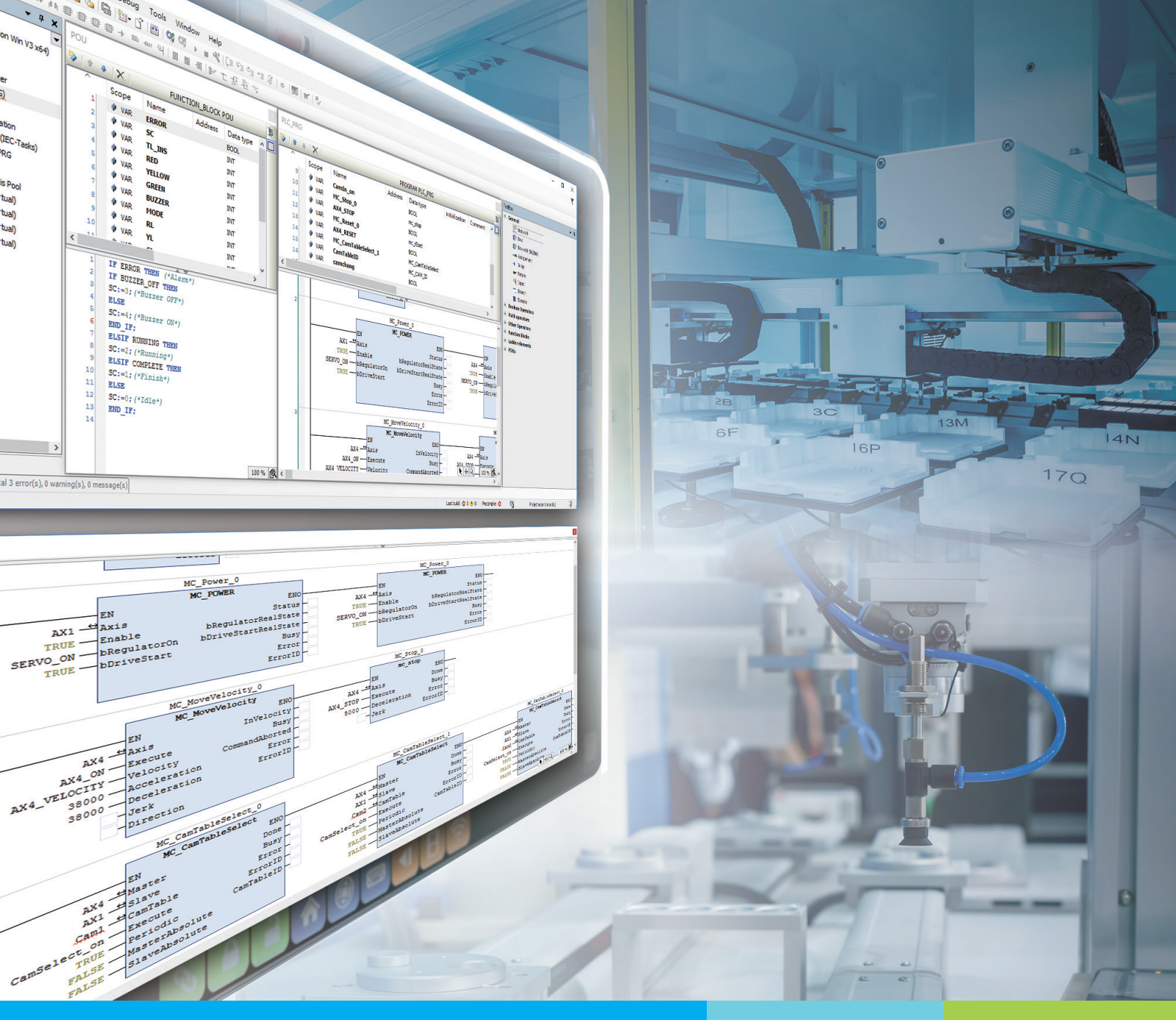


Digitized Automation for a Changing World

DIADesigner - AX Series Motion Controller Instructions Manual

www.deltaww.com

Revision History

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.	09/15/2022

Version	Revision	Date
	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.	
1.4.0	1. Updated Section 2.1.1: Updated function description. 2. Updated Section 2.3.1 and 2.3.2: Added version compatibility error troubleshooting. 3. Updated Section 2.3.1.1: Updated function description. 4. Updated Section 2.3.4.1–2.3.4.3: Added library supported version. 5. Added Section 2.3.2.29: Added DMC_GetCamSlaveData.	03/31/2023
1.5.0	1. Added Section 1.1.1: Added function blocks that support buffer. 2. Updated Section 1.2: Added function block behavior when meets software/hardware limit. 3. Updated Section 2.1.7: Added function introduction. 4. Revised Section 2.1.16: Revised output updating time. 5. Updated Section 2.2.1: Added function introduction. 6. Updated Section 2.2.13: Added example. 7. Revised Section 2.3.2.14: Added function introduction. 8. Revised Section 2.1.8: Revised parameter descriptions. 9. Added sections 2.2.18 and 2.2.19: Added description of new function blocks. 10. Updated Section 2.3.1.1: Added new parameters and function introduction. 11. Revised Section 2.3.1.5 and 2.3.1.6: Revised some descriptions. 12. Updated Section 2.3.1.3–2.3.1.6, 2.3.1.19–2.3.1.20: Updated TransitionMode. 13. Added Section 2.3.1.23–2.3.1.30: Added new function blocks. 14. Updated Section 2.3.1.1, 2.3.1.2, 2.3.2.19, 2.3.2.22, 2.3.3: Function blocks are not supported in PLC simulation mode. 15. Updated Section 2.3.2.14, 2.3.2.15: Added function introduction. 16. Add Section 2.3.2.30: Added DMC_GetDeltaServoDriveError. 17. Updated Section 2.3.3: Added function block input minimum value and positioning axis version. 18. Updated Section A.1: Added DMC_GetDeltaServoDriveError. 19. Updated Section A.2: Added DMC_GROUP_TRANSITION_MODE.	09/28/2023

Version	Revision	Date
	20. Updated Section A.3.1: Added corrective actions for DMC_TC_INVALID_PDO_MAPPING and DMC_CWP_INVALID_MASTEPOS. 21. Updated Section A.3.3: Added SM3_Drive_ETC error codes and troubleshooting.	
1.6.0	1. Section 1.2: Updated descriptions of motion control instructions. 2. Section 1.3: Added the version compatibility between Delta motion control instruction and Codesys Softmotion instruction library. 3. Section 1.4: Added supported devices. 4. Section 2.1: Added this section. 5. Section 2.1.1: Restructured the subsections. 6. Section 2.1.1.8: Updated the output updating time of InSync and EndofProfile. 7. Section 2.1.1.17: Added a new function description. 8. Section 2.1.2: Restructured this section. 9. Section 2.1.2.20: Added MC_SetOveridde. 10. Section 2.2: Changed the title to DL_MotionControl. 11. Section 2.2.1.1: Added function descriptions. 12. Section 2.2.1.2: Added function descriptions. 13. Section 2.2.1.23–2.2.1.28: Updated function descriptions. 14. Section 2.2.1.26: Updated the value scope and function description of diRatioDenominator. 15. Section 2.2.1.27: Update the bStatus output updating time. 16. Section 2.2.2.15: Updated the example. 17. Section 2.2.2.27: Updated bStatus output update timing. 18. Section 2.3: Added this section. 19. Section 2.3.1: Restructured this section. Added the differences between the position axes and the velocity axis and the supported instruction of the axis. 20. Section 2.3.1.4: Added new function descriptions. 21. Section 2.3.1.8: Added new function descriptions. 22. Section 2.3.2: Restructured this section. 23. Section 2.4: Changed the title to DL_ServoPress_AX. 24. Section 2.5: Added this section and the subsection of MC_JogToPosition, MC_MasterEngine, MC_FlyingShear, MC_Axes_Interlock. 25. Section A1.1: Added positioning, sync, and administrative instructions for the Single-axis Motion Control Instruction (Synchronous axis). 26. Section A1.2: Added instructions supported for the AX-8 controller.	03/29/2024

Version	Revision	Date
	27. Section A1.3: Added instructions: MC_Axes_Interlock, MC_FlyingShear, MC_JogToPosition, MC_MasterEngine, MC_SetOverride 28. Section A3.1: Added error codes: 0x0001B, 0x0079E, 0x01771, 0x18959.	
1.7.0	1. Section 1.5: Added PDO mapping descriptions. 2. Section 2.1.1.8: Updated function descriptions. 3. Section 2.1.1.15: Updated function descriptions. 4. Section 2.1.1.17: Updated function descriptions. 5. Section 2.1.1.18: Updated function descriptions. 6. Section 2.2.1.5–2.2.1.6: Updated the data type and setting value of the AuxPoint parameter, and added the descriptions of CircMode to Function. 7. Section 2.2.1.9: Updated function descriptions. 8. Section 2.2.1.24: Updated function descriptions. 9. Section 2.2.1.26: Updated function descriptions. 10. Section 2.2.1.27: Added the MasterResverseAction parameter and updated function descriptions. 11. Section 2.2.1.29: Updated function descriptions. 12. Section 2.2.1.30: Updated function descriptions. 13. Section 2.2.2.4: Modified the default value of PARAM_PLANNING_PRIORITY. 14. Section 2.2.2.15: Updated the image of the example. 15. Section 2.2.2.31: Added the new DMC_GetCamMasterData function block section. 16. Section 2.3.1: Added a mapping table and updated the firmware version of devices supporting positioning axis. 17. Section 2.3.1.8: Modified the default value range of input IrVelocity and updated function descriptions. 18. Section 2.3.1.17: Modified the default value range of input IrVelocity and updated function descriptions. 19. Section A.3.1: Added a corrective action for 0x00003 and an error code 0x18A1F.	9/30/2024
1.8.0	1. Section 2.1.1.7: Updated function descriptions. 2. Section 2.1.1.8: Updated function descriptions. 3. Section 2.1.1.15: Updated function descriptions. 4. Section 2.1.1.17: Updated function descriptions. 5. Section 2.1.1.18: Updated function descriptions. 6. Section 2.2.1.1: Added a new input parameter DisableMode and updated function descriptions. 7. Section 2.2.1.2: Modified the note of the parameter. 8. Section 2.2.1.27: Updated diagrams of function descriptions.	4/28/2025

Version	Revision	Date
	9. Section 2.2.2.22: Updated function descriptions. 10. Section 2.3: Added MX300 device and updated descriptions. 11. Section A1.2: Added AX-5 and AX-C devices. 12. Section A2: Updated DMC_GROUP_TRANSITION_MODE and DMC_CIRC_PLANE. Added DMC_DISABLEMODE.	

TOC

P1 Preface..... 1
P1.1 Introduction.....2
P1.1.1 Applicable Products.....3
P1.1.2 Associated Manuals.....4
Chapter 1 Introduction to Motion Control..... 5
1.1 Introductions of Motion Control.....6
1.1.1 Basic Knowledge of Motion Control Instructions.....7
1.2 Descriptions of Motion Control Instructions.....10
1.3 Overview of Delta Motion Instructions Library Version.....11
1.4 Device Categories.....12
1.5 PDO Mapping.....13
Chapter 2 Motion Control Instruction..... 14
2.1 SM3 Basic.....15
2.1.1 Motion Control Instructions.....16
2.1.1.1 MC_Home.....17
2.1.1.2 MC_Stop.....21
2.1.1.3 MC_Halt.....25
2.1.1.4 MC_MoveAbsolute.....29
2.1.1.5 MC_MoveRelative 35
2.1.1.6 MC_MoveAdditive.....39
2.1.1.7 MC_MoveSuperImposed.....43
2.1.1.8 MC_CamIn.....48
2.1.1.9 MC_CamOut.....65
2.1.1.10 MC_MoveVelocity.....69
2.1.1.11 MC_PositionProfile.....74
2.1.1.12 MC_VelocityProfile.....79
2.1.1.13 MC_AccelerationProfile.....84
2.1.1.14 MC_Jog.....89
2.1.1.15 MC_GearIn.....92
2.1.1.16 MC_GearOut.....96
2.1.1.17 MC_GearInPos.....101
2.1.1.18 MC_Phasing.....107
2.1.2 Administrative Motion Control Instructions.....112
2.1.2.1 MC_Power.....113
2.1.2.2 MC_SetPosition.....118
2.1.2.3 MC_ReadParameter.....124
2.1.2.4 MC_WriteParameter.....127
2.1.2.5 MC_ReadBoolParameter.....130
2.1.2.6 MC_WriteBoolParameter.....133
2.1.2.7 MC_ReadActualPosition.....135
2.1.2.8 MC_ReadActualVelocity.....138
2.1.2.9 MC_ReadActualTorque.....141
2.1.2.10 MC_Reset.....143
2.1.2.11 MC_ReadStatus.....146

2.1.2.12 MC_ReadAxisError..... 150
2.1.2.13 MC_CamTableSelect..... 153
2.1.2.14 MC_TouchProbe..... 159
2.1.2.15 MC_AbortTrigger..... 166
2.1.2.16 MC_DigitalCamSwitch..... 170
2.1.2.17 SMC_BacklashCompensation..... 177
2.1.2.18 SMC3_ETC_ReadParameter_CoE..... 183
2.1.2.19 SMC3_ETC_WriteParameter_CoE..... 186
2.1.2.20 MC_SetOverride..... 189
2.2 DL_MotionControl..... 193
2.2.1 Motion Control Instructions..... 194
2.2.1.1 DMC_TorqueControl..... 195
2.2.1.2 DMC_VelocityControl..... 200
2.2.1.3 DMC_MoveLinearAbsolute..... 206
2.2.1.4 DMC_MoveLinearRelative..... 211
2.2.1.5 DMC_MoveCircularAbsolute..... 216
2.2.1.6 DMC_MoveCircularRelative..... 224
2.2.1.7 DMC_GroupStop..... 231
2.2.1.8 DMC_GroupHalt..... 236
2.2.1.9 DMC_Home_P..... 241
2.2.1.10 DMC_ImmediateStop_P..... 246
2.2.1.11 DMC_MoveVelocityStopByPos..... 248
2.2.1.12 DMC_GroupInterrupt..... 254
2.2.1.13 DMC_GroupContinue..... 261
2.2.1.14 DMC_MoveLink..... 264
2.2.1.15 DMC_MoveFeed..... 273
2.2.1.16 DMC_GroupReadSetPosition..... 281
2.2.1.17 DMC_GroupReadActPosition..... 284
2.2.1.18 DMC_GroupJog..... 287
2.2.1.19 DMC_MoveDirectAbsolute..... 292
2.2.1.20 DMC_MoveDirectRelative..... 298
2.2.1.21 DMC_MoveModulo..... 302
2.2.1.22 DMC_Home_E..... 308
2.2.1.23 DMC_CombineAxis..... 313
2.2.1.24 DMC_GearIn..... 318
2.2.1.25 DMC_GearOut..... 322
2.2.1.26 DMC_GearInPos..... 325
2.2.1.27 DMC_CamIn..... 329
2.2.1.28 DMC_CamOut..... 349
2.2.1.29 DMC_PhasingAbsolute..... 351
2.2.1.30 DMC_PhasingRelative..... 355
2.2.2 Administrative Motion Control Instructions..... 361
2.2.2.1 DMC_GroupEnable..... 362
2.2.2.2 DMC_GroupDisable..... 367
2.2.2.3 DMC_GroupReadParameter..... 370
2.2.2.4 DMC_GroupWriteParameter..... 373

2.2.2.5 DMC_GroupReadStatus..... 376
2.2.2.6 DMC_GroupReadError..... 382
2.2.2.7 DMC_GroupReset..... 385
2.2.2.8 DMC_CamReadTappetStatus..... 387
2.2.2.9 DMC_CamReadTappetValue..... 392
2.2.2.10 DMC_CamWriteTappetValue..... 396
2.2.2.11 DMC_CamAddTappet..... 401
2.2.2.12 DMC_CamDeleteTappet..... 406
2.2.2.13 DMC_CamReadPoint..... 410
2.2.2.14 DMC_CamWritePoint..... 413
2.2.2.15 DMC_ChangeMechanismGearRation..... 417
2.2.2.16 DMC_ReadMotionState..... 421
2.2.2.17 DMC_AxesObserve 427
2.2.2.18 DMC_PositionLag..... 431
2.2.2.19 DMC_SetTorqueLimit..... 435
2.2.2.20 DMC_SetSoftwareLimit..... 438
2.2.2.21 DMC_CamKeyPointWrite..... 441
2.2.2.22 DMC_TouchProbeCyclically..... 450
2.2.2.23 DMC_CAMBounds..... 458
2.2.2.24 DMC_AddAxisToGroup..... 463
2.2.2.25 DMC_RemoveAxisFromGroup..... 466
2.2.2.26 DMC_UngroupAllAxes..... 469
2.2.2.27 DMC_GroupPower..... 472
2.2.2.28 DMC_GroupSetOverride..... 475
2.2.2.29 DMC_GetCamSlaveData..... 479
2.2.2.30 DMC_GetDeltaServoDriveError..... 482
2.2.2.31 DMC_GetCamMasterData..... 485
2.3 DL_MotionControlLight..... 489
2.3.1 Positioning Axis and Velocity Axis Instructions..... 490
2.3.1.1 MC_Power_DML..... 495
2.3.1.2 MC_Stop_DML..... 498
2.3.1.3 MC_Reset_DML..... 501
2.3.1.4 MC_Halt_DML..... 504
2.3.1.5 MC_Home_DML..... 507
2.3.1.6 MC_MoveAbsolute_DML..... 510
2.3.1.7 MC_MoveRelative_DML..... 513
2.3.1.8 MC_MoveVelocity_DML..... 516
2.3.1.9 MC_WriteBoolParameter_DML..... 519
2.3.1.10 MC_ReadBoolParameter_DML..... 522
2.3.1.11 MC_WriteParameter_DML..... 525
2.3.1.12 MC_ReadParameter_DML..... 528
2.3.1.13 MC_ReadStatus_DML..... 531
2.3.1.14 MC_TorqueControl_DML..... 533
2.3.1.15 MC_ChangeAxisConfig_DML..... 537
2.3.1.16 MC_ReinitDrive_DML..... 540
2.3.1.17 MC_VelocityControl_DML..... 542

2.3.2 Tension Control....545
2.3.2.1 R2R_Configuration....546
2.3.2.2 R2R_RollDiameter....553
2.3.2.3 R2R_Run....558
2.4 DL_ServoPress_AX....568
2.4.1 Servo Press Instruction....569
2.4.1.1 DFC_SP_DegreeToHeight....570
2.4.1.2 DFC_SP_HeightToDegreeDFC_SP_HeightToDegree....573
2.4.1.3 DMC_SP_CamCoinCurve....575
2.4.1.4 DMC_SP_CamCrankCurve....580
2.4.1.5 DMC_SP_CamCurveWrite....584
2.4.1.6 DMC_SP_CamLinkCurve....591
2.4.1.7 DMC_SP_CamPendulumCurve....596
2.4.1.8 DMC_SP_CamPulse1Curve....601
2.5 DL_PLCopenPart3....607
2.5.1 Motion Control Instructions....608
2.5.1.1 MC_JogToPosition....609
2.5.1.2 MC_MasterEngine....613
2.5.1.3 MC_FlyingShear....619
2.5.2 Administrative Motion Control Instructions....624
2.5.2.1 MC_Axes_Interlock....625
A. Appendix....628
A.1 Instruction Tables and Indexes....629
A.1.1 By Function....630
A.1.2 By Device....637
A.1.3 By Letter....641
A.2 Data Type: Enumeration and Structure....645
A.3 Error Codes and Troubleshooting....651
A.3.1 For Synchronous Axis....652
A.3.2 For Positioning Axis....680
A.3.3 For SM3_Drive_ETC Library....687
A.4 Explanation of DMC_Home_P....688

P1 Preface

P1.1 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.

P1.1.1 Applicable Products

This manual relates to the following products.

- AX-3 series
- AX-8 series
- AX-C series

P1.1.2 Associated Manuals

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

- *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.

- *AX-3 Series Operation 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.

- *AX-8 Series Operation 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.

- *AX-C Series Operation 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.

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_	

  ***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 running: Execute and Enable. When the instruction is run or turned on, 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-running of a Function Block**
  Re-running 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 interrupted by Aborting under the buffer mode.

- **Multi-running of Multiple Motion Control Instructions**
  Multi-running of motion control instructions means that multiple instructions on the same axis are run in the same task running period. The pattern of multi-running 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 run 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 of buffer mode.

- **Buffer Mode Supported Function Blocks**

Function Block	Buffer Mode Supported	Followed by Buffer Mode Supported Function Blocks	Timing to run the successive function blocks
MC_Power	○	○	
MC_Home	○	○	
MC_Stop	○	○	
MC_Halt	○	○	
MC_MoveAbsolute	●	●	*Done*
MC_MoveRelative	●	●	*Done*
MC_MoveSuperimposed	○	○	
MC_MoveVelocity	●	● Only Buffered	*InVelocity*
MC_CamIn	○	● Cyclic (Only Buffered)	*EndOfProfile*
MC_CamOut	○	● Only Buffered	*Done*
MC_GearIn	● Only BlendingPrevious	● Only Buffered	*InGear*
MC_GearOut	○	● Only Buffered	*Done*
MC_GearInPos	● Only BlendingPrevious	● Only Buffered	*InSync*
MC_Jog	○	○	
SMC_BacklashCompensation	○	○	

● = Supported

○ = Not supported

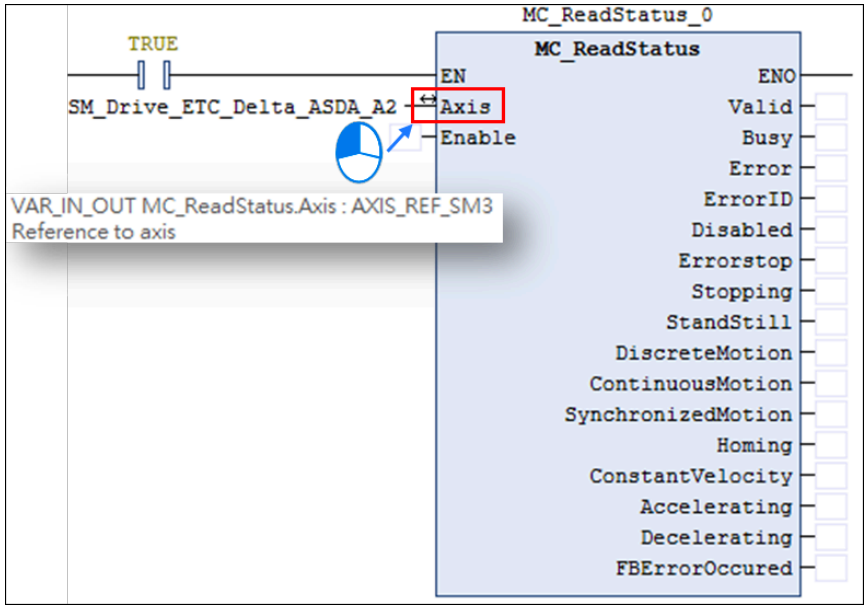
- **Structure Applicable for Motithe on Control**

  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.

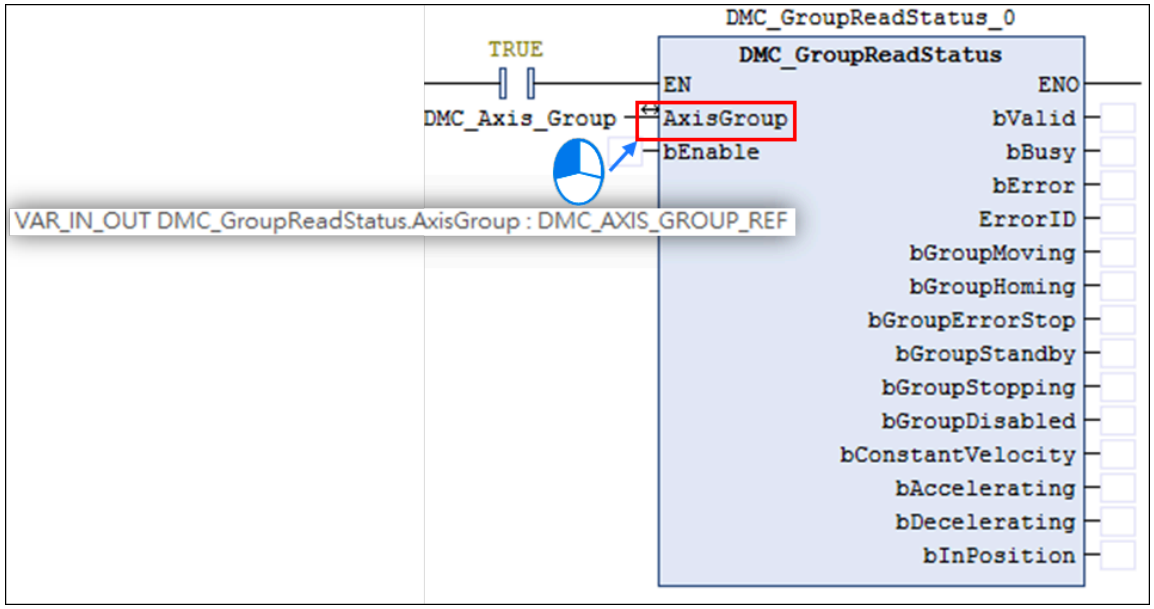
  For AX Motion CPUs, the applicable Structure is as below:

- **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 A.2 Data Types: Enumeration and Structure.

1.2 Descriptions of Motion Control Instructions

AX Motion Control CPU has the following five built-in libraries to support different scenarios.

Library	Description
SM3 Basic	Motion control instructions based on PLCopen, including motion control and administrative function blocks
DL_MotionControl	Delta's custom motion control instructions, including motion control and administrative function blocks
DL_MotionControlLight	Suitable for PP and PV modes, including motion control and administrative function blocks
DL_ServoPress_AX	Suitable for function blocks used in the servo press industry
DL_PLCopenPart3	Convenient function blocks that integrates FB for common applications

Note: The function blocks in the library are divided into the following by different prefixes:

- "MC_": PLCopen-based motion control instructions
- "DMC_": Delta customized motion control instructions
- "SMC_": Motion instructions

1.3 Overview of Delta Motion Instructions Library Version

This section describes the version relations of Delta's instructions library in AX series motion control CPUs.

Delta's motion instructions DL_MotionControl and DL_MotionControlLight are derived from Codesys Softmotion. They provide you 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	V4.10.0.0	V4.16.0.0
DL_MotionControl V1.1 and earlier versions	○		
DL_MotionControl V1.2 and later versions		○	
DL_MotionControl V1.4 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 later versions		○

Delta Motion Library	SM3_Basic V4.6.1.0	SM3_Basic V4.10.0.0	SM3_Basic V4.16.0.0
DL_PLCopenPart3		○	

Delta Motion Library	SML_Basic V4.5.1.0	SML_Basic V4.10.0.0
DL_ServoPress_AX		

Delta Motion Library	SM3_Basic V4.6.1.0	SM3_Basic V4.10.0.0	SM3_Basic V4.16.0.0
DL_PLCopenPart3		○	

1.4 Device Categories

- Logical controllers: Not support motion instructions.
- Basic motion controllers: Only support the DL_MotionControlLight library instructions.
- Motion controllers: Support the SM3_Basic, DL_MotionControl, DL_MotionControlLight library instructions.

AX-series	Logical controller	Basic motion controller	Motion controller
AX-300NA	○		
AX-324NA	○		
AX-304EL		○	
AX-308EA			○
AX-316EA			○
AX-364EL			○
AX-332EP			○
AX-C12EP			○
AX-816EP			○
AX-832EP			○
AX-864EP			○

1.5 PDO Mapping

This section describes the PDOs that need to be configured for different instructions.

- PDOs that must be configured
  - 0x6040
  - 0x6041
  - 0x6064
  - 0x607A
- Instructions
- PDO inputs

Instruction Name	0x6061	0x6064	0x606C	0x6077	0x60B9	0x60BA
MC_TouchProbe	○			○	○	○
DMC_TorqueControl	○			○		
DMC_VelocityControl	○		○			
DMC_MoveFeed					○	○
MC_TorqueControl_DML		○		○		

- PDO outputs

Instruction Name	0x6060	0x6071	0x60B8	0x60FF
MC_TouchProbe			○	
DMC_TorqueControl	○	○		
DMC_VelocityControl	○			○

Chapter 2 Motion Control Instruction

2.1 SM3 Basic

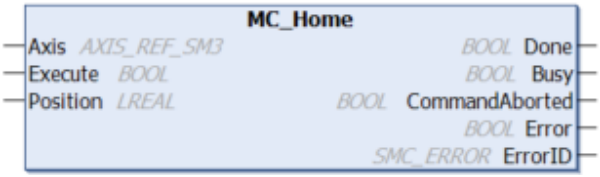
This section describes the usage and examples of function blocks in Delta's instruction library in AX series motion control CPU.

2.1.1 Motion Control Instructions

Motion control instructions are generally used to control motors to perform specific motion 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 Operation Manual*.

2.1.1.1 MC_Home

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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Position	Absolute position (Unit: user unit)	LREAL	Negative, positive, or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.

- **Outputs**

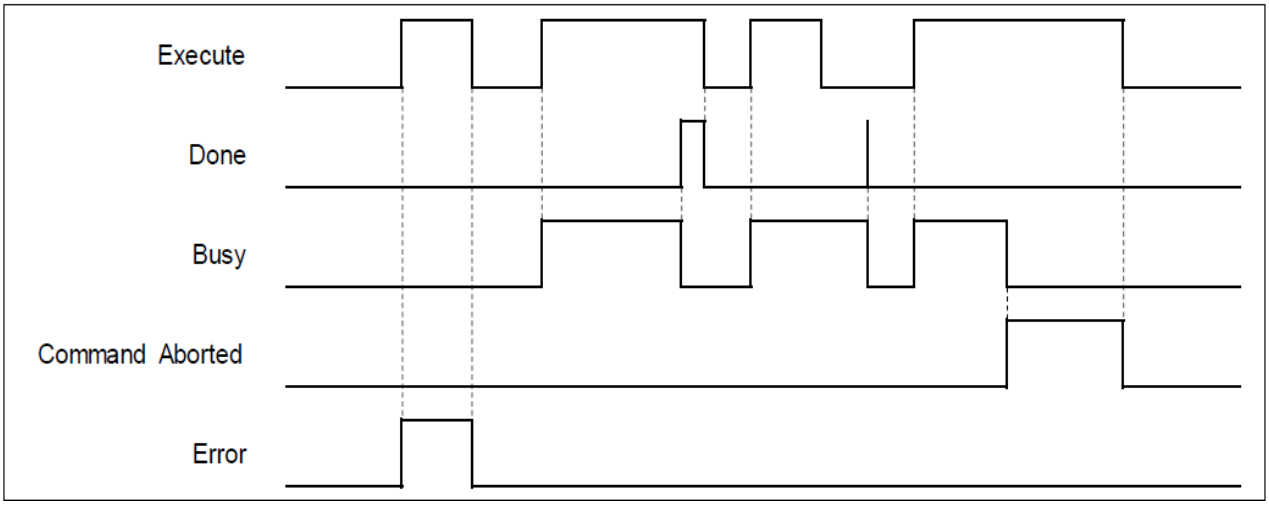
Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when homing is completed and the status is standstill	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The homing is complete.	• *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 then turn to FALSE immediately.
Busy	The instruction is running.	• *Done* turns to TRUE. • *Error* turns to TRUE. • *CommandAborted* turns to TRUE.
CommandAborted	The instruction is interrupted by other instructions or MC_Stop.	• *Execute* turns to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, *Done* will be TRUE for only one scan cycle and then turn to FALSE immediately.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	The error is cleared.
ErrorID		

- **Timing Diagram of Output Parameter Changes**



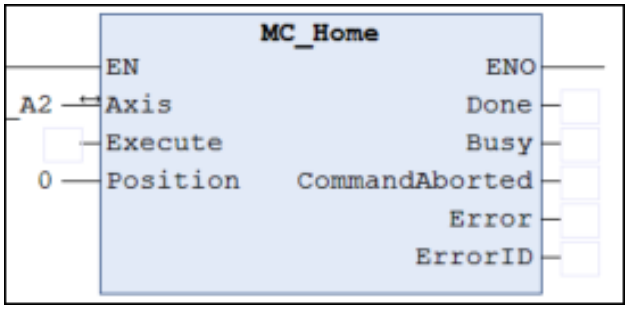
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.

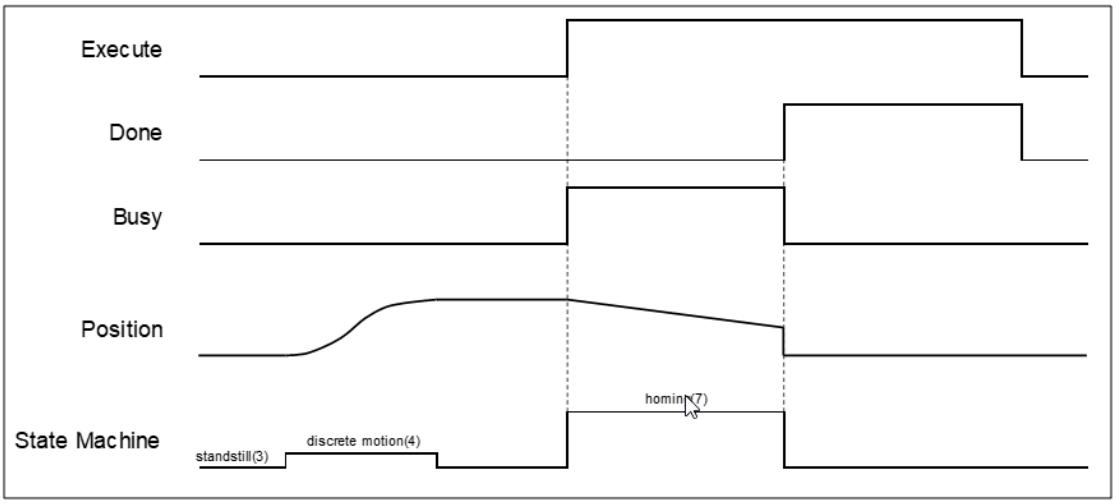
***Note**: AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.

- **Function**
  - This function block must be executed when the status is Standstill. When the instruction is running, the state is Homing, and the function block cannot be executed when the axis is in other states.
  - Position is used to specify the absolute position for homing.
  - If the MC_Home command is interrupted by MC_Stop and xWaitForHaltWhenStopInterruptsHome is TRUE, MC_Stop has to wait till the driver velocity reaches zero before setting it to *Done*. Instead, if *xWaitForHaltWhenStopInterruptsHome* is FALSE, *Done* will turn to TRUE once MC_Stop interrupts MC_Home.
  - When MC_Home is running, if the power is off, the function block status will change to *Error*. For SoftMotion V4.10.0.0 and later, when MC_Home is running, if the power is off, the \function block status will change to Abort.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  This example explains how to perform homing by using MC_Home and setting homing mode on the parameter configuration page with DIAdesigner-AX.



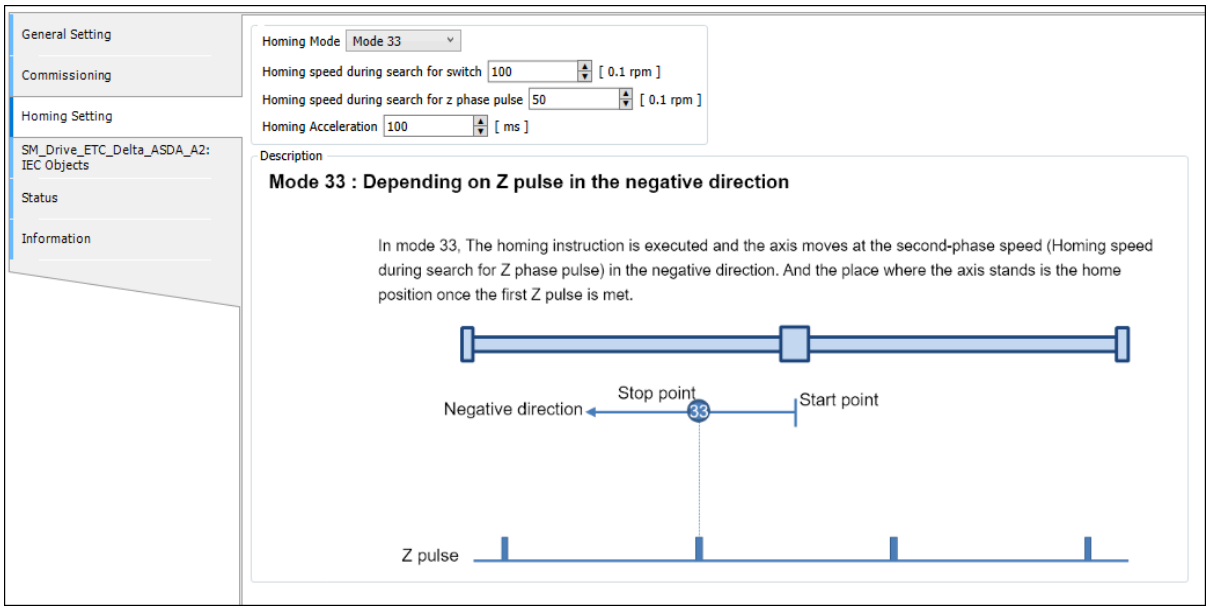
  - Timing Diagram



  - Use MC_Home with the following parameters:

Homing mode	Speed during search for switch	Speed during search for zero	Homing acc eleration
33	100	50	100

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

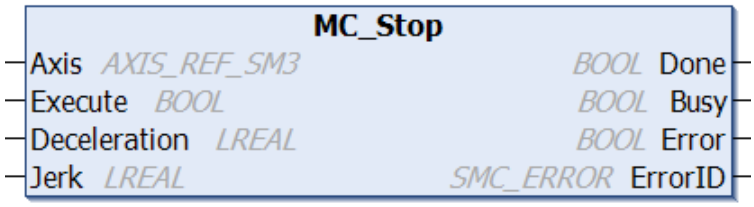


- After MC_Home is run, the axis will move in the negative direction till finds the Z phase pulse. And the place where the axis stands is the home position once the first Z pulse is met.

- **Supported Devices**
  - AX-series motion controller

2.1.1.2 MC_Stop

MC_Stop decelerates an axis to 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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Deceleration	Deceleration rate (Unit: user unit/s^2)	LREAL	Positive number or 0 (0)	When *Execute* turns to TRUE, update the deceleration rate.
Jerk	Jerk value (Unit: user unit/s^3)	LREAL	Positive number or 0 (0)	When *Execute* turns to TRUE, update the jerk value.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when the velocity reaches zero	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

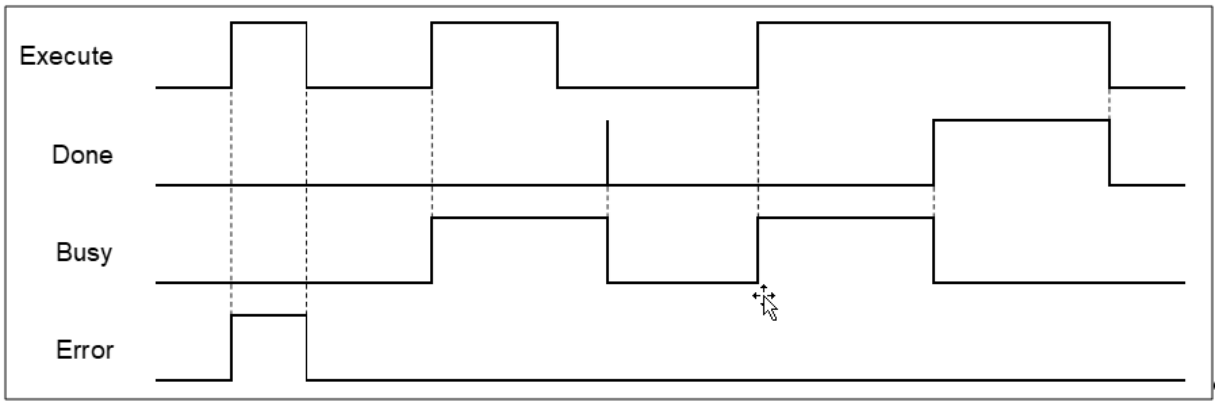
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Done	The axis decelerates to a stop and velocity reaches to zero.	• *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 then turn to FALSE immediately.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Busy	*Execute* changes to TRUE.	• *Done* turns to TRUE. • *Error* turns to TRUE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



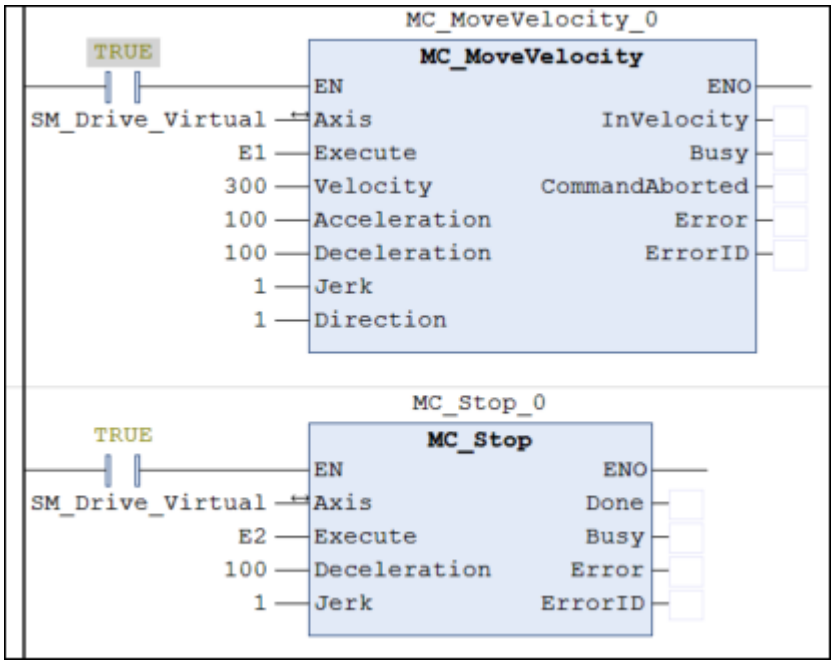
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.

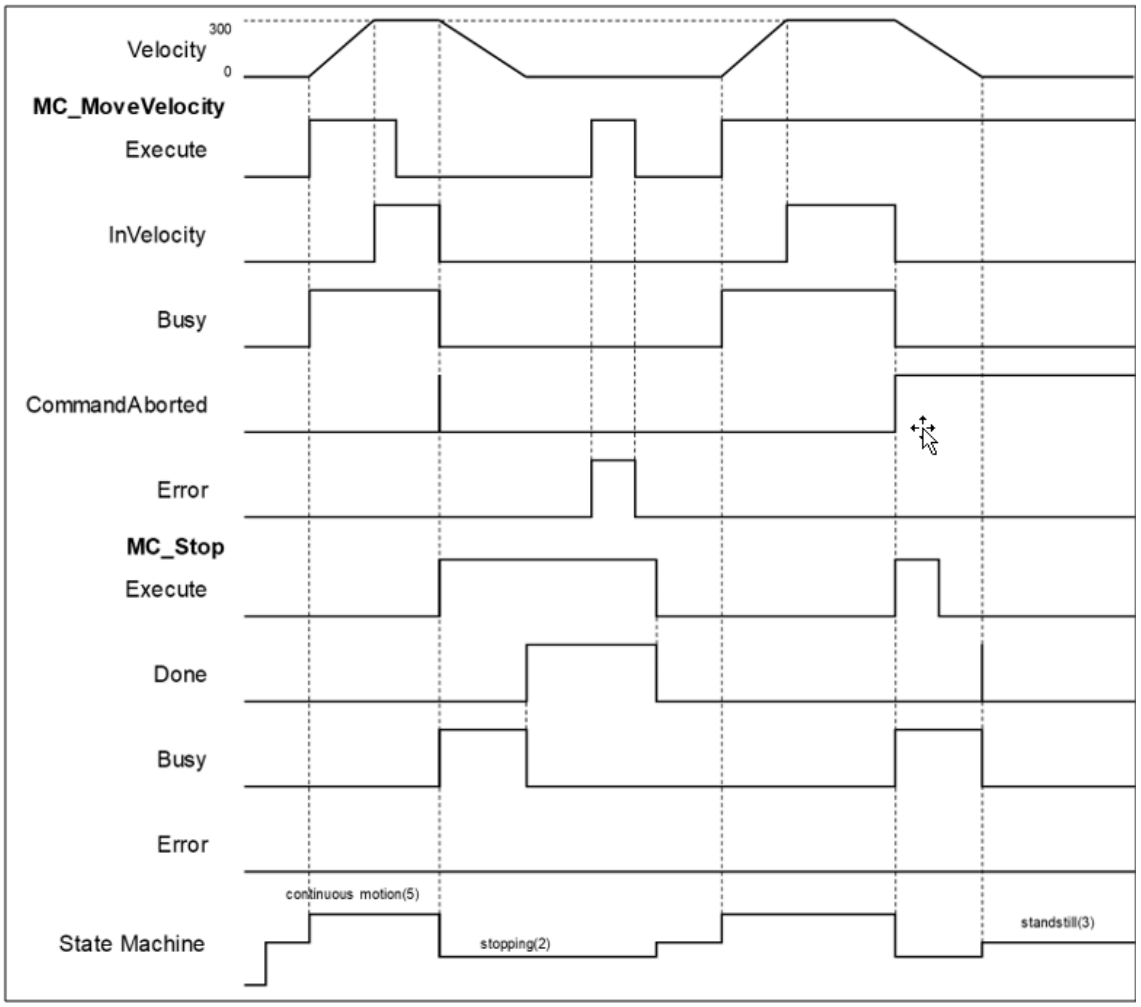
***Note**: AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.

- **Function**
  - Use MC-Stop to stop a running axis. Specify the deceleration rate when decelerating the moving axis to a stop. The State Machine will be in the stopping state.
  - When MC_Power is FALSE during deceleration, the motor is in the Free Run state.
  - The Done output is set to TRUE when the axis velocity reaches to zero. At the same time, the input *Execute* changes to FALSE, while the State Machine's state will change from stopping to standstill.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  This example explains how to execute MC_Stop after executing MC_MoveVelocity and the motion trace of MC_Stop.



- Timing Diagram



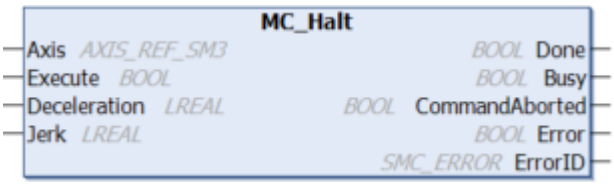
- When *Execute* of MC_Stop changes to TRUE, it triggers *CommandAboted* of MC_MoveVelocity to be TRUE, and the motion controller starts decelerating the axis to a stop. The Axis state changes to "Stopping".

- When the axis velocity reaches zero, the *Done* output will change to TRUE. *Execute* is still TRUE so the axis state remains in "Stopping". After the axis is stopped, *Execute* turns to FALSE, and the axis state will change to "Standstill".
- If MC_MoveVelocity runs again while the axis state is "Stopping", an error will be reported. (Error Code: SMC_AXIS_NOT_READY_FOR_MOTION).

- **Supported Devices**
  - AX-series motion controller

2.1.1.3 MC_Halt

MC_Halt stops the axis motion in a controllable 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	Runs the instruction 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 value will be updated.
Jerk	Jerk value (Unit: user unit/s3)	LREAL	Positive number or 0 (0)	When *Execute* turns to TRUE, the value will be updated.

- **Outputs**

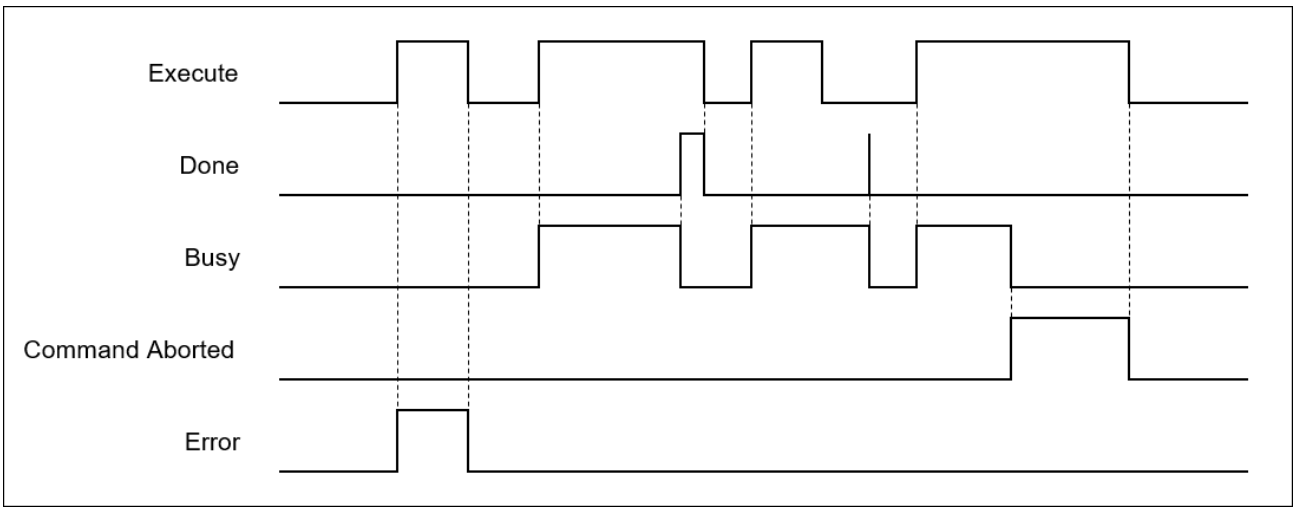
Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when the velocity reaches zero	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The axis decelerates to a stop and the velocity reaches zero.	• *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 turn to FALSE.
Busy	*Execute* turns to TRUE.	• When *Done* turns to TRUE. • When *Error* turns to TRUE. • When *CommandAborted* turns to TRUE.
CommandAborted	This instruction is aborted by other function blocks.	• *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 turn to FALSE.
Error ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



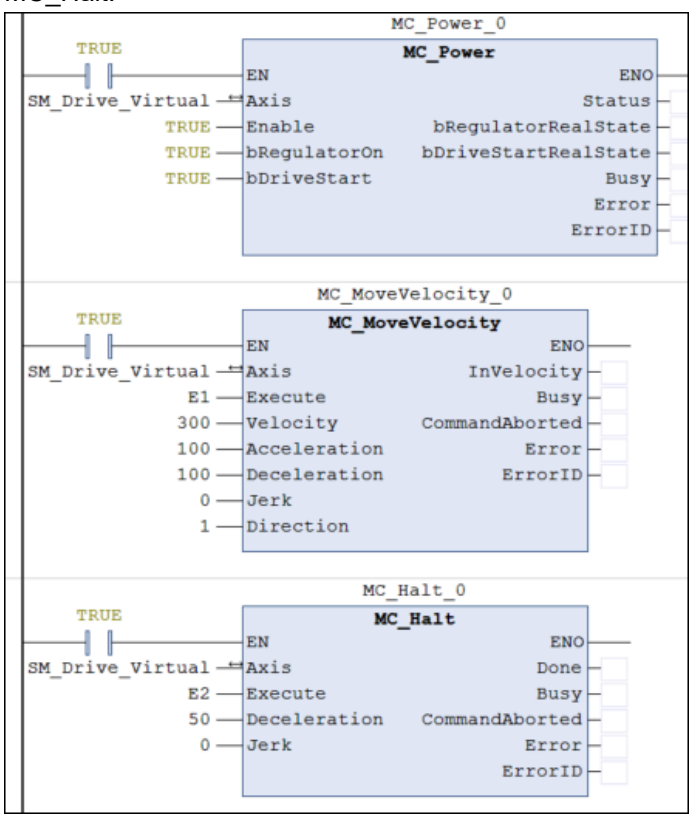
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.

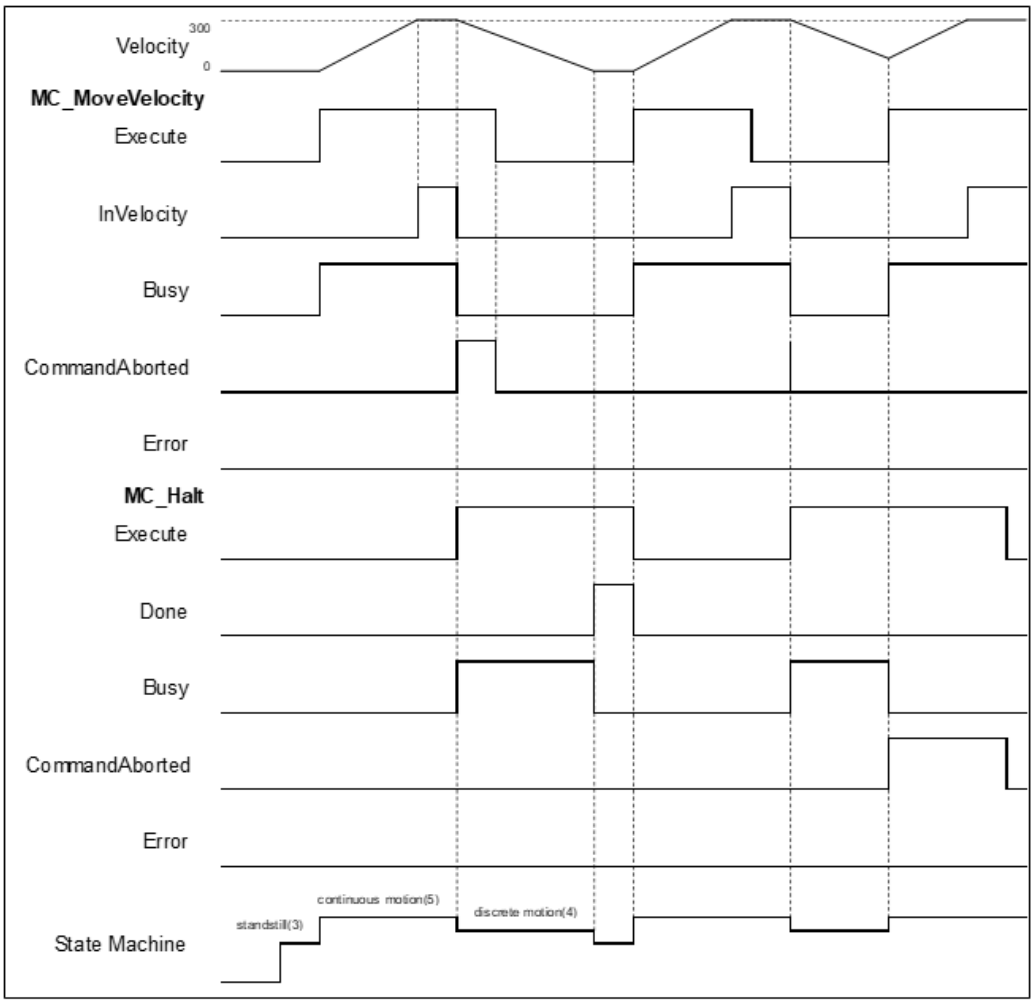
***Note**: AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.

- **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 while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  This example explains how to run MC_Halt after running MC_MoveVelocity and the motion trace of MC-Halt.
  - If no other motion instructions are run when the MC_Halt is stopping MC_MoveVelocity, the axis status will enter "Standstill" after the deceleration is completed.
  - If MC_MoveVelocity runs again during the deceleration, it will abort MC_Halt immediately and accelerate again without entering the "Standstill" state. This re-running behavior is allowed for MC_Halt.



  - Timing Diagram

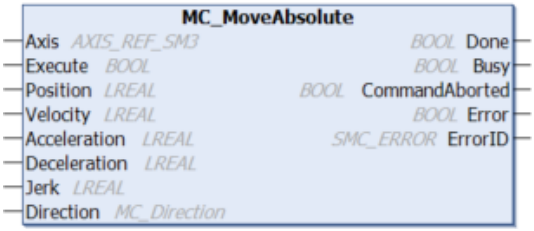


- When *Execute* of MC_Halt changes to TRUE, it triggers *CommandAborted* of MC_MoveVelocity at the same time, and the motion controller starts to decelerate the axis to a stop. The Axis state changes to the "DiscreteMotion".
- When the axis reaches zero velocity, *Done* will change to TRUE. The axis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

- **Supported Devices**
  - AX-series motion controller

2.1.1.4 MC_MoveAbsolute

MC_MoveAbsolute controls the axis to move to the specified absolute target position according to the motion mode.

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

- **Inputs**

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

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
BufferMode	Buffering behavior mode for this instruction.	MC_BUFFER_MODE*2	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	*Execute* turns to TRUE and *Busy* is FALSE.

***Note**:

1. MC_Direction: Enumeration (Enum)
2. MC_BUFFER_MODE: 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 running	BOOL	TRUE/FALSE (FALSE)
Active	TRUE when the axis is moving	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

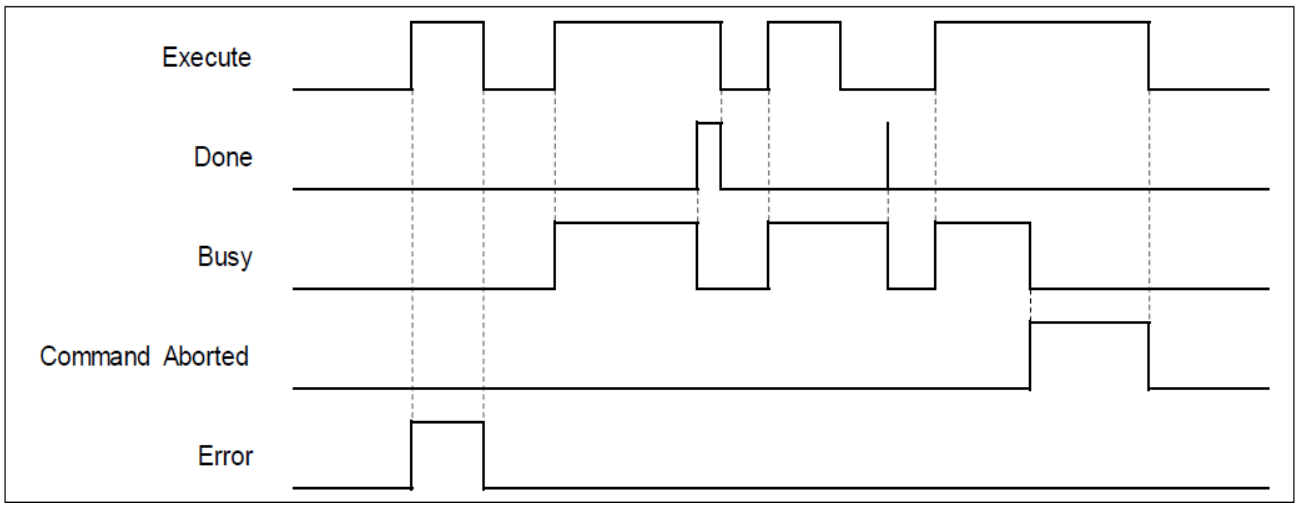
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Done	The absolute positioning is completed.	• *Execute* turns to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, *Done* will be TRUE for only one scan cycle and immediately turn to FALSE.
Busy	*Execute* turns to TRUE.	• *Done* turns to TRUE. • *Error* turns to TRUE. • *CommandAborted* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Active	*Execute* turns to TRUE.	• *Done* turns to TRUE. • *Error* turns to TRUE. • *CommandAborted* turns to TRUE.
CommandAborted	• This instruction is interrupted by another function block. • This instruction is aborted because of the running of MC_Stop instruction.	• *Execute* turns to FALSE • If *Execute* is FALSE and *CommandAborted* turns to TRUE, *Done* will be TRUE for only one scan cycle and immediately turn to FALSE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**

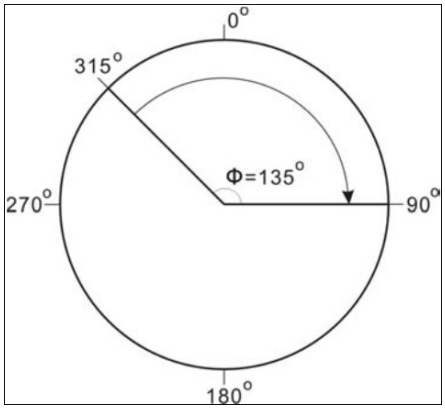
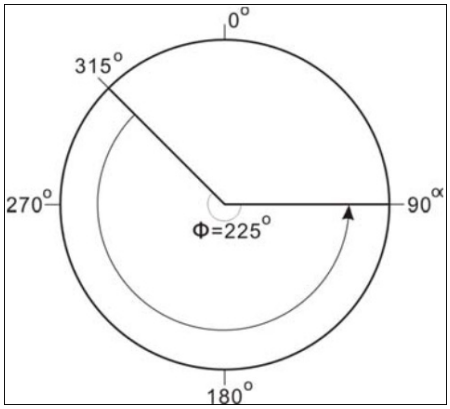


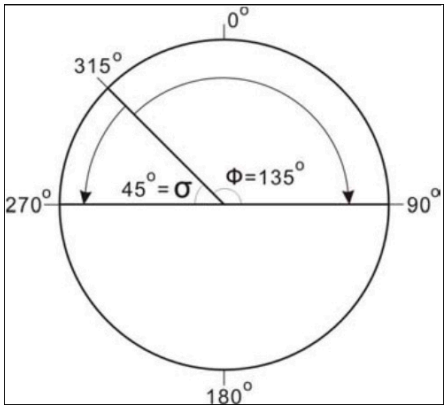
- **Inputs/Outputs**

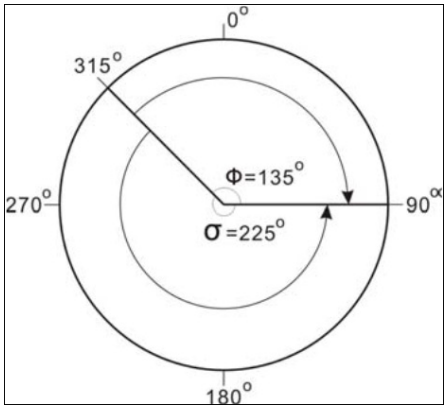
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.

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

- **Function**
  - **Direction**
    - Direction is used to define the rotation of the servo axis. The shortest, current, and fastest are only available for the rotary axis but not for linear axes.
    - When the direction value is different, the rotary axis's motion direction and travel distance will differ. If 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°
	

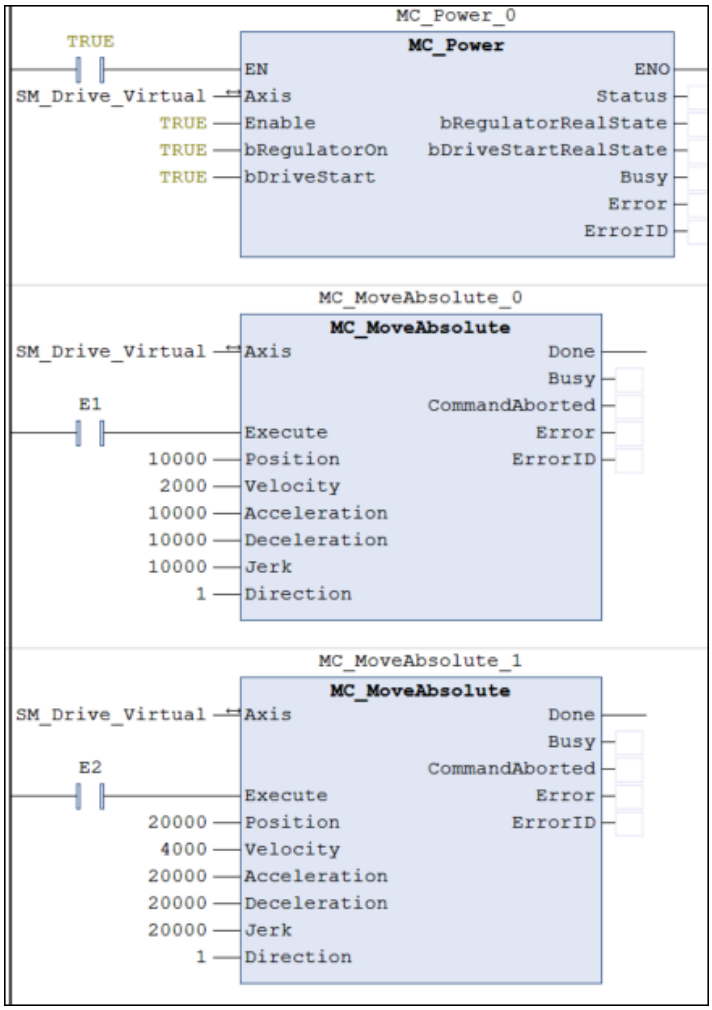
Direction: 0 (shortest), 3(fastest) Current position: 315° Target position: 90° Movement angle: 135°	Direction: 0 (shortest), 3 (fastest) Current position: 315° Target position: 270° Movement angle: 45°
	

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°
	

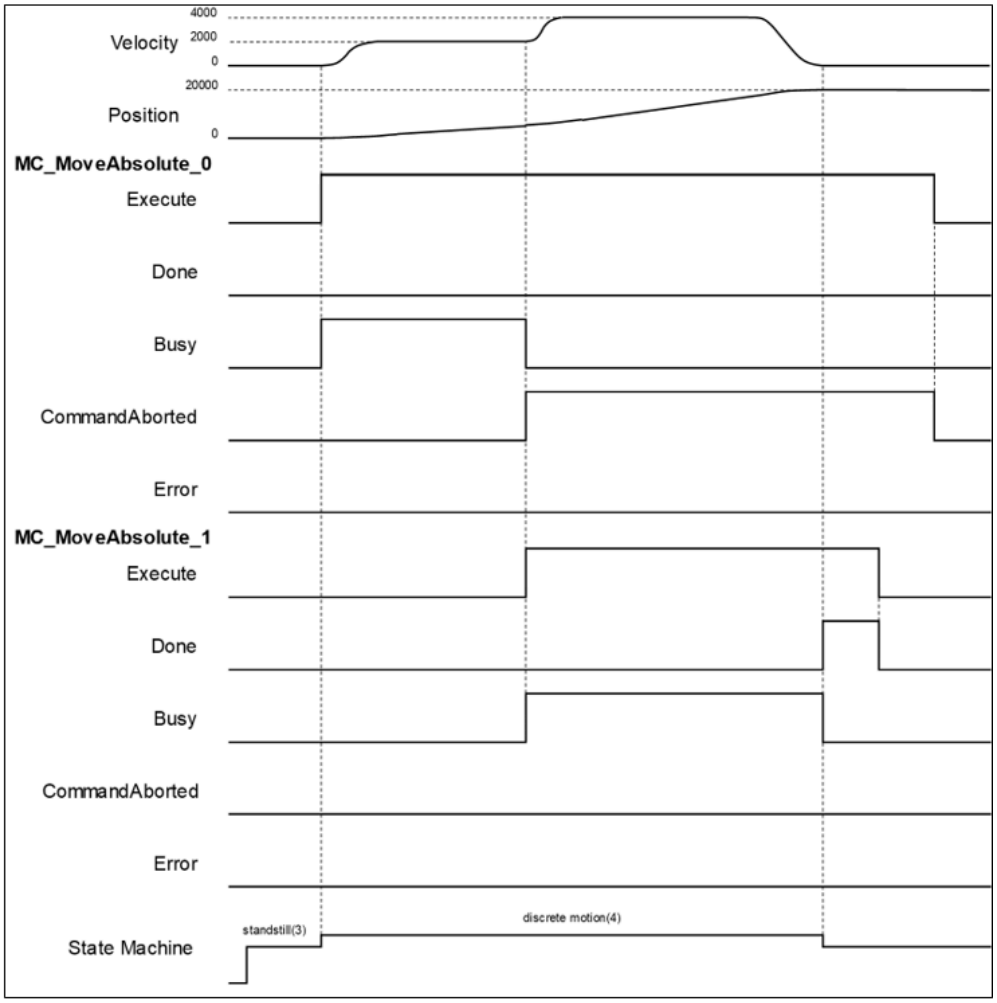
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

The example describes the behavior of 2 MC_MoveAbsolute instructions that are connected.



- Timing Diagram



  - When *Execute* of MC_MoveAbsolute_0 is TRUE, the axis will move towards the absolute target position. Once *Execute* of MC_MoveAbsolute_1 turns to TRUE, MC_MoveAbsolute_0 will be aborted. CommandAborted turns TRUE. The final position will be 20,000.
  - When MC_MoveAbsolute_1 is executed, the axis will move towards the absolute target position according to the MC_MoveAbsolute_1 parameter.
  - When the axis reaches the absolute position 20,000 set by MC_MoveAbsolute_1, *Done* of MC_MoveAbsolute_1 turns to TRUE and *Busy* turns to FALSE.
  - If *Execute* of MC_MoveAbsolute_1 switches to FALSE, the *Done* output will also change to FALSE.

- **Supported Devices**
  - AX-series motion controller

2.1.1.5 MC_MoveRelative

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

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

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	Runs the instruction 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)	*Execute* turns to TRUE and *Busy* is FALSE.
Velocity	Target velocity (Unit: user unit/s)	LREAL	Positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Acceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Deceleration	Deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Jerk	Jerk value (Unit: user unit/s^3)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
BufferMode	Buffering mode for this instruction	MC_BUFFER_MODE*2	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	*Execute* turns to TRUE and *Busy* is FALSE.

***Note**: MC_BUFFER_MODE: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when reaching the target distance	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Active	TRUE when the axis is moving	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

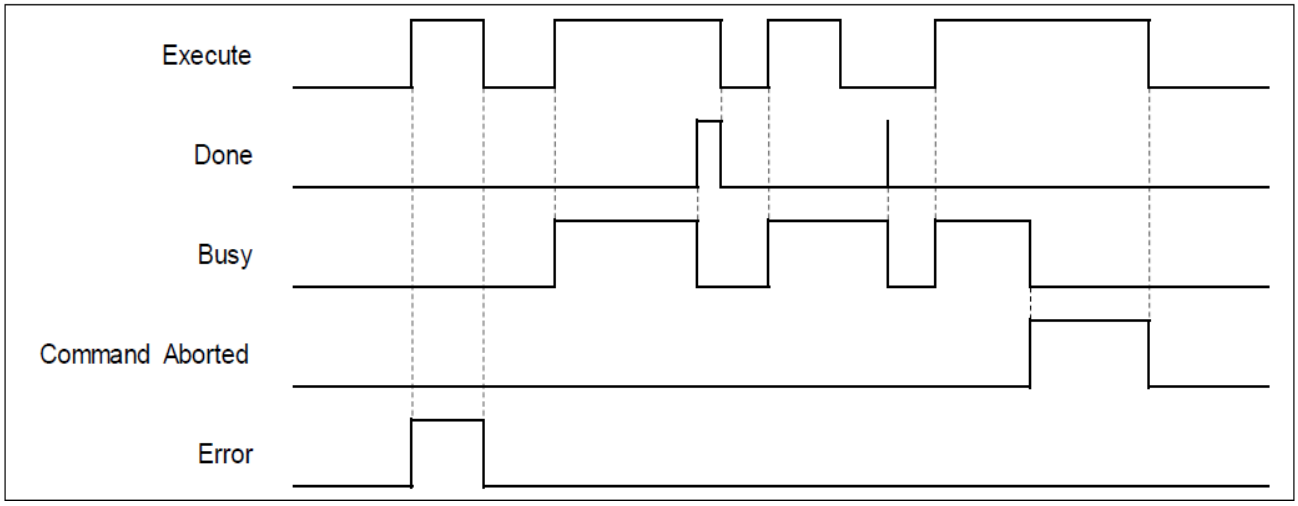
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Done	The relative positioning is complete.	• *Execute* turns from TRUE to FALSE. • If *Execute* is FALSE and *Done* turns to TRUE, *Done* keeps TRUE for only one scan cycle and then turn to FALSE.
Busy	.*Execute* changes to TRUE	• *Done* turns to TRUE. • *Error* turns to TRUE. • *CommandAborted* turns to TRUE.
Active	*Execute* turns to TRUE.	• *Done* turns to TRUE. • *Error* turns to TRUE. • *CommandAborted* turns to TRUE.
CommandAborted	• This instruction is interrupted by another function block. • This instruction is interrupted because of the running of MC_Stop instruction.	• *Execute* tuens to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, it keeps TRUE for only one scan cycle and then turns to FALSE.
Error		*Execute* turns from TRUE to FALSE. (Error Code is cleared.)

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*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 the function block.

- **Function**

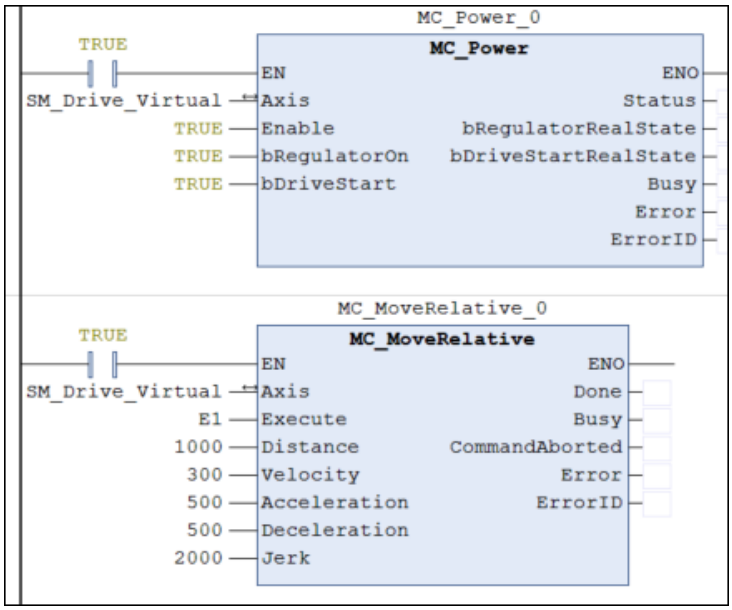
  The instruction performs relative positioning with specified target velocity (*Velocity*), acceleration rate (*Acceleration*), deceleration rate (*Deceleration*) and *Jerk* value (*Jerk*) when *Execute* turns to TRUE.

- **Troubleshooting**

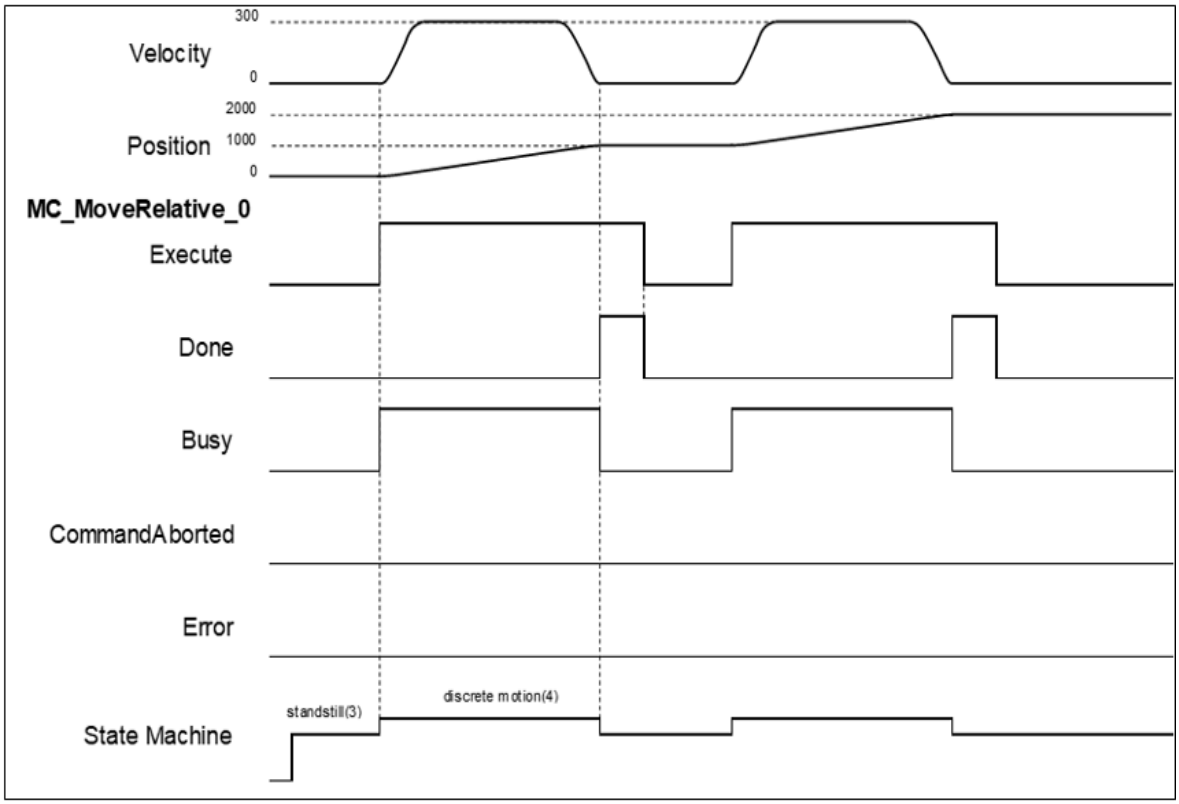
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

  This example describes the behavior of the MC_MoveRelative instruction.



- Timing Diagram

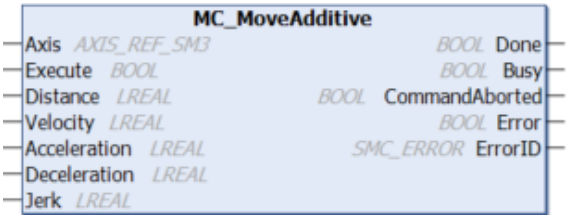


  - When *Execute* changes to TRUE, MC_MoveRelative drives the axis to the target position. During motion, *Busy* is TRUE in the state of Discrete motion.
  - When the axis moved the specified relative distance (1,000), *Done* changes to TRUE, and *Busy* changes to FALSE.
  - When *Execute* changes to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

- **Supported Devices**
  - AX-series motion controller

2.1.1.6 MC_MoveAdditive

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	Runs the instruction 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)	*Execute* turns to TRUE and *Busy* is FALSE.
Velocity	Target velocity (Unit: user unit/s)	LREAL	Positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Acceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Deceleration	Deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Jerk	Jerk value (Unit: user unit/s^3)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.

- **Outputs**

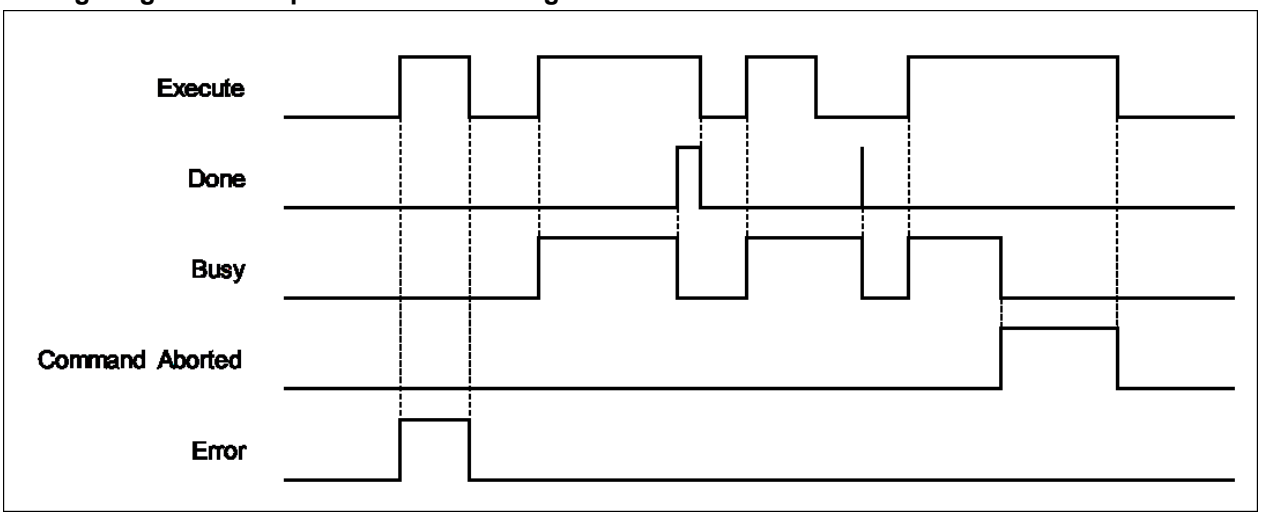
Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when additive distance is complete	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The additive positioning is completed.	• *Execute* changes to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, *Done* will be TRUE for only one scan cycle and immediately turn to FALSE.
Busy	*Execute* changes to TRUE.	• *Done* changes to TRUE. • *Error* changes to TRUE. • *CommandAborted* changes to TRUE.
CommandAborted	This instruction is aborted by MC_Stop instruction.	• *Execute* changes to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, *Done* will be TRUE for only one scan cycle and immediately turn to FALSE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error code is cleared)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



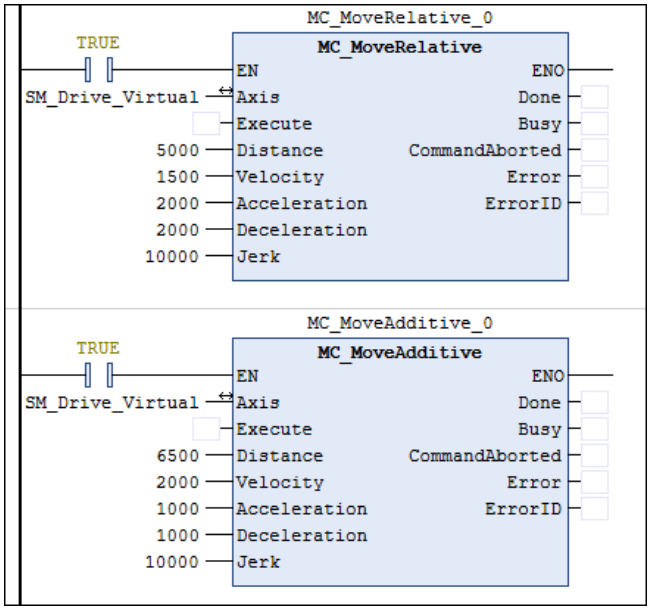
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.

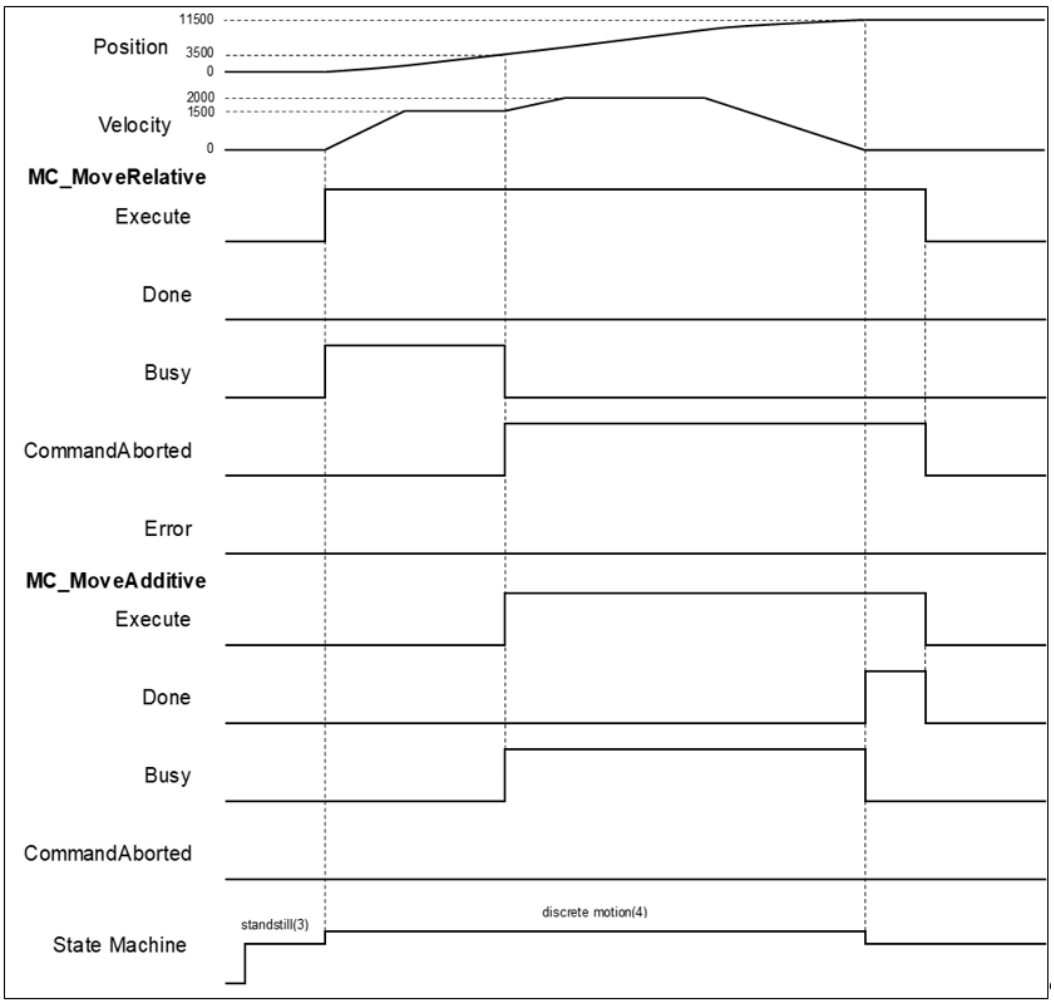
***Note**: AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.

