveAbsolute turns to True when the target position is reached. Output status turns from *Discretemotion* to *Standstill.*

2.2.12MC_ReadAxisError

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_ReadAxisError reads the error information of axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_ReadAxisError		MC_ReadAxisError_instance(Axis : =, Enable : =, Valid =>, Busy =>, Error =>, ErrorID =>, AxisError =>, AxisErrorID =>, SWEndSwitchActive =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	Run the function block.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Valid	True when the parameter to read exists and can be further processed.	BOOL	True/False (False)
Busy	True when the function block is being run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
AxisError	True if an error occurs in the axis.	BOOL	True/False (False)
AxisErrorID	Error codes specified by the	DWORD	Positive or 0(0)

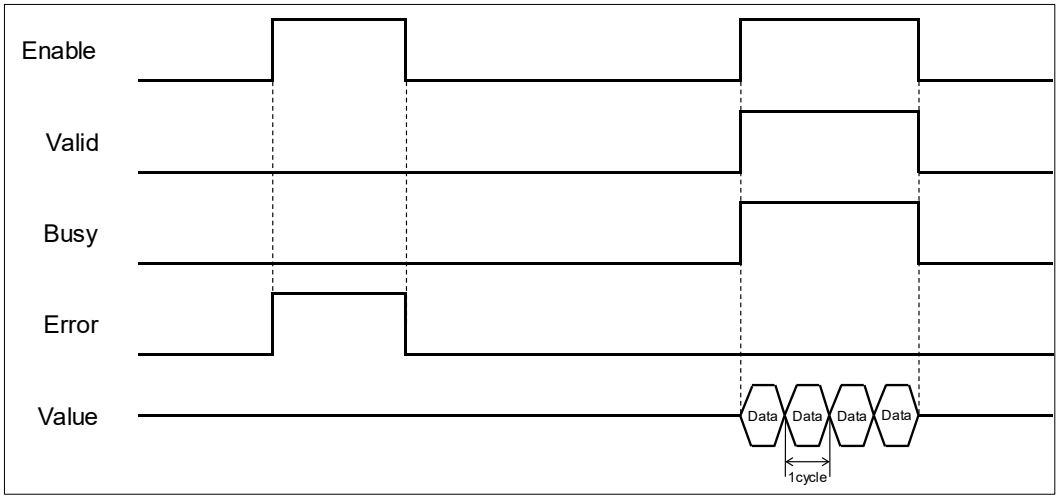
Name	Function	Data Type	Output Range (Default Value)
	vender.		
SWEndSwitchActive	True when the axis exceeds the software limit.	BOOL	True/False (False)

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Valid	• When *Enable* is triggered to True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.
Busy	• When *Enable* is triggered to True. • When the parameter to read exists.	• When *Enable* turns from True to False. • When *Error* is rising edge.
Error	• When an error occurs in executing conditions or input values.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		
AxisError	• When an error occurs in the axis.	• When the error is removed.
AxisErrorID		
SWEndSwitchActive	• When the axis exceeds the software limit.	• When runs MC_Reset.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Enable* turns to True.

***Note:** AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

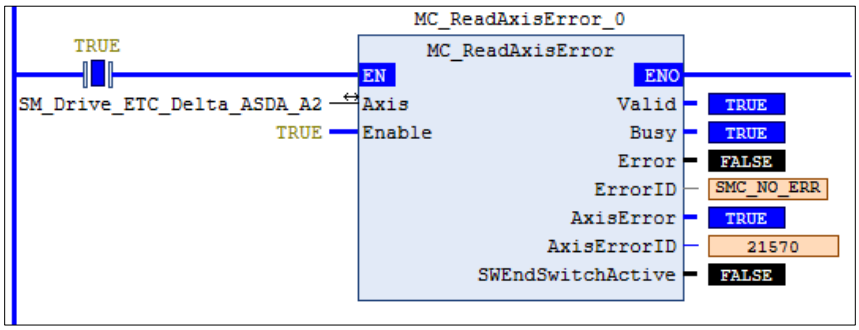
- Outputs*SWEndSwitchActive* will shift to True once the axis reaches the software limit.
- AxisErrorID displays the error codes of the servo motor itself. Take ASDA-A2-E for example, when error codes appear in the display on the servo panel, MC_ReadAxisError requests the servo for its error code by giving Error Code(0x603F) and the servo's error code will be displayed on the monitoring screen of axis.

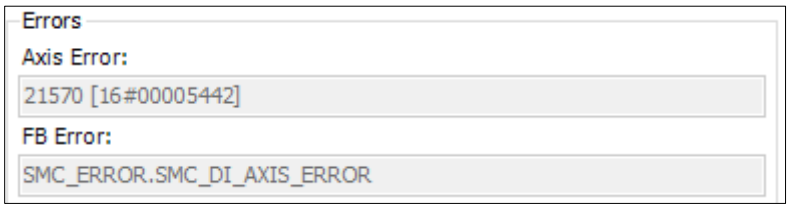
- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

  - The following example gives the status read by MC_ReadAxisError when the servo reaches hardware limit.



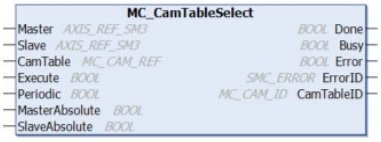


When ASDA-A2-E servo touches the positive hardware limit, "AL015" will be displayed on the servo panel. Meanwhile, use MC_ReadAxisError to read the corresponding error code. The error code for AL015 is 0x5442(refer to ASDA-A2-E user manual.) AxisErrorID is used to display the error code, which will also be displayed simultaneously on the monitoring screen of axis.

2.2.13 MC_CamTableSelect

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_CamTableSelect selects the cam table for use with MC_CamIn.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_CamTableSelect		MC_CamTableSelect_instance(Master : =, Slave : =, CamTable : =, Execute : =, Periodic : =, MasterAbsolute : =, SlaveAbsoulte : =, Done =>, Busy =>, Error =>, ErrorID =>, CamTableID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when Execute changes from False to True.	BOOL	True/False (False)	-
Periodic	Periodic mode	BOOL	True/False (True)	When *Execute* turns to True and *Busy* is False.
MasterAbsolute	MasterAbsolute mode	BOOL	True/False (True)	When *Execute* turns to True and *Busy* is False.
SlaveAbsoulte	SlaveAbsoulte mode	BOOL	True/False (True)	When *Execute* turns to True and *Busy* is False.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	True when the instruction is completed.	BOOL	True/False (False)
Busy	True when the instruction is triggered to run.	BOOL	True/False (False)

Name	Function	Data Type	Output Range (Default Value)
Error	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*1	SMC_ERROR (SMC_NO_ERROR)
CamTableID	Create CAM_ID for use by CamTableID of MC_CamIn.	MC_CAM_ID*2	MC_CAM_ID

*** Note:**

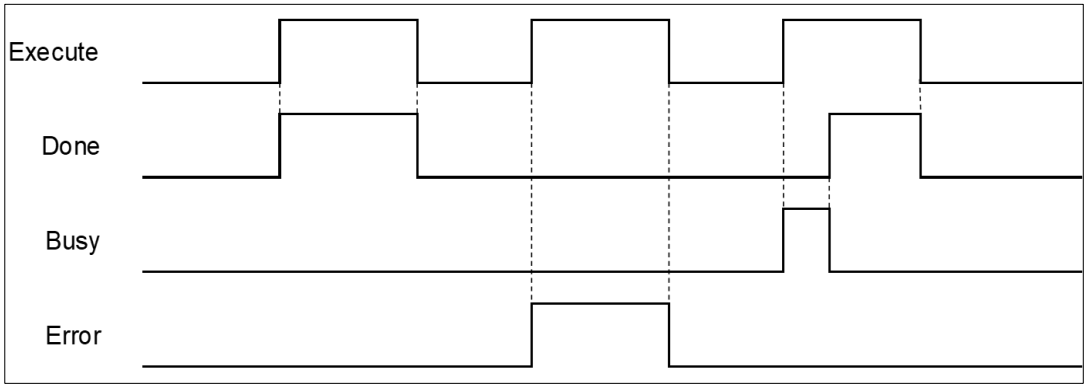
1. SMC_ERROR: Enumeration (Enum)
2. MC_CAM_ID: Structure (Struct)

Name	Function	Data Type	Output Range (Default Value)
pCT	The internal information described by the cam table.	POINTER TO BYTE	Positive or 0(0)
Periodic	Periodic mode	BOOL	True/False(True)
MasterAbsolute	MasterAbsolute mode	BOOL	True/False(True)
SlaveAbsolute	SlaveAbsolute mode	BOOL	True/False(True)
StartMaster	The master start position of the cam table.	LREAL	Positive, negative, or 0 (0)
EndMaster	The master end position of the cam table.	LREAL	Positive, negative, or 0 (0)
StartSlave	The slave start position of the cam table.	LREAL	Positive, negative, or 0 (0)
EndSlave	The slave end position of the cam table.	LREAL	Positive, negative, or 0 (0)
byCompatibilityMode	Compatibility Mode	BYTE	Positive or 0(0)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• When CamTableSelect is completed.	• When *Execute* turns from True to False. • If *Execute* is False and *Done* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Busy	• When the instruction is being run.	• When *Done* turns to True. • When *Error* turns to True.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Master	Specifies the master axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	When *Execute* turns to True and *Busy* is False.
Slave	Specifies the slave axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	When *Execute* turns to True and *Busy* is False.
CamTable	Specifies cam table.	MC_CAM_REF*2	MC_CAM_REF	When *Execute* turns to True and *Busy* is False.

***Note:**
1. AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.
2. MC_CAM_REF(FB): This data structure is used as reference to a cam table specified by users.

- **Function**

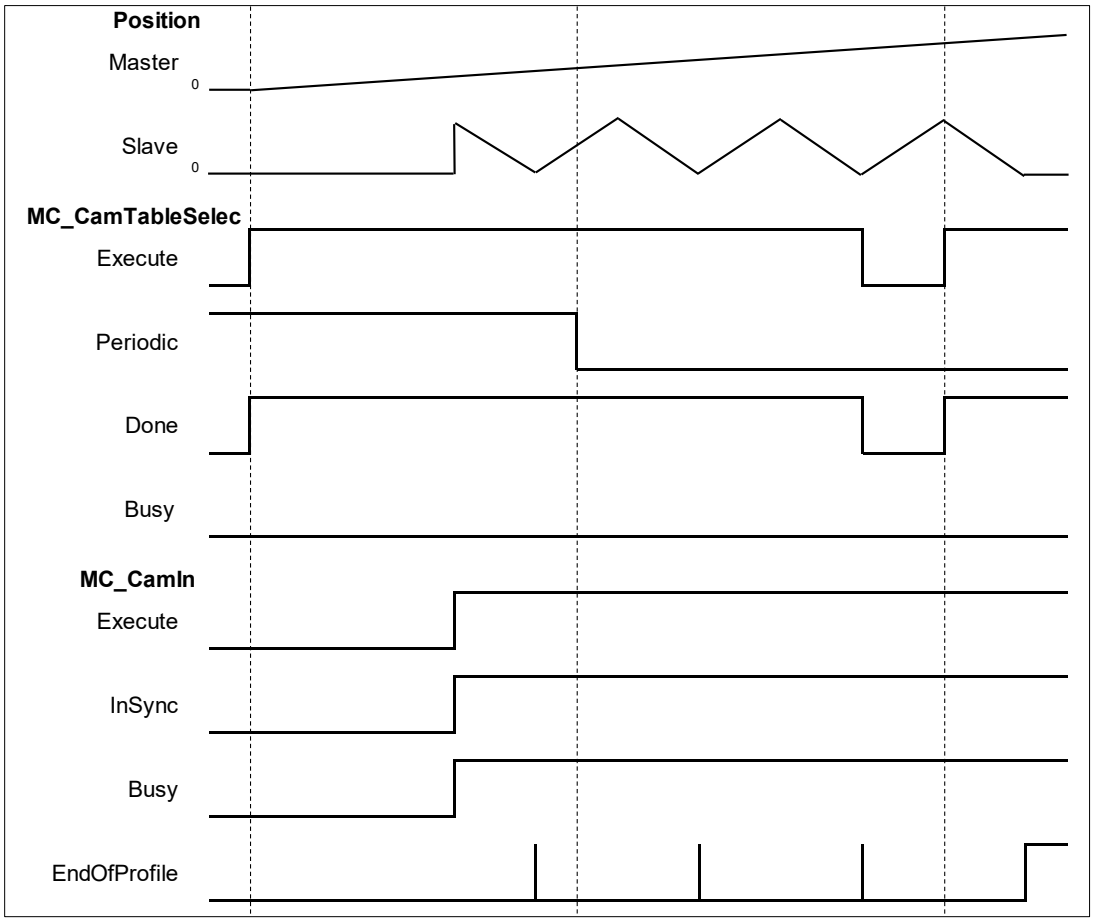
  - Use MC_CamTableSelect to select the cam table for operation.
  - Set Execute to be True so as to execute the specified or refreshed cam table. When Done turns to True, CamTableID is effective.
  - After the master-slave synchronization is completed, the modification of MC_CamTableSelect parameters can cause changes in the cam behavior.
    - After changes the variables of CamTable, the mode of cam behavior will be effective immediately.
    - The function block must be reboot after changing Periodic mode.

- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

  - The following example explains the impact on cam after changing Periodic mode.
  - Timing Diagram

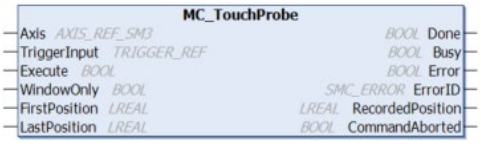


- To change the periodic mode, *Periodic* of MC_CamTable will be pulled down and shift to False, while the slave axis remains its periodicity.
- After rebooting MC_CamTable, the slave axis enters non-periodic mode. As soon the last period movement performed by the slave axis is completed, *EndOfProfile* will shift to True and remain unchanged.

2.2.14MC_TouchProbe

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_TouchProbe records an axis position at the time when a trigger event occurs.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_TouchProbe		MC_TouchProbe_instance(Axis: =, TriggerInput: =, Execute : =, WindowOnly: =, FirstPosition: =, LastPosition: =, Done =>, Busy =>, Error =>, ErrorID =>, RecordedPosition =>, CommandAborted =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when Execute changes from False to True.	BOOL	True/False (False)	-
WindowOnly	Activate the scope setting of Window.	BOOL	True/False (False)	When *Execute* turns to True and *Busy* is False.
FirstPosition	Define the start position of the Window mask. (User-defined unit)	LREAL	Negative, positive or 0(0)	When *Execute* turns to True and *Busy* is False.
LastPosition	Define the last position of the Window mask. (User-defined unit)	LREAL	Negative, positive or 0(0)	When *Execute* turns to True and *Busy* is False.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	If the trigger signal is True and the axis position has been recorded.	BOOL	True/False (False)
Busy	True when the instruction is triggered to run.	BOOL	True/False (False)
Error	True when an error occurs.	BOOL	True/False (False)

Name	Function	Data Type	Output Range (Default Value)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
RecordedPosition	Display the axis position recorded at the time of trigger signal being True.	LREAL	LREAL(0)
CommandAborted	True when the instruction is aborted by MC_AbortTrigger.	BOOL	True/False (False)

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• When the trigger signal is True and the axis position has been recorded.	• When *Execute* turns from True to False. • If *Execute* is False and *Done* turns to True, *Done* will be True for only one scan cycle and immediately shift to False.
Busy	• When *Execute* is triggered to be True.	• When *Done* turns to True. • When *Error* turns to True. • When *CommandAborted* turns to True.
Error	• When an error occurs in the execution conditions or input values for the instruction.	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		
CommandAborted	• When the function block is interrupted by MC_AbortTrigger.	• When *Execute* turns from True to False. • If *Execute* is False and *CommandAborted* turns to True, *CommandAborted* will be True for only one scan cycle and immediately shift to False.

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	-
TriggerInput	Trigger signal	TRIGGER_REF*2	TRIGGER_REF	When *Execute* turns to True and *Busy* is False.

***Note:**
1. AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.
2. TRIGGER_REF: Structure (STRUCT)

Name	Function	Data Type	Setting range (Default Value)
iTriggerNumber	Trigger channel	INT	0: Touch Probe 1, rising edge 1: Touch Probe 1, falling edge 2: Touch Probe 2, rising edge

			3: Touch Probe 2, falling edge (-1)
bFastLatching	Trigger signal	BOOL	True: Latching is done in drive False: Latching is done in motion controller (True)
bInput	Trigger signal when bFastLatching=FALSE	BOOL	Trigger signal
bActive	Validity of trigger signal	BOOL	True: Valid (False)

*Note: bActive is the output. Do not input signal.

- **Function**

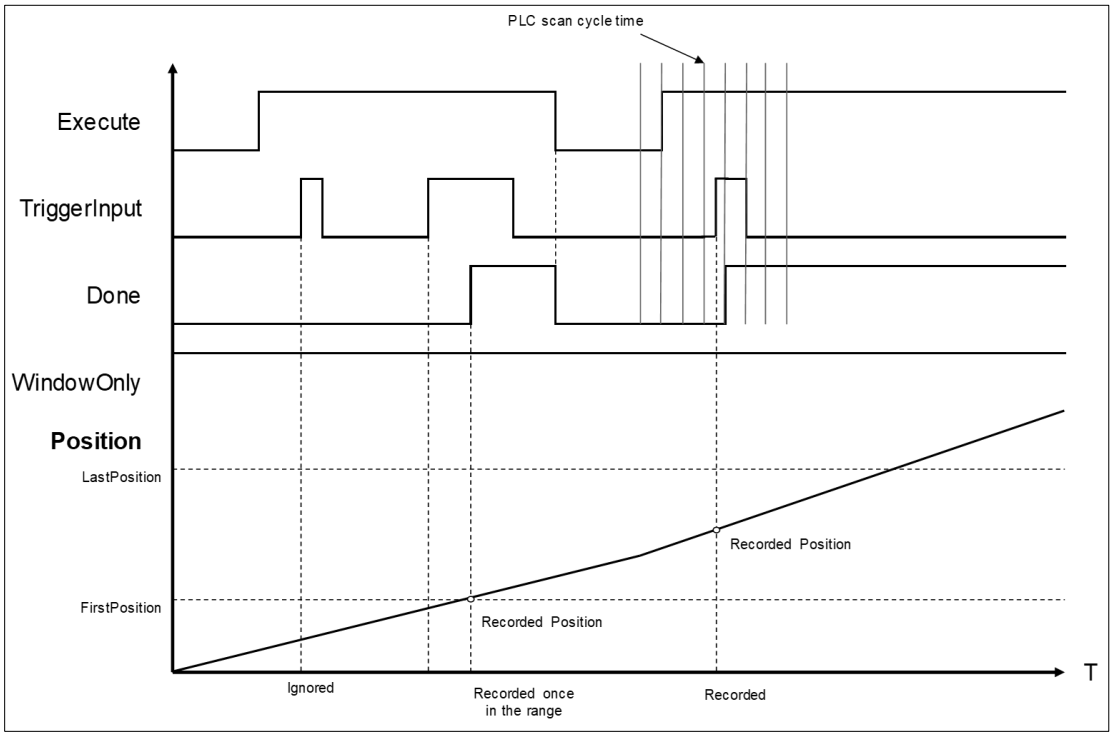
  - Drive mode
    - While using real axes, *bFastLatching* must be set to True (latching in drive) and configure *iTriggerNumber.* (The Setting Value cannot be set as default "-1" or there will be an error in the function block.)

      When *Execute* is True, the function block writes values to 0x60B8(Touch Probe Function) based on the setting of *iTriggerNumber* to open the corresponding Trigger channel.
    - If Inputs*Execute* of MC_TouchProbe is True, only the first position value of trigger signal will be captured and the following signal will be ignored, even when bit1 of 0x60B8 is set to 1 to create multiple triggers.
    - Under drive mode, RecordedPosition reads the values in 0x60BA (Position value positive edge) and then convert with the gear ratio.

  - Controller mode

    - bFastLatching must be set to False and the trigger signal changes to be controlled by bInput.
    - RecordedPosition records the command position and the current command position when bInput triggers signals successfully.

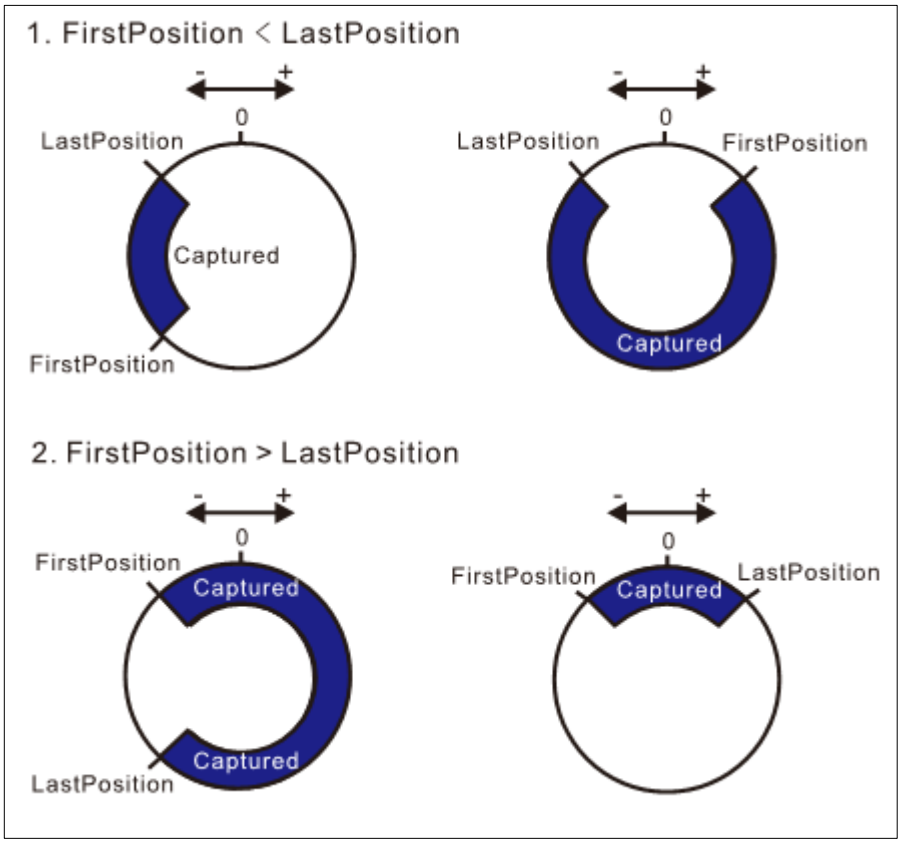
  - The operation of MC_TouhcProbe with window mask function is demonstrated as below:



- At the first activation of the trigger input signal, the signal is not accepted because the axis position hasn't reach the specified window mask section.
- When the axis position enters the window mask section, the second activation of the trigger input signal is accepted, and after a period Done changes to True.
- Time is needed until the touch probe operation is actually activated. The touch probe operation is not possibly to be activated immediately after WindowOnly turns to True.
- If the window mask is too small, the touch probe operation is not possible. The effective range for the window mask depends on EtherCAT communications and the performance of encoder input or the servo drive.
- In case that the servo drive does not support the window mask function, an error of SMC_TP_COULDNT_SET_WINDOW(401) will be reported by the function block. (Delta ASDA-A2-E has not yet supported *WindowsOnly* function.)

- Window Mask setting

  - You can observe the results of different window mask settings when the instruction is used for rotary/modulo axes as below. The difference is resulted from the set values between FirstPosition and LastPosition.

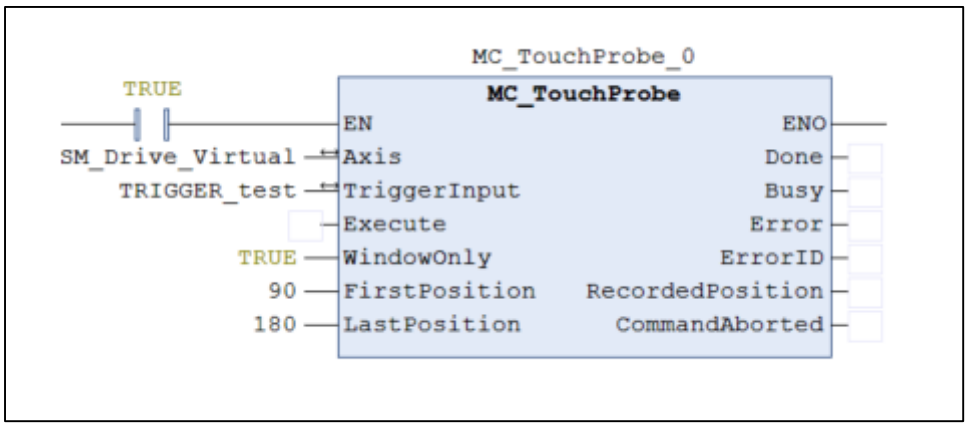


- Troubleshooting

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

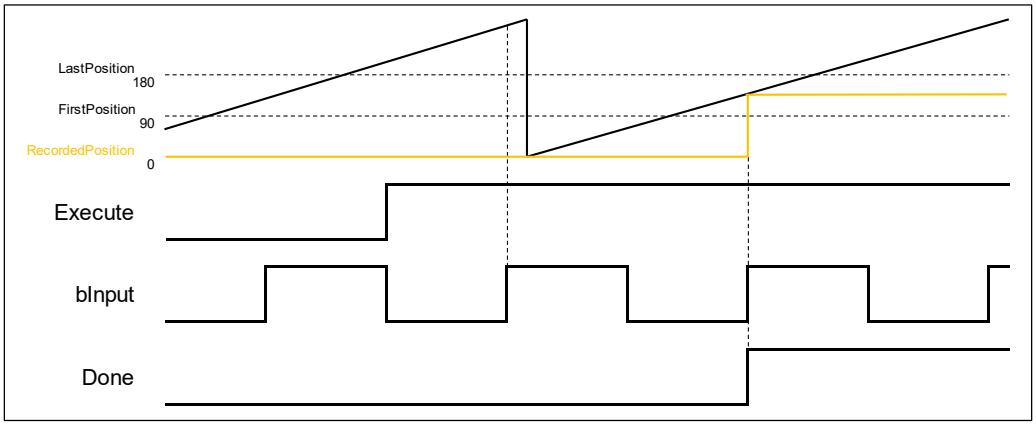
- Example

  - Example1: demonstrate the operation result of using MC_TouchProbe under controller mode.



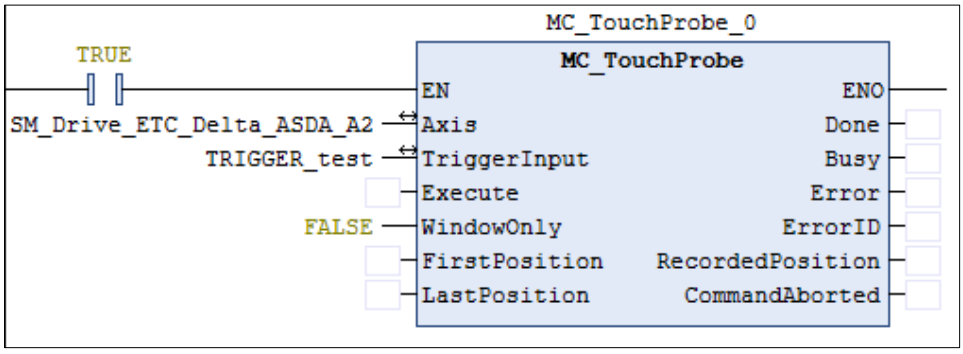

```
TRIGGER_test.bFastLatching := FALSE;
```

- Timing Diagram



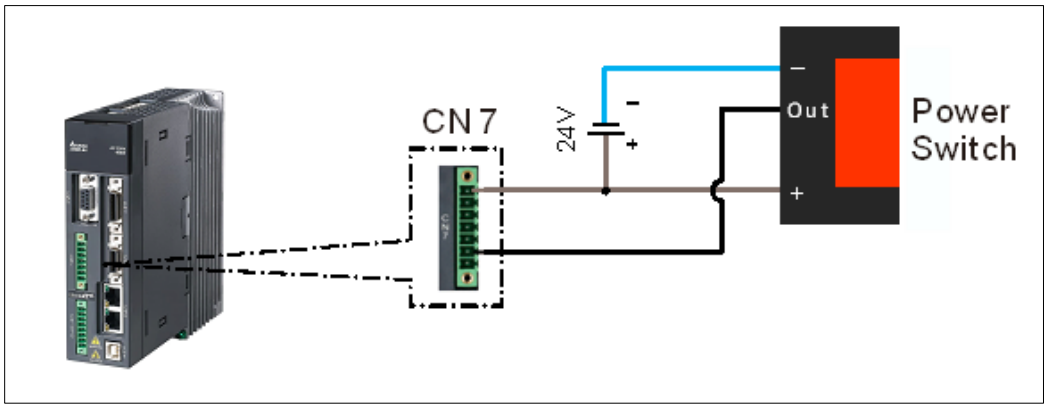
  - When Execute of MC_TouchProbe changes to True, it starts to capture the signal. Under controller mode, bInput will be the trigger signal.
  - At the first activation of the trigger input signal, the axis position is not recorded because the axis position hasn't reached the specified window mask section. The axis has entered window mask section when trigger for the second time, therefore the position will be recorded in the output RecordedPosition.

- Example 2: Explain how MC_TouchProbe takes the drive as the trigger signal, which demonstrates with ASDA-A2-E as the drive.

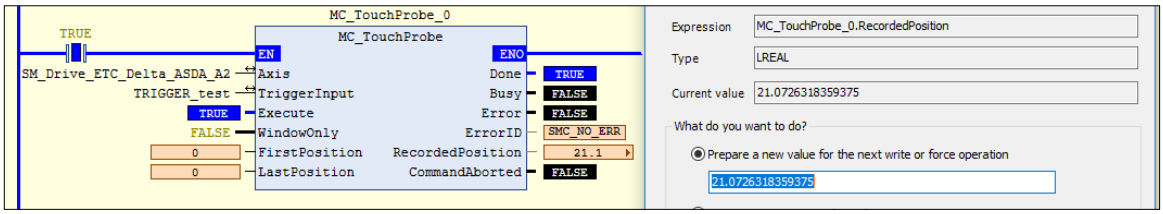



```
TRIGGER_test.iTriggerNumber := 0;
```

- Wiring diagram

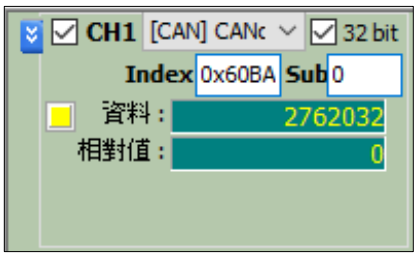


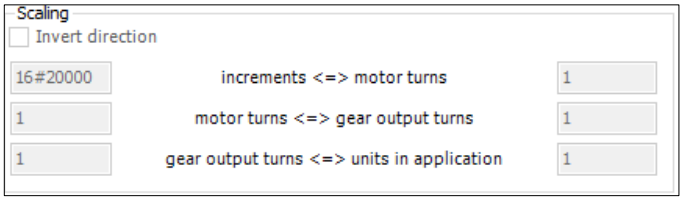
- Trigger signal is from DI13 of CNY extension DI connector. You can start the configuration with the diagram above.
- Trigger channel must be specified by the function block. The following example demonstrates with rising edge trigger.



aCaptDesc	ARRAY [0..7] OF SMC3_CaptureDescription	
aCaptDesc[0]	SMC3_CaptureDescription	
fCaptPosition	LREAL	21.0726318359375
bCaptureOccured	BOOL	FALSE
bStartCapturing	BOOL	FALSE
bAbortTrigger	BOOL	FALSE
fFirstCapturePosition	LREAL	0
fLastCapturePosition	LREAL	0
bCaptureWindowActive	BOOL	FALSE
bLatchInController	BOOL	FALSE

- When the signal on DI13 of the servo is triggered, MC_TouchProbeOutputsDone will be True. At the same time, MC_TouchProbe reads the value stored in the object 0x60BA(Touch Probe Pos1 Pos Value). After being converted with the gear ratio, the value will be stored in the axis parameter fCaptPosition, which will be output by RecordedPosition.





- As a result of the gear ratio being set to 0x20000: 1, when the drive is rising edge triggered, the value in 0x60BA must be divided with 0x20000. The signal is triggered by the 2762032 index pulses; therefore, the position is recorded at 21.0726318359375(2762032 / 131072).

2.2.15MC_AbortTrigger

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_AbortTrigger aborts the instruction MC_TouchProbe which are intended to capture trigger events.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_AbortTrigger		MC_AbortTrigger_instance(Axis : =, TriggerInput : =, Execute: =, Done =>, Busy =>, Error =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (True)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	True when trigger event is aborted	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
Error	True if an error occurs	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

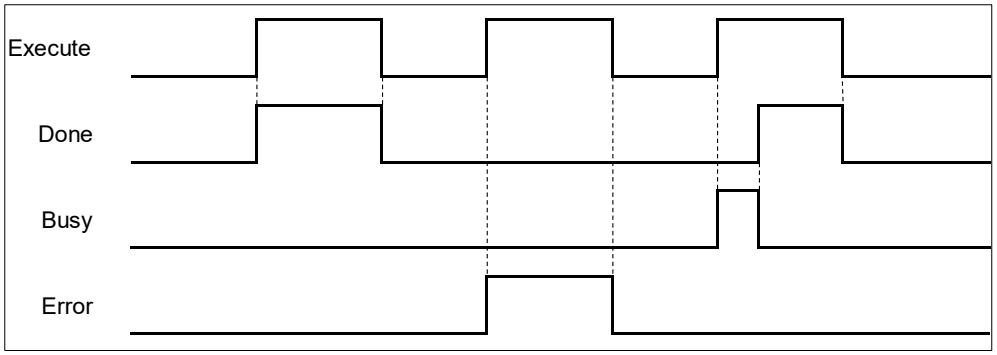
***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
Done	• When the capture operation is stopped.	• When *Execute* turns from True to False. • If *Execute* is False and *Done* turns to True, it will be True for only one scan cycle and immediately shift to False.
Busy	• When *Execute* turns to True.	• When *Done* turns to True. • When *Error* turns to True.
Error	• When an error occurs in the execution	• When *Execute* turns from True to False.

Name	Timing for Shifting to True	Timing for Shifting to False
ErrorID	conditions or input values for the instruction.(Error code is recorded)	• (Error Code is cleared)

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	When *Execute* turns to True and *Busy* is False.
TriggerInput	Specifies the reference to the source of the trigger signal.	TRIGGER_REF*2	TRIGGER_REF	When *Execute* turns to True and *Busy* is False.

***Note:**
1. AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function blocks.
2. TRIGGER_REF: Structure(STRUCT).

Name	Function	Data Type	Setting Value (Default Value)
iTriggerNumber	Trigger channel	INT	0: Touch Probe 1, rising edge 1: Touch Probe 1, falling edge 2: Touch Probe 2, rising edge 3: Touch Probe 2, falling edge (-1)
bFastLatching	Trigger signal	BOOL	True: Latching is done in drive False: Latching is done in motion controller (True)
bInput	Trigger signal when bFastLatching=FALSE	BOOL	Trigger signal
bActive	Validity of trigger signal	BOOL	True: Valid (False)

***Note**: bActive is the output, do not input signal.

- **Function**

  - You can cancel the touch probe operation by using MC_AbortTrigger.

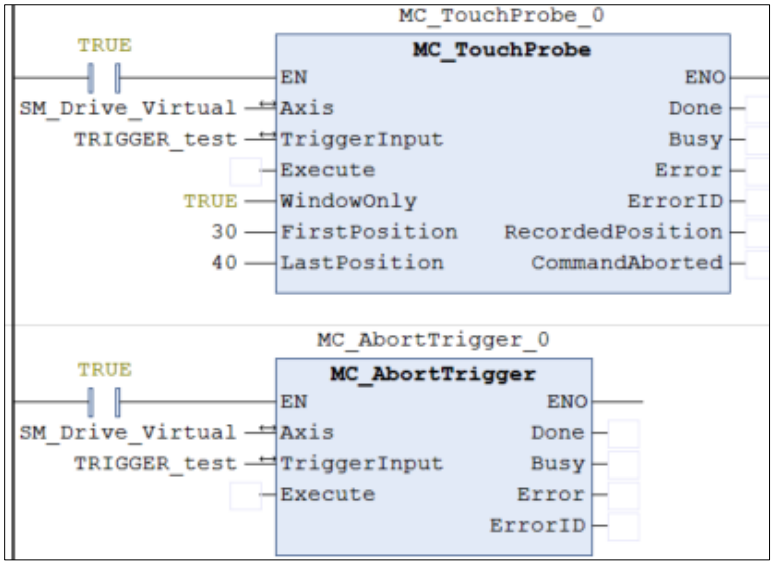
- By setting Axis and TriggerInput for this instruction you can define the touch probe operation to abort.

- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

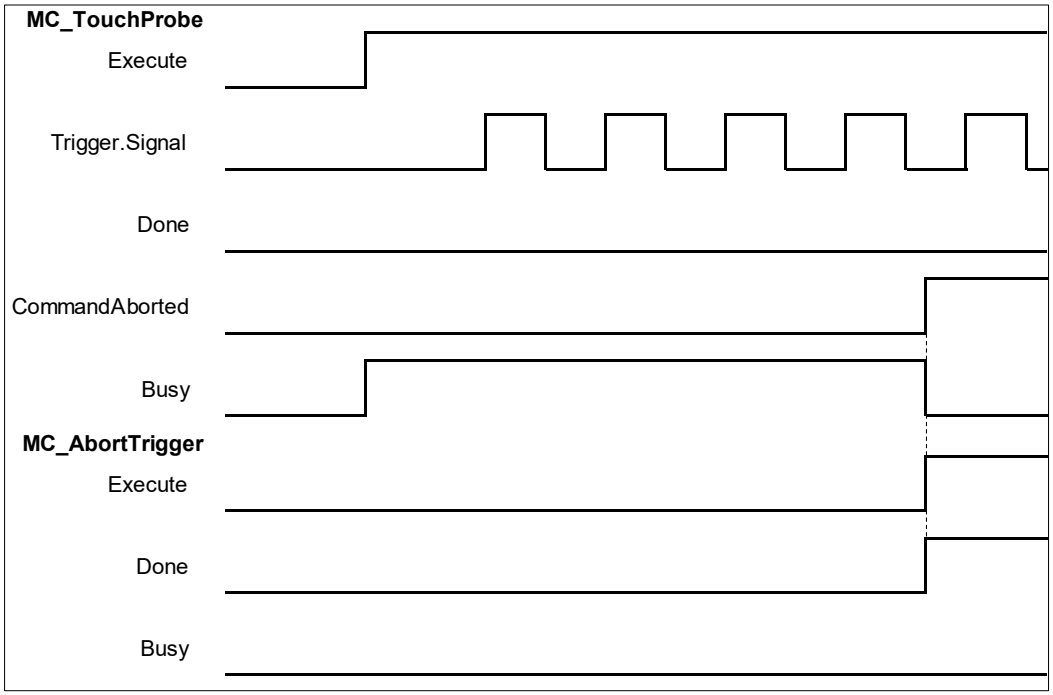
- **Example**

  - This example demonstrates the relating operation of the combination of MC_AbortTrigger and MC_TouchProbe.




```
TRIGGER_test.bFastLatching := FALSE;
```

  - Timing Diagram



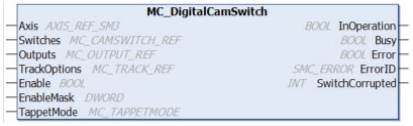
    - When a rising edge is detected on *Execute* of MC_AbortTrigger, *CommandAborted* of MC_TouchProbe turns to True.
    - If a rising edge is detected on *Execute* of MC_AbortTrigger when *Done* of MC_TouchProbe turns to True, an error

of SMC_AT_TRIGGERNOTOCCUPIED (410) will be reported by MC_AbortTrigger.

2.2.16MC_DigitalCamSwitch

- **Supported Devices**: AX-308E, AX-8, AX-364E

MC_DigitalCamSwitch uses the axis position to control a switch of a digital output.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_DigitalCam Switch		MC_DigitalCamSwitch_instance(Axis : =, Switches : =, Outputs: =, TrackOptions: =, Enable: =, EnableMask: =, TappetMode: =, InOperation =>, Busy =>, Error =>, ErrorID =>, SwitchCorrupted =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	The instruction is run when *Execute* changes from False to True.	BOOL	True/False (False)	-
EnableMask	Enable the different tracks.	DWORD	Positive or 0(16#FFFFFFFF)	When *Enable* turns to True.
TappetMode	Define the positions for the position-defined calculation of the tappets.	MC_TAPPETMODE*	0: tp_mode_auto 1: tp_mode_demandposition 2: tp_mode_actualposition (tp_mode_auto)	When *Enable* turns to True.

***Note:** MC_TAPPETMODE: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
InOperation	True when the track and instruction is activated.	BOOL	True/False (False)
Busy	True when the instruction is run.	BOOL	True/False (False)
Error	True if an error occurs	BOOL	True/False (False)

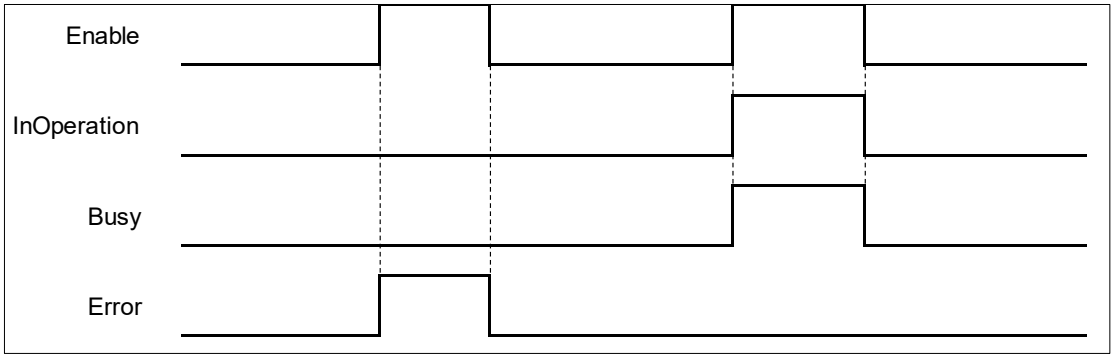
Name	Function	Data Type	Output Range (Default Value)
ErrorID	Indicates the error code when the error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
SwitchCorrupted	When the switch action is operated abnormally, the output value will not be -1.	INT	Positive, negative or 0 (-1)

***Note:** SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
InOperation	• When the track and instruction is activated.	• When *Enable* turns to False.
Busy	• When *Execute* turns to True.	• When *Enable* turns to False. • When *Error* turns to True.
Error	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	• When *Enable* turns to True.
Switches	Switch-related parameters.	MC_CAMSWITCH_REF*2	MC_CAMSWITCH_REF	• When *Enable* turns to True.
Outputs	Output signals of track	MC_OUTPUT_REF	ARRAY [1..32] OF BOOL(False)	• When *Enable* turns to True.
TrackOptions	Compensation and Hysteresis parameters for the cam track.	MC_TRACK_REF	ARRAY [1..32] OF MC_TRACK_TR*3	• When *Enable* turns to True.

***Note:**

1. AXIS_REF_SM3(FB): Every function block contains this variable, which works as the starting program for function

blocks.

2. MC_CAMSWITCH_REF: Structure (STRUCT)

Name	Function	Data Type	Setting Value (Default Value)
NoOfSwitches	Specify the number of switches.	BYTE	Positive or 0(0)
CamSwitchPtr	Points to the first element of the MC_CAMSWITCH_TR array	POINTER TO MC_CAMSWITCH_TR*	POINTER TO MC_CAMSWITCH_TR(0)

***Note:** MC_CAMSWITCH_TR: Structure (STRUCT)

Name	Function	Data Type	Setting Value (Default Value)
TrackNumber	Specify the track for the operation of tappets.	INT	Positive, negative, or 0 (0)
FirstOnPosition	Switch ON when the axis passes.	LREAL	POINTER TO MC_CAMSWITCH_TR(0)
LastOnPosition	Switch OFF when the axis passes.	LREAL	Positive, negative, or 0 (0)
AxisDirection	The switch is active only when the axis is moving in the specified direction.	INT	Positive, negative, or 0 (0)
CamSwitchMode	Switch mode	INT	Positive, negative, or 0 (0)
Duration	How long the switch is on.	TIME	Positive or 0(0)
bOn	Internal variables	BOOL	True/False (False)
CounterOff	Internal variables	INT	Positive or 0(0)

3. MC_TRACK_TR: Stucture(STRUCT).

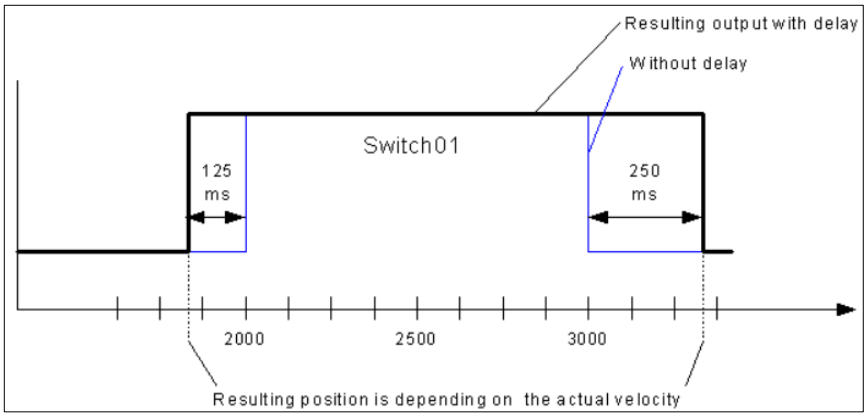
Name	Function	Data Type	Setting Value (Default Value)
OnCompensation	Compensation time with which the switch is turned on. (Unit: Sec.)	LREAL	Positive, negative, or 0 (0)
OffCompensation	Compensation time with which the switch is turned off. (Unit: Sec.)	LREAL	Positive, negative, or 0 (0)
Hysteresis	Hysteresis interval	LREAL	Positive, negative, or 0 (0)

- **Function**

  - EnableMask is a 32 bits of bool type parameter, used to enable different tracks. With the concept of the least significant bit representing the first track, the input value will be 16#FFFFFFFB to disable the third track.
  - MC_CAMSWITCH_REF defines switches for digital cam. NoOfSwitches calculates the number of switching positions. CamSwitchPtr is a pointer on an array of type MC_CAMSWITCH_TR.

- MC_CAMSWITCH_TR specifies the positions of tappets.
  - TrackNumber specifies the output number.
  - FirstOnPosition specifies the switch-on position of the output.
  - LastOnPosition specifies the switch-off position of the output (when CamSwitchMode = 0).
  - AxisDirection = 0: Output is switched in both directions. AxisDirection = 1: Only positive direction. AxisDirection = 2: Only negative direction.
  - Switch is OFF at LastOnPosition when CamSwitchMode = 0. Switch remains ON for a time set (Duration) and then changes to OFF when CamSwitchMode = 1.
  - Duration: Period of time for which the tappet output stays TRUE in case of CamSwitchMode = 1.

- MC_TRACK_REF is the Structure for managing the tracks, which contains OnCompensation, OffCompensation and Hysteresis.
  - OnCompensation is set for the delay of switch-on. If the input value is positive, switching to ON will be delayed, while an early switch-on can be set with a negative input value. The time is given in seconds. For example, if OnCompensation is set to 0.01, switching to ON will be delayed for 0.1 second.
  - OffCompensation is set for the delay of switch-off. If the input value is positive, switching to OFF will be delayed, while an early switch-off can be set with a negative input value.



  - The interval for Hysteresis is set to avoid switching errors and the specified axis position must exceeds the interval, so the switch will continue with the next action. The unit of Hysteresis is user-defined.
- Output will be switched to ON with all AxisDirection settings as long as the axis position is inside of the range.
- Multiple switch modes are allowed to be set in a single Track.

- **Troubleshooting**

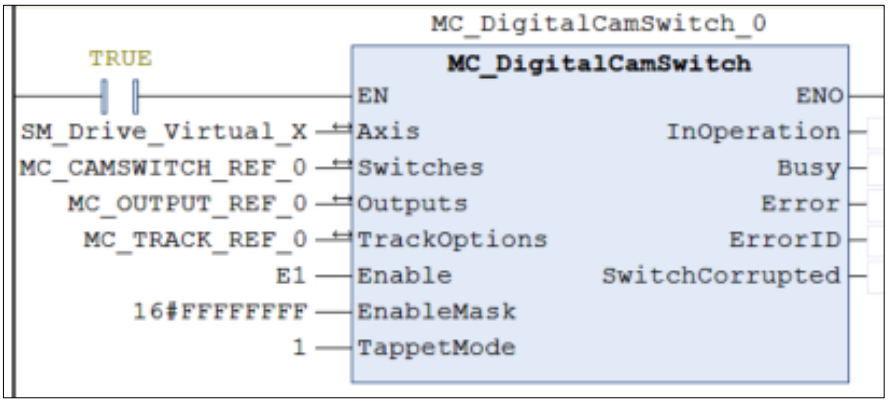
  - If an error occurs during the execution of the instruction, *Error* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Example**

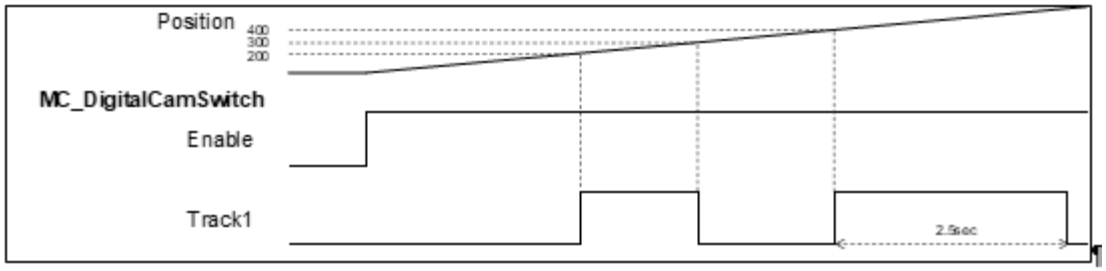
  - Example1: Demonstrates using 2 switches in the same Track in the following example.

Parameter	Type	Switch1	Switch2
TrackNumber	INT	1	1
FirstOnPosition [u]	REAL	200	400
LastOnPosition [u]	REAL	300	-

Parameter	Type	Switch1	Switch2
AxisDirection	INT	0=Both	0=Both
CamSwitchMode	INT	0=Position	1=TIME
Duration	TIME	-	2500ms



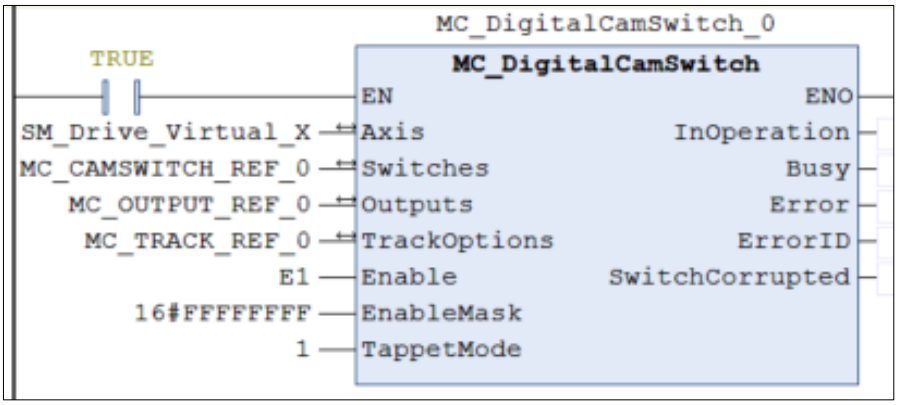
- Timing Diagram



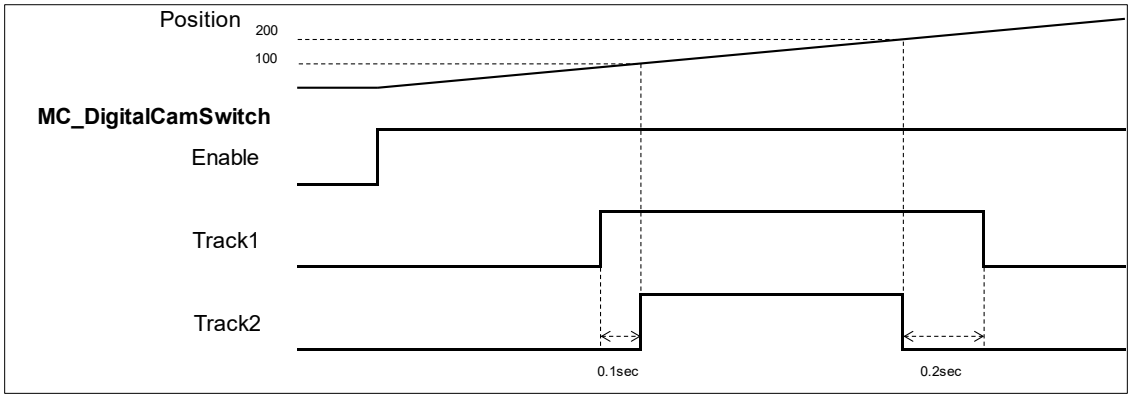
  - When the axis reaches the position 200, Switch1 on Track1 will be turned ON till the axis reaches the position 300, then changes to OFF.
  - Switch1 will be turned ON again when the position 400 is reached, and lasting for 2.5 seconds, then changes to OFF.

- Example 2: The operation result of OnCompensation/OffCompensation is given in the following example.

Parameter	Type	Switch1	Switch2
TrackNumber	INT	1	2
FirstOnPosition [u]	REAL	100	100
LastOnPosition [u]	REAL	200	200
AxisDirection	INT	0=Both	0=Both
CamSwitchMode	INT	0=Position	0=Position
Duration	TIME	-	-
OnCompensation	LREAL	- 0.1	0
OffCompensation	LREAL	0.2	0



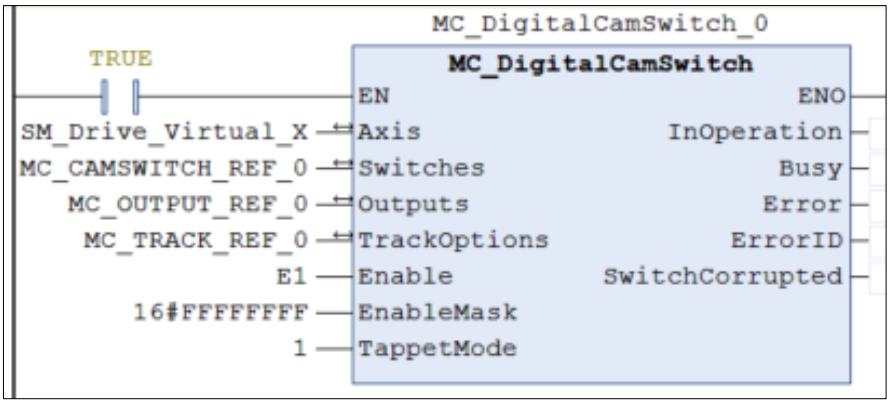
- Timing Diagram



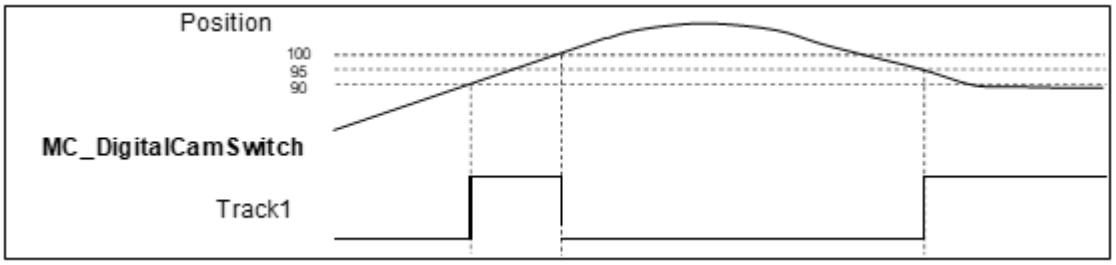
  - Once the position 100 is reached, Switch1 on Track1 and Switch2 on Track2 are both turned ON and will be turned OFF when position 200 is reached. The switch-on of Switch 1 is advanced for 0.1 second while OnCompensation = -0.1. By setting 0.2 to OffCompensation. Switch 1 is delayed for 0.2 second.

- Example 3: The operation result of *Hysteresis* is given in the following example.

Parameter	Type	Switch1
TrackNumber	INT	1
FirstOnPosition [u]	REAL	90
LastOnPosition [u]	REAL	95
AxisDirection	INT	0=Both
CamSwitchMode	INT	0=Position
Duration	TIME	-
Hysteresis	LREAL	10



- Timing Diagram

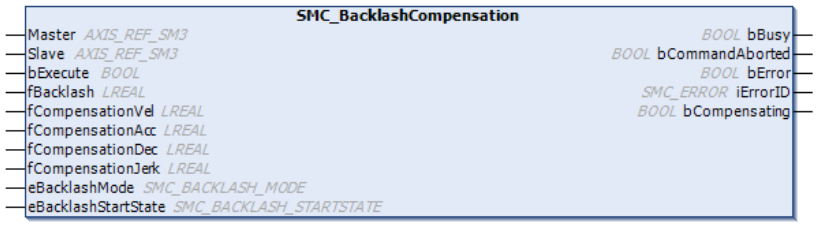


  - The FirstOnPosition and LastOnPosition of Switch 1 on Track1 are set to 90 and 95 respectively with Hysteresis set to 10, which means the switch will be turned off after the axis position passing the interval (80~100).
  - Track 1 is switched to ON when the axis reaches position 90 and not able to be switched to OFF at position 95 until the axis passes the hysteresis interval.
  - When the axis moves reversely to position 95, the switch will be turned ON again and remains, for the reason that the axis position stays within the hysteresis interval (105~85).

2.2.17 SMC_BacklashCompensation

- **Supported Devices**: AX-308E, AX-8, AX-364E

SMC_BacklashCompensation is used to compensate for the backlash of gears.

FB/FC	Instruction	Graphic Expression
FB	SMC_BacklashCompensation	

ST Language

```
SMC_BacklashCompensation_instance(
Master : =,
Slave : =,
bExecute : =,
fBacklash : =,
fCompensationVel : =,
fCompensationAcc : =,
fCompensationDec : =,
fCompensationJerk : =,
eBacklashMode : =,
eBacklashStartState : =,
bBusy =>,
bCommandAborted =>,
bError =>,
iErrorID =>,
bCompensating => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when bExecute changes from False to True.	BOOL	True/False (False)	-
fBacklash	The backlash distance to be compensated	LREAL	Positive, negative, or 0	When *bExecute* changes from False to True
fCompensationVel	The speed when compensating for backlash	LREAL	Positive or 0	When *bExecute* changes from False to True
fCompensationAcc	The acceleration when compensating for backlash	LREAL	Positive or 0	When *bExecute* changes from False to True
fCompensationDec	The deceleration when compensating for backlash	LREAL	Positive or 0	When *bExecute* changes from False to True
fCompensationJerk	The jerk when	LREAL	Positive or 0	When *bExecute*

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
	compensating for backlash			changes from False to True
eBacklashMode	The backlash compensation mode	SMC_BACKLASH_MODE*1	-1: SMC_BL_NEGATIVE 0: SMC_BL_OFF 1: SMC_BL_POSITIVE 2: SMC_BL_AUTO (SMC_BL_AUTO)	When *bExecute* changes from False to True
eBacklashStartState	The initial state of the axis	SMC_BACKLASH_STARTSTATE*2	-1: SMC_BL_START_NEGATIVE 0: SMC_BL_START_NONE 1: SMC_BL_START_POSITIVE (SMC_BL_START_NONE)	When *bExecute* changes from False to True

***Note**:

1. SMC_BACKLASH_MODE: Enumeration (Enum)
2. SMC_BACKLASH_STARTSTATE: Enumeration (Enum)

- **Outputs**

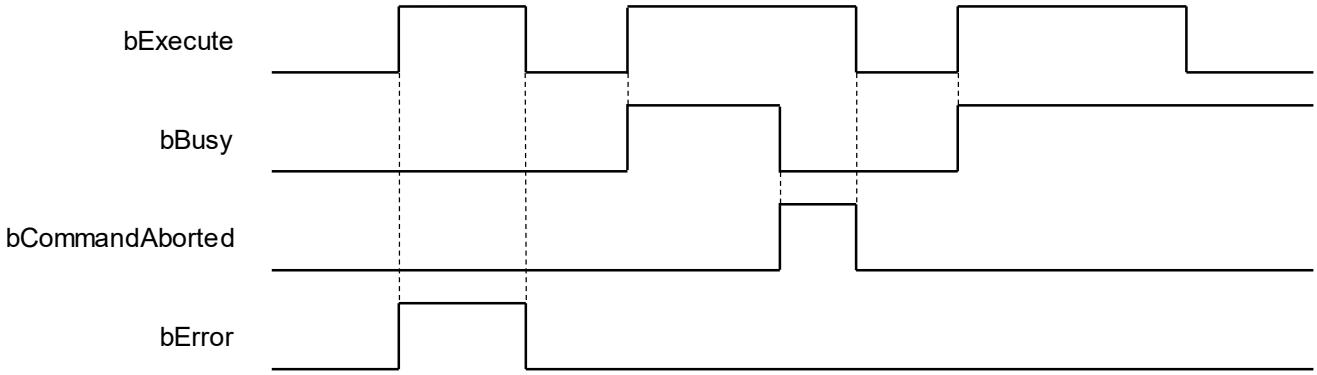
Name	Function	Data Type	Output Range (Default Value)
Busy	True when the instruction is triggered to run.	BOOL	True/False (False)
bCommandAborted	True when the function block is interrupted by another synchronous function block.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
bCompensating	True when compensating for backlash	BOOL	True/False (False)

***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to True	Timing for Shifting to False
bBusy	• When *bExecute* turns to True	• When *bError* turns to True • When *bCommandAborted* turns to True
bCommandAborted	• When MC_GearOut is run • When the function block instruction is interrupted by another function block instruction • When the function block instruction is interrupted by MC_Stop	• When *bExecute* turns to False • If *bExecute* is False and *bCommandAborted* is True, *bCommandAborted* will immediately change to False after maintaining a True state for a scan cycle.
bError	• When an error occurs in execution conditions or input values for the instruction	• When *bExecute* turns to False (Error Code is cleared)
ErrorID		
bCompensating	• When backlash compensation is undergoing	• When backlash compensation is not performed

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Master	Master axis referenced	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and Busy is False.
Slave	Slave axis referenced	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and Busy is False.

***Note**: AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for function blocks.

- **Function**

  - SMC_BacklashCompensation can be used to compensate for the backlash of drive belt devices or the transmission box.
    - SMC_BACKLASH_MODE

Compensation mode	Description
SMC_BL_NEGATIVE	Backlash compensation is only performed when the axis is running in the opposite direction.
SMC_BL_OFF	No backlash compensation
SMC_BL_POSITIVE	Backlash compensation is only performed when the axis is running in the positive direction.
SMC_BL_AUTO	Backlash is compensationd regardless of the direction in which the axis is running.

    - SMC_BACKLASH_STARTSTATE

The Initial State of the Master and Slave Axes	Description
SMC_BL_START_NEGATIVE	Reverse traction is initially applied to the slave axis by the master slave.
SMC_BL_START_NONE	No traction is initially applied to the slave axis by the master slave.
SMC_BL_START_POSITIVE	Positive traction is initially applied from the shaft to the main shaft.

    - When SMC_BacklashCompensation is run, even if the master axis is stationary, the function block will first perform compensation based on the MC_BL_START_NONE and SMC_BACKLASH_MODE, in which bCompensating will not turn to True.

The Initial State of the Master and Slave Axes	Compensation Mode	Behavior Pattern
SMC_BL_START_NONE	SMC_BL_POSITIVE	When the function block starts but the master axis is stationary, the slave axis is first compensationd forward to the fBacklash value set. Assuming fBacklash = 10, after the function block starts, the master axis position = 0, and the slave axis position = 5.
	SMC_BL_NEGATIVE	When the function block starts but the master axis is stationary, the slave axis is first compensationd backwards to the fBacklash value set. Assuming fBacklash = 10, after the function block starts, the master axis position = 0, and the slave axis position = -5.
SMC_BL_START_POSITIVE	SMC_BL_NEGATIVE	When the function block starts but the master axis is stationary, the slave axis is first compensationd backwards to the fBacklash value set. Assuming fBacklash = 10, after the function block starts, the master axis position = 0, and the slave axis position = -10.

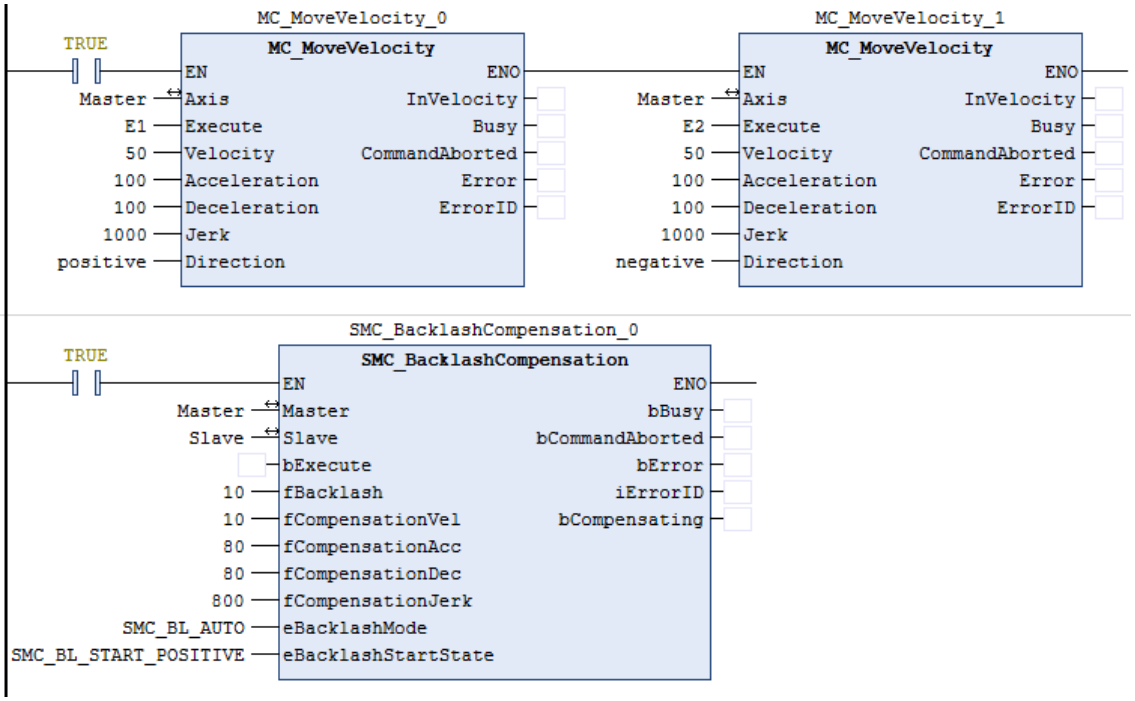
The Initial State of the Master and Slave Axes	Compensation Mode	Behavior Pattern
SMC_BL_START_NEGATIVE	SMC_BL_POSITIVE	When the function block starts but the master axis is stationary, the slave axis is first compensationd forward to the fBacklash value set. Assuming fBacklash = 10, after the function block starts, the master axis position = 0, and the slave axis position = 10.

- **Troubleshooting**

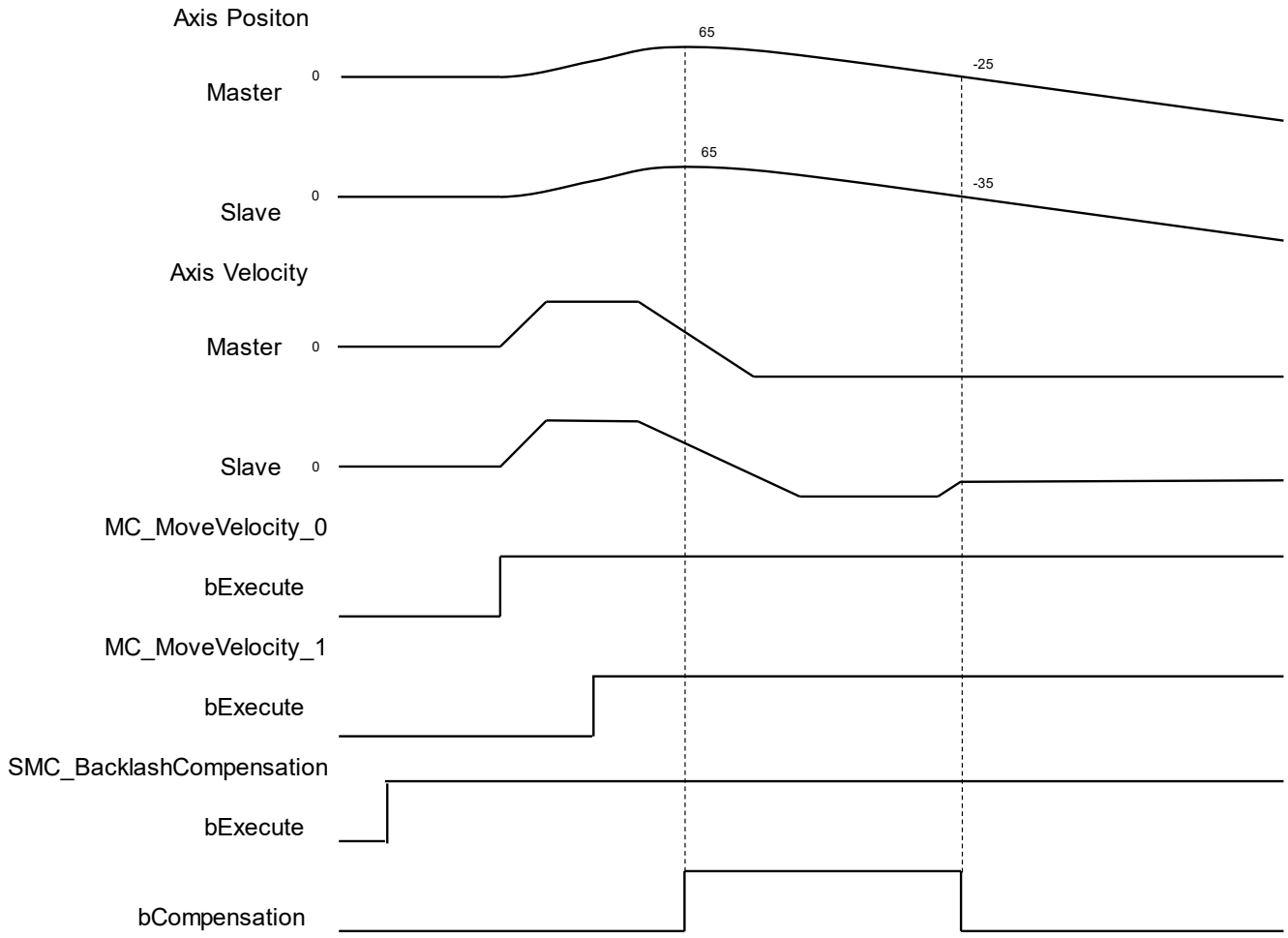
  - If an error occurs during the execution of the instruction, *bError* will change to True. You can refer to ErrorID (Error Code) to address the problem.

- **Programming Example**

  - This example illustrates the backlash compensation behavior of SMC_BacklashCompensation based on the following settings.



  - Timing Diagram



1. Run the SMC_BacklashCompensation first, then move forward, and then reverse to observe the backlash compensation.
2. Since the SMC_BACKLASH_MODE is set to SMC_BL_AUTO and the SMC_BACKLASH_STARTSTATE is set to SMC_BL_START_POSITIVE, the slave axis will not be compensationd at start-up.
3. The MC_MoveVelocity_0 is performed first, at which point the forward movement is carried out, and since the positive traction force is applied to the slave axis by the master axis at the beginning, there is no need for backlash compensation.
4. And then immediately run the MC_MoveVelocity_1. The reverse movement begins. At this time, the backlash affects the synchronization of the slave axis, so SMC_BacklashCompensation starts the backlash compensation, and the function block will move 10 distances in reverse in advance at the command position of the slave axis. After compensation, the actual master and slave axes are fully synchronized. bCompensating is True during the period of backlash compensation

2.3 Delta Motion Control Instructions

2.3.1 Motion Control Instructions

Motion instructions generally refer to the ability to control the motor to move after the instruction is run. The function blocks used in this section are from the library "DL_MotionControl*" and the function blocks used can be synchronized with the driver, so when setting the axis, select the synchronous axis.

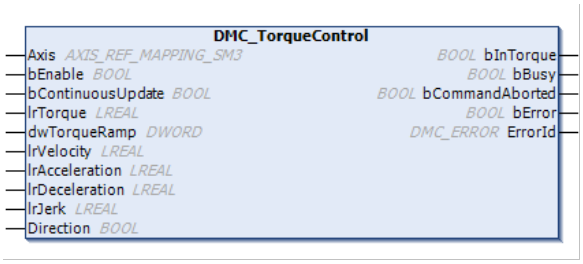
For setting up the synchronous axis, refer to section 7.4 in the *AX-3 Series Operation Manual*.

*Note: When the version of SM3_Basic is not V4.6.1.0 to match with V1.1.0.0 and earlier, an error "Type 'xxxxxx' is not equal to type 'Axis'VAR_IN_OUT 'AXIS_REF_SM3'" will appear when compiling. Change the Softmotion library version to V4.6.1.0.

2.3.1.1 DMC_TorqueControl

- **Supported Devices**: AX-308E, AX-364E

DMC_TorqueControl controls the torque according to the torque control mode of the servo drive.

FB/FC	Instruction	Graphic Expression
FB	DMC_TorqueControl	

ST Language

```
DMC_TorqueControl_instance(
Axis : =,
bEnable : =,
bContinuousUpdate : =,
lrTorque : =,
dwTorqueRamp : =,
lrVelocity : =,
lrAcceleration : =,
lrDeceleration : =,
lrJerk : =,
Direction : =,
bInTorque =>,
bBusy =>,
bCommandAborted =>,
bError =>,
ErrorID =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is enabled when *bEnable* changes from False to True.	BOOL	True/False (False)	-
bContinuousUpdate	The target torque maximum speed can be updated continuously when *bContinuousUpdate* is True[*1].	BOOL	True/False (False)	When *bEnable* turns to True and *Busy* is False.
lrTorque	Specify the target torque. (Unit: N.m)	LREAL	Negative, positive, 0 (0)	When *bEnable* turns to True and *Busy* is False.
dwTorqueRamp	Specify the change rate of the torque (Unit: ms) [*2]	DWORD	Positive (0)	When *bEnable* turns to True and *Busy* is False.
lrVelocity	Specify the maximum velocity.	LREAL	Positive (0)	When *bEnable* turns to True and *Busy* is False.
lrAcceleration	Reserved	LREAL	-	-
lrDeceleration	Reserved	LREAL	-	-
lrJerk	Reserved	LREAL	-	-
Direction	Reserved	BOOL	-	-

*__Note__:

1. DL_MotionControl version V1.0.1.0 includes the above support, when bContinuousUpdate is True, the torque and the maximum speed can be modified immediately.

2. Take ASDA-A2 for example here with the unit: μs (microsecond). For other models, refer to 0x6087 in their object dictionaries.

- **Outputs**

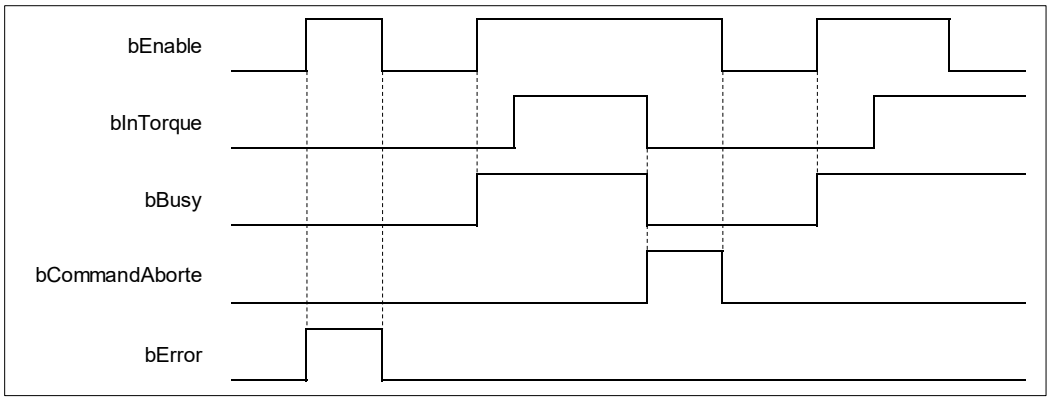
Name	Function	Data Type	Output Range Value (Default Value)
bInTorque	True when the target torque is reached.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note:** DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bInTorque	• When the *bEnable* is True and the axis motion state can be read.	• When *bEnable* turns to False. • When *bError* turns to True.
bBusy	• When *bEnable* turns to True.	• When *bEnable* turns to False. • When *bError* turns to True.
bCommandAborted	• When this instruction is aborted by another instruction.	• When *bEnable* turns to False.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bEnable* turns to False. (The value in ErrorID is cleared.)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

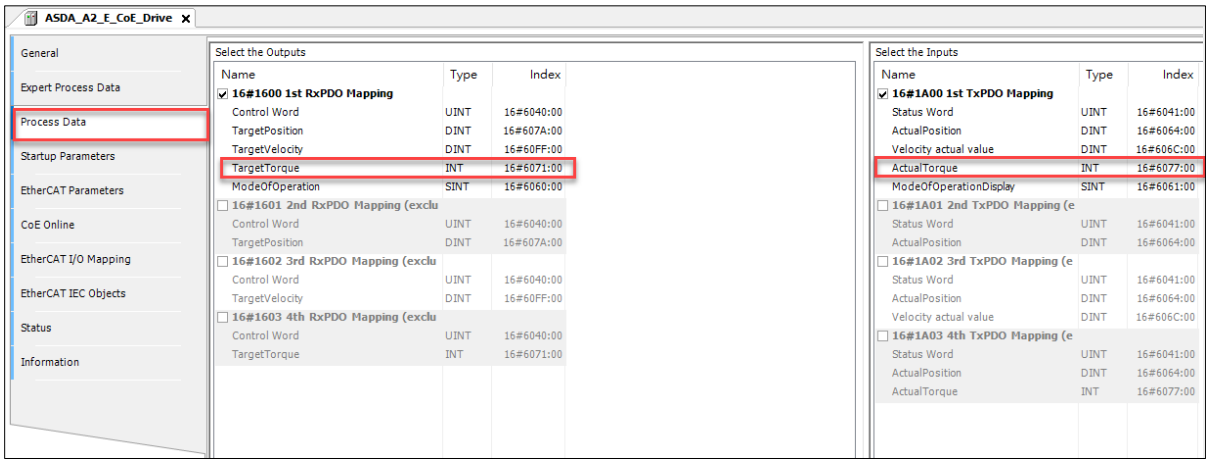
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *bEnable* turns to True and *bBusy* is False.

***Note:**

AXIS_REF_SM3 (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**
  - When *bEnable* turns to True, the values of *lrTorque*, *dwTorqueRamp* and *lrVelocity* of the DMC_TorqueControl instruction are sent to the servo for torque control.
  - When *bEnable* is False, set the target torque *lrTorque* to 0 to make the axis decelerate to a stop. The instruction execution is completed when the axis decelerates to a stop and *bBusy* turns to False.
  - Ensure that the axis is in Standstill state before instruction execution.
  - The servo will perform an immediate stop if SMC_SetControllerMode interrupts DMC_TorqueControl during instruction execution. Do not do so.
  - Only one DMC_TorqueControl instruction is allowed to run at a time. If the second DMC_TorqueControl instruction is also run at the same moment, an error “DMC_TC_FB_CONFLICT” will occur.

- When the DMC_TorqueControl instruction is run, 0x6071 (Target Torque), 0x6077 (Torque actual value), 0x6060 (ModeOfOperation) and 0x6061 (ModeOfOperationDisplay) OD must be included in the slave PDO mapping data. Otherwise, an error will occur.

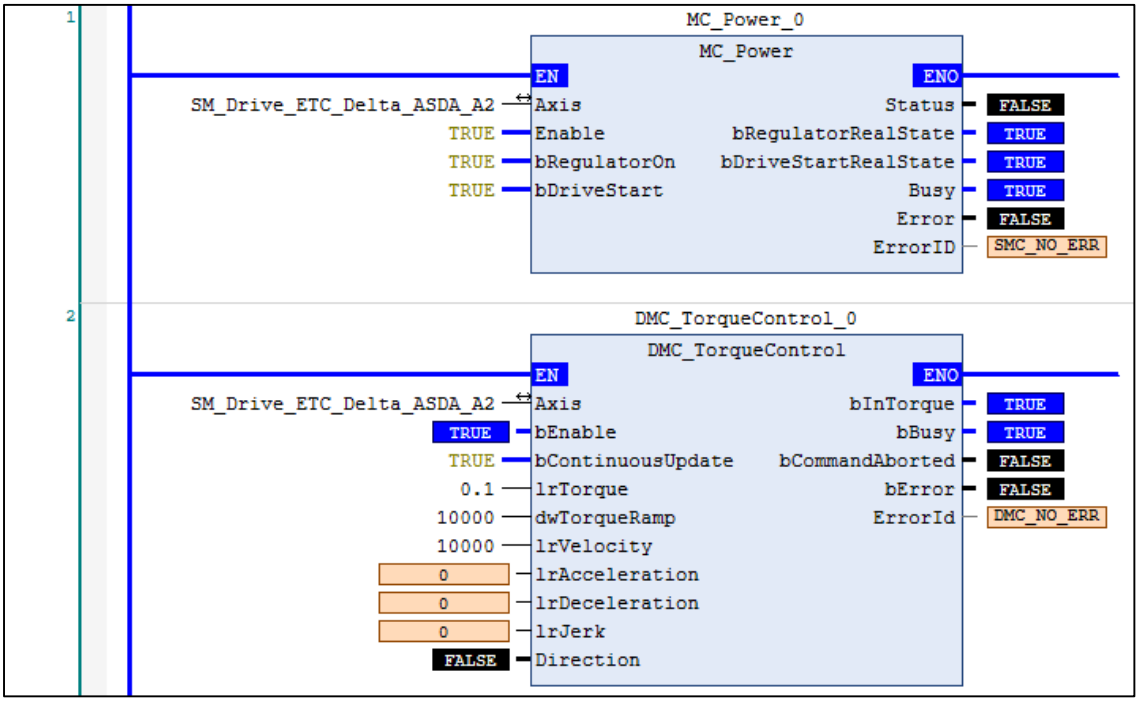


- **Troubleshooting**

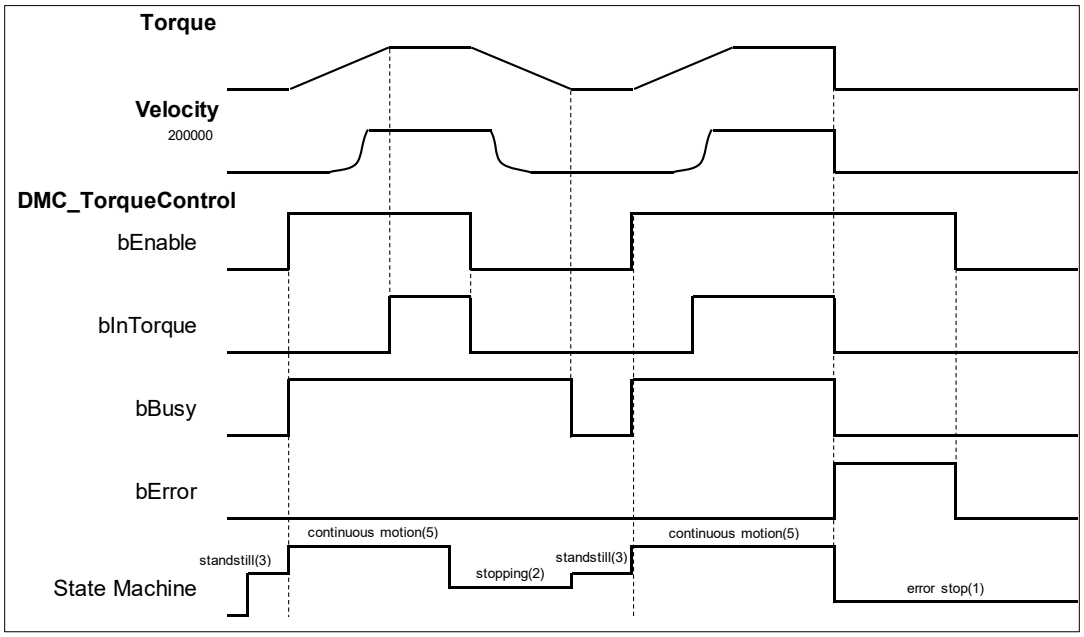
  - If an error occurs during the execution of the instruction, *bError w*ill change to True. You can refer to *ErrorID* (Error Code) to address the problem.

- **Programming Example**

  - This example shows the motion behavior performed by DMC_TorqueControl.



  - **Timing Diagram**

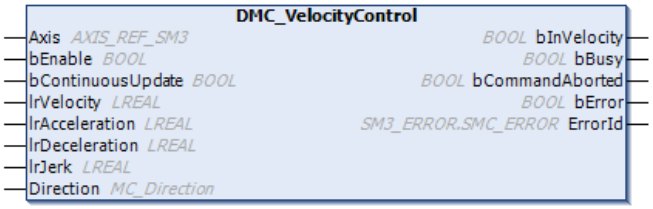


- After DMC_TorqueControl has started, the servo starts to run according to the input settings of the instruction for the target torque *lrTorque*, change rate of the torque *dwTorqueRamp* and maximum velocity *lrVelocity*.
- After *bEnable* of DMC_TorqueControl turns to False, the axis starts to decelerate till it stops. When the axis decelerates to a stop, *bBusy* turns to False.
- An error occurs on the axis while DMC_TorqueControl has been run for a period of time after being started one more time. At the moment, the axis performs an immediate stop for the error and then the instruction will report an error.

2.3.1.2 DMC_VelocityControl

- **Supported Devices**: AX-308E, AX-364E

DMC_VelocityControl performs a velocity control on a specified axis in the CSV speed mode with the specified behavior and an average velocity.

FB/FC	Instruction	Graphic Expression
FB	DMC_VelocityControl	

ST Language

```
DMC_VelocityControl_instance(
Axis : =,
bEnable : =,
bContinuousUpdate : =,
lrVelocity : =,
lrAcceleration : =,
lrDeceleration : =,
lrJerk : =,
Direction : =,
bInVelocity =>,
bBusy =>,
bCommandAborted =>,
bError =>,
ErrorID =>) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is enabled when *bEnable* changes from FALSE to TRUE.	BOOL	True/False (False)	-
bContinuousUpdate*1	The target velocity can be updated continuously when *bContinuousUpdate* is True	BOOL	True/False (False)	When *bEnable* turns to True and *Busy* is False.
lrVelocity	Target velocity (Unit: user unit/s)	LREAL	Positive (0)	When *bEnable* turns to True and *Busy* is False.
lrAcceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	When *bEnable* turns to True and *Busy* is False.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrDeceleration	Deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	When *bEnable* turns to True and *Busy* is False.
lrJerk	Jerk value. (Unit: user unit/s^3)	LREAL	Positive (0)	When *bEnable* turns to True and *Busy* is False.
Direction	Specifies the motion direction of the servo motor.	MC_ DIRECTION*2	3: fastest 2: current 1: positive 0: shortest -1: negative (current) *3	When *bEnable* turns to True and *Busy* is False.

***Note:**

1. After bContinuousUpdate has started, change the speed, and acceleration & deceleration will immediately take effect.
2. MC_DIRECTION: Enumeration (ENUM).
3. The options fastest, current and shortest are only for the rotary axis.

- **Outputs**

Name	Function	Data Type	Output Range Value (Default Value)
bInVelocity	True when the specified target velocity is reached.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

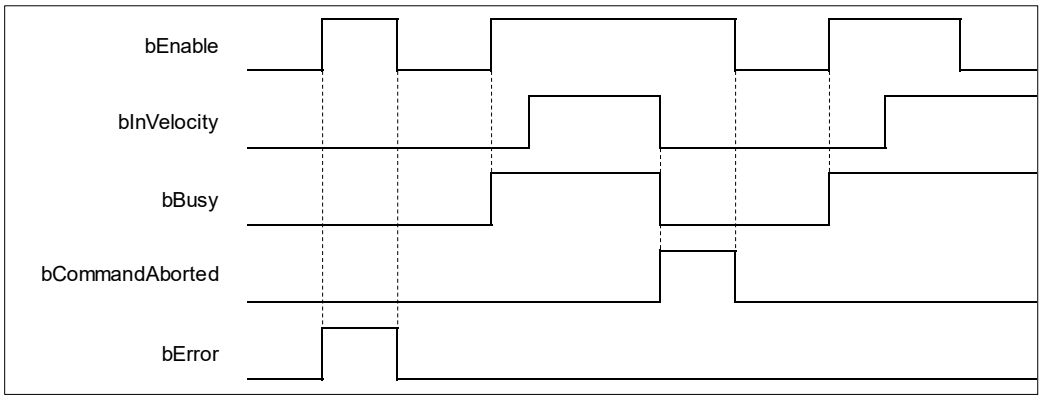
***Note:** DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bInVelocity	• When the specified target velocity is reached.	• When *bCommandAborted* turns to True. • When *bContinuousUpdate* is True. and lrVelocity value is changed. • When *bEnable* turns to False. • When *bError* turns to True.
bBusy	• When *bEnable* turns to True.	• When *bCommandAborted* turns to True. • When the axis decelerates to a stop after bEnable turns to False. • When *bError* turns to True.

Name	Timing for shifting to True	Timing for shifting to False
bCommandAborted	• When this instruction is aborted by another instruction. • When this instruction is aborted via MC_Stop instruction.	• When *bEnable* turns to False.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	• When *bEnable* turns from True to False. (Error Code is cleared)
ErrorID		

■ **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *bEnable* turns to True and *bBusy* is False.

***Note:**

AXIS_REF_SM3 (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

■ The instruction performs speed control with specified target velocity (*lrVelocity*), acceleration rate (*lrAcceleration*), deceleration rate (*lrDeceleration*) and Jerk value (*lrJerk*) when *bEnable* changes to True.

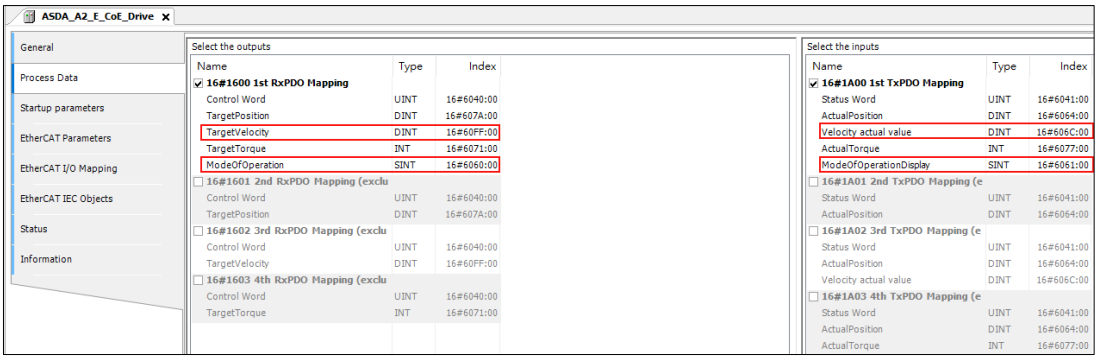
◆ You can execute another motion instruction to abort the ongoing motion of MC_VelocityControl. But the servo is in CSV mode and the control mode will not be switched to CSP control mode.

◆ When the instruction is interrupted by another instruction, the output *bInVelocity* turns to False and *bCommandAborted* turns to True.

◆ When the input *bContinuousUpdate* of the instruction turns to True and the target velocity is given a new value, the axis velocity will update to the new speed.

◆ When *bEnable* turns to False, the instruction makes the axis decelerate to a stop and the mode is switched to CSP control mode.

◆ When the instruction is run, 0x60FF (Target Velocity), 0x606C (Velocity actual value), 0x6060 (ModeOfOperation) and 0x6061 (ModeOfOperationDisplay) OD must be included in the slave PDO mapping data. Otherwise, the servo will not be able to work.



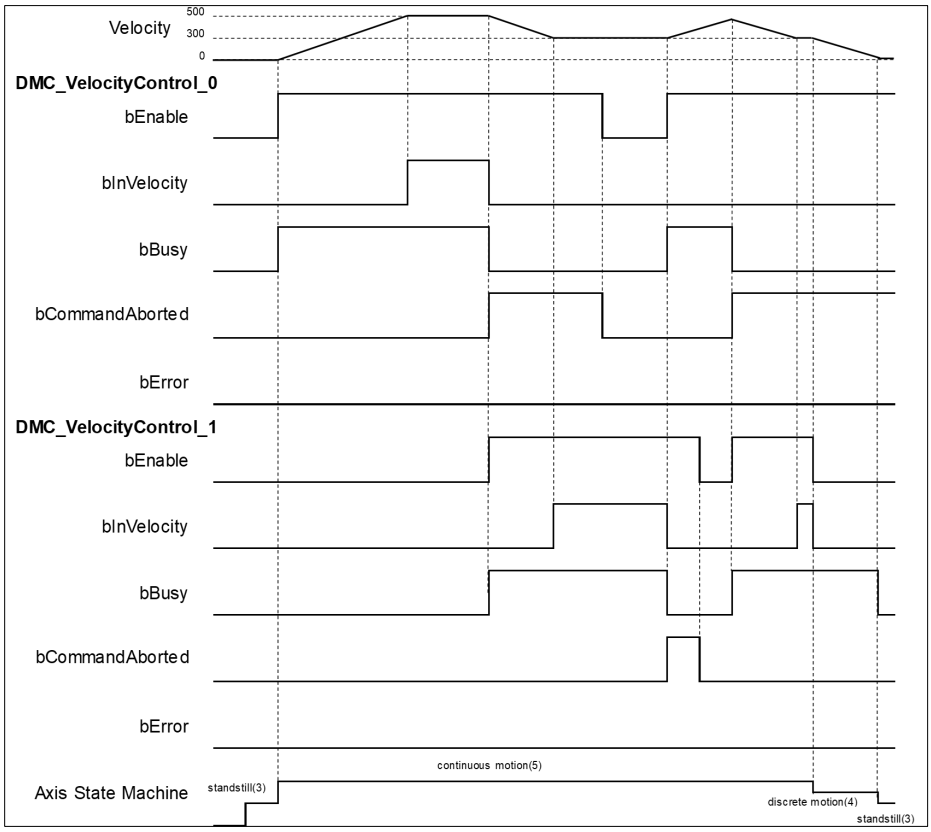
- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *bError* will change to True. You can refer to *ErrorID* (Error Code) to address the problem.

- **Programming Example**

  - The example shows the motion behavior performed by DMC_VelocityControl.

  - **Timing Diagram**

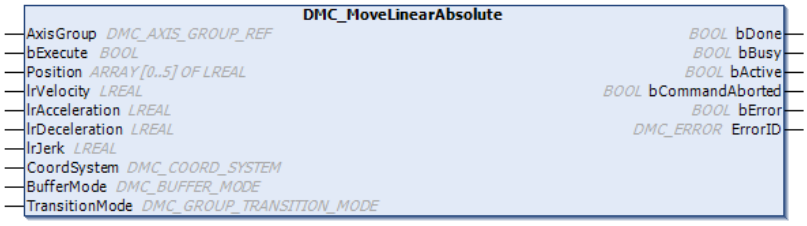


- When *bEnable* of DMC_VelocityControl_0 changes to True, the instruction controls the axis to reach the specified target velocity 500. When it reaches 500, *bInVelocity* of DMC_VelocityControl_0 changes to True.
- When *bEnable* of DMC_VelocityControl_1 changes to True, DMC_VelocityControl_0 is interrupted and *bInVelocity* of the instruction changes to False and *bCommandAborted* changes to True.
- The DMC_VelocityControl_1 instruction decelerates the axis to the velocity 300. When 300 is reached, *bInVelocity* of DMC_VelocityControl_1 will change to True and remain in this status as long as the velocity is not changed.
- When *bEnable* of DMC_VelocityControl_0 changes to False, *bCommanAborted* changes to False.
- When DMC_VelocityControl_0 has started again through changing *bEnable* of DMC_VelocityControl_0 to True, DMC_VelocityControl_0 will be aborted and the axis will accelerate to 500.
- If *bEnable* of DMC_VelocityControl_1 changes from False to True again when the target velocity of DMC_VelocityControl_0 has not been reached yet, DMC_VelocityControl_0 will be aborted. In this case, the axis will decelerate again without reaching the target velocity 500 of DMC_VelocityControl_0.
- *bInVelocity* of DMC_VelocityControl_1 changes to True when the target velocity of DMC_VelocityControl_1 is reached.
- When *bEnable* of DMC_VelocityControl_1 changes to True in the next cycle, the axis starts to decelerate to a stop and then *bBusy* of DMC_VelocityControl_1 changes to False.

2.3.1.3 DMC_MoveLinearAbsolute

- **Supported Devices**: AX-308E, AX-364E

DMC_MoveLinearAbsolute controls a specified axis group to perform the absolute linear interpolation for a specified absolute position.

FB/FC	Instruction	Graphic Expression
FB	DMC_MoveLinearAbsolute	

ST Language

```
DMC_MoveLinearAbsolute_instance(
AxisGroup: = ,
bExecute: = ,
Position: = ,
lrVelocity: = ,
lrAcceleration: = ,
lrDeceleration: = ,
lrJerk: = ,
CoordSystem: =,
BufferMode: = ,
TransitionMode: = ,
bDone=> ,
bBusy=> ,
bActive=> ,
bCommandAborted=> ,
bError=> ,
ErrorID=> ) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
Position	Specify the absolute target position for each axis in the specified axis group. (Unit: user unit)	LREAL[6]	[_, _, _, _, _, _] Positive or negative ([0, 0, 0, 0, 0, 0])	When *bExecute* turns to True.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrVelocity	Specify the target velocity for the specified axis group. (Unit: user unit/s)	LREAL	Positive (0)	When *bExecute* turns to True.
lrAcceleration	Specify the acceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True.
lrDeceleration	Specify the deceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True.
lrJerk	Specify the jerk. (Unit: user unit/s^3)	LREAL	Positive (0)	When *bExecute* turns to True.
CoordSystem	Coordinate system	DMC_COORD_SYSTEM*1	0: ACS 1: MCS 2: WCS (Reserved) 3: PCS_1 (Reserved) 4: PCS_2 (Reserved) 5: TCS (Reserved) (1)	When *bExecute* is on the rising edge, the setting parameters of CoordSystem will be updated.
BufferMode	Specify a buffer mode for the instruction *1	DMC_ BUFFER_ MODE	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	When *bExecute* turns to True.
TransitionMode	Specify a transition mode for the instruction*2.	DMC_ GROUP_ TRANSITION_ MODE	0: None 10: Overlap (0)	When *bExecute* turns to True.

***Note:**

1. Refer to *AX-3 Series Operation Manual* for details on BufferMode.
2. Refer to *AX-3 Series Operation Manual* for details on TransitionMode.

- **Outputs**

Name	Function	Data Type	Output Range Value (Default Value)
bDone	True when the absolute positioning is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bActive	True when the instruction is controlling axes.	BOOL	True/False (False)

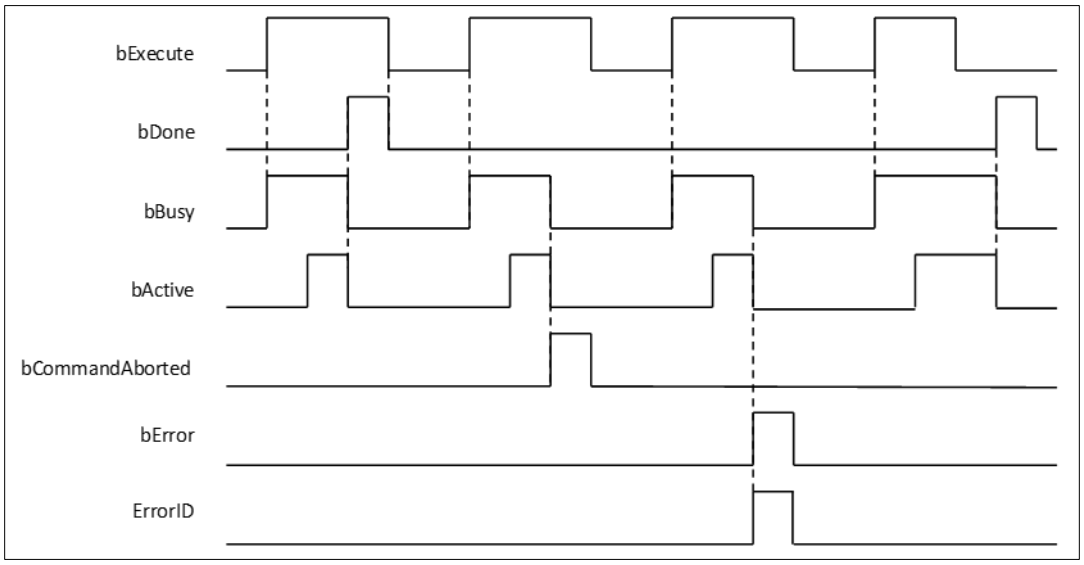
Name	Function	Data Type	Output Range Value (Default Value)
bCommand Aborted	True when the instruction execution is aborted.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the absolute positioning is completed.	• When *bExecute* turns to False. • When *bExecute* is False but *bDone* turns to True, *bDone* will remain True for one scan cycle and then change to False.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bActive	• When axes start being controlled by the instruction.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True. • *bActive* will change to False after remaining True for at least one cycle when *bExecute* is False but *bActive* changes to True.
bCommand Aborted	• When the instruction is aborted by another instruction BufferMode of which is set to Aborting. • When the instruction is aborted by MC_Stop. • When the instruction is aborted by DMC_GroupStop.	• When *bExecute* turns to False. • *bCommandAborted* will change to False after remaining True for one cycle when *bExecute* is False but *bCommandAborted* changes to True.
bError/ErrorID	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded in ErrorID)	• When *bExecute* turns from True to False. (Error Code is cleared)

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*	DMC_AXIS_ GROUP_REF	When *bExecute* turns to True and *bBusy* is False.

***Note:**

DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

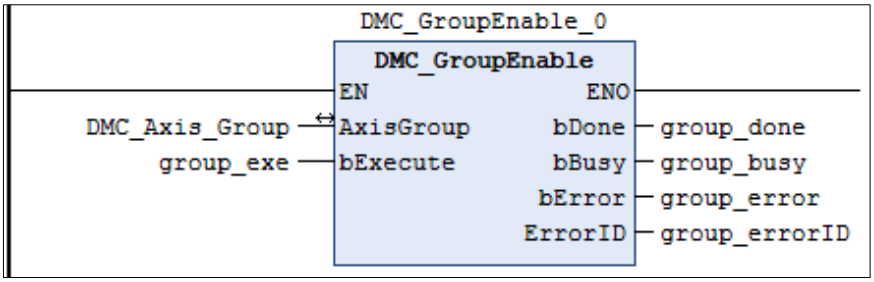
- **Function**
  - The instruction supports the absolute linear interpolation of maximum six axes, where the six axes can simultaneously start, stop as well as reach the specified absolute target position.
  - At least one axis is needed for the linear interpolation. An error will occur if there is a travel distance for the axis which is not set.
  - The function of CoordSystem needs to be supported by DL_MotionControl library V1.1.0.0 or above.

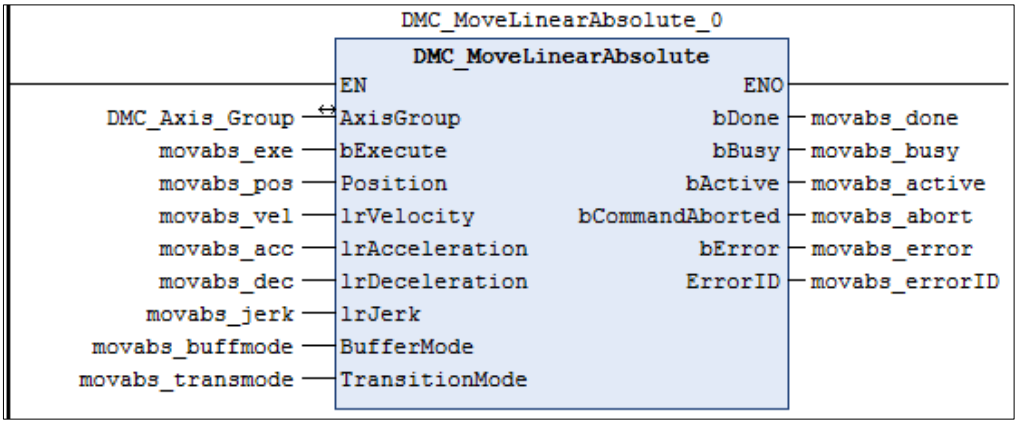
- **Troubleshooting**
  - When an error occurs during the execution of the instruction, *bError* changes to True and axes stops running. To confirm current error state, see the error code in *ErrorID*.
  - For error codes and corresponding trouble shootings, refer to **Appendix** for error code descriptions.

- **Programming Example**
  - In this example, the path for the six-axis absolute linear interpolation is planned and the six axes simultaneously reach the target position through traveling an absolute distance from current positon.

Axis group	Target position
Axis1	1000
Axis2	2000
Axis3	3000
Axis4	4000
Axis5	5000
Axis6	6000

```
DMC_MoveLinearAbsolute_0: DMC_MoveLinearAbsolute;
movabs_exe: BOOL;
movabs_pos: ARRAY [0..5] OF LREAL := [1000, 2000, 3000, 4000, 5000, 6000];
movabs_vel: LREAL := 1000;
movabs_acc: LREAL := 100;
movabs_dec: LREAL := 100;
movabs_jerk: LREAL := 0;
movabs_buffmode: DMC_BUFFER_MODE;
movabs_transmode: DMC_GROUP_TRANSITION_MODE;
movabs_done: BOOL;
movabs_busy: BOOL;
movabs_active: BOOL;
movabs_abort: BOOL;
movabs_error: BOOL;
movabs_errorID: DMC_ERROR;
```



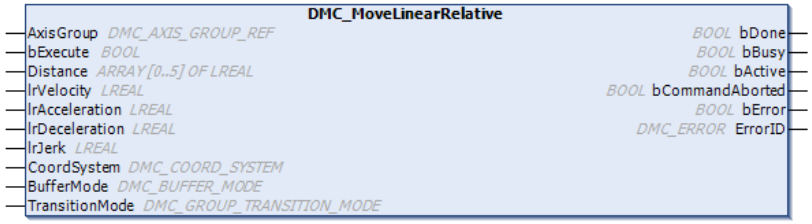


- When moveabs_exe (*bExecute*) changes to True, DMC_MoveLinearAbsolute starts to perform the absolute linear interpolation for six axes.
- When moveabs_done (*bDone*) changes to True, moveabs_busy (*bBusy*) changes to False, which means the specified absolute positioning (1000, 2000, 3000, 4000, 5000, 6000) is completed.
- moveabs_exe (*bExecute*) is switched to False after the absolute linear interpolation is completed. Then moveabs_done (*bDone*) will change to False automatically.
- If moveabs_exe (*bExecute*) is set to True again for the absolute linear interpolation, then no axes will move for positioning.

2.3.1.4 DMC_MoveLinearRelative

- **Supported Devices**: AX-308E, AX-364E

DMC_MoveLinearRelative controls a specified axis group to perform the relative linear interpolation.

FB/FC	Instruction	Graphic Expression
FB	DMC_MoveLinearRelative	

ST Language

```
DMC_MoveLinearRelative_instance(
AxisGroup: =,
bExecute: =,
Distance: =,
lrVelocity: =,
lrAcceleration: =,
lrDeceleration: =,
lrJerk: =,
CoordSystem: =,
BufferMode: =,
TransitionMode: =,
bDone=>,
bBusy=>,
bActive=>,
bCommandAborted=>,
bError=>,
ErrorID=>) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Distance	Specify the travel distance for each axis in the specified axis group. (Unit: user unit)	LREAL[6]	[_, _, _, _, _, _] Positive, negative or 0 ([0, 0, 0, 0, 0, 0])	When *bExecute* turns to True.
lrVelocity	Specify the target velocity for the axis group. (Unit: user unit/s)	LREAL	Positive (0)	When *bExecute* turns to True.
lrAcceleration	Specify the acceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True.
lrDeceleration	Specify the deceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True.
lrJerk	Specify the jerk. (Unit: user unit/s^3)	LREAL	Positive (0)	When *bExecute* turns to True.
CoordSystem	Coordinate system	DMC_COORD_SYSTEM*1	0: ACS 1: MCS 2: WCS （Reserved） 3: PCS_1 （Reserved） 4: PCS_2 （Reserved） 5: TCS（Reserved） （1）	When *bExecute* is on the rising edge, the setting parameters of CoordSystem will be updated.
BufferMode	Specify a buffer mode for the instruction.*1	DMC_ BUFFER_ MODE	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	When *bExecute* turns to True.
TransitionMode	Specify a transition mode for the instruction *2	DMC_ GROUP_ TRANSITION_MODE	0: None 10: Overlap (0)	When *bExecute* turns to True.

***Note:**

1. Refer to *AX-3 Series Operation Manual* for details on BufferMode.
2. Refer to *AX-3 Series Operation Manual* for details on TransitionMode.

- **Outputs**

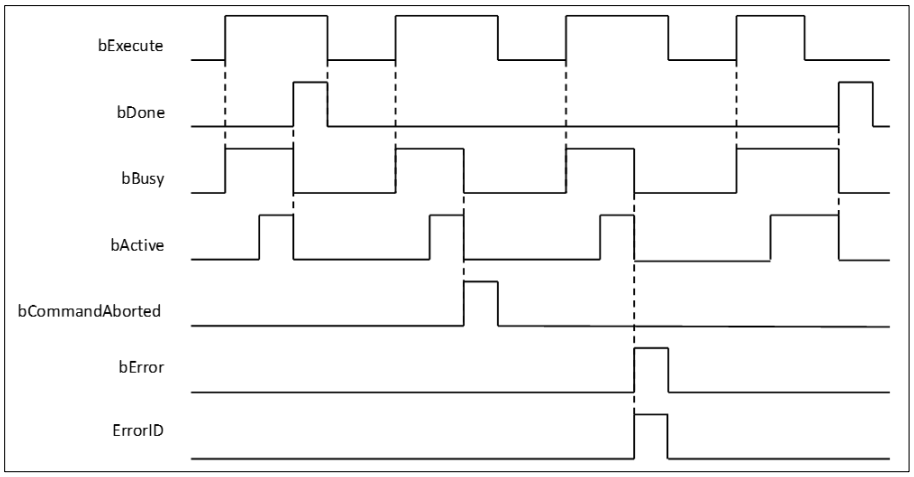
Name	Function	Data Type	Output Range Value (Default Value)
bDone	True when the relative positioning is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bActive	True when the instruction is controlling axes.	BOOL	True/False (False)
bCommand Aborted	True when the instruction execution is aborted.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to **Appendix** for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note:** DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the relative positioning is completed.	• When *bExecute* turns to False. • When *bExecute* is False but *bDone* turns to True, *bDone* will remain True for one scan cycle and then change to False.
bBusy	• When *bExecute* turns to TRUE.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bActive	• When axes start being controlled by the instruction.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True. • When *bExecute* is False but *bActive* turns to True, *bActive* will remain True for one cycle and then change to False.
bCommand Aborted	• When the instruction is interrupted by another instruction whose BufferMode is set to Aborting. • When the instruction is interrupted by MC_Stop. • When the instruction is aborted by DMC_GroupStop.	• When *bExecute* turns to False. • When *bExecute* is False but *bCommandAborted* turns to True, *bCommandAborted* will remain True for one cycle and then change to False.
bError/ErrorID	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded in ErrorID)	• When *bExecute* turns to False. (Error Code is cleared)

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Output range	Name
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	When *bExecute* turns to True and *bBusy* is False.

*__Note__: DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

  - The instruction supports the relative linear interpolation of maximum six axes, where the six axes can simultaneously start, stop as well as reach the specified target relative position.
  - At least one axis is needed for the linear interpolation. An error will occur if there is a travel distance for the axis which is not set.
  - The function of CoordSystem needs to be supported by DL_MotionControl library V1.1.0.0 or above.

- **Troubleshooting**

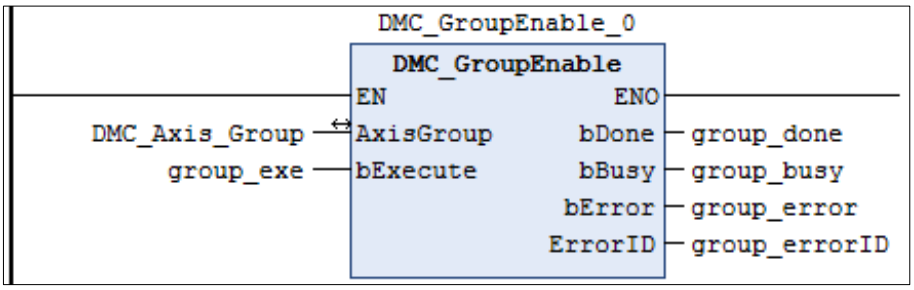
  - When an error occurs during the execution of the instruction, *bError* changes to True and axes stops running. To confirm current error state, see the error code in *ErrorID*.
  - For error codes and corresponding trouble shootings, refer to **Appendix** for error code descriptions.

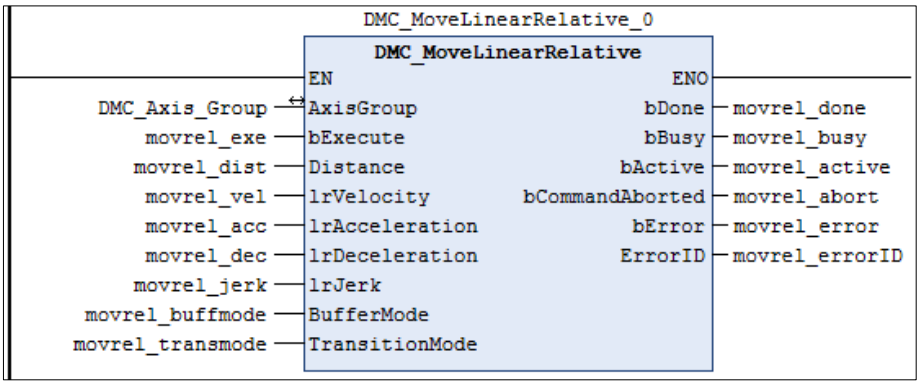
- **Programming Example**

  - In this example, the path for the six-axis relative linear interpolation is planned and six axes simultaneously reach the target relative position through traveling a relative distance from current positon.

The number of each axis in the axis group	Current position	Relative distance	Target position
Axis 1	1000	1000	2000
Axis 2	1000	2000	3000
Axis 3	1000	3000	4000
Axis 4	1000	4000	5000
Axis 5	1000	5000	6000
Axis 6	1000	0	1000

```
DMC_MoveLinearRelative_0: DMC_MoveLinearRelative;
movrel_exe: BOOL;
movrel_dist: ARRAY [0..5] OF LREAL := [1000, 2000, 3000, 4000, 5000, 0];
movrel_vel: LREAL := 1000;
movrel_acc: LREAL := 100;
movrel_dec: LREAL := 100;
movrel_jerk: LREAL := 0;
movrel_buffmode: DMC_BUFFER_MODE;
movrel_transmode: DMC_GROUP_TRANSITION_MODE;
movrel_done: BOOL;
movrel_busy: BOOL;
movrel_active: BOOL;
movrel_abort: BOOL;
movrel_error: BOOL;
movrel_errorID: DMC_ERROR;
```



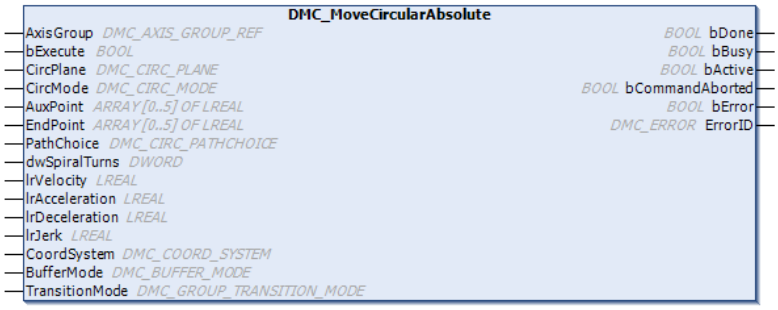


- When movrel_exe (*bExecute*) changes to True, DMC_GroupRelLinear starts to perform the relative linear interpolation for six axes.
- When movrel_done (*bDone*) changes to True, movrel_busy (*bBusy*) and movrel_abort (*bAborted*) change to False, which means the specified relative positioning (1000, 2000, 3000, 4000, 5000, 0) is completed.
- movrel_exe (*bExecute*) is switched to False after the relative linear interpolation is completed. Then movrel_done (*bDone*) will change to False automatically.
- If movrel_exe (*bExecute*) is set to True again, axes will perform the relative linear interpolation one more time to reach the target position (3000, 5000, 7000, 9000, 11000, 1000).
- When the target positioning is completed, movrel_done (*bDone*) changes to True again.

2.3.1.5 DMC_MoveCircularAbsolute

- **Supported Devices**: AX-308E, AX-364E

DMC_MoveCircularAbsolute controls the axis group to perform circular or helical interpolation for a specified absolute target position.

FB/FC	Instruction	Graphic Expression
FB	DMC_MoveCircularAbsolute	

ST Language

```
DMC_MoveCircularAbsolute_instance(
AxisGroup: = ,
bExecute: = ,
CircPlane: = ,
CircMode: = ,
AuxPoint: = ,
EndPoint: = ,
PathChoice: = ,
dwSpiralTurns: = ,
lrVelocity: = ,
lrAcceleration: = ,
lrDeceleration: = ,
lrJerk: = ,
CoordSystem: =,
BufferMode: = ,
TransitionMode: = ,
bDone=> ,
bBusy=> ,
bActive=> ,
bCommandAborted=> ,
bError=> ,
ErrorID=> ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
CircPlane	Specify the circular or helical plane. *1	DMC_CIRC_ PLANE	0: XY_plane 1: YZ_plane 2: ZX_plane 3: ARBITRARY_plane (0)	When *bExecute* turns to True.
CircMode	Specify the method for circular or helical interpolation. *1	DMC_CIRC_ MODE	0: Border 1: Center 2: Radius (0)	When *bExecute* turns to True.
AuxPoint	Specify the auxiliary point data. *1	LREAL[3]	[_, _, _] Positive, negative or 0 ([0, 0, 0])	When *bExecute* turns to True.
EndPoint	Specify the target position for each axis in the axis group. (Unit: user unit)	LREAL[6]	[_, _, _, _, _, _] Positive, negative or 0 ([0, 0, 0, 0, 0, 0])	When *bExecute* turns to True.
PathChoice	Specify the circular or helical interpolation direction.	DMC_CIRC_ PATHCHOICE	0: Clockwise 1: CounterClockwise (0)	When *bExecute* turns to True.
dwSpiralTurns	Specify the number of spiral turns.	DWORD	0~65535 (0)	When *bExecute* turns to True.
lrVelocity	Specify the target velocity for the axis group. (Unit: user unit/s)	LREAL	Positive (0)	When *bExecute* turns to True.
lrAcceleration	Specify the acceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True.
lrDeceleration	Specify the deceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True.
lrJerk	Specify the jerk. (Unit: user unit/s^3)	LREAL	Positive (0)	When *bExecute* turns to True.
CoordSystem	Coordinate system	DMC_COORD_ SYSTEM*1	0: ACS 1: MCS 2: WCS（Reserved） 3: PCS_1（Reserved） 4: PCS_2（Reserved） 5: TCS（Reserved）	When *bExecute* is on the rising edge, the setting parameters of CoordSystem will be updated.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
			(1)	
BufferMode	Specify a buffer mode for the instruction.*2	DMC_BUFFER_MODE	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	When *bExecute* turns to True.
TransitionMode	Specify a transition mode for the instruction. *3	DMC_GROUP_TRANSITION_MODE	0: None 10: Overlap (0)	When *bExecute* turns to True.

***Note:**

1. Setting parameters *CircPlane*, *CircMode* and *AuxPoint*.
2. Refer to *AX-3 Series Operation Manual* for details on BufferMode.
3. Refer to *AX-3 Series Operation Manual* for details on TransitionMode.

***CircMode* setting**	**Definition *CircMode – AuxPoint***	***CircPlane***		
		XY_Plane	**YZ_Plane**	**ZX_Plane**
		Actual input for *AuxPoint* [_, _, _]		
0	Three points – Absolute coordinate values for the border point (X_A, Y_A, Z_A)	Start point, end point and border point [X_A, Y_A, Z_A]		
1	A center point – Absolute coordinate values for the center point (C_X, C_Y)	[C_X, C_Y, N/A]	[N/A, C_X, C_Y]	[C_Y, N/A, C_X]
2	Radius – Radius (R)	[R, N/A, N/A]		

- **Outputs**

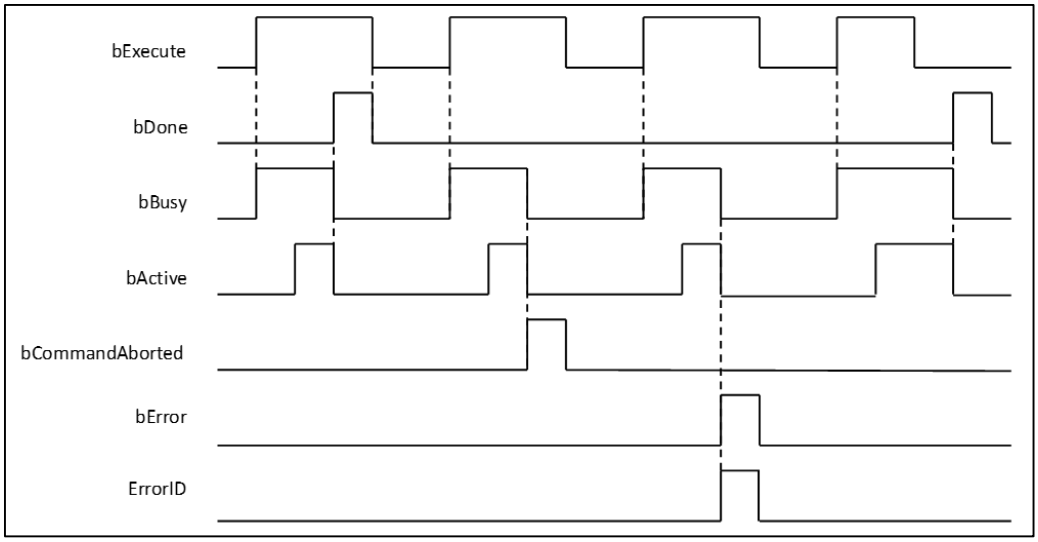
Name	Function	Data Type	Output Range Value (Default Value)
bDone	True when the absolute positioning is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bActive	True when the instruction is controlling axes.	BOOL	True/False (False)
bCommandAborted	True when the instruction execution is interruppted.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note:** DMC_ERROR: Enumeration (ENUM)

■ **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the absolute positioning is completed.	• When *bExecute* turns to False. • *bDone* will change to False after remaining True for one cycle when *bExecute* is False but *bDone* changes to True.
bBusy	• When *bExecute* changes to TRUE.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bActive	• When axes start being controlled by the instruction.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True. • *bActive* will change to False after remaining True for at least one cycle when *bExecute* is False but *bActive* changes to True.
bCommand Aborted	• When the instruction is interrupted by another instruction *BufferMode* of which is set to Aborting. • When the instruction is interrupted by MC_Stop. • When the instruction is interrupted by DMC_GroupStop.	• When *bExecute* turns to False. • *bCommandAborted* will change to False after remaining True for at least one cycle when *bExecute* is False but *bCommandAborted* changes to True.
bError/ErrorID	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded in ErrorID).	• When *bExecute* turns from True to False. (Error Code is cleared)

■ **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	When *bExecute* turns to True and *bBusy* is False.

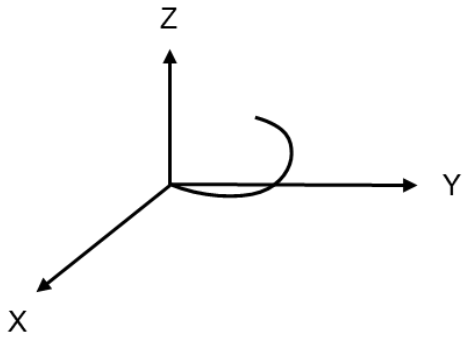
***Note:**

DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**
  - The instruction supports the absolute helical interpolation of maximum three axes, where the three axes can simultaneously start, stop as well as reach the specified absolute target position.
  - The instruction can be used to specify the circle drawing on the plane parallel to XY / YZ / ZX and set the height of the helix on Z / X / Y axis.
  - If the start point and end point for circular interpolation are set to the same point, use the center point mode (*CircMode* = Center) for the interpolation.
  - At least two axes are needed for circular interpolation. An error will occur if there is a travel distance for an axis which is not set.
  - When the start points and end point for circular interpolation are set as the same point, the instruction will perform the rotation for one complete circle.
  - The function of CoordSystem needs to be supported by DL_MotionControl library V1.1.0.0 or above.
  - CircPlane added ARBITRARY_plane in DL_MotionControl library V1.2.0.0 or later.
  - ARBITRARY_plane can draw ellipses in space.
    - Use the following parameters to draw the ellipse.

Name	Setting Value
CircPlane	DMC_CIRC_PLANE.ARBITRARY_plane
CircMode	DMC_CIRC_MODE.border
AuxPoint	[1000, 2000, 0, 3(0.0)]
EndPoint[2]	[3(7000), 3(0.0)]

dwSpiralTurrns	0



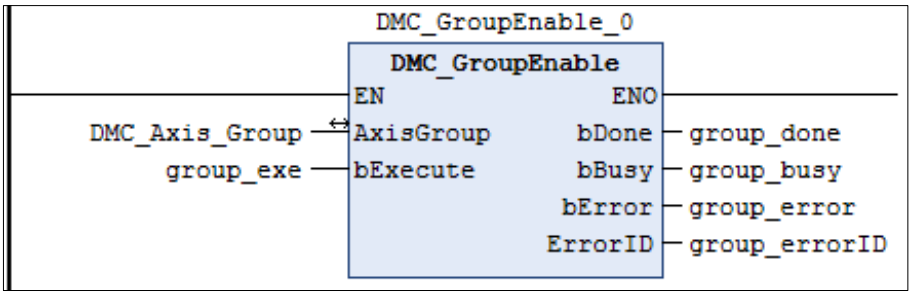
- **Troubleshooting**

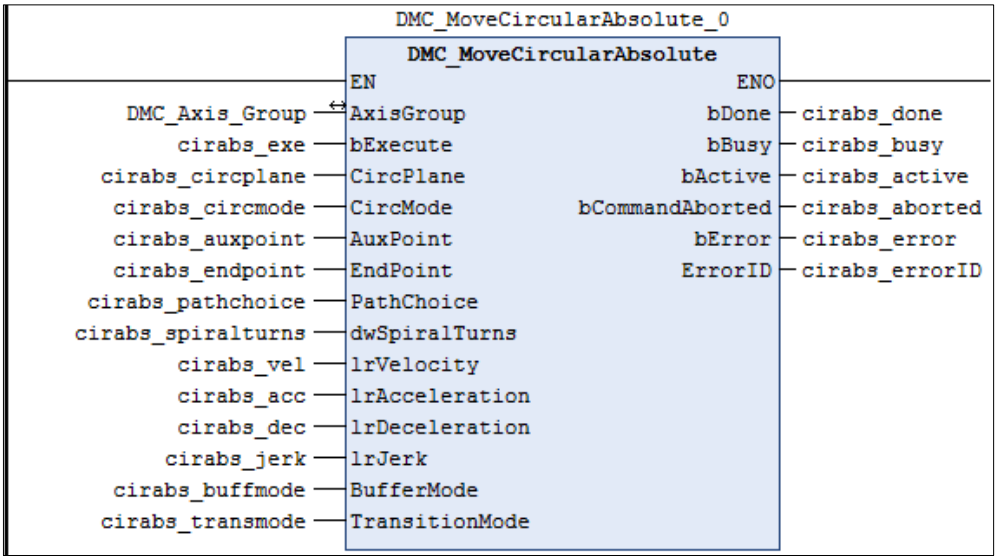
  - When an error occurs during the execution of instructions, *bError* changes to True and axes stops running. To confirm current error state, see the error code in *ErrorID*.
  - For error codes and corresponding troubleshootings, refer to **Appendix** for error code descriptions.

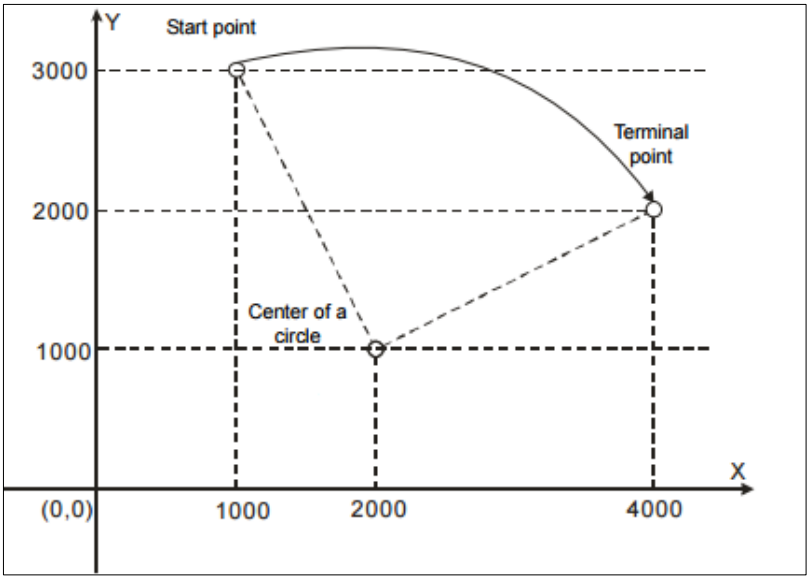
- **Programming Example**

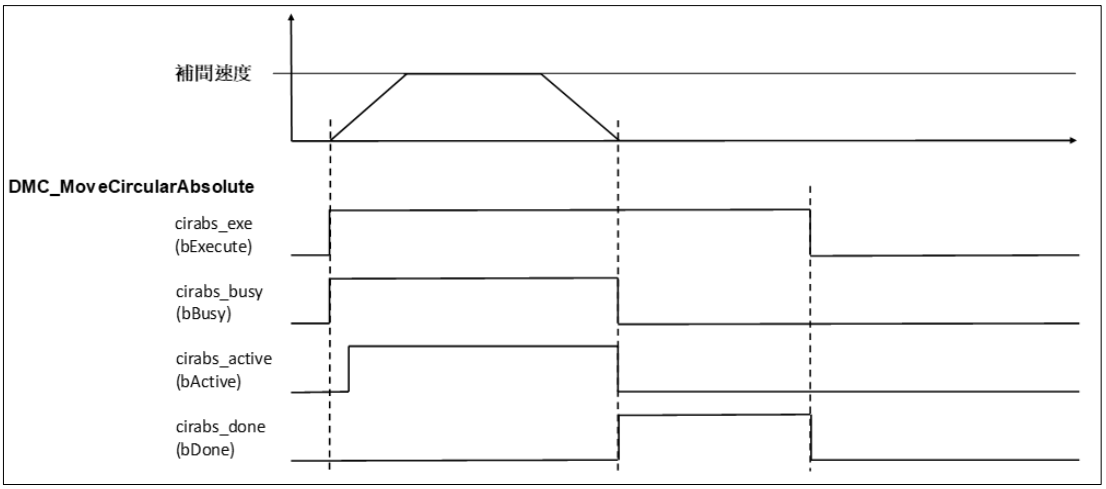
  - In this example, the instruction performs the circular interpolation from current positon (1000, 3000) until the absolute target position (4000, 2000) in the clockwise direction.

```
DMC_MoveCircularAbsolute_0: DMC_MoveCircularAbsolute;
cirabs_exe: BOOL;
cirabs_circplane: DMC_CIRC_PLANE := DMC_CIRC_PLANE.XY_plane;
cirabs_circmode: DMC_CIRC_MODE := DMC_CIRC_MODE.center;
cirabs_auxpoint: ARRAY [0..2] OF LREAL := [2000, 1000];
cirabs_endpoint: ARRAY [0..5] OF LREAL := [4000, 2000];
cirabs_pathchoice: DMC_CIRC_PATHCHOICE := DMC_CIRC_PATHCHOICE.CLOCKWISE;
cirabs_spiralturns: WORD := 0;
cirabs_vel: LREAL := 200;
cirabs_acc: LREAL := 100;
cirabs_dec: LREAL := 100;
cirabs_jerk: LREAL := 0;
cirabs_buffmode: DMC_BUFFER_MODE;
cirabs_transmode: DMC_GROUP_TRANSITION_MODE;
cirabs_done: BOOL;
cirabs_busy: BOOL;
cirabs_active: BOOL;
cirabs_aborted: BOOL;
cirabs_error: BOOL;
cirabs_errorID: DMC_ERROR;
```


DMC_GroupEnable_0
DMC_GroupEnable
EN
ENO
DMC_Axis_Group
AxisGroup
bDone
group_done
group_exe
bExecute
bBusy
group_busy
bError
group_error
ErrorID
group_errorID


DMC_MoveCircularAbsolute_0
DMC_MoveCircularAbsolute
EN
ENO
DMC_Axis_Group
AxisGroup
bDone
cirabs_done
cirabs_exe
bExecute
bBusy
cirabs_busy
cirabs_circplane
CircPlane
bActive
cirabs_active
cirabs_circmode
CircMode
bCommandAborted
cirabs_aborted
cirabs_auxpoint
AuxPoint
bError
cirabs_error
cirabs_endpoint
EndPoint
ErrorID
cirabs_errorID
cirabs_pathchoice
PathChoice
cirabs_spiralturns
dwSpiralTurns
cirabs_vel
lrVelocity
cirabs_acc
lrAcceleration
cirabs_dec
lrDeceleration
cirabs_jerk
lrJerk
cirabs_buffmode
BufferMode
cirabs_transmode
TransitionMode


Y
Start point
3000
Terminal point
2000
Center of a circle
1000
X
(0,0)
1000
2000
4000

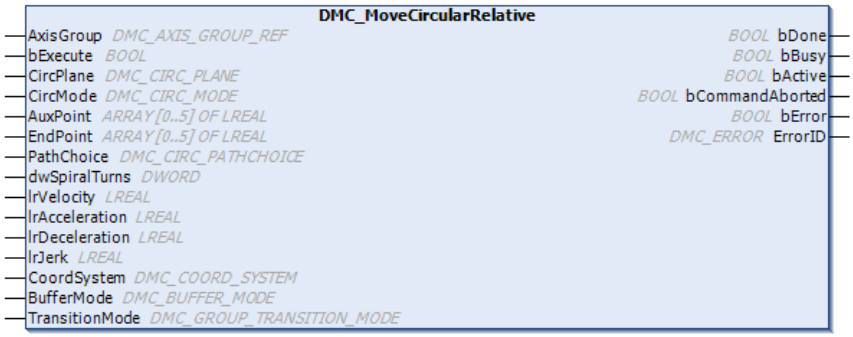


- When cirabs_exe (*bExecute*) changes to True, DMC_MoveCircularAbsolute performs the absolute positioning toward the terminal point (4000, 2000) from the start point (1000, 3000) in the clockwise direction.
- When cirabs_done (*bDone*) is True and cirabs_busy (*bBusy*) changes to False, which means the absolute target positioning (4000, 2000) is completed. When cirabs_exe (*bExecute*) is switched to False, cirabs_done (*bDone*) will change to False automatically.
- If cirabs_exe (*bExecute*) is set to True again, there will be no positioning motion any longer since the absolute target position has been reached

2.3.1.6 DMC_MoveCircularRelative

- **Supported Devices**: AX-308E, AX-364E

DMC_MoveCircularRelative controls the axis group to perform circular or helical interpolation for a specified relative target position.

FB/FC	Instruction	Graphic Expression
FB	DMC_MoveCircular Relative	

ST Language

```
DMC_MoveCircularRelative_instance(
AxisGroup: = ,
bExecute: = ,
CircPlane: = ,
CircMode: = ,
AuxPoint: = ,
EndPoint: = ,
PathChoice: = ,
dwSpiralTurns: = ,
lrVelocity: = ,
lrAcceleration: = ,
lrDeceleration: = ,
lrJerk: = ,
CoordSystem: =,
BufferMode: = ,
TransitionMode: = ,
bDone=> ,
bBusy=> ,
bActive=> ,
bCommandAborted=> ,
bError=> ,
ErrorID=> ) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
CircPlane	Specify the circular or helical plane. *1	DMC_CIRC_ PLANE	0: XY_plane 1: YZ_plane 2: ZX_plane 3: ARBITRARY_plane (0)	When *bExecute* turns to True.
CircMode	Specify the method for circular or helical interpolation. *1	DMC_CIRC_ MODE	0: Border 1: Center 2: Radius (0)	When *bExecute* turns to True.
AuxPoint	Specify the auxiliary point data. *1	LREAL[3]	[_, _, _] Positive, negative or 0 ([0, 0, 0])	When *bExecute* turns to True.
EndPoint	Specify the target position for each axis in the axis group. (Unit: user unit)	LREAL[6]	[_, _, _, _, _, _] Positive, negative or 0 ([0, 0, 0, 0, 0, 0])	When *bExecute* turns to True.
PathChoice	Specify the circular or helical interpolation direction.	DMC_CIRC_ PATHCHOICE	0: Clockwise 1: CounterClockwise (0)	When *bExecute* turns to True.
dwSpiralTurns	Specify the number of spiral turns.	DWORD	0~65535 (0)	When *bExecute* turns to True.
lrVelocity	Specify the target velocity for the axis group. (Unit: user unit/s)	LREAL	Positive (0)	When *bExecute* turns to True.
lrAcceleration	Specify the acceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True.
lrDeceleration	Specify the deceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True.
lrJerk	Specify the jerk. (Unit: user unit/s^3)	LREAL	Positive (0)	When *bExecute* turns to True.
CoordSystem	Coordinate system	DMC_COORD_SYSTEM*1	0: ACS 1: MCS 2: WCS (Reserved) 3: PCS_1	When *bExecute* is on the rising edge, the setting parameters of CoordSystem will be updated.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
			(Reserved) 4: PCS_2 (Reserved) 5: TCS (Reserved) (1)	
BufferMode	Specify a buffer mode for the instruction.*2	DMC_BUFFER_MODE	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	When *bExecute* turns to True.
TransitionMode	Specify a transition mode for the instruction.*3	DMC_GROUP_TRANSITION_MODE	0: None 10: Overlap (0)	When *bExecute* turns to True.

***Note:**

1. Setting parameters *CircPlane*, *CircMode* and *AuxPoint*.

***CircMode* setting**	**Definition CircMode – AuxPoint**	***CircPlane***		
		XY_Plane	**YZ_Plane**	**ZX_Plane**
		Actual input for *AuxPoint* [_, _, _]		
0	Three points – Relative coordinate values for the border point (X_A, Y_A, Z_A)	Start point, end point and border point [X_A, Y_A, Z_A]		
1	A center point – Relative coordinate values for the center point (C_X, C_Y)	[C_X, C_Y, N/A]	[N/A, C_X, C_Y]	[C_Y, N/A, C_X]
2	Radius – Radius (R)	[R, N/A, N/A]		

2. Refer to *AX-3 Series Operation Manual* for details on BufferMode.
3. Refer to *AX-3 Series Operation Manual* for details on TransitionMode.

- **Outputs**

Name	Function	Data Type	Output Range Value (Default Value)
bDone	True when the relative positioning is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bActive	True when the instruction is controlling axes.	BOOL	True/False (False)
bCommandAborted	True when the instruction execution is aborted.	BOOL	True/False (False)

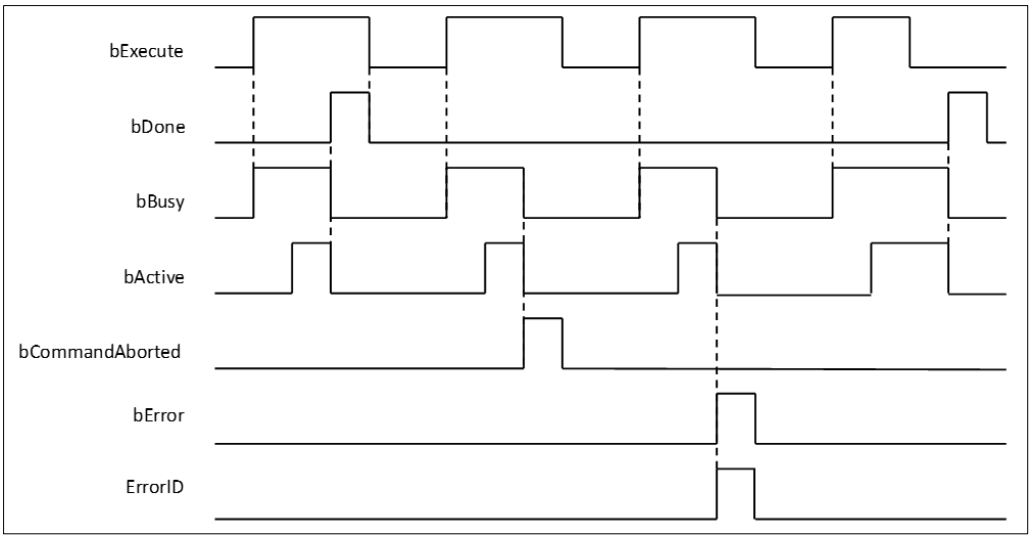
Name	Function	Data Type	Output Range Value (Default Value)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to **Appendix** for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note:** DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the relative positioning is completed.	• When *bExecute* turns to False. • *bDone* will change to False after remaining True for one cycle when *bExecute* is False but *bDone* changes to True.
bBusy	• When *bExecute* changes to TRUE.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bActive	• When axes start being controlled by the instruction.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True. • *bActive* will change to False after remaining True for at least one cycle when *bExecute* is False but *bActive* changes to True.
bCommand Aborted	• When the instruction is interrupted by another instruction *BufferMode* of which is set to *Aborting*. • When the instruction is interrupted by MC_Stop. • When the instruction is interrupted by DMC_GroupStop.	• When *bExecute* turns to False. • *bCommandAborted* will change to False after remaining True for at least one cycle when *bExecute* is False but *bCommandAborted* changes to True.
bError/ErrorID	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	• When *bExecute* turns from True to False. (Error Code is cleared)

- **Timing Diagram**



- **Inputs/Outputs**

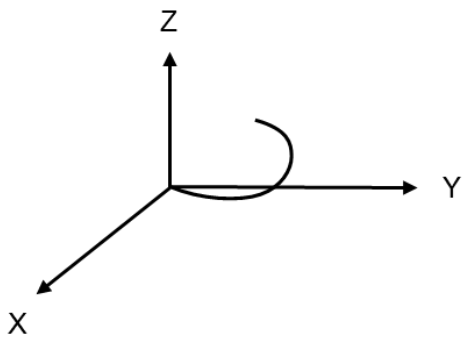
Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	When *bExecute* turns to True and *bBusy* is False.

***Note:**

DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**
  - The instruction supports the relative helical interpolation of maximum three axes, where the three axes can simultaneously start, stop as well as reach the specified relative target position.
  - The instruction can be used to specify the circle drawing on the plane parallel to XY / YZ / ZX and set the height of the helix on Z / X / Y axis.
  - If the start point and end point for circular interpolation are set to the same point, use the center point mode (*CircMode* = Center) for the interpolation.
  - At least two axes are needed for circular interpolation. An error will occur if there is a travel distance for an axis which is not set.
  - When the start point and end point for circular interpolation are set to the same point, the instruction will perform the rotation for one complete circle.
  - The function of CoordSystem needs to be supported by DL_MotionControl library V1.1.0.0 or above.
  - CircPlane added ARBITRARY_plane in DL_MotionControl library V1.2.0.0 or later.
  - ARBITRARY_plane can draw ellipses in space.
    - Use the following parameters to draw the ellipse.

Name	Setting Value
CircPlane	DMC_CIRC_PLANE.ARBITRARY_plane
CircMode	DMC_CIRC_MODE.border
AuxPoint	[1000, 2000, 0, 3(0.0)]
EndPoint[2]	[3(7000), 3(0.0)]
dwSpiralTurrns	0



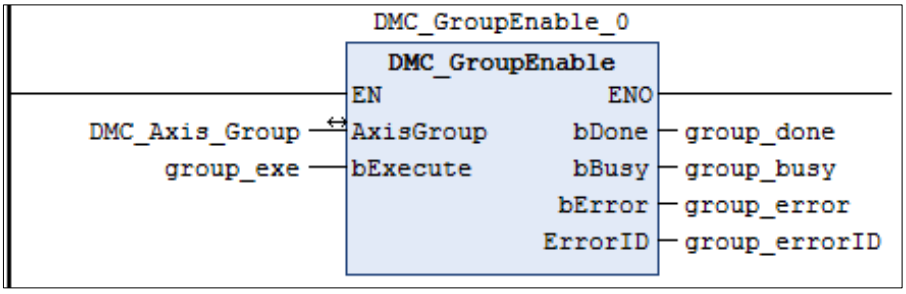
- **Troubleshooting**

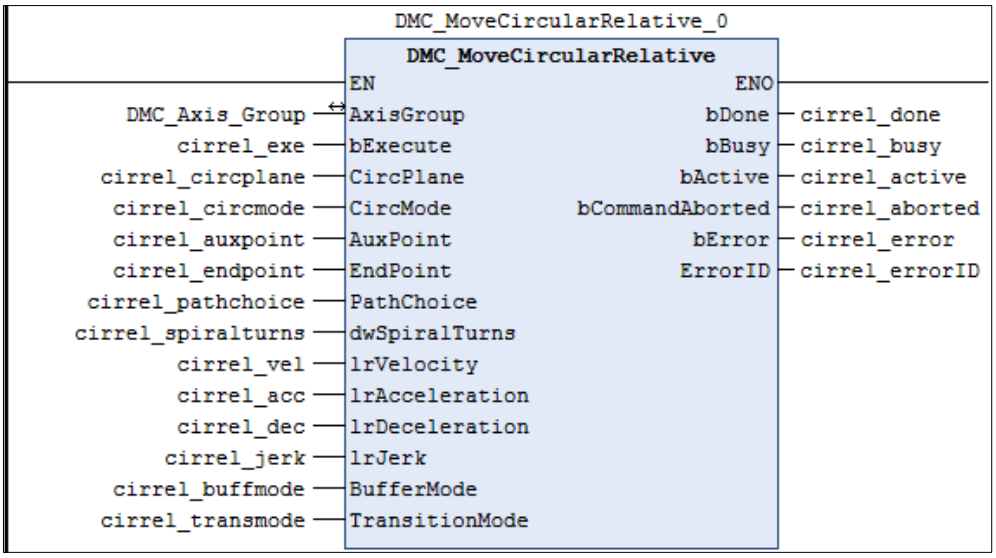
  - When an error occurs during the execution of the instruction, *bError* changes to True and axes stops running. To confirm current error state, see the error code in *ErrorID*.
  - For error codes and corresponding trouble shootings, refer to **Appendix** for error code descriptions.

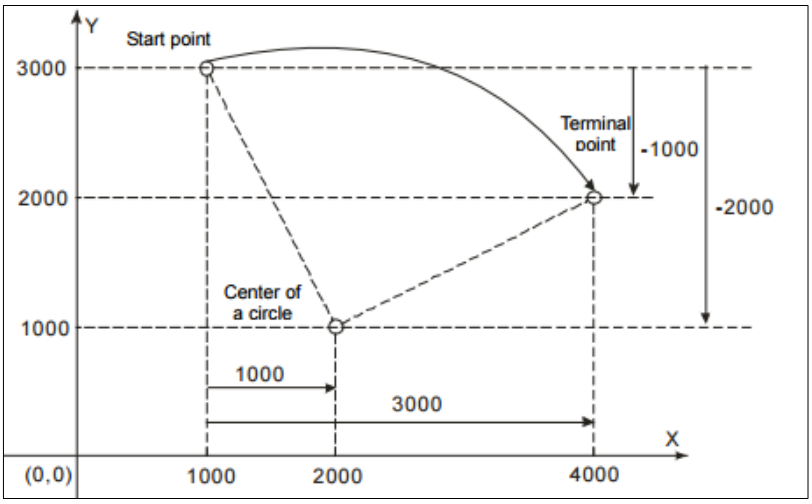
- **Programming Example**

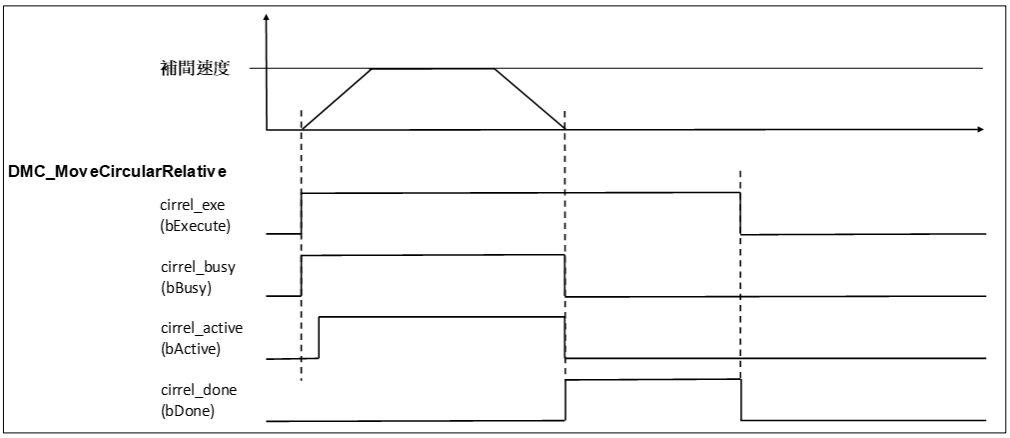
  - In this example, the instruction performs the circular interpolation from current positon (1000, 3000) until the target position (4000, 2000) in the clockwise direction.

```
DMC_MoveCircularRelative_0: DMC_MoveCircularRelative;
cirrel_exe: BOOL;
cirrel_circmode: DMC_CIRC_MODE := DMC_CIRC_MODE.center;
cirrel_auxpoint: ARRAY [0..2] OF LREAL := [1000, -2000];
cirrel_endpoint: ARRAY [0..5] OF LREAL := [3000, -1000];
cirrel_pathchoice: DMC_CIRC_PATHCHOICE := DMC_CIRC_PATHCHOICE.CLOCKWISE;
cirrel_spiralturns: WORD := 0;
cirrel_vel: LREAL := 200;
cirrel_acc: LREAL := 100;
cirrel_dec: LREAL := 100;
cirrel_jerk: LREAL := 0;
cirrel_buffmode: DMC_BUFFER_MODE;
cirrel_transmode: DMC_GROUP_TRANSITION_MODE;
cirrel_done: BOOL;
cirrel_busy: BOOL;
cirrel_active: BOOL;
cirrel_aborted: BOOL;
cirrel_error: BOOL;
cirrel_errorID: DMC_ERROR;
```







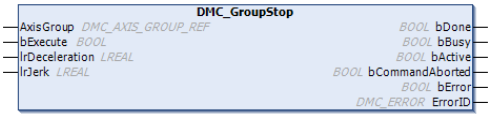


- When cirrel_exe (*bExecute*) changes to True, DMC_MoveCircularRelative performs the relative positioning toward the terminal point (4000, 2000) from the start point (1000, 3000) in the clockwise direction.
- When cirrel_done (*bDone*) is True and cirrel_busy (*bBusy*) changes to False, which means the relative target positioning (4000, 2000) is completed. When cirrel_exe (*bExecute*) is switched to False, cirrel_done (*bDone*) will change to False automatically.
- If cirrel_exe (*bExecute*) is set to True again, the instruction will perform the circular interpolation regarding current positon (4000, 2000) as the reference point.

2.3.1.7 DMC_GroupStop

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupStop decelerates the group axes to a stop.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupStop		DMC_GroupStop_instance (AxisGroup : =, bExecute : =, lrDeceleration : =, lrJerk : =, bDone =>, bBusy =>, bActive =>, bCommandAborted =>, bError =>, ErrorID =>) ;

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
lrDeceleration	Specify the deceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True.
lrJerk	Specify the jerk. (Unit: user unit/s^3)	LREAL	Positive (0)	When *bExecute* turns to True.

- **Outputs**

Name	Function	Data Type	Output Range Value (Default Value)
bDone	True when all axes stop with the velocity 0.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bActive	True when the instruction is controlling axes.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)

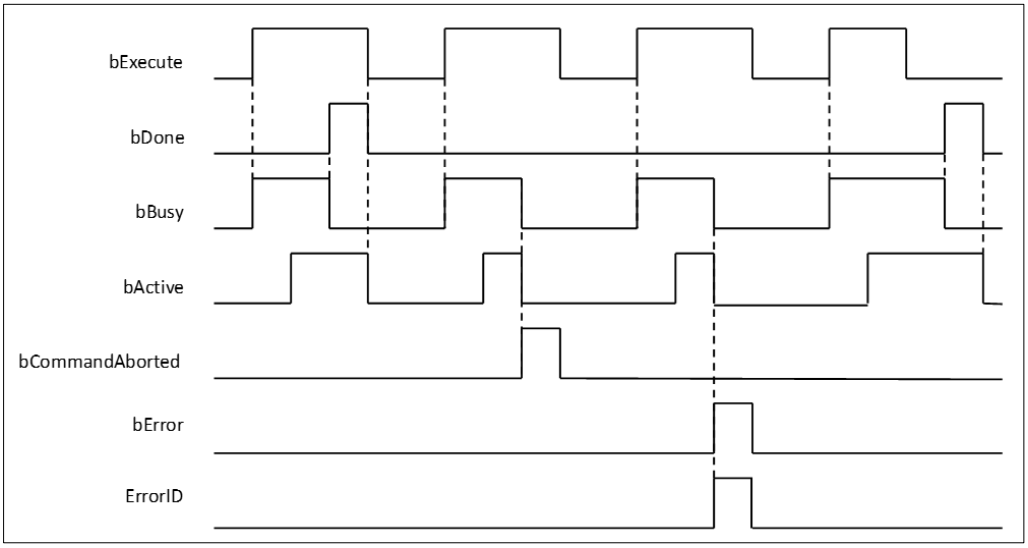
Name	Function	Data Type	Output Range Value (Default Value)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to **Appendix** for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

*Note: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the axis group decelerates to a stop.	• When *bExecute* turns to False. • *bExecute* will change to False after remaining True for one cycle when *bExecute* is False but *bDone* changes to True.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bActive	• When axes start being controlled by the instruction.	• When *bExecute* turns to False. • When *bError* turns to True. • When *bCommandAborted* turns to True. • *bActive* will change to False after remaining True for at least one cycle if *bExecute* changes to False but *bActive* changes to True.
bCommandAborted	• When the instruction is interrupted by another instruction.	• When *bExecute* turns to False. • *bCommandAborted* will change to False after remaining True for one cycle when *bExecute* is False but *bCommandAborted* changes to True.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	• When *bExecute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	When *bExecute* turns to True and *bBusy* is False.

***Note:**

DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

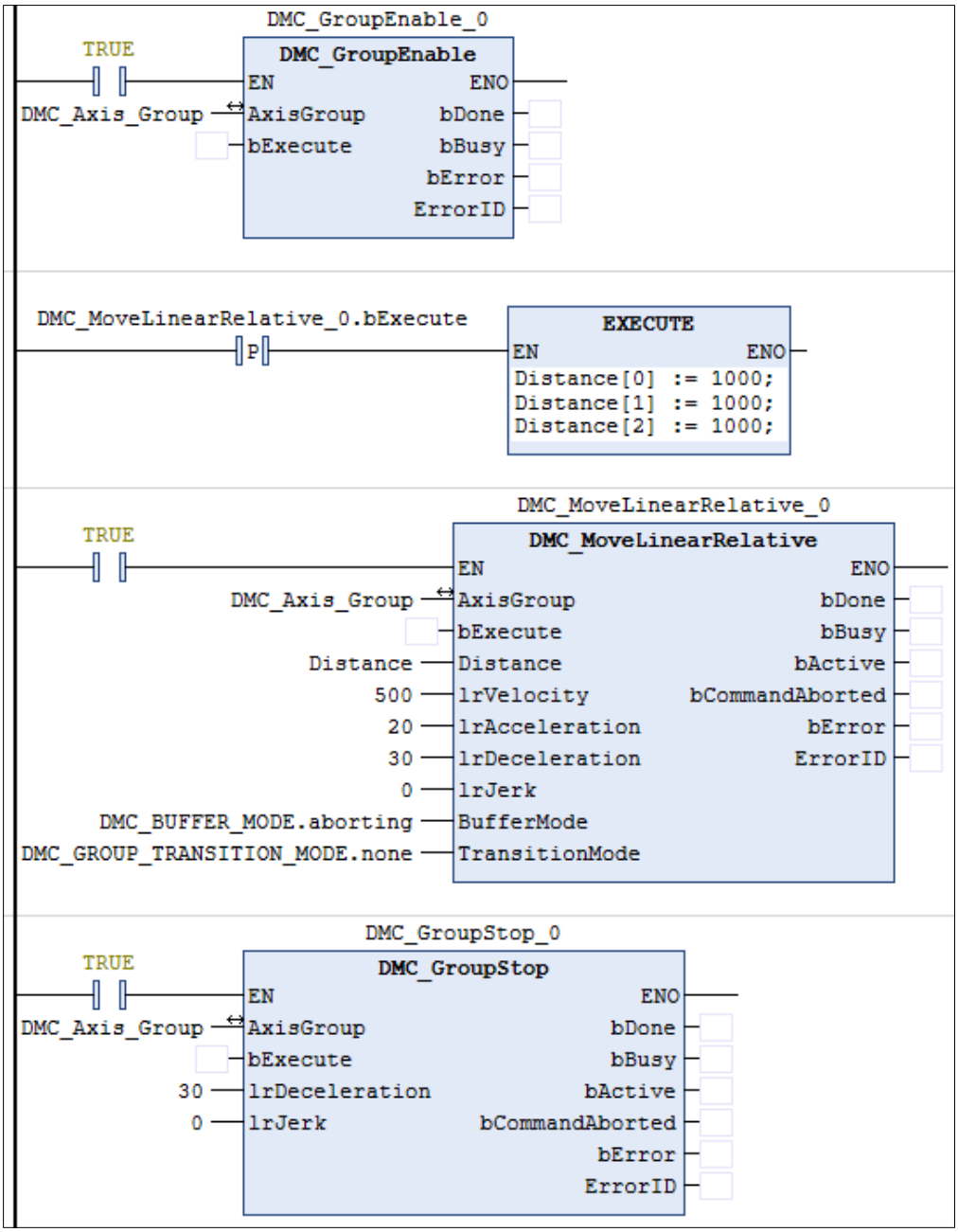
  - The instruction decelerates the group axes in motion to a stop.
  - The axis group state is switched to GroupStopping via the instruction.
  - The axis group state GroupStopping will continue until *bExecute* changes to False. *bDone* changes to True when the velocity 0 is reached.

- **Troubleshooting**

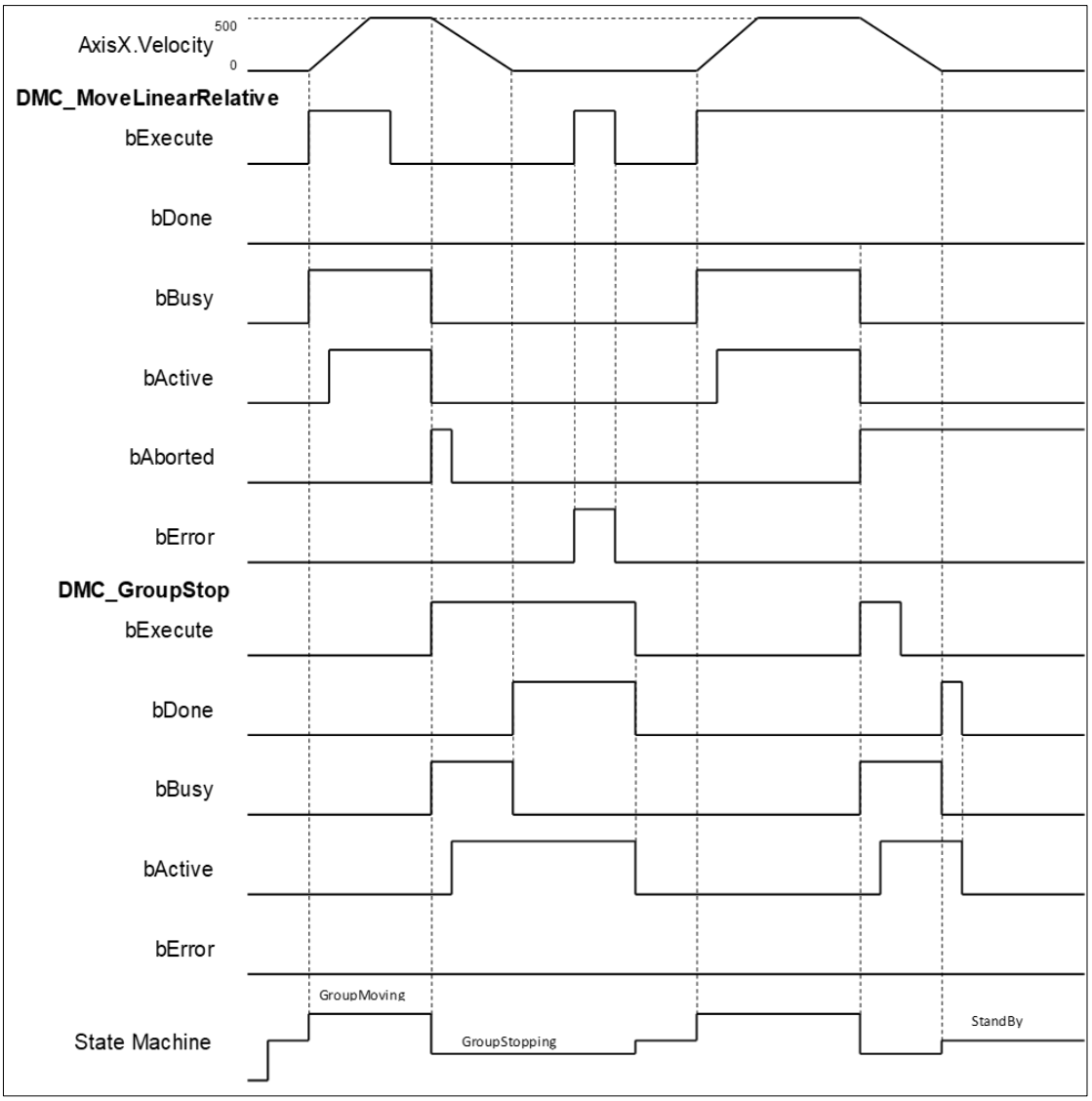
  - When an error occurs during the execution of the instruction or the axis group enters “Errorstop” state, *bError* changes to True and the axes stops running. To confirm the current error state, see the error code in *ErrorID*.

- **Programming Example**

  - This example shows the motion behavior which is performed by DMC_GroupStop during the execution of DMC_MoveLinearRelative.
  - When the execution of DMC_GroupStop is completed, the axis group enters GroupStandby state.



- **Timing Diagram**

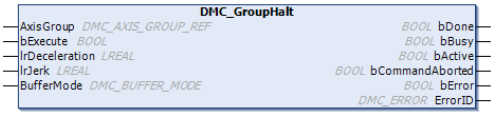


- When *bExecute* of DMC_GroupStop changes to True, *bCommandAboted* of MoveLinearRelative changes to True and axes start to decelerate to a stop. Meanwhile the axis group stays in GroupStopping state.
- When the velocities of axes reach 0, *bDone* of DMC_GroupStop changes to True and the axis group holds GroupStopping state.
- When *bExecute* of DMC_GroupStop changes to False, the state of axes changes from GroupStopping into StandBy.

2.3.1.8 DMC_GroupHalt

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupHalt decelerates the axis group in motion to a pause.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupHalt		DMC_GroupHalt_instance (AxisGroup : =, bExecute : =, lrDeceleration : =, lrJerk : =, BufferMode : =, bDone =>, bBusy =>, bActive =>, bCommandAborted =>, bError =>, ErrorID =>) ;

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
lrDeceleration	Specify the deceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True.
lrJerk	Specify the jerk. (Unit: user unit/s^3)	LREAL	Positive (0)	When *bExecute* turns to True.
BufferMode	Specify a buffer mode for the instruction.*	DMC_BUFFER_MODE	0: Aborting 1: Buffered (0)	When *bExecute* turns to True.

***Note**: Refer to *AX-3 Series Operation Manual* for details on BufferMode.

- **Outputs**

Name	Function	Data Type	Output range (Default Value)
bDone	True when all axes stop with the velocity 0.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bActive	True when the instruction is controlling axes.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error	DMC_ERROR*	DMC_ERROR

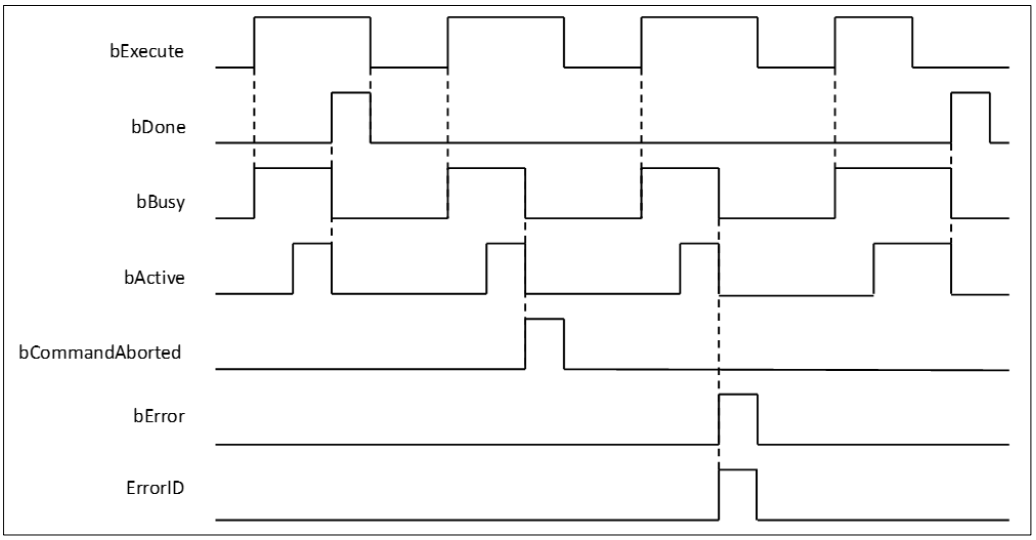
Name	Function	Data Type	Output range (Default Value)
	occurs. Refer to **Appendix** for error code descriptions.		(DMC_NO_ERROR)

*__Note__: DMC_ERROR: Enumeration (ENUM)

■ **Output Update Timing**

Name	Function	Data Type
bDone	• When the axis group decelerates to a stop.	• When *bExecute* turns to False. • *bDone* will change to False after remaining True for one cycle when *bExecute* changes to False but *bDone* changes to True.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bActive	• When axes start being controlled by the instruction.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True. • *bActive* will change to False after remaining True for at least one cycle when *bExecute* changes to False but *bActive* changes to True.
bCommandAborted	• When the instruction is interrupted by another instruction.	• When *bExecute* turns to False. • *bCommandAborted* will change to False after remaining True for one cycle when *bExecute* is False but *bCommandAborted* changes to True.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	• When *bExecute* turns from True to False. (Error code is cleared.)
ErrorID		

■ **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF	DMC_AXIS_GROUP_REF	When *bExecute* turns to True and *bBusy* is False.

***Note:**

AxisGroup_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

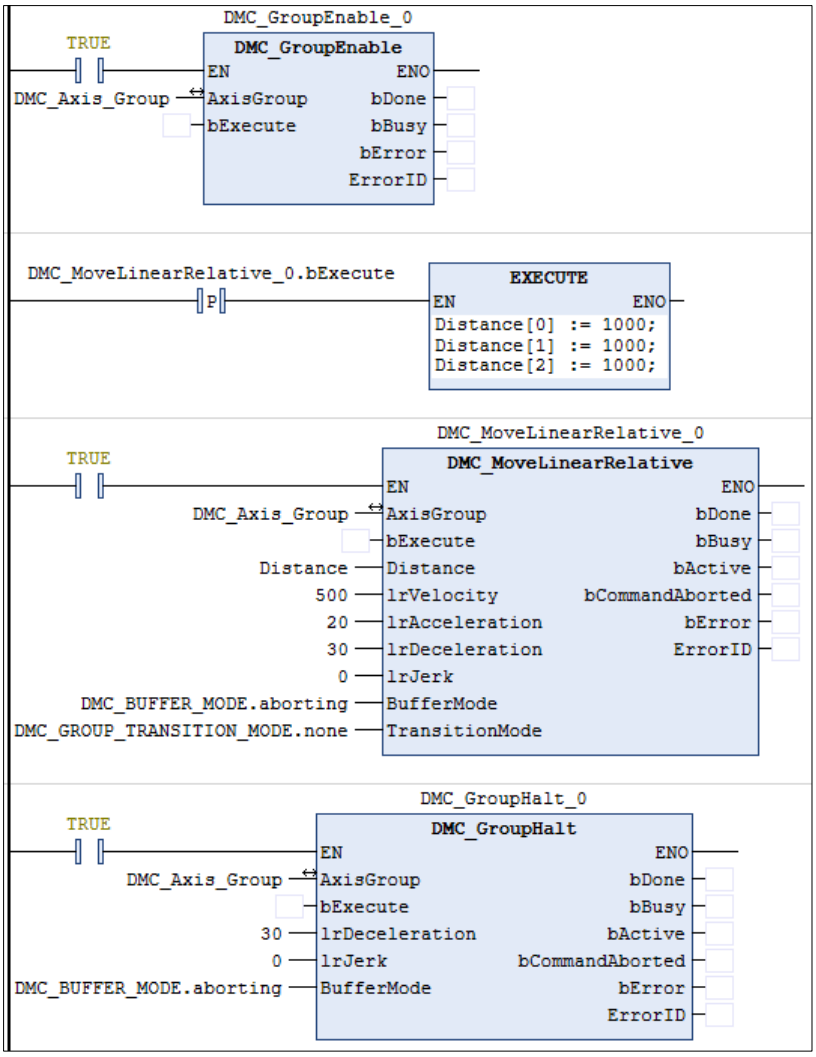
  - The instruction decelerates the group axes in motion to a pause.
  - The axis group enters the state of GroupMoving via the instruction.
  - When the velocity 0 is reached, *bDone* changes to True and the axis group changes to StandBy state.
  - BufferMode of DMC_GroupHalt only supports 0: Aborting and 1: Buffered. An error will occur with DMC_ERROR.DMC_GM_INVALID_BUFFER_MODE if other BufferMode is used.

- **Troubleshooting**

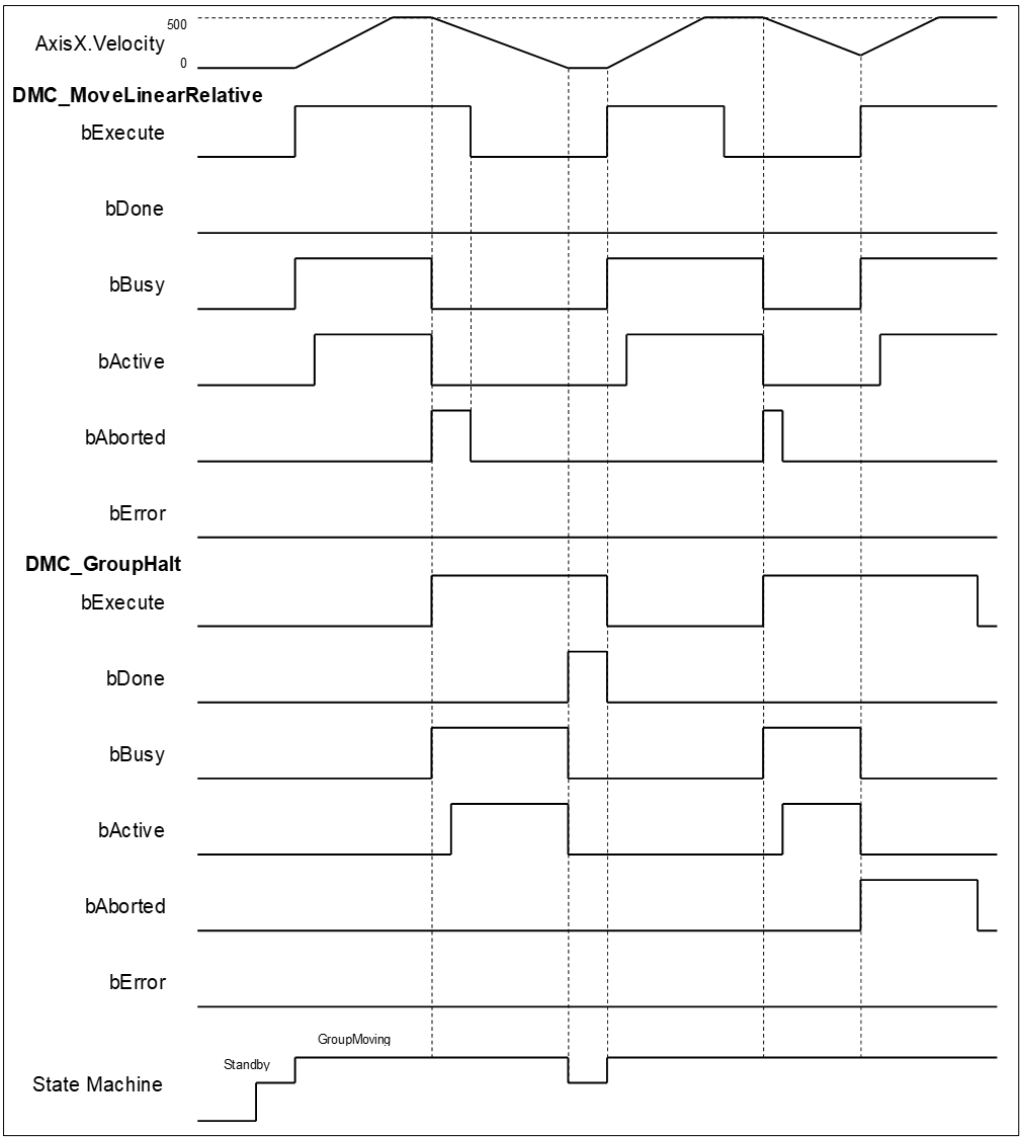
  - When an error occurs during the execution of the instruction or the axis group enters “Errorstop” state, *bError* changes to True and the axes stops running. To confirm the current error state, see the error code in *ErrorID*.

- **Programming Example**

  - This example shows the motion behavior which is performed by DMC_GroupHalt after DMC_MoveLinearRelative is run.
  - The axes will enter Standby state after deceleration is completed if no other motion instruction is run during the period when DMC_MoveLinearRelative is paused via DMC_GroupHalt.
  - When DMC_MoveLinearRelative is run again during deceleration, DMC_GroupHalt will be interrupted immediately and the axis group will accelerate again without staying in Standby state any more. The re-execution action described above is allowed for DMC_GroupHalt.



- **Timing Diagram**

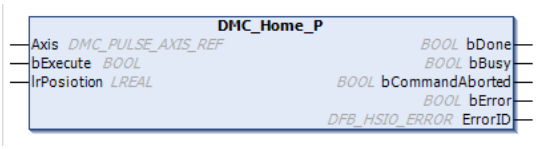


- When *bExecute* of DMC_GroupHalt changes to True, *bCommandAboted* of DMC_MoveLinearRelative changes to True and the axes start to decelerate to a stop. And the axis group stays in GroupMoving state.
- When the velocity 0 is reached, *bDone* of DMC_GroupHalt changes to True and the axis group changes to Standby state.
- When the velocity has not been reduced to 0 yet and *bExecute of* DMC_GroupHalt changes to True during the instruction execution, DMC_GroupHalt will be aborted by changing *bExecute* of DMC_MoveLinearRelative to True again and then its *bCommandAboted* will change to True.

2.3.1.9 DMC_Home_P

- **Supported Devices**: AX-308E, AX-364E

DMC_Home_P, an application function block of pulse output, drives the pulse axis to perform the homing in the set mode.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_Home_P		DMC_Home_P_instance (Axis : =, bExecute : =, lrPosition : =, bDone =>, bBusy =>, bCommandAborted =>, bError =>, ErrorID =>) ;

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
lrPosition	Specify the position after the homing is completed.	LREAL	Positive, negative or 0 (0)	When *bExecute* turns to True and *bBusy* is False.

- **Outputs**

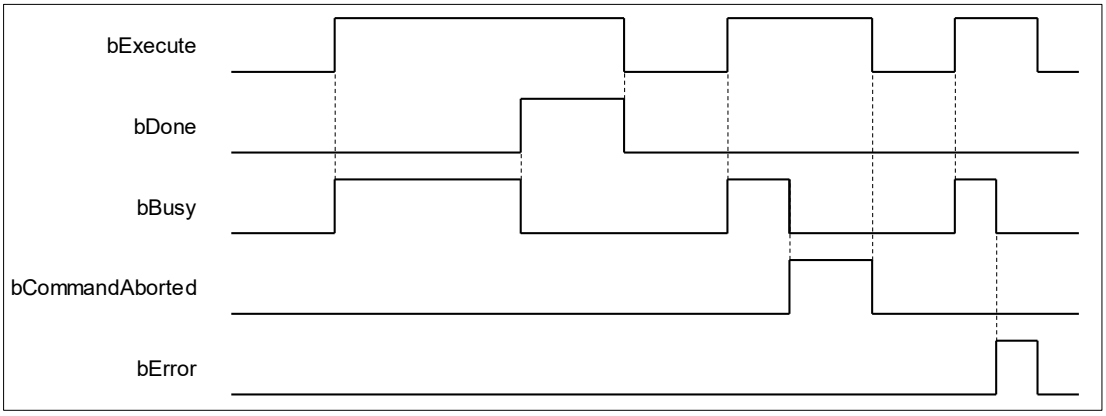
Name	Function	Data Type	Output range (Default Value)
bDone	True when the homing is completed.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted by another instruction.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to **Appendix** for error code descriptions.	DFB_HSIO_ERROR*	DFB_HSIO_ERROR (DFB_HSIO_NO_ERR)

***Note:** DFB_HSIO_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Function	Data Type
bDone	• When the homing is completed.	• When *bExecute* turns to False. • When *bError* turns to True.
bBusy	• When *bExecute* changes to TRUE.	• When *bExecute* turns to False. • When *bError* turns to True.
bCommandAborted	• When the instruction is aborted by MC_Stop.	• When *bExecute* turns to False. • *bCommandAborted* will change to False after remaining True for one cycle when *bExecute* is False but *bCommandAborted* changes to True.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	• When *bExecute* turns from True to False. (Error code is cleared.)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specifies the source of pulse output axis	DMC_PULSE_AXIS_REF (FB) *	DMC_PULSE_AXIS_REF	When *bExecute* turns to True and *bBusy* is False.

***Note:**

DMC_PULSE_AXIS_REF (FB): The function block serves as the drive interface for the pulse output axis, which contains the axis parameter call and the drive program.

- **Function**

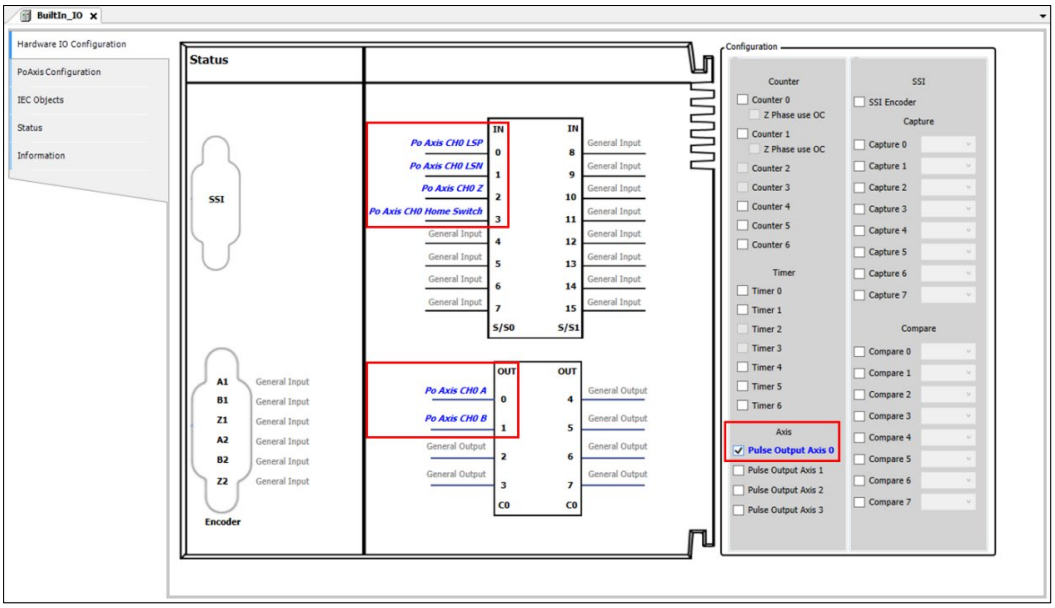
  - The pulse output axis specified by the instruction must be selected in **Hardware IO Configuration** so that the axis can output pulses and perform the homing action according to the pulse axis settings i.e. homing mode, acceleration rate and velocity.
  - The instruction can be used only when the pulse output axis is in Standstill state. An error will occur if the instruction is run in other axis state.
  - DMC_Home_P supports homing modes defined in CiA 402 protocol. For details on homing modes, refer to appendices.
  - Library of this function block: DL_BuiltInIO_AX3.library.

- **Troubleshooting**

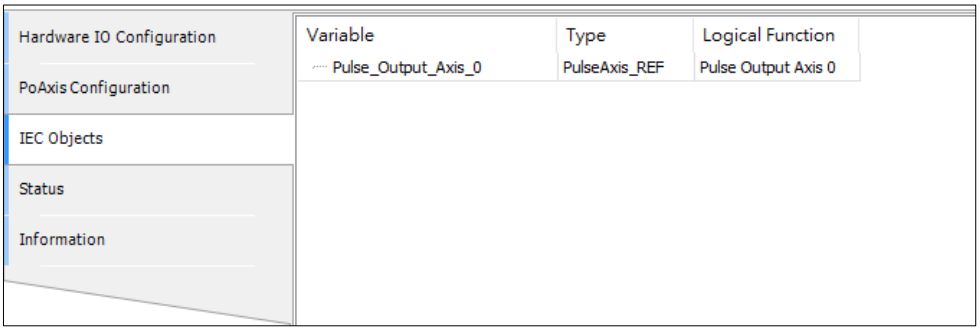
  - When an error occurs in the instruction execution, *bError* of the instruction changes to True. To confirm current error state, see the error code in *ErrorID*.

- **Programming Example**

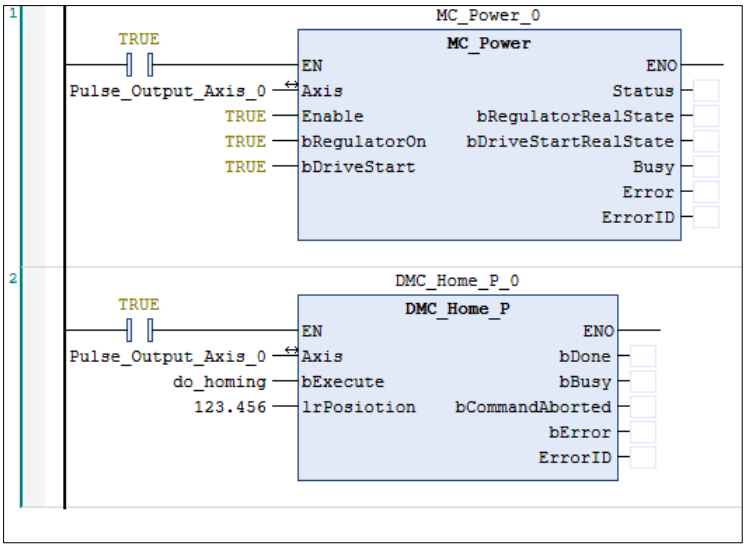
  - In this example, the pulse output axis performs the homing motion via DMC_Home_P after the axis is configured in the IO configuration interface.
  - Select the first pulse output axis (Pulse Output Axis 0) in Hardware IO Configuration of BuiltIn_IO as below. Then you can see corresponding output points (e.g. OUT0, OUT1) and signal trigger points for pulse output (e.g. IN0, IN1, IN2 and IN3) from the software. The homing motion cannot be performed until the signal trigger points for the homing mode have been configured to corresponding input signal sources.



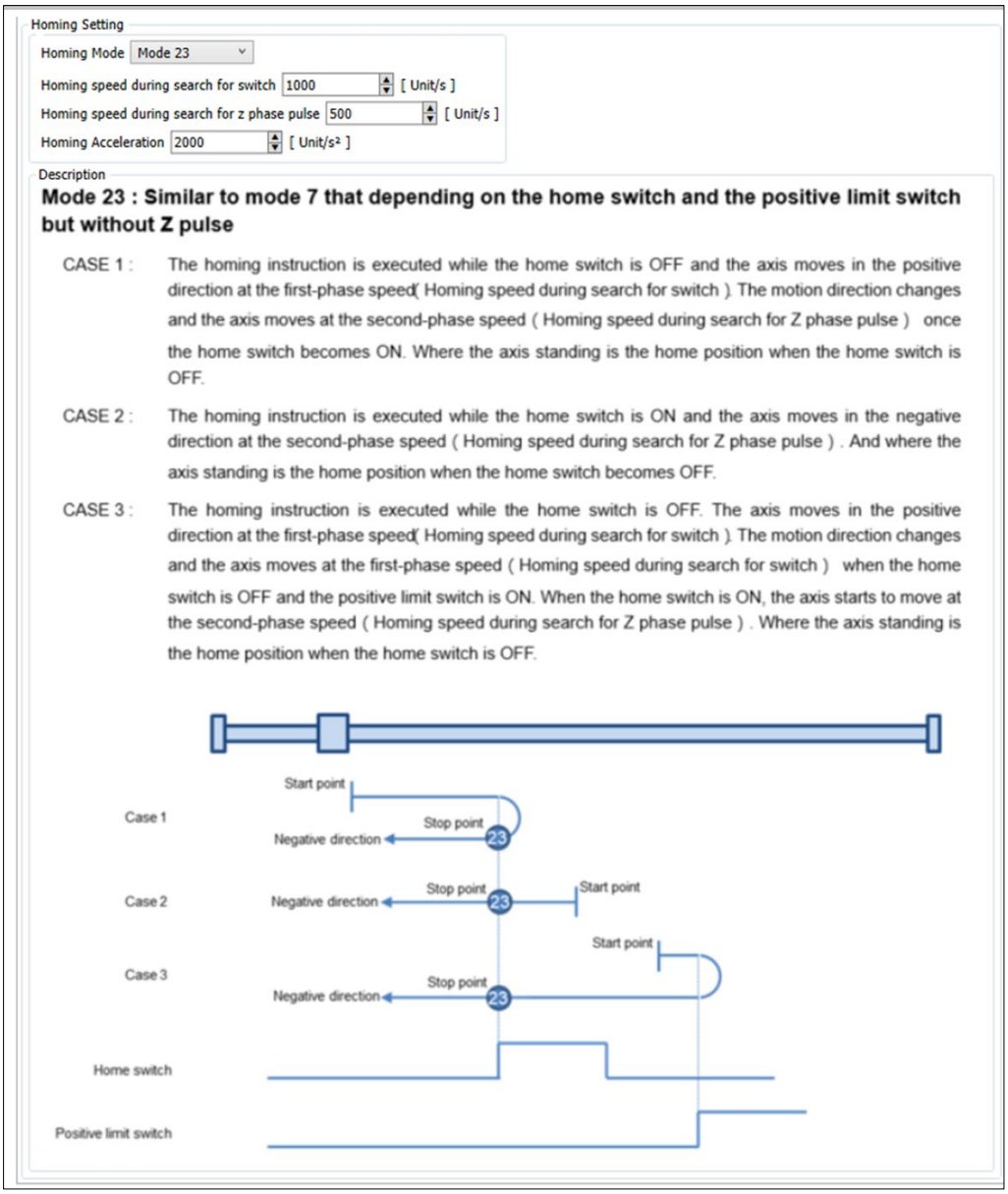
  - After the configuration of the pulse output axis, the variable Pulse_Output_Axis_0 configured in IEC Objects can be taken out as a Data Type to any function block, as shown below.



  - Pulse_Output_Axis_0 is connected to the input *Axis* of MC_ Power and DMC_ Home_P as shown in the figure below. When the axis is in Standstill state, the instruction has started to perform the homing motion according to the set homing mode. At the moment, the state machine will switch the state from Standstill to Homing.



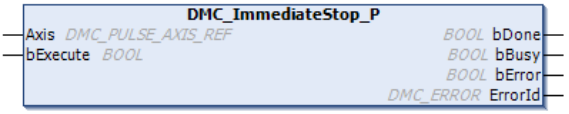
- After DMC_Home_P has started, the pulse axis Pulse_Output_Axis_0 will perform the homing motion according to the set Home Mode in PoAxis Configuration below. After the function block is run, the homing will be conducted according to different external signals and cases.
  - Homing Mode: Mode 23;
  - Homing speed during search for switch: 1000 (Unit: user unit /s);
  - Homing speed during search for z phase pulse: 500 (Unit: user unit /s);
  - Homing Acceleration: 2000 (Unit: user unit /s ^{2}).


Homing Setting
Homing Mode Mode 23
Homing speed during search for switch 1000 [Unit/s]
Homing speed during search for z phase pulse 500 [Unit/s]
Homing Acceleration 2000 [Unit/s²]
Description
Mode 23 : Similar to mode 7 that depending on the home switch and the positive limit switch but without Z pulse
CASE 1 : The homing instruction is executed while the home switch is OFF and the axis moves in the positive direction at the first-phase speed(Homing speed during search for switch). The motion direction changes and the axis moves at the second-phase speed (Homing speed during search for Z phase pulse) once the home switch becomes ON. Where the axis standing is the home position when the home switch is OFF.
CASE 2 : The homing instruction is executed while the home switch is ON and the axis moves in the negative direction at the second-phase speed (Homing speed during search for Z phase pulse) . And where the axis standing is the home position when the home switch becomes OFF.
CASE 3 : The homing instruction is executed while the home switch is OFF. The axis moves in the positive direction at the first-phase speed(Homing speed during search for switch). The motion direction changes and the axis moves at the first-phase speed (Homing speed during search for switch) when the home switch is OFF and the positive limit switch is ON. When the home switch is ON, the axis starts to move at the second-phase speed (Homing speed during search for Z phase pulse) . Where the axis standing is the home position when the home switch is OFF.
Start point
Case 1
Stop point
Negative direction
23
Case 2
Negative direction
Stop point
23
Start point
Start point
Case 3
Stop point
Negative direction
23
Home switch
Positive limit switch

2.3.1.10 DMC_ImmediateStop_P

- **Supported Devices**: AX-308E, AX-364E

DMC_ImmediateStop_P can stop the PO axis motion immediately and stop the pulse output.

FB/FC	Instruction	Graphic Expression
FB	DMC_ImmediateStop_P	

ST Language

```
DMC_ImmediateStop_P(
Axis : =,
bExecute : =,
bDone =>,
bBusy =>,
bError =>,
ErrorId =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is enabled when *bExecute* changes from False to True.	BOOL	True/False (False)	-

- **Outputs**

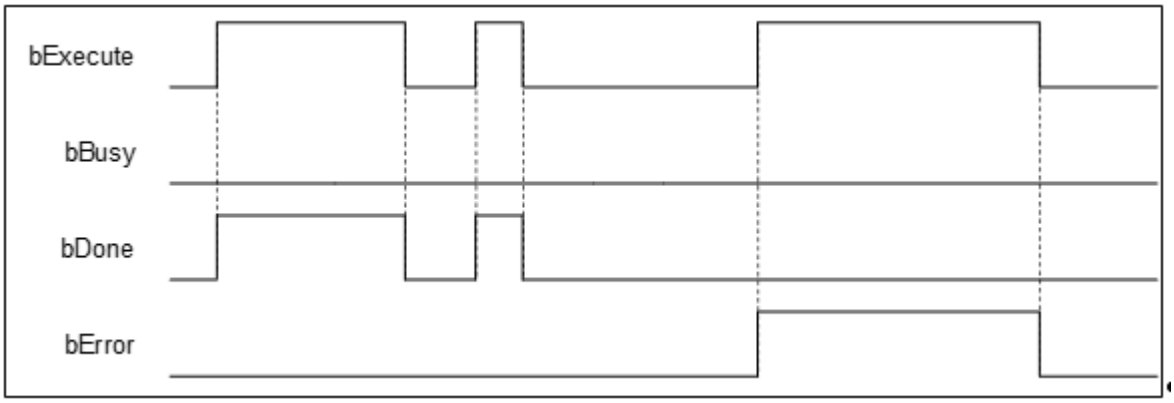
Name	Function	Data Type	Output range (Default Value)
bDone	True when the instruction execution is complete.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to execute.	BOOL	True/False (False)
bError	True when an instruction error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERR)

***Note:** DMC_ERROR: Enumeration (Enum)

  - **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the instruction execution is complete.	• When *bEexcute* turns to False. • *bDone* will change to False after remaining True for one period when *bExecute* is False but *bDone* changes to True.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded in ErrorID)	• When *bExecute* turns from True to False (Error Code is cleared).
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	DMC_PULSE_AXIS_REF *1	DMC_PULSE_AXIS_REF	When *bExecute* turns to True, and *bBusy* is False.

***Note:** DMC_PULSE_AXIS_REF (FB): The function block serves as the drive interface for the pulse output axis, which contains the axis parameter call and the drive program.

- **Function**

  - When *bExecute* turns to True, PO axis motion will be stopped immediately, and pulse output will be stopped immediately without deceleration.
  - Library of this function block is DL_BuiltInIO_AX3.library.

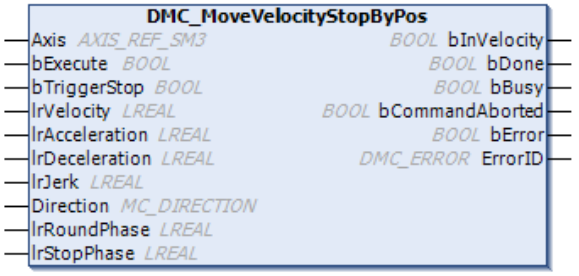
- **Troubleshooting**

  - If an error occurs during instruction execution and output pin bError changes to True, refer to *ErrorID* (Error Code) to address the problem.
  - For error codes and corresponding troubleshooting, refer to Appendix of this manual.

2.3.1.11 DMC_MoveVelocityStopByPos

- **Supported Devices**: AX-308E, AX-364E

DMC_MoveVelocityStopByPos controls an axis to stop at a specified position after a period of motion.

FB/FC	Instruction	Graphic Expression
FB	DMC_MoveVelocityStopByPos	

ST Language

```
DMC_MoveVelocityStopByPos_instance(
Axis : =,
bExecute : =,
bTriggerStop : =,
lrVelocity : =,
lrAcceleration : =,
lrDeceleration : =,
lrJerk : =,
Direction : =,
lrRoundPhase: =,
lrStopPhase: =,
bInVelocity =>,
bDone =>,
bCommandAborted =>,
bBusy =>,
bError =>,
ErrorID =>)
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
bTriggerStop	The stop command is run when *bExecute* is True.	BOOL	True/False (False)	When *bExecute* is True and the output *bBusy* is True.
lrVelocity	Specify the target velocity. (Unit: User unit/s)	LREAL	Positive or 0 (0)	When *bExecute* is True and the output *bBusy* is False.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrAcceleration	Specify the acceleration rate when the motion starts. (Unit: User unit/s^2)	LREAL	Positive (0)	When *bExecute* is True and the output *bBusy* is False.
lrDeceleration	Specify the deceleration rate when the motion ends. (Unit: User unit/s^2)	LREAL	Positive (0)	When *bExecute* is True and the output *bBusy* is False.
lrJerk	Specify the jerk. (Unit: User unit/s^3)	LREAL	Positive (0)	When *bExecute* is True and the output *bBusy* is False.
Direction	Specify the motion direction.	MC_DIRECTION*	-1: negative 1: positive (positive)	When *bExecute* is True and the output *bBusy* is False.
lrRoundPhase	Set the modulo.	LREAL	Positive (0)	When *bExecute* is True and the output *bBusy* is False.
lrStopPhase	Specify a position or a phase in the modulo.	LREAL	Positive or 0 (0)	When *bExecute* is True and the output *bBusy* is False.

*__Note__: MC_DIRECTION: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output range (Default Value)
bInVelocity	True when reaching the target velocity.	BOOL	True/False (False)
bDone	If the trigger signal is True and the axis position has been recorded.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted by another instruction.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Contains error codes.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

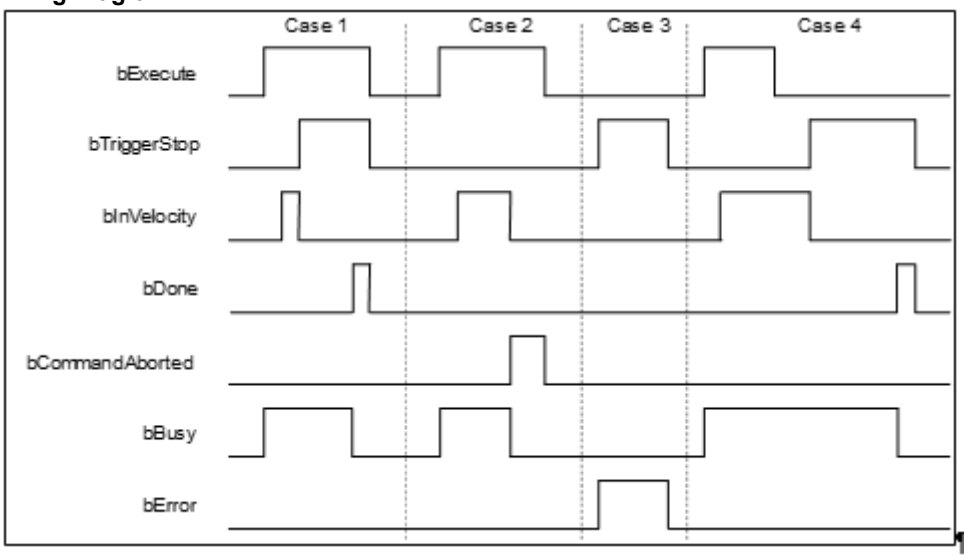
*__Note__: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for changing to TRUE	Timing for changing to FALSE
bInVelocity	• When axis velocity reaches the target speed.	• When *bCommandAborted* turns to True. • When *bExecute* is re-triggered, and Velocity is given a new value.

bDone	• When the trigger signal is True, and the axis position has been recorded.	• When *bExecute* turns to False. • *bDone* will change to False after remaining True for one period when *bExecute* is False but *bDone* changes to True.
bCommandAborted	• When the instruction is interrupted by another instruction.	• When *bExecute* turns to False.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns from True to False. (Error code is cleared.)
ErrorID		

- **Timing Diagram**



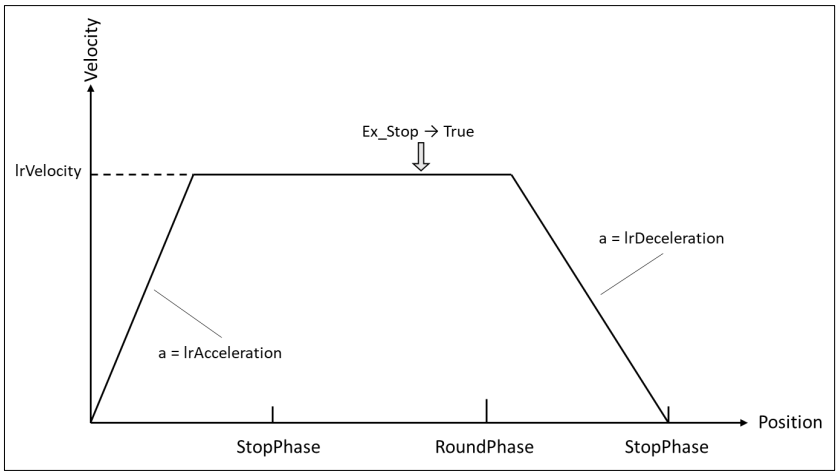
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *bExecute* turns to True.

*__Note__: AXIS_REF_SM3 (FB): The interface is built in every function block and works as the starting program of the function block.

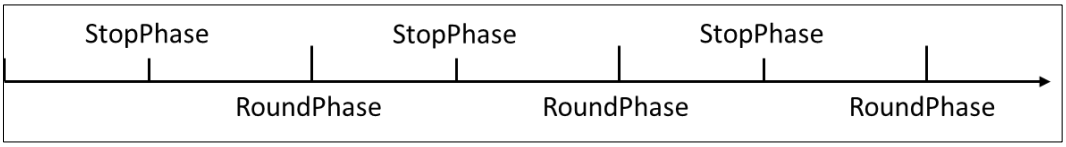
- **Function**

  - After *bExecute* of DMC_MoveVelocityStopByPos changes to True, the axis will move with the velocity specified by *lrVelocity* and acceleration specified by *lrAcceleration* until *bTriggerStop* changes to True. Then the axis positioning will start according to *lrDeceleration* and the target position converted from the values of *RoundPhase* and*StopPhase*.

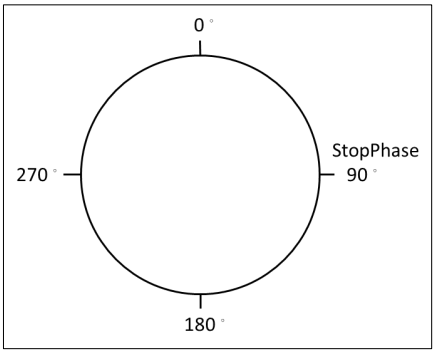


- RoundPhase & StopPhase

  - *RoundPhase* specifies a modulo. *StopPhase* is a position in the modulo. The value of *StopPhase* should be less than that of *RoundPhase.*
  - When the axis specified by the function block is a linear axis, *RoundPhase* is the length of the specified modulo. And *StopPhase* is a point in the specified modulo. When *bTriggerStop* changes to True, the axis will stop at the position specified by *StopPhase*, and the final stop position equals an integral multiple of *RoundPhase* value + *StopPhase* value.



  - When the axis specified by the function block is a rotary axis, *RoundPhase* specifies the entire phase of the modulo and *StopPhase* is a phase in the specified modulo. When *bTriggerStop* changes to True, the axis will stop at the phase specified by StopPhase, and the final stop position is (*StopPhase* value/ *RoundPhase* value) × **Modulo value** of the rotary axis.



- Special Case

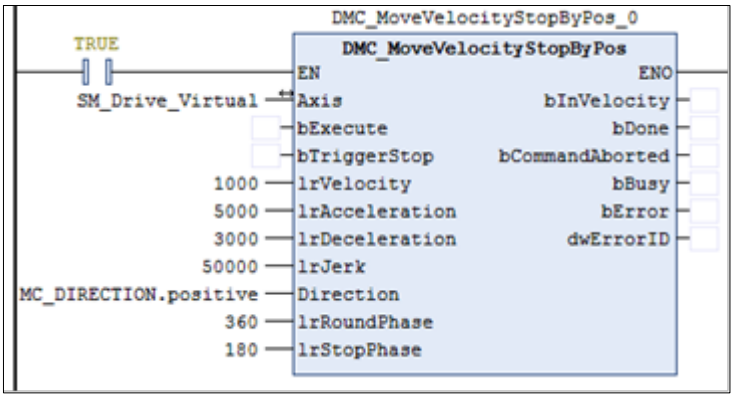
  - When the stop command is performed, the axis may not be able to complete the stop action with the deceleration rate specified by *lrdeceleration* if the position of the specified axis is too close to the target stop position. Therefore, the axis positioning will end in the next modulo. In that case, it is suggested to adjust the value of *lrDeceleration* or the position where the stop command is triggered so as to satisfy the path planned in the deceleration motion.

- **Troubleshooting**

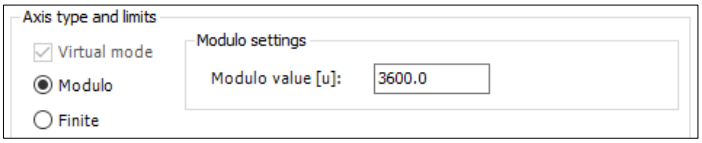
  - When an error occurs in the execution of instructions or the axis group enters “Errorstop state”, *bError* changes to True and the axes stops running. To confirm the current error state, see the error code in *ErrorID*.

- **Programming Examples**

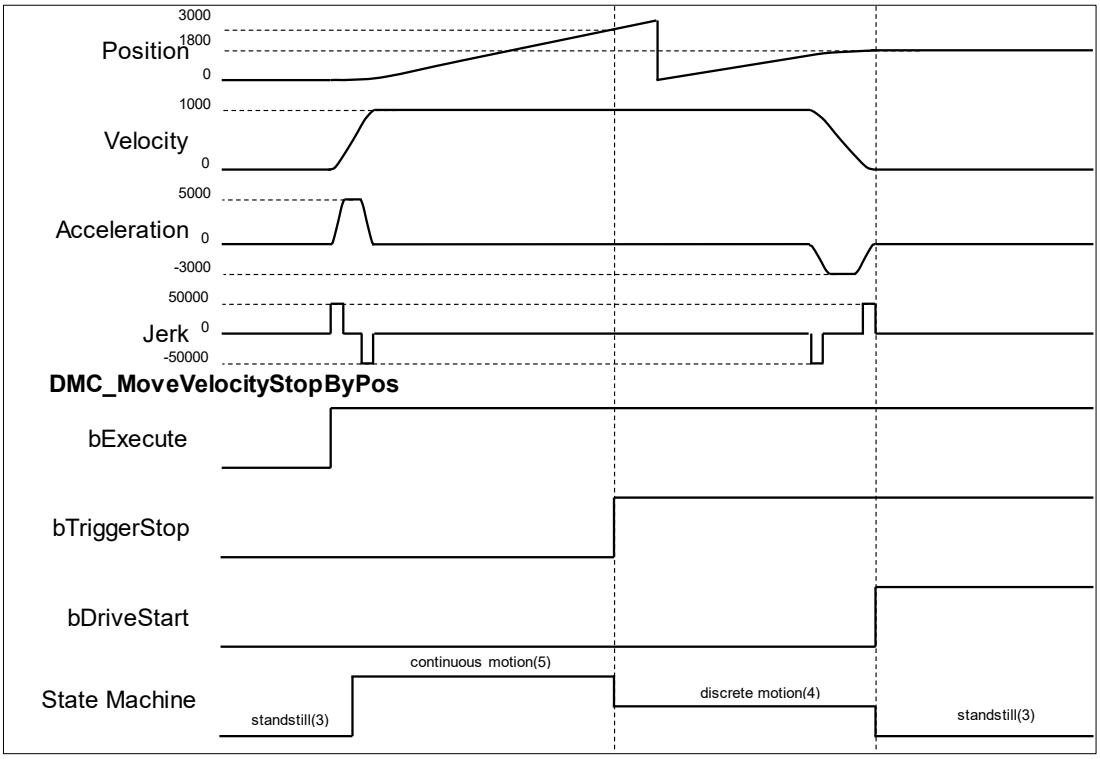
  - Programming Example 1:
    - This example illustrates how to use DMC_MoveVelocityStopByPos for phase positioning after the rotary axis motion starts.



    - Rotary axis setting



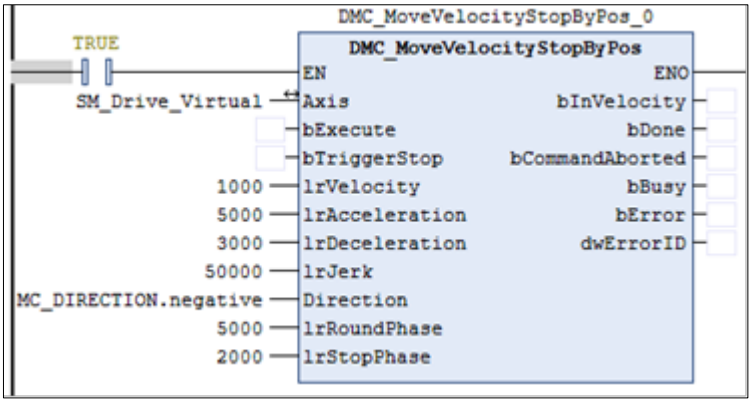
    - Timing Diagram



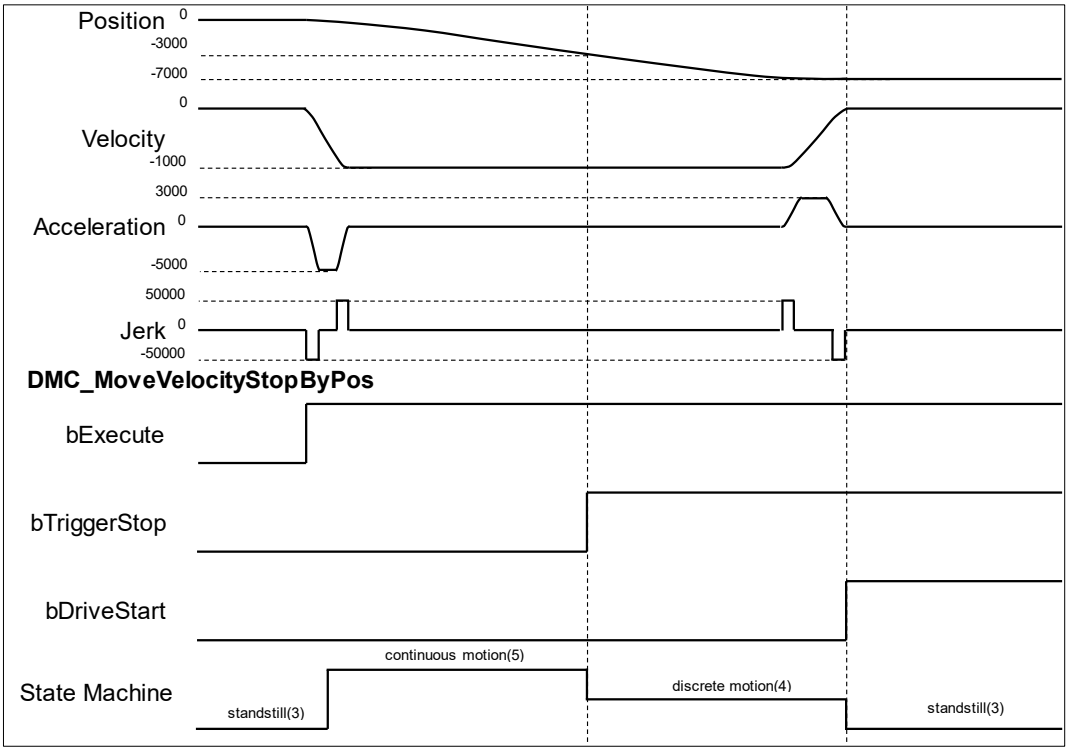
1. After *bExecute* changes to True, the axis starts to move at a constant speed in the set direction until *bTriggerStop* changes to True to start the positioning motion.
2. The *RoundPhase* and *StopPhase* of DMC_MoveVelocityStopByPos are set to 360 and 180, and the Modulo value of the rotary axis is 3600. Therefore, the rotary axis finally stops at 1800.
3. Since the axis position exceeds the position specified by *StopPhase* as *bTriggerStop* turns to True, the axis will stop at the next *StopPhase* position.

- Programming Example 2:

  - This example illustrates how to use DMC_MoveVelocityStopByPos for position locating after the linear axis motion starts.



  - Timing Diagram



1. After *bExecute* changes to True, the axis starts to move at a constant speed in the set direction until *bTriggerStop* changes to True to start the positioning motion.
2. The *RoundPhase* and *StopPhase* of DMC_MoveVelocityStopByPos are set to 5000 and 2000 respectively. Therefore, the linear axis finally stops at the position of an integral multiple of 5000 plus 2000.
3. Since the axis position exceeds 2000 as *bTriggerStop* turns to True, the axis will stop at the next 7000.

2.3.1.12 DMC_GroupInterrupt

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupInterrupt makes the current motion pause, and it can be used with DMC_GroupContinue to restore the motion.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupInterrupt	

ST Language

```
DMC_GroupInterrupt_instance(
AxisGroup: = ,
bExecute: = ,
lrDeceleration: = ,
lrJerk: = ,
bDone=> ,
bBusy=> ,
bCommandAborted=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
lrDeceleration	Deceleration (user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True.
lrJerk	Jerk ; Jump (Unit: user unit/s^3)	LREAL	Positive (0)	When *bExecute* turns to True.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	True when all axes stop with the velocity 0.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)

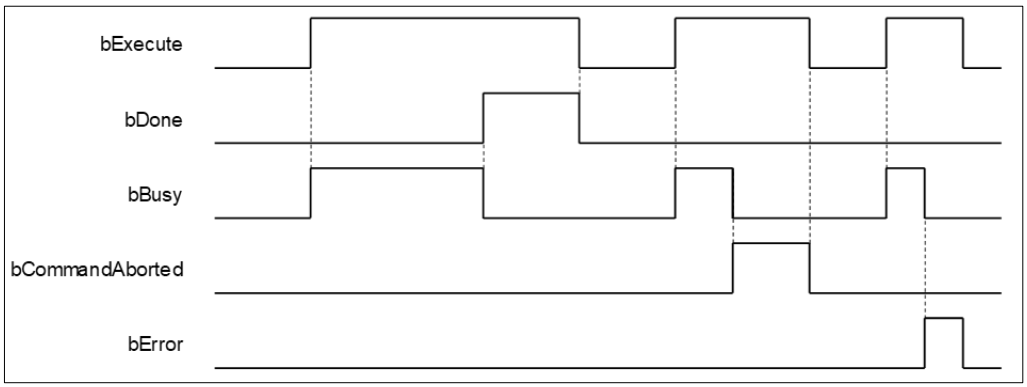
Name	Function	Data Type	Output Range (Default Value)
bError	True when an instruction error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to **Appendix** for error code descriptions.	DMC_ERROR*	DMC_ERROR(DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When decelerating to stop.	• When *bExecute* turns to False. • *bDone* will change to False after remaining True for one cycle when *bExecute* is False but *bDone* changes to True.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bCommandAborted	• When the instruction is aborted by another instruction.	• When *bExecute* turns to False. • *bCommandAborted* will change to False after remaining True for one cycle when *bExecute* is False but *bCommandAborted* changes to True.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	• When *bExecute* turns from True to False (Error Code is cleared).
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_ GROUP_REF*	DMC_AXIS_ GROUP_REF	When *bExecute* turns to True and *bBusy* is False.

***Note:** DMC_AXIS_GROUP_REF(FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

- This instruction can decelerate and stop the motion of axis groups other than DMC_GroupStop.
- During deceleration, the status of the axis groups remains GroupMoving.
- When the speed reaches 0, the output pin bDone will immediately changes to True, and the status of the axis groups will switch to StandBy.
- If Group state machine is GroupMoving during execution, after the instruction execution, the following Continue data will be recorded for subsequent DMC_GroupContinue to resume the motion.
  - The instructions that have not finished execution (including the instructions that have not finished execution in the instruction buffer area).
  - Position after motion stops (AxisGroup.ContinuePos).

After recording the dada, AxisGroup.bContinueDataWriten will be set to TRUE.

The execution will not be recorded if it is not completed.

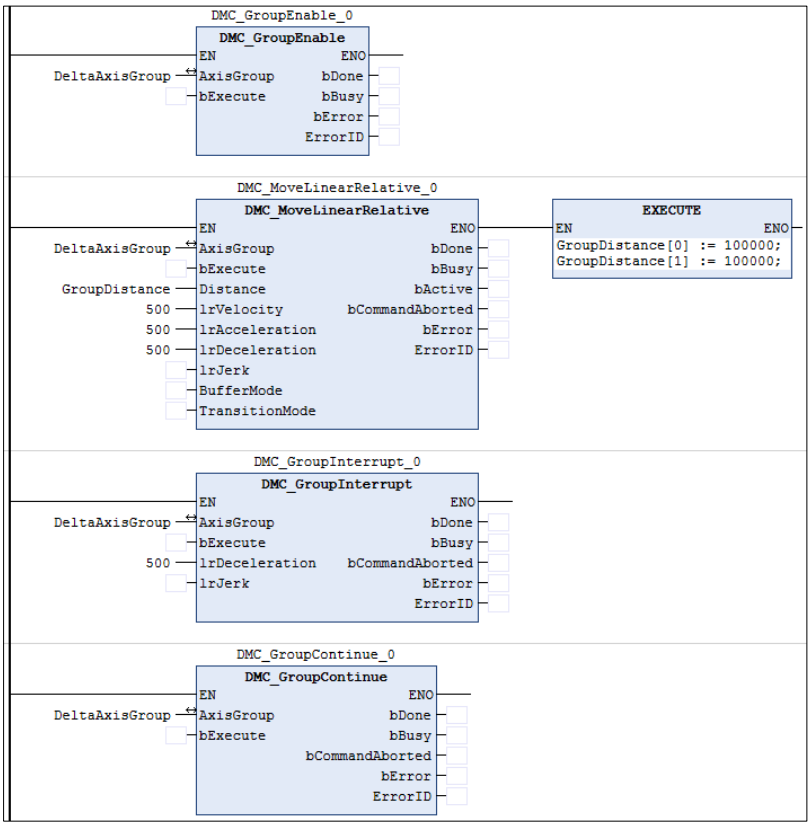
- When there is currently Continue data, re-execute DMC_GroupInterrupt. The current Continue data will firstly be cleared. Whether to record new Continue for follow-up depends on if the execution is successful.
- During the execution of DMC_GroupInterrupt, users might encounter the following situation:
  - During DMC_GroupInterrupt execution, DMC_GroupInterrupt will be interrupted when triggering DMC_GroupStop.
  - During DMC_GroupInterrupt execution, when the second function block DMC_GroupInterrupt is re-triggered, it will be interrupted.
  - During DMC_GroupInterrupt execution, DMC_GroupInterrupt continues running when axis groups motion instructions are carried out. Motion instructions are at the Busy state until DMC_GroupInterrupt execution is completed, and the instructions will be added into instruction buffer to start execution.

- **Troubleshooting**

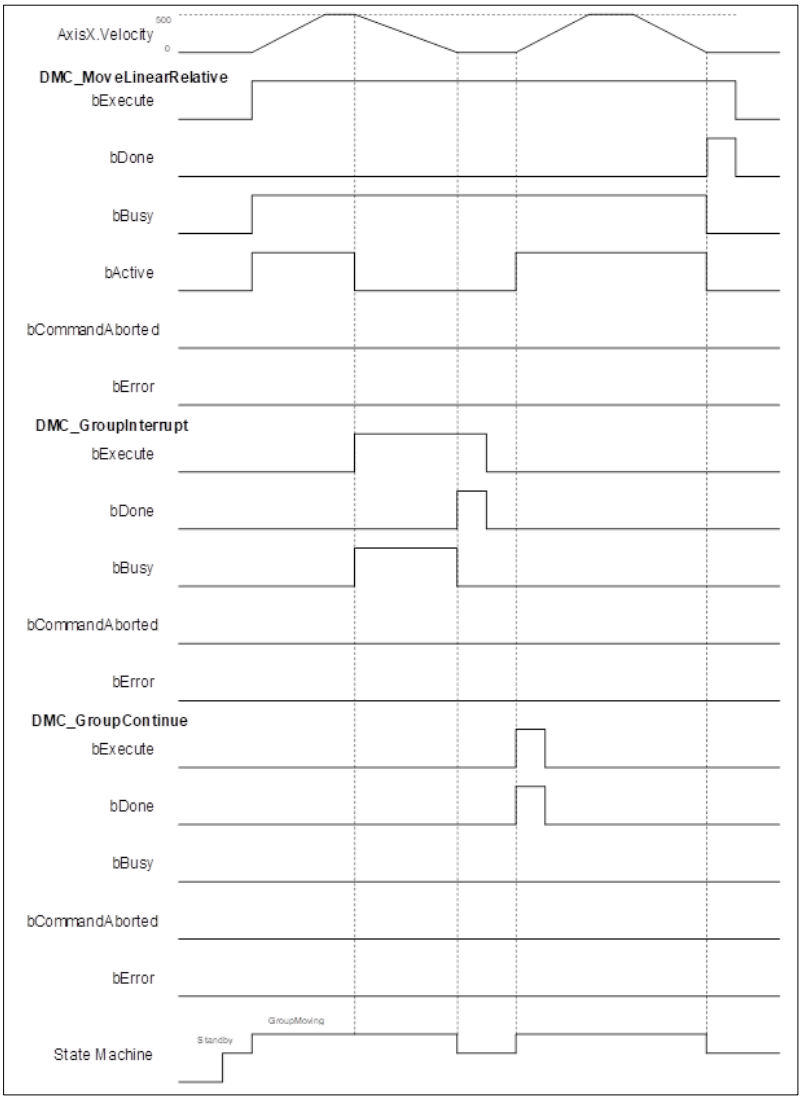
- When an error occurs in the execution of the instruction, *bError* turns to True, and the axis motion will stop. Refer to *ErrorID* (Error Code) to confirm the current error status.
- For error codes and corresponding troubleshooting, refer to **Appendix** of this manual.

- **Programming Example**

- This example shows that the DMC_GroupInterrupt function block is run when the axis group is running. After the axis group stops running, the DMC_GroupContinue function block is used again to restore the axis group motion.



■ Timing Diagram

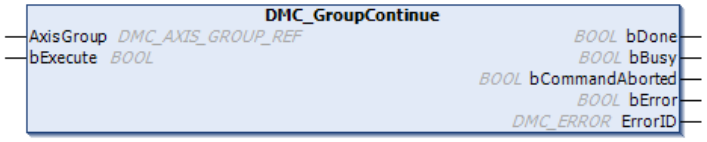


- When *bExecute* of DMC_MoveLinearRelative changes to True, the axis group starts to run.
- When *bExecute* of DMC_GroupInterrupt changes to True, the axis group will decelerate until the speed reaches 0 and stops. The status of DMC_GroupInterrupt will be changed from Busy to Done.
- At this time, *bExecute* of DMC_GroupContinue changes to True, and the unfinished motion path of previous DMC_MoveLinearRelative of the axis group will be completed.

2.3.1.13 DMC_GroupContinue

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupContinue restores the interrupted motion of DMC_GroupInterrupt.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupContinue	

ST Language

```
DMC_GroupContinue_instance(
AxisGroup: = ,
bExecute: = ,
bDone=> ,
bBusy=> ,
bCommandAborted=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output range value (Default Value)
bDone	True when motion is resumed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
bError	True when an instruction error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

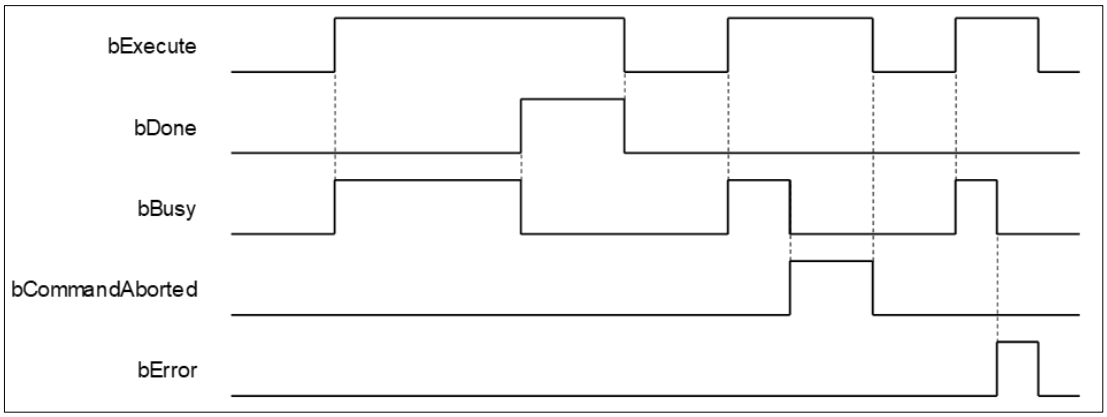
Name	Function	Data Type	Output range value (Default Value)
	for error code descriptions.		

*__Note__: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When motion is resumed.	• When *bExecute* turns to False. • *bDone* will change to False after remaining True for one cycle when *bExecute* is False but *bDone* changes to True.
bBusy	• When *bExecute* changes to TRUE.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bCommandAborted	• When the instruction is interrupted by another function block.	• When *bExecute* turns to False. • *bCommandAborted* will change to False after remaining True for one cycle when *bExecute* is False but *bCommandAborted* changes to True.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	• When *bExecute* turns from True to False (Error Code is cleared).
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_ GROUP_REF*	DMC_AXIS_ GROUP_REF	When *bExecute* turns to True and *bBusy* is False.

***Note:** DMC_AXIS_GROUP_REF(FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

  - This instruction can resume the axis group motion that is stopped by DMC_GroupInterrupt.
  - Three conditions for successfully execute this instruction:
    - The current status of the axis group is GroupStandby.

    - There is recorded Continue data (the axis group variable AxisGroup.bContinueDataWriten is True).
    - The current position is at AxisGroup.ContinuePos.
  - Output pin bDone will immediately change to True after the execution is successful, and clear the Continue data that is recorded.

- **Troubleshooting**

  - When an error occurs in the execution of instructions, *bError* will change to True and the axes will stop running. Refer to *ErrorID* (Error Code) to confirm the current error status.
  - For error codes and corresponding troubleshooting, refer to **Appendix** of this manual.

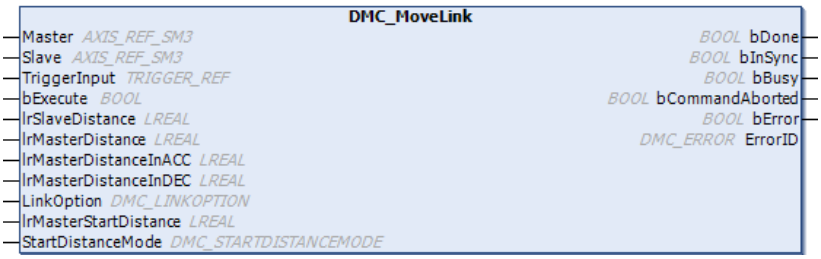
- **Programming Example**

  - Refer to DMC_GroupInterrupt programming example.

2.3.1.14 DMC_MoveLink

- **Supported Devices**: AX-308E, AX-364E

DMC_MoveLink makes the Slave Axis follow the Master Axis for synchronous positioning movement.

FB/FC	Instruction	Graphic Expression
FB	DMC_MoveLink	

ST Language

```
DMC_MoveLink_instance(
Master: = ,
Slave: = ,
TriggerInput: = ,
bExecute: = ,
lrSlaveDistance: = ,
lrMasterDistance: = ,
lrMasterDistanceInACC: = ,
lrMasterDistanceInDEC: = ,
LinkOption: = ,
lrMasterStartDistance: = ,
StartDistanceMode: = ,
bDone=> ,
bInSync=> ,
bBusy=> ,
bCommandAborted=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	When *bExecute* turns to True.
lrSlaveDistance	Slave axis travel distance (user units)	LREAL	Positive, negative, or 0 (0)	When *bExecute* turns to True.
LrMasterDistance	Spindle travel distance (user unit)	LREAL	positive (0)	When *bExecute* turns to True.
lrMasterDistanceInACC	Spindle acceleration travel distance (user unit)	LREAL	positive (0)	When *bExecute* turns to True.
lrMasterDistanceInDEC	Spindle deceleration travel distance (user unit)	LREAL	positive (0)	When *bExecute* turns to True.

LinkOption	Synchronization start condition selection	DMC_LINKOPTION[*1]	0: COMMANDEEXECUTION 1: TRIGGERDETECTION 2: MASTERREACH (COMMANDEEXECUTION)	When *bExecute* turns to True.
lrMasterStartDistance	Spindle following distance (user unit)	LREAL	Positive, negative, or 0 (0)	When *bExecute* turns to True.
StartDistanceMode	Spindle following distance mode	DMC_STARTDISTANCEMODE[*2]	0: ABSOLUTE 1: RELATIVE (ABSOLUTE)	When *bExecute* turns to True.

***Note**:

1. DMC_LINKOPTION: Enumeration (Enum)
2. DMC_STARTDISTANCEMODE: Enumeration (Enum)

- **Outputs**

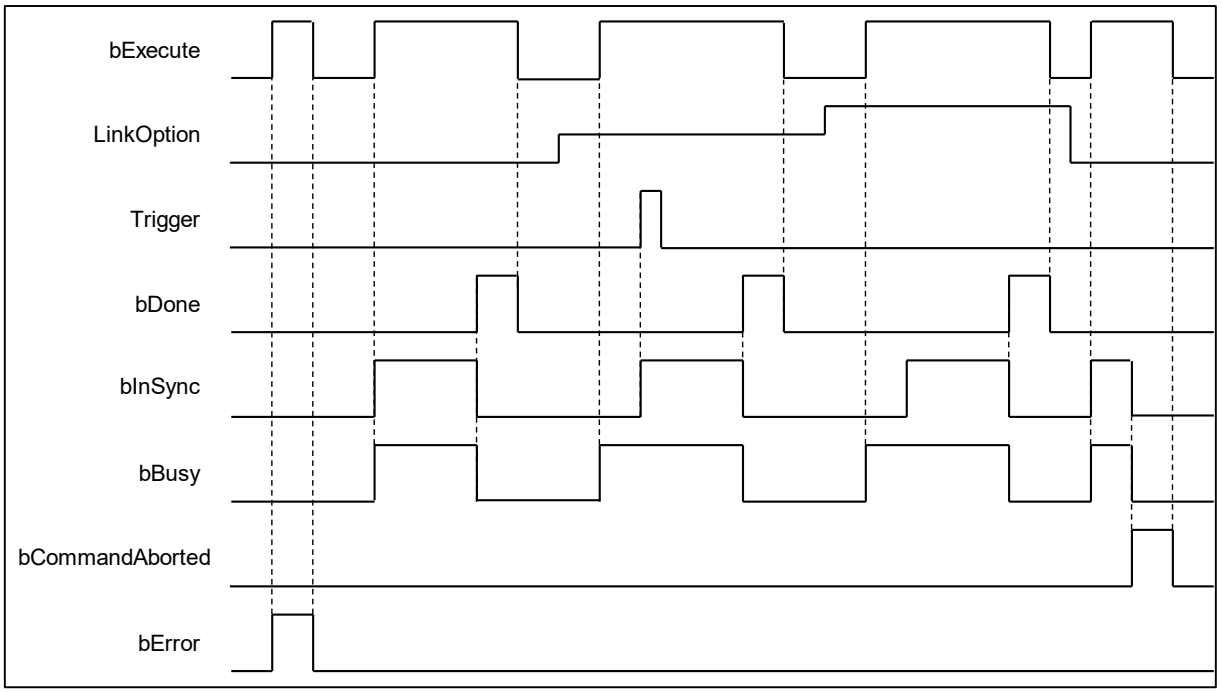
Name	Function	Data Type	Output range value (Default Value)
bDone	Turns True when slave axis positioning is complete.	BOOL	True/False (False)
bInSync	True when master and slave cams are synchronized.	BOOL	True/False (False)
bBusy	True when Instruction is running.	BOOL	True/False (False)
bCommandAborted	True when Instruction is interrupted.	BOOL	True/False (False)
bError	True when an Instruction error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when the instruction error occurs. For the detailed description of the error code, refer to the appendix of the manual.	DMC_ERROR[*]	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: enumerate（Enum）

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When slave axis positioning is completed	• When *bExecute* turns to False. • If *bExecute* is False and *bDone* turns to True, then *bDone* turns to False immediately after maintaining the True state for one scan cycle.
bInSync	• When the slave axis is synchronized	• After positioning
bBusy	• When the upper edge of *bExecute* fires	• When *bDone* goes up • when *bError* goes up • When *bCommandAborted* upper edge
bCommandAborted	• When The Instruction is interrupted by another Function block	• When *bExecute* turns to False. • *bCommandAborted* will change to False after remaining True for at least one cycle when *bExecute* is False but *bCommandAborted* changes to True.
bError	• When an error occurs in the execution condition or input value of the Insert (the error code is recorded in the ErrorID).	• When *bExecute* turns to False. (Clear the error code of the ErrorID record)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	Specifies the master axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	When *bExecute* rises and *bBusy* status is False
Slave	Specifies the slave axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	When *bExecute* rises and *bBusy* status is False
TriggerInput	Trigger signal	TRIGGER_REF*2	TRIGGER_REF	When *bExecute* turns to True.

***Note**:

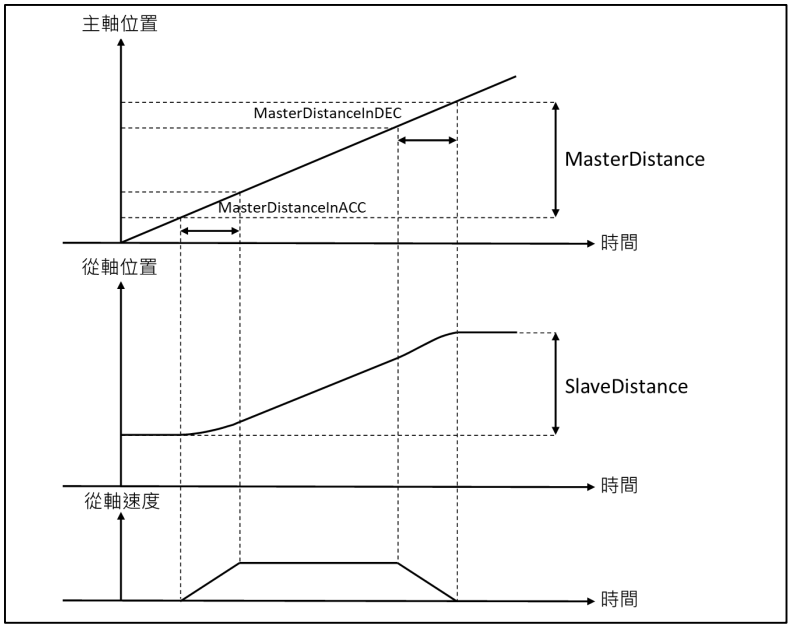
1. AXIS_REF_SM3（FB）: Every function block contains this variable, which works as the starting program for function blocks.
2. TRIGGER_REF: Structure（STRUCT）

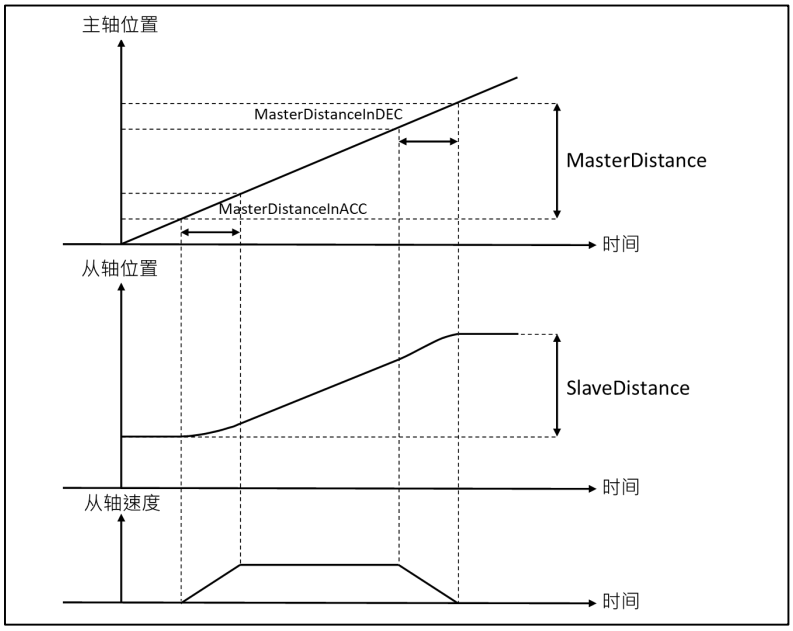
Name	Function	Data Type	Setting Range (Default)
iTriggerNumber	Trigger channel	INT	0: Touch Probe 1 upper edge data capture 1: Touch Probe 1 lower edge data capture 2: Touch Probe 2 upper edge data capture 3: Touch Probe 2 lower edge data capture (-1)
bFastLatching	Trigger signal	BOOL	True: the drive triggers False: Controller trigger (True)
bInput	When the controller is triggered, it is the trigger signal source	BOOL	Trigger source
bActive*	Whether the trigger signal is valid	BOOL	True: The trigger signal is valid (False)

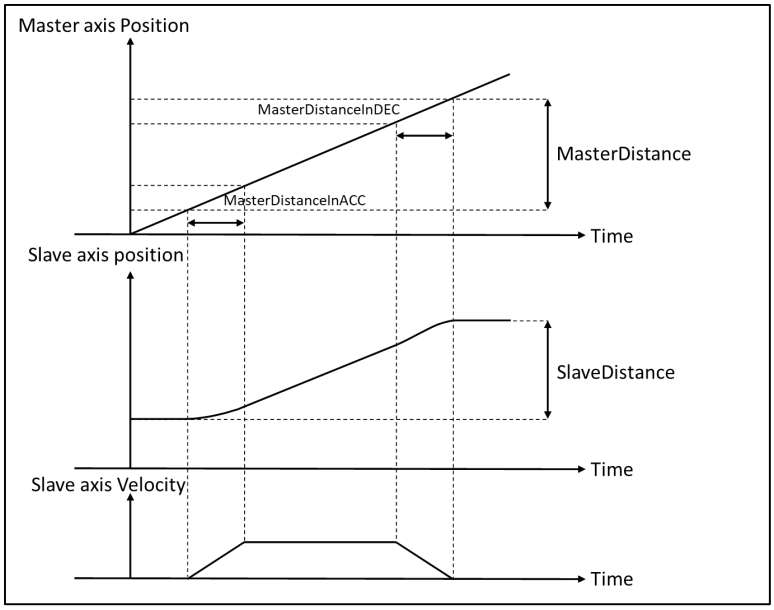
***Note:** bActive is the output contact, do not input signal.

- **Function**

  - The synchronous motion trigger of MC_MoveLinke is determined by LinkOption.
    - COMMANDEEXECUTION: When DMC_MoveLinke starts, the slave axis goes into synchronization and performs positioning movements.
    - TRIGGERDETECTION: After DMC_MoveLinke has started, it waits for an external signal to trigger, and after triggering, the slave axis enters the synchronization state and runs positioning motion.
  - StartDistanceMode determines, if LinkOption = MASTERREACH mode, the specified position of the spindle.
    - ABSOLUTE: The specified location is MasterStartDistance.
    - RELATIVE: The specified position is the current position triggered by the MasterStartDistance+Function block.
  - The positioning path of the slave axis is converted by four parameters: the moving distance of the spindle acceleration segment (MasterDistanceInACC), the moving distance of the spindle deceleration segment (MasterDistanceInDEC), the moving distance of the master axis (MasterDistance), and the moving distance of the slave axis (SlaveDistance).


主軸位置
MasterDistanceInDEC
MasterDistance
MasterDistanceInACC
時間
從軸位置
SlaveDistance
時間
從軸速度
時間


主轴位置
MasterDistanceInDEC
MasterDistance
MasterDistanceInACC
时间
从轴位置
SlaveDistance
时间
从轴速度
时间



◆ **Acceleration Section**

Interval	The relationship between the master axis and the slave axis movement amount	
Acceleration Section	Master Axis	MasterDistanceInACC
	Slave Axis	$\text{SlaveDistance}\times \frac{\frac{\text{MasterDistanceInACC}}{2}}{\frac{\text{MasterDistanceInACC}}{2}+(\text{MasterDistance}-\text{MasterDistanceInACC}-\text{MasterDistanceInDEC})+\frac{\text{MasterDistanceInDEC}}{2}}$

◆ **Constant Velocity Section**

Interval	The relationship between the master axis and the slave axis movement amount	
Constant Velocity Section	Master Axis	MasterDistance – MasterDistanceInACC – MasterDistanceInDEC
	Slave Axis	SlaveDistance –Slave AxisAcceleration Section moving distance – 下述Slave AxisDeceleration Section移動距離

◆ **Deceleration Section**

Interval	The relationship between the master axis and the slave axis movement amount	
Deceleration Section	Master Axis	MasterDistanceInDEC
	Slave Axis	$\text{SlaveDistance}\times \frac{\frac{\text{MasterDistanceInDEC}}{2}}{\frac{\text{MasterDistanceInACC}}{2}+(\text{MasterDistance}-\text{MasterDistanceInACC}-\text{MasterDistanceInDEC})+\frac{\text{MasterDistanceInDEC}}{2}}$

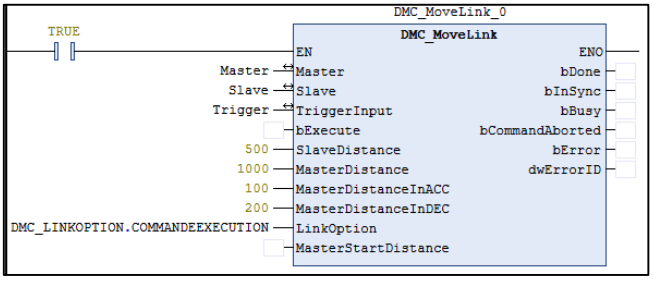
■ When TRIGGERDETECT has the same driver mode and controller mode as MC_TouchProbe, refer to the MC_TouchProbe instruction manual for the usage of the two modes and the usage settings of TriggerInput.

● **Troubleshooting**

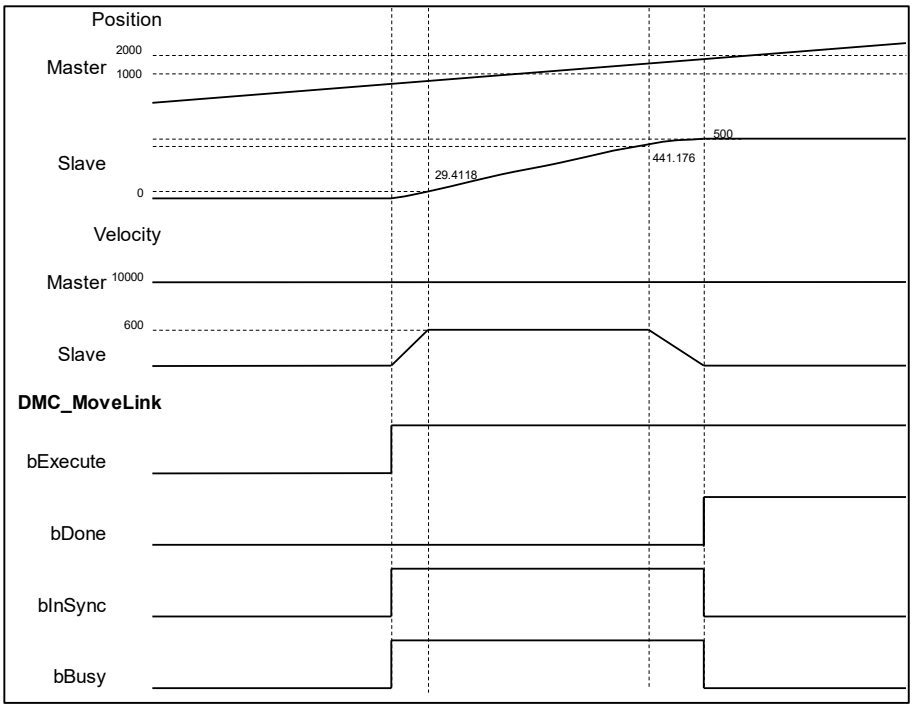
- If an error occurs during the execution of the instruction, you can refer to the content of ErrorID (error code) to confirm the current error status.

- **Programming Example**

  - **Sample Program1**:
    The example shows that DMC_MoveLink operates in COMMANDEEXECUTION mode.