- **Function**
  - MC_MoveAdditive drives the axis to move an additional distance based on parameters you set.
  - When MC_MoveAdditive runs alone, the behavior will be identical to MC_MoveRelative.
  - If the MC_MoveAdditive instruction is ongoing, triggering this instruction again will additionally increase the movement distance.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

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



- Timing Diagram

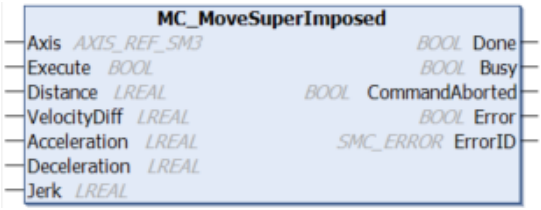


  - When *Execute* turns to TRUE, MC_MoveRelative drives the axis to the target position. After *Execute* turns to TRUE at the position of 3,500, the MC_MoveRelative instruction will be aborted, and *CommandAborted* turns to TRUE. The axis remains in the "Discrete motion" state.
  - The MC_MoveAdditive instruction is run and adds a relative distance of 6,500 to the previous target position of 5,000, and the new target position is 11,500.
  - When the axis reaches 13,500, *Done* changes to TRUE.

- **Supported Devices**
  - AX-series motion controller

2.1.1.7 MC_MoveSuperImposed

MC_MoveSuperimposed controls the axis to move a relative superimposed distance at a specified behavior without changing the target movement of the original motion instruction.

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	Runs the instruction 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)	*Execute* turns to TRUE and *Busy* is FALSE.
VelocityDiff	Additional target velocity (Unit: user unit/s)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Acceleration	Additional acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Deceleration	Additional deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Jerk	Additional jerk value (Unit: user unit/s^3)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when the superimposed motion is completed	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)

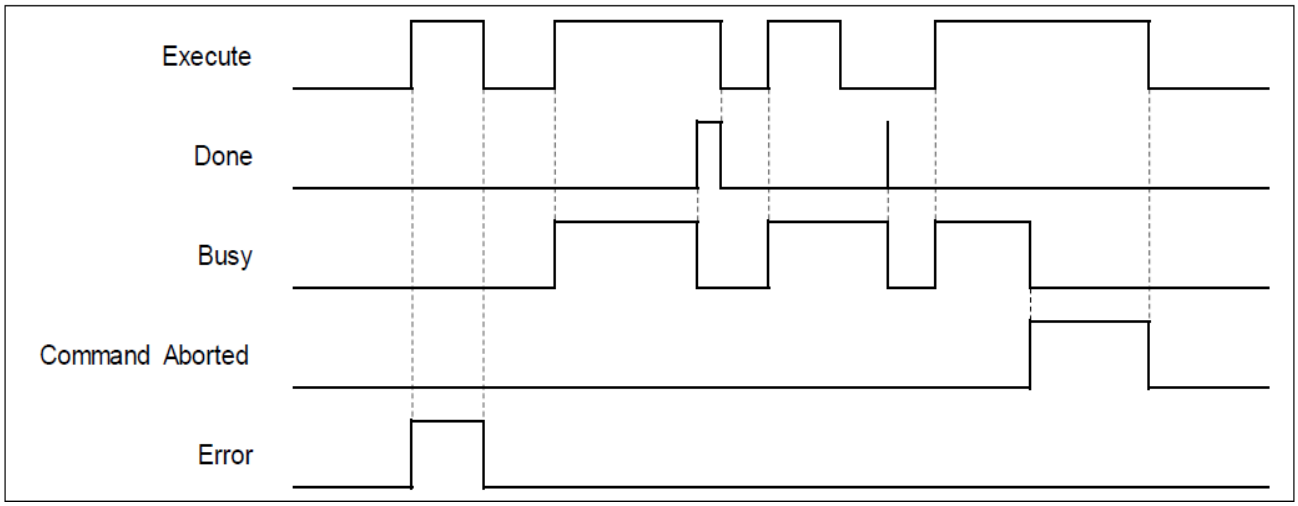
Name	Function	Data Type	Output Range (Default Value)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

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

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Done	The superimposed distance is completed.	• *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 turn to FALSE.
Busy	*Execute* changes to TRUE.	• *Done* changes to TRUE. • *Error* changes to TRUE. • *CommandAborted* turns to TRUE
CommandAborted	• One instruction is interrupted by another instruction with the Buffer Mode set to Aborting • This instruction is aborted because of the running of MC_Stop instruction	• *Execute* changes to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, it will be TRUE for only one period and immediately turn to FALSE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

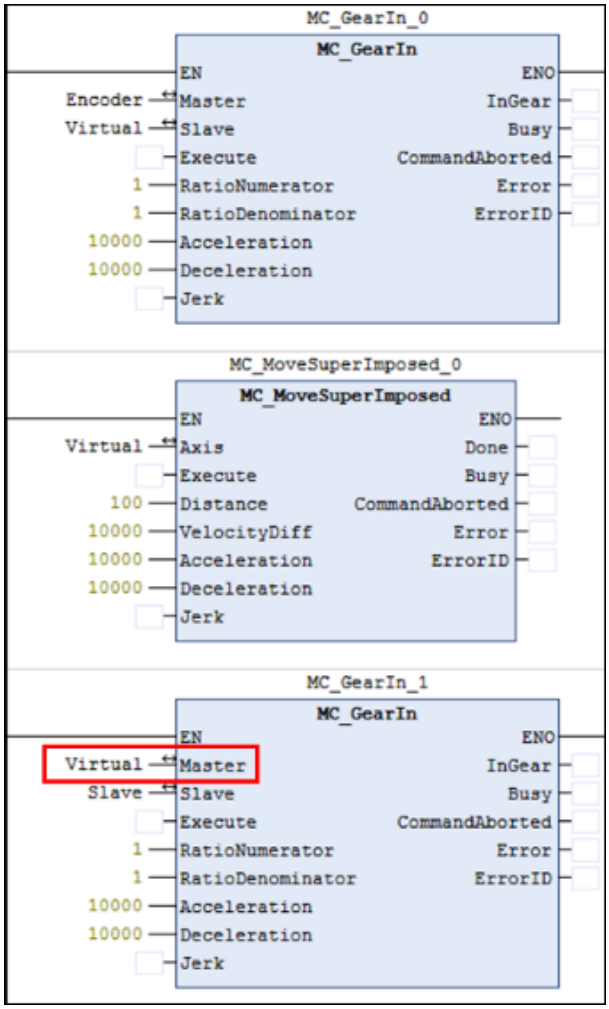
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.

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

- **Function**

  - The input values of *VelocityDiff*, *Distance*, *Acceleration*, *Deceleration* and *Jerk* are superimposed on the on-going motion instruction.
  - If running the MC_MoveSuperImposed block in Standstill state, the function will be the same as that of MC_MoveRelative.
  - MC_MoveSuperImposed can be interrupted by other function blocks.
  - An error will occur when MC_MoveSuperImposed is repeatedly run on the same axis.
  - If changing the input values when running MC_MoveSuperImposed or Re-run 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.
  - When MC_MoveSuperimposed is superimposed on another motion instruction, executing another motion instruction at this time will break the two instructions together.

- Depending on where this function block is placed in the program, the results of the synchronous motion are different. For more information, see *AX-3 Series Operation Manual* section 7.7.2.5 Synchronous Control.

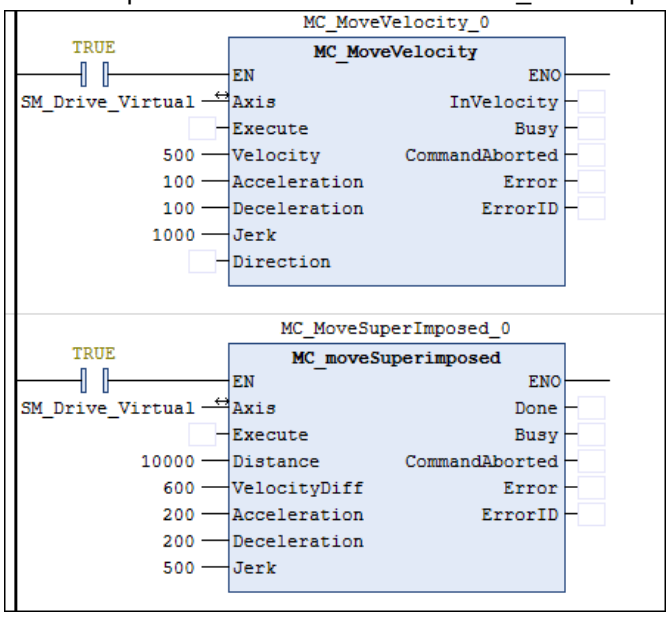


- **Troubleshooting**

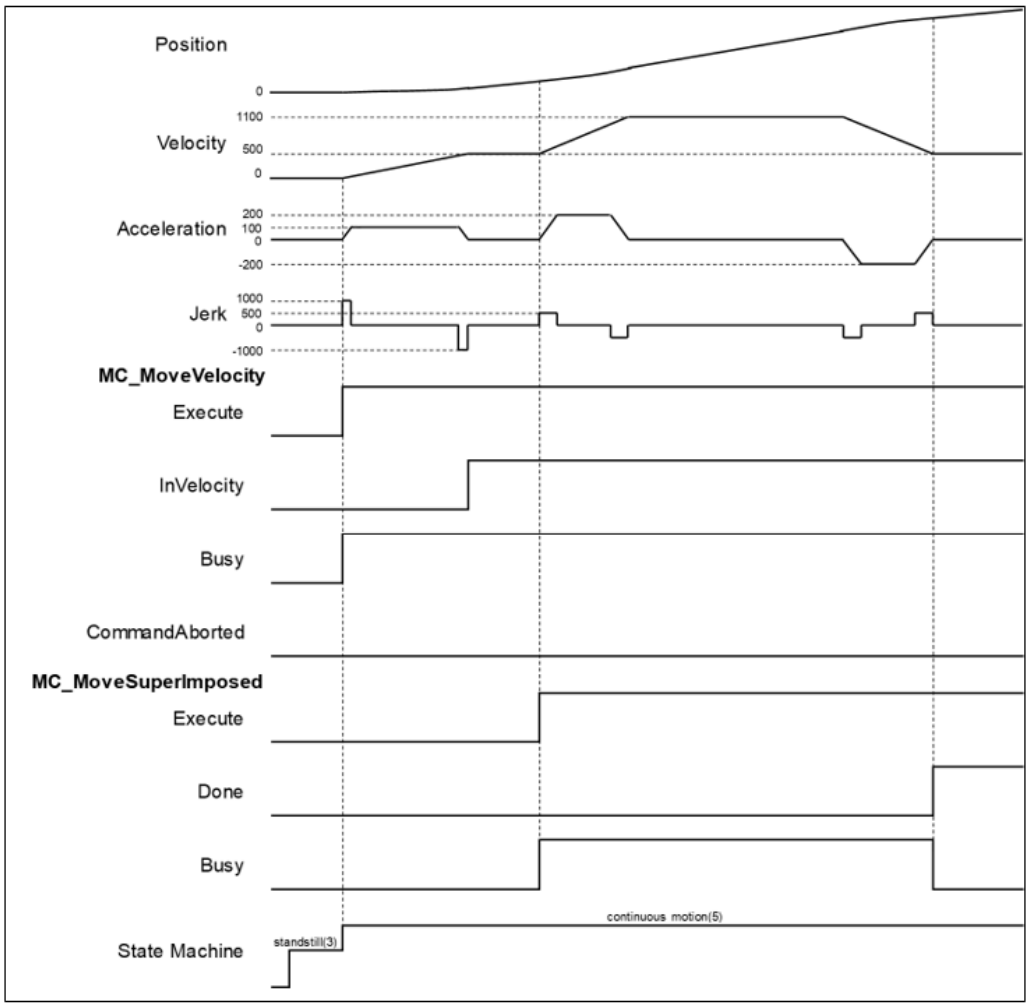
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

This example describes the behavior of MC_MoveSuperImposed applied to MC_MoveVelocity.



- Timing Diagram

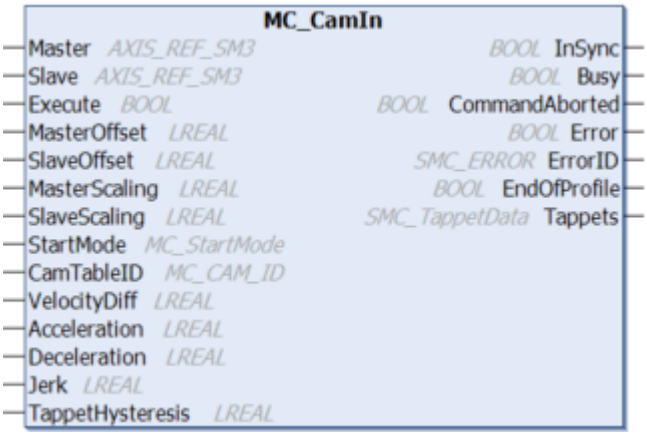


  - When *Execute* of MC_MoveVelocity changes to TRUE, the specific axis starts to move towards the target velocity (500) at the constant speed.
  - When *Execute* of MC_MoveSuperImposed changes to TRUE, the MC_MoveSuperImposed instruction starts and applies the additional values (velocity, distance, acceleration, 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 running MC_MoveSuperImposed is finished, *Done* will turn TRUE and MC_MoveVelocity will keep going.

- **Supported Devices**
  - AX-series motion controller

2.1.1.8 MC_CamIn

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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
MasterOffset	Master offset value (Unit: user unit)	LREAL	Negative, positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
SlaveOffset	Slave offset value (Unit: user unit)	LREAL	Negative, positive or 0 (0)	*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)	*Execute* turns to TRUE and *Busy* is FALSE.
SlaveScaling	Scales the slave axis up and down with the specified factor.	LREAL	Negative, positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
StartMode	The slave axis mode of MC_CamIn.	MC_*StartMode*	0: absolute 1: relative 2: ramp_in 3: ramp_in_pos 4: ramp_in_neg	*Execute* turns to TRUE and *Busy* is FALSE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
			(absolute)	
CamTableID	Cam table identifier, which is from the output of CamTableSelect	MC_CAM_ID	MC_CAM_ID*1	*Execute* turns to TRUE and *Busy* is FALSE.
VelocityDiff	Maximum velocity difference when running MC_CamIn*2 (Unit: user unit/s)	LREAL	Positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Acceleration	The acceleration rate when running MC_CamIn*2 (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Deceleration	The deceleration rate when running MC_CamIn*2 (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Jerk	The jerk value when running MC_CamIn*2(Unit: user unit/s^3)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
TappetHysteresis	The hysteresis rate of tappet	LREAL	Positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.

***Note**:

1. MC_CAM_ID (Struct): Cam table variables, from output of MC_CAMTableSelect, and input to MC_CamIn.
2. This setting is only effective if selecting ramp_in, ramp_in_pos, or ramp_in_neg in *StartMode*.

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	Master axis absolute mode	BOOL	TRUE/FALSE (TRUE)
SlaveAbsolute	Slave axis absolute mode	BOOL	TRUE/FALSE (TRUE)
StartMaster	The start position of the master axis in the cam table	LREAL	Negative, positive or 0 (0)
EndMaster	The end position of the master axis in the cam table	LREAL	Negative, positive or 0 (0)
StartSlave	The start position of the slave axis in the cam table	LREAL	Negative, positive or 0 (0)
EndSlave	The end position of the slave axis 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 master and slave cam operation is synchronized	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	Used with function block SMC_GetTappetValue.	SMC_TappetData*2	SMC_TappetData

***Note**:

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

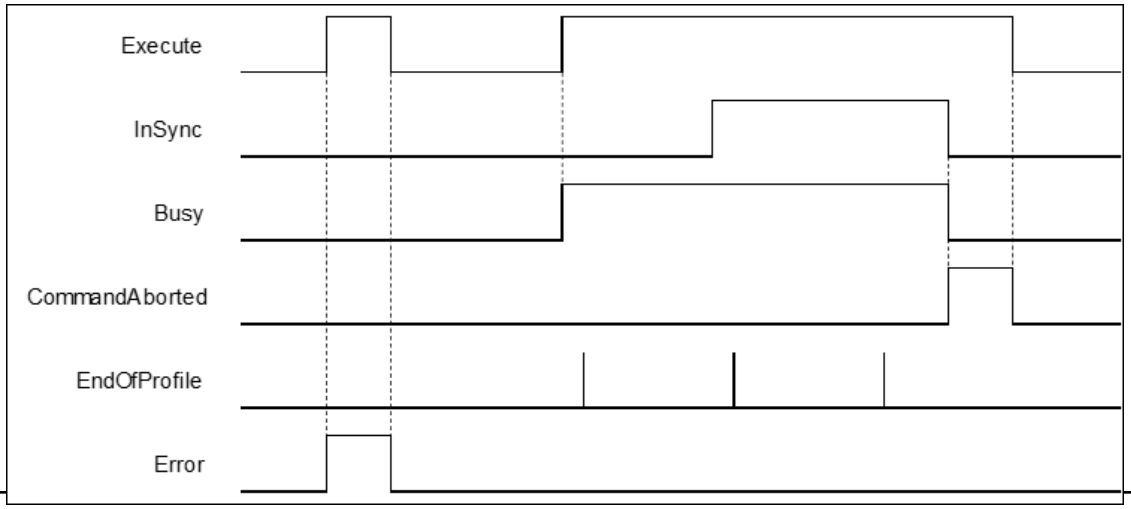
Name	Function	Data Type	Output Range (Default Value)
ctt	Tappet action active when axis passes tappets in the specified direction (positive or negative).	SMC_CAMTAPPETTYPE	0: TAPPET_pos (Pass in positive direction) 1: TAPPET_all (No specific direction) 2: TAPPET_neg (Pass in negative direction) (TAPPET_pos)
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 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)

Name	Function	Data Type	Output Range (Default Value)
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.	*Execute* is TRUE.
Busy	When the instruction is running	• *CommandAborted* is TRUE. • *Error* is TRUE.
CommandAborted	• When MC_CamOut is run • When the instruction is interrupted by another instruction • When the function block instruction is interrupted by MC_Stop	• *Execute* is FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, *CommandAborted* will be TRUE for only one period and immediately turn to FALSE.
Error	An error occurs during running the instruction or the input value of the instruction is incorrect.	When *Execute* is FALSE. (Error codes are cleared.)
ErrorID		
EndOfProfile	Cyclic end of the cam profile	• When the MC_CamTableSelect Periodic is 1 (cycle), turn to TRUE. After one cycle, turn to FALSE. • Turn to TRUE for only one cycle and immediately turn to FALSE if MC_CamTableSelect Periodic is 1 (cycle).

- **Timing Diagram of Output Parameter Changes**



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 cycle, then switch back to FALSE. Once the engaging 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, *Execute* and *CommandAborted* will change from TRUE to FALSE.

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	Specifies the master axis.	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.
Slave	Specifies the slave axis.	AXIS_REF_SM3*	AXIS_REF_SM3	*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 the function block.

- **Function**
  - Depending on where this function block is placed in the program, the results of the synchronous motion are different. For more information, see *AX-3 Series Operation Manual* section 7.7.2.5 Synchronous Control.
  - **Relationship between master axis position and slave axis position**
    - The cam mechanism set 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 and slave axis instead of the physical axis position. If the cam mechanism planned is seen as the function CAM, 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.

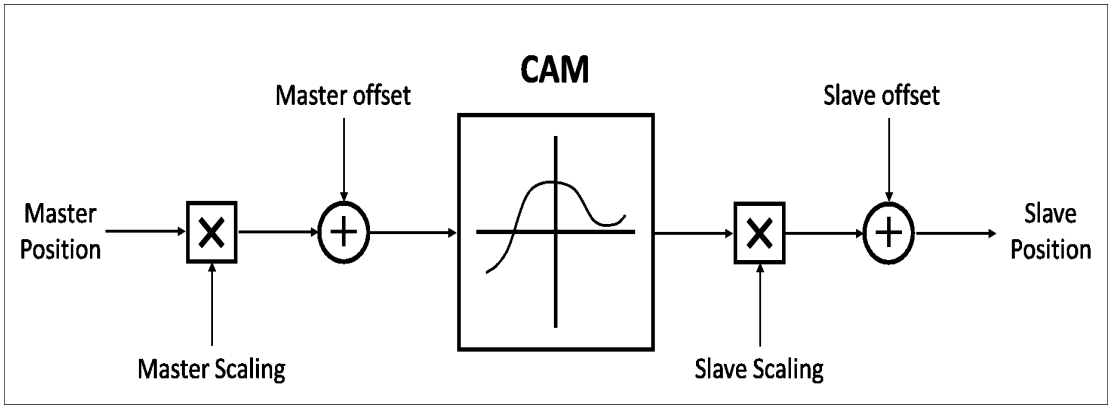
      y = CAM(x)

      x: The master axis cam phase

      y: The slave axis cam phase

    - The cam phase comes from the axis position and there is a conversion between them. The conversion is related to parameter MasterAbsolute, SlaveAbsolute, *MasterCompensation*, *SlaveCompensation*, *MasterScaling*, and *SlaveScaling*.

    - The slave axis follows the master axis to perform the synchronous cam motion by using the MC_CamIn instruction. In the synchronous cam motion, the relationship between the master axis position and slave axis position is based on the planned cam mechanism (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*

- When the master axis is in absolute mode, the master position is the remainder of the current master position divided by modulo. When the master axis is in relative mode, master position is the start point position (usually 0) of master axis of 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 runout will be occurred if the current position of slave axis and its start position from the Cam are not the same.

- Relative mode (*StartMode* = 1): The cam will change according to the current position of the slave axis. The slave axis position equals to the current position plus target position. If the slave axis position when engaging is different from the start position plus current position, a runout may occur.

- Ramp mode (*StartMode* = 2, 3, 4): Add a compensating motion curve to prevent the cam from runout when it starts engaging according to *VelocityDiff*, *Acceleration*, *Deceleration*, and *Jerk*.

  StartMode = 0; Absolute mode: The slave current position is not involved in the Cam calculation as the cam synchronization starts. However, a runout will occurr if the current position of slave axis and its start position from the Cam are not the same.

  StartMode = 1; Relative mode: The cam will change according to the current position of the slave axis. The slave axis position equals to the current postion plus target position. If the slave axis position when engaging is different from the start position plus current position, a runout may occur.

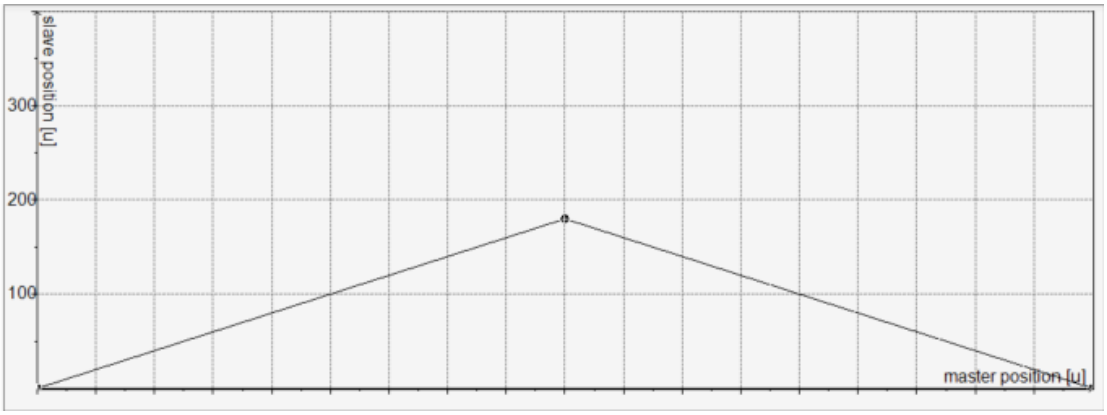
  StartMode = 2,3,4; Ramp mode: Add a compensating motion curve to prevent the cam from runout when it starts engaging according to *lrVelocityDiff*, *lrAcceleration*, *lrDeceleration*, and *lrJerk*.

MC_CamTableSelect.MasterAbsolute	Master axis
absolute	Absolute mode
relative	Relative mode

MC_CamIn.*StartMode*	MC_CamTableSelect.SlaveAbsolute	Slave axis mode
absolute	TRUE	Absolute
absolute	FALSE	Relative
relative	TRUE	Relative
relative	FALSE	Relative
ramp_in	TRUE	Ramp in Absolute
ramp_in	FALSE	Ramp in Relative
ramp_in_pos	TRUE	Ramp in Positive Absolute
ramp_in_pos	FALSE	Ramp in Positive Relative

MC_CamIn.*StartMode*	MC_CamTableSelect.SlaveAbsolute	Slave axis mode
ramp_in_neg	TRUE	Ramp in Negative Absolute
ramp_in_neg	FALSE	Ramp in Negative Relative

- Cam table



1. Absolute mode on master axis (*MasterAbsolute* = TRUE)

   Absolute mode on slave axis (*SlaveAbsolute* = TRUE)

   Note: Cam master-slave shaft meshing position: Master axis Position = Current Master Axis Position / Final Master Position in Cam Table

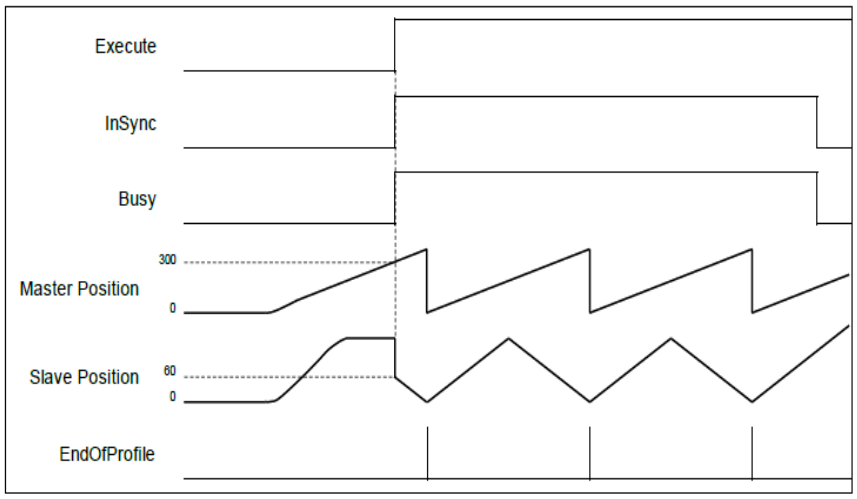
   Slave Axis Position = Final Slave Position in Cam Table* n.

   (n = Current Master Postion / Master Position in Cam Table)

   For example: Cam table, the master axis range is 360; the slave axis range is 200. The master axis start position is 800, which is converted into 360 * 2 + 80, and the slave axis start position is 200 * 2 + the cam position corresponding of the master axis position.

   a. Absolute mode (*StartMode* = 0)

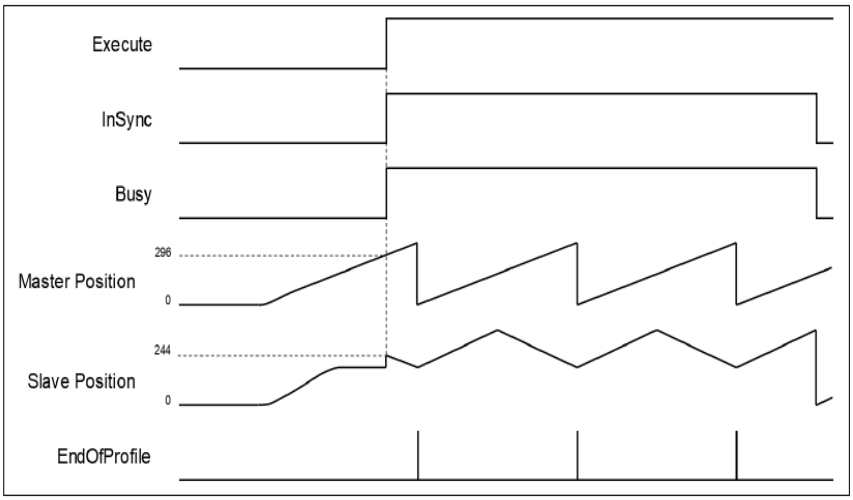
      Cam master and slave axis engaging position: for master axis, is the current position. For the slave axis, follows the cam table.



   b. Absolute mode (*StartMode* = 1)

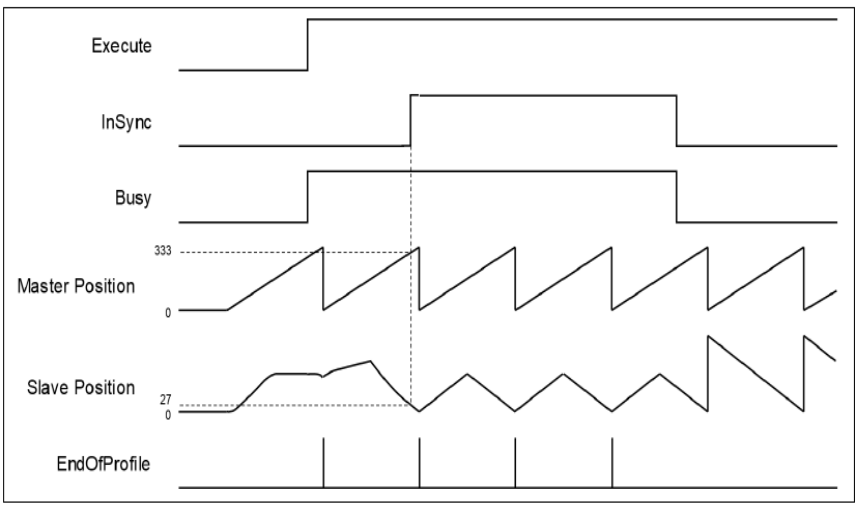
      Cam master and slave axis engaging position: for master axis, is the current position. For the slave axis, follows the cam table plus slave current position (180+64 = 244). In addition, a jump will occur if the

start point of master axis is not same as the start position on cam table.



c. Ramp in mode (*StartMode*=2)

Cam master and slave axes engaging 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.



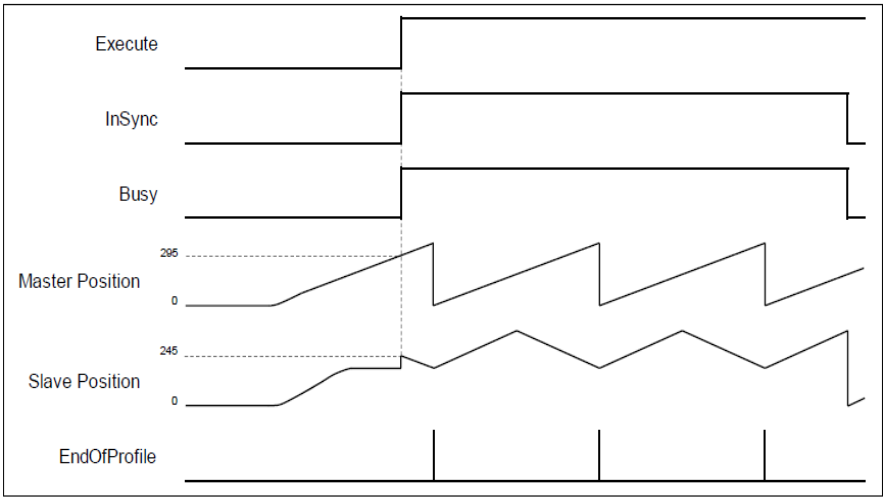
d. Ramp in positive, Ramp in negative (*StartMode* = 3, 4)

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)

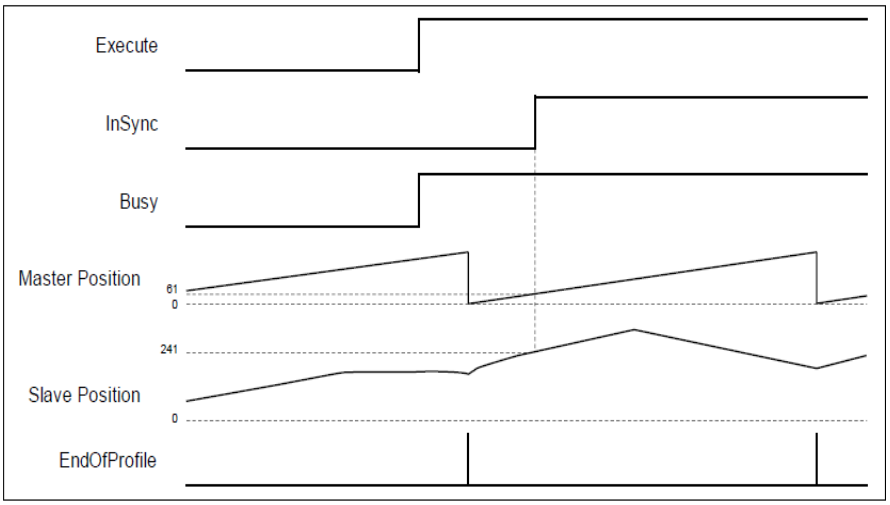
a. Absolute/ Relative mode (*StartMode* = 0, 1)

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.



b. Ramp in mode (*StartMode* = 2)

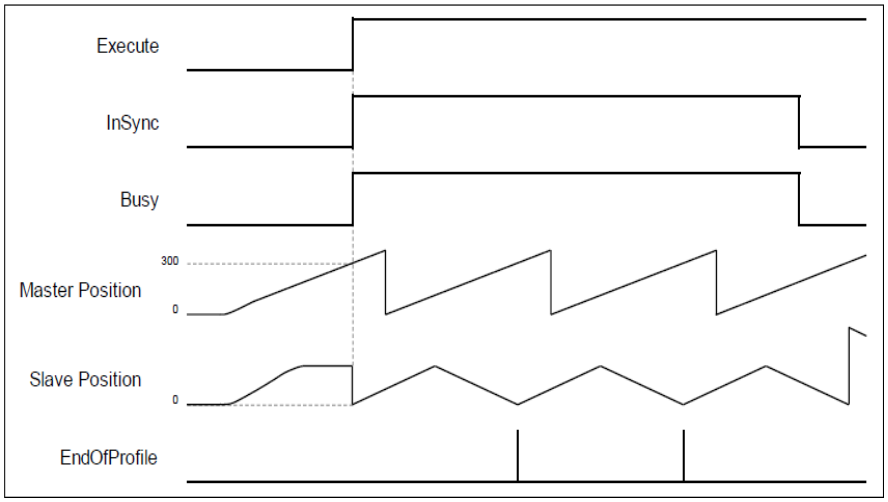
The defined positions of master and slave axis when cam is engaged, respectively, are master current position and the slave position added with a motion curve for compensation, which is configured via VelocituDiff, Acceleration and Deceleration settings, for the purpose of preventing a jump while ramping in. The slave engaging 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)

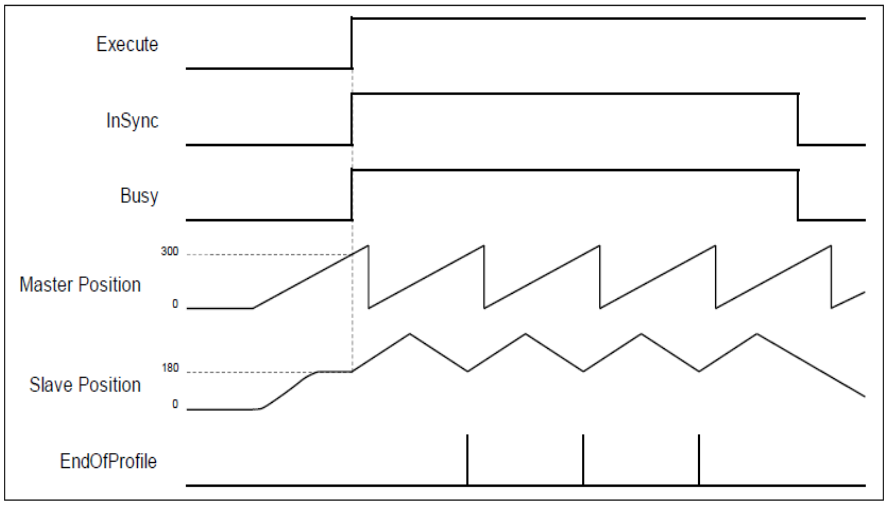
a. Absolute mode (*StartMode* = 0)

Master-slave engaging 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 (engaging) is zero under absolute mode.



b. Relative mode (*StartMode* = 1)

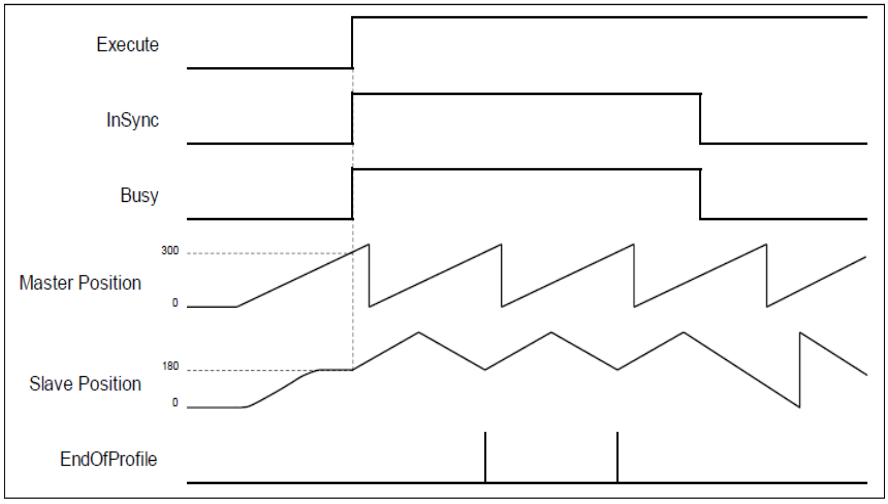
Master-slave engaging position: The current position of master axis will be the start position as well as the zero position on the cam table. The corresponding position of the slave axis on the cam table should also be zero, while the slave position (engaging) under relative mode should be the sum of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



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

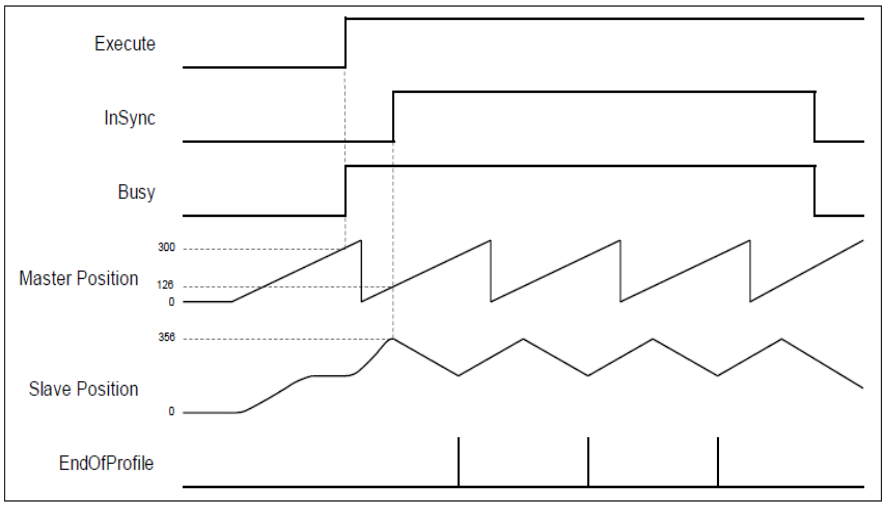
a. Absolute/ Relative mode (*StartMode* = 0, 1)

Master-slave engaging 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 (engaging) under relative mode should be the sum of corresponding position and slave current position (0+180=180).

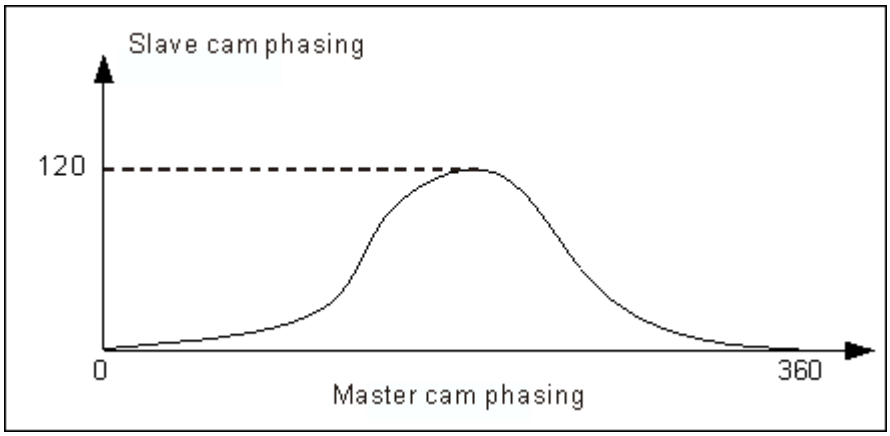


b. Ramp in mode (*StartMode* = 2)

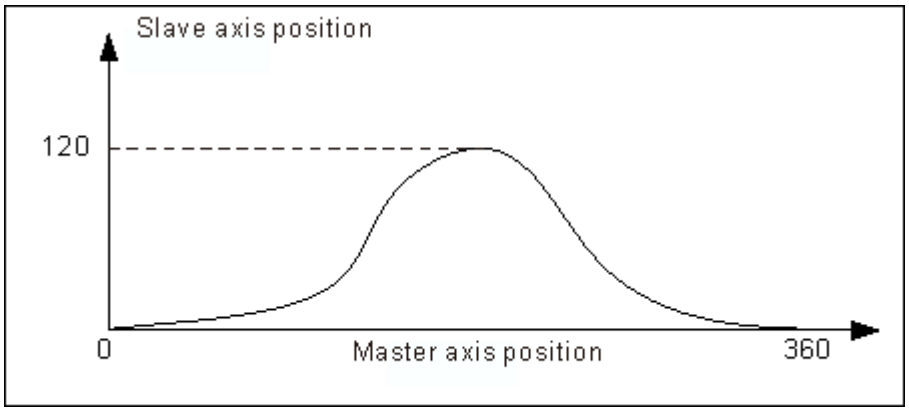
Master-slave engaging position: The current position of master axis will be the start position as well as the zero position on the cam table. A compensating curve is added to the slave position according to the settings of *VelocityDiff*, *Acceleration* and *Deceleration* to avoid jumps.



- 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.

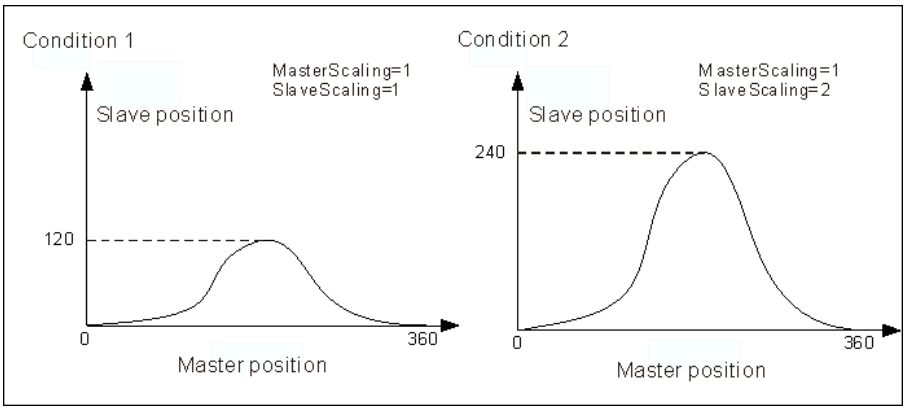


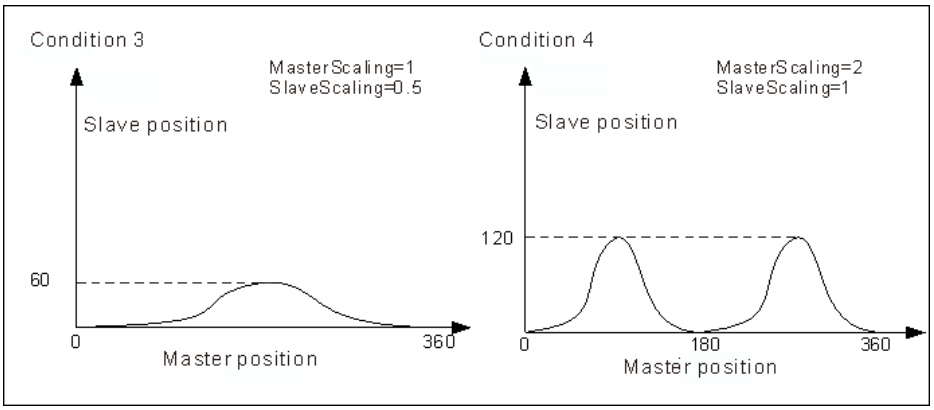
- If the master and slave axis are under absolute mode, the start position of master and slave axis will be zero while performing engaging action. Without any compensation and scaling (default setting), the relationship between the actual positions of master and slave axis are shown below.

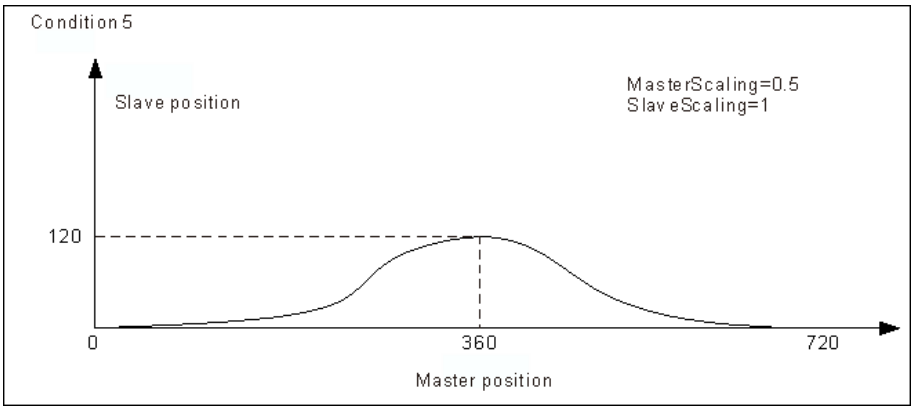


- When the position compensations and scaling are not set as default, impacts on the relationship between the actual positions of master and slave axis are shown as follows.

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.

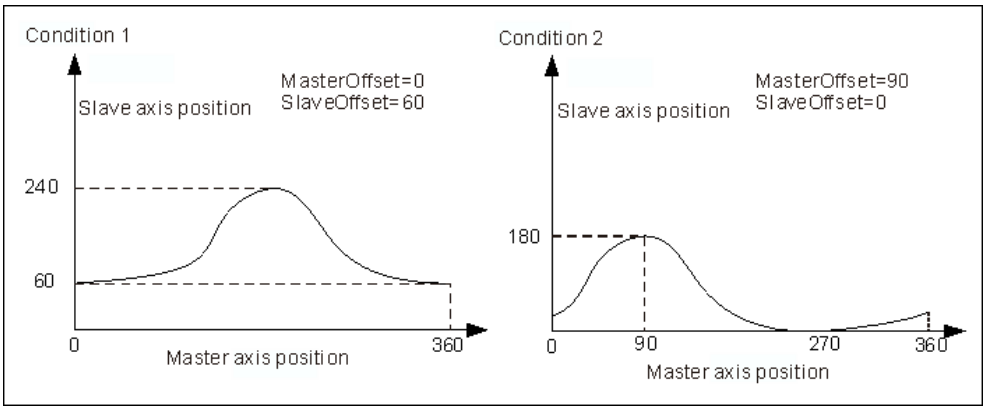
Condition 3: When *MasterScaling*=1 and *SlaveScaling*=0.5 with no Compensations, the slave position will be half of the preplanned.

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 *MasterScaling* and *SlaveScaling* are set to 1, the impact of *MasterCompensation* and *SlaveCompensation* on the actual operation of cam profile.

   The master axis offset is equivalent to a horizontal shift of the actual axis position curve when running the cam; the slave axis offset is equivalent to a vertical shift of the axis position curve when running the cam.



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 running.

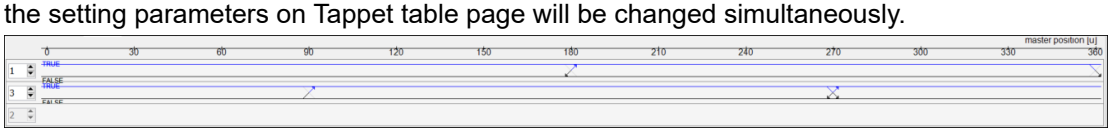
Condition 2: When *MasterScaling*=1, *SlaveScaling*=1, *MasterCompensation*=90 and *SlaveCompensation*=0, the master position relative to the slave position will be added with 90 based on the preplanned position. For example, the master position 180 corresponds to the slave position 180 in a planned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

- **Period mode**
  - Use *Periodic* of MC_CamTableSelect to control the period mode. Under non-periodic mode, *EndOfProfile* remains to be TRUE after running 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.

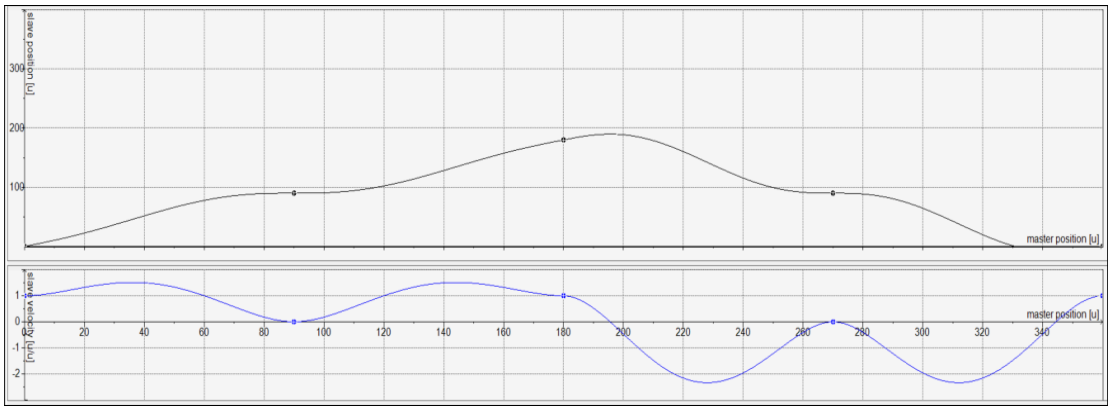
  - You can configure several tappets for each track ID on the Tappet table, then view the relationship between tappets and the master axis. While moving the points on Tappets page, the setting parameters on Tappet table page will be changed simultaneously.

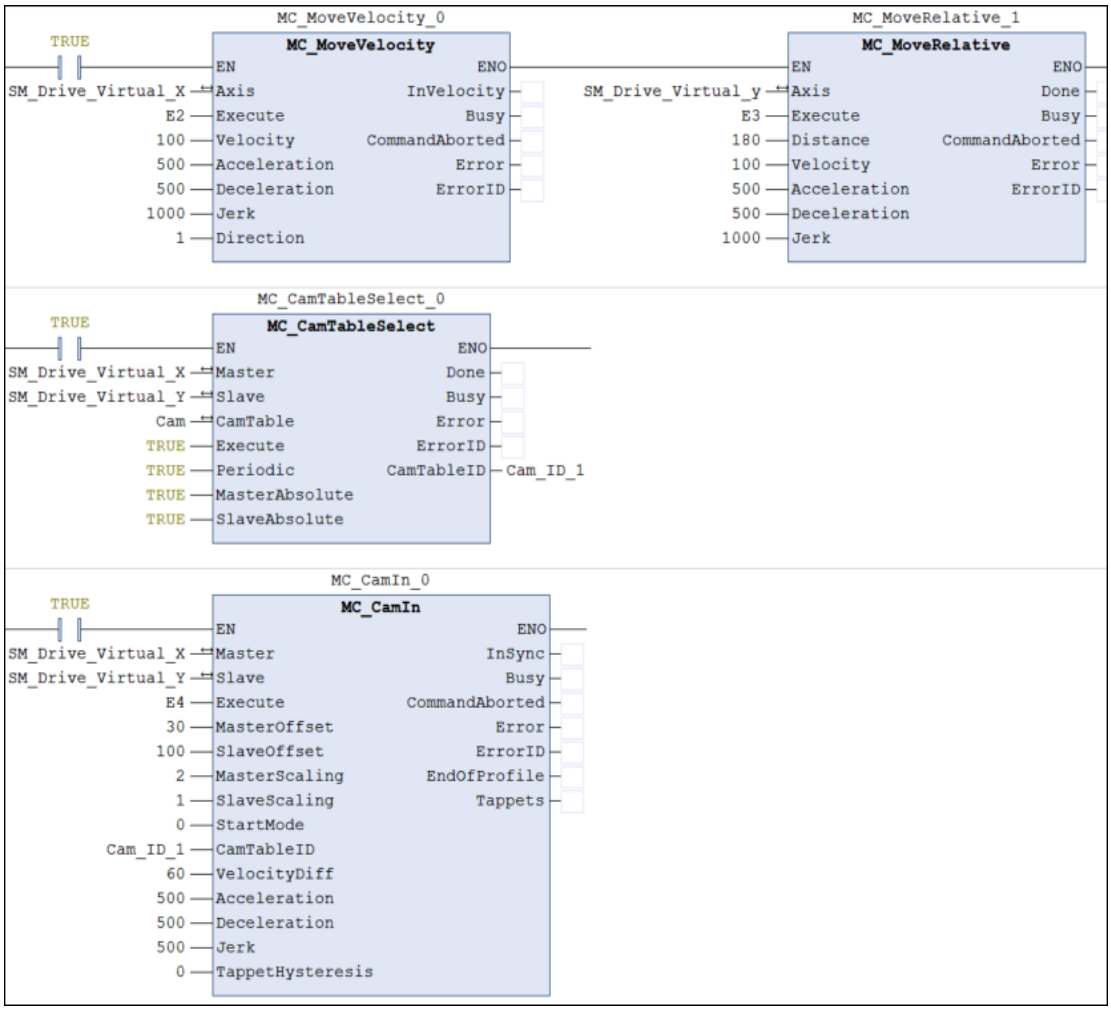


- **TappetHysteresis**
  - Set hysteresis intervals of tappet to avoid vibration in axes and encoders, which may lead to wrong switching actions. The specified axis position must exceed the interval so the next action will be run. The unit for Hysteresis is user-defined.

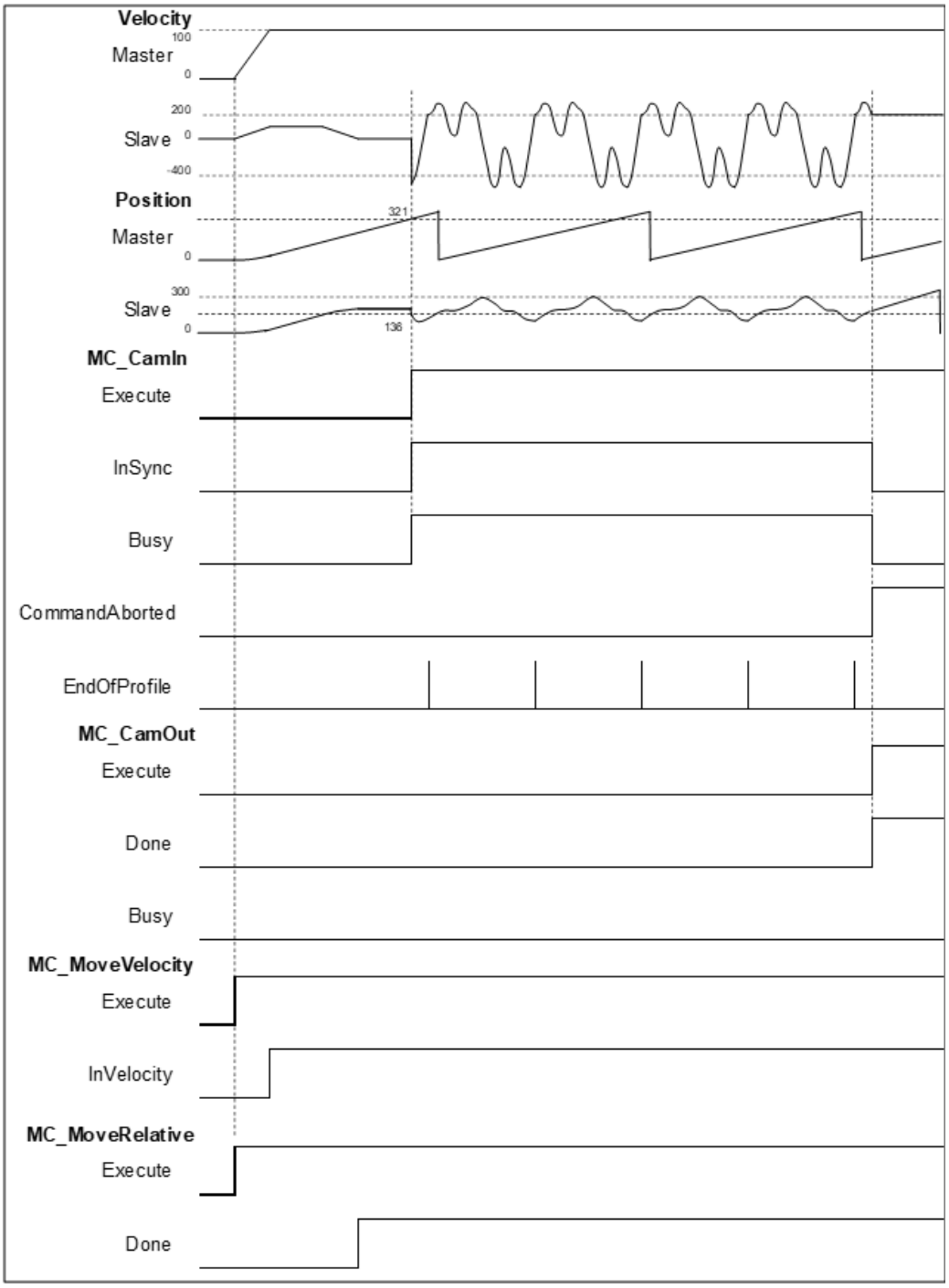
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**
  - **Example 1**: Demonstrate the running 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:





- Timing Diagram



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

  Position_Slave = *SlaveScaling*×CAM (*MasterScaling*×*MasterPosition* + *MasterOffset*) + *SlaveOffset*

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

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

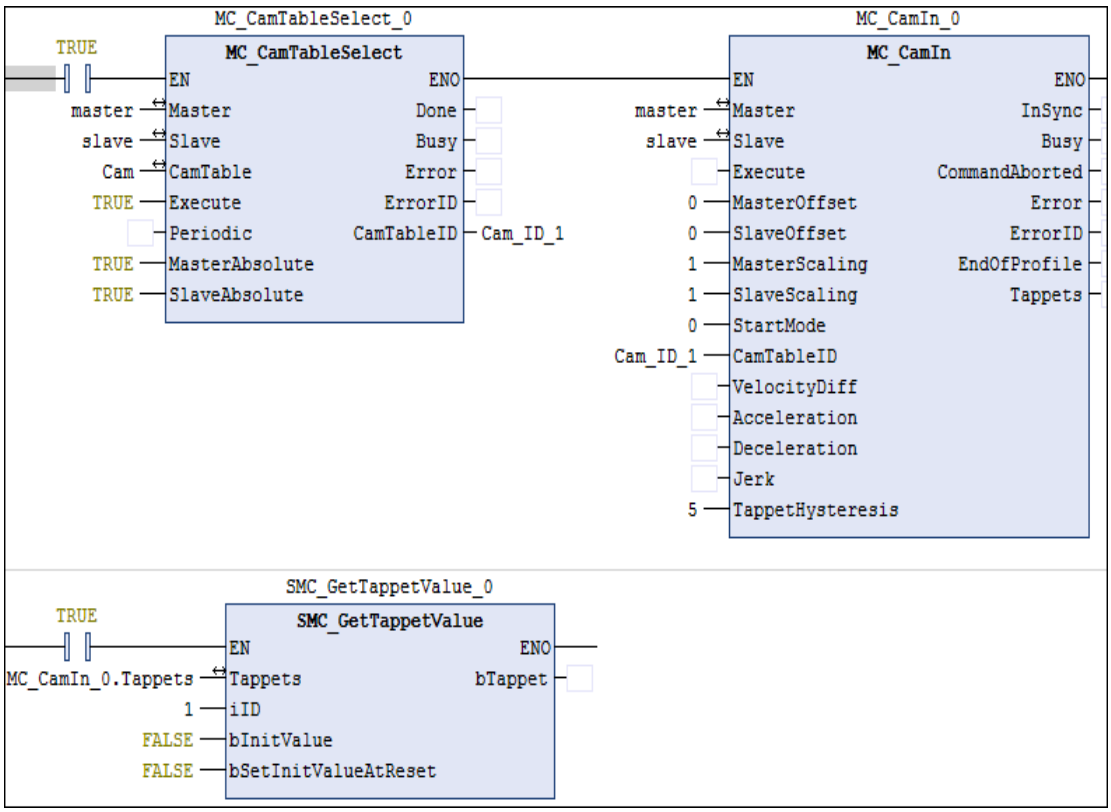
  = 36 + 100 = 136

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

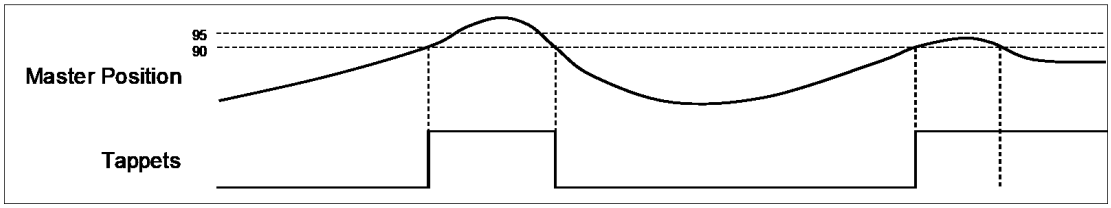
- **Example 2**: The operation of tappet after configuring TappetHysteresis as the following example demonstrates.

- Tappets

	Track ID	X	positive pass	negative pass
	1			
		90	invert	invert



- Timing Diagram

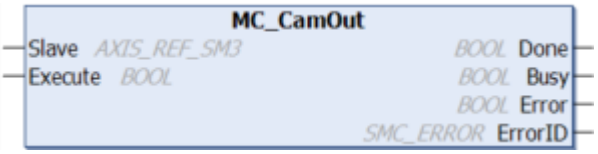


1. The tappets switch to ON when the master axis passes position 90. Master keeps moving forward until its position exceeds the hysteresis interval and the axis performs reversing. Then the master axis passes position 90 again and exceeds the hysteresis interval, which will make tappets switch to OFF.
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.

- **Supported Devices**
  - AX-series motion controller

2.1.1.9 MC_CamOut

MC_CamOut deactivates the cam 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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

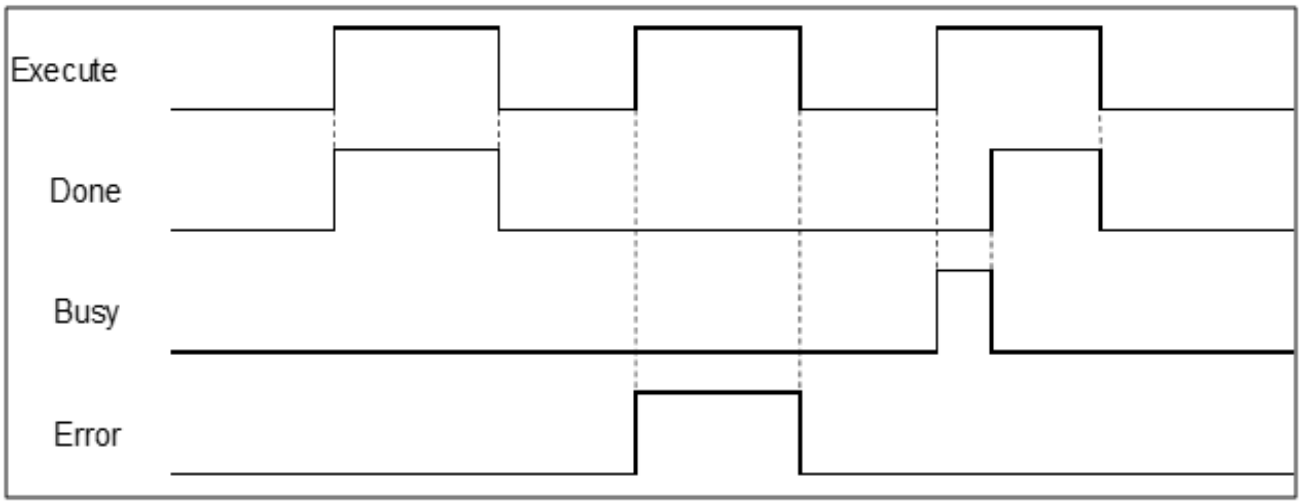
Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when the cam synchronization between master and slave axes are deactivated	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The instruction CamOut is completed.	• *Execute* is FALSE. • If *Execute* is FALSE and *Done* turns to TRUE, *Done* will be TRUE for only one scan cycle and immediately turn to FALSE.
Busy	The instruction is running.	*Error* and *Done* are TRUE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* is FALSE. (Error code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting 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 the function block.

- **Function**
  - When MC_CamOut decouples the slave axis from the master axis, the slave axis will keep the velocity and the slave state turns to 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.

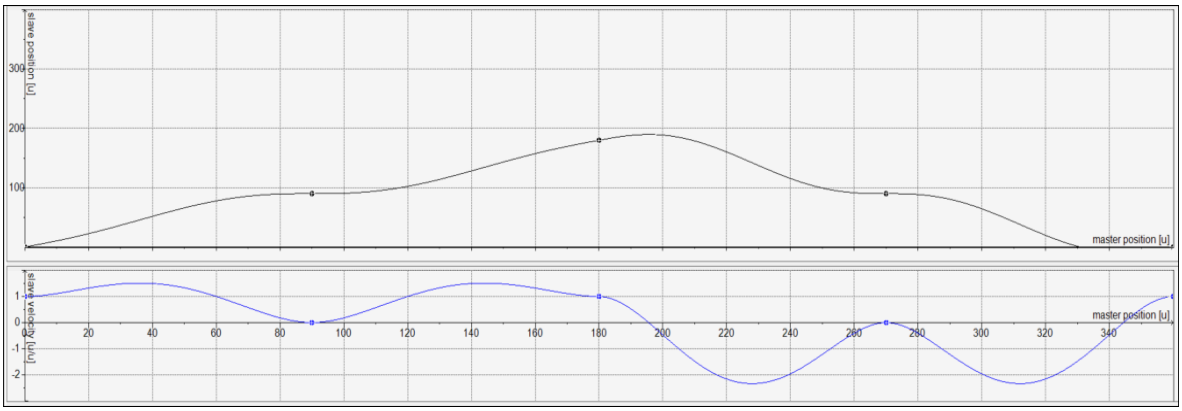
- **Troubleshooting**

  If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

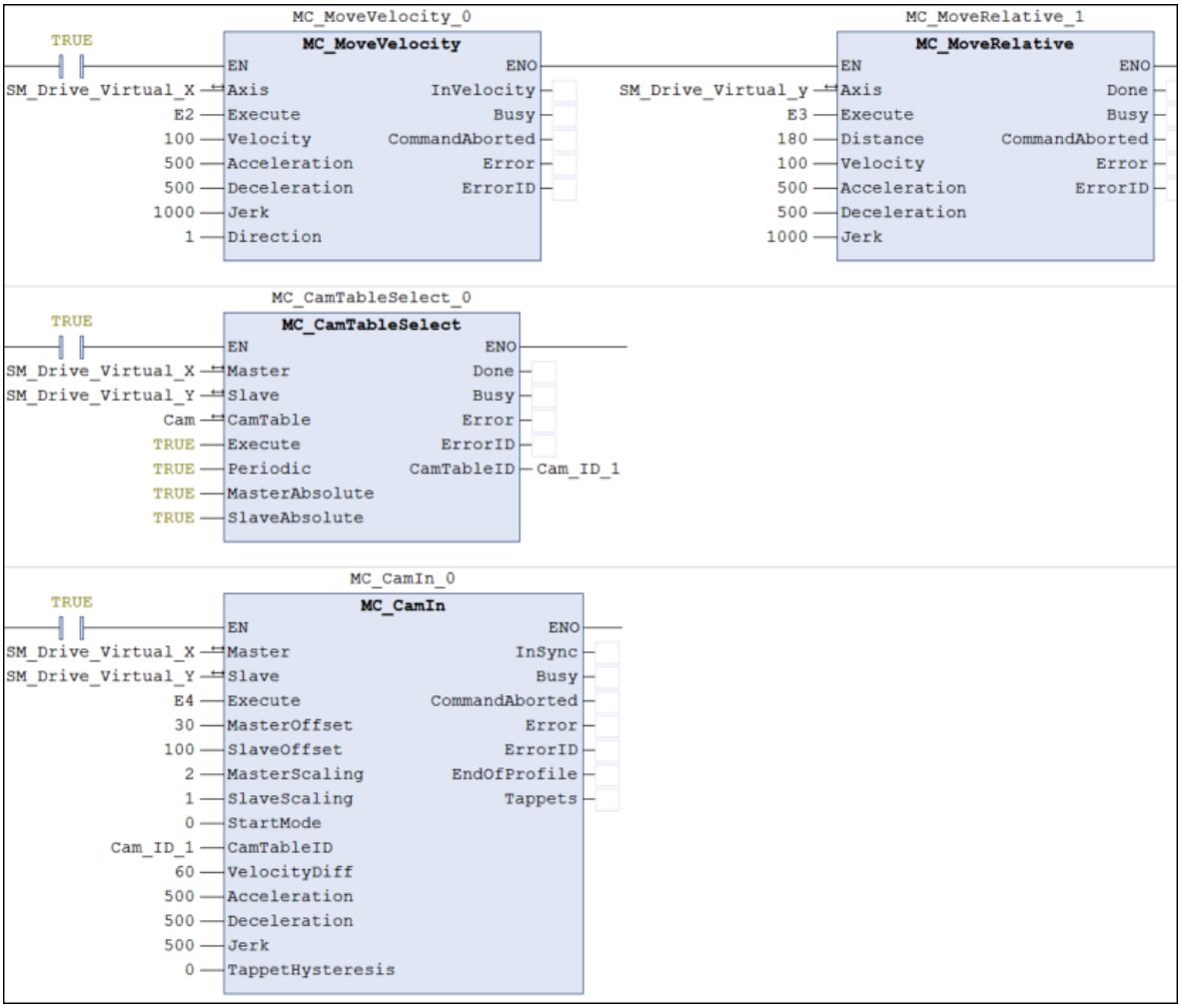
- **Example**

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

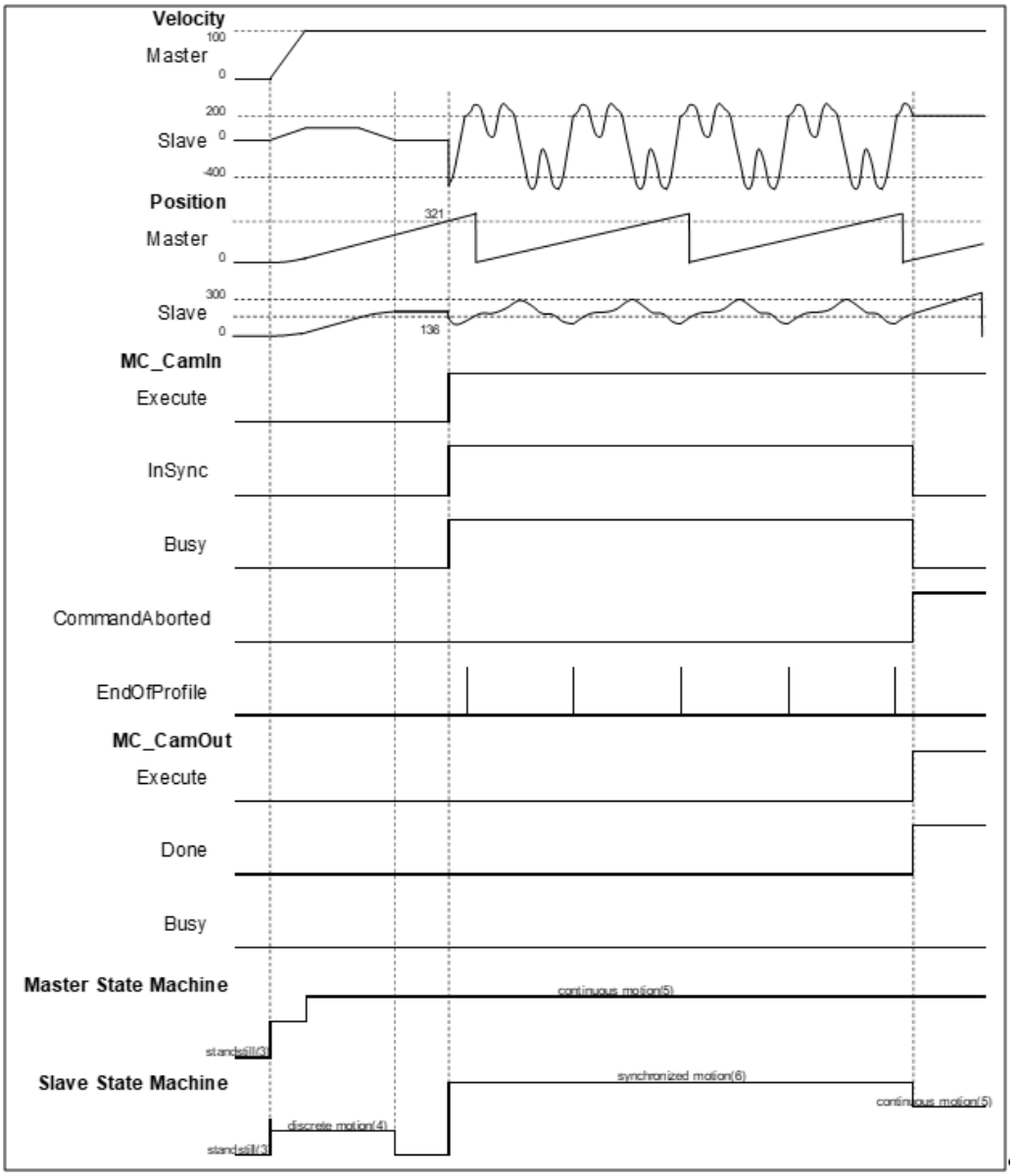
  - Cam curve planning



	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

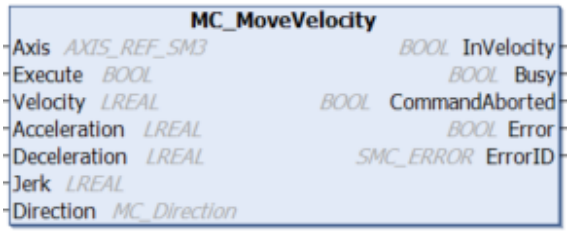


- When MC_CamOut is run, the master and slave axes will be desynchronized. At the same time, and MC_CamIn will be pulled up by the interrupting command CommandAborted.
- The slave axis continues to move at the current speed after being decoupled from the master axis, while the axis state changes to continuous_motion.

- **Supported Devices**
  - AX-series motion controller

2.1.1.10 MC_MoveVelocity

MC_MoveVelocity controls the uniform motion of the axis in position mode according to the specified motion mode and speed.

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

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Velocity	Target velocity (Unit: user unit/s)	LREAL	Positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Acceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Deceleration	Deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Jerk	Jerk value (Unit: user unit/s^3)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Direction	Direction for servo motor rotation	MC_Direction*1	3: fastest 2: current 1: positive 0: shortest -1: negative (current)	*Execute* turns to TRUE and *Busy* is FALSE.
BufferMode	Buffer mode for this instruction	MC_BUFFER_MODE*2	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	*Execute* turns to TRUE and *Busy* is FALSE

***Note**:

1. MC_Direction: Enumeration (Enum)
2. MC_BUFFER_MODE: Enumeration (Enum)

- **Outputs**

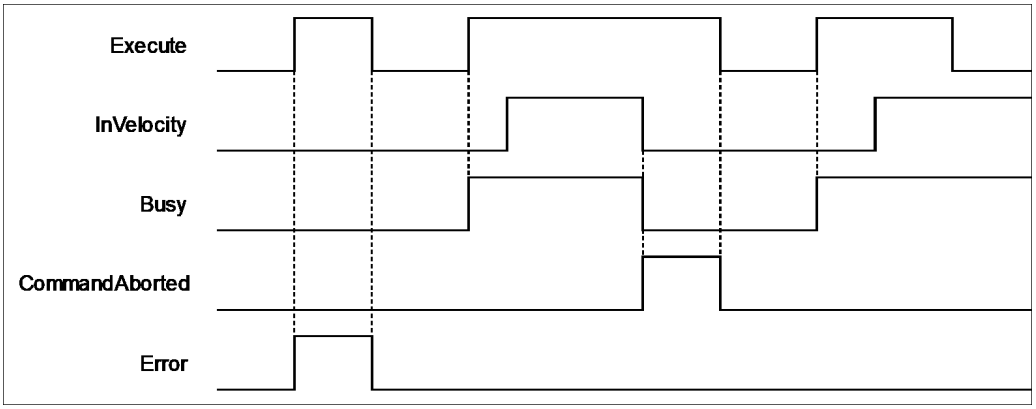
Name	Function	Data Type	Output Range (Default Value)
InVelocity	TRUE when the specified target velocity is reached	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Active	TRUE when the axis is moving	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The specified target velocity is reached.	• *CommandAborted* turns to TRUE. • *Execute* turns to TRUE and *Velocity* is changed.
Busy	*Execute* turns to TRUE.	• *Error* turns to TRUE. • *CommandAborted* turns to TRUE.
Active	*Execute* turns to TRUE.	• *Error* turns to TRUE. • *CommandAborted* turns to TRUE.
CommandAborted	• This instruction is interrupted by another instruction. • This instruction is aborted because of the running of MC_Stop instruction.	• *Execute* changes to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, it will be TRUE for only one period and immediately turn to FALSE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



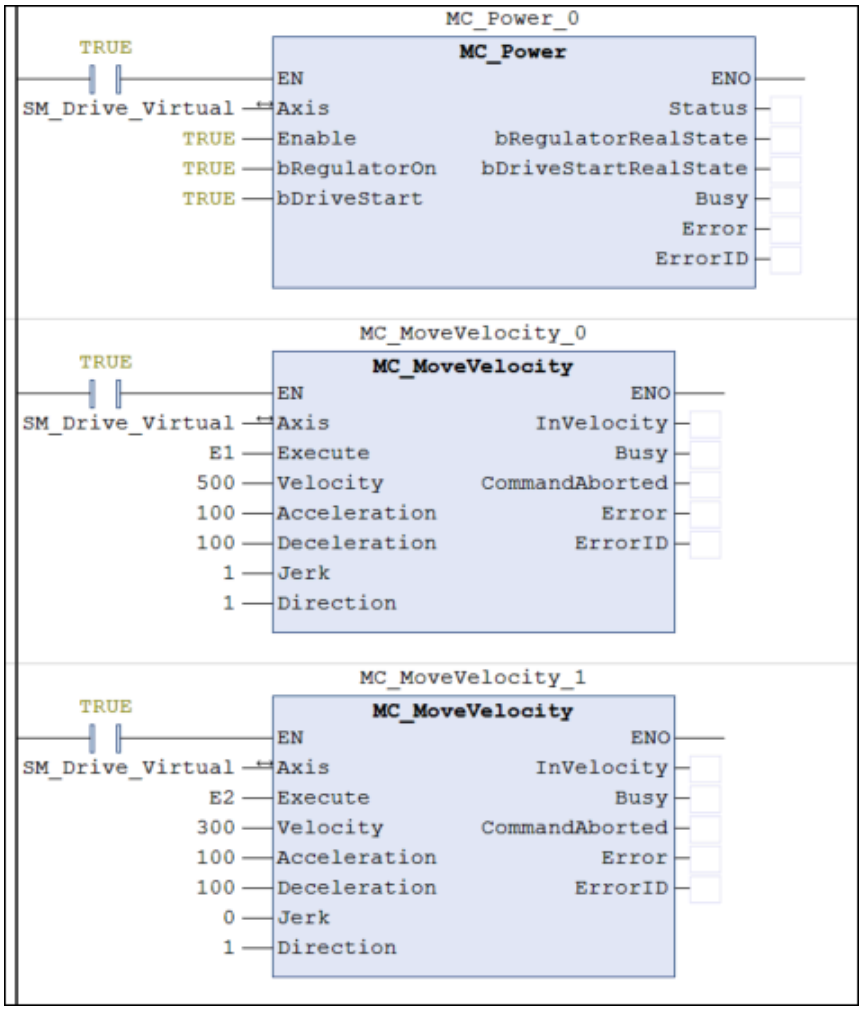
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.

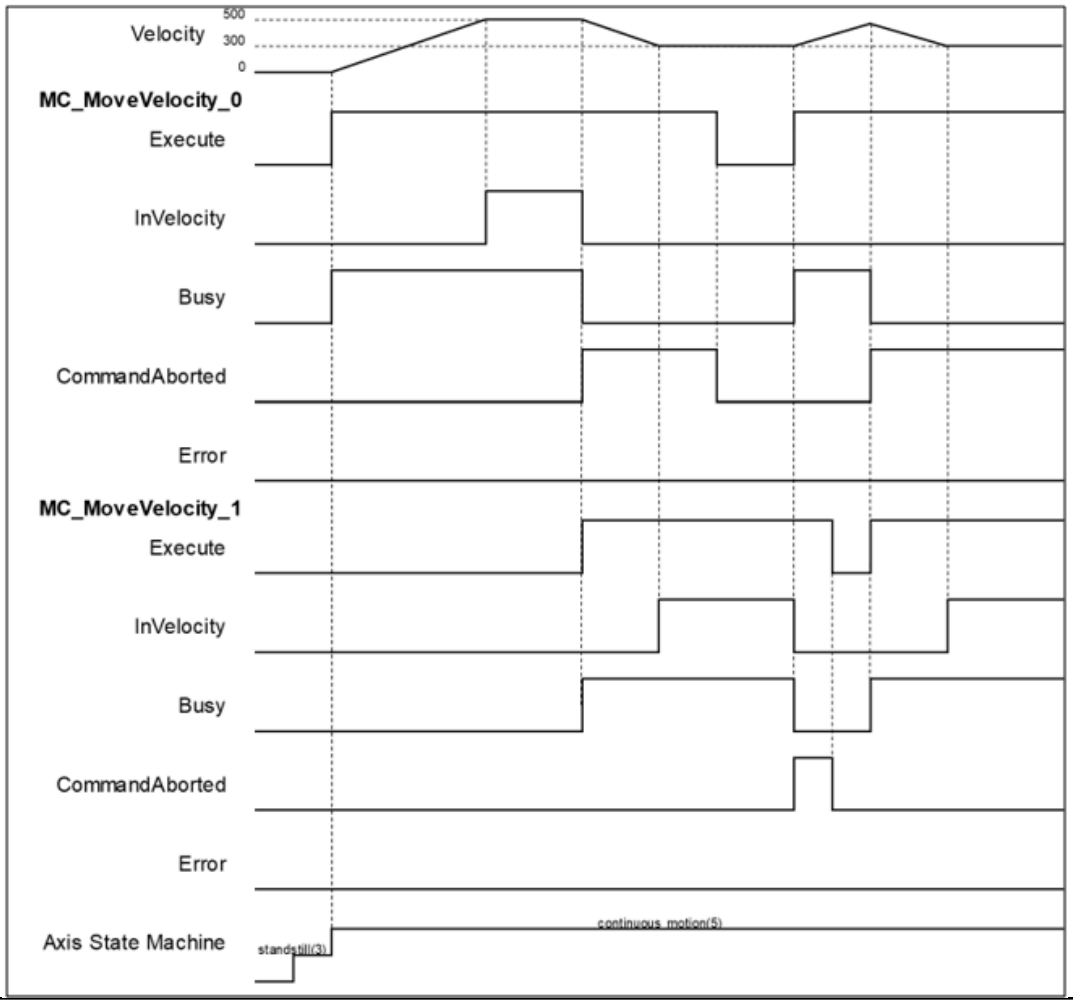
***Note**: AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.

- **Function**
  - When *Execute* changes to TRUE, the instruction performs uniform motion with the target velocity (*Velocity*), acceleration rate (*Acceleration*), deceleration rate (*Deceleration*) and Jerk value (*Jerk*).
  - You can run another motion instruction to abort the ongoing MC_MoveVelocity.
  - When interrupted by other instructions, output *InVelocity* will turn to FALSE, and *CommandAborted* will turn to TRUE.
  - When *Execute* of MC_MoveVelocity changes to TRUE, the axis starts to move at the target velocity. Even if *Execute* turns FALSE, the running of the function block will not be affected.
  - When *Execute* is triggered again and a new target velocity is specified, the axis velocity will change to the new one.
  - If *Execute* changes to FALSE after the function block runs, *InVelocity* will turn TRUE when the target velocity is reached. *InVelocity* will remain as TRUE, until it is interrupted by other instructions.
  - *InVelocity* remains as TRUE when MC_MoveVelocity reaches the target velocity. Even if the velocity is changed by MC_MoveSuperimposed, the motion of *InVelocity* will not be affected.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

This example describes the behavior of running two MC_MoveVelocity instructions.



- Timing Diagram



- When *Execute* changes to TRUE, the first MC_MoveVelocity instruction controls the axis to reach the target velocity. When it reaches the target velocity of 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 of 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 stop 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 turn FALSE to TRUE again and stops MC_MoveVelocity_0. The axis decelerates again without reaching the target velocity.