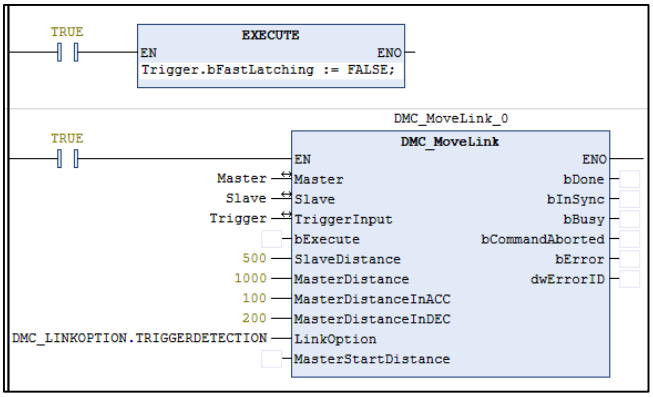
  - **Timing Diagram**



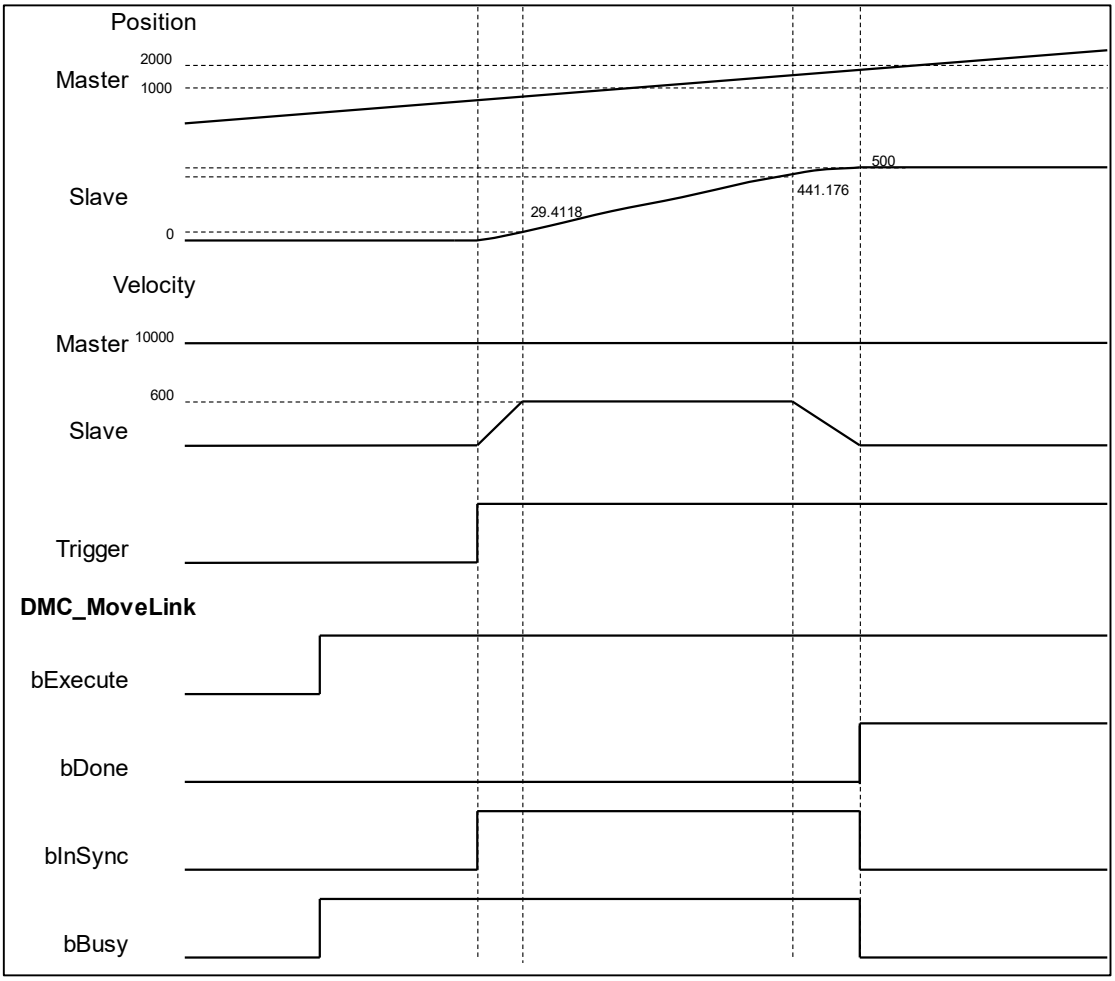
5. When the LinkOption is set to COMMANDEEXECUTION, the Slave Axis enters the synchronization mode immediately after DMC_MoveLink is activated, and performs positioning motion according to the speed of the Master Axis.
6. According to the four parameters of MasterDistanceInACC, MasterDistanceInDEC, MasterDistance and SlaveDistance, the distance of Slave AxisAcceleration Section can be calculated as 29.4118, and the distance of Slave AxisDeceleration Section is 58.824.
7. When the Slave Axis completes the dynamic movement, the Slave Axis leaves the synchronization relationship, and DMC_MoveLink completes the synchronous positioning movement.

  - **Sample Program 2**:

    The example shows that DMC_MoveLink operates in TRIGGERDETECTION mode. (External trigger using controller mode)



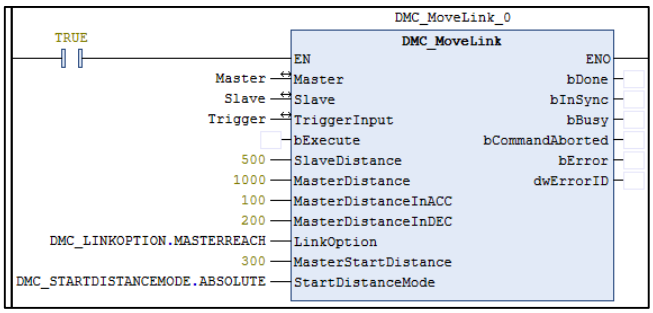
- **Timing Diagram**



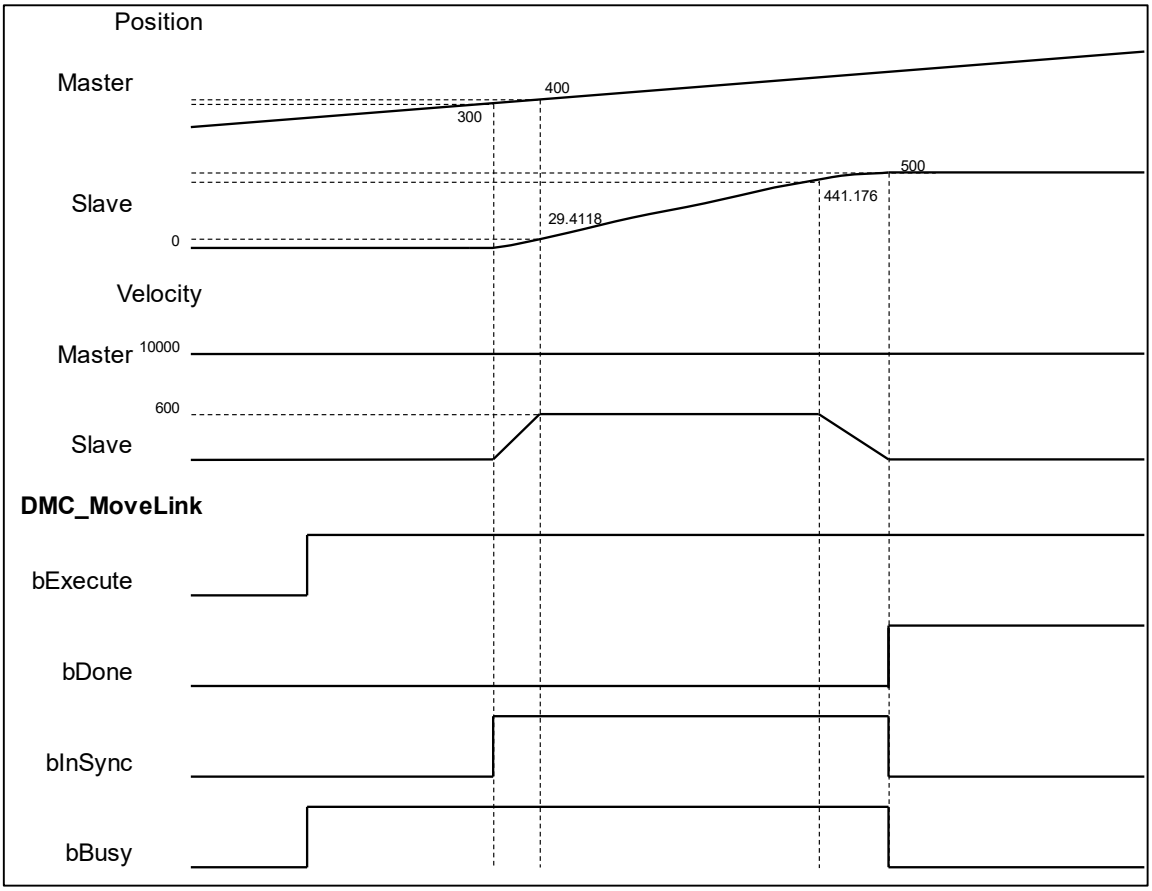
1. When the LinkOption is set to TRIGGERDETECTION, after the DMC_MoveLink is activated, the trigger signal must be used to make the Slave Axis enter the synchronization mode. After the trigger, the Slave Axis will follow the speed of the Master Axis to perform positioning motion.
2. According to the four parameters of MasterDistanceInACC, MasterDistanceInDEC, MasterDistance and SlaveDistance, the distance of Slave AxisAcceleration Section can be calculated as 29.4118, and the distance of Slave AxisDeceleration Section is 58.824.
3. When the Slave Axis completes the moving position movement, the Slave Axis leaves the synchronization relationship, and DMC_MoveLink completes the synchronous positioning movement.

- **Sample Program 3**:

Example to illustrate that DMC_MoveLink operates in MASTERREACH mode.



- **Timing Diagram**

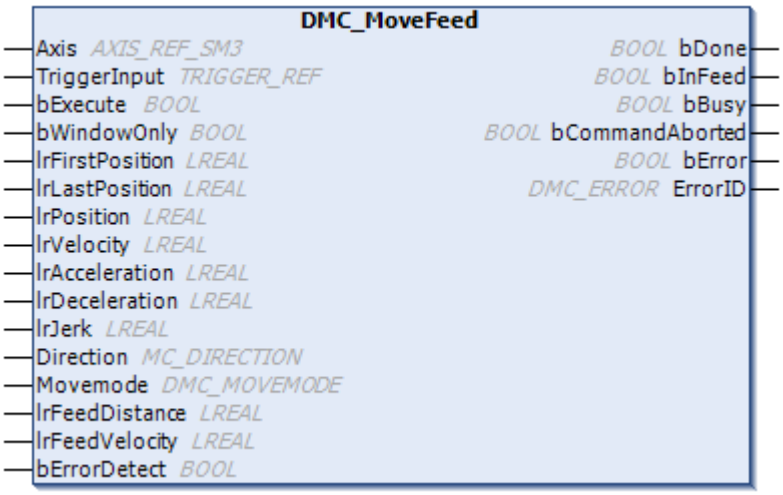


1. When the LinkOption is set to MASTERREACH, after DMC_MoveLink is activated, the Master Axis must pass the position set by the MasterStartDistance, the Slave Axis enters the synchronous mode, and the positioning motion is performed according to the speed of the Master Axis.
2. StartDistanceMode is set to ABSOLUTE mode, which means that when the Master Axis runs to 300, the Slave Axis starts to perform synchronous positioning motion.
3. According to the four parameters of MasterDistanceInACC, MasterDistanceInDEC, MasterDistance and SlaveDistance, the distance of Slave AxisAcceleration Section can be calculated as 29.4118, and the distance of Slave AxisDeceleration Section is 58.824.
4. When the Slave Axis completes the moving position movement, the Slave Axis leaves the synchronization relationship, and DMC_MoveLink completes the synchronous positioning movement.

2.3.1.15 DMC_MoveFeed

- **Supported Devices**: AX-308E, AX-364E

DMC_MoveFeed can specify an external interrupt input. During the movement of the target, the position where the interrupt occurs is used as the starting point to perform the positioning movement.

FB/FC	Instruction	Graphic Expression
FB	DMC_MoveFeed	

ST Language

```
DMC_MoveFeed _instance(
Axis: = ,
TriggerInput: = ,
bExecute: = ,
bWindowOnly: = ,
lrFirstPosition: = ,
lrLastPosition: = ,
lrPosition: = ,
lrVelocity: = ,
lrAcceleration: = ,
lrDeceleration: = ,
lrJerk: = ,
Direction: = ,
Movemode: = ,
lrFeedDistance: = ,
lrFeedVelocity: = ,
bErrorDetect: = ,
bDone=> ,
bInFeed=> ,
bBusy=> ,
bCommandAborted=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	When *bExecute* turns to True.
bWindowOnly	Enable the Window range setting.	BOOL	True/False (False)	When *bExecute* turns to True and *bBusy* is False.
lrFirstPosition	Defines the start position of Window. (Unit: user unit)	LREAL	Negative, Positive or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
lrLastPosition	Defines the end position of Window. (Unit: user unit)	LREAL	Negative, Positive or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
lrPosition	Absolute target position (Unit: user unit)	LREAL	Negative, Positive or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
lrAcceleration	Acceleration rate (Unit: user unit/s2)	LREAL	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
lrDeceleration	Deceleration rate. (Unit: user unit/s2)	LREAL	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
lrJerk	Specify the jerk. (Unit: user unit/s3)	LREAL	Positive or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
Direction	Specify the motion direction.	MC_DIRECTION*1	-1: negative 0 : shortest 1 : positive 2 : current 3 : fastest （shortest）	When *bExecute* turns to True and *bBusy* is False.
MoveMode	target movement mode	DMC_MOVEMODE*2	0 : ABSOLUTE 1 : RELATIVE 2 : VELOCITY (ABSOLUTE)	When *bExecute* turns to True and *bBusy* is False.
lrFeedDistance	standard distance (Unit: user unit)	LREAL	Negative, Positive or 0 (0)	When turns to True.
lrFeedVelocity	Standard speed (user unit)	LREAL	Positive (0)	When turns to True.
bErrorDetect	Error detection selection	BOOL	True/False (False)	When *bExecute* turns to True and *bBusy* is False.

***Note**:

1. MC_DIRECTION: Enumeration (Enum)
2. DMC_MOVEMODE: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output range value (Default Value)
bDone	Turns to True when the standard movement is completed or the target movement is completed and ErrorDetect is False.	BOOL	True/False (False)
bInFeed	True in standard motion	BOOL	True/False (False)

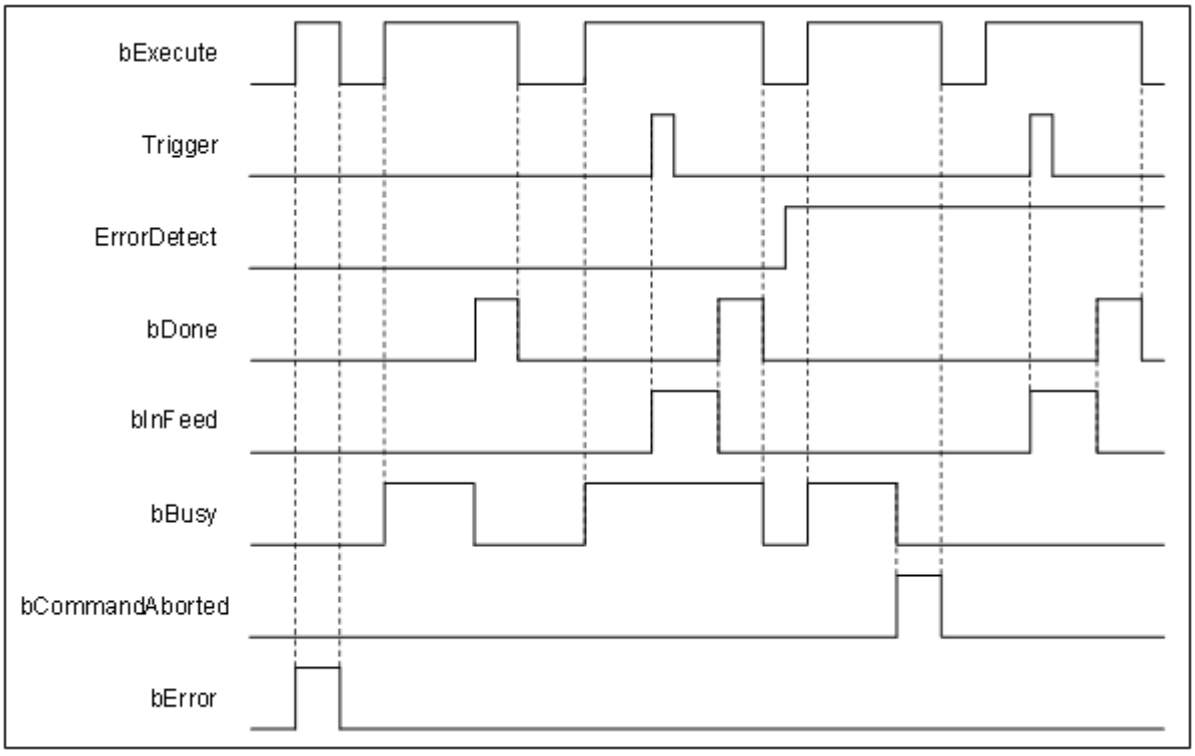
Name	Function	Data Type	Output range value (Default Value)
bBusy	True when Instruction is executing	BOOL	True/False (False)
bCommandAborted	True when Instruction is interrupted	BOOL	True/False (False)
bError	True when an Instruction error occurs	BOOL	True/False (False)
ErrorID	Record the error code when the instruction error occurs. For the detailed description of the error code, refer to the appendix of the manual.	DMC_ERROR*	DMC_ERROR （DMC_NO_ERROR）

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the standard movement is completed or the target movement is completed and ErrorDetect is False.	• When *bExecute* turns to False. • *bDone* will change to False after remaining True for one period when *bExecute* is False but *bDone* changes to True.
bInFeed	• During standard exercise	• When completing standard exercise.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True.
bCommandAborted	• The Function block is interrupted or when the target motion is complete and bErrorDetect is True.	• When *bExecute* turns to False.
bError	• When an error occurs in the execution condition of the Instruction or the input value.	• When *bExecute* turns to False. (Clear the error code of the ErrorID record)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	When bExcute turns to True.
TriggerInput	Trigger signal	TRIGGER_REF*2	TRIGGER_REF	When *bExecute* turns to True.

***Note**:

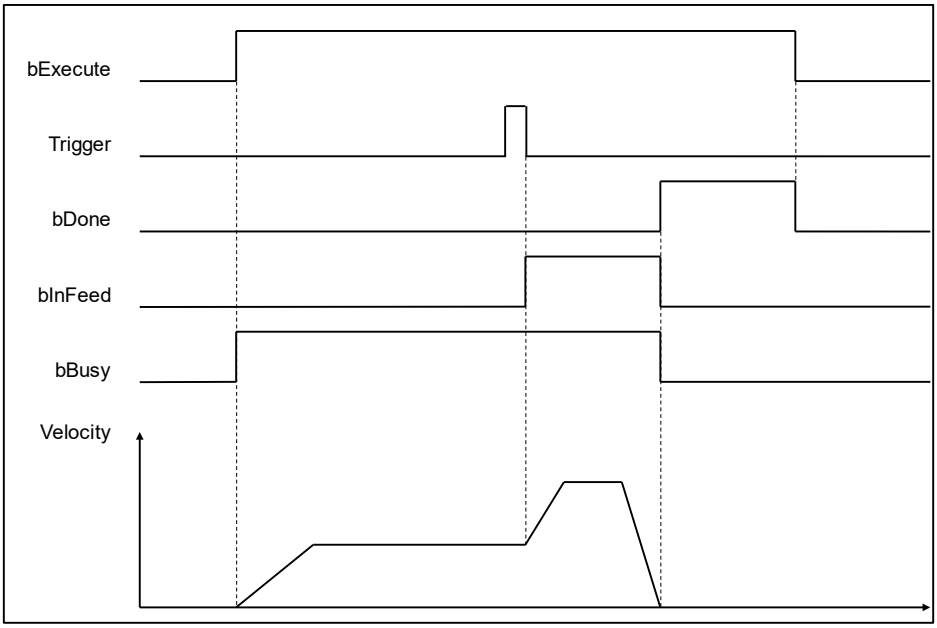
1. AXIS_REF_SM3 (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.
2. TRIGGER_REF: Structure (STRUCT).

Name	Function	Data Type	Setting Range (Default)
iTriggerNumber	Trigger channel	INT	0: Touch Probe 1 aquire data when True 1: Touch Probe 1 aquire data when False 2: Touch Probe 2 aquire data when True 3: Touch Probe 2 aquire data when False
bFastLatching	Trigger signal	BOOL	True: Drive trigger False: Controller trigger (True)
bInput	Trigger signal source when Controller trigger	BOOL	Trigger signal source
bActive*	Trigger signal valid or not	BOOL	True: Trigger signal valid (False)

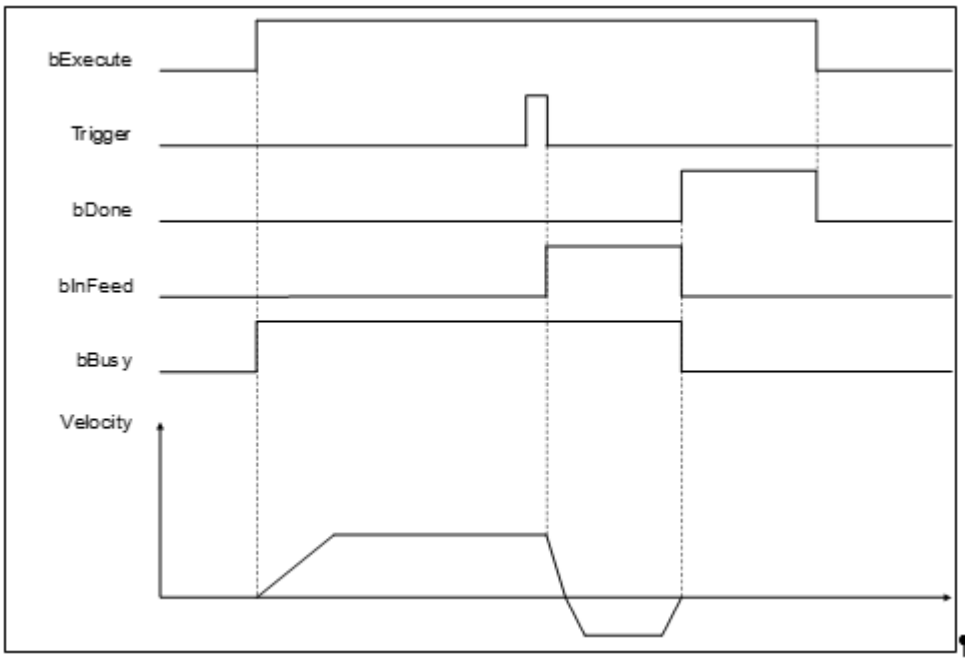
***Note**: bActive is the output contact. Do not input signal.

- **Function**

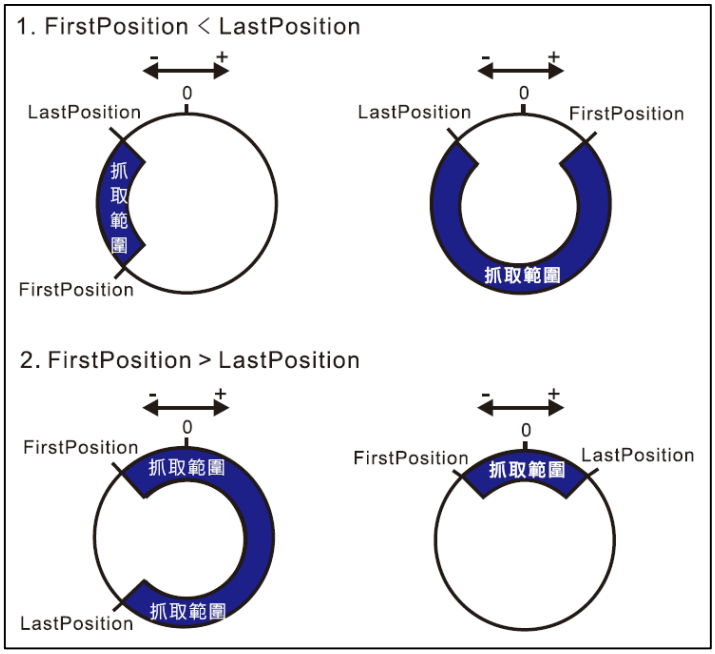
  - Like MC_TouchProbe, there are driver mode and controller mode. For the usage of the two modes and the usage settings of TriggerInput, refer to the MC_TouchProbe instruction manual.
  - The mode of the target movement (the first segment of movement) is set by MoveMode, in which the absolute movement (ABSOLUTE) and the relative movement (RELATIVE) are completed if there is no trigger signal. At this time, the DMC_MoveFeedFunction block will enter the next stage according to the ErrorDetect setting, and the velocity motion (VELOCITY) will continue to run regardless of the ErrorDetect state.
  - When ErrorDetect is False and the target motion (the first motion) is completed, bDone turns to TrueFunction block to complete; when ErrorDetect is True, and the target motion (first motion) is completed, bCommandAborted turns to TrueFunction block to interrupt.
  - During standard motion (second motion), relative motion will be performed according to the standard distance (IrFeedDistance). When IrFeedDistance is a positive value, the axis will maintain the original motion direction for standard motion.



- When lrFeedDistance is negative, the axis will perform standard motion in the opposite direction of the current motion.



- When lrFeedDistance is set to 0, the axis will stop immediately.
- In drive mode, if Touch Probe Status(60B9h) and Touch Probe Pos1 Value(60BAh) are not configured in PDO, you need to use SDO to ask the controller. Therefore, when the Trigger signal comes, the Function block will not respond immediately. In response configure the above two PDOs.
- Window Mask Setting
  - When the axis is set as a rotary axis, different results will be obtained with different Window Mask settings. The results obtained by setting different FirstPosition and LastPosition Interval are shown in the following figure.



- In the linear axis mode, the Windows Mask Setting Range value must be FirstPosition < LastPosition, and the trigger can work within the range value.

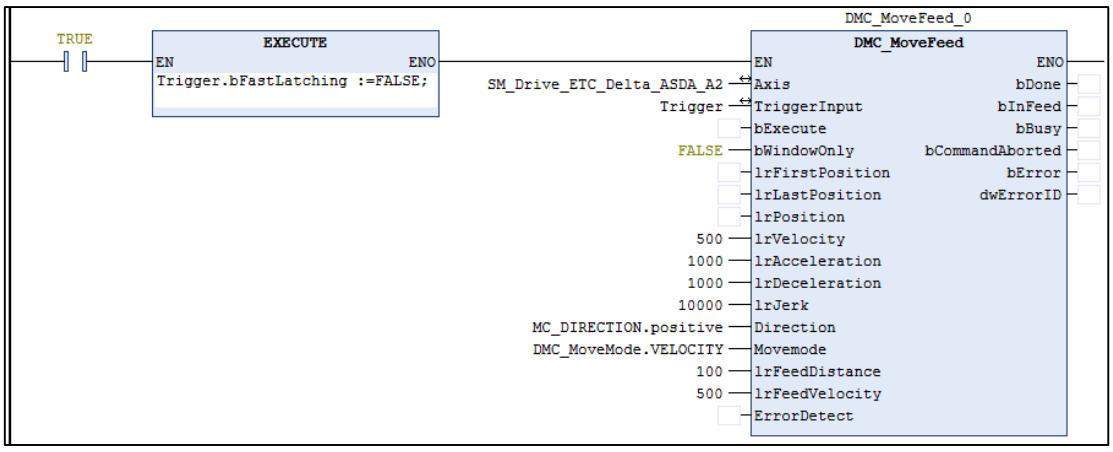
- **Troubleshooting**

  - If an error occurs during the execution of the Instruction, you can refer to the content of ErrorID (error code) to confirm the current error status.

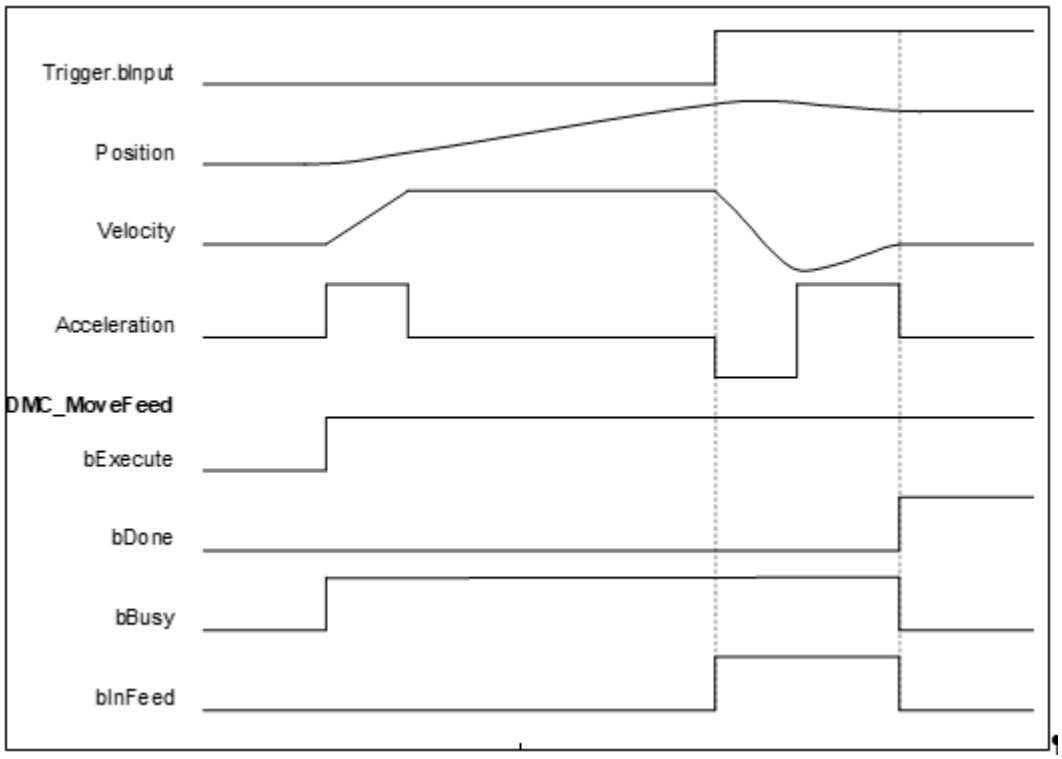
- **Programming Example**

  - **Sample Program 1**:

The example shows the execution result of using DMC_MoveFeed in controller mode.



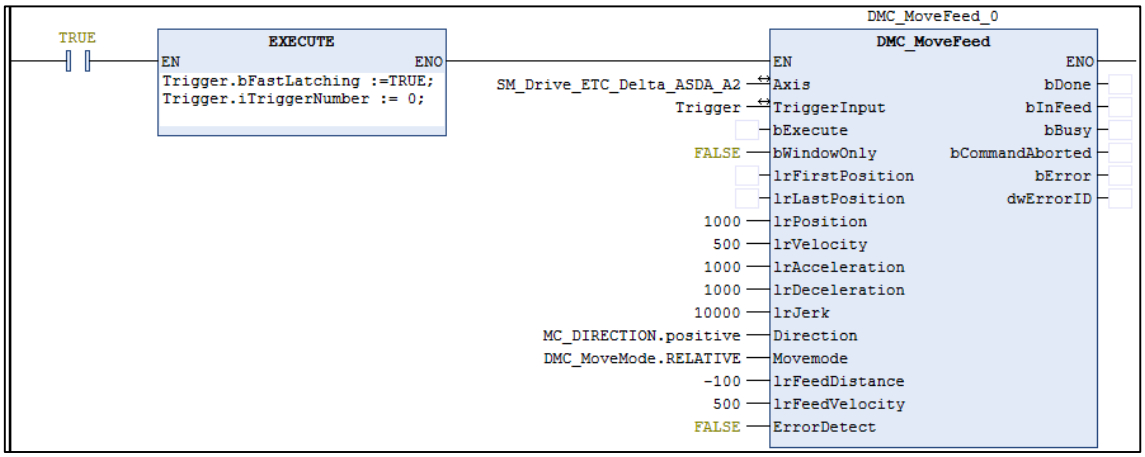
  - **Timing Diagram**



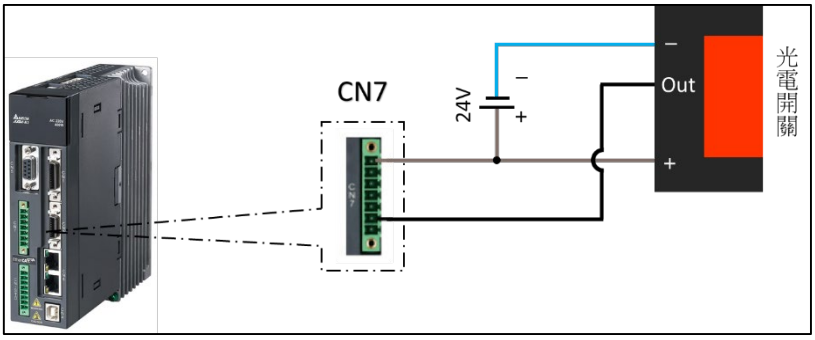
1. After DMC_MoveFeed has started, the axis starts to run with parameters such as target position, velocity, acceleration and motion mode input by the Function block, and waits for the Trigger signal of the controller mode.
2. After the trigger signal of the controller mode is triggered, the axis will move according to the position and speed of the second standard movement.
3. Since the standard distance (lrFeedDistance) is a positive value, the axis maintains the original movement direction and performs standard movement after triggering.

- **Sample Program 2**:

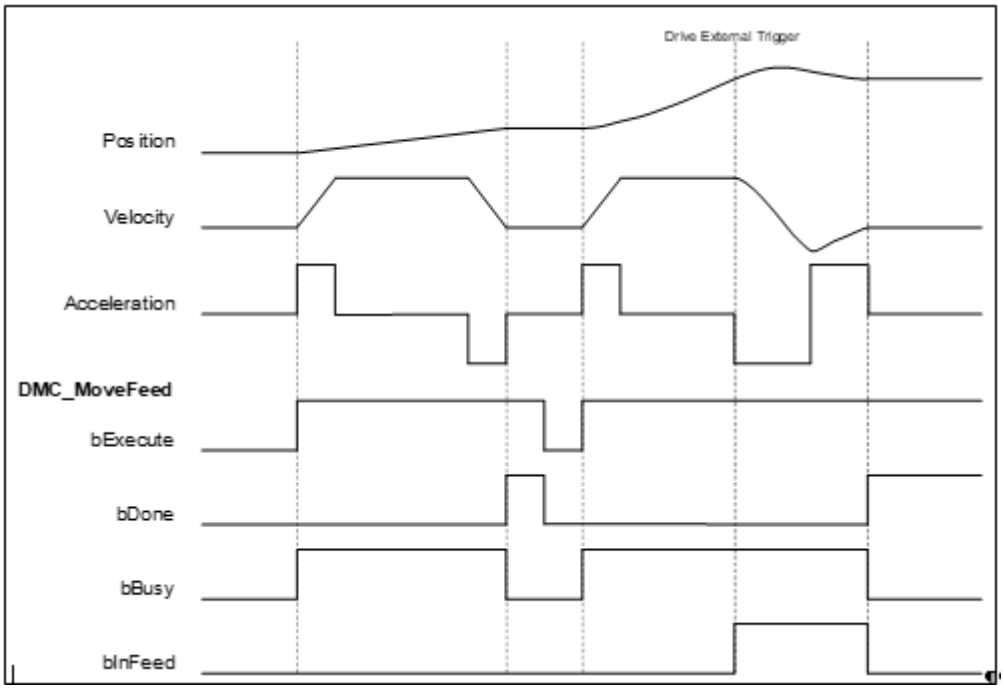
Example to illustrate the execution result of using DMC_MoveFeed in drive mode.



- **Wiring Diagram**



- **Timing Diagram**



1. After DMC_MoveFeed has started, the axis starts to run with parameters such as target position, velocity, acceleration and motion mode input in the Function block, and waits for the Trigger signal in the controller mode.
2. Since the first segment of the target movement uses the relative mode (RELATIVE), and the error detection selection (ErrorDetect) is False, when the target position is reached, the DMC_MoveFeed operation is completed, and bDone turns to True.
3. Restart DMC_MoveFeed, and trigger the external signal of the driver when the first segment of target movement has not been completed.
4. After triggering, the axis will follow the position and speed of the second standard movement. Since the standard distance (IrFeedDistance) is negative, the axis will run in the opposite direction after triggering.
5. The trigger position of the drive mode can be obtained by querying the Touch probe pos1 pos value (60BAh). Since the Drive trigger is more real-time than the controller, there will be a slight error in observing the relationship between the InFeed and the position.

2.3.1.16 DMC_GroupReadSetPosition

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupReadSetPosition reads the current Instruction position of the axis group.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupReadSetPosition	

ST Language

```
DMC_GroupReadSetPosition_instance(
AxisGroup: = ,
bEnable: = ,
CoordSystem: = ,
bValid=> ,
bBusy=> ,
bError=> ,
ErrorID=> ,
Position=> ,
KinematicConfig=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	Implement instruction when *bEnable* turns to True.	BOOL	True/False (False)	-
CoordSystem	Coordinate system	DMC_COORD_SYSTEM*	0: ACS 1: MCS 2: WCS (Reserved) 3: PCS_1 (Reserved) 4: PCS_2 (Reserved) 5: TCS (Reserved) (1)	When *bEnable* turns to True, the setting parameters of CoordSystem will be updated.

***Note**: DMC_COORD_SYSTEM: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output range value (Default Value)
bValid	True when the output value is valid.	BOOL	True/False (False)
bBusy	True when the Instruction is triggered to execute.	BOOL	True/False (False)
bError	True when an Instruction error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when the instruction error	DMC_ERROR*1	DMC_ERROR

Name	Function	Data Type	Output range value (Default Value)
	occurs. For the detailed description of the error code, refer to the appendix of the manual.		（DMC_NO_ERROR）
Position	The current Instruction position of the axis group in the set CoordSystem.	LREAL[6]	[_, _, _, _, _, _] Positive value, negative value or 0 ([0，0，0，0，0，0])
KinematicConfig	When the CoordSystem is set as the cassette coordinate system (that is, when it is not ACS), the configuration and Data Type corresponding to the current Instruction position of the axis group.	DL_Kinematics.CONFIG_DATA*2 (Reserved)	-

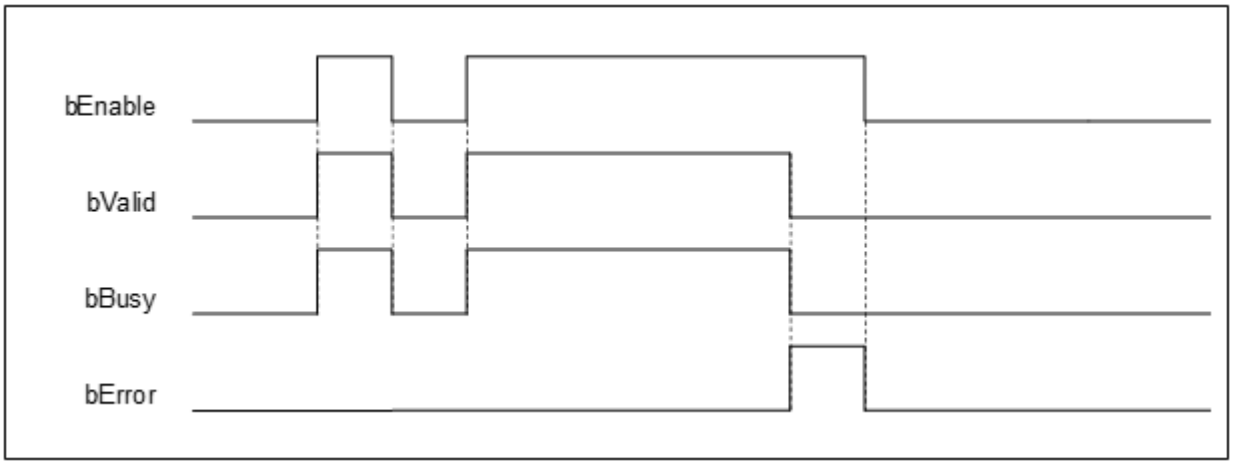
***Note**:

1. DMC_ERROR: Enumeration (Enum)
2. Depending on the configuration, there are different storage data.

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bValid	• When *bEnable* turns to True and IrValueOutput is valid.	• When *bEnable* turns to True. • When *bError* turns to True.
bBusy	• When *bEnable* turns to True.	• When *bValid* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution condition of the Instruction or the input value.	• When *bEnable* turns to True (Clear the error code of the ErrorID record).
ErrorID		
Position	• Continuously update the value when *bEnable* is True.	• Continuously update the value when *bEnable* is True.
KinematicConfig	• Continuously update the value when *bEnable* is True.	• Continuously update the value when *bEnable* is True.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect

AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*	DMC_AXIS_ GROUP_REF	When *bEnable* turns to True and *bBusy* is False

***Note**: DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

  - When the axis group state is not GroupDisable, this Function block will only effectively output the value.
  - If the read position is a cassette coordinate system, KinematicConfig will output the configuration and attitude data corresponding to the Position; if the coordinate system is ACS, this pin is meaningless.
  - Function of CoordSystem needs to be supported by DL_MotionControl library V1.1.0.0 or above.

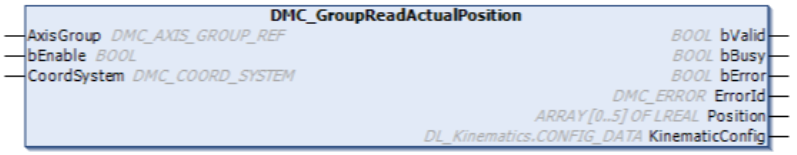
- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *bError* will turn to True and the axis motion will stop. You can refer to the content of ErrorID (error code) to confirm the current error status.
  - For the error codes and corresponding Troubleshooting methods, refer to the **Appendix** of this manual.

2.3.1.17 DMC_GroupReadActPosition

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupReadActPosition reads the current actual position of the axis group.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupReadActPosition	

ST Language

```
DMC_GroupReadActPosition_instance(
AxisGroup: = ,
bEnable: = ,
CoordSystem: = ,
bValid=> ,
bBusy=> ,
bError=> ,
ErrorID=> ,
Position=> ,
KinematicConfig=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	When *bEnable* turns to True, run the instruction.	BOOL	True/False (False)	-
CoordSystem	Coordinate system	DMC_COORD_SYSTEM*	0: ACS 1: MCS 2: WCS (Reserved) 3: PCS_1 (Reserved) 4: PCS_2 (Reserved) 5: TCS (Reserved) (1)	When *bEnable* turns to True, the setting parameters of CoordSystem will be updated.

***Note**: DMC_COORD_SYSTEM: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output range value (Default Value)
bValid	True when the output value is valid.	BOOL	True/False (False)
bBusy	True when the Instruction is triggered to run.	BOOL	True/False (False)
bError	True when an Instruction error occurs.	BOOL	True/False (False)

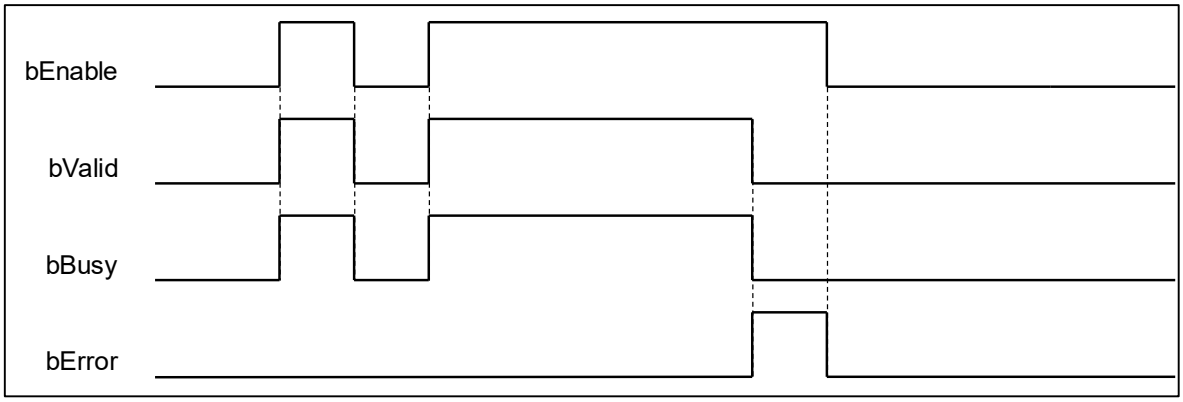
Name	Function	Data Type	Output range value (Default Value)
ErrorID	Record the error code when the instruction error occurs. For the detailed description of the error code, refer to the appendix of the manual.	DMC_ERROR*1	DMC_ERROR （DMC_NO_ERROR）
Position	The current Instruction position of the axis group in the set CoordSystem.	LREAL[6]	[_，_，_，_，_，_] Positive value, negative value or 0 ([0，0，0，0，0，0])
KinematicConfig	When the CoordinateSystem is set as the cassette Coordinate system (that is, when it is not ACS), the configuration and attitude data corresponding to the current Instruction position of the axis group.	DL_Kinematics.CONFIG_DATA*2 (Reserved)	-

***Note**:
1. DMC_ERROR: Enumeration (Enum)
2. Depending on the configuration, there are different storage data.

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bValid	• When *bEnable* turns to True and IrValueOutput is valid.	• When *bEnable* turns to True. • When *bError* turns to True.
bBusy	• When the rising edge of *bEnable* is triggered.	• When *bValid* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution condition of the Instruction or the input value.	• When *bEnable* turns to True (Clear the Error Code).
ErrorID		
Position	• Continuously update the value when *bEnable* is True.	• Continuously update the value when *bEnable* is True.
KinematicConfig	• Continuously update the value when *bEnable* is True.	• Continuously update the value when *bEnable* is True.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the	DMC_AXIS_GROUP_REF*	DMC_AXIS_	When *bEnable* turns to True and *bBusy*

	axis group.		GROUP_REF	is False.

***Note:** DMC_AXIS_GROUP_REF(FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

  - When the axis group state is not GroupDisable, this Function block will only effectively output the value.
  - If the read position is a cassette Coordinate system, KinematicConfig will output the configuration and attitude data corresponding to the Position; if the Coordinate system is ACS, this pin is meaningless.
  - Function of CoordSystem needs to be supported by DL_MotionControl library V1.1.0.0 or above.

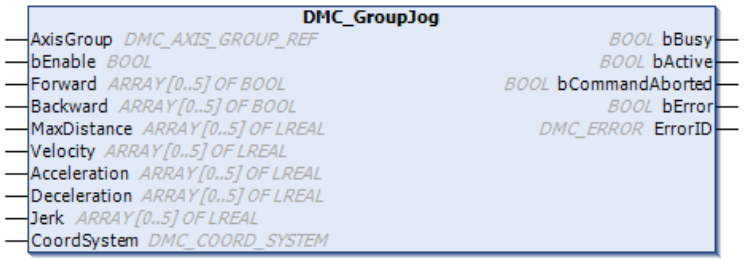
- **Troubleshooting**

  - If an error occurs during the execution of the Instruction, *bError* will turn to True and the axis motion will stop. You can refer to the content of ErrorID (error code) to confirm the current error status.
  - For error codes and corresponding Troubleshooting methods, refer to the **Appendix** of this manual.

2.3.1.18 DMC_GroupJog

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupJog is used to activate the forward and reverse jog Function of the axis group to the specified coordinates.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupJog	

ST Language

```
DMC_GroupJog_instance(
AxisGroup: = ,
bEnable: = ,
Forward: = ,
Backward: = ,
MaxDistance: = ,
Velocity: = ,
Acceleration: = ,
Deceleration: = ,
Jerk: = ,
CoordSystem: = ,
bBusy=> ,
bActive=> ,
bCommandAborted=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	When *bEnable* turns to True, implement instruction	BOOL	True/False (False)	-
Forward	Run the forward jog of each coordinate axis	BOOL[6]	[_, _, _, _, _, _] True/False ([_, _, _, _, _, _] False)	Only works when *Enable*=True

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Backward	Run the reverse jog of each coordinate axis	BOOL[6]	[_, _, _, _, _, _] True/False ([_, _, _, _, _, _] False)	Only works when *Enable*=True
MaxDistance	Set the maximum moving distance of one jog of each coordinate axis	LREAL[6]	[_, _, _, _, _, _] positive, negative or 0 ([0，0，0，0，0，0])	Only works when *Enable*=True
Velocity	Set the maximum speed of each coordinate axis inching	LREAL[6]	[_, _, _, _, _, _] Positive ([0，0，0，0，0，0])	Only works when *Enable*=True
Acceleration	Set the maximum acceleration of each coordinate axis inching	LREAL[6]	[_, _, _, _, _, _] Positive ([0，0，0，0，0，0])	Only works when *Enable*=True
Jerk	Set the maximum jerk of each coordinate axis inching	LREAL[6]	[_, _, _, _, _, _] Positive ([0，0，0，0，0，0])	Only works when *Enable*=True
CoordSystem	Coordinate system	DMC_COORD_SYSTEM	0: ACS 1: MCS 2: WCS (Reserved) 3: PCS_1 (Reserved) 4: PCS_2 (Reserved) 5: TCS (Reserved) (0)	Only works when *Enable*=True

***Note:** DMC_COORD_SYSTEM: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output Range Value (Default Value)
bValid	True when the Instruction is triggered to execute	BOOL	True/False (False)
bBusy	True when inching is run	BOOL	True/False (False)
bCommand Aborted	True when the Instruction is interrupted	BOOL	True/False (False)
bError	True when an Instruction error occurs	BOOL	True/False (False)
ErrorID	Record the error code when the instruction error occurs. For the detailed description of the error code, refer to the appendix of the manual.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

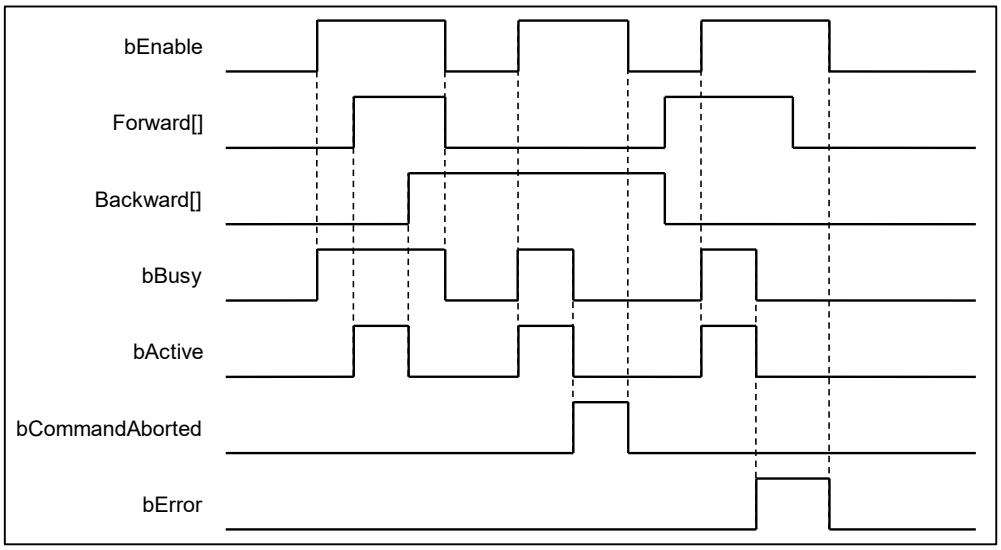
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bValid	• When *bEnable* turns to True.	• When *bEnable* turns to True.

Name	Timing for shifting to True	Timing for shifting to False
		• When *bError* turns to True.
bBusy	• When the forward/backward upper edge starts jogging.	• When *bValid* turns to True. • When *bError* turns to True.
bCommand Aborted	• When this function block instruction is interrupted by another instruction whose buffer mode is set to Aborting. • When this function block instruction is interrupted by MC_Stop. • When this function block instruction is interrupted by DMC_GroupStop.	• When *bEnable* turns to True. • *bCommandAborted* will change to False after remaining True for one cycle when *bExecute* is False but *bCommandAborted* changes to True.
bError ErrorID	• When an error occurs in the execution condition of the Instruction or the input value.	• When *bEnable* turns to True (Clear the Error Code).

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*	DMC_AXIS_ GROUP_REF	When *bEnable* turns to True and *bBusy* is False

***Note:** DMC_AXIS_GROUP_REF(FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

  - Function of CoordSystem needs to be supported by DL_MotionControl library V1.1.0.0 or above.
  - When bEnable is TRUE, according to the Coordinate system specified by CoordSystem, the value of MaxDistance / Velocity / Acceleration / Deceleration / Jerk determines the relevant motion parameters of each coordinate axis of the Coordinate system, and uses Forward / Backward as the switch to start the forward and reverse directions of each coordinate axis Inching.
  - When bEnable is FALSE, the axis group will stop jogging immediately and decelerate to 0.
  - Only when one of Forward and Backward is TRUE will start the Jog motion of the coordinate axis.
  - When MaxDistance is set to 0, there is no movement range limit.
  - MaxDistance / Velocity / Acceleration / Deceleration / Jerk will not affect the current inching movement after modification, and need to restart Forward / Backward to take effect.

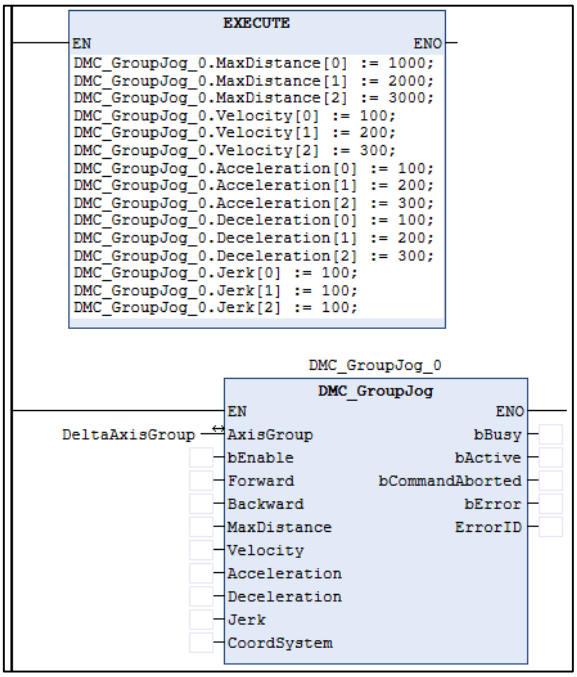
- Modifying the CoordSystem breaks jogging in all directions. A new jog needs to be restarted for Forward / Backward.
- When any axis in the axis group is jogging, the GroupState will become GroupMoving, and the axis states of all axes in the axis group will become synchronized_motion; after the jogging ends, the GroupState will become GroupStandby, and the axis states of all axes in the axis group will become standstill.
- DMC_GroupJog cannot interrupt other motion function blocks, and can only be run when the axis group state is GroupStandby.

- **Troubleshooting**

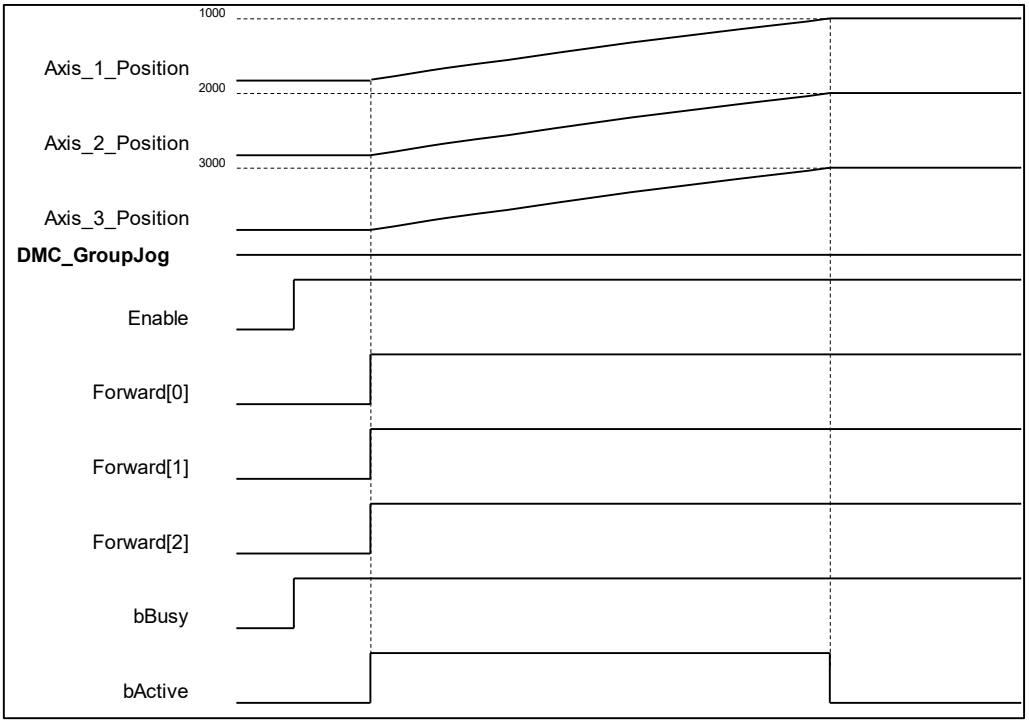
  - If an error occurs during the execution of the Instruction, *bError* will turn to TRUE and the axis motion will stop. You can refer to ErrorID (error code) to confirm the current error status.
  - For error codes and corresponding troubleshooting methods, refer to the **Appendix** of this manual.

- **Programming Example**

  - This example shows how to use DMC_GroupJog to control axis group motion and make 3 axes move.



  - Timing Diagram

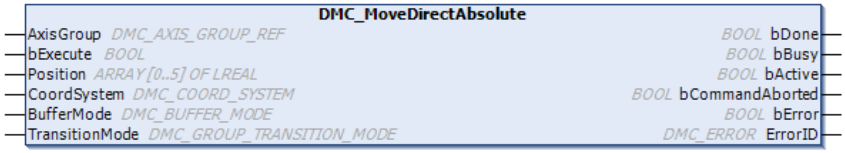


- When the Enable of DMC_GroupJog is True, the Forward[0]~ Forward[2] pins are activated, and the axis will start to run to the MaxDistance setting position and then stop running.
- When any axis in the axis group is running, bActive of DMC_GroupJog is True. bActive is False after the axis group is running.

2.3.1.19 DMC_MoveDirectAbsolute

- **Supported Devices**: AX-308E, AX-364E

DMC_MoveDirectAbsolute controls the axis group moving to the absolute position in the specified coordinate system. Each axis is calculated independently during the motion, and the motion path is not specified.

FB/FC	Instruction	Graphic Expression
FB	DMC_MoveDirectAbsolute	

ST Language

```
DMC_MoveDirectAbsolute_instance(
AxisGroup: = ,
bExecute: = ,
Position: = ,
CoordSystem: = ,
BufferMode: = ,
TransitionMode: = ,
bDone=> ,
bBusy=> ,
bActive=> ,
bCommandAborted=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
Position	Specify the absolute target position for each axis in the specified axis group. (Unit: user unit)	LREAL[6]	[_，_，_，_，_，_] Positive or negative value ([0，0，0，0，0，0])	When *bExecute* is on the rising edge, the setting parameters of *Position* will be updated.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
CoordSystem	Coordinate system	DMC_COORD_SYSTEM*1	0: ACS 1: MCS 2: WCS (Reserved) 3: PCS_1 (Reserved) 4: PCS_2 (Reserved) 5: TCS (Reserved) (0)	When *bExecute* is on the rising edge, the setting parameters of *CoordSystem* will be updated.
BufferMode	Specifies the buffer behavior mode for this function block instruction*2	DMC_BUFFER_MODE*2	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	When *bExecute* is on the rising edge, the setting parameters of *BufferMode* will be updated.
TransitionMode	Specifies the transition behavior mode for this function block instruction*3	DMC_GROUP_TRANSITION_MODE*3	0: None 10: Overlap (0)	When *bExecute* is on the rising edge, the setting parameters of *TransitionMode* will be updated.

***Note**:

1. DMC_COORD_SYSTEM: Enumeration (Enum)
2. About BufferMode, refer to the related information of BufferMode in **AX-3 series Instructions Manuals**.
3. About TransitionMode, refer to the related information of TransitionMode in **AX-3 series Instructions Manuals**.

- **Outputs**

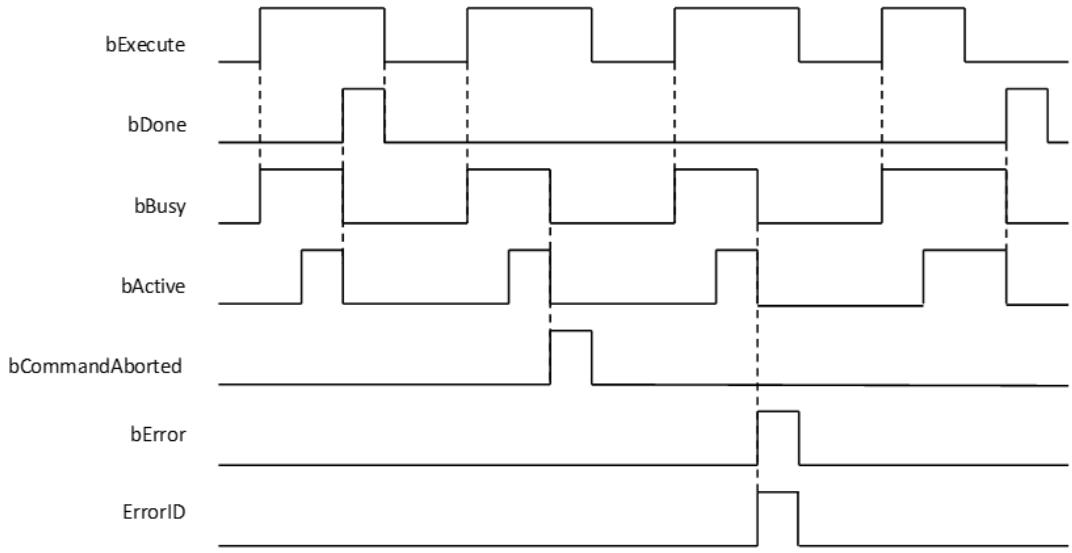
Name	Function	Data Type	Output Range Value (Default Value)
bDone	True when absolute positioning is complete	BOOL	True/False (False)
bBusy	True when the instruction is triggered for execution	BOOL	True/False (False)
bActive	True when the instruction is controlling axes.	BOOL	True/False (False)
bCommand Aborted	True when the instruction execution is aborted.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When absolute positioning is complete.	• When *bExecute* turns to False. • When *bExecute* is False but *bDone* turns to True, *bDone* will remain True for one cycle and then change to False.
bBusy	• When *bExecute* turns to TRUE.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bActive	• When axes motion starts.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True. • When *bExecute* is False but *bActive* turns to True, *bActive* will remain True for one cycle and then change to False.
bCommand Aborted	• When the instruction is interrupted by another instruction whose *BufferMode* is set to Aborting. • When the instruction is interrupted by MC_Stop. • When the instruction is interrupted by DMC_GroupStop.	• When *bExecute* turns to False. • When *bExecute* is False but *bCommandAborted* turns to True, *bCommandAborted* will remain True for one cycle and then change to False.
bError ErrorID	• When an error occurs in the execution conditions or input values of the instruction. (Error code is recorded in ErrorID)	• When *bExecute* turns to False. (Error Code is cleared)

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*	DMC_AXIS_ GROUP_REF	When *bExecute* turns to True and *bBusy* is False

***Note:** DMC_AXIS_GROUP_REF(FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

  - This function is available for DL_MotionControl V1.2.0.0 or later.
  - Each axis is calculated independently during the movement, so the movement path will vary according to the configuration used.
  - Buffer Mode only supports Aborting and Buffered. If the Buffer Mode of the subsequent motion function block is set to BlendingLow, BlendingPrevious, BlendingNext, and BlendingHigh, the actual execution will be according to Buffered.
  - The speed, acceleration, deceleration, and jerk of this function block are related to the set values of the axis group. The Velocity (1113), Acceleration (1123), Deceleration (1133), Jerk (1143) values of each single axis in the axis group can be modified by MC_WriteParameter.

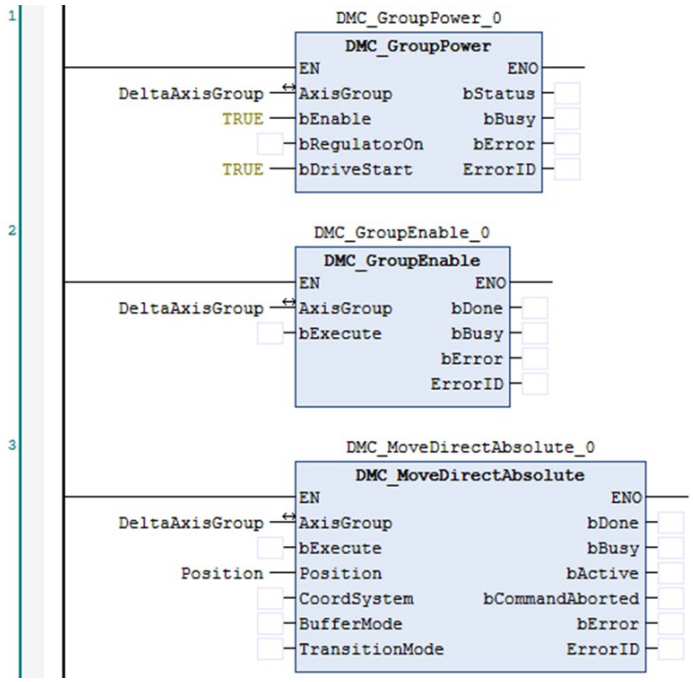
- **Troubleshooting**

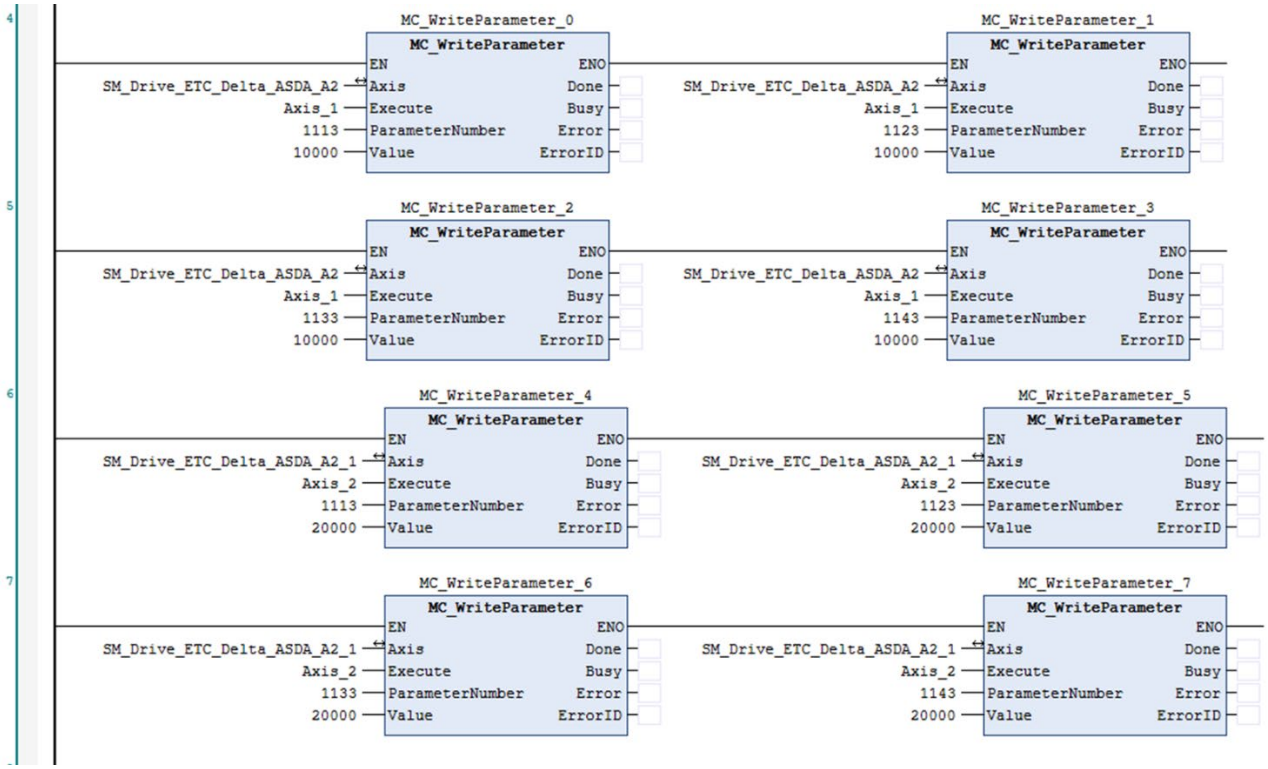
  - If an error occurs during the execution of the instruction, *bError* will turn to TRUE and the axis motion will stop. You can refer to ErrorID (error code) to confirm the current error status.
  - For error codes and corresponding troubleshooting methods, refer to the **Appendix** of this manual.

- **Programming Example**
  - This example shows how to use MoveDirectAbsolute to control axis group motion.

Axis Group	Target Position
Axis1	1000
Axis2	2000

```
DMC_GroupPower_0: DMC_GroupPower;
DMC_GroupEnable_0: DMC_GroupEnable;
DMC_MoveDirectAbsolute_0: DMC_MoveDirectAbsolute;
Position: ARRAY [0..5] OF LREAL := [10000, 20000, 4(0.0)];
MC_WriteParameter_0: MC_WriteParameter;
MC_WriteParameter_1: MC_WriteParameter;
MC_WriteParameter_2: MC_WriteParameter;
MC_WriteParameter_3: MC_WriteParameter;
MC_WriteParameter_4: MC_WriteParameter;
MC_WriteParameter_5: MC_WriteParameter;
MC_WriteParameter_6: MC_WriteParameter;
MC_WriteParameter_7: MC_WriteParameter;
Axis_1: BOOL;
Axis_2: BOOL;
```



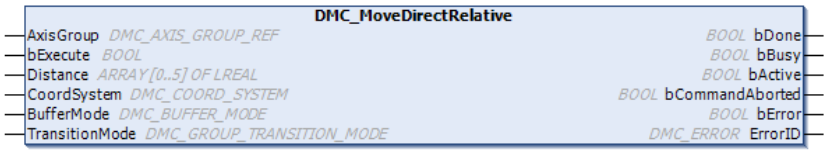


- When DMC_GroupPower bRegulatorOn is True, the single-axis status switches from Disabled to Standstill.
- When *bExecute* of DMC_GroupEnable is True, the axis group status switches from GroupDisabled to GroupStandby.
- When Axis_1 and Axis_2 are true, the parameters will be written to Velocity, Acceleration, Deceleration, and Jerk of each single axis.
- When DMC_MoveDirectAbsolute is True, each single axis will perform absolute positioning according to the set speed of the single-axis parameter.
- When the positioning of each single axis is completed, *bBusy* is False and bDone is True.

2.3.1.20 DMC_MoveDirectRelative

- **Supported Devices**: AX-308E, AX-364E

DMC_MoveDirectRelative controls the axis group moving to the relative position in the specified coordinate system. Each axis is calculated independently during the motion, and the motion path is not specified.

FB/FC	Instruction	Graphic Expression
FB	DMC_MoveDirectRelative	

ST Language

```
DMC_MoveDirectRelative_instance(
AxisGroup: = ,
bExecute: = ,
Distance: = ,
CoordSystem: = ,
BufferMode: = ,
TransitionMode: = ,
bDone=> ,
bBusy=> ,
bActive=> ,
bCommandAborted=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False （False）	-
Distance	Specify the absolute target position for each axis in the specified axis group. (User unit)	LREAL[6]	[_，_，_，_，_，_] Positive or negative value ([0，0，0，0，0，0])	When *bExecute* is on the rising edge, the setting parameters of *Position* will be updated.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
CoordSystem	Coordinate system	DMC_COORD_SYSTEM[*1]	0: ACS 1: MCS 2: WCS (Reserved) 3: PCS_1 (Reserved) 4: PCS_2 (Reserved) 5: TCS (Reserved) (0)	When *bExecute* is on the rising edge, the setting parameters of *CoordSystem* will be updated.
BufferMode	Specifies the buffer behavior mode for this function block instruction[*2]	DMC_BUFFER_MODE[*2]	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	When *bExecute* is on the rising edge, the setting parameters of *BufferMode* will be updated.
TransitionMode	Specifies the transition behavior mode for this function block instruction[*3]	DMC_GROUP_TRANSITION_MODE[*3]	0: None 10: Overlap (0)	When *bExecute* is on the rising edge, the setting parameters of *TransitionMode*will be updated.

***Note**:

1. DMC_COORD_SYSTEM: Enumeration (Enum)
2. About BufferMode, refer to the related information of BufferMode in **AX-3 series Instructions Manuals**.
3. About TransitionMode, refer to the related information of TransitionMode in **AX-3 series Instructions Manuals**.

- **Outputs**

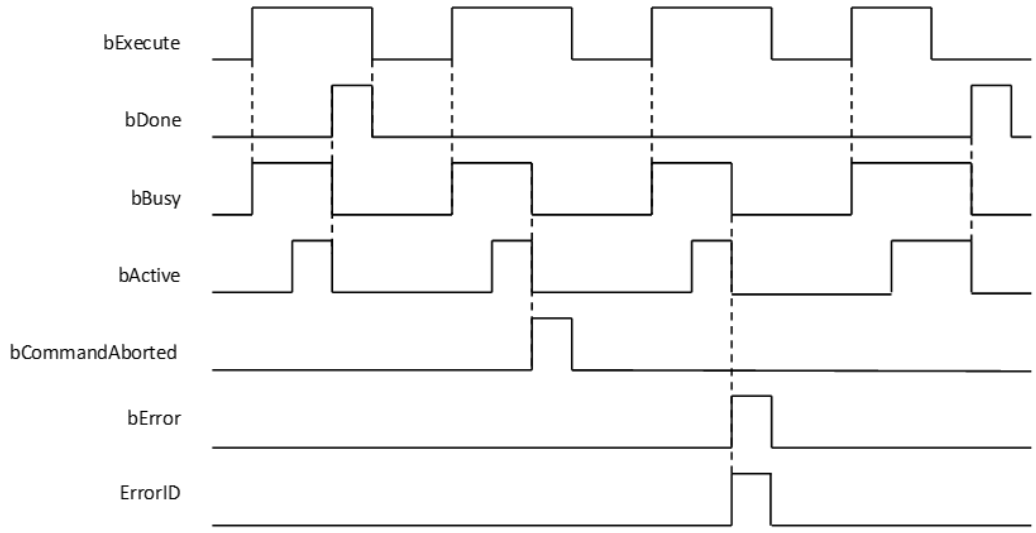
Name	Function	Data Type	Setting Value (Default Value)
bDone	When the relative positioning is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bActive	When axes start being controlled by the instruction.	BOOL	True/False (False)
bCommand Aborted	True when the instruction execution is interrupted.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the relative positioning is completed.	• When *bExecute* turns to False. • When *bExecute* is False but *bDone* turns to True, *bDone* will remain True for one cycle and then change to False.
bBusy	• When *bExecute* turns to TRUE.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bActive	• When axes start being controlled by the instruction.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True. • When *bExecute* is False but *bActive* turns to True, *bActive* will remain True for one cycle and then change to False.
bCommand Aborted	• When the instruction is interrupted by another instruction whose *BufferMode* is set to Aborting. • When the instruction is interrupted by MC_Stop. • When the instruction is interrupted by DMC_GroupStop.	• When *bEnable* turns to False. • When *bEnable* is False but *bCommandAborted* turns to True, *bCommandAborted* will remain True for one cycle and then change to False.
bError	• When an error occurs in the execution conditions or input values of the instruction. (Error code is recorded in ErrorID)	• When *bEnable* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*	DMC_AXIS_ GROUP_REF	When *bExecute* turns to True and *bBusy* is False

***Note:** DMC_AXIS_GROUP_REF(FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

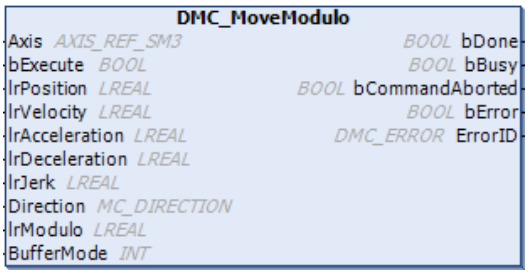
- **Function**
  - This function is available for DL_MotionControl V1.2.0.0 or later.
  - Each axis is calculated independently during the movement, so the movement path will vary according to the configuration used.
  - Buffer Mode only supports Aborting and Buffered. If the Buffer Mode of the subsequent motion function block is set to BlendingLow, BlendingPrevious, BlendingNext, and BlendingHigh, the actual execution will be according to Buffered.
  - The speed, acceleration, deceleration, and jerk of this function block are related to the set values of the axis group. The Velocity (1113), Acceleration (1123), Deceleration (1133), Jerk (1143) values of each single axis in the axis group can be modified by MC_WriteParameter.
- **Troubleshooting**
  - If an error occurs during the execution of the instruction, *bError* will turn to TRUE and the axis motion will stop. You can refer to ErrorID (error code) to confirm the current error status.
  - For error codes and corresponding troubleshooting methods, refer to the **Appendix** of this manual.
- **Programming Example**
  - Refer to DMC_MoveDirectAbsolute.

2.3.1.21 DMC_MoveModulo

- **Supported Devices**: AX-308E, AX-364E

DMC_MoveModulo is used for modulo positioning and specifies the number of rotation turns.

FB/FC	Instruction	Graphic Expression

FB	DMC_MoveModulo	

ST Language

```
DMC_MoveModulo_instance(
Axis : =,
bExecute : =,
lrPosition: =,
lrVelocity: =,
lrAcceleration: =,
lrDeceleration: =,
lrJerk: =,
Direction: =,
lrModulo: =,
BufferMode: =,
bDone=>,
bBusy =>,
bCommandAborted =>,
bError =>,
ErrorID =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
lrPosition	Absolute target position (User unit)	LREAL	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
lrVelocity	Target speed (User unit)	LREAL	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
lrAcceleration	Acceleration (User unit)	LREAL	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
lrDeceleration	Deceleration (User unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
lrJerk	Jerk (User unit/s^3)	LREAL	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
Direction	Motion direction	MC_DIRECTION*	3: fastest 2: current 1: positive 0: shortest -1: negative	When *bExecute* turns to True and *bBusy* is False.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
			(shortest)	
lrModulo	Modulo	LREAL	Positive or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
BufferMode	(Reserved)	-	-	-

***Note**: MC_DIRECTION: Enumeration (Enum)

- **Outputs**

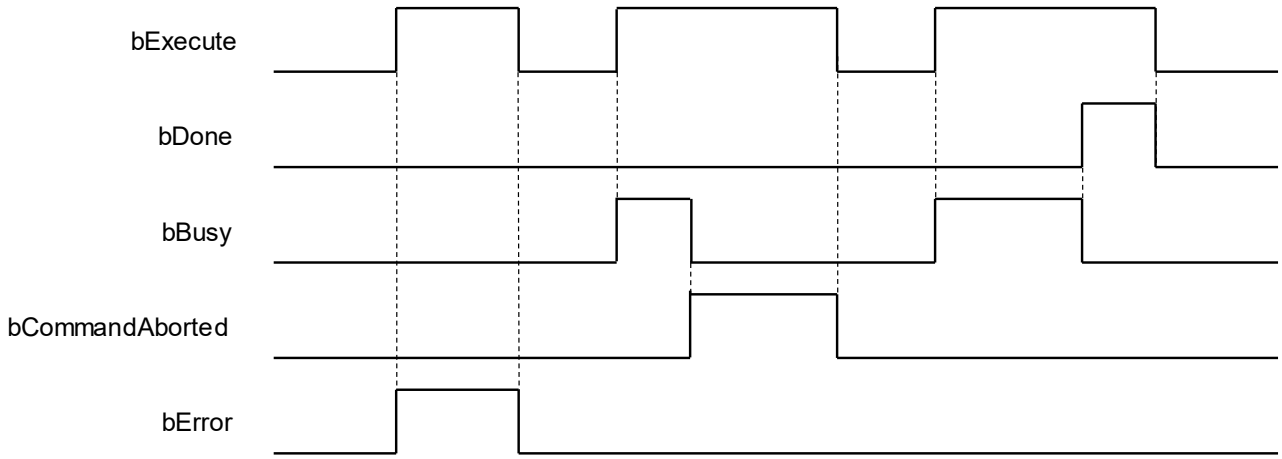
Name	Function	Data Type	Setting Value (Default Value)
bDone	True when the slave axis is performing positioning motion.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bCommand Aborted	True when the instruction execution is interrupted.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When motion is completed.	• When *bExecute* turns to False. • When *bExecute* is False but *bDone* turns to True, *bDone* will remain True for one cycle and then change to False.
bBusy	• When *bExecute* turns to TRUE.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bActive	• When the axis motion starts.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True. • When *bExecute* is False but *bActive* turns to True, *bActive* will remain True for one cycle and then change to False.
bCommand Aborted	• When the instruction is interrupted by another instruction whose *BufferMode* is set to Aborting. • When the instruction is interrupted by MC_Stop. • When the instruction is interrupted by DMC_GroupStop.	• When *bEnable* turns to False. • When *bEnable* is False but *bCommandAborted* turns to True, *bCommandAborted* will remain True for one cycle and then change to False.
bError	• When an error occurs in the execution conditions or input values of the instruction. (Error Code is recorded in *ErrorID*)	• When *bEnable* turns to False. (Error Code is cleared)
ErrorID		
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *Execute* turns to True and *bBusy* is False

***Note**: AXIS_REF_SM3(FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

  - This function is available for DL_MotionControl V1.2.0.0 or later.
  - Absolute position (lrPosition) and modulus (lrModulo) relationship
    - Absolute position in modulo (lrPosition<lrModulo)

      The final positioning is within the modulo.
    - Absolute position outside the modulo (lrPosition>lrModulo)

      The final positioning is outside the modulo, running n times modulo distance.
  - Direction mode:
    - Positive—Only allow forward positioning
      - Absolute position ahead of current position: move forward to the target position of the next modulo.
      - Absolute position lags behind the current position: move forward to the target position of the next modulo.
    - Negative—Only allow reverse positioning
      - Absolute position ahead of current position: move in reverse to the target position of the modulo.
      - Absolute position lags behind current position: move in reverse to the target position of the last modulo.
    - Current—Current motion direction positioning
      - Currently running forward, absolute position ahead of current position: move forward to the target position of the next modulo.
      - Currently running forward, absolute position lags behind current position: move forward to the target position of the modulo.
      - Currently running in reverse, absolute position ahead of current position: move in reverse to the target position of the modulo.
      - Currently running in reverse, absolute position lags behind current position: move in reverse to the target position of the last modulo.
    - Shortest—Shortest distance positioning
      - Absolute position ahead of the current position for greater than 0.5 modulo: move in reverse to the target position of the next modulo.
      - Absolute position ahead of the current position for smaller than 0.5 modulo: move in reverse to the target position of the modulo.
      - Absolute position lags behind the current position for greater than 0.5 modulo: move in reverse to the target position of the last modulo.

- Absolute position lags behind the current position for smaller than 0.5 modulo: move forward to the target position of the modulo.

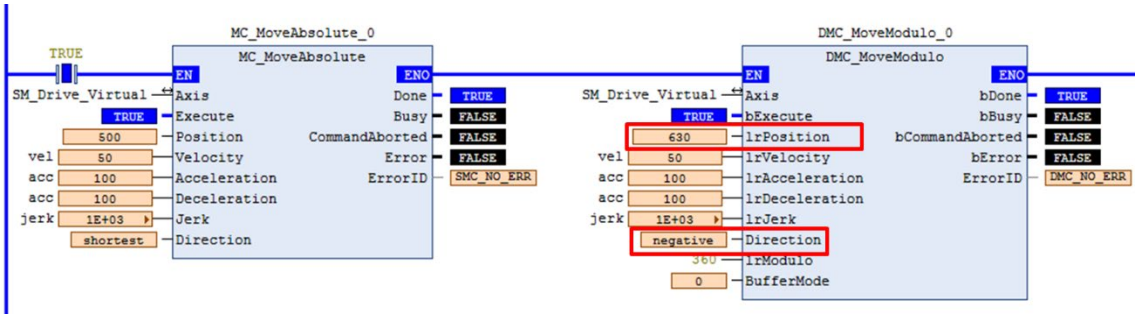
- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *bError* will turn to TRUE. You can refer to ErrorID (error code) to confirm the current error status.

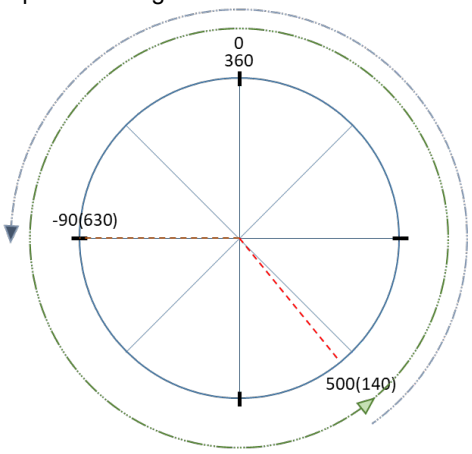
- **Programming Example**

  - Example 1:

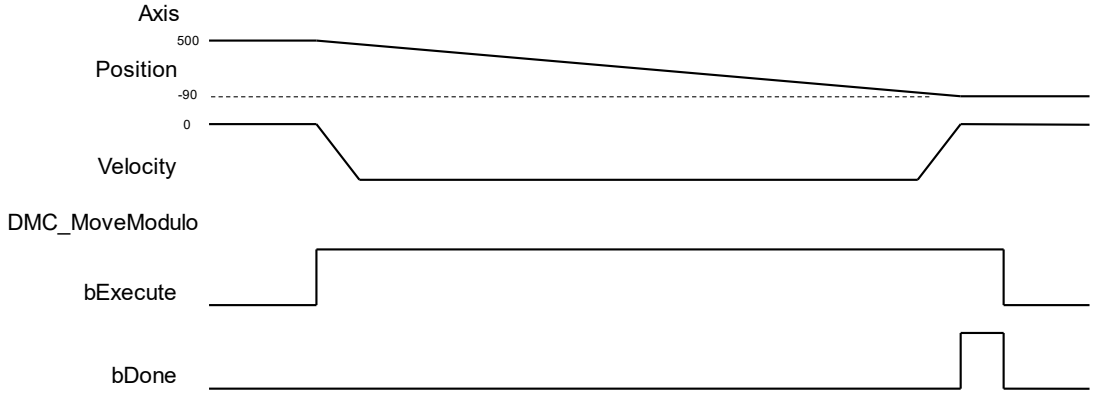
The following image shows the operation of target position exceeding the modulo and lagging behind the current position when DMC_MoveModulo is in negative mode.



  - Operation Diagram



■ Timing Diagram

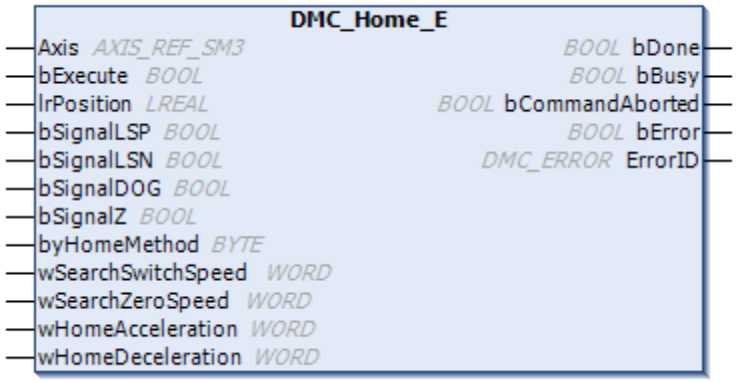


1. Enter 630 for absolute position (lrPosition), 360 for modulo, so the positioning will run more than one turn and then to the relative position in the corresponding module, which is 270.
2. Enter negative for Direction, which can only run in reverse to the target position.
3. The current position is 500, according to the above instructions, after running one modulo (one turn) in reverse, the current position is 140.
4. When it reaches 140, it will move to the -90 position in reverse. (Corresponding modulo position is 270)

2.3.1.22 DMC_Home_E

- **Supported Devices**: AX-308E, AX-364E

DMC_Home_E controls and plans homing.

FB/FC	Instruction	Graphic Expression
FB	DMC_Home_E	

ST Language

```
DMC_Home_E_instance(
Axis: = ,
bExecute: = ,
lrPosition: = ,
bSignalLSP: = ,
bSignalLSN: = ,
bSignalDOG: = ,
bSignalZ: = ,
byHomeMethod: = ,
wSearchSwitchSpeed: = ,
wSearchZeroSpeed: = ,
wHomeAcceleration: = ,
wHomeDeceleration: = ,
bDone=> ,
bBusy=> ,
bCommandAborted=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
lrPosition	Specifies the position of the axis after the homing motion is complete.	LREAL	Positive, negative, or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
bSignalLSP	Positive limit signal	BOOL	True/False (False)	When *bExecute* turns to True and *bBusy* is False.
bSignalLSN	Negative limit signal	BOOL	True/False	When *bExecute* turns to

			(False)	True and *bBusy* is False.
bSignalDOG	DOG signal	BOOL	True/False (False)	When *bExecute* turns to True and *bBusy* is False.
bSignalZ	Z signal	BOOL	True/False (False)	When *bExecute* turns to True and *bBusy* is False.
byHomeMethod*	Homing mode	BYTE	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
wSearchSwitchSpeed	First-phase speed (homing speed when searching for a switch)	WORD	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
wSearchZeroSpeed	Second-phase speed (homing speed when searching for zero)	WORD	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
wHomeAcceleration	Homing acceleration	WORD	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
wHomeDeceleration	Homing deceleration (Reserved)	WORD	-	-

***Note**: Refer to A.4DMC_Home_P

- **Outputs**

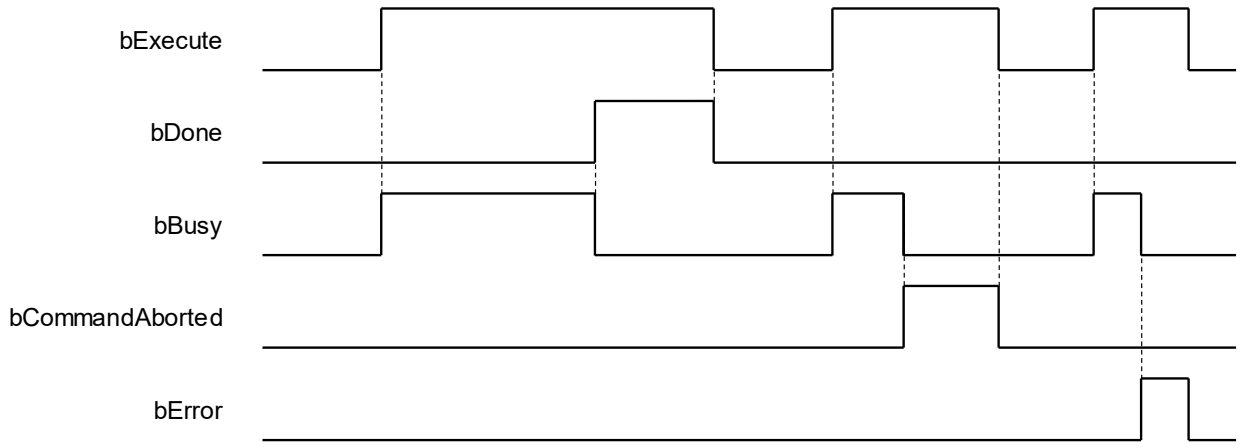
Name	Function	Data Type	Setting Value (Default Value)
bDone	True when homing is completed.	BOOL	True/False (False)
bBusy	True when the instruction is enabled.	BOOL	True/False (False)
bCommandAborted	True when the instruction execution is interrupted.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR(DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• True when homing is completed.	• When *bExecute* turns to False. • When *bError* turns to True.
bBusy	• When *bExecute* turns to TRUE.	• When *bExecute* turns to False. • When *bError* turns to True.
bCommandAborted	• When the instruction is interrupted by MC_Stop.	• When *bExecute* turns to False. • When *bExecute* is False but bCommandAborted turns to True, bCommandAborted will remain True for one cycle and then change to False.
bError	• When an error occurs in the execution conditions or input values of the instruction.	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *bEnable* turns to True and *bBusy* is False

***Note**: AXIS_REF_SM3(FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

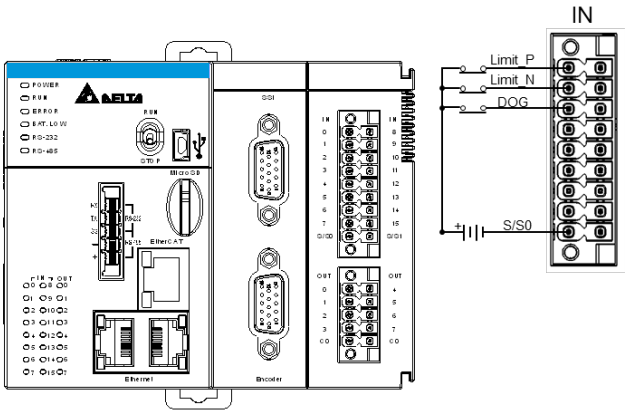
  - This function is available for DL_MotionControl V1.2.0.0 or later.
  - This function block is only available when the axis state is standstill. If run in other states, there will be errors.
  - DMC_Home_E supports a variety of homing modes defined in CiA 402. For more information on homing modes, see the Appendix.
  - bSignalLSP (positive limit signal), bSignalLSN (negative limit signal), bSignalDOG (DOG signal), bSignalZ (Z signal) signals, controlled by function block input, can be used with DIO to map function block input.
  - wSearchSwitchSpeed (first-phase speed), wSearchZeroSpeed (second-phase speed), wHomeAcceleration (homing acceleration) units are based on that defined in lower drivers.
  - If using bSignalZ, pay attention to the EtherCAT Task cycle time and signal response time. For example, if the EtherCAT Task cycle is 2ms, but the Z signal only maintained 1ms as ON, DMC_Home_E cannot capture the signal source response.

- **Troubleshooting**

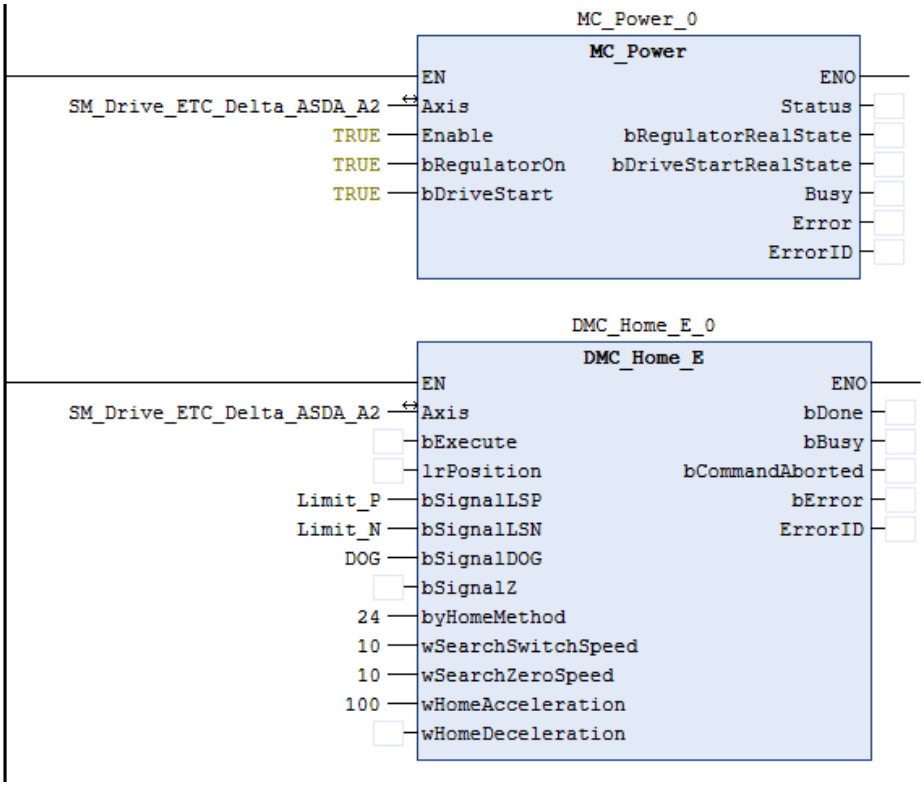
  - When an error occurs during the execution of instructions or the axis group enters “Errorstop” state, *bError* changes to True and the axes stops running. To confirm the current error state, see the error code in *ErrorID*.

- **Example**
  - This example shows how to use DMC_Home_E to run homing motion of upper computer.
  - The homing related signal can be set by DIO and program variables to map the inputs to DMC_Home_E.
  - Input MC_Power and DMC_Home_E input (Axis), as shown in the following figure, when the axis status is standstill, then start this function block, the homing motion is run as set, at which time the state will switch from standstill to Homing.

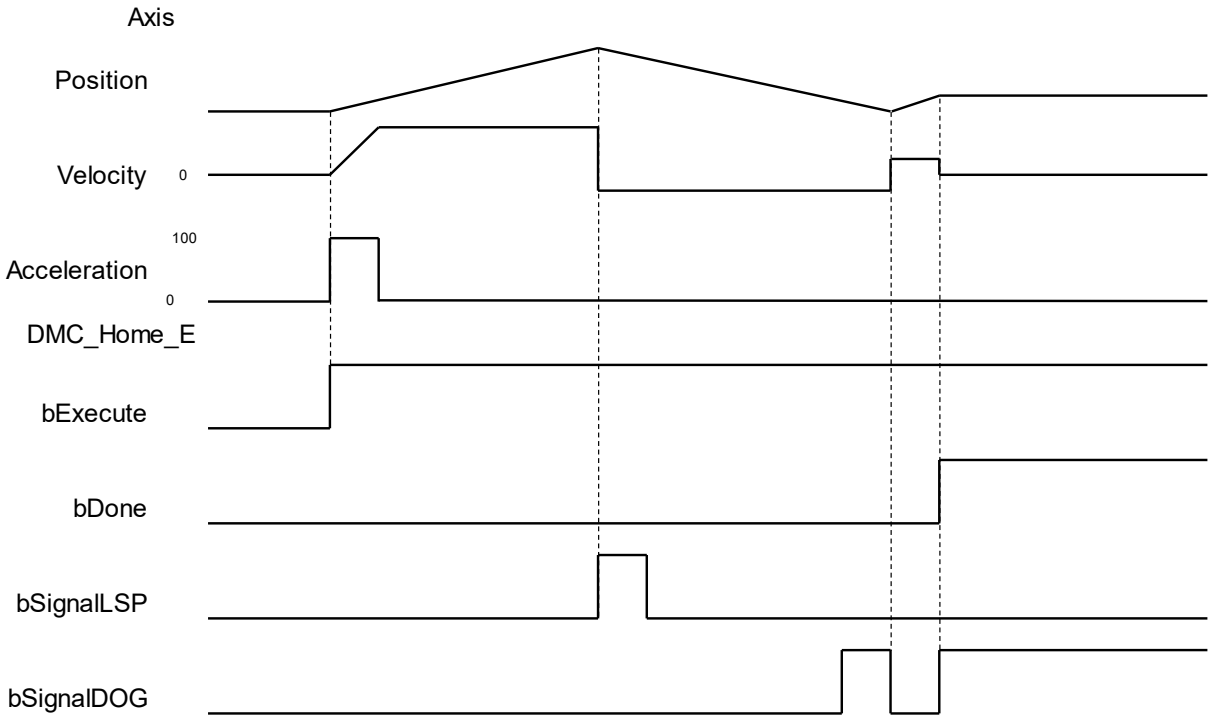
  - DOG Signal Hardware Configuration



- Homing mode 24 example



■ Timing Diagram



- Set Homing mode as Mode 24.
- After encountering the positive limit switch, reverse to search the DOG signal. Receive the DOG signal, maintain the direction, and wait to disengage the DOG signal; After disengaging, reverse until the DOG signal is True, and then homing is completed.

2.3.2 Administrative Instructions

Administrative instructions generally refer to the running of the instruction to set the driver or read the relevant information without causing the actual motion of the motor. The function blocks used in this section are from the library "DL_MotionControl*" and the function blocks used can be synchronized with the driver, so when setting the axis, select the synchronous axis.

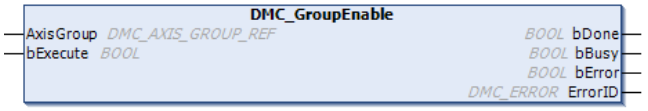
For setting up the synchronous axis, refer to section 7.4 in the *AX-3 Series Operation Manual*.

*Note: When the version of SM3_Basic is not V4.6.1.0 to match with V1.1.0.0 and earlier, an error "Type 'xxxxxx' is not equal to type 'Axis'VAR_IN_OUT 'AXIS_REF_SM3'" will appear when compiling. Change the Softmotion library version to V4.6.1.0.

2.3.2.1 DMC_GroupEnable

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupEnable switches the axis group state from GroupDisable to GroupStandby.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupEnable	

ST Language

```
DMC_GroupEnable_instance(
AxisGroup: = ,
bExecute: = ,
bDone=> ,
bBusy=> ,
bError=> ,
ErrorID=> ) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output range (Default Value)
bDone	True when the instruction is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)

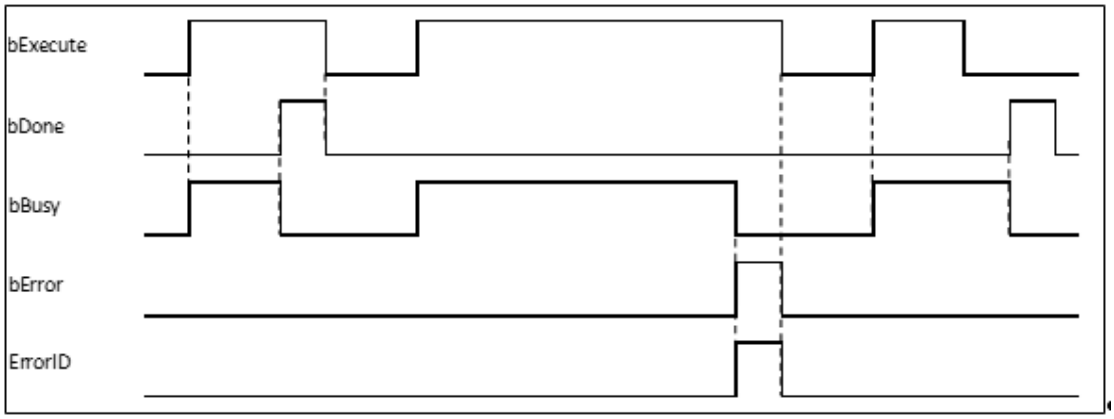
Name	Function	Data Type	Output range (Default Value)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note:** DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the instruction is completed.	• When *bExecute* turns to False. • *bDone* will change to False after remaining True for one cycle when *bExecute* is False but *bDone* changes to True.
bBusy	• When *bExecute* changes to TRUE.	• When *bDone* turns to True. • When *bError* turns to True.
bError (ErrorID)	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	• When bExecute turns from True to False. (Error Code is cleared)

- **Timing Diagram**



- **Inputs/Outputs**

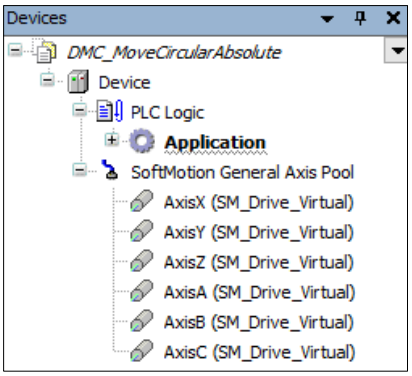
Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_ GROUP_REF*	DMC_AXIS_ GROUP_REF	When *bExecute* turns to True and *bBusy* is False.

***Note:**

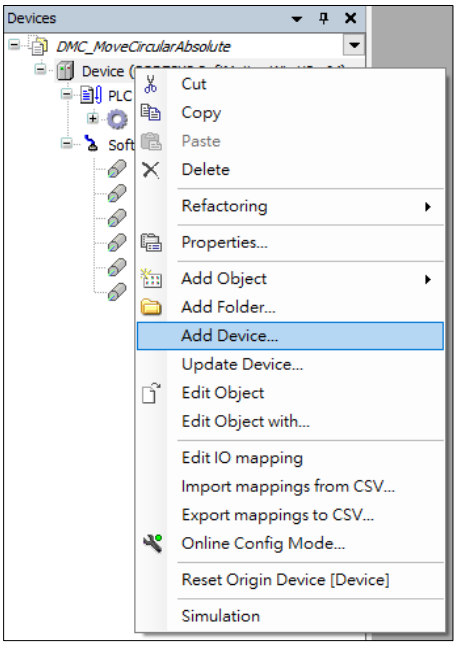
DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

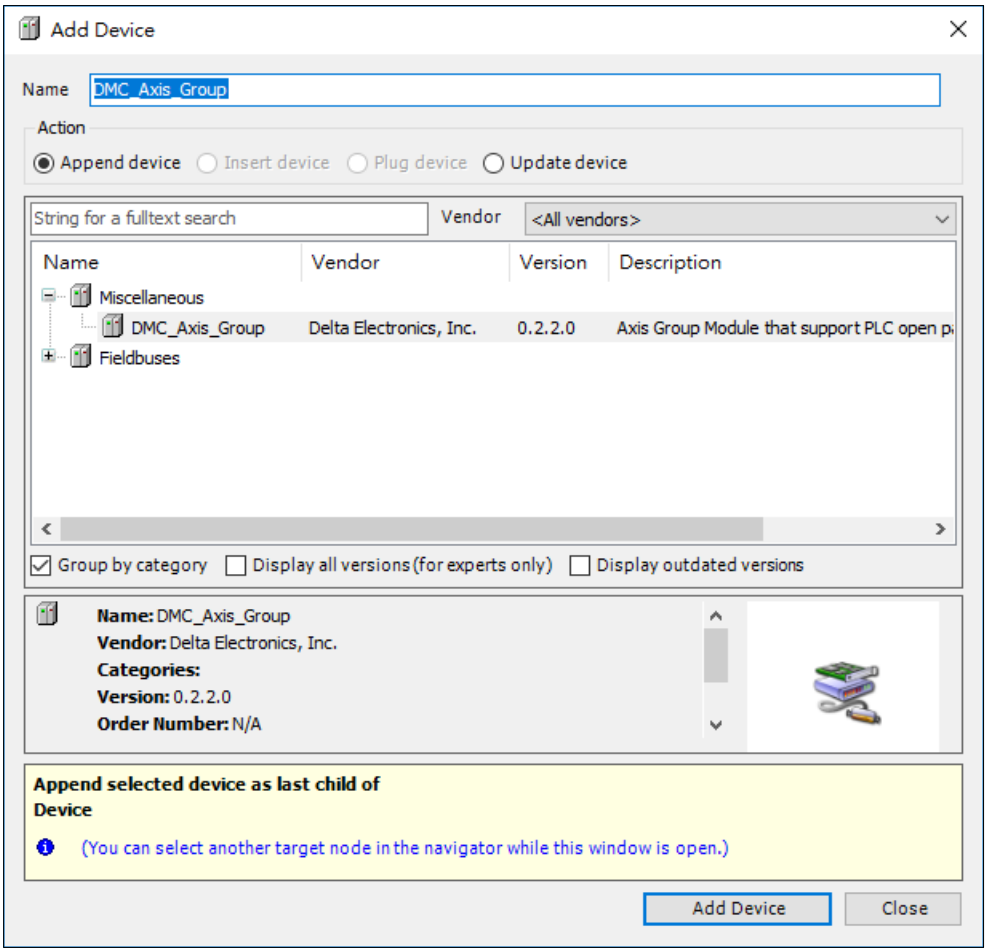
  - First, add axes to SoftMotion General Axis Pool in the project. In this example, six virtual axes have been established, i.e. AxisX, AxisY, AxisZ, AxisA, AxisB and AxisC.



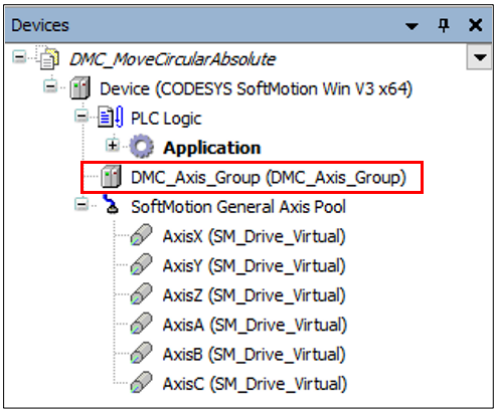
- Right-click **Device** in the project and then choose “Add Device”.



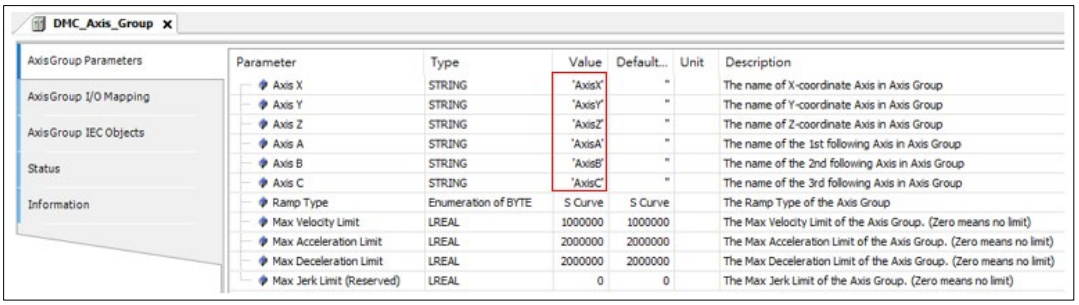
- After right-clicking **Device** and selecting **Add device,** find **DMC_ Axis_Group** and then click **Add Device**.



- Once **DMC_Axis_Group (DMC_Axis_Group)** appears in **Device**, it indicates that adding the axis group is successful.



- Click **DMC_Axis_Group** setting page and then select **AxisGroup Parameters** item. In the **Parameter** column, AxisX~AxisC represent axes 1 ~ 6 in the axis group. Fill in the value field of the Axis X ~ Axis C with the names of the previously created virtual axes “AxisX” ~ “AxisC”, as shown in the red box below. The axis group in this example uses 6 axes, AxisX, AxisY, AxisZ, AxisA, AxisB and AxisC.



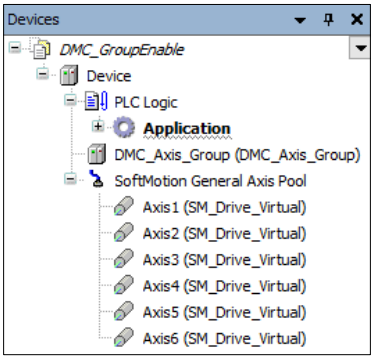
- AxisX ~ AxisC in the AxisGroup Parameters represent the axes 1 ~ 6 in the axis group respectively, which denotes a 6D space, i.e. coordinate axes X, Y, Z, A, B and C. No value is required for the coordinate axis which is not set.
- If the Value field for axes in the AxisGroup Parameters is not filled in with the names of axes, no error will occur when DMC_GroupEnable has started. However, the axis group will report an error if it starts to move the axis the name of which is not entered in the Value field.
- If there are same axis names or invalid axis names in the Value field for axes in the AxisGroup Parameters, an error will occur when DMC_GroupEnable is run.
- At least one axis is specified in the **Value** field for axes in the AxisGroup Parameters. Otherwise, an error will occur when DMC_GroupEnable is run.
- Whether the specified single axis is in Standstill state or not will not be judged when DMC_GroupEnable is run. After the axis group is created, the state of the axis group will be switched from GroupDisable to GroupStandby and the axes in the axis group will maintain the current state.
- If axes in the group are in ErrorSTOP state, the axis group state will change from GroupDisable->GroupStandby->GroupErrorStop.
- For more details on axis states, refer to Axis State Transitions in *AX-3 Series Operation Manual*.

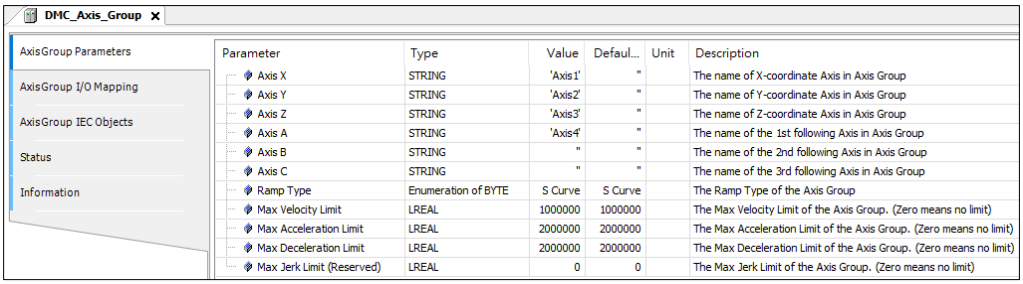
- **Troubleshooting**

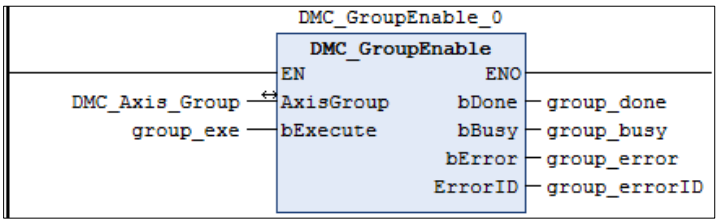
- When an error occurs during the execution of the instruction, *bError* will change to True and the axes will stop running. Refer to *ErrorID* (Error Code) to address the problem.
- For error codes and corresponding troubleshooting, refer to Appendix of this manual.

- **Programming Example**

- Based on the limitation of the number of axes for simultaneous motion and the actual demand of axes, DMC_GroupEnable switches the axis group state from GroupDisable to GroupStandby for the upcoming axis group motion.





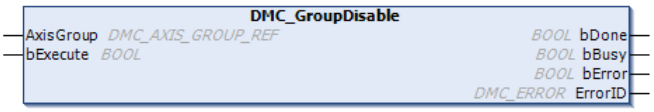


1. Before the absolute interpolation motion of Axis1~ Axis4 is performed, create Axis1~Axis4 first, add them to the axis group DMC_Axis_Group and then input Axis1~Axis4 in the Value field for Parameter AxisX~AxisA in the setting page.
2. Use DMC_GroupEnable to create the axis group first before Axis 1 ~ Axis 4 perform the absolute interpolation of simultaneous motion of four axes.
3. DMC_GroupEnable is triggered by changing group_exe (*bExecute*) to True. When group_done (bDone) changes to True, the axis group DMC_Axis_Group switches its state from GroupDisable to GroupStandby. The specified axes in the axis group maintain current state.
4. When DMC_GroupEnable is run after the axis group is created, no error occurs and the axes enter Standstill state. Then the axis group DMC_Axis_Group can be used for the interpolation of simultaneous motion.

2.3.2.2 DMC_GroupDisable

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupDisable sets the state of an axis group to GroupDisable.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupDisable	

ST Language

```
DMC_GroupDisable_instance(
AxisGroup: = ,
bExecute: = ,
bDone=> ,
bBusy=> ,
bError=> ,
ErrorID=> ) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-

- **Outputs**

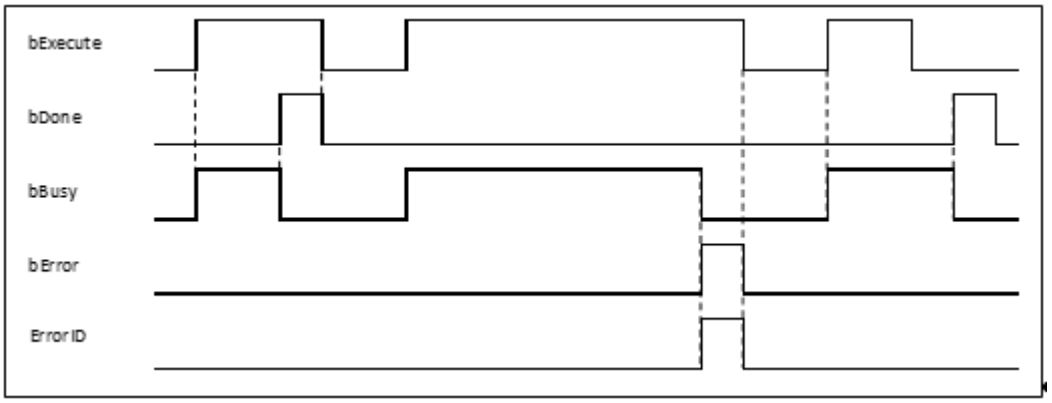
Name	Function	Data Type	Output range (Default Value)
bDone	True when the instruction is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the instruction is completed.	• When *bExecute* turns to False. • *bDone* will change to False after remaining True for one cycle when *bExecute* is False but *bDone* changes to True.
bBusy	• When *bExecute* changes to TRUE.	• When *bDone* turns to True. • When *bError* turns to True.
bError (ErrorID)	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	When *bExecute* turns from True to False. (Error Code is cleared)

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_ GROUP_REF*	DMC_AXIS_GROUP_REF	When *bExecute* turns to True and *bBusy* is False.

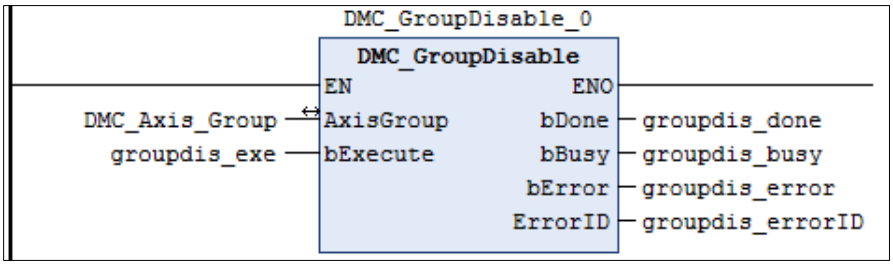
***Note:**

DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**
  - When this instruction is run for an axis group, the axis group state will switch from GroupStandby to GroupDisable but the state of axes in the axis group will remain unchanged.
  - If the axis group is not in GroupStandby state, an error will occur when DMC_GroupDisable is run.
  - For more details on axis states, refer to Axis State Transitions.

- **Troubleshooting**
  - When an error occurs during the execution of the instruction, *bError* will change to True and the axes will stop running. Refer to *ErrorID* (Error Code) to address the problem.
  - For error codes and corresponding troubleshooting, refer to Appendix of this manual.

- **Programming Example**
  - Switch the axis group state from GroupStandby to GroupDisable.

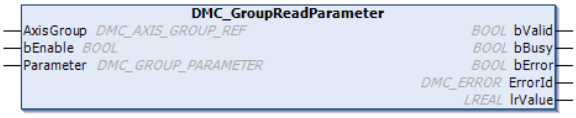


- This instruction enables the group axis DMC_Axis_Group specified by *AxisGroup* to enter the GroupDisable state.
- DMC_GroupDisable is run when groupdis_exe (*bExecute*) changes to true. When groupdis_done (*bDone*) changes to true, it indicates that DMC_Axis_Group axis group has successfully entered GroupDisable state.

2.3.2.3 DMC_GroupReadParameter

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupReadParameter reads axis group parameters.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupReadParameter	

ST Language

```
DMC_GroupReadParameter_instance(
AxisGroup: = ,
bEnable: = ,
Parameter: = ,
bValid=> ,
bBusy=> ,
bError=> ,
ErrorID=> ,
lrValue=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is enabled when *bEnable* changes from False to True.	BOOL	True/False (False)	-
Parameter	Set the parameter to be written.	DMC_GROUP_PARAMETER*	DMC_GROUP_PARAMETER* (PARAM_RAMP_TYPE)	Whew the function block *bEnable is* True, the setting parameter of Parameter will be updated.

*__Note__: DMC_GROUP_PARAMETER: Enumeration (Enum)

Setting Value	Name	Function
16	PARAM_RAMP_TYPE	Velocity ramp type
17	PARAM_MAX_VELOCITY_LIMIT	Max. velocity limit

Setting Value	Name	Function
18	PARAM_MAX_ACCELERATION_LIMIT	Limit on max. acceleration
19	PARAM_MAX_DECELERATION_LIMIT	Limit on max. deceleration
21*1	PARAM_PLANNING_PRIORITY	Velocity ramp planning is prioritized
22	PARAM_STOP_METHOD	Stop method
24	PARAM_VELOCITY_WARNING_PERCENTAGE	Velocity warning range
25	PARAM_ACCELERATION_WARNING_PERCENTAGE	Acceleration warning range
26	PARAM_DECELERATION_WARNING_PERCENTAGE	Deceleration warning range
28	PARAM_RADIUS_CORRECTION_PERCENTAGE	Allowable correction range of radius

***Note:**

1. DL_MotionControl Version 1.2.0.0 and later supports the above features.
2. Refer to DMC_GroupWriteParameter for setting parameter values.

- **Outputs**

Name	Function	Data Type	Output range (Default Value)
bValid	True when the output value is valid.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*1	DMC_ERROR (DMC_NO_ERROR)
lrValue	Read parameter value	LREAL*2	Positive, negative , or 0 (0)

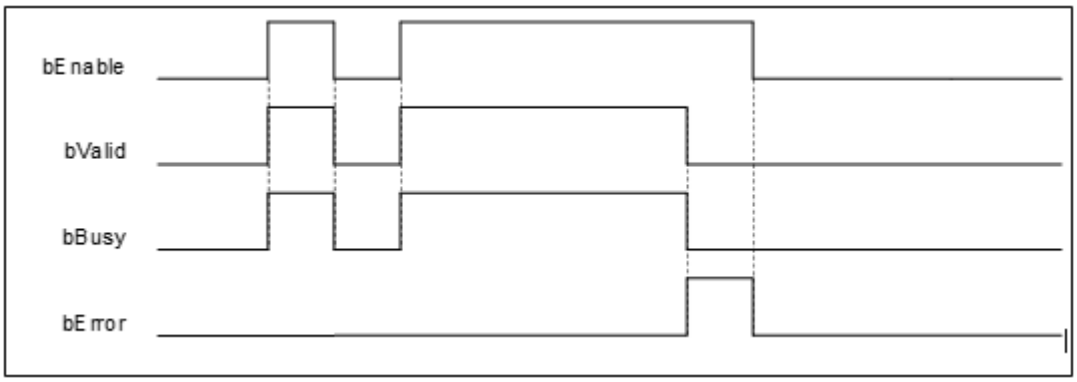
***Note**:

1. DMC_ERROR: Enumeration (Enum)
2. No matter what number type of the original parameter type is (including ENUM), the read parameter will be expressed as LREAL.

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bValid	• When *bEnable* turns to True, and the output pin lrValue is valid.	• When *bEnable* turns to False. • When *bError* turns to True.
bBusy	• When *bEnable* turns to True.	• When *bValid* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction (error code is recorded in ErrorID).	• When *bEnable* turns to False (clear the error code recorded in ErrorID).
ErrorID		
lrValue	• Continuously update the value when *bEnable* is True.	• Continuously update the value when *bEnable* is True.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_ GROUP_REF*	DMC_AXIS_ GROUP_REF	When *bEnable* turns to True, and *bBusy* is False.

*__Note__: DMC_AXIS_GROUP_REF(FB): The interface is built in every function block and works as the starting program of the function block.

- **Function**

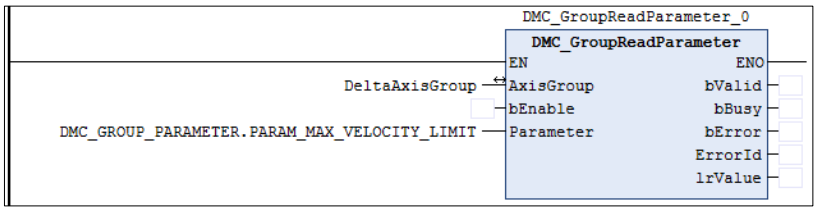
  - If the parameter to be read is of the ENUM type, the read parameter will be its corresponding number.

- **Troubleshooting**

  - When an error occurs during the execution of the instruction, *bError* changes to True and axes stops running. To confirm current error state, see the error code in *ErrorID*.
  - For error codes and corresponding trouble shootings, refer to **Appendix** for error code descriptions.

- **Programming Example**

  - This example shows how to directly use DMC_GroupReadParameter to read axis group parameters.



2.3.2.4 DMC_GroupWriteParameter

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupWriteParameter writes axis group parameters.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupWriteParameter	

ST Language

```
DMC_GroupWriteParameter_instance(
AxisGroup: = ,
bExecute: = ,
Parameter: = ,
lrValue: = ,
bDone=> ,
bBusy=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
Parameter	Set the parameter to be written	DMC_GROUP_PARAMETER*2	DMC_GROUP_PARAMETER (PARAM_RAMP_TYPE)	Whew the function block *bExecute is* True, the setting parameter of Parameter will be updated.
lrValue	The value to be written	LREAL*1	Positive, negative , or 0 (0)	When the function block *bExecute* turns to True, the setting parameter of Parameter will be updated.

***Note**:

1. No matter what number type of the original parameter type is (including ENUM), the read parameter will be expressed as LREAL.
2. DMC_GROUP_PARAMETER: Enumeration (Enum)

Setting Value	Name	Function
16	PARAM_RAMP_TYPE	Velocity ramp type
17	PARAM_MAX_VELOCITY_LIMIT	Max. velocity limit
18	PARAM_MAX_ACCELERATION_LIMIT	Limit on max. acceleration
19	PARAM_MAX_DECELERATION_LIMIT	Limit on max. deceleration
21*	PARAM_PLANNING_PRIORITY	Velocity ramp planning is prioritized
22	PARAM_STOP_METHOD	Stop method
24	PARAM_VELOCITY_WARNING_PERCENTAGE	Velocity warning range
25	PARAM_ACCELERATION_WARNING_PERCENTAGE	Acceleration warning range
26	PARAM_DECELERATION_WARNING_PERCENTAGE	Deceleration warning range
28	PARAM_RADIUS_CORRECTION_PERCENTAGE	Allowable correction range of radius

***Note:** DL_MotionControl Version 1.2.0.0 and later supports the above features.

■ **Parameter Values**

Data Type	Value (Default Value)	Description
PARAM_RAMP_TYPE	0: Trapezoid 1: S_Curve (0)	0: Trapezoidal curve 1: S curve
PARAM_PLANNING_PRIORITY	0: Velocity 1: Acceleration (0)	0: Velocity first 1: Acceleration first

- **Outputs**

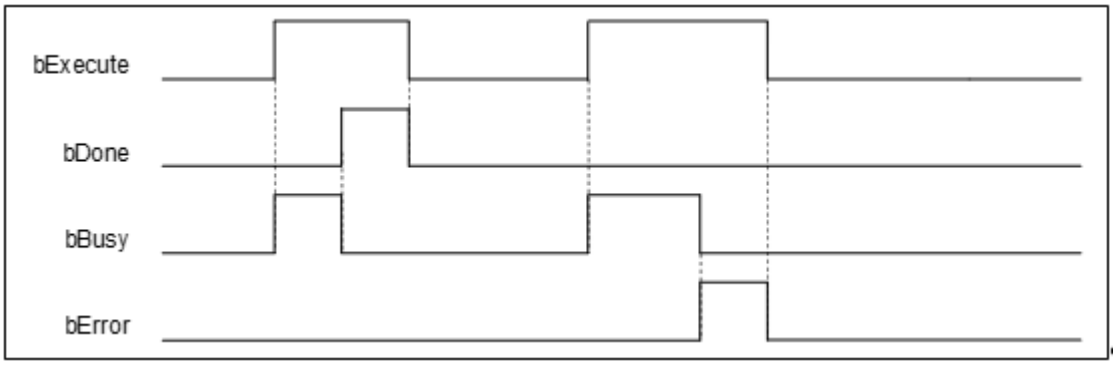
Name	Function	Data Type	Output range (Default Value)
bDone	True when the parameter is written.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an instruction error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR(DMC_NO_ERR)

*__Note__: DMC_ERROR: Enumeration (Enum)

■ **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the parameter is written	• When *bEexcute* turns to False. • When *bError* turns to True.
bBusy	• When *bExexcute* turns to True	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction (error code is recorded in *ErrorID*).	• When *bEexcute* turns to False (clear the error code recorded in *ErrorID*).
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify	DMC_AXIS_	DMC_AXIS_	When *bExecute* turns to True, and *bBusy* is

	the axis group.	GROUP_REF*	GROUP_REF	False.

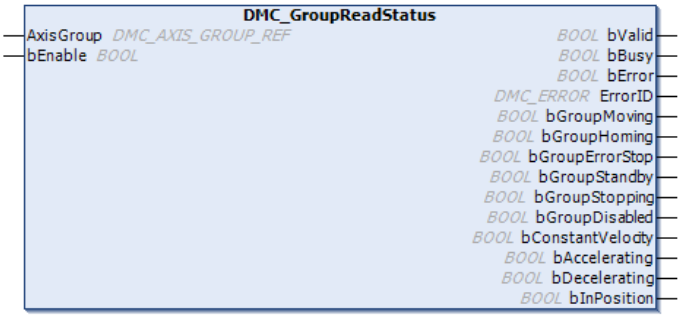
*Note: DMC_AXIS_GROUP_REF(FB): The interface is built in every function block and works as the starting program of the function block.

- **Function**
  - If the parameter to be write is of the ENUM type, set lrValue as the corresponding number.
- **Troubleshooting**
  - When an error occurs during the execution of the instruction, *bError* changes to True and axes stops running. To confirm current error state, see the error code in ErrorID.
  - For error codes and corresponding trouble shootings, refer to **Appendix** for error code descriptions.
- **Programming Example**
  - Refer to the programming example of DMC_GroupReadParameter.

2.3.2.5 DMC_GroupReadStatus

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupReadStatus reads the state of an axis group.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupReadStatus	

ST Language

```
DMC_GroupReadStatus_instance(
AxisGroup: = ,
bEnable: = ,
bValid=> ,
bBusy=> ,
bError=> ,
ErrorID=> ,
bGroupMoving=> ,
bGroupHoming=> ,
bGroupErrorStop=> ,
bGroupStandby=> ,
bGroupStopping=> ,
bGroupDisabled=> ,
bConstantVelocity=> ,
bAccelerating=> ,
bDecelerating=> ,
bInPosition=> ) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is enabled when *bEnable* changes from False to True.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output range (Default Value)
bValid	True when the output values are valid.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
bGroupMoving	True when the axis group state is *bGroupMoving*.	BOOL	True/False (False)
bGroupHoming	True when the axis group state is *bGroupHoming*.	BOOL	True/False (False)
bGroupErrorStop	True when the axis group state is *bGroupErrorStop*.	BOOL	True/False (False)
bGroupStandby	True when the axis group state is *bGroupStandby*.	BOOL	True/False (False)
bGroupStopping	True when the axis group state is *bGroupStopping*.	BOOL	True/False (False)
bGroupDisabled	True when the axis group state is *bGroupDisabled*.	BOOL	True/False (False)
bConstantVelocity	True when the axis group runs at a constant velocity.	BOOL	True/False (False)
bAccelerating	True when the axis group accelerates.	BOOL	True/False (False)
bDecelerating	True when the axis group decelerates.	BOOL	True/False (False)
bInPosition	True when the axis group reaches the target position.	BOOL	True/False (False)

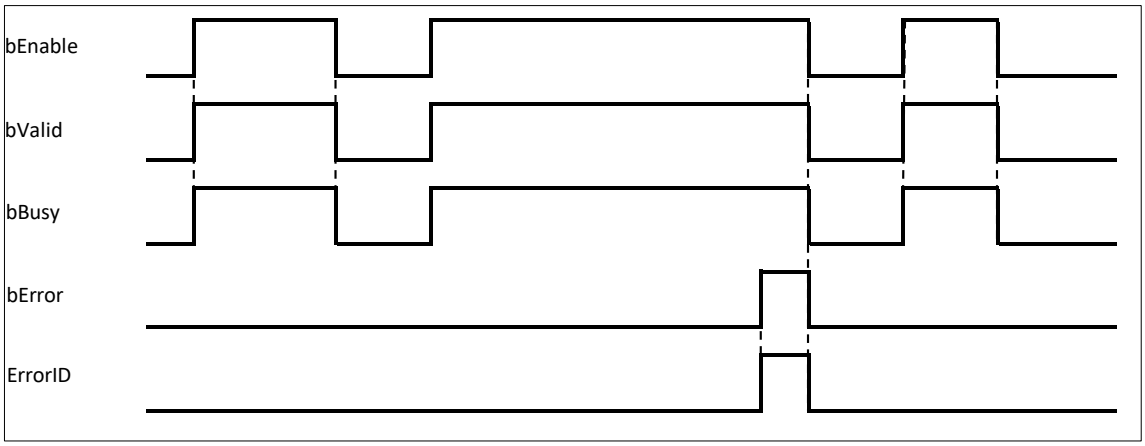
***Note:** DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bValid	• When *bEnable* is True and other axis group state outputs are valid.	• When *bEnable* turns to False. • When *bError* turns to True.
bBusy	• When *bEnable* changes to TRUE.	• When *bEnable* turns to False. • When *bError* turns to True.
bError (ErrorID)	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	When *bEnable* turns from True to False. (Error Code is cleared).
bGroupMoving	• When *bEnable* is True and the output keeps updating its value.	• When *bEnable* is True and the output keeps updating its value. • When *bEnable* turns to False. • When *bError* turns to True.
bGroupHoming	• When *bEnable* is True and the output keeps updating its value.	• When *bEnable* is True and the output keeps updating its value.

Name	Timing for shifting to True	Timing for shifting to False
		• When *bEnable* turns to False. • When *bError* turns to True.
bGroupErrorStop	• When *bEnable* is True and the output keeps updating its value.	• When *bEnable* is True and the output keeps updating its value. • When *bEnable* turns to False. • When *bError* turns to True.
bGroupStandby	• When *bEnable* is True and the output keeps updating its value.	• When *bEnable* is True and the output keeps updating its value. • When *bEnable* turns to False. • When *bError* turns to True.
bGroupStopping	• When *bEnable* is True and the output keeps updating its value.	• When *bEnable* is True and the output keeps updating its value. • When *bEnable* turns to False. • When *bError* turns to True.
bGroupDisabled	• When *bEnable* is True and the output keeps updating its value.	• When *bEnable* is True and the output keeps updating its value. • When *bEnable* turns to False. • When *bError* turns to True.
bConstantVelocity	• When *bEnable* is True and the output keeps updating its value.	• When *bEnable* is True and the output keeps updating its value. • When *bEnable* turns to False. • When *bError* turns to True.
bAccelerating	• When *bEnable* is True and the output keeps updating its value.	• When *bEnable* is True and the output keeps updating its value. • When *bEnable* turns to False. • When *bError* turns to True.
bDecelerating	• When *bEnable* is True and the output keeps updating its value.	• When *bEnable* is True and the output keeps updating its value. • When *bEnable* turns to False. • When *bError* turns to True.
bInPosition	• When *bEnable* is True and the output keeps updating its value.	• When *bEnable* is True and the output keeps updating its value. • When *bEnable* turns to False. • When *bError* turns to True.

■ **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_ GROUP_REF*	DMC_AXIS_ GROUP_REF	When *bEnable* turns to True and *bBusy* is False.

***Note:**

DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

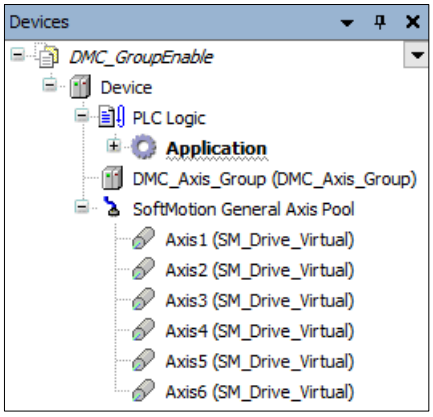
  - DMC_GroupReadStatus can be used to read the state of an axis group.

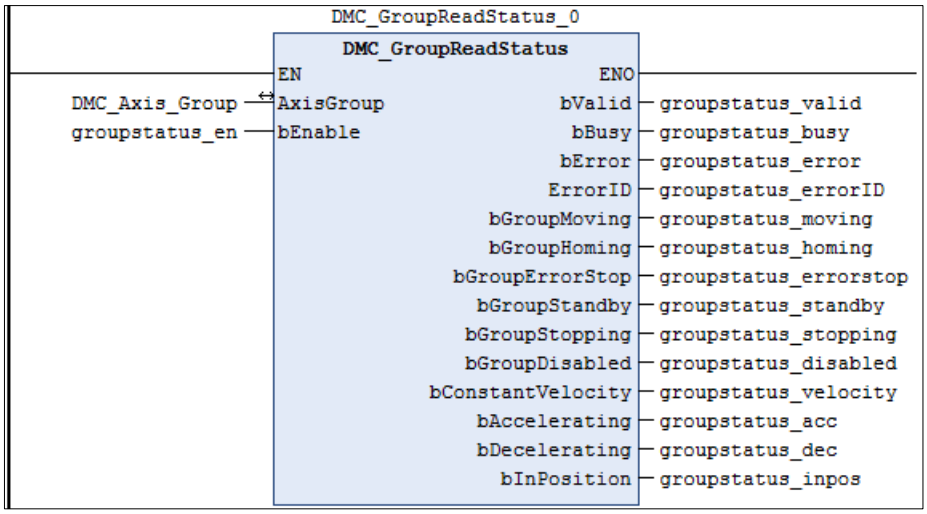
- **Troubleshooting**

  - When an error occurs during the execution of the instruction, *bError* will change to True and the axes will stop running. Refer to *ErrorID* (Error Code) to address the problem.
  - For error codes and corresponding troubleshooting, refer to **Appendix** of this manual.

- **Programming Example**

  - The example shows how DMC_GroupReadStatus is used to read the current state of the specified axis group.



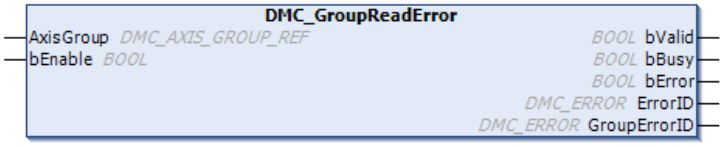


- Add DMC_Axis_Group in **Device**.
- When groupstatus_valid (*bValid*) changes to True after groupstatus_en (*bEnable*) changes to True, DMC_GroupReadStatus reads the state of the axis group DMC_Axis_Group via its outputs.

2.3.2.6 DMC_GroupReadError

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupReadError reads axis group errors.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupReadError	

ST Language

```
DMC_GroupReadError_instance(
AxisGroup: = ,
bEnable: = ,
bValid=> ,
bBusy=> ,
bError=> ,
ErrorID=> ,
GroupErrorID=> ) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is enabled when *bEnable* changes from False to True.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output range (Default Value)
bValid	True when the output value is valid.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to **Appendix** for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
GroupErrorID	When the axis group is in ErrorStop	DMC_ERROR	DMC_ERROR

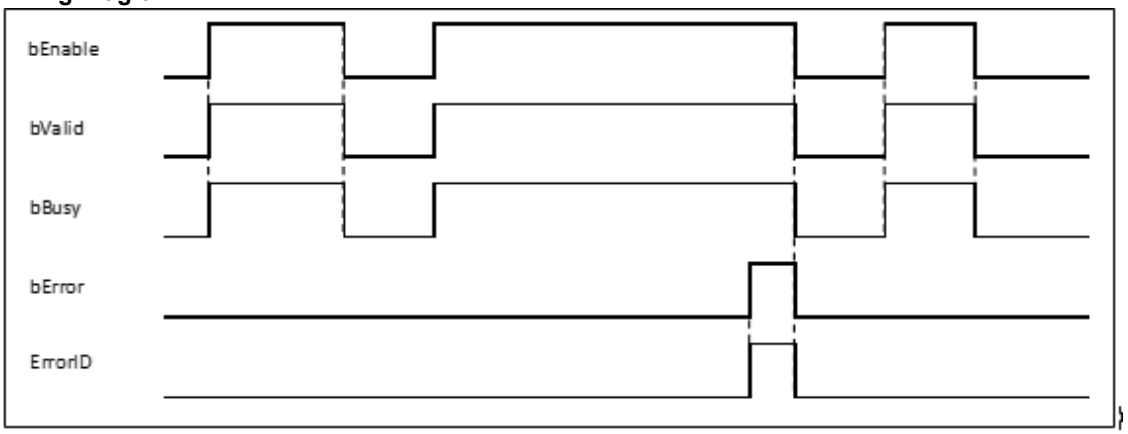
Name	Function	Data Type	Output range (Default Value)
	state, the output shows an error code for the current axis group. Refer to **Appendix** for error code descriptions.		(DMC_NO_ERROR)

***Note:** DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bValid	• When *bEnable* is True and the output value is valid.	• When *bEnable* turns to False. • When *bError* turns to True.
bBusy	• When *bEnable* is True	• When *bEnable* turns to False. • When *bError* turns to True.
bError (ErrorID)	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded in ErrorID and axis group error code is recorded in GroupErrorID).	• When *bEnable* turns from True to False. (Both the error code in ErrorID and axis group error code in GroupErrorID are cleared)
GroupErrorID	• When *bEnable* is True and the output keeps updating.	• When *bEnable* is True and the output keeps updating.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_ GROUP_REF*	DMC_AXIS_ GROUP_REF	When *bEnable* turns to True and *bBusy* is False.

***Note:**

DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

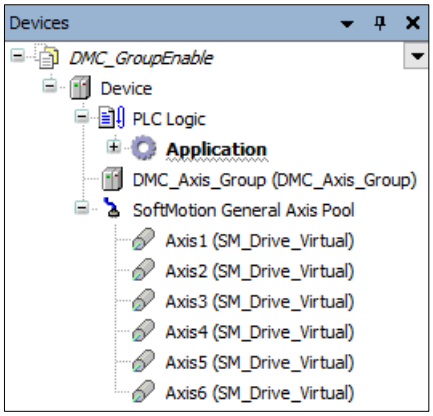
  - DMC_GroupReadError can be used to read axis group errors.
  - The value of *GroupErrorID* is 0 if no axis group error occurs.

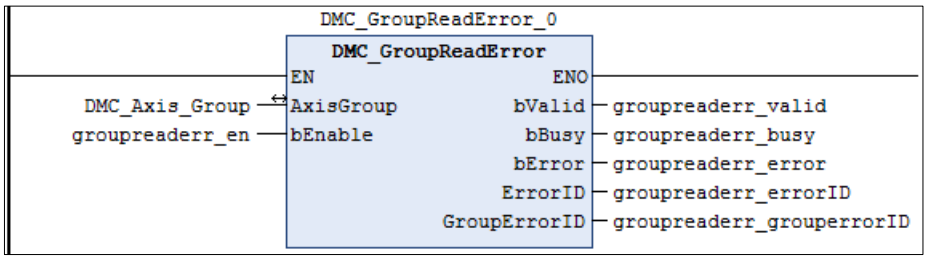
- **Troubleshooting**

  - When an error occurs during the execution of the instruction, *bError* will change to True and the axes will stop running. Refer to *ErrorID* (Error Code) to address the problem.
  - For error codes and corresponding troubleshooting, refer to **Appendix** in this manual.

- **Programming Example**

  - The example shows how DMC_GroupReadError is used to read an axis group error after the axis group is created.





    - Add DMC_Axis_Group in **Device**.
    - When groupreaderr_valid (*bValid*) changes to True after groupreaderr_en (*bEnable*) changes to True, DMC_GroupReadError reads the state of the axis group DMC_Axis_Group via its output.

2.3.2.7 DMC_GroupReset

- **Supported Devices**: AX-308E, AX-364E

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupReset	

ST Language

```
DMC_GroupReset_instance(
AxisGroup: = ,
bExecute: = ,
bDone=> ,
bBusy=> ,
bError=> ,
ErrorID=> ) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-

- **Outputs**

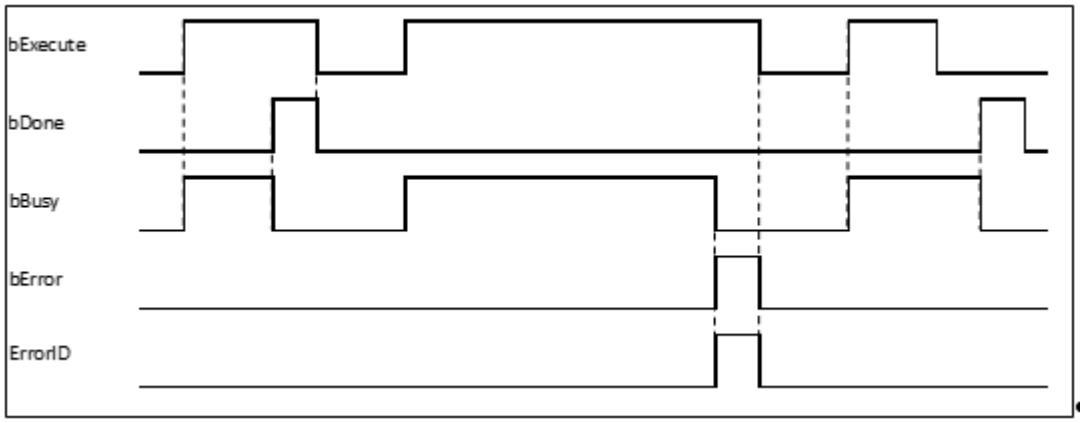
Name	Function	Data Type	Output range (Default Value)
bDone	True when the instruction is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When axis group errors clearing is completed.	• When *bExecute* turns to False. • *bDone* will change to False after remaining True for one cycle when *bExecute* is False but *bDone* changes to True.
bBusy	• When *bExecute* changes to TRUE.	• When *bDone* turns to True. • When *bError* turns to True.
bError (ErrorID)	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	• When *bExecute* turns from True to False. (Error Code is cleared)

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_ GROUP_REF*	DMC_AXIS_ GROUP_REF	When *bExecute* turns to True and *bBusy* is False.

***Note:**

DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

  - When an axis group is in GroupErrorstop state, DMC_GroupReset can be used to clear axis group errors and switch the axis group state to GroupStandby.
  - When the axis group enters the GroupStandby state, it indicates that the axis group motion can be performed.

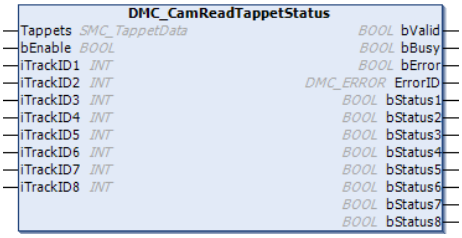
- **Troubleshooting**

  - When an error occurs during the execution of the instruction, *bError* will change to True and the axes will stop running. Refer to *ErrorID* (Error Code) to address the problem.
  - For error codes and corresponding troubleshooting, refer to **Appendix** of this manual.

2.3.2.8 DMC_CamReadTappetStatus

- **Supported Devices**: AX-308E, AX-364E

DMC_CamReadTappetStatus reads the status of multiple tappets.

FB/FC	Instruction	Graphic Expression
FB	DMC_CamReadTappetStatus	

ST Language

```
DMC_CamReadTappetStatus_instance(
Tappets : =,
bEnable : =,
iTrackID1 : =,
iTrackID2 : =,
iTrackID3 : =,
iTrackID4 : =,
iTrackID5 : =,
iTrackID6 : =,
iTrackID7 : =,
iTrackID8 : =,
bValid =>,
bBusy =>,
bError =>,
ErrorID =>,
bStatus1 =>,
bStatus2 =>,
bStatus3 =>,
bStatus4 =>,
bStatus5 =>,
bStatus6 =>,
bStatus7 =>,
bStatus8 =>) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is enabled when *bEnable* changes from False to True.	BOOL	True/False (True)	-

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
iTrackID1	Specify the tappet number.	INT	0~512 (0) *	When *bEnable* is True.
iTrackID2	Specify the tappet number.	INT	0~512 (0)	When *bEnable* is True.
iTrackID3	Specify the tappet number.	INT	0~512 (0)	When *bEnable* is True.
iTrackID4	Specify the tappet number.	INT	0~512 (0)	When *bEnable* is True.
iTrackID5	Specify the tappet number.	INT	0~512 (0)	When *bEnable* is True.
iTrackID6	Specify the tappet number.	INT	0~512 (0)	When *bEnable* is True.
iTrackID7	Specify the tappet number.	INT	0~512 (0)	When *bEnable* is True.
iTrackID8	Specify the tappet number.	INT	0~512 (0)	When *bEnable* is True.

***Note:**

If the Track ID is set to 0, the corresponding output will not be used to read the tappet status.

- **Outputs**

Name	Function	Data Type	Output range (Default Value)
bValid	True when the outputs are valid.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
bStatus1	The status of the tappet specified by *iTrackID1.*	BOOL	True/False (False)
bStatus2	The status of the tappet specified by *iTrackID2.*	BOOL	True/False (False)
bStatus3	The status of the tappet specified by *iTrackID3.*	BOOL	True/False (False)
bStatus4	The status of the tappet specified by *iTrackID4.*	BOOL	True/False (False)
bStatus5	The status of the tappet specified by *iTrackID5.*	BOOL	True/False (False)

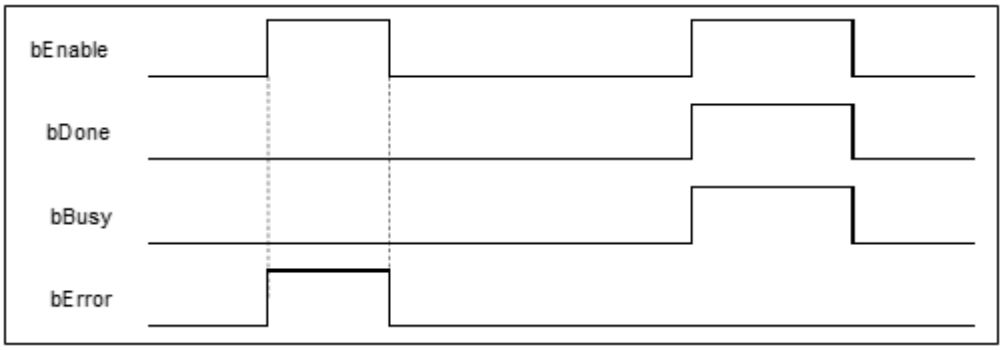
Name	Function	Data Type	Output range (Default Value)
bStatus6	The status of the tappet specified by *iTrackID6.*	BOOL	True/False (False)
bStatus7	The status of the tappet specified by *iTrackID7.*	BOOL	True/False (False)
bStatus8	The status of the tappet specified by *iTrackID8.*	BOOL	True/False (False)

***Note:** DMC_ERROR: Enumeration (ENUM)

■ **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bValid	• When *bEnable* turns to True.	• When *bError* turns to True. • When *bEnable* turns to False.
bBusy	• When *bEnable* turns to True.	• When bError turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bEnable* turns to False.
ErrorID		
bStatus1	• When the status of the specified tappet is True.	• When the status of the specified tappet is False.
bStatus2	• When the status of the specified tappet is True.	• When the status of the specified tappet is False.
bStatus3	• When the status of the specified tappet is True.	• When the status of the specified tappet is False.
bStatus4	• When the status of the specified tappet is True.	• When the status of the specified tappet is False.
bStatus5	• When the status of the specified tappet is True.	• When the status of the specified tappet is False.
bStatus6	• When the status of the specified tappet is True.	• When the status of the specified tappet is False.
bStatus7	• When the status of the specified tappet is True.	• When the status of the specified tappet is False.
bStatus8	• When the status of the specified tappet is True.	• When the status of the specified tappet is False.

■ **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Tappets	Tappet signal source	SMC_TappetData	SMC_TappetData*	When *bEnable* turns to True.

***Note:**

SMC_TappetData (STRUCT): the internal interface between MC_CamIn and SMC_GetTappetValue for tappet data transmission.

Name	Function	Data Type	Setting range (Default Value)
ctt	Specify the direction in which a tappet point is passed so that there will be an action then.	SMC_CAMTAPPETTYPE	0: TAPPET_pos (pass in the positive direction) 1: TAPPET_all (pass in both positive and negative directions) 2: TAPPET_neg (pass in the negative direction) (TAPPET_pos)
cta	Specify the action when the tappet point is passed.	SMC_CAMTAPPETACTION	0: TAPPETACTION_on (switch to ON) 1: TAPPETACTION_off (switch to OFF) 2: TAPPETACTION_inv (Invert) 3: TAPPETACTION_time (be ON for a period of time and then switch to OFF.) (TAPPETACTION_on)
dwDelay	Specify the delay time before the tappet changes to ON under TAPPETACTION_time mode.	DWORD	Positive or 0 (0)
dwDuration	For how long the tappet is ON under TAPPETACTION_time mode.	DWORD	Positive or 0 (0)
iGroupID	Specify the track ID of the tappet.	INT	Positive, negative or 0 (0)
x	Tappet position	LREAL	Positive, negative or 0 (0)
dwActive	The internal variable	DWORD	Positive or 0 (0)

- **Function**

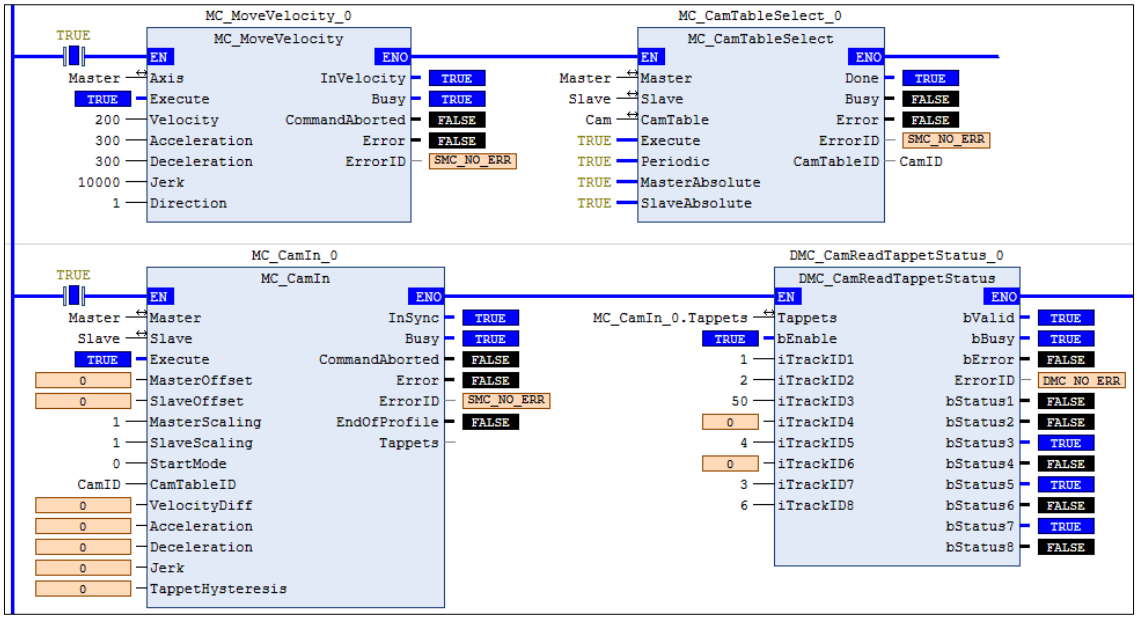
- The instruction allows users to watch the states of eight tappets. The tappet number range for iTrackID1~8 is 0~512. If the setting is outside the range, the instruction will report an error, which indicates that the output status is disabled.
- After *bEnable* changes to False, the instruction will not update the states of tappets anymore and then outputs will maintain current tappet states.

- **Troubleshooting**

- When an error occurs during the execution of the instruction, *bError* will change to True and the axes will stop running. Refer to *ErrorID* (Error Code) to address the problem.

- **Programming Example**

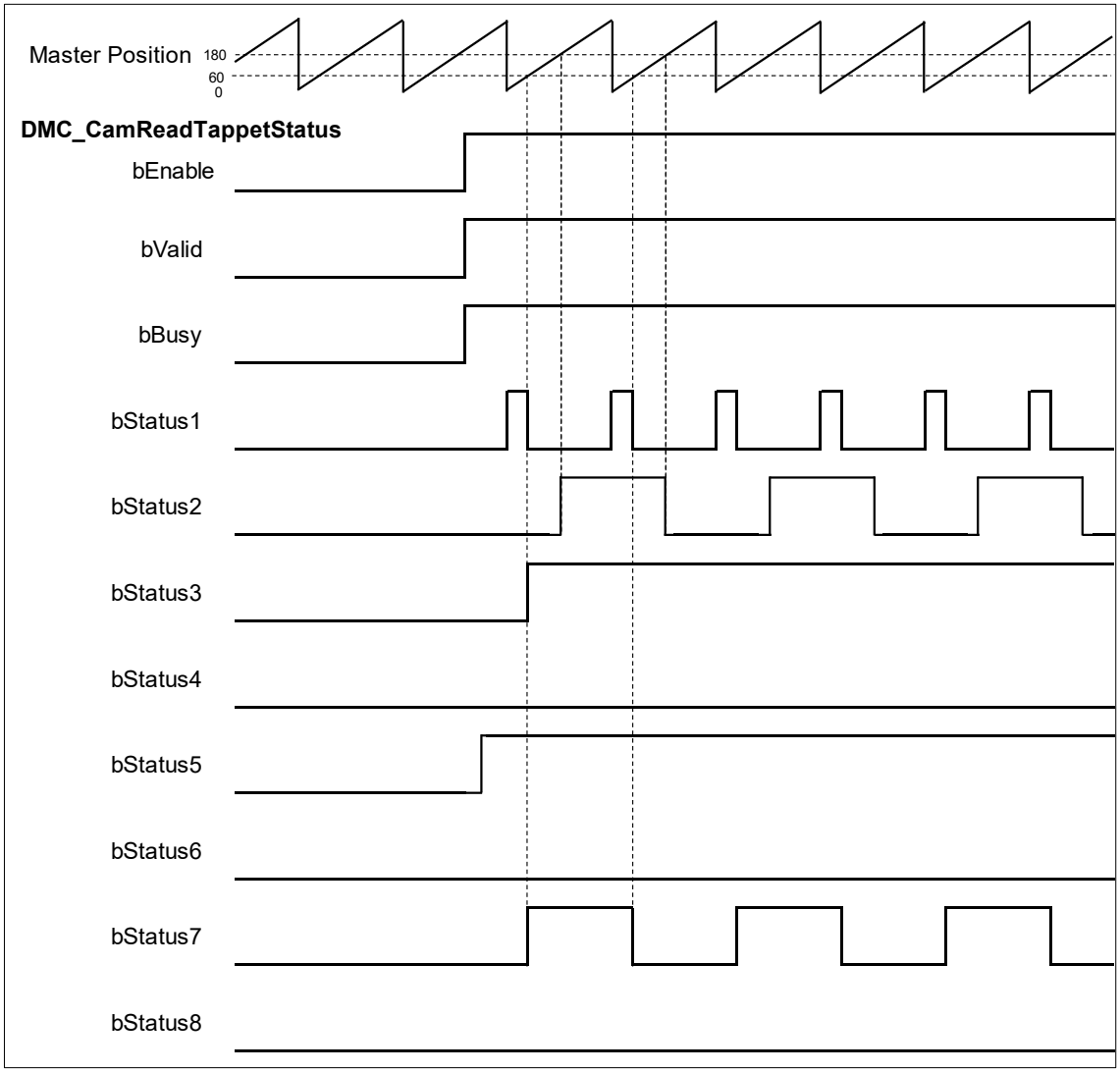
- The example explains the action principle for CamReadTappetStatus.
- Input 1 for *iTrackID1*, 2 for *iTrackID2*, 50 for *iTrackID3*, 4 for *iTrackID4*, 3 for *iTrackID7*. No tappet numbers for *iTrackID4*, *iTrackID6* and *iTrackID8* are given and thus there will be no actions on these tappets.



- Tappets Setting

	Track ID	X	positive pass	negative pass
	1			
		0	invert	switch OFF
		60	switch OFF	switch OFF
	2			
		180	invert	none
	50			
		60	switch ON	switch OFF
	3			
		300	invert	none
	4			
		270	switch ON	switch OFF

■ **Timing Diagram**

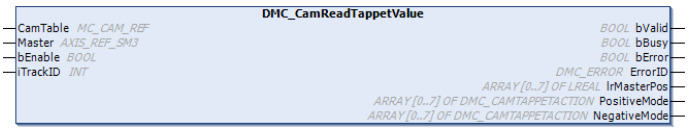


- When *bEnable* changes to True, DMC_CamReadTappetStatus starts to update the statuses of tappets.
- Take the second output point (*bStatus2*) for example. The corresponding tappet ID is 2 and the action is to invert its status when the position 180 is reached.
- When *bEnable* changes to False, the outputs maintain current statuses of tappets.

2.3.2.9 DMC_CamReadTappetValue

- **Supported Devices**: AX-308E, AX-364E

DMC_CamReadTappetValue reads the data of one single tappet.

FB/FC	Instruction	Graphic Expression
FB	DMC_CamReadTappetValue	

ST Language

```
DMC_CamReadTappetValue_instance(
bEnable : =,
CamTable : =,
Master: =,
iTrackID: =,
bValid =>,
bBusy =>,
bError =>,
ErrorID =>,
lrMasterPos =>,
PositiveMode =>,
NegativeMode =>) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is enabled when *bEnable* changes from False to True.	BOOL	True/False (True)	-
iTrackID	Specify the ID of the Track to be read.	INT	1~512 (0)	When *bEnable* is True.

- **Outputs**

Name	Function	Data Type	Output range (Default Value)
bValid	True when the outputs are valid.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)

Name	Function	Data Type	Output range (Default Value)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*1	DMC_ERROR (DMC_NO_ERROR)
lrMasterPos	The tappet position which is corresponded to master axis position (Unit: user unit)	LREAL[0..7]*2	Positive, negative or 0 (0)*3
PositiveMode	Specify the mode for the tappet point when it is passed in the positive direction.	DMC_CAMTAPPETACTION[0..7]*2	0: TAPPETACTION_none 1: TAPPETACTION_on 2: TAPPETACTION_off 3: TAPPETACTION_inv 4: TAPPETACTION_time (TAPPETACTION_none)
NegativeMode	Specify the mode for the tappet point when it is passed in the negative direction.	DMC_CAMTAPPETACTION[0..7]*2	0: TAPPETACTION_none 1: TAPPETACTION_on 2: TAPPETACTION_off 3: TAPPETACTION_inv 4: TAPPETACTION_time (TAPPETACTION_none)

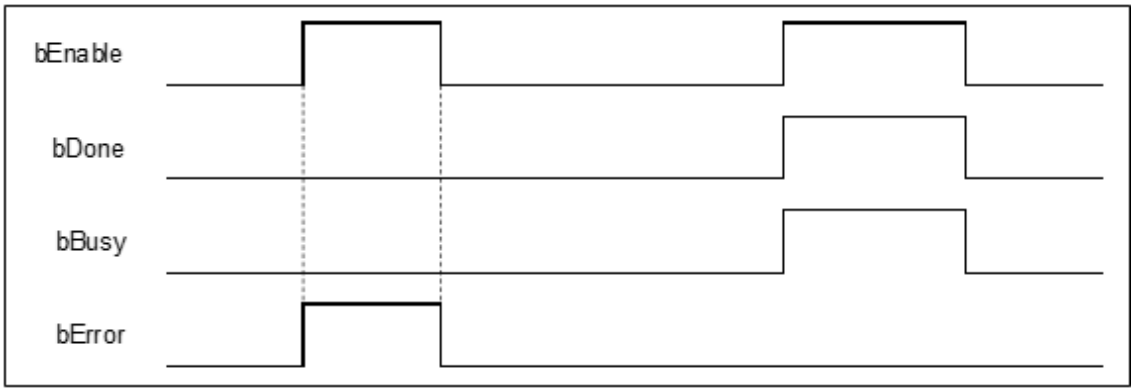
***Note:**

1. DMC_ERROR: Enumeration (ENUM)
2. One Track can have multiple tappet points set inside it. 8 tappet points at most can be read from the same Track via this instruction by default.
3. There is no tappet data to be output when *lrMasterPos* is set to 0 and *PositiveMode* and *NegativeMode* are both set to TAPPETACTION_none.

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bValid	• When *bEnable* turns to True.	• When *bError* turns to True. • When *bEnable* turns to False.
bBusy	• When *bEnable* turns to True.	• When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bEnable* turns to False.
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF*1	MC_CAM_REF	When *bEnable* turns to True.
Master	The reference master axis	AXIS_REF_SM3*2	AXIS_REF_SM3	

***Note:**

1. MC_CAM_REF (FB): The basic CAM
2. AXIS_REF_SM3 (FB): Generally, all motion function blocks have the InOut AXIS_REF_SM3.

- **Function**

  - A tappet table can be set in the Cam table. Multiple tappet points can be set for one Track. 8 tappet points at most can be read from the same tappet track via the function block.
  - The tappet data contains the master axis position corresponding to the tappet point, the positive passing mode and the negative passing mode. The modes include TAPPETACTION_none, TAPPETACTION_on, TAPPETACTION_off, TAPPETACTION_inv and TAPPETACTION_time
  - See the meanings of the modes in the following table.

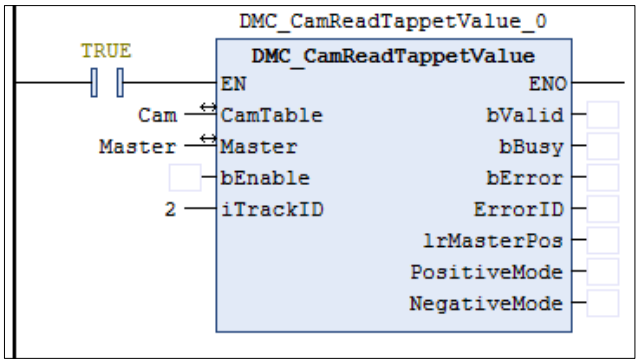
Mode	Function	Action
TAPPETACTION_none	No action	The tappet does not take an action when the master axis passes the point.
TAPPETACTION_on	ON	The tappet is enabled when the master axis passes the point.
TAPPETACTION_off	OFF	The tappet is disabled when the master axis passes the point.
TAPPETACTION_inv	Invert	The tappet status is inverted when the master axis passes the point.
TAPPETACTION_time	ON	When the master axis passes the point, the tappet is ON for a set period of time and then turns OFF.

- **Troubleshooting**

  - When an error occurs during the execution of the instruction, *bError* will change to True. Refer to *ErrorID* (Error Code) to address the problem.

- **Programming Example**

  - The example explains the action principle for DMC_CamReadTappetValue and how to read the tappet data of Track ID 2.
  - The example explains the action principle for DMC_CamReadTappetValue by reading the tappet data of Track ID 2.



- Setting tappet points

	Track ID	X	positive pass	negative pass
	1			
		60	switch OFF	switch OFF
	2			
		50	switch ON	switch OFF
		180	none	invert
	50			
		60	switch ON	switch OFF

lrMasterPos	ARRAY [0..7] OF LREAL	
lrMasterPos[0]	LREAL	50
lrMasterPos[1]	LREAL	180
lrMasterPos[2]	LREAL	0
lrMasterPos[3]	LREAL	0
lrMasterPos[4]	LREAL	0
lrMasterPos[5]	LREAL	0
lrMasterPos[6]	LREAL	0
lrMasterPos[7]	LREAL	0
PositiveMode	ARRAY [0..7] OF DMC_CAMTAPPETACTION	
PositiveMode[0]	DMC_CAMTAPPETACTION	TAPPETACTION_on
PositiveMode[1]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PositiveMode[2]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PositiveMode[3]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PositiveMode[4]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PositiveMode[5]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PositiveMode[6]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PositiveMode[7]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NegativeMode	ARRAY [0..7] OF DMC_CAMTAPPETACTION	
NegativeMode[0]	DMC_CAMTAPPETACTION	TAPPETACTION_off
NegativeMode[1]	DMC_CAMTAPPETACTION	TAPPETACTION_inv
NegativeMode[2]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NegativeMode[3]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NegativeMode[4]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NegativeMode[5]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NegativeMode[6]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NegativeMode[7]	DMC_CAMTAPPETACTION	TAPPETACTION_none

  - The tappet of Track ID 2 has two switch points: 50 and 180. Array 1 stores the data of switch position 50 and array 2 stores the data of switch position 180. The position -1 indicates no tappet switch data.

Track ID	MasterPosition	Positive Pass	Negative Pass
2	50	TAPPETACTION_on	TAPPETACTION_off
2	180	TAPPETACTION_none	TAPPETACTION_inv

2.3.2.10 DMC_CamWriteTappetValue

- **Supported Devices**: AX308E, AX-364E

DMC_CamWriteTappetValue modifies the tappet data for the specified existing track.

FB/FC	Instruction	Graphic Expression
FB	DMC_ CamWriteTappetValue	

ST Language

```
DMC_CamWriteTappetValue_instance(
CamTable : =,
bExecute : =,
iTrackID: =,
lrMasterPosition : =,
PositiveMode : =,
NegativeMode : =,
bDone =>,
bBusy =>,
bError =>,
ErrorID =>) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (True)	-
iTrackID	Specify the Track ID where the tappet data is to be modified.	INT	1~512 (0)	When *bExecute* turns from False to True.
lrMasterPos	The master axis position of the tappet point (Unit: user unit)	LREAL[0..7]*1	Positive, negative or 0 (-1)	When *bExecute* turns from False to True.
PositiveMode	Specify the mode for the tappet point when it has passed in the	DMC_CAMTAPPETACTION[0..7]*	0: TAPPETACTION_none 1: TAPPETACTION_on 2: TAPPETACTION_off 3: TAPPETACTION_inv	When *bExecute* turns from False to True.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
	positive direction.		4: TAPPETACTION_time (TAPPETACTION_none)	
NegativeMode	Specify the mode for the tappet point when it has passed in the negative direction.	DMC_CAMTAPPETACTION[0..7]*	0: TAPPETACTION_none 1: TAPPETACTION_on 2: TAPPETACTION_off 3: TAPPETACTION_inv 4: TAPPETACTION_time (TAPPETACTION_none)	When *bExecute* turns from False to True.

***Note:**

One tappet track can be set with multiple tappet points. Maximum 8 tappet points can be written for one tappet track via the function block.

- **Outputs**

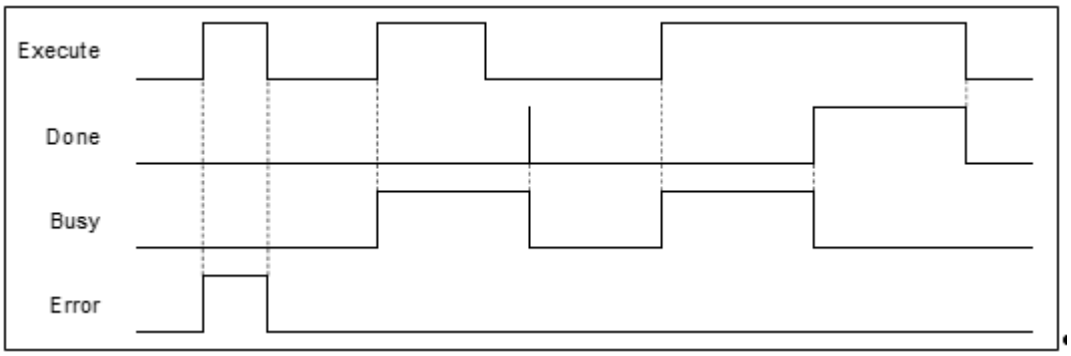
Name	Function	Data Type	Output range (Default Value)
bDone	When the output is valid.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	When the instruction is completed.	• When *bError* turns to True. • When *bExecute* turns to False.
bBusy	When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True.
bError	When an error occurs in the execution conditions or input values for the instruction.	When *bExecute* turns from True to False.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF*	MC_CAM_REF	When *bExecute* is True.

***Note:** MC_CAM_REF (FB): The basic CAM

- **Function**

  - The tappet table can be set in the cam table. Tappets can be used to track the current position of the master axis and output a signal to trigger an event under particular conditions.
  - One tappet table has multiple tappet tracks.
  - This function block can delete all original tappet points in the specified tappet track and replace them with maximum 8 tappet points which are described in the inputs of the function block.
  - The Tappet Track
    - One tappet track contains one track ID, one tappet switch (Boolean signal) and multiple tappet points.
  - The Tappet
    - One tappet point includes the Track ID of the tappet track where the tappet point is, the master axis position corresponding to the tappet point, the positive passing mode and the negative passing mode.

      See the modes and their meanings in the following table.

Mode	Function	Action
TAPPETACTION_none	No action	The tappet switch does not take an action when the master axis passes the point.
TAPPETACTION_on	ON	The tappet switch is enabled when the master axis passes the point.
TAPPETACTION_off	OFF	The tappet switch is disabled when the master axis passes the point.
TAPPETACTION_inv	Invert	The tappet switch status is inverted when the master axis passes the point.
TAPPETACTION_time	ON	When the master axis passes the point, the tappet switch is ON for a set period of time and then turns OFF.

***Note**: When the mode is set to TAPPETACTION_time in this function block, the tappet switch will turn off after being ON for a fixed 100 ms.

- **Troubleshooting**

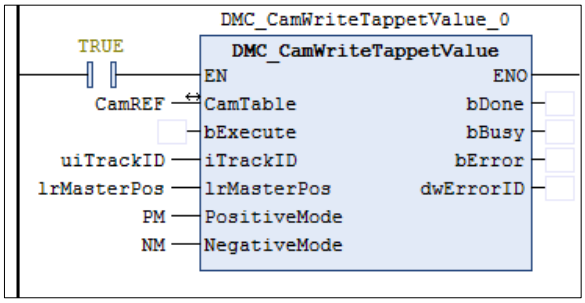
  - When an error occurs during the execution of the instruction or the axis is in “Errorstop” state, *bError* will change to True. Refer to *ErrorID* (Error Code) to address the problem.

- **Programming Example**

  - The example explains the action principle for DMC_CamWriteTappetValue by writing the tappet data of Track ID 1.
  - Initial setting for tappet points

	Track ID	X	positive pass	negative pass
	1			
		100	switch ON	switch OFF
		500	switch OFF	switch OFF
		1000	switch ON	switch OFF
	7			
		7000	invert	none
	2			
		0	switch ON	switch OFF

  - Function block setting



  - Input 1 for uiTrackID. Refer to the figure below for the setup of lrMasterPos, PositiveMode and NegativeMode.

lrMasterPos	ARRAY [0..(GVL.MAX_FB_SWITCH_NUM - 1)] OF LREAL	
lrMasterPos[0]	LREAL	1250
lrMasterPos[1]	LREAL	7050
lrMasterPos[2]	LREAL	3050
lrMasterPos[3]	LREAL	0
lrMasterPos[4]	LREAL	0
lrMasterPos[5]	LREAL	0
lrMasterPos[6]	LREAL	0
lrMasterPos[7]	LREAL	0
PM	ARRAY [0..(GVL.MAX_FB_SWITCH_NUM - 1)] OF DMC_CAMTAPPE...	
PM[0]	DMC_CAMTAPPETACTION	TAPPETACTION_on
PM[1]	DMC_CAMTAPPETACTION	TAPPETACTION_off
PM[2]	DMC_CAMTAPPETACTION	TAPPETACTION_inv
PM[3]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PM[4]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PM[5]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PM[6]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PM[7]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NM	ARRAY [0..(GVL.MAX_FB_SWITCH_NUM - 1)] OF DMC_CAMTAPPE...	
NM[0]	DMC_CAMTAPPETACTION	TAPPETACTION_off
NM[1]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NM[2]	DMC_CAMTAPPETACTION	TAPPETACTION_inv
NM[3]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NM[4]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NM[5]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NM[6]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NM[7]	DMC_CAMTAPPETACTION	TAPPETACTION_none

  - The tappet table before the function block is run

Track ID	Master axis position	Direction	Passing mode
1	100	Negative	TAPPETACTION_off

Track ID	Master axis position	Direction	Passing mode
1	100	Positive	TAPPETACTION_on
1	500	Negative	TAPPETACTION_off
1	500	Positive	TAPPETACTION_off
1	1000	Negative	TAPPETACTION_off
1	1000	Positive	TAPPETACTION_on
7	7000	Positive	TAPPETACTION_inv
2	0	Negative	TAPPETACTION_off
2	0	Positive	TAPPETACTION_on

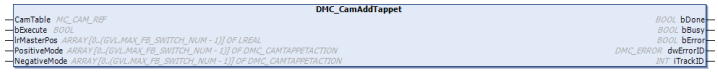
- The tappet table after the function block is run

Track ID	Master axis position	Direction	Passing mode
1	1250	Negative	TAPPETACTION_off
1	1250	Positive	TAPPETACTION_on
1	7050	Positive	TAPPETACTION_off
1	3050	Negative	TAPPETACTION_inv
1	3050	Positive	TAPPETACTION_inv
7	7000	Positive	TAPPETACTION_inv
2	0	Negative	TAPPETACTION_off
2	0	Positive	TAPPETACTION_on

2.3.2.11 DMC_CamAddTappet

- **Supported Devices**: AX308E, AX-364E

DMC_CamAddTappet adds a new tappet track at the end of the tappet table.

FB/FC	Instruction	Graphic Expression
FB	DMC_CamAddTappet	

ST Language

```
DMC_CamAddTappet_instance(
CamTable : =,
bExecute : =,
lrMasterPosition : =,
PositiveMode : =,
NegativeMode : =,
bDone =>,
bBusy =>,
bError =>,
ErrorID =>,
iTrackID => ) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (True)	-
lrMasterPos	The master axis position of the tappet point (Unit: user unit)	LREAL[0..7]*	Positive, negative or 0 (-1)	When *bExecute* turns from False to True.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
PositiveMode	Specify the mode for the tappet point when it is passed in the positive direction.	DMC_CAM TAPPETACTION [0..7]*	0: TAPPETACTION_none 1: TAPPETACTION_on 2: TAPPETACTION_off 3: TAPPETACTION_inv 4: TAPPETACTION_time (TAPPETACTION_none)	When *bExecute* turns from False to True.
NegativeMode	Specify the mode for the tappet point when it is passed in the negative direction.	DMC_CAM TAPPETACTION [0..7]*	0: TAPPETACTION_none 1: TAPPETACTION_on 2: TAPPETACTION_off 3: TAPPETACTION_inv 4: TAPPETACTION_time (TAPPETACTION_none)	When *bExecute* turns from False to True.

***Note:**
One tappet track can be set with multiple tappet points. Maximum 8 tappet points can be written for one tappet track via the function block.

- **Outputs**

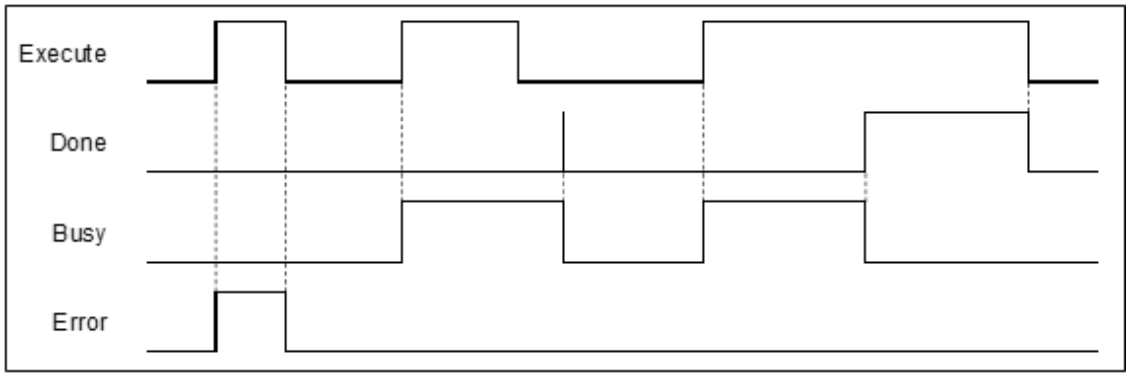
Name	Function	Data Type	Output range (Default Value)
bDone	True when the instruction is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
iTrackID	The Track ID of the new tappet track	INT	1~512

***Note**: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the instruction is completed.	• When *bError* is True. • When *bExecute* turns to False.
bBusy	• When *bExecute* is True.	• When *bDone* is True. • When *bError* is True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns from True to False

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF	MC_CAM_REF	When *bExecute* changes to True.

*__Note__: MC_CAM_REF (FB): The basic CAM.

- **Function**

  - The tappet table can be set in the cam table. Tappets can be used to track the current position of the master axis and output a signal to trigger an event under particular conditions.
  - One tappet table has multiple tappet tracks.
  - This function block adds a tappet track and outputs its track ID to its output *uiTappetNum*. The track ID is the smallest one which has not been used yet.
  - The tappet track
    - One tappet track contains one track ID, one tappet switch (Boolean signal) and multiple tappet points.
  - The tappet
    - One tappet point includes the Track ID of the tappet track where the tappet point is, the master axis position corresponding to the tappet point, the positive passing mode and the negative passing mode.

      See the modes and their meanings in the following table.

Mode	Function	Action
TAPPETACTION_none	No action	The tappet switch does not take an action when the master axis passes the point.
TAPPETACTION_on	ON	The tappet switch is enabled when the master axis passes the point.
TAPPETACTION_off	OFF	The tappet switch is disabled when the master axis passes the point.
TAPPETACTION_inv	Invert	The tappet switch status is inverted when the master axis passes the point.
TAPPETACTION_time	ON	When the master axis passes the point, the tappet switch is ON for a set period of time and then turns OFF.

***Note:**

When the mode is set to TAPPETACTION_time in this function block, the tappet switch will turn off after being ON for a fixed 100 ms.

- **Troubleshooting**

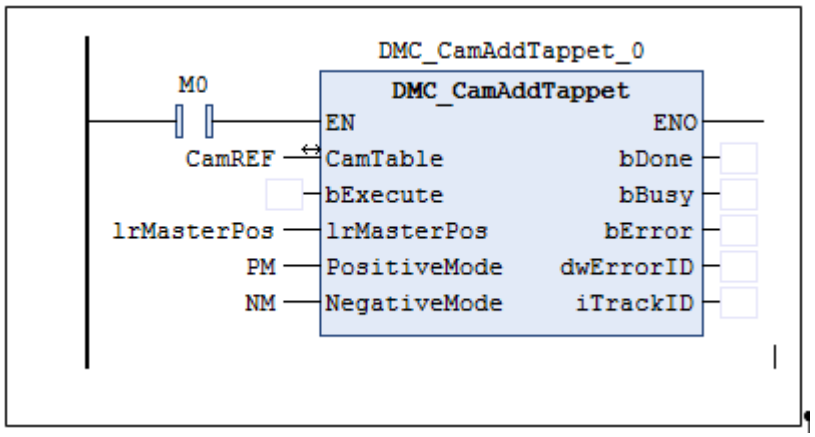
- When an error occurs during the execution of the instruction or the axis is in “Errorstop” state, *bError* will change to True and the axis stops running. Refer to *ErrorID* (Error Code) to address the problem.

- **Programming Example**

  - The example explains the action principle for DMC_CamAddTappetValue by adding a new track of tappet points.
  - Initial setting for tappet points

	Track ID	X	positive pass	negative pass
⊕	1			
		100	switch ON	switch OFF
		500	switch OFF	switch OFF
		1000	switch ON	switch OFF
⊕	7			
		7000	invert	none
⊕	2			
		0	switch ON	switch OFF
⊕				

  - Function block setting



lrMasterPos	ARRAY [0..(GVL.MAX_FB_SWITCH_NUM - 1)] OF LREAL	
lrMasterPos[0]	LREAL	1250
lrMasterPos[1]	LREAL	7050
lrMasterPos[2]	LREAL	3050
lrMasterPos[3]	LREAL	0
lrMasterPos[4]	LREAL	0
lrMasterPos[5]	LREAL	0
lrMasterPos[6]	LREAL	0
lrMasterPos[7]	LREAL	0
PM	ARRAY [0..(GVL.MAX_FB_SWITCH_NUM - 1)] OF DMC_CAMTAPPE...	
PM[0]	DMC_CAMTAPPETACTION	TAPPETACTION_on
PM[1]	DMC_CAMTAPPETACTION	TAPPETACTION_off
PM[2]	DMC_CAMTAPPETACTION	TAPPETACTION_inv
PM[3]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PM[4]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PM[5]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PM[6]	DMC_CAMTAPPETACTION	TAPPETACTION_none
PM[7]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NM	ARRAY [0..(GVL.MAX_FB_SWITCH_NUM - 1)] OF DMC_CAMTAPPE...	
NM[0]	DMC_CAMTAPPETACTION	TAPPETACTION_off
NM[1]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NM[2]	DMC_CAMTAPPETACTION	TAPPETACTION_inv
NM[3]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NM[4]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NM[5]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NM[6]	DMC_CAMTAPPETACTION	TAPPETACTION_none
NM[7]	DMC_CAMTAPPETACTION	TAPPETACTION_none

- See the tappet table before the function block is run

Track ID	Master axis position	Direction	Mode
1	100	Negative	TAPPETACTION_off
1	100	Positive	TAPPETACTION_on
1	500	Negative	TAPPETACTION_off
1	500	Positive	TAPPETACTION_off
1	1000	Negative	TAPPETACTION_off
1	1000	Positive	TAPPETACTION_on
7	7000	Positive	TAPPETACTION_inv
2	0	Negative	TAPPETACTION_off
2	0	Positive	TAPPETACTION_on

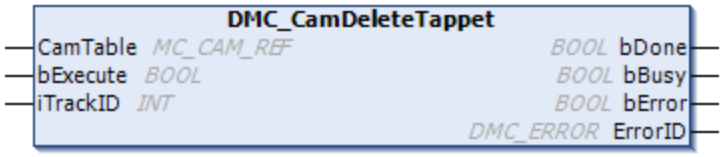
- See the tappet table after the function block is run

Track ID	Master axis position	Direction	Mode
1	100	Negative	TAPPETACTION_off
1	100	Positive	TAPPETACTION_on
1	500	Negative	TAPPETACTION_off
1	500	Positive	TAPPETACTION_off
1	1000	Negative	TAPPETACTION_off
1	1000	Positive	TAPPETACTION_on
7	7000	Positive	TAPPETACTION_inv
2	0	Negative	TAPPETACTION_off
2	0	Positive	TAPPETACTION_on
3	1250	Negative	TAPPETACTION_off
3	1250	Positive	TAPPETACTION_on
3	7050	Positive	TAPPETACTION_off
3	3050	Negative	TAPPETACTION_inv
3	3050	Positive	TAPPETACTION_inv

2.3.2.12 DMC_CamDeleteTappet

- **Supported Devices**: AX308E, AX-364E

DMC_CamDeleteTappet deletes the specified tappet track.

FB/FC	Instruction	Graphic Expression
FB	DMC_CamDeleteTappet	

ST Language

```
DMC_CamDeleteTappet_instance(
CamTable : =,
bExecute : =,
iTrackID : =,
bDone =>,
bBusy =>,
bError =>,
ErrorID =>) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (True)	-
iTrackID	Specify the ID of the track to be deleted.	INT	1~512 (0)	When *bExecute* is True.

- **Outputs**

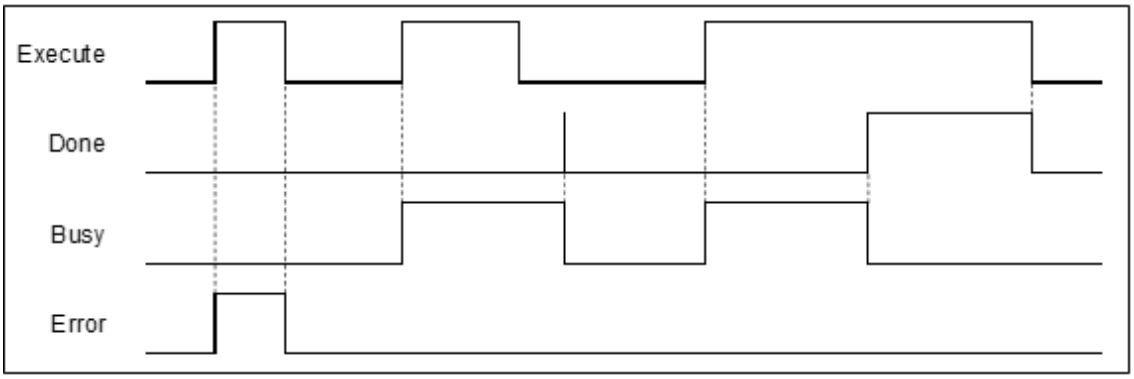
Name	Function	Data Type	Output range (Default Value)
bDone	True when the instruction is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the instruction is completed.	• When *bError* is True. • When *bExecute* turns to False.
bBusy	• When *bExecute* is True.	• When *bDone* is True. • When *bError* is True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns from True to False.
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF	MC_CAM_REF*	When *bExecute* changes to True.

***Note**: MC_CAM_REF (FB): The basic CAM.

- **Function**

  - The tappet table can be set in the cam table. Tappets can be used to track the current position of the master axis and output a signal to trigger an event under particular conditions.
  - One tappet table has multiple tappet tracks.
  - The function block deletes a specified tappet track from the tappet table.
  - The tappet track
    - One tappet track contains one track ID, one tappet switch (Boolean signal) and multiple tappet points.
  - The tappet
    - One tappet point includes the Track ID of the tappet track where the tappet point is, the master axis position corresponding to the tappet point, the positive passing mode and the negative passing mode.

See the modes and their meanings in the following table.

Mode	Function	Action
TAPPETACTION_none	No action	The tappet switch does not take an action when the master axis passes the point.
TAPPETACTION_on	ON	The tappet switch is enabled when the master axis passes the point.
TAPPETACTION_off	OFF	The tappet switch is disabled when the master axis passes the point.
TAPPETACTION_inv	Invert	The tappet switch status is inverted when the master axis passes the point.
TAPPETACTION_time	ON	When the master axis passes the point, the tappet switch is ON for a set period of time and then turns OFF.

***Note:**

When the mode is set to TAPPETACTION_time in this function block, the tappet switch will turn off after being ON for a fixed 100 ms.

- **Troubleshooting**

  - When an error occurs during the execution of the instruction or the axis enters “Errorstop” state, *bError* will change to True and the axis stops running. Refer to *ErrorID* (Error Code) to address the problem.

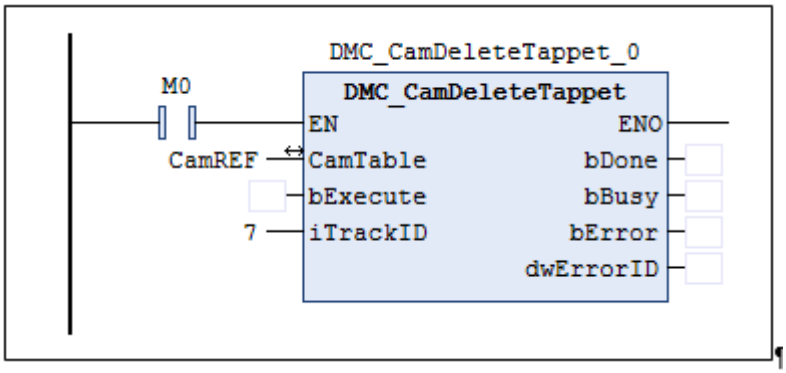
- **Programming Example**

  The example explains the action principle for DMC_CamDeleteTappet by deleting the specified track data from the tappet table.

  - Initial tappets setting

	Track ID	X	positive pass	negative pass
⊕	1			
		100	switch ON	switch OFF
		500	switch OFF	switch OFF
		1000	switch ON	switch OFF
⊕	7			
		7000	invert	none
⊕	2			
		0	switch ON	switch OFF
⊕				

  - Function block setting



  - See the tappet table before the function block is run

Track ID	Master axis position	Direction	Mode
1	100	Negative	TAPPETACTION_off

Track ID	Master axis position	Direction	Mode
1	100	Positive	TAPPETACTION_on
1	500	Positive and negative	TAPPETACTION_off
1	1000	Negative	TAPPETACTION_off
1	1000	Positive	TAPPETACTION_on
7	7000	Positive	TAPPETACTION_inv
2	0	Negative	TAPPETACTION_off
2	0	Positive	TAPPETACTION_on

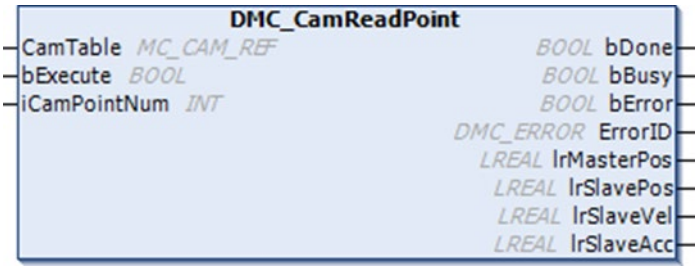
- See the tappet table after the function block is run

Track ID	Master axis position	Direction	Mode
1	100	Negative	TAPPETACTION_off
1	100	Positive	TAPPETACTION_on
1	500	Positive and negative	TAPPETACTION_off
1	1000	Negative	TAPPETACTION_off
1	1000	Positive	TAPPETACTION_on
2	0	Negative	TAPPETACTION_off
2	0	Positive	TAPPETACTION_on

2.3.2.13 DMC_CamReadPoint

- **Supported Devices**: AX308E, AX-364E

DMC_CamReadPoint reads the data of one single cam point.

FB/FC	Instruction	Graphic Expression
FB	DMC_CamReadPoint	

ST Language

```
DMC_CamReadPoint_instance(
CamTable : =parameter,
bExecute: =parameter,
iCamPointNum: =parameter,
bDone =>parameter,
bBusy =>parameter,
bError =>parameter,
ErrorID =>parameter,
lrMasterPos =>parameter,
lrSlavePos =>parameter,
lrSlaveVel =>parameter,
lrSlaveAcc =>parameter) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (True)	-
iCamPointNum	Specify the number of the cam point to be read.	INT	0~256 (0)	When *bExecute* is True.

- **Outputs**

Name	Function	Data Type	Output range (Default Value)
bDone	True when the instruction is completed.	BOOL	True/False (False)

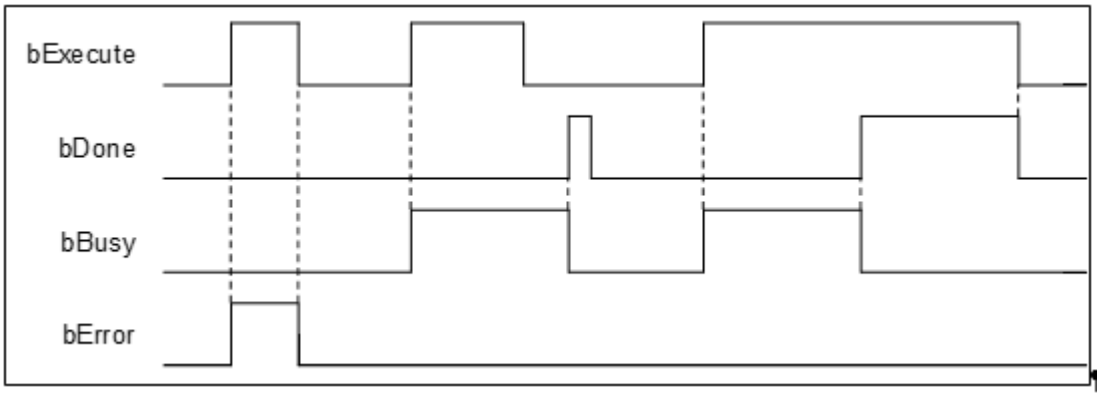
Name	Function	Data Type	Output range (Default Value)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERR)
lrMasterPos	The position of the cam master axis	LREAL	Positive, negative or 0 (0)
lrSlavePos	The position of the cam slave axis	LREAL	Positive, negative or 0 (0)
lrSlaveVel	The velocity of the cam slave axis	LREAL	Positive, negative or 0 (0)
lrSlaveAcc	The acceleration of the cam slave axis	LREAL	Positive, negative or 0 (0)

***Note:** DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When *bExecute* changes to True.	• When *bError* turns to True. • When *bExecute* turns to False.
bBusy	• When *bExecute* changes to True.	• When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns to False.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified	MC_CAM_REF	MC_CAM_REF*	When *bExecute* changes to True.

	cam table			

***Note:** MC_CAM_REF (FB): The basic CAM.

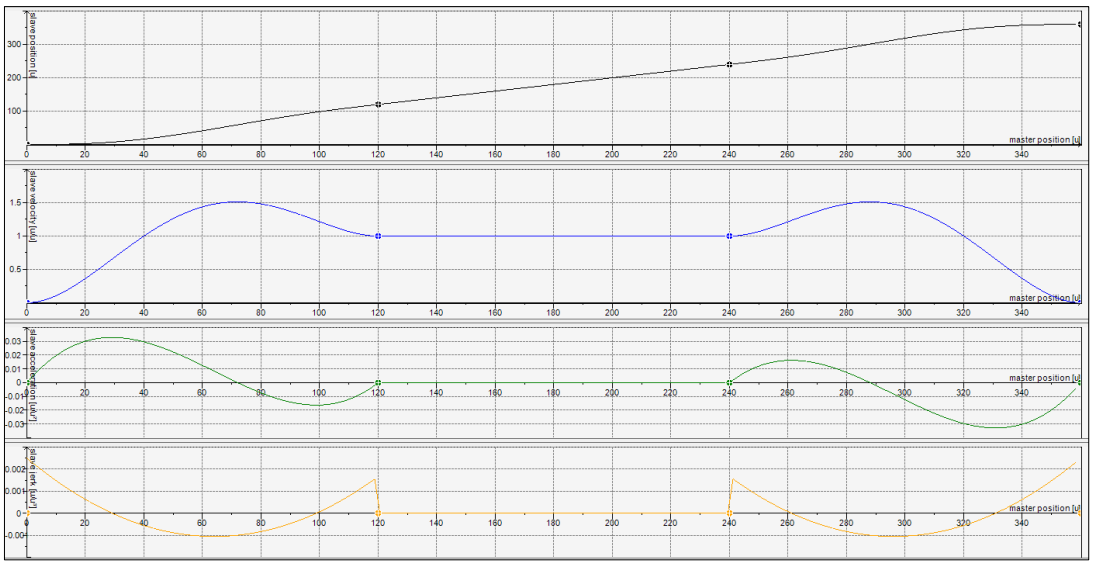
- **Function**

  - *CamTable* determines which cam table is to read. *iCamPointNum* determines the number of the cam point to read. *lrMasterPos* shows the master position that the cam point corresponds to. *lrSlavePos* shows the slave position that the cam point corresponds to, i.e. coordinates of the cam point. *lrSlaveVel* is the slave velocity that the cam point corresponds to. *lrSlaveAcc* is the slave acceleration rate that the cam point corresponds to.
  - If no data of the specified cam point exists, the output will show Infinity.

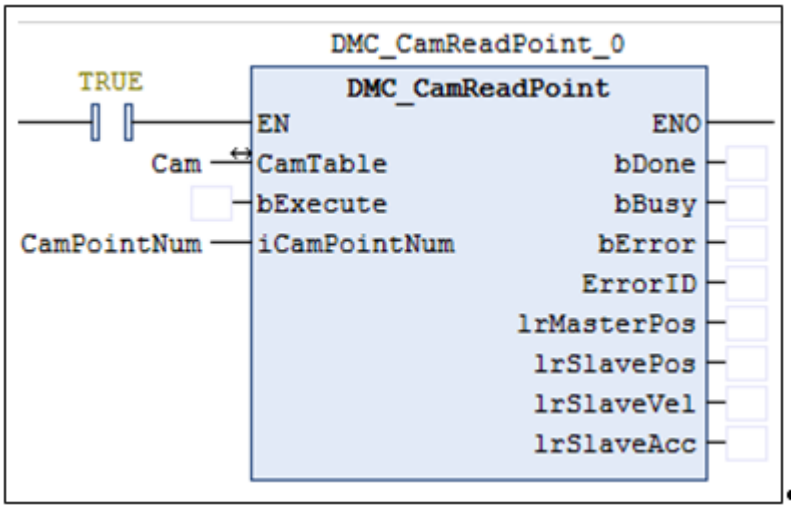
- **Programming Example**

1. Build a cam table: “Cam”.

	X	Y	V	A	J	Segm...	min(P...	max(P...	max(|V...	max(|A...
	0	0	0	0	0					
						Poly5	0	120	1.5120...	0.0328...
	120	120	1	0	0					
						Poly5	120	240	1	0
	240	240	1	0	0					
						Poly5	240	360	1.512	0.0328...
	360	360	0	0	0					



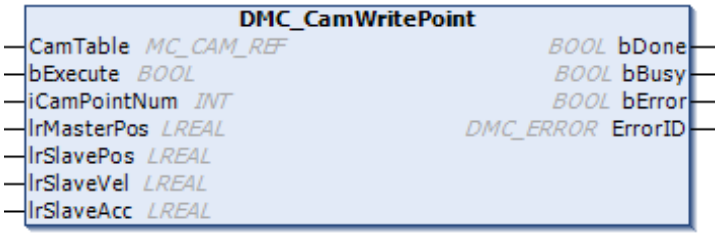
2. Input Cam for the specified *CamTable* and 2 for *CamPointNum*, the number of the cam point to be read.
3. Set DMC_CamReadPoint_0.*bExecute* to True.
4. DMC_CamReadPoint_0.lrMasterPos: 240, DMC_CamReadPoint_0.lrSlavePos: 240, DMC_CamReadPoint_0.lrSlaveVel: 1 and DMC_CamReadPoint_0.lrSlaveAcc: 0 can be read while the instruction is waiting until DMC_CamReadPoint_0.bDone changes from False to True.


DMC_CamReadPoint_0
DMC_CamReadPoint
TRUE
EN
ENO
Cam
CamTable
bDone
bExecute
bBusy
CamPointNum
iCamPointNum
bError
ErrorID
lrMasterPos
lrSlavePos
lrSlaveVel
lrSlaveAcc

2.3.2.14 DMC_CamWritePoint

- **Supported Devices**: AX308E, AX-364E

DMC_CamWritePoint writes the data of one single cam point.

FB/FC	Instruction	Graphic Expression
FB	DMC_CamWritePoint	

ST Language

```
DMC_CamWritePoint_instance(
CamTable : =,
bExecute: =,
bCamChangedPoint: =,
lrMasterPos : =,
lrSlavePos : =,
lrSlaveVel : =,
lrSlaveAcc : =,
bDone =>,
bBusy =>,
bError =>,
ErrorID => ) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (True)	-
iCamPointNum	Specify the number of the cam point to be written.	INT	0~256 (0)	When *bExecute* is True.
lrMasterPos	Specify the position of the cam master axis	LREAL	Positive, negative or 0 (0)	When *bExecute* is True.
lrSlavePos	Specify the position of the cam slave axis	LREAL	Positive, negative or 0 (0)	When *bExecute* is True.
lrSlaveVel	Specify the velocity of the cam slave axis	LREAL	Positive, negative or 0 (0)	When *bExecute* is True.
lrSlaveAcc	Specify the acceleration of the cam slave axis	LREAL	Positive, negative or 0 (0)	When *bExecute* is True.

- **Outputs**

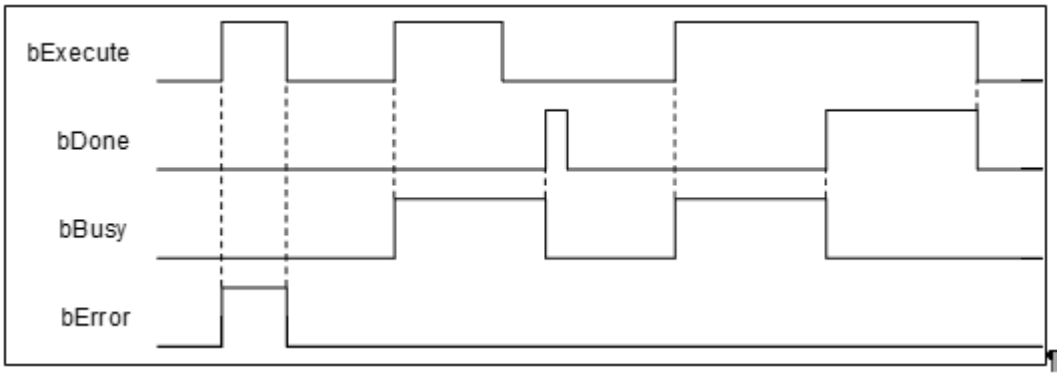
Name	Function	Data Type	Output range (Default Value)
bDone	True when the instruction is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERR)

***Note:** DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When *bExecute* changes to True.	• When *bError* turns to True. • When *bExecute* turns to False.
bBusy	• When *bExecute* changes to True.	• When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns to False.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF	MC_CAM_REF*	When *bExecute* changes to True.

***Note:** MC_CAM_REF (FB): The basic CAM.

- **Function**

  - The *CamTable* determines which cam table to write data to and *iCamPointNum* determines the cam point number to be written. *lrMasterPos* and *lrSlavePos* write respectively the master axis position and slave axis position of the cam

data point (i.e. cam point coordinate positions). *lrSlaveVel* writes the slave axis velocity corresponding to the cam data point and *lrSlaveAcc* writes the slave axis acceleration corresponding to the cam data point.

- The data point information accessed in the cam table corresponds to different data according to different cam Data Types. When DMC_CamWritePoint writes data, the cam operation will not be affected even if parameters are input to the inputs of the instruction if no specific data (e.g. *lrSlaveVel* and *lrSlaveAcc*) is accessed in the cam table.
- When DMC_CamWritePoint modifies the cam table data in the synchronized cam motion, the slave axis in synchronization will change its path immediately, which may cause a jolt of the mechanism.
- When the starting or ending cam data points are modified and the master axis position written by *lrMasterPos* exceeds the range of the original cam table, the running cam will have no change. And the cam table with the modified boundary range cannot work until the MC_Camtableselect is restarted.

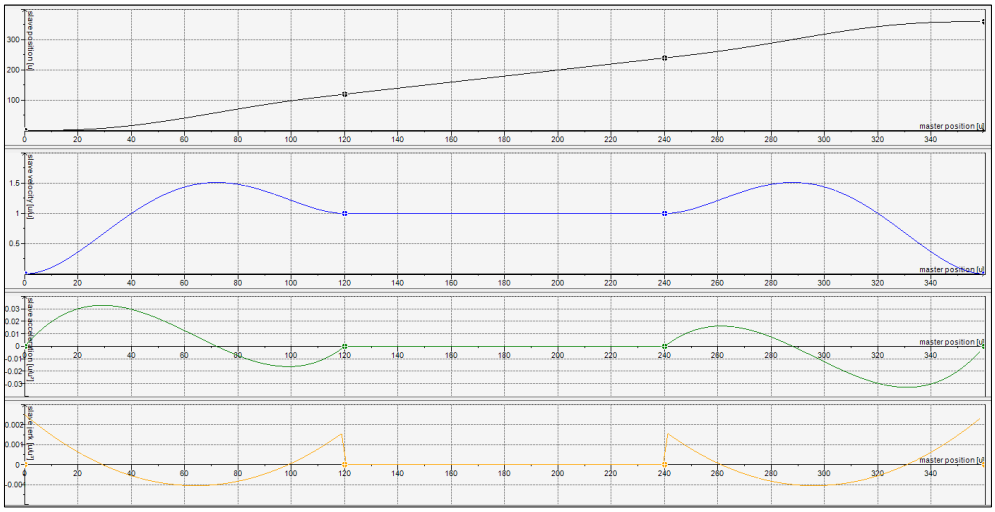
- **Troubleshooting**

  - When an error occurs during the execution of the instruction or the axis enters Errorstop state, *bError* will change to True and the axis stops running. Refer to *ErrorID* (Error Code) to address the problem.

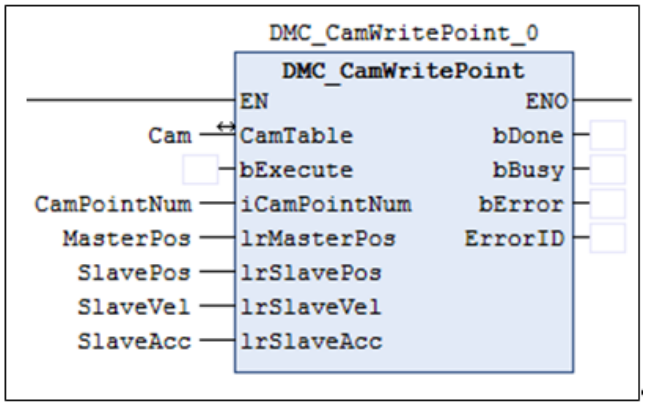
- **Programming Example**

1. Build a cam table: “Cam”.

	X	Y	V	A	J	Segm...	min(P...	max(P...	max(|V...	max(|A...
	0	0	0	0	0					
						Poly5	0	120	1.5120...	0.0328...
	120	120	1	0	0					
						Poly5	120	240	1	0
	240	240	1	0	0					
						Poly5	240	360	1.512	0.0328...
	360	360	0	0	0					



2. Input Cam for the specified *CamTable* and 2 for *CamPointNum*, the number of the cam point to be written.
3. Input 300 for *MasterPos*, 250 for *SlavePos*, 2 for *SlaveVel*, 4 for *SlaveAcc* in the selected cam point data.
4. Set DMC_CamWritePoint_0.*bExecute* to True.
5. The data writing is completed when DMC_CamWritePoint_0.bDone changes from False to True.



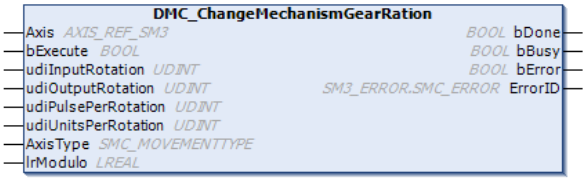
6. After the writing is finished, the actual values for the cam table “Cam” are shown as below.

	X	Y	V	A
0	0	0	0	0
1	120	120	1	0
2	300	250	2	4
3	360	360	0	0

2.3.2.15 DMC_ChangeMechanismGearRation

- **Supported Devices**: AX-308E, AX-364E

DMC_ChangeMechanismGearRation modifies the ratio between user units and pulses, axis type and user units per rotation of the rotary axis.

FB/FC	Instruction	Graphic Expression
FB	DMC_ChangeMechanismGearRation	

ST Language

```
DMC_ReadMotionState_instance(
Axis : =,
bExecute: =,
udiInputRotation: =,
udiOutputRotation: =,
udiPulsePerRotation: =,
udiUnitsPerRotation: =,
AxisType: =,
lrModulo: =,
bDone=>,
bBusy=>,
bError=>,
ErrorID=>,) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
udiInputRotation	Specify the input of the gearbox.	UDINT	Positive (0)	When *bExecute* is True and *bBusy* is False.
lrOutputRotation	Specify the output of the gearbox.	LREAL	Positive (0)	When *bExecute* is True and *bBusy* is False.
udiPulsePerRotation	Specify how many pulses	UDINT	Positive	When *bExecute* is

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
	per rotation of the input end of the gearbox. (Unit: pulses/ rotation)		(0)	True and *bBusy* is False.
udiUnitsPerRotation	Specify how many units the terminal actuator travels per rotation of the output end of the gearbox	UDINT	Positive (0)	When *bExecute* is True and *bBusy* is False.
AxisType	Specify the axis type.	SMC_MOVEMENTTYPE*	0: rotary 1: linear (rotary)	When *bExecute* is True and *bBusy* is False.
lrModulo	Specify how many units per rotation of the rotation axis.	LREAL	Positive (0)	When *bExecute* is True and *bBusy* is False.

*__Note__: SMC_MOVEMENTTYPE: Enumeration (ENUM)

- **Outputs**

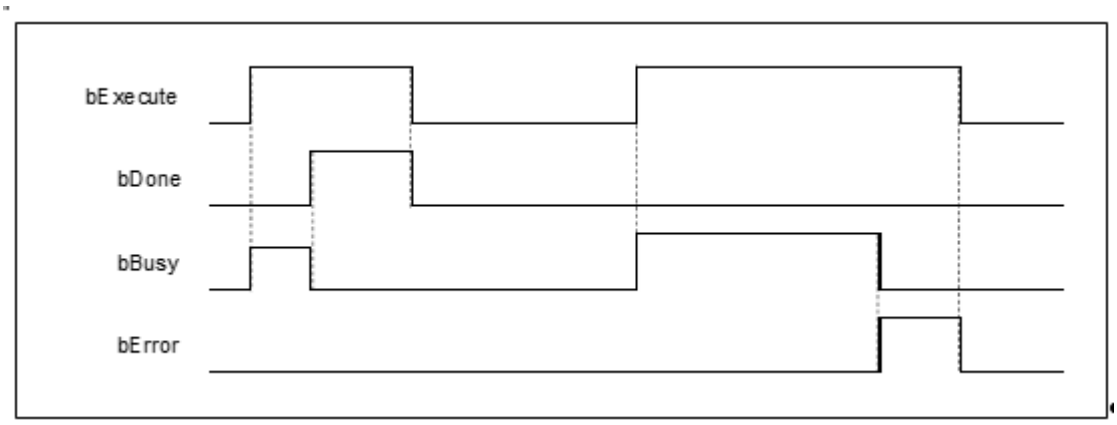
Name	Function	Data Type	Output range (Default Value)
bDone	True when the instruction is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note:** DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the instruction is completed.	• When *bExecute* turns to False. • When *bError* turns to True.
bBusy	• When *bExecute* turns to True.	• When *bExecute* turns to False. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	• When *bExecute* turns from True to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**

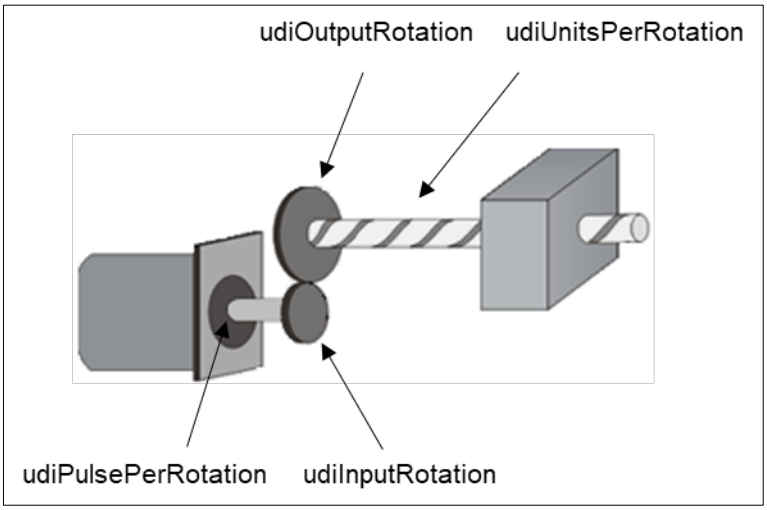


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *bExecute* is True and *bBusy* is False.

*__Note__: AXIS_REF_SM3 (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**



  - DMC_ChangeMechanismGearRation modifies the ratio between user units and pulses, axis type and user units per rotation of the rotary axis.
  - The corresponding relationship between the function block inputs and the mechanism is shown in the figure above. *udiPulsePerRotation* is the number of pulses required per rotation of the input end of the gearbox, *udiInputRotation* is the input of the gearbox, *udiOutputRotation* is the output of the gearbox and *IrUnitsPerRotation* is the number of terminal actuator travel units per rotation of the output end of the gearbox.

- **Troubleshooting**

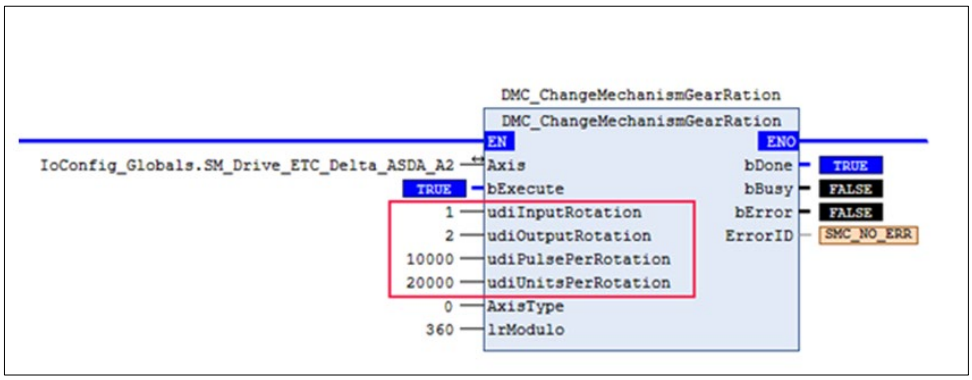
  - The instruction can be run only when the state machine is power_off. Refer to *ErrorID* (Error Code) to address the problem if an error occurs during the instruction execution.

- **Programming Example**

- The example shows the behavior of DMC_ChangeMechanismGearRation.
- Relevant parameters setting:

udiOutputRotation: udiInputRotation= 2: 1
udiPulsePerRotation: 10000 pulses
udiUnitsPerRotation: 20000 us
AxisType: 0 (rotary axis)
lrModulo: 360

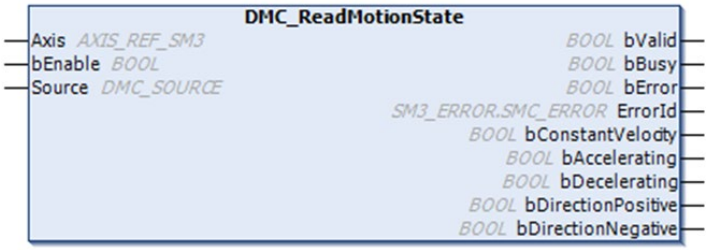
- **Function block setting**



2.3.2.16 DMC_ReadMotionState

- **Supported Devices**: AX-308E, AX-364E

DMC_ReadMotionState reads the behavior state of the axis in motion.

FB/FC	Instruction	Graphic Expression
FB	DMC_ReadMotionState	

ST Language

```
DMC_ReadMotionState_instance(
Axis : =,
bEnable : =,
Source : =,
bValid =>,
bBusy =>,
bError =>,
bConstantVelocity=>,
bAcclerating =>,
bDecelerating =>,
bDirectionPositive =>,
bDirectionNegative =>,) ;
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is enabled when *Enable* changes from FALSE to TRUE.	BOOL	True/False (False)	-
Source	Select the data source. Commanded: instruction-specified value. Actual: actual value of the axis.	DMC_ SOURCE*	0: dmcCommandedValue (0)	When *bEnable* turns to True and *bBusy* is False.

***Note:** MC_SOURCE: Enumeration (ENUM)

- **Outputs**

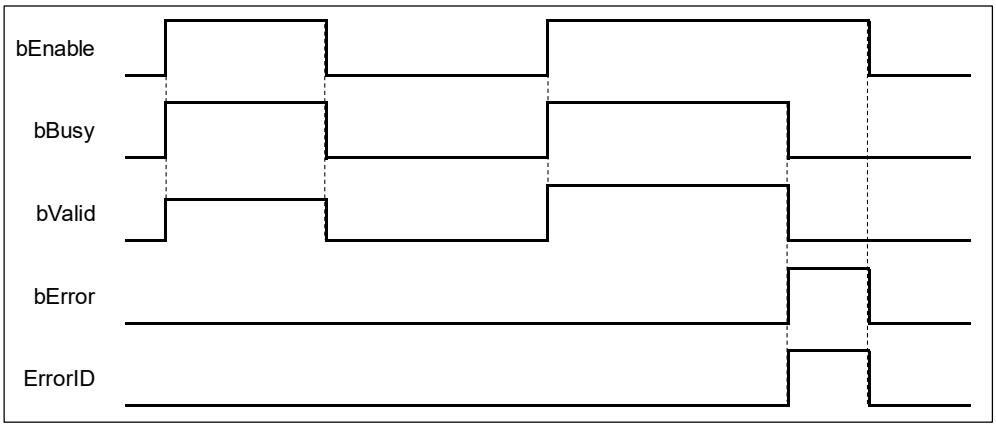
Name	Function	Data Type	Output range (Default Value)
bValid	True when the axis stops and the velocity reaches 0.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
bConstantVelocity	Shows that the current velocity is constant.	BOOL	True/False (False)
bAccelerating	Shows that the absolute value of the current velocity is increasing.	BOOL	True/False (False)
bDecelerating	Shows that the absolute value of the current velocity is decreasing.	BOOL	True/False (False)
bDirectionPositive	Shows that the current position is increasing.	BOOL	True/False (False)
bDirectionNegative	Shows that the current position is decreasing.	BOOL	True/False (False)

***Note:** DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bValid	• When *bEnable* is True and the axis motion state can be read.	• When *bEnable* turns to False. • When *bError* turns to True.
bBusy	• When *bEnable* is True.	• When *bEnable* turns to False. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	• When *bEnable* turns to False. (The error code in ErrorID is cleared.)
ErrorID		
bConstantVelocity	• When the current velocity is constant.	• When *bEnable* is True but the velocity is not constant.
bAccelerating	• When the absolute value of the current velocity is increasing.	• When *bEnable* is True but the velocity does not increase.
bDecelerating	• When the absolute value of the current velocity is decreasing.	• When *bEnable* is True but the velocity does not decrease.
bDirectionPositive	• When the current position is increasing.	• When *bEnable* is True and the direction of motion is not positive. • When *bEnable* is True and the axis does not move any more.
bDirectionNegative	• When the current position is decreasing.	• When *bEnable* is True and the direction of motion is not negative. • When *bEnable* is True and the axis does not move any more.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *bEnable* is True and *bBusy* is False.

***Note:** AXIS_REF_SM3 (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

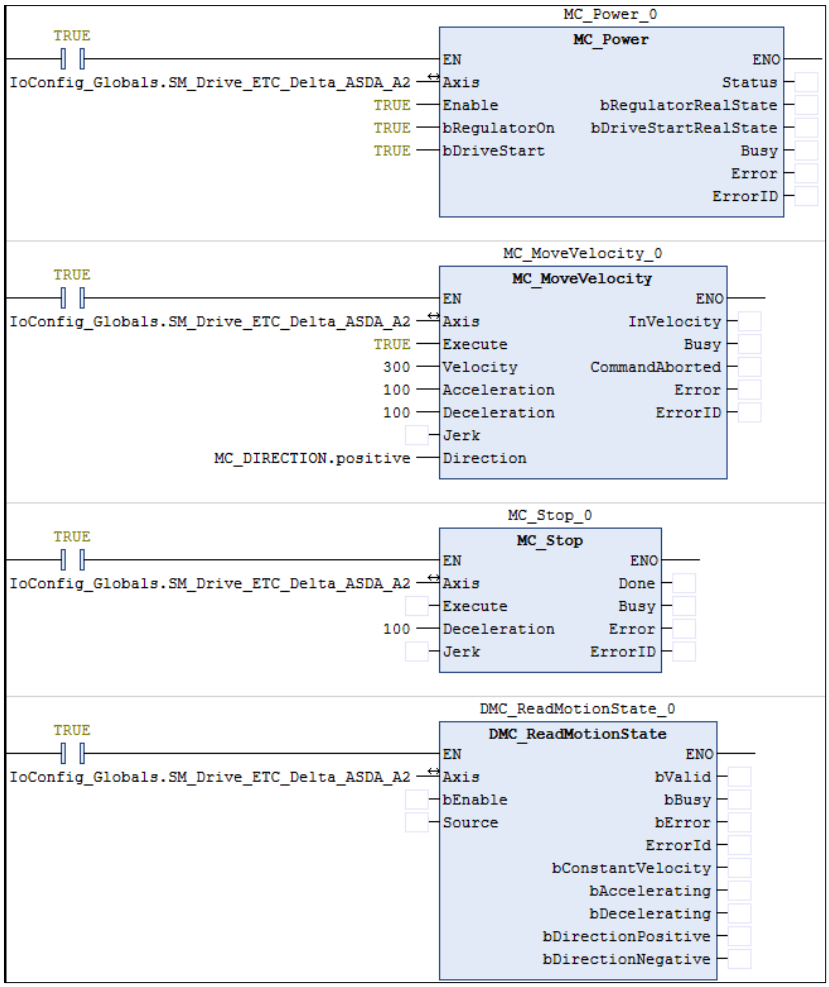
  - DMC_ReadMotionState reads the behavior state of the axis in motion (i.e. acceleration/deceleration, constant velocity, positive/negative direction of motion).
  - When the velocity is 0, the output *bConstantVelocity* changes to True.

- **Troubleshooting**

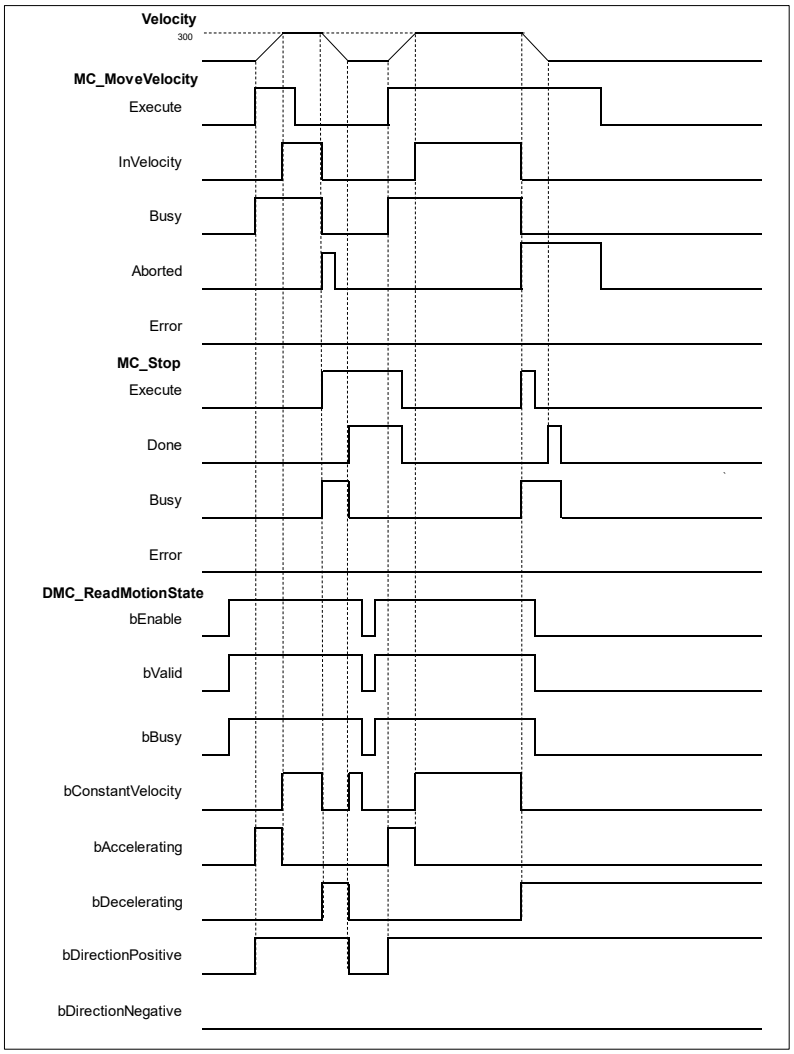
  - When an error occurs during the execution of the instruction, *bError* will change to True. Refer to ErrorID (Error Code) to address the problem.

- **Programming Example**

  - The example shows the motion behavior that MC_ReadMotionState reads MC_MoveVelocity and MC_Stop.



- **Timing Diagram**



1. When *bvalid* and *bBusy* change to True after DMC_ReadMotionState has started, it indicates that the motion state can be read.
2. The axis starts to accelerate until the target velocity after MC_MoveVelocity has started. When *bAccelerating* and *bDirectionPositive* change to True, it indicates that the axis is accelerating in the positive direction.
3. When the axis reaches the specified target velocity, the instruction maintains a constant velocity, *bAccelerating* changes to False and *bConstantVelocity* changes to True.
4. When MC_Stop has started, MC_MoveVelocity is interrupted and the axis starts to decelerate to a stop. Then *bConstantVelocity* changes to False and *bDecelerating* changes to True.
5. When the velocity of the axis reaches 0, *bDecelerating* and *bDirectionPositive* change to False and *bConstantVelocity* changes to True.
6. When DMC_ReadMotionState is disabled during the deceleration of the axis in the next motion cycle, both *bDecelerating* and *bDirectionPositive* will remain True and will not update any longer no matter how motion instructions work.

2.3.2.17 DMC_AxesObserve

- **Supported Devices**: AX-308E, AX-364E

DMC_AxesObserve monitors the deviation between the master axis position and slave axis position and it will output a reminder when the deviation exceeds the allowed value.

FB/FC	Instruction	Graphic Expression
FB	DMC_AxesObserve	

ST Language

```
DMC_AxesObserve_instance(
Master : =,
Slave : =,
bEnable : =,
iReferenceType : =,
bRotarySelectDeviation: =,
lrPermittedDeviation : =,
bEnabled =>,
bInvalid=>,
bBusy =>,
lrDeviatedValue=>,
bError =>,
ErrorID => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is run when *bEnable* changes from False to True.	BOOL	True/False (False)	-
iReferenceType	Specify the position type.	INT	0: Command position 1: Actual position (0)	When *bEnable* is True.
bRotarySelect Deviation	Specify the shorter or longer distance between axes. The parameter is valid only when the master axis and slave axis are both rotary axes.	BOOL	True/False (False)	When *bEnable* is True.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrPermitted Deviation	Specify the permitted deviation between the two axes.	LREAL	Positive or 0(0)	When *bEnable* is True.

- **Outputs**

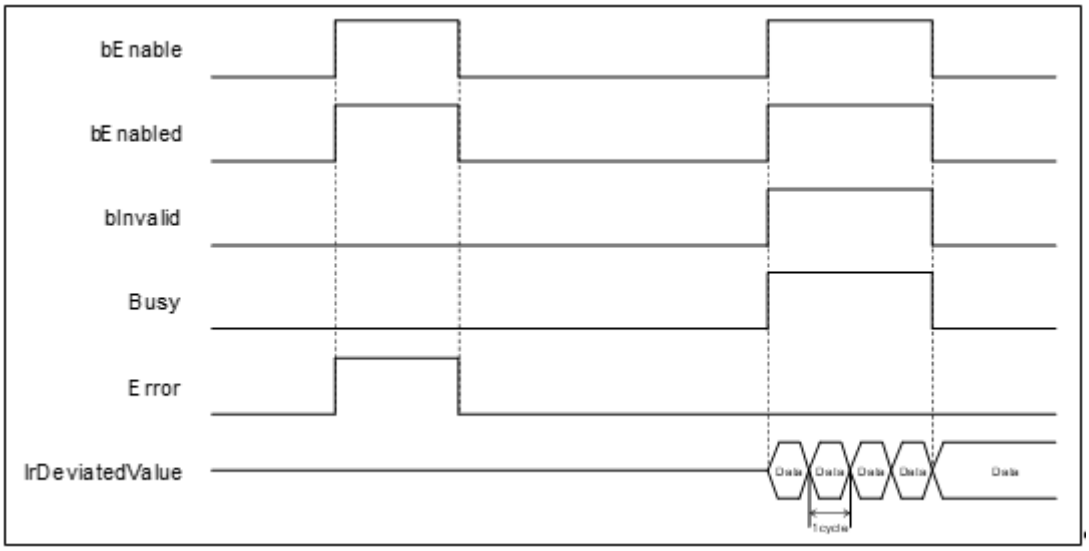
Name	Function	Data Type	Output range (Default Value)
bEnabled	True when the instruction outputs are valid.	BOOL	True/False (False)
bInvalid	True when the difference between the two axes exceeds the allowed value.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
lrDeviatedValue	The error value between the two axes	LREAL	Positive, negative or 0 (0)
bError	True when an error occurs during instruction execution.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERR)

***Note:** DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bEnabled	• When *bEnable* turns to True.	• When *bError* turns to True. • When *bEnable* turns to False.
bInvalid	• When *bEnable* turns to True. • When the difference between the two axes exceeds the allowed value.	• When *bError* turns to True. • When *bEnable* turns to False.
bBusy	• When *bEnable* turns to True.	• When *bError* turns to True. • When *bEnable* turns to False.
lrDeviatedValue	• When *bEnable* turns to True.	• When *bEnable* turns to False, the data update stops.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bEnable* turns to False.
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	Specify the master axis.	AXIS_REF_SM3	AXIS_REF_SM3*	When *bEnable* turns to True.
Slave	Specify the slave axis.	AXIS_REF_SM3	AXIS_REF_SM3*	

***Note:**

AXIS_REF_SM3 (FB): The interface is built in every function block and works as the starting program of the function block.

- **Function**

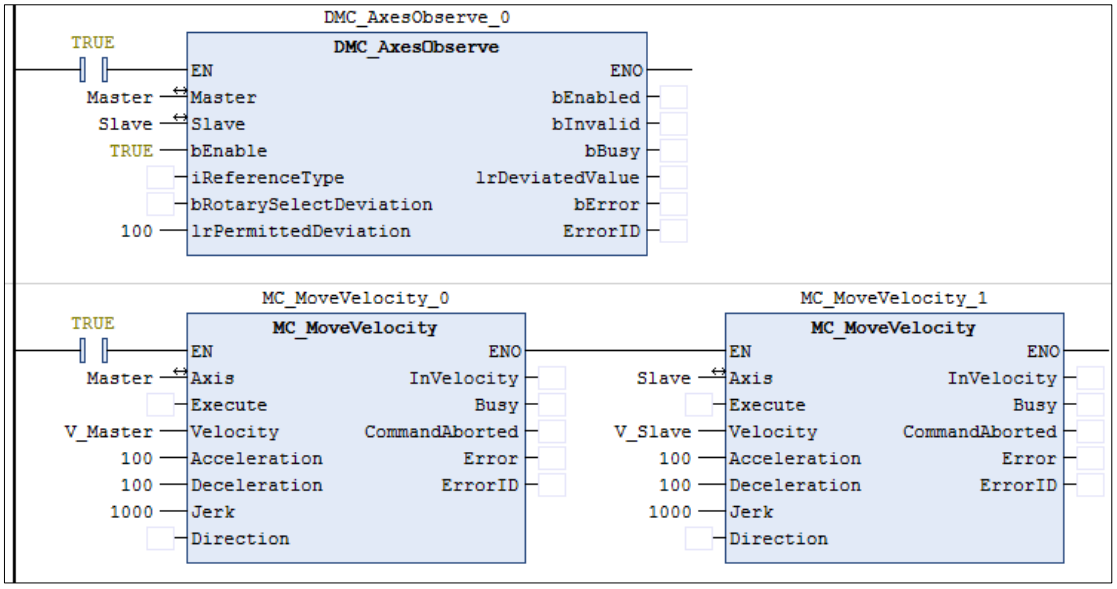
  - DMC_AxesObserve checks if the deviation between the master axis position and slave axis position exceeds the allowed Setting Value.

    When ｜lrDeviatedValue｜is > lrPermittedDeviation, *bInvalid* changes to True.
  - It is suggested that the same mode should be set for the master and slave axes. If one is set as a linear axis and the other is set as a rotary axis, the calculation of the error between axes will be done in linear axis mode.
  - When both the master axis and slave axis are rotary axes but their distances per rotation are different, the calculation of *lrDeviatedValue* (error between axes) is done in linear axis mode.
  - *bRotarySelectDeviation* is valid only when the distances per rotation for the master axis and slave axis are the same. False means to read the shorter direction and True means to read the longer direction.
  - Calculation of *lrDeviatedValue*

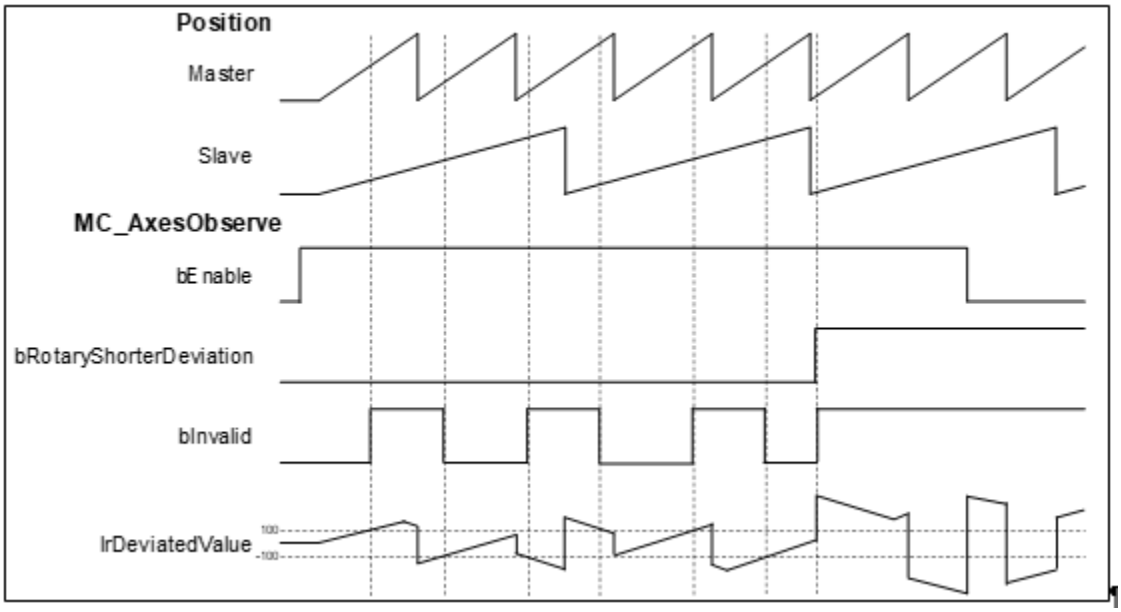
Master axis mode	Slave axis mode	Calculation method
Linear axis	Linear axis	*lrDeviatedValue* (Error between axes) = Master axis position – Slave axis position
Rotary axis	Linear axis	
Linear axis	Rotary axis	
Rotary axis	Rotary axis	*lrDeviatedValue* (Error between axes) = Master axis position – Slave axis position *lrDeviatedValue* (Error between axes) = Distance per rotation– (Master axis position – Slave axis position) *lrDeviatedValue* outputs the value of the longer or shorter distance according to the

Master axis mode	Slave axis mode	Calculation method
		setting of *bRotarySelectDeviation.* When the current positon of the master axis > the current positon of the slave axis, the sign of *lrDeviatedValue* (Error between axes) is positive (+). When the current positon of the master axis < he current positon of the slave axis, the sign of *lrDeviatedValue* (Error between axes) is negative (-).

- **Troubleshooting**
  - When an error occurs during the execution of the instruction, see the error code in *ErrorID* to confirm the current error state.
- **Programming Example**
  - The example illustrates the behavior performed by DMC_AxesObserve when both the master and slave axes are rotary axes with the cycle of 360.



  - **Timing Diagram**



1. When the master and slave axes operate together, the master axis is faster than the slave axis in velocity. Therefore, the starting master axis position is greater than the slave axis position, and the value of *lrDeviatedValue* is positive. When the *lrDeviatedValue* value is greater than the Setting Value of *lrPermittedDeviation* 100, *bInvalid* changes to True.
2. When the master axis rotates one circle, the master axis returns to 0 behind the slave axis in position, then the slave axis position is negative.
3. When *bRotarySelectDeviation* changes to True, it means that *lrDeviatedValue* selects the longer distance between axes and the value must exceed 180 based on the calculation of *lrDeviatedValue* value mentioned in Function section. Since the *lrPermittedDeviation* is set to 100, *bInvalid* must be True.

2.3.2.18 MC_PositionLag

- **Supported Devices**: AX-308E, AX-364E

DMC_PositionLag sets the allowed range of lag error and observe whether the allowed position lag is exceeded.

FB/FC	Instruction	Graphic Expression
FB	DMC_PositionLag	

ST Language

```
DMC_PositionLag_instance(
Axis: =,
bEnable : =,
eStopMode : =,
fMaxPositionLag : =,
fSetActTimeLagCycles : =,
bOutOfRange=>,
bBusy =>,
bError =>,
ErrorID =>,
lrPosLag =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is run when *bEnable* changes from False to True.	BOOL	True/False (False)	-
eStopMode	Specify the stop mode for axis motion when the lag error occurs.	SMC3_CheckPositionLagMode	0: SMC3_PCL_OFF 1: SMC3_PCL_DISABLE 2: SMC3_PCL_HALT 3: SMC3_PCL_ENABLE (SMC3_PCL_OFF)	When *bEnable* and *bBusy* are True.
fMaxPositionLag	Specify the maximum lag error value.	LREAL	Positive or 0 (0)	When *bEnable* and *bBusy* are True.
fSetActTimeLagCycles	Specify the lag cycle between command value and actual	LREAL	Positive or 0 (3)	When *bEnable* and *bBusy* are True.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
	value.			

- **Outputs**

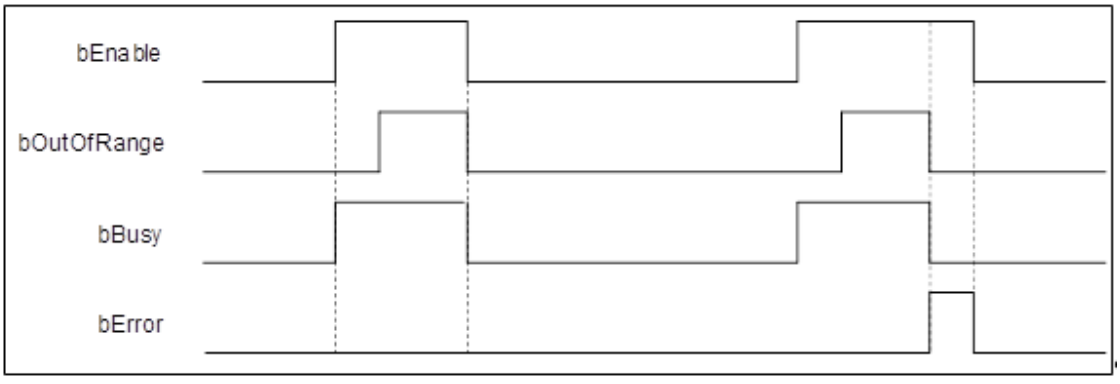
Name	Function	Data Type	Output range (Default Value)
bOutOfRange	True when LagTime exceeds the Setting Value.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	When a command error occurs, record the error code. For the detailed description of the error code, refer to the appendix of the manual.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
lrPosLag	Contains current LagError value	LREAL	Positive or 0 (0)

***Note:** DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bOutOfRange	• True when LagTime exceeds the setting value.	• When *bEnable* turns to False.
bBusy	• When the instruction is being run.	• When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bEnable* turns to False. (Error code is cleared.)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *bExecute* turns to True.

***Note:** AXIS_REF_SM3 (FB): The interface is built in every function block and works as the starting program of the function block.

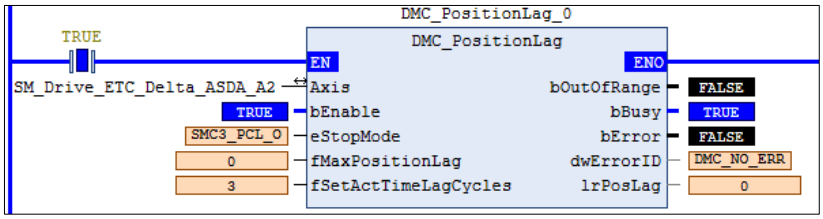
- **Function**
  - Explanation of *eStopMode*

ENUM	ENUM Name
0	SMC3_PCL_OFF
1	SMC3_PCL_DISABLE
2	SMC3_PCL_HALT
3	SMC3_PCL_ENABLE

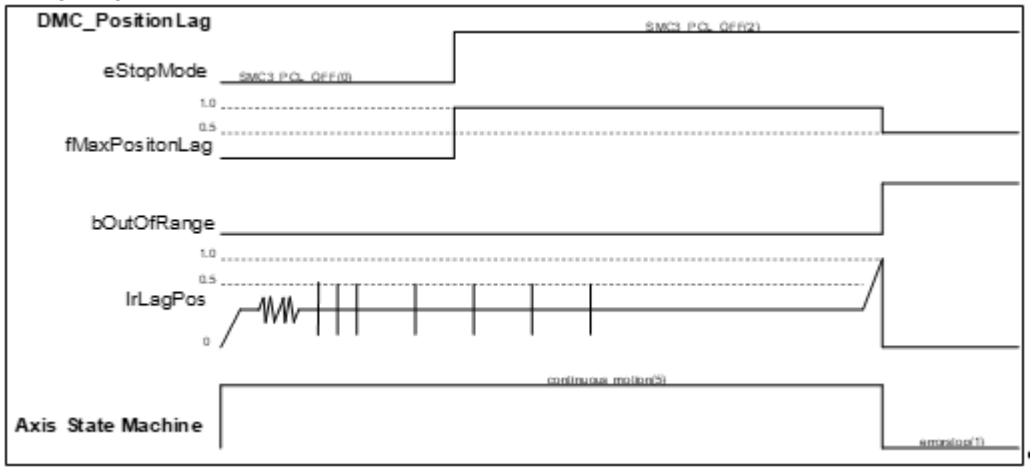
    - SMC3_PCL_OFF: When LagError is out of the allowed range, the axis is still running.
    - SMC3_PCL_DISABLE: When LagError is out of the allowed range, the axis parameter *bRegulatorOn* changes to False.
    - SMC3_PCL_HALT: When LagError is out of the allowed range, the axis parameter *bDriveStart* changes to False.
    - SMC3_PCL_ENABLE: When LagError is out of the allowed range and the axis stops, there are no change for *bRegulatorOn* and *bDriveStar*t states.
  - Calculation and Judgement of Lag Error
    - | (fActPosition + fSetActTimeLagCycles * [Task cycle time] * fActVelocity – fSetPosition) | > fMaxPositionLag
    - *fSetActTimeLagCycles* sets the lag cycle between the command value and actual value. The bigger the difference between the set lag cycle and actual lag cycle, the more likely the lag error will occur due to the error between the command position and actual position.
  - When LagError is out of the range, *bOutOfRange* will change to True and the axis will perform corresponding action based on the setting of *eStopMode*.