- **Supported Devices**
  - AX-series motion controller

2.1.1.11 MC_PositionProfile

MC_PositionProfile sets 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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
ArraySize	The number of motion profile arrays	INT	Positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
PositionScale	Overall scaling of position	LREAL	Negative, positive or 0 (1)	*Execute* turns to TRUE and *Busy* is FALSE.
Compensation	Overall profile compensation (Unit: user unit/s)	LREAL	Negative, positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.

- **Outputs**

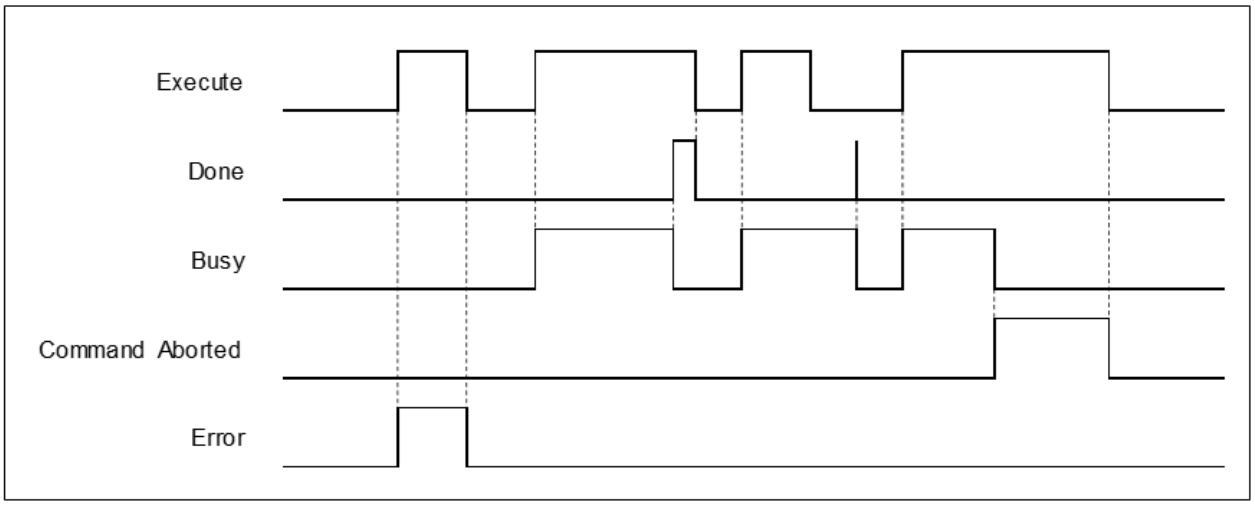
Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when path planning is finished	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The running of path planning is complete.	• *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 turn to FALSE.
Busy	*Execute* changes to TRUE.	• *Done* changes to TRUE. • *Error* changes to TRUE. • *CommandAborted* turns to TRUE.
CommandAborted	This instruction is interrupted by another function block	• *Execute* changes to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, it will be TRUE for only one period and immediately turn to FALSE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*1	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.
TimePosition	Time and position during running.	MC_TP_REF*2	MC_TP_REF	*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 the function block.
2. MC_TP_REF: Structure(STRUCT).

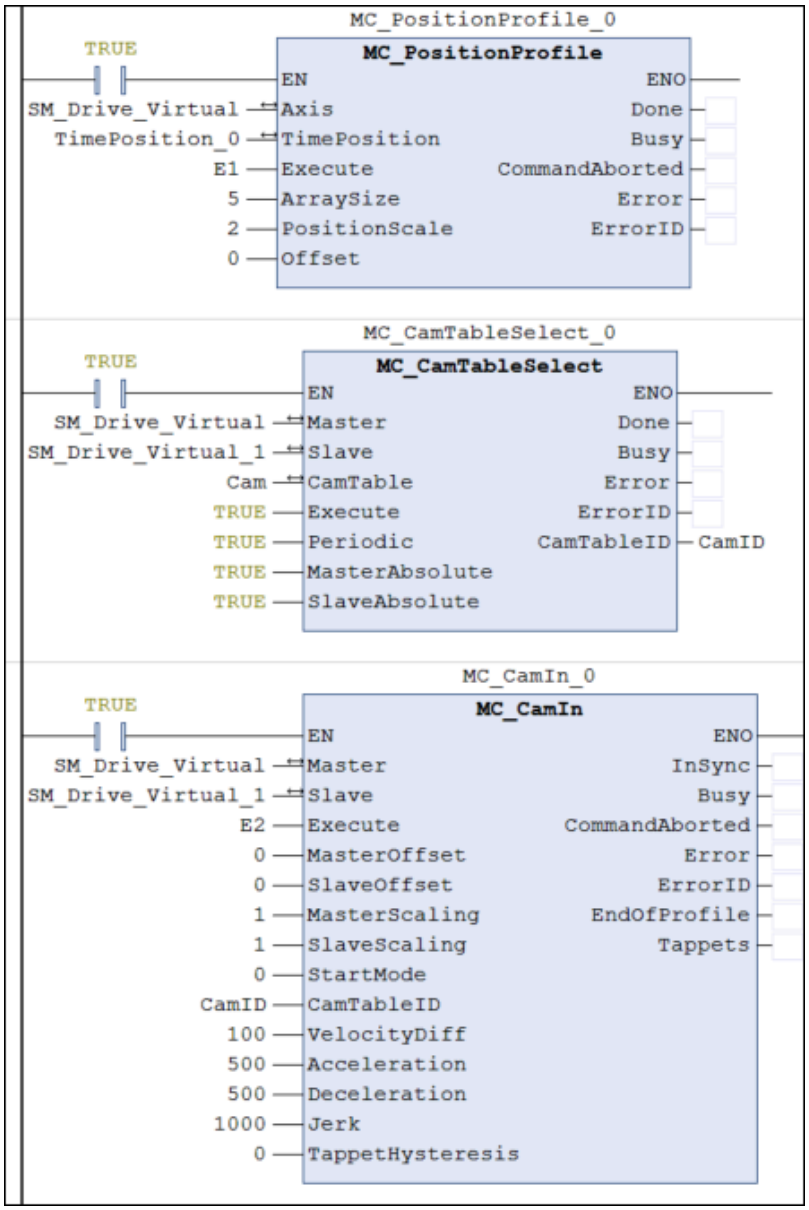
Name	Function	Data Type	Setting Value (Default Value)
Number_of_pairs	This variable does not need to be set and has been replaced by the input *ArraySize*.	INT	-
IsAbsolute	Set the mode of position.	BOOL	TRUE: Absolute mode FALSE: Relative mode (TRUE)
MC_TP_Array	Time and position data during the running 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 motion.
  - MC_MoveSuperimposed will not be able to function while MC_PositionProfile is being used.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  The example below describes the behavior of MC_PositionProfile instruction.

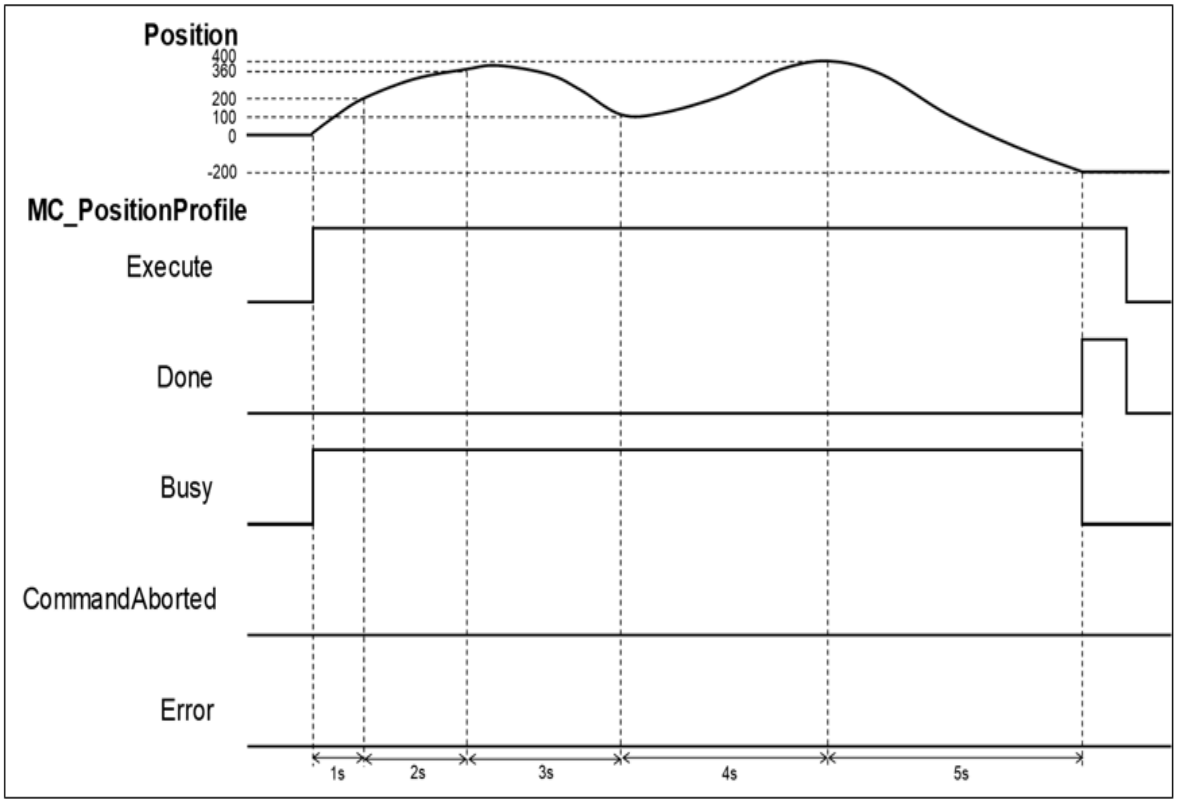


- Configure for the curve of motion profile:

```
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

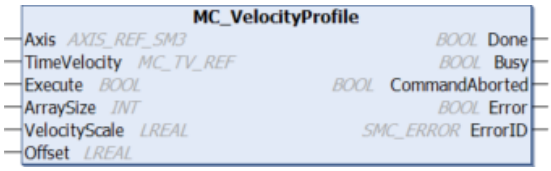


- When *Execute* changes to TRUE, the target axis moves along the curve according to the settings of delta_time and position in TimePosition.
- The value of *IsAbsolute* is TRUE, MC_PositionProfile plans motion curves in the Absolute mode.
- The motion curve of MC_PositionProfile is executed according to the time set in *TimePosition*. If *PositionScale*=2, the position will be 200 after running for 1s; the position will be 360 after 2s, and so on. When running after 5s, the position should be –200.

- **Supported Devices**
  - AX-series motion controller

2.1.1.12 MC_VelocityProfile

MC_VelocityProfile sets 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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
ArraySize	The number of motion profile arrays	INT	Positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
VelocityScale	Overall scaling of velocity	LREAL	Negative, positive or 0 (1)	*Execute* turns to TRUE and *Busy* is FALSE.
Compensation	Overall profile Compensation (Unit: user unit/s)	LREAL	Negative, positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.

- **Outputs**

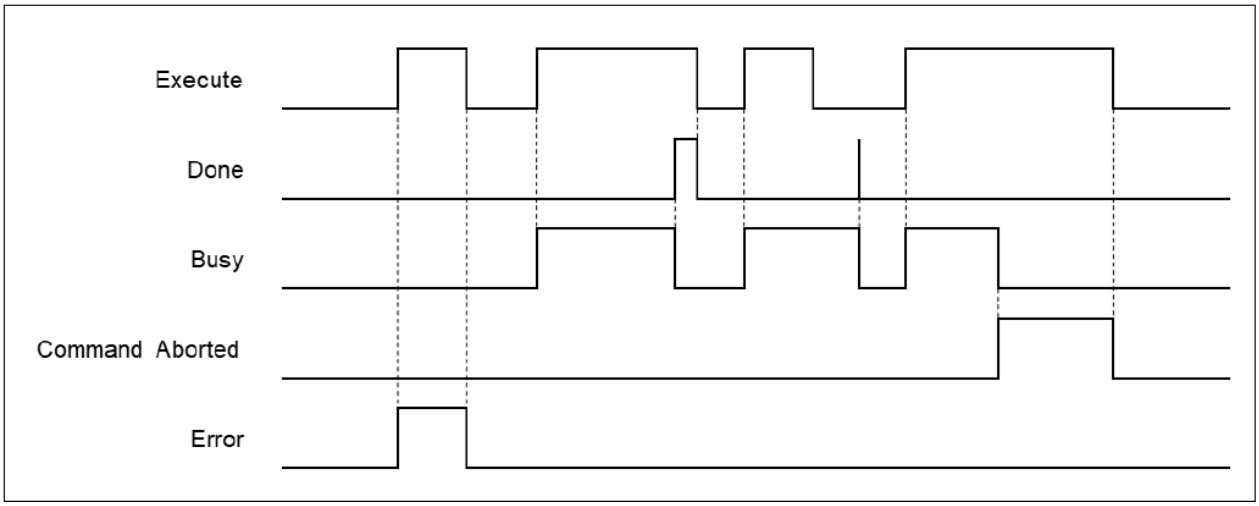
Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when the planned path is executed	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	A speed-based planned motion path is completed.	• *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 turn to FALSE.
Busy	*Execute* changes to TRUE.	• *Done* changes to TRUE. • *Error* changes to TRUE. • *CommandAborted* turns to TRUE.
CommandAborted	This instruction is interrupted by another function block.	• *Execute* changes to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, it will be TRUE for only one period and immediately turn to FALSE.
Error ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3 *1	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.
TimeVelocity	Time and velocity during running	MC_TV_REF *2	MC_TV_REF	*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 the function block.
2. MC_TV_REF: Structure (STRUCT)

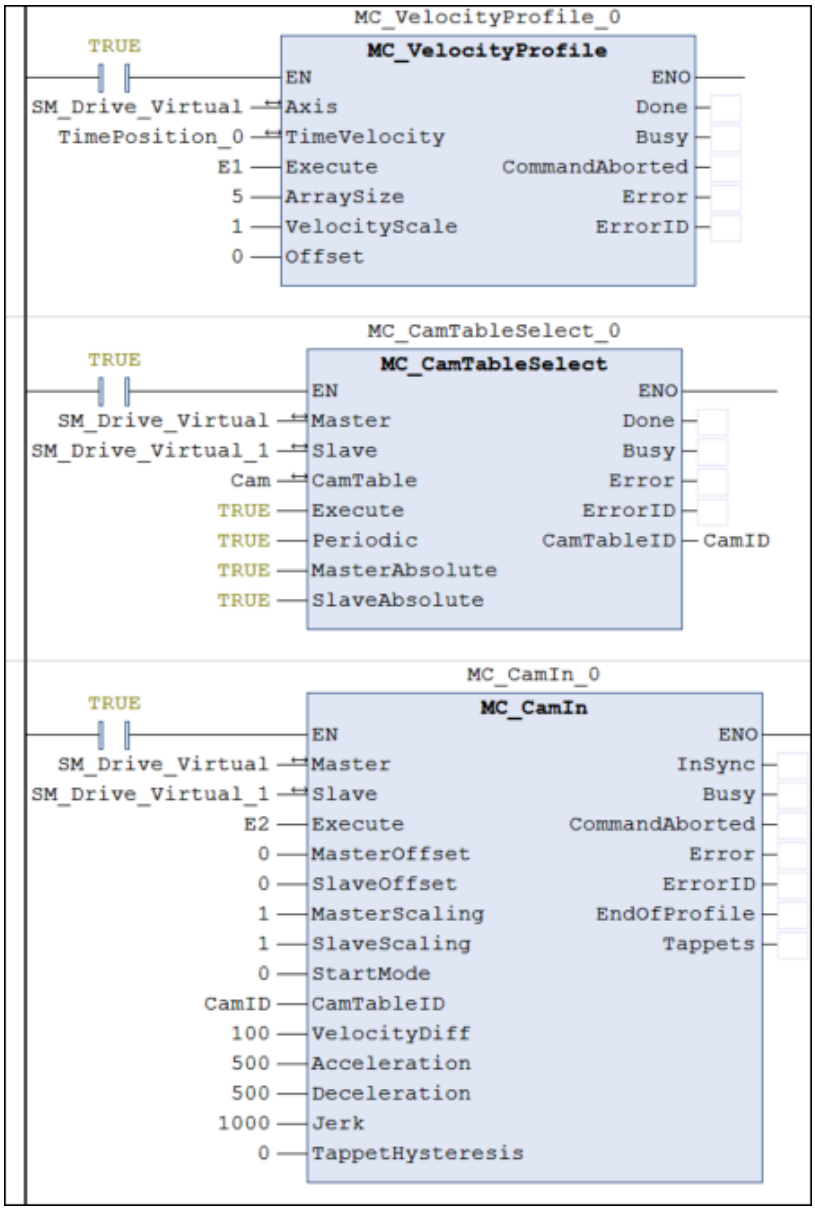
Name	Function	Data Type	Setting Value (Default Value)
Number_of_pairs	There's no need to set this variable, which has been replaced by input *ArraySize*.	INT	-
IsAbsolute	Set the mode of velocity.	BOOL	TRUE: Absolute mode FALSE: Relative mode (TRUE)
MC_TV_Array	Time and velocity data during running of 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 motion.
  - MC_VelocityProfile will not be able to function while MC_PositionProfile is being used.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  This example describes the behavior of MC_VelocityProfile.

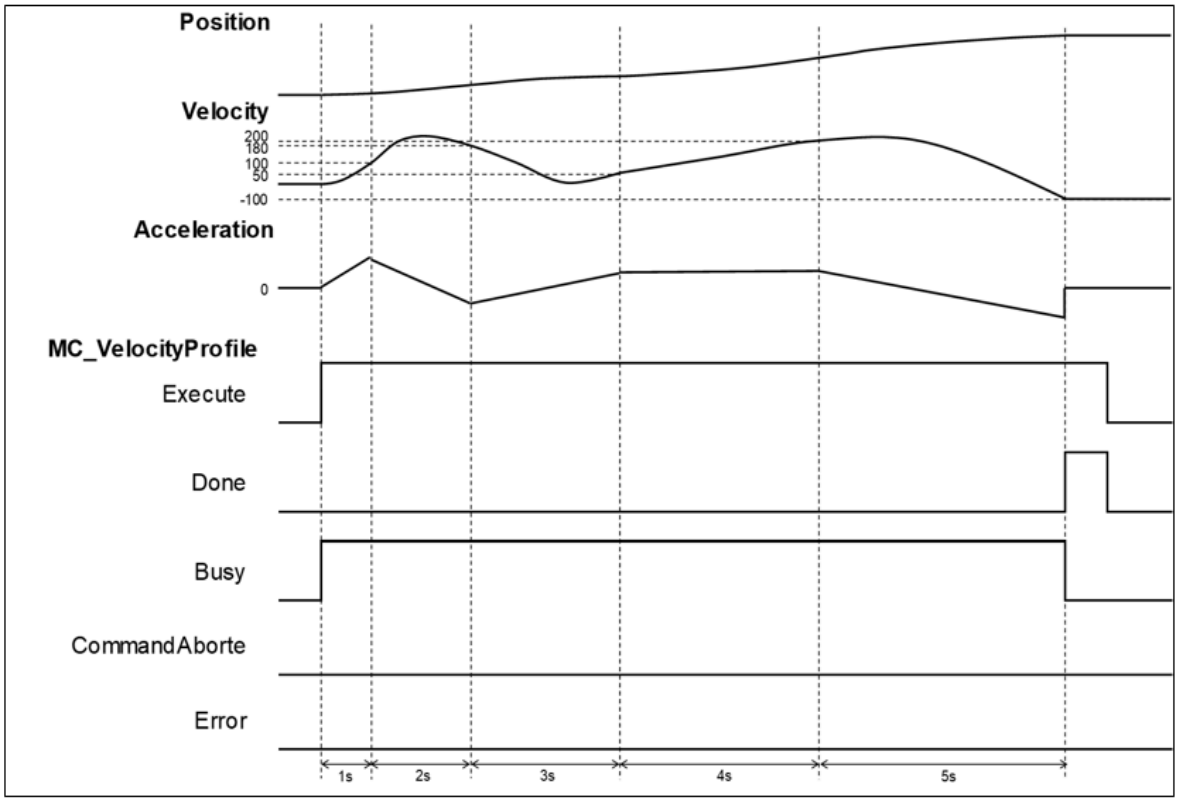


- Configure for the curve of motion profile:

```
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* changes to TRUE, the target axis moves along the curve according to the settings of delta_time and velocity setting in TimeVelocity.
- If *IsAbsolute* is TRUE, MC_VelocityProfile plans motion curves in the Absolute mode.
- The motion curve of MC_PositionProfile is executed according to the time velocity set in *TimeVelocity*. The velocity will be 100 after running for 1s; the velocity will be 180 after 2s, and so on. When running after 5s, the velocity should be –100.

- **Supported Devices**
  - AX-series motion controller

2.1.1.13 MC_AccelerationProfile

MC_AccelerationProfile sets time and acceleration to plan motion profiles.

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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
ArraySize	The number of motion profile arrays	INT	Negative, positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
AccelerationScale	Overall scaling of acceleration	LREAL	Negative, positive or 0 (1)	*Execute* turns to TRUE and *Busy* is FALSE.
Offset	Overall offset (Unit: user unit/s)	LREAL	Negative, positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.

- **Outputs**

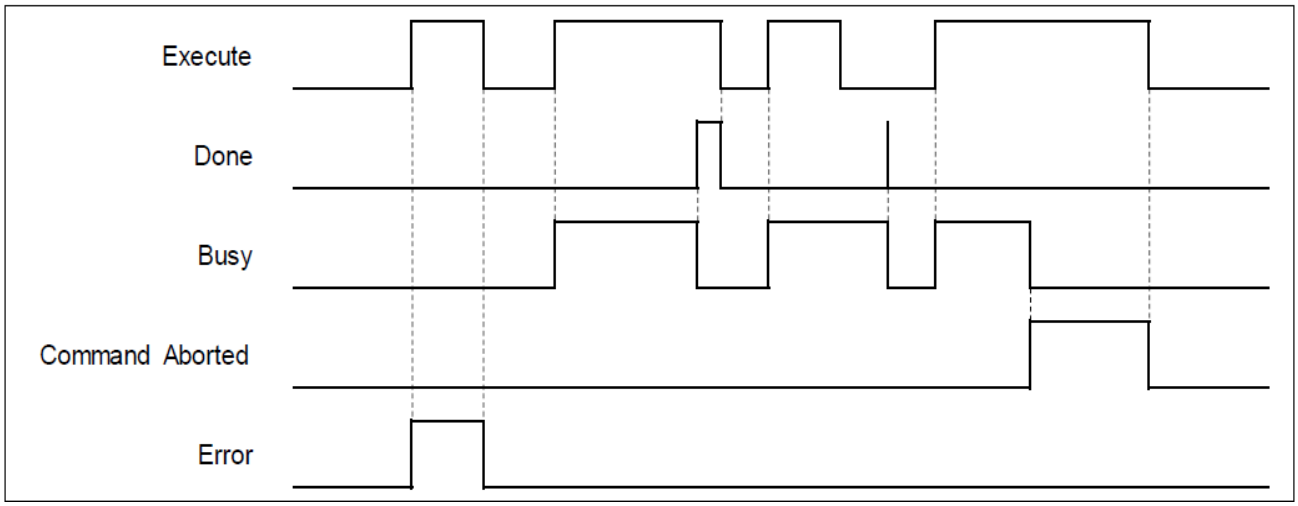
Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when the running of path planning is finished	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The path planning is complete.	• *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 turn to FALSE.
Busy	*Execute* changes to TRUE.	• *Done* changes to TRUE. • *Error* changes to TRUE. • *CommandAborted* turns to TRUE.
CommandAborted	This instruction is interrupted by another function block.	• *Execute* changes to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, it will be TRUE for only one period and immediately turn to FALSE.
Error ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*1	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.
TimeAcceleration	Time and acceleration during running	MC_TA_REF*2	MC_T_REF	*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 the function block.
2. MC_TA_REF: Structure (STRUCT)

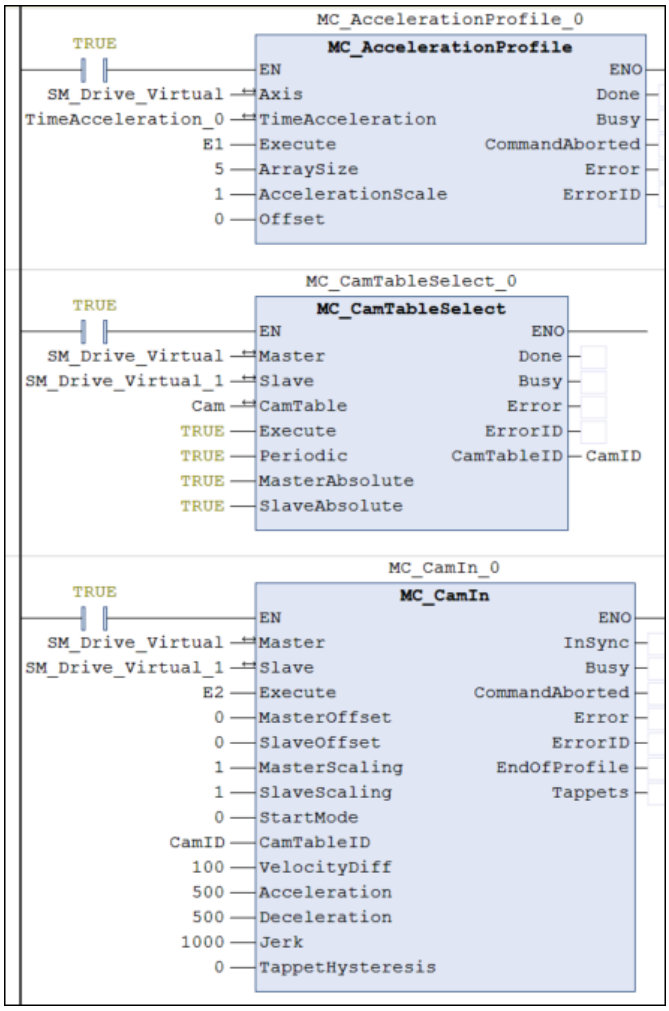
Name	Function	Data Type	Setting Value (Default Value)
Number_of_pairs	This variable does not need to be set and has been replaced by the input *ArraySize*.	INT	-
IsAbsolute	Set the mode of acceleration.	BOOL	TRUE: Absolute mode FALSE: Relative mode (TRUE)
MC_TA_Array	Time and acceleration data during the running 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	Duration 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 motion.
  - MC_MoveSuperimposed cannot add a distance to an existing motion of MC_AccelerationProfile.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

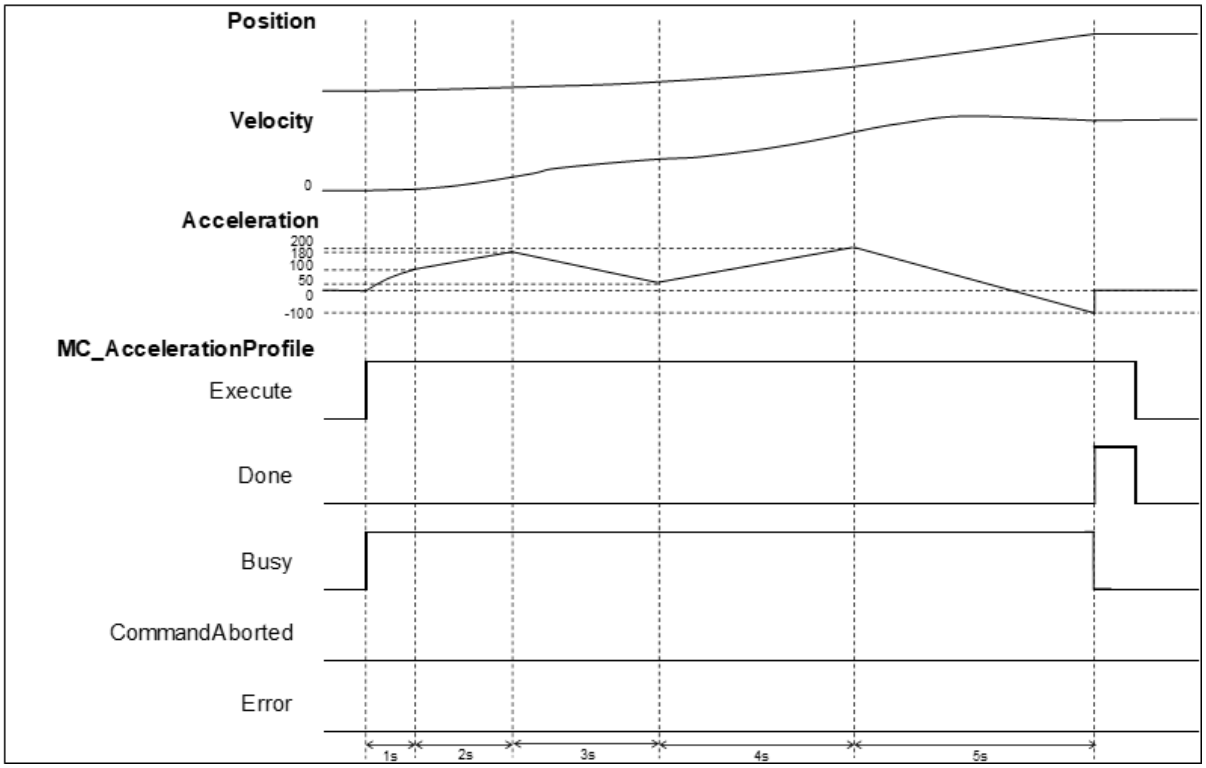
  This example describes the behavior of MC_AccelerationProfile.




```
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

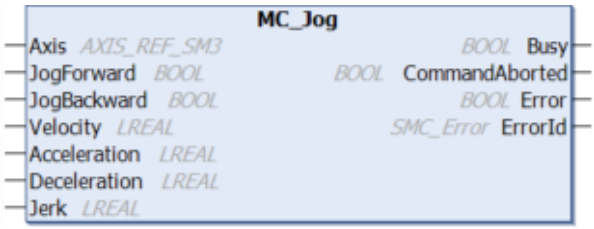


  - When *Execute* changes to TRUE, the target axis moves along the curve, which is generated by the settings of delta_time and *Acceleration* in *TimeAcceleration*.
  - If *IsAbsolute* is TRUE, MC_AccelerationProfile plans motion curves in the Abosolue mode.
  - The motion curve of MC_PositionProfile is executed according to the time acceleration set in *TimeAcceleration*. The acceleration will be 100 after running for 1s; the acceleration will be 180 after 2s, and so on. When running after 5s, the acceleration should be –100.

- **Supported Devices**
  - AX-series motion controller

2.1.1.14 MC_Jog

MC_Jog enables an axis to move 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 =>);

- **Inpts**

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)	*Execute* turns to TRUE and *Busy* is FALSE
Acceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE
Deceleration	Deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*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)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	*JogForward* or *JogBackward* turns to TRUE	• *Error* turns to TRUE. • *CommandAborted* turns to TRUE. • When status is standstill and there's no complementarity between the status of *JogForward* and *JogBackward*.
CommandAborted	The instruction is interrupted.	*JogForward* and *JogBackward* turn to FALSE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*JogForward* and *JogBackward* turn to FALSE. (Error code is cleared)
ErrorID		

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*JogForward* or *JogBackward* turns to TRUE.

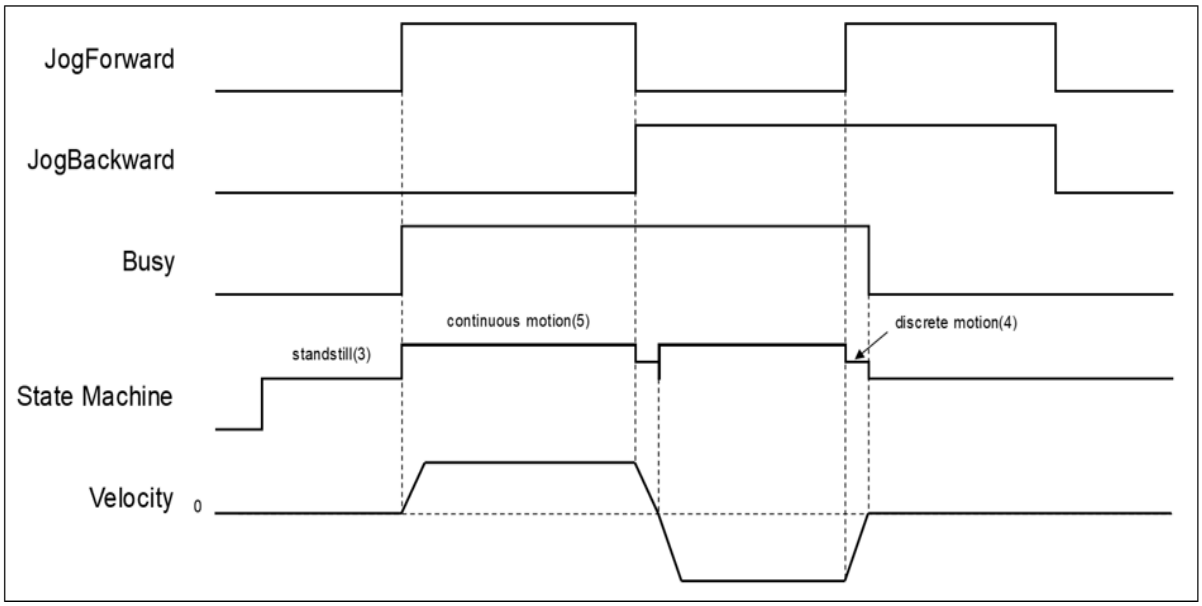
***Note**: AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.

- **Function**

  - MC_Jog performs a continuous motion with the specified *Velocity*, while *JogForward* and *JogBackward* determine the direction of axis motion (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 parameters to perform Jog operation.



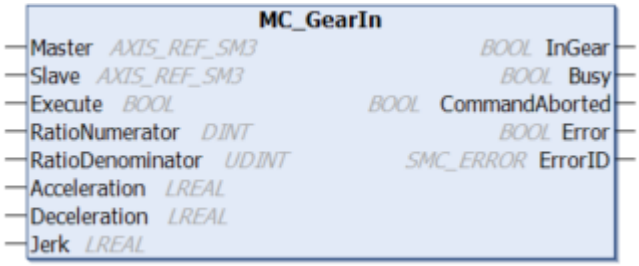
- The following are the detailed descriptions of the timing program:
  - When *JogFoward* is TRUE, and *JogBackward* is FALSE, the axis starts to move forward. The state machine is continuous_motion.
  - When *JogFoward* is FALSE, and *JogBackward* is TRUE, the axis starts to move backward. The state machine is discrete_motion.
  - When the decelerating velocity reaches 0, the state machine will turn to Standstill. Then, the axis starts to accelerate in negative direction, and the state machine is continuous_motion. *Busy* is still TRUE.
  - When both *JogForward* and *JogBackward* are TRUE, the axis will accelerate or decelerate to reach a velocity of 0, and the state machine will be discrete_motion.
  - When both *JogForward* and *JogBackward* are TRUE, the axis will stop moving and the state machine will be Standstill. *Busy* output changes from TRUE to FALSE.

- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Supported Devices**
  - AX-series motion controller

2.1.1.15 MC_GearIn

MC_GearIn couples the master and slave axis, and applies a certain velocity ratio between the two.

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

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Execute	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
RatioNumerator	Gear ratio numerator[*1] between master and slave axis	DINT	Negative, positive or 0 (1)	*Execute* turns to TRUE and *Busy* is FALSE.
RatioDenominator	Gear ratio denominator[*1] between master and slave axis	UDINT	Positive (1)	*Execute* turns to TRUE and *Busy* is FALSE.
Acceleration	Acceleration (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Deceleration	Deceleration (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Jerk	Jerk (Unit: user unit/s^3)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
BufferMode	Buffer mode for this instruction	MC_BUFFER_MODE[*2]	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	*Execute* turns to TRUE and *Busy* is FALSE.

***Note**:

1. A negative gear ratio will make the master and slave axis move in an opposite direction.
2. MC_BUFFER_MODE: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Active	TRUE when the axis is moving	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
InGear	TRUE when engaging is complete	BOOL	TRUE/FALSE (FALSE)

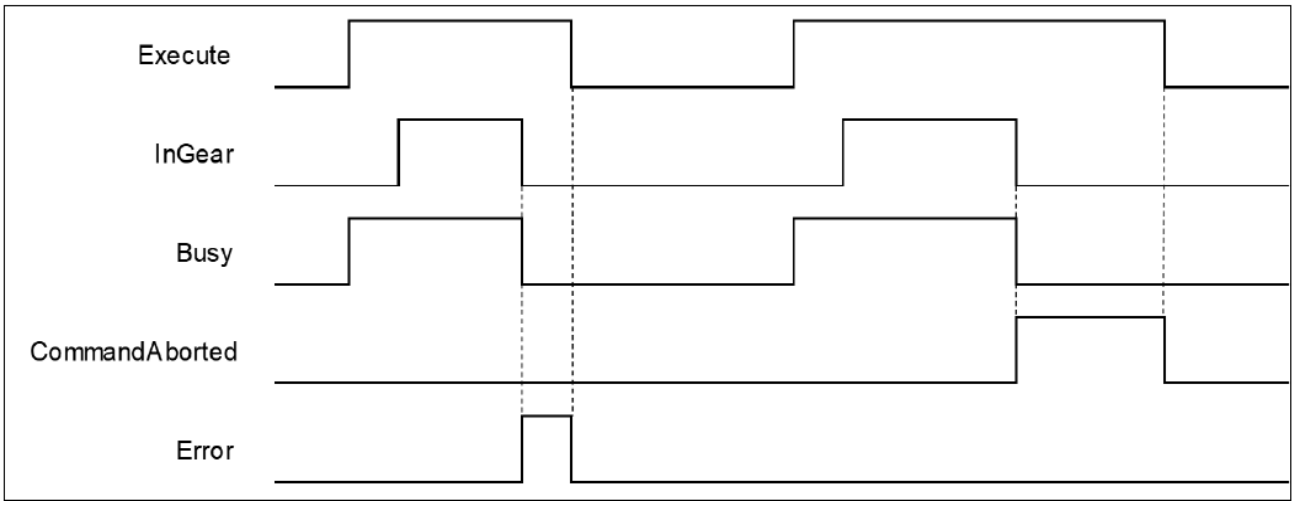
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
InGear	The slave axis reaches the target velocity and the engaging is successful.	• *Error* turns to TRUE. • *CommandAborted* turns to TRUE. • The gear ratio changes.
Busy	The engaging is processed.	• *Error* turns to TRUE. • *CommandAborted* turns to TRUE.
Active	*Execute* turns to TRUE.	• *Error* turns to TRUE • *CommandAborted* turns to TRUE.
CommandAborted	• MC_GearOut is run. • The instruction is interrupted by another function block. • This instruction is aborted by MC_Stop.	• *Execute* changes to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, *CommandAborted* will be TRUE for only one scan cycle and immediately turn to FALSE.
Error		*Execute* turns from TRUE to FALSE. (Error Code is cleared.)

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	

- **Timing Diagram of Output Parameter Changes**

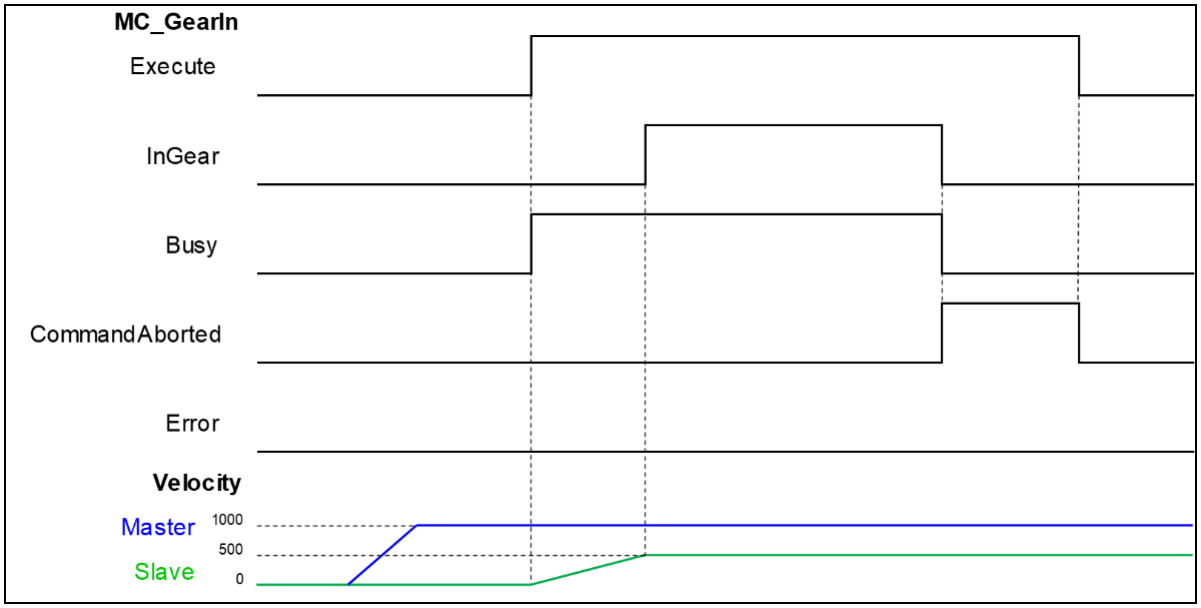


- **Inputs/Outputs**

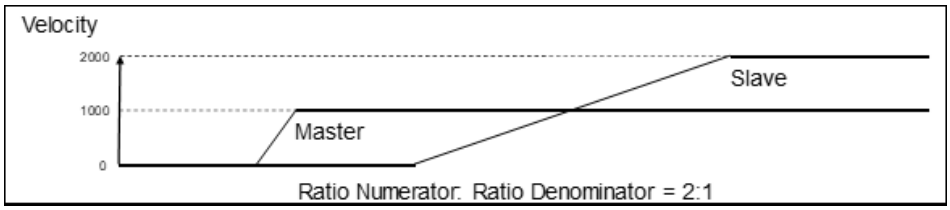
Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	Specifies the master axis number.	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.
Slave	Specifies the slave axis number.	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.

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

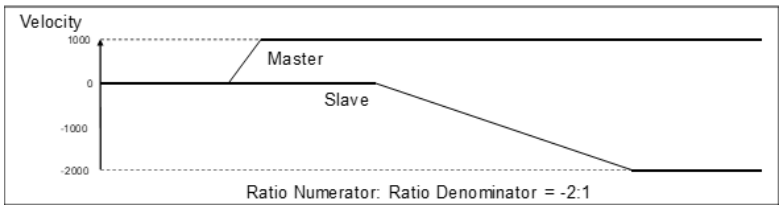
- **Function**
  - In MC_GearIn, the slave axis will follow master axis's Set Position.
  - When *Execute* is TRUE, the target speed of slave axis is the velocity of master axis times gear ratio (*Velocity* * *RatioNumerator* / *RatioDenominator*).



- After the gear mechanism is set, the slave axis will follow the master axis to move at the given proportion to accomplish the synchronized control of master and slave axis. The master axis could be physical or virtual axis or external encoder.
  - 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.



- Depending on where this function block is placed in the program, the results of the synchronous motion are different. For more information, see *AX-3 Series Operation Manual* section 7.7.2.5 Synchronous Control.

- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Supported Devices**
  - AX-series motion controller

2.1.1.16 MC_GearOut

MC_GearOut disengages 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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

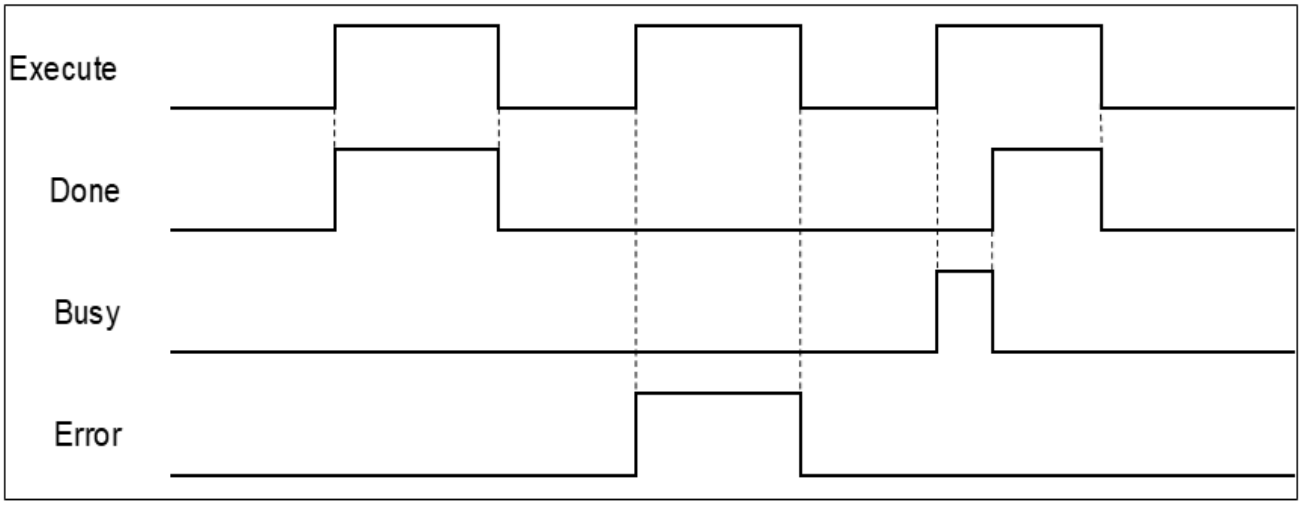
Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when the gear disconnection is complete	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The gear is disengaged.	• *Execute* changes to FALSE. • If *Execute* is FALSE and *Done* turns to TRUE, *Done* will be TRUE for only one scan cycle and immediately turn to FALSE. • *Error* turns to TRUE.
Busy	Performs the gear disengaging.	*Done* turns to TRUE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



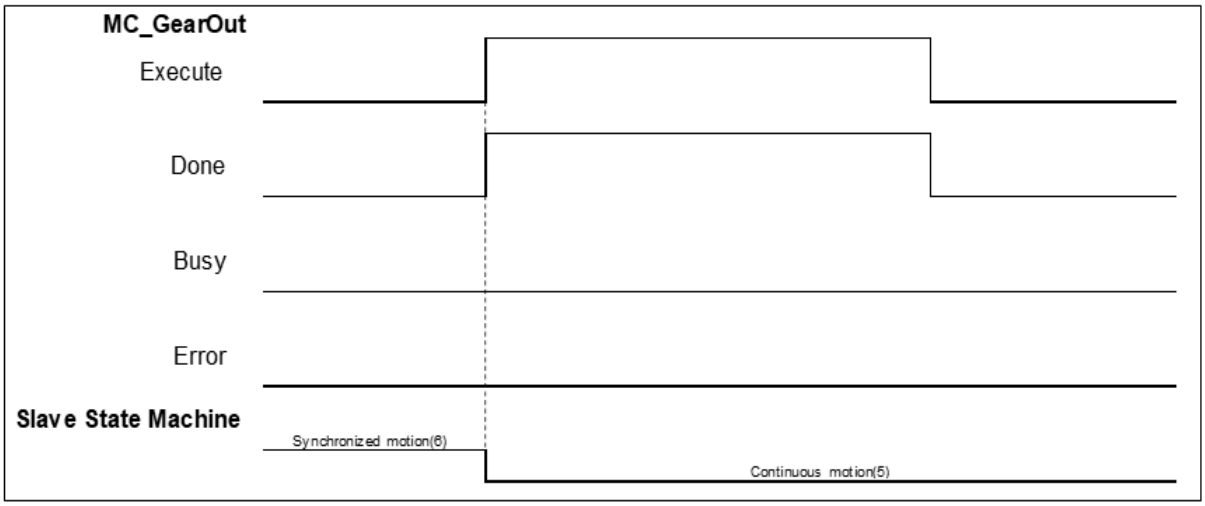
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Slave	The specified 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 the function block.

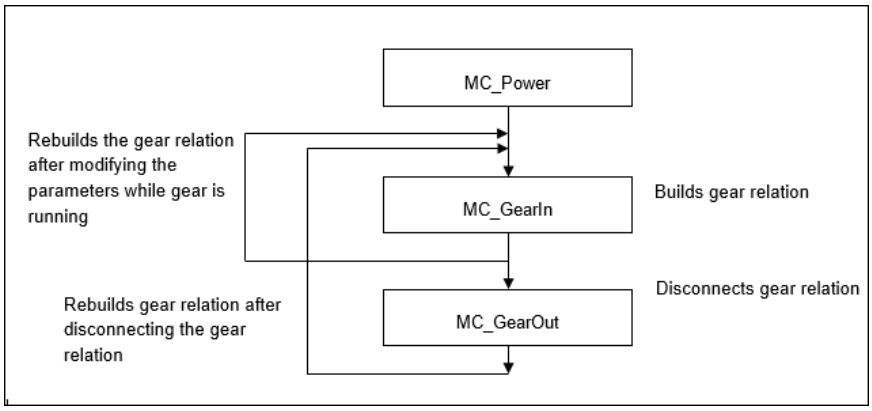
- **Function**

  - After the gear is disengaged, the slave axis will keep the moving speed. The state machine will be in the state of ContinuousMotion (not related to the slave axis).



  - After the slave axis is disengaged, and the velocity is zero, the status will be continuous_motion and remain unchanged.

- The following is the sequence for executing instructions related to electronic gear.

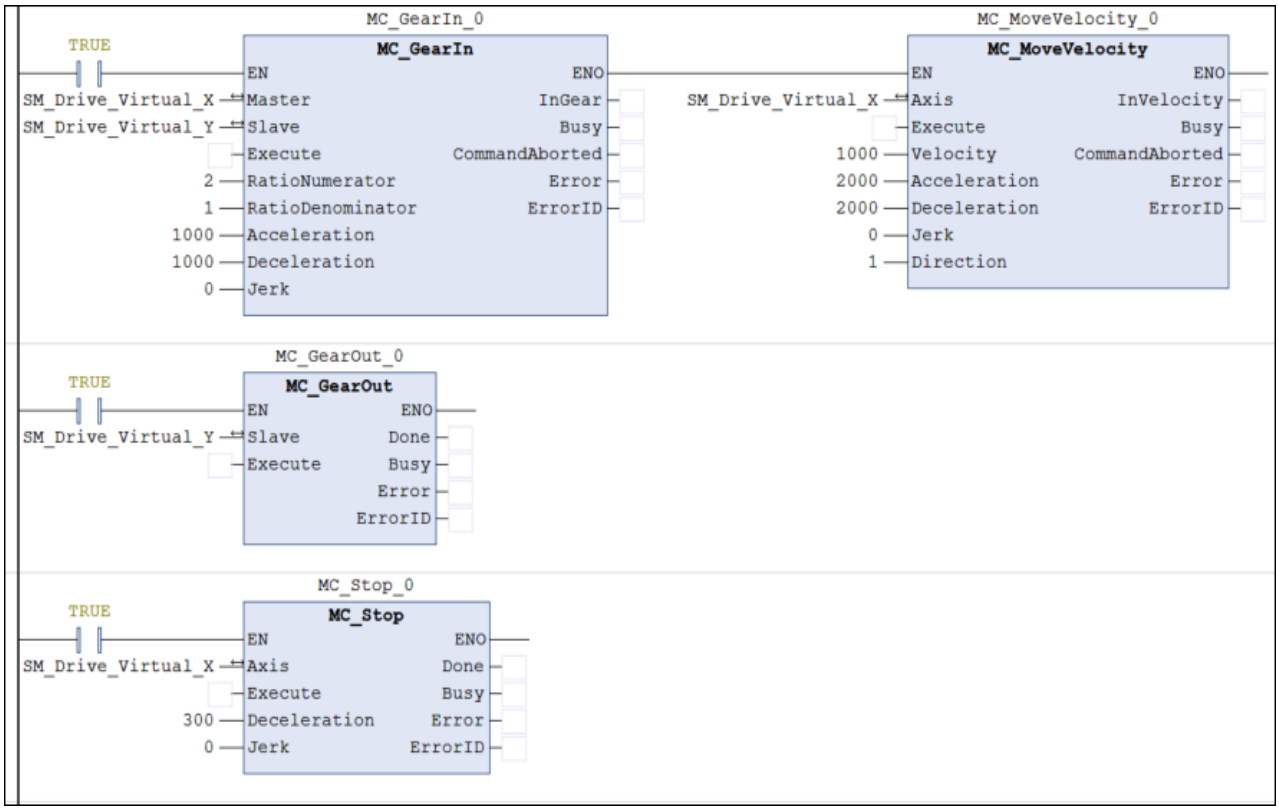


- **Troubleshooting**

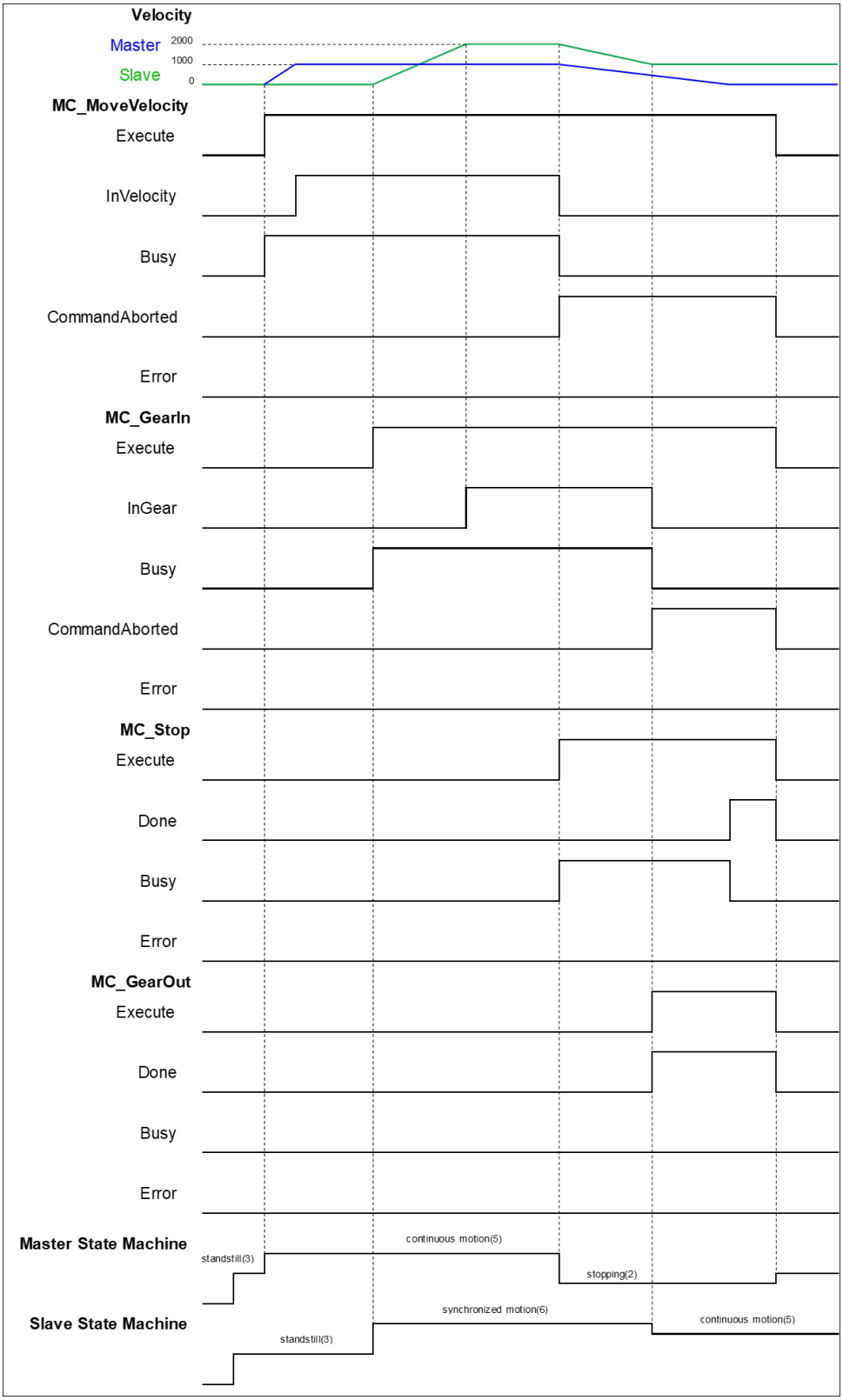
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

  This example describes the execution sequence of gear-related instructions the corresponding motion states.



◦ Timing Diagram



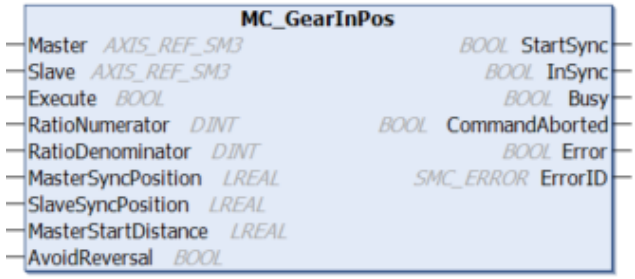
- When *Execute* of MC_MoveVelocity changes to TRUE, the master axis starts to move.
- When *Execute* of MC_GearIn changes to TRUE, the slave axis starts to catch the master axis.
- If the velocity of salve axis reaches 2 times of the master axis' velocity (RatioNumerator: RatioDenominator = 2: 1), *InGear* 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 running MC_Stop, when *Execute* of MC_GearOut changes to TRUE, master and slave axes will be out of synchronization and the slave axis will keep moving at the speed (Continuous Motion state) when the gear relation is decoupled.

- **Supported Devices**
  - AX-series motion controller

2.1.1.17 MC_GearInPos

MC_GearInPos couples the master and the slave axis at the specific 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	Runs the instruction 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)	*Execute* turns to TRUE and *Busy* is FALSE.
RatioDenominator	Gear ratio denominator* between master and slave axis	UDINT	Positive or 0 (1)	*Execute* turns to TRUE and *Busy* is FALSE.
MasterSyncPosition	Master Position at which the axes are synchronized	LREAL	Negative, positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
SlaveSyncPosition	Slave Position at which the axes are synchronized	LREAL	Negative, positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
MasterStartDistance	Master Distance for synchronization procedure	LREAL	Negative, positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
AvoidReversal	Prohibit reversal	BOOL	TRUE/FALSE (FALSE)	*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**

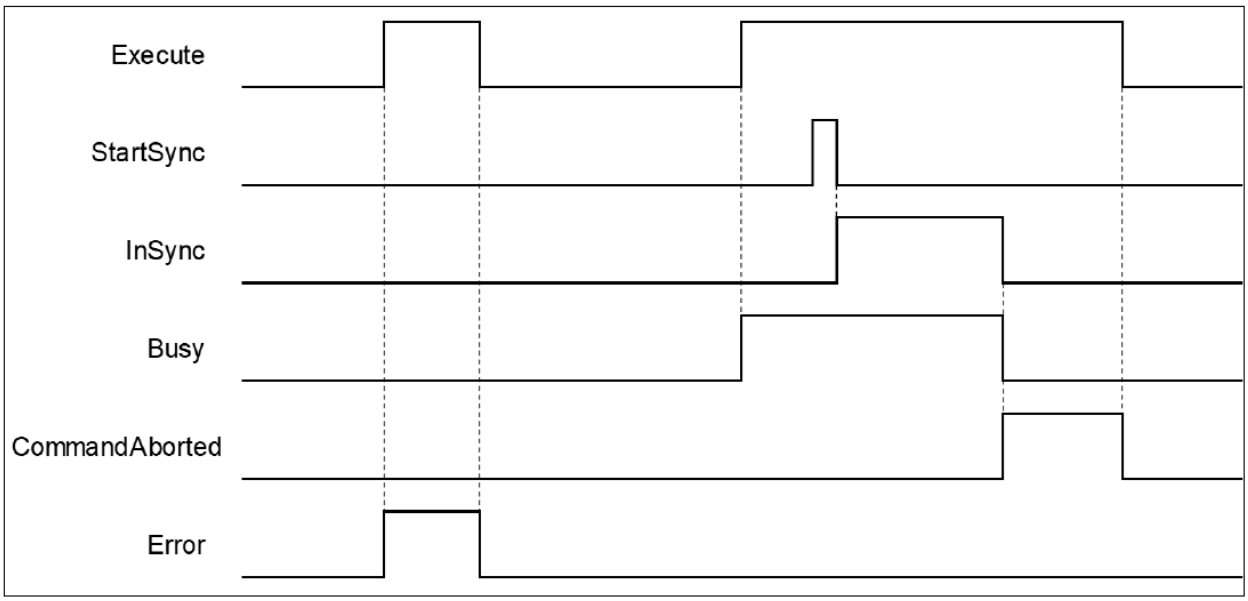
Name	Function	Data Type	Output Range (Default Value)
StartSync	TRUE when the gearing starts	BOOL	TRUE/FALSE (FALSE)
InSync	TRUE when the gearing is ongoing	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The gearing has been started.	• *InSync* turns to TRUE. • *Error* turns to TRUE.
InSync	The synchronization between master and slave axis is completed.	• *CommandAborted* turns to TRUE. • *Error* turns to TRUE.
Busy	The synchronization is ongoing.	• *CommandAborted* turns to TRUE. • *Error* turns to TRUE.
CommandAborted	• MC_GearOut is run. • The instruction is interrupted by another function block. • This instruction is aborted because of the MC_Stop instruction.	• *Execute* changes to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, *CommandAborted* will be TRUE for only one scan cycle and immediately turn to FALSE.
Error ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (ErrorID is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	Specifies the master axis.	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.
Slave	Specifies the slave axis.	AXIS_REF_SM3*	AXIS_REF_SM3	*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 the function block.

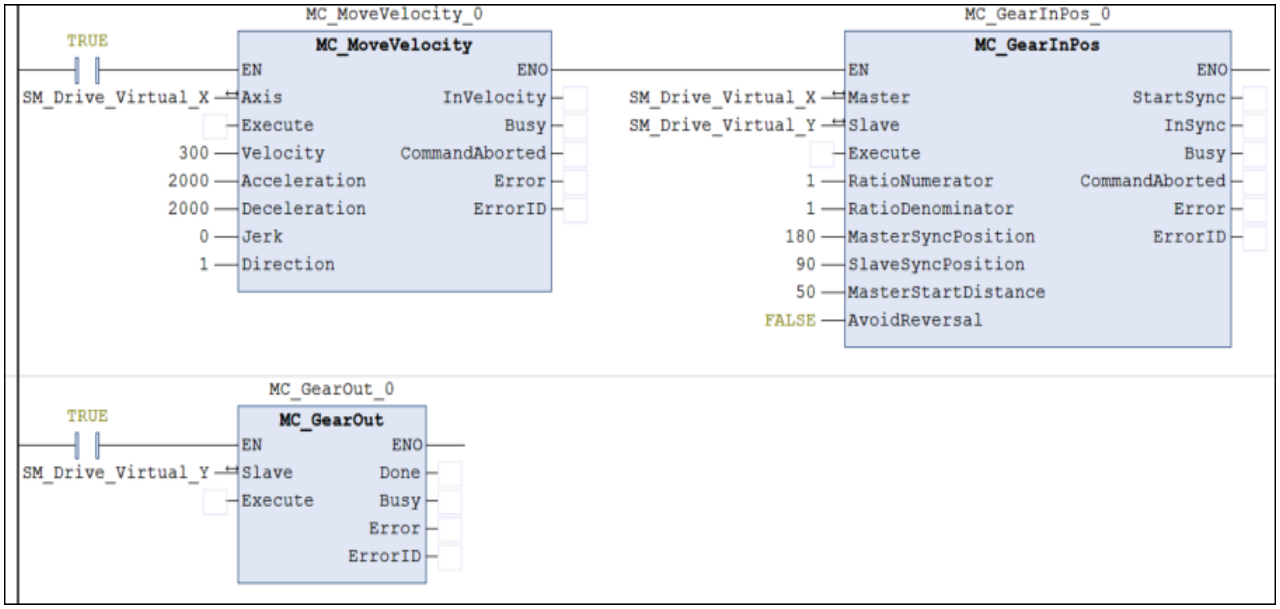
- **Function**

  - The position where the master axis runs StartSync = MasterSyncPosition–MasterStartDistance.
  - Ensure appropriate parameter settings of sync position if both master and slave axes are working under Finite mode. Suppose the master and slave axis are 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 axes operating under Modulo mode.
  - During the progress of synchronization between the master and slave axis, MC_GearInPos begins to plan the motion path of the slave axis automatically with gear ratio based on parameters of the position where the master axis runs *StartSync*, *MasterSyncPosition*, and *SlaveSyncPosition*. After synchronizing , the slave axis will start to move by following master axis.
  - When MasterStartDistance = 0 or a negative number, CAM is completed immediately.
  - When the master position does not pass the *MasterSyncPosition* setpoint, the slave axis will not perform synchronization. If the master position passes the *MasterSyncPosition* setpoint, the slave axis will perform synchronization.
  - Depending on where this function block is placed in the program, the results of the synchronous motion are different. For more information, see *AX-3 Series Operation Manual* section 7.7.2.5 Synchronous Control.

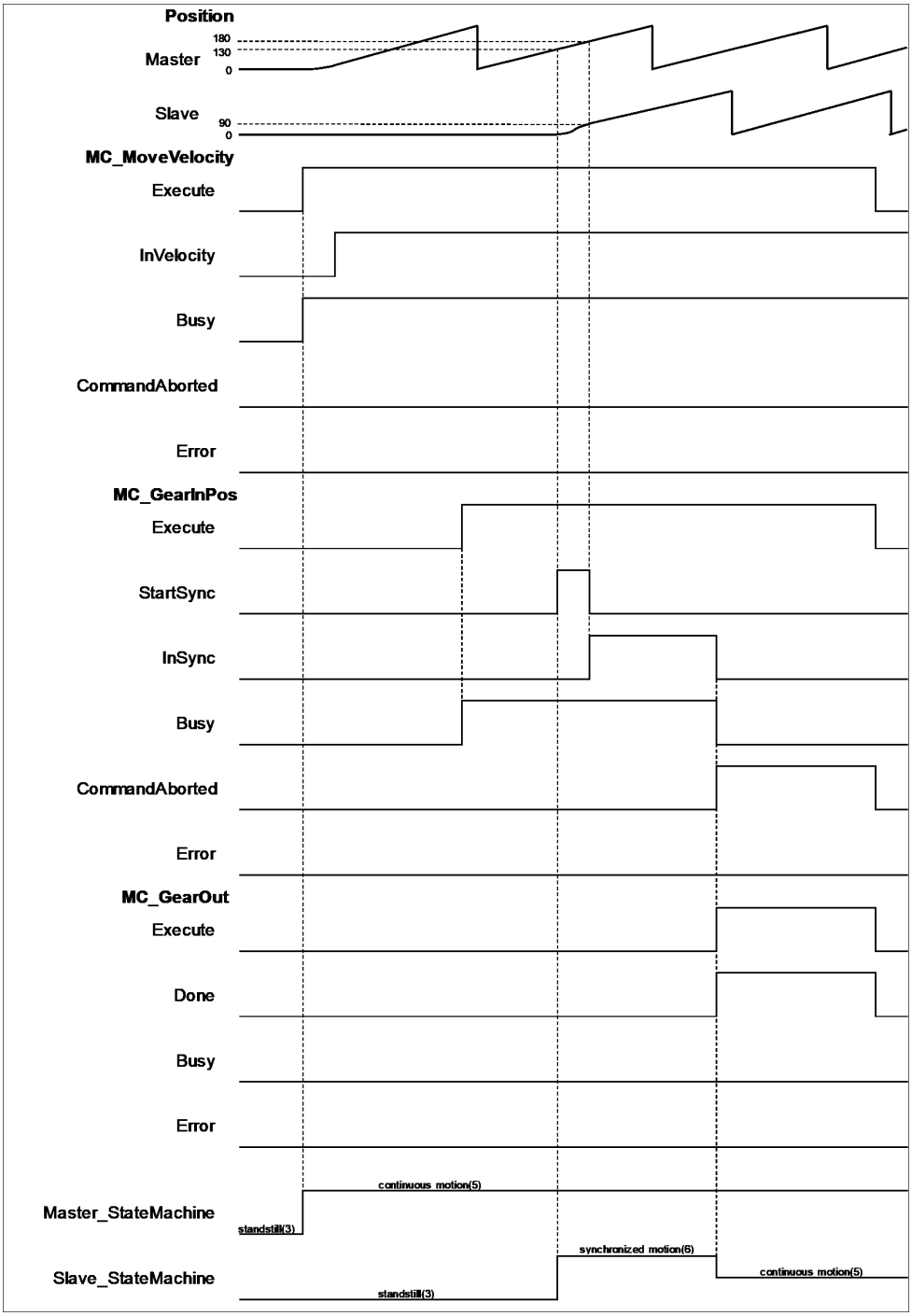
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

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



◦ Timing Diagram

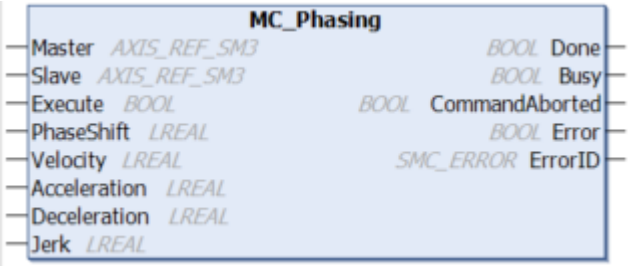


- When *Execute* of MC_MoveVelocity changes to TRUE, the master axis starts to move.
- When *Execute* of MC_GearInPos changes to TRUE, it's waiting for the master axis to reach the *StartSync* position.
- When the *StartSync* position is reached, *StartSync* of MC_GearInPos turns to TRUE. At the same time, a motion curve is planned for the slave axis to move, and the axis will be in the Synchronized Motion state.
- When both the master and slave axis reach the synchronization position, *InSync* of MC_GearInPos turns to TRUE, and *StartSync* changes to FALSE.
- When *Execute* of MC_GearOut turns to TRUE, the master and slave axes move asynchronously, entering the Continuous Motion state.

- **Supported Devices**
  - AX-series motion controller

2.1.1.18 MC_Phasing

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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
PhaseShift	Phase difference between the master and slave*	LREAL	Negative, positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Velocity	The maximum velocity for reaching the phase difference (Unit: user unit/s)	LREAL	Positive or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Acceleration	The Maximum acceleration for reaching the phase difference (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Deceleration	The maximum deceleration for reaching the phase difference (Unit: user unit/s^2)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Jerk	The max jerk for reach the phase difference (Unit: user unit/s^3)	LREAL	Positive (0)	*Execute* turns to TRUE and *Busy* is FALSE.

***Note**: The positive values indicate that the slave axis is behind the master axis. The negative values indicate that the slave axis is ahead of the master axis.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when phase compensation is completed	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
CommandAborted	TRUE when the instruction is interrupted	BOOL	TRUE/FALSE (FALSE)

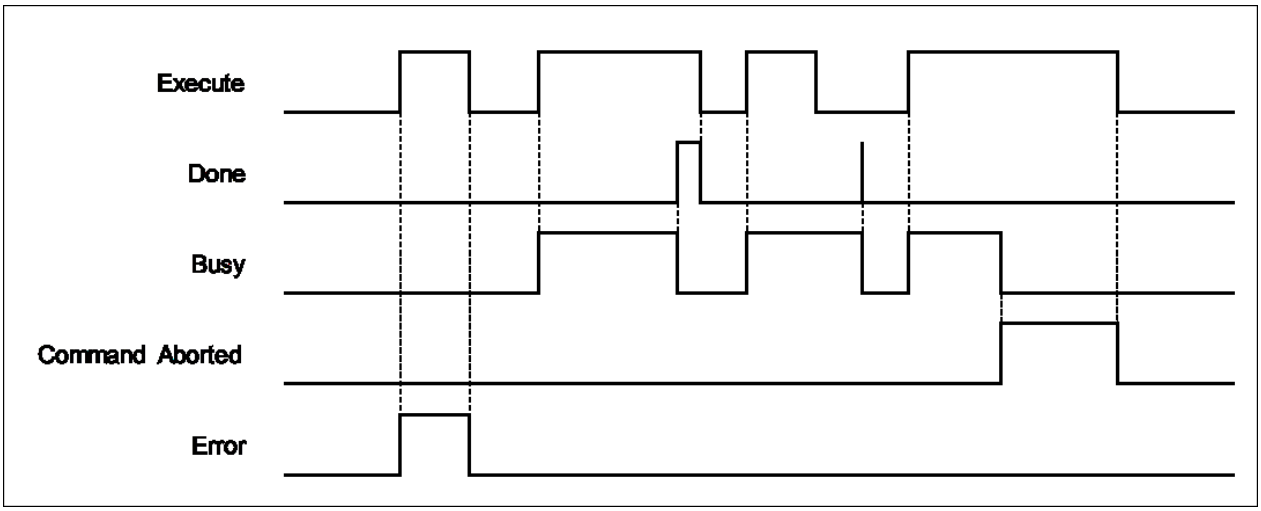
Name	Function	Data Type	Output Range (Default Value)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

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

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Done	The phasing operation is completed.	• *Execute* changes to FALSE. • If *Execute* is FALSE and *Done* turns to TRUE, *Done* will be TRUE for only one scan cycle and then immediately turn to FALSE.
Busy	The phasing operation is running.	• *Error* turns to TRUE. • *CommandAborted* turns to TRUE.
CommandAborted	• TRUE when the instruction is aborted • TRUE when this instruction is aborted by the MC_Stop instruction	• *Execute* changes to FALSE. • If *Execute* is FALSE and *CommandAborted* turns to TRUE, *CommandAborted* will be TRUE for only one cycle and then immediately turn to FALSE.
Error	An error occurs during running the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (ErrorID is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**

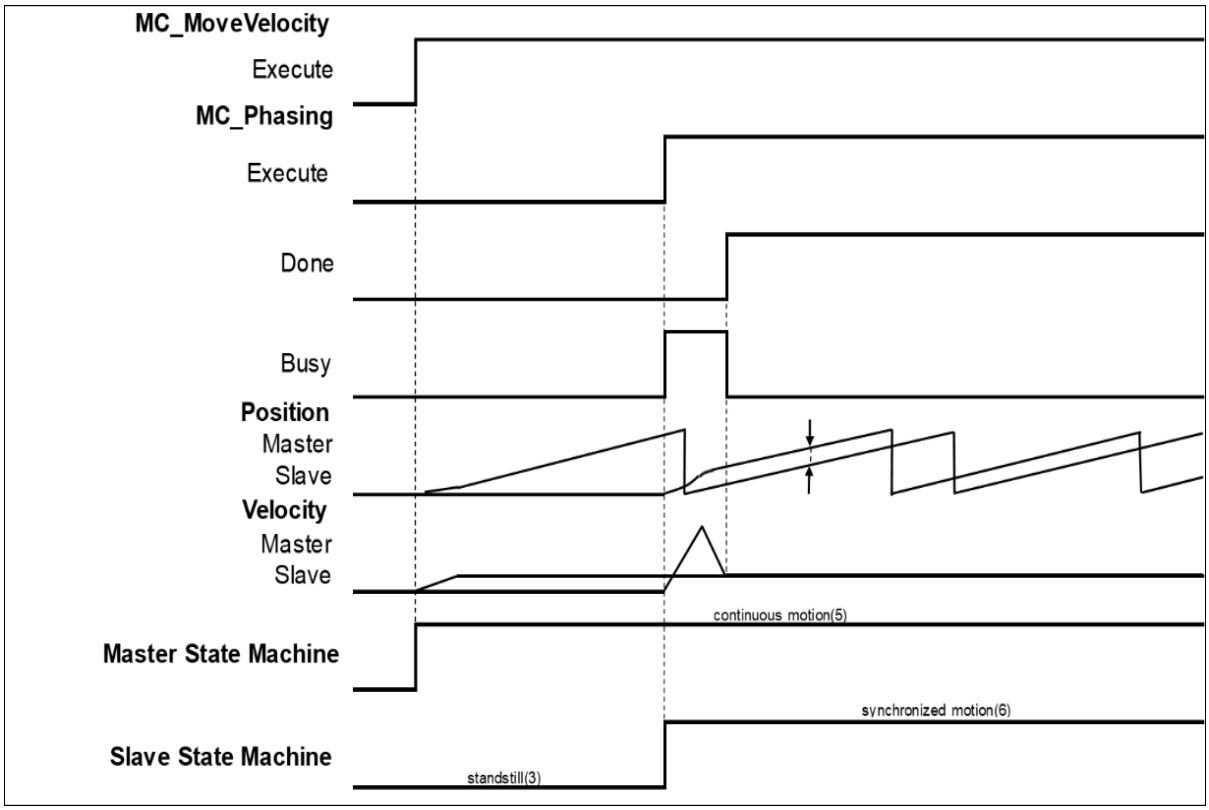


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	Specifies the master axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.
Slave	Specifies the slave axis	AXIS_REF_SM3*	AXIS_REF_SM3	*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 the function block.

- **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. If the *PhaseShift* value is positive, 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.

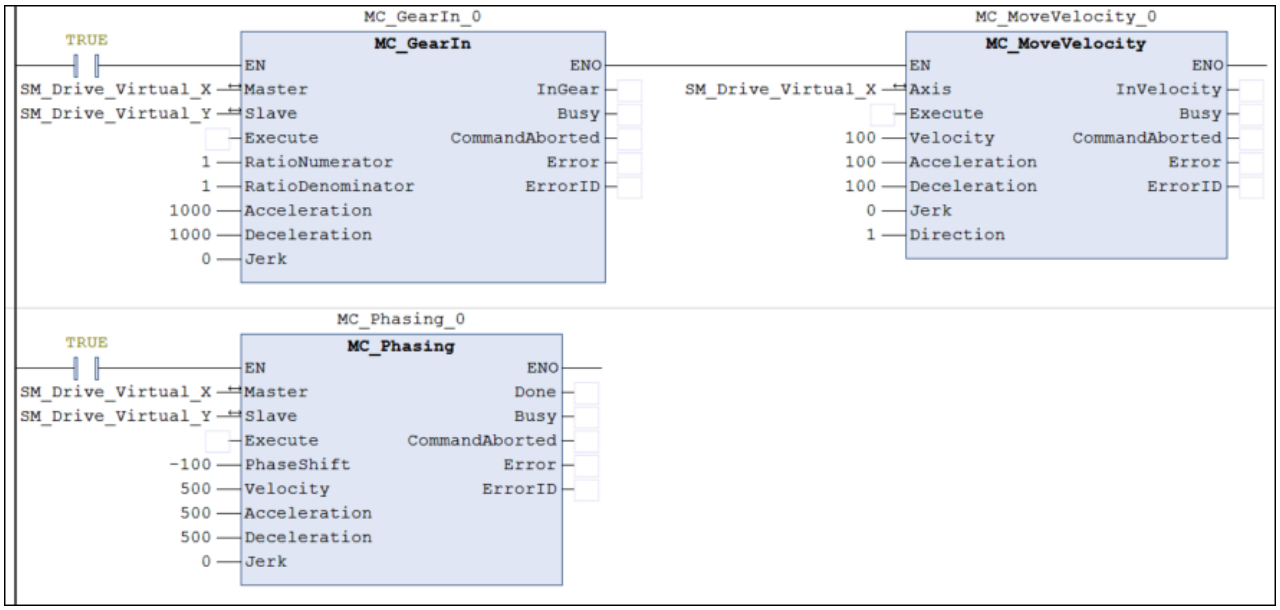


  - MC_Phasing can be used when the state is not under Synchronized motion.
    - When MC_Phasing is run, the state of the slave axis will remain at Synchronized motion.
    - When MC_Phasing is executed by the slave axis before establishing a gear relationship between the master and slave axes, the slave axis will be directly synchronized with the master axis, and both move based on the gear ratio of 1: 1.
    - When the slave axis runs MC_Phasing, it can be interrupted by other single-axis function blocks, and the synchronous relationship will be disconnected.
  - Depending on where this function block is placed in the program, the results of the synchronous motion are different. For more information, see *AX-3 Series Operation Manual* section 7.7.2.5 Synchronous Control.

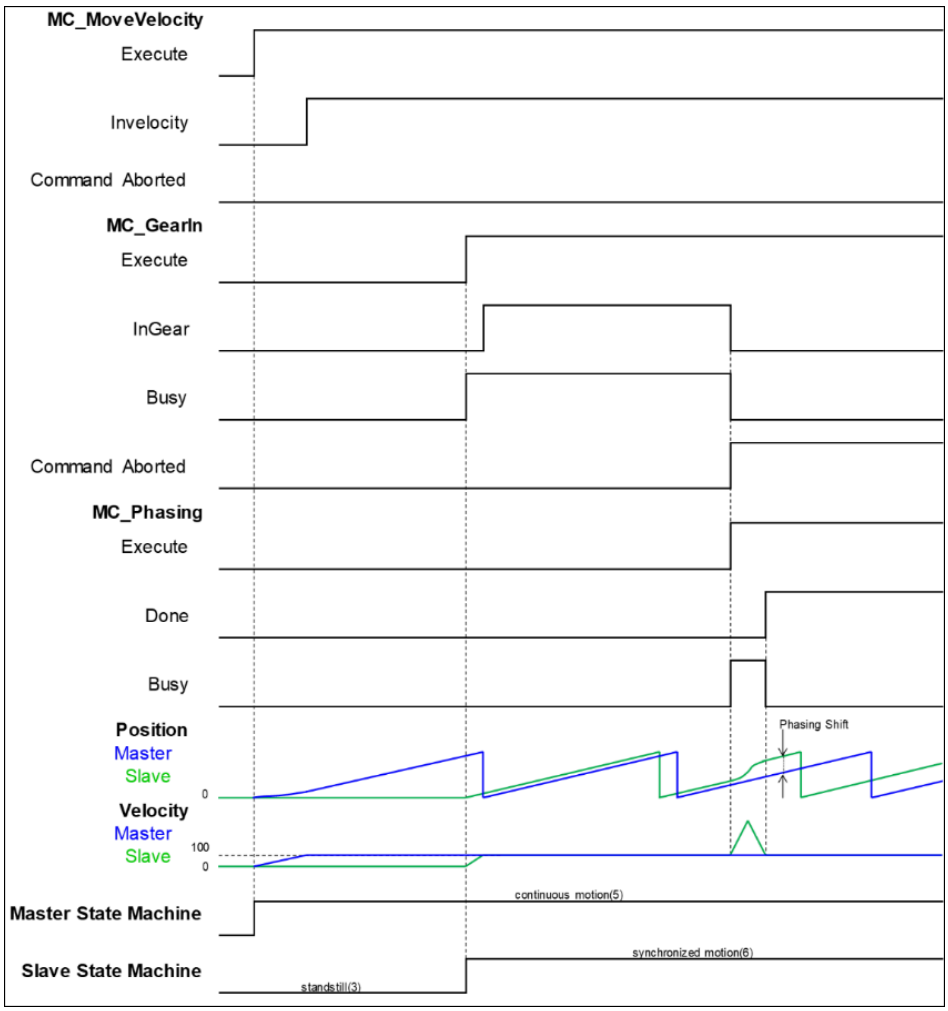
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

This example describes the corresponding motion state and path throughout the gear operation.



- Timing Diagram



- Run MC_MoveVelocity to make the master axis run at a constant velocity, and then run MC_GearIn to establish a gear relationship between the master and slave axes.
- 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.

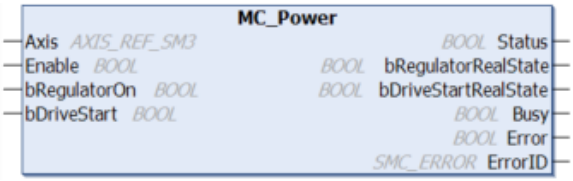
- **Supported Devices**
  - AX-series motion controller

2.1.2 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 SM3_Drive_ETC 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 Operation Manual*.

2.1.2.1 MC_Power

MC_Power enables or disables the 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	Runs the instruction when *Enable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
bRegulatorOn	Enables the power.	BOOL	TRUE/FALSE (FALSE)	Only when *Enable* = TRUE
bDriveStart	Disables the Quickstop mechanism.	BOOL	TRUE/FALSE (FALSE)	Only when *Enable* = TRUE.

- **Outputs**

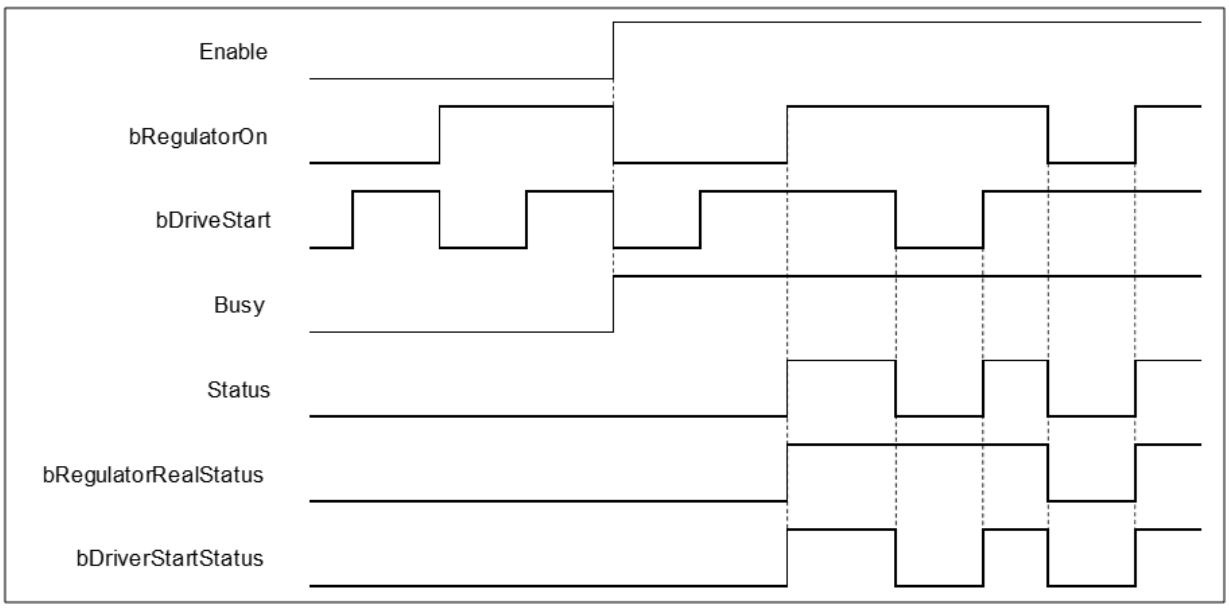
Name	Function	Data Type	Output Range (Default Value)
Status	The specific axis is ready to be moved	BOOL	TRUE/FALSE (FALSE)
bRegulatorRealState	The power is turned ON.	BOOL	TRUE/FALSE (FALSE)
bDriveStartRealState	Quickstop function is available to the device.	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	*Enable*, *bRegulatorRealState,* and *bDriveStartRealState* are TRUE.	• When *Enable* is TRUE, *bRegulatorRealState* or bDriveStartRealState shifts to FALSE • *Error* turns to TRUE.
bRegulatorRealState	*Enable* and *bRegulatorRealState* are TRUE.	• When *Enable* is TRUE, *bRegulatorRealState* turns to FALSE • *Error* turns to TRUE.
bDriveStartRealState	*Enable*, *bRegulatorRealState*, and *bDriveStartRealState* are TRUE.	• When *Enable* is TRUE, *bRegulatorRealState* or *bDriveStartRealState* shifts to FALSE • *Error* turns to TRUE.
Busy	*Enable* turns to TRUE.	• *Enable* turns to FALSE. • *Error* turns to TRUE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	Errors are cleared.
ErrorID		

- **Timing Diagram of Output Parameter Changes**

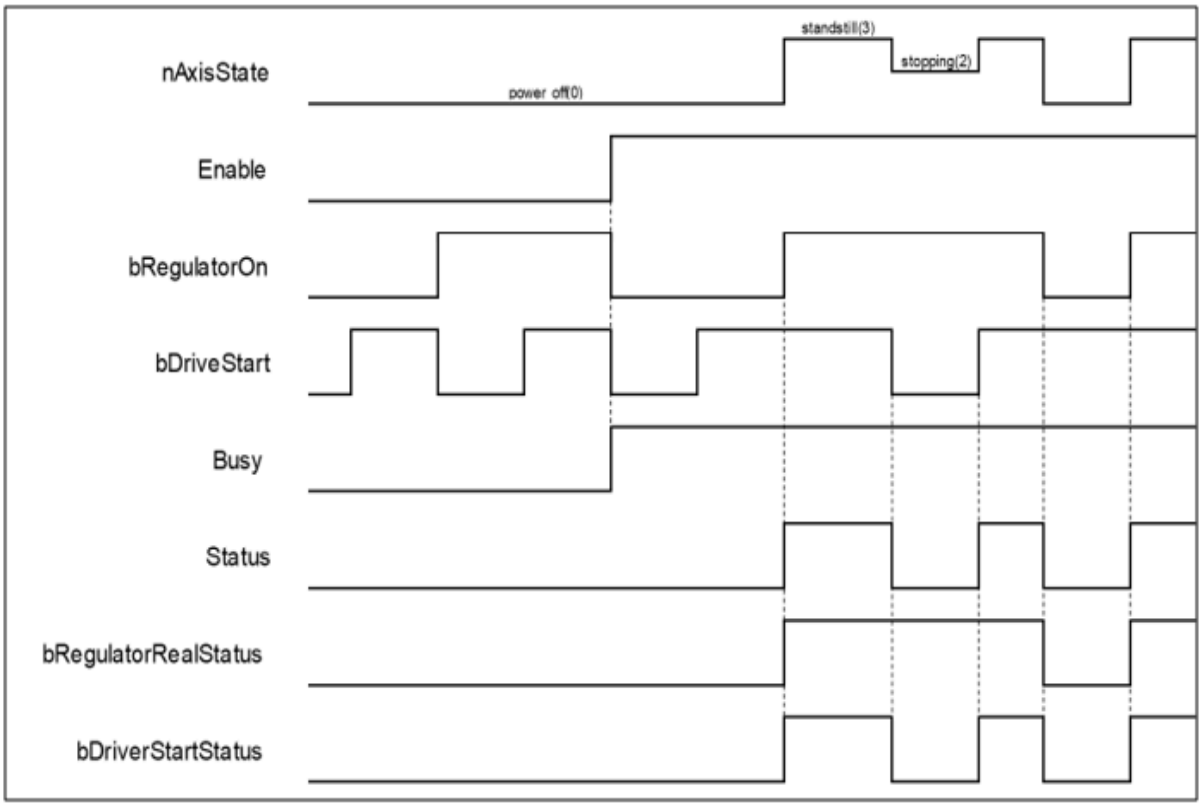


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Enable* turns to TRUE.

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

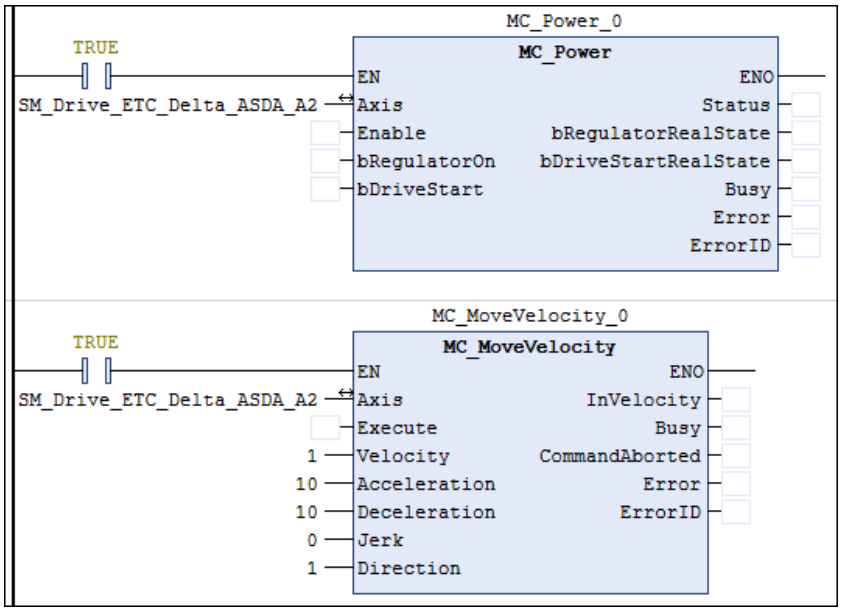
- **Function**
  - When *Enable* is TRUE, *bRegulatorOn* and *bDriveStart* are effective.
  - When *Enable*, *bRegulatorOn*, and *bDriveStart* are TRUE. *Status* turns to TRUE and *nAxisState* turns to Standstill.
  - When *Enable* and *bRegulatorOn* are TRUE, *DriveStart* is set to FALSE , and *nAxisState* (axis status) turns to Stopping.
  - When *Enable* and *bDriveStart* are TRUE, *RegulatorOn* is set to FALSE, and *nAxisState* (axis status) turns to Disabled.
  - When using SoftMotion version V4.10.0.0, the axis state machine switches among Standstill → Stopping → Standstill. It is recommended to use MC_Power.Status to check whether the axis state machine is enabled.
  - Timing Diagram



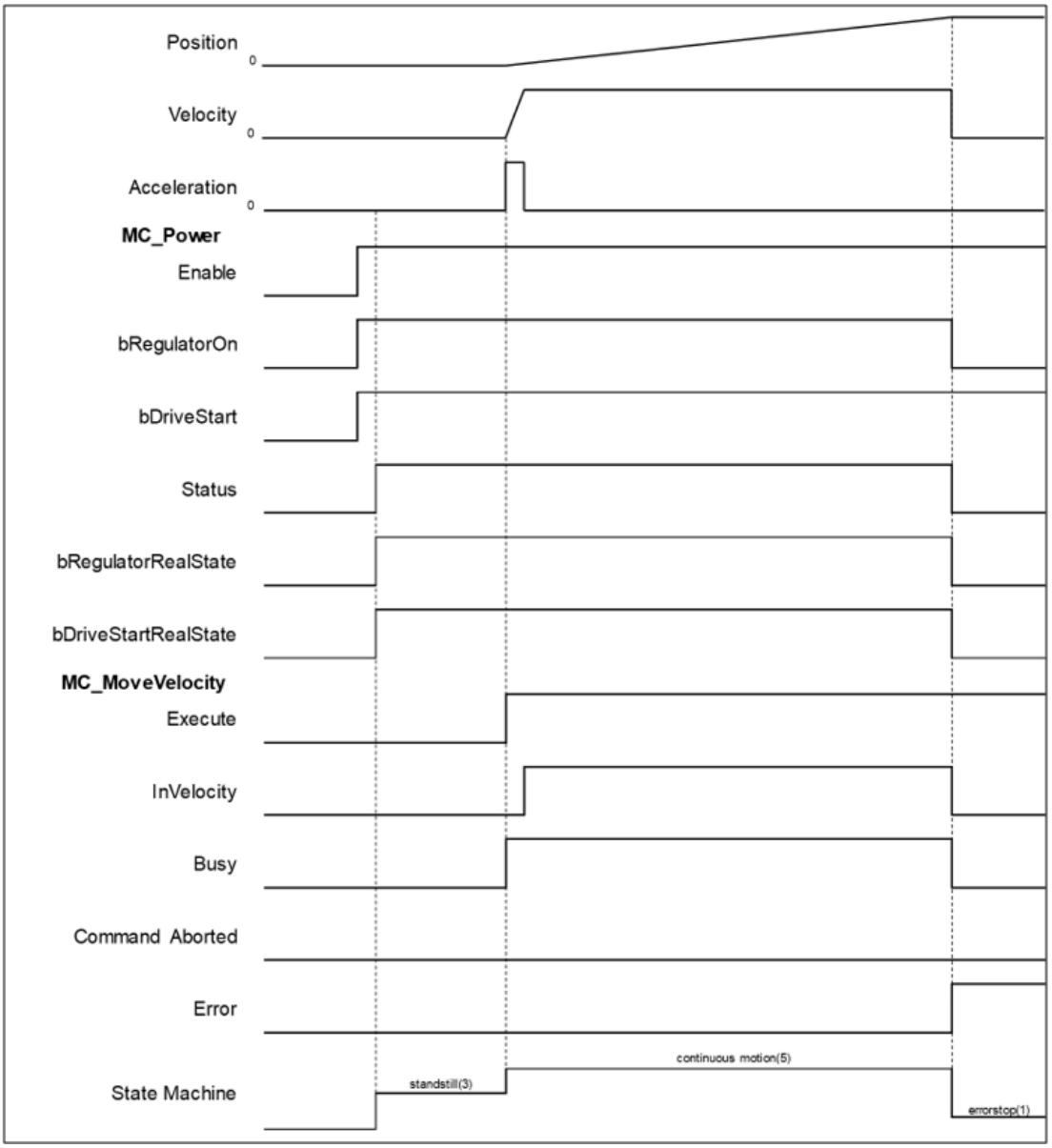
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

  - **Example 1**: The following example explains the motion of a moving axis when bRegulatorOn of MC_Power turns to FALSE.



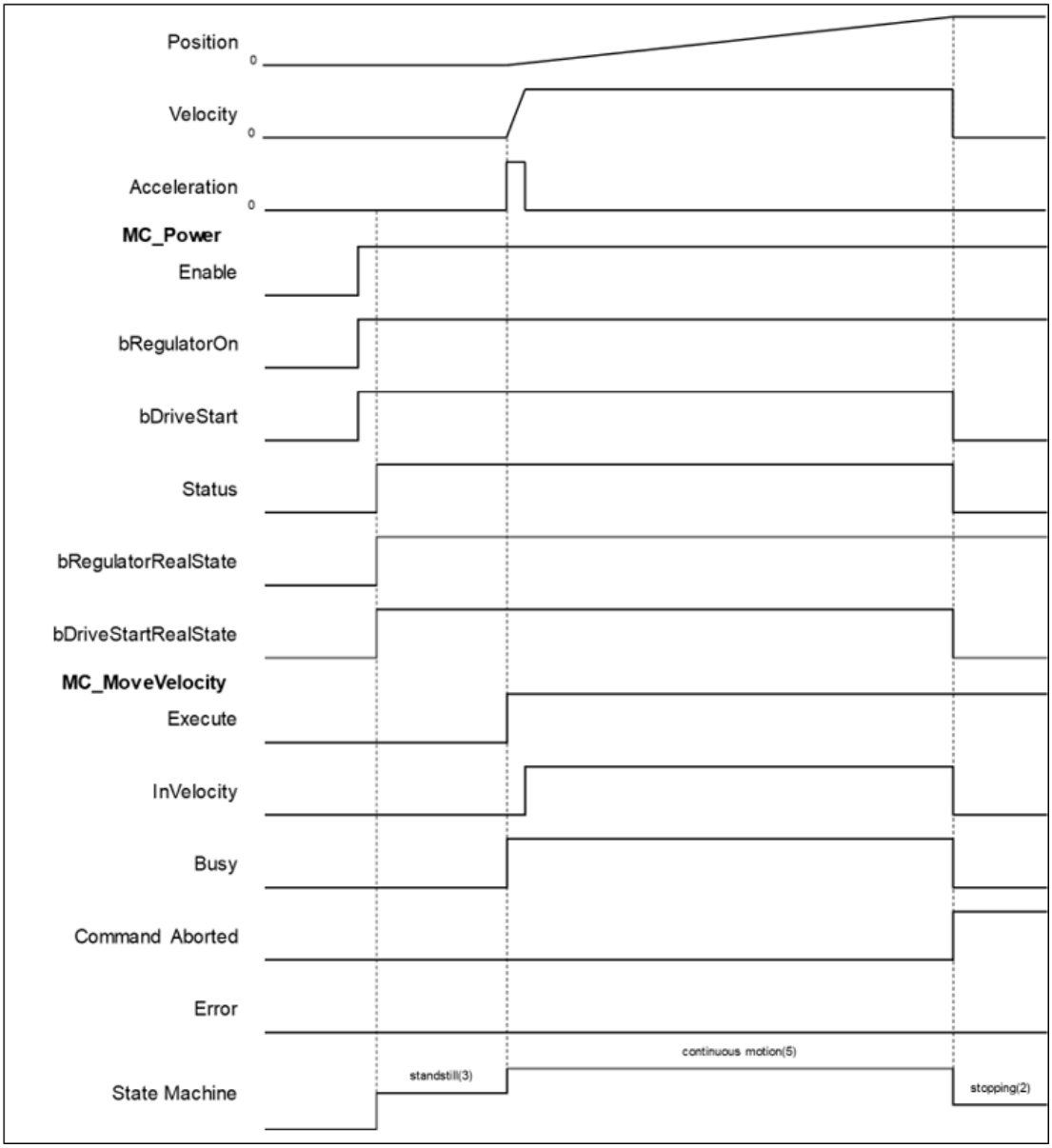
    - Timing Diagram



- Change *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 switches from continuous_motion to errorstop.

◦ **Example 2**: Continue with **Example 1**. The following example explains the axis motion when *bDriveStart* of MC_Powers turns to FALSE.

- Timing Diagram



- Change *bDriveStart* of MC_Power from TRUE to FALSE while the axis is moving. The axis will stop immediately.
- When *CommandAborted* turns to TRUE and MC_MoveVelocity is interrupted, the axis status changes from continuous_motion to stopping.

• **Supported Devices**

◦ AX-series motion controller

2.1.2.2 MC_SetPosition

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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (TRUE)	-
Position	Axis position (User-defined unit)	LREAL	Positive, negative, or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Mode	Relative position (TRUE) or the absolute position (FALSE)	BOOL	TRUE/FALSE (FALSE)	*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 within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

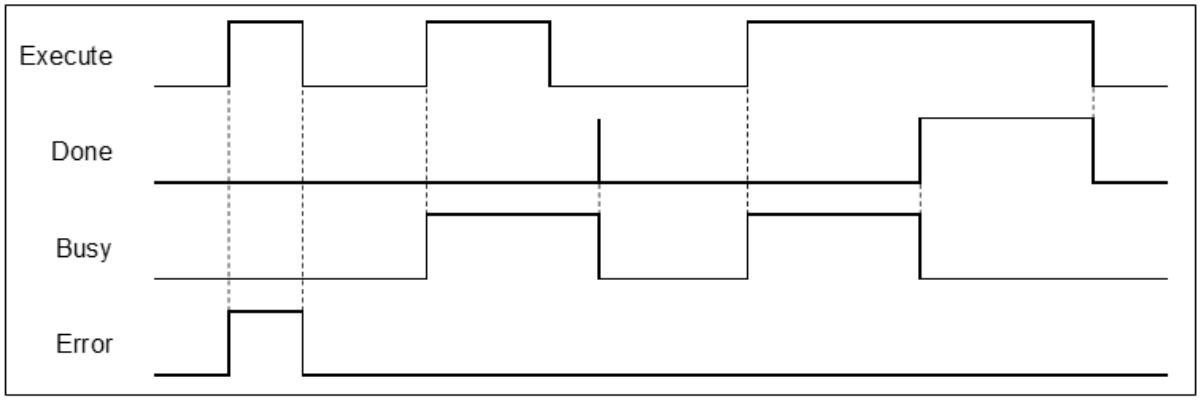
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Done	The coordinate modification is complete.	• *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 turn to FALSE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Busy	*Execute* is triggered.	• *Done* turns to TRUE. • *Error* turns to TRUE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

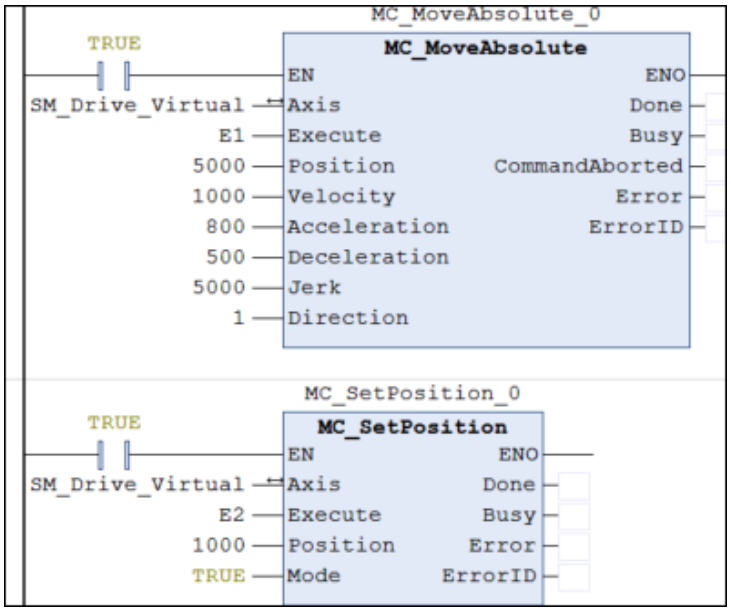
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.

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

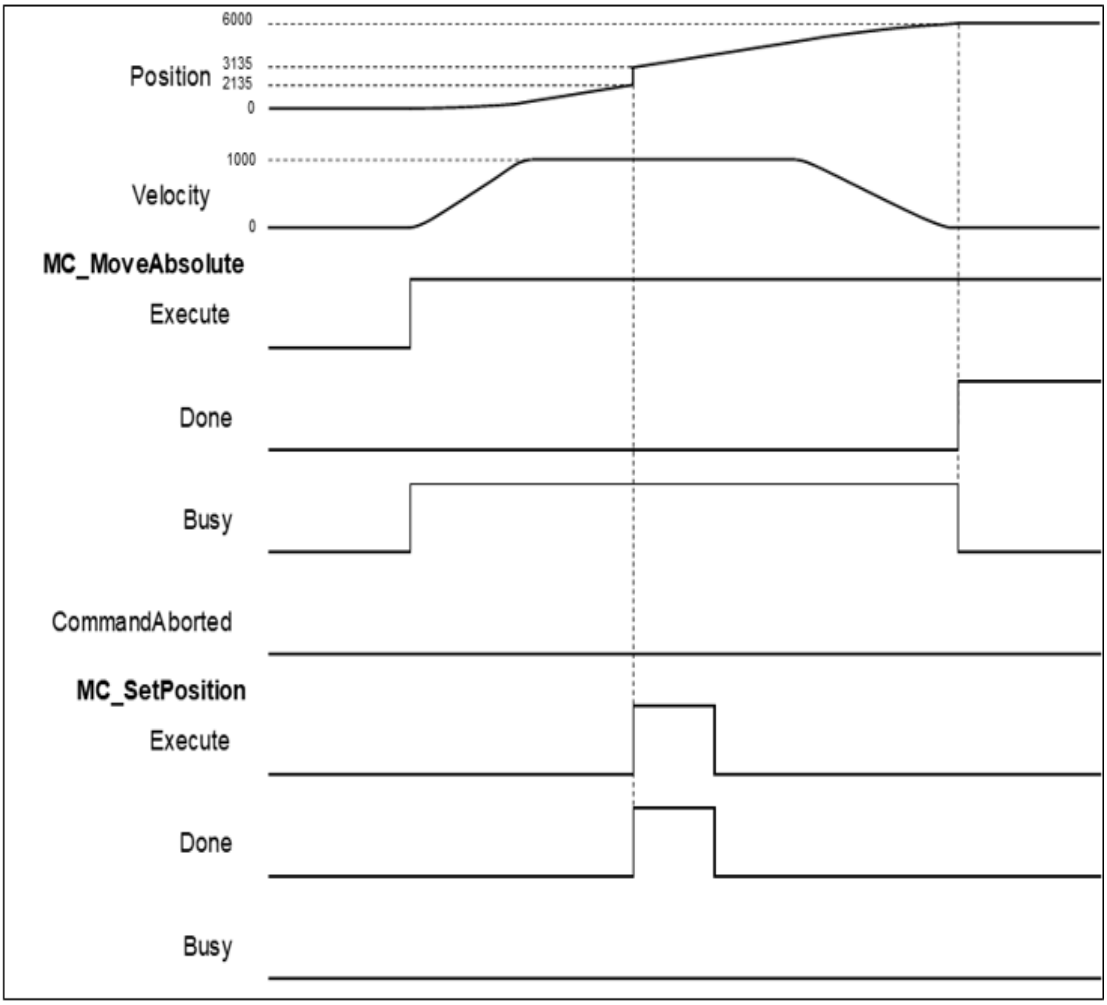
- **Function**
  - When the axis position parameters are set via MC_Position, the axis does not move, but only the coordinate system moves.
  - To avoid possible position jumps occurring to the slave axis, you should avoid running MC_SetPosition to the synchronized master axis.
  - In the relative mode, the value of *Position* will be added to the current position to make the new coordinate position. In the absolute mode, the value of *Position* will be set to the current coordinate position.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

  - **Example 1**: The following example describe the situation when MC_MoveAbsolute is executed and MC_SetPosition is used to change the coordinate system by selecting the relative mode.

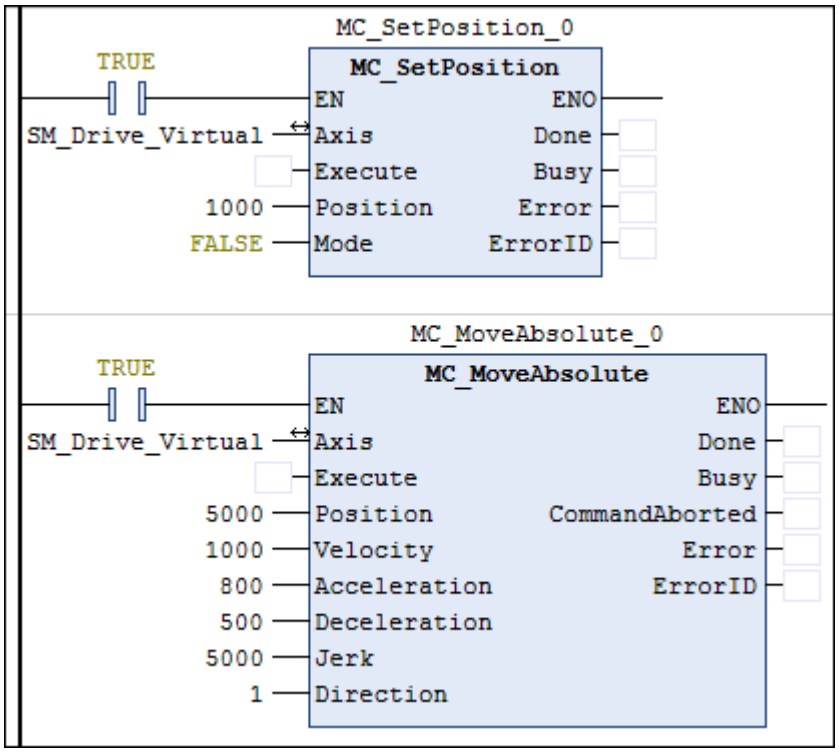


    - Timing Diagram

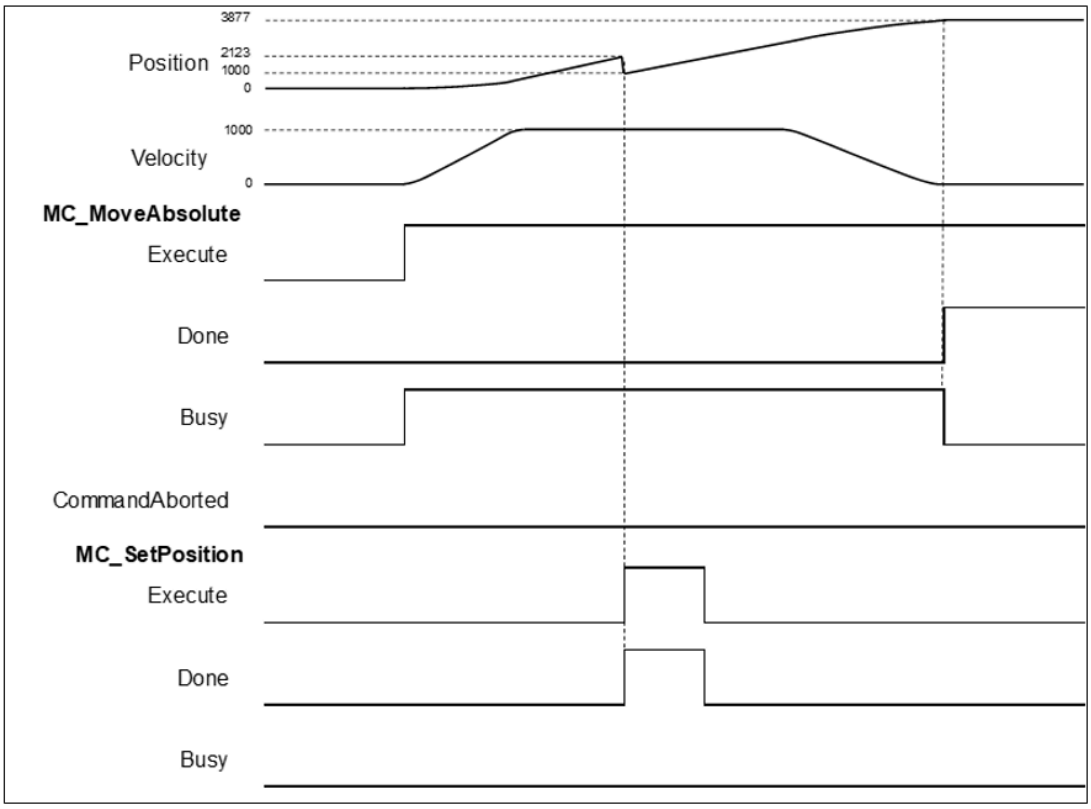


- After MC_MoveAbsolute is executed, use MC_SetPosition to define a new coordinate system under the relative mode.
- When the *Execute* of MC_SetPosition is TRUE, the new position of the axis will be changed from 2135 to 3135 after adjusting the coordinate system.
- When *Done* of MC_MoveAbsolute is TRUE, the current position will be 6000. At this time, MC_MoveAbsolute still moves to 5000 on the old coordinate system, and the position will be 6000 after the coordinate adjustment.

◦ **Example 2**: The following example describe the situation when MC_MoveAbsolute is executed, and MC_SetPosition is used to change the coordinate system by selecting the absolute mode.

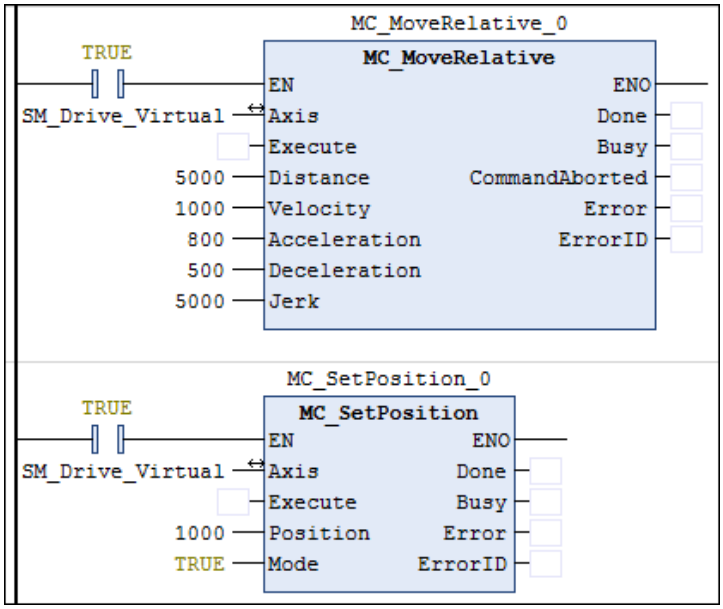


- Timing Diagram

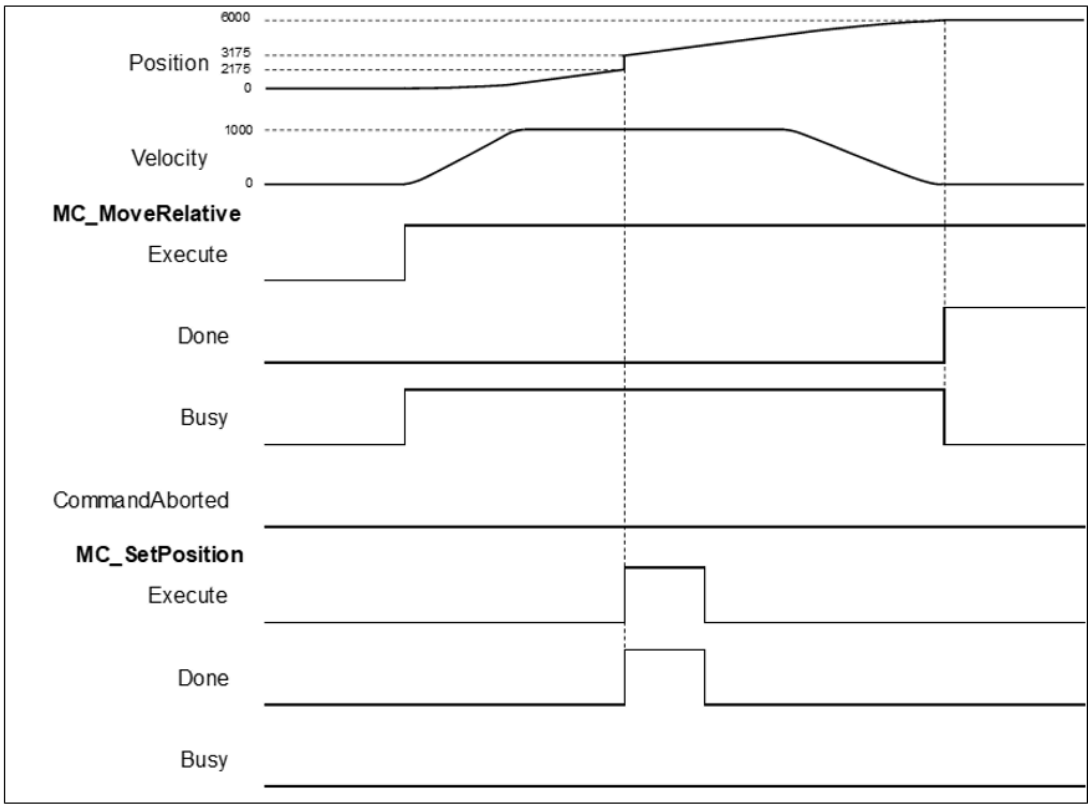


  - After MC_MoveAbsolute runs, use MC_SetPosition to define a new coordinate system under absolute mode.
  - When the of *Execute* of MC_SetPosition changes to TRUE, the new position of the axis will change to 1000 from 2123 after adjusting the coordinate system by moving a distance of 1123.
  - When *Done* of MC_MoveAbsolute is TRUE, the current position will be 3877 (5000–1123). At this time, MC_MoveAbsolute still moves to 5000 on the old coordinate system, and the position is 3877 after the coordinate adjustment.

- **Example 3**: The following example describes the situation when MC_MoveRelative is executed, MC_SetPosition is used to change the coordinate system by selecting the relative mode.



- Timing Diagram

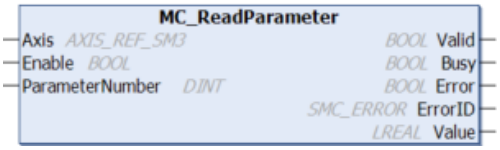


  - After MC_MoveRelative runs, use MC_SetPosition to define a new coordinate system under relative mode.
  - When *Execute* of MC_SetPosition changes to TRUE, the new position of the axis will be changed from 2175 to 3175 after adjusting the coordinate system.
  - When *Done* of MC_MoveAbsolute is TRUE, the current position will be 6000. At this time, MC_MoveAbsolute still moves to 5000 on the old coordinate system, and it will be 6000 after the coordinate adjustment.

- **Supported Devices**
  - AX-series motion controller

2.1.2.3 MC_ReadParameter

MC_ReadParameter reads the value of a 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	Runs the instruction when *Enable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
ParameterNumber	The number of the parameter to be read	DINT	Positive, negative, or 0 (0)	*Enable* turns to TURE

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Valid	TRUE when the parameter is available	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
Value	Value of the parameter to read	LREAL*2	Positive, negative, or 0 (0)

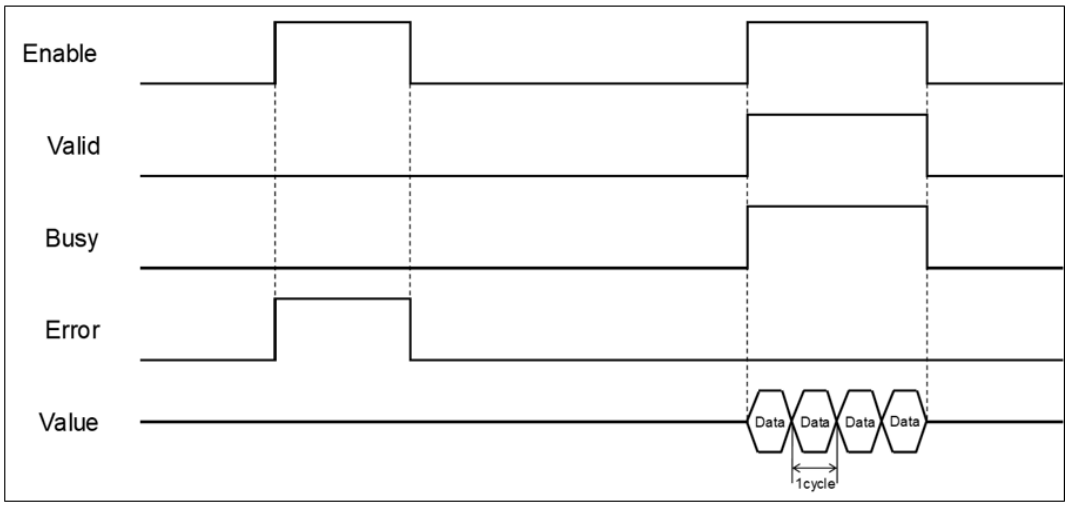
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Valid	• *Enable* is triggered to TRUE. • The parameter to read exists.	• When *Enable* turns from TRUE to FALSE. • *Error* turns to true.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Busy	• *Enable* changes to TRUE • The parameter is available.	• *Enable* turns from TRUE to FALSE. • *Error* turns to TRUE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		
Value	Continuously update when *Valid* is TRUE.	*Valid* is FALSE and stops updating.

- **Timing Diagram of Output Parameter Changes**



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

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	When *Enable* is TRUE

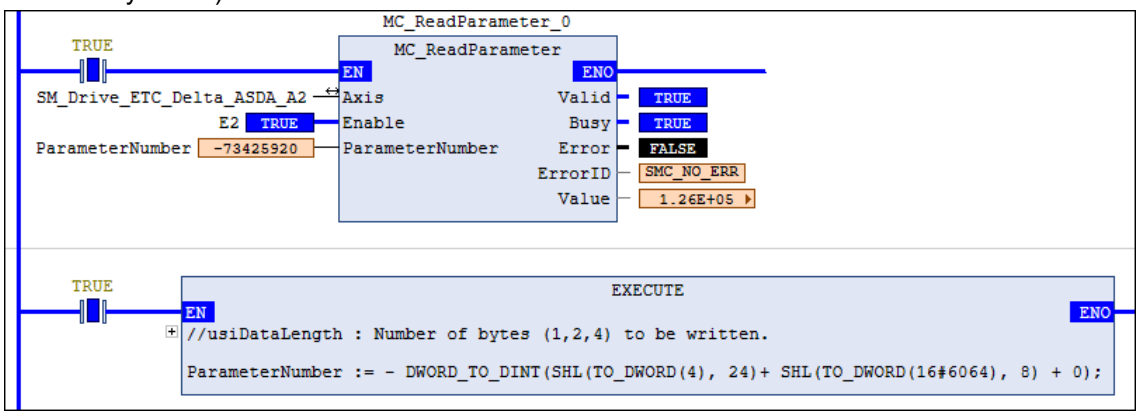
***Note**: AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.

- **Function**
  - To use MC_ReadParameter to read parameter ID in EtherCAT Object Dictionary:
    - Use SHL instruction to move the data length of the object to the left by 24 bits.
    - Use SHL instruction to move the index of the object to the left by 8 bits.
    - Sum up the above parameters and adds the subindexes. 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 need to enter the parameter ID of AXIS_REF_SM3 (FB) into the *ParameterNumber* input.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

  This example describes using 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 executed, 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

- **Supported Devices**

  - AX-series motion controller

2.1.2.4 MC_WriteParameter

MC_WriteParameter writes the value of a axis 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	Runs the instruction when *Enable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
ParameterNumber	The number of the parameters 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 instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

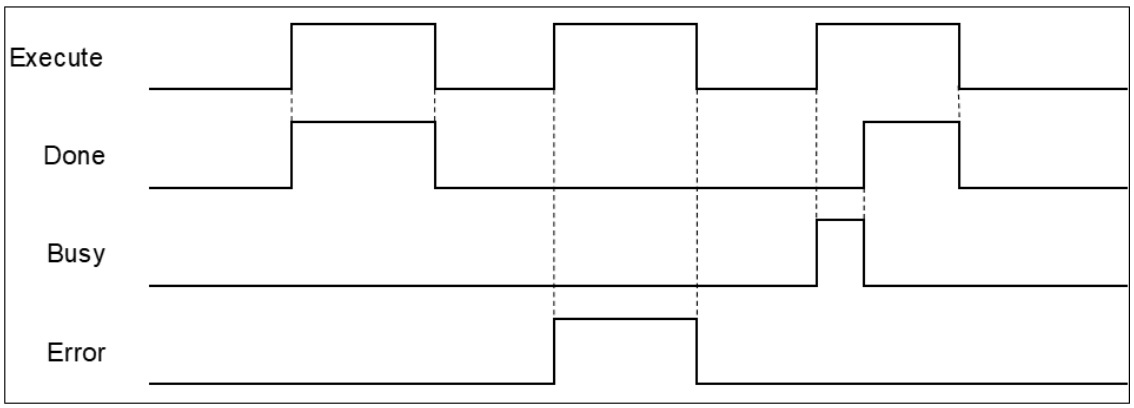
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Done	Writing the value is completed.	*Execute* turns from TRUE to FALSE.
Busy	• *Execute* turns to TRUE. • The value is being written to the parameter.	• *Done* turns to TRUE. • *Error* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

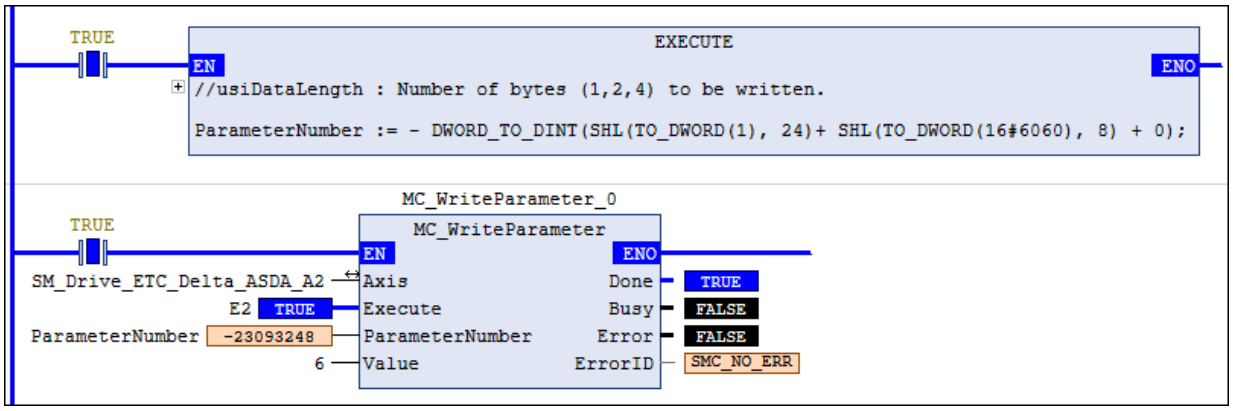
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.

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

- **Function**
  - To use MC_ReadParameter to read the parameter ID in EtherCAT Object Dictionary:
    - Use SHL instruction to move the data length of the object to the left by 24 bits.
    - Use SHL instruction to move the index of the object to the left by 8 bits.
    - Sum up the above parameters and add the subindexes. 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 need to input the parameter number of AXIS_REF_SM3 (FB) to *ParameterNumber*.
  - When the axis is moving, MC_WriteParameter will write parameter values to the input *fSetPosition* . The value of *fSetPosition* will be changed to the set value for only one EtherCAT task cycle. After that, *fSetPosition* will restore to its original planned motion curve.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

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



- Enter 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, 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

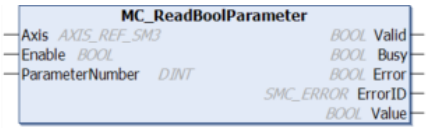
INDEX	6060_h
Name	Modes of operation
Object Code	VAR
Data Type	INTEGER8
Access	RW
PDO Mapping	Yes
Value Range	INTEGER8
Default Value	0
Comment	0: Reserved

- **Supported Devices**

  - AX-series motion controller

2.1.2.5 MC_ReadBoolParameter

MC_ReadBoolParameter reads the value of a 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	Runs the instruction when *Enable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
ParameterNumber	The number of the specific Boolean parameters	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 is available	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
Value	Value of the parameter to be read.	BOOL	TRUE/FALSE (FALSE)

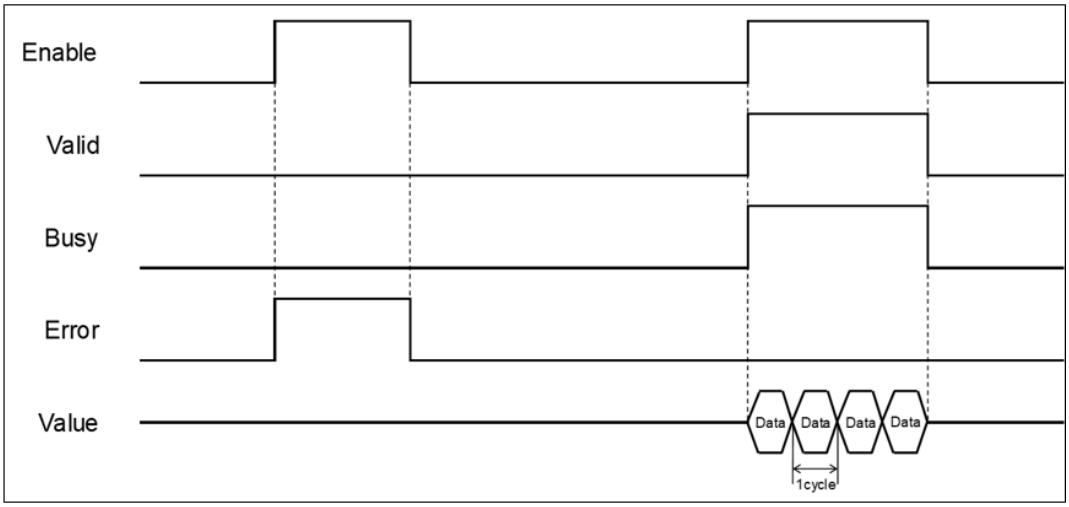
***Note**: SMC_ERROR: Enumeration (Enum)

- **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

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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	When *Execute* turns from TRUE to FALSE (Error Code is cleared.)
ErrorID		
Value	Continuously update when *Valid* is TRUE	When *Valid* is FALSE and stop updating

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified 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 the function block.

- **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 the object to the left by 24 bits.
    - Use SHL instruction to move the index of the object to the left by 8 bits.
    - Sum up the above parameters and adds the subindexes. 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 need to enter the parameter number of AXIS_REF_SM3 (FB) to *ParameterNumber* input.

- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Supported Devices**
  - AX-series motion controller

2.1.2.6 MC_WriteBoolParameter

MC_WriteBoolParameter writes a Boolean value to a 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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
ParameterNumber	The number of the parameters to be written	DINT	Positive, negative, or 0 (0)	*Execute* turns to TRUE and *Busy* is FALSE.
Value	Boolean value to be written to the parameter	BOOL	TRUE/FALSE (FALSE)	*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 instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

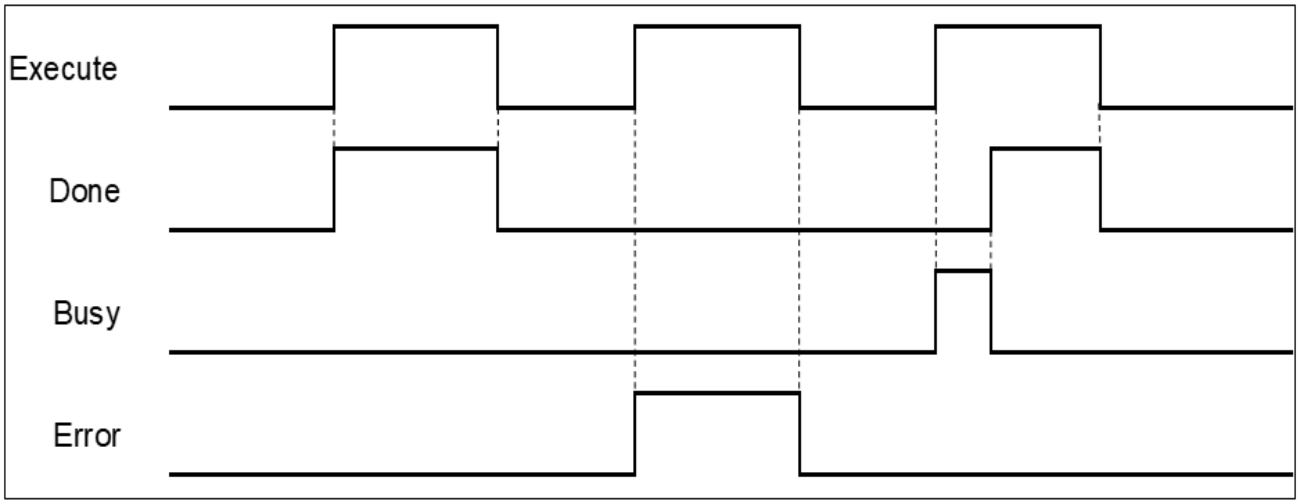
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Done	Completes writing the value.	*Execute* turns from TRUE to FALSE.
Busy	• *Execute* is triggered to be TRUE. • The value is being written to the parameter.	• *Done* turns to TRUE. • *Error* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.

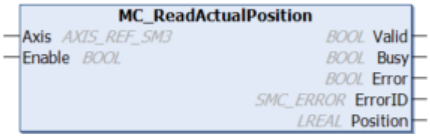
***Note**: AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.

- **Function**
  - To use MC_WriteBoolParameter to write the parameter ID in EtherCAT Object Dictionary:
    - Use SHL instruction to move the data length of the object to the left by 24 bits.
    - Use SHL instruction to move the index of the object to the left by 8 bits.
    - Sum up the above parameters and add the subindexes. . 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 need to enter the parameter number of AXIS_REF_SM3 (FB) into input *ParameterNumber*.
  - For the operation example, refer to the example in MC_WriteParameter.
- **Troubleshooting** .
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Supported Devices**
  - AX-series motion controller

2.1.2.7 MC_ReadActualPosition

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	Runs the instruction when *Enable* changes from FALSE to TRUE.	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 instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
Position	The current position of axis	LREAL	Positive, negative, or 0 (0)

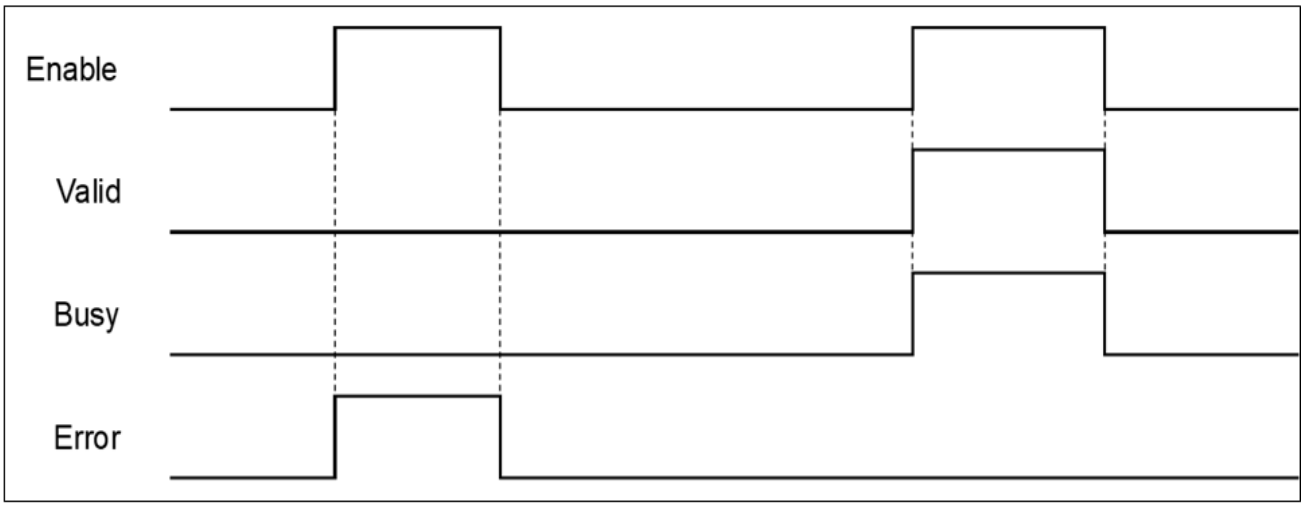
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Valid	• *Enable* is triggered to TRUE. • The parameter to read exists.	• *Enable* turns from TRUE to FALSE. • *Error* is rising edge.
Busy	• *Enable* is triggered to TRUE. • The parameter to read exists.	• *Enable* turns from TRUE to FALSE. • *Error* is rising edge.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Position	*Valid* is TRUE and there are ongoing updates.	*Valid* is FALSE and stop updating.

- **Timing Diagram of Output Parameter Changes**



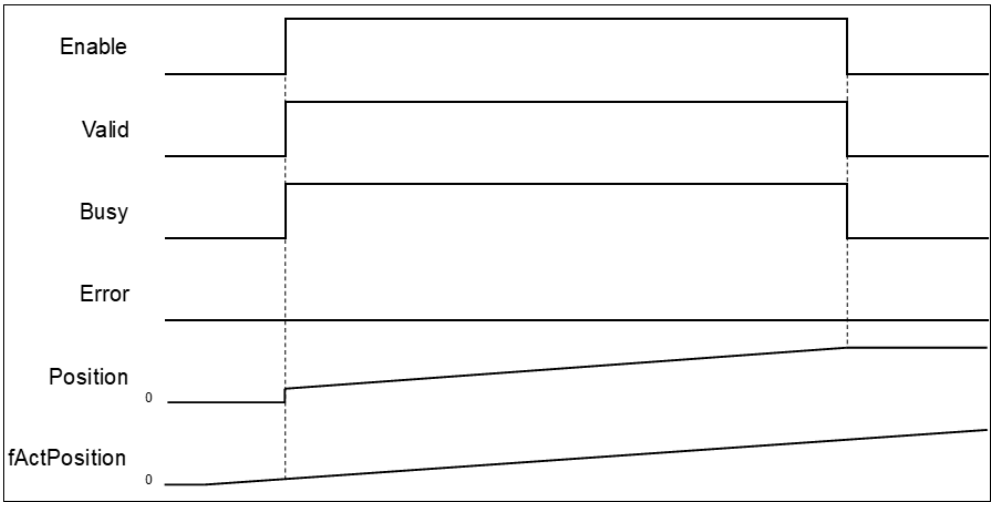
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	When *Enable* turns to TRUE

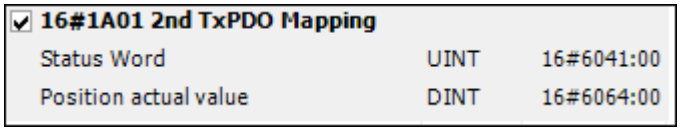
***Note**: AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.

- **Function**

  - The value read from *Position* of MC_ReadActualPosition is the 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.

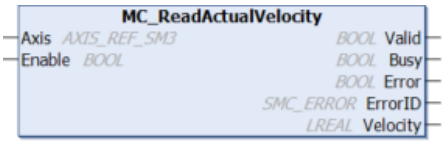


  - If TxPDO OD 0x6064 is not configured, the values read by the function blocks are all zero.

- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Supported Devices**
  - AX-series motion controller

2.1.2.8 MC_ReadActualVelocity

MC_ReadActualVelocity reads the actual axis velocity value.

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

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	Runs the instruction when *Enable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Valid	TRUE when the parameter is available	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
Velocity	The current velocity of axis.	LREAL	Positive, negative, or 0 (0)

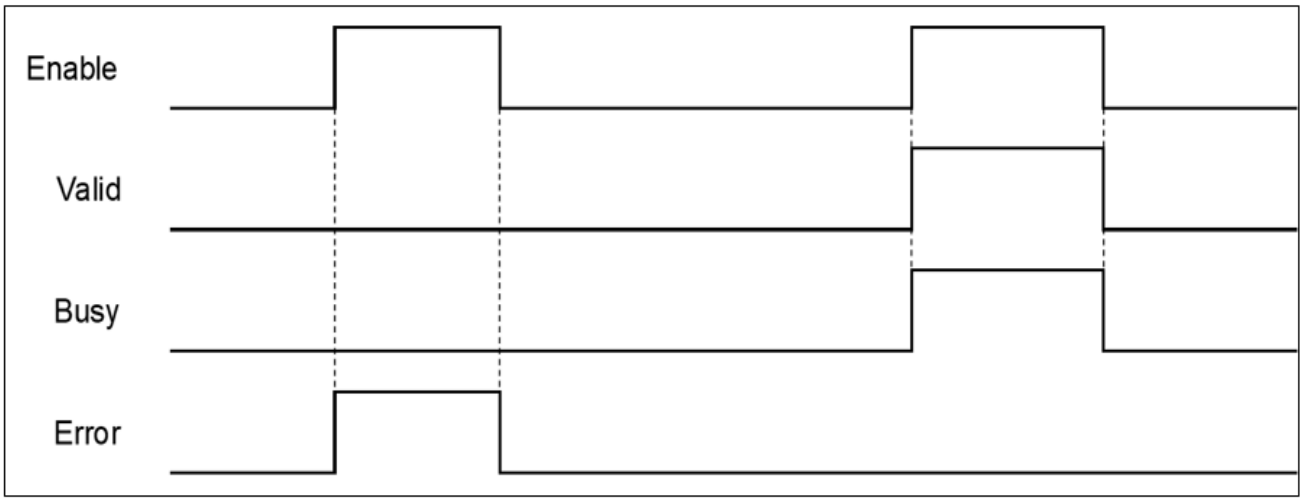
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Valid	• *Enable* is triggered to TRUE. • The parameter to read exists.	• *Enable* turns from TRUE to FALSE. • *Error* turns to TRUE.
Busy	• *Enable* is triggered to TRUE, • The parameter to read exists.	• *Enable* turns from TRUE to FALSE • *Error* turns to TRUE.
Error		When *Execute* turns from TRUE to FALSE (Error Code is cleared.)

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	
Velocity	Continuously update when *Valid* is TRUE.	When *Valid* is FALSE and stop updating

- **Timing Diagram of Output Parameter Changes**

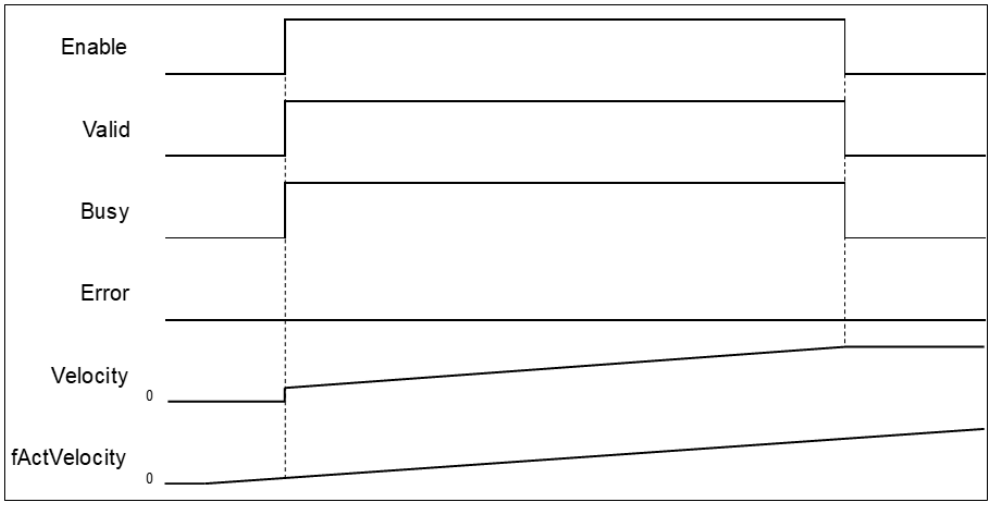


- **Inputs/Outputs**

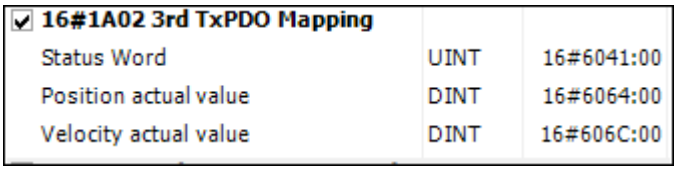
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	When *Enable* turns to TRUE

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

- **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 mapped to TxPDO so as to read the actual velocity of the servo.



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

- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Supported Devices**
  - AX-series motion controller

2.1.2.9 MC_ReadActualTorque

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	Runs the instruction when *Enable* changes from FALSE to TRUE.	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 instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
Torque	The actual torque of axis	LREAL	Positive or 0 (0)

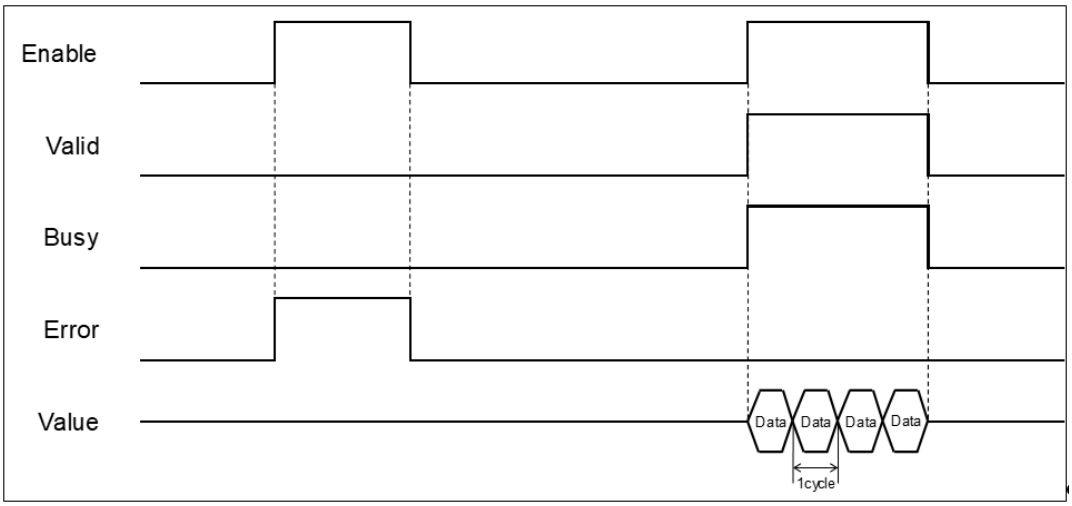
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Valid	• *Enable* is triggered to TRUE. • The parameter to read exists.	• *Enable* turns from TRUE to FALSE. • *Error* turns to TRUE.
Busy	• *Enable* is triggered to TRUE. • The parameter to read exists.	• *Enable* turns from TRUE to FALSE. • *Error* turns to TRUE.
Error		*Execute* turns from TRUE to FALSE. (Error Code is cleared.)

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	
Torque	Continuously update when *Valid* is TRUE.	*Valid* is FALSE and stop updating.

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Enable* turns to TRUE.

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

- **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 while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Supported Devices**
  - AX-series motion controller

2.1.2.10 MC_Reset

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	Runs the instruction 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 running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

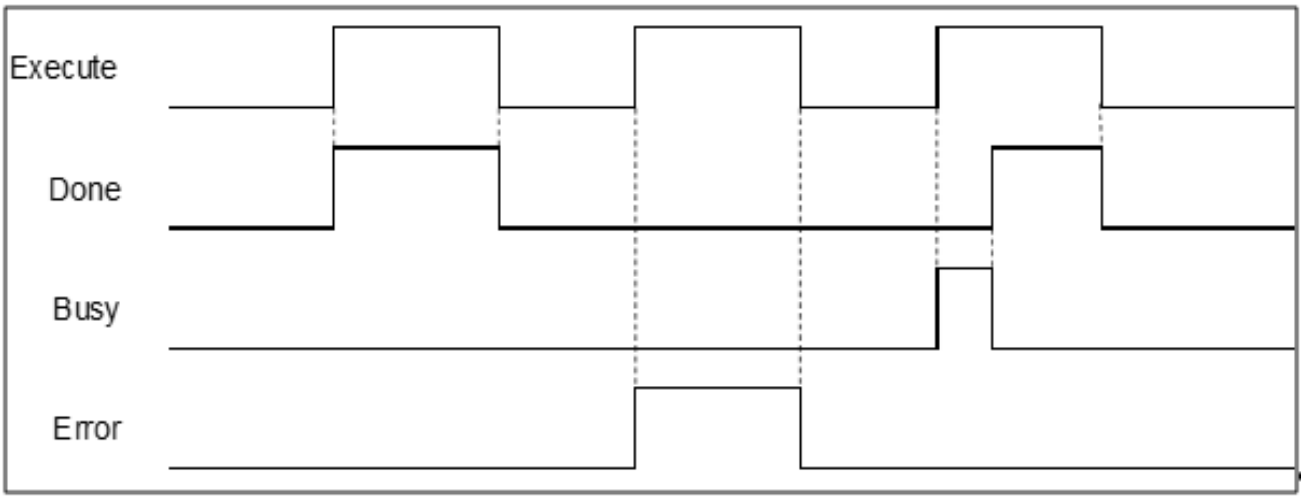
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Done	Complete clearing axis-related errors.	• *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 turn to FALSE.
Busy	*Execute* is triggered to TRUE.	• *Done* turns to TRUE • *Error* turns to TRUE
Error		*Execute* turns from TRUE to FALSE. (Error Code is cleared.)

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	

- **Timing Diagram of Output Parameter Changes**



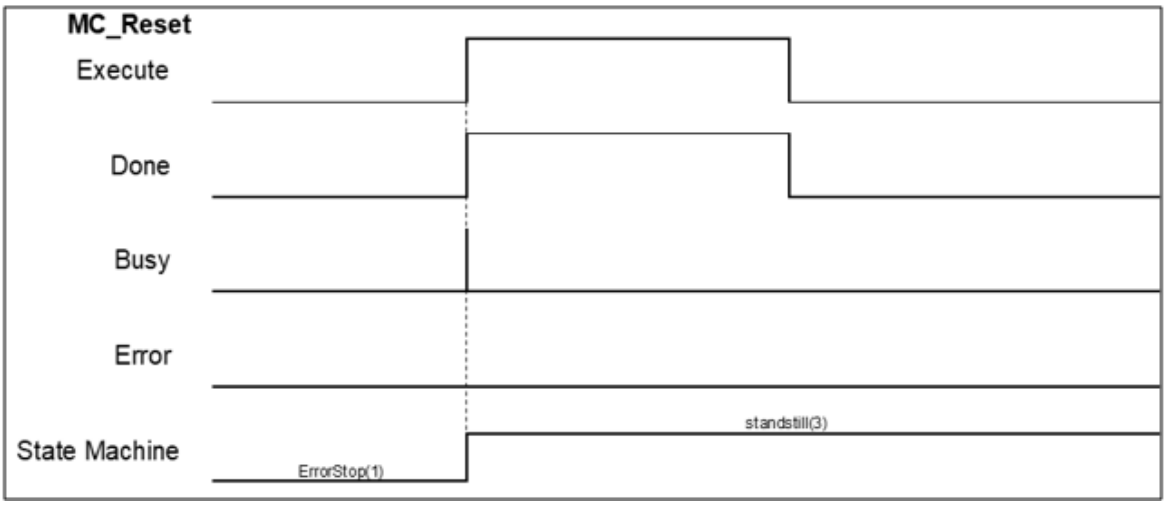
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*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 the function block.

- **Function**

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



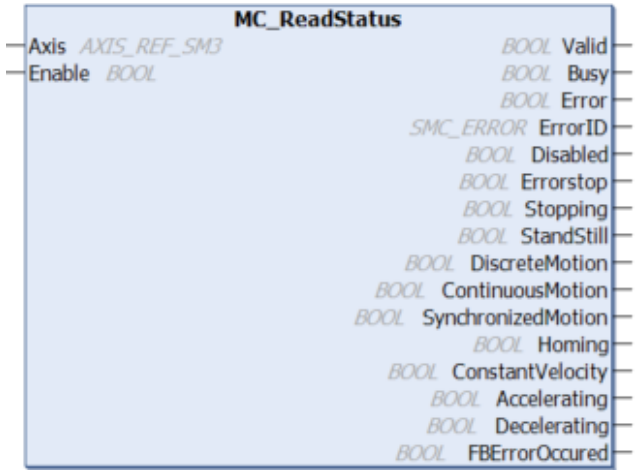
  - After the servo controller report errors, you can use MC_Reset to clear them and then the axis state will return to Standstill.

  - If axis errors cannot be cleared by MC_Resets, such as the communication error, SMC_R_ERROR_NOT_RESETTABLE 122 (Error could not be reset.) will be reported by MC_Reset.

- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Supported Devices**
  - AX-series motion controller

2.1.2.11 MC_ReadStatus

MC_ReadStatus reads the status of the 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	Runs the instruction when *Enable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Valid	TRUE when the parameter is available	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*1	SMC_ERROR (SMC_NO_ERROR)
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)

Name	Function	Data Type	Output Range (Default Value)
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	• .*Enable* turns to TRUE. • The parameter to read exists.	• *Enable* turns from TRUE to FALSE. • *Error* turns to TRUE.
Busy	*Enable* turns to TRUE.	• *Enable* turns from TRUE to FALSE. • *Error* turns to TRUE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		
Disabled	The axis is in the Disabled state.	The axis is not in Disabled state.
Errorstop	The axis is in the Errorstop state.	The axis is not in Errorstop state.
Stopping	The axis is in the Stopping state.	The axis is not in Stopping state.
StandStill	The axis is in the StandStill state.	The axis is not in StandStill state.
DiscreteMotion	The axis is in the Discrete Motion state.	The axis is not in Discrete Motion state.
ContinuousMotion	The axis is in the Continuous Motion state.	The axis is not in Continuous Motion state.
SynchronizedMotion	The axis is in the Synchronized Motion state.	The axis is not in Synchronized Motion state.
Homing	The axis is in the Homing state.	The axis is not in Homing state.
ConstantVelocity	The axis moves at a constant speed.	The axis moves at a non-constant speed.
Accelerating	The axis moves with acceleration.	The axis moves without acceleration.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Decelerating	The axis moves with deceleration.	The axis moves without deceleration.
FBErrorOccured	Errors occur.	Errors are cleared.

- **Inputs/Outputs**

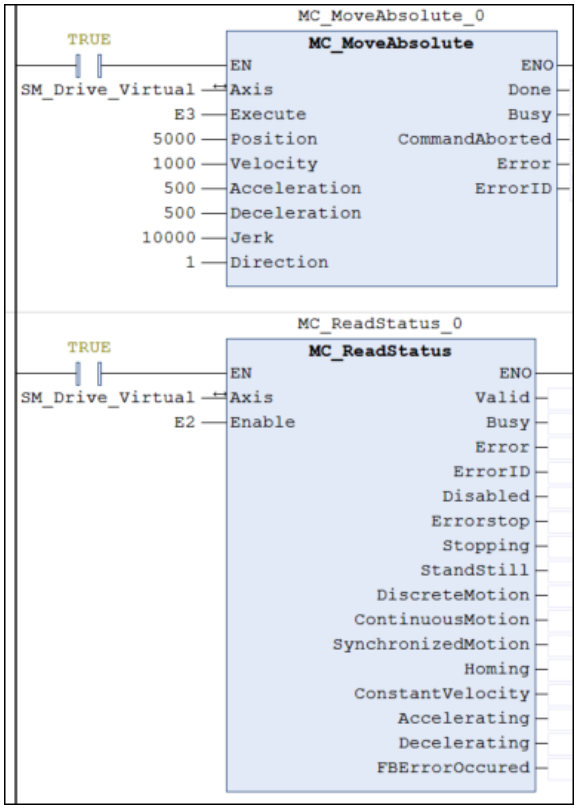
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Enable* turns to TRUE.

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

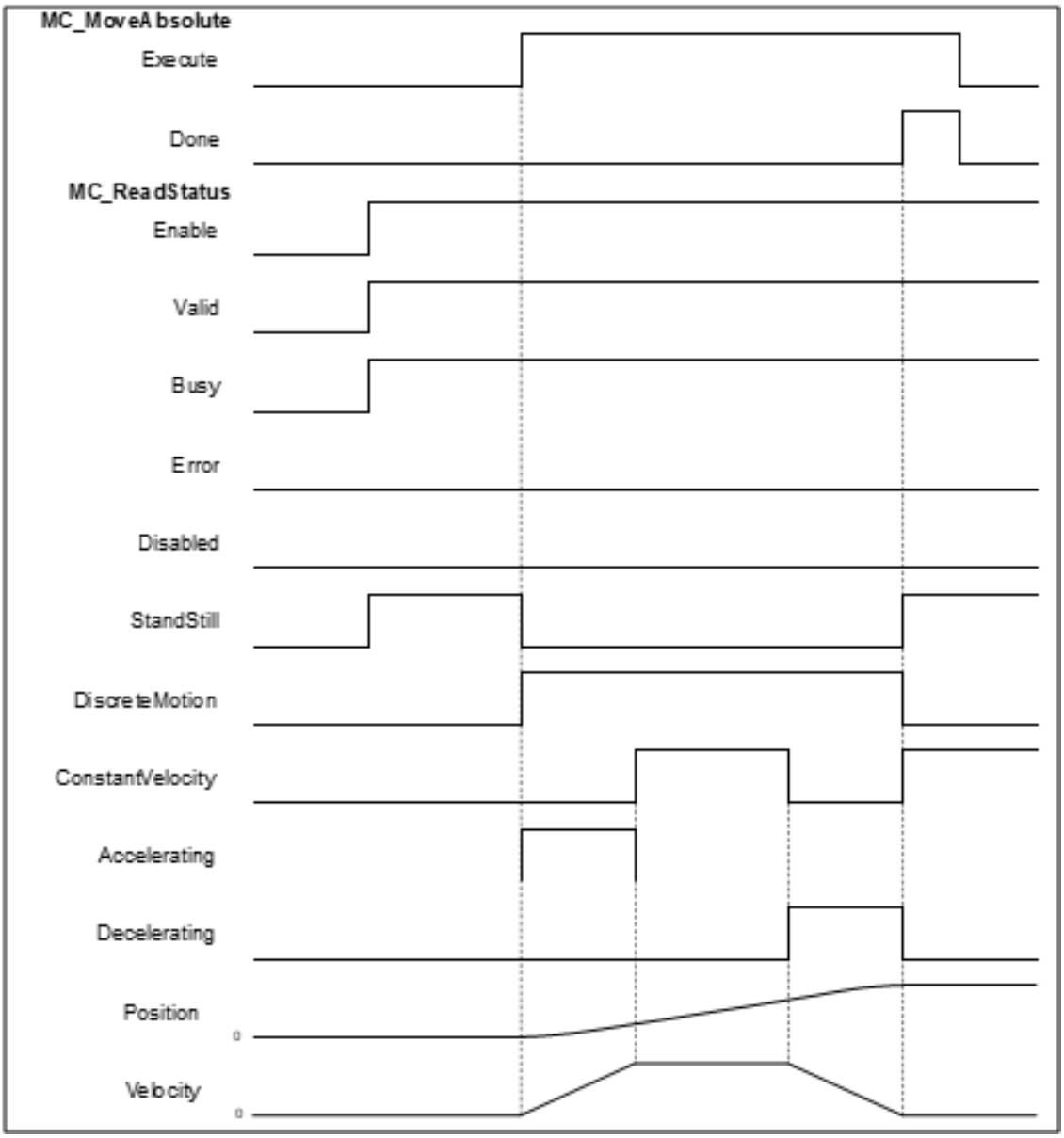
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

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



- Timing Diagram

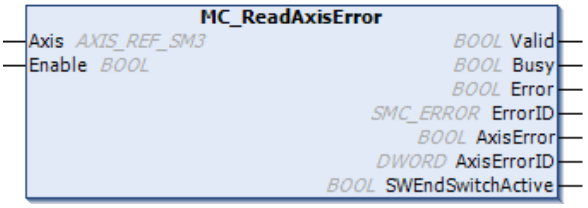


  - After MC_MoveAbsolute is executed, the axis state turns from Standstill to Discrete_motion. At the same time, the axis begins to accelerate and the output *Accelerating* turns to TRUE.
  - When the axis velocity reaches the setting in MC_MoveAbsolute, the axis moves at a constant speed. Meanwhile, the output *ConstantVelocity* turns to TRUE and *Accelerating* turns to FALSE. When the axis reaches the target position, the axis starts decelerating, and *Decelerating* turns to TRUE and *ConstantVelocity* turns to FALSE.
  - *Done* of MC_MoveAbsolute turns to TRUE when the target position is reached. The output status turns from Discretemotion to Standstill.

- **Supported Devices**
  - AX-series motion controller

2.1.2.12 MC_ReadAxisError

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	Runs the instruction when *Enable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Valid	TRUE when the parameter is available	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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 vender	DWORD	Positive or 0 (0)
SWEndSwitchActive	TRUE when the axis exceeds the software limit	BOOL	TRUE/FALSE (FALSE)

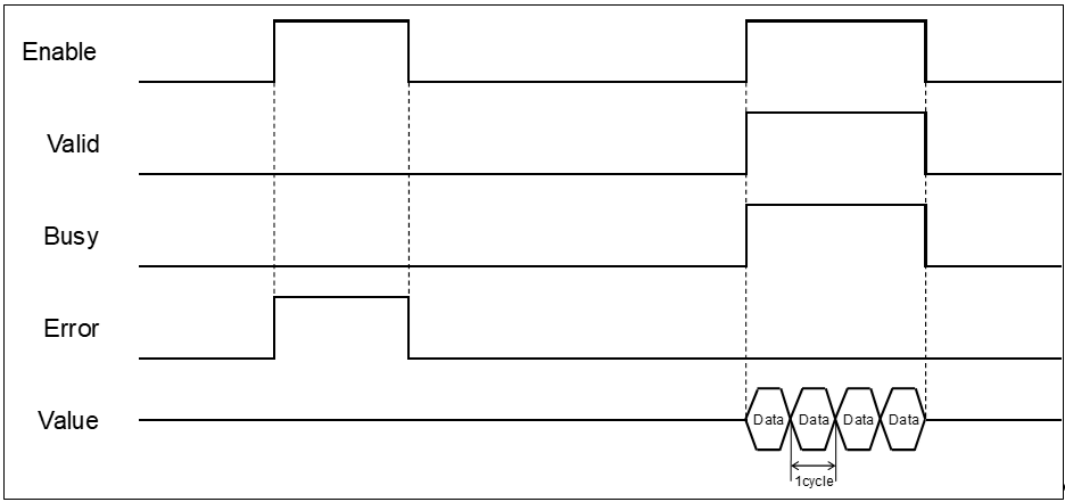
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Valid	• *Enable* is triggered to TRUE. • The parameter is available.	• *Enable* turns from TRUE to FALSE. • *Error* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Busy	• *Enable* is triggered to TRUE. • The parameter is available.	• *Enable* turns from TRUE to FALSE • *Error* turns to TRUE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		
AxisError	An error occurs in the axis.	Errors are cleared.
AxisErrorID		
SWEndSwitchActive	The axis exceeds the software limit.	MC_Reset is ran.

- **Timing Diagram of Output Parameter Changes**



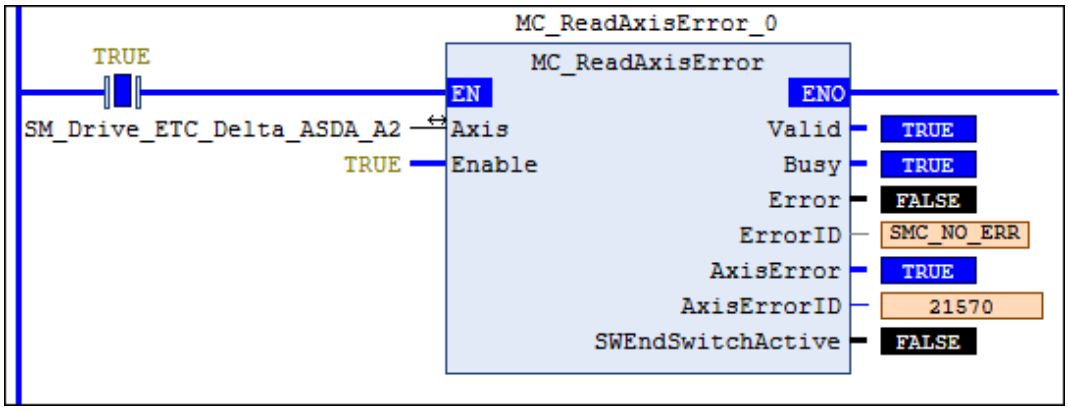
- **Inputs/Outputs**

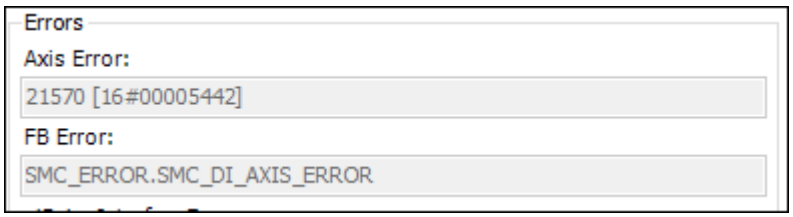
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Enable* turns to TRUE.

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

- **Function**
  - The output *SWEndSwitchActive* will turn to TRUE once the axis reaches the software limit.
  - *AxisErrorID* displays the error codes of the servo motor. Taking ASDA-A2-E as an 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 the axis.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

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



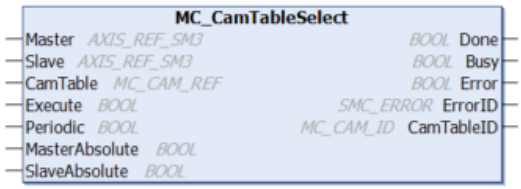


When the ASDA-A2-E servo touches the positive hardware limit, “AL015” will be displayed on the servo panel. You can use MC_ReadAxisError to read the error code. The error code for AL015 is 0x5442 (refer to ASDA-A2-E user manual.) AxisErrorID is used to display the error code and will also be displayed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

- **Supported Devices**
  - AX-series motion controller

2.1.2.13 MC_CamTableSelect

MC_CamTableSelect selects the cam tables. It is used 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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Periodic	Periodic mode	BOOL	TRUE/FALSE (TRUE)	*Execute* turns to TRUE and *Busy* is FALSE.
MasterAbsolute	The absolute mode of the master axis	BOOL	TRUE/FALSE (TRUE)	*Execute* turns to TRUE and *Busy* is FALSE.
SlaveAbsoulte	The absolute mode of the slave axis	BOOL	TRUE/FALSE (TRUE)	*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 running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*1	SMC_ERROR (SMC_NO_ERROR)
CamTableID	Generate CAM_ID to be used by *CamTableID* of MC_CamIn.	MC_CAM_ID*2	MC_CAM_ID

***Note**:

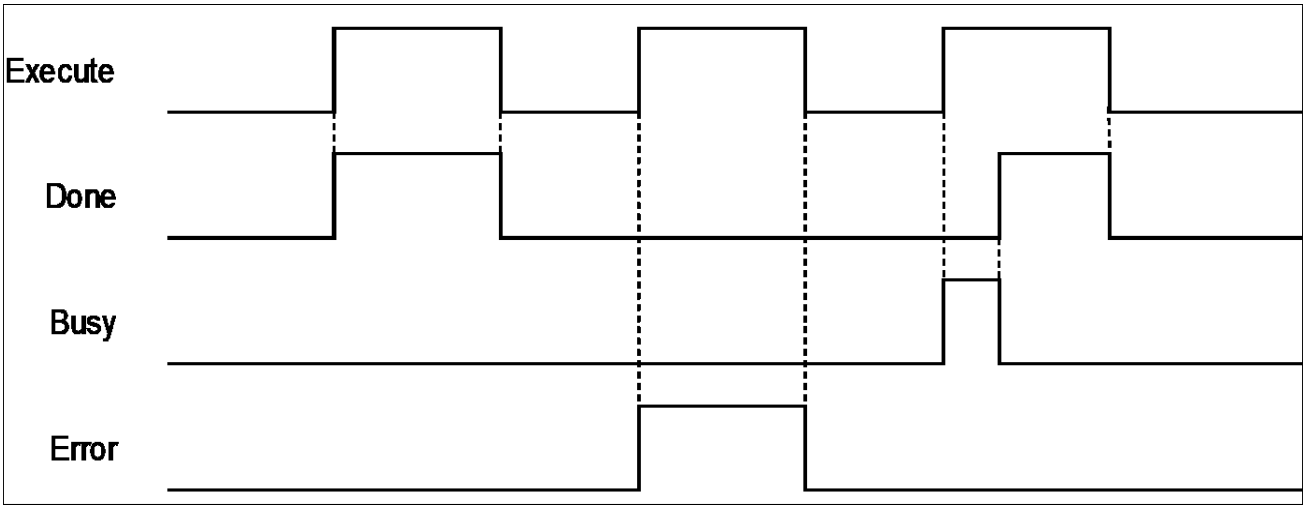
1. SMC_ERROR: Enumeration (Enum)
2. MC_CAM_ID: Structure (Struct)

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	MC_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 turn to FALSE.
Busy	The instruction is running.	• *Done* turns to TRUE. • *Error* turns to TRUE.
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	Specifies the master axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.
Slave	Specifies the slave axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	*Execute* turns to TRUE and *Busy* is FALSE.
CamTable	Specifies the cam table.	MC_CAM_REF*2	MC_CAM_REF	*Execute* turns to TRUE and *Busy* is FALSE.

***Note**:

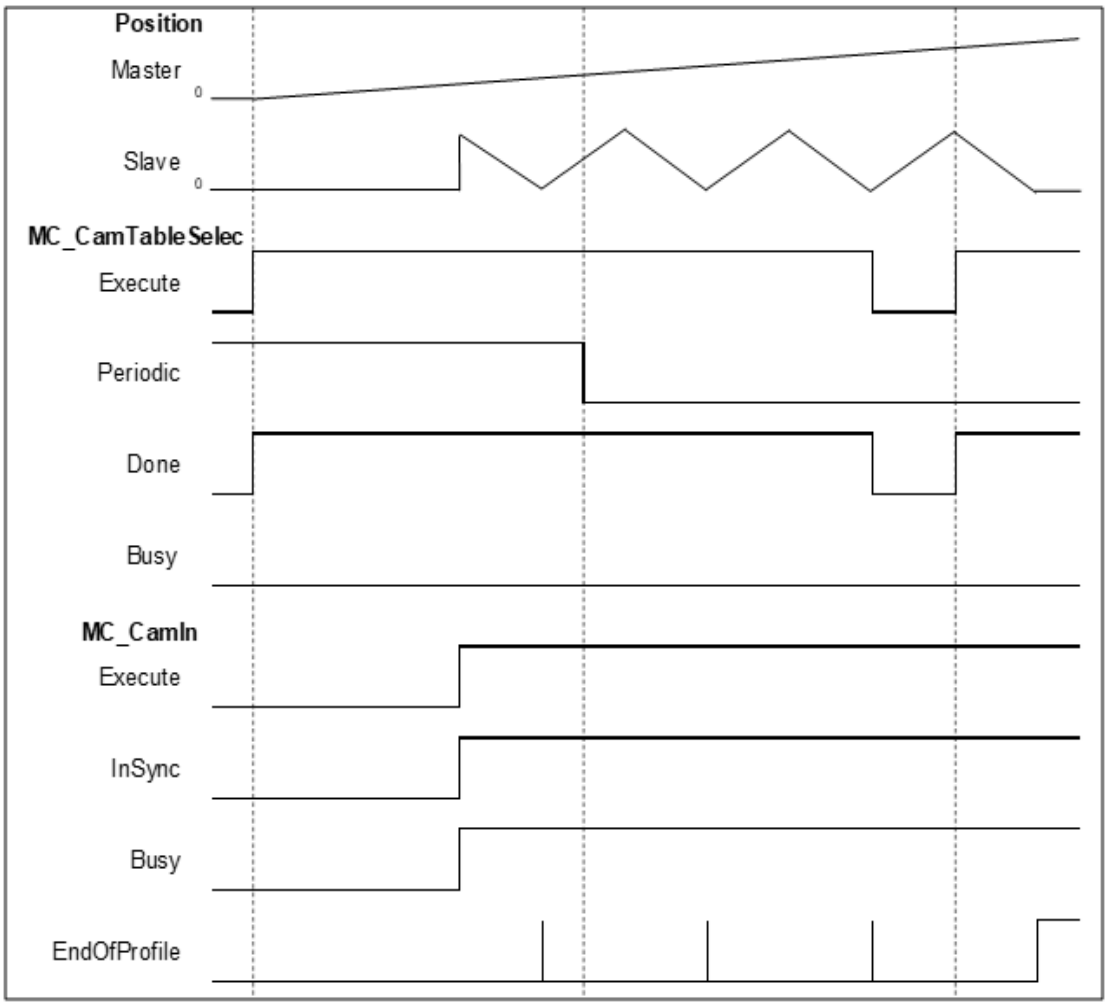
1. AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.
2. MC_CAM_REF(FB): This data structure is used as a reference to a cam table.

- **Function**
  - Use MC_CamTableSelect to select the cam table that will be followed.
  - Set *Execute* to TRUE to specify or refresh the cam table. When *Done* turns to TRUE, *CamTableID* is valid.
  - After the master-slave synchronization is completed, the modification of MC_CamTableSelect parameters can cause changes in the cam.
    - After changes the variables of *CamTable*, the cam mode will be effective immediately.
    - The function block must be reboot after changing to the Periodic mode.

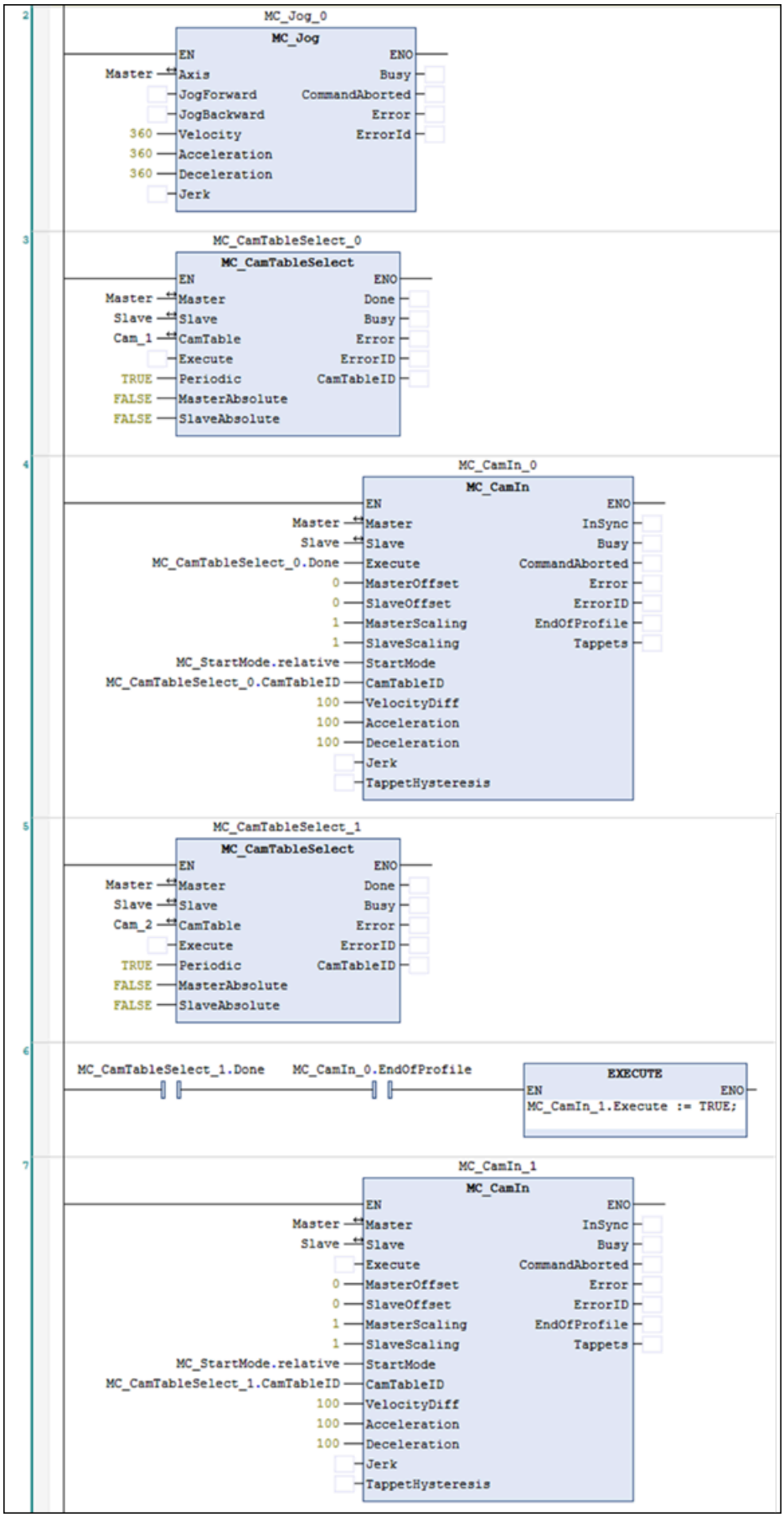
- **Troubleshooting**

  If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**
  - **Example 1**: This example explains the impact on cam after changing the Periodic mode.
    - Timing Diagram



    - To change the periodic mode, *Periodic* of MC_CamTable will be pulled down and turn to FALSE, while the slave axis remains the periodic mode.
    - After rebooting MC_CamTable, the slave axis enters non-periodic mode. The slave axis will run the last cycle of cam motion, and then *EndOfProfile* will be kept at TRUE.
  - **Example 2**: This example explains how to switch cam tables.

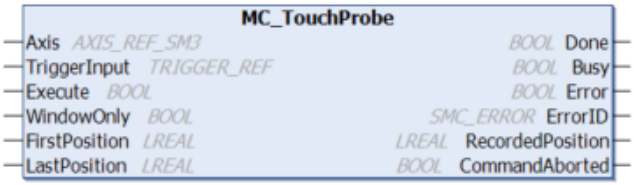

2
MC_Jog_0
MC_Jog
EN
ENO
Master
Axis
Busy
JogForward
CommandAborted
JogBackward
Error
360
Velocity
ErrorId
360
Acceleration
360
Deceleration
Jerk
3
MC_CamTableSelect_0
MC_CamTableSelect
EN
ENO
Master
Master
Done
Slave
Slave
Busy
Cam_1
CamTable
Error
Execute
ErrorID
TRUE
Periodic
CamTableID
FALSE
MasterAbsolute
FALSE
SlaveAbsolute
4
MC_CamIn_0
MC_CamIn
EN
ENO
Master
Master
InSync
Slave
Slave
Busy
MC_CamTableSelect_0.Done
Execute
CommandAborted
0
MasterOffset
Error
0
SlaveOffset
ErrorID
1
MasterScaling
EndOfProfile
1
SlaveScaling
Tappets
MC_StartMode.relative
StartMode
MC_CamTableSelect_0.CamTableID
CamTableID
100
VelocityDiff
100
Acceleration
100
Deceleration
Jerk
TappetHysteresis
5
MC_CamTableSelect_1
MC_CamTableSelect
EN
ENO
Master
Master
Done
Slave
Slave
Busy
Cam_2
CamTable
Error
Execute
ErrorID
TRUE
Periodic
CamTableID
FALSE
MasterAbsolute
FALSE
SlaveAbsolute
6
MC_CamTableSelect_1.Done
MC_CamIn_0.EndOfProfile
EXECUTE
EN
ENO
MC_CamIn_1.Execute := TRUE;
7
MC_CamIn_1
MC_CamIn
EN
ENO
Master
Master
InSync
Slave
Slave
Busy
Execute
CommandAborted
0
MasterOffset
Error
0
SlaveOffset
ErrorID
1
MasterScaling
EndOfProfile
1
SlaveScaling
Tappets
MC_StartMode.relative
StartMode
MC_CamTableSelect_1.CamTableID
CamTableID
100
VelocityDiff
100
Acceleration
100
Deceleration
Jerk
TappetHysteresis

- When *MC_CamTableSelect_0.Execute* is TRUE and *Done* changes to TRUE, the MC_CamIn_0 function block is triggered.
- When *MC_CamIn_0.InSync* is TRUE, run MC_Jog_0. At this point, the master and slave axes start to synchronize.
- When *MC_CamTableSelect_1.Execute* is TRUE, and then *MC_CamIn_0.EndOfProfile* turns to TRUE, run MC_CamIn_1. At this time, the master and slave axes will synchronize based on the MC_CamIn_0 and MC_CamIn_1 cam table.

- **Supported Devices**
  - AX-series motion controller

2.1.2.14 MC_TouchProbe

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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
WindowOnly	Activates the scope setting of Window.	BOOL	TRUE/FALSE (FALSE)	*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)	*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)	*Execute* turns to TRUE and *Busy* is FALSE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	When the trigger signal is TRUE and the axis position has been recorded	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)
RecordedPosition	The axis position recorded at the time of trigger signal being TRUE	LREAL	LREAL(0)
CommandAborted	TRUE when the instruction is interrupted 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	The trigger signal is TRUE and the axis position has been recorded.	• *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 turn to FALSE.
Busy	*Execute* is triggered to be TRUE.	• *Done* turns to TRUE. • *Error* turns to TRUE. • *CommandAborted* turns to TRUE.
Error ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE. (Error Code is cleared.)
CommandAborted	The function block is interrupted by MC_AbortTrigger.	• *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 turn to FALSE.

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*1	AXIS_REF_SM3	-
TriggerInput	Trigger signal	TRIGGER_REF*2	TRIGGER_REF	*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 the function block.
2. TRIGGER_REF: Structure (STRUCT)

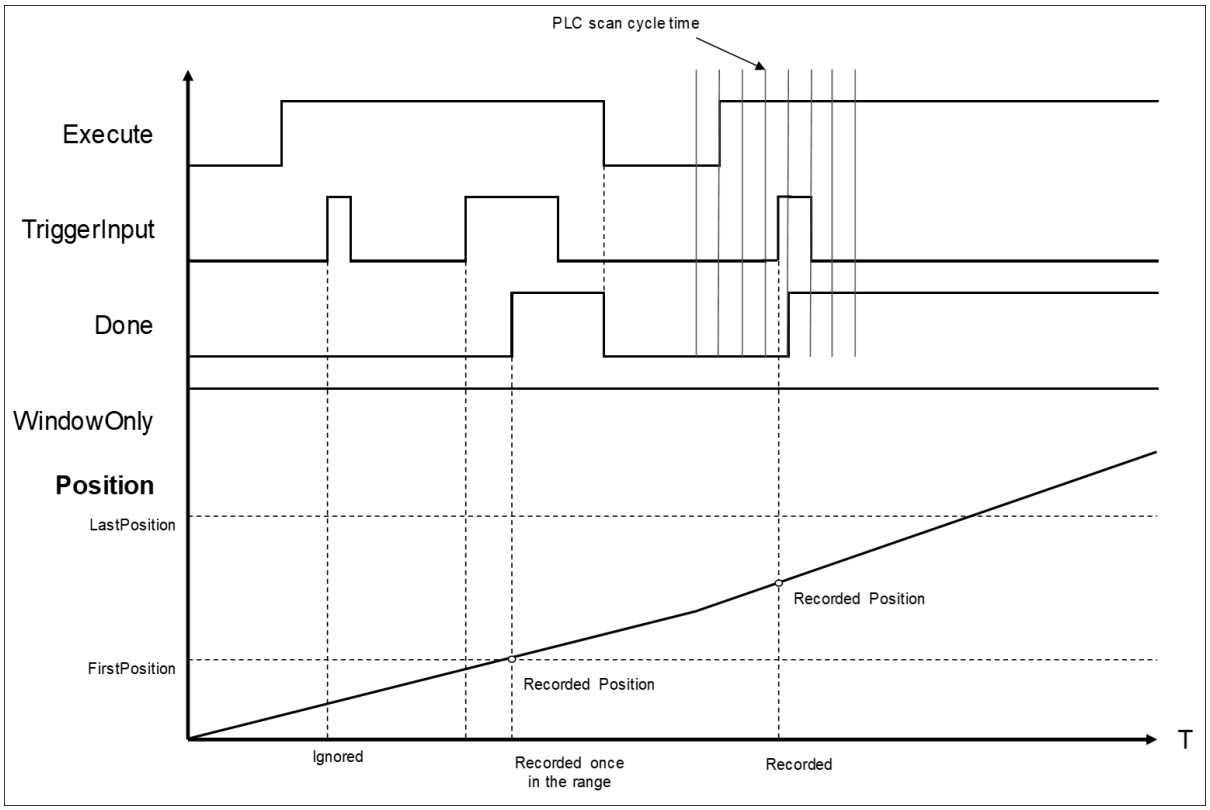
Name	Function	Data Type	Setting Value (Default Value)
iTriggerNumber	Trigger channel	INT	0: Touch Probe 1, rising edge 1: Touch Probe 1, falling edge 2: Touch Probe 2, rising edge 3: Touch Probe 2, falling edge (–1)
bFastLatching	Trigger signal	BOOL	TRUE: Latching is done in drive

Name	Function	Data Type	Setting Value (Default Value)
			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**
  - Driver mode
    - When using physical axes, it must be set as driver-triggered. *bFastLatching* must be set to TRUE, and *iTriggerNumber* should be configured. (The value cannot be set as default "-1", or there will be an error in the function block.).

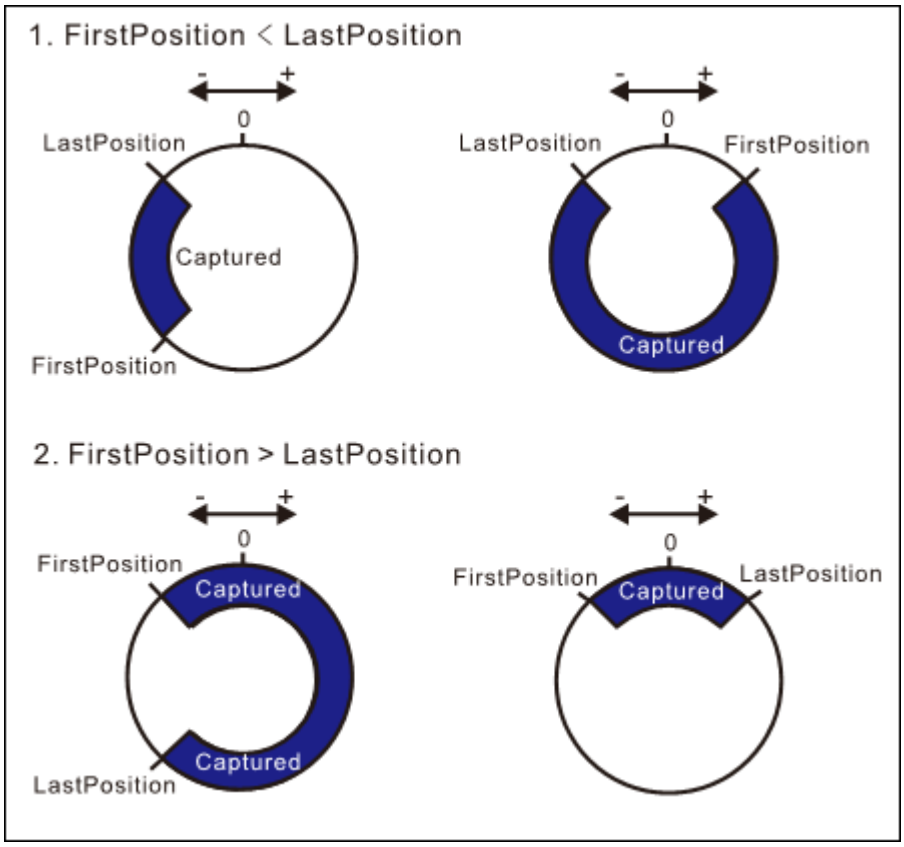
      When *Execute* is TRUE, the function block writes values to 0x60B8 (Touch Probe Function) based on the setting of *iTriggerNumber* to open the corresponding Trigger channel.
    - If *Execute* of MC_TouchProbe is TRUE, only the first position value of the trigger signal will be captured, and the following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



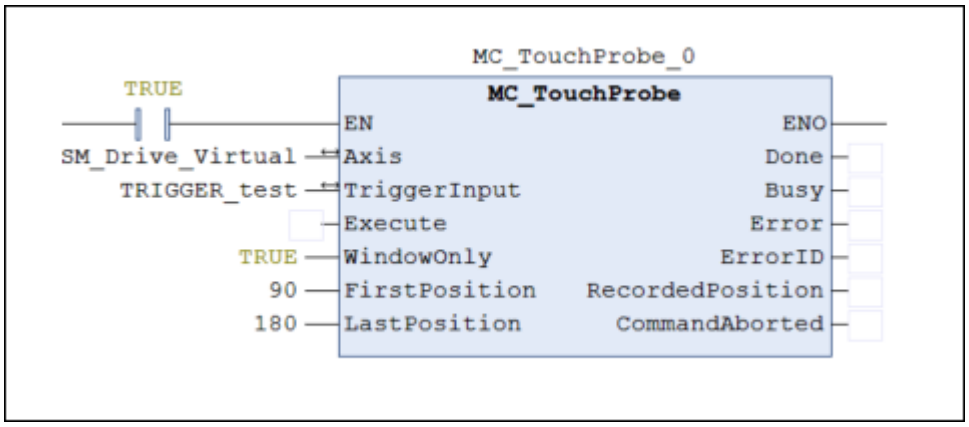
- When the signal input is triggered for the first time, the signal is not accepted because the axis position does not reach the window mask section.
- When the axis position reaches the window mask section, the second triggered input signal is accepted. The output signal *Done* turns to TRUE in the next cycle.
- When *WindowOnly* turns to TRUE, the TouchProbe function does not work immediately and will take some time to work.
- If the window mask is too small, MC_TouchProbe will not work. The effective range for the window mask depends on EtherCAT communications and the performance of the encoder input or the servo drive.
- If 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

- When the axis is set to rotary/modulo, different Window Mask settings will result differently. The result of setting different FirstPosition and LastPosition is shown in the following figure.

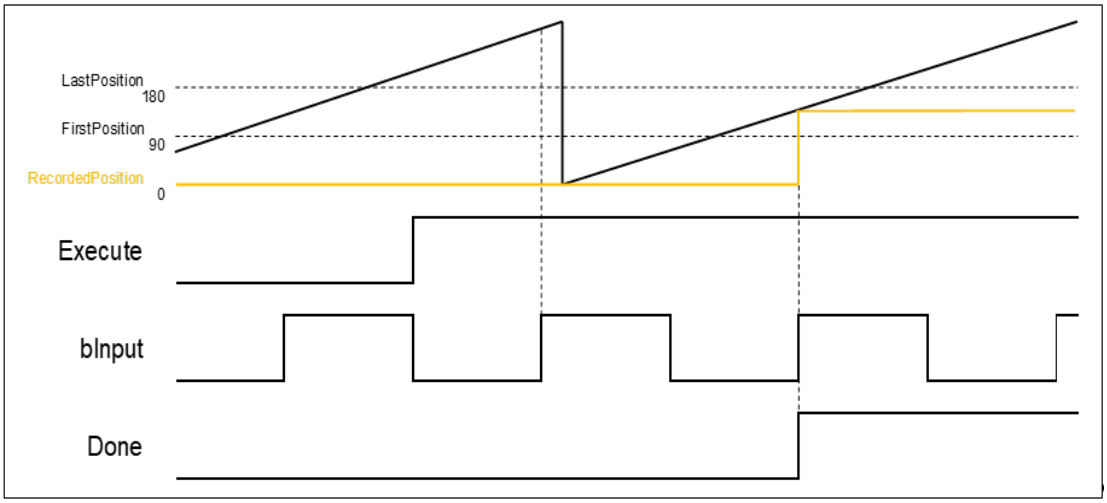


- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - **Example 1**: This example demonstrates the operation result of using MC_TouchProbe under the controller mode.



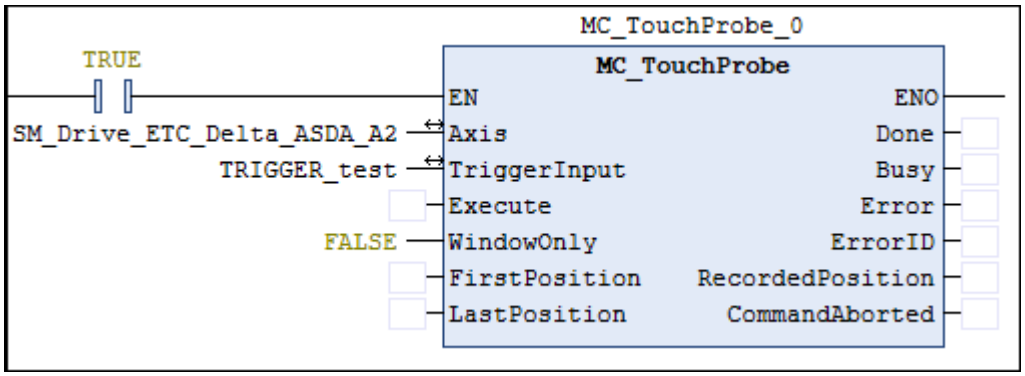