- **Troubleshooting**
  - When an error occurs during the execution of the instruction or the axis enters “Errorstop” state, bError changes to True and the axis stops running. To confirm current error state, see the error code in ErrorID.

- **Programming Example**
  - The example illustrates how to observe the position lag state and modify the position lag parameters via DMC_PositionLag.



- Timing Diagram



1. Firstly, set *eStopMode* to SMC3_PCL_OFF. The axis starts to run. Then no matter whether the lag error occurs, the axis will not stop running with *bOutOfRange* of the instruction always being False.
2. Then set *eStopMode* to SMC3_PCL_HALT and set *fMaxPositionLag* to a value which is greater than LagError value. Then the position lag value will never be out of the range for the constant-velocity motion.
3. Eventually, adjust *fMaxPositionLag* to a value which is less than LagError value. Then it can be found that the axis stops running and enters ErrorStop state. And *bOutOfRange* of the instruction turns to True.

2.3.2.19 DMC_SetTorqueLimit

- **Supported Devices**: AX-308E, AX-364E

DMC_SetTorqueLimit sets the maximum torque of an axis.

FB/FC	Instruction	Graphic Expression
FB	DMC_SetTorqueLimit	

ST Language

```
DMC_SetTorqueLimit_instance(
Axis: =,
bExecute : =,
lrMaxTorque : =,
bDone =>,
bBusy =>,
bCommandAborted =>,
bError =>,
ErrorID =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
lrMaxTorque	Specify the maximum rated torque. (Unit: Nm)	LREAL	Positive or 0(0)	When *bExecute* is True and the output *bBusy* is False.

- **Outputs**

Name	Function	Data Type	Output range (Default Value)
bDone	True when the setting is done.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)

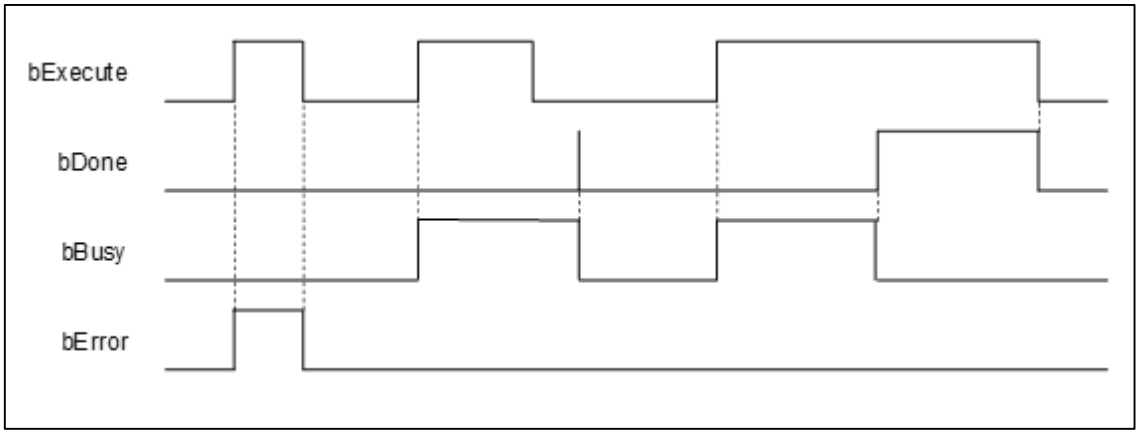
Name	Function	Data Type	Output range (Default Value)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	When a command error occurs, record the error code. For the detailed description of the error code, refer to the Appendix of the manual	DMC_ERROR*1	DMC_ERROR(DMC_NO_ERROR)

***Note:** DMC_ERROR: Enumeration (Enum)

■ **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the setting is completed.	• When *bExecute* turns to False. • *bDone* will change to False after remaining True for one period when *bExecute* is False but *bDone* changes to True.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns from True to False. (Error code is cleared.)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *bExecute* turns to True.

***Note:** AXIS_REF_SM3 (FB): The interface is built in every function block and works as the starting program of the function block.

- **Function**

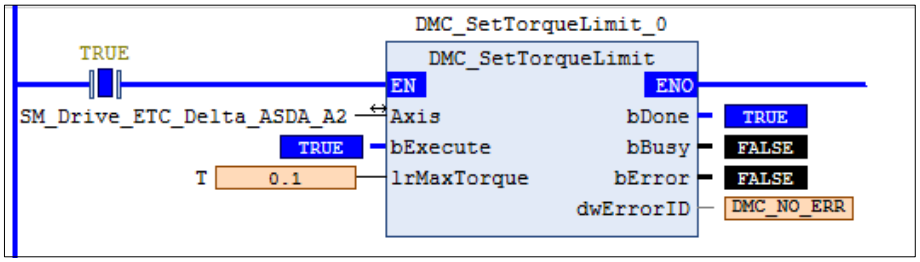
- After the maximum torque is set, the torque of the motor will be limited to the Setting Value to protect the motor from damage caused by the excessive torque when large resistance is encountered.
- DMC_SetTorqueLimit can be used to set the maximum torque of an axis with the unit: Nm.
- DMC_SetTorqueLimit can be used only in CSP or CSV mode

- **Troubleshooting**

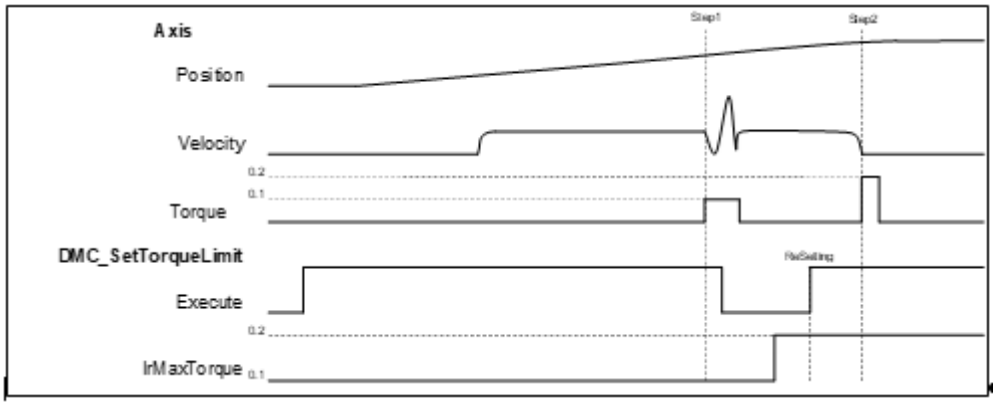
  - When an error occurs during the execution of the instruction or the axis enters “Errorstop” state, *bError* changes to True and the axis stops running. To confirm current error state, see the error code in *ErrorID*.

- **Programming Example**

  - The example illustrates how to use DMC_SetTorqueLimit.



  - **Timing Diagram**



1. Set the maximum rated torque of the axis to 0.1Nm before the operation. Then the operation is performed at a constant velocity.
2. Use the external force to make the axis stop (Step 1) during operation. It can be found that the actual torque of the axis reaches 0.1Nm. Then remove the external force.
3. Set the maximum rated torque to 0.2 Nm and use the external force to make the axis stop (Step2). It is found that the actual torque of the axis reaches 0.2 Nm at the moment. Finally, the Following Error is generated and Servo ON is disabled.

2.3.2.20 DMC_SetSoftwareLimit

- **Supported Devices**: AX-308E, AX-364E

DMC_SetSoftwareLimit is used to enable, disable and set the upper and lower software limits

FB/FC	Instruction	Graphic Expression
FB	DMC_SetSoftwareLimit	

ST Language

```
DMC_SetSoftwareLimit_instance(
Axis: =,
bEnable : =,
bSoftLimitSwitch : =,
lrSWLimitNegative : =,
lrSWLimitPositive : =,
bValid =>,
bBusy =>,
bError =>,
ErrorID =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is run when *bEnable* changes from False to True.	BOOL	True/False (False)	-
bSoftLimitSwitch	Enables or disables software limits.	BOOL	True/False (False)	When *bEnable* and *bBusy* are True.
lrSWLimitNegative	Negative software limit (User unit)	LREAL	Positive, negative or 0	When *bEnable* and *bBusy* are True.
lrSWLimitPositive	Positive software limit (User unit)	LREAL	Positive, negative or 0	When *bEnable* and *bBusy* are True.

- **Outputs**

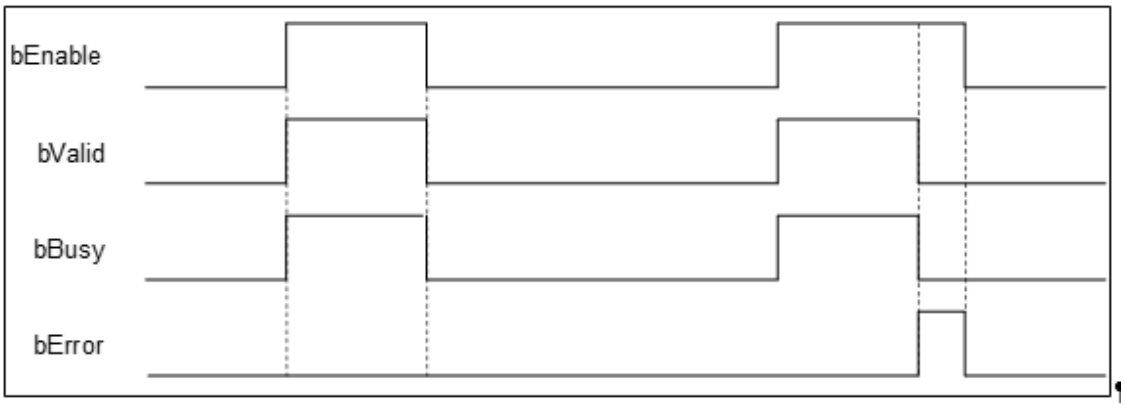
Name	Function	Data Type	Output range (Default Value)
bValid	True when the control over software limit parameters is valid.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	When a command error occurs, record the error code. For the detailed description of the error code, refer to the appendix of the manual.	DMC_ERROR*	DMC_ERROR(DMC_NO_ERROR)

***Note:** DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for changing to TRUE	Timing for changing to FALSE
bValid	• When the control over software limit parameters is valid.	• When *bEnable* turns to False. • When *bError* turns to True.
bBusy	• When *bEnable* turns to True.	• When *bEnable* turns to False. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bEnable* is False. (Error code is cleared.)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*	AXIS_REF_SM3	When *bEnable* turns to True.

***Note:** AXIS_REF_SM3 (FB): The interface is built in every function block and works as the starting program of the function block.

- **Function**

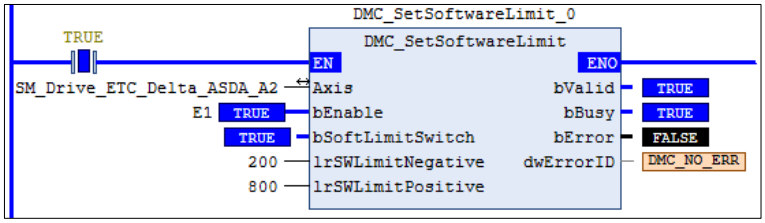
- After DMC_SetSoftwareLimit is run, the writing and monitoring of *bSoftLimitSwitch*, fSWLimitNegative and *fSWLimitPositive* axis parameters will be conducted according to the settings on *bSoftLimitSwitch*, *lrSWLimitNegative* and *lrSWLimitPositive* of the instruction.

- **Troubleshooting**

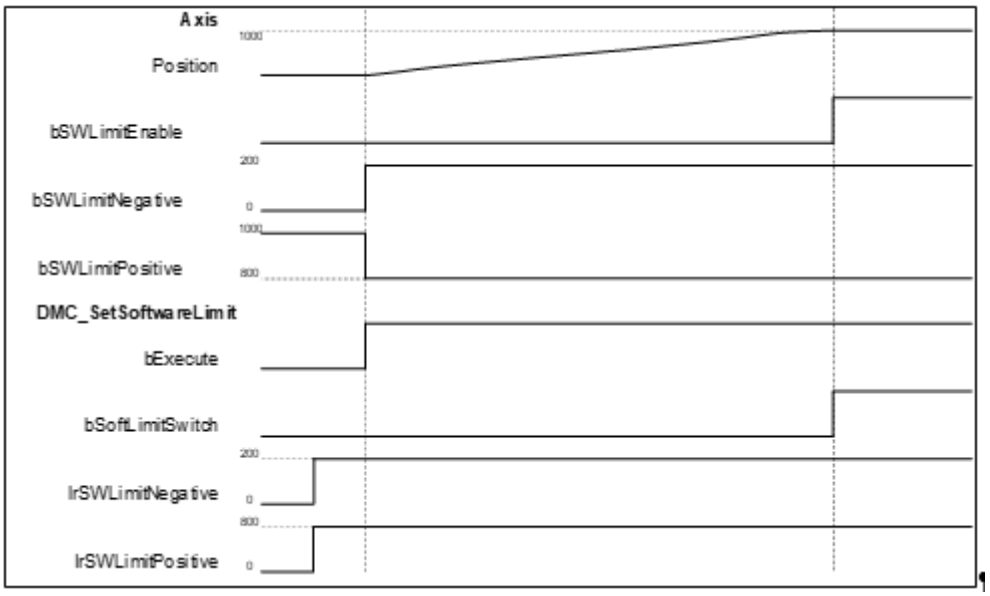
  - When an error occurs during the execution of the instruction or the axis enters “Errorstop” state, *bError* changes to True and the axis stops running. To confirm current error state, see the error code in *ErrorID*.

- **Programming Example**

  - The example illustrates how to use DMC_SetSoftwareLimit to set the software limits.



  - **Timing Diagram**

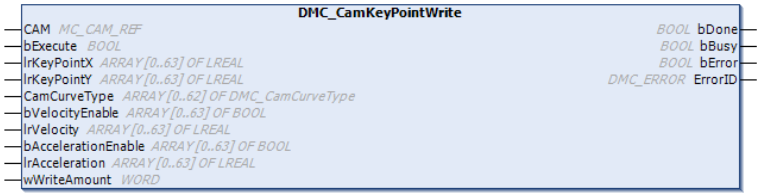


1. After DMC_SetSoftwareLimit starts, the axis parameters writing is conducted based on the set input parameters of the instruction.
2. When *bSoftLimitSwitch* is True, the axis stopping starts as the axis position is outside the software limits.

2.3.2.21 DMC_CamKeyPointWrite

- **Supported Devices**: AX-308E, AX-364E

DMC_CamKeyPointWrite writes key cam points by selecting a curve type and generating corresponding cam curve based on related parameters. After the new cam curve is generated, the selected cam table will be changed accordingly.

FB/FC	Instruction	Graphic Expression
FB	DMC_CamKeyPointWrite	

ST Language

```
DMC_CamKeyPointWrite_instance(
CAM : =,
bExecute : =,
lrKeyPointX : =,
lrKeyPointY : =,
CamCurveType : =,
bVelocityEnable : =,
lrVelocity : =,
bAccelerationEnable : =,
lrAcceleration : =,
wWriteAmount : =,
bDone =>,
bBusy =>,
bError =>,
ErrorID =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
lrKeyPointX	The master axis positions of key points which are set (Unit: user unit)	LREAL [0..63]	Negative, Positive or 0 (0)	When *bExecute* is True and the output *bBusy* is False.
lrKeyPointY	The slave axis positions of key points which are set. (Unit: user unit)	LREAL [0..63]	Negative, Positive or 0 (0)	When *bExecute* is True and the output *bBusy* is False.
CamCurve	Select types of cam	DMC_	0: Line	When *bExecute* is True

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Type	curves between key cam points.	Cam Curve Type [0..62]*	1: Quadratic_Parabola 2: Poly5 3: Basic_Sine 4: Inclined_Sine 5: Mod_Acc_Sine 6: Mod_Acc_Trapezoidal 7: Cubic_Spline_Nature 8: Cubic_Spline_Clamp 9: Cubic_Spline (Line)	and the output *bBusy* is False.
bVelocity Enable	Enable or disable velocity settings of key points.	BOOL [0..63]	Negative, Positive or 0 (0)	When *bExecute* is True and the output *bBusy* is False.
lrVelocity	Velocities of key cam points	LREAL [0..63]	Negative, Positive or 0 (0)	When *bExecute* is True and the output *bBusy* is False.
bAccelerationEnable	Enable or disable acceleration settings of key points	BOOL [0..63]	Negative, Positive or 0 (0)	When *bExecute* is True and the output *bBusy* is False.
lrAcceleration	Acceleration rates of key cam points	LREAL [0..63]	Negative, Positive or 0 (0)	When *bExecute* is True and the output *bBusy* is False.
wWriteAmount	The amount of key cam points which are set	WORD	2~64 (2)	When *bExecute* is True and the output *bBusy* is False.

***Note**: DMC_CamCurveType: Enumeration (ENUM)

- **Outputs**

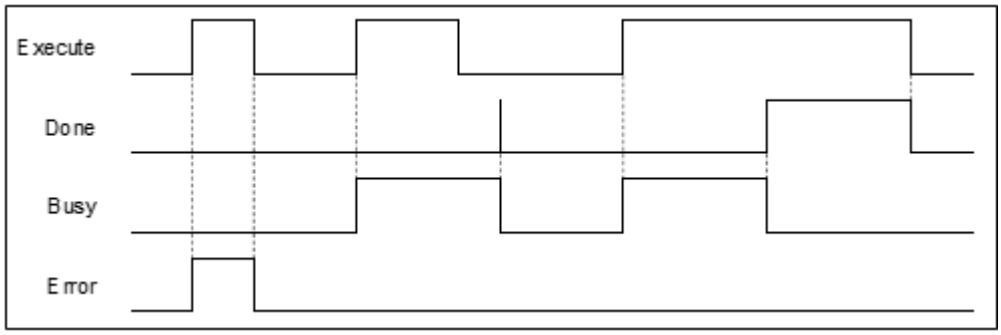
Name	Function	Data Type	Output range (Default Value)
bDone	True when the instruction is complete.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	When a command error occurs, record the error code. For the detailed description of the error code, refer to the Appendix of the manual.	DMC_ERROR*	DMC_ERROR(DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for changing to TRUE	Timing for changing to FALSE
bDone	• When the instruction is completed.	• When *bExecute* turns to False. • *bDone* will change to False after remaining True for one period when *bExecute* is False but *bDone* changes to True.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns from True to False. (Error code is cleared.)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	Specify a cam table.	MC_CAM_REF*	MC_CAM_REF	When *bExecute* turns to True.

***Note:** MC_CAM_REF (FB): The basic CAM.

- **Function**

  - This function block may take a long time to perform the calculation of curves and cam points. Therefore, it is suggested that this function block should be used in a non-EtherCAT Task in order to avoid the issue of Lost Sync in EtherCAT Task when DMC_CamKeyPointWrite is run.

  - CamCurveType

CamCurveType	Description
Line (0)	Used in the situation where the velocity-constant motion is maintained. There will be a large force on the start point and end point of a straight line (the accelerations for the start point and end point of the line segment approaches infinity), which is very obvious in the high speed operation. So the curve type is suitable for use in low-speed operation.
Quadratic_Parabola (1)	Used in the situation where the acceleration must maintain a constant-velocity motion. This type of curve (with non-zero acceleration rates at the start point and end point of the line segment) is more likely to cause shock as well as vibration. So the curve type is suitable for use in lower speed operation.

CamCurveType	Description
Poly5 (2)	Users can set the velocity and acceleration boundaries of the start point and the end point, or automatically continue the velocity and acceleration boundary value of the previous or next segment (via disabling *bVelocityEnable* / *bAccelerationEnable*)
Basic_Sine (2)	Used in the situation where the follower needs to do a simple harmonic motion. This curve is a cosine curve in the acceleration diagram. The positive maximum acceleration rate and negative maximum acceleration rate are at the start position and the end position respectively and it is zero at the middle point. So the Jerk is infinite at the start position and end position, which is prone to shock and vibration. So the curve type is suitable for applications in the medium and low speed operation.
Inclined_Sine (3)	Used in the situation where the follower needs to perform a cycloid motion. This curve is a sine curve in the acceleration diagram, and the acceleration at the start position and the end position is zero, so the jump produced will not cause the acceleration to reach infinity. And thus the curve type can be applied for high-speed operation due to smooth operation.
Mod_Acc_Sine (4)	The acceleration graph of the curve is a sine curve changed from a typical ladder graph, so that the acceleration is smoother. The curve type is applied for high-speed operation.
Mod_Acc_Trapezoidal (5)	The acceleration graph of the curve is a sine curve changed from oblique straight lines for the acceleration and deceleration segments of a typical ladder diagram. So the acceleration has better smoothness. The curve type is applied for high-speed operation.
Cubic_Spline_Nature (6)	The acceleration at the start and end points of the cubic curve is zero. That is, there is no force on both ends of the follower.
Cubic_Spline_Clamp (7)	The velocities for the start and end points of the cubic curve are user-set values. The acceleration rates for both ends are the positive maximum and negative maximum, so shock and vibration are likely to occur.
Cubic_Spline (8)	The cubic curve is used when four or more key points are used as interpolation points in order to link two boundary curves as well as avoid the Runge phenomenon of multi-order curves.

- Velocity Enable / Acceleration Enable
  - Users can enable or disable the velocities and accelerations of key points through the *bVelocityEnable* and *bAccelerationEnable* parameters of the instruction.
  - False means that the user-set velocity or acceleration value is not enabled for curve planning. The boundary condition values of key points will automatically obtain the velocity or acceleration calculated for the previous or next curve segment so as to achieve continuous velocity or acceleration for the intersection of curves. True means that a curve will be produced based on the velocities and accelerations of key points, which are the condition values of *lrVelocity* and *lrAcceleration* set by user.
  - For some of the following curves, the velocity and acceleration of their key points can be specified via *bVelocityEnable* and *bAccelerationEnable.* See details in the following table.

No.	Curve type	VelocityEnable	AccelerationEnable	Velocity	Acceleration
0	Straight line	Not possible *1	Not possible *1	Automatically calculated	0
1	Parabola	Not possible	Not possible	0	Automatically calculated
2	Poly5	Possible	Possible	User can define	User can define
3	Acceleration cosine curve	Not possible	Not possible	0	Automatically calculated
4	Acceleration sine curve	Not possible	Not possible	0	0

No.	Curve type	VelocityEnable	AccelerationEnable	Velocity	Acceleration
5	Modified acceleration sine curve	Possible	Not possible	User can define	0
6	Modified acceleration trapezoidal curve	Not possible	Not possible	0	0
7	Cubic spline curve (nature boundary)*2	Not possible	Not possible	Automatically calculated	0
8	Cubic spline curve (clamp boundary)*2	Possible	Not possible	User can define	Automatically calculated
9	Cubic spline curve*3	Not possible	Not possible	Automatically calculated	Automatically calculated

***Note:**

1. Not possible: The Setting Value is invalid; Possible: the Setting Value is valid.
2. The boundary conditions of the cubic spline curve are classified into nature boundary and clamp boundary. The nature boundary means that the acceleration of the spline curve is specified as 0 and the velocity for both ends of the curve cannot be specified. The clamp boundary means the velocity for both ends of the curve can be specified but the acceleration cannot be specified.
3. The Cubic_Spline curve is a curve for connecting two boundaries and the boundary curves at the two ends of the cubic spline curve must be the same as follows.

CamCurve_Type[0] : = Cubic_Spline_Nature;

CamCurve_Type[1] : = Cubic_Spline;

CamCurve_Type[2] : = Cubic_Spline_Nature;

- Key point number specified by *WriteAmount*
  - The amount of key points specified by *WriteAmount* is up to 64 points, but the key point amount cannot exceed the total resolution of the cam table.
  - Each key point (except the last point) needs to select a curve type, the resolution between the straight lines is fixed as 1, and the resolution of the remaining curves is averaged by the remaining analytical points; but when there are only straight lines in the entire cam table, then the points of the entire cam table will be divided equally by all straight lines.

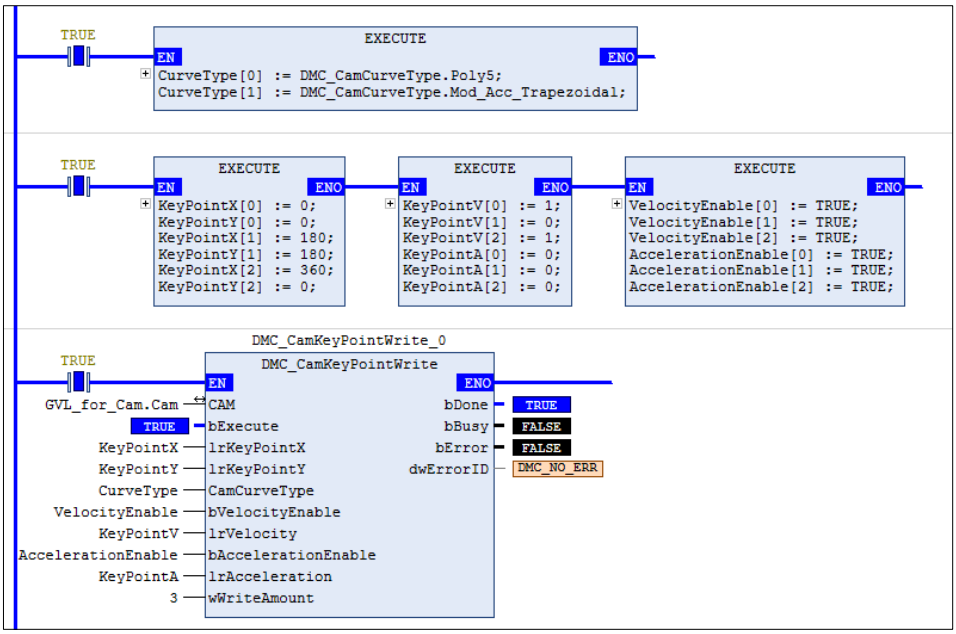
- Curve Types

Curve Type	Description
Line (0)	Used in the situation where the velocity-constant motion is maintained. There will be a large force on the start point and end point of a straight line (the accelerations for the start point and end point of the line segment approaches infinity), which is very obvious in the high speed operation. So the curve type is suitable for use in low-speed operation.
Parabola	Used in the situation where the acceleration must maintain a constant-velocity motion. This type of curve (with non-zero acceleration rates at the start point and end point of the line segment) is more likely to cause shock as well as vibration. So the curve type is suitable for use in lower speed operation.
Poly5	Users can set the velocity and acceleration boundaries of the start point and the end point, or automatically continue the velocity and acceleration boundary value of the previous or next segment (via disabling *bVelocityEnable* / *bAccelerationEnable*)
Acceleration cosine curve	Used in the situation where the follower needs to do a simple harmonic motion. This curve is a cosine curve in the acceleration diagram. The positive maximum acceleration rate and negative maximum acceleration rate are at the start position and the end position respectively

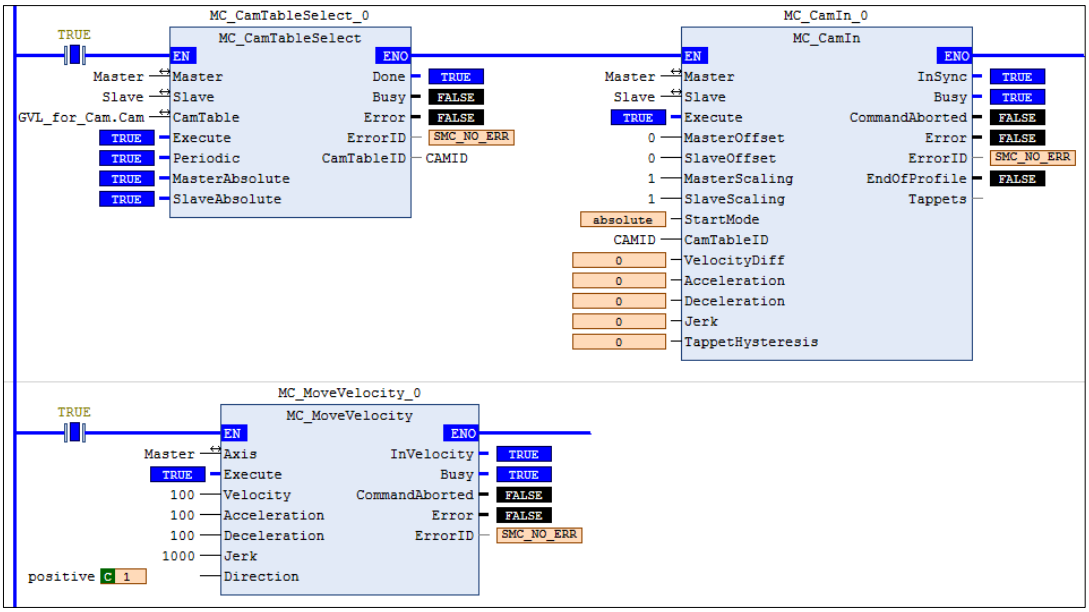
Curve Type	Description
	and it is zero at the middle point. So the Jerk is infinite at the start position and end position, which is prone to shock and vibration. So the curve type is suitable for applications in the medium and low speed operation.
Acceleration sine curve	Used in the situation where the follower needs to perform a cycloid motion. This curve is a sine curve in the acceleration diagram, and the acceleration at the start position and the end position is zero, so the jump produced will not cause the acceleration to reach infinity. And thus the curve type can be applied for high-speed operation due to smooth operation.
Modified acceleration sine curve	The acceleration graph of the curve is a sine curve changed from a typical ladder graph, so that the acceleration is smoother. The curve type is applied for high-speed operation.
Modified acceleration trapezoidal curve	The acceleration graph of the curve is a sine curve changed from oblique straight lines for the acceleration and deceleration segments of a typical ladder diagram. So the acceleration has better smoothness. The curve type is applied for high-speed operation.
Cubic spline curve (nature boundary)	The acceleration at the start and end points of the cubic curve is zero. That is, there is no force on both ends of the follower.
Cubic spline curve (clamp boundary)	The velocities for the start and end points of the cubic curve are user-set values. The acceleration rates for both ends are the positive maximum and negative maximum, so shock and vibration are likely to occur.
Cubic spline curve	The cubic curve is used when four or more key points are used as interpolation points in order to link two boundary curves as well as avoid the Runge phenomenon of multi-order curves.

- **Troubleshooting**
  - When an error occurs during the execution of the instruction or the axis enters “Errorstop” state, *bError* changes to True and the axis stops running. To confirm current error state, see the error code in *ErrorID*.

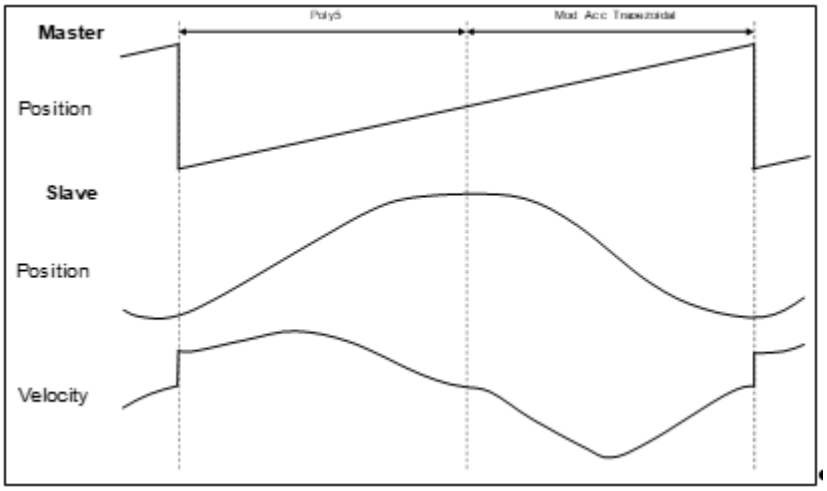
- **Programming Example**
  - Programming Example1:
    - The example illustrates the synchronized motion based on the cam table generated from DMC_ DMC_CamKeyPointWrite.



    - The cam table generated from DMC_CamKeyPointWrite can be used by MC_CamTableSelect directly.



- **Timing Diagram**



Three key points are used to make up a curve. The first segment of the curve is a 5th Polynomial curve, and the second segment is a Mod_Acc Trapezoidal curve.

- Programming Example2:
  - The example illustrates the special applications of cubic interpolation curves:

```
CamCurve_Type[0] := Line;
CamCurve_Type[1] := Cubic_Spline_Nature;   } Case 1.
CamCurve_Type[2] := Cubic_Spline_Nature;   }
CamCurve_Type[3] := Line;
CamCurve_Type[4] := Cubic_Spline_Nature;   }
CamCurve_Type[5] := Cubic_Spline_Nature;   } Case 2.
CamCurve_Type[6] := Cubic_Spline_Nature;   }
CamCurve_Type[7] := 5th Polynomial;
CamCurve_Type[8] := Cubic_Spline_Nature;   }
CamCurve_Type[9] := Cubic_Spline;          }
CamCurve_Type[10] := Cubic_Spline;         } Case 3.
CamCurve_Type[11] := Cubic_Spline_Nature;  }
CamCurve_Type[12] := 5th Polynomial;
```

Case 1. If you want to plan a cubic curve with three key points, just select the curve types with the same boundaries.

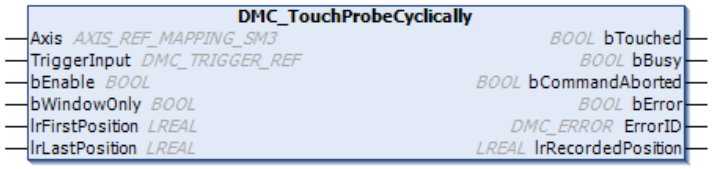
Case 2. If there are three or more boundary condition curves, each two curves will be counted as a segment in the curve planning calculation. If there are no continuous boundary curves, the single curve will be calculated as a boundary condition curve.

Case 3. If there are four or more key points to be on the same curve, you can use spline curves as the continuous line segments of the continuous boundary condition curves at both ends. Then the key points between the two ends will be used as inner interpolation points of the cubic curve.

2.3.2.22 DMC_TouchProbeCyclically

- **Supported Devices**: AX-308, AX-364E

DMC_TouchProbeCyclically can continuously record the captured position of an axis.

FB/FC	Instruction	Graphic Expression
FB	DMC_TouchProbeCyclically	

ST Language

```
DMC_TouchProbeCyclically_instance(
Axis : =,
TriggerInput : =,
bEnable : =,
bWindowOnly : =,
lrFirstPosition : =,
lrLastPosition : =,
bTouched =>,
bBusy =>,
bCommandAborted =>,
bError =>,
ErrorID =>,
lrRecordedPosition =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction is run when *bEnable* changes from False to True.	BOOL	True/False (False)	-
bWindowOnly	Enable the Window range setting.	BOOL	True/False (False)	When *bEnable* and *bBusy* are True.
lrFirstPosition	Defines the start position of Window. (Unit: user unit)	LREAL	Negative, Positive or 0 (0)	When *bEnable* and *bBusy* are True.
lrLastPosition	Defines the end position of Window. (Unit: user unit)	LREAL	Negative, Positive, or 0 (0)	When *bEnable* and *bBusy* are True.

- **Outputs**

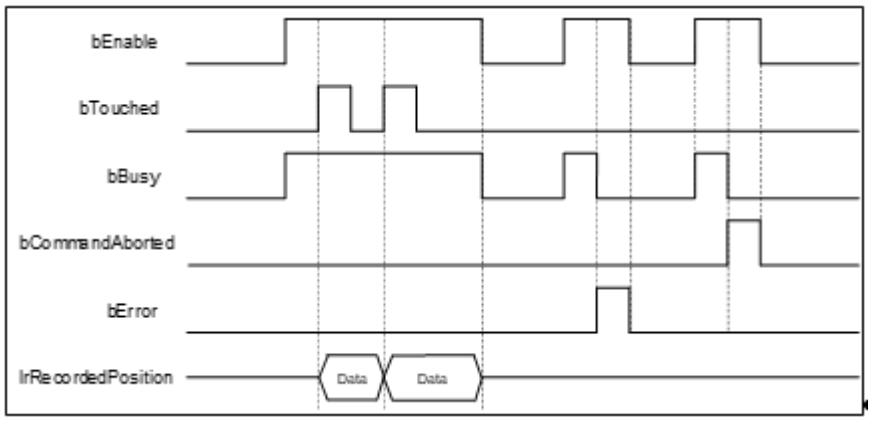
Name	Function	Data Type	Output range (Default Value)
bTouched	True when the trigger signal is True and axis position recording is completed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bCommand Aborted	True when the instruction is aborted by another instruction.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	When a command error occurs, record the error code. For the detailed description of the error code, refer to the appendix of the manual.	DMC_ERROR*1	DMC_ERROR(DMC_NO_ERROR)
lrRecorded Position	Contains the position when a trigger occurs.	LREAL	Positive, negative or 0 (0)

***Note:** DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for changing to TRUE	Timing for changing to FALSE
bTouched	• When the trigger signal is True and axis position recording is completed.	• When *bEnable* turns to False. • When *bCommandAborted* turns to True. • When *bError* turns to True. • After a period when *bEnable* turns to True.
bBusy	• True when the instruction execution starts.	• When *bEnable* turns to False. • When *bCommandAborted* turns to True. • When *bError* turns to True.
bCommand Aborted	• When the instruction is aborted by another instruction.	• When *bEnable* turns to False.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bEnable* changes from True to False.
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_SM3*1	AXIS_REF_SM3 (Must be specified)	-
TriggerInput	Trigger signal	DMC_TRIGGER_REF *2	TRIGGER_REF (-1)	When *bEnable* turns to True and *bBusy* is False.

***Note:**
1. AXIS_REF_SM3 (FB): The interface is built in every function block and works as the starting program of the function block.
2. DMC_TRIGGER_REF: Structure (STRUCT)

Name	Function	Data Type	Setting Value (Default)
iTriggerNumber	The trigger channel	INT	0: Touch Probe 1 1: Touch Probe 2 (-1)
eFastLatching	The trigger signal	DMC_LATCH_MODE	0: DRIVE_MODE 1: CONTRL_MODE (DRIVE_MODE)
bInput	The trigger signal source when the controller is triggered	BOOL	The trigger signal source
bActive	Activate or deactivate the trigger signal	BOOL	True: Activate the trigger signal (False)
iCtrlTriggerSource	The recorded position source	INT	0: Set Position 1: Act Position (0)
iCtrlTriggerNumber	The mode of triggering the controller	INT	0: Rising edge data capture 1: Falling edge data capture 2: Rising/falling edge data capture (-1)

- **Function**

  - When the trigger signal (*eFastLatching*) is DRIVE_MODE, then the position is provided by the servo and *iCtrlTriggerSource* is meaningless. *iCtrlTriggerSource* is available only for CONTRL_MODE.
  - When DMC_TouchProbeCyclically is used, the Touch Probe Function (60B8h) cannot be configured to PDO. If users configure it to PDO, the function block will report an error when being run.
  - DMC_TouchProbeCyclically cannot be used with MC_TouchProbe together. If MC_TouchProbe is already run, an error will occur when DMC_TouchProbeCyclically is run. DMC_TouchProbeCyclically will also report an error if MC_TouchProbe is run during DMC_TouchProbeCyclically execution.
  - If the trigger signal is DRIVE_MODE, the position stored in the servo is read directly and then the *iCtrlTriggerSource* of *TriggerInput* is an invalid parameter.
  - *bInput* of *TriggerInput* is the trigger signal source under CONTRL_MODE, and it is an invalid parameter under DRIVE_MODE mode.

- When DRIVE_MODE is used, TouchProbe1 and TouchProbe2 can be started respectively in two independent DMC_TouchProbeCyclically instructions.

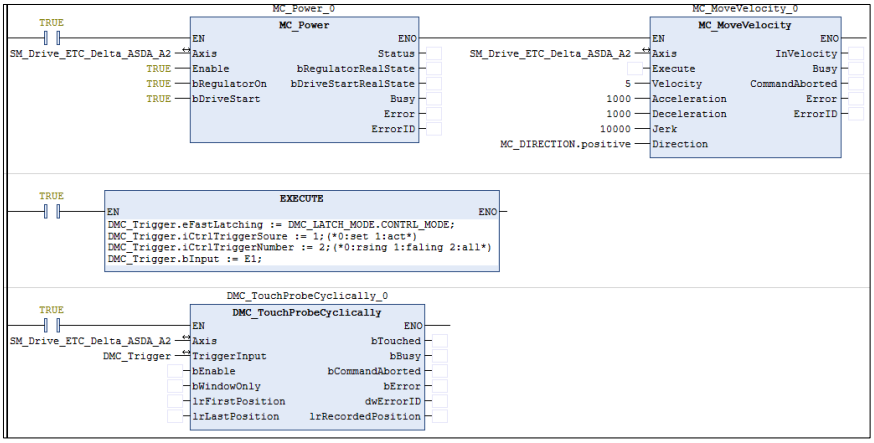
- **Troubleshooting**

  - When an error occurs during the execution of the instruction or the axis enters “Errorstop” state, *bError* changes to True and the axis stops running. To confirm current error state, see the error code in ErrorID.

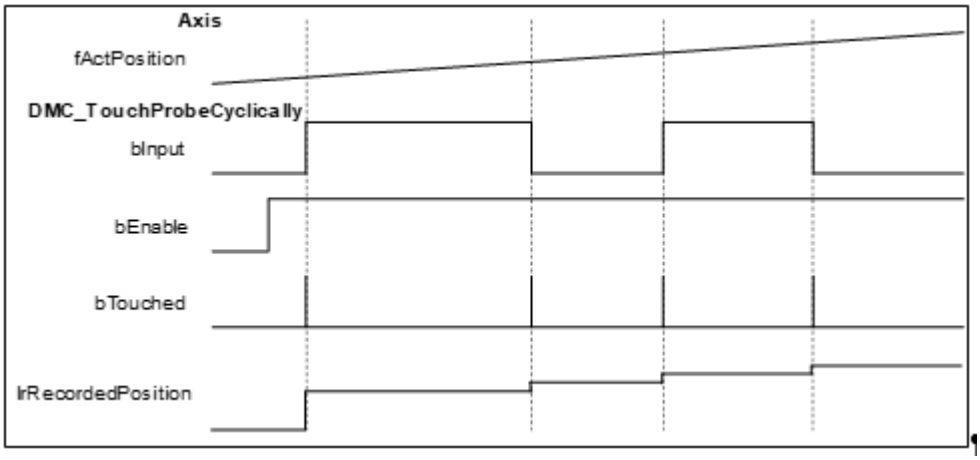
- **Programming Example**

  - Programming Example1:

    - This example illustrates how to use DMC_TouchProbeCyclically in CONTRL_MODE mode.



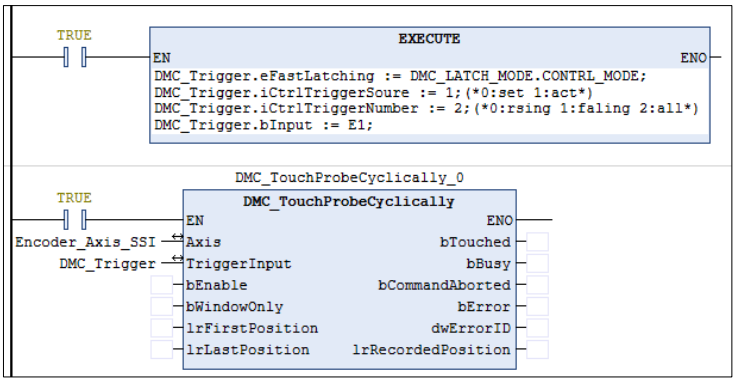
    - Timing Diagram



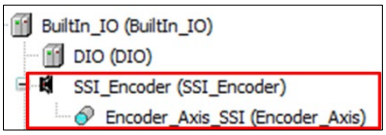
1. DMC_TouchProbeCyclically specifies CONTRL_MODE and uses the rising edge or falling edge as the trigger signal with the actual position of the axis (*fActPosition*) as the reference position.
2. When the signal source *bInput* of *TriggerInput* is triggered in CONTRL_MODE mode and the state of *bInput* changes, the function block will record the actual position of the current axis, and *bTouched* will remain True for one period.

  - Programming Example 2:

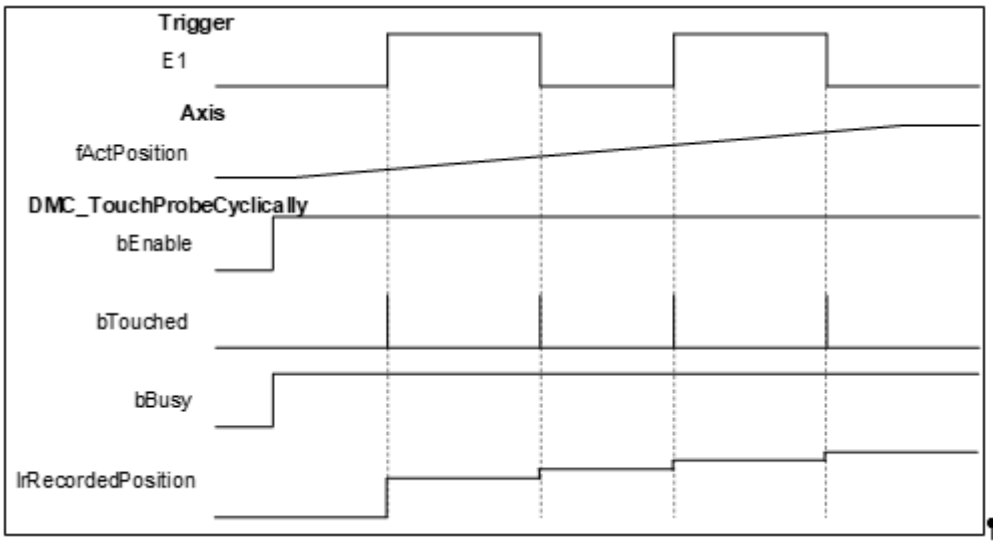
    - The example illustrates how to use DMC_TouchProbeCyclically by using SSI Encoder as the signal source in CONTRL_MODE.



- Device tree setting



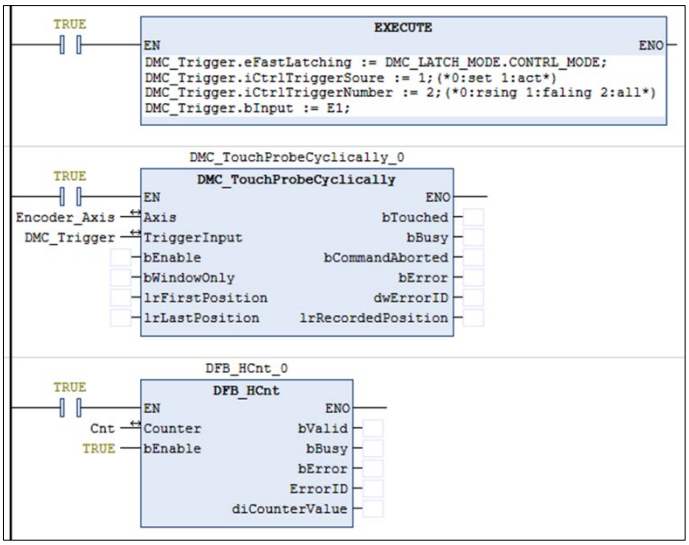
- **Timing Diagram**



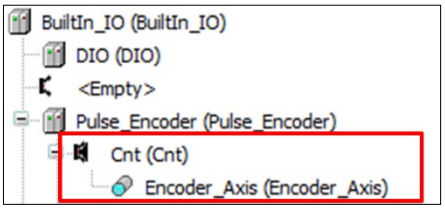
1. Select SSI Encoder as the signal source for DMC_TouchProbeCyclically.
2. Add an SSI Encoder to the device tree and then connect the SSI Encoder to the AX-308 module. For wiring, refer to section 2.2.4 CPU Module Input and Output Terminals in **AX-3 Series Operation Manual**.
3. When the *bInput* of *TriggerInput* is triggered, DMC_TouchProbeCyclically will record the position of the current SSI Encoder.

■ Programming Example 3:

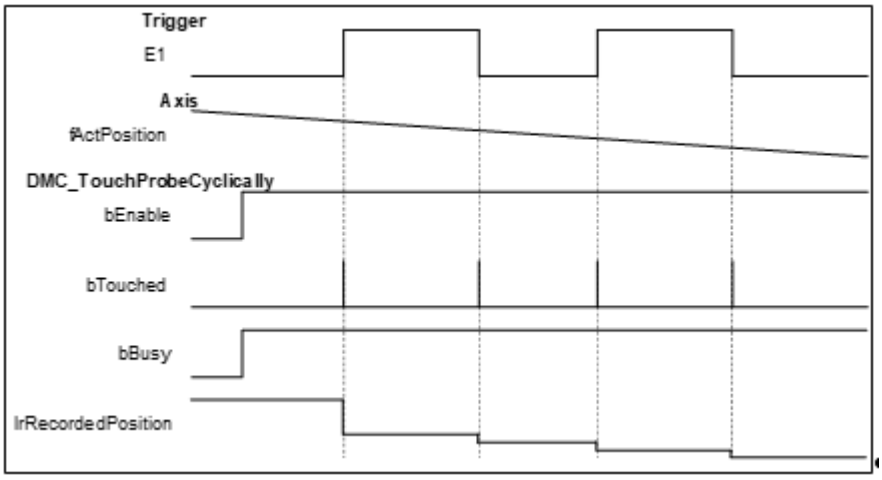
- The example illustrates how to use DMC_TouchProbeCyclically with Pulse Encoder as the signal source in CONTRL_MODE mode.



- Device tree setting



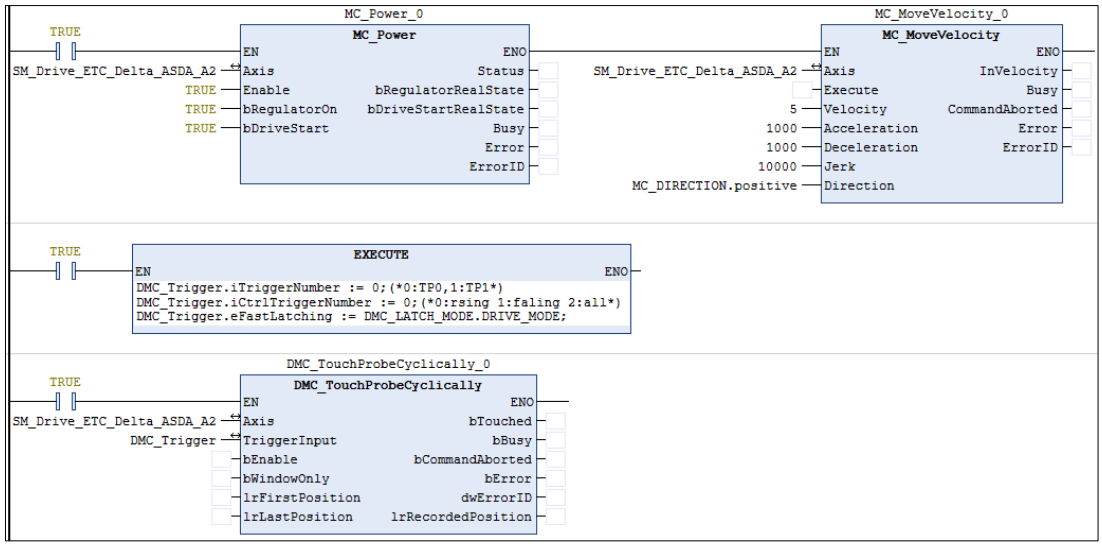
- Timing Diagram



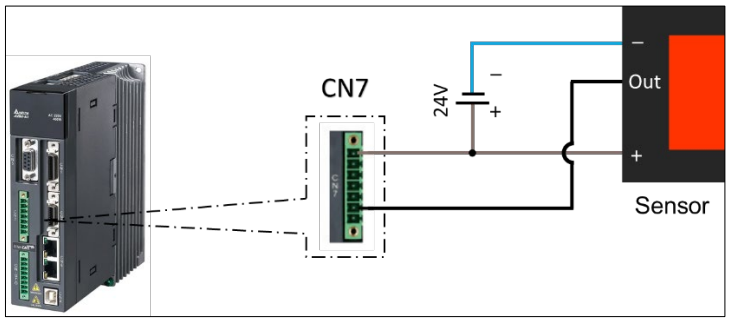
1. Select Pulse Encoder as the signal source for DMC_TouchProbeCyclically.
2. Add a Count to the device tree (here is Count 1), and then add DFB_HCnt to the program to read the value of the Pulse Encoder. Finally connect the Pulse Encoder. For wiring, refer to Section 2.2.4 CPU Module Input and Output Terminals in **AX-3 Series Operation Manual**.
3. When the *bInput* of *TriggerInput* is triggered, DMC_TouchProbeCyclically will record the position of the current Pulse Encoder.

- Programming Example 4:

  - The example illustrates how to use DMC_TouchProbeCyclically in DRIVE_MODE.

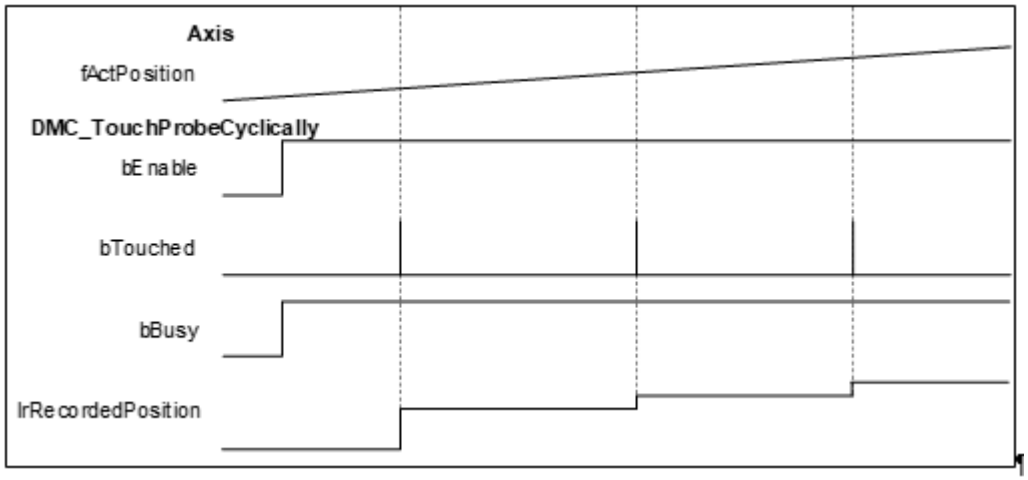


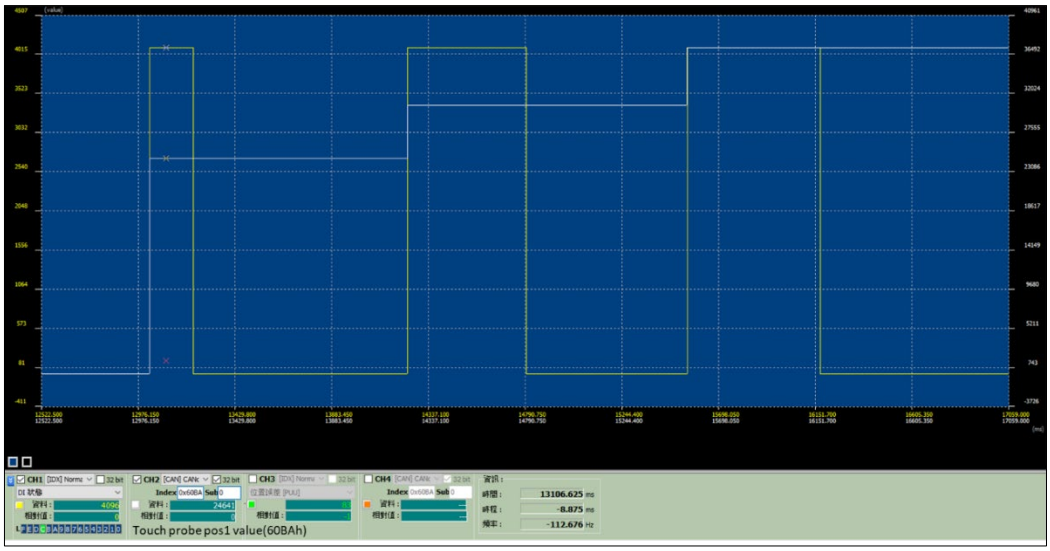
◆ Wiring figure



1. The trigger signal comes from DI13 of extension DIs of the servo drive's CN7. refer to the wiring figure above for configuration.
2. In this example, the TouchProbe 1 trigger is taken as an example and so the photoelectric switch is connected to DI13. If the TouchProbe 2 trigger is selected, the photoelectric switch should be connected to DI14.

■ **Timing Diagram**



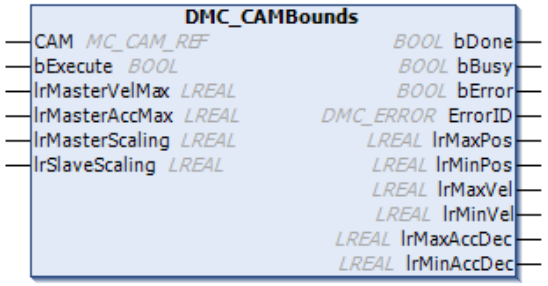


1. DMC_TouchProbeCyclically specifies DRIVE_MODE with TouchProbe 1 which is triggered by the rising edge signal.
2. When the switch trigger occurs, the drive will record the current position, send it back to the controller and record it in the function block IrRecordedPosition, and *bTouched* will remain True for one period.
3. In DRIVE_MODE, the drive will record the current position in real time and thus the recorded position will be earlier than the actual feedback position of the controller.

2.3.2.23 DMC_CAMBounds

- **Supported Devices**: AX-308E, AX-364E

DMC_CAMBounds using the cam table and the expected maximum speed and acceleration of the master axis to obtain the maximum and minimum values of the estimated position, velocity and acceleration of the slave axis.

FB/FC	Instruction	Graphic Expression
FB	DMC_CAMBounds	

ST Language

```
DMC_CAMBounds_instance(
CAM: = ,
bExecute: = ,
lrMasterVelMax: = ,
lrMasterAccMax: = ,
lrMasterScaling: = ,
lrSlaveScaling: = ,
bDone=> ,
bBusy=> ,
bError=> ,
ErrorID=> ,
lrMaxPos=> ,
lrMinPos=> ,
lrMaxVel=> ,
lrMinVel=> ,
lrMaxAccDec=> ,
lrMinAccDec=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
lrMasterVelMax	Expected maximum velocity of the master axis (User unit/sec)	LREAL	Positive or 0 (1)	When *bExecute* and *bBusy* are True.
lrMasterAccMax	Expected maximum acceleration of the master	LREAL	Positive or 0 (1)	When *bExecute* turns to True and *bBusy* is False.

	axis (User unit/sec^2)			
lrMasterScaling	The scaling factor of the master axis	LREAL	Positive (1)	When *bExecute* turns to True and *bBusy* is False.
lrSlaveScaling	The scaling factor of the slave axis	LREAL	Positive (1)	When *bExecute* turns to True and *bBusy* is False.

- **Outputs**

Name	Function	Data Type	Setting Value (Default Value)
bDone	True when the calculation is complete.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bError	True when an error occurs in execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
lrMaxPos	The maximum position of the slave axis (User unit)	LREAL	Positive, negative, or 0 (0)
lrMinPos	The minimum position of the slave axis (User unit)	LREAL	Positive, negative, or 0 (0)
lrMaxVel	The maximum velocity of the slave axis (User units/sec)	LREAL	Positive, negative, or 0 (0)
lrMinVel	The minimum velocity of the slave axis (User units/sec)	LREAL	Positive, negative, or 0 (0)
lrMaxAccDec	The maximum acceleration and deceleration of the slave axis (User units/sec^2)	LREAL	Positive, negative, or 0 (0)
lrMinAccDec	The minimum acceleration and deceleration of the slave axis (User units/sec^2)	LREAL	Positive, negative, or 0 (0)

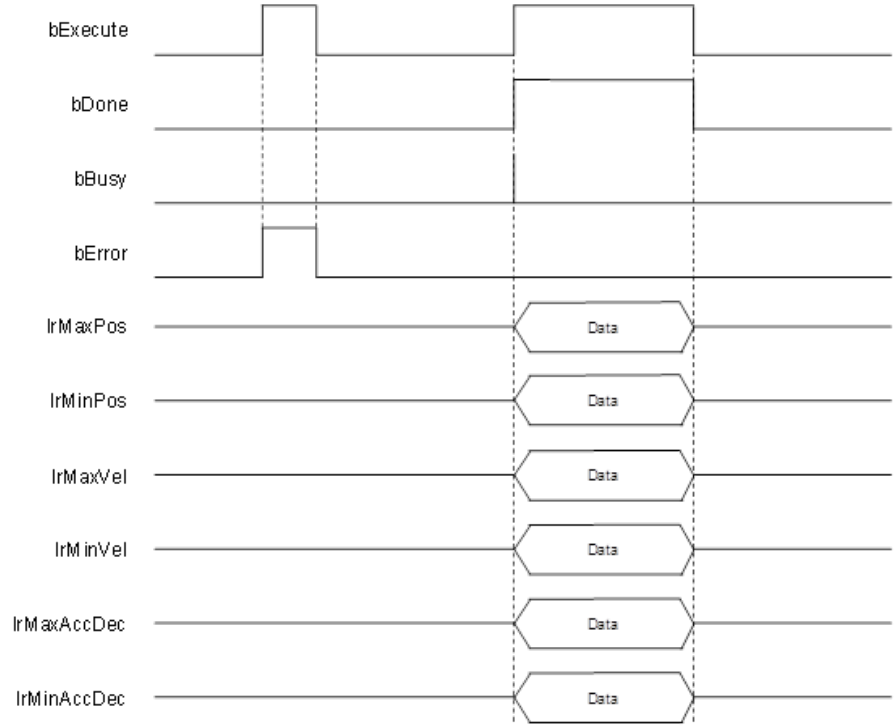
***Note:** DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the trigger signal is True and axis position recording is complete.	• When *bExecute* turns to False. • When *bExecute* is False but *bDone* turns to True, *bDone* will remain True for one cycle and then change to False.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input invalid values of the instruction.	• When *bExecute* turns to False.
ErrorID		
dMaxPos	• Update values after calculation completes.	• The value goes to zero when *bExecute* turns to False.
dMinPos		

Name	Timing for shifting to True	Timing for shifting to False
dMaxVel		
dMinVel		
dMaxAccDec		
dMinAccDec		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	Specify cam table	MC_CAM_REF*	MC_CAM_REF	When *bExecute* is True.

***Note**: MC_CAM_REF (FB): Basic cam.

- **Function**

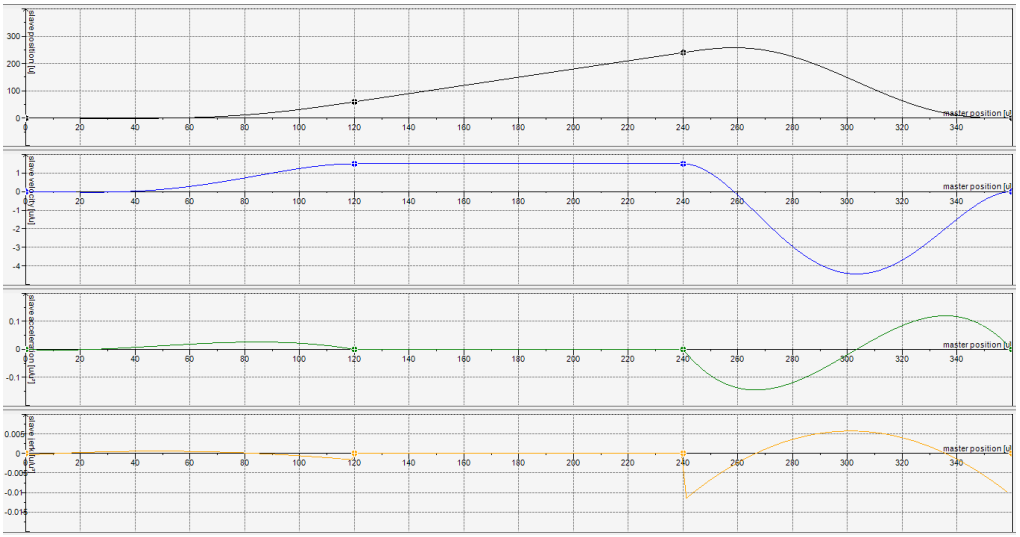
  - DMC_CAMBounds is used to check whether the user-defined cam table curve is correct. Use the maximum velocity, acceleration and decelaration limit of the master axis to calculate the maximum (minimum) position, deceleration, and acceleration of the slave axis.
  - This function block supports the following two cam formats:
    - XYVA
    - Two dimensional point array

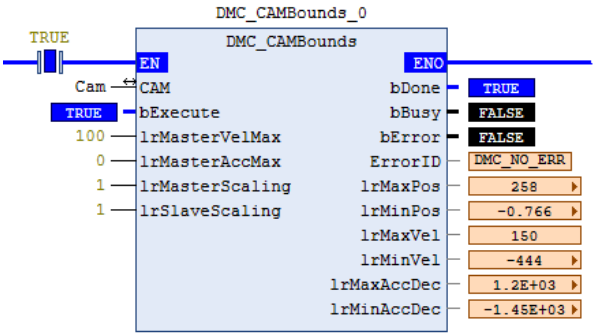
- **Troubleshooting**

  - When an error occurs during the execution of the instruction, *bError* changes to True. To confirm current error state, see the error code in ErrorID.

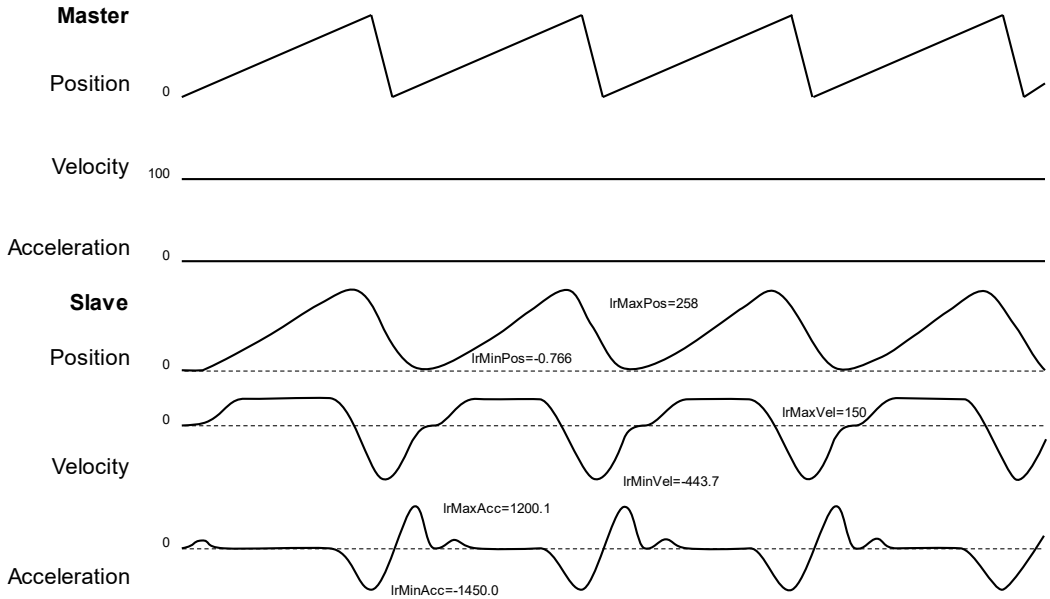
- **Example**

- The example shows the use of DMC_CAMBounds.
- Cam table:





- Timing Diagram

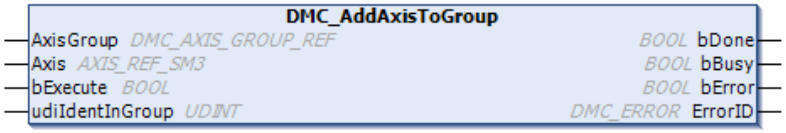


From the oscillogram, you can see that the maximum and minimum positions, velocity, and acceleration of the slave axis are consistent with the output of the DMC_CAMBounds.

2.3.2.24 DMC_AddAxisToGroup

- **Supported Devices**: AX-308E, AX-364E

DMC_AddAxisToGroup is used to add a single axis to the axis group.

FB/FC	Instruction	Graphic Expression
FB	DMC_AddAxisToGroup	

ST Language

```
DMC_AddAxisToGroup_instance(
AxisGroup: = ,
Axis: = ,
udiIdentInGroup: = ,
bExecute: = ,
bDone=> ,
bBusy=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction will be run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
udiIdentInGroup	Specifies to add the single axis to which axis of the axis group.	UDINT	1 ~ 6 (1)	When *bExecute* isTrue, the parameters of udiIdentInGroup are updated.

- **Outputs**

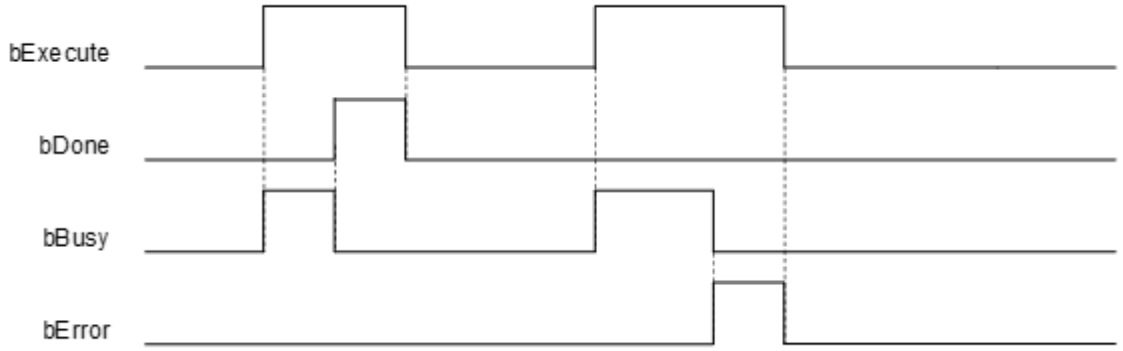
Name	Function	Data Type	Setting Value (Default Value)
bDone	True when the single axis is added.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs in the execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note:** DMC_ERROR: Enumeration (Enum)

■ **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the single axis is added.	• When *bExecute* turns to False. • When *bError* turns to True.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values of the instruction.	• When *bEexcute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



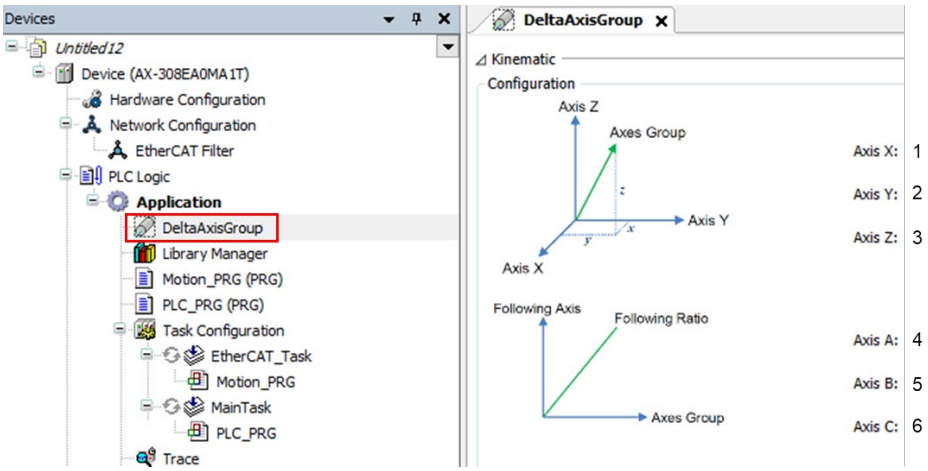
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*1	DMC_AXIS_GROUP_REF	When bExecute turns to True and *bBusy* is False
Axis	Specify the axis.	AXIS_REF_SM3*2	AXIS_REF_SM3	When bExecute turns to True and *bBusy* is False

***Note**:

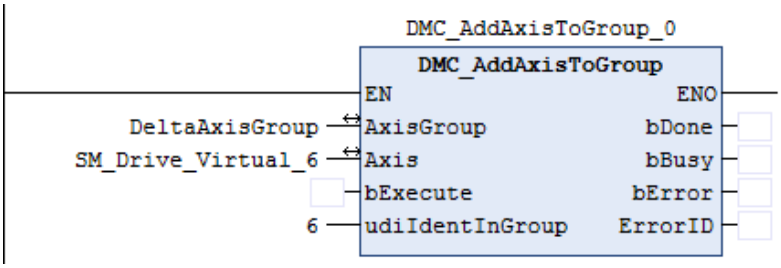
1. DMC_AXIS_GROUP_REF (FB): All axis group function blocks for an axis group contain this variable, which works as the starting program for function blocks.
2. AXIS_REF_SM3 (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**
  - This function is available for DL_MotionControl V1.2.0.0 or later.
  - Adds the specified axis to the axis group.
  - If an axis already exists on the specified udiIdentInGroup, it will be directly overwritten.
  - The axis group state must be Disabled to run this function block.
  - The udiIdentInGroup settings for DIADesigner-AX axis group is as follows:



- **Troubleshooting**
  - If an error occurs during the execution of the instruction, *bError* will turn to True and the axis motion will stop. To confirm current error state, see the error code in ErrorID.
  - For the error codes and corresponding troubleshooting methods, refer to the **Appendix** of this manual.

- **Example**
  - This example shows how to use DMC_AddAxisToGroup to add a single axis at a specified position within an axis group.

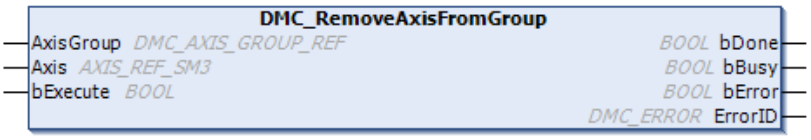


- Enter the name of the single axis that you want to add, and then enter 6 in udiIdentInGroup.
- When DMC_RemoveAxisFromGroup.bExecute is true, the SM_Drive_Virtual_6 will be configured in the position of axis group 6.
- When DMC_RemoveAxisFromGroup.bDone is true, it means that a single axis has been configured at the specified position in the axis group.

2.3.2.25 DMC_RemoveAxisFromGroup

- **Supported Devices**: AX-308E, AX-364E

The DMC_RemoveAxisFromGroup removes a single axis from an axis group.

FB/FC	Instruction	Graphic Expression
FB	DMC_RemoveAxisFromGroup	

ST Language

```
DMC_RemoveAxisFromGroup_instance(
AxisGroup: = ,
Axis: = ,
bExecute: = ,
bDone=> ,
bBusy=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction will be run when *bExecute* changes from False to True.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Setting Value (Default Value)
bDone	True when the single axis is removed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs in the execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

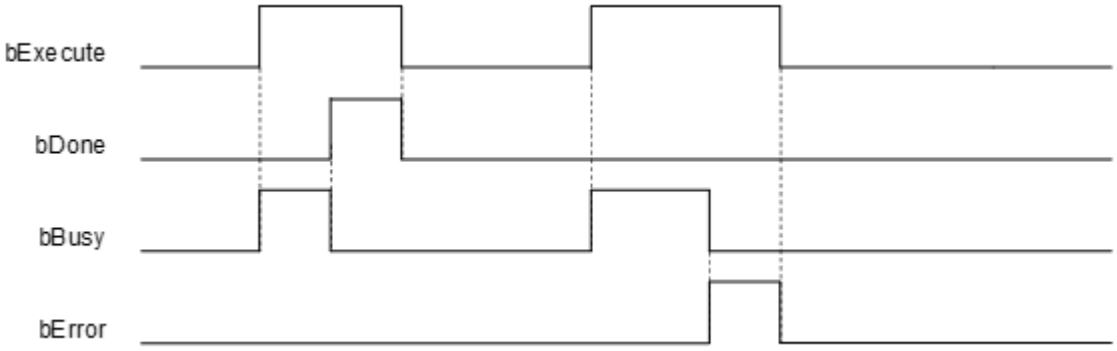
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False

bDone	• When the single axis is removed.	• When *bExecute* turns to False. • When *bError* turns to True.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values of the instruction.	• When *bEexcute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*1	DMC_AXIS_GROUP_REF	When *bExecute* turns to True and *bBusy* is False
Axis	Specify the axis.	AXIS_REF_SM3*2	AXIS_REF_SM3	When *bExecute* turns to True and *bBusy* is False

***Note**:

1. DMC_AXIS_GROUP_REF (FB): All axis group function blocks for an axis group contain this variable, which works as the starting program for function blocks.
2. AXIS_REF_SM3 (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

  - Removes the specified axis from the axis group.
  - The axis group state must be Disabled to run this function block.

- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *bError* will turn to True and the axis motion will stop. To confirm current error state, see the error code in ErrorID.
  - For the error codes and corresponding troubleshooting methods, refer to the **Appendix** of this manual.

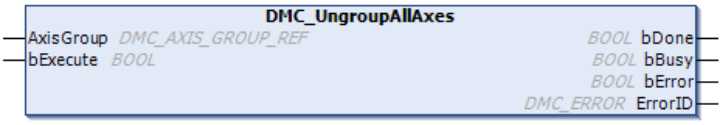
- **Example**
  - This example shows how to use DMC_RemoveAxisFromGroup to remove a single axis from an axis group.

- Enter the name of the single axis that you want to remove, and then run DMC_RemoveAxisFromGroup.bExecute. When *bDone* turns to True, the single axis has been removed.

2.3.2.26 DMC_UngroupAllAxes

- **Supported Devices**: AX-308E, AX-364E

DMC_UngroupAllAxes removes all axes in the axis group.

FB/FC	Instruction	Graphic Expression
FB	DMC_UngroupAllAxes	

ST Language

```
DMC_UngroupAllAxes_instance(
AxisGroup: = ,
bExecute: = ,
bDone=> ,
bBusy=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction will be run when *bExecute* changes from False to True.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Setting Value (Default Value)
bDone	True when the single axis is removed.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs in the execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

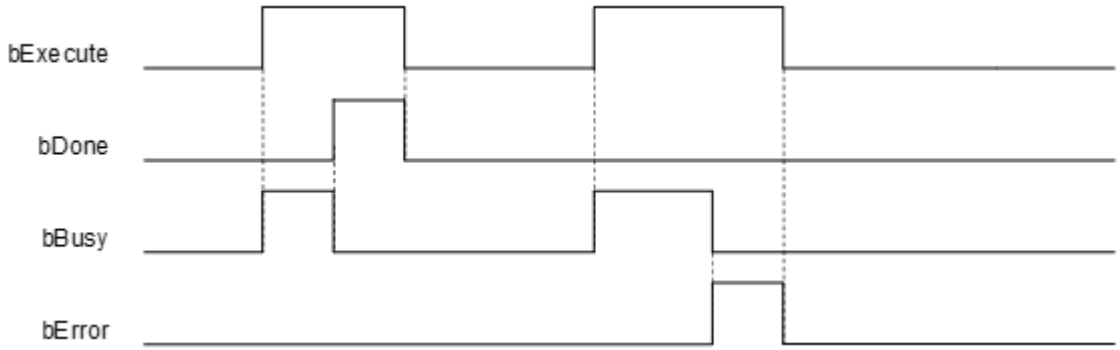
***Note:** DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the single axis is removed.	• When *bExecute* turns to False. • When *bError* turns to True.
bBusy	• When *bExecute* turns to True.	• When *bDone* turns to True. • When *bError* turns to True.

bError ErrorID	• When an error occurs in the execution conditions or input values of the instruction.	• When *bEexcute* turns to False. (Error Code is cleared)

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*1	DMC_AXIS_GROUP_REF	When *bExecute* turns to True and *bBusy* is False

***Note**: DMC_AXIS_GROUP_REF (FB): All axis group function blocks for an axis group contain this variable, which works as the starting program for function blocks.

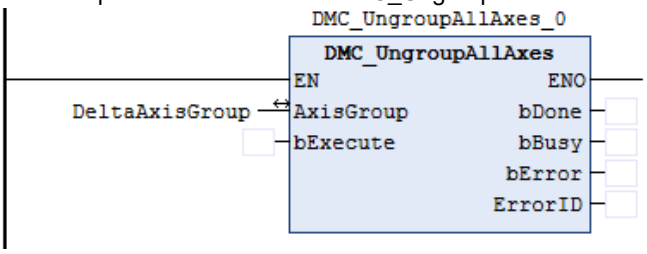
- **Function**

  - Remove all axes in the axis group.
  - The axis group state must be Disabled to run this function block.

- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *bError* will turn to True and the axis motion will stop. To confirm current error state, see the error code in ErrorID.
  - For the error codes and corresponding troubleshooting methods, refer to the **Appendix** of this manual.

- **Example**
  - This example shows how to use DMC_UngroupAllAxes to remove all single axes from an axis group.

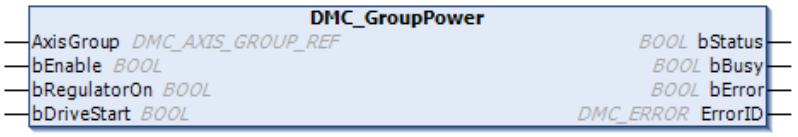


  - Enter the name of the single axis that you want to remove, and then run DMC_UngroupAllAxes.bExecute. When *bDone* turns to True, all single axes in the axis group have been removed.

2.3.2.27 DMC_GroupPower

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupPower controls the enablement, shutdown and immediate stop of all axes in the axis group.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupPower	

ST Language

```
DMC_GroupPower_instance(
AxisGroup: = ,
bEnable: = ,
bRegulatorOn: = ,
bDriveStart: = ,
bStatus=> ,
bBusy=> ,
bError=> ,
ErrorID=> , );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction will be run when *bEnable* changes from False to True.	BOOL	True/False (False)	-
bRegulatorOn	Power ON	BOOL	True/False (False)	Only when Enable=True
bDriveStart	Disable the immediate stop mechanism.	BOOL	True/False (False)	Only when Enable=True

- **Outputs**

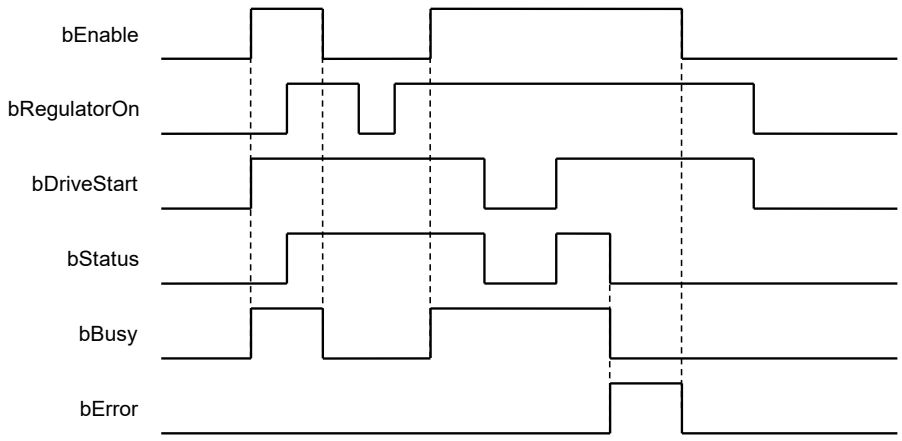
Name	Function	Data Type	Setting Value (Default Value)
bStatus	True when all axes in the axis group are enabled.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs in the execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note:** DMC_ERROR: Enumeration (Enum)

  - **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bStatus	• When *bEnable* turns to True and all axes in the axis group are enabled and enter a movable state.	• When *bEnable* turns to False. • When *bError* turns to True.
bBusy	• When *bEnable* turns to True.	• When *bEnable* turns to False. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values of the instruction.	• When *bEnable* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*1	DMC_AXIS_GROUP_REF	When *bEnable* turns to True and *bBusy* is False

***Note**: DMC_AXIS_GROUP_REF (FB): All axis group function blocks for an axis group contain this variable, which works as the starting program for function blocks.

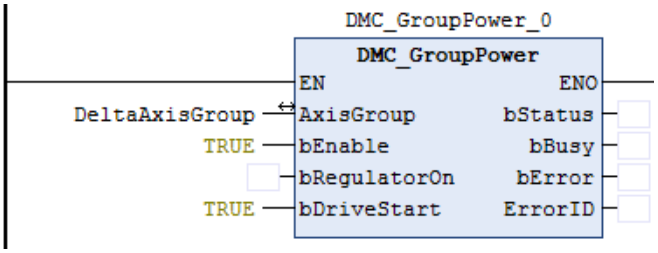
- **Function**

  - This function is available for DL_MotionControl V1.2.0.0 or later.
  - DMC_GroupPower enables on all single axes in the axis group without affecting the axis group status. Originally, the axis group status was GroupDisabled, but it remained GroupDisabled after using DMC_GroupPower.

- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *bError* will turn to True and the axis motion will stop. To confirm current error state, see the error code in ErrorID.
  - For the error codes and corresponding troubleshooting methods, refer to the **Appendix** of this manual.

- **Example**
  - This example shows how to use DMC_GroupPower to enable all single axes in an axis group.



- Enter the name of the axis group that you want to enable, and then run DMC_GroupPower.bRegulatorOn. When *bStatus* turns to True, all single axes in the axis group have been enabled.

2.3.2.28 DMC_GroupSetOverride

- **Supported Devices**: AX-308E, AX-364E

DMC_GroupSetOverride changes the velocity of the axis group movement by override control factor.

FB/FC	Instruction	Graphic Expression
FB	DMC_GroupSetOverride	

ST Language

```
DMC_GroupSetOverride_instance(
AxisGroup: = ,
bEnable: = ,
lrVelFactor: = ,
lrAccFactor: = ,
lrJerkFactor: = ,
bEnabled=> ,
bBusy=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction will be run when *bEnable* changes from False to True.	BOOL	True/False (False)	-
lrVelFactor	Override control velocity factor	LREAL	0.0 ~ 5.0 (1.0)	When *bEnable* is True, *VelFactor* will be updated.
lrAccFactor	Override control acceleration and deceleration factor (Reserved)	LREAL	0.0 ~ 1.0 (1.0)	When *bEnable* is True, *VelFactor* will be updated.
lrJerkFactor	Override control jerk factor (Reserved)	LREAL	0.0 ~ 1.0 (1.0)	When *bEnable* is True, *VelFactor* will be updated.

- **Outputs**

Name	Function	Data Type	Setting Value (Default Value)
bEnabled	True when the factor is successfully set.	BOOL	True/False (False)
bBusy	True when the instruction is triggered to run.	BOOL	True/False (False)
bError	True when an error occurs in the execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs.	DMC_ERROR*	DMC_ERROR

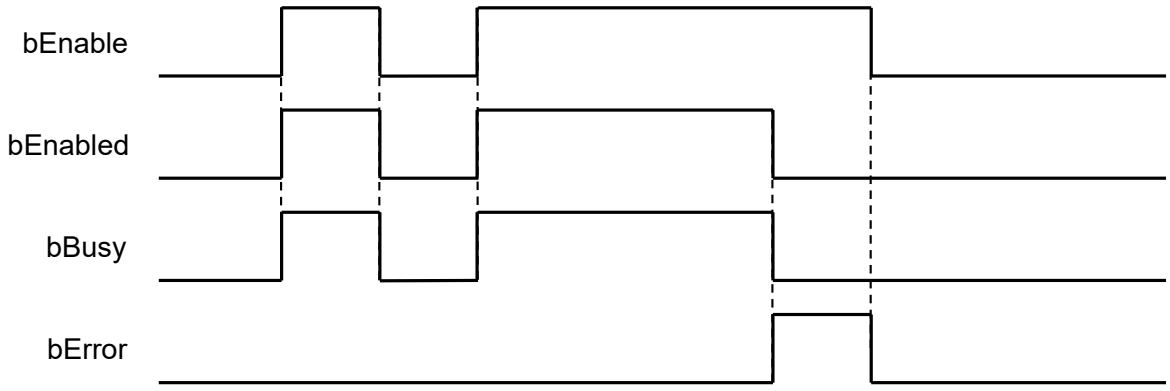
Name	Function	Data Type	Setting Value (Default Value)
	Refer to Appendix for error code descriptions.		(DMC_NO_ERROR)

***Note:** DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bEnabled	• When *bEnable* turns to True and the factor is successfully set.	• When *bEnable* turns to False. • When *bError* turns to True.
bBusy	• When *bEnable* turns to True.	• When *bEnable* turns to False. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values of the instruction.	• When *bEnable* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*1	DMC_AXIS_GROUP_REF	When *bEnable* turns to True and *bBusy* is False

***Note**: DMC_AXIS_GROUP_REF (FB): All axis group function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

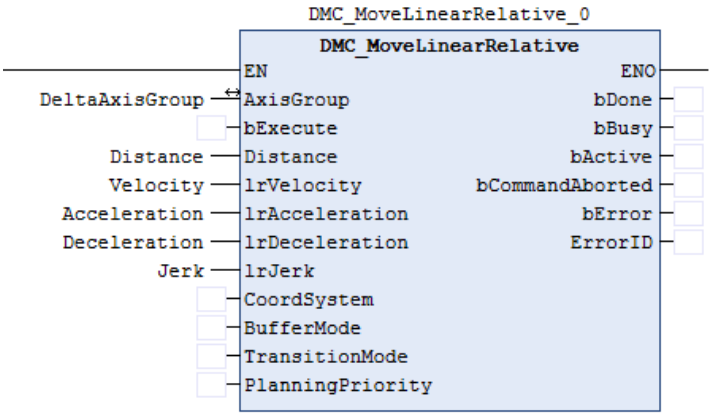
  - This function is available for DL_MotionControl V1.2.0.0 or later.
  - When *bEnable* is True, the override control factor will be continuously updated; When *bEnable* is False, the override control factor remains at the last updated value.
  - When the IrVelFactor is 0, the current axis group movement will stop, but the axis group status will not change. After switching to a non-zero value, the movement will be continued.
  - The factor of this function block has no effect on the DMC_GroupStop and the deceleration and stop of ErrorStop.
  - The acceleration and deceleration velocity of this function block is based on the current motion command of the axis group.
    - If DMC_MoveLinearAbsolote acceleration and deceleration is set to 100, the override control will perform the acceleration and deceleration changes at 100.

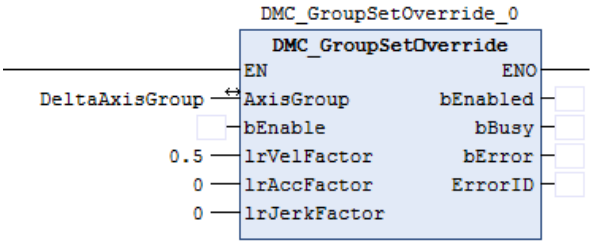
- **Troubleshooting**

- If an error occurs during the execution of the instruction, *bError* will turn to True and the axis motion will stop. To confirm current error state, see the error code in ErrorID.
- For the error codes and corresponding troubleshooting methods, refer to the **Appendix** of this manual.

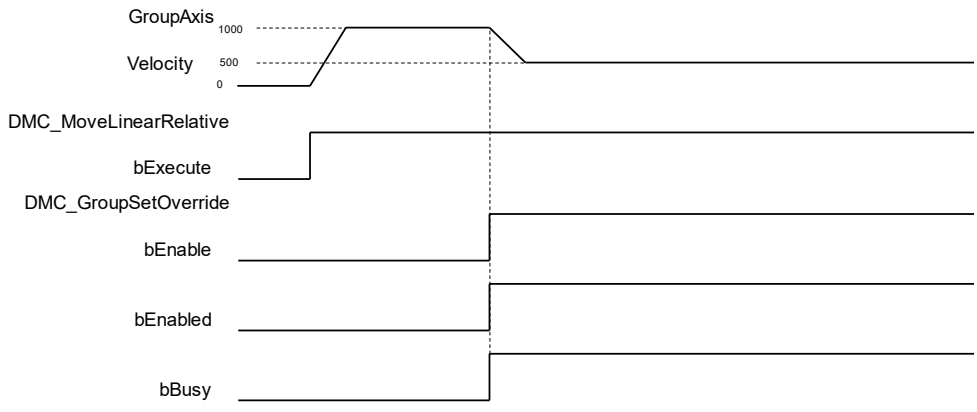
- **Example**
  - This example shows how to use DMC_GroupSetOverride to change the velocity during axis group motion.

```
DMC_MoveLinearRelative_0: DMC_MoveLinearRelative;
DMC_GroupSetOverride_0: DMC_GroupSetOverride;
Distance: ARRAY [0..5] OF LREAL := [2(10000), 4(0.0)];
Velocity: LREAL := 1000;
Acceleration: LREAL := 1000;
Deceleration: LREAL := 1000;
Jerk: LREAL := 0;
```





  - **Timing Diagram**

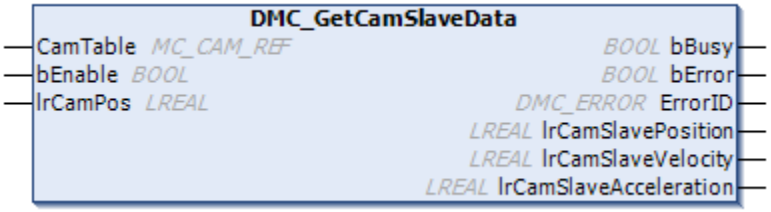


1. When DMC_MoveLinearRelative.bExecute starts, the axis group runs at the velocity of 1000 set by DMC_MoveLinearRelative.lrVelocity.
2. At this time, DMC_GroupSetOverride.bEnable is True, and then DMC_GroupSetOverride.lrVelFactor is set to 0.5. The axis group velocity is 1000 * 0.5 = 500. The axis group will continue to run at the deceleration of 500 set by DMC_MoveLinearRelative.

2.3.2.29 DMC_GetCamSlaveData

- **Supported Devices**: AX-308E, AX-364E

Input the axis position for DMC_GetCamSlaveData to get information about the slave axis of the specified cam table.

FB/FC	Instruction	Graphic Expression
FB	DMC_GetCamSlaveData	

ST Language

```
DMC_GetCamSlaveData_instance(
CamTable :=,
bEnable :=,
lrCamPos :=,
bBusy =>,
bError =>,
ErrorID =>
lrCamSlavePosition =>
lrCamSlaveVelocity =>
lrCamSlaveAcceleration =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction will be run when *bEnable* changes from False to True.	BOOL	True/False (False)	-
lrCamPos	Cam axis position (User Unit)	LREAL	Positive or 0 (0)	When *bEnable* changes from False to True.

- **Outputs**

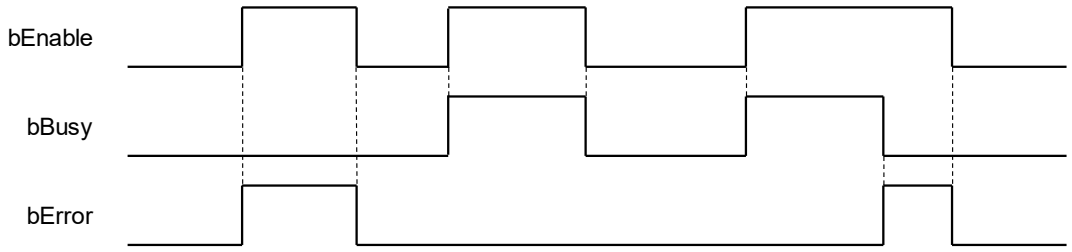
Name	Function	Data Type	Setting Value (Default Value)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bError	True when an error occurs in the execution of the instruction.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DMC_ERROR*1	DMC_ERROR (DMC_NO_ERROR)
lrCamSlavePosition	Cam slave axis position	LREAL	Positive, negative, or 0 (0)
lrCamSlaveVelocity	Cam slave axis velocity ratio	LREAL	Positive, negative, or 0 (0)
lrCamSlaveAcceleration	Cam slave axis acceleration ratio (This feature is not available when CamTable Type is one/two dimension)	LREAL	Positive, negative, or 0 (0)

***Note:** DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bBusy	• When *bEnable* turns to True.	• When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values of the instruction.	• When *bEnable* turns to False. (Error Code is cleared)
ErrorID		
lrCamSlavePosition	• Update information when *bEnable* is True.	• Will not update information when *bEnable* is False.
lrCamSlaveVelocity	• Update information when *bEnable* is True.	• Will not update information when *bEnable* is False.
lrCamSlaveAcceleration	• Update information when *bEnable* is True.	• Will not update information when *bEnable* is False.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	Specify the cam table.	MC_CAM_REF*	MC_CAM_REF	When *bEnable* turns to True

***Note**: MC_CAM_REF (FB): User-defined camtable parameters.

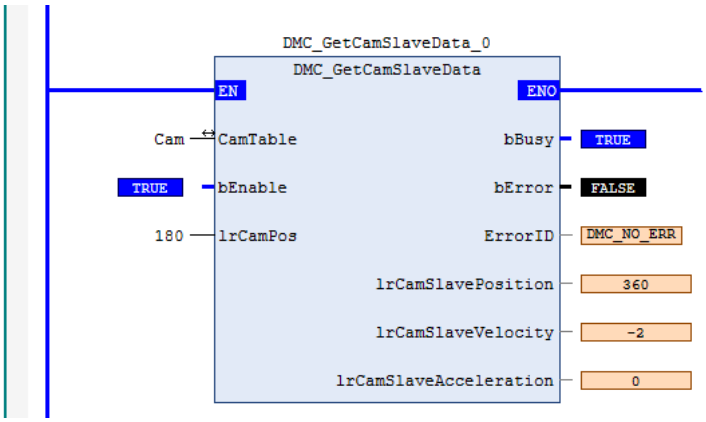
- **Function**

  - This function is available for DL_MotionControl V1.2.3.0 or later.
  - By entering the master slave position with this function block, you can get the slave axis position (lrCamSlavePosition), the slave axis velocity ratio (lrCamSlaveVelocity), and the slave axis acceleration ratio (lrCamSlaveAcceleration) of the specified cam table.
  - When the type of cam table is polynomial (XYVA Type), you can get complete information. If the tye is one-dimensional table of slave positions or two-dimensional table of related master/slave positions, then the function block does not provide information about the acceleration ratio of the slave axis (lrCamSlaveAcceleration).
  - When the cam table slave axis starts and ends at the same position, the velocity and acceleration will both be NaN.

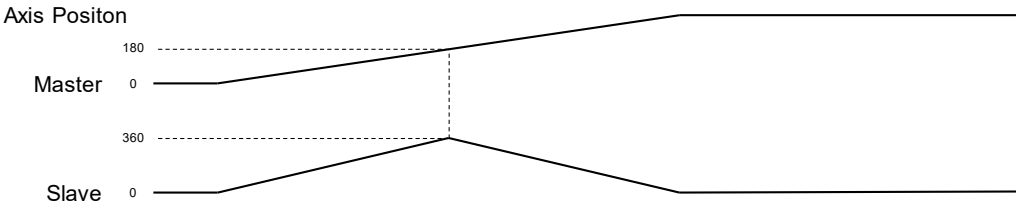
- **Troubleshooting**

  - If an error occurs during the execution of the instruction, *bError* will turn to True. To confirm current error state, see the error code in ErrorID.
  - For the error codes and corresponding troubleshooting methods, refer to the **Appendix** of this manual.

- **Example**
  - This example shows how to use DMC_GetCamSlaveData function blocks to get the cam table master axis position and the corresponding position of slave axis.



    - Timing Diagram



    - The figure above shows the changes of master and slave axis when the cam table runs for a cycle.
    - Input 180 for DMC_GetCamSlaveData.lrCamPos (master axis position is 180), and then you can know that when the master axis runs to 180, the slave axis position will be 360.

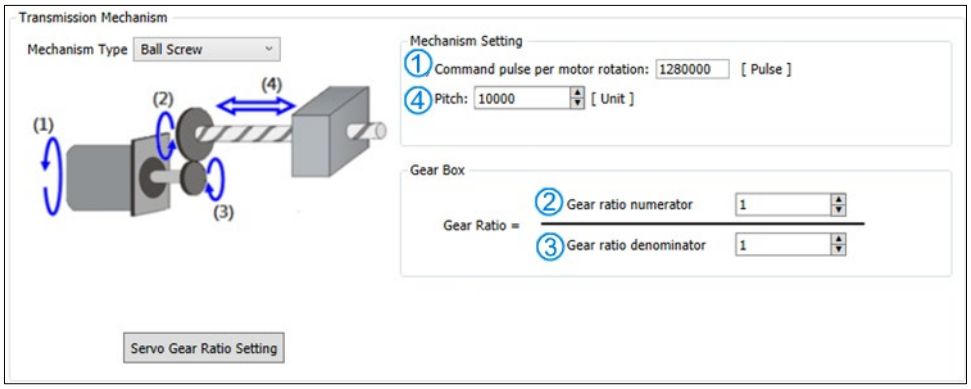
2.3.3 Positioning Axis Instructions

The function blocks in this section come from the function library DL_MotionControlLight. The drive handles the main motion curve planning and calculation of function blocks. So select the positioning axis when setting the axis. Refer to section 7.4 in *AX-3 Series Operation Manual* for related settings on a positioning axis.

- Positioning axis speed range introduction

  The positioning axis speed range is related to the speed range in the EtherCAT servo drive. Take the A2-E servo as an example. The servo drive speed unit is rpm, and the acceleration and deceleration time unit is ms.

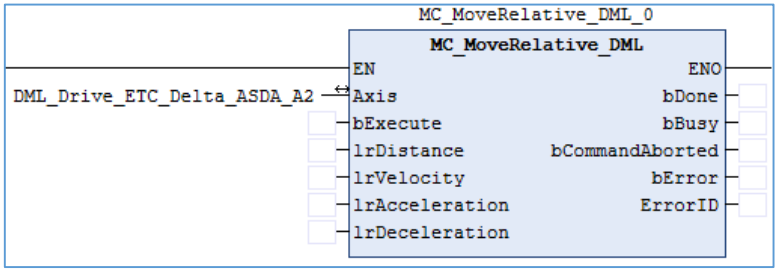
  - Gear ratio of the DIA-AX software end



  - Take the above figure as a calculation example

Factor =（① / ②）*（③ / ④）

The maximum speed and maximum acceleration and deceleration range of the function block are calculated as follows:



Max. lrVelocity = allowable rated rpm of servo drive / 60 * servo motor one-round resolution / Factor

Max. lrAcceleration = speed time allowable range / fastest acceleration time for the servo drive

Max. lrDeceleration = speed time allowable range / fastest deceleration time for the servo drive

- Example:

  If

  - A2-E servo drive allowable rated speed is 3,000 rpm
  - A2-E servo motor one-round resolution is 1,280,000 (P1-44 = 1 · P1-45 = 1)
  - The fastest acceleration and deceleration time is 1 ms for EtherCAT OD 0x6083 and 0x6084
  - DIA-AX Factor = 128, then

  Max. lrVelocity = 3000/60*128000/128 = 500000 unit/s

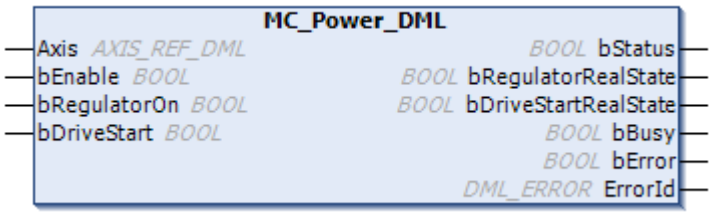
  Max. lrAcceleration = Max. lrDeceleration = 500000 / (1/1000) = 500000000 unit/s2

*Note: When the conversion unit exceeds the pulse unit, it will run at the maximum allowable pulse unit of the drive.

2.3.3.1 MC_Power_DML

- **Supported Devices**: AX-308E, AX-364E

MC_Power_DML is used to enable, disable and immediately stop the specified axis.

FB/FC	Instruction	Graphic Expression
FB	_DML	

ST Language

```
MC_Power_DML_instance(
Axis : =,
bEnable : =,
bRegulatorOn: =,
bDriveStart : =,
bStatus =>,
bRegulatorRealState =>,
bDriveStartRealState =>,
bBusy =>,
bError =>,
ErrorID =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bEnable	The instruction is run when *bEnable* turns from False to True.	BOOL	True/False (False)	-
bRegulatorOn	Power ON	BOOL	True/False (False)	Only valid when *bEnable* is True.
bDriveStart	Disable the immediate stop mechanism.	BOOL	True/False (False)	Only valid when *bEnable* is True.

- **Outputs**

Name	Function	Data Type	Output Range (Default)
bStatus	True when the specified axis can move.	BOOL	True/False (False)
bRegulatorRealState	True when the power is ON	BOOL	True/False (False)

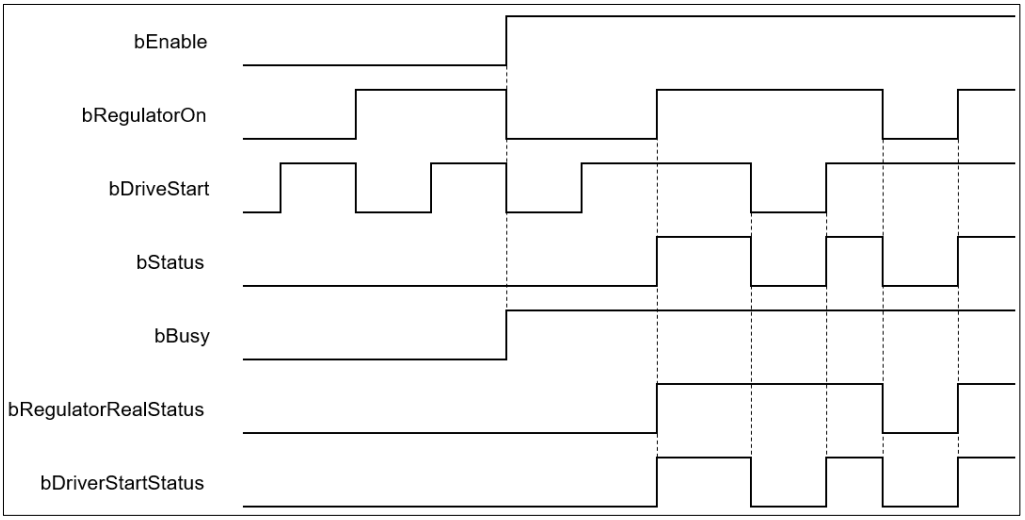
Name	Function	Data Type	Output Range (Default)
bDriveStartRealState	True when the immediate stop mechanism can be used.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bError	True when an error occurs in the execution of the instruction.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendix** for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bStatus	• When *bEnable* is True and *bRegulatorRealState* and *bDriveStartRealState* shift to True.	• When *bEnable* is True and *bRegulatorRealState* or *bDriveStartRealState* turns to False. • When *bError* turns to True.
bRegulatorRealState	• When *bEnable* and *bRegulatorRealState* are True.	• When *bEnable* is True and *bRegulatorRealState* turns to False. • When *bError* turns to True.
bDriveStartRealState	• When *bEnable* and *bRegulatorRealState*, *bDriveStartRealState* are True.	• When *bEnable* is True, and *bRegulatorRealState* or *bDriveStartRealState* turns to False. • When *bError* turns to True.
bBusy	• When *bEnable* turns to True.	• When *bEnable* turns to False. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When Error Code is cleared.
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bEnable* turns to True.

***Note:**
AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

  - *bRegulatorOn* and *bDriveStart* are effective only when *bEnable* is True.
  - When *bEnable*, *bRegulatorOn* and *bDriveStart* are all True, *bStatus* changes to True and *nAxisState* (state machine) changes to Standstill
  - When *bEnable* and *bRegulatorOn* are True and then *bDriveStart* is set to False, *nAxisState* (state machine) changes to Stopping.
  - When *bEnable* and *bDriveStart* are True and then *bRegulatorOn* is set to False, *nAxisState* (state machine) changes to Disabled.
  - When the axis state machine is under Standstill, Delta servo ASDA-xx-E Series runs MC_Stop_DML, and the bStatus of MC_Power_DML will be False.

- **Troubleshooting**

  - When an error occurs in the instruction execution or the axis enters Errorstop state, *bError* changes to True and the axis stops running. To confirm current error state, see the error code in *ErrorID*.

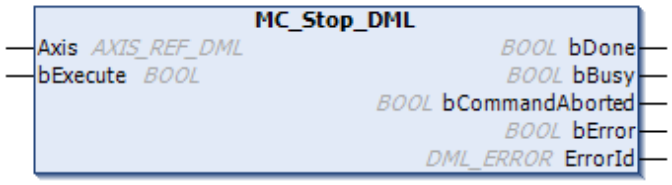
- **Programming Example**

  - For the example, refer to the programming example for MC_Power function block
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.2 MC_Stop_DML

- **Supported Devices**: AX-308E, AX-364E

MC_Stop_DML decelerates the specified axis to a stop.

FB/FC	Instruction	Graphic Expression
FB	MC_Stop_DML	
ST Language		

```
MC_Stop_DML_instance(
Axis : =,
bExecute : =,
bDone =>,
bBusy =>,
bCommandAborted=>,
bError =>,
ErrorID => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* turns from False to True.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default)
bDone	True when the velocity reaches 0.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Contains error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

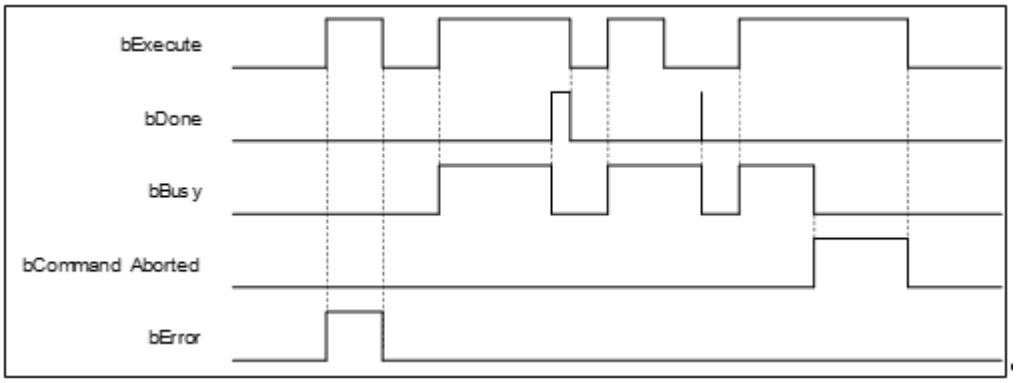
***Note:**
DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the axis decelerates to a stop or the velocity is 0.	• When *bExecute* turns from True to False. • If *bExecute* is False and bDone turns to True, bDone will be True for one period and then immediately shift to False.
bCommandAborted	• When the axis state switches to Disabled during instruction execution.	• When *bExecute* turns to False. • If *bExecute* is False and *bCommandAborted* is True, *bCommandAborted* will immediately change to False after maintaining a True state for a scan cycle.
bBusy	• When *bExecute* turns to True and the instruction is run.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution	• When *bExecute* turns to False. (Error Code is

Name	Timing for shifting to True	Timing for shifting to False
ErrorID	conditions or input values for the instruction.	cleared)

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy* is False.

*__Note__: AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

  - MC_Stop_DML can be used to stop the axis in motion and State Machine enters Stopping.
  - When MC_Power is set to False during deceleration, the motor will be in Free Run.
  - When the axis velocity is already decreased to 0 and *Done* of MC_Stop changes to True, *Execute* of MC_Stop changes to False and State Machine changes from Stopping to Standstill.
  - The deceleration rate can follow the Setting Value of Quick stop deceleration (16#6085) in the CiA402 object dictionary.

- **Troubleshooting**

  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the error code in ErrorID.

- **Programming Example**

  - For the example, refer to the programming example for MC_Stop function block
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.3 MC_Reset_DML

- **Supported Devices**: AX-308E, AX-364E

MC_Reset_DML clears axis-related errors.

FB/FC	Instruction	Graphic Expression
FB	MC_Reset_DML	

ST Language

```
MC_Reset_DML_instance(
Axis : =,
bExecute : =,
bDone =>,
bBusy =>,
bError =>,
ErrorID =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* turns from False to True.	BOOL	True/False (False)	-

- **Outputs**

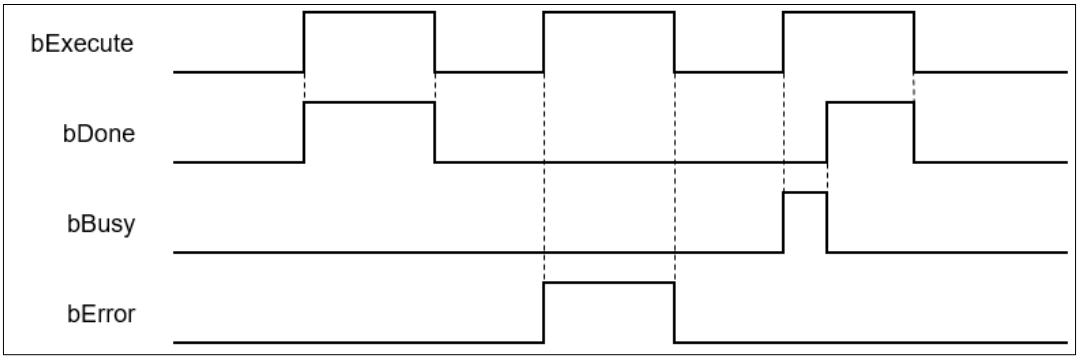
Name	Function	Data Type	Output Range (Default)
bDone	True when axis error clearing is completed and the axis enters Standstill or Disabled.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When axis error clearing is completed.	• When *bExecute* turns to False. • If *bExecute* is False and *bDone* turns to True, *bDone* will be True for one period and then immediately shift to False.
bBusy	• When *bExecute* turns to True and the instruction is run.	• When *bError* turns to True. • When *bDone* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy* is False.

***Note:**

AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

  - MC_Reset_DML can change the axis from an abnormal error state to a normal operational state.
    When MC_Power_DML.Enable is True, the axis state changes from Errorstop to Standstill.
    When MC_Power_DML.Enable is False, the axis state changes from Errorstop to Disabled.
  - When the servo controller reports an error, MC_Reset_DML can be used to clear the error. After the error is cleared, the axis state will return to Standstill or Disabled.
  - If errors (e.g., a communication error) cannot be cleared by MC_Reset_DML, the instruction will report DML_R_ERROR_NOT_RESETTABLE (122) error.

- **Troubleshooting**

  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the error code in ErrorID.

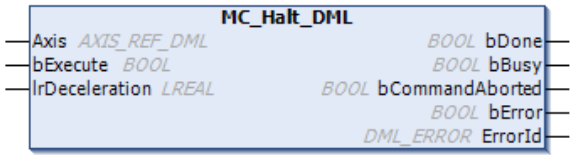
- **Programming Example**

  - For the example, refer to the programming example for MC_Reset function block
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.4 MC_Halt_DML

- **Supported Devices**: AX-308E, AX-364E

MC_Halt_DML halts an axis in a controllable way.

FB/FC	Instruction	Graphic Expression
FB	MC_Halt_DML	

ST Language

```
MC_Halt_DML_instance(
Axis : =,
bExecute : =,
lrDeceleration : =,
bDone =>,
bBusy =>,
bCommandAborted =>,
bError =>,
ErrorID => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* turns from False to True.	BOOL	True/False (False)	-
lrDeceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True.

- **Outputs**

Name	Function	Data Type	Output Range (Default)
bDone	True when the axis stops and the velocity is 0.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)

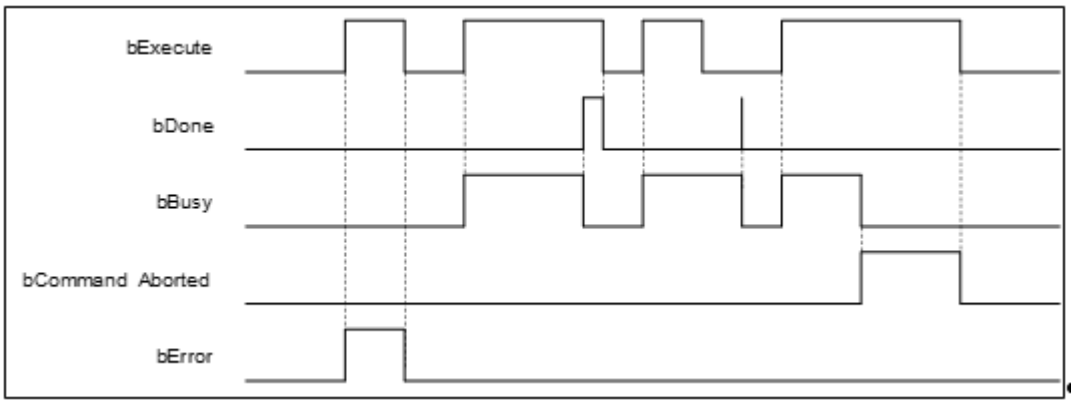
Name	Function	Data Type	Output Range (Default)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the axis decelerates to a stop.	• When *bExecute* turns to False. • If *bExecute* is False and *bDone* turns to True, *bDone* will be True for one period and then immediately shift to False.
bBusy	• When *bExecute* turns to True and the instruction is run.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bCommandAborted	• When this instruction is aborted by another function block.	• When *bExecute* turns to False. • If *bExecute* is False and *bCommandAborted* turns to True, *bCommandAborted* will be True for one period and immediately shift to False.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy* is False.

***Note**:

AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

- MC_Halt_DML is different from MC_Stop_DML in stopping the axis motion. MC_Halt_DML can be interrupted by other motion function blocks.
- When MC_Halt_DML is run, the axis will enter discrete_motion state. When the velocity of the axis reaches zero, the axis will enter Standstill state.

- **Troubleshooting**

  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the error code in ErrorID.

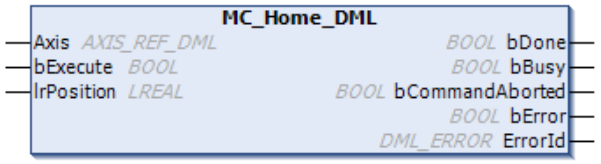
- **Programming Example**

  - For the example, refer to the programming example of the MC_Halt function block.
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.5 MC_Home_DML

- **Supported Devices**: AX-308E, AX-364E

MC_Home_DML controls the axis to perform the homing operation.

FB/FC	Instruction	Graphic Expression
FB	MC_Home_DML	

ST Language

```
MC_Home_DML_instance(
Axis : =,
bExecute: =,
lrPosition: =,
bDone =>,
bBusy =>,
bCommandAborted =>,
bError =>,
ErrorID => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* turns from False to True.	BOOL	True/False (False)	-
lrPosition	Specify the absolute home position. (Unit: user unit)	LREAL	Positive, negative or 0 (0)	When *bExecute* turns to True and *bBusy* is False.

- **Outputs**

Name	Function	Data Type	Output Range (Default)
bDone	True when homing is completed and the axis is in Standstill state.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bCommandAborted	True when the instruction is	BOOL	True/False (False)

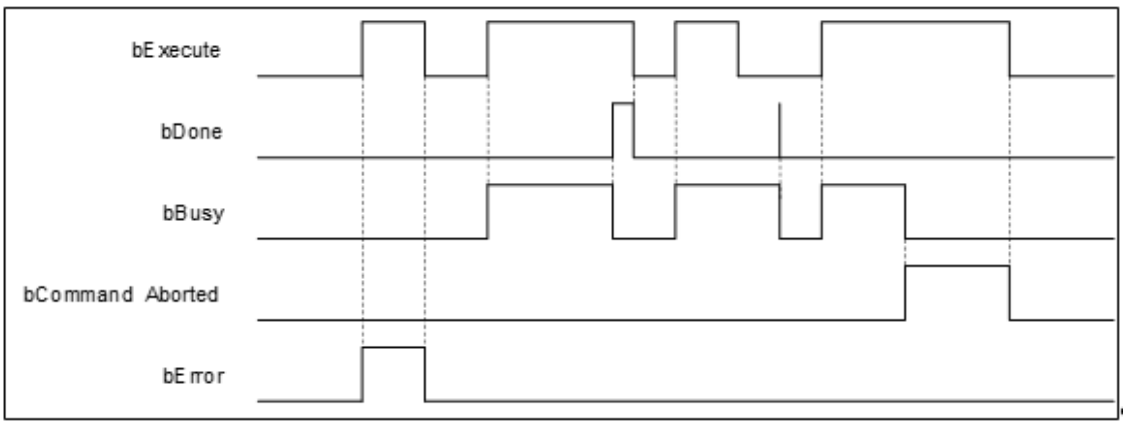
Name	Function	Data Type	Output Range (Default)
	interrupted.		
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML _NO_ERROR)

***Note:** DML_ERROR: Enumeration (ENUM)

■ **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the homing is completed.	• When *bExecute* turns to False. • If *bExecute* is False and *bDone* turns to True, *bDone* will be True for one period and then immediately shift to False.
bBusy	• When *bExecute* turns to True and the instruction is run.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bCommandAborted	• When this instruction is aborted by another instruction. • When the instruction is aborted by MC_Stop_DML.	• When *bExecute* turns to False. • If *bExecute* is False and *bCommandAborted* turns to True, *bCommandAborted* will be True for one period and immediately shift to False.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When Error Code is cleared.
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy* is

	axis.			False.

***Note:**

AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

  - This function block is run only when the axis is in Standstill state and the state is Homing during the instruction execution. The function block cannot be run when the axis is in any state else.
  - *Position* is the absolute position when the homing is completed.
  - The home mode can be selected from the axis parameter page.

- **Troubleshooting**

  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the Error Code in ErrorID.

- **Programming Example**

  - For the example, refer to the programming example of the MC_Home function block.
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.6 MC_MoveAbsolute_DML

- **Supported Devices**: AX-308E, AX-364E

MC_MoveAbsolute_DML controls the specified axis to move to the specified absolute target position based on the specified motion behavior.

FB/FC	Instruction	Graphic Expression
FB	MC_MoveAbsolute_DML	

ST Language

```
MC_MoveAbsolute_DML_instance(
Axis : =,
bExecute : =,
lrPosition : =,
lrVelocity : =,
lrAcceleration : =,
lrDeceleration : =,
bDone =>,
bBusy =>,
bCommandAborted =>,
bError =>,
ErrorID => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* turns from False to True.	BOOL	True/False (False)	-
lrPosition	Absolute target position (Unit: user unit)	LREAL	Negative, positive or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
lrVelocity	Target velocity (Unit: user unit/s)	LREAL	Positive or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
lrAcceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
lrDeceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True and *bBusy* is False.

- **Outputs**

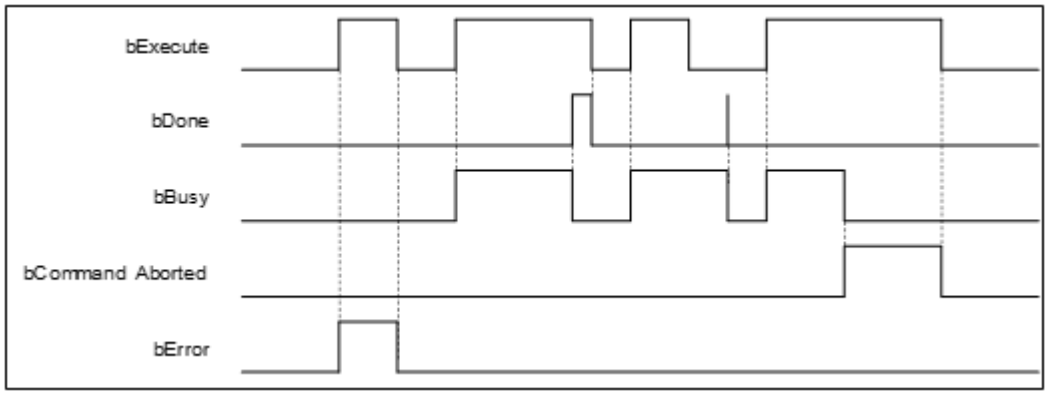
Name	Function	Data Type	Output Range (Default)
bDone	True when the absolute target position is reached.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• True when the absolute target position is reached.	• When *bExecute* turns to False. • If *bExecute* is False and *bDone* turns to True, *bDone* will be True for one period and then immediately shift to False.
bBusy	• When *bExecute* turns to True and the instruction is run.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bCommandAborted	• When this instruction is aborted by another instruction. • When the instruction is aborted by MC_Stop_DML.	• When *bExecute* turns to False. • If *bExecute* is False and *bCommandAborted* is True, *bCommandAborted* will immediately change to False after maintaining a True state for a scan cycle.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy* is False.

***Note:**
AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

  - MC_MoveAbsolute_DML performs absolute positioning according to specified target velocity (*lrVelocity*), acceleration rate (*lrAcceleration*) and deceleration rate (*lrDeceleration*) when *bExecute* turns to True.

- **Troubleshooting**

  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the error code in ErrorID.

- **Programming Example**

  - For the example, refer to the programming example of the MC_MoveAbsolute function block.
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.7 MC_MoveRelative_DML

- **Supported Devices**: AX-308E, AX-364E

MC_MoveRelative_DML controls the specified axis to move to the specified relative target position according to the specified motion behavior.

FB/FC	Instruction	Graphic Expression
FB	MC_MoveRelative_DML	

ST Language

```
MC_MoveRelative_DML_instance(
Axis : =,
bExecute : =,
lrDistance : =,
lrVelocity : =,
lrAcceleration : =,
lrDeceleration : =,
bDone =>,
bBusy =>,
bCommandAborted =>,
bError =>,
ErrorID => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* turns to True.	BOOL	True/False (False)	-
lrDistance	Relative distance to be moved (Unit: user unit)	LREAL	Negative, positive or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
lrVelocity	Target velocity (Unit: user unit/s)	LREAL	Positive or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
lrAcceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
lrDeceleration	Deceleration rate	LREAL	Positive	When *bExecute* turns to True and *bBusy* is False.

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
	(Unit: user unit/s^2)		(0)	

- **Outputs**

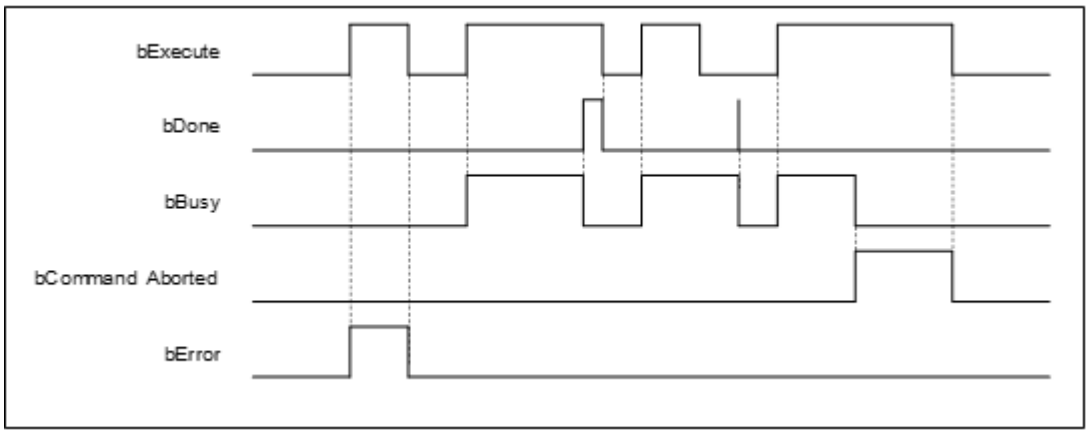
Name	Function	Data Type	Output Range (Default)
bDone	True when the relative distance is completed.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the relative positioning is completed.	• When *bExecute* turns to False. • If *bExecute* is False and *bDone* turns to True, *bDone* will be True for one period and immediately shift to False.
bBusy	• When *bExecute* turns to True and the instruction is run.	• When *bDone* turns to True. • When *bError* turns to True. • When *bCommandAborted* turns to True.
bCommandAborted	• When this instruction is aborted by another instruction. • When the instruction is aborted by MC_Stop_DML.	• When *bExecute* turns to False. • If *bExecute* is False and *bCommandAborted* is True, *bCommandAborted* will immediately change to False after maintaining a True state for a scan cycle.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy* is False.

***Note:**
AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

  - MC_MoveRelative_DML performs relative positioning according to specified target velocity (*lrVelocity*), acceleration rate (*lrAcceleration*) and deceleration rate (*lrDeceleration*) when *bExecute* turns to True.

- **Troubleshooting**

  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the Error Code in ErrorID.

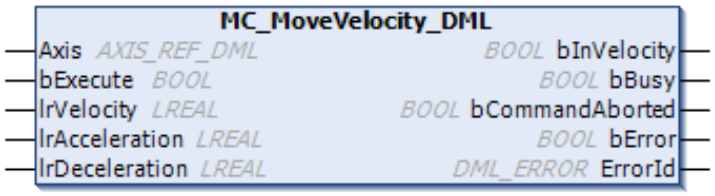
- **Programming Example**

  - For the example, refer to the programming example of the MC_MoveRelative function block.
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.8 MC_MoveVelocity_DML

- **Supported Devices**: AX-308E, AX-364E

MC_MoveVelocity_DML performs velocity control on an axis in the position mode with a specified behavior and a constant velocity.

FB/FC	Instruction	Graphic Expression
FB	MC_MoveVelocity_DML	

ST Language

```
MC_MoveVelocity_DML_instance(
Axis : =,
bExecute : =,
lrVelocity : =,
lrAcceleration : =,
lrDeceleration : =,
bInVelocity =>,
bBusy =>,
bCommandAborted =>,
bError =>,
ErrorID => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* turns from False to True.	BOOL	True/False (False)	-
lrVelocity	Target velocity (Unit: user unit/s)	LREAL	Positive or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
lrAcceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True and *bBusy* is False.
lrDeceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive (0)	When *bExecute* turns to True and *bBusy* is False.

- **Outputs**

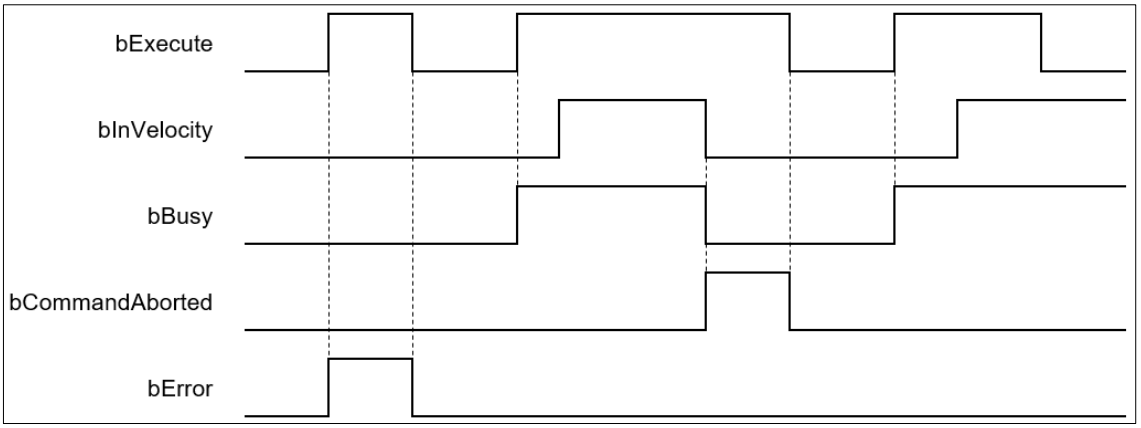
Name	Function	Data Type	Output Range (Default)
bInVelocity	True when the target velocity is reached.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bInVelocity	• When the target velocity is reached.	• When *bCommandAborted* turns to True. • When *bExecute* turns to True again and *lrVelocity* value is changed.
bBusy	• When *bExecute* turns to True and the instruction is run.	• When *bError* turns to True. • When *bCommandAborted* turns to True.
bCommandAborted	• When this instruction is aborted by another instruction. • When the instruction is aborted by MC_Stop_DML.	• When *bExecute* turns to False. • If *bExecute* is False and *bCommandAborted* is True, *bCommandAborted* will immediately change to False after maintaining a True state for a scan cycle.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy* is False.

***Note:**
AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**
  - When *bExecute* turns to True, the instruction will perform constant-velocity motion according to the specified target speed (*lrVelocity*), acceleration (*lrAcceleration*), and deceleration (*lrDeceleration*).
  - The executing MC_MoveVelocity_DML can be aborted by another motion instruction.
  - When the instruction is aborted by another instruction, the output *bInVelocity* changes to False and the output *bCommandAborted* changes to True.
  - When *bExecute* of MC_MoveVelocity_DML switches to True, the axis will start to move at the target velocity. Even if *bExecute* switches to False, the operation of the function block will not be affected.
  - When *bExecuteInputs* of MC_MoveVelocity_DML changes to True again and a new target velocity is assigned, the axis velocity is adjusted to the new velocity.
  - When *bExecute* changes to False after the function block is run and then the target velocity is reached, the *bInVelocity* of MC_MoveVelocity_DML changes to True. Afterward, *bInVelocity* will be True until it is aborted by another instruction.

- **Troubleshooting**
  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the Error Code in ErrorID.

- **Programming Example**
  - For the example, refer to the programming example of the MC_MoveVelocity function block.
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.9 MC_WriteBoolParameter_DML

- **Supported Devices**: AX-308E, AX-364E

MC_WriteBoolParameter_DML writes a Boolean value in the specified parameter.

FB/FC	Instruction	Graphic Expression
FB	MC_WriteBoolParameter_DML	
ST Language		

```
MC_WriteBoolParameter_instance(
Axis : =,
bExecute : =,
diParameterNumber : =,
bValue : =,
bDone =>,
bBusy =>,
bError =>,
ErrorID => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* turns to True.	BOOL	True/False (False)	-
diParameterNumber	Specify the number of the axis parameter.	DINT	Positive, negative or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
bValue	Set a Boolean value of the parameter to write.	BOOL	True/False (False)	When *bExecute* turns to True and *bBusy* is False.

- **Outputs**

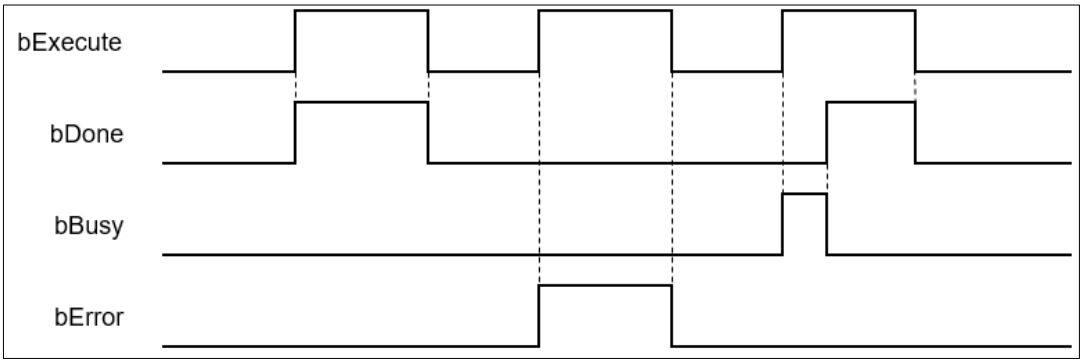
Name	Function	Data Type	Output Range (Default)
bDone	True when the parameter writing is completed.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the parameter writing is completed.	• When *bExecute* turns from True to False.
bBusy	• When *bExecute* turns to True and the instruction is run. • When parameter writing is in progress.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy* is False.

***Note:** AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

  - How to use MC_WriteBoolParameter_DML to write the number of an EtherCAT object dictionary
    - Use the SHL instruction to shift the data length of the object dictionary where a value is to be written to the left by 24 bits
    - Use the SHL instruction to shift the index of the object dictionary where a value is to be written to the left by 8 bits
    - Add up the above parameters and the sub-index.

      See the reference formula as follows.

      diParameterNumber: =-DWORD_TO_DINT (SHL (TO_DWORD (object dictionary data length), 24) + SHL (TO_DWORD (object dictionary index), 8) + object sub-index);
  - To write a value in an axis parameter, refer to the axis parameter AXIS_REF_DML (FB) and fill in its number in the *diParameterNumber* input parameter.

- **Troubleshooting**

  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the Error Code in ErrorID.

- **Programming Example**

  - For the example, refer to the programming example of the MC_WriteParameter function block.
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.10 MC_ReadBoolParameter_DML

- **Supported Devices**: AX-308E, AX-364E

MC_ReadBoolParameter_DML reads the Boolean value of a specified parameter.

FB/FC	Instruction	Graphic Expression
FB	MC_ReadBoolParameter_DML	

ST Language

```
MC_ReadBoolParameter_DML_instance(
Axis : =,
bEnable : =,
diParameterNumber : =,
bValid =>,
bBusy =>,
bError =>,
ErrorID =>,
bValue => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bEnable	The instruction is run when *bEnable* turns to True.	BOOL	True/False (False)	-
diParameterNumber	Specify the number of the axis parameter.	DINT	Positive, negative or 0 (0)	When *bEnable* turns to True.

- **Outputs**

Name	Function	Data Type	Output Range (Default)
bValid	True when the read parameter value is available.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)

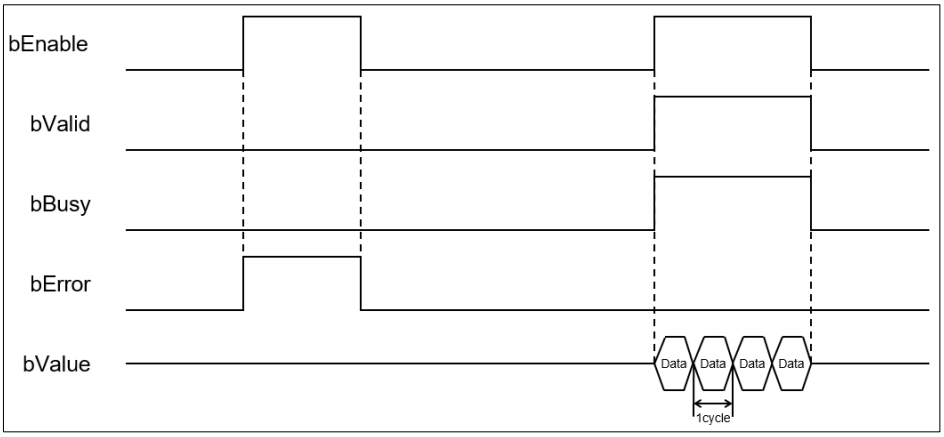
Name	Function	Data Type	Output Range (Default)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)
bValue	The read parameter value	BOOL	True/False (False)

***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bValid	• When *bEnable* turns to True. • When the parameter to be read is available.	• When *bEnable* turns from True to False. • When *bError* turns to True.
bBusy	• When *bEnable* turns to True and the instruction is run. • When the parameter to be read is available.	• When *bEnable* turns from True to False. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bEnable* is False. (Error Code is cleared)
ErrorID		
bValue	• Updates continuously when bValid is True.	• Update stops when *bValid* is False.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bEnable* changes to True.

***Note:**
AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

- How to use MC_ReadBoolParameter_DML to read the number of an EtherCAT object dictionary.
  - Use the SHL instruction to shift the data length of the object dictionary to be read to the left by 24 bits
  - Use the SHL instruction to shift the index of the object dictionary to be read to the left by 8 bits
  - Add up the above parameters and the sub-index.

    See the reference formula as follows.
  - diParameterNumber: = - DWORD_TO_DINT (SHL (TO_DWORD (object dictionary data length), 24) + SHL (TO_DWORD (object dictionary index), 8) + object sub-index);
- To read an axis parameter value, refer to the axis parameter AXIS_REF_DML (FB) and fill in its number in the *diParameterNumber* input parameter.

- **Troubleshooting**

  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the Error Code in ErrorID.

- **Programming Example**

  - For the example, refer to the programming example of the MC_ReadParameter function block.
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.11 MC_WriteParameter_DML

- **Supported Devices**: AX-308E, AX-364E

MC_WriteParameter_DML writes a value in the specified parameter.

B/FC	Instruction	Graphic Expression
FB	MC_WriteParameter_DML	

ST Language

```
MC_WriteParameter_DML_instance(
Axis : =,
bExecute : =,
diParameterNumber : =,
lrValue : =,
bDone =>,
bBusy =>,
bError =>,
ErrorID => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* turns to True.	BOOL	True/False (False)	-
diParameter Number	Specify the number of the axis parameter.	DINT	Positive, negative or 0 (0)	When *bExecute* is True and *bBusy* is False.
lrValue	Set a parameter value to write.	LREAL	Positive, negative or 0 (0)	When *bExecute* is True and *bBusy* is False.

- **Outputs**

Name	Function	Data Type	Output Range (Default)
bDone	True when the parameter writing is completed.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)

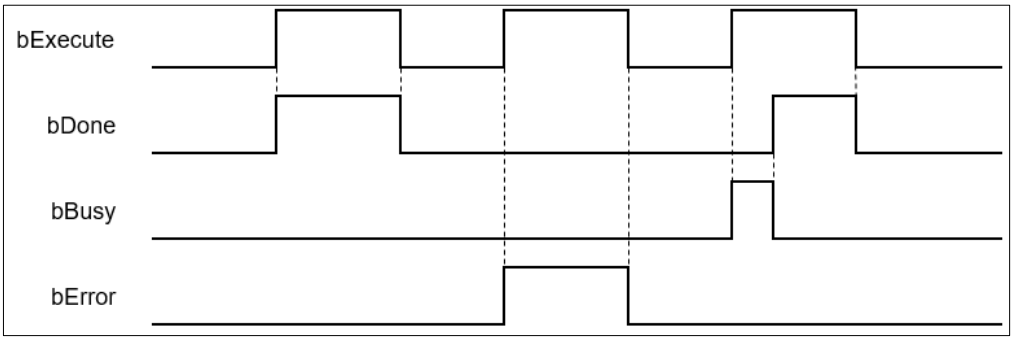
Name	Function	Data Type	Output Range (Default)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the parameter writing is completed.	• When *bExecute* turns from True to False.
bBusy	• When *bExecute* turns to True and the instruction is run. • When parameter writing is in progress.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy* is False.

***Note:**
AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

  - How to use MC_WriteParameter_DML to write the number of an EtherCAT object dictionary.
    - Use the SHL instruction to shift the data length of the object dictionary where a value is to be written to the left by 24 bits
    - Use the SHL instruction to shift the index of the object dictionary where a value is to be written to the left by 8 bits
    - Add up the above parameters and the sub-index.

      See the reference formula as follows.

diParameterNumber : =-DWORD_TO_DINT (SHL (TO_DWORD (object dictionary data length), 24) + SHL (TO_DWORD (object dictionary index), 8) + object sub-index);

- To write a value in an axis parameter, refer to the axis parameter AXIS_REF_DML (FB) and fill in its number in the *diParameterNumber* input parameter.

- **Troubleshooting**

  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the Error Code in ErrorID.

- **Programming Example**

  - For the example, refer to the programming example of the MC_WriteParameter function block.
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.12 MC_ReadParameter_DML

- **Supported Devices**: AX-308E, AX-364E

MC_ ReadParameter_DML reads the value of a specified parameter.

FB/FC	Instruction	Graphic Expression
FB	MC_ReadParameter_DML	
ST Language		
MC_ReadParameter_DML_instance(Axis : =, bEnable : =, diParameterNumber : =, bValid =>, bBusy =>, bError =>, ErrorID =>, lrValue =>);		

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bEnable	The instruction is run when *bEnable* turns to True.	BOOL	True/False (False)	-
diParameterNumber	Specify the number of the axis parameter.	DINT	Positive, negative or 0 (0)	*When bEnable turns to True.*

- **Outputs**

Name	Function	Data Type	Output Range (Default)
bValid	True when the read parameter value is available.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	When a command error occurs,	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

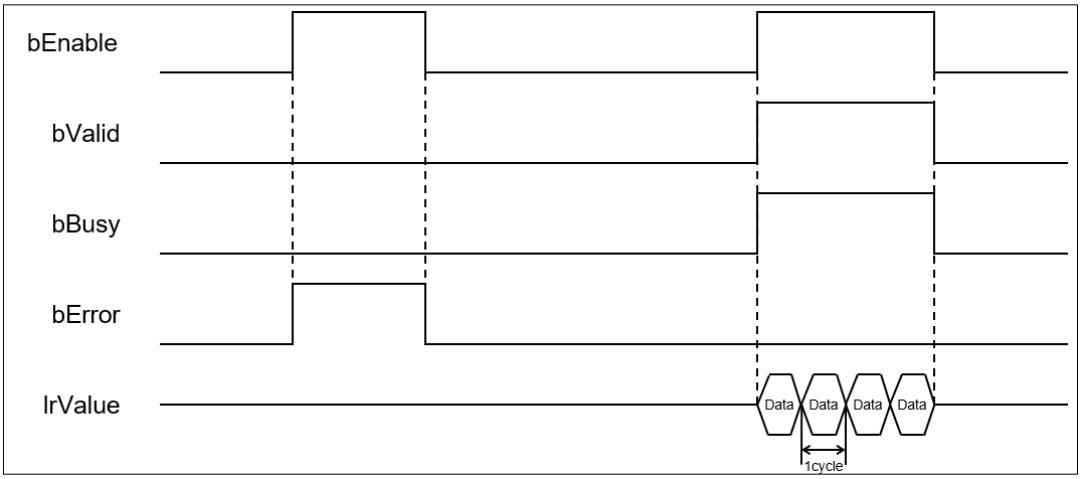
Name	Function	Data Type	Output Range (Default)
	record the error code. For the detailed description of the error code, refer to the appendix of the manual		
lrValue	The read parameter value	LREAL	Positive, negative or 0 (0)

***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bValid	• When *bEnable* turns to True. • When the read parameter value is available.	• When *bEnable* turns from True to False. • When *bError* turns to True.
bBusy	• When *Enable* turns to True and the instruction is run. • When the read parameter value is available.	• When *bEnable* turns from True to False. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bEnable* is False. (Error Code is cleared)
ErrorID		
lrValue	• Updates continuously when *bValid* is True.	• Update stops when *bValid* is False.

- **Timing Diagram**



***Note:**
Data = Parameter values
1 cycle = One task cycle

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bEnable* turns to True.

***Note:**

AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

  - How to use MC_ReadParameter_DML to read the number of an EtherCAT object dictionary
    - Use the SHL instruction to shift the data length of the object dictionary to be read to the left by 24 bits
    - Use the SHL instruction to shift the index of the object dictionary to be read to the left by 8 bits
    - Add up the above parameters and the sub-index.

      See the reference formula as follows.

    - diParameterNumber: = - DWORD_TO_DINT (SHL (TO_DWORD (object dictionary data length), 24) + SHL (TO_DWORD (object dictionary index), 8) + object sub-index);

  - To read an axis parameter, refer to the axis parameter AXIS_REF_DML (FB) and fill in its number in the *diParameterNumber* input parameter.

- **Troubleshooting**

  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the Error Code in ErrorID.

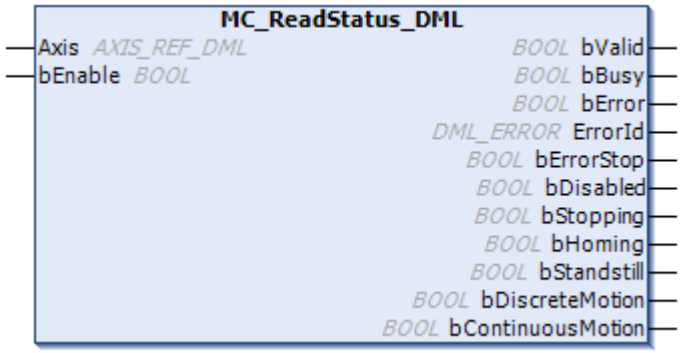
- **Programming Example**

  - For the example, refer to the programming example of the MC_ReadParameter function block.
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.13 MC_ReadStatus_DML

- **Supported Devices**: AX-308E, AX-364E

MC_ReadStatus_DML reads the state of a specified axis.

FB/FC	Instruction	Graphic Expression
FB	MC_ReadStatus_DML	

ST Language

```
MC_ReadStatus_DML_instance(
Axis : =,
bEnable : =,
bValid =>,
bBusy =>,
bError =>,
ErrorID =>,
bErrorStop=>,
bDisabled=>,
bStopping=>,
bHoming=>,
bStandStill=>,
bDiscreteMotion=>,
bContinuousMotion=>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bEnable	The instruction is run when *bEnable* turns to True.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default)
bValid	True when the axis state at the output is available.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*1	DML_ERROR (DML_NO_ERROR)
bErrorStop	To know details on the axis state machine, refer to SML_AXIS_STATE.*2	BOOL	True/False (False)
bDisabled		BOOL	True/False (False)
bStopping		BOOL	True/False (False)
bHoming		BOOL	True/False (False)
bStandStill		BOOL	True/False (False)
bDiscreteMotion		BOOL	True/False (False)
bContinuousMotion		BOOL	True/False (False)

***Note:**

1. DML_ERROR: Enumeration (ENUM)
2. SML_AXIS_STATE: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bValid	• When *bEnable* turns to True. • When the axis state at the output is available.	• When *bEnable* turns from True to False. • When *bError* turns to True.
bBusy	• When *bEnable* turns to True and the instruction is run.	• When *bEnable* turns from True to False. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bEnable* is False. (Error Code is cleared)
ErrorID		
bDisabled	• When the axis is in Disabled state.	• When the axis is not in Disabled state.
bErrorstop	• When the axis is in Errorstop state.	• When the axis is not in Errorstop state.
bStopping	• When the axis is in Stopping state.	• When the axis is not in Stopping state.
bStandStill	• When the axis is in StandStill state.	• When the axis is not in StandStill state.
bDiscreteMotion	• When the axis is in Discrete Motion state.	• When the axis is not in Discrete Motion state.
bContinuousMotion	• When the axis is in Continuous Motion state.	• When the axis is not in Continuous Motion state.

Name	Timing for shifting to True	Timing for shifting to False
bHoming	• When the axis is in Homing state.	• When the axis is not in Homing state.

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bEnable* turns to True.

***Note:**

AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Troubleshooting**

  - When an error occurs in the instruction execution, *bError* changes to True. To confirm the current error state, see the Error Code in ErrorID.

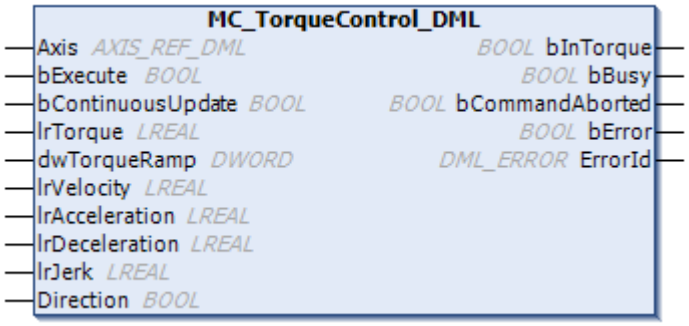
- **Programming Example**

  - For the example, refer to the programming example of the MC_ReadStatus function block.
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.14 MC_TorqueControl_DML

- **Supported Devices**: AX-308E, AX-364E

MC_TorqueControl_DML controls the torque by using the torque control mode of the applied servo drive.

FB/FC	Instruction	Graphic Expression
FB	MC_TorqueControl	

ST Language

```
MC_TorqueControl_DML_instance(
Axis : =,
bExecute : =,
bContinuousUpdate : =,
lrTorque : =,
dwTorqueRamp : =,
lrVelocity : =,
lrAcceleration : =,
lrDeceleration : =,
lrJerk : =,
Direction : =,
bInTorque =>,
bBusy =>,
bCommandAborted =>,
bError =>,
ErrorID =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* turns from False to True.	BOOL	True/False (False)	-
bContinuousUpdate	Continuously updates the target torque when *Continuousupdate* is True.	BOOL	True/False (False)	When *bExecute* turns to True and *Busy* is False.
lrTorque	Specify the target torque. (Unit: N.m)	LREAL	Positive, negative or 0 (0)	When *bExecute* turns to True and *Busy* is False.

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
dwTorqueRamp	Specify the change rate of the torque from current torque to target torque. (Unit: ms) *	DWORD	Positive (0)	When *bExecute* turns to True and *Busy* is False.
lrVelocity	Specify the maximum velocity.	LREAL	Positive (0)	When *bExecute* turns to True and *Busy* is False.
lrAcceleration	Reserved	LREAL	-	-
lrDeceleration	Reserved	LREAL	-	-
lrJerk	Reserved	LREAL	-	-
Direction	Reserved	BOOL	-	-

***Note:**

Here is ASDA-A2 as an example with the unit of microsecond. For other servo models, refer to 0x6087 in the object dictionary.

- **Outputs**

Name	Function	Data Type	Output Range (Default)
bInTorque	True when the target torque is reached.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NoError)

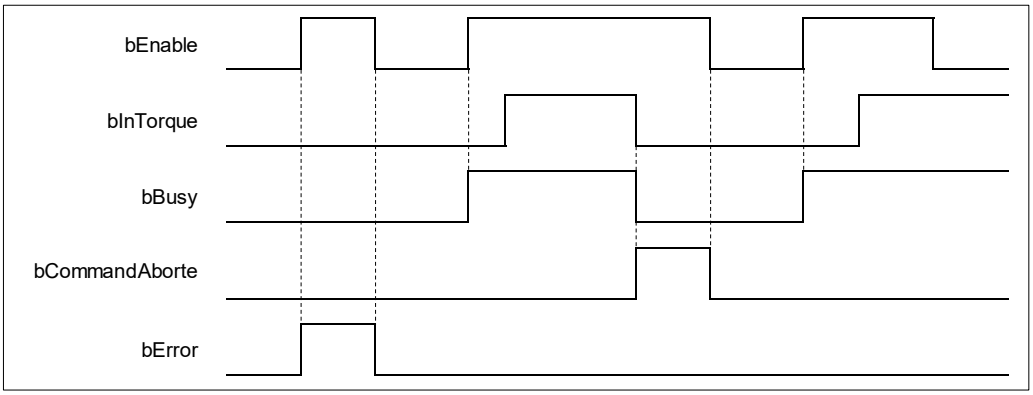
***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bInTorque	• When *bExecute* turns to True and the axis state is available.	• When *bError* turns to True. • When *bCommandAborted* turns to True. • When *bExecute* turns to True again and *lrTorque* value changes.
bBusy	• When *bExecute* turns to True and the instruction is run.	• When *bError* turns to True. • When *bCommandAborted* turns to True.
bCommandAborted	• When the instruction is aborted.	• When *bExecute* turns to False. • If *bExecute* is False and *bCommandAborted* is True, *bCommandAborted* will immediately

Name	Timing for shifting to True	Timing for shifting to False
		change to False after maintaining a True state for a scan cycle.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy* is False.

***Note:**

AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

  - When *bExecute* of the instruction turns to True, the target torque (*lrTorque*), torque ramp (*dwTorqueRamp*) and maximum velocity (*lrVelocity*) will be sent to the servo for servo torque control.
  - When *bExecute* is True, the instruction is run for a velocity-constant motion according to the specified target velocity (*lrVelocity*), acceleration rate (*lrAcceleration*) and deceleration rate (*lrDeceleration*).
  - The executing MC_TorqueControl_DML can be aborted by executing another motion instruction.
  - When MC_TorqueControl_DML is aborted by another motion instruction, the output *bInTorque* will shift to False and the output *bCommandAborted* will shift to True.
  - When *bExecute* of MC_TorqueControl_DML turns to True, the axis starts to move according to the target velocity. Even if *bExecute* turns to False, the instruction execution will not be affected.
  - When *bExecute* of MC_TorqueControl_DML turns to True again and a new *lrTorque* value is set, the torque of the axis is adjusted to the new torque value.
  - When its *bExecute* turns to False after the instruction is run and the target torque is reached, *bInTorque* of MC_TorqueControl_DML turns to True. Afterward, *bInTorque* will remain True until it is aborted by another instruction.
  - When using C2000+ or CH2000 Series AC Motor Drives, it is necessary to configure 0x6064 (Position actual value) and 0x6077 (Torque actual value) to the Slave PDO (Process data) mapping data.

- **Troubleshooting**

- When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the Error Code in *ErrorID*.

- **Programming Example**

  - For the example, refer to the programming example of the DMC_TorqueControl function block.
  - For function block Axis parameters, enter that of the positioning axis.

2.3.3.15 MC_ChangeAxisConfig_DML

- **Supported Devices**: AX-308E, AX-364E

MC_ChangeAxisConfig_DML modifies basic axis settings including the ratio between user units and pulse number, axis type and user units per rotary axis rotation.

FB/FC	Instruction	Graphic Expression
FB	MC_ChangeAxisConfig_DML	

ST Language

```
MC_ChangeAxisConfig_DML_instance(
Axis : =,
bExecute : =,
dwRatioTechUnitsDenom : =,
iRatioTechUnitsNum : =,
fModuloPeriodU : =,
fMovementType : =,
bDone =>,
bBusy =>,
bError =>,
ErrorID => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* turns to True.	BOOL	True/False (False)	-
dwRatioTechUnitsDenom	Electronic gear ratio denominator (Pulse number)	DWORD	Positive or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
iRatioTechUnits Num	Electronic gear ratio numerator (User units)	DINT	Positive, negative or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
fModuloPeriodU	Maximum position of the rotary axis	LREAL	Positive, negative or 0 (0)	When *bExecute* turns to True and *bBusy* is False.
fMovementType	Linear axis/rotary axis	SML_MOVE MENTTYPE	0: SML_MT_MOD ULO	When *bExecute* turns to True and *bBusy* is False.

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
			1: SML_MT_FINITE	

- **Outputs**

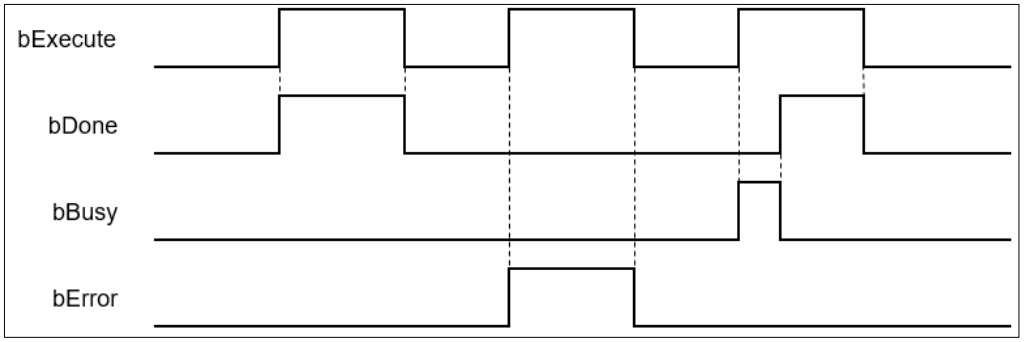
Name	Function	Data Type	Output Range (Default)
bDone	True when the parameter writing is completed.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the parameter writing is completed.	• When *bExecute* turns from True to False.
bBusy	• When *bExecute* turns to True and the instruction is run. • When the parameter writing is in progress.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy*

Name	Function	Data Type	Setting Value	Timing to Take Effect
	axis.			is False.

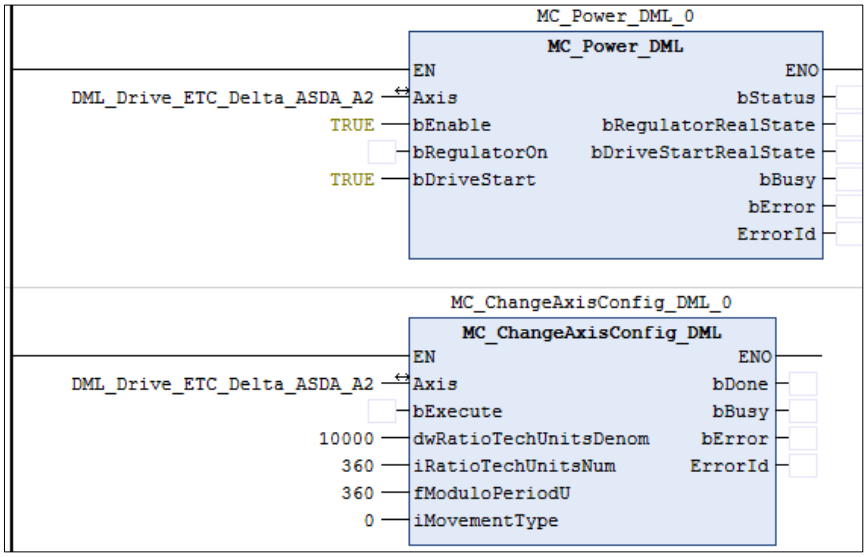
***Note:**

AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**
  - MC_ChangeAxisConfig_DML can be used to modify basic axis settings including the ratio between user units and pulse number (electronic gear ratio), axis type and user units per rotary axis rotation.
  - The axis state must be Disabled if this function block is used.
  - After modification, the new axis settings cannot be retained when power off and so they will disappear after repowering or resetting. And the axis will still run based on the settings on the axis parameter page next time.

- **Troubleshooting**
  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the Error Code in *ErrorID*.

- **Programming Example**
  - This example explains how MC_ChangeAxisConfig_DML is used to modify axis parameters.

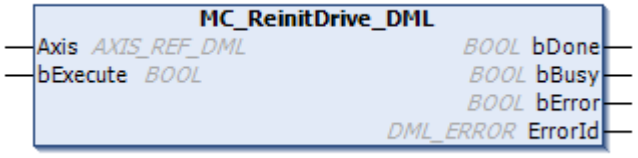


  - Set the gear ratio denominator and gear ratio numerator (10000: 360), maximum position of the rotary axis (360) and axis type (0). Change *bRegulatorOn* of MC_Power to False before MC_ChangeAxisConfig is run.

2.3.3.16 MC_ReinitDrive_DML

- **Supported Devices**: AX-308E, AX-364E

MC_ReinitDrive_DML reinitializes the specified axis.

FB/FC	Instruction	Graphic Expression
FB	MC_ReinitDrive_DML	

ST Language

```
MC_ReinitDrive_DML_instance(
Axis : =,
bExecute : =,
bDone =>,
bBusy =>,
bError =>,
ErrorID => );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	The instruction is run when *bExecute* turns to True.	BOOL	True/False (False)	-

- **Outputs**

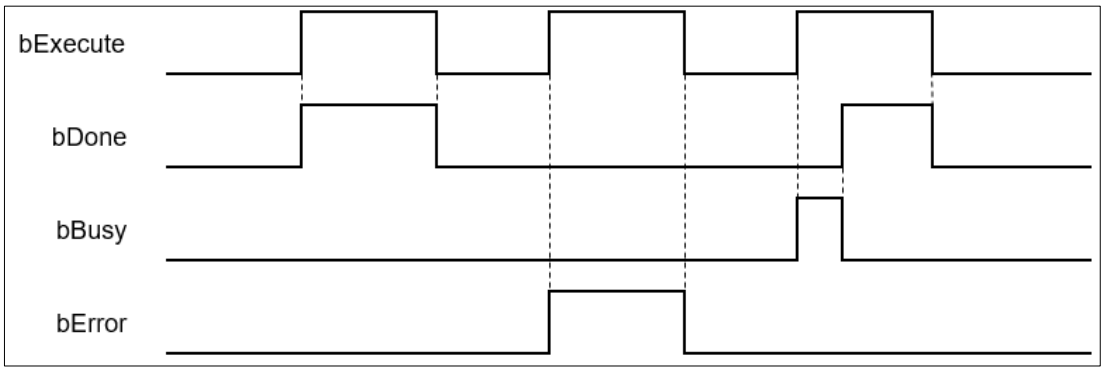
Name	Function	Data Type	Output Range (Default)
bDone	True when initialization is completed.	BOOL	True/False (False)
bBusy	True when the instruction is run.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When initialization is completed.	• When *bExecute* turns from True to False.
bBusy	• When *bExecute* turns to True and the instruction is run. • When initialization is in progress.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction.	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy* is False.

***Note:**

AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

  - MC_ReinitDrive_DML is used to initialize the drive which has an error or is no longer in synchronization.
  - Resetting the network cannot clear errors of the positioning axis itself, e.g. left and right limits, emergency stop and etc.
  - The function block resets EtherCAT network of the positioning axis and keeps the servo in the previous servo state. Refer to the following table for details.

Before the reset action	Resetting	After the reset action
Servo Off	Servo Off	Servo Off
Servo On	Servo Off	Servo On

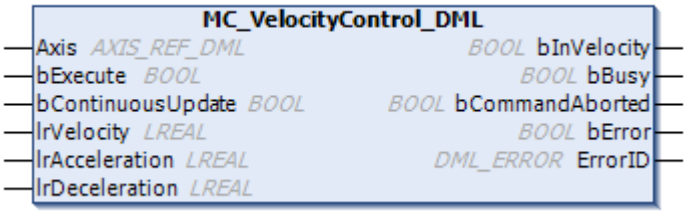
- **Troubleshooting**

  - When an error occurs during the execution of the instruction, *bError* changes to True. To confirm the current error state, see the error code in *ErrorID*.

2.3.3.17 MC_VelocityControl_DML

- **Supported Devices**: AX-308E, AX-364E

MC_VelocityControl_DML controls the specified axis to move evenly according to the specified motion mode and speed in speed mode (VL).

FB/FC	Instruction	Graphic Expression
FB	MC_VelocityControl_DML	

ST Language

```
MC_VelocityControl_DML_instance(
Axis : =,
bExecute : =,
bContinuousUpdate : =,
lrVelocity : =,
lrAcceleration : =,
lrDeceleration : =,
bInVelocity =>,
bBusy =>,
bCommandAborted =>,
bError =>,
ErrorID =>);
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	Run the funclion block	BOOL	True/False (False)	-
bContinuousUpdate*	When bContinuousUpdate is True, the target speed will be updated continuously	BOOL	True/False (False)	When *bExecute* turns to True and *Busy* is False
lrVelocity	Target speed (user unit/sec)	LREAL	Positive (0)	When *bExecute* turns to True and *Busy* is False

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
lrAcceleration	Acceleration (user unit/sec^2)	LREAL	Positive (0)	When *bExecute* turns to True and *Busy* is False
lrDeceleration	Deceleration (user unit/sec^2)	LREAL	Positive (0)	When *bExecute* turns to True and *Busy* is False

***Note**: When bContinuousUpdate is activated, the speed, acceleration and deceleration are modified immediately.

- **Outputs**

Name	Function	Data Type	Setting Value (Default Value)
bInVelocity	When the target speed is reached.	BOOL	True/False (False)
bBusy	True when the instruction is running.	BOOL	True/False (False)
bCommandAborted	True when the instruction is interrupted.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendices for error code descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

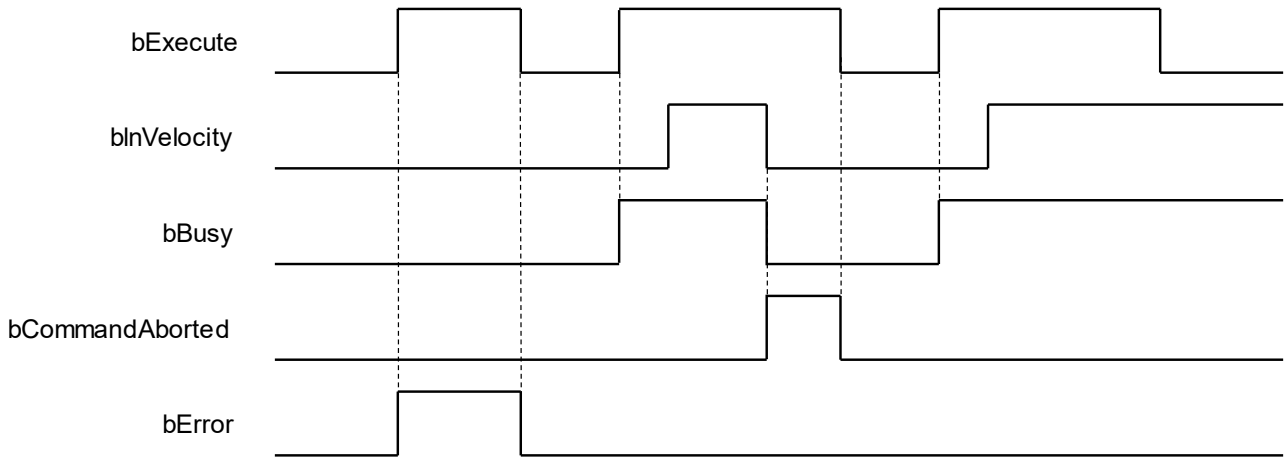
***Note:** DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bInVelocity	• When the axis speed reaches the target speed	• When *bCommandAborted* turns to True. • When *bContinuousUpdate* is true and write a new value to *lrVelocity*. • When *bError* turns to True.
bBusy	• When *bEnable* turns to True.	• When *bError* turns to True. • When *bCommandAborted* turns to True.

Name	Timing for shifting to True	Timing for shifting to False
bCommandAborted	• When the function block instruction is interrupted by another function block instruction. • When the function block instruction is interrupted by MC_Stop.	• When *bExecute* turns to False. • If *bExecute* is False and *bCommandAborted* is True, *bCommandAborted* will immediately change to False after maintaining a True state for a scan cycle.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded in *ErrorID*).	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When bExecute turns to True and *bBusy* is False

***Note**: AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

  - When *bExecute* is True, this function block runs in an even speed based on the target velocity (lrVelocity), acceleration (lrAcceleration), deceleration (lrDeceleration) and jerk (lrJerk) specified by the user.
  - When the bContinuousUpdate input parameter of the function block is True and a new target speed is assigned, the axis speed is adjusted to the new speed.

- **Troubleshooting**
  - When an error occurs during the execution of the instruction, *bError* will turn to True. To confirm the current error state, see the Error Code in *ErrorID*.

- **Example**

  - Refer to DMC_VelocityControl.
  - For function block Axis parameters, enter that of the positioning axis.

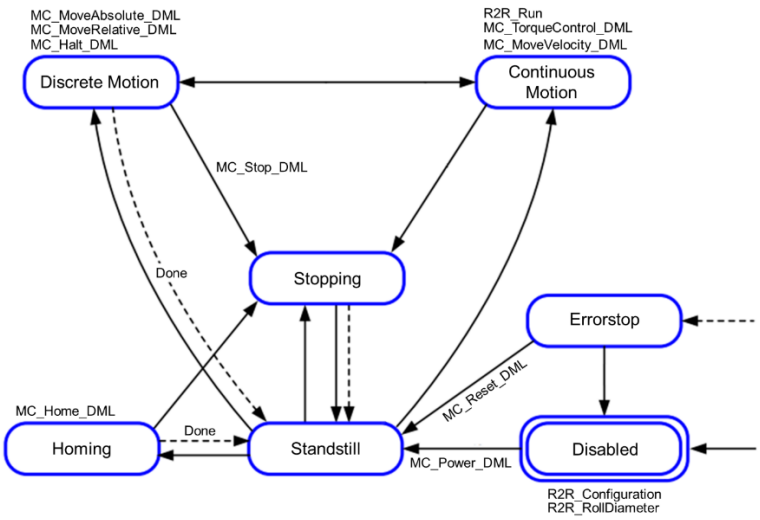
2.3.4 Tension Control

The function blocks mentioned in this section are from library "DL_MotionControlLight", and the frequency converter is mainly set up through communication to achieve tension control. The relevant settings of the frequency converter can be found in section 7.4.2.1 in *AX-3 Series Operation Manual*.

- Introduction to tension control status

  When running R2R_Configuration and R2R_RollDiameter function blocks, the axis status should be Disabled. When running R2R_Run function block，the axis status will change from Disabled to Standstill to Continuous Motion. Otherwise R2R_Run will switch to Disabled when the function block is turned off.

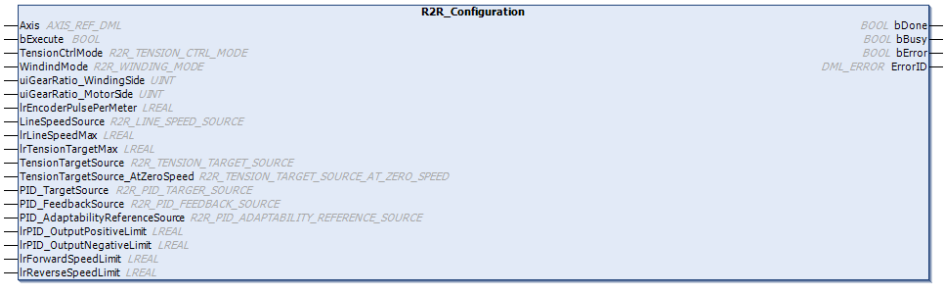
  - Positioning axis status:



2.3.4.1 R2R_Configuration

- **Supported Devices**: AX-308E, AX-364E

R2R_Configuration configures tension control.

FB/FC	Instruction	Graphic Expression
FB	R2R_Configuration	

ST Language

```
R2R_Configuration_instance(
Axis: = ,
bExecute: = ,
TensionCtrlMode: = ,
WindindMode: = ,
uiGearRatio_WindingSide: = ,
uiGearRatio_MotorSide: = ,
lrEncoderPulsePerMeter: = ,
LineSpeedSource: = ,
lrLineSpeedMax: = ,
lrTensionTargetMax: = ,
TensionTargetSource: = ,
TensionTargetSource_AtZeroSpeed: = ,
PID_TargetSource: = ,
PID_FeedbackSource: = ,
PID_AdaptabilityReferenceSource: = ,
lrPID_OutputPositiveLimit: = ,
lrPID_OutputNegativeLimit: = ,
bDone=> ,
bBusy=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction will be run when *bExecute* changes from False to True.	BOOL	True/False (False)	-
TensionCtrlMode	Tension control mode	R2R_TENSION_CTRL_MODE[*1]	0: TensionCloseLoop_SpeedMode 1: LineSpeedCloseLoop_SpeedMode (Reserved) 2: TensionCloseLoop_TorqueMode	When *bExecute* turns to True and *bBusy* is False

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
			3: TensionOpenLoop_TorqueMode (TensionCloseLoop_SpeedMode)	
WindindMode	Winding mode	R2R_WINDING_MODE*2	0: Rewind 1: Unwind (Rewind)	When *bExecute* turns to True and *bBusy* is False
uiGearRatio_WindingSide	Winding side mechanical gear A	UINT	1~65535 (100)	When *bExecute* turns to True and *bBusy* is False
uiGearRatio_MotorSide	Motor side mechanical gear B	UINT	1~65535 (100)	When *bExecute* turns to True and *bBusy* is False
lrEncoderPulsePerMeter	Number of pulses per meter (Pulse/m)	LREAL	0~6000 (0)	When *bExecute* turns to True and *bBusy* is False
LineSpeedSource	Line speed input source	R2R_LINE_SPEED_SOURCE*3	0: R2R_Run_lrLineSpeedValue 1: AVI 2: ACI 3: PG_CARD 4: DFM_DCM 5: MI6MI7 (R2R_Run_lrLineSpeedValue)	When *bExecute* turns to True and *bBusy* is False
lrLineSpeedMax	Maximum line speed (m/mm)	LREAL	0.0~3000.0 (1000.0)	When *bExecute* turns to True and *bBusy* is False
lrTensionTargetMax	Maximum tension value (N)	LREAL	0~65535 (0)	When *bExecute* turns to True and *bBusy* is False
TensionTargetSource	Source of tension command	R2R_TENSION_TARGET_SOURCE*4	0: R2R_Run_uiTensionTargetValue 1: AVI 2: ACI (R2R_Run_uiTensionTargetValue)	When *bExecute* turns to True and *bBusy* is False
TensionTargetSource_AtZeroSpeed	Source of zero-speed tension setting	R2R_TENSION_TARGET_SOURCE_AT_ZERO_SPEED*5	0: Disable 1: R2R_Run_uiTensionTargetValue_AtZeroSpeed 2: AVI 3: ACI (Disable)	When *bExecute* turns to True and *bBusy* is False
PID_TargetSource	PID target source	R2R_PID_TARGER_SOURCE*6	0: R2R_Run_lrPID_TargetValue 1: AVI 2: ACI (R2R_Run_lrPID_TargetValue)	When *bExecute* turns to True and *bBusy* is False
PID_FeedbackSource	PID feedback source	R2R_PID_FEEDBACK_SOURCE*7	0: AVI 1: ACI 2: MI6MI7 (AVI)	When *bExecute* turns to True and *bBusy* is False
PID_AdaptabilityReferenceSource	Tension PID reference basis	R2R_PID_ADAPTABILITY_REFERENCE_SOURCE*8	0: Disable 1: RollDiameter 2: Freq (Disable)	When *bExecute* turns to True and *bBusy* is False
lrPID_OutputPositiv	Tension PID	LREAL	0~655.35	When *bExecute*

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
eLimit	positive output limit (%)		(20.0)	turns to True and *bBusy* is False
lrPID_OutputNegativeLimit	Tension PID negative output limit (%)	LREAL	0~655.35 (1.0)	When *bExecute* turns to True and *bBusy* is False
lrForwardSpeedLimit	Torque mode forward speed limit (%)	LREAL	0~120 (10)	When *bExecute* turns to True and *bBusy* is False
lrReverseSpeedLimit	Torque mode reverse speed limit (%)	LREAL	0~120 (10)	When *bExecute* turns to True and *bBusy* is False

***Note**:

1. R2R_TENSION_CTRL_MODE: Enumeration (Enum)
2. R2R_WINDING_MODE: Enumeration (Enum)
3. R2R_LINE_SPEED_SOURCE: Enumeration (Enum)
4. R2R_TENSION_TARGET_SOURCE: Enumeration (Enum)
5. R2R_TENSION_TARGET_SOURCE_AT_ZERO_SPEED: Enumeration (Enum)
6. R2R_PID_TARGER_SOURCE: Enumeration (Enum)
7. R2R_PID_FEEDBACK_SOURCE: Enumeration (Enum)
8. R2R_PID_ADAPTABILITY_REFERENCE_SOURCE: Enumeration (Enum)

- **Outputs**

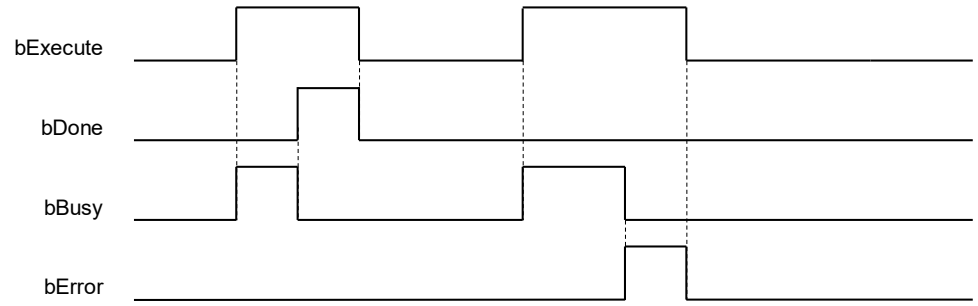
Name	Function	Data Type	Setting Value (Default Value)
bDone	True when complete to write the parameters.	BOOL	True/False (False)
bBusy	True when the instruction is running.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML _ERROR (DML _NO_ERROR)

***Note:** DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the movement resumes.	• When *bExecute* turns to False. • When *bExecute* is False but *bDone* turns to True, *bDone* will remain True for one scan cycle and then change to False.
bBusy	• When *bExecute* turns to TRUE.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded in *ErrorID*).	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When *bExecute* turns to True and *bBusy* is False

***Note**: AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

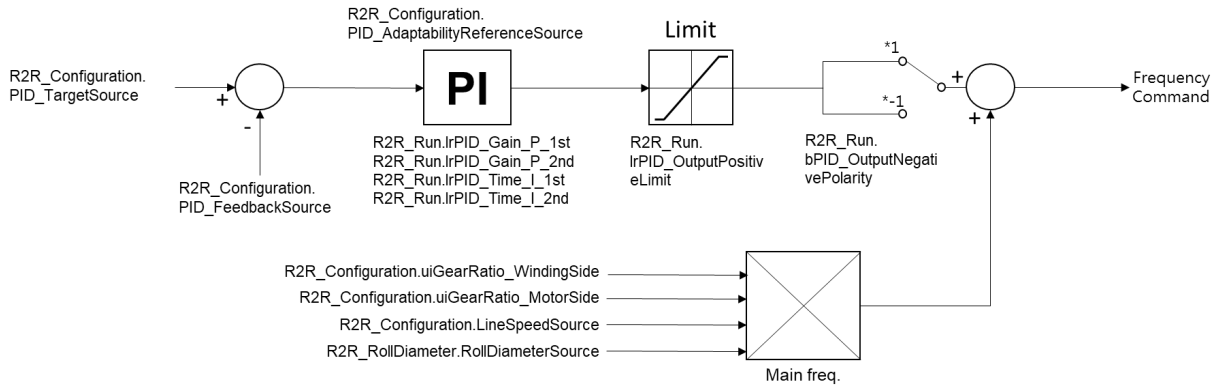
  - This function is only available when DL_ MotionControlLight is V1.2.0.0 or later.
  - This instruction is used to set the parameters related to tension control. Before running tension control, we must use this function block to configure settings.
  - There are 4 modes of TensionCtrlMode. The following describles each mode architecture.
    - TensionCloseLoop_SpeedMode (tension closed-loop control, speed mode)
      - In this mode, the driver is set to speed mode for tension control.

      Tension control main frequency: $f(Hz) = \frac{V}{\pi D} \cdot \frac{A}{B}$

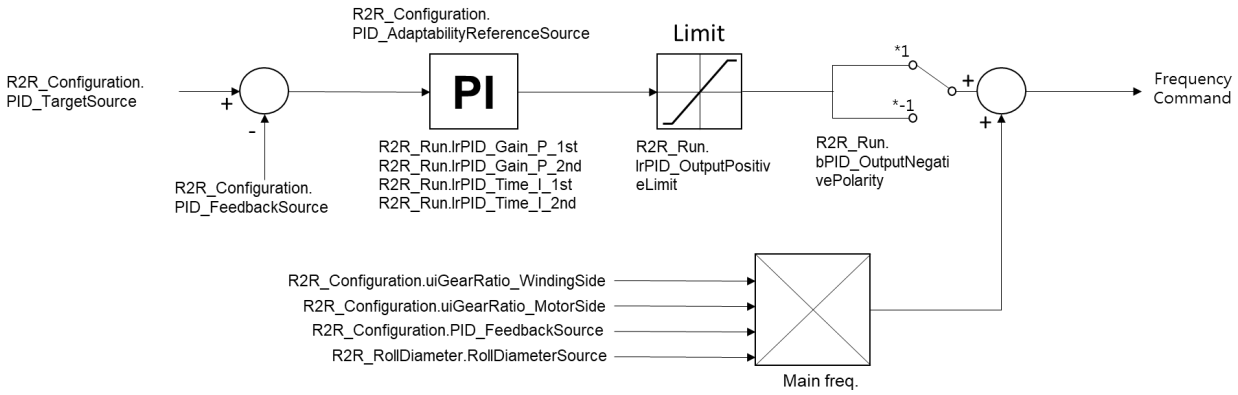
      V: Line speed (m/min.)

      D: Reel diameter (m)

      A/B: Mechanical gear ratio



    - LineSpeedCloseLoop_SpeedMode (Line speed closed-loop control, speed mode)
      - In this mode, the driver is set to speed mode for tension control.



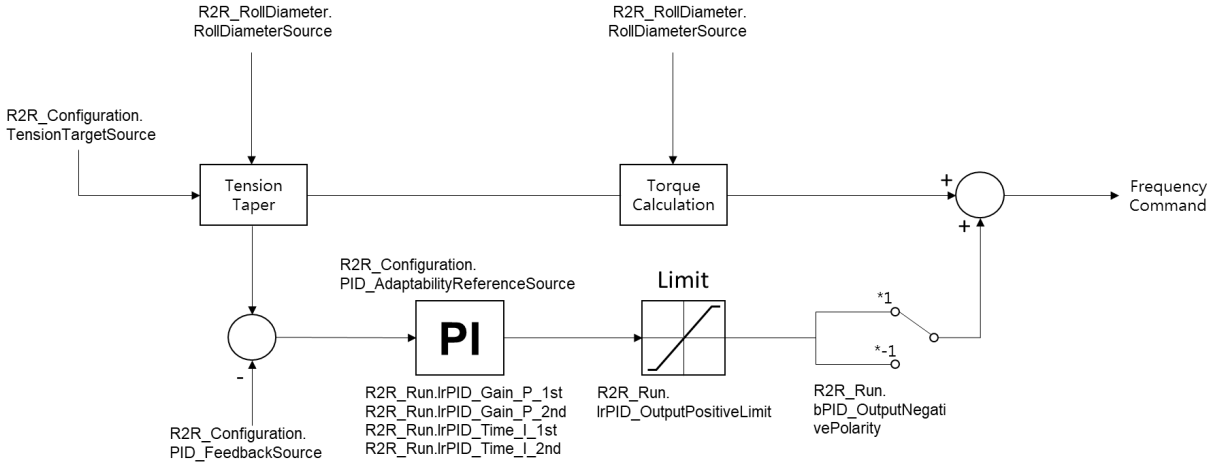
◆ TensionCloseLoop_TorqueMode (Tension closed-loop control, torque mode)

- In this mode, the driver is set to torque mode for tension control.

F: Tension (N)

D: Reel diameter (m)

$$\text{Torque (N-m)} = \frac{F \cdot D}{2}$$



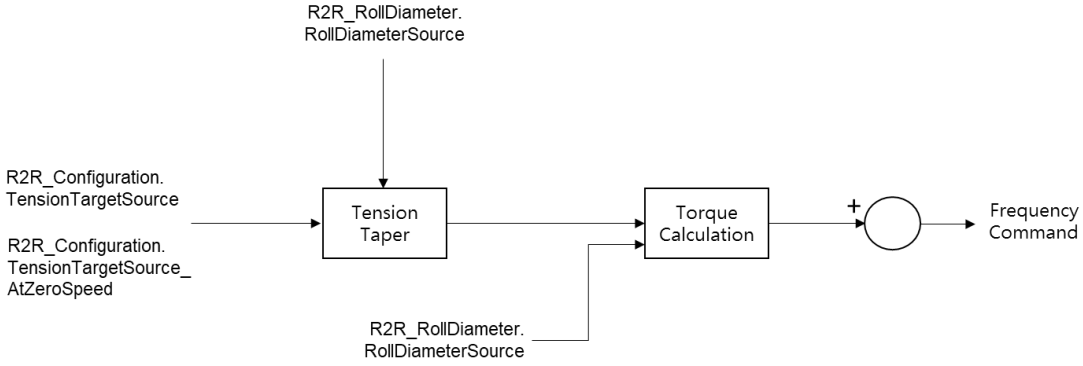
◆ TensionOpenLoop_TorqueMode (Tension open-loop control, torque mode)

- In this mode, the driver is set to torque mode for tension control.

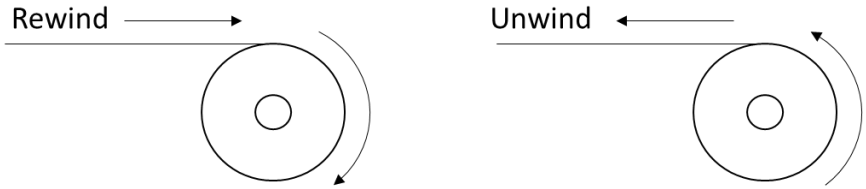
F: Tension (N)

D: Reel diameter (m)

$$\text{Torque } (N-m) = \frac{F \cdot D}{2}$$

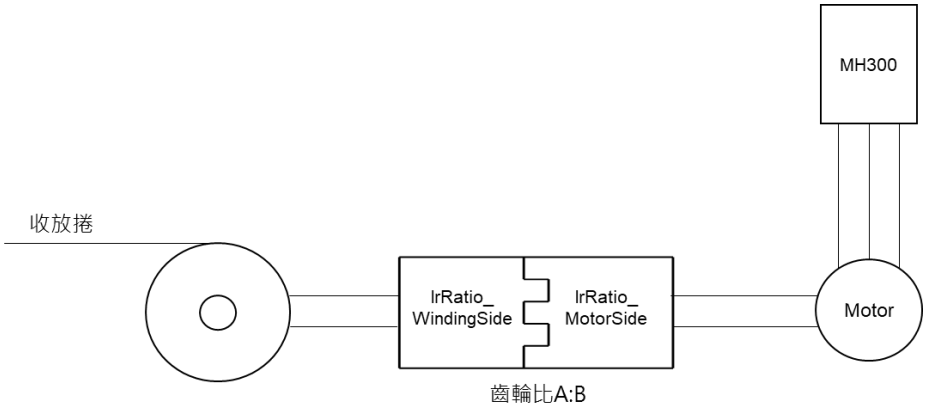


- WindingMode has Rewind and Uwind modes.



Note: When the winding mode is selected, the reel diameter (D) will increment; When the unwinding mode is selected, the reel diameter (D) decreases. As shown in the image.

- Use uiGearRatio_WindingSide and uiGearRatio_MotorSide parameters to set the gear ratio.



- Select PG_CARD in the LineSpeedSource input parameter. You must install the EMM-PG01x accessory card on the MH300 driver. Refer to *MH300 Series User Manual* Chapter 8 for details.
- MH300 driver supports the following signal inputs. Refer to *MH300 Series User Manual* Chapter 6 for the hardware configuration of signals.
  - DFM_DMC: Pulse voltage as output monitoring signal
  - AVI: Analog voltage frequency instruction, ACI: Analog current frequency instruction
  - MI6MI7: Frequency command function
- The PulseInput of PID_FeedbackSource refers to MI6MI7.
- The axis group state must be Disabled to run this function block.
- This instruction is only supported by Delta MH300 with EtherCAT (CMM-EC02) communication card.

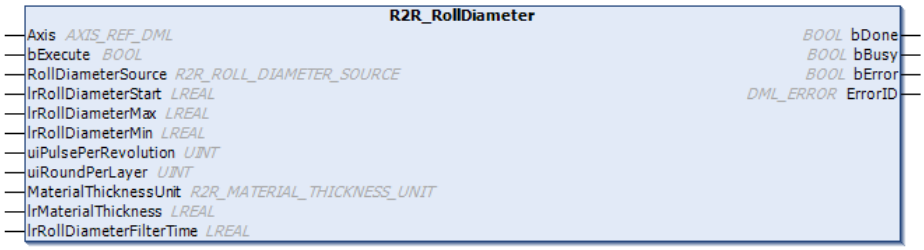
- **Troubleshooting**
  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the Error Code in ErrorID.

- **Example**
  - Refer to R2R_Run function block.

2.3.4.2 R2R_RollDiameter

- **Supported Devices**: AX-308E, AX-364E

R2R_RollDiameter sets the roll diameter.

FB/FC	Instruction	Graphic Expression
FB	R2R_RollDiameter	

ST Language

```
R2R_ RollDiameter_instance(
Axis: = ,
bExecute: = ,
RollDiameterSource: = ,
lrRollDiameterStart: = ,
lrRollDiameterMax: = ,
lrRollDiameterMin: = ,
uiPulsePerRevolution: = ,
uiRoundPerLayer: = ,
MaterialThicknessUnit: = ,
lrMaterialThickness: = ,
lrRollDiameterFilterTime: = ,
bDone=> ,
bBusy=> ,
bError=> ,
ErrorID=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	The instruction will be run when *bExecute* changes from False to True.	BOOL	True/False (False)	-

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
RollDiameter Source	Roll diameter source	R2R_ROLL_DIAMETER_SOURCE*1	0: R2R_Run_lrLineSpeedValue 1: AVI 2: ACI 3: ThicknessIntegrate_Motor_Encorder_PG1 4: ThicknessIntegrate_Motor_Encorder_PG2 5: ThicknessIntegrate_Motor_Encorder_MI67 6: ThicknessIntegrate_Motor_CloseSW_MI7 7: ThicknessIntegrate_Winding_Encorder_PG2 8: ThicknessIntegrate_Winding_Encorder_MI67 9: ThicknessIntegrate_Winding_CloseSW_MI7 (LineSpeed)	When *bExecute* turns to True and *bBusy* is False
lrRollDiameterStart	Current roll diameter (mm)	LREAL	1.0~6000.0 (6000.0)	When *bExecute* turns to True and *bBusy* is False
lrRollDiameterMax	Maximum roll diameter (mm)	LREAL	1.0~6000.0 (6000.0)	When *bExecute* turns to True and *bBusy* is False
lrRollDiameterMin	Empty roll diameter (mm)	LREAL	1.0~6000.0 (1.0)	When *bExecute* turns to True and *bBusy* is False
uiPulsePerRevolution	Number of pulses per revolution (Pulse/lr)	UINT	1~60000 (1)	When *bExecute* turns to True and *bBusy* is False
uiRoundPerLayer	Number of rounds per layer (round/layer)	UINT	1~10000 (1)	When *bExecute* turns to True and *bBusy* is False
MaterialThicknessUnit	Material thickness multiple	R2R_MATERIAL_THICKNESS_GAIN*2	0: millimeter 1: centimeter (millimeter)	When *bExecute* turns to True and *bBusy* is False
lrMaterialThickness	Material thickness (mm)	LREAL	0.001~65.0 (0.001)	When *bExecute* turns to True and *bBusy* is False
lrRollDiameter	Roll diameter filter time (s)	LREAL	0~100.0	When *bExecute* turns

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
rFilterTime			(1.0)	to True and *bBusy* is False

***Note**:
1. R2R_ROLL_DIAMETER_SOURCE: Enumeration (Enum)
2. R2R_MATERIAL_THICKNESS_GAIN: Enumeration (Enum)

- **Outputs**

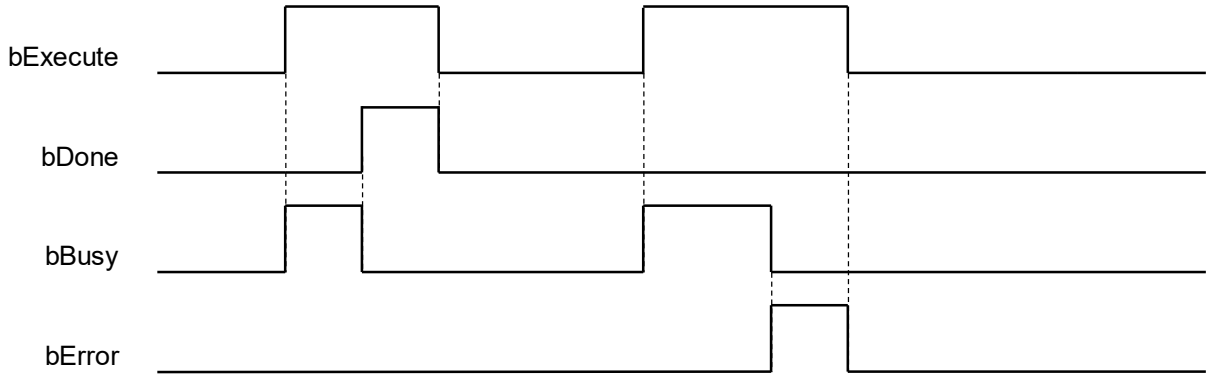
Name	Function	Data Type	Setting Value (Default Value)
bDone	True when complete to write the parameters.	BOOL	True/False (False)
bBusy	True when the instruction is running.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML _ERROR (DML _NO_ERROR)

***Note:** DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	• When the movement resumes.	• When *bExecute* turns to False. • When *bExecute* is False but *bDone* turns to True, *bDone* will remain True for one scan cycle and then change to False.
bBusy	• When *bExecute* turns to TRUE.	• When *bDone* turns to True. • When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded in *ErrorID*).	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		

- **Timing Diagram**



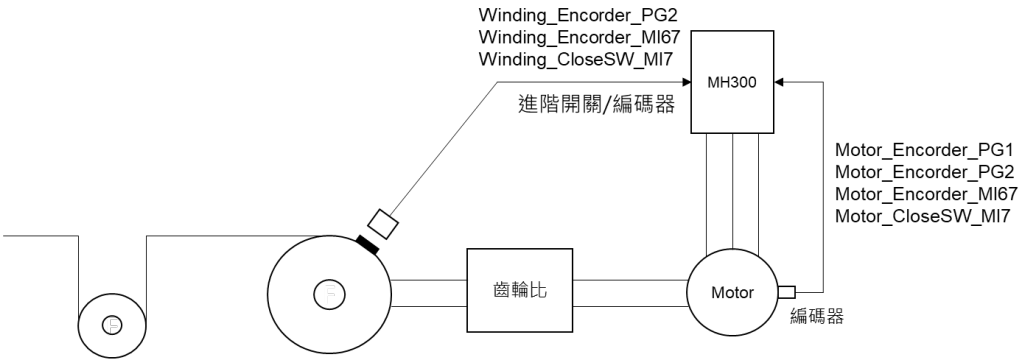
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When bExecute turns to True and *bBusy* is False

***Note**: AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

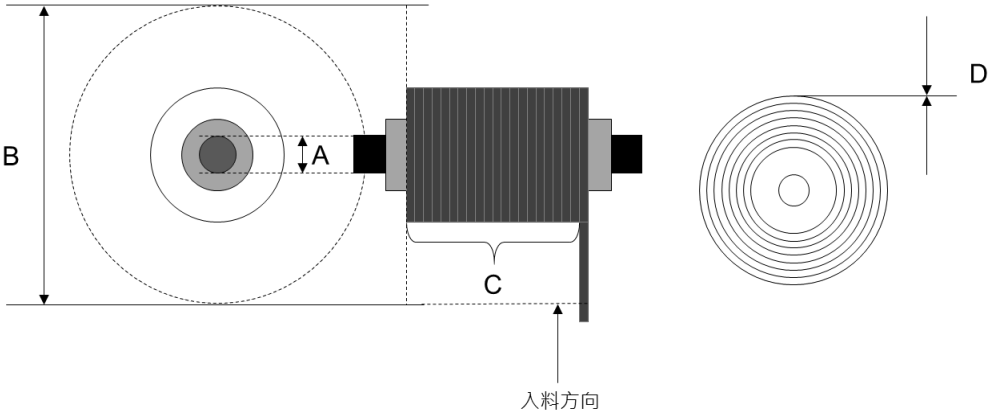
- **Function**

  - This function is only available when DL_ MotionControlLight is V1.2.0.0 or later.
  - This instruction is an industry function block, which can set the relevant parameters of the roll diameter.
  - When RollDiameterSource is set to "ThicknessIntegrate_Motor_Encorder_PG1", "ThicknessIntegrate_Motor_Encorder_PG2", and "ThicknessIntegrate_Winding_Encorder_PG2" mode, PG card is required.
  - This function is only available when the axis status is Disabled.
  - This instruction is only supported by Delta MH300 with EtherCAT (CMM-EC02) communication card.
  - The Thickness Aggregate parameter of RollDiameterSource has many types. Refer to the following:

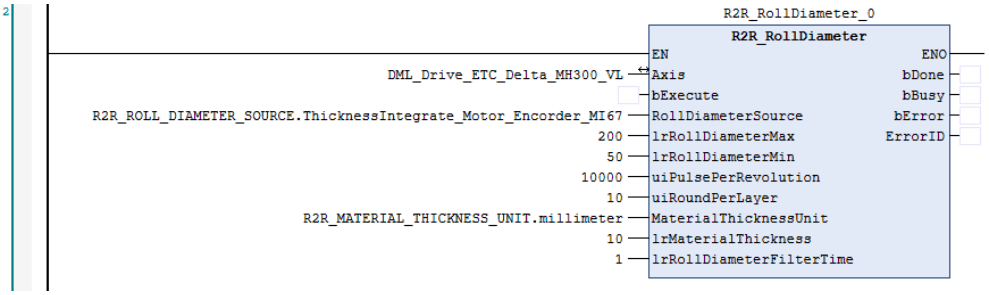


- **Troubleshooting**
  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the Error Code in ErrorID.

- **Example**
  - This example shows how to use R2R_RollDiameter to set the roll diameter parameter.



    - A: Empty roll diameter: 50 mm
    - B: Maximum roll diameter: 200 mm
    - C: Number of pulses per revolution: 10000, Number of rounds per layer: 10
    - D: Material thickness: 10 mm

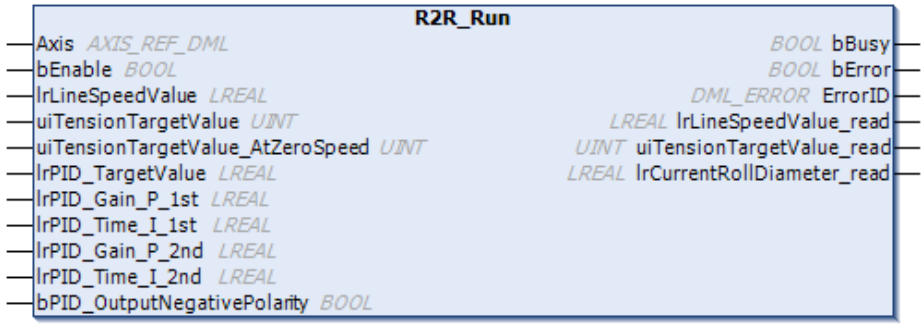


  - Set the corresponding parameters according to the organization parameters, and then run this function. When *bDone* of the R2R_RollDiameter_0 is True, parameters have been written to the driver.

2.3.4.3 R2R_Run

- **Supported Devices**: AX-308E, AX-364E

R2R_Run activates the tension control function.

FB/FC	Instruction	Graphic Expression
FB	R2R_Run	

ST Language