```
TRIGGER_test.bFastLatching := FALSE;
```

- Timing Diagram



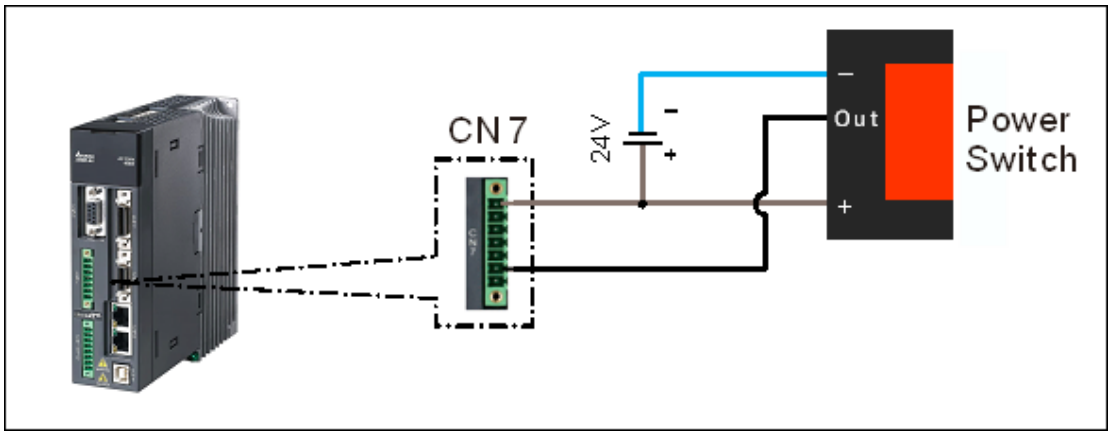
  - When *Execute* of MC_TouchProbe changes to TRUE, it starts to capture the signal. Under the controller mode, bInput will be the trigger signal.
  - At the first activation of the trigger input signal, the axis position is not recorded because the axis position hasn't reached the window mask section.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

- **Example 2**: This example uses ASDA-A2-E as the driver to demonstrate how MC_TouchProbe uses the driver as the trigger signal.

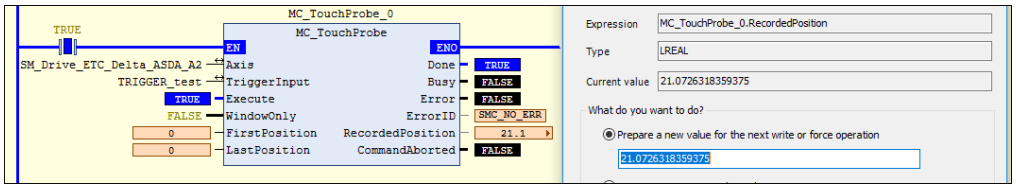



```
TRIGGER_test.iTriggerNumber := 0;
```

- Wiring diagram

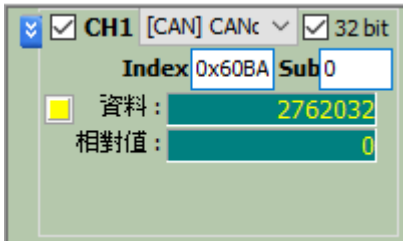


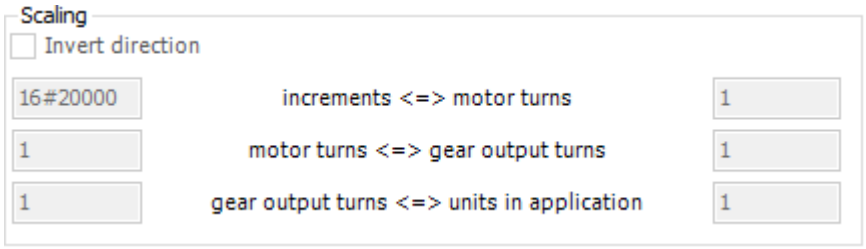
- The trigger signal is from DI13 of Servo Drive CN7 Expansion DI. It can be configured according to the above wiring diagram.
- The trigger channel must be specified by the function block. The following example demonstrates with rising edge trigger.



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*.





- Since the gear ratio is set to 0x20000: 1, when the driver is rising edge triggered, the value in 0x60BA must be divided with 0x20000. The signal is triggered by the 2762032 index pulses, so the recorded position is 21.0726318359375(2762032 / 131072).

- **Supported Devices**
  - AX-series motion controller

2.1.2.15 MC_AbortTrigger

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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (TRUE)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when the trigger event is aborted	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

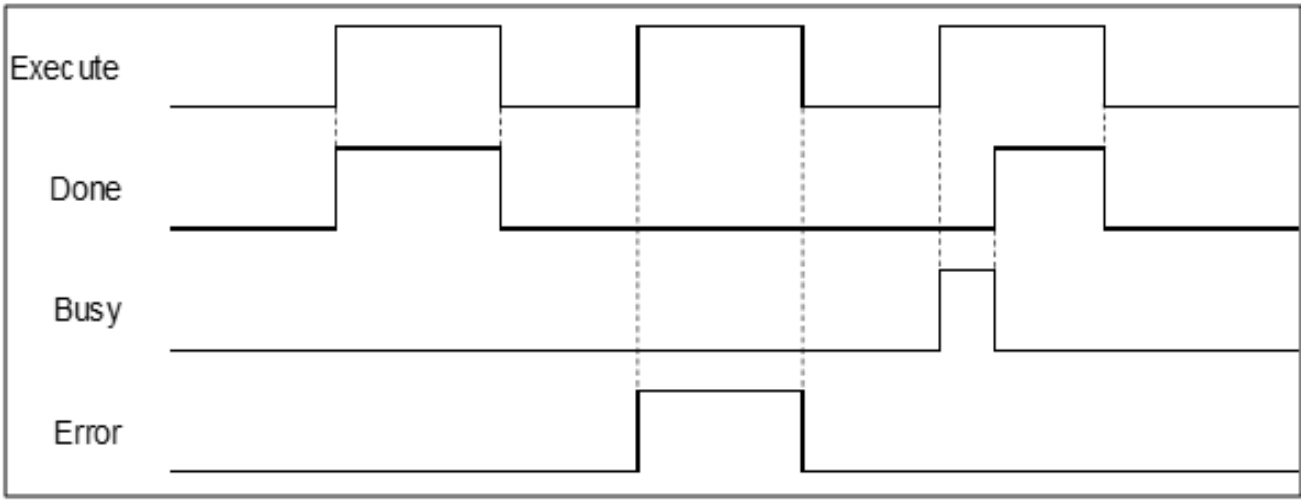
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Done	The capture operation is stopped.	• *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 turn to FALSE.
Busy	*Execute* turns to TRUE.	• *Done* turns to TRUE. • *Error* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	• *Execute* turns from TRUE to FALSE (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Change**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*1	AXIS_REF_SM3	*Execute* turns to TRUE and Busy is FALSE.
TriggerInput	Specifies the reference to the source of the trigger signal.	TRIGGER_REF*2	TRIGGER_REF	*Execute* turns to TRUE and Busy is FALSE.

***Note**:

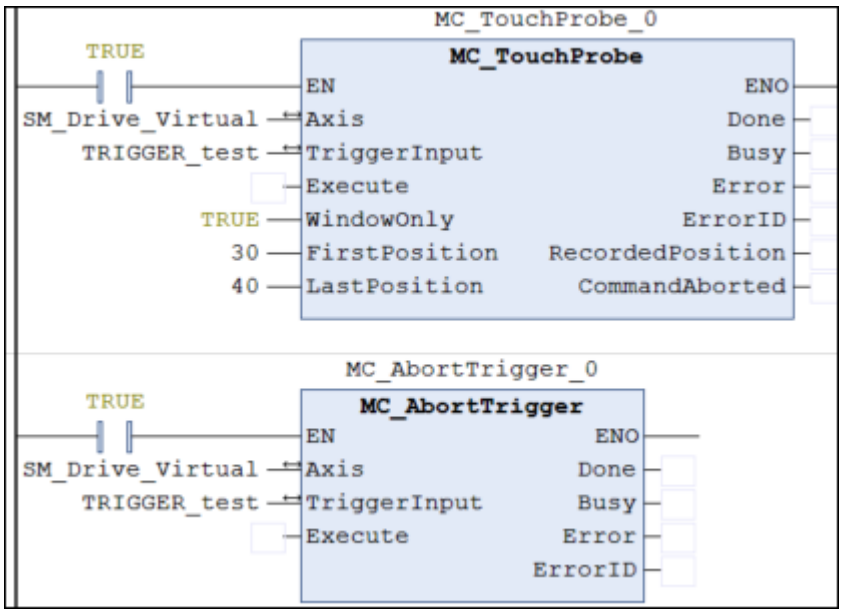
1. AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.
2. TRIGGER_REF: Structure(STRUCT).

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.

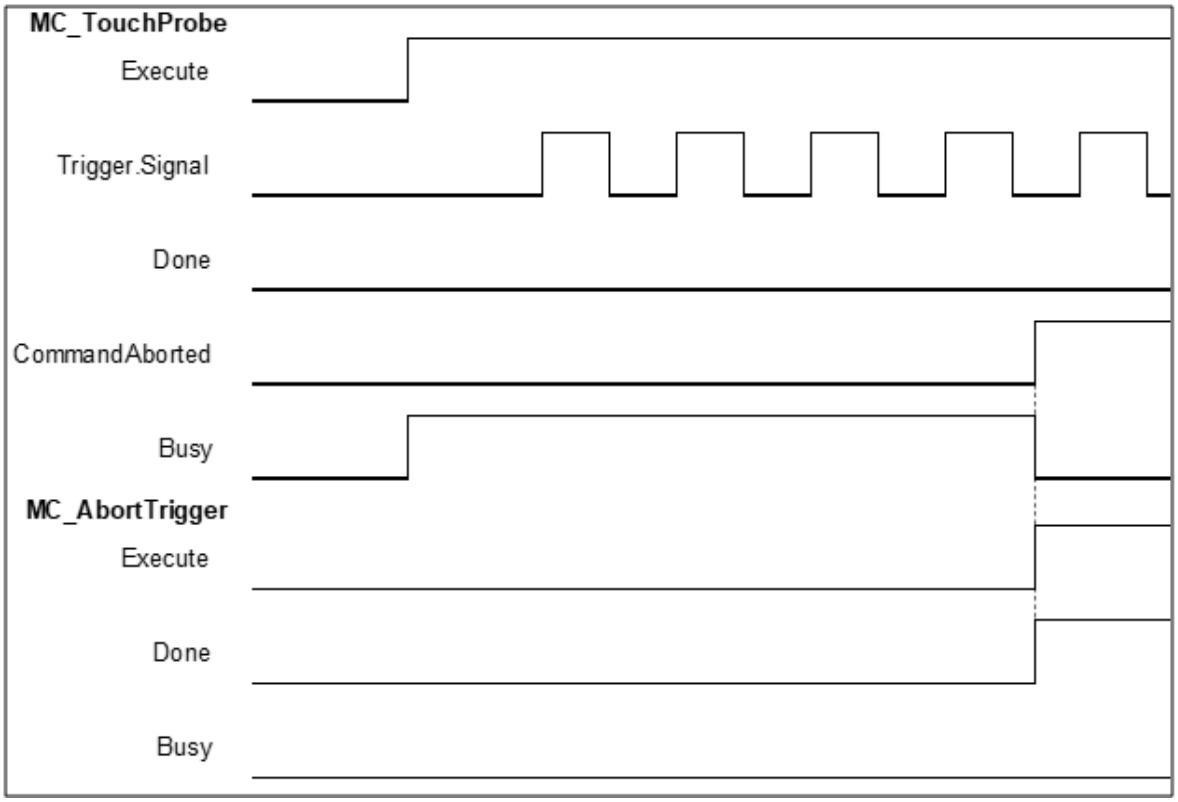
- **Function**
  - MC_AbortTrigger is used to absorb the operation that are connected to MC_AbortTrigger.
  - The touch probe operation to abort is determined by *Axis* and *TriggerInput* in the instruction.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  This example demonstrates the operations of MC_AbortTrigger and MC_TouchProbe.




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

  - Timing Diagram

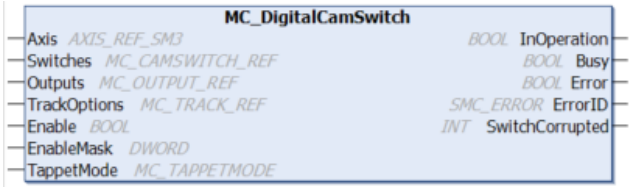


- When a rising edge is detected on *Execute* of MC_AbortTrigger, *CommandAborted* of MC_TouchProbe turns to TRUE.
- If a rising edge is detected on *Execute* of MC_AbortTrigger when *Done* of MC_TouchProbe turns to TRUE, an error of SMC_AT_TRIGGERNOTOCCUPIED (410) will be reported by MC_AbortTrigger.

- **Supported Devices**
  - AX-series motion controller

2.1.2.16 MC_DigitalCamSwitch

MC_DigitalCamSwitch switches a motor on or off according to the axis position.

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	Runs the instruction when *Enable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
EnableMask	Enables the different tracks.	DWORD	Positive or 0 (16#FFFFFFFF)	*Enable* turns to TRUE
TappetMode	Defines 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)	*Enable* turns to TRUE

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

- **Outputs**

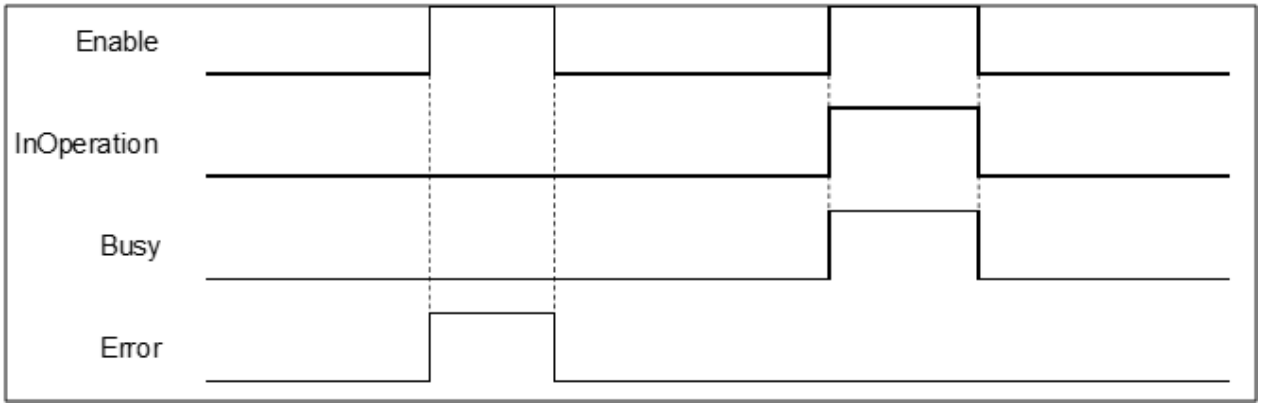
Name	Function	Data Type	Output Range (Default Value)
InOperation	TRUE when the track and instruction is activated	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
Error	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The track and instruction is activated.	*Enable* turns to FALSE
Busy	The instruction is running.	• *Enable* turns to FALSE. • *Error* turns to TRUE.
Error ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Execute* turns from TRUE to FALSE (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
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
Axis	The specified axis	AXIS_REF_SM3*1	AXIS_REF_SM3	When *Enable* turns to TRUE

***Note**:

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

Name	Function	Data Type	Setting Value (Default Value)
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)

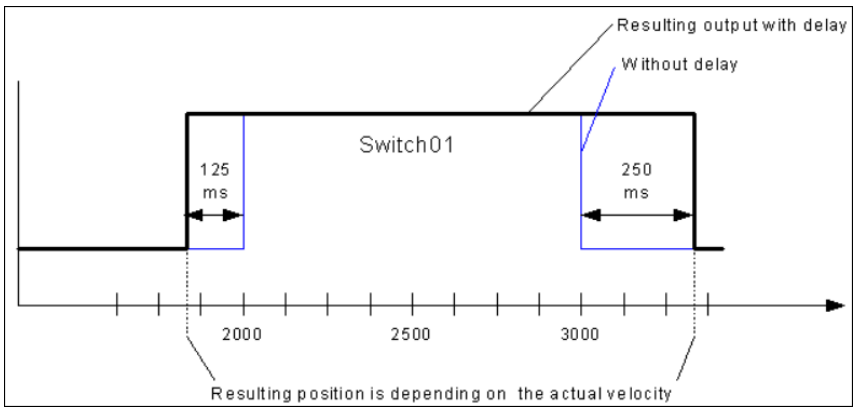
Name	Function	Data Type	Setting Value (Default Value)
OnCompensation	Compensation time with which the switch is turned on. (Unit: Sec.)	LREAL	Positive, negative, or 0 (0)
OffCompensation	Compensation time with which the switch is turned off. (Unit: Sec.)	LREAL	Positive, negative, or 0 (0)
Hysteresis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

- **Function**
  - *EnableMask* is a 32-bits bool type parameter used to enable different tracks. The smallest valid bit indicates the first track. For example, if you want to disable the third track, the input value will be 16#FFFFFFFB.
  - MC_CAMSWITCH_REF defines switches for digital cam. *NoOfSwitches* calculates the number of switching positions. *CamSwitchPtr* is a pointer on an array of 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 is OFF after the time set in *Duration* when *CamSwitchMode* = 1.
- *Duration* defines when the switch will be turned off after it is turned on. Only works when *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 0.1s.
- *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 exceed 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 in the range.

◦ Multiple switch modes are allowed to be set in a single Track.

- **Troubleshooting**

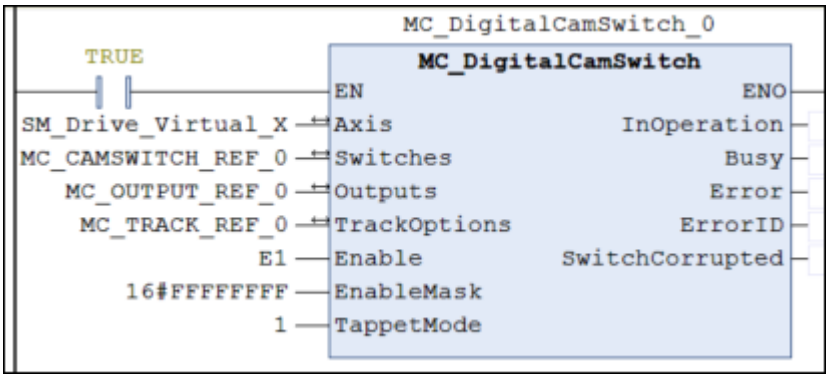
◦ If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

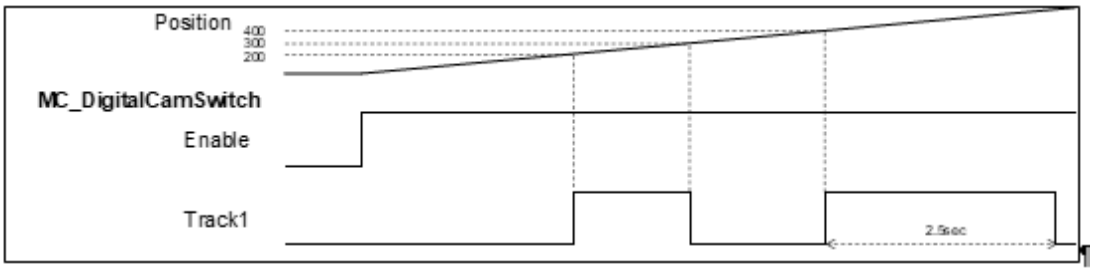
◦ **Example 1**: This example demonstrates the use of 2 switches in the same Track in MC_DigitalCamSwitch.

Parameter	Type	Switch1	Switch2
TrackNumber	INT	1	1
FirstOnPosition [u]	REAL	200	400
LastOnPosition [u]	REAL	300	-
AxisDirection	INT	0=Both	0=Both

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



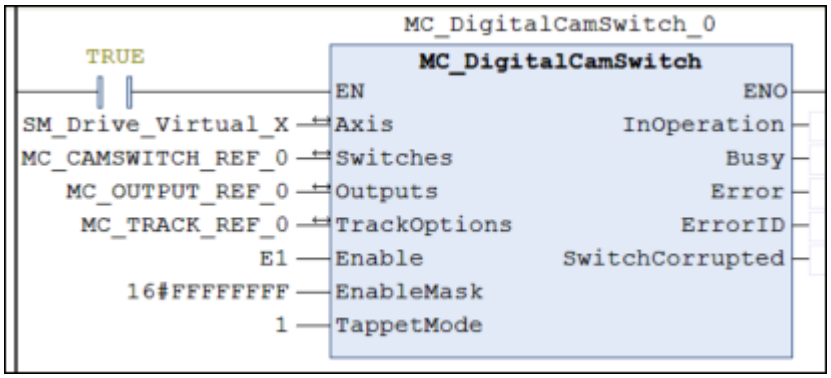
- Timing Diagram



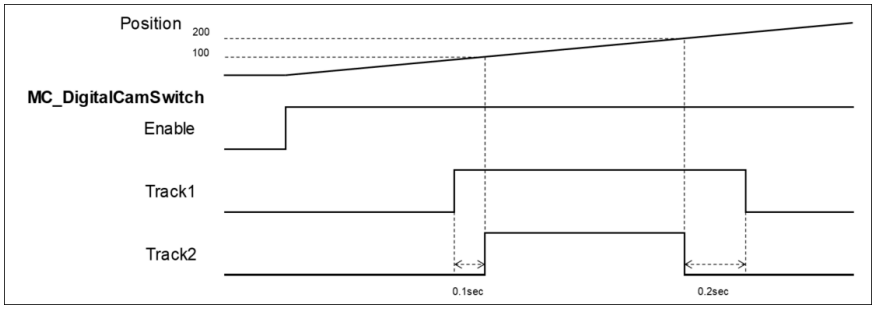
  - When the axis reaches the position 200, Switch1 on Track1 will turn ON, and Switch will turn OFF when the axis position reaches 300.
  - Switch1 turns on again when the position reaches 400, and Switch1 turns off after 2500ms.

- **Example 2**: This example demonstrates the operation result of *OnCompensation*/*OffCompensation* used in MC_DigitalCamSwitch.

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



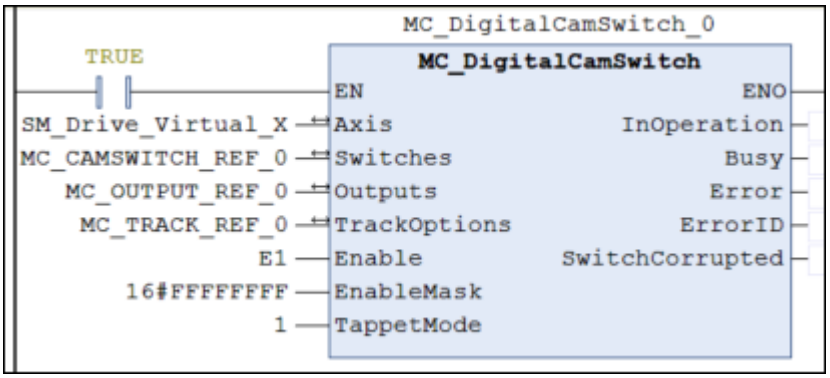
- Timing Diagram



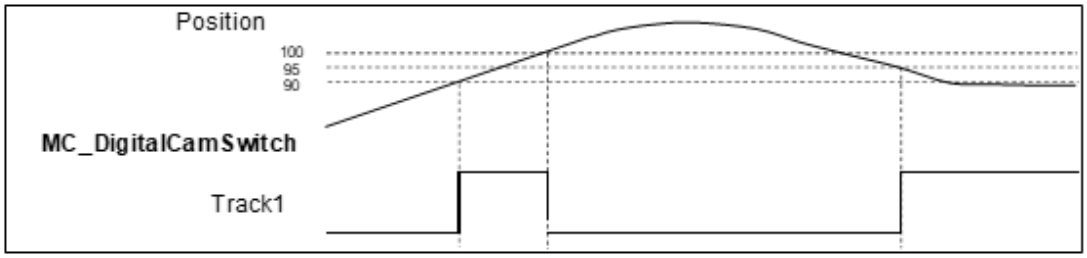
  - Switch1 on Track1 and Switch2 on Track2 turn on when the position reaches 100 and turn off when it reaches 200. OnCompensation = -0.1 for Switch1 means turning on 0.1 seconds earlier, and OffCompensation = 0.2 means turning off 0.2 second later.

- **Example 3**: This example demonstrates the operation result of Hysteresis used in MC_DigitalCamSwitch.

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

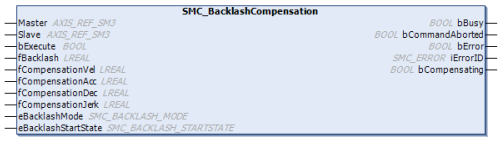


  - The *FirstOnPosition* and *LastOnPosition* of Switch 1 on Track1 are set to 90 and 95 respectively with *Hysteresis* set to 10, which means the switch will be turned off after the axis position passing the interval (80–100).
  - Track 1 is switched to ON when the axis reaches position 90 and not able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

- **Supported Devices**
  - AX-series motion controller

2.1.2.17 SMC_BacklashCompensation

SMC_BacklashCompensation is used to compensate for the backlash of gears.

FB/FC	Instruction	Graphic Expression	ST Language
FB	SMC_Backlash Compensation		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	Runs the instruction when *Execute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
fBacklash	The backlash distance to be compensated	LREAL	Positive, negative, or 0	*bExecute* changes from FALSE to TRUE.
fCompensationVel	The speed when compensating for backlash	LREAL	Positive or 0	*bExecute* changes from FALSE to TRUE.
fCompensationAcc	The acceleration when compensating for backlash	LREAL	Positive or 0	*bExecute* changes from FALSE to TRUE.
fCompensationDec	The deceleration when compensating for backlash	LREAL	Positive or 0	*bExecute* changes from FALSE to TRUE.
fCompensationJerk	The jerk when compensating for backlash	LREAL	Positive or 0	*bExecute* 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)	*bExecute* changes from FALSE to TRUE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
eBacklashStartState	The start condition 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)	*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)
bBusy	TRUE when the instruction is running	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 within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.		SMC_ERROR (SMC_NO_ERROR)
bCompensating	TRUE when compensating for backlash	BOOL	TRUE/FALSE (FALSE)

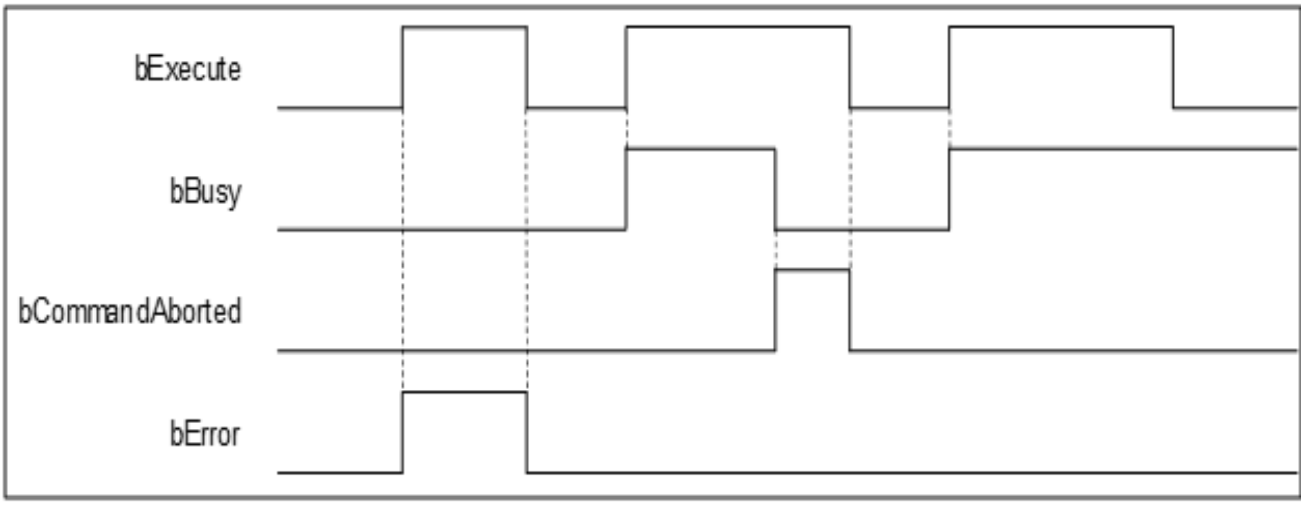
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	The instruction is running.	• *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bCommandAborted	• MC_GearOut is run. • The instruction is aborted. • The function block instruction is aborted by MC_Stop.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bCompensating	Backlash compensation is undergoing.	Backlash compensation is not performed.

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	The specified master axis		AXIS_REF_SM3	*bExecute* turns to TRUE and *bBusy* is FALSE.
Slave	The specified slave axis		AXIS_REF_SM3	*bExecute* turns to TRUE and *bBusy* is FALSE.

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

- **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 performed when the axis is running in the negative direction.
SMC_BL_OFF	No backlash compensation
SMC_BL_POSITIVE	Backlash compensation is performed when the axis is running in the positive direction.
SMC_BL_AUTO	Backlash is compensation 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.

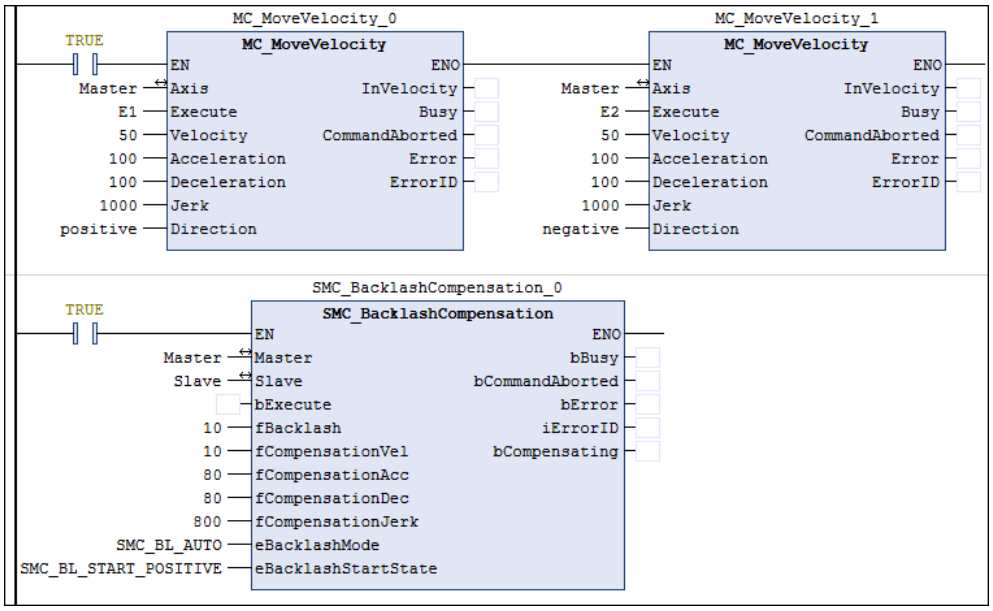
The Initial State of the Master and Slave Axes	Description
SMC_BL_START_POSITIVE	Positive traction is initially applied from the shaft to the main shaft.

- When SMC_BacklashCompensation is executed, 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, and *bCompensating* will not turn to TRUE.

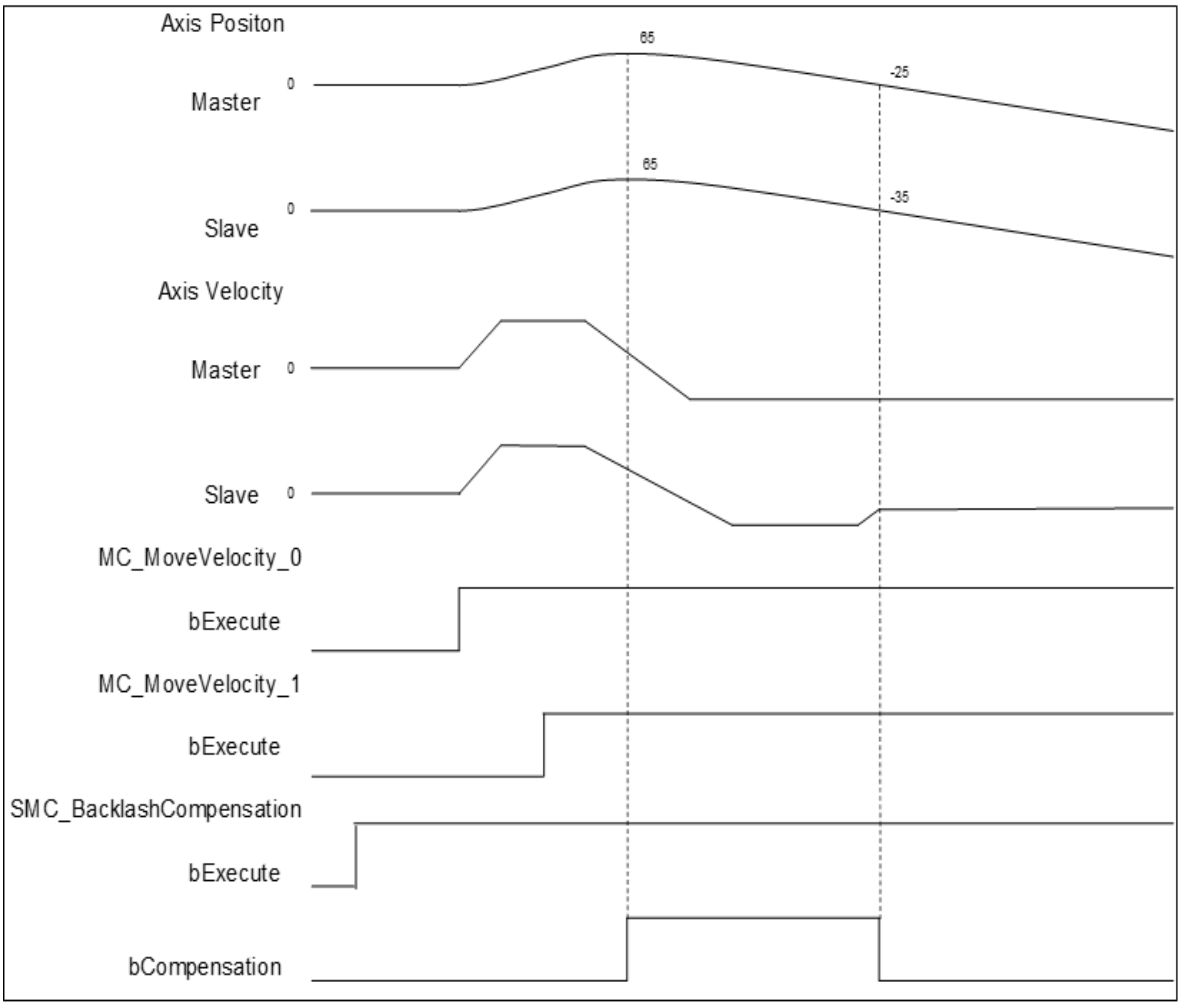
The Initial State of the Master and Slave Axes	Compensation Mode	Behavior Pattern
SMC_BL_START_NONE	SMC_BL_POSITIVE	When the function block starts, but the master axis is stationary, the slave axis will compensate positively to the set value of *fBacklash*. Assuming *fBacklash* = 10, after the function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

- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

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



- Timing Diagram



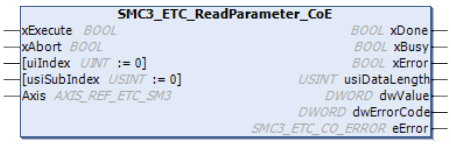
1. Run the SMC_BacklashCompensation first, then move forward, and then reverse to observe the backlash compensation.
2. Since the SMC_BACKLASH_MODE is set to SMC_BL_AUTO and the SMC_BACKLASH_STARTSTATE is set to SMC_BL_START_POSITIVE, the slave axis will not perform the compensation first.
3. The MC_MoveVelocity_0 is performed first, at which point the positive motion is carried out. Since the master axis applies positive traction to the slave axis at the beginning, there is no need to compensate the backlash.

4. The MC_MoveVelocity_1 is performed immediately after performing MC_MoveVelocity_0. The negative motion 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 synchronized. bCompensating is TRUE during the period of backlash compensation.

- **Supported Devices**
  - AX-series motion controller

2.1.2.18 SMC3_ETC_ReadParameter_CoE

SMC3_ETC_ReadParameter_CoE reads the slave object dictionary. (Applies to SM3_Drive_ETC library)

FB/FC	Instruction	Graphic Expression	ST Language
FB	SMC3_ETC_ReadParameter_CoE		SMC3_ETC_ReadParameter_CoE_instance (xExecute:= , xAbort:= , uiIndex:= , usiSubIndex:= , Axis:= , xDone=> , xBusy=> , xError=> , usiDataLength=> , dwValue=> , dwErrorCode=> , eError=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
xExecute	Runs the instruction when *xExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
xAbort	Stop reading parameters.	BOOL	TRUE/FALSE (FALSE)	*xExecute* changes from FALSE to TRUE.
uiIndex	Object dictionary index	UINT	Positive or 0	*xExecute* changes from FALSE to TRUE.
usiSubIndex	Object dictionary subindex	USINT	Positive or 0	*xExecute* changes from FALSE to TRUE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
xDone	TRUE when the parameter reading is complete	BOOL	TRUE/FALSE (FALSE)
xBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
xError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
usiDataLength	The length of the data read The unit is byte.	USINT	TRUE/FALSE (FALSE)
dwValue	Read the value of the parameter.	DWORD	Positive or 0 (0)

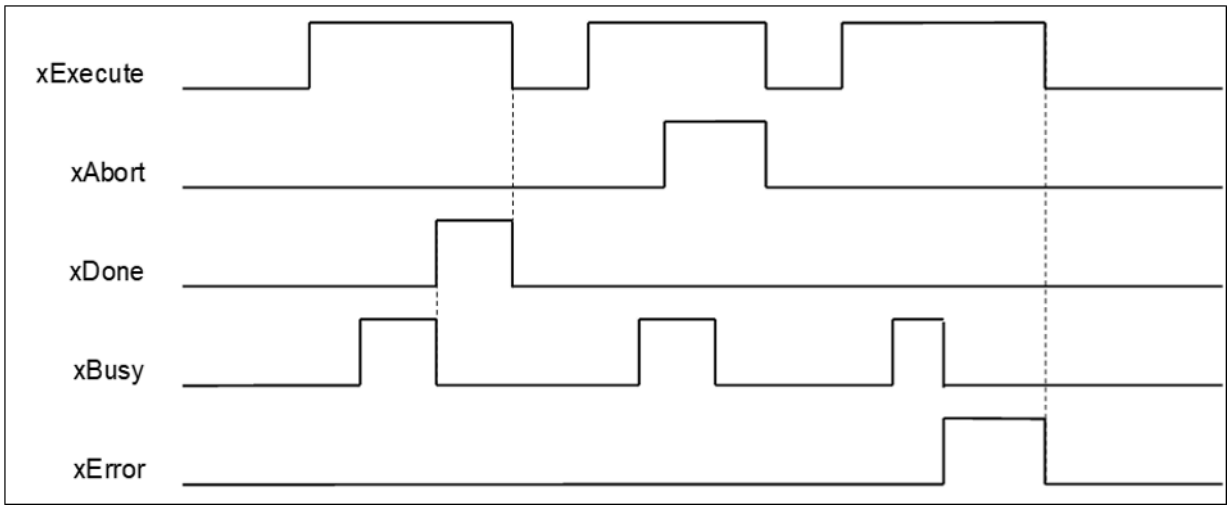
Name	Function	Data Type	Output Range (Default Value)
dwErrorCode	SDO error code	DWORD	Positive or 0 (0)
eError	Records the error code. Refer to **Appendix** for descriptions.		SMC3_ETC_CO_ERROR (SMC3_ETC_CO_NO_ERROR)

***Note**: SMC3_ETC_CO_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
xDone	*xExecute* turns to TRUE.	• *xExecute* turns to FALSE. • *xError* turns to TRUE.
xBusy	*xExecute* turns to TRUE.	• *xExecute* turns to FALSE. • *xError* turns to TRUE.
usiDataLength	*xDone* turns to TRUE.	• *xExecute* turns to FALSE. • *xError* turns to TRUE.
dwValue	*xDone* turns to TRUE.	• *xExecute* turns to FALSE. • *xError* turns to TRUE.
xError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*xExecute* turns to FALSE. (dwErrorCode and error codes are cleared.)
dwErrorCode		
eError		

- **Timing Diagram of Output Parameter Changes**



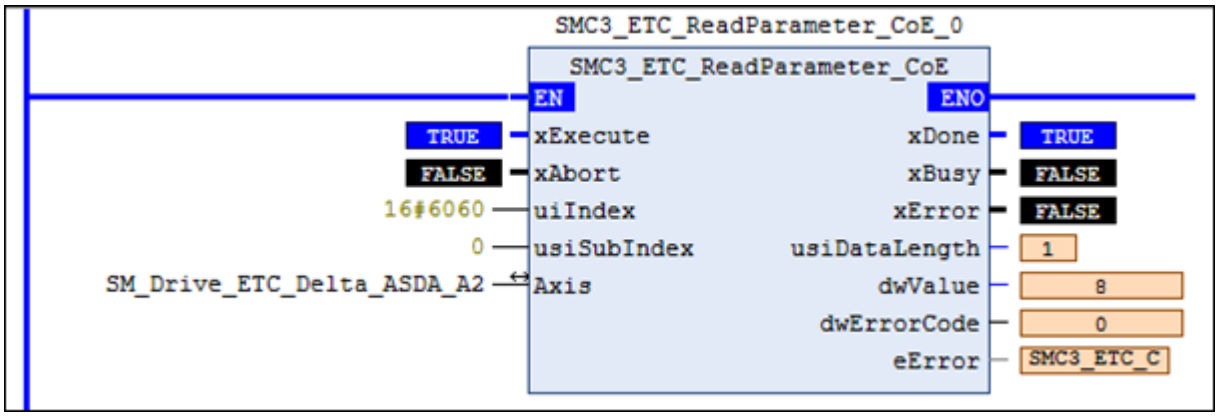
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified master axis		AXIS_REF_SM3	*xExecute* turns to TRUE and *xBusy* is FALSE.

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

- **Function**
  - Use SMC3_ETC_ReadParameter_CoE to read the EtherCAT Object Dictionary.
  - To set the range of uiIndex and usiSubIndex, refer to the slave manual.
- **Troubleshooting**
  - If an error occurs while running the instruction, *xError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  This example shows how to use SMC3_ETC_ReadParameter_CoE to read the values of the 0x6060 (mode setting) object in a drive.



  - The following figure shows the parameters of 0x6060 object dictionary.

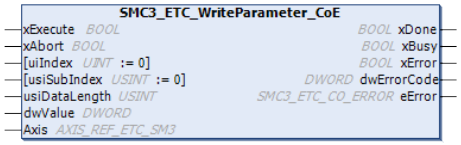
Object 6060_h: Modes of operation

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 1: Profile position mode 3: Profile velocity mode 4: Profile torque mode 6: Homing mode 7: Interpolated position mode 8: Cyclic synchronous position mode 9: Cyclic synchronous velocity mode 10: Cyclic synchronous torque mode

- **Supported Devices**
  - AX-series motion controller

2.1.2.19 SMC3_ETC_WriteParameter_CoE

SMC3_ETC_WriteParameter_CoE writes to the slave object dictionary. (Applies to SM3_Drive_ETC library)

FB/FC	Instruction	Graphic Expression	ST Language
FB	SMC3_ETC_WriteParameter_CoE		SMC3_ETC_WriteParameter_CoE_instance (xExecute:= , xAbort:= , uiIndex:= , usiSubIndex:= , usiDataLength:= , dwValue:= , Axis:= , xDone=> , xBusy=> , xError=> , dwErrorCode=> , eError=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
xExecute	Runs the instruction when *xExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
xAbort	Stops writing parameters.	BOOL	TRUE/FALSE (FALSE)	*xExecute* changes from FALSE to TRUE.
uiIndex	Object dictionary index	UINT	Positive or 0	*xExecute* changes from FALSE to TRUE.
usiSubIndex	Object dictionary subindex	USINT	Positive or 0	*xExecute* changes from FALSE to TRUE.
usiDataLength	The length of the data read The unit is byte (Range 1–4)	USINT	Positive or 0	*xExecute* changes from FALSE to TRUE.
dwValue	The value of the parameter to write	DWORD	Positive or 0	*xExecute* changes from FALSE to TRUE.

- **Outputs**

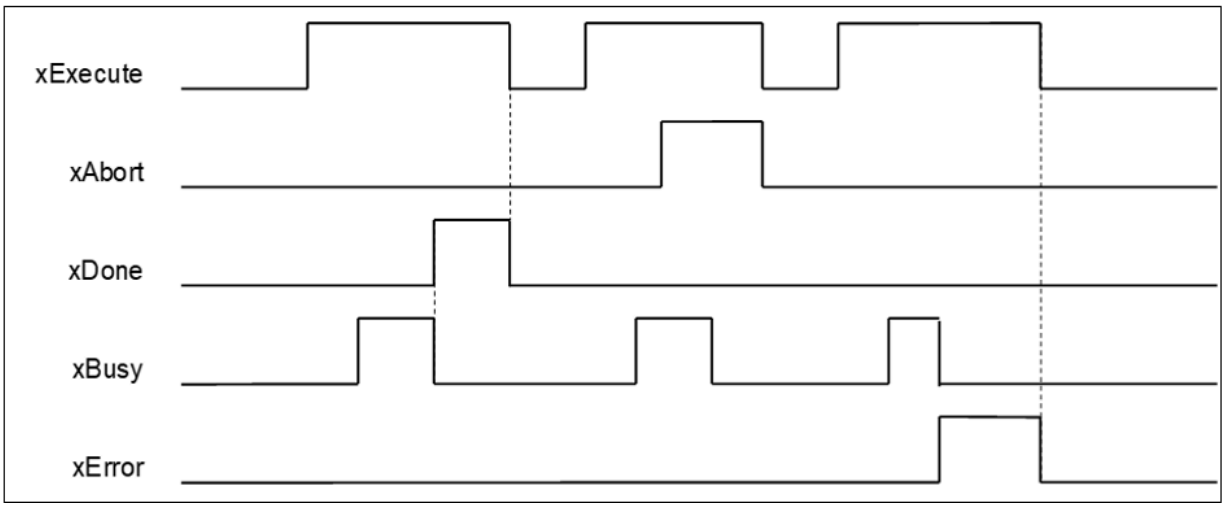
Name	Function	Data Type	Output Range (Default Value)
xDone	TRUE when writing parameter is complete	BOOL	TRUE/FALSE (FALSE)
xBusy	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
xError	TRUE when an error occurs	BOOL	TRUE/FALSE (FALSE)
dwErrorCode	SDO error code	DWORD	Positive or 0
eError	Records the error code. Refer to **Appendix** for descriptions.		SMC3_ETC_CO_ERROR (SMC3_ETC_CO_NO_ERROR)

***Note**: SMC3_ETC_CO_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
xDone	*xExecute* turns to TRUE.	• *xExecute* turns to FALSE. • *xError* turns to TRUE.
xBusy	*xExecute* turns to TRUE.	• *xExecute* turns to FALSE. • *xError* turns to TRUE.
xError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*xExecute* turns to FALSE. (dwErrorCode and error codes are cleared)
dwErrorCode		
eError		

- **Timing Diagram of Output Parameter Changes**



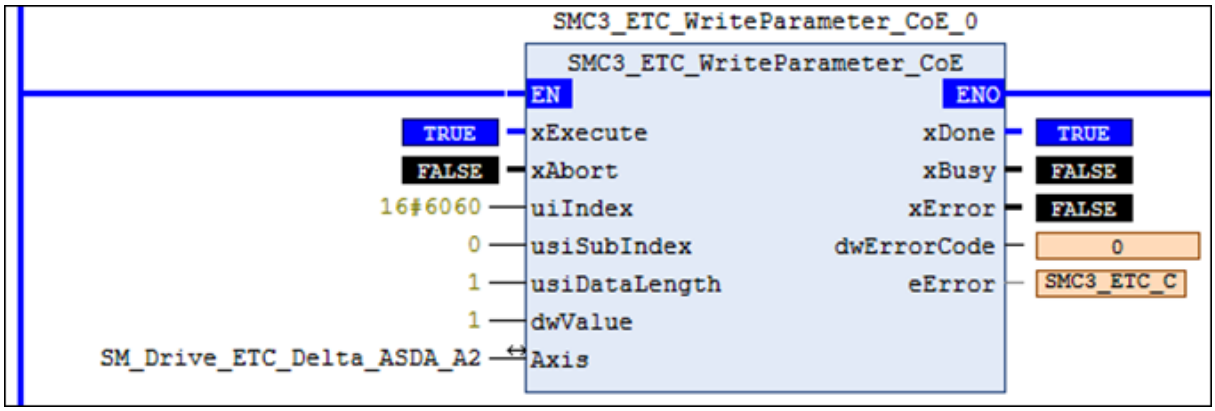
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified master axis		AXIS_REF_SM3	*xExecute* turns to TRUE and *xBusy* is FALSE.

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

- **Function**
  - Use SMC3_ETC_WriteParameter_CoE to write to the EtherCAT Object Dictionary.
  - To set the range of uiIndex and usiSubIndex, refer to the slave manual.
- **Troubleshooting**
  - If an error occurs while running the instruction, *xError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  This example shows how to use SMC3_ETC_WriteParameter_CoE to write values to the 0x6060 (mode setting) object in a drive.



  - The following figure shows the parameters of 0x6060 object dictionary.

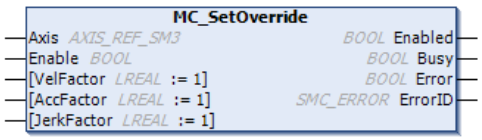
Object 6060_h: Modes of operation

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 1: Profile position mode 3: Profile velocity mode 4: Profile torque mode 6: Homing mode 7: Interpolated position mode 8: Cyclic synchronous position mode 9: Cyclic synchronous velocity mode 10: Cyclic synchronous torque mode

- **Supported Devices**
  - AX-series motion controller

2.1.2.20 MC_SetOverride

MC_SetOverride changes the target axis speed by overriding control factors.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_SetOverride		MC_SetOverride(Axis:= , Enable:= , VelFactor:= , AccFactor:= , JerkFactor:= , Enabled=> , Busy=> , Error=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	Runs the instruction when *Enable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
VelFactor	Velocity factor (Unit: %)	LREAL	0-1 (1)	*Enable* turns to TRUE.
AccFactor	Acceleration factor (Unit: %)	LREAL	0-1 (1)	*Enable* turns to TRUE.
JerkFactor	Jerk factor (Unit: %)	LREAL	0-1 (1)	*Enable* turns to TRUE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Enabled	TRUE when processing the control	BOOL	TRUE/FALSE (FALSE)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

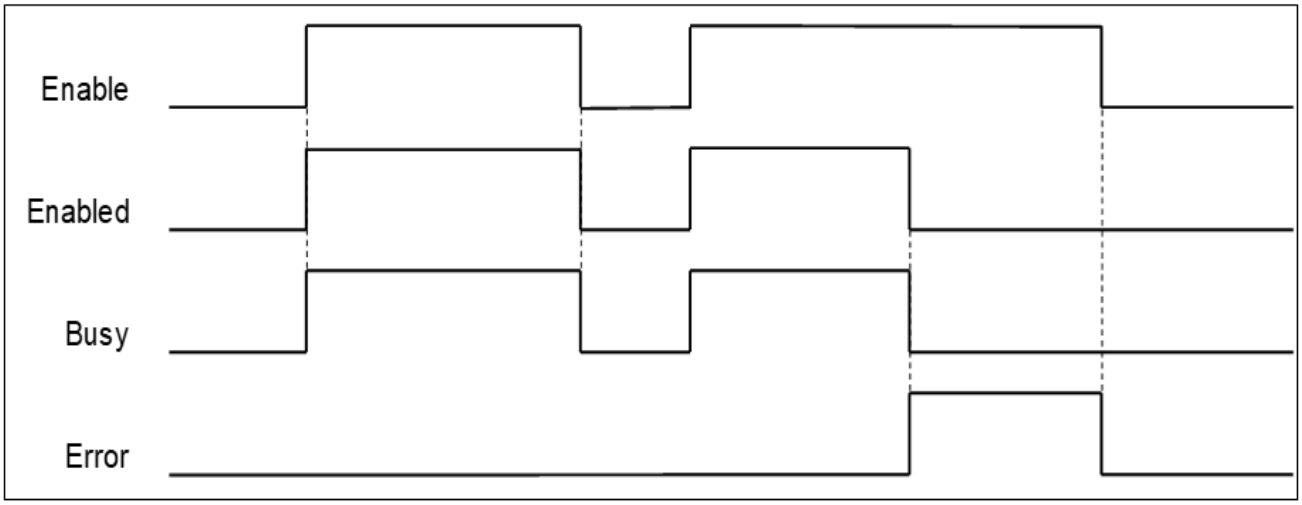
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Enabled	*Enable* is triggered to TRUE.	• *Enable* turns to FALSE. • *Error* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Busy	*Enable* is triggered to TRUE.	• *Enable* turns to FALSE. • *Error* turns to TRUE.
Error ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Enable* turns to FALSE.

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Enable* turns to TRUE.

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

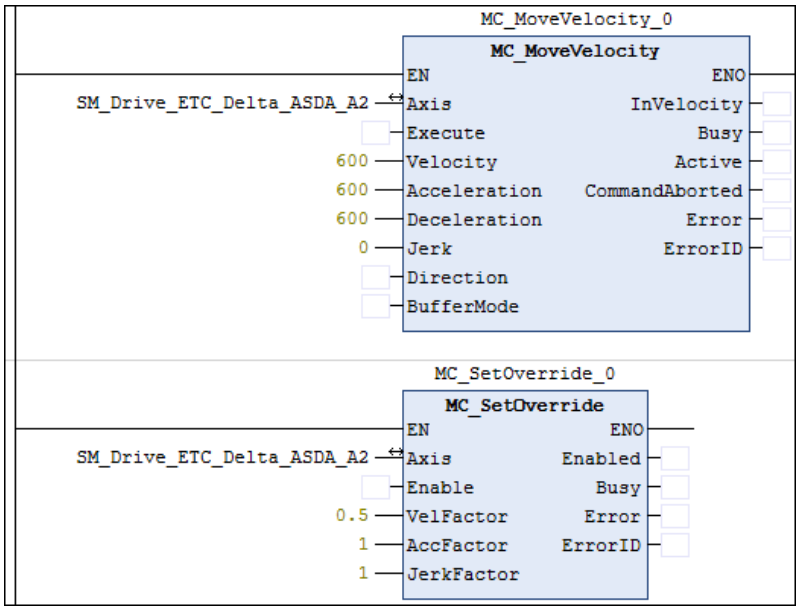
- **Function**
  - This function is supported in SoftMotion V4.16.0.0 or later.
  - The target velocity can be calculated by the following formula.

    New Target Velocity = Current Target Velocity × The Override Factor
  - When *Enable* is TRUE, the inputs of *VelFactor*, *AccFactor*, and *JerkFactor* can be continuously updated.
  - When an error occurs, *VelFactor*, *AccFactor*, and *JerkFactor* will not change.
  - You can temporarily stop the motion axis by setting the *VelFactor* value to 0. When *VelFactor* is set to 0, the target speed will be changed to 0, and the axis will slow down to 0 and remain controlled.
  - This function cannot change the velocity of the slave axis in the synchronous motion of the master and slave axes.
  - This function does not change the velocity of a single axis in the axis group movement.

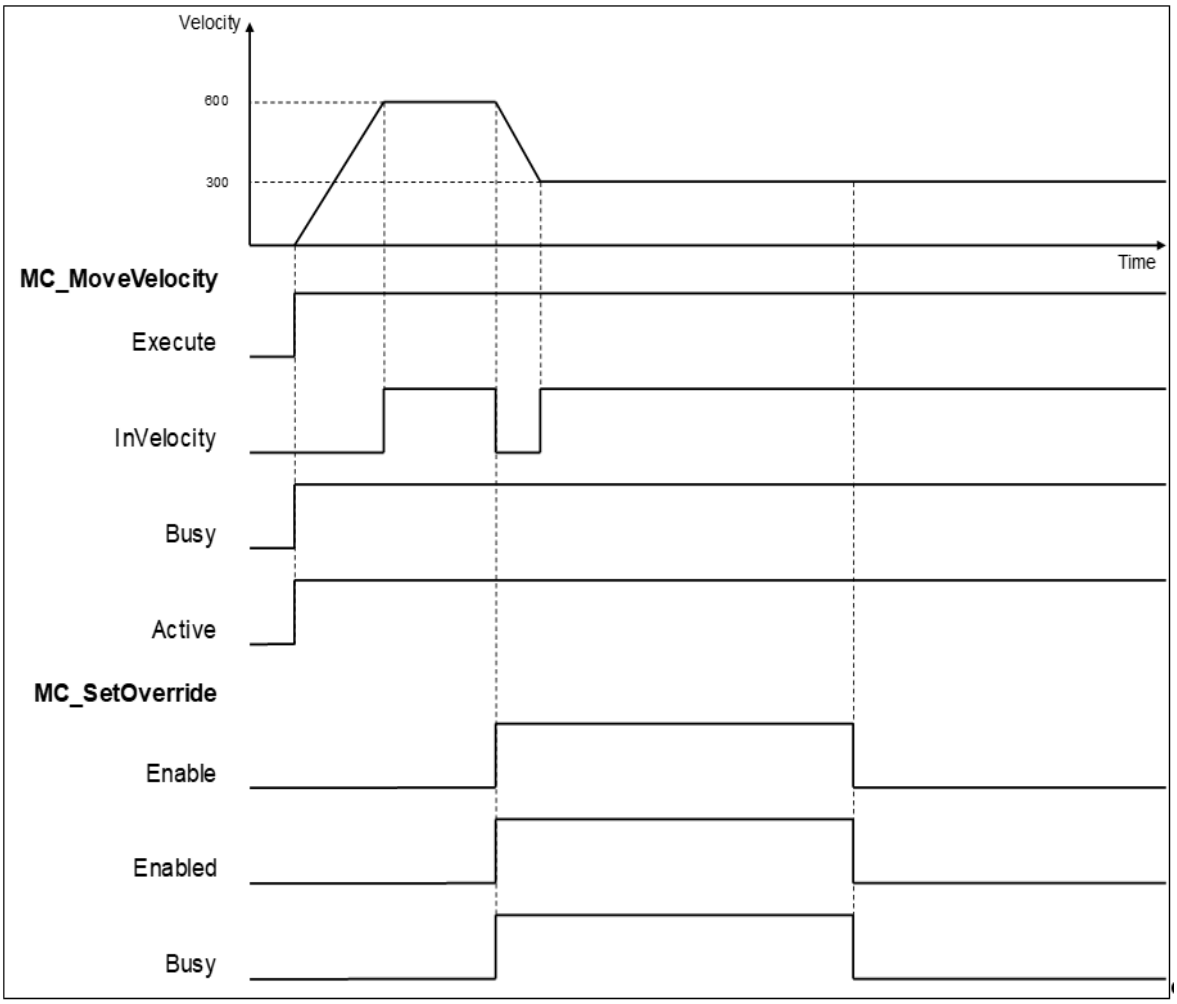
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

The following example explains how to use MC_SetOverride to change the velocity.



- Timing diagram



1. When the MC_MoveVelocity reaches the target speed, the MC_SetOverride starts, *MC_SetOverride.VelFactor* is set to 0.5, MC_SetOverride instruction will change the target speed from 600 to 300. When the MC_MoveVelocity's target speed reaches the new target speed of 300, *MC_MoveVelocity.InVelocity* turns to TRUE.
2. When *MC_SetOverride.Enable* is set to FALSE, the target speed of the axis remains unchanged at 300.

- **Supported Devices**
  - AX-seriers motion controllers

2.2 DL_MotionControl

2.2.1 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.
- W3 firmware version V1.0006 slave devices are supported in V1.4.0.0 and V1.0006.

2.2.1.1 DMC_TorqueControl

DMC_TorqueControl controls the torque that applied to the servo drive.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_TorqueControl		DMC_TorqueControl_instance(Axis :=, bEnable :=, bContinuousUpdate :=, lrTorque :=, dwTorqueRamp :=, lrVelocity :=, lrAcceleration :=, lrDeceleration :=, lrJerk :=, Direction :=, lrZeroTolerance :=, lrInTorqueRange:= , DisableMode:= , bInTorque =>, bBusy =>, bCommandAborted =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	Runs the instruction 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)	*bEnable* turns to TRUE and *Busy* is FALSE.
lrTorque	Target torque (Unit: N.m)	LREAL	Negative, positive, or 0 (0)	*bEnable* turns to TRUE and *Busy* is FALSE.
dwTorqueRamp	Change rate of the torque (Unit: ms) [*2]	DWORD	Positive (0)	*bEnable* turns to TRUE and *Busy* is FALSE.
lrVelocity	Maximum velocity	LREAL	Positive (0)	*bEnable* turns to TRUE and *Busy* is FALSE.
lrAcceleration	Reserved	LREAL	-	-
lrDeceleration	Reserved	LREAL	-	-
lrJerk	Reserved	LREAL	-	-
Direction	Reserved	BOOL	-	-

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrZeroTolerance[3]	The value to disable the torque function (Unit: N.m.)	LREAL	Negative, positive, or 0 (0)	*bEnable* turns to TRUE and *Busy* is FALSE.
DisableMode[*4]	Specify the torque disable mode	DMC_DISABLEMODE	Positive or 0 (0)	*bEnable* turns to TRUE and *Busy* is FALSE.

***Note**:

1. Supported by DL_MotionControl version V1.0.1.0 and later. When *bContinuousUpdate* is TRUE, the torque and the maximum velocity can be modified.
2. Take ASDA-A2 as an example, the unit is μs (microsecond). For other devices, refer to the 0x6087 index in their object dictionaries.
3. Supported by DL_MotionControl version V1.3.1.0 and later. This parameter sets the range to turn *bBusy* FALSE: *acutalTorque* ± *ZeroTolerance*.
4. Supported by DL_MotionControl version V1.3.1.0 and later. When the torque function block is disabled, the current torque is retained.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bInTorque	TRUE when the target torque is reached	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

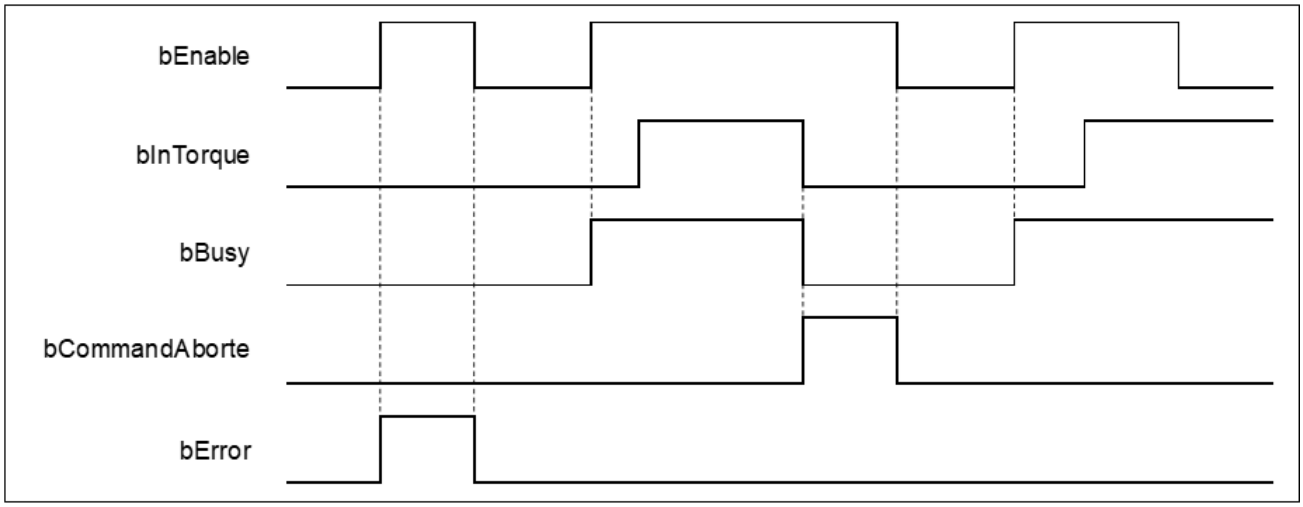
***Note**: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bInTorque	*bEnable* is TRUE and the axis motion state can be read.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bEnable* turns to TRUE.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bCommandAborted	This instruction is interrupted by another instruction.	*bEnable* turns to FALSE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* turns to FALSE (The value in ErrorID is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

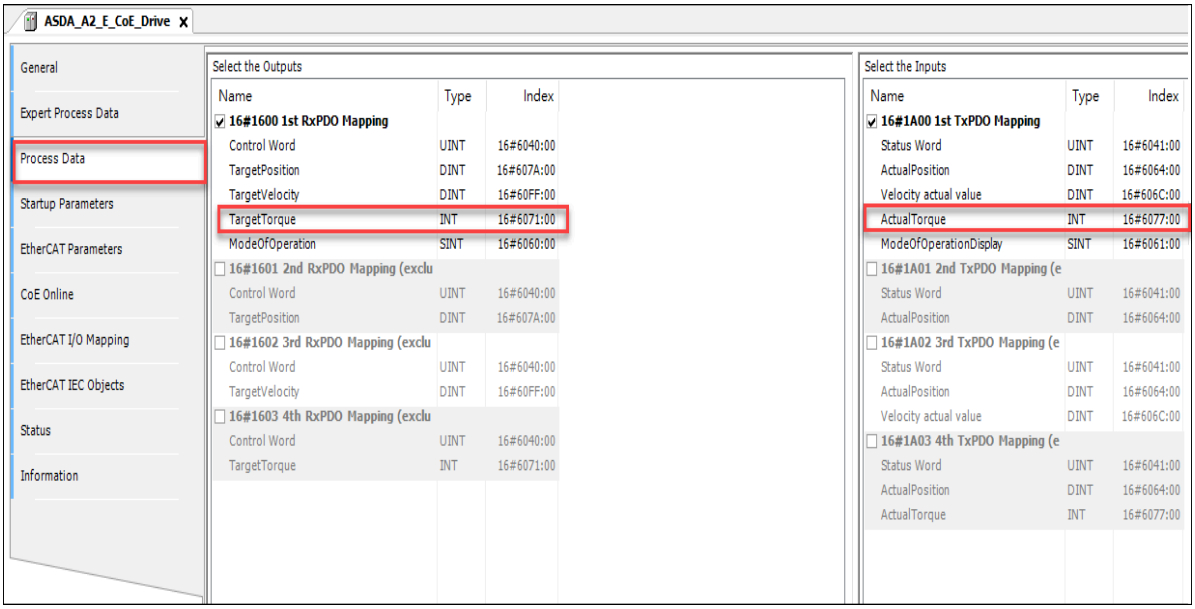
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	When *bEnable* turns to TRUE and *bBusy* is FALSE

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

- **Function**

  - This function block is not available in the PLC simulation mode. If used, the function block will report the DMC_TC_INVALID_PDO_MAPPING error.
  - 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 completes running when the axis decelerates to a stop, and *bBusy* turns to FALSE.
  - Ensure that the axis is in the Standstill state before executing the instruction.
  - When the instruction is running, do not use SMC_SetControllerMode. Otherwise, DMC_TorqueControl will be interrupted, and the servo will stop abruptly.
  - When running the DMC_TorqueControl, the MC_Power *Status* parameter will be FALSE and will not turn to TRUE until the DMC_TorqueControl is stopped. If you need to check whether the axis can be enabled, use the MC_ReadStatus function block to check whether the axis status is in Standstill.
  - One axis only allows only one DMC_TorqueControl to run at a time. If the second DMC_TorqueControl instruction is run at the same time, the DMC_TC_FB_CONFLICT error will occur.
  - When the *lrTorque* input exceeds the 0x6071 (Target Torque) range, it will be written as the maximum or minimum values of the OD data type.

- 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.

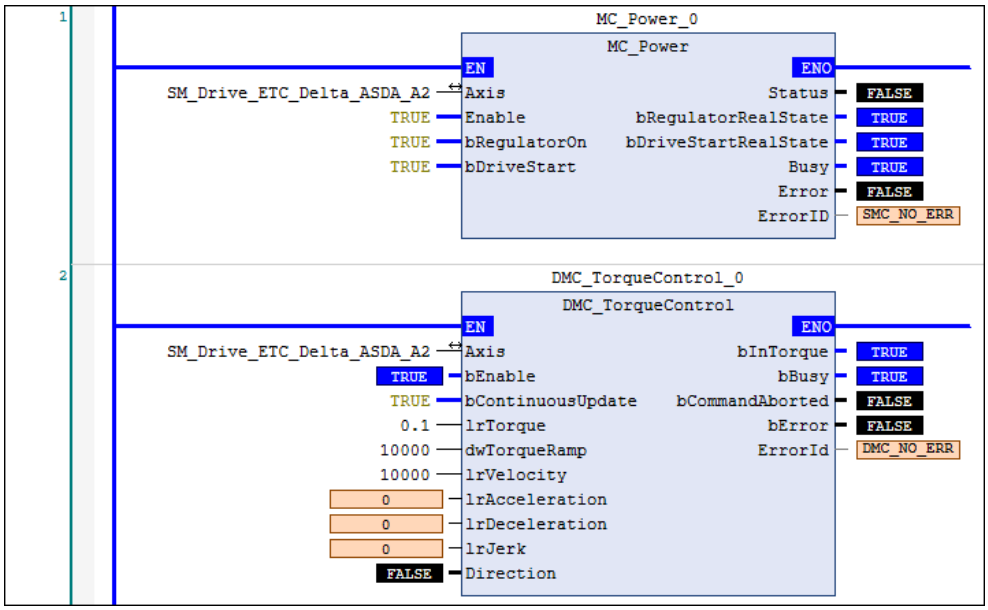


- bInTorque = TRUE, should be:
  - lrTorque > 0
    - ActualToruque >= lrTorque - lrTorqueRange
  - lrTorque < 0
    - The axis variable ActualToruque >= lrTorque + lrTorqueRange
  - lrTorque = 0
    - -lrTorqueRange <= the axis parameter ActualToruque <= lrTorqueRange

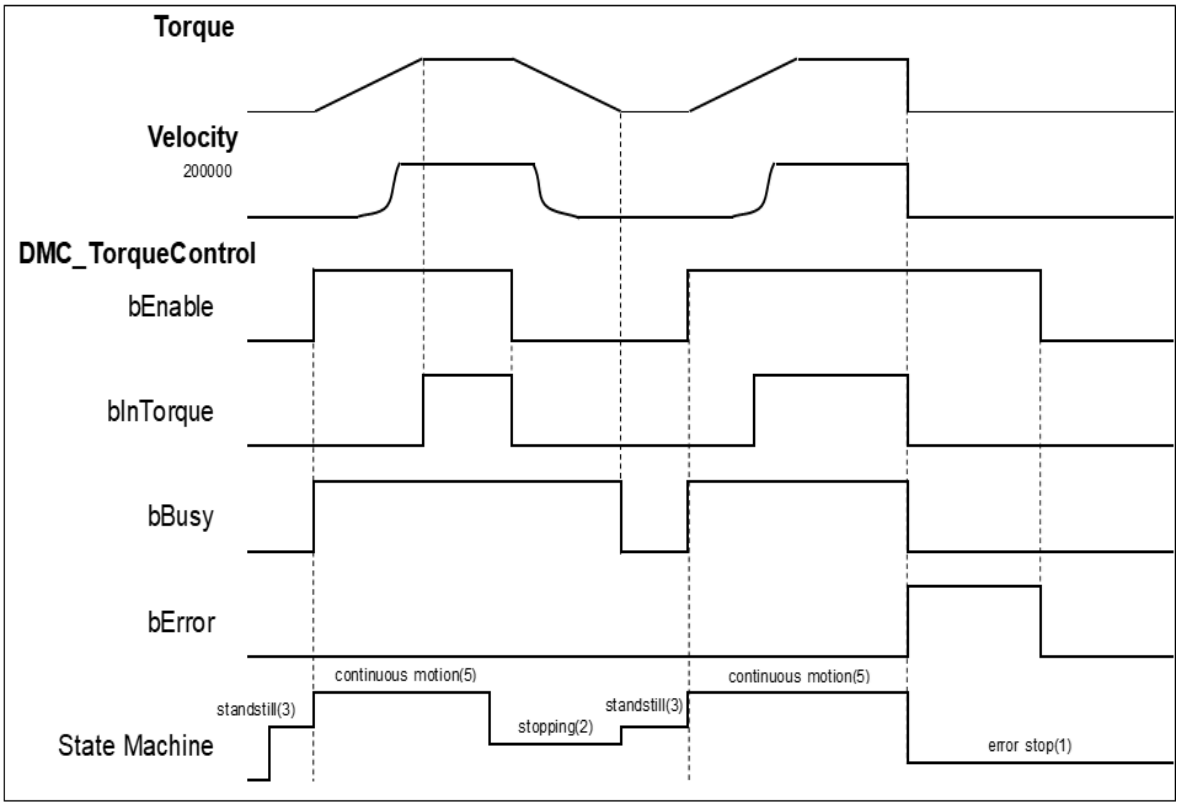
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

  This example shows the motion behavior performed by DMC_TorqueControl.



- Timing Diagram

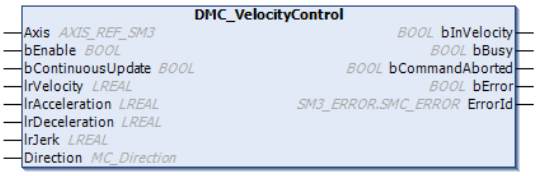


  - After DMC_TorqueControl is started, the servo starts to run according to the input settings of the instruction for the target torque *lrTorque*, change rate of the torque *dwTorqueRamp*, and maximum velocity *lrVelocity*.
  - After bEnable of DMC_TorqueControl turns to FALSE, the axis decelerates until 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.

- **Supported Devices**
  - AX-series motion controller

2.2.1.2 DMC_VelocityControl

DMC_VelocityControl controls the axis that is under the CSV mode to move in a specified velocity and mode, and maintain that velocity.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_VelocityControl		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	Runs the instruction when *Enable* 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)	*bEnable* turns to TRUE and *Busy* is FALSE.
lrVelocity	Target velocity (Unit: user unit/s)	LREAL	Positive (0)	*bEnable* turns to TRUE and *Busy* is FALSE.
lrAcceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bEnable* turns to TRUE and *Busy* is FALSE.
lrDeceleration	Deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bEnable* turns to TRUE and *Busy* is FALSE.
lrJerk	Jerk value (Unit: user unit/s^3)	LREAL	Positive (0)	*bEnable* turns to TRUE and *Busy* is FALSE.
Direction	Direction of the servo motor	MC_DIRECTION *2	3: fastest 2: current 1: positive 0: shortest -1: negative (current) *3	*bEnable* turns to TRUE and *Busy* is FALSE.

***Note**:

1. Supported by DL_MotionControl V1.2.0.0 and later. After *bContinuousUpdate* is started, change the speed. The acceleration, deceleration, and position will immediately take effect.
2. MC_DIRECTION: Enumeration (ENUM).
3. The values of fastest, current, and shortest can only used for the rotary axis.

- **Outputs**

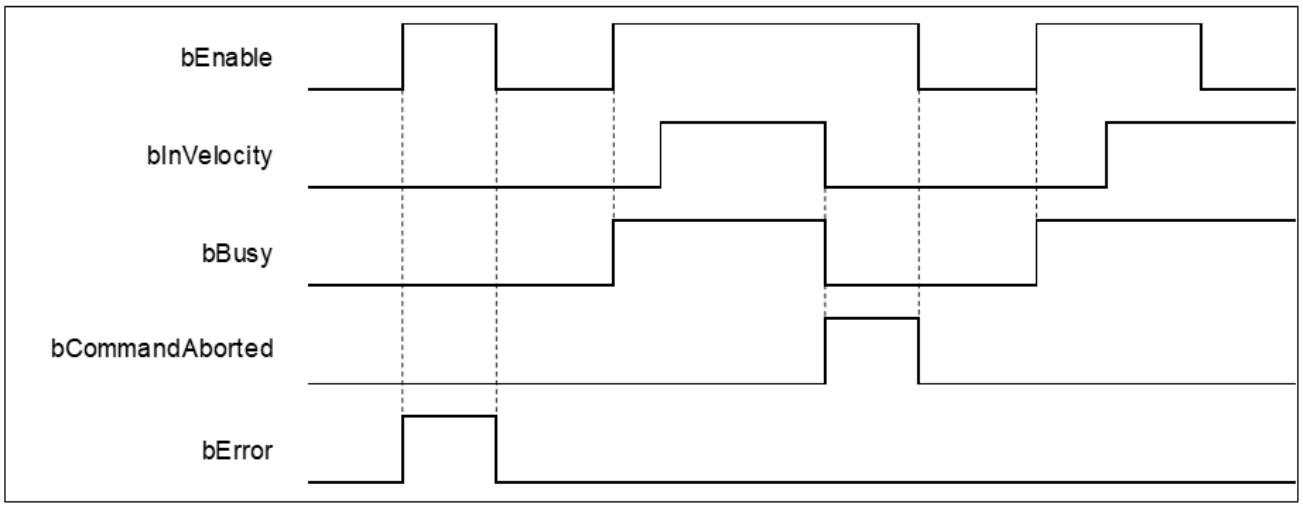
Name	Function	Data Type	Output Range (Default Value)
bInVelocity	TRUE when the specified target velocity is reached	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The specified target velocity is reached.	• *bCommandAborted* turns to TRUE. • *bContinuousUpdate* is TRUE and *IrVelocity* value is changed. • *bEnable* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bEnable* turns to TRUE.	• *bCommandAborted* turns to TRUE. • The axis decelerates to a stop after *bEnable* turns to FALSE. • *bError* turns to TRUE.
bCommandAborted	• This instruction is interrupted by another instruction. • This instruction is stopped by MC_Stop.	*bEnable* turns to FALSE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

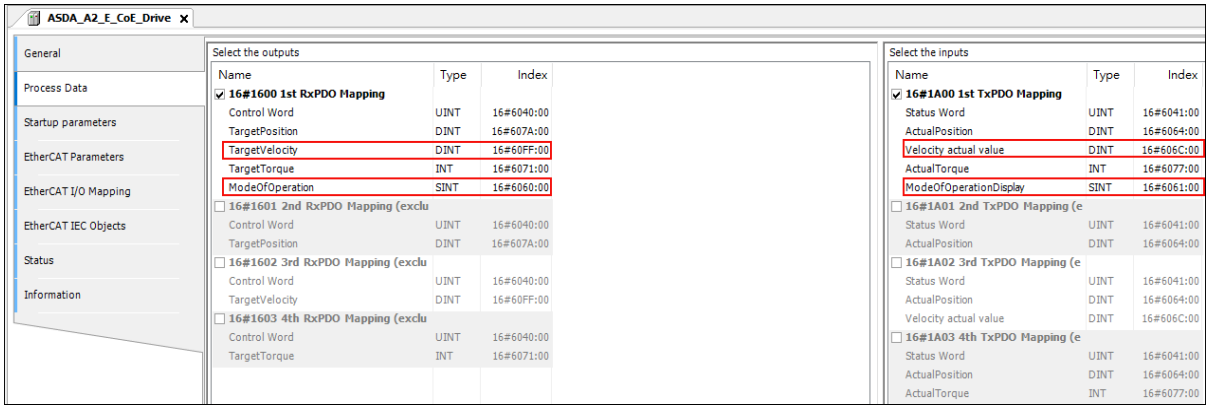
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bEnable* turns to TRUE and *bBusy* is FALSE.

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

- **Function**

  - This function block is not available in the PLC simulation mode. If used, the function block will report the DMC_VC_INVALID_PDO_MAPPING error.
  - When *bEnable* turns to TRUE, the instruction performs speed control with the target velocity (*lrVelocity*), acceleration rate (*lrAcceleration*), deceleration rate (*lrDeceleration*), and Jerk value (*lrJerk*).
  - You can run another motion instruction to interrupt the DMC_VelocityControl, but the servo will be in the CSV mode and will not switch to the CSP control mode.
  - When interrupted by another instruction, the output *bInVelocity* will turn to FALSE and the output *bCommandAborted* will turn to TRUE.
  - When the *bContinuousUpdate* input turns to TRUE and a new target velocity is given, the axis velocity is adjusted to the new one.
  - When *bEnable* turns to FALSE, the instruction will decelerate to stop, and will switch to the CSP control mode.
  - When the function block is used, the 0x60FF (Target Velocity), 0x606C (Velocity actual value), 0x6060 (ModeOfOperation), and 0x6061 (ModeOfOperationDisplay) OD must be included in the slave PDO (Process data) mapping data. Otherwise, the servo will not work.
  - It is not recommended to use this function directly on synchronous motion function blocks such as electronic cams and gears. DMC_VelocityControl is run in CSV mode, so if you want the axis to act as

the master axis electronic cam and electronic gear, the actual encoder feedback command needs to be connected back to the controller and controlled in a fully closed-loop architecture.

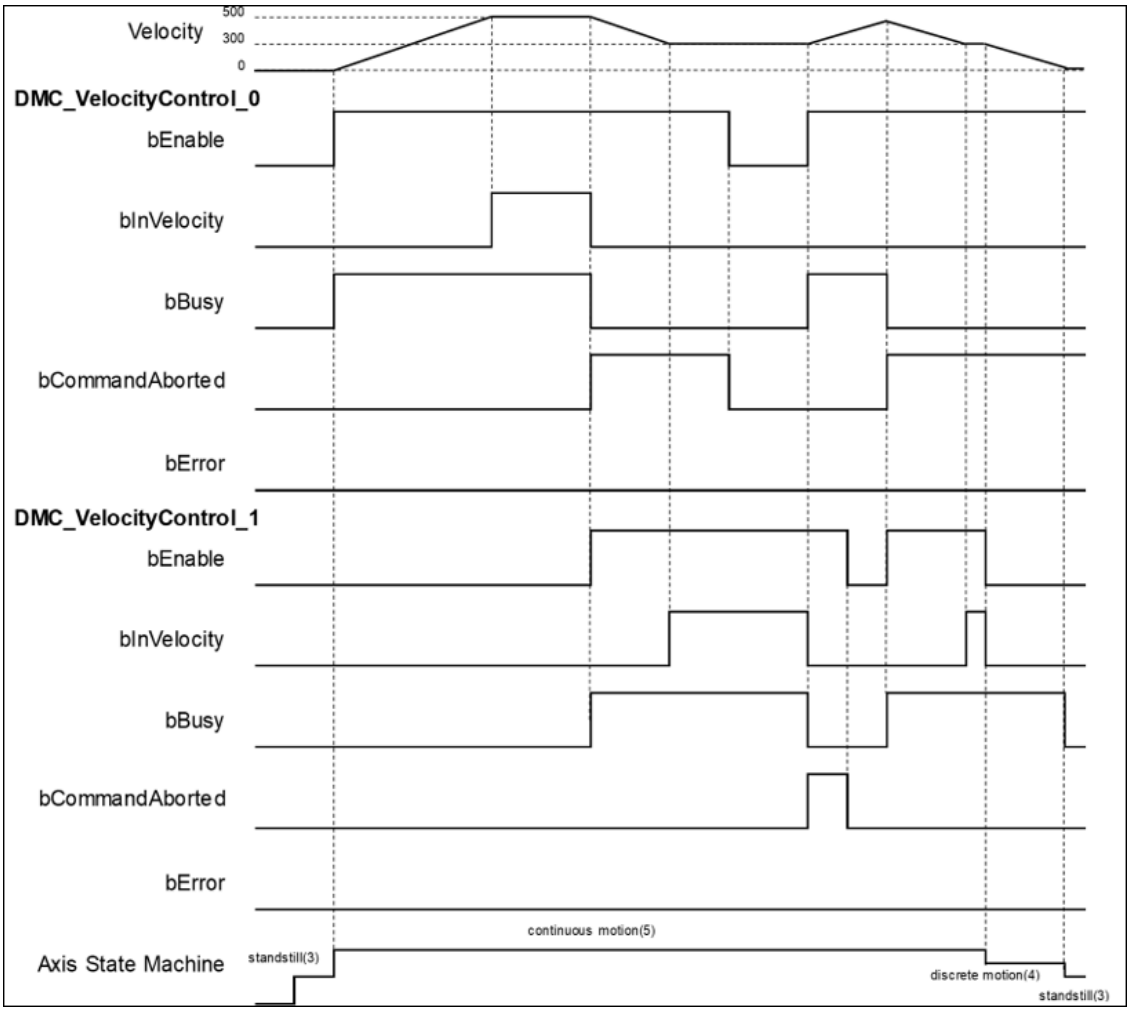


- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

The example shows the motion behavior performed by DMC_VelocityControl.

◦ Timing Diagram



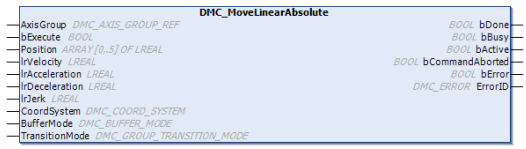
- When *bEnable* of DMC_VelocityControl_0 changes to TRUE, the instruction controls the axis to reach the specified target velocity 500. When it reaches 500, *bInVelocity* of DMC_VelocityControl_0 changes to TRUE.
- When *bEnable* of DMC_VelocityControl_1 changes to TRUE, DMC_VelocityControl_0 is interrupted, 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 by 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 *bBusy* of DMC_VelocityControl_1 changes to FALSE.

- **Supported Devices**
  - AX-series motion controller

2.2.1.3 DMC_MoveLinearAbsolute

DMC_MoveLinearAbsolute controls the axis group to move to the absolute position by linear interpolation.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_MoveLinearAbsolute		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Position	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])	*bExecute* turns to TRUE.
lrVelocity	The target velocity for the specified axis group (Unit: user unit/s)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrAcceleration	The acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrDeceleration	The deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrJerk	The jerk (Unit: user unit/s^3)	LREAL	Positive (0)	*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)	*bExecute* is on the rising edge, the setting parameters of CoordSystem will be updated.
BufferMode	Buffer mode for the instruction *1	DMC_BUFFER_MODE	0: Aborting 1: Buffered 2: BlendingLow	*bExecute* turns to TRUE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
			3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	
TransitionMode	Specify a transition mode for the instruction[*2].	DMC_GROUP_TRANSITION_MODE[*3]	0: None 10: Overlap 11：Single_axis (0)	*.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 (Default Value)
bDone	TRUE when the absolute positioning is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bActive	TRUE when the instruction is controlling axes	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR[*]	DMC_ERROR (DMC_NO_ERROR)

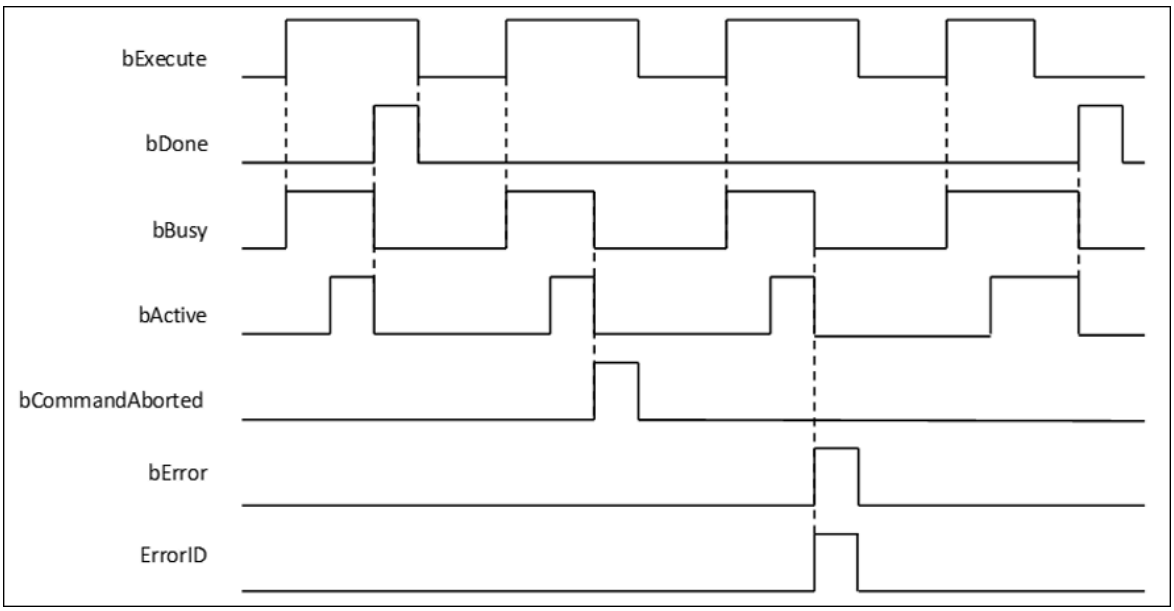
***Note**: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The absolute positioning is completed.	• *bExecute* turns to FALSE. • *bExecute* is FALSE but *bDone* turns to TRUE, *bDone* will remain TRUE for one scan cycle and then change to FALSE.
bBusy	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bActive	Axes start being controlled by the instruction.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *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.
bCommandAborted	• The instruction is interrupted by another instruction BufferMode of which is set to Aborting. • The instruction is interrupted by MC_Stop. • The instruction is interrupted by DMC_GroupStop.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID*)	*bExecute* turns from TRUE to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **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	*bExecute* turns to TRUE and *bBusy* is FALSE.

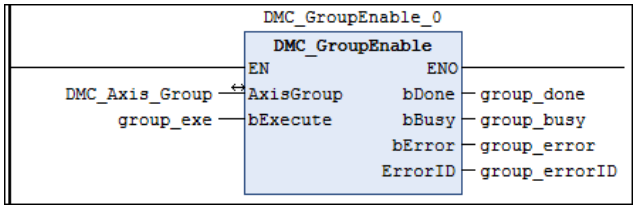
***Note**: DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

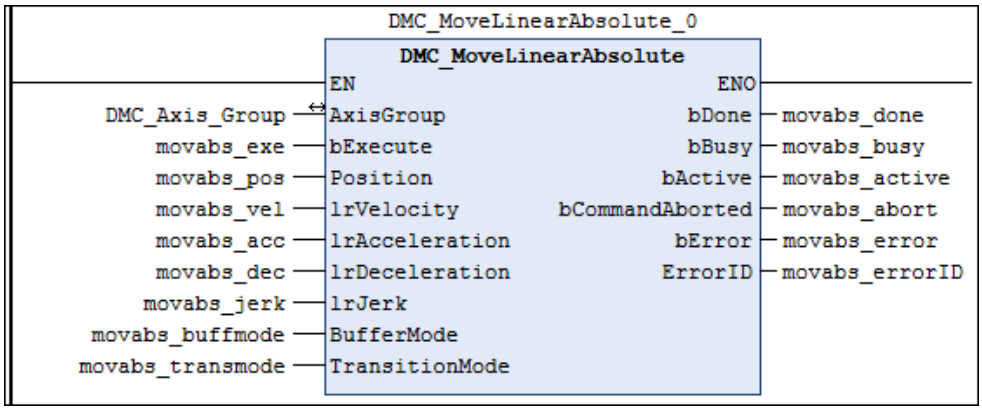
- **Function**
  - The instruction supports up to six axes to move to the absolute position by the linear interpolation. The six axes will start, stop, and reach the target absolute position at the same time.
  - At least one axis is needed for the linear interpolation. An error will occur if there is a travel distance for the axis which is not set.
  - The function of CoordSystem is supported by DL_MotionControl library V1.1.0.0 or later.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding troubleshooting, refer to **Appendix** for error code descriptions.
- **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 position.

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;
```



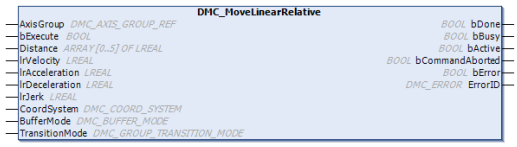


- When moveabs_exe (*bExecute*) changes to TRUE, DMC_MoveLinearAbsolute starts to perform the absolute linear interpolation for six axes.
- When moveabs_done (*bDone*) changes to TRUE, moveabs_busy (*bBusy*) changes to FALSE, which means the specified absolute positioning (1000, 2000, 3000, 4000, 5000, 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, no axes will move.

- **Supported Devices**
  - AX-series motion controller

2.2.1.4 DMC_MoveLinearRelative

DMC_MoveLinearRelative controls the axis group to move to the relative position by linear interpolation.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_MoveLinearRelative		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Distance	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])	*bExecute* turns to TRUE.
lrVelocity	The target velocity for the axis group (Unit: user unit/s)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrAcceleration	The acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrDeceleration	The deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrJerk	The jerk (Unit: user unit/s^3)	LREAL	Positive (0)	*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)	*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
			5: TCS (Reserved) (1)	
BufferMode	Buffer mode for the instruction *1	DMC_BUFFER_MODE	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	*bExecute* turns to TRUE.
TransitionMode	Transition mode for the instruction *2	DMC_GROUP_TRANSITION_MODE	0: None 10: Overlap 11: Single_axis (0)	*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 (Default Value)
bDone	TRUE when the relative positioning is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bActive	TRUE when the instruction is controlling axes	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

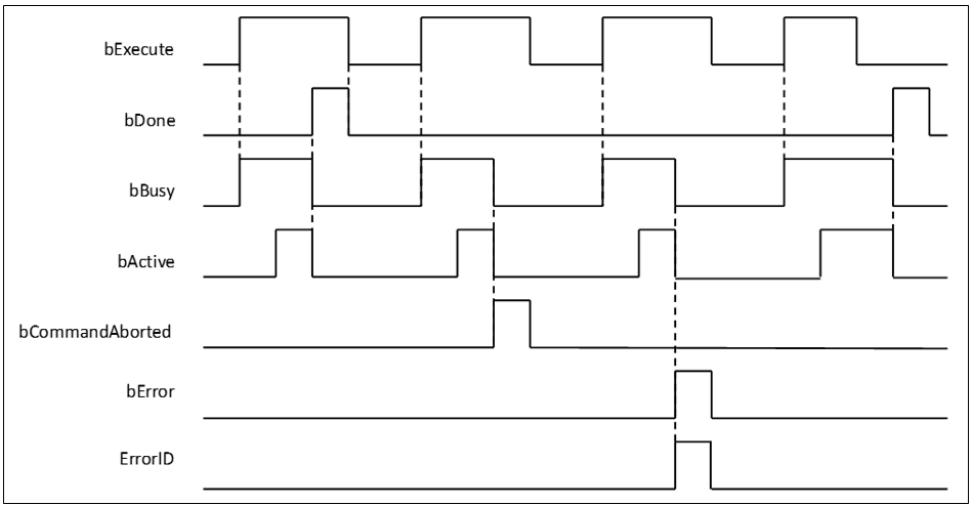
***Note**: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	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.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	*bExecute* turns to TRUE.	• When *bDone* turns to TRUE • When *bError* turns to TRUE • When *bCommandAborted* turns to TRUE
bActive	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.
bCommandAborted	• The instruction is interrupted by another instruction whose BufferMode is set to Aborting. • The instruction is interrupted by MC_Stop. • The instruction is interrupted by DMC_GroupStop.	• *bExecute* turns to FALSE. • *bExecute* is FALSE but *bCommandAborted* turns to TRUE, *bCommandAborted* will remain TRUE for one cycle and then change to FALSE.
bError/ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID*.)	*bExecute* turns to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

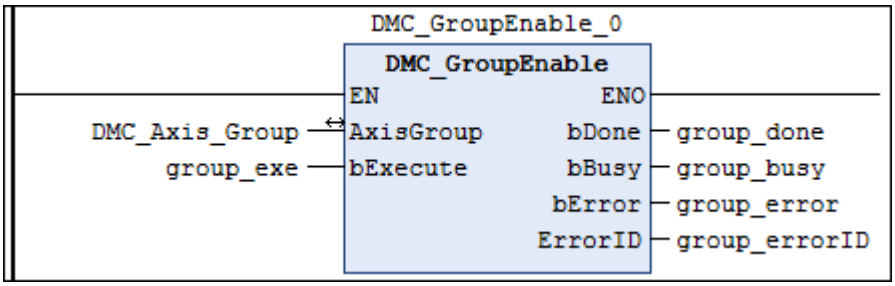
Name	Function	Data Type	Setting Value	Name
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	When *bExecute* turns to TRUE and *bBusy* is FALSE.

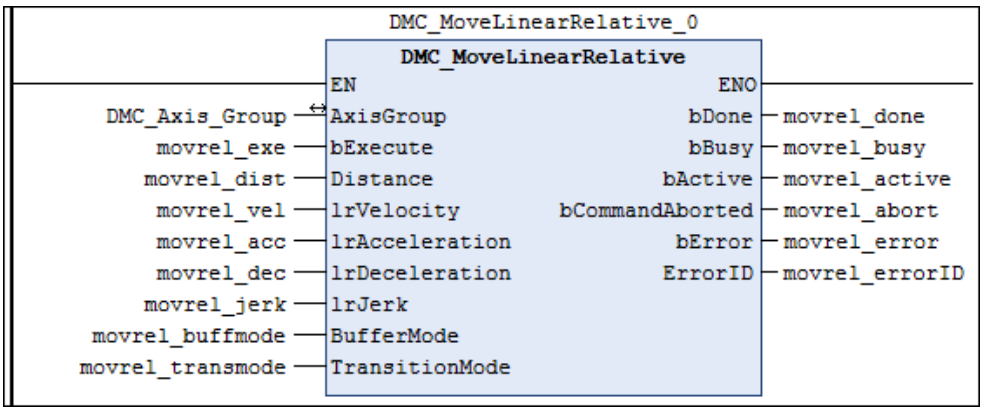
***Note**: DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**
  - The instruction supports up to six axes to move to the relative position by the linear interpolation. The six axes will start, stop, and reach the target absolute position at the same time.
  - 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 is supported by DL_MotionControl library V1.1.0.0 or later.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding trouble shootings, refer to **Appendix** for error code descriptions.
- **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 position.

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;
```





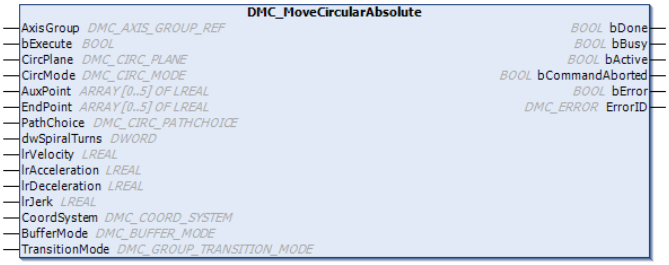
- When movrel_exe (*bExecute*) changes to TRUE, DMC_GroupRelLinear starts to perform the relative linear interpolation for six axes.
- When movrel_done (*bDone*) changes to TRUE, movrel_busy (*bBusy*) and movrel_abort (*bCommandAborted*) change to FALSE, which means the specified relative positioning (1000, 2000, 3000, 4000, 5000, 0) is completed.
- movrel_exe (*bExecute*) is switched to FALSE after the relative linear interpolation is completed. 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.

• Supported Devices

- AX-series motion controller

2.2.1.5 DMC_MoveCircularAbsolute

DMC_MoveCircularAbsolute controls the axis group to move to a absolute position by circular or helical interpolation.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_MoveCircularAbsolute		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
CircPlane	The circular or helical plane*1	DMC_CIRC_PLANE	0: XY_plane 1: YZ_plane 2: ZX_plane 3: ARBITRARY_plane (0)	*bExecute* turns to TRUE.
CircMode	The method for circular or helical interpolation*1	DMC_CIRC_MODE	0: Border 1: Center 2: Radius (0)	*bExecute* turns to TRUE.
AuxPoint	The auxiliary point data*1	LREAL[6]	[_, _, _, _, _, _] Positive, negative, or 0 ([0, 0, 0, 0, 0, 0])	*bExecute* turns to TRUE.
EndPoint	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])	*bExecute* turns to TRUE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
PathChoice	The circular or helical interpolation direction	DMC_CIRC_PATHCHOICE	0: Clockwise 1: CounterClockwise (0)	*bExecute* turns to TRUE.
dwSpiralTurns	The number of spiral turns	DWORD	0–65535 (0)	*bExecute* turns to TRUE.
lrVelocity	The target velocity for the axis group (Unit: user unit/s)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrAcceleration	The acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrDeceleration	The deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrJerk	The jerk (Unit: user unit/s^3)	LREAL	Positive (0)	*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)	*bExecute* is on the rising edge, the setting parameters of *CoordSystem* will be updated.
BufferMode	Buffer mode for the instruction*2	DMC_BUFFER_MODE	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	*bExecute* turns to TRUE.
TransitionMode	Transition mode for the instruction*3	DMC_GROUP_TRANSITION_MODE	0: None 10: Overlap 11: Single_axis (0)	*bExecute* turns to TRUE.

***Note**:

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

CircMode setting	Definition CircMode–AuxPoint	CircPlane		
		XY_Plane	YZ_Plane	ZX_Plane
		Actual input for AuxPoint [_, _, _]		
2	Radius–Radius (R)	[R, N/A, N/A]		

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the absolute positioning is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bActive	TRUE when the instruction is controlling axes	BOOL	TRUE/FALSE (FALSE)
bCommand Aborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

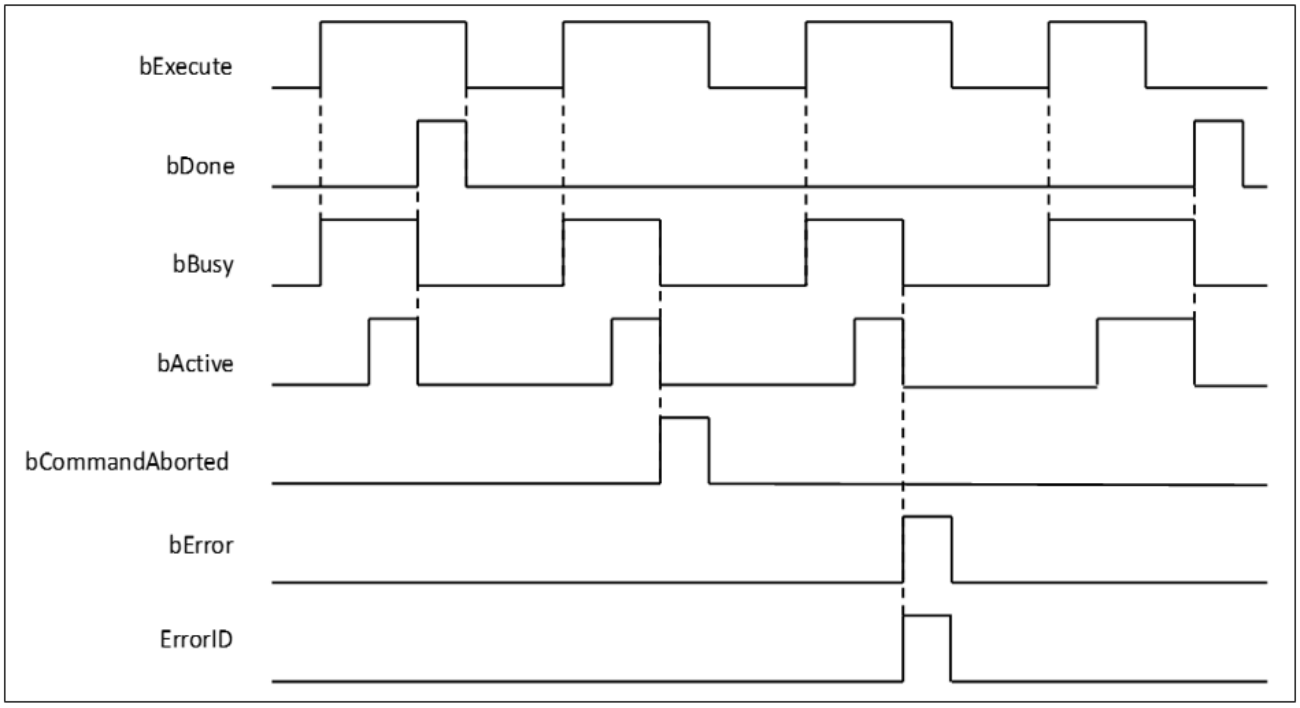
***Note**: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	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	*bExecute* changes to TRUE.	• When *bDone* turns to TRUE • When *bError* turns to TRUE • When *bCommandAborted* turns to TRUE
bActive	Axes start being controlled by the instruction.	• When *bDone* turns to TRUE • When *bError* turns to TRUE • When *bCommandAborted* turns to TRUE

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
		• *bActive* will change to FALSE after remaining TRUE for at least one cycle when *bExecute* is FALSE but bActive changes to TRUE.
bCommandAborted	• The instruction is interrupted by another instruction BufferMode of which is set to Aborting. • The instruction is interrupted by MC_Stop. • The instruction is interrupted by DMC_GroupStop.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID*.)	*bExecute* turns from TRUE to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



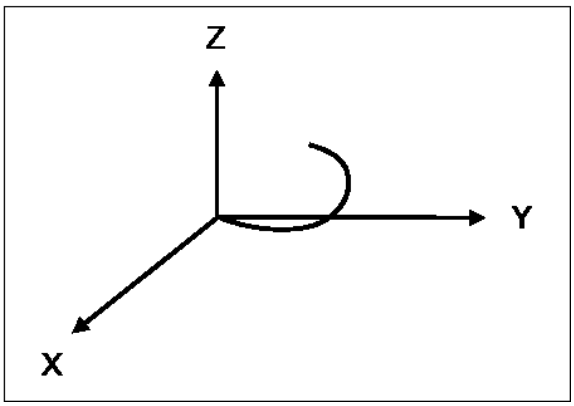
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*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.

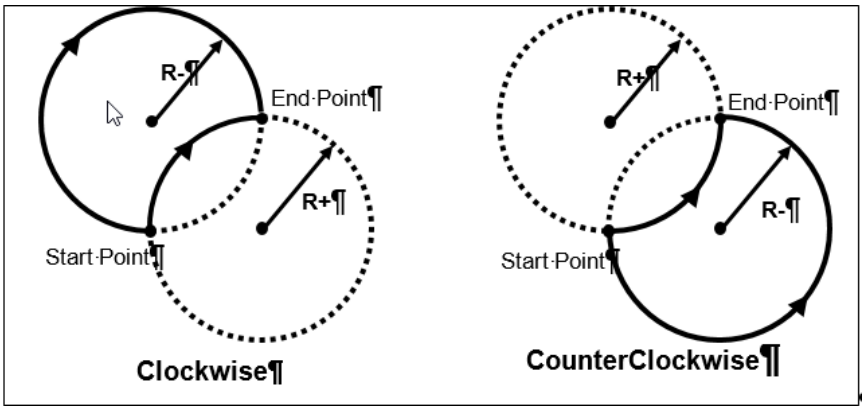
- **Function**
  - The instruction supports up to three axes to move to the absolute possition by the helical interpolation. The three axes will start, stop, and reach the absolute target position at the same time.
  - The instruction can be used to draw the circle on the plane parallel to XY / YZ / ZX and set the height of the helix on Z / X / Y axis.
  - If you want to set the start point and end point 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.
  - CoordSystem is supported by DL_MotionControl library V1.1.0.0 or later.
  - CircPlane added ARBITRARY_plane in DL_MotionControl library V1.2.0.0 or later.
  - ARBITRARY_plane can draw arc in space.
  - Use the following parameters to draw an arc.

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



  - When *CircMode* is set to Radius, the entered positive and negative values of *AuxPoint* will affect the size of the circle.

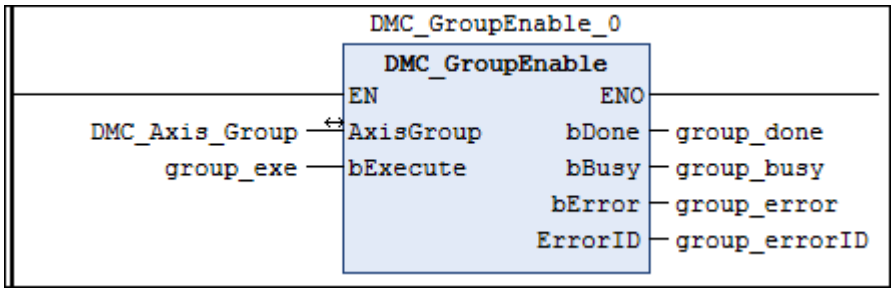
- The size of the circle will be determined by the positive or negative value entered by the Radius.

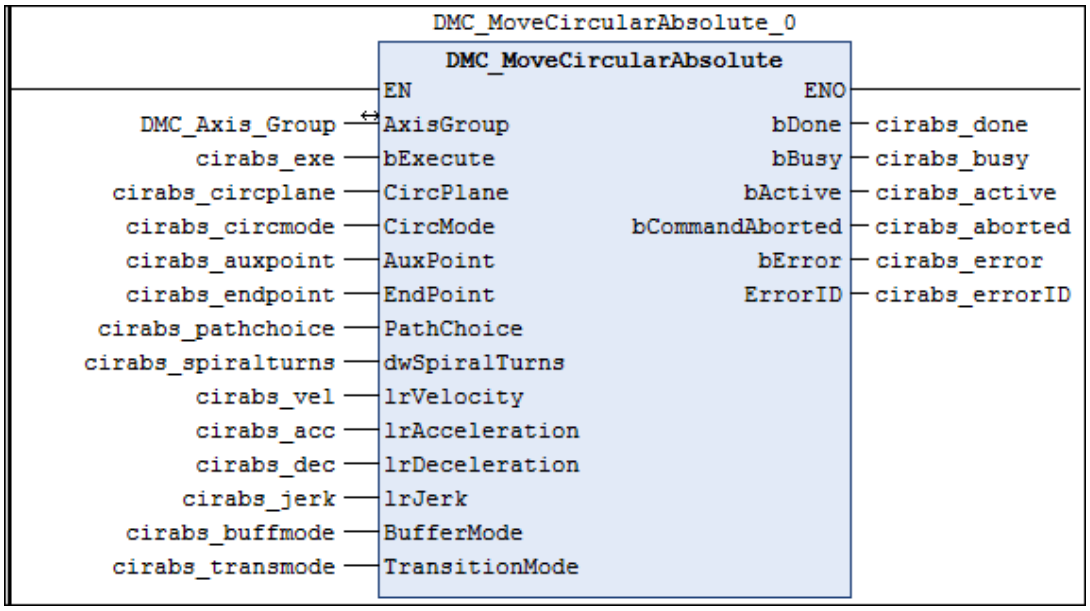


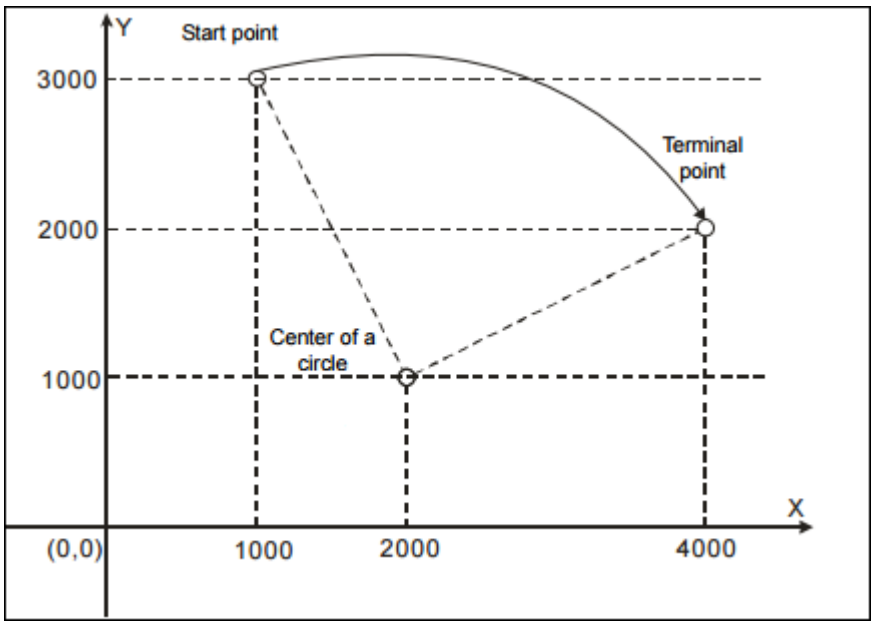
Note: Library DL_MotionControl version 1.4.1.0 and later supports this feature.

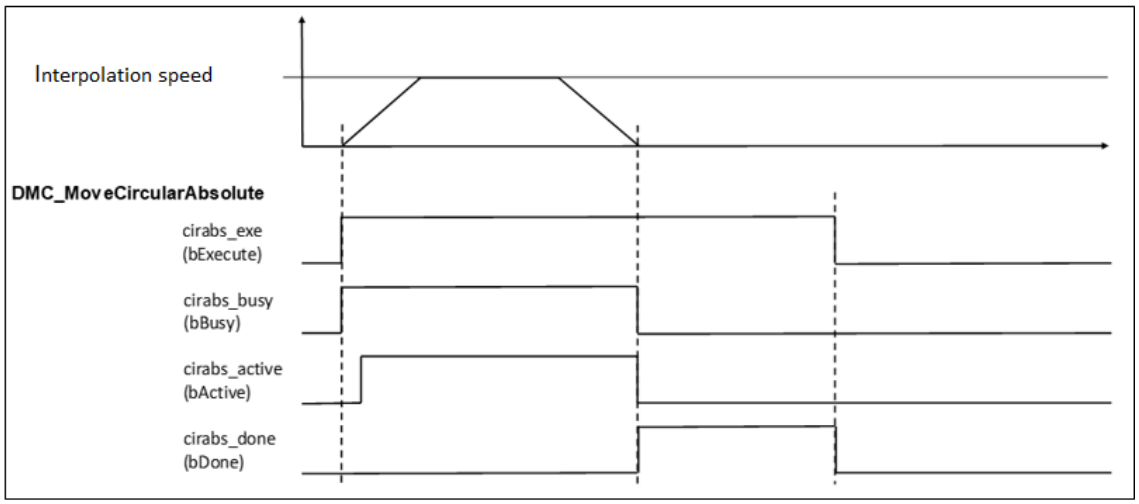
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding troubleshootings, refer to **Appendix** for error code descriptions.
- **Example**
  - In this example, the instruction performs the circular interpolation from current position (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;
```









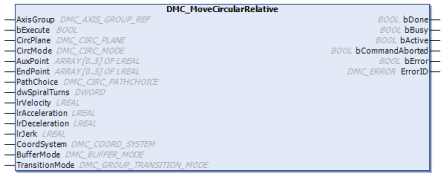
- When cirabs_exe (*bExecute*) changes to TRUE, DMC_MoveCircularAbsolute performs the absolute positioning toward the terminal point (4000, 2000) from the start point (1000, 3000) in the clockwise direction.

- 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.

- **Supported Devices**
  - AX-series motion controller

2.2.1.6 DMC_MoveCircularRelative

DMC_MoveCircularRelative controls the axis group to move to a relative position by circular or helical interpolation.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_MoveCircular Relative		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
CircPlane	The circular or helical plane[*1]	DMC_CIRC_PLANE	0: XY_plane 1: YZ_plane 2: ZX_plane 3: ARBITRARY_plane (0)	*bExecute* turns to TRUE.
CircMode	The method for circular or helical interpolation[*1]	DMC_CIRC_MODE	0: Border 1: Center 2: Radius (0)	*bExecute* turns to TRUE.
AuxPoint	The auxiliary point data[*1]	LREAL[6]	[_, _, _, _, _, _] Positive, negative, or 0 ([0, 0, 0, 0, 0, 0])	*bExecute* turns to TRUE.
EndPoint	The target position for each axis in the axis group (Unit: user unit)	LREAL[6]	[_, _, _, _, _, _] Positive, negative, or 0	*bExecute* turns to TRUE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
			([0, 0, 0, 0, 0, 0])	
PathChoice	The circular or helical interpolation direction	DMC_CIRC_PATHCHOICE	0: Clockwise 1: CounterClockwise (0)	*bExecute* turns to TRUE.
dwSpiralTurns	The number of spiral turns	DWORD	0–65535 (0)	*bExecute* turns to TRUE.
lrVelocity	The target velocity for the axis group (Unit: user unit/s)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrAcceleration	The acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrDeceleration	The deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrJerk	The jerk (Unit: user unit/s^3)	LREAL	Positive (0)	*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	The buffer mode for the instruction*2	DMC_BUFFER_MODE	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	*bExecute* turns to TRUE.
TransitionMode	The transition mode for the instruction*3	DMC_GROUP_TRANSITION_MODE	0: None 10: Overlap 11: Single_axis (0)	*bExecute* turns to TRUE.

***Note**:

1. Settings of 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]

CircMode setting	Definition CircMode–AuxPoint	CircPlane		
		XY_Plane	YZ_Plane	ZX_Plane
		Actual input for AuxPoint [_, _, _]		
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 (Default Value)
bDone	TRUE when the relative positioning is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bActive	TRUE when the instruction is controlling axes	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

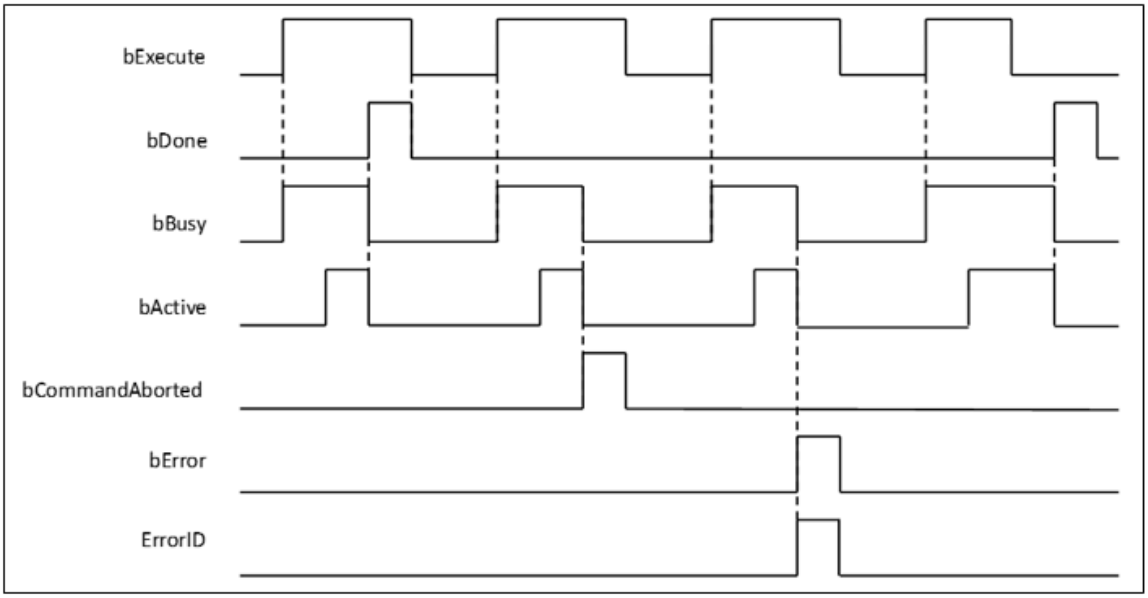
***Note**: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The relative positioning is completed.	• *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	*bExecute* changes to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bActive	Axes start being controlled by the instruction.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
		• *bActive* will change to FALSE after remaining TRUE for at least one cycle when *bExecute* is FALSE but *bActive* changes to TRUE.
bCommand Aborted	• The instruction is interrupted by another instruction BufferMode of which is set to Aborting. • The instruction is interrupted by MC_Stop. • The instruction is interrupted by DMC_GroupStop.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	*bExecute* turns from TRUE to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Change**



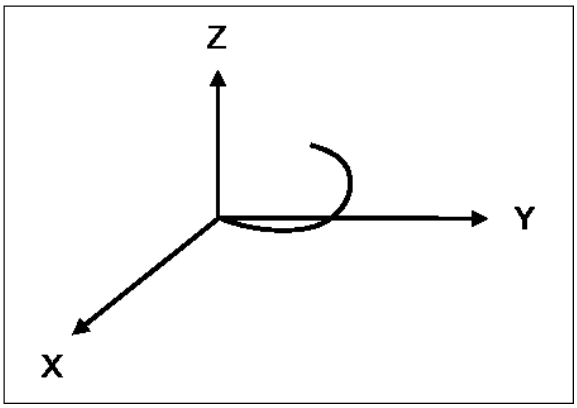
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*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.

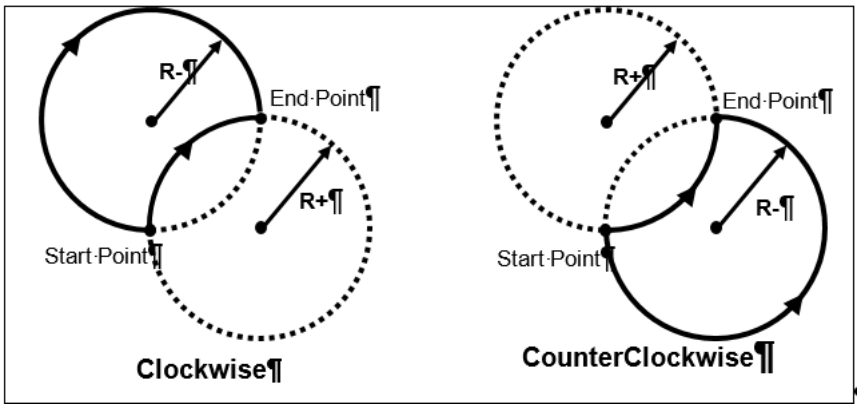
- **Function**
  - The instruction supports the relative helical interpolation of maximum three axes, where the three axes can simultaneously start, stop as well as reach the specified relative target position.
  - The instruction can be used to specify the circle drawing on the plane parallel to XY / 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 arc in space.
    - Use the following parameters to draw the arc.

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



  - When *CircMode* is set to Radius, the entered positive and negative values of *AuxPoint* will affect the size of the circle.

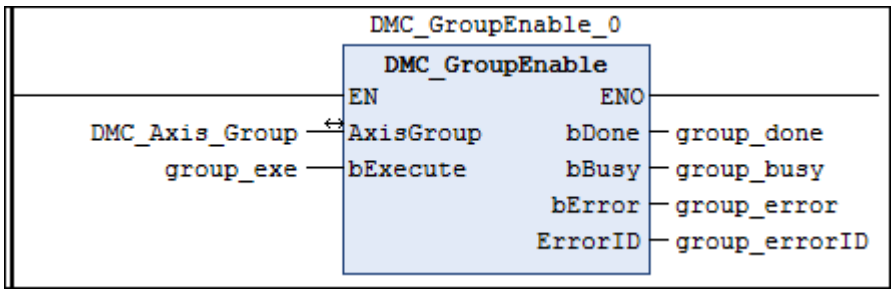
- The size of the circle will be determined by the positive or negative value entered by the Radius.

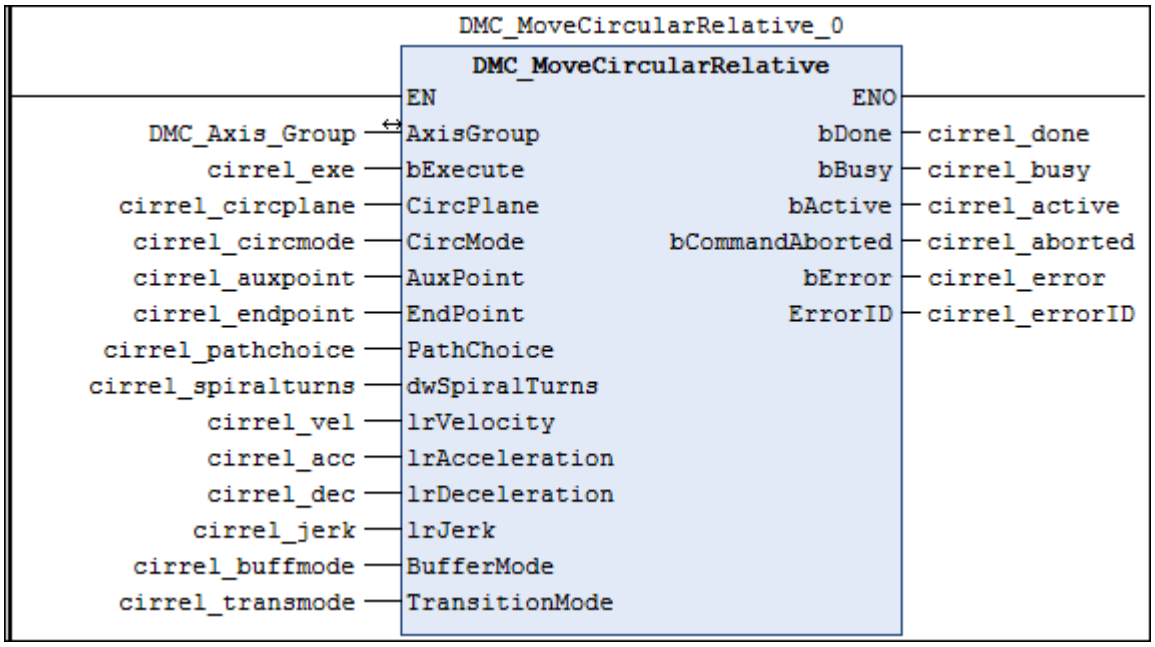


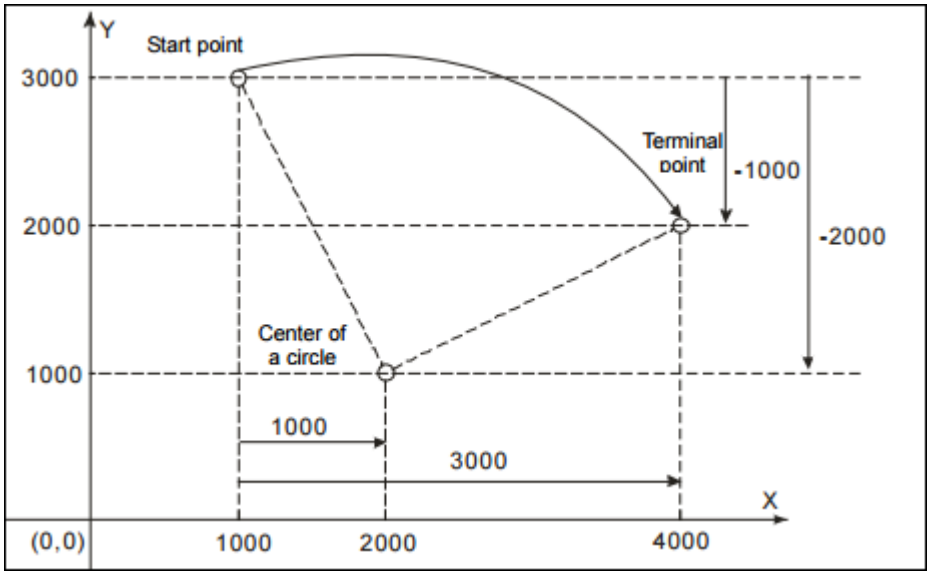
Note: Library DL_MotionControl version 1.4.1.0 and later supports this feature.

- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding trouble shootings, refer to **Appendix** for error code descriptions.
- **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;
```







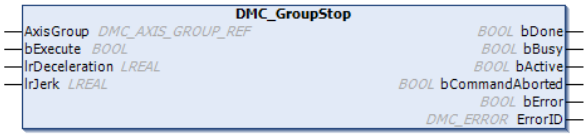
- When cirrel_exe (*bExecute*) changes to TRUE, DMC_MoveCircularRelative performs the relative positioning toward the terminal point (4000, 2000) from the start point (1000, 3000) in the clockwise direction.
- When cirrel_done (*bDone*) is TRUE and cirrel_busy (*bBusy*) changes to FALSE, which means 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.

• Supported Devices

- AX-series motion controller

2.2.1.7 DMC_GroupStop

DMC_GroupStop decelerates the axis group to stop at the current position.

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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrDeceleration	The deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrJerk	The jerk (Unit: user unit/s^3)	LREAL	Positive (0)	*bExecute* turns to TRUE.

- **Outputs**

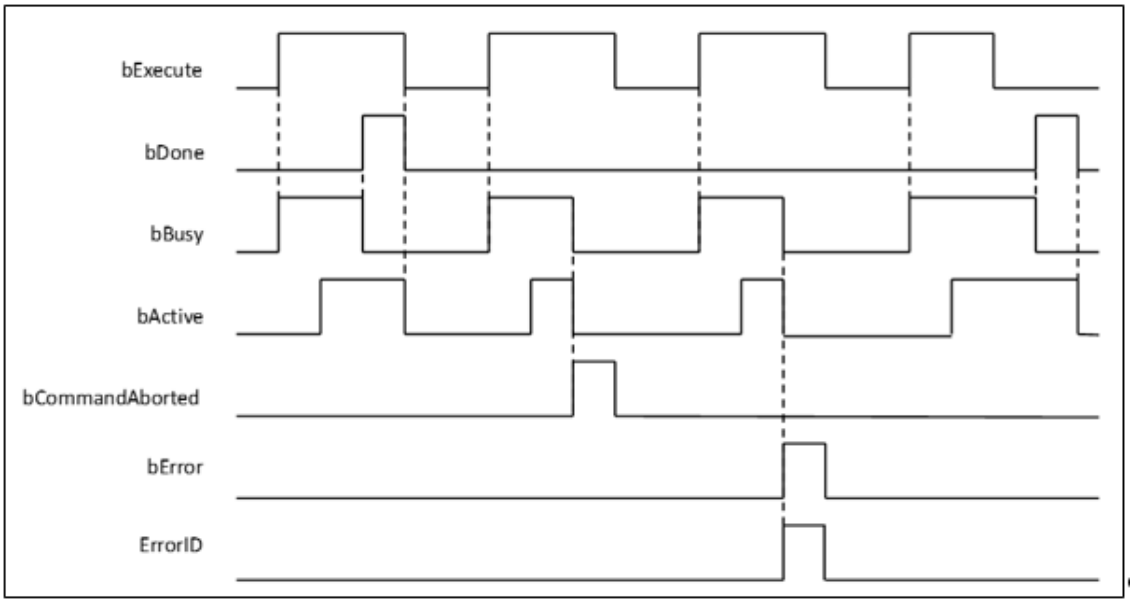
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when all axes stop with the velocity 0	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bActive	TRUE when the instruction is controlling axes	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The axis group decelerates to a stop.	• *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	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bActive	Axes start being controlled by the instruction.	• *bExecute* turns to FALSE. • *bError* turns to TRUE. • *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	The instruction is interrupted by another instruction	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	*bExecute* turns from TRUE to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **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	*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.

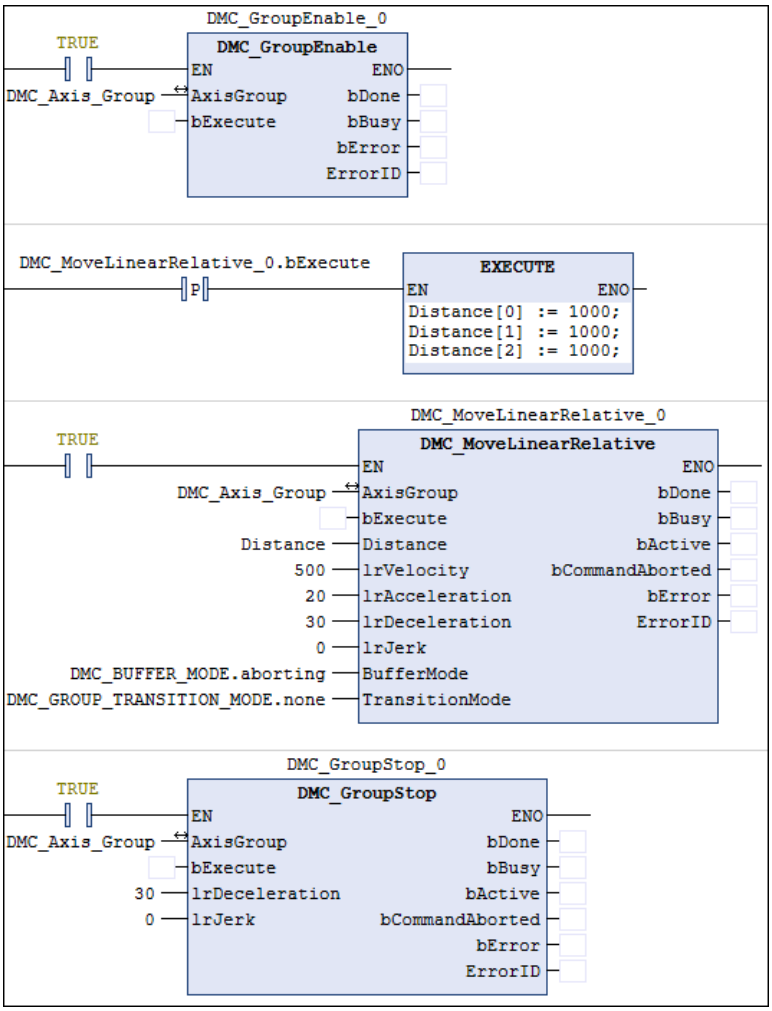
- **Function**
  - The instruction stops the moving axis group.
  - The axis group state is switched to GroupStopping.
  - The axis group state GroupStopping will continue until *bExecute* changes to FALSE. When the velocity reaches 0, *bDone* changes to TRUE

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

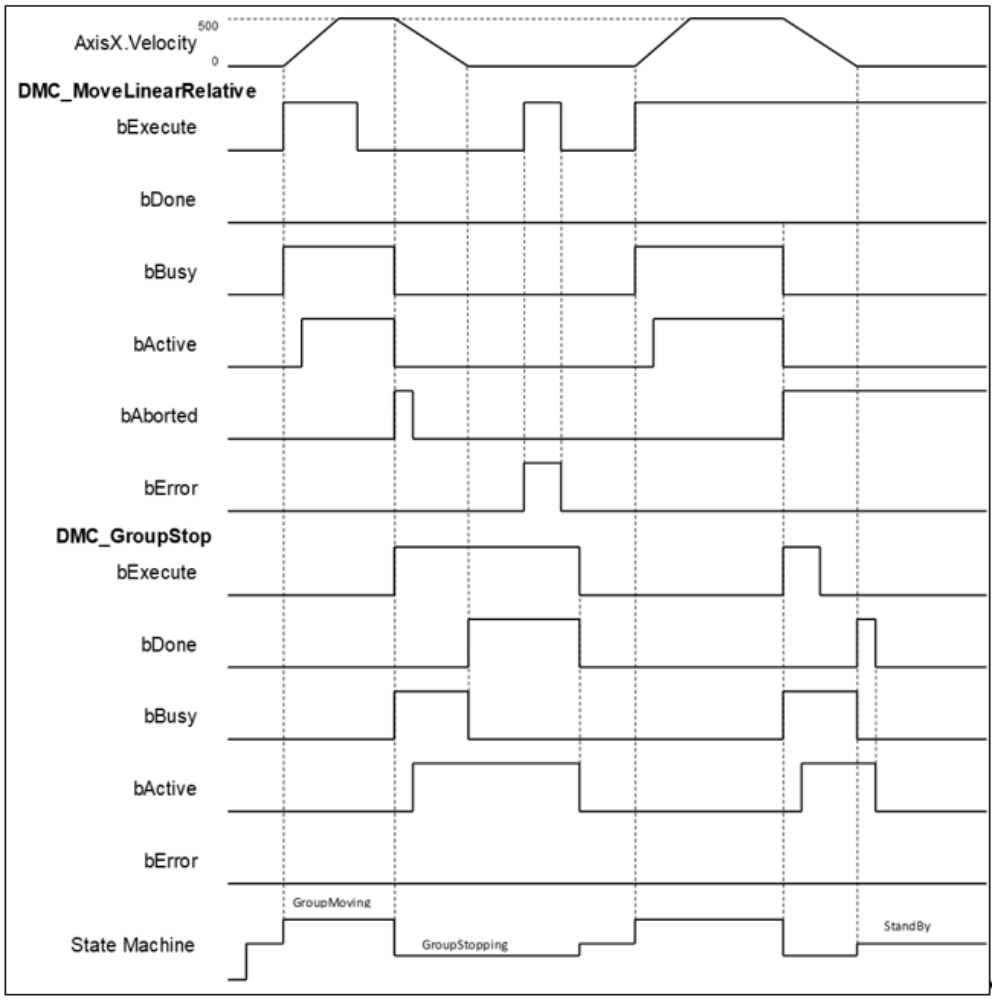
- **Example**

  This example shows the motion behavior performed by DMC_GroupStop when running DMC_MoveLinearRelative.

- When DMC_GroupStop is completed, the axis group changes to GroupStandby state.



- Timing Diagram

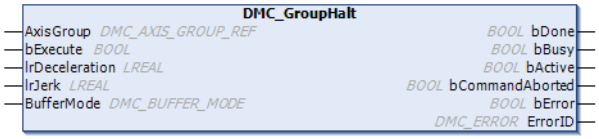


- When *bExecute* of DMC_GroupStop changes to TRUE, *bCommandAboted* of MoveLinearRelative changes to TRUE, and axes start to decelerate to a stop, meanwhile, the axis group stays in GroupStopping state.
- When the velocity of axes reaches 0, *bDone* of DMC_GroupStop changes to TRUE, and the axis group holds GroupStopping state.
- When *bExecute* of DMC_GroupStop changes to FALSE, the state of axes changes from GroupStopping to StandBy.

- **Supported Devices**
  - AX-series motion controller

2.2.1.8 DMC_GroupHalt

DMC_GroupHalt decelerates the moving axis group to pause at the current position.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupHalt	,k 	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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrDeceleration	The deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrJerk	The jerk (Unit: user unit/s^3)	LREAL	Positive (0)	*bExecute* turns to TRUE.
BufferMode	Buffer mode for the instruction*	DMC_BUFFER_MODE	0: Aborting 1: Buffered (0)	*bExecute* turns to TRUE.

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

- **Outputs**

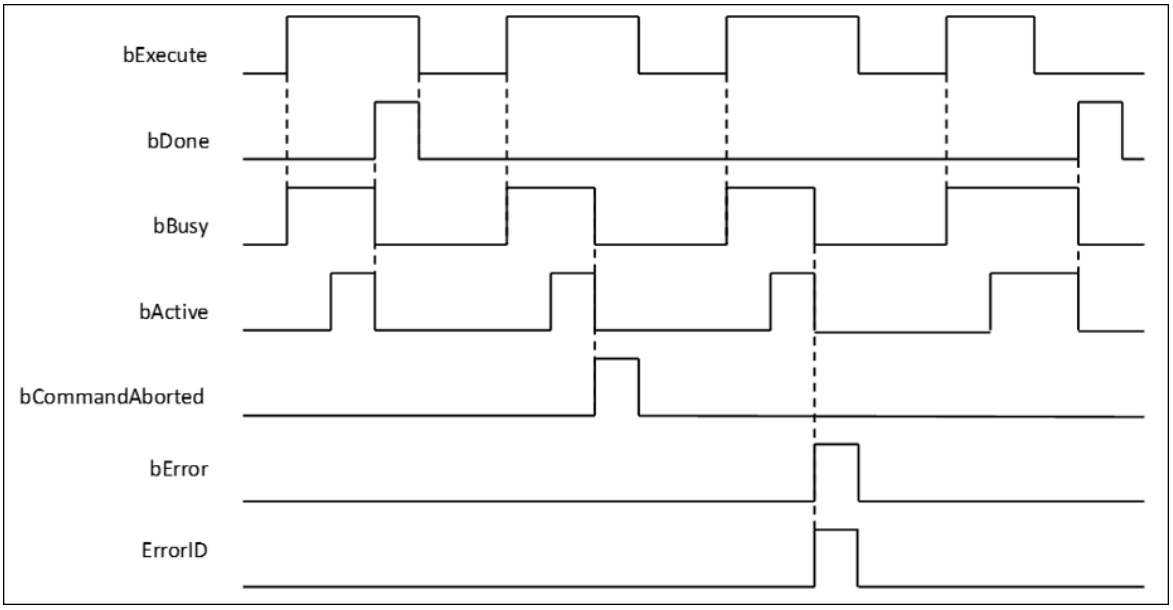
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when all axes stop with the velocity 0	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bActive	TRUE when the instruction is controlling axes	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

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

- **Output Update Timing**

Name	Function	Data Type
bDone	The axis group decelerates to a stop.	• *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	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bActive	Axes start being controlled by the instruction.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *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	The instruction is interrupted by another instruction.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	*bExecute* turns from TRUE to FALSE. (Error code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

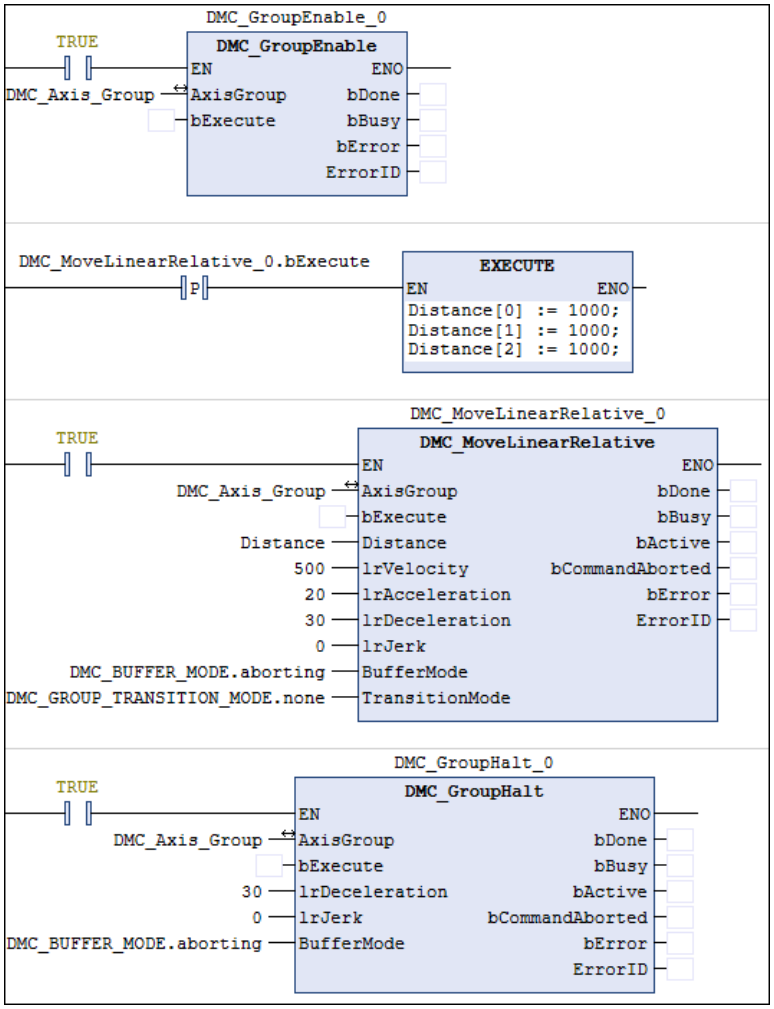
Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	Specify the axis group.	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*bExecute* turns to TRUE and *bBusy* is FALSE.

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

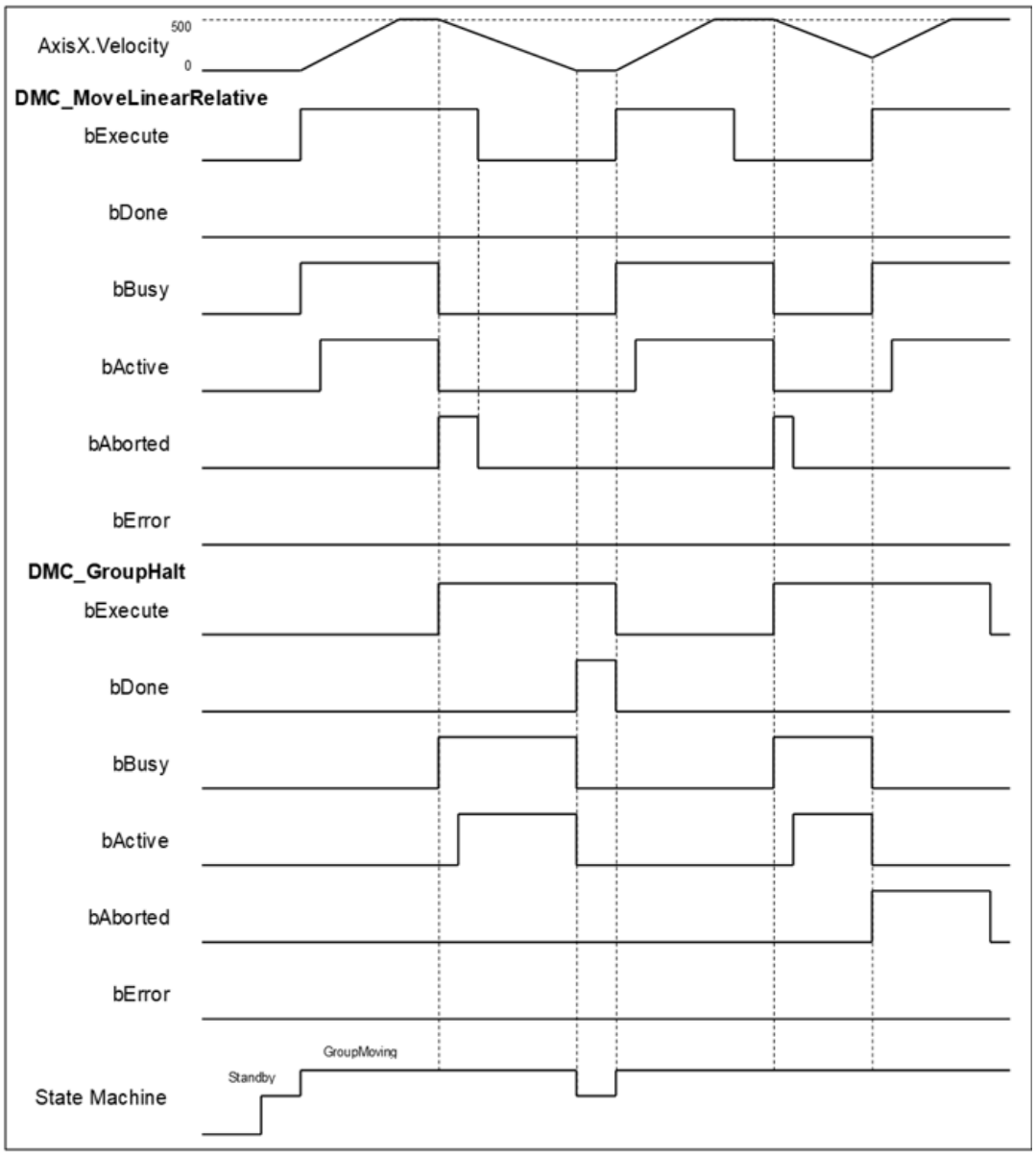
- **Function**
  - The instruction decelerates the moving axis group to pause.
  - The axis group state keeps at GroupMoving.
  - When the velocity 0 is reached, *bDone* changes to TRUE, and the axis group changes to *StandBy* state.
  - BufferMode of DMC_GroupHalt only supports *0: Aborting* and *1: Buffered*. An error will occur with DMC_ERROR.DMC_GM_INVALID_BUFFER_MODE if another BufferMode is used.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  This example shows how the DMC_GroupHalt runs after running DMC_MoveLinearRelative.
  - If no other motion instructions are executed during DMC_GroupHalt pauses DMC_MoveLinearRelative, the axes state will enter “Standby” after the deceleration is completed.

- When DMC_MoveLinearRelative is run again during deceleration, DMC_GroupHalt will be interrupted, and the axis group will accelerate again without staying in the Standby state any more. The re-running action described above is allowed for DMC_GroupHalt.



- Timing Diagram

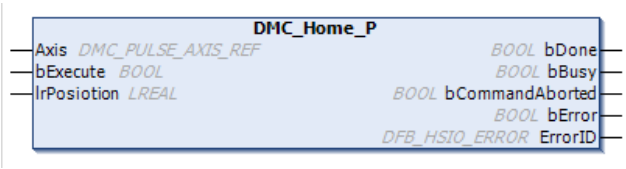


- When *bExecute* of DMC_GroupHalt changes to TRUE, *bCommandAboted* of DMC_MoveLinearRelative changes to TRUE and the axes start to decelerate to a stop. The axis group stays in GroupMoving.
- When the velocity 0 is reached, *bDone* of DMC_GroupHalt changes to TRUE, and the axis group changes to Standby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

- **Supported Devices**
  - AX-series motion controller

2.2.1.9 DMC_Home_P

DMC_Home_P drives the pulse axis to perform the homing to 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 Values (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrPosition	The position after the homing is completed	LREAL	Positive, negative, or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when homing is complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DFB_HSIO_ERROR*	DFB_HSIO_ERROR (DFB_HSIO_NO_ERR)

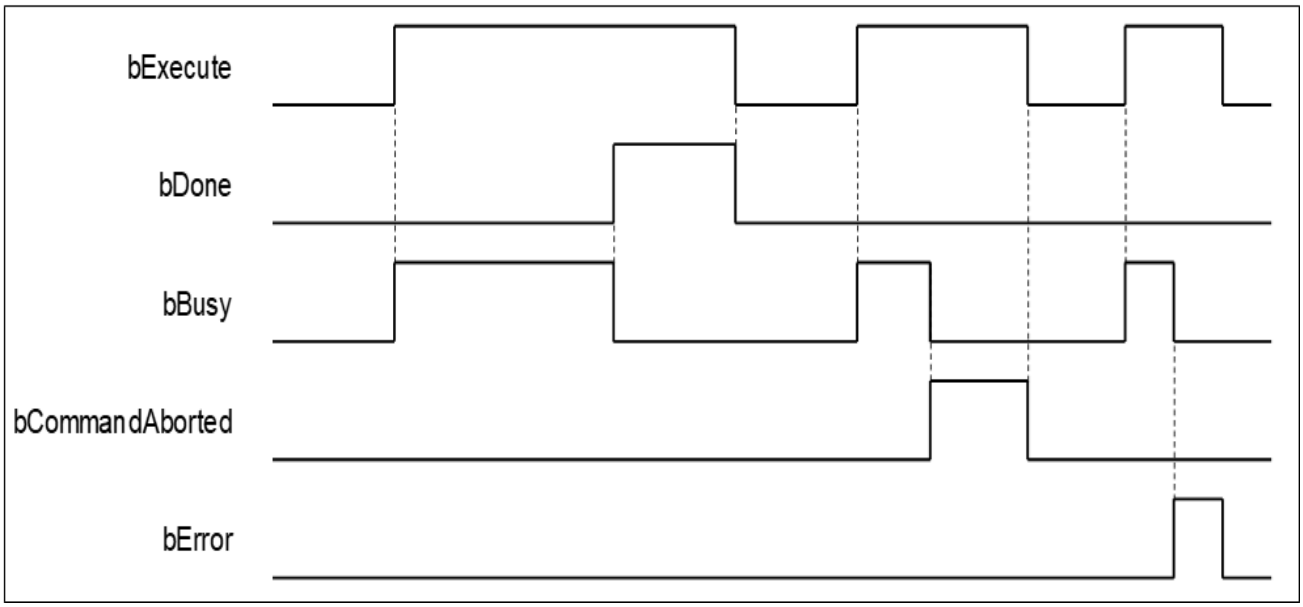
***Note**: DFB_HSIO_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Function	Data Type
bDone	Homing is complete	• *bExecute* turns to FALSE. • *bError* turns to TRUE.

Name	Function	Data Type
bBusy	*bExecute* changes to TRUE.	• *bExecute* turns to FALSE. • *bError* turns to TRUE.
bCommandAborted	The instruction is interrupted by MC_Stop.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	*bExecute* turns from TRUE to FALSE. (Error code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **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	*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 must be selected in **Hardware IO Configuration** to output pulses and perform the homing action according to the pulse axis settings, such as homing mode, acceleration rate, and velocity.
  - The instruction can be used only when the pulse output axis is in the Standstill state. An error will occur if the instruction is run in other axis states.

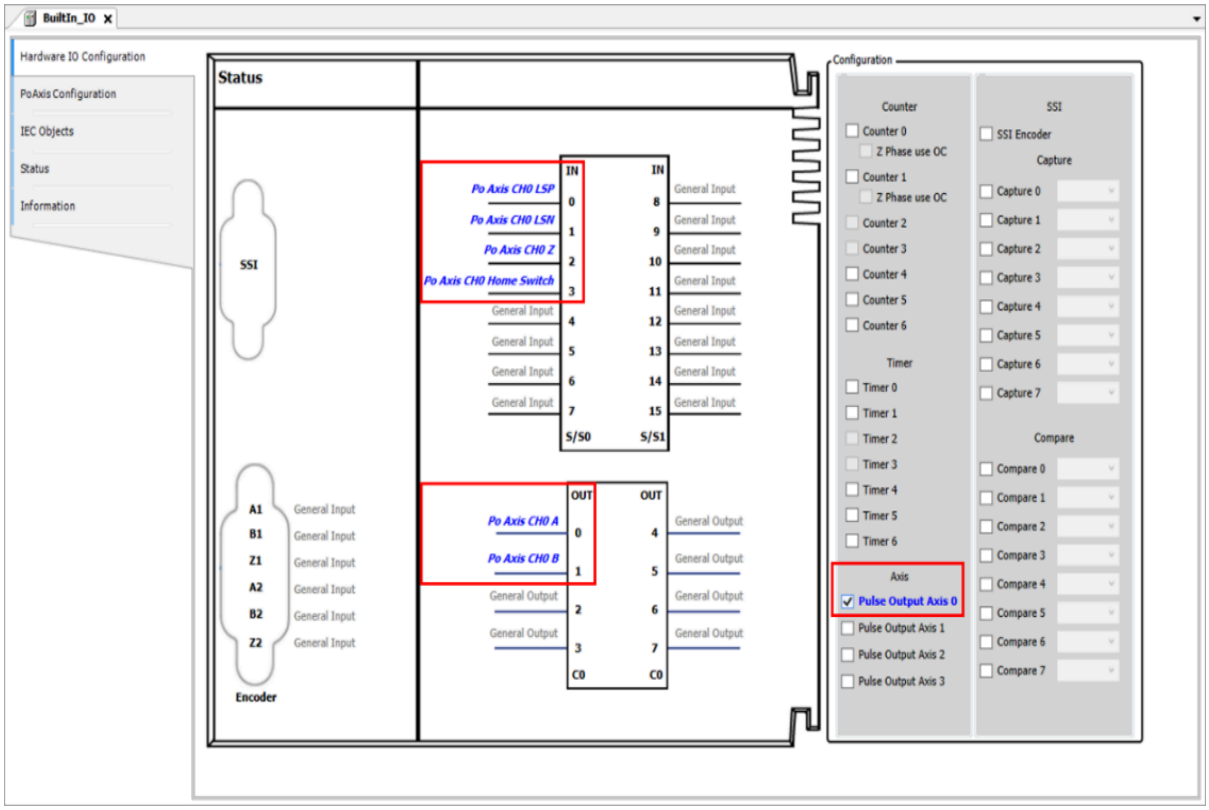
- DMC_Home_P supports homing modes defined in the CiA 402 protocol. For details on homing modes, refer to A.4 Explanation of DMC_Home_P.
- Library of this function block: DL_BuiltInIO_AX3.library.
- This function block only supports trapezoid curves.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

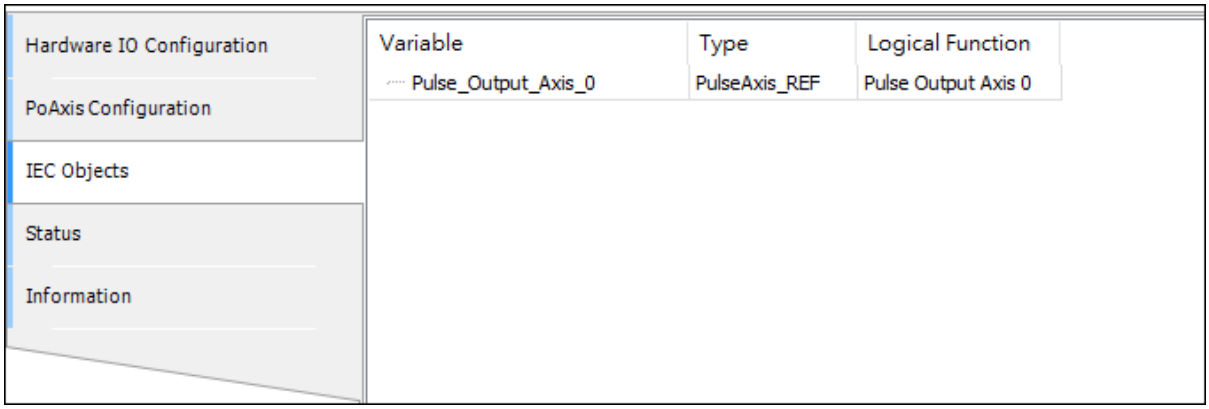
- **Example**

  In this example, the pulse output axis performs the homing motion via DMC_Home_P after configuring the axis in the **Hardware IO Configuration** tab.

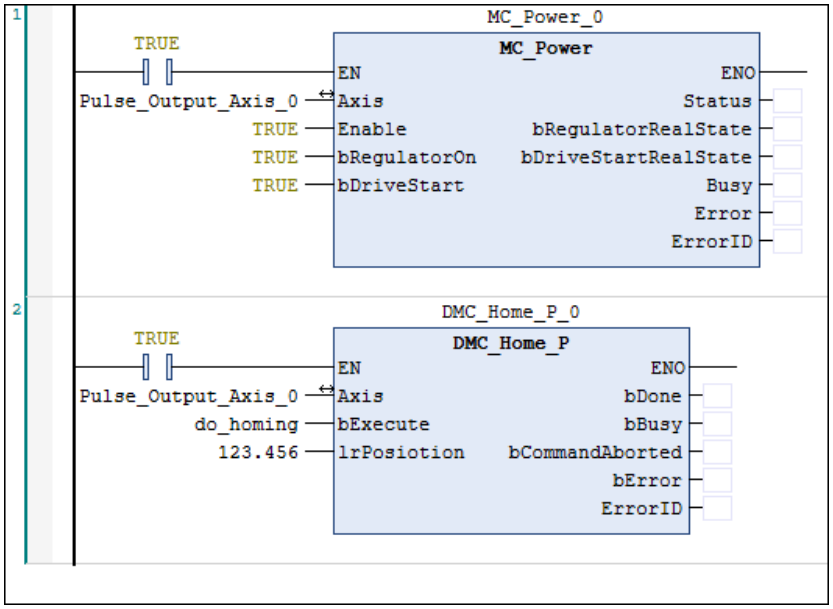
  - Select the first pulse output axis (**Pulse Output Axis 0**) in **Hardware IO Configuration** of **BuiltIn_IO**, and then output points (e.g., **OUT0**, **OUT1**) and signal trigger points for pulse output (e.g., **IN0**, **IN1**, **IN2**, and **IN3**) will show. You must configure the signal trigger points that will be used by Homing Mode to the corresponding input signal in order to run correctly.



  - After the pulse output axis is configured, the variable **Pulse_Output_Axis_0** configured in **IEC Objects** can be used in the pulse axis parameter to any function block.



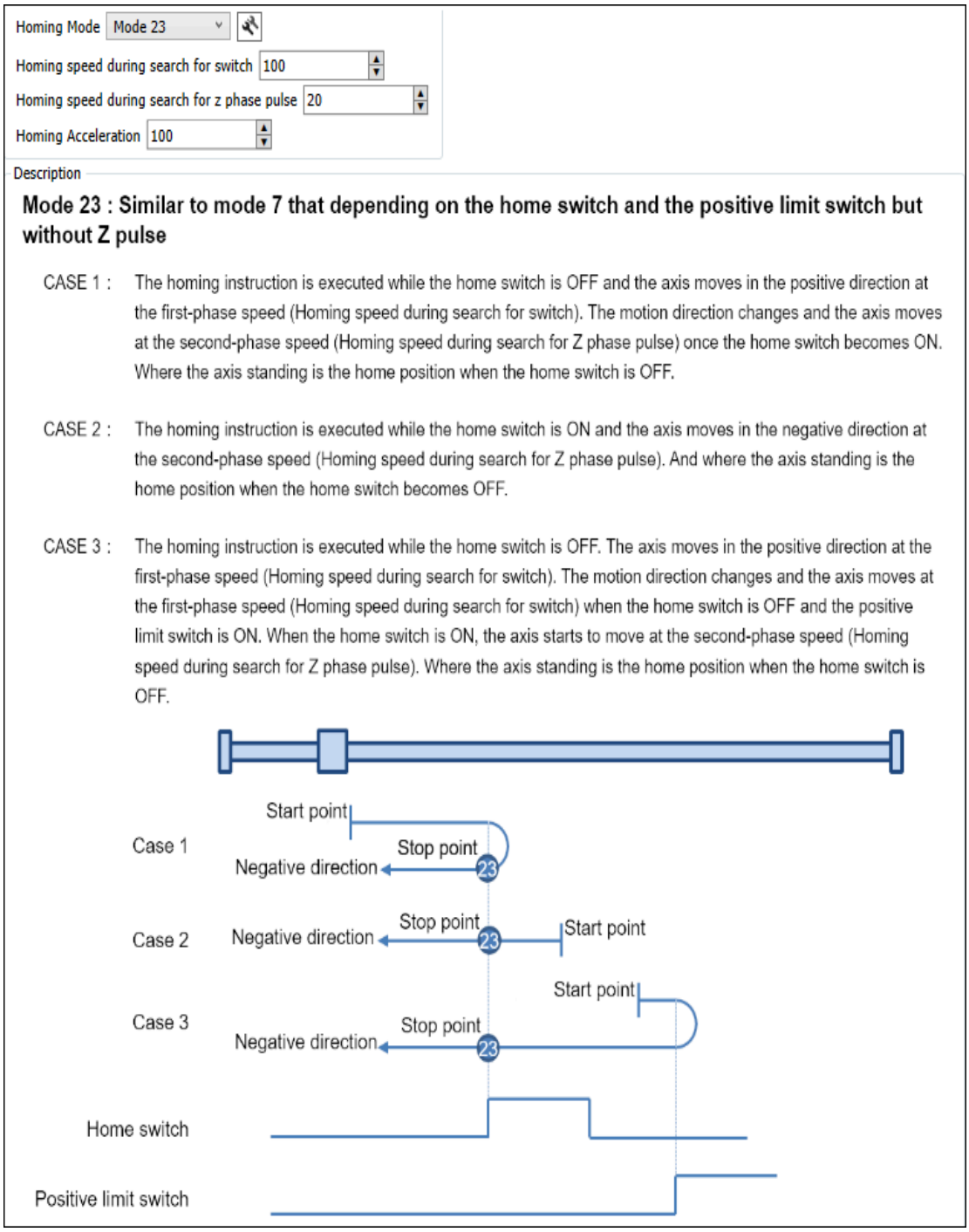
- **Pulse_Output_Axis_0** is connected to the input *Axis* of MC_ Power and DMC_ Home_P. When the axis is in the Standstill state, the instruction performs the homing motion according to the set homing mode. The state machine will switch the state from Standstill to Homing.



- After starting DMC_Home_P , **Pulse_Output_Axis_0** will perform the homing motion according to the mode selected in **Homing Mode** in the **PoAxis Configuration** tab. After the function block is run, the homing will be executed according to different external signals and conditions.

  - 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)

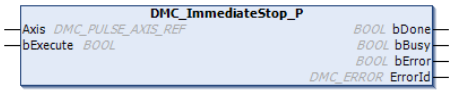
- Homing Acceleration: 2000 (Unit: user unit/s ^{2})



- **Supported Devices**
  - AX-series motion controller

2.2.1.10 DMC_ImmediateStop_P

DMC_ImmediateStop_P stops the PO axis motion immediately and stop the pulse output.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_ImmediateStop_P		DMC_ImmediateStop_P(Axis Axis : =, bExecute : =, bDone =>, bBusy =>, bError =>, ErrorId =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction 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 running is complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERR)

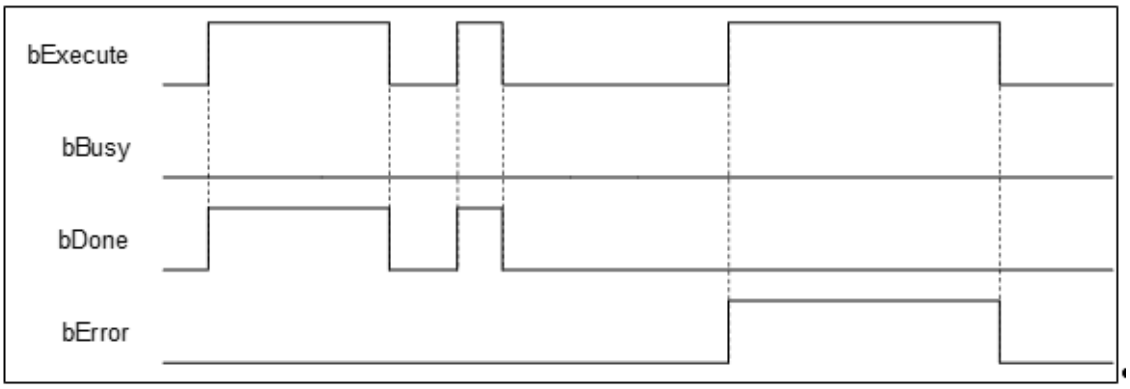
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The instruction running is complete.	• *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	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError		*bExecute* turns from TRUE to FALSE. (Error Code is cleared.)

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID*.)	

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

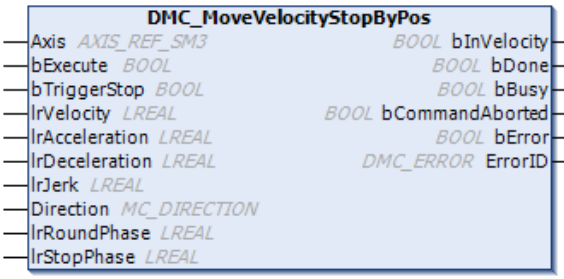
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	DMC_PULSE_AXIS_REF *1	DMC_PULSE_AXIS_REF	When *bExecute* turns to TRUE, and *bBusy* is FALSE

***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 while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding troubleshooting, refer to **Appendix** of this manual.
- **Supported Devices**
  - AX-series motion controller

2.2.1.11 DMC_MoveVelocityStopByPos

DMC_MoveVelocityStopByPos stops the moving axis at a specified position.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_MoveVelocityStopByPos		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	Runs the instruction 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)	*bExecute* is TRUE and the output *bBusy* is TRUE.
lrVelocity	The target velocity (Unit: User unit/s)	LREAL	Positive or 0 (0)	*bExecute* is TRUE and the output *bBusy* is FALSE.
lrAcceleration	The acceleration rate when the motion starts (Unit: User unit/s^2)	LREAL	Positive (0)	*bExecute* is TRUE and the output *bBusy* is FALSE.
lrDeceleration	The deceleration rate when the motion ends (Unit: User unit/s^2)	LREAL	Positive (0)	*bExecute* is TRUE and the output *bBusy* is FALSE.
lrJerk	The jerk (Unit: User unit/s^3)	LREAL	Positive (0)	*bExecute* is TRUE and the output *bBusy* is FALSE.
Direction	The motion direction	MC_DIRECTION*	-1: negative 1: positive (positive)	*bExecute* is TRUE and the output *bBusy* is FALSE.
lrRoundPhase	The modulo	LREAL	Positive (0)	*bExecute* is TRUE and the output *bBusy* is FALSE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrStopPhase	The position or a phase in the modulo	LREAL	Positive or 0 (0)	*bExecute* is TRUE and the output *bBusy* is FALSE.

*__Note__: MC_DIRECTION: Enumeration (Enum)

- **Outputs**

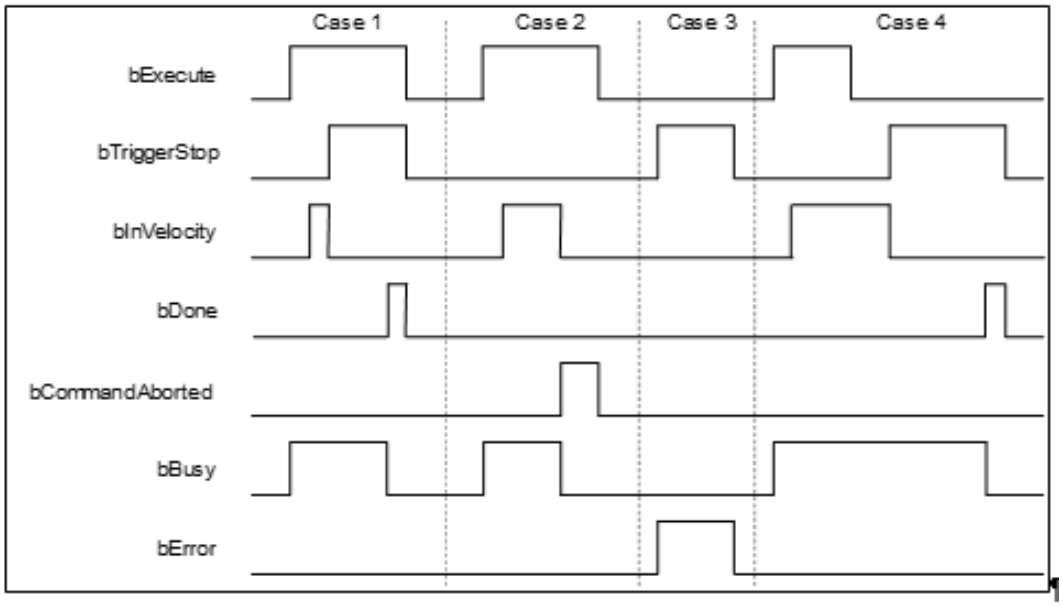
Name	Function	Data Type	Output Range (Default Value)
bInVelocity	TRUE when reaching the target velocity	BOOL	TRUE/FALSE (FALSE)
bDone	When the trigger signal is TRUE and the axis position has been recorded	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is interrupted by another instruction	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	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	The axis velocity reaches the target speed.	• *bCommandAborted* turns to TRUE. • When *bExecute* is re-triggered and *IrVelocity* is given a new value.
bDone	The trigger signal is TRUE, and the axis position has been recorded.	• *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	The instruction is aborted.	*bExecute* turns to FALSE.
bBusy	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns from TRUE to FALSE. (Error code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



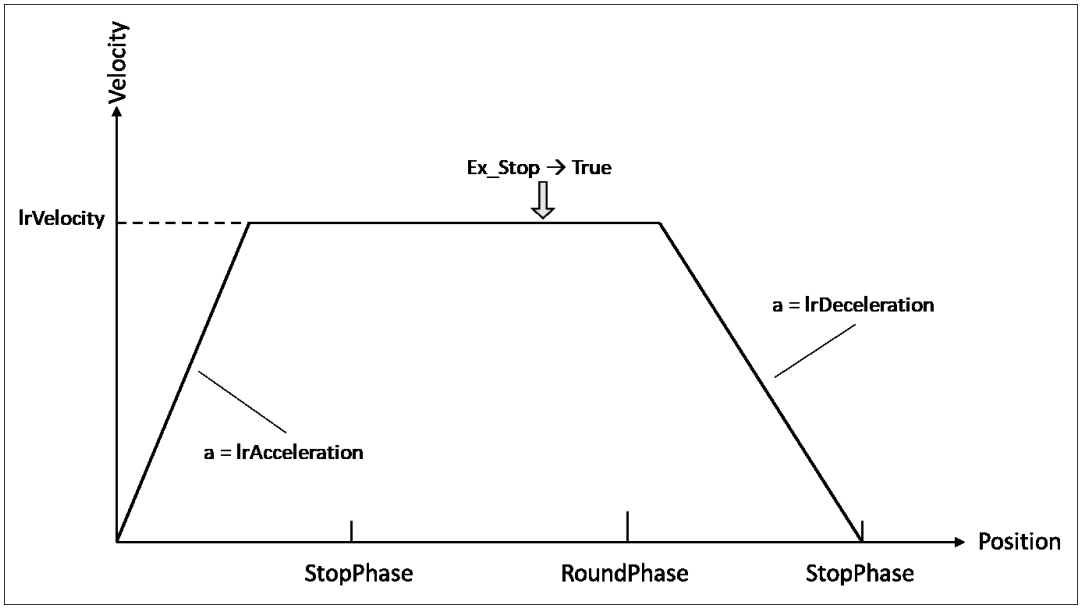
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*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.

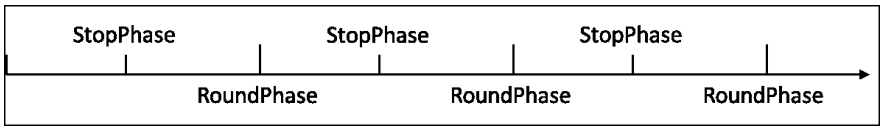
- **Function**

  - After *bExecute* of DMC_MoveVelocityStopByPos changes to TRUE, the axis will move with the velocity specified by *lrVelocity* and acceleration specified by *lrAcceleration*. When *bTriggerStop* changes to TRUE, the axis positioning will start according to *lrDeceleration* and the target position converted from the values of *RoundPhase* and *StopPhase*.

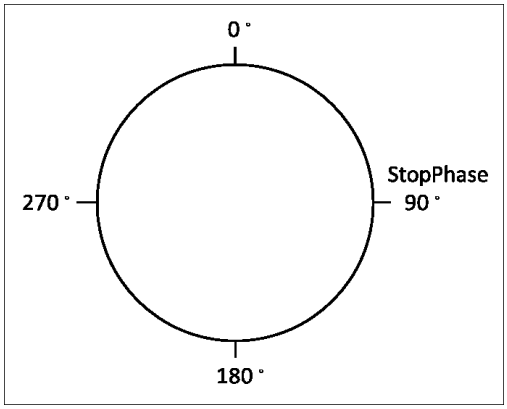


  - RoundPhase & StopPhase

- *RoundPhase* is the specified 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.

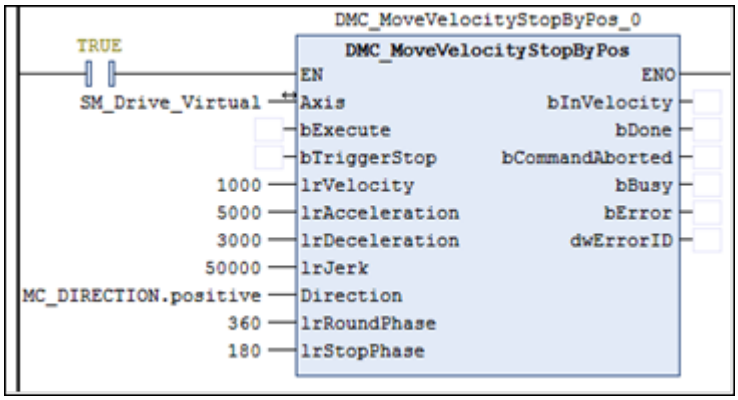


- When the axis specified by the function block is a rotary axis, *RoundPhase* specifies the entire phase of the modulo and *StopPhase* is a phase in the specified modulo. When *bTriggerStop* changes to TRUE, the axis will stop at the phase specified by *StopPhase*, and the final stop position is (*StopPhase* value / *RoundPhase* value) × **Modulo value** of the rotary axis.

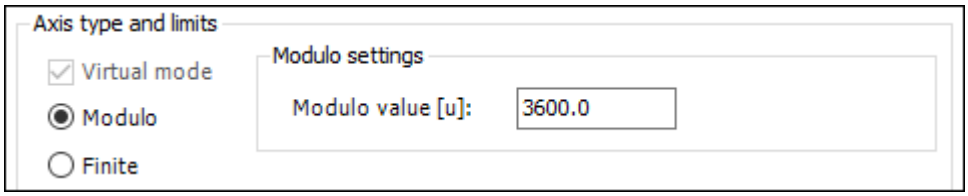


- Special case
  - If the specified axis position is too close to the target stop position when executing the stop command, the axis may not be able to complete the stopping action with the set deceleration *lrDeceleration*, and therefore the axis will be positioned in the next modulus. When this happens, it is recommended that the *lrDeceleration* or the trigger position of the stop command be adjusted to meet the planned path of deceleration.

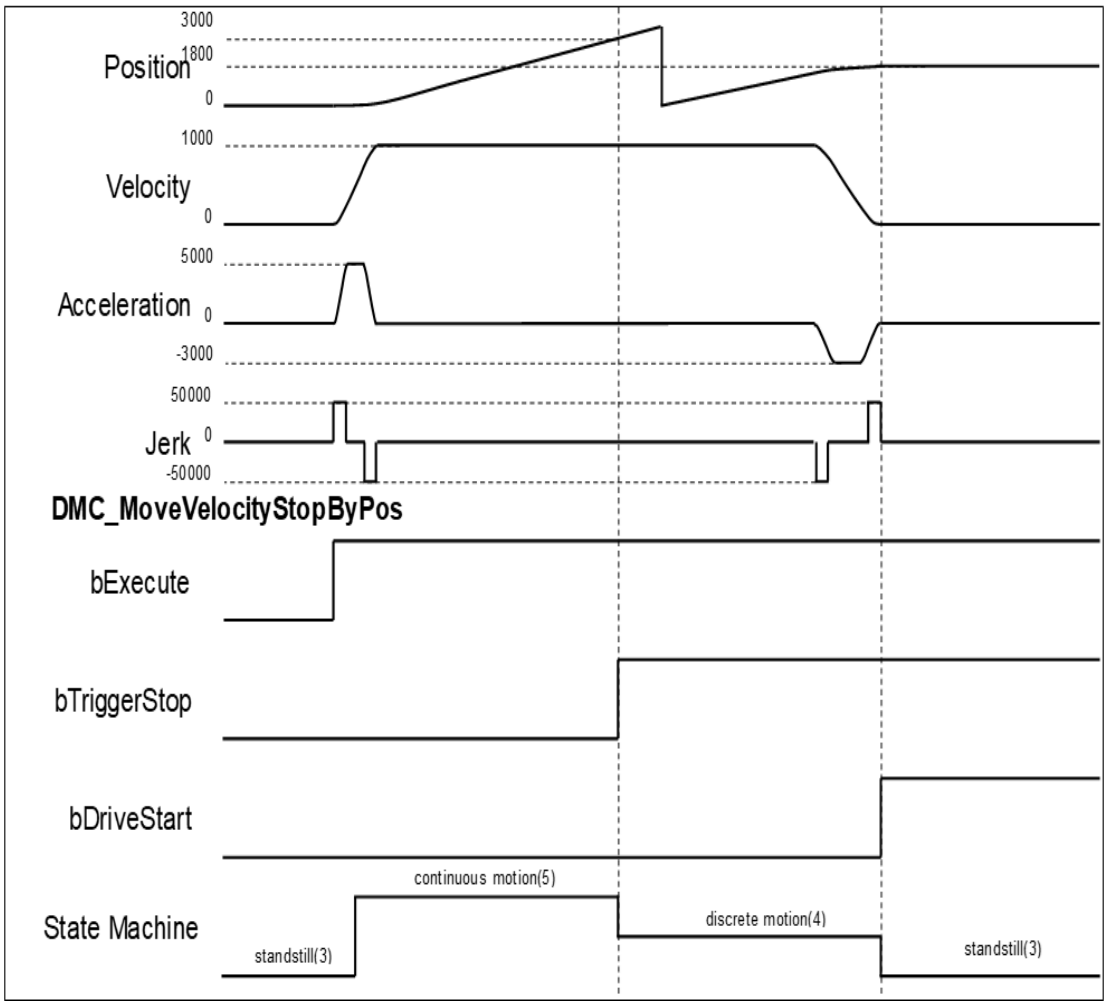
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - **Example 1**: This example illustrates how to use DMC_MoveVelocityStopByPos for phase positioning after the rotary axis motion starts.



- Rotary axis setting

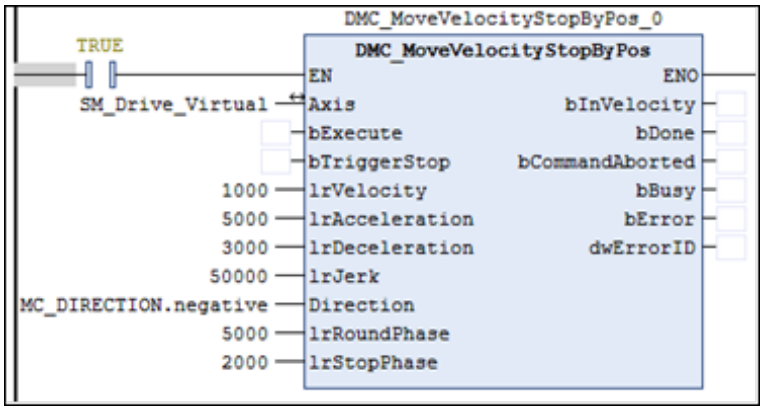


- Timing Diagram

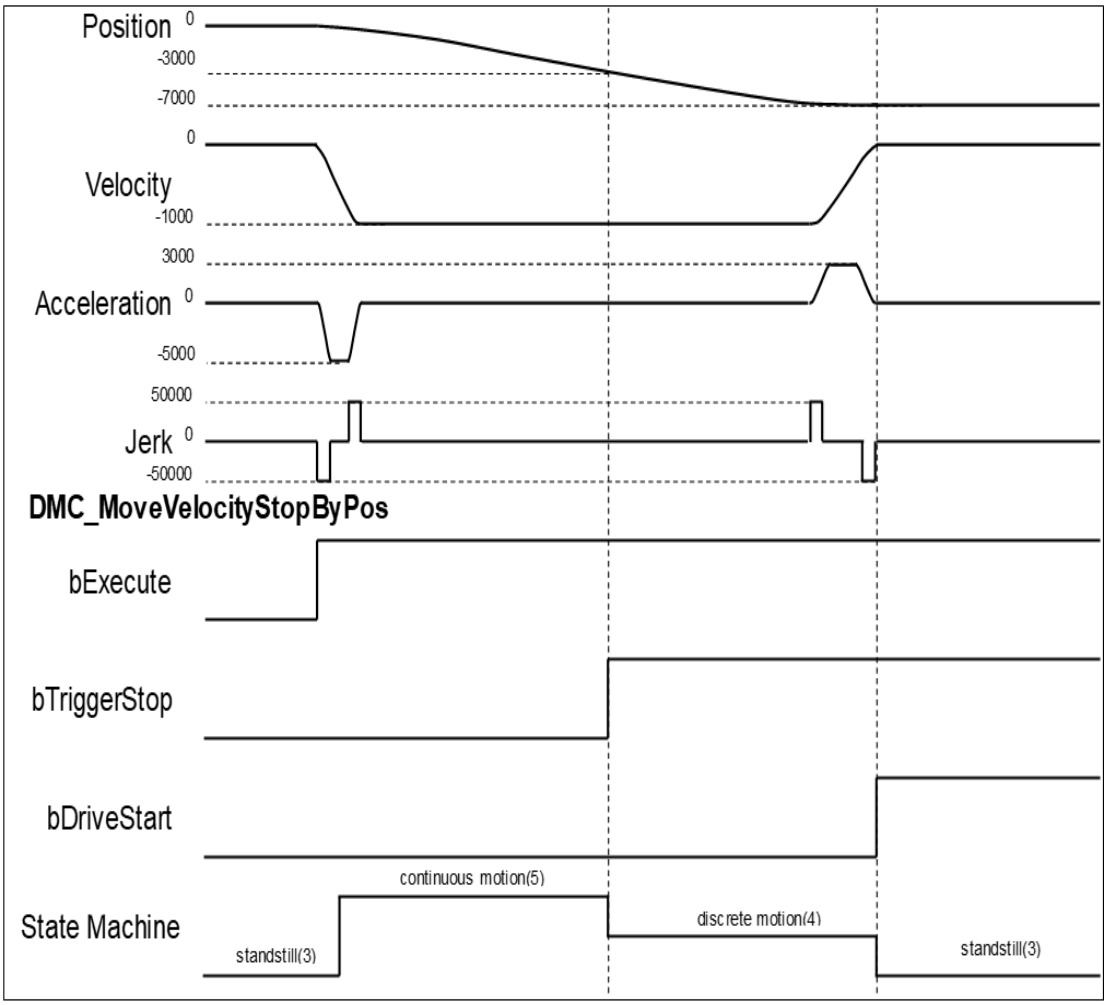


1. After *bExecute* changes to TRUE, the axis starts to move at a constant speed in the set direction until *bTriggerStop* changes to TRUE to start the positioning motion.
2. The *RoundPhase* and *StopPhase* of DMC_MoveVelocityStopByPos are set to 360 and 180, and the **Modulo** value of the rotary axis 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.

- **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.

- **Supported Devices**
  - AX-series motion controller

2.2.1.12 DMC_GroupInterrupt

DMC_GroupInterrupt pauses the current motion. Use DMC_GroupContinue to resume the paused motion.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupInterrupt		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrDeceleration	Deceleration (user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrJerk	Jerk; Jump (Unit: user unit/s^3)	LREAL	Positive (0)	*bExecute* turns to TRUE.

- **Outputs**

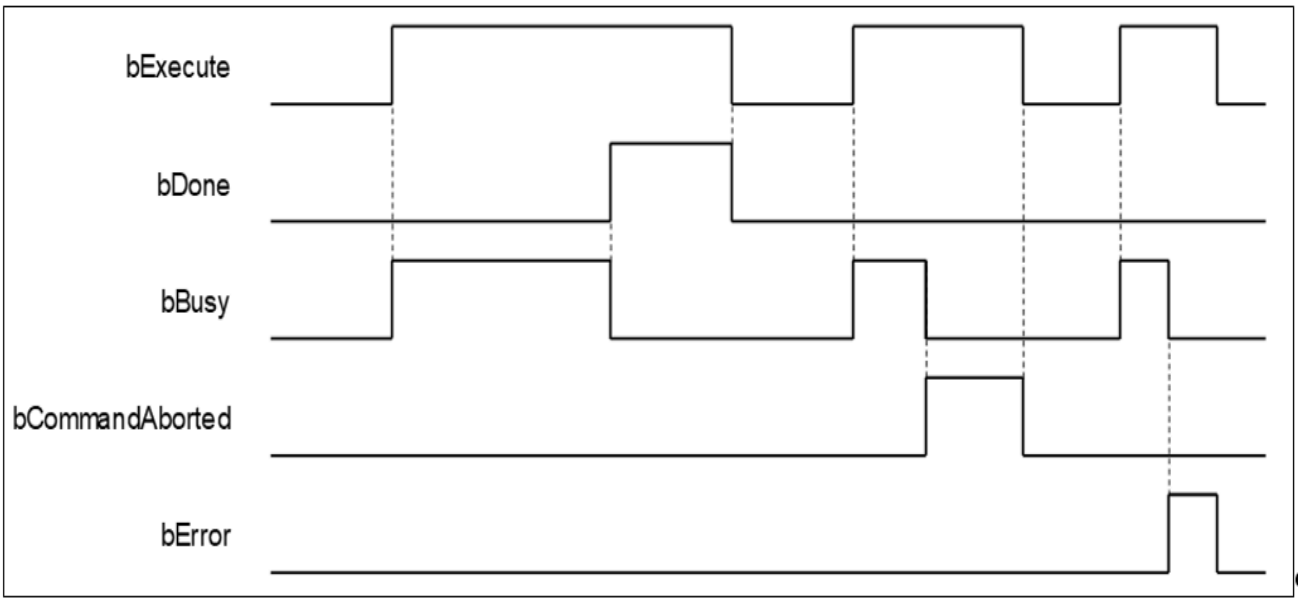
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when all axes stop with the velocity 0	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is interrupted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	Decelerates to a stop.	• *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	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bCommandAborted	The instruction is aborted.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	*bExecute* turns from TRUE to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



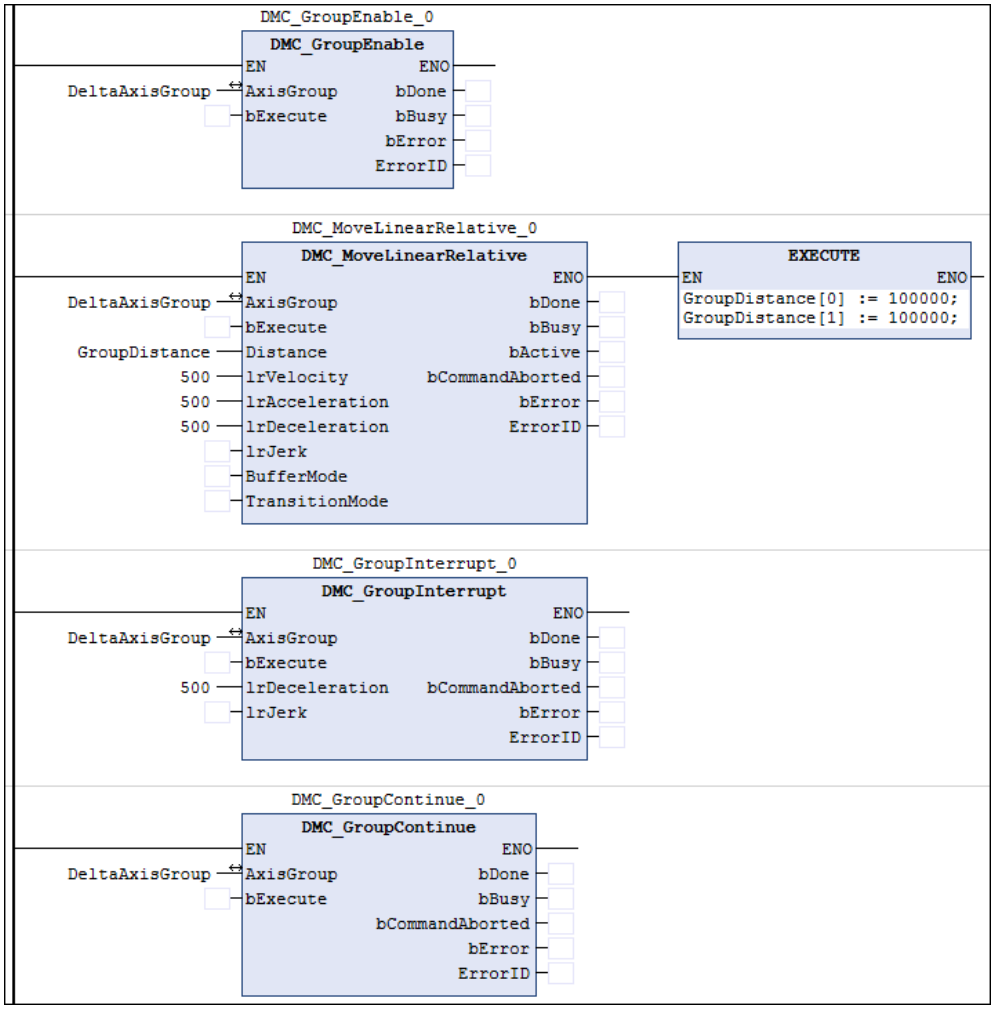
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*bExecute* turns to TRUE and *bBusy* is FALSE.

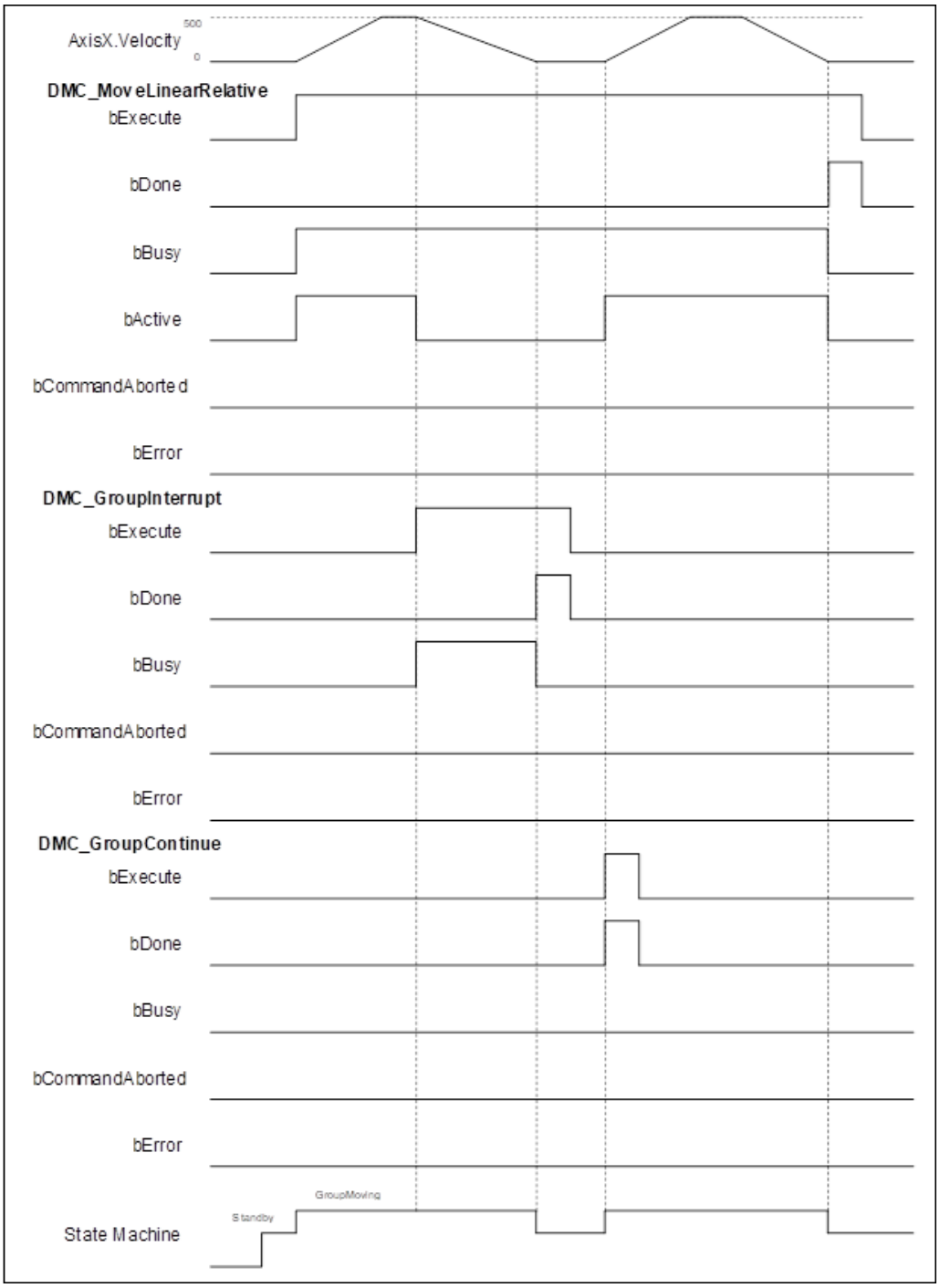
***Note**: DMC_AXIS_GROUP_REF(FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**
  - This instruction can decelerate and stop the motion of axis groups except DMC_GroupStop.
  - During deceleration, the status of the axis groups remains GroupMoving.
  - When the velocity reaches 0, the output parameter *bDone* changes to TRUE, and the status of the axis groups will switch to StandBy.
  - If the Group state machine is GroupMoving while running, the following Continue data will be recorded for subsequent DMC_GroupContinue to resume the motion after the instruction is done.
    - Instructions have not been executed (including the instructions in the buffer area that have not been executed).
    - Position after motion stops (*AxisGroup.ContinuePos*). After recording the data, *AxisGroup.bContinueDataWriten* will be set to TRUE. The running will not be recorded if it is not completed.
  - When Continue data is currently available, re-run DMC_GroupInterrupt. The current Continue data will be cleared. Whether the new Continue data is recorded depends on whether the running is successful.
  - When the DMC_GroupInterrupt lrDeceleration and *lrJerk* are set too small, it may cause the deceleration distance to exceed the target position of the axis group. After the DMC_GroupContinue is run, it will reverse to the target position of the axis group.
  - During the running of DMC_GroupInterrupt, users might encounter the following situation:
    - During DMC_GroupInterrupt running, DMC_GroupInterrupt will be interrupted when triggering DMC_GroupStop.
    - During DMC_GroupInterrupt running, when the second function block DMC_GroupInterrupt is re-triggered, it will be interrupted.
    - During DMC_GroupInterrupt running, DMC_GroupInterrupt continues running when axis groups motion instructions are running. Motion instructions are in the Busy state until DMC_GroupInterrupt running is completed, and the instructions will be added to the instruction buffer area to start running.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding troubleshooting, refer to **Appendix** of this manual.
- **Example**

- **Example 1**: 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.

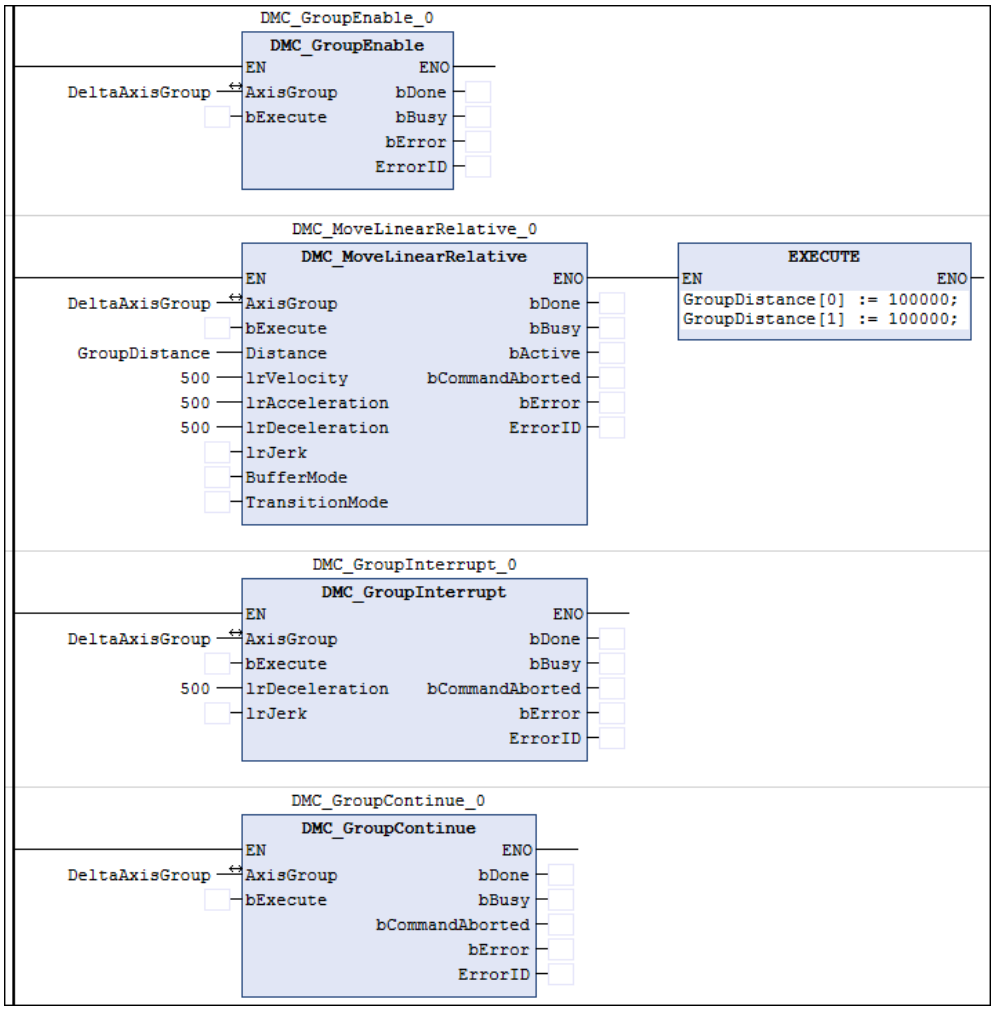


- Timing Diagram

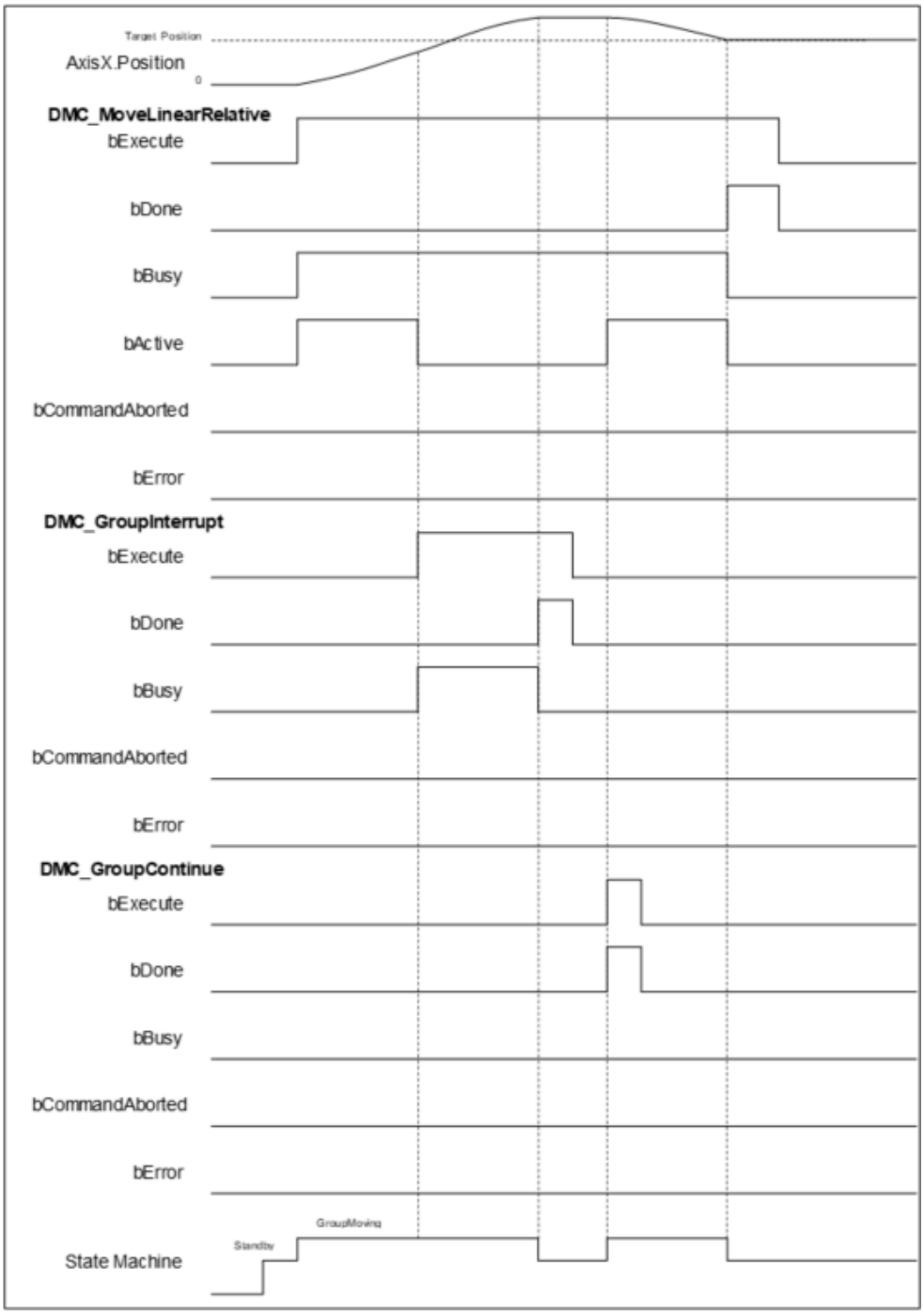


  - When *bExecute* of DMC_MoveLinearRelative changes to TRUE, the axis group starts to run.
  - When *bExecute* of DMC_GroupInterrupt changes to TRUE, the axis group will decelerate until the speed reaches 0 and stops. The status of DMC_GroupInterrupt will be changed from Busy to Done.
  - At this time, *bExecute* of DMC_GroupContinue changes to TRUE, and the unfinished motion path of previous DMC_MoveLinearRelative of the axis group will be completed.

- **Example 2**: This example shows that when the axis group is close to the target position, run the DMC_GroupInterrupt function block. After the axis group stops, use the DMC_GroupContinue function block to resume the motion.


DMC_GroupEnable_0
DMC_GroupEnable
EN
ENO
DeltaAxisGroup
AxisGroup
bExecute
bDone
bBusy
bError
ErrorID
DMC_MoveLinearRelative_0
DMC_MoveLinearRelative
EN
ENO
DeltaAxisGroup
AxisGroup
bExecute
GroupDistance
Distance
500
lrVelocity
500
lrAcceleration
500
lrDeceleration
lrJerk
BufferMode
TransitionMode
bDone
bBusy
bActive
bCommandAborted
bError
ErrorID
EXECUTE
EN
ENO
GroupDistance[0] := 100000;
GroupDistance[1] := 100000;
DMC_GroupInterrupt_0
DMC_GroupInterrupt
EN
ENO
DeltaAxisGroup
AxisGroup
bExecute
500
lrDeceleration
lrJerk
bDone
bBusy
bCommandAborted
bError
ErrorID
DMC_GroupContinue_0
DMC_GroupContinue
EN
ENO
DeltaAxisGroup
AxisGroup
bExecute
bDone
bBusy
bCommandAborted
bError
ErrorID

- Timing Diagram

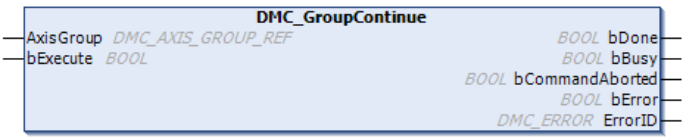


  - When the DMC_MoveLinearRelative's *bExecute* turns to TRUE, the axis group starts moving.
  - When the axis group is close to the target position, run DMC_GroupInterrupt, and when *bExecute* turns to TRUE, the axis group will slow down and to until the speed is 0, but it will exceed the target position.
  - At this time, the DMC_GroupContinue *bExecute* turns to TRUE. Use the previous setting of DMC_MoveLinearRelative to reverse the axis to the target position.

- **Supported Devices**
  - AX-series motion controller

2.2.1.13 DMC_GroupContinue

DMC_GroupContinue restores the interrupted motion of DMC_GroupInterrupt.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupContinue		DMC_GroupContinue_instance(AxisGroup: = , bExecute: = , bDone=> , bBusy=> , bCommandAborted=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when motion is resumed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

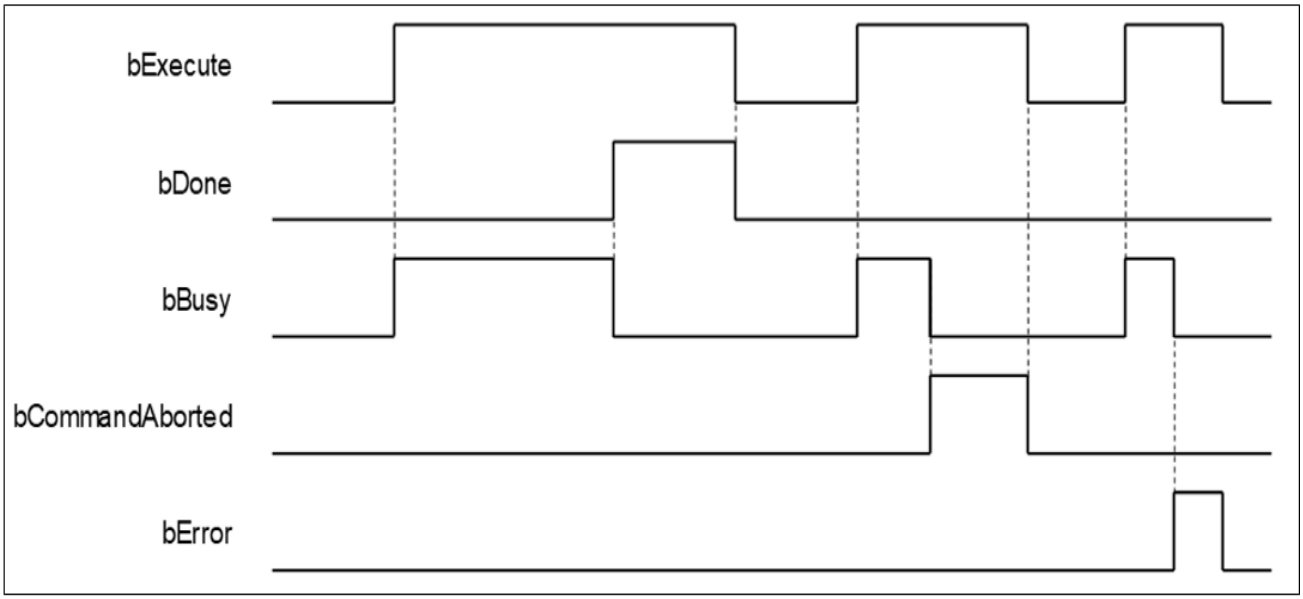
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The motion is resumed.	• *bExecute* turns to FALSE. • *bDone* will change to FALSE after remaining TRUE for one cycle when *bExecute* is FALSE but *bDone* changes to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	*bExecute* changes to TRUE.	• *bDone* turns to TRUE • *bError* turns to TRUE • *bCommandAborted* turns to TRUE
bCommandAborted	The instruction is interrupted by another function block.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	*bExecute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*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 to successfully run 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*.

- After successful executing the command, the output paremeter *bDone* changes to TRUE and clears the recorded Continue data.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding troubleshooting, refer to **Appendix** of this manual.

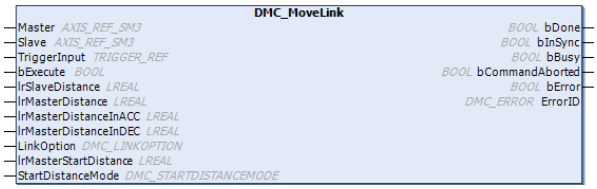
- **Example**

  Refer to *DMC_GroupInterrupt* programming example.

- **Supported Devices**
  - AX-series motion controller

2.2.1.14 DMC_MoveLink

DMC_MoveLink enables the slave axis to move following the master axis for synchronous positioning.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_MoveLink		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	*bExecute* turns to TRUE.
lrSlaveDistance	Slave axis travel distance (user units)	LREAL	Positive, negative, or 0 (0)	*bExecute* turns to TRUE.
LrMasterDistance	Master axis travel distance (user unit)	LREAL	positive (0)	*bExecute* turns to TRUE.
lrMasterDistanceInACC	Master axis acceleration travel distance (user unit)	LREAL	positive (0)	*bExecute* turns to TRUE.
lrMasterDistanceInDEC	Master axis deceleration travel distance (user unit)	LREAL	positive (0)	*bExecute* turns to TRUE.
LinkOption	Synchronization start condition selection	DMC_LINKOPTION[*1]	0: COMMANDEEXECUTION 1: TRIGGERDETECTION 2: MASTERREACH (COMMANDEEXECUTION)	*bExecute* turns to TRUE.
lrMasterStartDistance	Master axis following distance (user unit)	LREAL	Positive, negative, or 0 (0)	*bExecute* turns to TRUE.
StartDistanceMode	Master axis following distance mode	DMC_STARTDISTANCEMODE[*2]	0: ABSOLUTE 1: RELATIVE (ABSOLUTE)	*bExecute* turns to TRUE.

***Note**:

1. DMC_LINKOPTION: Enumeration (Enum)
2. DMC_STARTDISTANCEMODE: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	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 the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR（DMC_NO_ERROR)

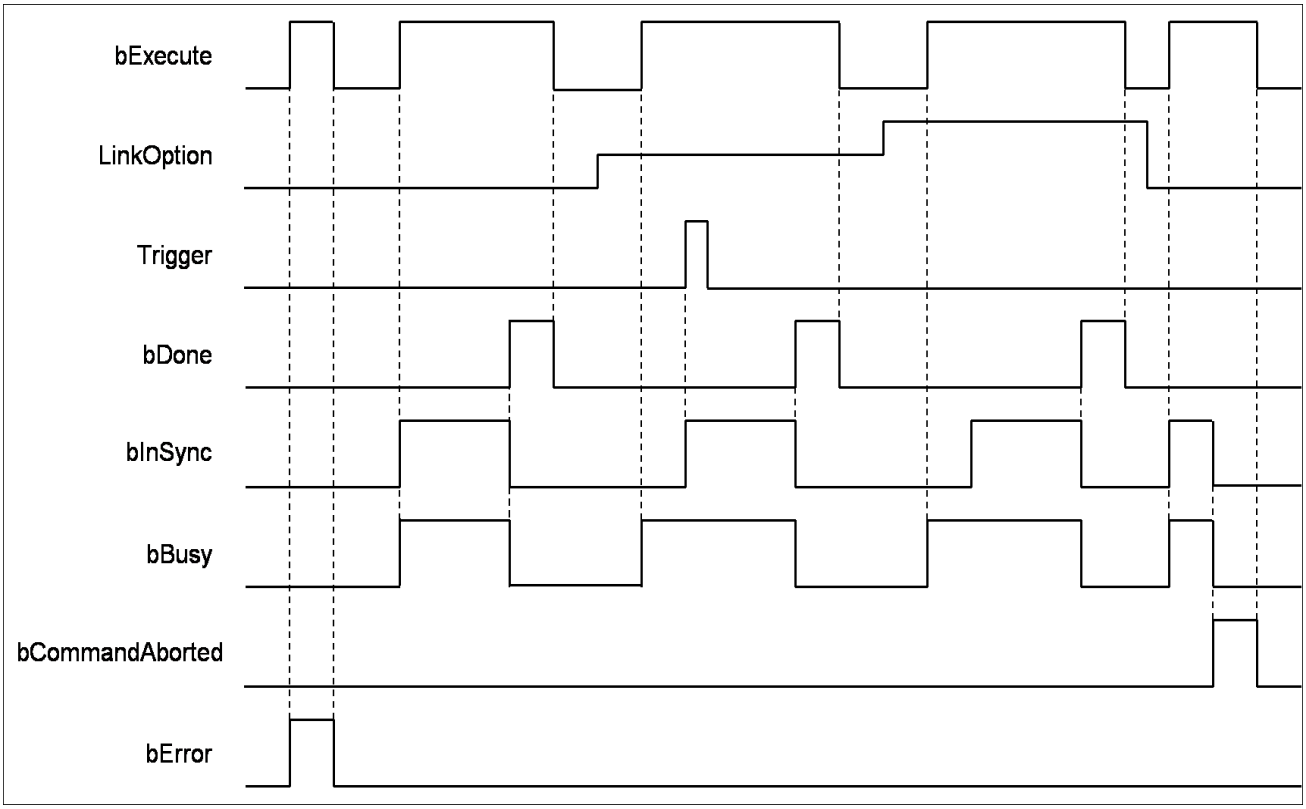
***Note**: DMC_ERROR: enumerate (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The slave axis positioning is completed.	• *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	The slave axis is synchronized.	When completing positioning
bBusy	The upper edge of *bExecute* fires.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bCommandAborted	The Instruction is interrupted by another Function block.	• *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	An error occurs during the execution of the instruction or the input value of the instruction	*bExecute* turns to FALSE (Clear the error code recorded in *ErrorID*.)
ErrorID		

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
	is incorrect. (Error code is recorded in *ErrorID*.).	

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Slave	Specifies the slave axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	*bExecute* rises and *bBusy* status is FALSE.
TriggerInput	Trigger signal	TRIGGER_REF*2	TRIGGER_REF	*bExecute* turns to TRUE.
Master	Specifies the master axis.	AXIS_REF_SM3*1	AXIS_REF_SM3	*bExecute* rises and *bBusy* status is FALSE.

***Note**:

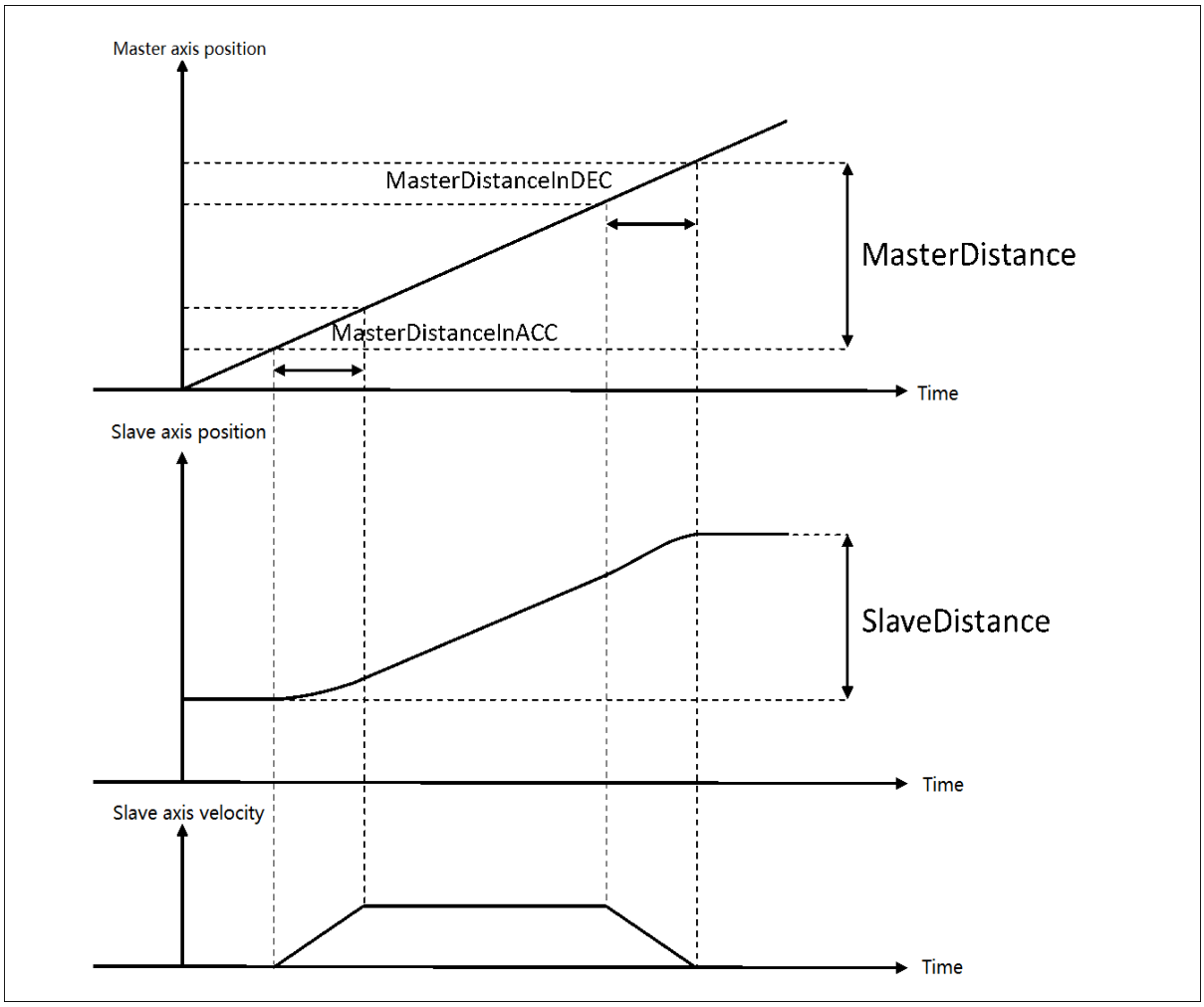
1. AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.
2. TRIGGER_REF: Structure (STRUCT)

Name	Function	Data Type	Setting Value (Default Value)
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)

Name	Function	Data Type	Setting Value (Default Value)
bInput	When the controller is triggered, it is the trigger signal source.	BOOL	Trigger source
bActive*	Shows 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 set by *LinkOption*.
    - COMMANDEEXECUTION: When DMC_MoveLinke starts, the slave axis starts the synchronization and executes positioning motions.
    - TRIGGERDETECTION: After DMC_MoveLinke starts and an external signal triggered, the slave axis starts the synchronization and executes positioning motion.
    - MASTERREACH: After the master axis reaches the specified position, the slave axis starts the synchronization and executes the positioning motion.
  - If *LinkOption* = MASTERREACH mode, *StartDistanceMode* determines the specified position of the master axis:
    - ABSOLUTE: The specified position 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*).



- Acceleration section

Interval	The relationship between the master axis and the slave axis motion amount	
Acceleration section	Master Axis	MasterDistanceInACC
	Slave Axis	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 motion amount	
Constant Velocity Section	Master Axis	MasterDistance–MasterDistanceInACC–MasterDistanceInDEC
	Slave Axis	SlaveDistance –Slave Axis Acceleration Section moving distance–Slave Axis Deceleration Section moving distance

- Deceleration section

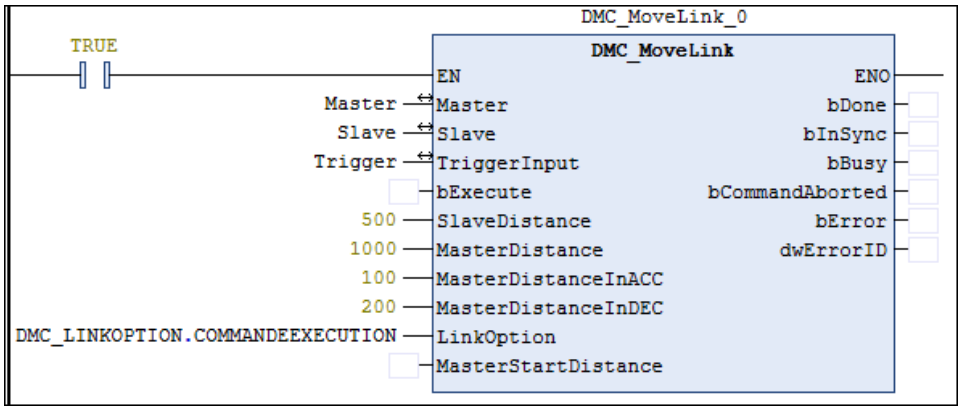
Interval	The relationship between the master axis and the slave axis motion amount	
Deceleration	Master Axis	MasterDistanceInDEC
	Slave Axis	SlaveDistance

Interval	The relationship between the master axis and the slave axis motion amount	
Section		$\times \frac{\frac{MasterDistanceInDEC}{2}}{\frac{MasterDistanceInACC}{2}+(MasterDistance-MasterDistanceInACC-MasterDistanceInDEC)+\frac{MasterDistanceInDEC}{2}}$

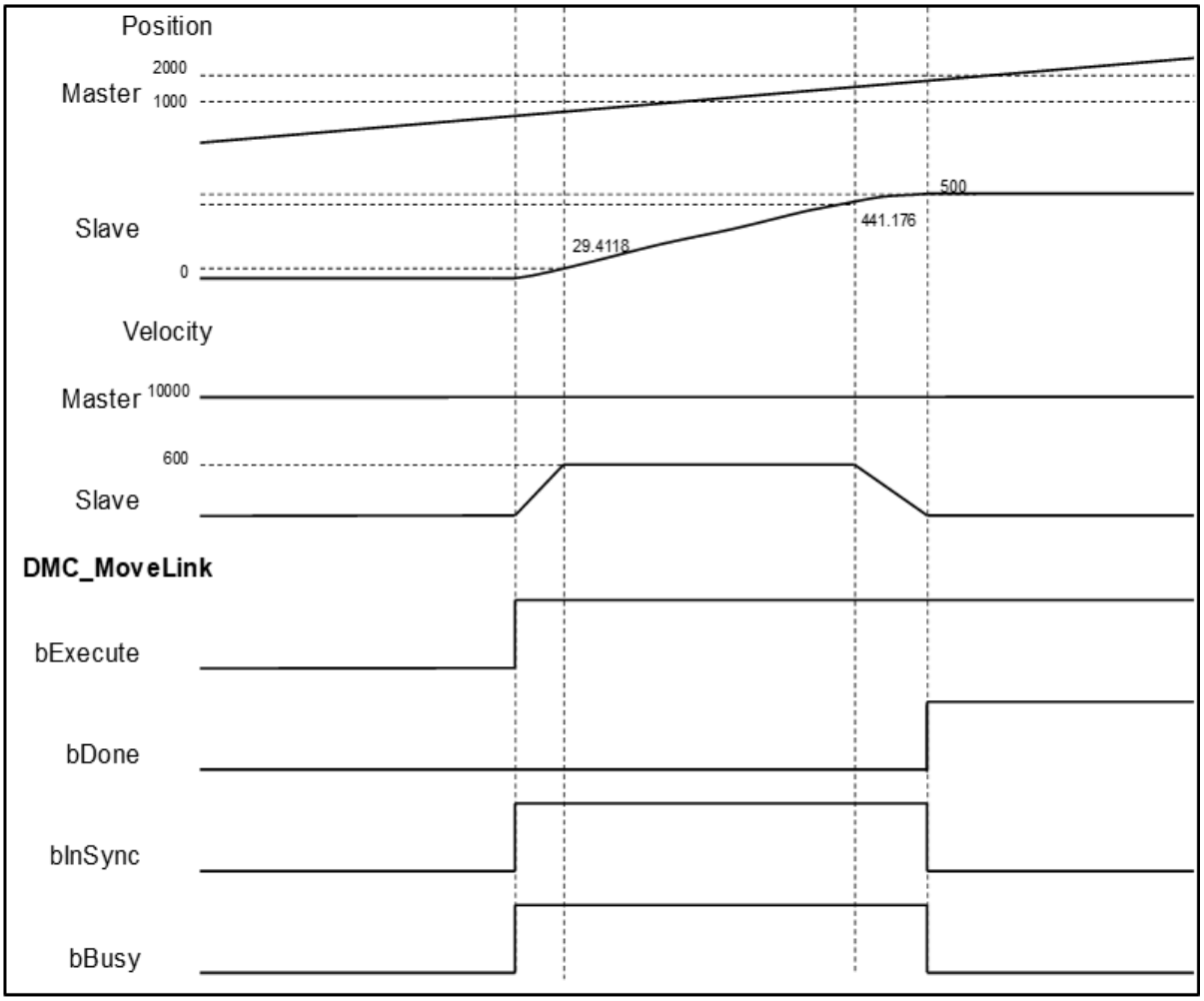
- When TRIGGERDETECT has the same driver mode and controller mode as MC_TouchProbe, refer to the MC_TouchProbe instruction manual for the usage of the two modes and the usage settings of TriggerInput.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**
  - **Example 1**: The example shows that DMC_MoveLink operates in COMMANDEEXECUTION mode.

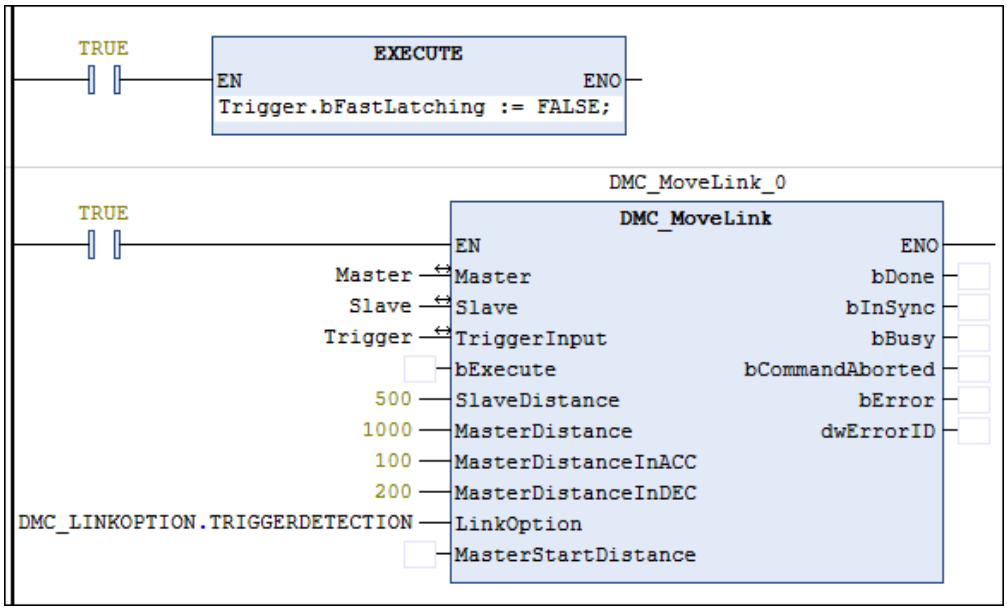


  - Timing Diagram

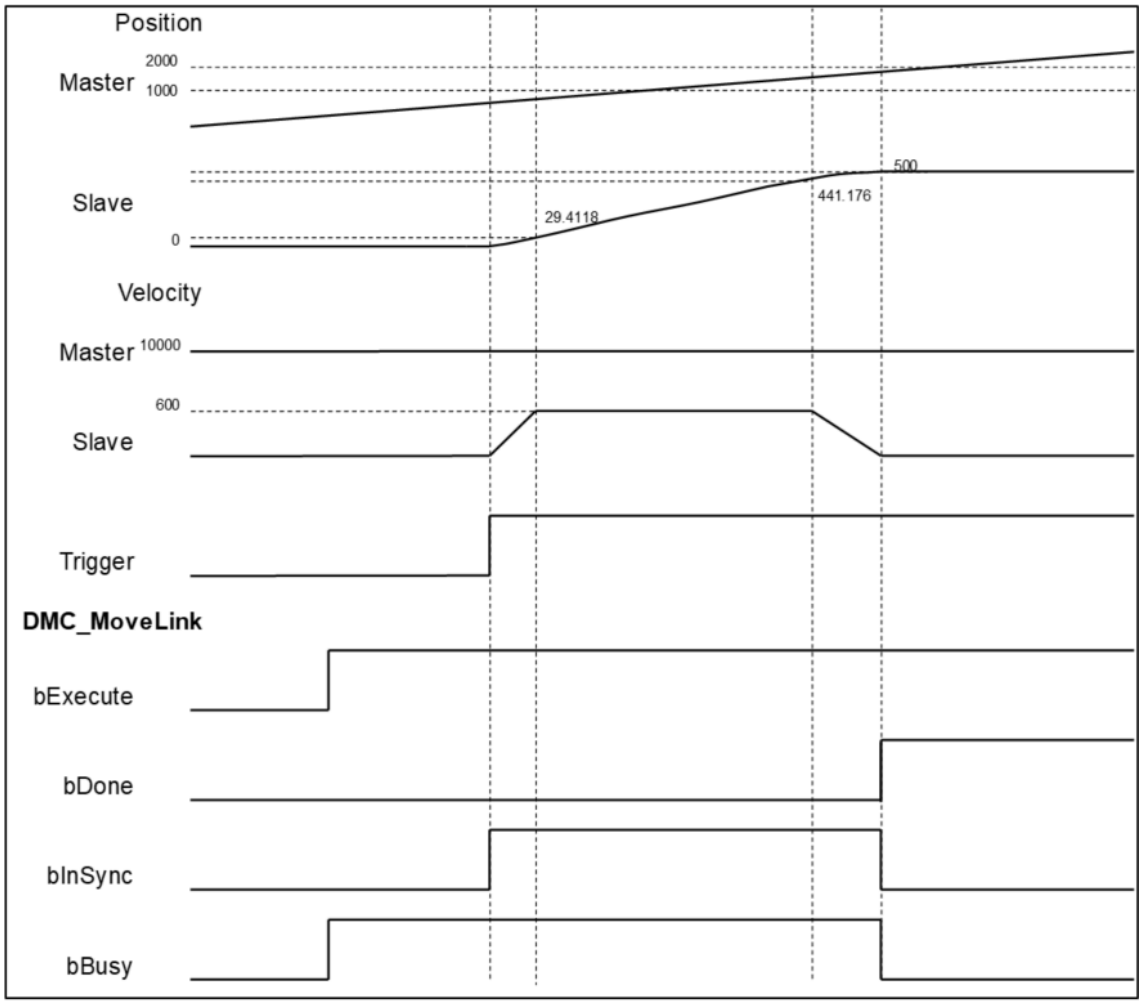


1. When *LinkOption* is set to COMMANDEEXECUTION, the slave axis starts the synchronization right after DMC_MoveLink is activated and performs positioning motion according to the speed of the master axis.
2. According to the four parameters of *MasterDistanceInACC*, *MasterDistanceInDEC*, *MasterDistance*, 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 dynamic motion, the slave axis stops the synchronization, and DMC_MoveLink completes the synchronous positioning motion.

- **Example 2**: The example shows that DMC_MoveLink operates in TRIGGERDETECTION mode. (External trigger using controller mode)

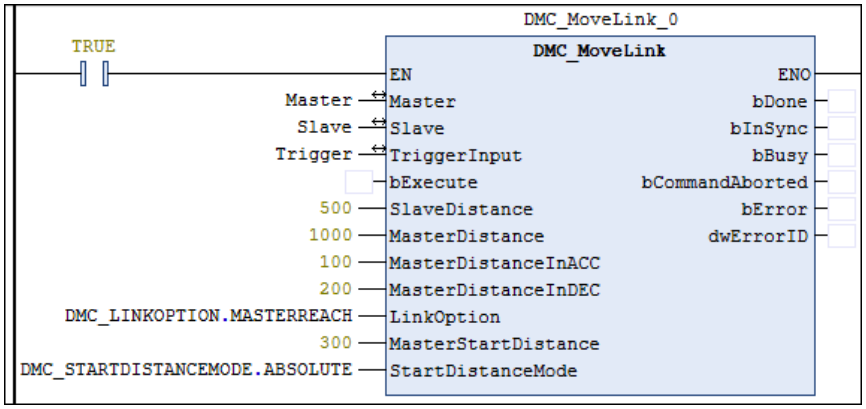


- Timing Diagram

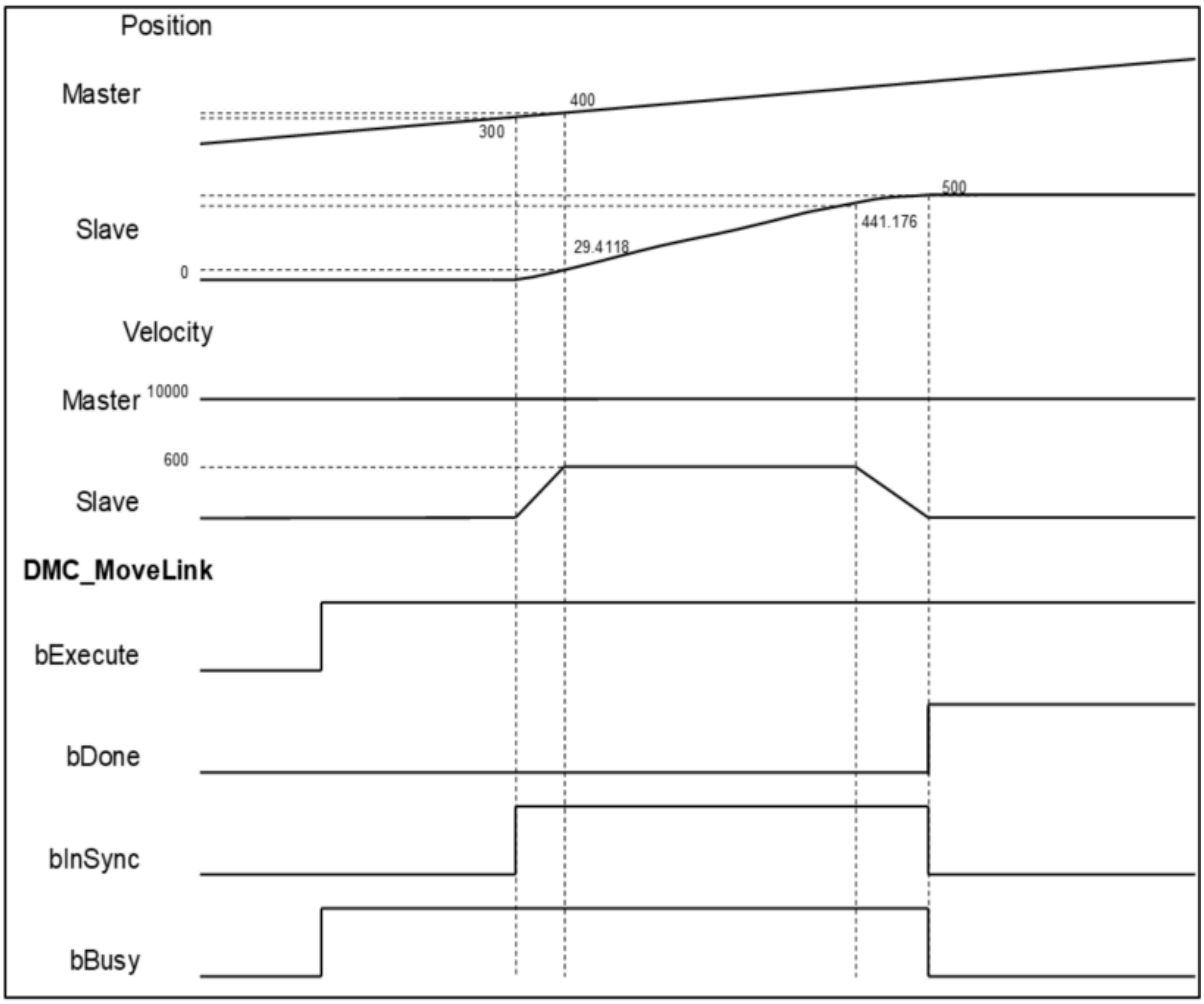


1. When the *LinkOption* is set to TRIGGERDETECTION, after the DMC_MoveLink is activated, the trigger signal must be used to make the slave axis enter the synchronization mode. After the trigger, the slave axis will follow the speed of the master axis to perform positioning motion.
2. According to the 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 motion, the slave axis leaves the synchronization relationship, and DMC_MoveLink completes the synchronous positioning motion.

- **Example 3**: The example illustrates that DMC_MoveLink operates in MASTERREACH mode.



- Timing Diagram

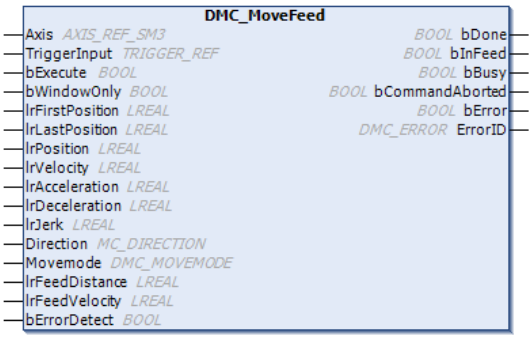


1. When the *LinkOption* is set to MASTERREACH, after DMC_MoveLink is activated, the master axis must pass the position set by the *MasterStartDistance*, the slave axis enters the synchronous mode, and the positioning motion is performed according to the speed of the master axis.
2. *StartDistanceMode* 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 motion, the slave axis leaves the synchronization relationship, and DMC_MoveLink completes the synchronous positioning motion.

- **Supported Devices**
  - AX-series motion controller

2.2.1.15 DMC_MoveFeed

DMC_MoveFeed can specify an external interrupt input. During the motion of the target, the position where the interrupt occurs is used as the starting point to perform the positioning motion.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_MoveFeed		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	*bExecute* turns to TRUE.
bWindowOnly	Enables the Window range setting.	BOOL	TRUE/FALSE (FALSE)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrFirstPosition	Defines the start position of Window (Unit: user unit)	LREAL	Negative, Positive, or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrLastPosition	Defines the end position of Window (Unit: user unit)	LREAL	Negative, Positive, or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrPosition	Absolute target position (Unit: user unit)	LREAL	Negative, Positive, or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrAcceleration	Acceleration rate (Unit: user unit/s2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrDeceleration	Deceleration rate (Unit: user unit/s2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrJerk	The jerk (Unit: user unit/s3)	LREAL	Positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
Direction	The motion direction	MC_DIRECTION[*1]	–1: negative 0 : shortest 1 : positive 2 : current 3 : fastest (shortest)	*bExecute* turns to TRUE and *bBusy* is FALSE.
MoveMode	Target motion mode	DMC_MOVEMODE[*2]	0 : ABSOLUTE 1 : RELATIVE 2 : VELOCITY (ABSOLUTE)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrFeedDistance	Standard distance (Unit: user unit)	LREAL	Negative, Positive, or 0 (0)	The signal is at the rising edge
lrFeedVelocity	Standard speed (user unit)	LREAL	Positive (0)	When the signal is at the rising edge
bErrorDetect	Error detection selection	BOOL	TRUE/FALSE (FALSE)	*bExecute* turns to TRUE and *bBusy* is FALSE.

***Note**:

1. MC_DIRECTION: Enumeration (Enum)
2. DMC_MOVEMODE: Enumeration (Enum)

- **Outputs**

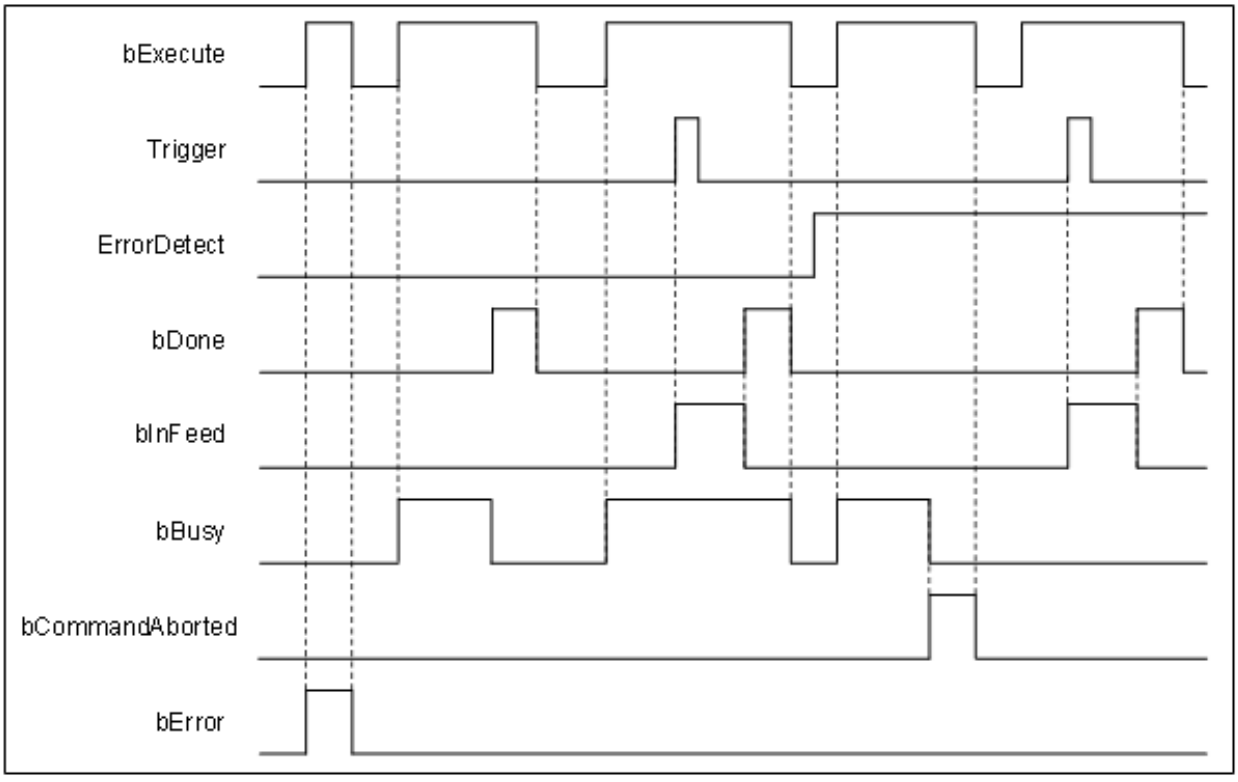
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the standard motion is complete or the target motion is completed and *bErrorDetect* is FALSE	BOOL	TRUE/FALSE (FALSE)
bInFeed	TRUE in standard motion	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The standard motion is completed or the target motion is completed and *bErrorDetect* is FALSE.	• *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 motion	Completing standard motion.
bBusy	*bExecute* turns to TRUE	• *bDone* turns to TRUE. • *bError* urns to TRUE.
bCommandAborted	The function block is interrupted or when the target motion is complete and *bErrorDetect* is TRUE.	*bExecute* turns to FALSE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Clear the error code recorded in *ErrorID*.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3 (FB)*1	AXIS_REF_SM3	*bExcute* turns to TRUE.
TriggerInput	Trigger signal	TRIGGER_REF*2	TRIGGER_REF	*bExecute* turns to TRUE.

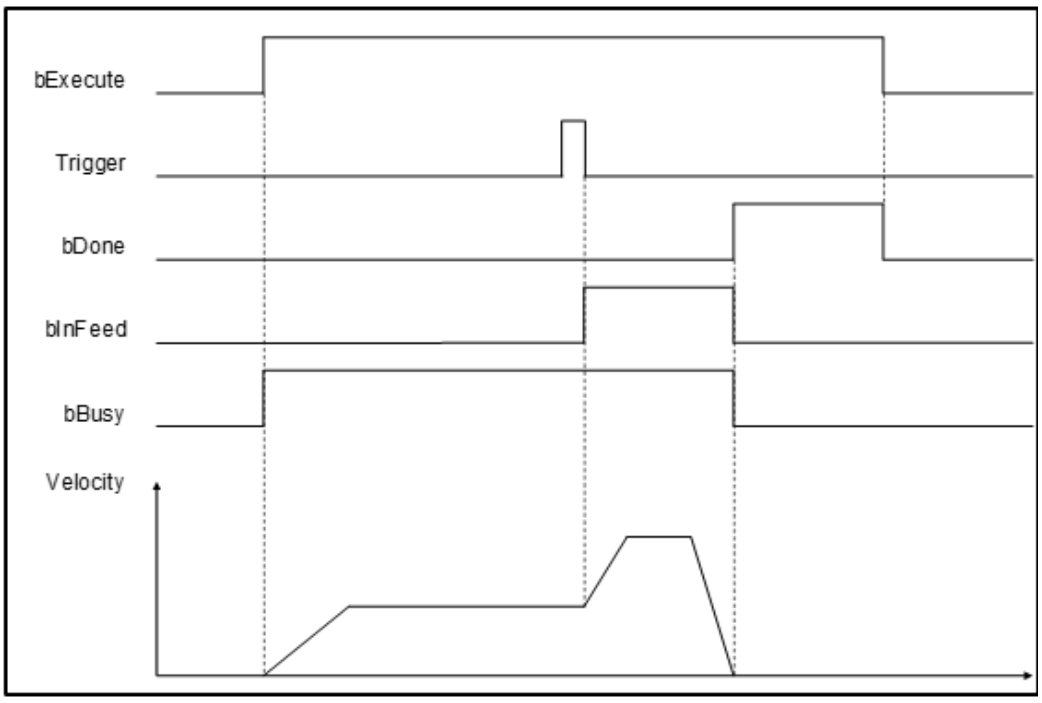
***Note**:

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).

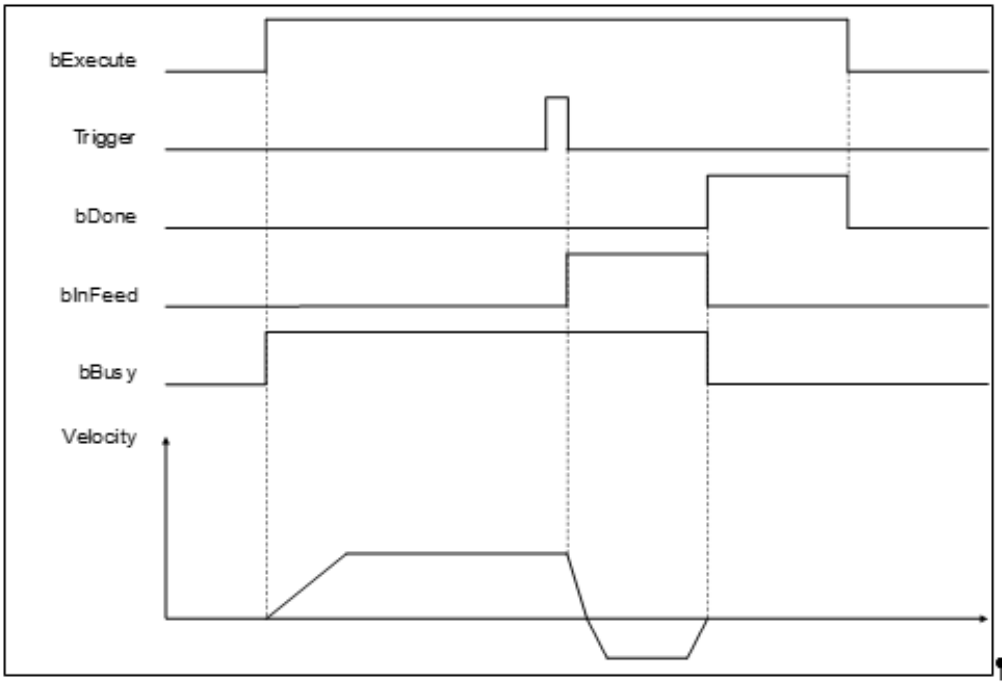
Name	Function	Data Type	Setting Value (Default Value)
iTriggerNumber	Trigger channel	INT	0: Touch Probe 1 captures data when TRUE 1: Touch Probe 1 captures data when FALSE 2: Touch Probe 2 captures data when TRUE 3: Touch Probe 2 captures data when FALSE
bFastLatching	Trigger signal	BOOL	TRUE: Drive 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**
  - TRIGGERDETECT has driver mode and controller mode like MC_TouchProbe. Refer to MC_TouchProbe for these two modes and the TriggerInput setting.
  - MoveMode sets the target motion (the first motion) mode. The absolute motion (ABSOLUTE) and the relative motion (RELATIVE) will be completed if there is no trigger signal. At this time, the DMC_MoveFeed function block will go to the next stage according to the *bErrorDetect* setting, and the velocity motion (VELOCITY) will continue to run regardless of the *bErrorDetect* state.
  - When *bErrorDetect* is FALSE, and the target motion (the first motion) is completed, *bDone* turns to TRUE, and the function block completes executing. When *bErrorDetect* is TRUE, and the target motion (first motion) is completed, *bCommandAborted* turns to TRUE, and the function block stops.
  - During standard motion (second motion), relative motion will be performed according to the standard distance (*lrFeedDistance*). When *lrFeedDistance* is a positive value, the axis will maintain the original motion direction for standard motion.



- When *IrFeedDistance* is negative, the axis will perform standard motion in the opposite direction of the current motion.

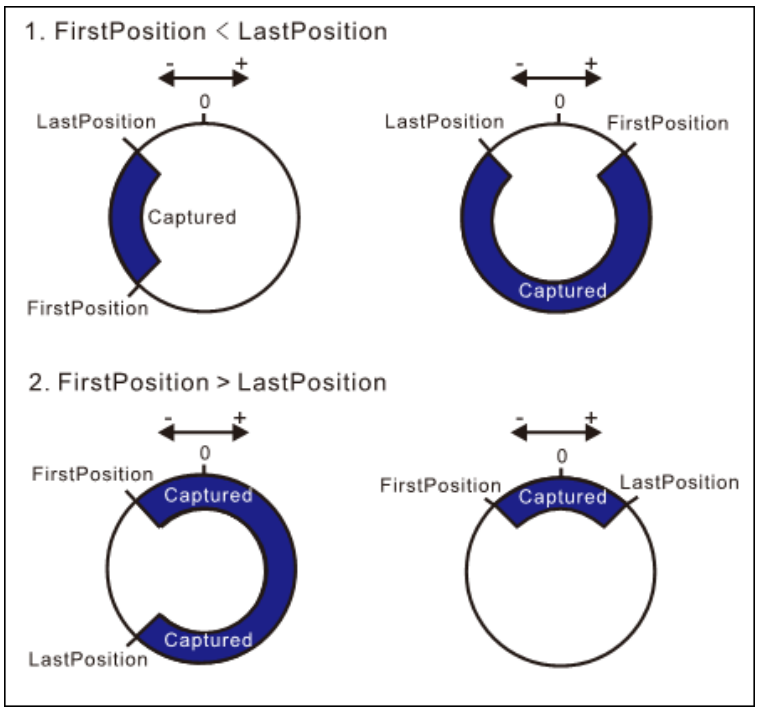


- When *IrFeedDistance* is set to 0, the axis will stop immediately.

- In the 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, the function block will not respond immediately when the Trigger signal comes. To react in time, configure the above two PDOs.

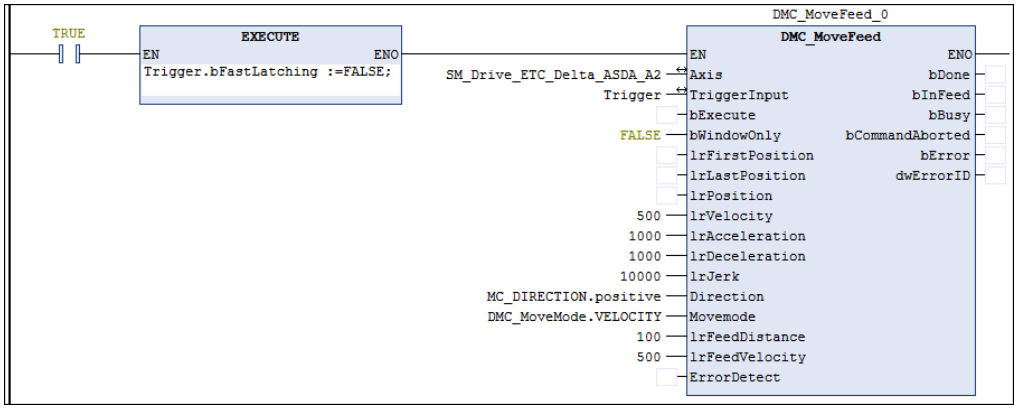
- Window Mask setting

  - 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* intervals are shown in the following figure.

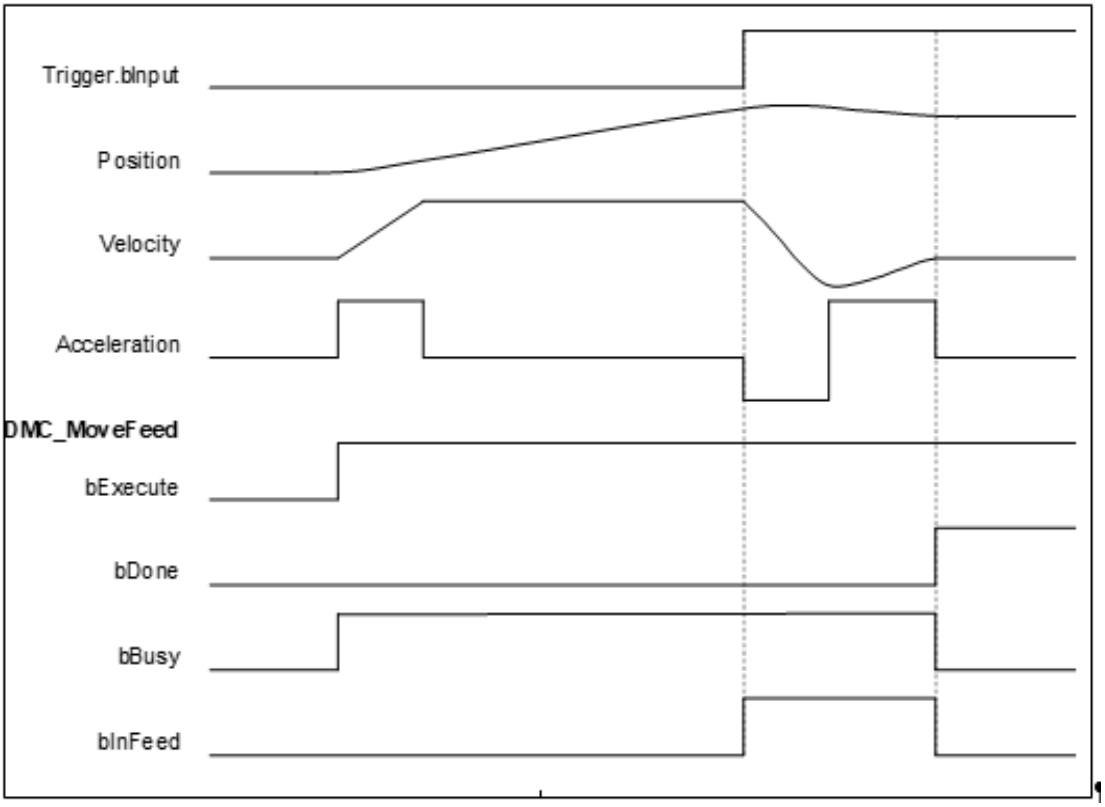


- In the linear axis mode, the Windows Mask setting range must be *FirstPosition* < *LastPosition*, and the trigger can only work within the range value.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - **Example 1**: The example shows the running result of using DMC_MoveFeed in the controller mode.

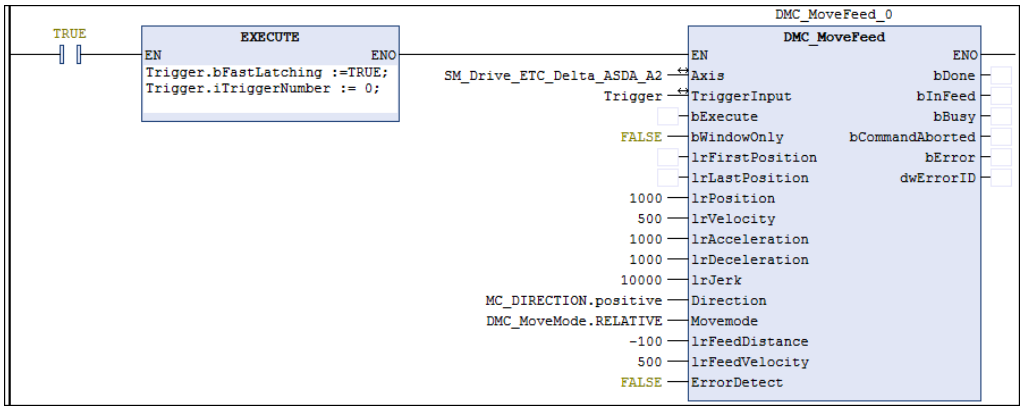


    - Timing Diagram

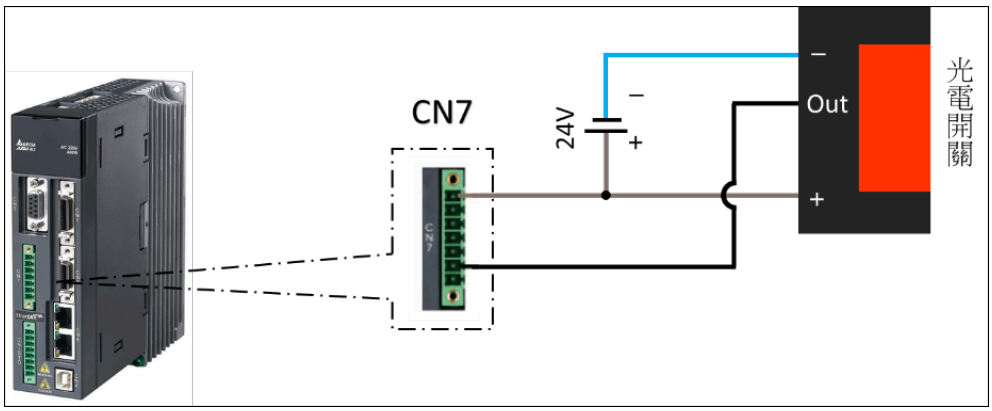


1. After DMC_MoveFeed is started, the axis starts to run with the target position, velocity, acceleration, and motion mode input from the function block and waits for the trigger signal of the controller mode.
2. After the controller mode is triggered, the axis will move according to the position and velocity of the second standard motion.
3. Since the standard distance (*lrFeedDistance*) is a positive value, the axis maintains the original motion direction and performs standard motion after triggering.

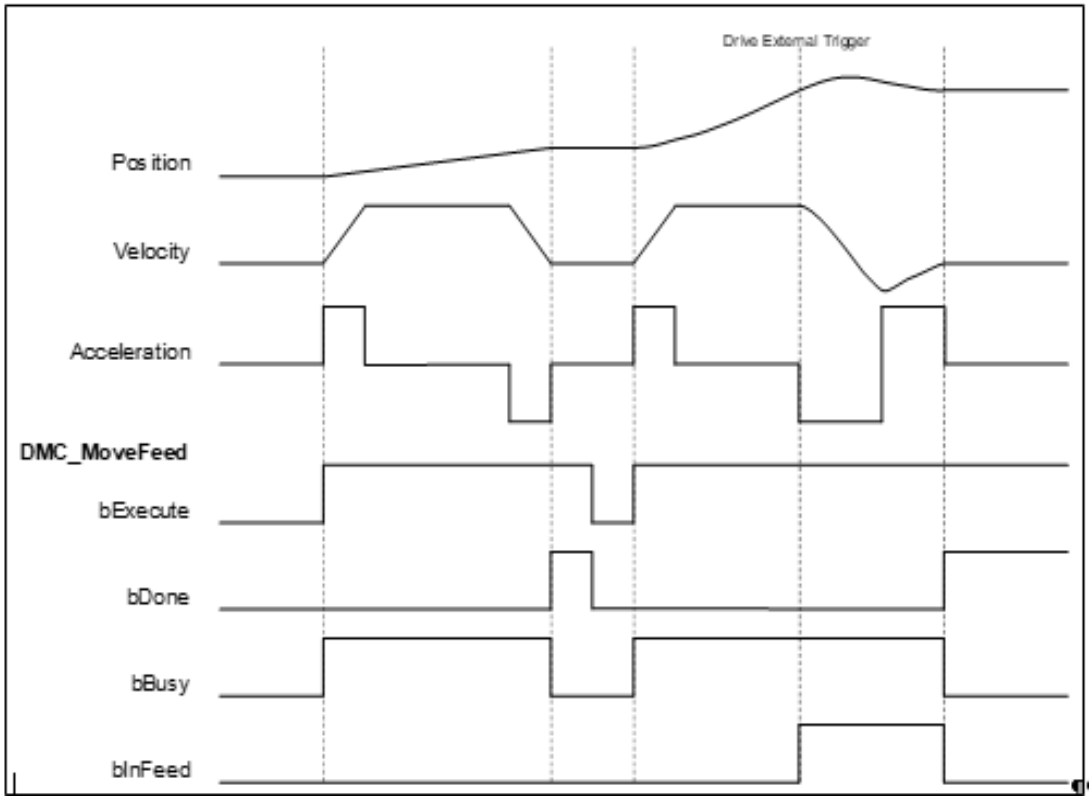
  - **Example 2**: The example illustrates the running result of using DMC_MoveFeed in drive mode.



- 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 motion 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 driver's external signal when the first segment of target motion has not been completed.
4. After triggering, the axis will follow the position and speed of the second standard motion. 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 mode trigger is more real-time than the controller, there will be a slight error when observing the relationship between *InFeed* and the position.

- **Supported Devices**
  - AX-series motion controller

2.2.1.16 DMC_GroupReadSetPosition

DMC_GroupReadSetPosition reads returns the current set-position of an axis group.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupReadSetPosition		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	Runs the instruction when *bEnable* changes from FALSE 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)	*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 (Default Value)
bValid	TRUE when the output value is valid	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*1	DMC_ERROR (DMC_NO_ERROR)
Position	The current 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	DL_Kinematics.CONFIG_DATA*2 (Reversed)	-

Name	Function	Data Type	Output Range (Default Value)
	Data Type corresponding to the current Instruction position of the axis group.		

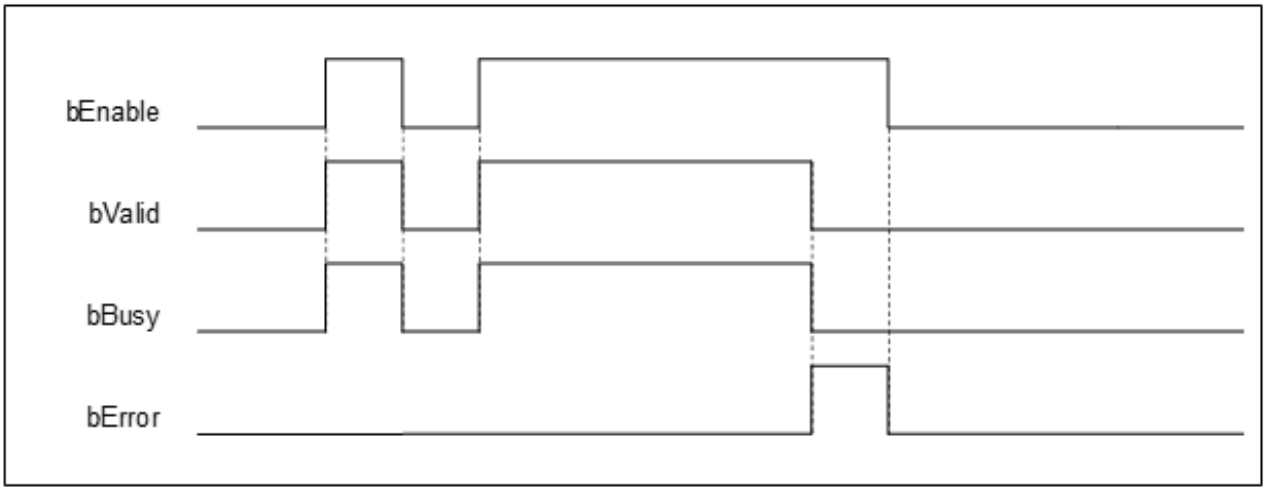
***Note**:

1. DMC_ERROR: Enumeration (Enum)
2. Different configurations have different stored data.

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bValid	*bEnable* turns to TRUE and *IrValueOutput* is valid.	• *bEnable* turns to TRUE. • *bError* turns to TRUE.
bBusy	*bEnable* turns to TRUE.	• *bValid* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* turns to TRUE. (Clear the error code recorded in *ErrorID*)
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 of Output Parameter Changes**



- **Inputs/Outputs**

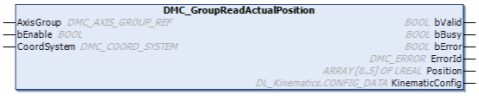
Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*bEnable* turns to TRUE and *bBusy* is FALSE.

***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**
  - This function is available for DL_MotionControl V1.2.0.0 or later.
  - This function block will output the valid value only when the axis group state is not GroupDisable.
  - If the read position is a cassette coordinate system, *KinematicConfig* will output the configuration data type corresponding to the Position. If the coordinate system is ACS, this parameter does not take effect.
  - CoordSystem function is supported by DL_MotionControl library V1.1.0.0 or later.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For the error codes and corresponding troubleshooting methods, refer to the **Appendix** of this manual.
- **Supported Devices**
  - AX-series motion controller

2.2.1.17 DMC_GroupReadActPosition

DMC_GroupReadActPosition reads the current position of the axis group.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupReadActPosition		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	Runs the instruction when *bEnable* changes from FALSE 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)	*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 (Default Value)
bValid	TRUE when the output value is valid	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*1	DMC_ERROR (DMC_NO_ERROR)
Position	The current 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 (when it is not ACS),	DL_Kinematics.CONFIG_DATA*2 (Reversed)	-

Name	Function	Data Type	Output Range (Default Value)
	the configuration data type corresponding to the current position of the axis group.		

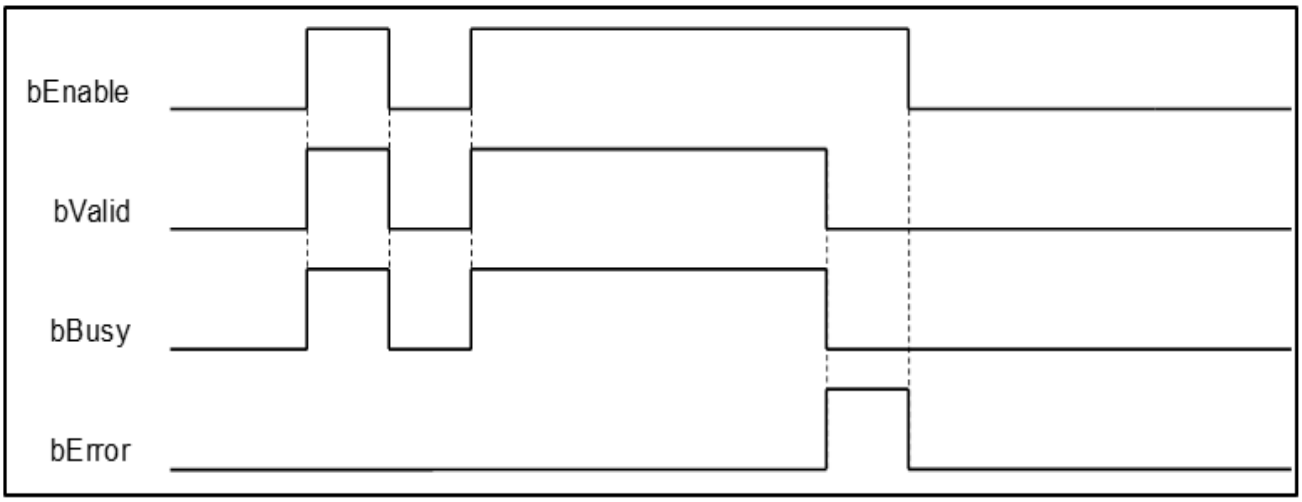
***Note**:

1. DMC_ERROR: Enumeration (Enum)
2. Different configurations have different stored data.

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bValid	*bEnable* turns to TRUE and the *lrValue*output is valid.	• *bEnable* turns to TRUE. • *bError* turns to TRUE.
bBusy	*bEnable* is TRUE.	• *bValid* turns to TRUE. • *bError* turns to TRUE.
bError ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	When *bEnable* turns to TRUE (Clear the Error Code)
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 of Output Parameter Changes**



- **Inputs/Outputs**

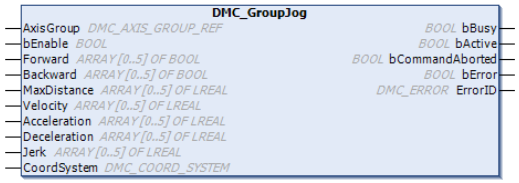
Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*bEnable* turns to TRUE and *bBusy* is FALSE.

***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**
  - This function is available for DL_MotionControl V1.2.0.0 or later.
  - This function block will output the valid value only when the axis group state is not GroupDisable.
  - If the read position is a cassette coordinate system, *KinematicConfig* will output the configuration data type corresponding to the Position. If the coordinate system is ACS, this parameter does not take effect.
  - CoordSystem function is supported by DL_MotionControl library V1.1.0.0 or later.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding troubleshooting methods, refer to **Appendix** of this manual.
- **Supported Devices**
  - AX-series motion controller

2.2.1.18 DMC_GroupJog

DMC_GroupJog activates the forward and reverse jogging of the axis group to the specified coordinates.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupJog		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	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Forward	Run the forward jog of each coordinate axis	BOOL[6]	[_, _, _, _, _, _] TRUE/FALSE ([_, _, _, _, _, _] FALSE)	Only works when *bEnable* = TRUE
Backward	Run the reverse jog of each coordinate axis	BOOL[6]	[_, _, _, _, _, _] TRUE/FALSE ([_, _, _, _, _, _] FALSE)	Only works when *bEnable* = 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 *bEnable* = TRUE
Velocity	Set the maximum speed of each coordinate axis inching	LREAL[6]	[_, _, _, _, _, _] Positive ([0, 0, 0, 0, 0, 0])	Only works when *bEnable* = TRUE
Acceleration	Set the maximum acceleration of each coordinate axis inching	LREAL[6]	[_, _, _, _, _, _] Positive ([0, 0, 0, 0, 0, 0])	Only works when *bEnable* = TRUE

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Jerk	Set the maximum jerk of each coordinate axis inching	LREAL[6]	[_, _, _, _, _, _] Positive ([0, 0, 0, 0, 0, 0])	Only works when *bEnable* = 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 *bEnable* = TRUE

***Note**: DMC_COORD_SYSTEM: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bValid	TRUE when the instruction is triggered to run	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 within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

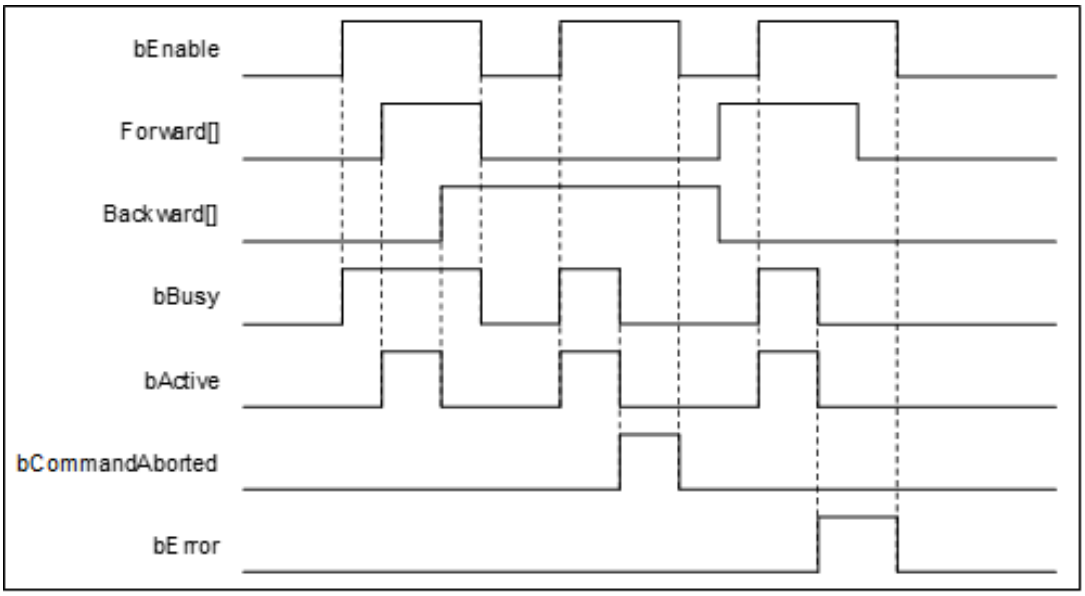
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bValid	*bEnable* turns to TRUE.	• *bEnable* turns to TRUE. • *bError* turns to TRUE.
bBusy	*Forward*/*Backward* is jogging.	• *bValid* turns to TRUE. • *bError* turns to TRUE.
bCommandAborted	• The instruction is aborted • This instruction is interrupted by MC_Stop • This instruction is interrupted by DMC_GroupStop	• *bEnable* turns to TRUE • *bCommandAborted* will change to FALSE after remaining TRUE for one cycle when *bExecute* is FALSE but *bCommandAborted* changes to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* turns to TRUE. (Clear the Error Code)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*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.

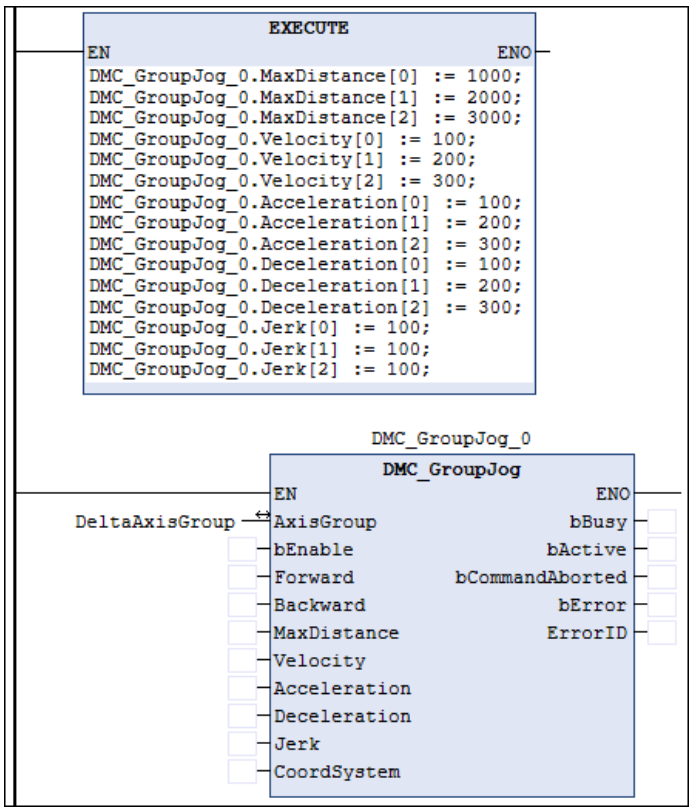
- **Function**
  - This function is supported by DL_MotionControl V1.2.0.0 or later.
  - CoordSystem function is supported by DL_MotionControl library V1.1.0.0 or later.
  - When *bEnable* is TRUE, according to the coordinate system specified by *CoordSystem*, the value of *MaxDistance*, *Velocity*, *Acceleration*, *Deceleration*, and *Jerk* determines the relevant motion parameters of each coordinate axis of the coordinate system, and uses *Forward* and *Backward* as the switch to start the forward and reverse directions of each coordinate axis jogging.
  - When *bEnable* is FALSE, the axis group will stop jogging and decelerate to 0.
  - The jog motion of the coordinate axis will start only when one of *Forward* and *Backward* is TRUE.
  - When *MaxDistance* is set to 0, there is no motion range limit.
  - *MaxDistance*, *Velocity*, *Acceleration*, *Deceleration*, and *Jerk* modification will not affect the current jogging. You need to restart *Forward* or *Backward* to take effect.
  - Modifying the CoordSystem breaks jogging in all directions. A new jog needs to be restarted for Forward or 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.

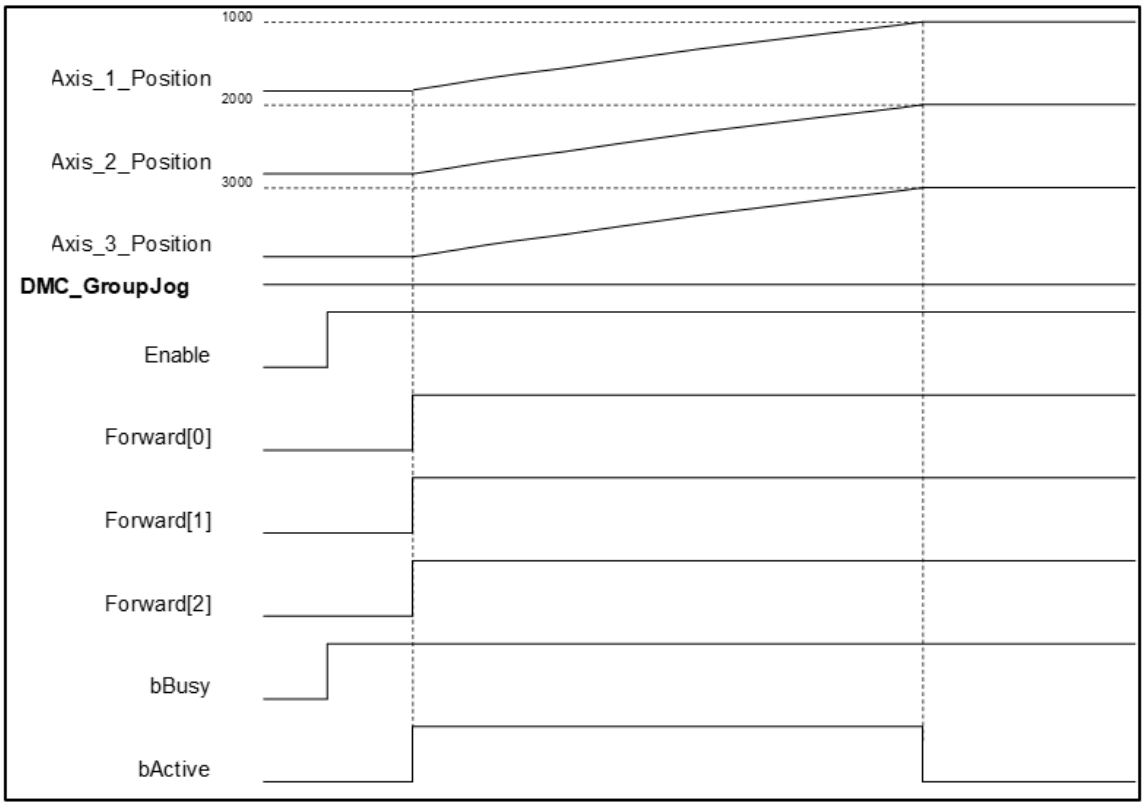
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding troubleshooting methods, refer to the **Appendix** of this manual.

- **Example**

This example shows how to use DMC_GroupJog to control axis group motion and make 3 axes move.



- Timing Diagram

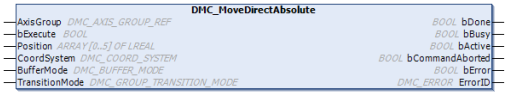


  - When the *Enable* of DMC_GroupJog is TRUE, Forward[0]– Forward[2] pins are activated, and the axis will start to run to the MaxDistance setting position and then stop running.
  - When any axis in the axis group is running, *bActive* of DMC_GroupJog is TRUE. *bActive* is FALSE after the axis group runs.

- **Supported Devices**
  - AX-series motion controller

2.2.1.19 DMC_MoveDirectAbsolute

DMC_MoveDirectAbsolute controls the axis group moving to the absolute position in the coordinate system. Each axis is operated independently during the motion, and the motion path is not specified.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_MoveDirectAbsolute		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
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Position	Specify the absolute target position for each axis in the axis group. (Unit: user unit)	LREAL[6]	[_, _, _, _, _, _] Positive or negative value ([0, 0, 0, 0, 0, 0])	*bExecute* is on the rising edge, the setting parameters of *Position* will be updated.
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)	*bExecute* is on the rising edge, the parameters of *CoordSystem* will be updated.
BufferMode	Specifies the buffer mode for this instruction*2	DMC_BUFFER_MODE*2	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh (0)	*bExecute* is on the rising edge, the parameters of *BufferMode* will be updated.
TransitionMode	Specifies the transition mode for this instruction*3	DMC_GROUP_TRANSITION_MODE*3	0: None 10: Overlap 11: Single_axis (0)	*bExecute* is on the rising edge, the 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 Operation Manual*.
3. About TransitionMode, refer to the related information of TransitionMode in *AX-3 Series Operation Manual*.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when absolute positioning is complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bActive	TRUE when the instruction is controlling axes	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

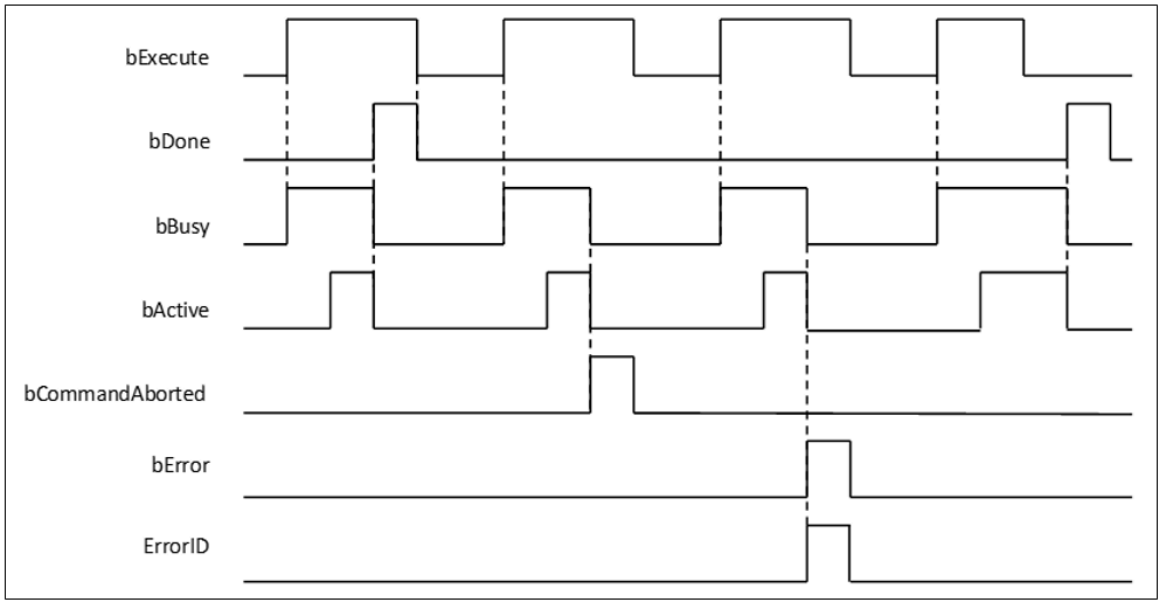
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The absolute positioning is complete.	• *bExecute* turns to FALSE. • *bExecute* is FALSE but *bDone* turns to TRUE, *bDone* will remain TRUE for one cycle and then change to FALSE.
bBusy	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bActive	The axis motion starts.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE • *bExecute* is FALSE but *bActive* turns to TRUE, *bActive* will remain TRUE for one cycle and then change to FALSE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bCommand Aborted	• When the instruction is interrupted by another instruction whose *BufferMode* is set to Aborting • The instruction is interrupted by MC_Stop. • The instruction is interrupted by DMC_GroupStop.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID*.)	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*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 operated independently during the motion, so the motion path will vary according to the configuration.

- 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 action will be performed 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.

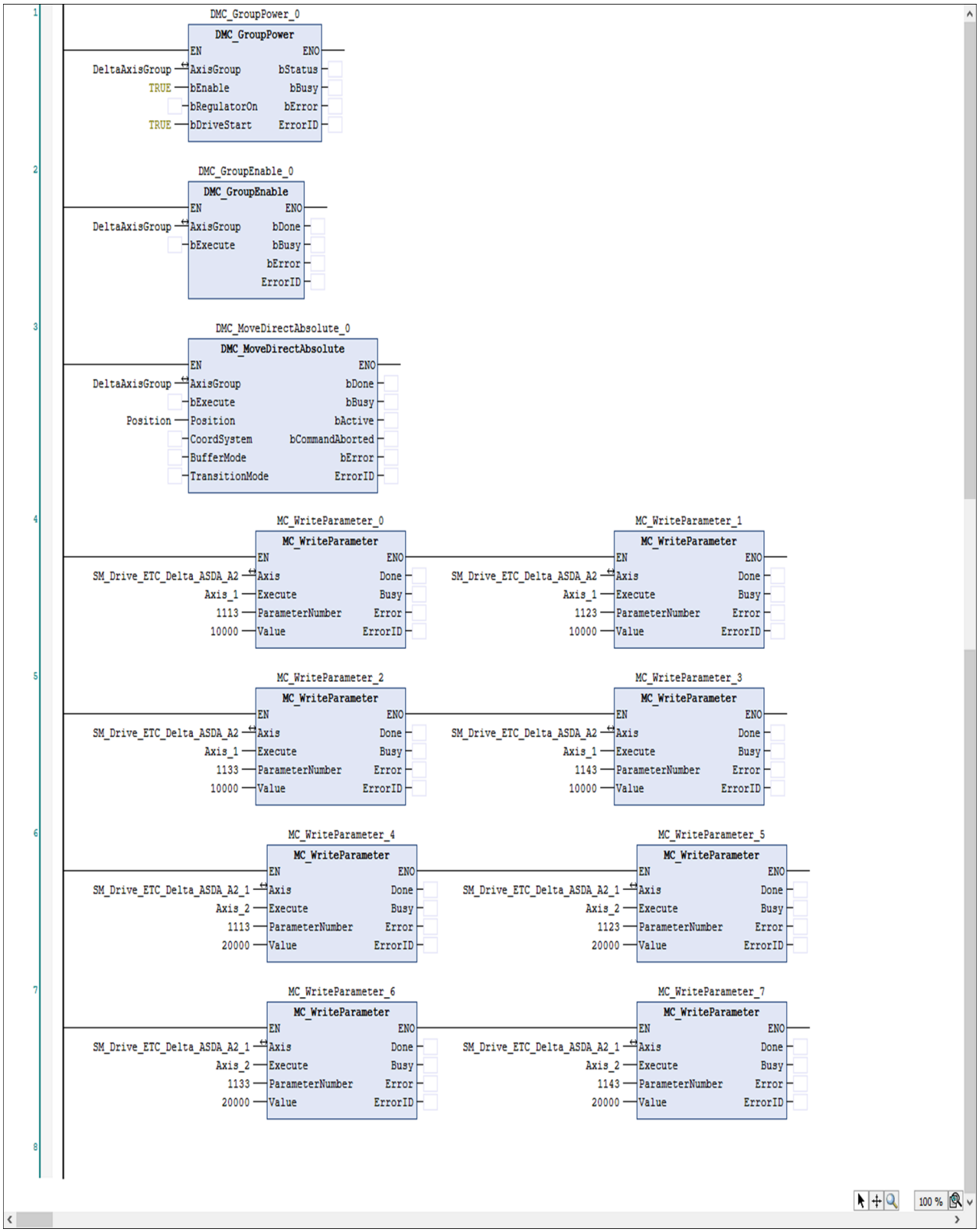
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding troubleshooting methods, refer to **Appendix** of this manual.

- **Example**

This example shows how to use MoveDirectAbsolute to control axis group motion.

Axis Group	Target Position
Axis2	2000
Axis1	1000