```
R2R_Run_instance(
Axis: = ,
bEnable: = ,
lrLineSpeedValue: = ,
uiTensionTargetValue: = ,
uiTensionTargetValue_AtZeroSpeed: = ,
lrPID_TargetValue: = ,
lrPID_Gain_P_1st: = ,
lrPID_Time_I_1st: = ,
lrPID_Gain_P_2nd: = ,
lrPID_Time_I_2nd: = ,
bPID_OutputNegativePolarity: = ,
bBusy=> ,
bError=> ,
ErrorID=> ,
lrLineSpeedValue_read=> ,
uiTensionTargetValue_read=> ,
lrCurrentRollDiameter_read=> );
```

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	The instruction will be run when *bExecute* changes from False to True.	BOOL	True/False (False)	-

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrLineSpeedValue	Current line speed (m/min)	LREAL	0.0~3000.0 (0)	When *Enable* turns to True
uiTensionTargetValue	Tension command value (N)	UINT	0~65535 (0)	When *Enable* turns to True
uiTensionTargetValue_AtZeroSpeed	Zero speed tension value (N)	UINT	0~65535 (0)	When *Enable* turns to True
lrPID_TargetValue	PID target value (%)	LREAL	0~100 (50.0)	When *Enable* turns to True
lrPID_Gain_P_1st	Tension PID P gain 1(%)	LREAL	0.0~1000.0 (50.0)	When *Enable* turns to True
lrPID_Time_I_1st	Tension PID I integration time 1	LREAL	0.0~500.0 (1.0)	When *Enable* turns to True
lrPID_Gain_P_2nd	Tension PID P gain 2(%)	LREAL	0.0~1000.0 (50.0)	When *Enable* turns to True
lrPID_Time_I_2nd	Tension PID I integration time 2	LREAL	0.0~500.0 (1.0)	When *Enable* turns to True
bPID_OutputNegativePolarity*	Tension PID output status selection	BOOL	True/False (False)	When *Enable* turns to True

***Note**: When the bPID_OutputNegativePolarity is False, the PID output is positive.

- **Outputs**

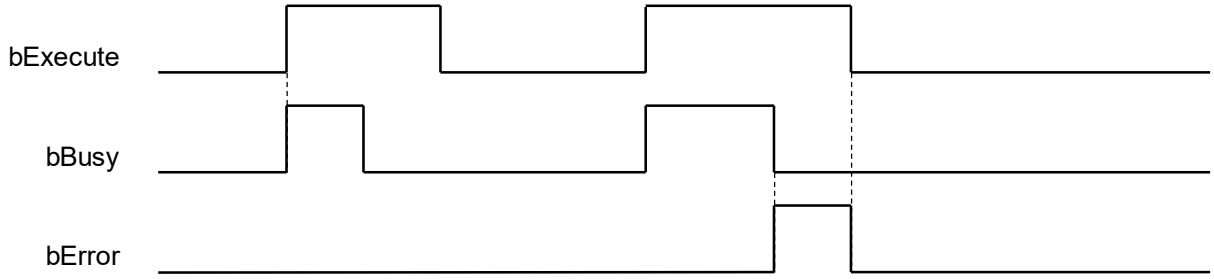
Name	Function	Data Type	Setting Value (Default Value)
bBusy	True when the instruction is running.	BOOL	True/False (False)
bError	True when an error occurs.	BOOL	True/False (False)
ErrorID	Record the error code when an error occurs. Refer to Appendix for error code descriptions.	DML_ERROR*	DML _ERROR (DML _NO_ERROR)
lrLineSpeedValue_read	Read the current line speed.	LREAL	正数或 0 (0)
uiTensionTargetValue_read	Read the current roll diameter.	UINT	正数或 0 (0)
lrCurrentRollDiameter_read	Read the tension command value.	LREAL	正数或 0 (0)

***Note:** DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bBusy	• When *bExecute* turns to TRUE.	• When *bError* turns to True.
bError	• When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded in *ErrorID*).	• When *bExecute* turns to False. (Error Code is cleared)
ErrorID		
lrLineSpeedValue_read	• Continuous update when *bBusy* is True.	• Stop updating when *bBusy* is False.
uiTensionTargetValue_read	• Continuous update when *bBusy* is True.	• Stop updating when *bBusy* is False.
lrCurrentRollDiameter_read	• Continuous update when *bBusy* is True.	• Stop updating when *bBusy* is False.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	Specify the axis.	AXIS_REF_DML*	AXIS_REF_DML	When bExecute turns to True and *bBusy* is False

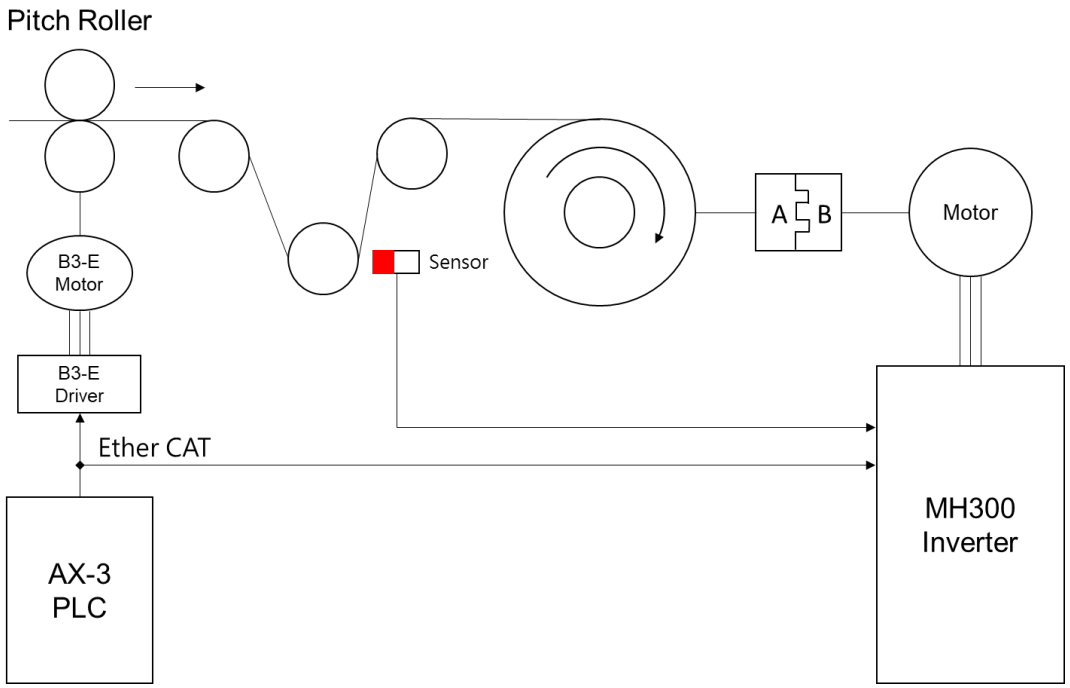
***Note**: AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**

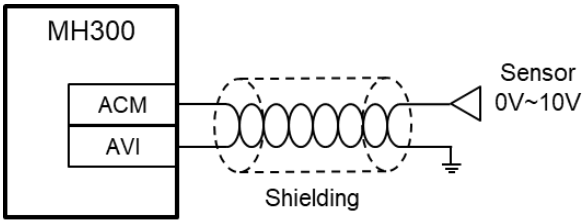
  - This function is only available when DL_ MotionControlLight is V1.2.0.0 or later.
  - This instruction is an industry function block that activates tension control.
  - If you perform this function, you need to change the axis status to Standstill, and if the function is successfully run, the axis status will be changed to Continuous.
  - This instruction is only supported by Delta MH300 with EtherCAT (CMM-EC02) communication card.
  - The way to set the bPID_OutputNegativePolarity can be given back by tension according to the different needs of customers. Refer to the following table below to select the appropriate method:

Name	Loose 0~100% Tight	Tight 0~100% Loose
Wind	Positive output	Negative output
Unwind	Negative output	Positive output

- **Troubleshooting**
  - When an error occurs in the execution of the instruction, *bError* will change to True. To confirm the current error state, see the Error Code in ErrorID.
- **Example**
  - Example 1: This example shows how to use the TensionCloseLoop_SpeedMode (Tension closed-loop control, speed mode) mode. Refer to the following device architecture:



- Sensor Wiring Diagram



- R2R_Configuration Parameters

Name	Value	Description
TensionCtrlMode	TensionCloseLoop_SpeedMode	Set the tension closed-loop speed mode
A (uiGearRatio_WindingSide)	200	Set the mechanical gears on the winding side A
B (uiGearRatio_MotorSide)	100	Set the mechanical gears on the motor side B
LineSpeedSource	R2R_Run_lrLineSpeedValue	Set the line speed source to the function block R2R_Run.lrLineSpeedValue
lrLineSpeedMax	500	Maximum line speed
PID_TargetSource	AVI	The hardware signal is the return voltage signal of the swing rod of the tractor (Pitch roller)

- R2R_RollDiameter Parameters

Name	Value	Description
RollDiameterSource	LineSpeed	Set the roll diameter source to line speed
lrRollDiameterStart	82	Set the current roll diameter
lrRollDiameterMax	6000	Set the maximum roll diameter
lrRollDiameterMin	80	Set the minimum roll diameter
MaterialThicknessUnit	millimeter	Material thickness multiple
lrMaterialThickness	0.001	Material thickness

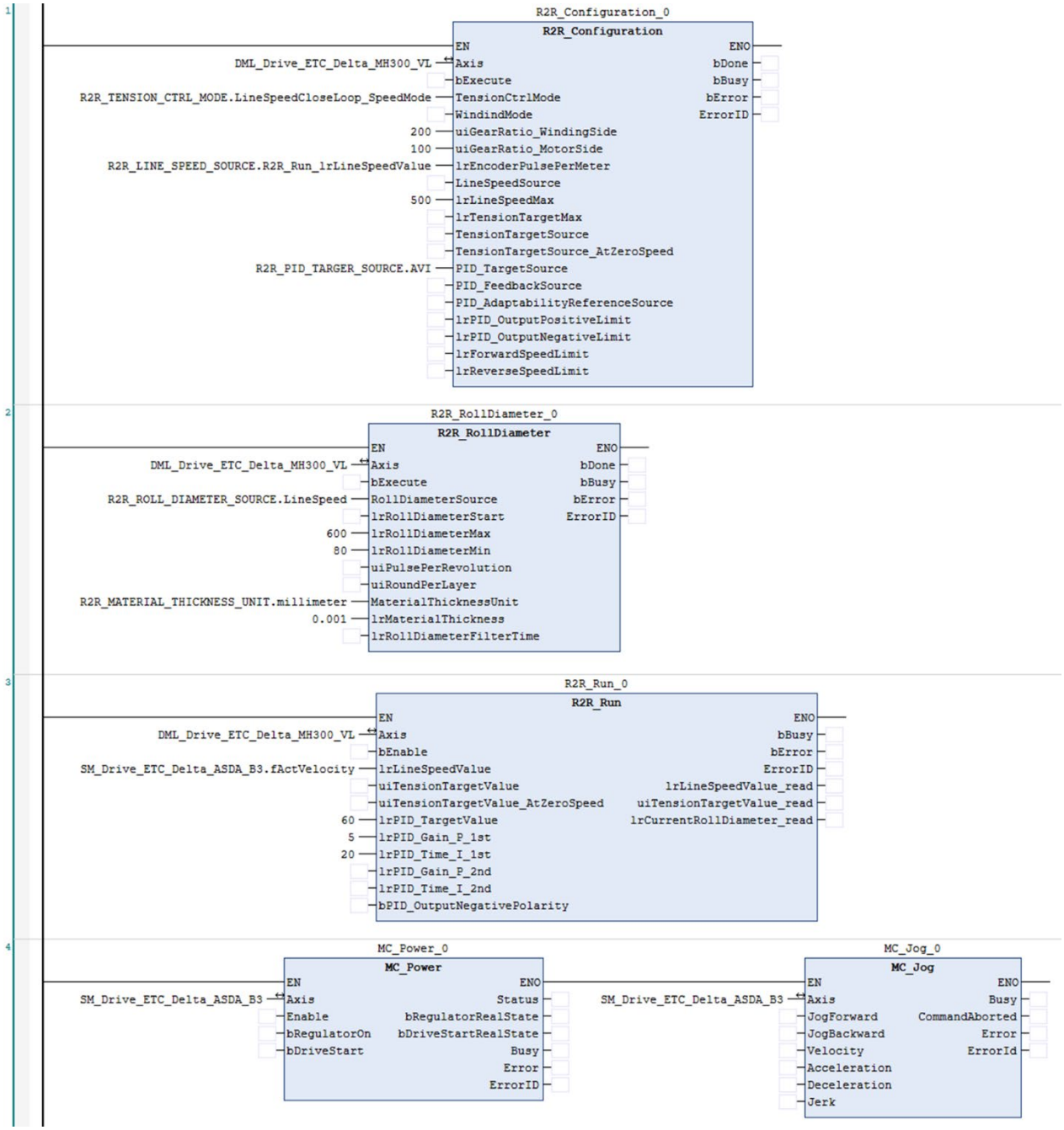
- R2R_Run Parameters

Name	Value	Description
lrPID_TargetValue	60	PID target value (%)
lrPID_Gain_P_1st	5	Tension PID P gain 1(%)
lrPID_Time_I_1st	20	Tension PID I integration time 1

- MH300 Parameters

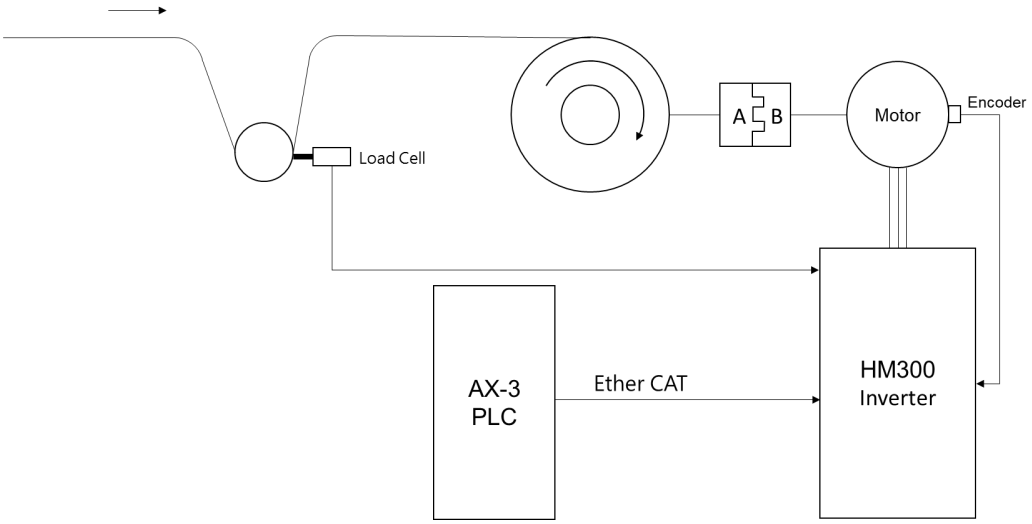
Name	Value	Description
03-28	0	Select AVI terminal input
03-29	1	Select ACI terminal input

- Tension closed-loop speed mode program

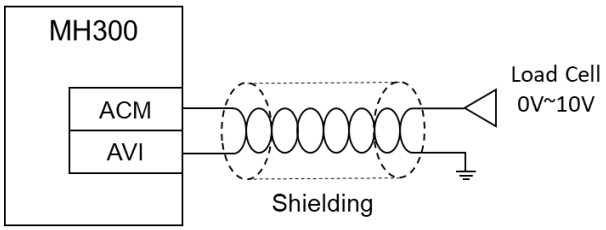


- Set the R2R_Configuration, R2R_RollDiameter, and R2R_Run parameters according to the above table.
- Starting sequence: start the R2R_Configuration, R2R_RollDiameter, and then start the R2R_Run.
- After the tension function is activated, start the feeding axis, and the line speed source of MH300 runs according to the line speed fed back by the feeding axis.

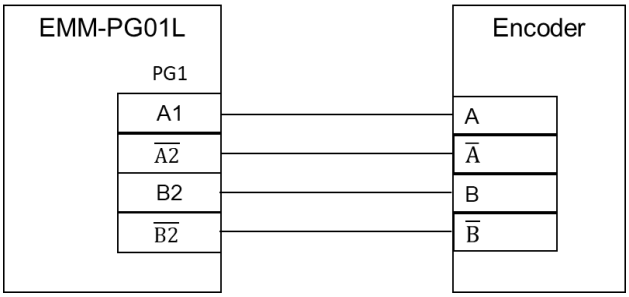
- Example 2: This example shows how to use the TensionCloseLoop_TorqueMode (Tension closed-loop, torque mode) mode. Refer to the following device architecture:



- Load Cell Wiring Diagram.



- EMM-PG01L Wiring Diagram



- R2R_Configuration Parameters

Name	Value	Description
TensionCtrlMode	TensionCloseLoop_TorqueMode	Set the tension closed-loop torque mode
A（uiGearRatio_WindingSide）	200	Set the mechanical gears on the winding side A
B（uiGearRatio_MotorSide）	100	Set the mechanical gears on the motor side B
lrLineSpeedMax	500	Maximum line speed
lrTensionTargetMax	350	Maximum tension value (N)
PID_TargetSource	AVI	The hardware signal is the Load Cell voltage signal of the tractor (Pitch roller).

- R2R_RollDiameter Parameters

Name	Value	Description
RollDiameterSource	LineSpeed	Set the roll diameter source to line speed

lrRollDiameterStart	82	Set the current roll diameter
lrRollDiameterMax	6000	Set the maximum roll diameter
lrRollDiameterMin	80	Set the minimum roll diameter
MaterialThicknessUnit	millimeter	Material thickness multiple
lrMaterialThickness	0.001	Material thickness

- R2R_Run Parameters

Name	Value	Description
uiTensionTargetValue	118	Tension command value (N)
uiTensionTargetValue_AtZeroSpeed	20	Zero speed tension value (N)
lrPID_TargetValue	50	PID target value (%)
lrPID_Gain_P_1st	1	Tension PID P gain 1(%)
lrPID_Time_I_1st	20	Tension PID I integration time 1

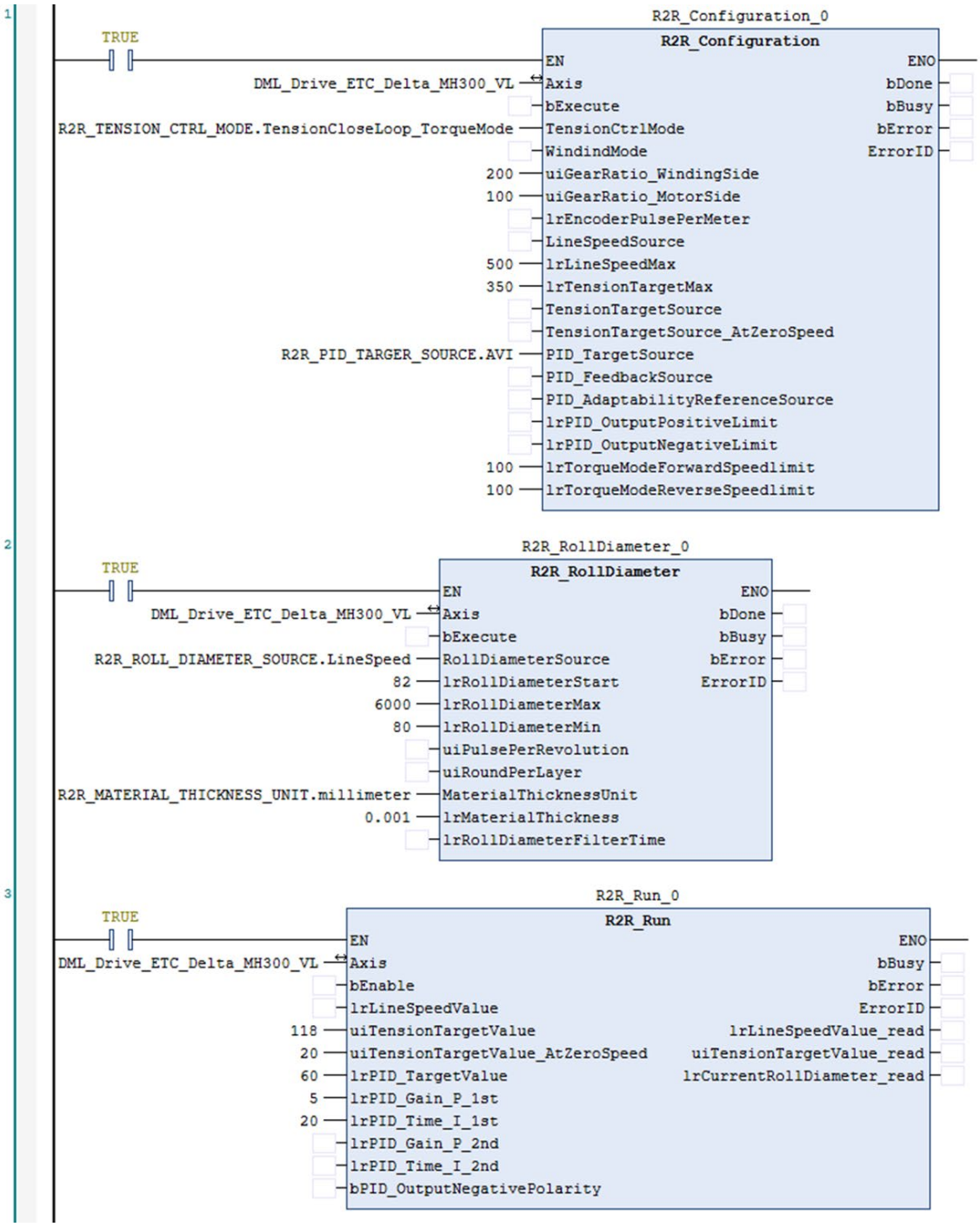
- Motor Settings Encoder Settings

Name	Value	Description
Encoder Type Selection	ABZ Pulse	Select the encoder type
Encoder Pulses Per Revolution	1024	Number of pulses per revolution of the encoder
Encoder Input Type Setting	FWD A Leads B	Set the encoder input type

- HM300 Parameters

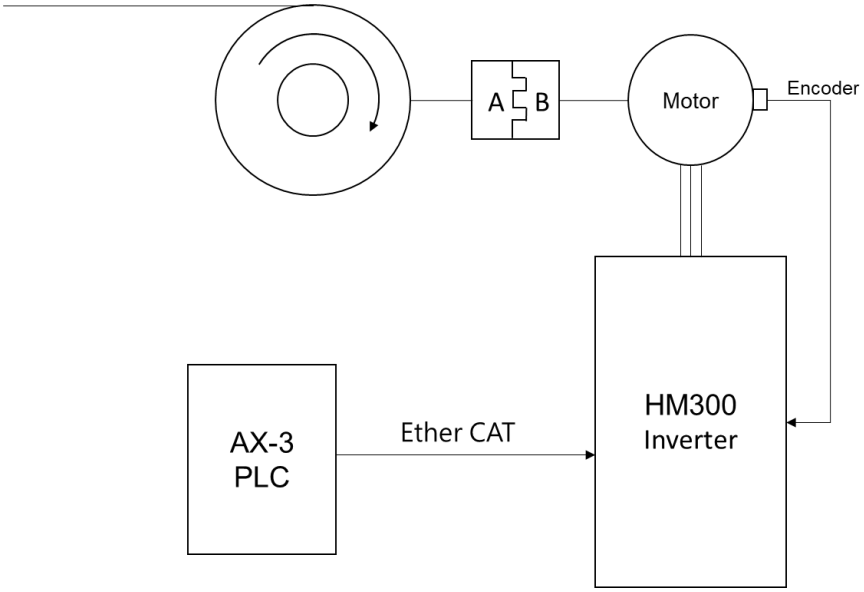
Name	Value	Description
03-28	0	Select AVI terminal input
10-16	0	Pulse input type setting
10-17	1	Electronic gears A
10-18	1	Electronic gears B

- Tension closed-loop torque mode program

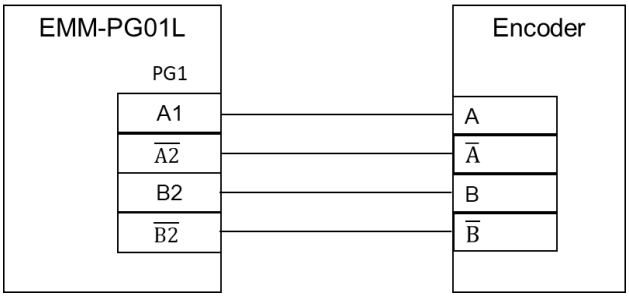


- Set the R2R_Configuration, R2R_RollDiameter, and R2R_Run parameters according to the above table.
- Starting sequence: start the R2R_Configuration, R2R_RollDiameter, and then start the R2R_Run.
- After the tension function is activated, it will run according to the tension value of the function block.

- Example 3: This example shows how to use the TensionOpenLoop_TorqueMode (Tension ope-loop, torque mode) mode. Refer to the following device architecture:



- EMM-PG01L Wiring Diagram



- R2R_Configuration Parameters

Name	Value	Description
TensionCtrlMode	TensionOpenLoop_TorqueMode	Set the tension open-loop torque mode
A（uiGearRatio_WindingSide）	200	Set the mechanical gears on the winding side A
B（uiGearRatio_MotorSide）	100	Set the mechanical gears on the motor side B
LineSpeedSource	R2R_Run_lrLineSpeedValue	Line speed input source
lrLineSpeedMax	500	Maximum line speed
lrTensionTargetMax	350	Maximum tension value (N)
TensionTargetSource	R2R_Run_uiTensionTargetValue	Select the tension command source

- R2R_RollDiameter Parameters

Name	Value	Description
RollDiameterSource	LineSpeed	Set the roll diameter source to line speed
lrRollDiameterStart	82	Set the current roll diameter
lrRollDiameterMax	6000	Set the maximum roll diameter
lrRollDiameterMin	80	Set the minimum roll diameter
MaterialThicknessUnit	millimeter	Material thickness multiple
lrMaterialThickness	0.001	Material thickness

- R2R_Run Parameters

Name	Value	Description
uiTensionTargetValue	118	Tension command value (N)
uiTensionTargetValue_AtZeroSpeed	20	Zero speed tension value (N)
lrPID_TargetValue	50	PID target value (%)
lrPID_Gain_P_1st	50	Tension PID P gain 1(%)
lrPID_Time_I_1st	1	Tension PID I integration time 1

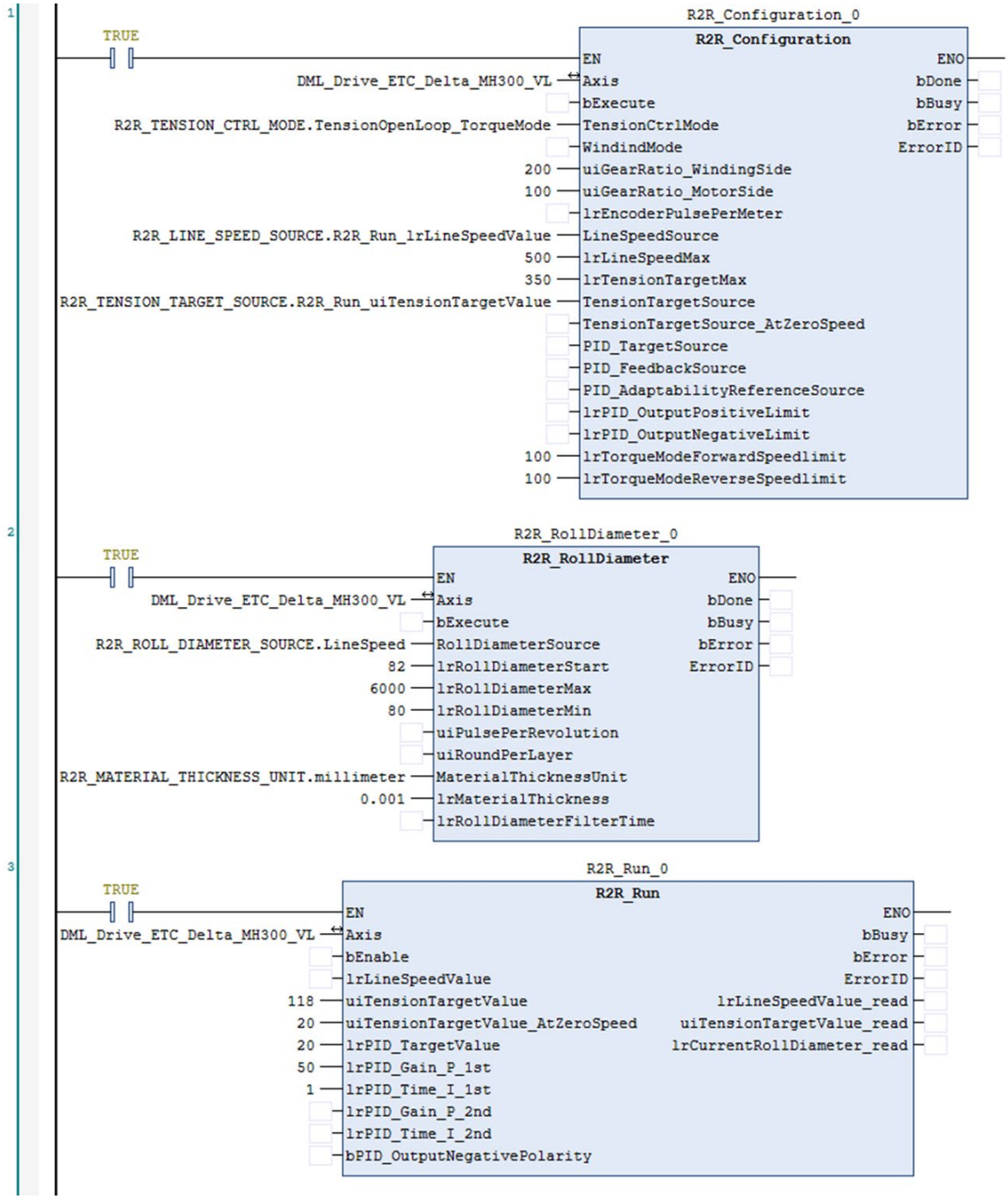
- Motor Settings Encoder Settings

Name	Value	Description
Encoder Type Selection	ABZ Pulse	Select the encoder type
Encoder Pulses Per Revolution	1024	Number of pulses per revolution of the encoder
Encoder Input Type Setting	FWD A Leads B	Set the encoder input type

- HM300 Parameters

Name	Value	Description
10-16	0	Pulse input type setting
10-17	1	Electronic gears A
10-18	1	Electronic gears B

- Tension open-loop torque mode program



- Set the R2R_Configuration, R2R_RollDiameter, and R2R_Run parameters according to the above table.
- Starting sequence: start the R2R_Configuration, R2R_RollDiameter, and then start the R2R_Run.
- After the tension function is activated, it will run according to the tension value of the function block. The line speed can be set by R2R_Run.lrLinSpeedValue to match the speed of the discharge axis.

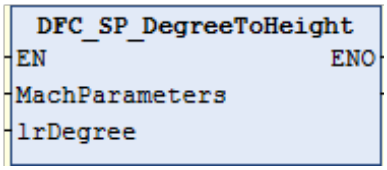
2.4 Industry-Specific Instructions

2.4.1 Servo Press Instruction

2.4.1.1 DFC_SP_DegreeToHeight

- **Supported Devices**: AX-308E, AX-8

Servo Press industry-specific input crankshaft length, connecting rod length and crankshaft angle, and the corresponding slider height are obtained by this Function. In motion control, the current height of the slider can be calculated from the current crankshaft angle, crankshaft length, and connecting rod length. When an invalid value is entered, the value 0 is returned.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFC_SP_DegreeToHeigh		_parameter : = DFC_SP_DegreeToHeight(MachParameters: = _parameter, lrDegree: = _parameter);

***Note:** If the input parameter range is invalid, the value of 0 will be output, indicating that no calculation will be performed.
Valid Input Range:
MachParameters.lrLLength > (2 * MachParameters.lrRLength)
0 <= lrDegree <= 360

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
MachParameters	ServoPress machine electromechanical parameters	Reference DMC_SP_MACHINE_PARAMETERS*2	-	-
lrDegree	The degree of the slave axis	LREAL*1	Positive(0)	When Execute turns to True and Busy state is False

***Note1**: LREAL Range:
Positive values: -1.7976931348623157E+308 to -4.9406564584124654E-324
Zero: 0
Negative values: 4.9406564584124654E-324 to 1.7976931348623157E+308

***Note2**: DMC_SP_MACHINE_PARAMETERS

Name	Function	Data Type	Setting Value
lrRLength	Crank length (unit: mm), provided by machine maker	LREAL	Positive
lrLLength	Link length (unit: mm), provided by machine maker	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's SPM. (six times lrSPMsys is equal to the	LREAL	Positive

	unit deg/sec)		
lrGearRatio	Reduction gear ratio	LREAL	Positive
lrRPMmotor	Max RPM of the motor	LREAL	Positive

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
lrHeight	The height of slider.	LREAL*1	0 < lrHeight < 2*lrRLength

***Note1**: LREAL Range:
Negative values: between -1.7976931348623157E+308 and -4.9406564584124654E-324
Zero: 0
Positive values: between 4.9406564584124654E-324 and 1.7976931348623157E+308

- **Output Update Timing**

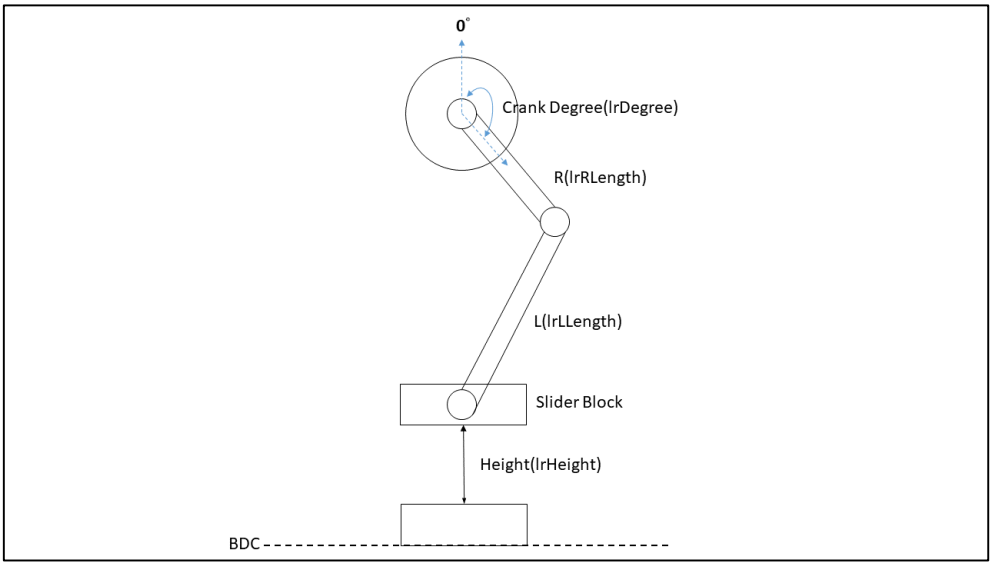
Name	Timing for shifting to True
-	-

- **Timing Diagram**
  -

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

- **Function**
  - Input MachParameters, and lrDegree. The function block will calculate out Slider Height (lrHeight).
  - In this case, invalid values are set, lrHeight will keep output 0.
  - Schematic diagram:

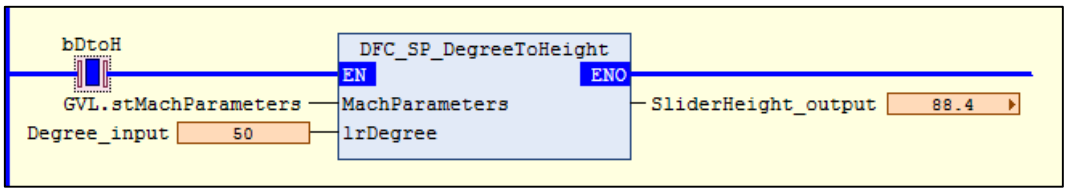


- **Troubleshooting**

-

- **Sample Program**

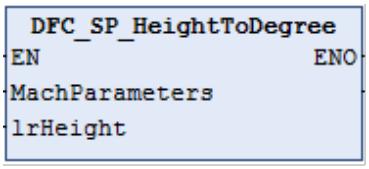
  Setup MachParameters.lrRLength = 50mm, MachParameters.lrLLength = 120mm, and lrDegree = 50, after that function will return lrHeight =88.4 (88.416296732977969).



2.4.1.2 DFC_SP_HeightToDegree

- **Supported Devices**: AX-308E, AX-8

A dedicated FB for the Servo Press industry, the user inputs the mechanical-related information (crankshaft length, connecting rod length, slider height) and uses this Function to obtain the corresponding angle (0 ~ 180); if the position has passed the bottom dead center (BDC, Bottom Dead Center), you can do your own mirroring (360.0 - return value). When an illegal parameter (ex: less than 0) is input, the angle cannot be obtained, and 0 is returned at this time.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFC_SP_HeightToDegree		_parameter : = DFC_SP_HeightToDegree(MachParameters: = _parameter, lrHeight: = _parameter)

***Note**: Valid Input Range:
MachParameters.lrLLength > (2 * MachParameters.lrRLength)
0 <= lrHeight <= (2 * MachParameters.lrRLength)

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
MachParameters	ServoPress machine electromechanical parameters	ReferenceDMC_SP_MACHIME_PARAMETERS*3	-	-
lrHeight	The height of slider	LREAL*1	Positive*2	-

***Note1**: LREAL Range:
Positive values: Between -1.7976931348623157E+308 and -4.9406564584124654E-324
Zero: 0
Negative values: Between 4.9406564584124654E-324 and 1.7976931348623157E+308

***Note2**: lrHeightValid Input Range:
0 ~ (2 * rfMachParameters.lrRLength)

***Note3**: DMC_SP_MACHINE_PARAMETERS

Name	Function	Data Type	Setting Value
lrRLength	Crank length (unit: mm), provided by machine maker	LREAL	Positive
lrLLength	Link length (unit: mm), provided by machine maker	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's SPM. (six times lrSPMsys is equal to the unit deg/sec)	LREAL	Positive
lrGearRatio	Reduction gear ratio	LREAL	Positive
lrRPMmotor	Max RPM of the motor	LREAL	Positive

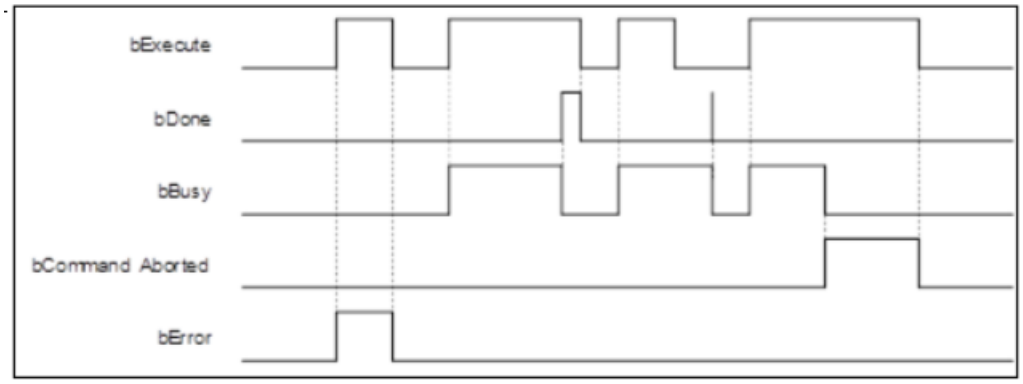
- **Output**

Name	Function	Data Type	Output Range (Default Value)
-	-	-	-

  - **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
-	-	-

- **Timing Diagram**

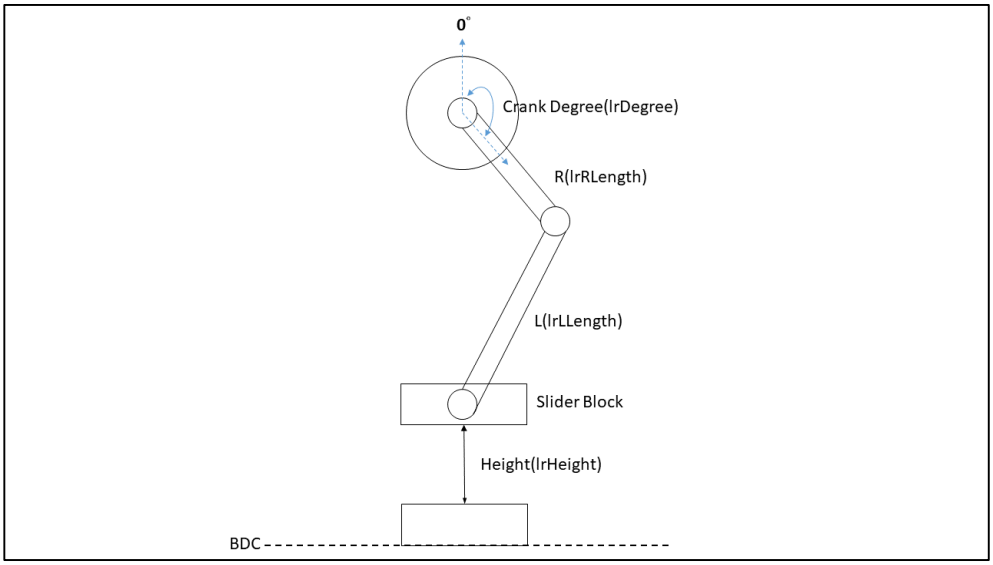


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

- **Function**

  - Input MachParameters, and lrHeight, then the function will calculate out lrDegree.
  - This function will be used on key points generation related function blocks for Servo Press application (e.g., DMC_SP_CamCrankCurve, DMC_SP_CamLinkCurve, DMC_SP_CamCoinCurve, etc).
  - Input valid range MachParameters.lrRLength, MachParameters.lrLLegnth, and lrHeight, then the function will calculate out lrDegree. In this case, invalid values are set, this function will keep returning 0.
  - Schematic diagram:

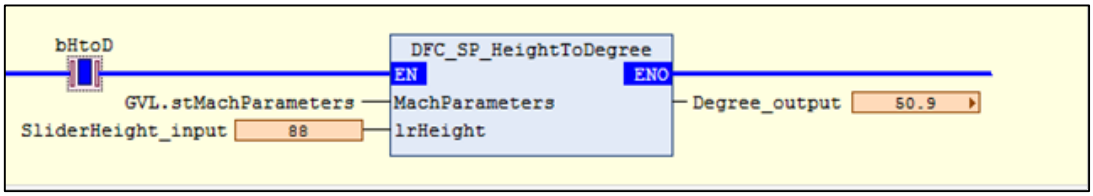


- **Troubleshooting**

  -

- **Sample Program**

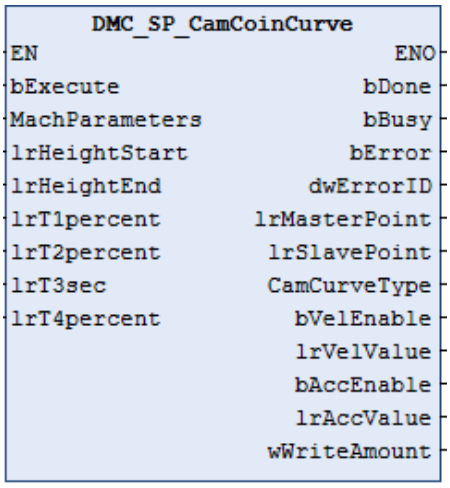
  Set up MachParameters.lrRLength = 50.0mm, MachParameters.lrLLength = 120.0mm, and lrHeight = 88.=mm. The function will return lrDegree = 50.9°(50.859…..).



2.4.1.3 DMC_SP_CamCoinCurve

- **Supported Devices**: AX-308E, AX-8

Servo Press industry-specific FB, generates cam key point information according to the imprint curve algorithm and generates an electronic cam table through DMC_SP_CamCurveWriteFunction block. Users can use the electronic cam table for Servo Press processing.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_SP_CamCoinCruve		DMC_SP_CamCoinCurve(bExecute: = _parameter, MachParameters: = _parameter, lrHeightStart: = _parameter, lrHeightEnd: = _parameter, lrT1percent: = _parameter, lrT2percent: = _parameter, lrT3sec: = _parameter, lrT4percent: = _parameter, bDone=> _parameter, bBusy=> _parameter, bError=> _parameter, dwErrorID=> _parameter, lrMasterPoint=> _parameter, lrSlavePoint=> _parameter, CamCurveType=> _parameter, bVelEnable=> _parameter, lrVelValue=> _parameter, bAccEnable=> _parameter, lrAccValue=> _parameter, wWriteAmount=> _parameter);

- **Inputs**

Name	Function	Data Type	Setting Range (Default)	Timing to Take Effect
bExecute	Run the function block control bit	BOOL	False/True (False)	-
MachParameters	ServoPress machine electromechanical parameters	Reference DMC_SP_MACHIME_PARAMETERS*2	-	*bBusy*=FALSE & *bExecute* is at rising-edge
lrHeightStart	Process start position - the height of slider	LREAL*1	Positive (0) [Range] 0 < lrHeightStart < (2 * lrRLength)	*bBusy*=FALSE & *bExecute* is at rising-edge
lrHeigthEnd	Process end position – the height of slider	LREAL*1	Positive (0) [Range] 0 < lrHeightStart < (2 * lrRLength)	*bBusy*=FALSE & *bExecute* is at rising-edge
lrT1percent	T1 percentage of time	LREAL*1	Positive (0) [Range] 0 < lrT1percent < 100	*bBusy*=FALSE & *bExecute* is at rising-edge
lrT2percent	T2 percentage of time	LREAL*1	Positive (0) [Range] 0 < lrT2percent < 100	*bBusy*=FALSE & *bExecute* is at rising-edge
lrT3sec	T3 seconds	LREAL*1	Positive(0) [Range] 0 < lrT3Ssec < (60/MachParameters.lrSPMsys)	*bBusy*=FALSE & *bExecute* is at rising-edge
lrT4percent	T4 percentage of time	LREAL*1	Positive(0) [Range] 0 < lrT4percent < 100	*bBusy*=FALSE & *bExecute* is at rising-edge

***Note1**: LREAL Range:
Positive values: -1.7976931348623157E+308 to -4.9406564584124654E-324
Zero: 0
Negative values: 4.9406564584124654E-324 to 1.7976931348623157E+308

***Note2**: DMC_SP_MACHINE_PARAMETERS

Name	Function	Data Type	Setting Value
lrRLength	Crank length (unit: mm), provided by machine maker	LREAL	Positive
lrLLength	Link length (unit: mm), provided by machine maker	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's SPM. (six times lrSPMsys is equal to the unit deg/sec)	LREAL	Positive
lrGearRatio	Reduction gear ratio	LREAL	Positive
lrRPMmotor	Max RPM of the motor	LREAL	Positive

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)

bDone	Status when FB finish generating coin key-points	BOOL	False/True (False)
bBusy	Status when FB is busy to generate key-points	BOOL	False/True (False)
bError	Status when FB has problems generating key-points	BOOL	False/True (False)
dwErrorID	Error Code when running the FB	DWORD	16#00000000~16#FFFFFFFF(16#00000000)
lrMasterPoint	Master positions for DMC_SP_CamCurveWrite	[0..63] LREAL Array	0.0 ~ 360.0(0.0)
lrSlavePoint	Slave positions for DMC_SP_CamCurveWrite	[0..63] LREAL Array	0.0 ~ 360.0(0.0)
CamCurveType	Key point to key point cruve type for DMC_SP_CamCurveWrite	[0..62] DMC_CamCurveType Array	0 ~ 9*2(0)
bVelEnable	Velocity Enable Configurations for DMC_SP_CamCurveWrite	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrVelValue	Velocity Value for DMC_SP_CamCurveWrite	[0..63] LREAL Array	LREAL*1(0)
bAccEnable	Acceleration Eanble Configurations for DMC_SP_CamCurveWrite	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrAccValue	Accerleration Value for DMC_SP_CamCurveWrite	[0..63] LREAL Array	LREAL*1(0)
wWriteAmount	Key Point write amount for DMC_SP_CamCurveWrite	WORD	0~64(0)

***Note1**: LREAL Range:
Positive values: -1.7976931348623157E+308 to -4.9406564584124654E-324
Zero: 0
Negative values : 4.9406564584124654E-324 to 1.7976931348623157E+308

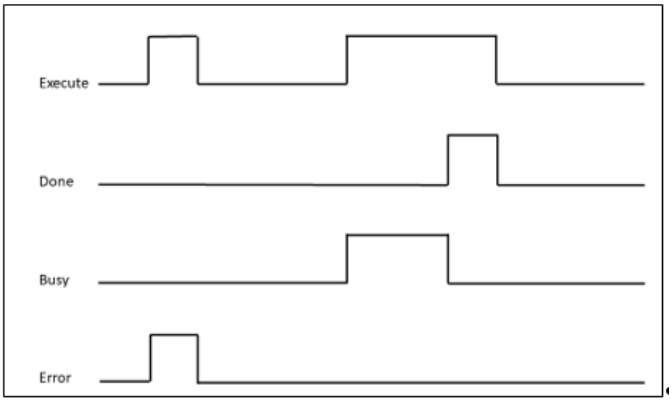
***Note2**: Curve Type (0 ~ 9)
Straight line (0)
Quadratic Parabola (1)
5th Polynomial (2)
Basic Sine (3)
Inclined Sine (4)
Mod_Acc Sine (5)
Mod_Acc Trapezoidal (6)
Cubic_Spline_Nature (7)
Cubic_Spline_Clamp (8)
Cubic_Spline(9)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	bExecute=TRUE + FB finish curve key points generation	• bExecute=FALSE • bExecute=TRUE + FB is Error

		bExecute=TRUE + FB is busy
bBusy	bExecute=TRUE + FB is generating curve key points	• bExecute=FALSE • bExecute=TRUE + FB is error bExecute=TRUE + FB is done
bError	bExecute=TURE + FB raise error code(non-zero)	• bExecute=FALSE • bExecute=TRUE + FB is busy bExecute=TRUE + FB is done

- **Timing Diagram**

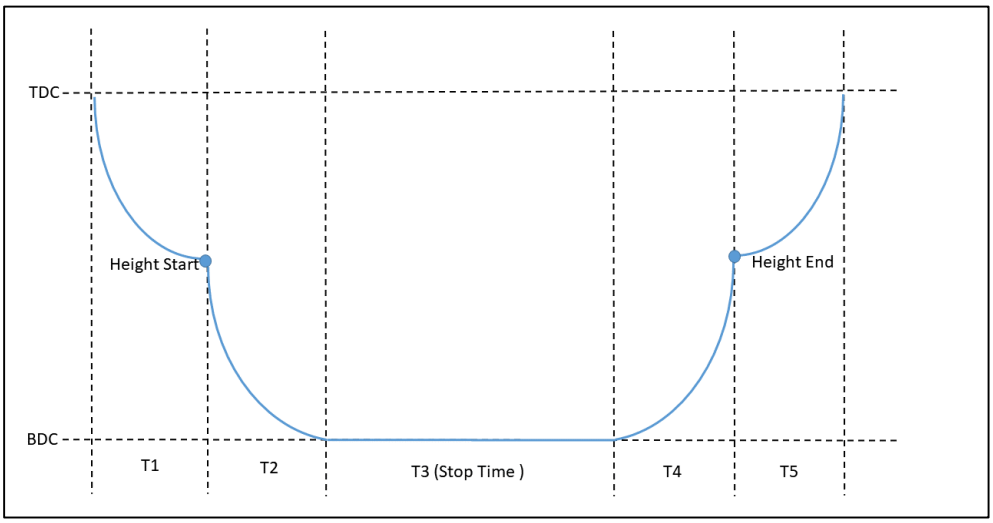


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

- **Function**

  - To generate key-points information for Servo Press coin mode, the data should be entered into DMC_SP_CamCurveWrite.
  - Invoke DMC_SP_CamCurveWrite to generate and write ECAM table after this FB is done.
  - Coin Curve Setup Page:

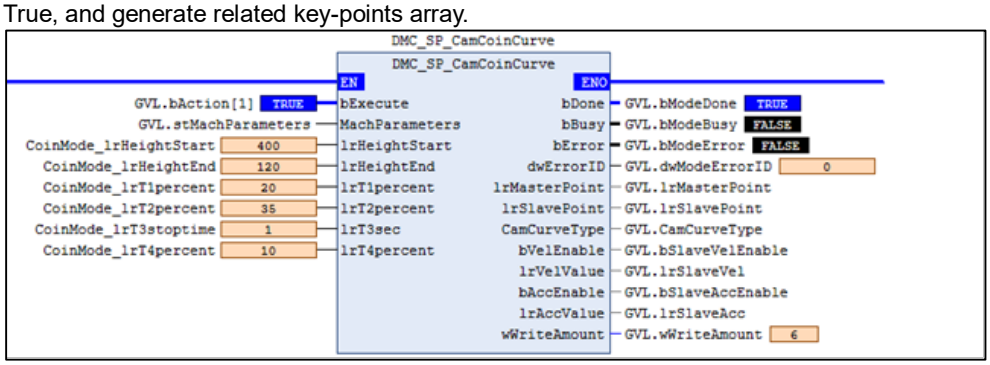


- **Troubleshooting**

When error happens, *bError* will be True and dwErrorID will show non-zero. Refer to the following table for your next step.

Error Code	Description	Contents	Corrective Action
0	No Error	-	-
1	Invalid machine parameters	Invalid electromechanical parameters were set	Check if these values are within the range • MachParameters.lrRLength • MachParameters.lrLLength • MachParameters.lrSPMsys
2	Invalid slider height	Incorrect range of slider height was set	Check if the values are within the range • lrHeightStart • lrHeightEnd
3	Invalid time percentage	Invalid time percentage was set (0% or sum of time percentage exceeds 100%)	Check if the values are within the range • lrT1percent • lrT2percent • lrT3sec • lrT4percent

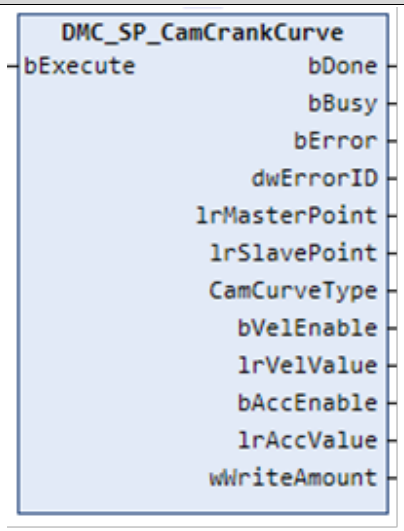
- **Sample Program**
  - Set MachParameters.lrLlength=900, MachParameters lrRlength=300, lrHightStart=400, lrHightEnd=120, lrT1_percent=20, rT2_percent=35 lrT3_StopTime=1, lrT4_percent=10
  - Set *bExecute* from False to True to trigger curve key-points calculation. Once calculation completes, *bDone* will be True, and generate related key-points array.



2.4.1.4 DMC_SP_CamCrankCurve

- **Supported Devices:** AX-308E, AX-8

Servo Press industry-specific FB, generates cam key point information according to the crankshaft curve algorithm, and generates an electronic cam table through DMC_SP_CamCurveWrite FB. Users can use the electronic cam table for Servo Press processing.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_SP_CamCrankCurve		DMC_SP_CamCrankCurve(bExecute: = _parameter, bDone=> _parameter, bBusy=> _parameter, bError=> _parameter, dwErrorID=> _parameter, lrMasterPoint=> _parameter, lrSlavePoint=> _parameter, CamCurveType=> _parameter, bVelEnable=> _parameter, lrVelValue=> _parameter, bAccEnable=> _parameter, lrAccValue=> _parameter, wWriteAmount=> _parameter);

- **Inputs**

Name	Function	Data Type	Setting Range (Default)	Timing to Take Effect
bExecute	Run the function block control bit	BOOL	False/True (False)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	Status when FB finish generating coin key-points	BOOL	False/True (False)
bBusy	Status when FB is busy to generate key-points	BOOL	False/True (False)
bError	Status when FB has problems generating key-points	BOOL	False/True (False)
dwErrorID	Error Code when running FB	DWORD	DWORD(0)
lrMasterPoint	Master positions for DMC_SP_CamCurveWrite	[0..63] LREAL Array	0.0 ~ 360.0(0.0)
lrSlavePoint	Slave positions for DMC_SP_CamCurveWrite	[0..63] LREAL Array	0.0 ~ 360.0(0.0)

CamCurveType	Key point to key point curve type for DMC_SP_CamCurveWrite	[0..62] DMC_CamCurveType Array	0 ~ 9*[2](0)
bVelEnable	Velocity Enable Configurations for DMC_SP_CamCurveWrite	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrVelValue	Velocity Value for DMC_SP_CamCurveWrite	[0..63] LREAL Array	LREAL*[1](0)
bAccEnable	Acceleration Eanble Configurations for DMC_SP_CamCurveWrite	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrAccValue	Accerleration Value for DMC_SP_CamCurveWrite	[0..63] LREAL Array	LREAL*[1](0)
wWriteAmount	Key Point write amount for DMC_SP_CamCurveWrite	WORD	0~64(0)

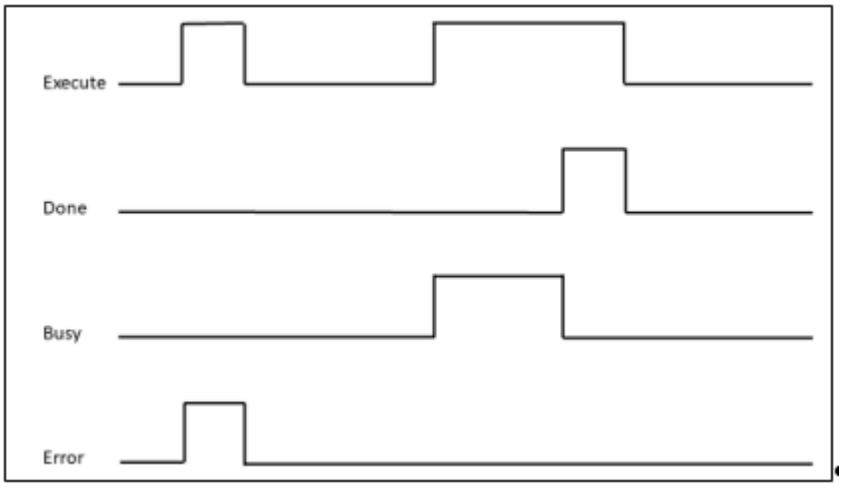
***Note1**: LREAL Range:
Positive values: -1.7976931348623157E+308 to -4.9406564584124654E-324
Zero: 0
Negative values: 4.9406564584124654E-324 to 1.7976931348623157E+308

***Note2**: Curve Type (0 ~ 9)
Straight line (0)
Quadratic Parabola (1)
5th Polynomial (2)
Basic Sine (3)
Inclined Sine (4)
Mod_Acc Sine (5)
Mod_Acc Trapezoidal (6)
Cubic_Spline_Nature (7)
Cubic_Spline_Clamp (8)
Cubic_Spline (9)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	*bExecute*=TRUE + FB finish curve key points generation	• When *bExecute* turns to False. If *bExecute* is False and bDone turns to True, then bDone turns to False immediately after maintaining the True state for one scan cycle.
bBusy	*bExecute*=TRUE + FB is generating curve key points	• When *bDone* turns to True When *bError* turns to True
bError	*bExecute*=TURE + FB raise error code(non-zero)	When *bExecute* turns to False.

- **Timing Diagram**

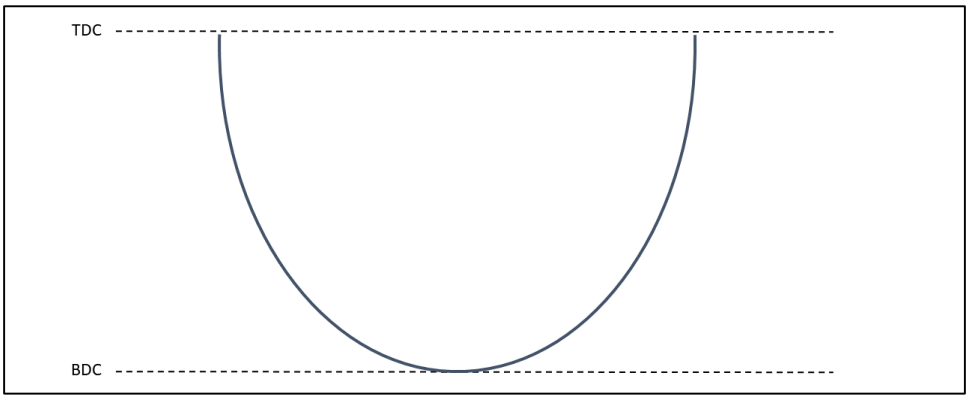


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

- **Function**

  - To generate KeyPoint for DMC_CamKeyPointWrite FB, the Slider move from TDC to BDC.

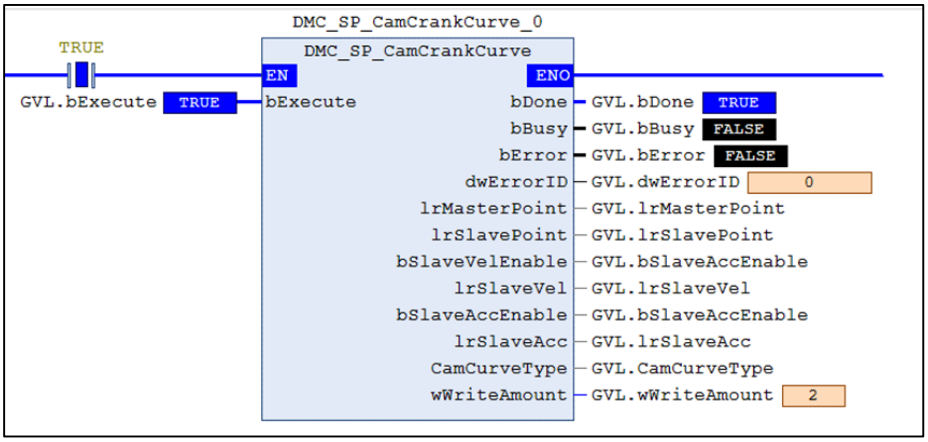


- **Troubleshooting**

Error Code	Description	Contents	Corrective Action
0	No Error	-	-

- **Sample Program**

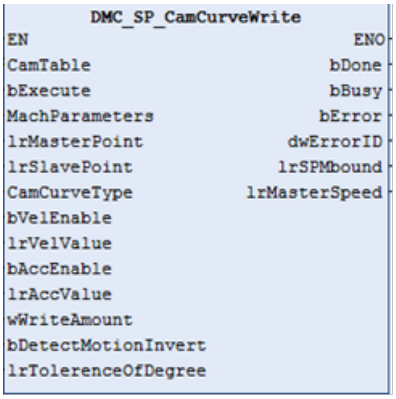
  - Input refMachParameters.lrRLength, refMachParameters.lrLLength (Link > 2* Crank).
  - Set *bExecute* to True.
  - Wait *bDone* to change to True. The FB will create Key points data.


DMC_SP_CamCrankCurve_0
DMC_SP_CamCrankCurve
TRUE
EN
ENO
GVL.bExecute
TRUE
bExecute
bDone
GVL.bDone
TRUE
bBusy
GVL.bBusy
FALSE
bError
GVL.bError
FALSE
dwErrorID
GVL.dwErrorID
0
lrMasterPoint
GVL.lrMasterPoint
lrSlavePoint
GVL.lrSlavePoint
bSlaveVelEnable
GVL.bSlaveAccEnable
lrSlaveVel
GVL.lrSlaveVel
bSlaveAccEnable
GVL.bSlaveAccEnable
lrSlaveAcc
GVL.lrSlaveAcc
CamCurveType
GVL.CamCurveType
wWriteAmount
GVL.wWriteAmount
2

2.4.1.5 DMC_SP_CamCurveWrite

- **Supported Devices:** AX-308E, AX-8

Servo Press industry-specific Function block, the purpose is to generate an electronic cam table according to the input key point information, and calculate the SPM upper limit value and the Master Axis speed according to the user input motor speed and gear ratio. At the same time, it provides a function of detecting curve reversal. Once a reversal is detected, a warning will appear to prevent the curve from reversing after processing.

FB/FC	Instruction	Graphic Expression	ST LANGUAGE
FB	DMC_SP_CamCurveWrite		DMC_SP_CamCurveWrite(CamTable: = _parameter, bExecute: = _parameter, MachParameters: = _parameter, lrMasterPoint: = _parameter, lrSlavePoint: = _parameter, CamCurveType: = _parameter, bVelEnable: = _parameter, lrVelValue: = _parameter, bAccEnable: = _parameter, lrAccValue: = _parameter, wWriteAmount: = _parameter, bDetectMotionInvert: = _parameter, lrTolerenceOfDegree: = _parameter, bDone=> _parameter, bBusy=> _parameter, bError=> _parameter, dwErrorID=> _parameter, lrSPMbound=> _parameter, lrMasterSpeed=> _parameter);

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	Run the function block control bit	BOOL	False/True (False)	Start to run with rising-edge signal, only one time.
MachParameters	ServoPress machine electromechanical parameters	ReferenceDMC_SP_MACHINE_PARAMETERS*3	-	*bBusy*=FALSE & *bExecute* is at rising-edge
lrMasterPoint	Key-points of master axis DMC_CamKeyPointWrite	[0..63] LREAL Array	0~360(0.0)	*bBusy*=FALSE & *bExecute* is at rising-edge
lrSlavePoint	Key-points of slave axis for DMC_CamKeyPointWrite	[0..63] LREAL Array	0~360(0.0)	*bBusy*=FALSE & *bExecute* is at rising-edge
CamCurveType	Curve Type Setup for DFB_CamKeyPointWrite	DMC_CamCuryveType Array[0..62]	0~9*2 (0)	*bBusy*=FALSE & *bExecute* is at rising-edge
bVelEnable	Velocity Enable Configurations for DMC_CamKeyPointWrite	WORD Array [0..63]	False/True (False)	*bBusy*=FALSE & *bExecute* is at rising-edge
lrVelValue	Velocity Value for	[0..63] LREAL	LREAL*1(0)	*bBusy*=FALSE &

	DMC_CamKeyPointWrite	Array		*bExecute* is at rising-edge
bAccEnable	Acceleration Enable Configurations for DMC_CamKeyPointWrite	WORD Array [0..63]	BOOL(FALSE)	*bBusy*=FALSE & *bExecute* is at rising-edge
lrAccValue	Acceleration Value for DMC_CamKeyPointWrite	[0..63] LREAL Array	LREAL*[1](0)	*bBusy*=FALSE & *bExecute* is at rising-edge
wWriteAmount	Key-points write amount for DMC_CamKeyPointWrite	WORD	2~64(0)	*bBusy*=FALSE & *bExecute* is at rising-edge
bDetectMotionInvert	Enable detecting motion invert functionality	BOOL	False/True (False)	*bBusy*=FALSE & *bExecute* is at rising-edge.
lrTolerenceOfDegree*[4]	The tolerance invert degree during motion between two key-points	LREAL	0~180(0)	*bBusy*=FALSE & *bExecute* is at rising-edge.

***Note1**: LREAL Range:
Positive values: -1.7976931348623157E+308 to -4.9406564584124654E-324
Zero: 0
Negative values: 4.9406564584124654E-324 to 1.7976931348623157E+308

***Note2**: CAM Curve Types (0 ~ 9)
Straight line (0)
Quadratic Parabola (1)
5th Polynomial (2)
Basic Sine (3)
Inclined Sine (4)
Mod_Acc Sine (5)
Mod_Acc Trapezoidal (6)
Cubic_Spline_Nature (7)
Cubic_Spline_Clamp (8)
Cubic_Spline (9)

***Note3**: DMC_SP_MACHINE_PARAMETERS

Name	Function	Data Type	Setting Value
lrRLength	Crank length (unit: mm), provided by machine maker	LREAL	Positive
lrLLength	Link length(unit: mm), provided by machine maker	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's SPM. (six times lrSPMsys is equal to the unit deg/sec)	LREAL	Positive
lrGearRatio	Reduction gear ratio	LREAL	Positive
lrRPMmotor	Max RPM of the motor	LREAL	Positive

***Note4** : When bDetectMotionInvert is set as FALSE, lrTolerenceOfDegree will be ignored.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	Status when FB finish writing ECAM table and calculating ECAM velocity upper bound value	BOOL	False/True (False)

bBusy	Status when FB is busy to write ECAM table	BOOL	False/True (False)
bError	Status when FB has problems writing ECAM	BOOL	False/True (False)
dwErrorID	Error Code when running FB	DWORD	DWORD(0)
lrSPMbound*2	Upper bound SPM value, based on input key-points	LREAL*1	LREAL(0)
lrMasterSpeed	Master axis speed(unit = deg/sec)	LREAL*1	LREAL(0)

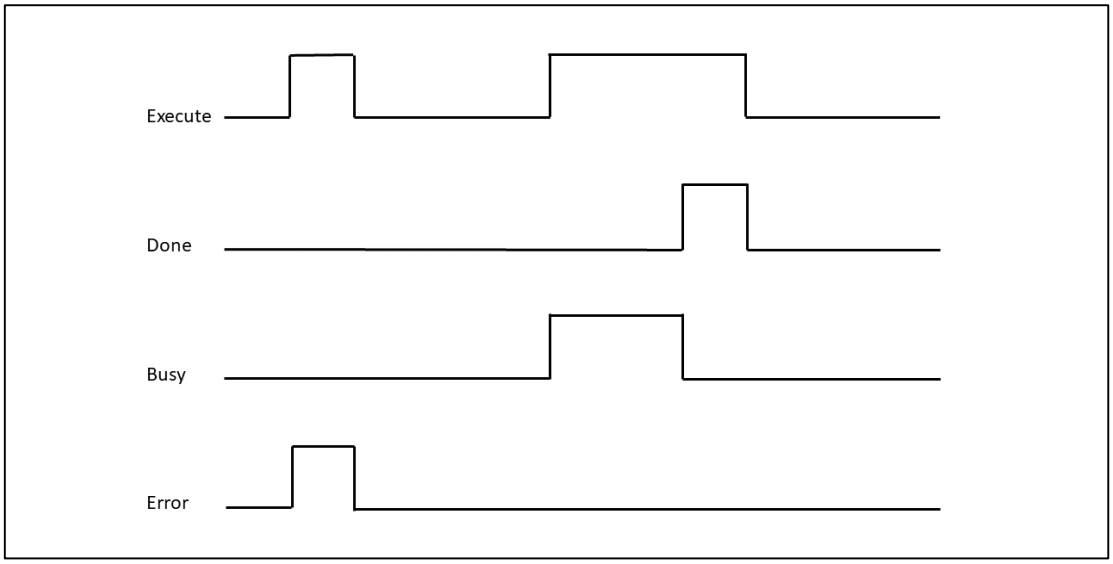
***Note1**: LREAL Range:
Positive values: -1.7976931348623157E+308 to -4.9406564584124654E-324
Zero: 0
Negative values: 4.9406564584124654E-324 to 1.7976931348623157E+308

***Note2**: Valid MachParameters.lrSPMsys should not exceed lrSPMbound value. lrSPMbound is the max speed of the motor.

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	*bExecute*=True + FB finish ECAM table write	• When *bExcute* turns to False If *bExecute* is False and *bDone* turns to True, then *bDone* turns to False immediately after maintaining the True state for one scan cycle.
bBusy	*bExecute*=True + FB is writing ECAM table	• When *bDone* turns to True When *bError* turns to True
bError	*bExecute*=True + FB raise error code(non-zero)	When *bExecute* turns to False.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	Specify the cam table.	MC_CAM_REF	MC_CAM_REF	*bBusy*=FALSE & *bExecute* is at rising-edge.

- **Function**
  - Input ECAM key-points data to DMC_SP_CamCurveWrite to generate ECAM curve and write the curve into ECAM table with a specific CAM table ID.
  - DMC_SP_CamCurveWrite will also calculate the upper bound SPM value. If system SPM value exceeds the upper bound SPM, the error message will be shown. Then users should adjust the system SPM if needed.
  - If Curve invert happen, DMC_SP_CamCurveWrite can detect when user set bDetectMotionInvert to True and set lrTolerenceOfDegree

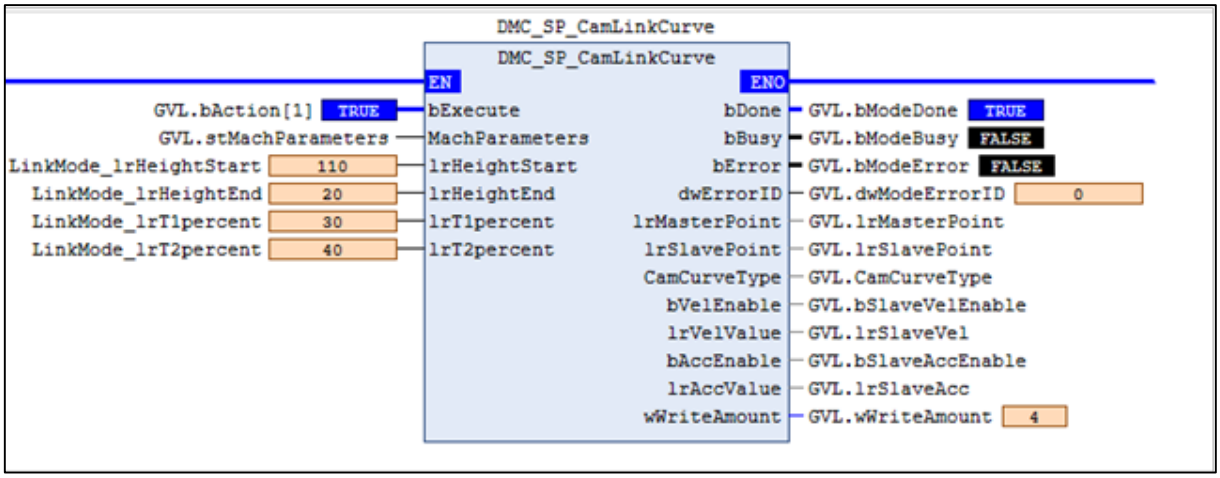
- **Troubleshooting**

When an error happens, *bError* will be True and dwErrorID will show non-zero. Refer to the following table for the error code.

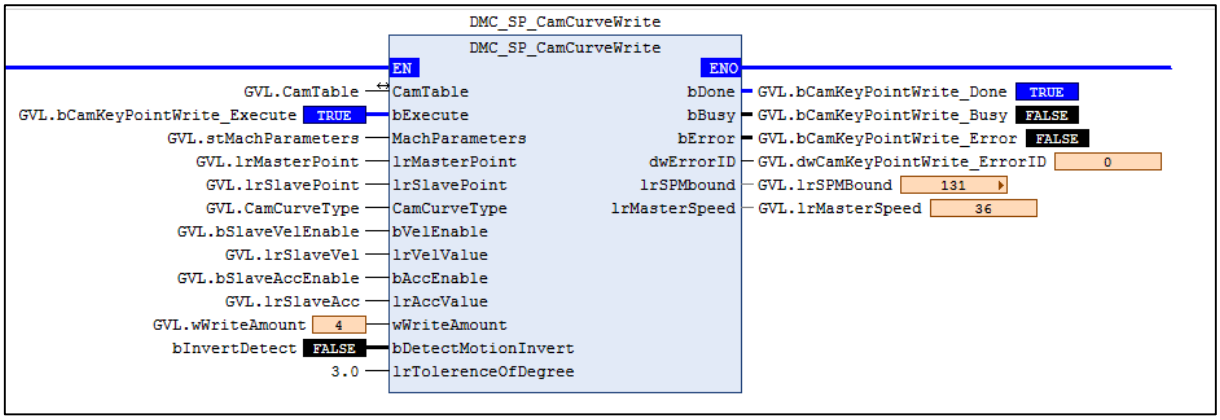
Error Code	Description	Contents	Corrective Action
0	No Error	-	-
1	Invalid machine paramters	Invalid electromechanical parameters were set	Check if these values are within the range • MachParameters.lrGearRatio • MachParameters.lrRPMmotor • MachParameters.lrSPMsys
4	Exceeds supported amount of key-points	wWriteAmount is out of range	Check if the values are within the range • wWriteAmount
5	SPM is over motor upper bound	On-demand master axis SPM is over motor upper bound	Check if the values are within the range • MachParameters.lrSPMsys
6	CAM Data Type is not supported	CAM Data Type is not supported	Check if CAM Data Type is supported • CamTable.byType • CamTable.byVarType
7	Tolerence invert degree is out of range	Tolerence of invert degree is out of range	Check if the values are within the range • lrTolerenceOfDegree
8	Curve motion invertion detected	One or more motion invertion are found	To adjust the input values • lrVelValue • lrAccValue • lrTolerenceOfDegree
Errors raised by inside reference function blocks			
100533	wWriteAmount out of range	Refer to AX3 user manual (DMC_ERROR)	Refer to AX3 user manual (DMC_ERROR)
100534	Invalid Master value of key-points	Refer to AX3 user manual (DMC_ERROR)	Refer to AX3 user manual (DMC_ERROR)
100535	Invalid acceleration values of key-points	Refer to AX3 user manual (DMC_ERROR)	Refer to AX3 user manual (DMC_ERROR)
100536	Invalid acceleration settings	Refer to AX3 user manual (DMC_ERROR)	Refer to AX3 user manual (DMC_ERROR)
100537	The curve type is not supported	Refer to AX3 user manual (DMC_ERROR)	Refer to AX3 user manual (DMC_ERROR)
100538	There is no boundary condition or wrong boundary condition	Refer to AX3 user manual (DMC_ERROR)	Refer to AX3 user manual (DMC_ERROR)
100539	The cam table data is	Refer to AX3 user manual	Refer to AX3 user manual (DMC_ERROR)

	written by other function	(DMC_ERROR)	

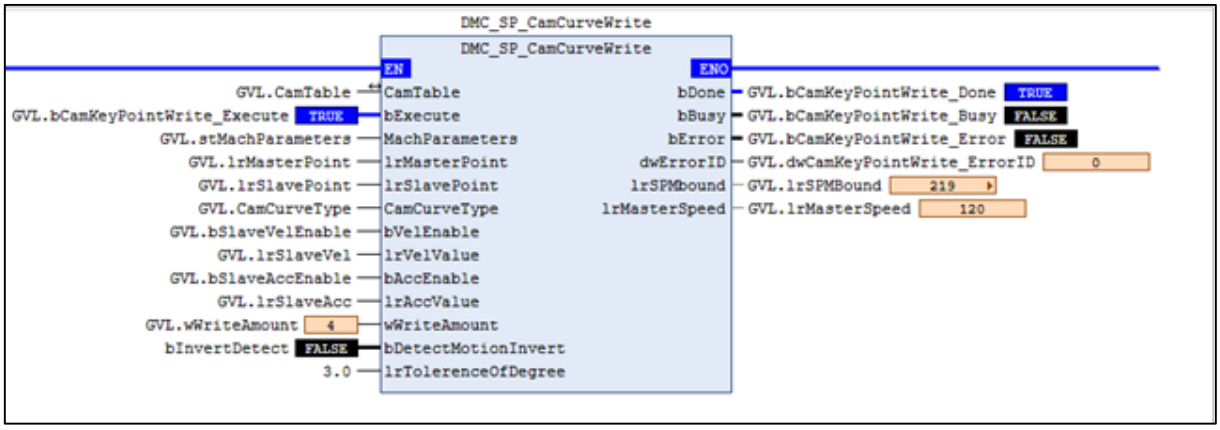
- **Sample Program**
  - Generate ECAM table:
    For Example, Select DMC_SP_CamLinkCurve FB (or other FB, like DMC_SP_CamPulse1Curve) to generate Key-Point Data.



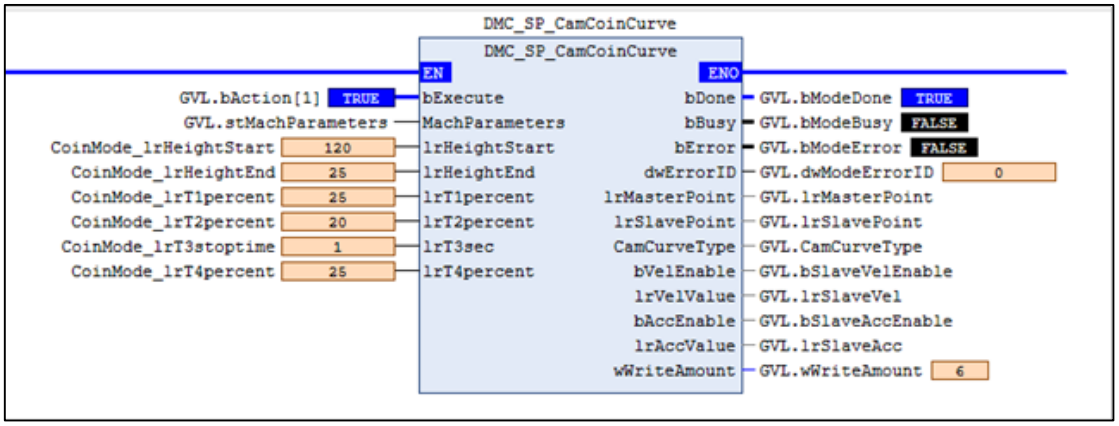
  - After FB successfully generates key-points data, invoking DMC_SP_CamCurveWrite to write the specific ECAM table.

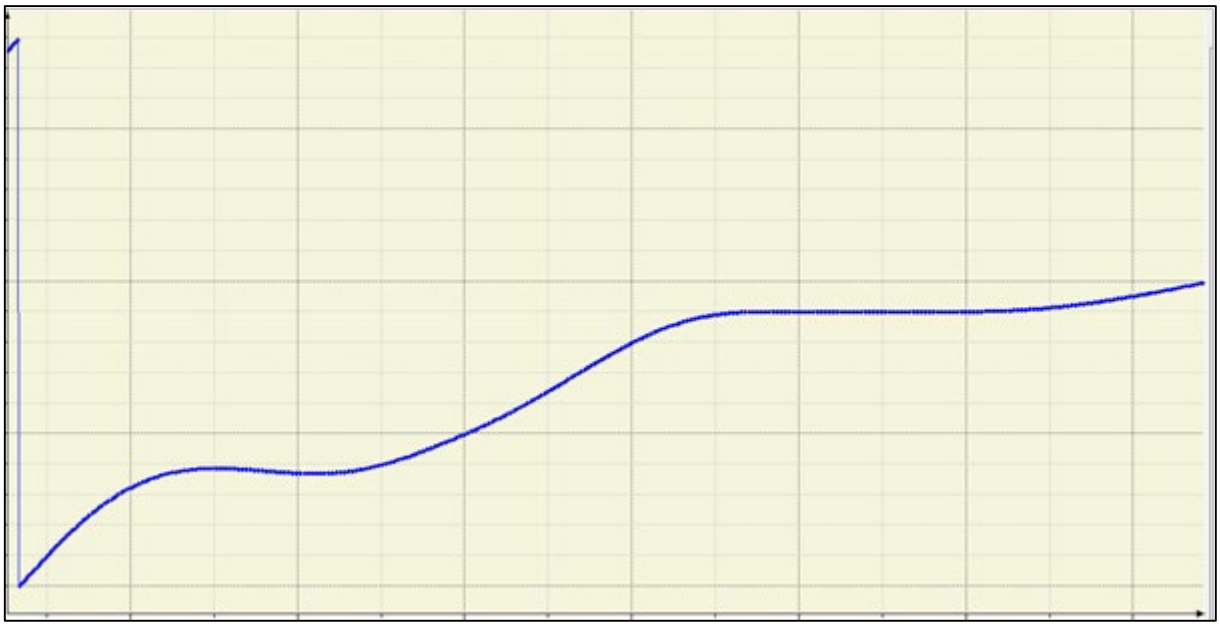


  - Calculate the upper bound SPM. Check if SPM exceeds the SPM bound value.
    Set Motor RPM=300, SPM = 350, GearRatio = 1.0, After DMC_SP_CamxxxCurve FB Generate Key-points, DMC_SP_CamCurveWrite will calculate the upper bound SPM. If the input SPM exceeds the SPM bound value, the function block will raise an error code. In this case, adjust SPM to run DMC_SP_CamCurveWrite again.

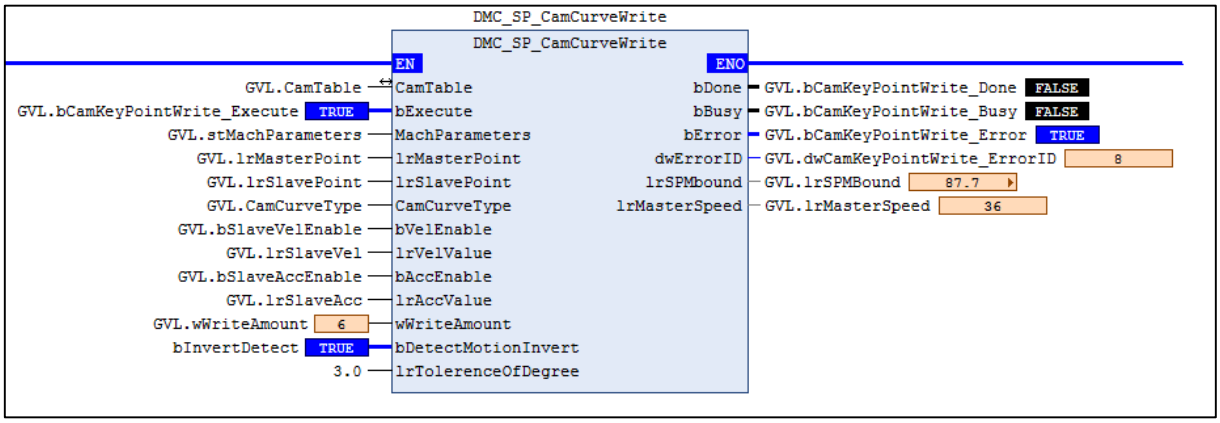


- Curve invert Detect
  Perform DMC_SP_CamCoinCurve and set lrSPM = 6, lrLlength = 220, lrRlength = 100, lrHightStart = 120, lrHightEnd = 25, lrT1_percent = 25, lrT2_percent = 20, lrT3_StopTime = 1, lrT4_percent = 25.





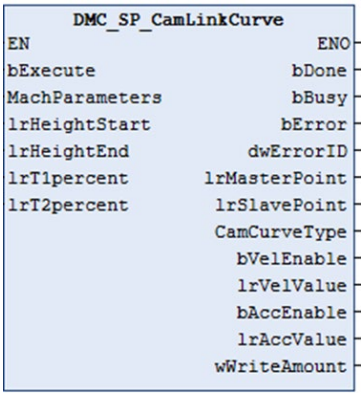
- Set bInvertDetect to True, and set lrTolerenceOfDegree to 3.0 and run DMC_SP_CamCurveWrite again. Error will be True, and ErrorID will show 8.



2.4.1.6 DMC_SP_CamLinkCurve

- **Supported Devices**: AX-308E, AX-8

Servo Press industry-specific FB, generates cam key point information according to the extended curve algorithm and generates an electronic cam table through DMC_SP_CamCurveWrite FB. Users can use the generated electronic cam table for Servo Press processing.

FB/FC	Instruction	Graphic Expression	ST Express
FB	DMC_SP_CamLinkCurve		DMC_SP_CamLinkCurve bExecute: = _parameter, MachParameters: = _parameter, lrHeightStart: = _parameter, lrHeightEnd: = _parameter, lrT1percent: = _parameter, lrT2percent: = _parameter, bDone=> _parameter, bBusy=> _parameter, bError=> _parameter, dwErrorID=> _parameter, lrMasterPoint=> _parameter, lrSlavePoint=> _parameter, CamCurveType=> _parameter, bVelEnable=> _parameter, lrVelValue=> _parameter, bAccEnable=> _parameter, lrAccValue=> _parameter, wWriteAmount=> _parameter);

- **Inputs**

Name	Function	Data Type	Setting Range (Default)	Timing to Take Effect
bExecute	Run the function block control bit	BOOL	False/True (False)	-
MachParameters	ServoPress machine electromechanical parameters	ReferenceDMC_SP_MACHINE_PARAMETERS *2	-	*bBusy*=FALSE & *bExecute* is at rising-edge
lrHeightStart	Process start position - the height of slider	LREAL*1	Positive (0) [Range] 0 < lrHeightStart < (2 * lrRLength)	*bBusy*=FALSE & *bExecute* is at rising-edge
lrHeigthEnd	Process end position – the height of slider	LREAL*1	Positive (0) [Range] 0 < lrHeightStart < (2 * lrRLength)	*bBusy*=FALSE & *bExecute* is at rising-edge
lrT1percent	T1 percentage of time	LREAL*1	Positive (0) [Range] 0 < lrT1percent < 100	*bBusy*=FALSE & *bExecute* is at rising-edge
lrT2percent	T2 percentage of time	LREAL*1	Positive (0) [Range] 0 <	*bBusy*=FALSE & *bExecute* is at rising-

			lrT2percent < 100	edge

***Note1**: LREAL Range:
Positive values: -1.7976931348623157E+308 to -4.9406564584124654E-324
Zero: 0
Negative values: 4.9406564584124654E-324 to 1.7976931348623157E+308

***Note2**: DMC_SP_MACHINE_PARAMETERS

Name	Function	Data Type	Setting Value
lrRLength	Crank length (unit: mm), provided by machine maker	LREAL	Positive
lrLLength	Link length (unit: mm), provided by machine maker	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's SPM. (six times lrSPMsys is equal to the unit deg/sec)	LREAL	Positive
lrGearRatio	Reduction gear ratio	LREAL	Positive
lrRPMmotor	Max RPM of the motor	LREAL	Positive

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	Status when FB finish generating coin key-points	BOOL	False/True (False)
bBusy	Status when FB is busy to generate key-points	BOOL	False/True (False)
bError	Status when FB has problems generating key-points	BOOL	False/True (False)
dwErrorID	Error Code when running FB	DWORD	16#00000000~16#FFFFFFFF(16#00000000)
lrMasterPoint	Master positions for DMC_SP_CamCurveWrite	[0..63] LREAL Array	0.0 ~ 360.0(0.0)
lrSlavePoint	Slave positions for DMC_SP_CamCurveWrite	[0..63] LREAL Array	0.0 ~ 360.0(0.0)
CamCurveType	Key point to key point curve type for DMC_SP_CamCurveWrite	[0..62] DMC_CamCurveType Array	0 ~ 9[*2](0)
bVelEnable	Velocity Enable Configurations for DMC_SP_CamCurveWrite	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrVelValue	Velocity Value for DMC_SP_CamCurveWrite	[0..63] LREAL Array	LREAL[*1](0)
bAccEnable	Acceleration Eanble Configurations for DMC_SP_CamCurveWrite	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrAccValue	Accerleration Value for DMC_SP_CamCurveWrite	[0..63] LREAL Array	LREAL[*1](0)
wWriteAmount	Key Point write amount for DMC_SP_CamCurveWrite	WORD	0~64(0)

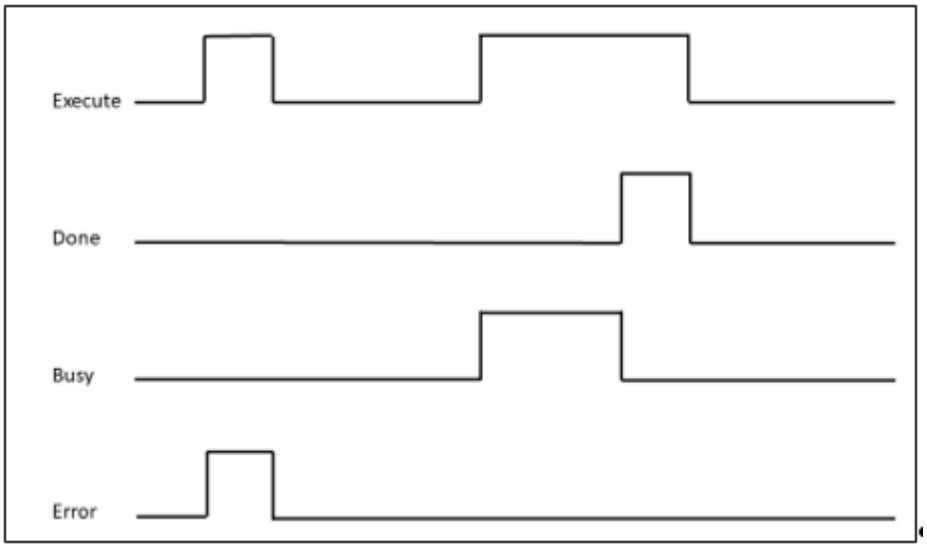
***Note1**: LREAL Range:
Positive values: -1.7976931348623157E+308 to -4.9406564584124654E-324
Zero: 0
Negative values: 4.9406564584124654E-324 to 1.7976931348623157E+308

***Note2**: Curve Type (0 ~ 9)
Straight line (0)
Quadratic Parabola (1)
5th Polynomial (2)
Basic Sine (3)
Inclined Sine (4)
Mod_Acc Sine (5)
Mod_Acc Trapezoidal (6)
Cubic_Spline_Nature (7)
Cubic_Spline_Clamp (8)
Cubic_Spline(9)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	*bExecute*=TRUE + FB finish curve key points generation	• When *bExecute* turns to False If *bExecute* is False and *bDone* turns to True, then *bDone* turns to False immediately after maintaining the True state for one scan cycle.
bBusy	*bExecute*=TRUE + FB is generating curve key points	• When *bDone* turns to True When *bError* turns to True
bError	*bExecute*=TURE + FB raise error code(non-zero)	When *bExecute* turns to False.

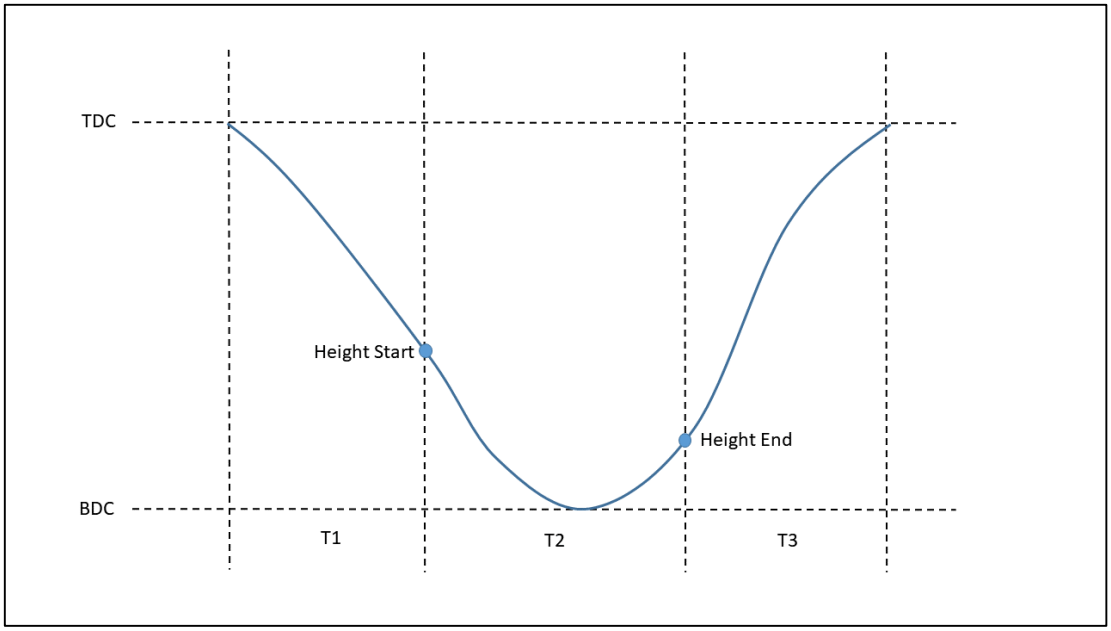
- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

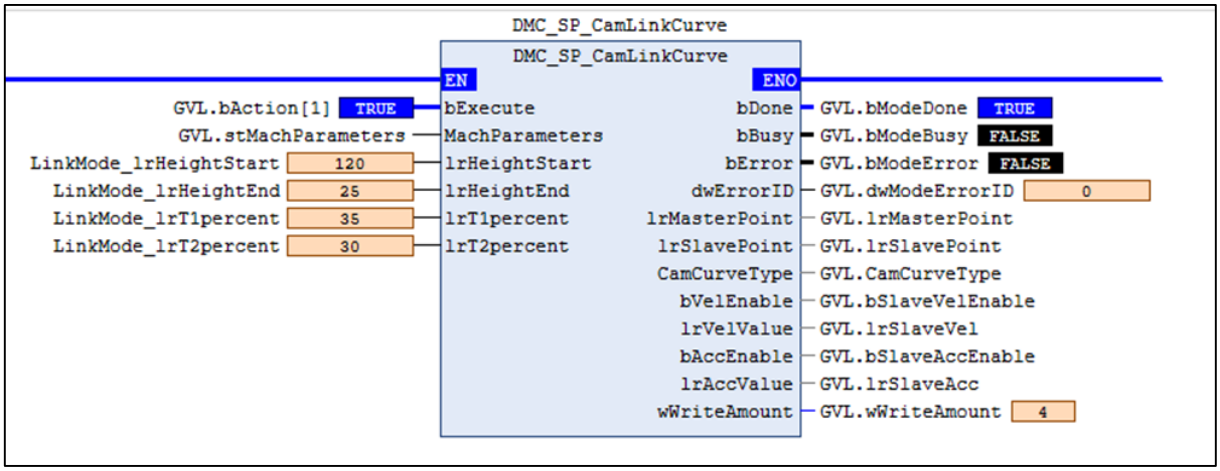
- **Function**
  - User input T1, T2 and FB will calculate T3 time percent.
  - The Slider will move from TDC to Height Start at T1 time percent.
  - The Slider will move to Height End at T2 time percent.
  - Finally, back to TDC at T3 time percent.



- **Troubleshooting**

Error Code	Description	Contents	Corrective Action
0	No Error	-	-
1	Invalid machine parameters	Invalid mechanical parameters were set	Check if these values are within the range - MachParameters.lrRLength - MachParameters.lrLLength
2	Invalid slider height	Incorrect range of slider height was set	Check if the values are within the range - lrHeightStart - lrHeightEnd
3	Invalid time percentage	Invalid time percentage was set. (0% or sum of time percentage exceeds 100%)	Check if the values are within the range - lrT1percent - lrT2percent

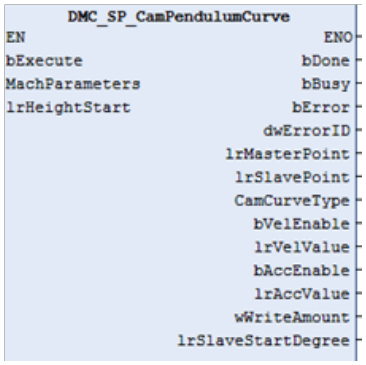
- **Sample Program**
  - Input MachParameters.lrRLength, MachParameters.lrLLength (Link > 2* Crank).
  - Input Height Start and Height End.
  - Input T1 time percent, T2 time percent.
  - Set *bExecute* to True.
  - Wait *bDone* to change to True. The FB will generate key-points data.


DMC_SP_CamLinkCurve
DMC_SP_CamLinkCurve
EN
ENO
GVL.bAction[1]
TRUE
bExecute
bDone
GVL.bModeDone
TRUE
GVL.stMachParameters
MachParameters
bBusy
GVL.bModeBusy
FALSE
LinkMode_lrHeightStart
120
lrHeightStart
bError
GVL.bModeError
FALSE
LinkMode_lrHeightEnd
25
lrHeightEnd
dwErrorID
GVL.dwModeErrorID
0
LinkMode_lrT1percent
35
lrT1percent
lrMasterPoint
GVL.lrMasterPoint
LinkMode_lrT2percent
30
lrT2percent
lrSlavePoint
GVL.lrSlavePoint
CamCurveType
GVL.CamCurveType
bVelEnable
GVL.bSlaveVelEnable
lrVelValue
GVL.lrSlaveVel
bAccEnable
GVL.bSlaveAccEnable
lrAccValue
GVL.lrSlaveAcc
wWriteAmount
GVL.wWriteAmount
4

2.4.1.7 DMC_SP_CamPendulumCurve

- **Supported Devices**: AX-308E, AX-8

Servo Press industry-specific FB, generates cam key point information according to the pendulum curve algorithm, and generates an electronic cam table through DMC_SP_CamCurveWrite FB. Users can use the generated electronic cam table for Servo Press processing.

FB/FC	Instruction	Graphic Expression	ST LANGUAGE
FB	DMC_SP_CamPendulumCruve		DMC_SP_CamPendulumCurve(bExecute: = _parameter, MachParameters: = _parameter, lrHeightStart: = _parameter, bDone=> _parameter, bBusy=> _parameter, bError=> _parameter, dwErrorID=> _parameter, lrMasterPoint=> _parameter, lrSlavePoint=> _parameter, CamCurveType=> _parameter, bVelEnable=> _parameter, lrVelValue=> _parameter, bAccEnable=> _parameter, lrAccValue=> _parameter, wWriteAmount=> _parameter, lrSlaveStartDegree=> _parameter);

- **Inputs**

Name	Function	Data Type	Setting Value (Default)	Timing to Take Effect
bExecute	Run the function block control bit	BOOL	False/True (False)	Start to run with the rising-edge signal, only one time.
MachParameters	ServoPress machine electromechanical parameters	ReferenceDMC_SP_MACHINE_PARAMETERS*2	-	*bBusy*=FALSE & *bExecute* is at rising-edge
lrHeightStart	The height of slider, and it is the start position of the pressing process	LREAL*1	Positive (0) [Range] 0 < lrHeightStart < (2 * lrRLength)	*bBusy*=FALSE & *bExecute* is at rising-edge

***Note1**: LREAL Range:
Positive values: -1.7976931348623157E+308 to -4.9406564584124654E-324
Zero: 0
Negative values: 4.9406564584124654E-324 to 1.7976931348623157E+308

***Note2**: DMC_SP_MACHINE_PARAMETERS

Name	Function	Data Type	Setting Value

lrRLength	Crank Length (unit: mm), provided by machine maker	LREAL	Positive
lrLLength	Link Length(unit: mm), provided by machine maker	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's SPM. (six times lrSPMsys is equal to the unit deg/sec)	LREAL	Positive
lrGearRatio	Reduction gear ratio	LREAL	Positive
lrRPMmotor	Max RPM of the motor	LREAL	Positive

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	Status when FB completed generating pendulum key-points	BOOL	False/True (False)
bBusy	Status when FB is generating key-points	BOOL	False/True (False)
bError	Status when FB has problems generating key-points	BOOL	False/True (False)
dwErrorID	Error code of this FB	DWORD	16#00000000~16#FFFFFFFF (16#00000000)
lrMasterPoint	Master positions for DMC_SP_CamCurveWrite	[0..63] LREAL Array	0.0 ~ 360.0 (0.0)
lrSlavePoint	Slave positions for DMC_SP_CamCurveWrite	[0..63] LREAL Array	0.0 ~ 360.0 (0.0)
CamCurveType	Cruve type between key-points for DMC_SP_CamCurveWrite	[0..62] DMC_CamCurveType Array	0 ~ 9[*2] (0)
bVelEnable	Velocity Enable Configurations for DMC_SP_CamCurveWrite	[0..63] BOOL Array	False/True (False)
lrVelValue	Velocity Value for DMC_SP_CamCurveWrite	[0..63] LREAL Array	LREAL (0)
bAccEnable	Acceleration Eanble Configurations for DMC_SP_CamCurveWrite	[0..63] BOOL Array	False/True (False)
lrAccValue	Accerleration Value for DMC_SP_CamCurveWrite	[0..63] LREAL Array	LREAL (0)
wWriteAmount	Key Point value	WORD	0~64 (0)
lrSlaveStartDegree	Based on MachParameters.lrRLength, MachParameters.lrLLength, and lrHeightStart to calculate out the angel of slave-axis (via DFC_SP_HeightToDegree).	LREAL[*1]	0.0~180.0 (0)

***Note1**: LREAL Range:
Positive values: -1.7976931348623157E+308 to -4.9406564584124654E-324
Zero: 0

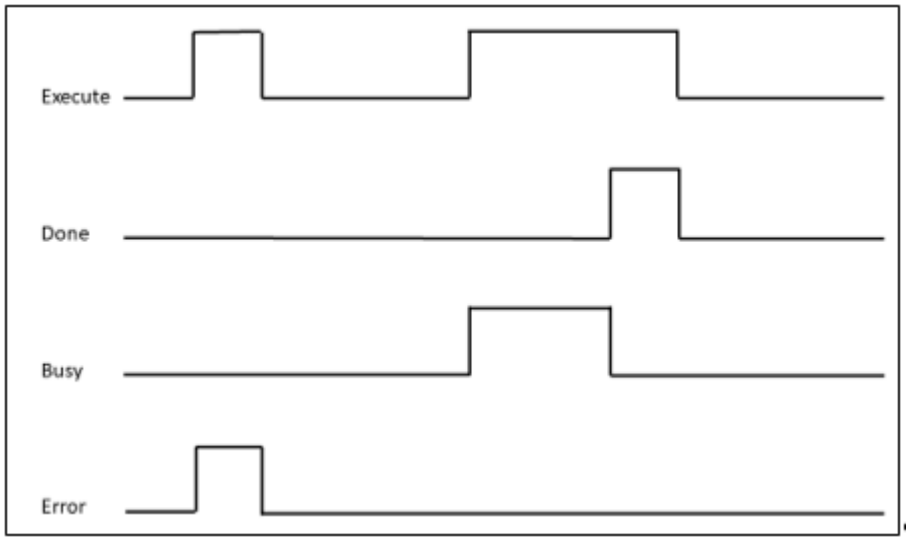
Negative values: 4.9406564584124654E-324 to 1.7976931348623157E+308

***Note2**: Curve Type (0 ~ 9) Straight line (0)
Quadratic Parabola (1)
5th Polynomial (2)
Basic Sine (3)
Inclined Sine (4)
Mod_Acc Sine (5)
Mod_Acc Trapezoidal (6)
Cubic_Spline_Nature (7)
Cubic_Spline_Clamp (8)
Cubic_Spline (9)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	*bExecute*=True + FB finish curve key points generation	• When *bExecute* turns to False If *bExecute* is False and *bDone* turns to True, then *bDone* turns to False immediately after maintaining the True state for one scan cycle.
bBusy	*bExecute*=TRUE + FB is generating curve key points	• When *bDone* turns to True When *bError* turns to True
bError	*bExecute*=True + FB raise error code(non-zero)	When *bExecute* turns to False.

- **Timing Diagram**

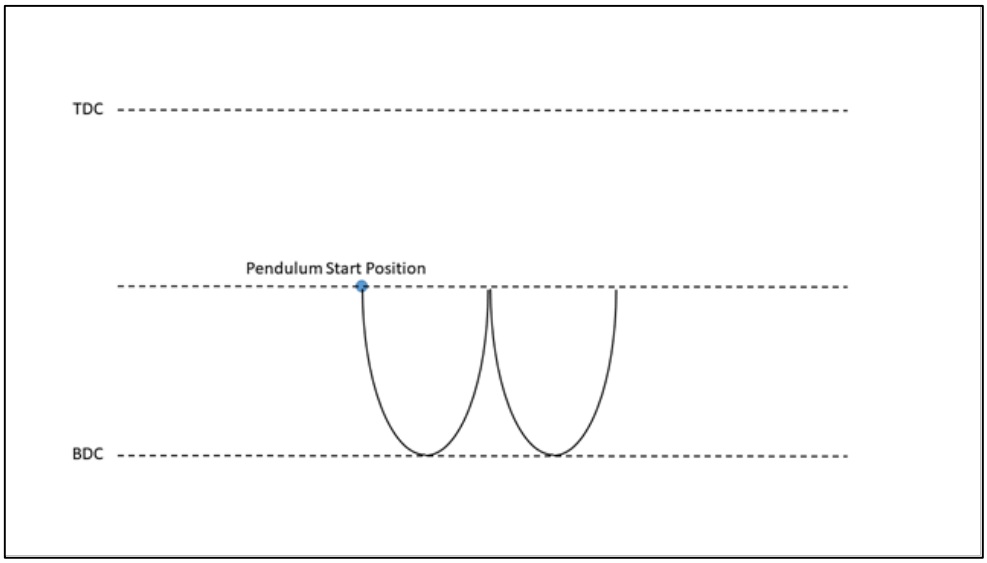


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

- **Function**
  - Start Motion and the slider will move from Start Postion to End Postion.
  - And then back and forth on BDC for process time.
  - Finally, back to TDC.

- Invoke DMC_SP_CamCurveWrite to generate and write ECAM table after this FB is done.
- Pendunlum Curve Setup Page.

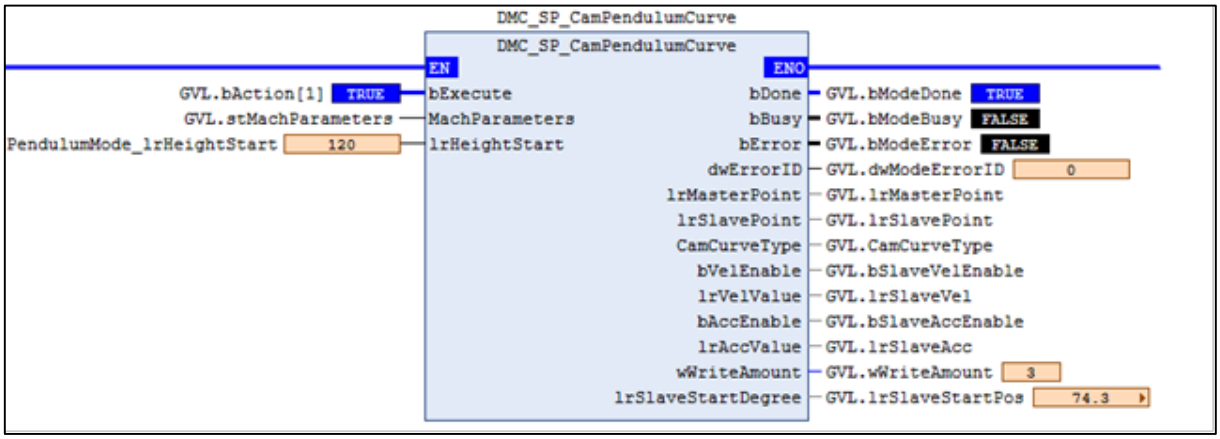


- **Troubleshooting**

Error Code	Description	Contents	Corrective Action
0	No Error	-	-
1	Invalid machine parameters	Invalid electromechanical parameters were set	Check if the values are within the range • MachParameters.lrRLength • MachParameters.lrLLength
2	Invalid slider height	Incorrect range of slide height was set	Check if the values are within the range • lrHeightStart

- **Sample Program**

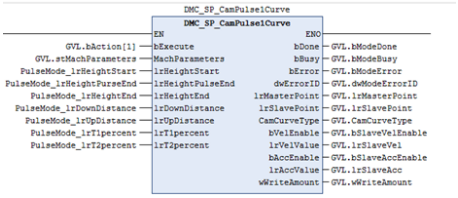
- Setre fMachParameters.lrRLength=90.0mm, MachParameters.lrLLength=670mm, and lrHeightStart=120.0mm.
- Set *bExecute* from False to True to trigger curve key-points calculation. Once calculation completes, *bDone* will be True, and generates related key-points arrays.



2.4.1.8 DMC_SP_CamPulse1Curve

- **Supported Devices:** AX-308E, AX-8

Servo Press industry-specific FB, generates cam key point information according to the pulse 1 curve algorithm, and generates an electronic cam table through DMC_SP_CamCurveWrite FB.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_SP_Pulse1Curve		DMC_SP_CamPulse1Curve(bExecute: = _parameter, MachParameters: = _parameter, lrHeightStart: = _parameter, lrHeightPulseEnd: = _parameter, lrHeightEnd: = _parameter, lrDownDistance: = _parameter, lrUpDistance: = _parameter, lrT1percent: = _parameter, lrT2percent: = _parameter, bDone=> _parameter, bBusy=> _parameter, bError=> _parameter, dwErrorID=> _parameter, lrMasterPoint=> _parameter, lrSlavePoint=> _parameter, CamCurveType=> _parameter, bVelEnable=> _parameter, lrVelValue=> _parameter, bAccEnable=> _parameter, lrAccValue=> _parameter, wWriteAmount=> _parameter);

***Note:** Recommend to set lrTolerenceOfDegree as 5~10 degree for Pulse1 Curve to invoke DMC_SP_CamCurveWrite. Due to poly5 characteristics, Pulse1 curve trajectory will invert at the lower and upper junction. But, this will not impact the user scenario.

- **Inputs**

Name	Function	Data Type	Setting Range (Default)	Timing to Take Effect
bExecute	Run the function block control bit	BOOL	False/True (False)	Start to run with the rising-edge signal, only one time.
lrRLength	The length of crank which provided by machine maker	LREAL*1	Positive(0)	*bBusy*=FALSE & *bExecute* is at rising-edge
lrLLength	The length of link which provided by machine maker	LREAL*1	Positive (0) [Range]lrLLength > 2 * lrRLength	*bBusy*=FASLE & *bExecute* is at rising-edge
MachParameters	ServoPress machine electromechanical parameters	ReferenceDMC_SP_MACHINE_PARAMETERS*2	-	*bBusy*=FALSE & *bExecute* is at rising-edge
lrHeightStart	Process start position - the height of slider	LREAL*1	Positive (0) [Range] 0 < lrHeightStart < (2 * lrRLength)	*bBusy*=FALSE & *bExecute* is at rising-edge
lrHeightPulseEnd	Pulse Porcess End Position	LREAL*1	[Range] 0<lrHeightPuserEnd< lrHeightStart	*bBusy*=FALSE & *bExecute* is at rising-edge
lrHeigthEnd	Process end position – the height of slider	LREAL*1	Positive (0) [Range] 0 < lrHeightStart < (2 * lrRLength)	*bBusy*=FALSE & *bExecute* is at rising-edge
lrDownDistance	The distance under the slider	LREAL*1	Positive (0) [Range] 0<lrDownDistance<lrHeightStart	*bBusy*=FALSE & *bExecute* is at rising-edge
lrUpDistance	The distance above the slider	LREAL*1	Positive (0) [Range] 0<lrUPDistance<lrDownDistance	*bBusy*=FALSE & *bExecute* is at rising-edge
lrT1percent	T1 time proportion	LREAL*1	Positive (0) [Range] 0 < lrT1percent < 100	*bBusy*=FALSE & *bExecute* is at rising-edge
lrT2percent	T2 time proportion	LREAL*1	Positive (0) [Range] 0 < lrT2percent < 100	*bBusy*=FALSE & *bExecute* is at rising-edge

***Note1**: LREAL Range:
Positive values: -1.7976931348623157E+308 to -4.9406564584124654E-324
Zero: 0
Negative values: 4.9406564584124654E-324 to 1.7976931348623157E+308

***Note2**: DMC_SP_MACHINE_PARAMETERS structure

Name	Function	Data Type	Setting Value
lrRLength	Crank length, provided by machine maker	LREAL	Positive

lrLLength	Link length, provided by machine maker	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's SPM. (six times lrSPMsys is equal to the unit deg/sec)	LREAL	Positive
lrGearRatio	Reduction gear ratio	LREAL	Positive
lrRPMmotor	Max RPM of the motor	LREAL	Positive

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	True when FB finish generating key-points	BOOL	False/True (False)
bBusy	True when FB is busy to generate key-pionts	BOOL	False/True (False)
bError	True when FB has problems generating key-points	BOOL	False/True (False)
dwErrorID	Error code when running FB	DWORD	16#00000000~16#FFFFFFFF(16#00000000)
lrMasterPoint	Master positions for DMC_SP_CamCurveWrite	[0..63] LREAL Array	0.0 ~ 360.0(0.0)
lrSlavePoint	Slave positions for DMC_SP_CamCurveWrite	[0..63] LREAL Array	0.0 ~ 360.0(0.0)
CamCurveType	Key point to key point curve type for DMC_SP_CamCurveWrite	[0..62] DMC_CamCurveType Array*1	0 ~ 9*2(0)
bVelEnable	Velocity Enable Configurations for DMC_SP_CamCurveWrite	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrVelValue	Velocity Value for DMC_SP_CamCurveWrite	[0..63] LREAL Array	LREAL*1(0)
bAccEnable	Acceleration Eanble Configurations for DMC_SP_CamCurveWrite	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrAccValue	Accerleration Value for DMC_SP_CamCurveWrite	[0..63] LREAL Array	LREAL*1(0)
wWriteAmount	Key Point value for DMC_SP_CamCurveWrite	WORD	0~64(0)

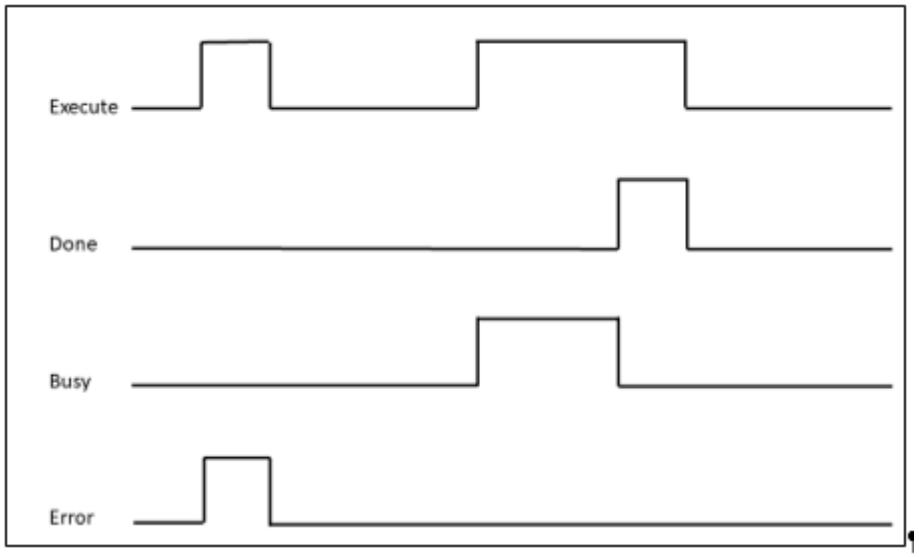
***Note**: Curve Type (0 ~ 9)
Straight line (0)
Quadratic Parabola (1)
5th Polynomial (2)
Basic Sine (3)
Inclined Sine (4)
Mod_Acc Sine (5)
Mod_Acc Trapezoidal (6)
Cubic_Spline_Nature (7)

Cubic_Spline_Clamp (8)
Cubic_Spline (9)

- **Output Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
bDone	*bExecute*=TRUE + FB finish curve key points generation	• When *bExecute* turns to False If *bExecute* is False and *bDone* turns to True, then *bDone* turns to False immediately after maintaining the True state for one scan cycle.
bBusy	*bExecute*=TRUE + FB is generating curve key points	• When *bDine* turns to True When *bError* turns to True
bError	*bExecute*=TURE + FB raise error code(non-zero)	When *bExecute* turns to False.

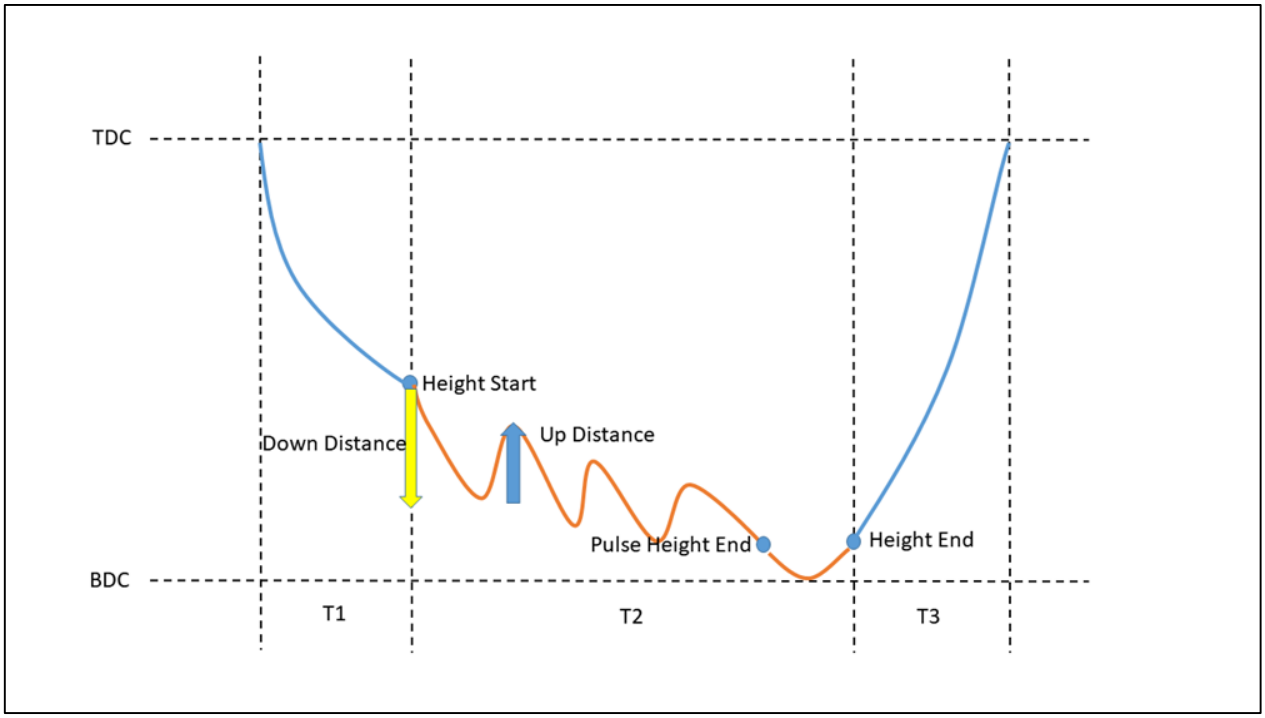
- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

- **Function**
  - To generate Key-Point data for DMC_SP_CamCurveWrite FB.
  - User input T1 and T2 time percent. FB will calculate T3 time percent.
  - The Slider will move from TDC to the processing starting position at T1 time percent.
  - The Slider will down and up until Slider Height move to Height PulseEnd at T2 time percent.
  - And then, the slider will get back to the end position.
  - Finally, back to TDC at T3 percent.

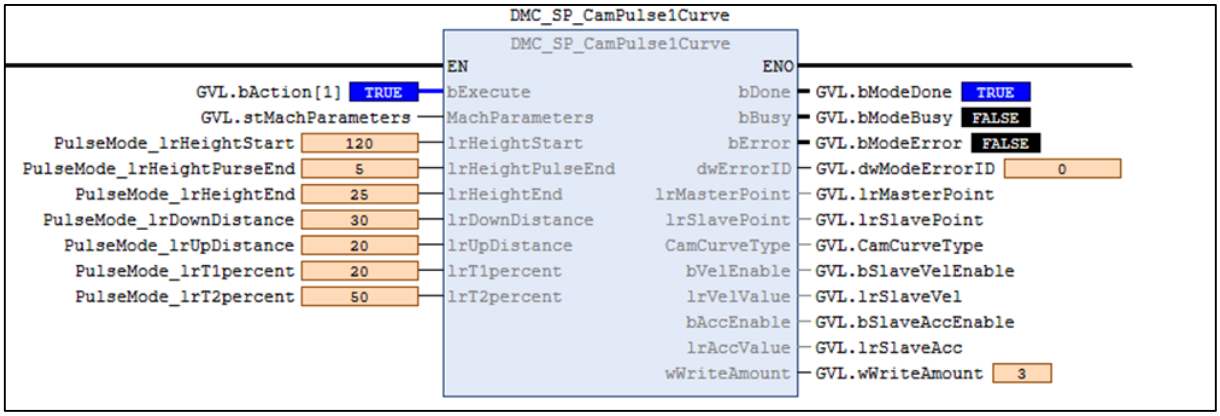


- **Troubleshooting**

Error Code	Description	Contents	Corrective Action
0	No Error	-	-
1	Invalid machine parameters	Invalid electromechanical parameters were set	Check if these values are within the range • MachParameters.lrRLength • MachParameters.lrLLength
2	Invalid slider height	Incorrect range of slider height was set	Check if the values are within the range • lrHeightStart • lrHeightPulseEnd • lrHeightEnd • lrDownDistance • lrUpDistance
3	Invalid time percentage	Invalid time percentage was set. (0% or sum of time percentage exceeds 100%)	Check if the values are within the range • lrT1percent • lrT2percent
4	Exceeds supported amount of key-points	wWriteAmount will be out of range in generating key-points.	Check the following values • lrDownDistance • lrUpDistance

- **Sample Program**

  - Input Crank Length, Link Length (Link > 2* Crank).
  - Input Height Start, Height End, and Pulse Height End (Height Start > Height PulseEnd).
  - Input slider down and up distance (Down > Up).
  - Input T1percent, T2percent.
  - Set *bExecute* to True.
  - Wait *bDone* to change to True. The Function block will generate Key-points data.
  - Transfer Key-point Data to DMC_SP_CamCurveWrite to generate CamTable.


DMC_SP_CamPulse1Curve
DMC_SP_CamPulse1Curve
EN
ENO
GVL.bAction[1] TRUE
bExecute
bDone
GVL.bModeDone TRUE
GVL.stMachParameters
MachParameters
bBusy
GVL.bModeBusy FALSE
PulseMode_lrHeightStart 120
lrHeightStart
bError
GVL.bModeError FALSE
PulseMode_lrHeightPurseEnd 5
lrHeightPulseEnd
dwErrorID
GVL.dwModeErrorID 0
PulseMode_lrHeightEnd 25
lrHeightEnd
lrMasterPoint
GVL.lrMasterPoint
PulseMode_lrDownDistance 30
lrDownDistance
lrSlavePoint
GVL.lrSlavePoint
PulseMode_lrUpDistance 20
lrUpDistance
CamCurveType
GVL.CamCurveType
PulseMode_lrT1percent 20
lrT1percent
bVelEnable
GVL.bSlaveVelEnable
PulseMode_lrT2percent 50
lrT2percent
lrVelValue
GVL.lrSlaveVel
bAccEnable
GVL.bSlaveAccEnable
lrAccValue
GVL.lrSlaveAcc
wWriteAmount
GVL.wWriteAmount 3

Appendix A

A.1 Instruction Tables and Indexes

A.1.1 By Function

- **Single-axis Motion Control Instruction (Synchronous axes)**

Categories	Name	Description
Positioning control	MC_Home	MC_Home controls the axis to perform the homing operation.
	MC_Stop	MC_Stop decelerates an axis to a stop.
	MC_Halt	MC_Halt stops the axis movement in a controlled way.
	MC_MoveAbsolute	MC_MoveAbsolute controls the axis to move to the specified absolute target position at a specified behavior.
	MC_MoveRelative	MC_MoveRelative controls the axis to move a specified relative distance with a specified behavior.
	MC_MoveAdditive	MC_MoveAdditive controls the axis to move an additional distance at a given speed and acceleration.
	MC_MoveSuperimposed	MC_MoveSuperimposed controls the axis to move a relative superimposed distance at a specified behavior while the axis is moving.
	MC_PositionProfile	MC_PositionProfile is used to set time and position to plan motion profiles.
	MC_Jog	MC_Jog enables an axis to be moved forward or backward.
	DMC_Home_P	DMC_Home_P, an application function block of pulse output, drives the pulse axis to perform the homing in the set mode.
	DMC_ImmediateStop_P	DMC_ImmediateStop_P can stop the PO axis motion immediately and stop the pulse output.
	DMC_MoveVelocityStopByPos	DMC_MoveVelocityStopByPos controls an axis to stop at a specified position after a period of motion.
	DMC_MoveFeed	DMC_MoveFeed can specify an external interrupt input. During the movement of the target, the position where the interrupt occurs is used as the starting point to perform the positioning movement.
	DMC_MoveModulo	DMC_MoveModulo is used for modulo positioning and specifies the number of rotation turns.
	DMC_Home_E	DMC_Home_E controls and plans homing.
Velocity control	MC_MoveVelocity	MC_MoveVelocity performs velocity control on an axis in the position mode with a specified behavior and an average velocity.
	MC_VelocityProfile	MC_VelocityProfile is used to set time and velocity to plan motion profiles.
	MC_AccelerationProfile	Similar to MC_PositionProfile, MC_AccelerationProfile is used to set time and acceleration to plan motion profiles. However, its position points are defined by acceleration variables in MC_TV_REF.

Categories	Name	Description
	DMC_VelocityControl	DMC_VelocityControl performs a velocity control on a specified axis in the CSV speed mode with the specified behavior and an average velocity.
Torque control	DMC_TorqueControl	DMC_TorqueControl controls the torque according to the torque control mode of the servo drive.
Sync control	MC_CamIn	MC_CamIn performs cam operation.
	MC_CamOut	MC_CamOut deactivates the meshing between master and slave axis.
	MC_GearIn	The function block MC_GearIn activates a linear master-slave meshing.
	MC_GearOut	MC_GearOut disconnects the gear relation (velocity) between master and slave axis.
	MC_GearInPos	MC_GearInPos establish a gear synchronization relationship between the master axis and the slave axis at the specified location.
	MC_Phasing	MC_Phasing specifies the phase shift value between the master and slave axis.
	DMC_MoveLink	DMC_MoveLink enables the slave axis to follow the master axis for synchronous positioning movement.
Administrative	MC_Power	MC_Power enables or disables the specific axis.
	MC_SetPosition	MC_SetPositionn changes the current position by shifting the coordinates of an axis.
	MC_ReadParameter	MC_ReadParameter reads a value of a specific axis parameter.
	MC_WriteParameter	MC_WriteParameter writes a value to a specific parameter.
	MC_ReadBoolParameter	MC_ReadBoolParameter reads the value of a specific Boolean parameter.
	MC_WriteBoolParameter	MC_WriteBoolParameter writes a Boolean value to a specific parameter.
	MC_ReadActualPosition	MC_ReadActualPosition reads the current axis position.
	MC_ReadActualVelocity	MC_ReadActualVelocity reads the actual axis velocity value.
	MC_ReadActualTorque	MC_ReadActualTorque reads the actual torque value of axis.
	MC_Reset	MC_Reset clears axis-related errors so that the error memory is available for new error messages.
	MC_ReadStatus	MC_ReadStatus reads the status of the specified axis.
	MC_ReadAxisError	MC_ReadAxisError reads the error information of axis.
	MC_CamTableSelect	MC_CamTableSelect selects the cam table for use with MC_CamIn.

Categories	Name	Description
	MC_TouchProbe	MC_TouchProbe records an axis position at the time when a trigger event occurs.
	MC_AbortTrigger	MC_AbortTrigger aborts the instruction MC_TouchProbe which are intended to capture trigger events.
	MC_DigitalCamSwitch	MC_DigitalCamSwitch uses the axis position to control a switch of a digital output.
Administrative	SMC_BacklashCompensation	SMC_BacklashCompensation is used to compensate for the backlash of gears.
	DMC_ChangeMechanism GearRation	DMC_ChangeMechanismGearRation modifies the ratio between user units and pulses, axis type and user units per rotation of the rotary axis.
	DMC_ReadMotionState	DMC_ReadMotionState reads the behavior state of the axis in motion.
	DMC_CamReadTappetStatus	DMC_CamReadTappetStatus reads the status of multiple tappets.
	DMC_CamReadTappetValue	DMC_CamReadTappetValue reads the data of one single tappet.
	DMC_CamWriteTappetValue	DMC_CamWriteTappetValue modifies the tappet data for the specified existing track.
	DMC_CamAddTappet	DMC_CamAddTappet adds a new tappet track at the end of the tappet table.
	DMC_CamDeleteTappet	DMC_CamDeleteTappet deletes the specified tappet track.
	DMC_CamReadPoint	DMC_CamReadPoint reads the data of one single cam point.
	DMC_CamWritePoint	DMC_CamWritePoint writes the data of one single cam point.
	DMC_AxesObserve	DMC_AxesObserve monitors the deviation between the master axis position and slave axis position and it will output a reminder when the deviation exceeds the allowed Setting Value.
	DMC_PositionLag	DMC_PositionLag sets the allowed range of lag error and observe whether the allowed position lag is exceeded
	DMC_SetTorqueLimit	DMC_SetTorqueLimit sets the maximum torque of an axis.
	DMC_SetSoftwareLimit	DMC_SetSoftwareLimit is used to enable, disable and set the upper and lower software limits
	DMC_CamKeyPointWrite	DMC_CamKeyPointWrite writes key cam points by selecting a curve type and generating corresponding cam curve based on related parameters. After the new cam curve is generated, the selected cam table will be changed accordingly.
	DMC_TouchProbeCyclically	DMC_TouchProbeCyclically can continuously record the captured position of an axis.
	DMC_CAMBounds	DMC_CAMBounds uses the cam table and inputs the expected maximum speed and acceleration of the master axis to obtain the maximum and minimum values of the estimated position, speed and acceleration of the slave axis when following.

Categories	Name	Description
	DMC_GetCamSlaveData	Input the axis position for DMC_GetCamSlaveData to get information about the slave axis of the specified cam table.

- **Single-axis Motion Control Instruction (Positioning axes)**

Categories	Name	Description
Positioning control	MC_Home_DML	MC_Home_DML controls the axis to perform the homing operation.
	MC_Stop_DML	MC_Stop_DML decelerates an axis to a stop.
	MC_Halt_DML	MC_Halt_DML halts an axis in a controllable way.
Positioning control	MC_MoveAbsolute_DML	MC_MoveAbsolute_DML controls the axis to move to the specified absolute target position at a specified behavior.
	MC_MoveRelative_DML	MC_MoveRelative_DML controls the axis to move a specified relative distance with a specified behavior.
Velocity Control	MC_MoveVelocity_DML	MC_MoveVelocity_DML performs velocity control on an axis in the position mode with a specified behavior and an average velocity.
	MC_VelocityControl_DML	MC_VelocityControl_DML controls the specified axis to move evenly according to the specified motion mode and speed in speed mode (VL).
Torque Control	MC_TorqueControl_DML	MC_TorqueControl_DML controls the torque according to the torque control mode of the servo drive.
Administrative	MC_Power_DML	MC_Power_DML enables or disables the specific axis.
	MC_ReadParameter_DML	MC_ReadParameter_DML reads a value of a specific axis parameter.
	MC_WriteParameter_DML	MC_WriteParameter_DML writes a value to a specific parameter.
	MC_ReadBoolParameter_DML	MC_ReadBoolParameter_DML reads the value of a specific Boolean parameter.
	MC_WriteBoolParameter_DML	MC_WriteBoolParameter_DML writes a Boolean value to a specific parameter.
	MC_Reset_DML	MC_Reset_DML clears axis-related errors.
	MC_ReadStatus_DML	MC_ReadStatus_DML reads the status of the specified axis.
	MC_ChangeAxisConfig_DML	MC_ChangeAxisConfig_DML modifies basic axis settings including the ratio between user units and pulse number, axis type and user units per rotary axis rotation.
	MC_ReinitDrive_DML	MC_ReinitDrive_DML re-initializes the axis.

- **Multi-axis Motion Control Instruction**

Categories	Name	Description
Group Motion	DMC_MoveLinearAbsolute	DMC_MoveLinearAbsolute controls a specified axis group to perform the absolute linear interpolation for a specified absolute position.
	DMC_MoveLinearRelative	DMC_MoveLinearRelative controls a specified axis group to perform the relative linear interpolation.
	DMC_MoveCircularAbsolute	DMC_MoveCircularAbsolute controls the axis group to perform circular or helical interpolation for a specified absolute target position.
	DMC_MoveCircularRelative	DMC_ MoveCircularRelative controls the axis group to perform circular or helical interpolation for a specified relative target position.
	DMC_GroupStop	DMC_GroupStop decelerates the group axes to a stop.
	DMC_GroupHalt	DMC_GroupHalt decelerates the axis group in motion to a pause.
	DMC_GroupInterrupt	DMC_GroupInterrupt makes the current motion pause but not stop, and it can be used with DMC_GroupContinue to restore the motion.
	DMC_GroupContinue	DMC_GroupContinue restores the interrupted motion of DMC_GroupInterrupt.
	DMC_GroupJog	DMC_GroupJog is used for the forward and reverse jog function of the axis group to the specified coordinates.
Administrative	DMC_GroupEnable	DMC_GroupEnable switches the axis group state from GroupDisable to GroupStandby.
	DMC_GroupDisable	DMC_GroupDisable sets the state of an axis group to GroupDisable.
	DMC_GroupReadParameter	DMC_GroupReadParameter reads axis group parameter.
	DMC_GroupWriteParameter	DMC_GroupWriteParameter writes axis group parameter.
	DMC_GroupReadStatus	DMC_GroupReadStatus reads the state of an axis group.
	DMC_GroupReadError	DMC_GroupReadError reads axis group errors.
	DMC_GroupReset	DMC_AddAxisToGroup adds a single axis to the axis group.
	DMC_AddAxisToGroup	DMC_RemoveAxisFromGroup removes a single axis from the axis group.
	DMC_RemoveAxisFromGroup	DMC_UngroupAllAxes removes all axes in the axes group.
	DMC_UngroupAllAxes	DMC_GroupReadSetPosition reads the current command position of the axis group.
	DMC_GroupReadSetPosition	DMC_GroupReadActPosition reads the current actual position of the axis group.
	DMC_GroupReadActPosition	DMC_AddAxisToGroup adds a single axis to the axis group.
	DMC_GroupPower	DMC_GroupPower controls the enablement, shutdown and immediate stop of all axes in the axis group.

Categories	Name	Description
	DMC_GroupSetOverride	DMC_GroupSetOverride changes the velocity of the axis group movement by override control factor.

A.1.2 By Model

- The supported model types are listed as follows:

Function Block	Model Type	
	AX-3	AX-8
MC_Home	●	●
MC_Stop	●	●
MC_Halt	●	●
MC_MoveAbsolute	●	●
MC_MoveRelative	●	●
MC_MoveAdditive	●	●
MC_MoveSuperimposed	●	●
MC_CamIn	●	●
MC_CamOut	●	●
MC_MoveVelocity	●	●
MC_PositionProfile	●	●
MC_VelocityProfile	●	●
MC_AccelerationProfile	●	●
MC_Jog	●	●
MC_GearIn	●	●
MC_GearOut	●	●
MC_GearInPos	●	●
MC_Phasing	●	●
MC_Power	●	●
MC_SetPosition	●	●
MC_ReadParameter	●	●
MC_WriteParameter	●	●
MC_ReadBoolParameter	●	●
MC_WriteBoolParameter	●	●
MC_ReadActualPosition	●	●
MC_ReadActualVelocity	●	●
MC_ReadActualTorque	●	●
MC_Reset	●	●
MC_ReadStatus	●	●
MC_ReadAxisError	●	●
MC_CamTableSelect	●	●
MC_TouchProbe	●	●
MC_AbortTrigger	●	●
MC_DigitalCamSwitch	●	●
MC_Home_DML	●	
MC_Stop_DML	●	

Function Block	Model Type	
	AX-3	AX-8
MC_Halt_DML	●	
MC_MoveAbsolute_DML	●	
MC_MoveRelative_DML	●	
MC_MoveVelocity_DML	●	
MC_TorqueControl_DML	●	
MC_Power_DML	●	
MC_ReadParameter_DML	●	
MC_WriteParameter_DML	●	
MC_ReadBoolParameter_DML	●	
MC_WriteBoolParameter_DML	●	
MC_Reset_DML	●	
MC_ReadStatus_DML	●	
MC_ChangeAxisConfig_DML	●	
MC_ReinitDrive_DML	●	
DMC_MoveVelocityStopByPos	●	
DMC_MoveLink	●	
DMC_MoveFeed	●	
DMC_CAMBounds	●	
DMC_TorqueControl	●	
DMC_VelocityControl	●	
DMC_MoveLinearAbsolute	●	
DMC_MoveLinearRelative	●	
DMC_MoveCircularAbsolute	●	
DMC_MoveCircularRelative	●	
DMC_GroupStop	●	
DMC_GroupHalt	●	
DMC_Home_P	●	
DMC_ImmediateStop_P	●	
DMC_GroupEnable	●	
DMC_GroupDisable	●	
DMC_GroupReadParameter	●	
DMC_GroupWriteParameter	●	
DMC_GroupReadStatus	●	
DMC_GroupReadError	●	
DMC_GroupReset	●	
DMC_GroupInterrupt	●	
DMC_GroupContinue	●	
DMC_CamReadTappetStatus	●	

Function Block	Model Type	
	AX-3	AX-8
DMC_CamReadTappetValue	●	
DMC_CamWriteTappetValue	●	
DMC_CamAddTappet	●	
DMC_CamDeleteTappet	●	
DMC_CamReadPoint	●	
DMC_CamWritePoint	●	
DMC_ChangeMechanismGearRation	●	
DMC_ReadMotionState	●	
DMC_AxesObserve	●	
DMC_PositionLag	●	
DMC_SetTorqueLimit	●	
DMC_SetSoftwareLimit	●	
DMC_CamKeyPointWrite	●	
DMC_TouchProbeCyclically	●	
DMC_GroupReadSetPositio	●	
DMC_GroupReadActPosition	●	
DMC_GroupJog	●	
DMC_AddAxisToGroup	●	
DMC_RemoveAxisFromGroup	●	
DMC_UngroupAllAxes	●	
DMC_MoveModulo	●	
DMC_Home_E	●	
SMC_BacklashCompensation	●	●
MC_VelocityControl_DML	●	
DMC_GroupSetOverride	●	
DMC_GetCamSlaveData	●	

A.1.3 By Letter

- A
  - MC_AbortTrigger
  - MC_AccelerationProfile
  - DMC_AxesObserve
  - DMC_AddAxisToGroup
- B
  - SMC_BacklashCompensation
- C
  - MC_CamIn
  - MC_CamOut
  - MC_CamTableSelect
  - MC_ChangeAxisConfig_DML
  - DMC_CamAddTappet
  - DMC_CamDeleteTappet
  - DMC_CamReadPoint
  - DMC_CamReadTappetStatus
  - DMC_CamReadTappetValue
  - DMC_CamWritePoint
  - DMC_CamWriteTappetValue
  - DMC_ChangeMechanismGearRation
  - DMC_CamKeyPointWrite
  - DMC_CAMBounds
- D
  - MC_DigitalCamSwitch
- G
  - MC_GearIn
  - MC_GearInPos
  - MC_GearOut
  - DMC_GroupDisable
  - DMC_GroupEnable
  - DMC_GroupHalt
  - DMC_GroupReadError
  - DMC_GroupReadStatus

DMC_GroupReset
DMC_GroupStop
DMC_GroupInterrupt
DMC_GroupContinue
DMC_GroupReadParameter
DMC_GroupWriteParameter
DMC_GroupReadSetPosition
DMC_GroupReadActPosition
DMC_GroupJog
DMC_GroupSetOverride
DMC_GetCamSlaveData

- H
  MC_Halt
  MC_Halt_DML
  MC_Home
  MC_Home_DML
  DMC_Home_P
  DMC_Home_E
- I
  DMC_ImmediateStop_P
- J
  MC_Jog
- M
  MC_MoveAbsolute
  MC_MoveAbsolute_DML
  MC_MoveAdditive
  MC_MoveRelative
  MC_MoveRelative_DML
  MC_MoveSuperimposed
  MC_MoveVelocity
  MC_MoveVelocity_DML
  DMC_MoveCircularAbsolute
  DMC_MoveCircularRelative
  DMC_MoveLinearAbsolute
  DMC_MoveLinearRelative
  DMC_MoveVelocityStopByPos
  DMC_MoveLink
  DMC_MoveFeed
  DMC_MoveModulo
- P
  MC_Phasing
  MC_PositionProfile
  MC_Power
  MC_Power_DML
  DMC_PositionLag
- R
  MC_ReadActualPosition
  MC_ReadActualTorque
  MC_ReadActualVelocity
  MC_ReadAxisError
  MC_ReadBoolParameter
  MC_ReadBoolParameter_DML
  MC_ReadParameter
  MC_ReadParameter_DML
  MC_ReadStatus
  MC_ReadStatus_DML
  MC_ReinitDrive_DML
  MC_Reset
  MC_Reset_DML
  DMC_ReadMotionState
  DMC_RemoveAxisFromGroup
- S
  MC_SetPosition
  MC_Stop
  MC_Stop_DML
  DMC_SetTorqueLimit
  DMC_SetSoftwareLimit
- T
  MC_TouchProbe
  MC_TorqueControl_DML
  DMC_TorqueControl
  DMC_TouchProbeCyclically
- U
  DMC_UngroupAllAxes
- V
  MC_VelocityProfile
  DMC_VelocityControl
  MC_VelocityControl_DML
- W
  MC_WriteBoolParameter
  MC_WriteBoolParameter_DML
  MC_WriteParameter
  MC_WriteParameter_DML
  MC_WriteBoolParameter

A.2 Data Type: Enumeration and Structure

- The Data Types listed below are Enumeration type:

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
MC_SOURCE	0: mcCommandedValue 1: mcActualValue	0: The commanded value of instruction 1: The actual value of motion axis	Function block: DMC_ReadMotionState Interface: *Source*
MC_StartMode	0: absolute 1: relative 2: ramp_in 3: ramp_in_pos 4: ramp_in_neg	0: Absolute mode 1: Relative mode 2: Ramp in mode 3: Positive Ramp in 4: Negative Ramp in mode	Function block: MC_CamIn Interface: *StartMode*
SMC_CAM TAPPETTYPE	0: TAPPET_pos 1: TAPPET_all 2: TAPPET_neg	0: When pass in a positive direction 1: When pass in both direction 2: When pass in a negative direction	Function block: MC_CamIn. Tappets.pTaps Interface: *ctt*
SMC_CAM TAPPET ACTION	0: TAPPETACTION_on 1: TAPPETACTION_off 2: TAPPETACTION_inv 3: TAPPETACTION_time	0: Switches ON 1: Switches OFF 2: Inverts 3: Switches on after a delay for a certain time period.	Function block: MC_CamIn. Tappets.pTaps Interface: *cta*
MC_TAPPET MODE	0: tp_mode_auto 1: tp_mode_demandposition 2: tp_mode_actualposition	0: Auto mode 1: Use set values 2: Use actual values	Function block: MC_DigitalCamSwitch Interface: *TappetMode*
DMC_BUFFER_MODE	0: aborting 1: buffered 2: blending_low 3: blending_previous 4: blending_next 5: blending_high	0: Any ongoing motion is aborted. 1: Start FB after current move has finished. 2: The velocity is blended with lowest velocity of both FBs 3: The velocity is blended with the velocity of the first FB. 4: The velocity is blended with velocity of the second FB. 5: The velocity is blended with highest velocity of both FBs	Function block: DMC_MoveLinearAbsolute DMC_MoveLinearRelative DMC_MoveCircularAbsolute DMC_MoveCircularRelative DMC_GroupHalt Interface: BufferMode

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
DMC_GROUP_TRANSITION_MODE	0: None 1: Overlap	0: The previous & next instructions follow BufferMode setting during blending, and there is no special transition mode. 1: The previous & next instructions do not follow BufferMode setting during blending, which makes the deceleration period of the previous instruction overlap the next instruction.	Function block: DMC_MoveLinearAbsolute DMC_MoveLinearRelative DMC_MoveCircularAbsolute DMC_MoveCircularRelative Interface: TransitionMode
DMC_CIRC_PLANE	0: XY_plane 1: YZ_plane 2: ZX_plane	0: The circle is parallel to the XY plane. 1: The circle is parallel to the YZ plane. 2: The circle is parallel to the ZX plane.	Function block: DMC_MoveCircularAbsolute DMC_MoveCircularRelative Interface: CircPlane
DMC_CIRC_MODE	0: radius 1: center 2: border	0: Defines radius of a circle. 1: Defines a center point of a circle. 2: Defines a point on the circle which is crossed on the path from the starting to the end point.	Function block: DMC_MoveCircularAbsolute DMC_MoveCircularRelative Interface: CircMode
DMC_CIRC_PATHCHOICE	0: CLOCKWISE 1: COUNTER_CLOCKWISE	0: Clockwise 1: Counterclockwise	Function block: DMC_MoveCircularAbsolute DMC_MoveCircularRelative Interface: PathChoice
DMC_GROUP_STATE	0: GroupDisabled 1: GroupStandby 2: GroupMoving 3: GroupHoming 4: GroupStopping 5: GroupErrorstop	0: Group state is Disabled 1: Group state is Standby 2: Group state is Moving 3: Group state is Homing 4: Group state is Stopping 5: Group state is Errorstop	Function block: DMC_AXIS_GROUP_REF Interface: GroupState
DMC_GROUP_RAMP_TYPE	0: Trapezoid 1: S_Curve	0: The velocity curve is trapezoidal. 1: The velocity curve is S Curve.	Function block: DMC_AXIS_GROUP_REF Interface: RampType

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
DMC_GROUP_PARAMETER	16: PARAM_RAMP_TYPE 17: PARAM_MAX_VELOCITY_LIMIT 18: PARAM_MAX_ACCELERATION_LIMIT 19: PARAM_MAX_DECELERATION_LIMIT 21: PARAM_PLANNING_PRIORITY 22: PARAM_STOP_METHOD 23: PARAM_FB_VADJ_TARGET 24: PARAM_VELOCITY_WARNING_PERCENTAGE 25: PARAM_ACCELERATION_WARNING_PERCENTAGE 26: PARAM_DECELERATION_WARNING_PERCENTAGE 28: PARAM_RADIUS_CORRECTION_PERCENTAGE	16: Velocity ramp type 17: Max. velocity limit 18: Limit on max. acceleration 19: Limit on max. deceleration 21: Priority items of velocity ramp planning 22: Stop method 23: Applied target of function block velocity/acceleration/ deceleration/jerk 24: Velocity warning range 25: Acceleration warning range 26: Deceleration warning range 28: Allowable correction range of radius	Function block: DMC_GroupReadParameter DMC_GroupWriteParameter Pin: Parameter
R2R_TENSION_CTRL_MODE	0: TensionCloseLoop_SpeedMode 1: LineSpeedCloseLoop_SpeedMode 2: TensionCloseLoop_TorqueMode 3: TensionOpenLoop_TorqueMode	0: Tension closed loop, speed mode 1: Linear velocity closed loop, velocity mode 2: Tension closed loop, torque mode 3: Tension open circuit, torque mode	Function Block: R2R_Configuration Pin: TensionCtrlMode
R2R_WINDING_MODE	0: Rewind 1: Unwind	0: Rewind mode 1: Unwinding mode	Function Block: R2R_Configuration Pin: WindingMode
R2R_LINE_SPEED_SOURCE	0: R2R_Run_lrLineSpeedValue 1: AVI 2: ACI 3: PG_CARD 4: DFM_DCM 5: MI6MI7	0: R2R_Run_lrLineSpeedValue 1: Analog input AVI 2: Analog input ACI 3: Pulse PG card input 4: DFM-DCM pulse input 5: Pulse input via MI6 / MI7 terminals	Function Block: R2R_Configuration Pin: LineSpeedSource
R2R_TENSION_TARGET_SOURCE	0: R2R_Run_uiTensionTargetValue 1: AVI 2: ACI	0: R2R_Run_uiTensionTargetValue 1: Analog input AVI 2: Analog input ACI	Function Block: R2R_Configuration Pin: TensionTargetSource

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
R2R_TENSION_TARGET_SOURCE_AT_ZERO_SPEED	0: Disable 1: R2R_Run_uiTensionTargetValue_AtZeroSpeed 2: AVI 3: ACI	0: no function 1: R2R_Run_uiTensionTargetValue_AtZeroSpeed 2: Analog input AVI 3: Analog input ACI	Function Block: R2R_Configuration Pin: TensionTargetSource_AtZeroSpeed
R2R_PID_TARGER_SOURCE	0: R2R_Run_lrPID_TargetValue 1: AVI 2: ACI	0: R2R_Run_lrPID_TargetValue 1: Analog input AVI 2: Analog input ACI	Function Block: R2R_Configuration Pin: PIDTargetSource
R2R_PID_FEEDBACK_SOURCE	0: AVI 1: ACI 2: MI6MI7	0: Analog input AVI 1: Analog input ACI 2: Pulse input	Function Block: R2R_Configuration Pin: PIDFeedbackSource
R2R_PID_ADAPTABILITY_REFERENCE_SOURCE	0: Disable 1: RollDiameter 2: Freq	0: no function 1: Roll diameter 2: Frequency	Function Block: R2R_Configuration Pin: PIDAdaptabilityReferenceSource
R2R_ROLL_DIAMETER_SOURCE	0: R2R_Run_lrLineSpeedValue 1: AVI 2: ACI 3: ThicknessIntegrate_Motor_Encorder_PG1 4: ThicknessIntegrate_Motor_Encorder_PG2 5: ThicknessIntegrate_Motor_Encorder_MI67 6: ThicknessIntegrate_Motor_CloseSW_MI7 7: ThicknessIntegrate_Winding_Encorder_PG2 8: ThicknessIntegrate_Winding_Encorder_MI67 9: ThicknessIntegrate_Winding_CloseSW_MI7	0: Calculated via linear velocity 1: Analog input AVI 2: Analog input ACI 3: Calculated by the thickness integral, the encoder at the motor end is input through the PG1 card 4: Calculated by the thickness integral, the encoder at the motor end is input through the PG2 card 5: Calculated by the thickness integral, the encoder at the motor end is input via MI67 6: Calculated by the thickness integral, the encoder at the motor end is input via MI7 7: Calculated by the thickness integral, the encoder at the reel end is input via the PG2 card 8: Calculated by the thickness integral, the encoder at the reel end is input via MI67 9: Calculated by the thickness integral, the encoder at the reel end is input via MI7	function block: R2R_RollDiameter Pin: RollDiameterSource
R2R_MATERIAL_THICKNESS_GAIN	0: millimeter 1: centimeter	0: mm unit 1: cm unit	function block: R2R_RollDiameter Pin: MaterialThicknessGain

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
MC_DIRECTION	-1: negative 0: shortest 1: positive 2: current 3: fastest	-1: reverse rotation 0: shortest path 1: Forward rotation 2: current direction 3: The fastest path	function block: DMC_MoveFeed Pin: Direction
DMC_MOVEMODE	0: ABSOLUTE 1: RELATIVE 2: VELOCITY	0: absolute mode 1: Relative mode 2: Speed Mode	function block: DMC_MoveFeed Pin: MoveMode
DMC_LINKOPTION	0: COMMANDEEXECUTION 1: TRIGGERDETECTION 2: MASTERREACH	0: start immediately 1: Drive side startup 2: Set the spindle position to start	function block: DMC_MoveMoveLink Pin: LinkOption
DMC_STARTDISTANCEMODE	0: ABSOLUTE 1: RELATIVE	0: absolute mode 1: Relative mode	function block: DMC_MoveMoveLink Pin: StartDistanceMode
DMC_COORD_SYSTEM	0: ACS 1: MCS 2: WCS（Reserved） 3: PCS_1（Reserved） 4: PCS_2（Reserved） 5: TCS（Reserved）	0: Motion axis coordinates 1: Mechanical coordinates 2: world coordinates (reserved) 3: Workpiece coordinate 1 (reserved) 4: Workpiece coordinate 2 (reserved) 5: Tool coordinates (reserved)	function block: DMC_MoveDirectAbsolute DMC_MoveDirectRelative DMC_GroupReadSetPosition DMC_GroupReadSetPosition Pin: CoordSystem

- The Data Types listed below are Structure type:

Data Type	Function Block	Definition
AXIS_REF_SM3*	Applied to MC_ / DMC_ function block.	This structure contains all the required data and parameters for axis motion.
DMC_AXIS_GROUP_REF*	Applied to DMC_Group function block.	This structure contains all the required data and parameters for group motion.
AXIS_REF_VIRTUAL_SM3	Applied to MC_ / DMC_ function block.	This structure contains all the required data and parameters for virtual axis motion.
TRIGGER_REF	MC_TouchProbe MC_AbortTrigger	Includes the trigger information ● The specified trigger channel ● The trigger condition and the mode (Triggered on the rising or falling edge of the trigger signal.)
MC_CAM_REF	MC_CamTableSelect	This structure contains information of the CAM table and points.

***Note**: refer to AX-3 Series operational manual for more details of structural type parameters.

A.3 Error Codes and Troubleshooting

A.3.1 For Synchronous Axes

When an error occurs, you can troubleshoot errors through error codes and the corresponding indicators. refer to **AX-3 Operational Manual** for more details of troubleshooting process.

The following table lists the error codes and the contents of the errors:

Error code	Description	Contents	Corrective Action
0x00000	SMC_NO_ERROR	No error messages.	
0x00001	SMC_DI_GENERAL_ COMMUNICATION_ ERROR	Communication error.	Make sure the servo's network cable is properly plugged, then reset EtherCAT master and execute MC_Reset.
0x00002	SMC_DI_AXIS_ERROR	Axis error.	Check the error information and troubleshoot the error according to each servo's user manual, then execute MC_Reset.
0x00003	SMC_DI_FIELDBUS_ LOST_SYNCRONICITY	Loss of synchronicity.	Execute SMC3_ReinitDrive. If the error occurs frequently, refer to Task max cycle time and increase EtherCAT DC time.
0x0000A	SMC_DI_SWLIMITS_ EXCEEDED	Software limit errors.	Execute MC_Reset and run reversely away from the limit.
0x0000B	SMC_DI_HWLIMITS_ EXCEEDED	Hardware limit errors.	Execute MC_Reset.
0x0000C	SMC_DI_LINEAR_AXIS_ OUTOFRANGE	Incremental position of a linear axis is out of range.	Execute MC_Reset
0x0000D	SMC_DI_HALT_OR_ QUICKSTOP_NOT_ SUPPORTED	Not support Halt or Quickstop state of drivers.	Execute MC_Reset
0x00010	SMC_DI_ POSITIONLAGERROR	Excessive position error.	Execute MC_Reset
0x00011	SMC_DI_HOMING_ ERROR	Homing error occurs.	Execute MC_Reset
0x00014	SMC_REGULATOR_OR_ START_NOT_SET	The motion FB cannot be run under the current axis state.	Enable the servo and execute MC_Reset, then re-execute the motion FB.
0x00015	SMC_WRONG_ CONTROLLER_MODE	The axis is under wrong controller mode.	Execute SMC_SetControllerMode to switch the axis to the proper controller mode.
0x00019	SMC_INVALID_ACTION_ FOR_LOGICAL	Invalid action for logical axes.	Do not perform improper operation to logical axes, such as powering on logical axes.
0x0001E	SMC_FB_WASNT_ CALLED_DURING_ MOTION	Function blocks cannot be called in movement state.	execute FBs in bus cycle task.
0x0001F	SMC_AXIS_IS_NO_AXIS_REF	AXIS_REF variable type errors.	The pointer must points to the register.
0x00020	SMC_AXIS_REF_ CHANGED_DURING_ OPERATION	AXIS_REF variables have been changed while the modules being activated.	Execute MC_Reset and do not change the axis input of the function block.
0x00021	SMC_FB_ACTIVE_AXIS_ DISABLED	Execute servo off while axis is in motion.	Power on the servo and execute MC_Reset.

Error code	Description	Contents	Corrective Action
0x00022	SMC_AXIS_NOT_READY_FOR_MOTION	The motion instruction cannot be run under the current axis state.	As a result that the axis is not able to be controlled, check if the state is power ON or an error exists, then enables the axis or execute MC_Reset depending on the situation.
0x00023	SMC_AXIS_ERROR_DURING_MOTION	Error occurs during motion.	refer to the servo user manual to check on the error information, then execute MC_Reset.
0x00028	SMC_VD_MAX_VELOCITY_EXCEEDED	Exceeds the maximum velocity limit fMaxVelocity.	Troubleshoots with MC_Reset.
0x00029	SMC_VD_MAX_ACCELERATION_EXCEEDED	Exceeds the maximum acceleration limit fMaxAcceleration.	Troubleshoots with MC_Reset.
0x0002A	SMC_VD_MAX_DECELERATION_EXCEEDED	Exceeds the maximum deceleration limit fMaxDeceleration.	Troubleshoots with MC_Reset.
0x00032	SMC_3SH_INVALID_VELACC_VALUES	Invalid Setting Values of velocity or acceleration.	insert the value of velocity or acceleration again and then re-run the function block.
0x00033	SMC_3SH_MODE_NEEDS_HWLIMIT	The current mode needs the hardware limit to be activated.	bIgnoreHWLimit should not be True in the current mode. select the proper mode.
0x00046	SMC_SCM_NOT_SUPPORTED	The mode is not supported.	Device does not support this mode, re-select the proper mode then activate the function block.
0x00047	SMC_SCM_AXIS_IN_WRONG_STATE	The controller mode cannot be changed in the current state.	Use MC_Reset to troubleshoot the error.
0x00048	SMC_SCM_INTERRUPTED	SMC_SetControllerMode is interrupted by MC_Stop or errorstop.	reactivate the function block.
0x0004B	SMC_ST_WRONG_CONTROLLER_MODE	The axis is under the wrong controller mode.	Use MC_Reset to troubleshoot the error.
0x00050	SMC_RAG_ERROR_DURING_STARTUP	Error occurs when the axis group is activated.	Make sure the configuration is normal and re-execute SMC3_ReinitDrive.
0x00051	SMC_RAG_ERROR_AXIS_NOT_INITIALIZED	The axis is not in the required state.	SMC3_ReinitDrive cannot be run when EtherCAT Master is in Initial state.
0x00055	SMC_PP_WRONG_AXIS_TYPE	Virtual axes or logical axes are not supported by the function block	SMC3_PersistPosition cannot be used on the virtual axis.
0x00056	SMC_PP_NUMBER_OF_ABSOLUTE_BITS_INVALID	Invalid absolute bits, which must be within 8~32 bits.	The value input to usiNumberOfAbsoluteBits of SMC3_PersistPositionSingleturn is incorrect, re-enter the value.
0x0005A	SMC_CGR_ZERO_VALUES	Invalid value.	Change the values of dwRatioTechUnitsDenomand to non-zero values and then re-run the function block.
0x0005B	SMC_CGR_DRIVE_POWERED	The gear ratio parameters of the drive cannot be modified when it is under controlled.	Make the axis enter Disable state, then re-run the function block.

Error code	Description	Contents	Corrective Action
0x0005C	SMC_CGR_INVALID_ POSPERIOD	Invalid position period (less than or equal to 0, or exceeds half the width of the band)	When iMovementType = 0, fPositionPeriod is set to a value greater than zero and smaller than half the value of dwBusBandWidth.
0x0005D	SMC_CGR_POSPERIOD_NOT_INTEGRAL	The increment of the period is not integral and the case of modulo values is completed by the drive.	After modifying the parameters of fPositionPeriod, re-run the function block.
0x0006E	SMC_P_FTASKCYCLE_ EMPTY	There's no cycle information in the axis.(fTaskCycle = 0)	Change the value of TaskCycle into a non-zero value.
0x00078	SMC_R_NO_ERROR_TO_RESET	There's no errors after using MC_Reset.	Before execute the function block, check if there're any errors in the specified axis.
0x0007A	SMC_R_ERROR_NOT_ RESETTABLE	The error is not resettable.	Before reactivate MC_Reset, make sure all errors in the drive have been removed.
0x00083	SMC_RP_REQUESTING_ERROR	check the error code output by the FB ReadDriveParameter if an error occurs when attempts to communicate to the drive.	1. The OD you're trying to access does not exist, confirm the correct OD input. 2. Adjust MAX_MAILBOX_CHANNELS and MAX_SDO_CHANNELS in IODrvEtherCAT to 128.
0x00084	SMC_RP_DRIVE_ PARAMETER_NOT_ MAPPED	The parameter is not mapped to a specific drive.	The parameter you're trying to access does not exist.
0x0008D	SMC_WP_SENDING_ ERROR	Error code to the FB WriteDriveParameter.	The OD you're writing does not exist, confirm the correct OD input.
0x0008E	SMC_WP_DRIVE_ PARAMETER_NOT_ MAPPED	Enter a parameter number of a non-existent axis.	The written parameter does not exist.
0x000AA	SMC_H_AXIS_WASNT_ STANDSTILL	The axis is not in standstill state.	Re-execute the FB after the axis enters standstill state.
0x000AB	SMC_H_AXIS_DIDNT_ START_HOMING	Errors occur while homing.	Before execute SMC3_ReinitDrive, make sure the drive you're using is complied with standards and there's no existing error.
0x000AC	SMC_H_AXIS_DIDNT_ ANSWER	The drive does not answer after the homing is completed.	Before execute SMC3_ReinitDrive, make sure the drive you're using is complied with standards and there's no existing error.
0x000AE	SMC_H_AXIS_IN_ ERRORSTOP	The homing mode cannot be run as a result because the drive is in errorstop state.	Re-execute the FB after the axis leaving ErrorStop state.
0x000B5	SMC_MS_INVALID_ ACCDEC_VALUES	Invalid Setting Value of velocity or acceleration.	Re-execute the FB after correcting the input value of "Deceleration".
0x000B7	SMC_MS_AXIS_IN_ ERRORSTOP	Drive in operating state Errorstop.	Re-execute the FB after the axis leaving ErrorStop state.
0x000B8	SMC_BLOCKING_MC_ STOP_WASNT_CALLED	set Execute to falling edge when the axis is locked and MC_Stop cannot be called.	Check the input setting of MC_Stop and re-execute the FB.
0x000B9	SMC_MS_AXIS_ ALREADY_STOPPING	A stop cannot be aborted while executing MC_Stop.	Re-execute the FB after the axis leaving Stopping state.

Error code	Description	Contents	Corrective Action
0x000C9	SMC_MA_INVALID_ VELACC_VALUES	Invalid values of velocity or acceleration.	Re-execute the FB after correcting the input values of “Velocity”, “Acceleration”, “Deceleration” and “Jerk”.
0x000E2	SMC_MR_INVALID_ VELACC_VALUES	Invalid values of velocity or acceleration.	Re-execute the FB after correcting the input values of “Velocity”, “Acceleration”, “Deceleration” and “Jerk”.
0x000FB	SMC_MAD_INVALID_ VELACC_VALUES	Invalid values of velocity or acceleration.	Re-execute the FB after correcting the input values of “VelocityDiff”, “Acceleration”, “Deceleration” and “Jerk”.
0x00114	SMC_MSI_INVALID_ VELACC_VALUES	Invalid values of velocity or acceleration.	Re-execute the FB after correcting the input values of “VelocityDiff”, “Acceleration” and “Deceleration”.
0x00116	SMC_MSI_INVALID_ EXECUTION_ORDER	An error will occur if activates the second MC_MoveSuperimposed while the first one is still being run.	Re-execute the FB after the execution of first MC_MoveSuperimposed is completed.
0x0012D	SMC_MV_INVALID_ ACCDEC_VALUES	Invalid values of velocity or acceleration.	Re-execute the FB after correcting the input values of “Velocity”, “Acceleration”, “Deceleration” and “Jerk”.
0x0012E	SMC_MV_DIRECTION_ NOT_APPLICABLE	Direction = shortest/fastest is not applicable.	After correcting the input value of “Direction” to be not in shortest / fastest state, re-run the function block.
0x00145	SMC_PP_ARRAYSIZE	Incorrect array size.	re-enter the correct ArraySize, then re-execute the function blocks.
0x00146	SMC_PP_STEP0MS	Delta_time is not allowed to be set to 0.	re-enter the correct Delta_Time, then re-execute the function blocks.
0x0015E	SMC_VP_ARRAYSIZE	Incorrect array size	re-enter the correct ArraySize, then re-execute the function blocks.
0x0015F	SMC_VP_STEP0MS	Delta_time is not allowed to be set to 0.	re-enter the correct Delta_Time, then re-execute the function blocks.
0x00177	SMC_AP_ARRAYSIZE	Incorrect array size.	re-enter the correct ArraySize, then re-execute the function blocks.
0x00178	SMC_AP_STEP0MS	Delta_time is not allowed to be set to 0.	re-enter the correct Delta_Time, then re-execute the function blocks.
0x00190	SMC_TP_ TRIGGEROCCUPIED	The trigger has been activated.	Correct TriggerInput.bActive back to False, then re-run the function block.
0x00191	SMC_TP_COULDNT_SET_WINDOW	The driver interface does not support Mask function.	The specified devices do not support Window functions, turn off Window functions to re-run the function block.
0x0019A	SMC_AT_ TRIGGERNOTOCCUPIED	Triggering has been reset.	Check if MC_TouchProbe is run and the axis position has not been captured then re-activate the function block.
0x001AA	SMC_MCR_INVALID_ VELACC_VALUES	Invalid values of velocity or acceleration.	Correct the input value of “Velocity”, “EndVelocity”, “Deceleration” and “Jerk”, then re-run the function block.
0x001C3	SMC_MCA_INVALID_ VELACC_VALUES	Invalid values of velocity or acceleration.	Correct the input value of “Velocity”, “EndVelocity”, “Deceleration” and “Jerk”, then re-run the function block.

Error code	Description	Contents	Corrective Action
0x001C5	SMC_MCA_DIRECTION_ NOT_APPLICABLE	Cannot set the shortest distance.	After correcting the input value of “EndVelocityDirection” to be not in shortest / fastest state, re-run the function block.
0x001DB	SMC_SDL_INVALID_ AXIS_STATE	SMC_ChangeDynamic Limits can only be called in standstill or power_off state.	Check if the axis is in power_off or standstill state and then re-activate the function block.
0x001DC	SMC_SDL_INVALID_ VELACC_VALUES	Invalid values of velocity, acceleration, deceleration and jerk.	After fixing the input value if "fMaxVelocity", fMaxAcceleration", "fMaxDeceleration" and "fMaxJerk", re-run the function block.
0x00258	SMC_CR_NO_TAPPETS_ IN_CAM	There’re no tappets set in the CAM.	set tappets in the cam table and then re-run the function block.
0x00259	SMC_CR_TOO_MANY_ TAPPETS	The number of tappet groupID exceeds MAX_NUM_TAPPETS	As a result of too many tappets in the cam table, you have to modify the number before re-executing the function block.
0x00271	SMC_CI_NO_CAM_ SELECTED	No cam is selected.	enter the correct value given by MC_CamTableSelect after it’s successfully run to the input “CamTableID” and then re-run the function block.
0x00272	SMC_CI_MASTER_OUT_ OF_SCALE	The master exceeds the valid scale.	1. Execute MC_Reset to make the axis back to standstill state and check the inputs of MC_CamTableSelect. 2. Make sure that the cam master position, which is calculated by "Periodic" /"MasterAbsolute" of MC_CamTableSelect and "MasterCompensation" of MC_CamIn, is in the range of master scale on the cam table before you re-run the function block.
0x00273	SMC_CI_RAMPIN_ NEEDS_VELACC_ VALUES	The value of Velocity and acceleration must be set in the function of ramp_in.	1. Execute MC_Reset to make the axis back to standstill state and check the StartMode input. 2. When "StartMode" is set to "ramp_in"/ "ramp_in_pos"/"ramp_in_neg", the input values of "VelocityDiff"/"Acceleration"/ "Deceleration" need to be non-zero. Then you can re-run the function block.
0x00274	SMC_CI_SCALING_ INCORRECT	Scaling variables fEditor/TableMasterMin/Max are not correct	1. Execute MC_Reset to make the axis back to standstill state and check the inputs. 2. Make sure that the max values must be bigger than the min values in fEditor / fTable while using a cam table not in "XYVA" format. Then you can re-run the function block.

Error code	Description	Contents	Corrective Action
0x00275	SMC_CI_TOO_MANY_ TAPPETS_PER_CYCLE	Activate too many tappets in the same period.	Modify the tappets on the cam table and make sure that there're no too many tappets gathering on the same position. After download the cam table again, re-run the function block.
0x00280	SMC_CB_NOT_ IMPLEMENTED	The selected cam format is not run.	Modify the cam table format to the one supported by the function block, then re-run the function block.(Currently only support "XYVA" format)
0x002A3	SMC_GI_RATIO_DENOM	RatioDenominator = 0	Change the value of "RatioDenominator" to be non-zero and re-run the function block.
0x002A4	SMC_GI_INVALID_ACC	Invalid value of acceleration.	1. Execute MC_Reset to make the axis back to standstill state and check the inputs. 2. Make sure the value of "Acceleration" is greater than zero, then re-run the function block.
0x002A5	SMC_GI_INVALID_DEC	Invalid value of deceleration.	1. Execute MC_Reset to make the axis back to standstill state and check the inputs. 2. Make sure the value of "Deceleration" is greater than zero, then re-run the function block.
0x002A6	SMC_GI_MASTER_ REGULATOR_CHANGED	The master state (Enable/Disable) is changed without the permission.	Execute MC_Reset to make the axis back to standstill state and re-run the function block.
0x002A7	SMC_GI_INVALID_JERK	Invalid jerk value.	1. Execute MC_Reset to make the axis back to standstill state and check the inputs. 2. Make sure the value of "Jerk" is greater than zero, then re-run the function block.
0x002D5	SMC_PH_INVALID_ VELACCDEC	Invalid values of velocity, acceleration and deceleration.	Make sure the values of "Velocity", "Acceleration" and "Deceleration" are non-zero before re-run the function block.
0x002EE	SMC_NO_CAM_REF_ TYPE	The chosen cam type is not MC_CAM_REF	Correct the input variable to the correct variable in "MC_CAM_REF" type.
0x002EF	SMC_CAM_TABLE_ DOES_NOT_COVER_ MASTER_SCALE	The curve data does not include the master scale, xStart and xEnd, on the CamTable.	Correct the values of "xStart" and "xEnd" to make these two values are included in the master scale.
0x002F0	SMC_CAM_TABLE_ EMPTY_MASTER_ RANGE	There's no master range on the cam table.	Fix the "xStart" and "xEnd" on the cam table to make the "xEnd" value greater than the value of "xStart".
0x002F2	SMC_CAM_TABLE_ INVALID_SLAVE_ MINMAX	Invalid min/ max values of the slave axis on the cam table.	Make sure that the values of fTableSlaveMin and fTableSlaveMax are not equal before you re-run the function block.
0x00307	SMC_GIP_MASTER_ DIRECTION_CHANGE	The master axis changes its direction while being synchronizing with the slave axis.	Execute MC_Reset to make the axis back to standstill state and re-execute the FB. At the same time, do not reverse the master direction when "StartSync" is True.

Error code	Description	Contents	Corrective Action
0x00308	SMC_GIP_SLAVE_ REVERSAL_CANNOT_BE_AVOIDED	AvoidReversal is set but cannot avoid the slave axis to be reversed.	Adjust the input values of "MasterSyncPosition", "SlaveSyncPosition" and "MasterStartDistance" as well as the velocity of master and slave axis after being coupled. Then re-run the function block.
0x00309	SMC_GIP_AVOID_ REVERSAL_FOR_ FINITE_AXIS	AvoidReversal cannot be configured while using linear axes.	Change the "Axis type" of slave to "Modulo"(Need to re-download) or set the input "AvoidReversal" to False, then re-run the function block.
0x186A0	DMC_TPC_INVALID_PDO_MAPPIN G	PDO mapping error.	do not configure Touch probe function (60B8h) in PDO.
0x186A1	DMC_TPC_TRIGGER OCCUPIED	Trigger has been created.	do not execute the function block with MC_TouchProbe, which has been run.
0x186A2	DMC_TPC_ETC_CO_ FIRST_ERROR	SDO read-write error	Invalid SDO command, check the related configuration.
0x186A3	DMC_TPC_ETC_CO_ OTHER_ERROR	Communication error	Cannot find the corresponding master station, check the master status first.
0x186A4	DMC_TPC_ETC_CO_ DATA_OVERFLOW	Communication error	The size of SDO is too large to be sent. re-execute the FB after modification.
0x186A5	DMC_TPC_ETC_CO_ TIMEOUT	Communication error	SDO time outs. check if there's a corresponding OD to the servo.
0x186A8	DMC_TPC_ECAT_ MASTER_DISABLE	Communication error	Master initialization failed. check the status of the master station.
0x186B3	DMC_MF_INVALID_ACCDEC_VALU ES	Invalid velocity or acceleration value	After correcting the "Velovity" , "Acceleration" , "Deceleration" input values, execute the function block again.
0x186B4	DMC_MF_AXIS_NOT_READY_FOR_ MOTION	Axis status cannot execute motion control instruction	The axis is in a state that cannot be controlled. confirm whether the target axis is Power on or in an error state, and enable the axis or execute MC_Reset on the axis according to the situation.
0x186B5	DMC_MF_AXIS_ERROR_DURING_ MOTION	An error occurred during operation	confirm the servo error information, refer to the servo manual to eliminate the error, and execute MC_Reset.
0x186B6	DMC_MF_REGULATOR_OR_START _NOT_SET	Axis status cannot execute motion control instruction	After starting the servo, execute MC_Reset and re-execute the motion function block.
0x186B7	DMC_MF_TP_TRIGGEROCCUPIED	There are other MC_TouchProbes executing	After making sure that no other MC_TouchProbe is executing in the program, re-run the function block.
0x186B8	DMC_MF_TP_COULDNT_SET_WIN DOW	Cannot support window mode	The drive cannot support the window mode, turn off the window mode and restart the function block.
0x186B9	DMC_MF_TP_COMM_ERROR	MC_TouchProbe function block command error	TouchProbe related function block command error, remove the error and re-run the function block.
0x186C4	DMC_ML_MASTER_DISTANCE_INV ALID_VALUES	Target distance entered incorrectly	confirm that the sum of the acceleration and deceleration distances cannot be greater than or equal to the total moving distance, and the three inputs cannot be

Error code	Description	Contents	Corrective Action
			negatives; restart the function block after correction.
0x186C5	DMC_ML_AXIS_NOT_READY_FOR_MOTION	Axis status cannot execute motion control instruction	The axis is in a state that cannot be controlled. confirm whether the target axis is Power on or in an error state, and enable the axis or execute MC_Reset on the axis according to the situation.
0x186C6	DMC_ML_AXIS_ERROR_DURING_MOTION	An error occurred during operation	confirm the servo error information, refer to the servo manual to eliminate the error, and execute MC_Reset.
0x186C7	DMC_ML_REGULATOR_OR_START_NOT_SET	Axis status cannot execute motion control instruction	After starting the servo, execute MC_Reset and re-execute the motion function block.
0x186C8	DMC_ML_TP_TRIGGEROCCUPIED	There are other MC_TouchProbes executing	After making sure that no other MC_TouchProbe is executing in the program, re-run the function block.
0x186C9	DMC_ML_TP_COMM_ERROR	MC_TouchProbe function block command error	TouchProbe related function block command error, remove the error and re-run the function block.
0x186D4	DMC_CB_CAM_TABLE_DATA_EMPTY	Cam table no information	Check if the Cam table has no data
0x186D5	DMC_CB_CAM_DATATYPE_NOT_SUPPORT	Cam table format error	Check if the Cam table format is correct
0x187CC	DMC_CRTS_TAPPETID_VALUE_OUTOFRANGE	The value of track ID of the tappet is set out of range.	Re-execute the FB after correcting Track ID.
0x187D2	DMC_CRTV_TAPPETID_VALUE_OUTOFRANGE	The value of track ID of the tappet is set out of range.	Re-execute the FB after correcting Track ID.
0x187D3	DMC_CRTV_NO_TAPPETID	The track ID to read does not exist.	Re-execute the FB after checking the tappet inputs.
0x187D4	DMC_CRTV_NO_TAPPETS_IN_CAM	There's no tappets set in the cam table.	Re-execute the FB after adding new tappets.
0x187DA	DMC_CWTV_INVALID_TAPPETID	Invalid Track ID.	Re-execute the FB after correcting Track ID.
0x187DB	DMC_CWTV_INVALID_MASTER_POS	Invalid master position.	correct the input of master position, then re-execute the FB.
0x187DC	DMC_CWTV_CAM_TABLE_NUM_EXCEED_LIMIT	The number of cam table exceeds the limit.	The limit has been reached. Cannot write in more tappets.
0x187DD	DMC_CWTV_TAPPETID_NOT_FOUND	The track ID to modify does not exist.	Re-execute the FB after correcting Track ID.
0x187DE	DMC_CWTV_TAPPET_NUM_EXCEED_LIMIT	The number of tappets exceeds the limit.	Re-execute the FB after checking the tappet number.
0x187DF	DMC_CWTV_INVALID_MODE	Tappet input is not an existed mode.	Correct the tappet mode and re-execute the FB.
0x187E4	DMC_CAT_INVALID_MASTER_POS	The user-defined master position is out of range.	Re-execute the FB after correcting the master position.
0x187E5	DMC_CAT_CAM_TABLE_NUM_EXCEED_LIMIT	The number of cam table exceeds the limit.	The limit has been reached. Cannot write in more tappets.
0x187E6	DMC_CAT_TAPPET_NUM_EXCEED_LIMIT	The number of tappets exceeds the limit.	Re-execute the FB after checking the tappet number.
0x187E7	DMC_CAT_NO_TAPPET_TO_BE_ADDED	No tappet action set in the input variable.	There're no newly-added tappets in the input data. confirm that either PositiveMode or NegativeMode is not

Error code	Description	Contents	Corrective Action
			set to TAPPETACTION_none before re-run the function block.
0x187E8	DMC_CAT_INVALID_MODE	Tappet input is not an existed mode.	Correct the tappet mode and re-execute the FB.
0x187ED	DMC_CDT_NO_ TAPPETS_IN_CAM	There's no tappet in the tappet table.	Re-execute the FB after specifying a tappet table which has tappets in it.
0x187EE	DMC_CDT_CAM_TABLE_NUM_EXCEED_LIMIT	The number of cam table exceeds the limit.	The limit has been reached. Cannot write in more tappets.
0x187F4	DMC_CRP_INVALID_ POINTNUM	Invalid point number.	Check if the point number of specified data is more than the point number of cam data. Re-execute the FB after modification.
0x187FA	DMC_CWP_INVALID_ POINTNUM	Invalid point number.	Check if the point number of specified data is more than the point number of cam data. Re-execute the FB after modification.
0x187FB	DMC_CWP_INVALID_ MASTERPOS	Invalid master position.	check if the master position of data point to be modified exceeds the master position of the front and back point. Re-execute the FB after modification.
0x18801	DMC_TC_INVALID_VALUES	Invalid value	Confirm pin input parameter value. Re-execute the FB after modification.
0x18802	DMC_TC_FB_CONFLICT	Function trigger repeat	FB DMC_TorqueControl is being run, and only one FB DMC_TorqueControl is allowed to be run at the same time.
0x18803	DMC_TC_SDO_RW_FAIL	Wrong communication	SDO read & write failed. Reply to the servo communication, and execute this FB.
0x18804	DMC_TC_SCM_NOT_SUPPORTED	Wrong PDO configuration	Confirm the slave OD setting. Need to open TargetTorque, ActualTorque, ModeOfOperation, and ModeOfOperationDisplay.
0x18805	DMC_TC_SCM_AXIS_IN_WRONG_ STATE	Axis at wrong state	Use MC_Reset to eliminate the error.
0x18806	DMC_TC_SCM_INTERRUPTED	Function block execution error	Re-run the function block.
0x18807	DMC_TC_AXIS_NOT_READY_FOR_ MOTION	Axis state error	Power on servo and re-run the function block.
0x18808	DMC_TC_REGULATOR_OR_START_NOT_SET	The axis state cannot execute motion control instruction.	After starting servo, execute MC_Reset, and re-execute motion function block.
0x18809	DMC_TC_INVALID_PDO_MAPPING	Slave does not configure the related OD on PDO.	Confirm PDO configuration

Error code	Description	Contents	Corrective Action
0x18811	DMC_VC_SCM_NOT_SUPPORTED	Slave does not configure the related OD on PDO.	Confirm the slave OD setting. Need to open TargetVelocity, ActualVelocity, ModeOfOperation, and ModeOfOperationDisplay.
0x18812	DMC_VC_SCM_AXIS_IN_WRONG_STATE	Axis at wrong state	Use MC_Reset to eliminate the error.
0x18813	DMC_VC_SCM_INTERRUPTED	Wrong function block execution	Re-run the function block.
0x18814	DMC_VC_INVALID_ACCDEC_VALUES	Wrong value	Confirm pin input parameter value. Re-execute the FB after modification.
0x18815	DMC_VC_DIRECTION_NOT_APPLICABLE	Wrong value	Confirm pin input parameter value. Re-execute the FB after modification.
0x18816	DMC_VC_AXIS_NOT_READY_FOR_MOTION	Wrong axis state	Power on servo, and re-run the function block.
0x18817	DMC_VC_AXIS_ERROR_DURING_MOTION	Axis error	Confirm servo error information. Refer to Servo manual for error elimination, and execute MC_Reset.
0x18818	DMC_VC_REGULATOR_OR_START_NOT_SET	Axis error	Power on servo, execute MC_Rest, and re-execute motion function block.
0x18819	DMC_VC_WRONG_CONTROLLER_MODE	Axis is in the wrong controller mode.	Function block does not support execution in the current mode. To execute this function block, execute SMC_SetControllerMode first to switch the axis to the appropriate mode.
0x1881A	DMC_VC_INVALID_PDO_MAPPING	Slave does not configure the related OD to PDO.	Confirm PDO configuration.
0x1881B	DMC_CMGR_ZERO_VALUES	Wrong value	After modifying udiInputRotation, udiPulsePerRotation, udiOutputRotation, and udiUnitsPerRotation to non-zero values, re-run the function block.
0x1881C	DMC_CMGR_DRIVE_POWERED	Wrong axis state	After making the axis state goes into Disable, re-run the function block.
0x1881D	DMC_CMGR_INVALID_POSPERIOD	Wrong value	When setting iMovementType = 0, set fPositionPeriod to a value greater than 0 and less than half of dwBusBandWidth. Then, re-run the function block.
0x1881E	DMC_CMGR_POSPERIOD_NOT_INTEGRAL	Wrong value	After correcting fPositionPeriod parameter, re-run the function block.

Error code	Description	Contents	Corrective Action
0x1881F	DMC_CMGR_RAG_ERROR_DURING_STARTUP	Communication error	Confirm if the bus configuration is normal, and re-execute DMC_ChangeMechanismGearRation.
0x18820	DMC_CMGR_RAG_ERROR_AXIS_NOT_INITIALIZED	Axis initializing	EtherCAT Master cannot execute DMC_ChangeMechanismGearRation during Initialization.
0x1882E	DMC_GM_NO_ERROR_ TO_RESET	There's no error to be reset.	Re-execute DMC_GroupReset when an error occurs in the axis group.
0x1882F	DMC_GM_DRIVE_ DOESNT_ANSWER	One or more axes in the group does not execute the reset action.	After the communication status of the axis is back to normal, re-execute the FB. (DFB_ResetECATMaster/DFB_ResetECATSlave)
0x18830	DMC_GM_ERROR_NOT_RESETTABLE	Error is not resettable.	Remove the error in axis group (Modify parameter settings/ check on a normal axis path) before download the program once again.
0x18831	DMC_GM_DRIVE_ DOESNT_ANSWER_IN_ TIME	Communication timeout	After the communication status of the axis is back to normal (DFB_ResetECATMaster/DFB_ResetECATSlave), re-execute the FB.
0x18832	DMC_GM_CANNOT_ RESET_ COMMUNICATION_ ERROR	Communication error cannot be reset.	After the communication status of the axis is back to normal (DFB_ResetECATMaster/DFB_ResetECATSlave), re-execute the FB.
0x18833	DMC_GM_AXIS_GROUP_RESET_FAILED	Fail to reset the axis group.	Remove the error in axis group (Modify parameter settings/ check on a normal axis path) before download the program once again.
0x18839	DMC_GM_LINEAR_AXIS_MAPPING_ERROR	Command a non-zero displacement to an axis, which does not exist.	Execute MC_GroupReset to make the axis group back to GroupStandby state. Then check the parameter setting and the input position of axis group so as to make sure the existing displacement has been mapped to an appointed axis.
0x1883B	DMC_GM_JOG_AXIS_MAPPING_ERROR	A displacement other than 0 is specified for an axis that does not exist in the jog motion command.	Execute MC_GroupReset to return the axis group to GroupStandby state. And check the parameter setting of the axis group and the position of the input axis group motion command, and confirm that each axis in the axis group with the displacement amount has the correct designated single axis.
0x1883F	DMC_GM_CIRCULAR_ AXIS_MAPPING_ERROR	Command a non-zero displacement to an axis, which does not exist, in a circular movement.	Execute MC_GroupReset to make the axis group back to GroupStandby state. Then check the parameter setting and the input position of axis group so as to make sure the existing displacement has been mapped to an appointed axis.

Error code	Description	Contents	Corrective Action
0x18840	DMC_GM_HELIX_AXIS_ MAPPING_ERROR	Command a non-zero displacement to an axis, which does not exist, in a helical movement.	Execute MC_GroupReset to make the axis group back to GroupStandby state. Then check the parameter setting and the input position of axis group so as to make sure the existing displacement has been mapped to an appointed axis.
0x18841	DMC_GM_CIRCLE_ DISTANCE_LARGER_ THAN_DIAMETER	Under the DMC_CIRC_MODE. radius mode, the distance between the start and end point is larger than the diameter.	1. Execute MC_GroupReset to make the group state back to GroupStandby. 2. While using DMC_CIRC_MODE.radius, the input value of radius must be larger than half of the distance between the start and end point. 3. Re-run the function block.
0x18842	DMC_GM_CIRCLE_ START_AND_ENDPOINT_EQUAL	Under DMC_CIRC_MODE. radius / DMC_CIRC_ MODE.border mode, the start point and the end point are at the same position.	1. Execute MC_GroupReset to make the group state back to GroupStandby. 2. While using DMC_CIRC_MODE.radius / DMC_CIRC_MODE.border the input value of radius must be larger than half of the distance between the start and end point. 3. Re-run the function block.
0x18843	DMC_GM_CIRCLE_ COLLINEAR_POINTS	Under DMC_CIRC_MODE. border mode, three points are defined to lie on a same line.	1. Execute MC_GroupReset to make the group state back to GroupStandby. 2. While using DMC_CIRC_MODE.border, start point, end point and assist point should not be set on the same line. 3. Re-run the function block.
0x18844	DMC_GM_CIRCLE_ CENTER_NOT_ON_ BISECTOR	Under DMC_CIRC_MODE. center mode, the center of a circle is not on the bisector line.	1. Execute MC_GroupReset to make the group state back to GroupStandby. Make sure that the center must locates on the bisector line between the start and end point. 2. Re-run the function block.
0x18845	DMC_GM_CIRCLE_ RADIUS_ZERO	Under DMC_CIRC_MODE. radius mode, the radius is zero.	1. Make sure the radius is not 0 while using DMC_CIRC_MODE.radius mode. 2. Re-run the function block.
0x1884B	DMC_GM_CONTINUE_ WRONG_POSITION	The current position is not the start position recorded in continue data.	1. Move the axis group to the position recorded in Continue Data. (DMC_AXIS_GROUP_REF.ContinuePos) 2. Re-run the function block.
0x1884C	DMC_GM_CONTINUE_ DATA_NOT_WRITTEN	ContinueData is not written.	After confirming there's Continue Data in the axis group

Error code	Description	Contents	Corrective Action
			(DMC_AXIS_GROUP_REF.bContinueDataWriten), then execute DMC_GroupContinue.
0x18852	DMC_GM_NO_AXIS_IN_AXIS_GROUP	There're no axes in the axis group.	At least one axis must be specified in the parameter setting of axis group before re-run the function block.
0x18853	DMC_GM_SINGLE_AXIS_ERROR	Axis error occurs in the axis group.	1. After troubleshoot the error, execute MC_GroupReset to make the group state back to GroupStandby, while each axis leaves errorstop state. 2. Re-run the function block.
0x18854	DMC_GM_AXIS_NOT_READY_FOR_MOTION	One or more axes in the group are not ready for motion.	1. Execute MC_GroupReset to make the group state back to GroupStandby, while each axis leaves errorstop state. 2. Make sure that each axis has been successfully powered on and entered standstill state. 3. Re-run the function block.
0x18855	DMC_GM_AXIS_LIMIT_VIOLATED	One or more limits for an axis are violated.	1. Execute MC_GroupReset to make the group state back to GroupStandby. 2. Make sure that the position, velocity, acceleration and jerk of each axis do not exceed the limits. 3. Re-run the function block.
0x18856	DMC_GM_AXIS_GROUP_WRONG_STATE	Axis group is in wrong state.	Make sure the axis group is under the proper state and ready to be run before Run the function block.
0x18857	DMC_GM_AXIS_GROUP_AXIS_IN_DIFFERENT_TASK	Some axes in the group and the axis group itself are not in the same task.	Correct the settings of the axis and the group so as to make both bus cycle tasks are appointed to the same task...
0x18858	DMC_GM_INVALID_VEL_ACC_DEC_JERK	Invalid values of velocity, acceleration, deceleration and jerk.	1. Adjust the values to be reasonable and non-zero. 2. Re-run the function block.
0x18859	DMC_GM_INVALID_BUFFER_MODE	Invalid buffer mode.	1. Change to a supported buffer mode. 2. Re-run the function block.
0x1885A	DMC_GM_CMD_ABORTED_DUE_TO_ERROR	Command is aborted due to an error.	1. Troubleshoot the error. 2. Execute MC_GroupReset to make the group state back to GroupStandby. 3. Re-run the function block.
0x1885B	DMC_GM_TRANSITIONING_FROM_SINGLE_AXIS_MOVEMENT_NOT_SUPPORTED	Transitioning from the single-axis movement is not supported.	1. Execute MC_GroupReset to make the group state back to GroupStandby. 2. Make sure each axis is back to standstill. 3. Re-run the function block.

Error code	Description	Contents	Corrective Action
0x1885C	DMC_GM_AXIS_GROUP_VELOCITY_EXCEED_LIMIT	The velocity of axis group exceeds the limit set in the parameter setting.	1. Execute MC_GroupReset to make the group state back to GroupStandby. 2. Make sure the group velocity does not exceed the limit set in the parameter setting. 3. Re-run the function block.
0x1885D	DMC_GM_AXIS_GROUP_ACCELERATION_EXCEED_LIMIT	The acceleration of axis group exceeds the limit set in the parameter setting.	1. Execute MC_GroupReset to make the group state back to GroupStandby. 2. Make sure the group acceleration does not exceed the limit set in the parameter setting. 3. Re-run the function block.
0x1885E	DMC_GM_AXIS_GROUP_DECELERATION_EXCEED_LIMIT	The deceleration of axis group exceeds the limit set in the parameter setting.	1. Execute MC_GroupReset to make the group state back to GroupStandby. 2. Make sure the group deceleration does not exceed the limit set in the parameter setting. 3. Re-run the function block.
0x1885F	DMC_GM_AXIS_GROUP_JERK_EXCEED_LIMIT	The jerk of axis group exceeds the limit set in the parameter setting.	1. Execute MC_GroupReset to make the group state back to GroupStandby. 2. Make sure the group jerk does not exceed the limit set in the parameter setting. 3. Re-run the function block.
0x18860	DMC_GM_AXIS_GROUP_PLANNING_ERROR	Axis group planning error.	1. Execute MC_GroupReset to make the group state back to GroupStandby. 2. Make sure the parameters set for the motion instruction are reasonable for planning paths. 3. Re-run the function block.
0x18861	DMC_GM_AXIS_GROUP_MOVE_ERROR	Axis group move error.	1. Execute MC_GroupReset to make the group state back to GroupStandby. 2. Make sure the parameters set for the motion instruction are reasonable for planning paths. 3. Re-run the function block.
0x18862	DMC_GM_CMD_BUF_FULL	Command buffer is full.	1. Make sure there's still some space in the command buffer. 2. Re-run the function block.

Error code	Description	Contents	Corrective Action
0x18881	DMC_GM_AXIS_GROUP_INIT_FAILED	Axis group initialization failed.	1. use the axis group in the device tree as the input to the instruction. 2. Re-run the function block.
0x18882	DMC_GM_INVALID_AXIS_IN_AXIS_GROUP	Invalid axes in axis group.	1. Make sure all the axes specified in the parameter setting exist in the device tree. 2. Download the program again. 3. Re-run the function block.
0x18883	DMC_GM_DUPLICATE_AXIS_IN_AXIS_GROUP	Duplicated axes in axis group.	1. Make sure there's no duplicated axis specified in the parameter setting. 2. Download the program again. 3. Re-run the function block.
0x18884	DMC_GM_AXIS_ALREADY_IN_OTHER_ENABLED_AXIS_GROUP	Some axes have been already existed in another enabled axis group.	1. Make sure the specified axis does not exist in other enabled axis group or disable the axis group which has the axis in it. 2. Re-run the function block.
0x18885	DMC_GM_AXIS_GROUP_INVALID_TASK_CONFIGURATION	Task is not configured correctly.	1. Make sure that the Setting Values of bus cycle task meet the requirement. (Type: Cyclic, Interval: > 1ms) 2. Download the program again. 3. Re-run the function block.
0x18886	DMC_GM_AXIS_GROUP_COUNT_REACH_LIMIT	The axis group count has reached the limit.	1. To activate more groups, make sure the number of activated axis group is less than the max.value. 2. Re-run the function block.
0x18890	DMC_GM_AXIS_GROUP_INVALID_PARAMETER	Invalid axis group parameter	After confirming that Parameter input pin has correct readable and writable parameters, re-run the function block.
0x18891	DMC_GM_AXIS_GROUP_CANT_WRITE_PARAMETER_DURING_GROUP_ENABLED	Cannot modify parameter during the axis group is enabled.	After using DMC_GroupDisable to disable this axis group, re-run the function block.
0x18892	DMC_GM_AXIS_GROUP_INVALID_PARAMETER_SETTING	Invalid axis group parameter	After confirming that lrValue input pin has correct parameter Setting Value, re-run the function block.
0x1889A	DMC_GM_INVALID_IDENT_IN_GROUP	The value of the input pin "IdentInGroup" is not within the legal range.	Correct the value of the input pin "IdentInGroup". (Range starts at 1) Re-run the function block.
0x1889B	DMC_GM_AXIS_NOT_PART_OF_AXIS_GROUP	The specified axis does not belong to this axis group and cannot be removed.	Make sure that the specified single axis is included in the axis group. Re-run the function block.
0x1889C	DMC_GM_AXIS_GROUP_CANNOT_ADD_SAME_AXIS	It is forbidden to add the same axis to the axis group multiple times.	Confirm that the specified single axis is not currently included in the axis group. Re-run the function block.
0x188B5	DMC_CKPW_WRITE AMOUNT_OUTOFRANGE	WriteAmount input error	Check and correct the input value of WriteAmount before Run the function block.

Error code	Description	Contents	Corrective Action
0x188B6	DMC_CKPW_INVALID_ MASTERPOS	Invalid master position.	Re-execute the FB after correcting the input of master position.
0x188B7	DMC_CKPW_INVALID_ ACC	Invalid acceleration.	Re-execute the FB after correcting the acceleration input value of master position.
0x188B8	DMC_CKPW_INVALID_ ACC_SETTING	Invalid acceleration setting.	Re-execute the FB after determining the velocity, acceleration and curve type.
0x188B9	DMC_CKPW_INVALID_ CURVE_TYPE_SETTING	Invalid curve type setting.	The input curve type is not supported. Re-execute the FB after correcting the curve type.
0x188BA	DMC_CKPW_SPLINE_ HAS_NO_BOUNDARY	Spine has no boundary.	Make sure there's boundary condition (Nature or Clamp) set for the previous and the latter part of the selected curve "Spline", which the condition should be the same at the start and end of the boundary. Then re-execute the FB.
0x188BB	DMC_CKPW_CAM_IS_ WRITING_BY_OTHER_ FUNCTION	Failure to write CAM.	Check if the cam table you're currently using is being written by other FBs, then wait for the writing completed before you re-execute the FB.
0x188C5	DMC_HP_INVALID_ HOME_SPEED	Invalid home speed value.	set "Search for switch" and "Search for Z phase pulse" with non-zero values for the home speed setting on Pulse Axis configuration page.
0x188C6	DMC_HP_INVALID_ HOME_ACC_DEC	Invalid home acceleration or deceleration value.	set the homing acceleration and deceleration with non-zero values on Pulse Axis configuration page.
0x188C7	DMC_HP_INVALID_ HOME_POSITION	Invalid Setting Value of home position.	Set "lrPosiotion" to be in the rotary range of pulse axis. [0 ~ PulseAxis.Modulo Value]
0x188C8	DMC_HP_AXIS_NOT_ PULSEAXIS	The input variable type is not set to be PulseAxis_REF.	After select "Pulse Axis" in IO Configuration, enter the IEC Object variable to the input "Axis" of FB DMC_Home_P.
0x188C9	DMC_HP_HOMING_ METHOD_RESERVED	Homing method is not supported by current version.	Check if the homing method is supported by the version you're currently using. refer to the specification document for mode modification.
0x188CA	DMC_HP_HOMING_ MOVEMENT_HW_LIMIT	Positive or negative limit signal is activated and axis cannot perform homing in this circumstances.	Check if the hardware limit signal you're using is supported by the current homing mode. refer to the specification document for changing the mode and hardware limit signal configuration.
0x188CB	DMC_HP_HOMING_AXIS_STATE_N OT_STAND STILL	Axis state is not Standstill.	Confirm that DMC_Home_P is run when the axis state is Standstill.
0x188D5	DMC_ISP_AXIS_NOT_READY_FOR _MOTION	Wrong axis state	Power on servo and re-run the function block.
0x188D6	DMC_ISP_WRONG_CONTROLLER_ MODE	Wrong axis state	Switch the control mode to SMC_position, and re-run the function block.

Error code	Description	Contents	Corrective Action
0x1896C	DMC_STL_WP_PARAM_ INVALID	Invalid parameter.	The input parameter is too large. Re-execute the FB after correcting the input parameter.
0x1896D	DMC_STL_WP_SENDING_ERROR	No corresponding OD or the OD is not allowed to be written.	No such error should occur while matching ASDA-A2-E to use. check if the servo you're currently using meets Cia402, or the function block cannot be run.
0x1896E	DMC_STL_WP_DRIVE_ PARAMETER_NOT_ MAPPED	The input parameter number does not exist.	No such error should occur while matching ASDA-A2-E to use. check if the servo you're currently using meets Cia402, or the function block cannot be run.
0x1896F	DMC_STL_WP_PARAM_CONVERSI ON_ERROR	Parameter conversion error.	No such error should occur while matching ASDA-A2-E to use. check if the servo you're currently using meets Cia402, or the function block cannot be run.
0x1897A	DMC_SSWL_LIMIT_ SETTING_OPPOSITE	Negative limit input error.	Negative software limit is greater than positive software limit. correct the input limit before you re-execute the FB.
0x1897B	DMC_SSWL_NEGPOS_ LIMT_EQUAL	Negative limit input error.	Negative software limit is equal to positive software limit. correct the input limit before you re-execute the FB.
0x1898A	DMC_PL_INVALID_ POSITIONLAG	Invalid MaxPositionLag input.	The input value of fMaxPositionLag is negative, correct the value before re-execute the FB.
0x1898B	DMC_PL_INVALID_ LAGCYCIES	Invalid SetActTimeLagCycles input.	The input value of SetActTimeLagCycles is negative, correct the value before re-execute the FB.
0x18996	DMC_MVSBP_INVALID_DIRECTION	Invalid direction.	Only positive and negative direction are allowed, correct the direction of movement before re-execute the FB.
0x18997	DMC_MVSBP_INVALID_PHASE	Invalid phase input.	RoundPhase/ StopPhase input error. correct the input parameters before re-execute the FB.
0x18998	DMC_MVSBP_AXIS_NOT_READY_F OR_MOTION	Slave axis is not ready for motion.	The slave is not under control. check if the target axis is powered on or in error, then enable the axis or execute MC_Reset depending on the situation.
0x18999	DMC_MVSBP_AXIS_ ERROR_DURING_ MOTION	Errors occur during motion.	check the error information. Refer to the corresponding servo's user manual to troubleshoot the error and execute MC_Reset.
0x1899A	DMC_MVSBP_ REGULATOR_OR_ START_NOT_SET	The motion control instruction cannot be run under the current axis state.	After activating the servo, execute MC_Reset before re-execute the FB.
0x1899B	DMC_MVSBP_INVALID_ACCDEC_V ALUES	Invalid velocity, acceleration, deceleration, and jerk	After correcting the parameter, re-run the function block.

Error code	Description	Contents	Corrective Action
0x189A5	DMC_AO_INVALID_REFERENCE_TYPE	Invalid reference type	Wrong reference type. Correct the reference type and re-run the function block.
0x189C6	DMC_VC_WRONG_AXIS_TYPE	Specify wrong axis	Confirm that the function block specifies the EtherCAT axis.
0x189D4	DMC_MM_INVALID_ACCDEC_VALUES	Invalid velocity or acceleration value	Enter the velocity or acceleration value and restart the function block
0x189D5	DMC_MM_AXIS_NOT_READY_FOR_MOTION	Current axis status cannot run the motion control command	The axis is in an uncontrollable state. Confirm whether the target axis is powered on or in an error state. Enable the axis or MC_Reset the axis according to the situation.
0x189D6	DMC_MM_AXIS_ERROR_DURING_MOTION	An error occurs during motion	Confirm the servo error message. Refer to the servo manual to troubleshoot the error, and run MC_Reset.
0x189D7	DMC_MM_REGULATOR_OR_START_NOT_SET	Current axis status cannot run the motion control command	Start the servo, run MC_Reset, and then run motion function block again.
0x189D8	DMC_MM_INVALID_DIRECTION	Direction error	Only forward and reverse motion are allowed. Modify the direction and restart the function block.
0x189D9	DMC_MM_INVALID_MODULO	lrModulo input error	Check if lrModulo is set to the correct range.
0x189DA	DMC_MM_INVALID_POS_VALUES	lrPosition input error	Check if lrPosition is set to the correct range.
0x189E0	DMC_WT_INVALID_PARAMENT	Input value error	Check the value.
0x189EB	DMC_GCSD_MASTER_OUT_OF_RANG	The target entered exceeds the cam master axis range	Check if the input value is out of range.

A.3.2 For Positioning Axis

When an error occurs, you can troubleshoot errors through error codes and the corresponding indicators. Refer to *AX-3 Operational Manual* for more details of troubleshooting.

The following table lists the error codes and the contents of the errors:

Error code	Description	Contents	Corrective Action
0x00000	SML_NO_ERROR	No error messages	-
0x00001	SML_DI_GENERAL_COMMUNICATION_ERROR	Communication error	Confirm if the Slave network cable is properly plugged. Run DFB_ResetECATMaster to reset EtherCAT Master, and then re-run MC_ReinitDrive_DML.
0x00002	SML_DI_AXIS_ERROR	Axis error	Confirm Slave error information and eliminate the error, and then re-execute MC_Reset_DML.
0x00015	SML_WRONG_OPMODE	Wrong control mode	Function block does not support execution in the current mode. To execute this function block, execute SMC_SetControllerMode first to switch the axis to the appropriate mode.
0x00022	SML_AXIS_NOT_READY_FOR_MOTION	The Slave state cannot execute the motion control instruction.	Axis is at the state that cannot be controlled. Confirm whether it is at the Power-on or error state. Start the axis or run MC_Reset_DML depending on the situation.
0x00023	SML_MA_MR_MODULO_ACT_POS_NOT_MAPPED	PDO lacks the essential parameter.	Configure Actual Position (16#6064) to PDO.
0x00024	SML_MV_INVALID_VELACCDEC_VALUES	Invalid velocity or acceleration/deceleration Setting Value	Use MC_Reset_DML to eliminate error.
0x00050	SMC_RAG_ERROR_DURING_STARTUP	Error occurs during axis re-startup	Confirm if the bus configuration is normal, and re-run MC_ReinitDrive_DML.
0x0005A	SML_CGR_ZERO_VALUES	Cannot enter 0 for dwRatioTechUnitsDenom and iRatioTechUnitsNum	After modifying dwRatioTechUnitsDenom and iRatioTechUnitsNum to non-zero values, re-run the function block.
0x0005B	SML_CGR_AXIS_POWERED	Cannot change gear ratio parameter at the wrong state.	After making the axis state goes into Disable, re-run the function block.
0x0005D	SML_CGR_MODULOPERIOD_NOT_INTEGRAL	Module period is not an integer.	After modifying the fModuloPeriodU parameter, re-run the function block.
0x0005E	SML_CGR_MOVEMENTTYPE_INVALID	Wrong axis type (Must be either a linear axis or rotary axis).	After modifying the iMovementType parameter, re-run the function block.
0x0005F	SML_CGR_MODULOPERIOD_NON_POSITIVE	Module period cannot be a negative.	After modifying the fPositionPeriod parameter, re-run the function block.
0x00060	SML_CGR_MODULOPERIOD_TOO_SMALL	Module period is too small.	After modifying the fPositionPeriod parameter, re-run the function block.
0x00061	SML_CGR_MODULOPERIOD_TOO_LARGE	Module period is too large.	After modifying the fPositionPeriod parameter, re-run the function block.
0x00078	SML_R_NO_ERROR_TO_RESET	No axis error after using MC_Reset_DML	Confirm whether the axis is correct, and then re-run the function block.
0x0007A	SML_R_ERROR_NOT_RESETTABLE	Error, non-resettable.	Confirm whether the Slave error has been eliminated. After error disappeared, restart MC_Reset_DML.
0x00083	SML_RP_REQUESTING_ERROR	Slave has no corresponding OD, or reading the OD is not allowed.	The OD you visit does not exist or is not allowed to be accessed. Confirm the input OD is correct and can be read.
0x00084	SML_RP_RCV_PARAM_CONVERSION_ERROR	Conversion error of the axis parameter to servo OD. Unknown SoftMotionLight parameter.	The parameter you visit does not exist.
0x0008D	SML_WP_SENDING_ERROR	Slave has no corresponding OD, or writing the OD is not allowed.	The OD you visit does not exist or is not allowed to be written. Confirm the input OD is correct and can be written.

Error code	Description	Contents	Corrective Action
0x0008E	SML_WP_TMT_PARAM_CONVERSION_ERROR	Conversion error of the axis parameter to servo OD. Unknown SoftMotionLight parameter.	The written parameter does not exist.
0x000AA	SML_H_AXIS_WASNT_STANDSTILL	Axis is not at the Standstill state.	Make axis enter the Standstill state, and re-run the function block.
0x000B7	SML_MS_AXIS_IN_ERRORSTOP	Driver is at the Errorstop state. Cannot execute MC_Stop_DML.	Make axis leave the ErrorStop state, and re-run the function block.
0x186A0	DML_MA_SDO_RW_FAIL	SDO read & write failed.	Reply to the slave communication, confirm the pin input parameter value meets the definition range of slave Object, and then re-run the function block.
0x186A1	DML_MA_AXIS_NOT_READY_FOR_MOTION	The axis state cannot execute motion control instructions.	After confirming the axis at the state that can execute motion instructions, re-run the function block.
0x186A2	DML_MA_INVALID_VALUES	The input parameter is invalid Setting Value.	Confirm the pin input parameter value. After the confirmation, re-run the function block.
0x186A4	DML_MA_AXIS_NOT_SUPPORT_PP_MODE	Slave does not support the PP mode.	The current selected slave does not support Profile Position Mode. use another model.
0x186AA	DML_MR_SDO_RW_FAIL	SDO read & write failed.	Reply to the slave communication, confirm the pin input parameter value meets the definition range of slave Object, and then re-run the function block.
0x186AB	DML_MR_AXIS_NOT_READY_FOR_MOTION	The motion FB cannot be run under the current axis state.	After confirming the axis at the state that can execute motion instructions, re-run the function block.
0x186AC	DML_MR_INVALID_VALUES	The input parameter is invalid Setting Value.	Confirm pin input parameter value. Re-run the FB after modification.
0x186AE	DML_MR_AXIS_NOT_SUPPORT_PP_MODE	Slave does not support the PP mode.	The current selected slave does not support Profile Position Mode. use another model.
0x186B4	DML_MV_SDO_RW_FAIL	SDO read & write failed.	Reply to the slave communication, confirm the pin input parameter value meets the definition range of slave Object, and then re-run the function block.
0x186B5	DML_MV_AXIS_NOT_READY_FOR_MOTION	The axis state cannot execute motion control instructions.	After confirming the axis at the state that can execute motion instructions, re-run the function block.
0x186B6	DML_MV_INVALID_VALUES	The input parameter is invalid Setting Value.	Confirm pin input parameter value. Re-run the FB after modification.
0x186B8	DML_MV_AXIS_NOT_SUPPORT_PV_MODE	Slave does not support the PV mode.	The current selected slave does not support Profile Velocity Mode. use another model.
0x186BE	DML_TC_SDO_RW_FAIL	SDO read & write failed.	Reply to the slave communication, confirm the pin input parameter value meets the definition range of slave Object, and then re-run the function block.
0x186BF	DML_TC_AXIS_NOT_READY_FOR_MOTION	The axis state cannot execute motion control instructions.	After confirming the axis at the state that can run motion instructions, re-run the function block.
0x186C0	DML_TC_INVALID_VALUES	The input parameter is invalid Setting Value.	Confirm pin input parameter value. Re-execute the FB after modification.
0x186C2	DML_TC_AXIS_NOT_SUPPORT_PT_MODE	Slave does not support the PT mode.	The current selected slave does not support Profile Torque Mode. Use another model.
0x186C8	DML_VC_SDO_RW_FAIL	SDO read & write failed.	Reply to the slave communication, confirm the pin input parameter value meets the definition range of slave Object, and then re-run the function block.

Error code	Description	Contents	Corrective Action
0x186C9	DML_VC_AXIS_NOT_READY_FOR_MOTION	The axis state cannot execute motion control instructions.	After confirming the axis at the state that can execute motion instructions, re-run the function block.
0x186CA	DML_VC_INVALID_VALUES	The input parameter is invalid Setting Value.	Confirm pin input parameter value. Re-execute the FB after modification.
0x186CC	DML_VC_AXIS_NOT_SUPPORT_VL_MODE	Slave does not support the VL mode.	The current selected slave does not support Velocity Mode. use another model.
0x186D2	DML_HA_SDO_RW_FAIL	SDO read & write failed.	Reply to the slave communication, confirm the pin input parameter value meets the definition range of slave Object, and then re-run the function block.
0x186D3	DML_HA_AXIS_NOT_READY_FOR_MOTION	The axis state cannot execute motion control instructions.	After confirming the axis at the state that can run motion instructions, re-run the function block.
0x186D4	DML_HA_INVALID_VALUES	The input parameter is invalid Setting Value.	Confirm pin input parameter value. Re-execute the FB after modification.
0x186D6	DML_HA_AXIS_NOT_SUPPORT_PV_MODE	Slave does not support the PV mode.	The current selected slave does not support Profile Velocity Mode. use another model.
0x186DC	DML_MS_SDO_RW_FAIL	SDO read & write failed.	Reply to the slave communication, confirm the pin input parameter value meets the definition range of slave Object, and then re-run the function block.
0x186DD	DML_MS_AXIS_NOT_READY_FOR_MOTION	The axis state cannot execute motion control instructions.	After confirming the axis at the state that can run motion instructions, re-run the function block.
0x186EA	DML_H_AXIS_NOT_SUPPORT_HM_MODE	Slave does not support the HM mode.	The current selected slave does not support Homing Mode. Use another model.
0x186F0	DML_R_SDO_RW_FAIL	SDO read & write failed.	Reply to the slave communication, confirm the pin input parameter value meets the definition range of slave Object, and then re-run the function block.
0x18A88	DML_R2R_CIG_TENSION_CTRL_UNSUPPORTED	Tension control is not supported.	Tension control is not supported.
0x18A89	DML_R2R_CIG_COMMUNICATION_ERROR	SDO communication error.	SDO communication error.
0x18A8A	DML_R2R_CIG_REUSED_ANALOG_INPUT_AVI	AVI analog input is reused.	AVI analog input is reused.
0x18A8B	DML_R2R_CIG_REUSED_ANALOG_INPUT_ACI	ACI analog input is reused.	ACI analog input is reused.
0x18A8C	DML_R2R_CIG_EXE_NOT_ON_STATE_POWER_OFF	R2R_Configuration is not run in power off state.	R2R_Configuration is not run in power off state.
0x18A8D	DML_R2R_CIG_GEAR_RATIO_IS_OVER_RAMGE	Gear ratio is out of range.	Gear ratio is out of range.
0x18A8E	DML_R2R_CIG_LINE_SPEED_MAX_IS_OVER_RAMGE	The maximum linear velocity is out of range.	The maximum linear velocity is out of range.
0x18A8F	DML_R2R_CIG_TENSION_MAX_IS_OVER_RANGE	The maximum tension is out of range.	The maximum tension is out of range.
0x18A90	DML_R2R_CIG_OUTPUT_LIMIT_IS_OVER_RANGE	Control output limit out of range.	Control output limit out of range.
0x18A91	DML_R2R_CIG_UNSUPPORTED_TENSION_TARGET_SOURCE	Tension target source not supported.	Tension target source not supported.
0x18A92	DML_R2R_CIG_UNSUPPORTED_TENSION_TARGET_SOURCE_AT_0_SPEED	Zero speed tension target source not supported.	Zero speed tension target source not supported.
0x18A93	DML_R2R_CIG_UNSUPPORTED_PID_TARGET_SOURCE	PID target source not supported.	PID target source not supported.
0x18A94	DML_R2R_CIG_UNSUPPORTED_PID_FEEDBACK_SOURCE	PID feedback source not supported.	PID feedback source not supported.

Error code	Description	Contents	Corrective Action
0x18A95	DML_R2R_CIG_UNSUPPORTED_PID_ADAPTABILITY_REFERENCE_SOURCE	Adaptive PID reference source not supported.	Adaptive PID reference source not supported.
0x18A96	DML_R2R_CIG_UNSUPPORTED_LINE_SPEED_SOURCE	Line speed source not supported.	Line speed source not supported.
0x18AA6	DML_R2R_RD_TENSION_CTRL_UNSUPPORTED	Tension control is not supported.	Tension control is not supported.
0x18AA7	DML_R2R_RD_COMMUNICATION_ERROR	SDO communication error.	SDO communication error.
0x18AA8	DML_R2R_RD_REUSED_ANALOG_INPUT_AVI	AVI analog input is reused.	AVI analog input is reused.
0x18AA9	DML_R2R_RD_REUSED_ANALOG_INPUT_ACI	ACI analog input is reused.	ACI analog input is reused.
0x18AAA	DML_R2R_RD_EXE_NOT_ON_STATE_POWER_OFF	R2R_RollDiameter is not run in power off state.	R2R_RollDiameter is not run in power off state.
0x18AAB	DML_R2R_RD_UNSUPPORTED_ROLL_DIAMETER_SOURCE	Roll diameter source not supported.	Roll diameter source not supported.
0x18AAC	DML_R2R_RD_ROLL_DIAMETER_MAX_IS_OVER_RANGE	The maximum roll diameter is out of range.	The maximum roll diameter is out of range.
0x18AAD	DML_R2R_RD_ROLL_DIAMETER_MIN_IS_OVER_RANGE	The minimum roll diameter is out of range.	The minimum roll diameter is out of range.
0x18AAE	DML_R2R_RD_PULSE_PER_REVOLUTION_IS_OVER_RANGE	The number of pulses per revolution is out of range.	The number of pulses per revolution is out of range.
0x18AAF	DML_R2R_RD_ROUND_PER_LAYER_IS_OVER_RANGE	The number of turns per layer is out of range.	The number of turns per layer is out of range.
0x18AB0	DML_R2R_RD_MATERIAL_THICKNESS_IS_OVER_RANGE	Coil thickness is out of range.	Coil thickness is out of range.
0x18AB1	DML_R2R_RD_ROLL_DIAMETER_FILTER_TIME_IS_OVER_RANGE	Roll diameter calculation filter time is out of range.	Roll diameter calculation filter time is out of range.
0x18AC4	DML_R2R_RU_TENSION_CTRL_UNSUPPORTED	Tension control is not supported.	Tension control is not supported.
0x18AC5	DML_R2R_RU_COMMUNICATION_ERROR	SDO communication error.	SDO communication error.
0x18AC6	DML_R2R_RU_RUN_BEFORE_CFIG	R2R_Run runs before R2R_Configuration completes.	R2R_Run runs before R2R_Configuration completes.
0x18AC7	DML_R2R_RU_EXE_NOT_ON_STATE_STANDSTILL	R2R_Run is not run in standstill state.	R2R_Run is not run in standstill state.
0x18AC8	DML_R2R_RU_CURRENT_LINE_SPEED_IS_OVER_RANGE	Out of range.	Out of range.
0x18AC9	DML_R2R_RU_TENSION_COMMAND_IS_OVER_RANGE	Out of range.	Out of range.
0x18ACA	DML_R2R_RU_TENSION_COMMAND_AT_0_SPEED_IS_OVER_RANGE	Out of range.	Out of range.
0x18ACB	DML_R2R_RU_PID_GAIN_P_1ST_IS_OVER_RANGE	Out of range.	Out of range.
0x18ACC	DML_R2R_RU_PID_TIME_I_1ST_IS_OVER_RANGE	Out of range.	Out of range.
0x18ACD	DML_R2R_RU_PID_GAIN_P_2ND_IS_OVER_RANGE	Out of range.	Out of range.
0x18ACE	DML_R2R_RU_PID_TIME_I_2ND_IS_OVER_RANGE	Out of range.	Out of range.
0x18ACF	DML_R2R_RU_NOT_IN_STATE_CONTINUOUS_MOTION	R2R_Run is forced out of continuous motion state.	R2R_Run is forced out of continuous motion state.

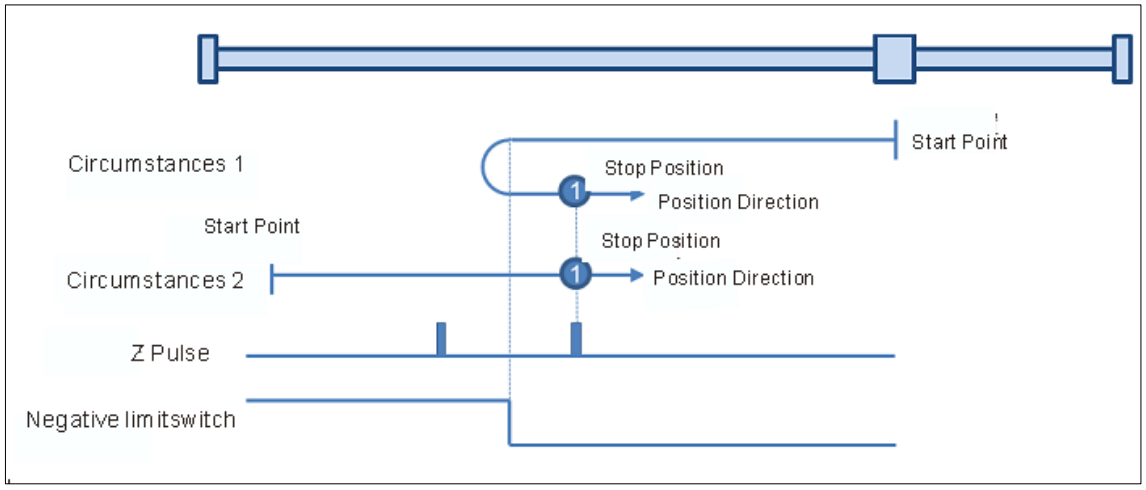
A.4 Explanation of DMC_Home_P

DFB_Home_P provides many homing modes from which user can choose the appropriate one in accordance with the field condition and technical requirement.

- **Mode 1: Homing which depends on the negative limit switch and Z pulse.**

Circumstance 1: MC_Home instruction is run when the negative limit switch is OFF and the axis moves in the negative direction at the first-phase speed. The motion direction changes and the axis moves at the second-phase speed when the axis encounters that the negative limit switch is ON. Where the first Z pulse is met is the home position when the negative limit switch is OFF.

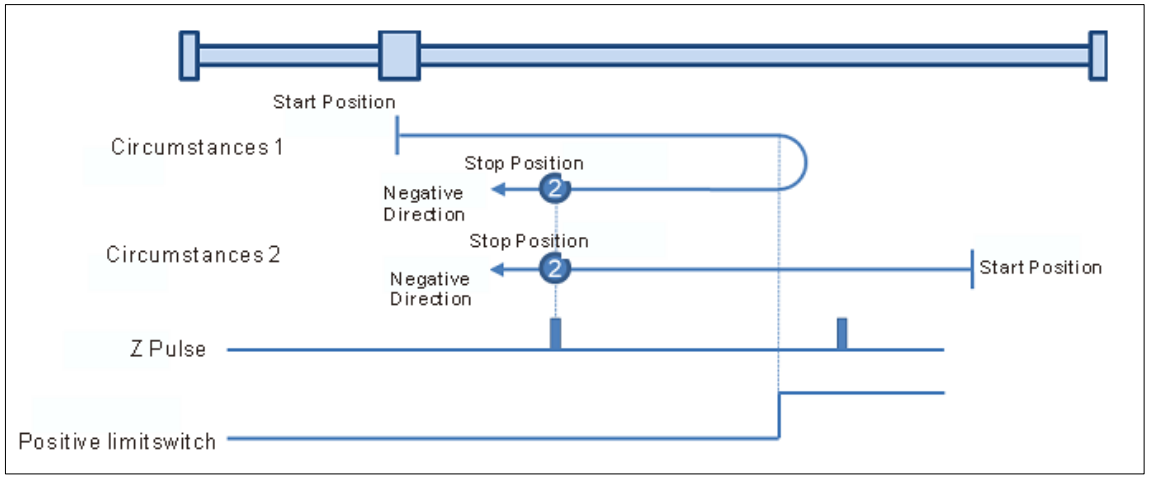
Circumstance 2: MC_Home instruction is run when the negative limit switch is ON and the axis moves in the positive direction at the second-phase speed. Where the first Z pulse is met is the home position when the negative limit switch is OFF.



- **Mode 2: Homing which depends on the positive limit switch and Z pulse**

Circumstance 1: MC_Home instruction is run when the positive limit switch is OFF and the axis moves in the positive direction at the first-phase speed. The motion direction changes and the axis moves at the second-phase speed when the axis encounters that the positive limit switch is ON. Where the first Z pulse is met is the home position while the positive limit switch is OFF.

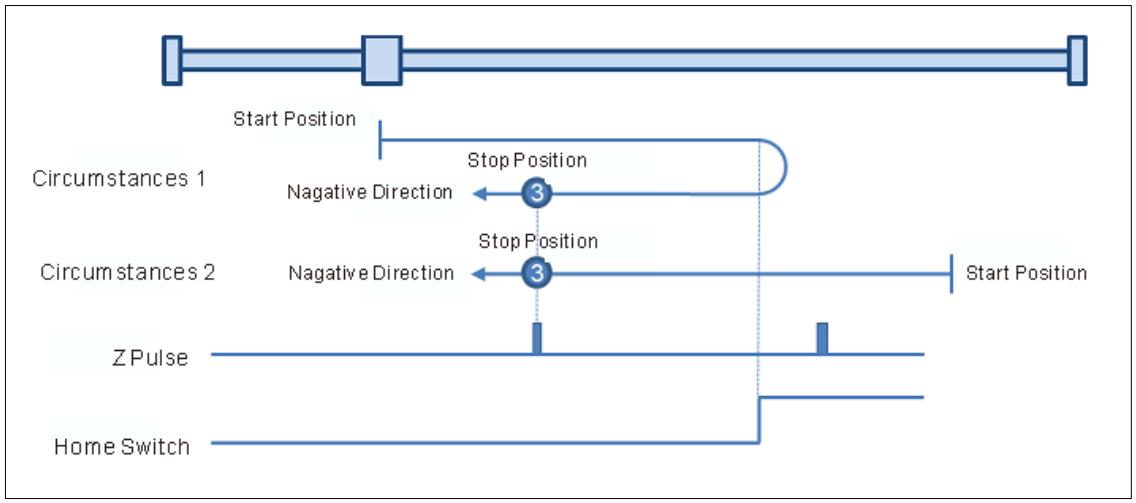
Circumstance 2: MC_Home instruction is run when the positive limit switch is ON and the axis moves in the negative direction at the second-phase speed. Where the first Z pulse is met is the home position while the positive limit switch is OFF.



- **Mode 3: Homing which depends on the home switch and Z pulse**

Circumstance 1: When the home switch is OFF, MC_Home instruction is run and the axis moves in the positive direction at the first-phase speed. When the axis encounters that the home switch is ON, the motion direction changes and the axis moves at the second-phase speed. Where the first Z pulse is met is the home position when the home switch is OFF.

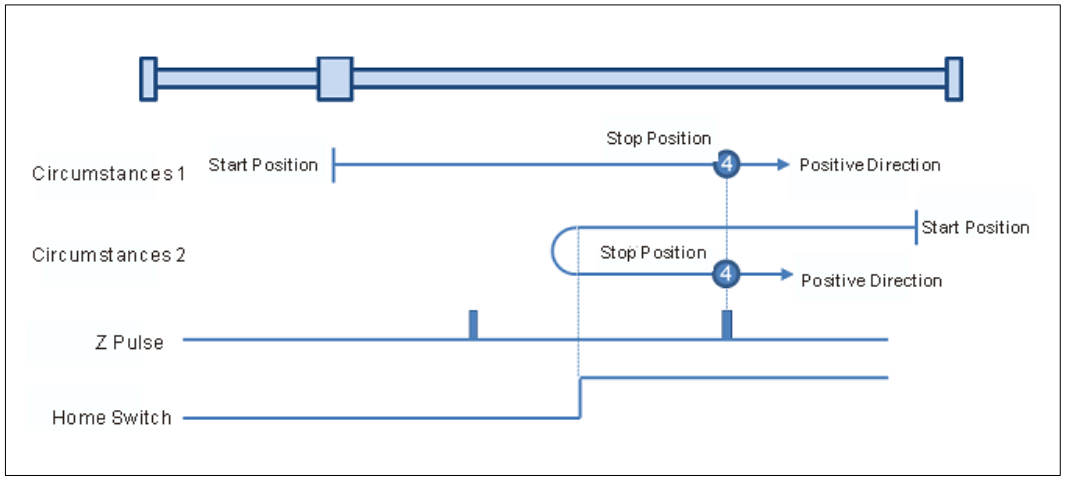
Circumstance 2: When the home switch is ON, MC_Home instruction is run and the axis directly moves in the negative direction at the second-phase speed. Where the first Z pulse is met is the home position while the home switch is OFF.



- **Mode 4: Homing which depends on the home switch and Z pulse**

Circumstance 1: When the home switch is OFF, MC_Home instruction is run and the axis moves in the positive direction at the first-phase speed. The axis moves at the second-phase speed when the axis encounters that the home switch is ON. Where the first Z pulse is met is the home position.

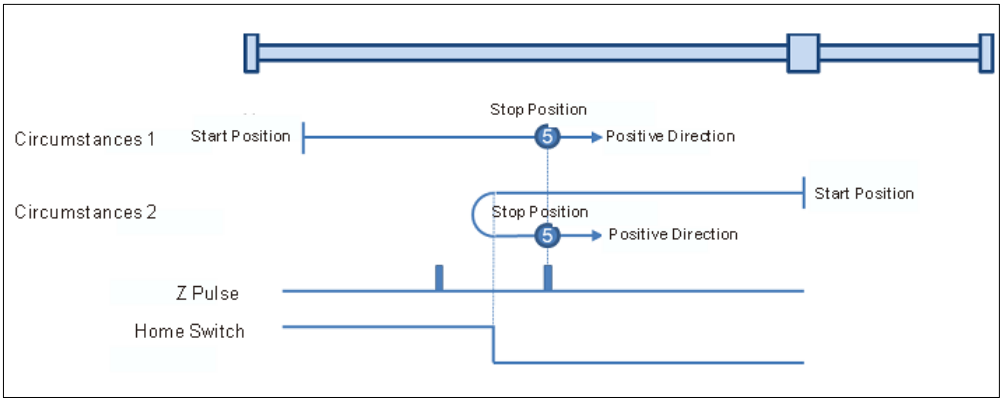
Circumstance 2: When the home switch is ON, MC_Home instruction is run and the axis moves in the negative direction at the second-phase speed. When the axis encounters that the home switch is OFF, the motion direction changes and the axis moves at the second-phase speed. Where the first Z pulse is met is the home position.



- **Mode 5 : Homing which depends on the home switch and Z pulse**

Circumstance 1: When the home switch is ON, MC_Home instruction is run and the axis moves in the positive direction at the second-phase speed. Where the first Z pulse is met is the home position while the home switch is OFF.

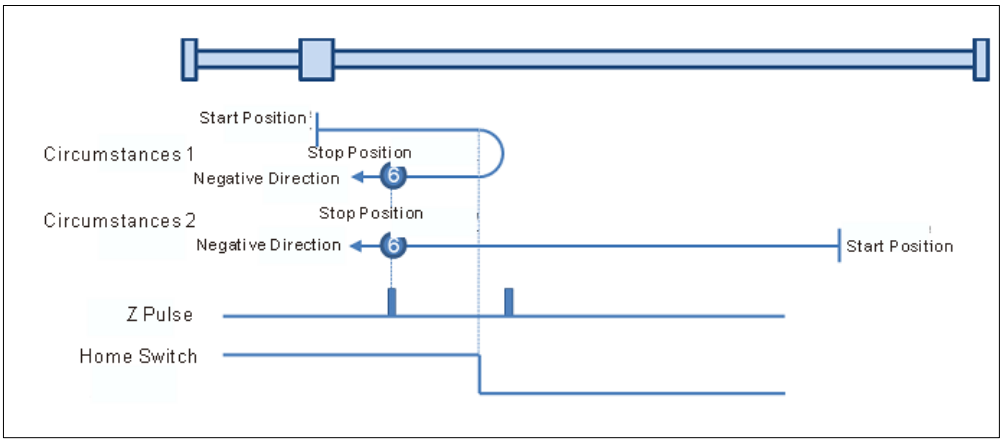
Circumstance 2: When the home switch is OFF, MC_Home instruction is run and the axis moves in the negative direction at the first-phase speed. When the home switch is ON, the motion direction changes and the axis moves at the second-phase speed. Where the first Z pulse is met is the home position when the home switch is OFF.



- **Mode 6: Homing which depends on the home switch and Z pulse**

Circumstance 1: When the home switch is ON, MC_Home instruction is run and the axis moves in the positive direction at the second-phase speed. When the home switch is OFF, the motion direction changes and the axis moves at the second-phase speed. Where the first Z pulse is met is the home position.

Circumstance 2: When the home switch is OFF, MC_Home instruction is run and the axis moves in the negative direction at the first-phase speed. While the home switch is ON, the axis moves at the second-phase speed and where the first Z pulse is met is the home position.

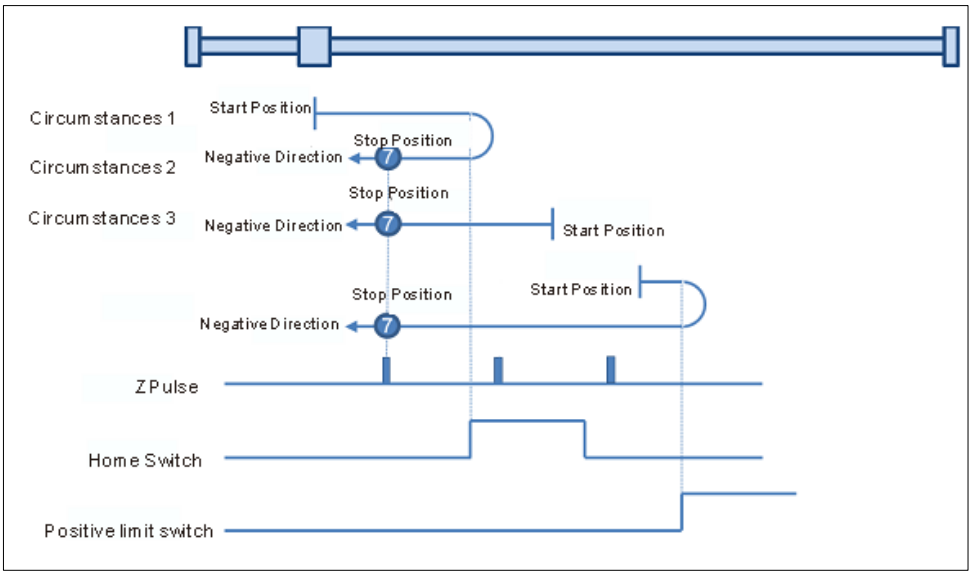


- **Mode 7: Homing which depending on the home switch, positive limit switch and Z pulse**

Circumstance 1: When the home switch is OFF, MC_Home instruction is run and the axis moves in the positive direction at the first-phase speed. The motion direction changes and the axis moves at the second-phase speed when the home switch is ON. Where the first Z pulse is met is the home position when the home switch is OFF.

Circumstance 2: When the home switch is ON, MC_Home instruction is run and the axis moves in the negative direction at the second-phase speed. Where the first Z pulse is met is the home position when the home switch is OFF.

Circumstance 3: When the home switch is OFF, MC_Home instruction is run and the axis moves in the positive direction at the first-phase speed. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF and the positive limit switch is ON. The axis starts to move at the second-phase speed when the home switch is ON. Where the first Z pulse is met is the home position when the home switch is OFF.

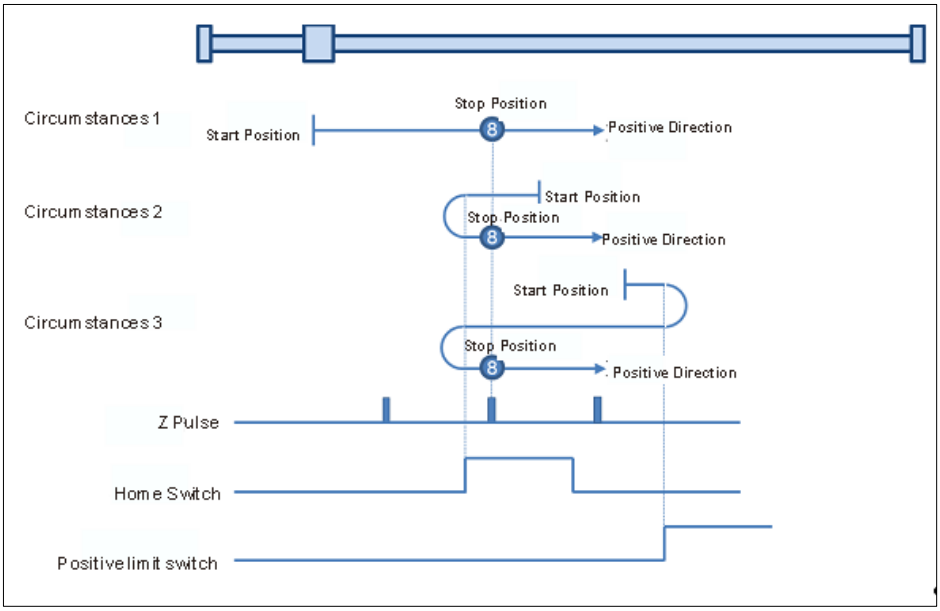


- **Mode 8: Homing depending on the home switch, positive limit switch and Z pulse.**

Circumstance 1: When the home switch is OFF, MC_Home instruction is run and the axis moves in the positive direction at the first-phase speed. The axis moves at the second-phase speed when the home switch is ON and where the first Z pulse is met is the home position.

Circumstance 2: MC_Home instruction is run and the axis moves in the negative direction at the second-phase speed when the home switch is ON. The motion direction changes and the axis moves at the second-phase speed when the home switch is OFF. And where the first Z pulse is met is the home position.

Circumstance 3: When the home switch is OFF, MC_Home instruction is run and the axis moves in the positive direction at the first-phase speed. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF and the positive limit switch is ON. The axis still moves at the first-phase speed when the home switch is ON. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF. The axis moves at the second-phase speed and where the first Z pulse is met is the home position when the home switch is ON.

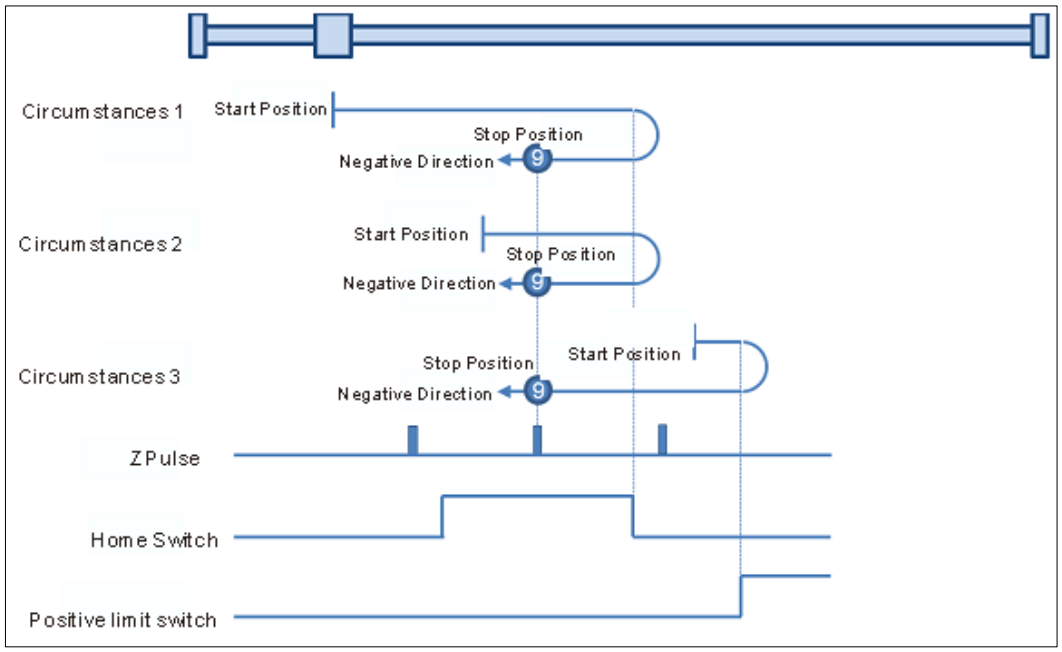


- **Mode 9: Homing depending on the home switch, positive limit switch and Z pulse**

Circumstance 1: MC_Home instruction is run and the axis moves in the positive direction at the first-phase speed when the home switch is OFF. The axis moves at the second-phase speed when the home switch is ON. The motion direction changes and the axis moves at the second-phase speed when the home switch is OFF. And where the first Z pulse is met is the home position.

Circumstance 2: When the home switch is ON MC_Home instruction is run and the axis moves in the positive direction at the second-phase speed. The motion direction changes and the axis moves at the second-phase speed when the home switch is OFF. And where the first Z pulse is met is the home position.

Circumstance 3: MC_Home instruction is run and the axis moves in the positive direction at the first-phase speed when the home switch is OFF. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF and the positive limit switch is ON. The axis moves at the second-phase speed and where the first Z pulse is met is the home position when the home switch is ON.

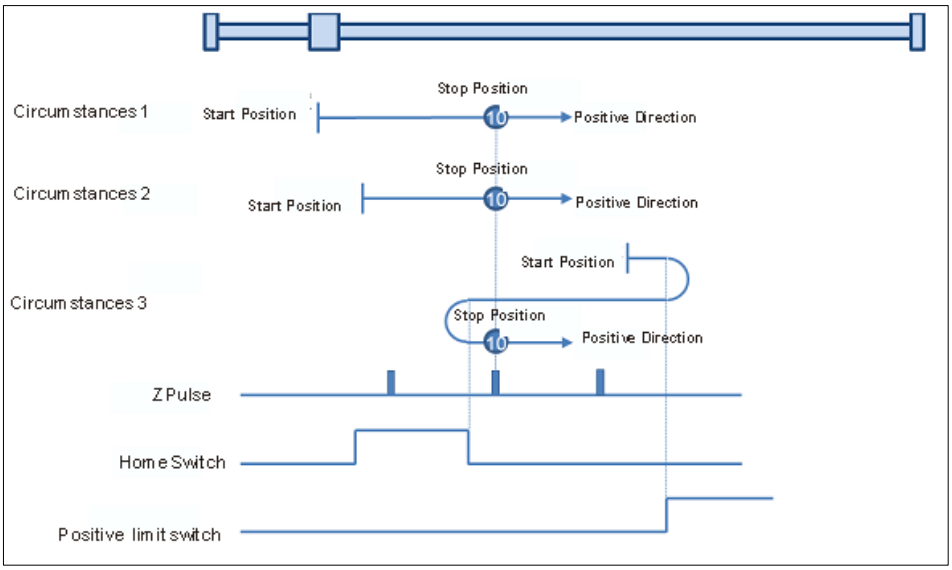


- **Mode 10: Homing depending on the home switch, positive limit switch and Z pulse.**

Circumstance 1: MC_Home instruction is run and the axis moves in the positive direction at the first-phase speed when the home switch is OFF. The axis moves at the second-phase speed when the home switch is ON. And where the first Z pulse is met is the home position while the home switch is OFF.

Circumstance 2: MC_Home instruction is run and the axis moves in the positive direction at the second-phase speed when the home switch is ON. And where the first Z pulse is met is the home position while the home switch is OFF.

Circumstance 3: MC_Home instruction is run and the axis moves in the positive direction at the first-phase speed when the home switch is OFF. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF and the positive limit switch is ON. The motion direction changes again and the axis moves at the second-phase speed when the home switch is ON. Where the first Z pulse is met is the home position while the home switch is OFF.



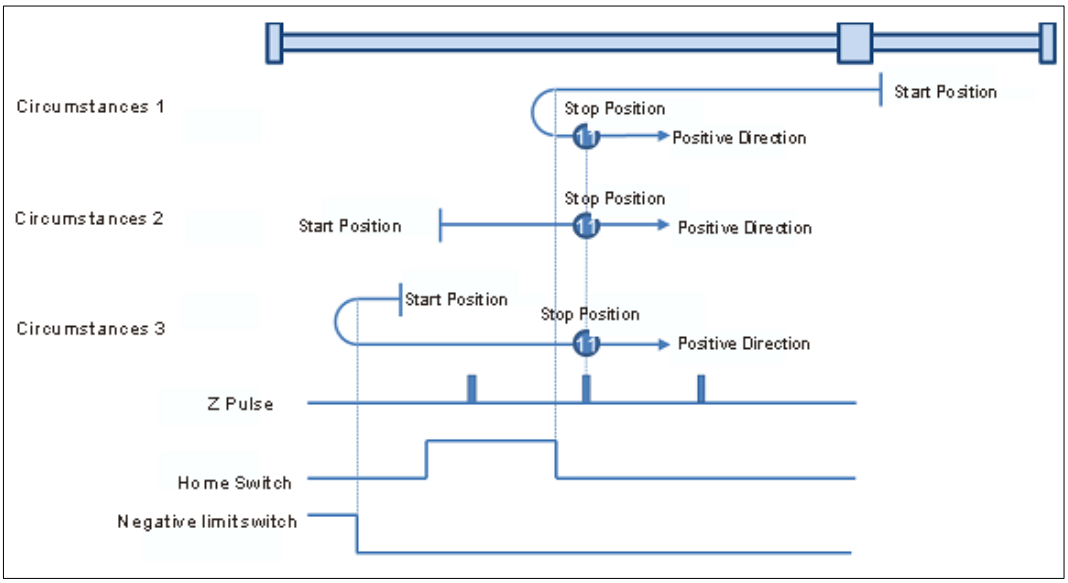
Mode 11~ mode 14 Homing which depends on the home switch, negative limit switch and Z pulse

- **Mode 11:**

Circumstance 1: MC_Home instruction is run and the axis moves in the negative direction at the first-phase speed when the home switch is OFF. The motion direction changes and the axis moves at the second-phase speed when the home switch is ON. And where the first Z pulse is met is the home position while the home switch is OFF.

Circumstance 2: MC_Home instruction is run and the axis moves in the positive direction at the second-phase speed while the home switch is ON. And where the first Z pulse is met is the home position while the home switch is OFF.

Circumstance 3: MC_Home instruction is run and the axis moves in the negative direction at the first-phase speed while the home switch is OFF. The motion direction changes and the axis moves at the first-phase speed while the home switch is OFF and the negative limit switch is ON. The axis moves at the second-phase speed when the home switch is ON. Where the first Z pulse is met is the home position while the home switch is OFF.



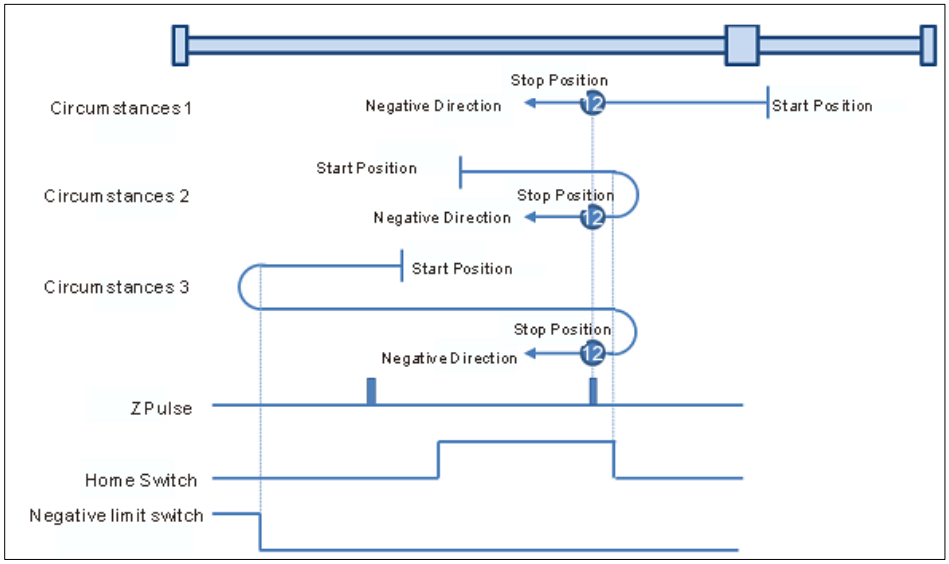
- **Mode 12: Homing depending on the home switch, negative limit switch and Z pulse**

Circumstance 1: MC_Home instruction is run and the axis moves in the negative direction at the first-phase speed when the home switch is OFF. The axis moves at the second-phase speed when the home switch is ON. And where the first Z pulse is met is the home position.

Circumstance 2: MC_Home instruction is run and the axis moves in the positive direction at the second-phase speed while the

home switch is ON. The motion direction changes and the axis moves at the second-phase speed while the home switch is OFF. And where the first Z pulse is met is the home position.

Circumstance 3: MC_Home instruction is run and the axis moves in the negative direction at the first-phase speed while the home switch is OFF. The motion direction changes and the axis moves at the first-phase speed while the home switch is OFF and the negative limit switch is ON. The axis still moves at the first-phase speed when the home switch is ON. The motion direction changes and the axis moves at the first-phase speed while the home switch is OFF. The axis moves at the second-phase speed while the home switch is ON. And where the first Z pulse is met is the home position.

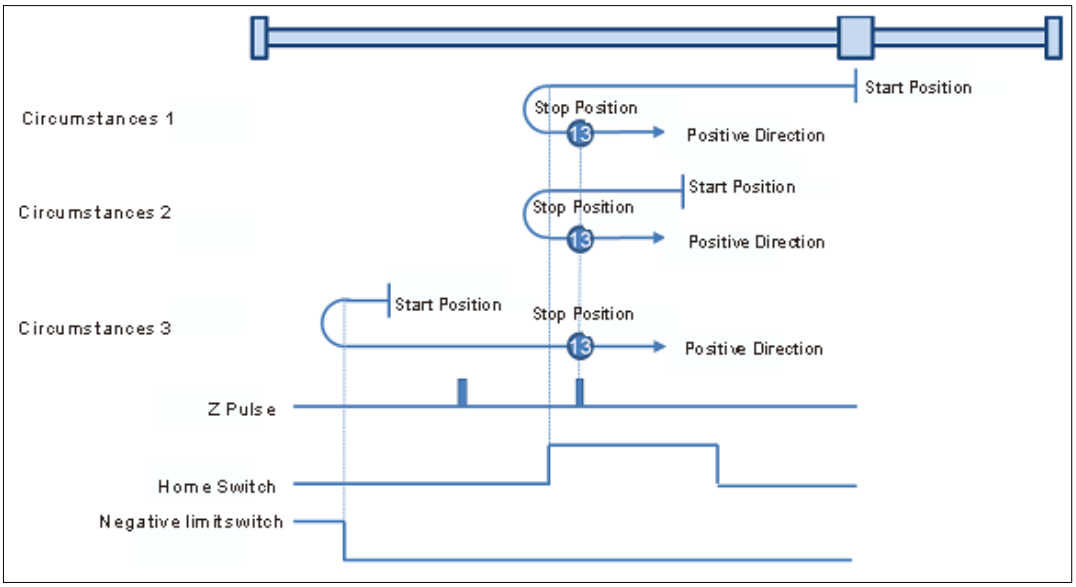


- **Mode 13: Homing depending on the home switch, negative limit switch and Z pulse**

Circumstance 1: MC_Home instruction is run and the axis moves in the negative direction at the first-phase speed while the home switch is OFF. The axis moves at the second-phase speed while the home switch is ON. The motion direction changes and the axis moves at the second-phase speed while the home switch is OFF. And where the first Z pulse is met is the home position.

Circumstance 2: MC_Home instruction is run and the axis moves in the negative direction at the second-phase speed while the home switch is ON. The motion direction changes and the axis moves at the second-phase speed while the home switch is OFF. And where the first Z pulse is met is the home position.

Circumstance 3: MC_Home instruction is run and the axis moves in the negative direction at the first-phase speed while the home switch is OFF. The motion direction changes and the axis moves at the first-phase speed while the home switch is OFF and the negative limit switch is ON. The axis moves at the second-phase speed and where the first Z pulse is met is the home position when the home switch is ON and the negative limit switch is OFF.

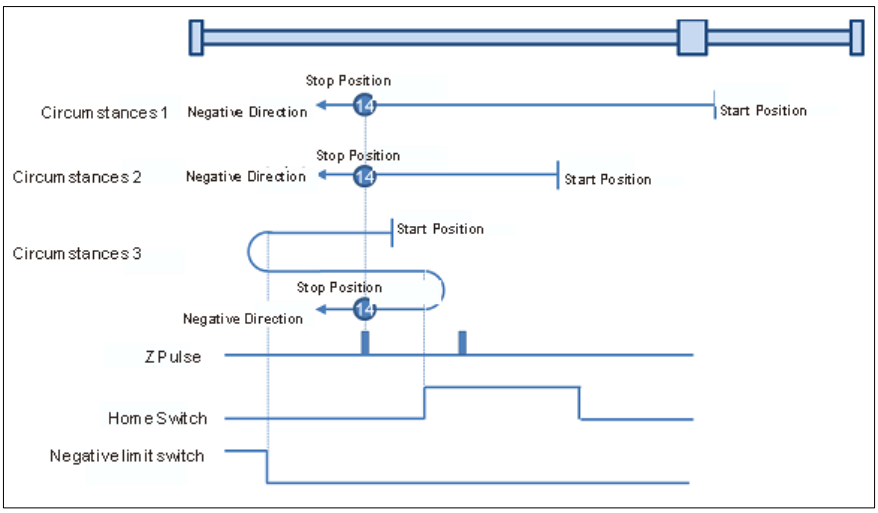


- **Mode 14: Homing depending on the home switch, negative limit switch and Z pulse**

Circumstance 1: MC_Home instruction is run and the axis moves in the negative direction at the first-phase speed while the home switch is OFF. The axis moves at the second-phase speed once the home switch is ON. And where the first Z pulse is met is the home position while the home switch is OFF.

Circumstance 2: MC_Home instruction is run and the axis moves in the negative direction at the second-phase speed while the home switch is ON. Where the first Z pulse is met is the home position while the home switch is OFF.

Circumstance 3: MC_Home instruction is run and the axis moves in the negative direction at the first-phase speed while the home switch is OFF. The motion direction changes and the axis moves at the first-phase speed while the home switch is OFF and the negative limit switch is ON. The motion direction changes again and the axis moves at the second-phase speed when the home switch is ON. Where the first Z pulse is met is the home position while the home switch is OFF.



Mode 15 and mode 16 are reserved for future development.

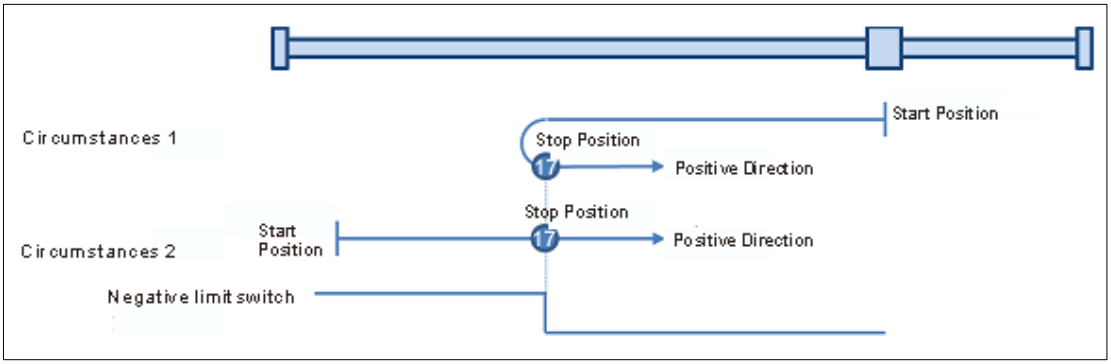
Mode 17~mode 30 Homing which has nothing to do with Z pulse

In mode 17~mode 30 which are respectively similar to mode1~mode 14 mentioned previously, the axis has nothing to do with Z pulse but the relevant home switch and limit switch status while returning to the home position.

- **Mode 17: Homing which depends on the negative limit switch, similar to mode 1, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run when the negative limit switch is OFF and the axis moves in the negative direction at the first-phase speed. The motion direction changes and the axis moves at the second-phase speed when the axis encounters that the negative limit switch is ON. Where the servo is when the negative limit switch is OFF is the home position.

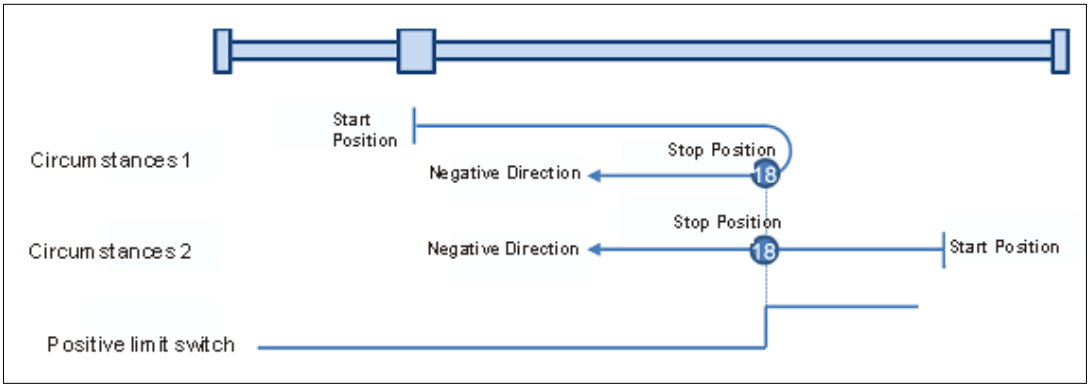
Circumstance 2: MC_Home instruction is run when the negative limit switch is ON and the axis moves in the positive direction at the second-phase speed. Where the servo is the home position when the negative limit switch is OFF.



- **Mode 18: Homing which depends on the positive limit switch, similar to mode 2, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run when the positive limit switch is OFF and the axis moves in the positive direction at the first-phase speed. The motion direction changes and the axis moves at the second-phase speed when the axis encounters that the positive limit switch is ON. Where the servo is the home position while the positive limit switch is OFF.

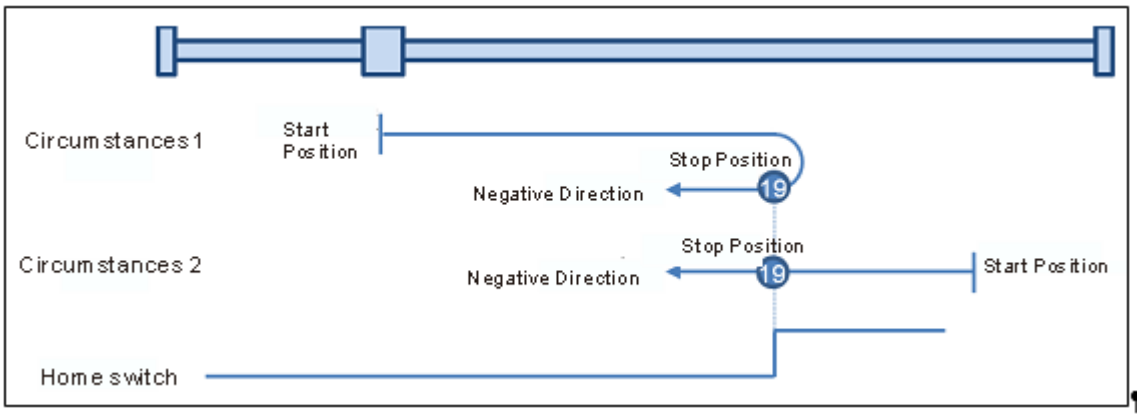
Circumstance 2: MC_Home instruction is run when the positive limit switch is ON and the axis moves in the negative direction at the second-phase speed. Where the servo is the home position while the positive limit switch is OFF.



- **Mode 19: Homing which depends on the home switch, similar to mode 3, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run and the axis moves in the positive direction at the first-phase speed while the home switch is OFF. The motion direction changes and the axis moves at the second-phase speed once the home switch becomes ON. And where the axis stands is the home position at the moment the home switch becomes OFF.

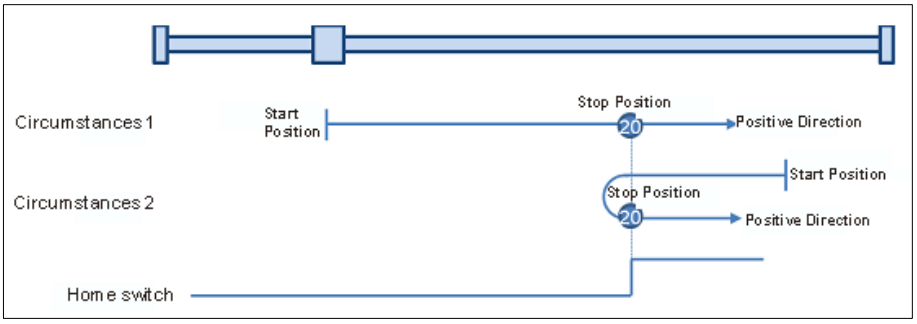
Circumstance 2: MC_Home instruction is run and the axis directly moves in the negative direction at the second-phase speed while the home switch is ON. And where the axis stands is the home position at the moment when the home switch becomes OFF.



- **Mode 20: Homing which depends on the home switch, similar to mode 4, but has nothing to do with Z pulse.**

Circumstance 1 : MC_Home instruction is run when the home switch is OFF and the axis moves in the positive direction at the first-phase speed. Where the servo is the home position when the home switch is ON.

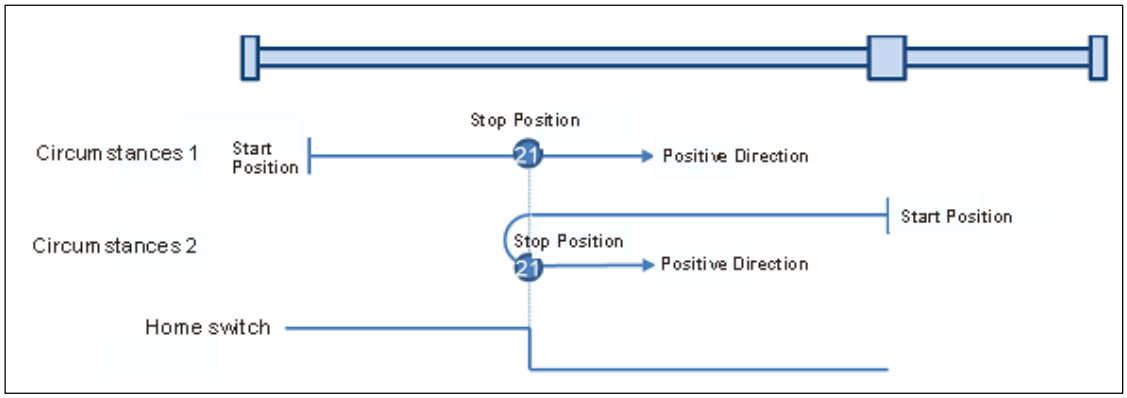
Circumstance 2 : MC_Home instruction is run when the home switch is ON and the axis moves in the negative direction at the second-phase speed. The motion direction changes and the axis moves at the second-phase speed when the home switch becomes OFF. Where the servo is the home position when the home switch is ON.



- **Mode 21: Homing which depends on the home switch, similar to mode 5, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run and the axis moves in the positive direction at the second-phase speed while the home switch is ON. And where the axis stands is the home position at the moment the home switch becomes OFF.

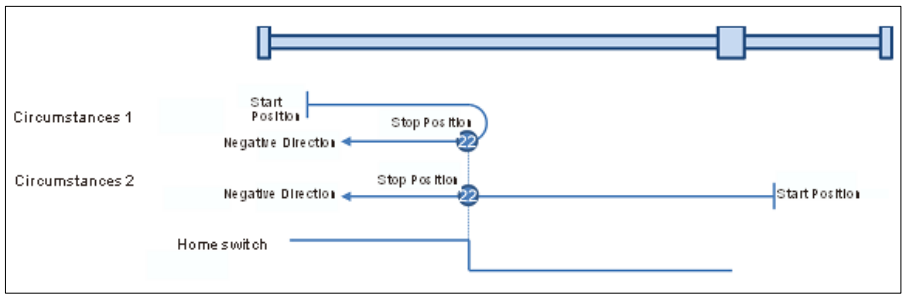
Circumstance 2: MC_Home instruction is run and the axis moves in the negative direction at the first-phase speed while the home switch is OFF. The motion direction changes and the axis moves at the second-phase speed once the home switch becomes ON. And where the axis stands is the home position at the moment the home switch becomes OFF.



- **Mode 22: Homing which depends on the home switch, similar to mode 6, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run while the home switch is ON and the axis moves in the positive direction at the second-phase speed. The motion direction changes and the axis moves at the second-phase speed once the home switch becomes OFF. Where the axis stands is the home position when the home switch is ON.

Circumstance 2: MC_Home instruction is run while the home switch is OFF and the axis moves in the negative direction at the first-phase speed. Where the axis stands is the home position when the home switch becomes ON.

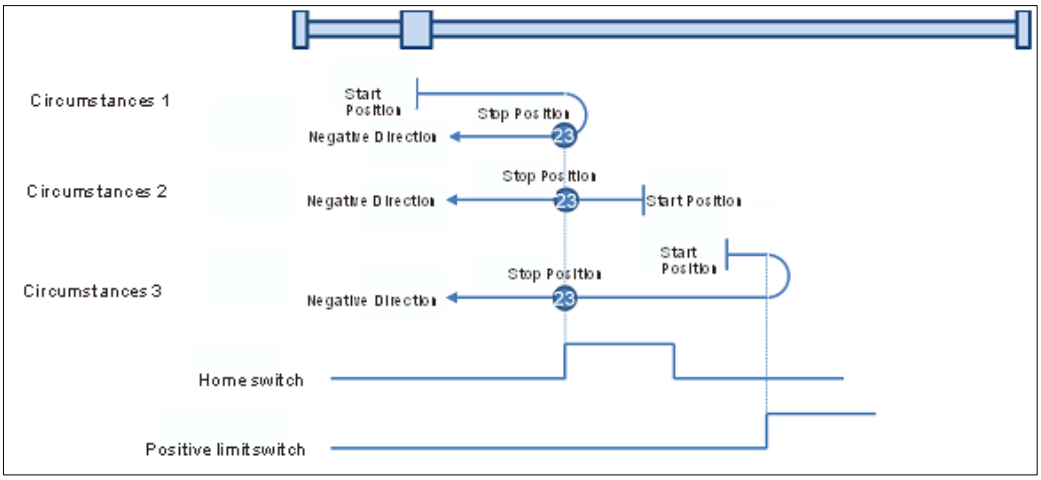


- **Mode 23: Homing which depends on the home switch and positive limit switch, similar to mode 7, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run while the home switch is OFF and the axis moves in the positive direction at the first-phase speed. The motion direction changes and the axis moves at the second-phase speed once the home switch becomes ON. Where the axis stands is the home position when the home switch is OFF.

Circumstance 2: MC_Home instruction is run while the home switch is ON and the axis moves in the negative direction at the second-phase speed. And where the axis stands is the home position when the home switch becomes OFF.

Circumstance 3: MC_Home instruction is run while the home switch is OFF. The axis moves in the positive direction at the first-phase speed. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF and the positive limit switch is ON. When the home switch is ON, the axis starts to move at the second-phase speed. Where the axis stands is the home position when the home switch is OFF.

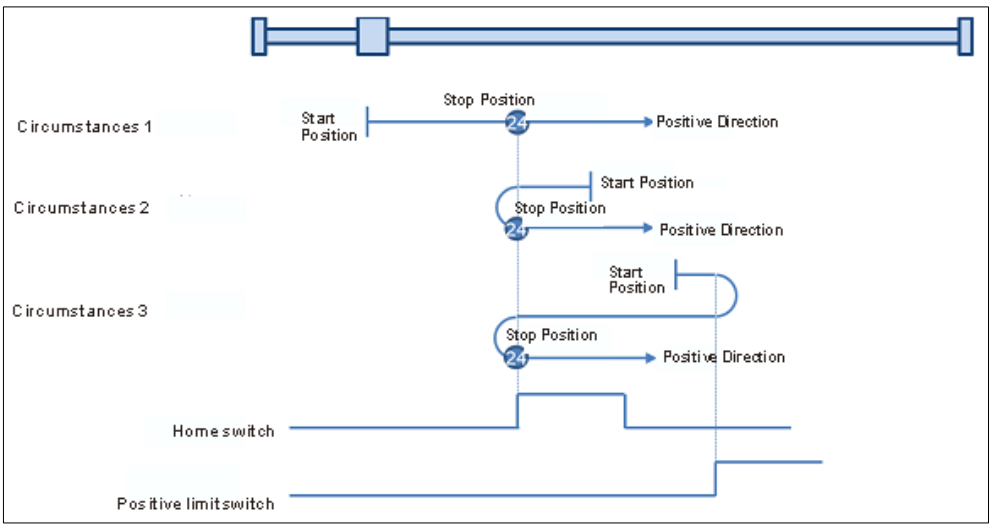


- **Mode 24: Homing which depends on the home switch and positive limit switch, similar to mode 8, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run while the home switch is OFF and the axis starts to move in the positive direction at the first-phase speed. Where the axis stands is the home position when the home switch is ON.

Circumstance 2: MC_Home instruction is run while the home switch is ON and the axis moves in the negative direction at the second-phase speed. The motion direction changes and the axis moves at the second-phase speed when the home switch is OFF. Where the axis stands is the home position when the home switch is ON.

Circumstance 3: MC_Home instruction is run while the home switch is OFF. The axis moves in the positive direction at the first-phase speed. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF and the positive limit switch is ON. When the home switch is ON, the axis still moves at the first-phase speed. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF. Where the axis stands is the home position when the home switch is ON.

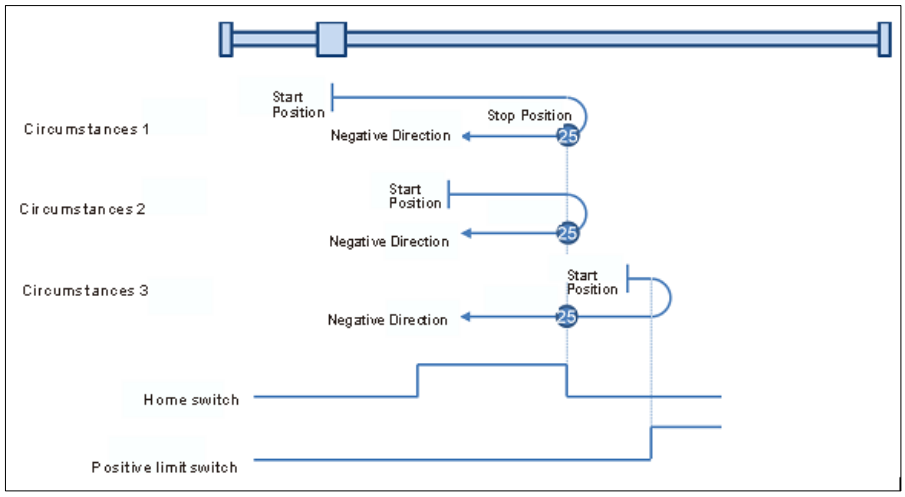


- **Mode 25: Homing which depends on the home switch and positive limit switch, similar to mode 9, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run while the home switch is OFF and the axis starts to move in the positive direction at the first-phase speed. The axis moves at the second-phase speed when the home switch is ON. The motion direction changes and the axis moves at the second-phase speed when the home switch is OFF. Where the axis stands is the home position when the home switch is ON.

Circumstance 2: MC_Home instruction is run while the home switch is ON and the axis moves in the positive direction at the second-phase speed. The motion direction changes and the axis moves at the second-phase speed when the home switch is OFF. Where the axis stands is the home position when the home switch is ON.

Circumstance 3: MC_Home instruction is run while the home switch is OFF. The axis moves in the positive direction at the first-phase speed. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF and the positive limit switch is ON. Where the axis stands is the home position when the home switch is ON.

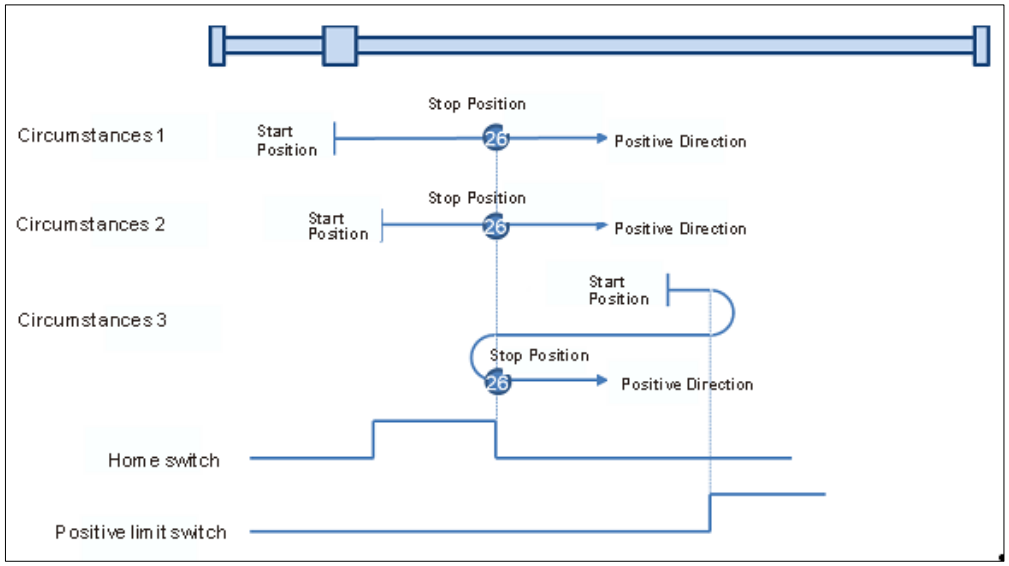


- **Mode 26: Homing which depends on the home switch and positive limit switch, similar to mode 10, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run while the home switch is OFF and the axis starts to move in the positive direction at the first-phase speed. The axis moves at the second-phase speed when the home switch is ON. Where the axis stands is the home position when the home switch is OFF.

Circumstance 2: MC_Home instruction is run while the home switch is ON and the axis moves in the positive direction at the second-phase speed. Where the axis stands is the home position when the home switch is OFF.

Circumstance 3: MC_Home instruction is run while the home switch is OFF. The axis moves in the positive direction at the first-phase speed. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF and the positive limit switch is ON. The motion direction changes again and the axis moves at the second-phase speed when the home switch is ON. Where the axis stands is the home position when the home switch is OFF.

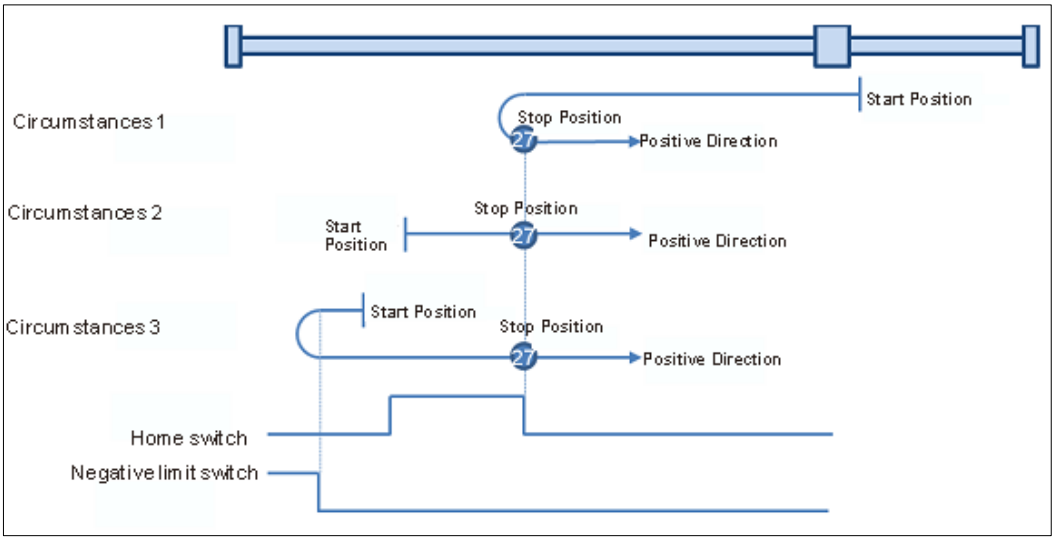


- **Mode 27: Homing which depends on the home switch and negative limit switch, similar to mode 11, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run while the home switch is OFF and the axis starts to move in the negative direction at the first-phase speed. The motion direction changes and the axis moves at the second-phase speed when the home switch is ON. Where the axis stands is the home position when the home switch is OFF.

Circumstance 2: MC_Home instruction is run while the home switch is ON and the axis moves in the positive direction at the second-phase speed. Where the axis stands is the home position when the home switch is OFF.

Circumstance 3: MC_Home instruction is run while the home switch is OFF. The axis moves in the negative direction at the first-phase speed. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF and the negative limit switch is ON. When the home switch is ON, the axis starts to move at the second-phase speed. Where the axis stands is the home position when the home switch is OFF.

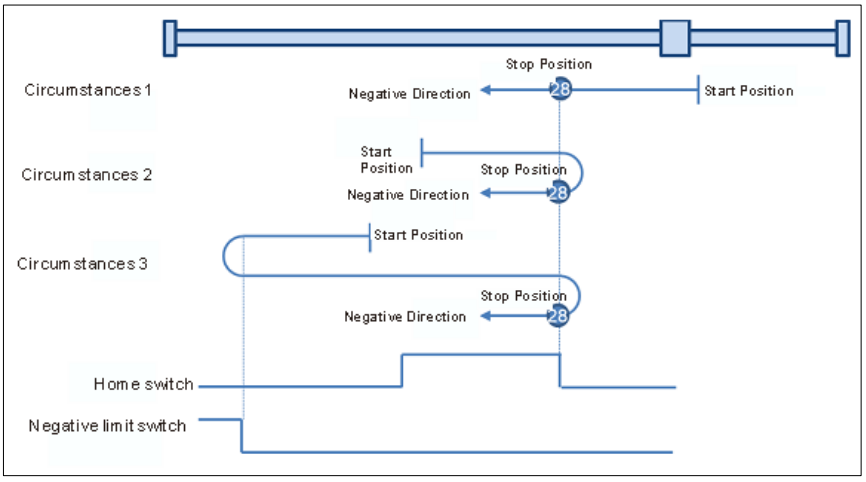


- **Mode 28: Homing which depends on the home switch and negative limit switch, similar to mode 12, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run while the home switch is OFF and the axis starts to move in the negative direction at the first-phase speed. Where the axis stands is the home position when the home switch is ON.

Circumstance 2: MC_Home instruction is run while the home switch is ON and the axis moves in the positive direction at the second-phase speed. The motion direction changes and the axis moves at the second-phase speed when the home switch is OFF. Where the axis stands is the home position when the home switch is ON.

Circumstance 3: MC_Home instruction is run while the home switch is OFF. The axis moves in the negative direction at the first-phase speed. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF and the negative limit switch is ON. When the home switch is ON, the axis still moves at the first-phase speed. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF. Where the axis stands is the home position when the home switch is ON.

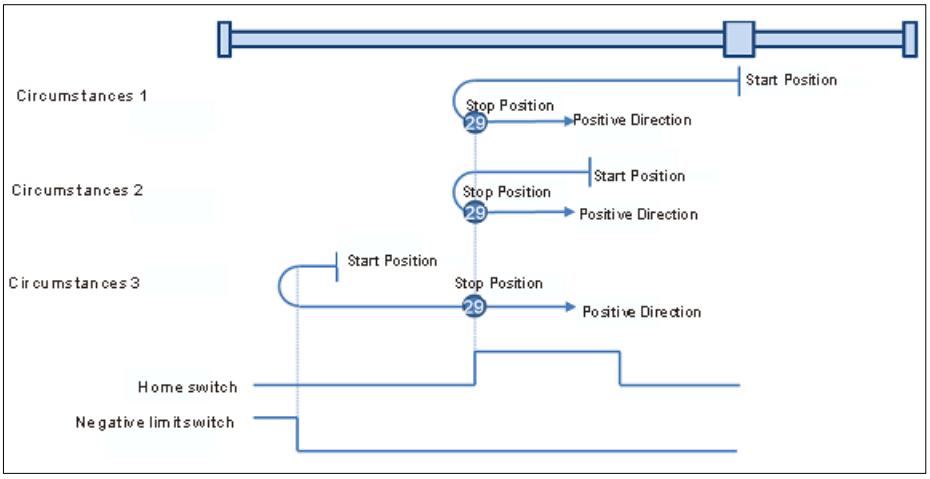


- **Mode 29: Homing which depends on the home switch and negative limit switch, similar to mode 13, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run while the home switch is OFF and the axis starts to move in the negative direction at the first-phase speed. When the home switch is ON, the axis starts to move at the second-phase speed. The motion direction changes and the axis moves at the second-phase speed when the home switch is OFF. Where the axis stands is the home position when the home switch is ON.

Circumstance 2: MC_Home instruction is run while the home switch is ON and the axis moves in the negative direction at the second-phase speed. The motion direction changes and the axis moves at the second-phase speed when the home switch is OFF. Where the axis stands is the home position when the home switch is ON.

Circumstance 3: MC_Home instruction is run while the home switch is OFF. The axis moves in the negative direction at the first-phase speed. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF and the negative limit switch is ON. Where the axis stands is the home position when the home switch is ON.

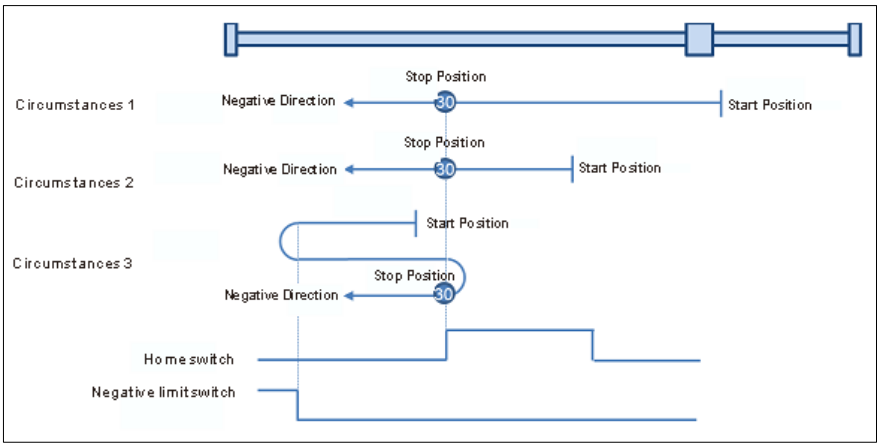


- **Mode 30: Homing which depends on the home switch and negative limit switch, similar to mode 14, but has nothing to do with Z pulse.**

Circumstance 1: MC_Home instruction is run while the home switch is OFF and the axis starts to move in the negative direction at the first-phase speed. When the home switch is ON, the axis starts to move at the second-phase speed. Where the axis stands is the home position when the home switch is OFF.

Circumstance 2: MC_Home instruction is run while the home switch is ON and the axis moves in the negative direction at the second-phase speed. Where the axis stands is the home position when the home switch is OFF.

Circumstance 3: MC_Home instruction is run while the home switch is OFF. The axis moves in the negative direction at the first-phase speed. The motion direction changes and the axis moves at the first-phase speed when the home switch is OFF and the negative limit switch is ON. When the home switch is ON, the motion direction changes again and the axis moves at the second-phase speed. Where the axis stands is the home position when the home switch is OFF



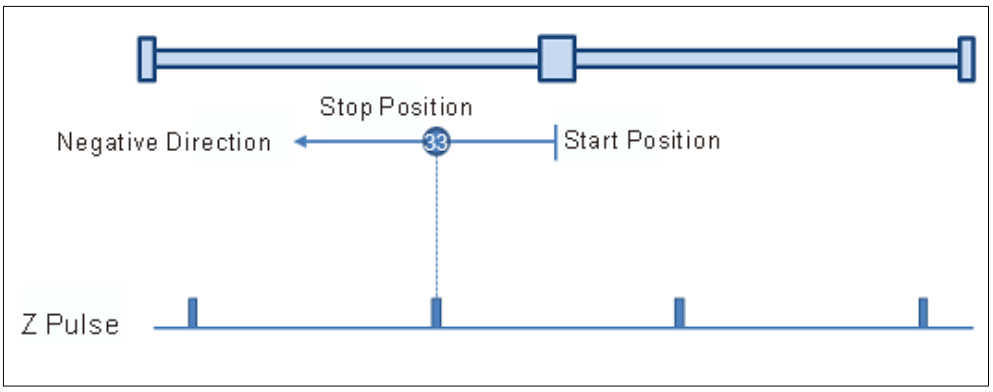
Mode 31 and mode 32: Reserved

Mode 31 and mode 32 Reserved for future homing.

Mode 33 ~ mode 34 Homing which only depends on Z pulse

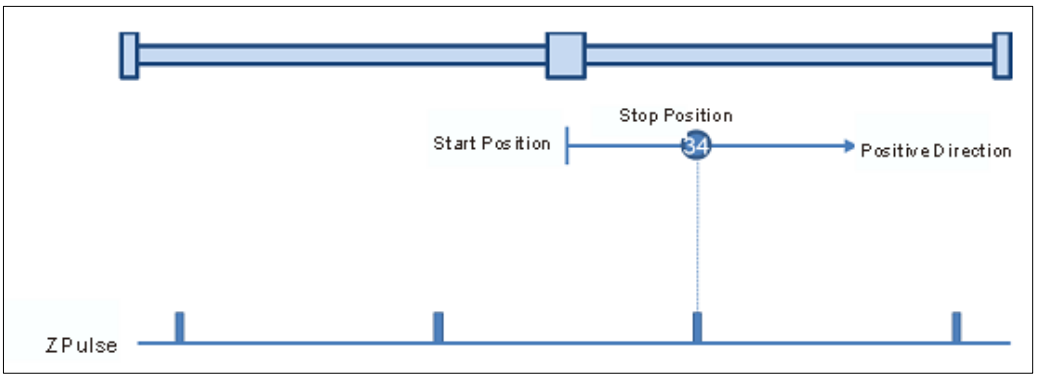
- **Mode 33: Homing depending on Z pulse (Negative direction)**

  MC_Home instruction is run and the axis moves at the second-phase speed in the negative direction. And the place where the axis stands is the home position once the first Z pulse is met.



- **Mode 34: Homing depending on Z pulse (Positive direction)**

MC_Home instruction is run and the axis moves at the second-phase speed in the positive direction. And the place where the axis stands is the home position once the first Z pulse is met.



- **Mode 35: Homing which depends on the current position**

MC_Home instruction is run, the axis does not move and its current position is regarded as the home position.