```
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;
```



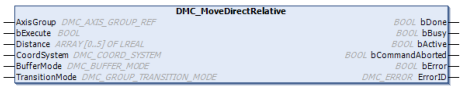
- When *DMC_GroupPower* of bRegulatorOn is TRUE, each axis status switches from Disabled to Standstill.
- When *bExecute* of DMC_GroupEnable is TRUE, the axis group status switches from GroupDisabled to GroupStandby.
- When *Axis_1* and *Axis_2* are TRUE, the parameters will be written to Velocity, Acceleration, 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.

- **Supported Devices**
  - AX-series motion controller

2.2.1.20 DMC_MoveDirectRelative

DMC_MoveDirectRelative controls the axis group moving to the relative position in the coordinate system. Each axis is operated independently during the motion, and the motion path is not specified.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_MoveDirectRelative		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Distance	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 *Distance* will be updated.
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 parameters of *CoordSystem* will be updated.
BufferMode	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 parameters of *BufferMode* will be updated.
TransitionMode	The transition behavior mode for this function block instruction[*3]	DMC_GROUP_TRANSITION_MODE[*3]	0: None 10: Overlap 11: Single_axis (0)	When *bExecute* is on the rising edge, the 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 Operation Manual*.
3. About TransitionMode, refer to the related information of TransitionMode in *AX-3 Series Operation Manual*.

- **Outputs**

Name	Function	Data Type	Setting Value (Default Value)
bDone	TRUE when the relative positioning is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bActive	TRUE when axes start being controlled by the instruction.	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

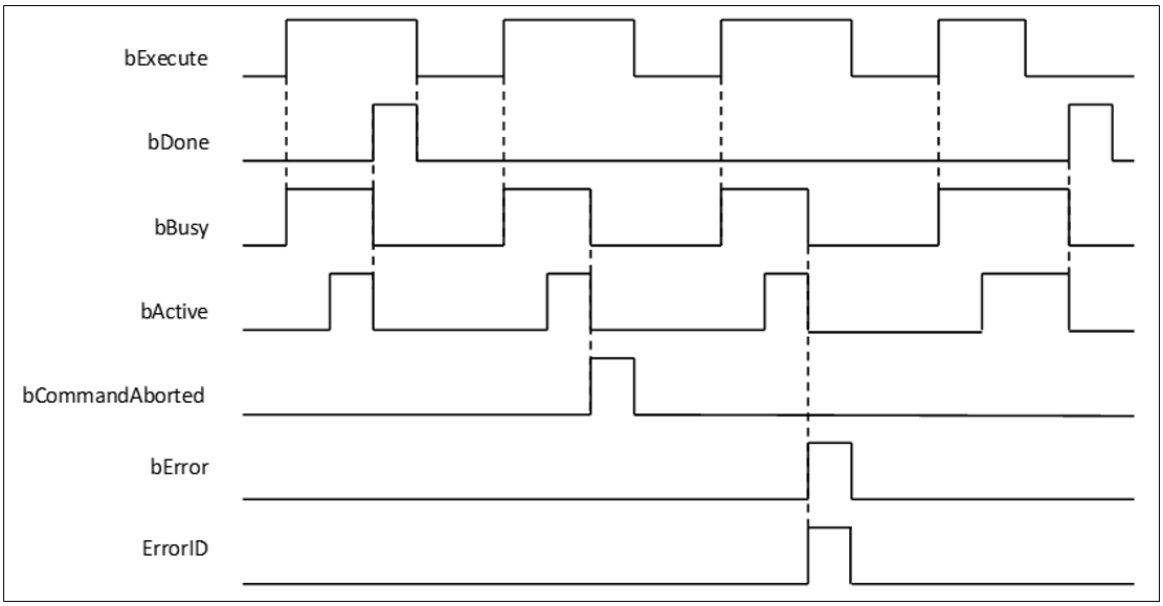
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The relative positioning is completed.	• *bExecute* turns to FALSE. • *bExecute* is FALSE but *bDone* turns to TRUE, *bDone* will remain TRUE for one cycle and then change to FALSE.
bBusy	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bActive	The axis is controlled by the instruction.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE • *bExecute* is FALSE but *bActive* turns to TRUE, *bActive* will remain TRUE for one cycle and then change to FALSE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bCommand Aborted	• The instruction is interrupted by another instruction whose BufferMode is set to Aborting. • The instruction is interrupted by MC_Stop. • The instruction is interrupted by DMC_GroupStop.	• *bEnable* turns to FALSE. • *bEnable* is FALSE but *bCommandAborted* turns to TRUE, *bCommandAborted* will remain TRUE for one cycle and then change to FALSE.
bError ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID*.)	*bEnable* turns to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*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.

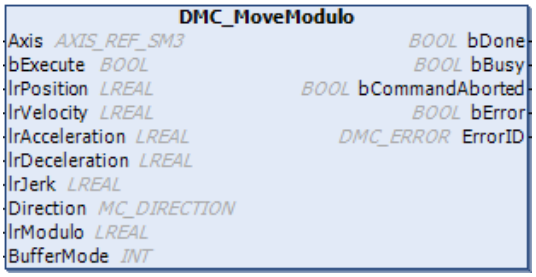
- **Function**
  - This function is available for DL_MotionControl V1.2.0.0 or later.
  - Each axis is operated independently during the motion, so the motion path will vary according to the configuration.

- 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 action will be performed 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 while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and troubleshooting methods, refer to **Appendix** of this manual.
- **Example**
  - Refer to DMC_MoveDirectAbsolute.
- **Supported Devices**:
  - AX-series motion controller

2.2.1.21 DMC_MoveModulo

DMC_MoveModulo is used for modulo positioning and specifies the number of rotation turns.

FB/FC	Instruction	Graphic Expression	ST language
FB	DMC_MoveModulo		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrPosition	Absolute target position (User unit)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrVelocity	Target speed (User unit)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrAcceleration	Acceleration (User unit)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrDeceleration	Deceleration (User unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrJerk	Jerk (User unit/s^3)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
Direction	Motion direction	MC_DIRECTION*	3: fastest 2: current 1: positive 0: shortest -1: negative (shortest)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrModulo	Modulo	LREAL	Positive or 0 (0)	*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 running	BOOL	TRUE/FALSE (FALSE)
bCommand Aborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

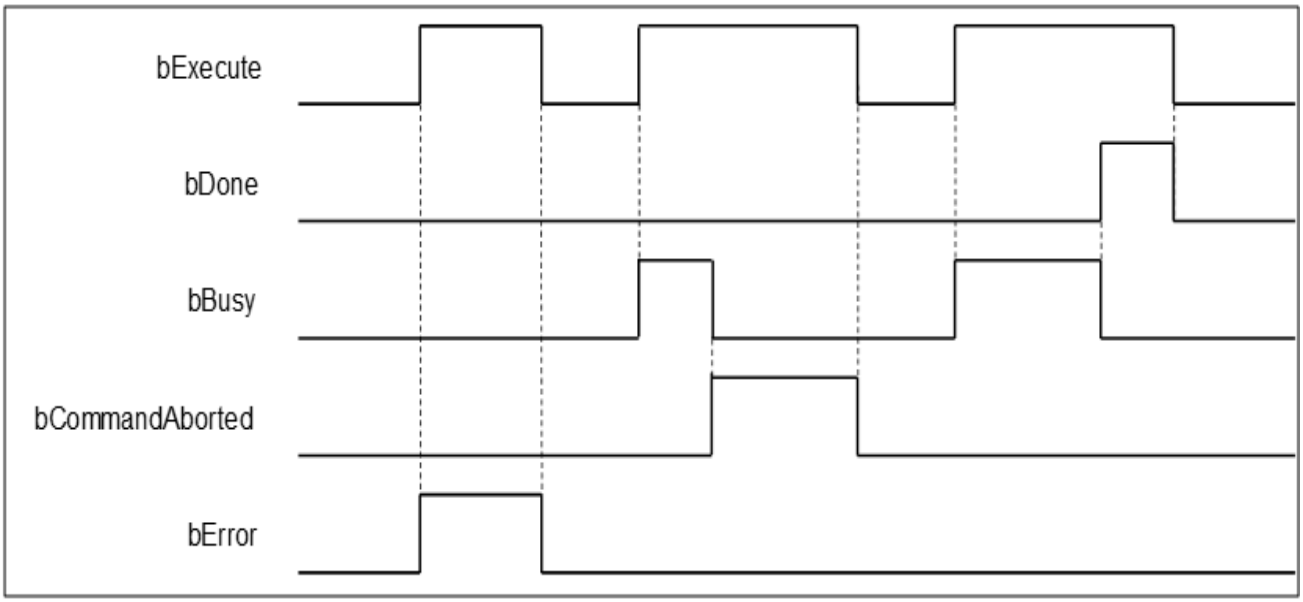
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The motion is completed.	• *bExecute* turns to FALSE. • *bExecute* is FALSE but *bDone* turns to TRUE, *bDone* will remain TRUE for one cycle and then change to FALSE.
bBusy	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bActive	The axis motion starts.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE. • *bExecute* is FALSE but *bActive* turns to TRUE, *bActive* will remain TRUE for one cycle and then change to FALSE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bCommand Aborted	• The instruction is interrupted by another instruction whose BufferMode is set to Aborting • The instruction is interrupted by MC_Stop • The instruction is interrupted by DMC_GroupStop	• *bExecute* turns to FALSE. • *bExecute* is FALSE but *bCommandAborted* turns to TRUE, *bCommandAborted* will remain TRUE for one cycle and then change to FALSE.
bError ErrorID ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded in *ErrorID*.)	*bExecute* turns to FALSE (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	When *bExecute* 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 within modulo (*lrPosition*<*lrModulo*)

      The final positioning is within the modulo.
    - Absolute position outside the modulo (*lrPosition*>*lrModulo*)

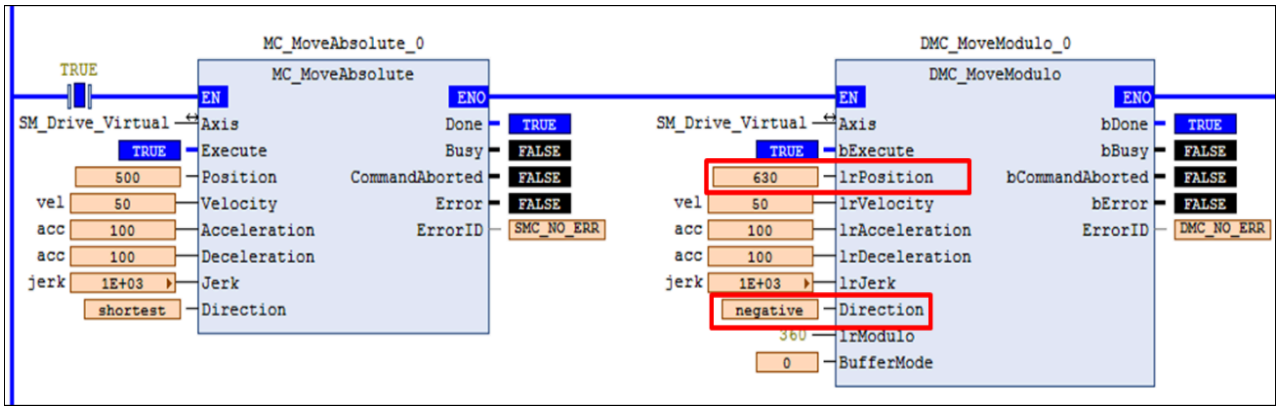
      The final positioning is outside the modulo. When running, it will move n times of the modulus distance first.

- 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 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 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 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 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 by greater than 0.5 modulo: move in reverse to the target position of the next modulo.
    - Absolute position ahead of the current position by smaller than 0.5 modulo: move in reverse to the target position of the modulo.
    - Absolute position behind the current position by greater than 0.5 modulo: move in reverse to the target position of the last modulo.
    - Absolute position behind the current position by smaller than 0.5 modulo: move forward to the target position of the modulo.

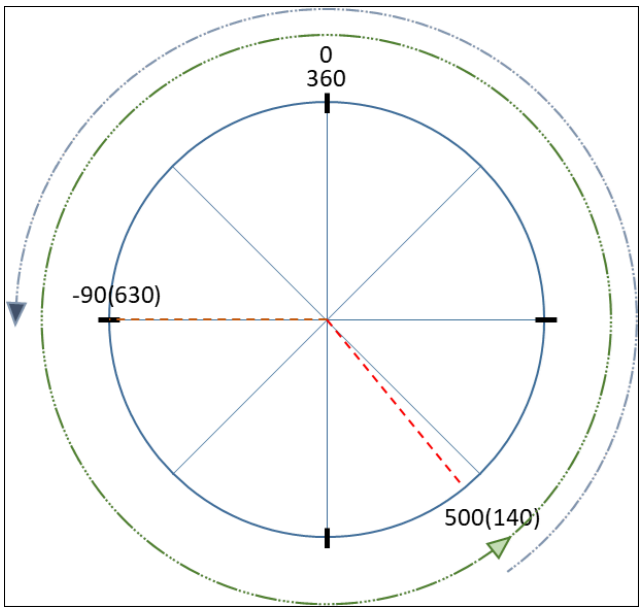
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

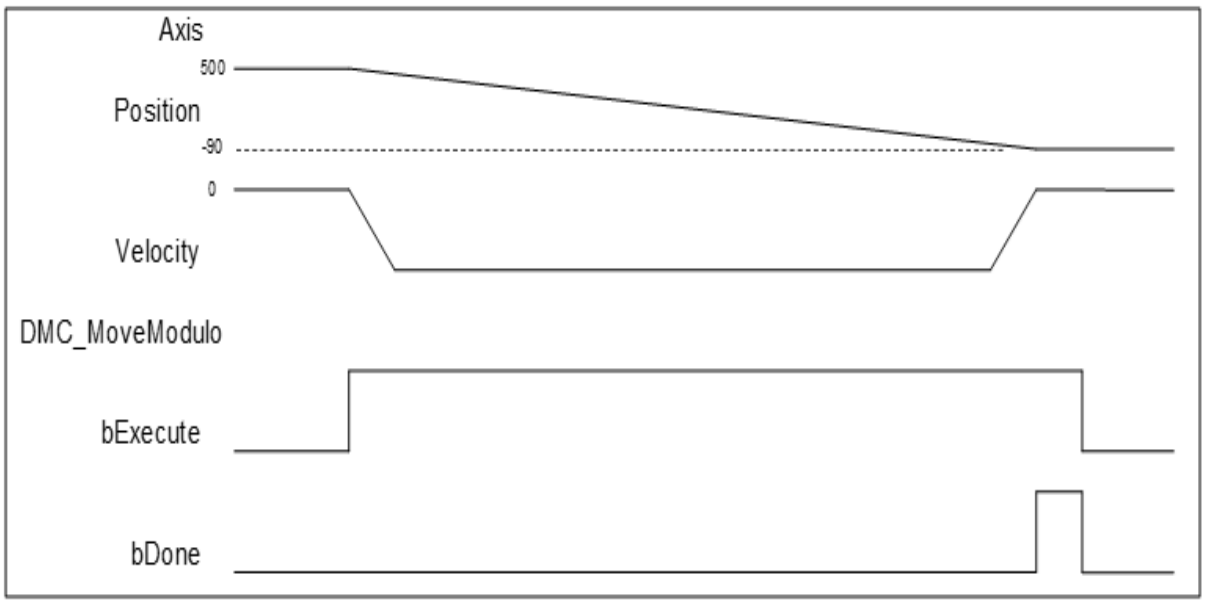
  The following image shows the operation of DMC_MoveModulo is in negative mode when the target position exceeds the modulo and behind the current position.



- Operation Diagram



- Timing Diagram



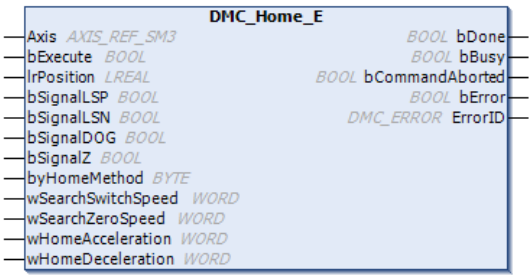
1. The input of absolute position (*lrPosition*) is 630, and the modulo is 360, so the positioning will run more than one turn and then to the relative position in the corresponding module, which is 270.
2. The input of *Direction* is negative, so it can only run backward to the target position.

3. The current position is 500, so 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)

- **Supported Devices**
  - AX-series motion controller

2.2.1.22 DMC_Home_E

DMC_Home_E controls and plans homing.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_Home_E		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrPosition	The position of the axis after the homing is complete	LREAL	Positive, negative, or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
bSignalLSP	Positive limit signal	BOOL	TRUE/FALSE (FALSE)	*bExecute* turns to TRUE and *bBusy* is FALSE.
bSignalLSN	Negative limit signal	BOOL	TRUE/FALSE (FALSE)	*bExecute* turns to TRUE and *bBusy* is FALSE.
bSignalDOG	DOG signal	BOOL	TRUE/FALSE (FALSE)	*bExecute* turns to TRUE and *bBusy* is FALSE.
bSignalZ	Z signal	BOOL	TRUE/FALSE (FALSE)	*bExecute* turns to TRUE and *bBusy* is FALSE.
byHomeMethod*	Homing mode	BYTE	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
wSearchSwitchSpeed	First-phase speed (homing speed when searching for a switch)	WORD	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
wSearchZeroSpeed	Second-phase speed (homing speed when searching for zero)	WORD	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
wHomeAcceleration	Homing acceleration	WORD	Positive (0)	*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 complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction running is interrupted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

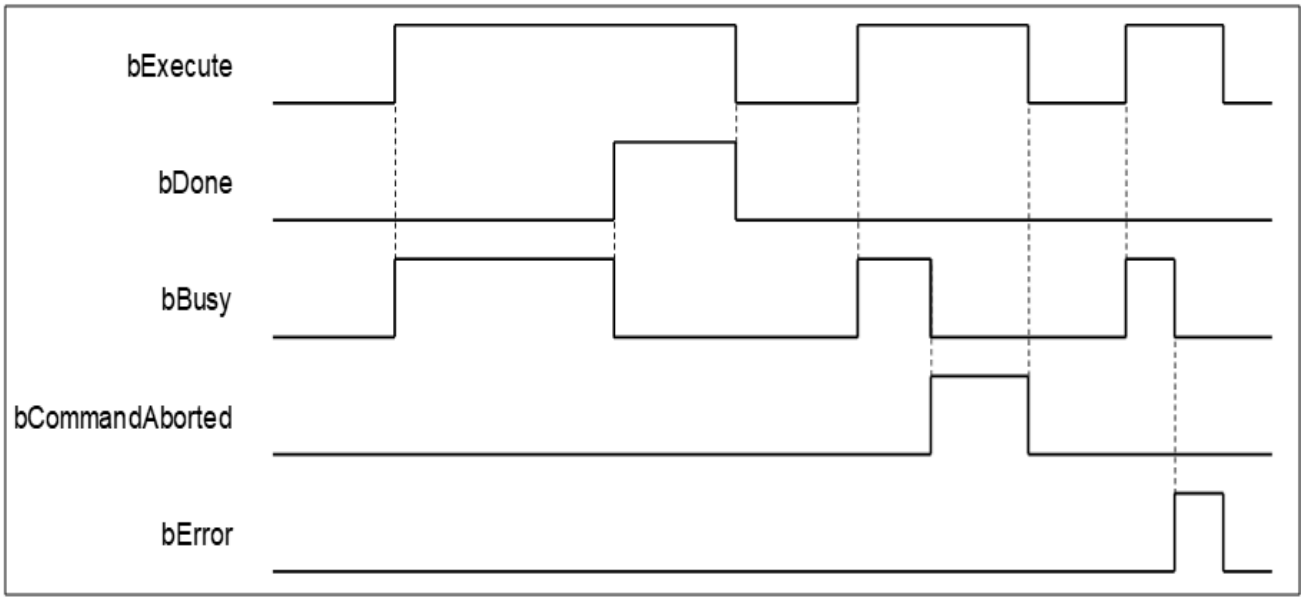
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	*bExecute* turns to TRUE.	• *bExecute* turns to FALSE. • *bError* turns to TRUE.
bCommandAborted	The instruction is interrupted by MC_Stop.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	Homing is complete.	• *bExecute* turns to FALSE. • *bError* turns to TRUE.

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*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 about homing modes, see **Appendix**.
  - *bSignalLSP* (positive limit signal), *bSignalLSN* (negative limit signal), *bSignalDOG* (DOG signal), and *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), and *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**

- If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

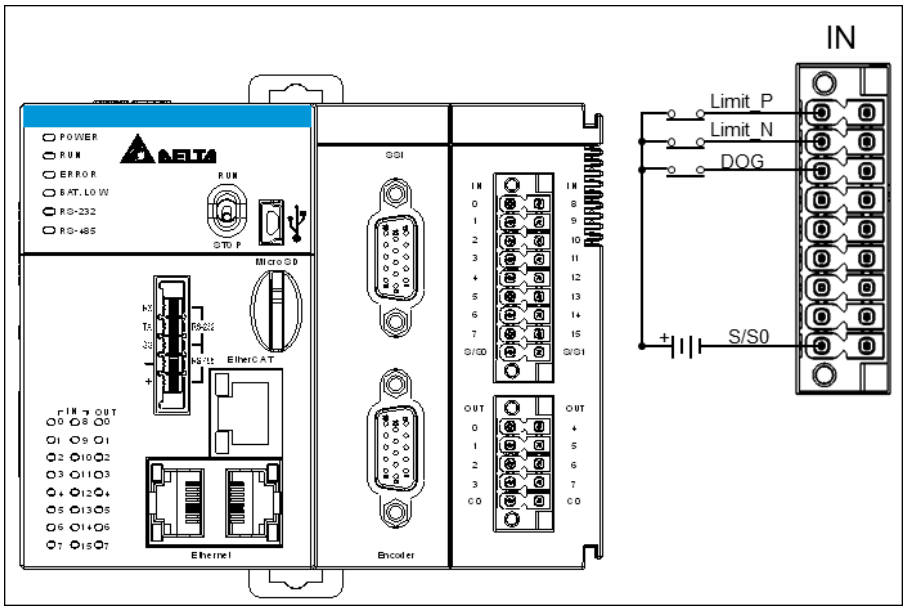
- **Example**

  This example demostrates how to use DMC_Home_E to execute a homing motion planned by the host computer.

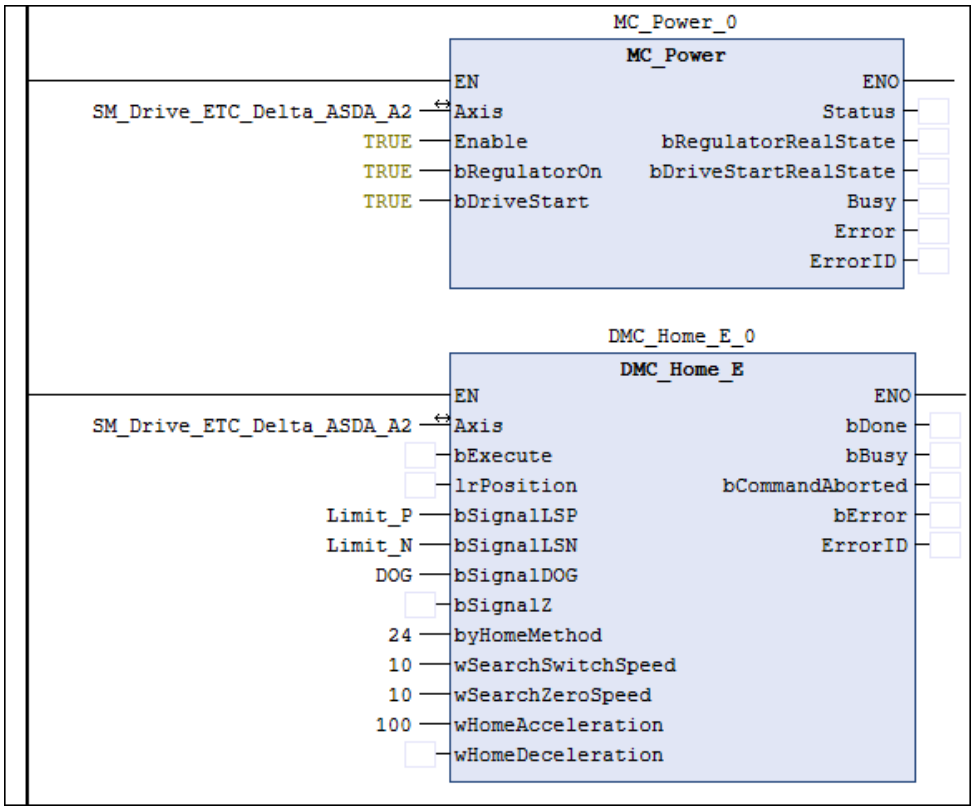
  - The homing signal can be set by DIO and program variables to map to the inputs of DMC_Home_E.

  - Set input parameter (Axis) for MC_Power and DMC_Home_E, as shown in the following figure. When the axis status is standstill, then start this function block, the homing motion will run. The state will switch from Standstill to Homing.

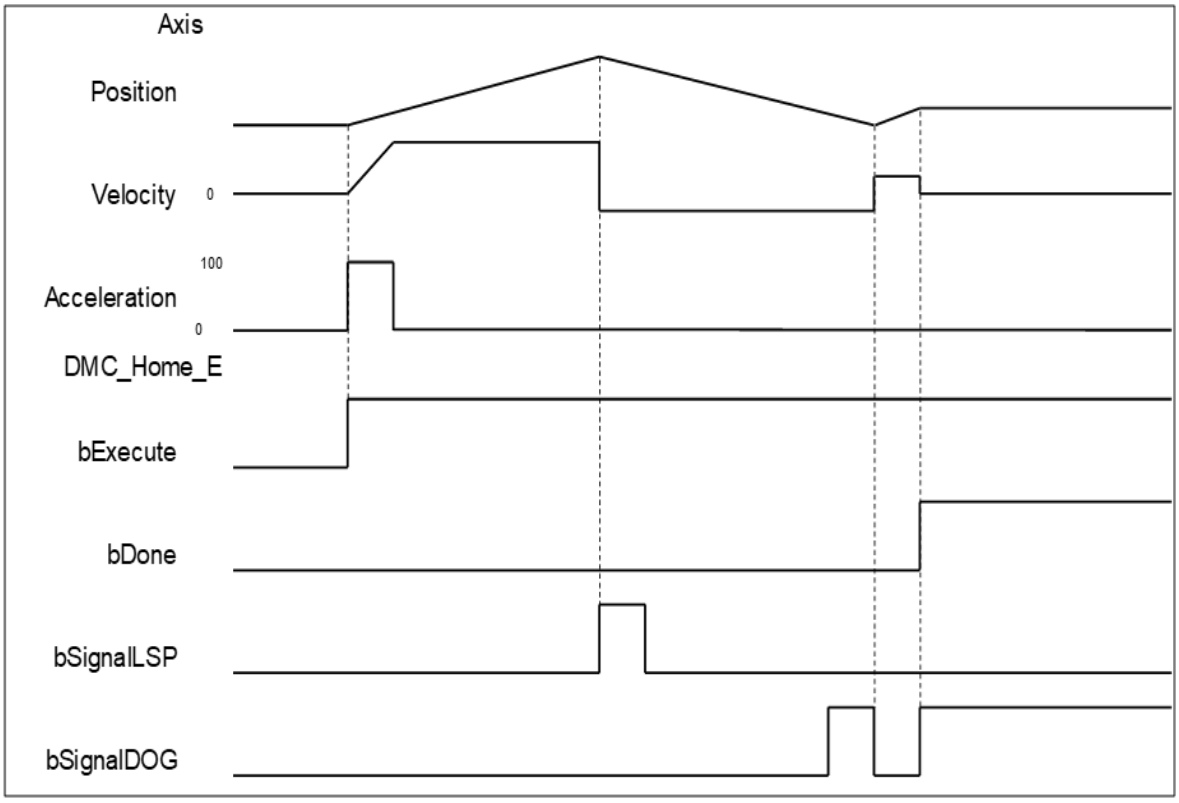
  - DOG Signal Hardware Configuration



  - Homing mode 24 example



- Timing Diagram

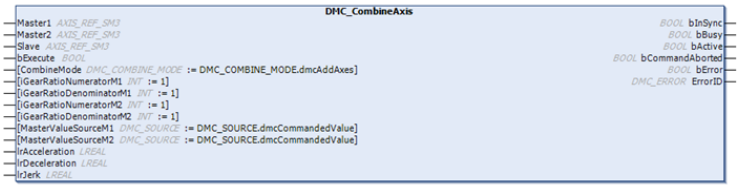


  - Set **Homing Mode** as **Mode 24**.
  - After reaching the positive limit switch, reverse to search the homing signal. When the homing signal is received, it will keep the running direction until there's no homing signal. After there's no homing signals,reverse the rotation until the home signal is TRUE to complete the homing.

- **Supported Devices**
  - AX-series motion controller

2.2.1.23 DMC_CombineAxis

DMC_CombineAxis allows the slave axis to follow the master axis at a fixed velocity ratio.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_CombineAxis		DMC_CombineAxis _instance(Master1:= , Master2:= , Slave:= , bExecute:= , CombineMode:= , iGearRatioNumeratorM1:= , iGearRatioDenominatorM1:= , iGearRatioNumeratorM2:= , iGearRatioDenominatorM2:= , MasterValueSourceM1:= , MasterValueSourceM2:= , lrAcceleration:= , lrDeceleration:= , lrJerk:= , bInSync=> , bBusy=> , bActive=> , bCommandAborted=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
CombineMode	Combination mode	DMC_COMBINE_MODE*1	dmcAddAxes~ dmcSubAxes (dmcAddAxes)	*bExecute* turns to TRUE.
iGearRatioNumerator M1	Gear ratio numerator (master axis 1)	INT	–(2^31)–(2^31 –1) (1)	*bExecute* turns to TRUE.
iGearRatioDenominator M1	Gear ratio denominator (master axis 1)	INT	1–(2^31–1) (1)	*bExecute* turns to TRUE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
iGearRatioNumerator M2	Gear ratio numerator (master axis 2)	INT	–(2^31)–(2^31–1) (1)	*bExecute* turns to TRUE.
iGearRatioDenominator M2	Gear ratio denominator (master axis 2)	INT	1–(2^31–1) (1)	*bExecute* turns to TRUE.
MasterValueSourceM1	Master axis source (master axis 1)	DMC_SOURCE*2	dmcCommandedValue~dmcActualValue (dmcCommandedValue)	*bExecute* turns to TRUE.
MasterValueSourceM2	Master axis source (master axis 2)	DMC_SOURCE*2	dmcCommandedValue~dmcActualValue (dmcCommandedValue)	*bExecute* turns to TRUE.
lrAcceleration	Acceleration	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrDeceleration	Deceleration	LREAL	Positive (0)	*bExecute* turns to TRUE.
lrJerk	Jerk	LREAL	Positive (0)	*bExecute* turns to TRUE.

***Note**:

1. DMC_COMBINE_MODE: Enumeration (Enum)
2. DMC_SOURCE: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Setting Value (Default Value)
bInSync	TRUE when engaging is complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

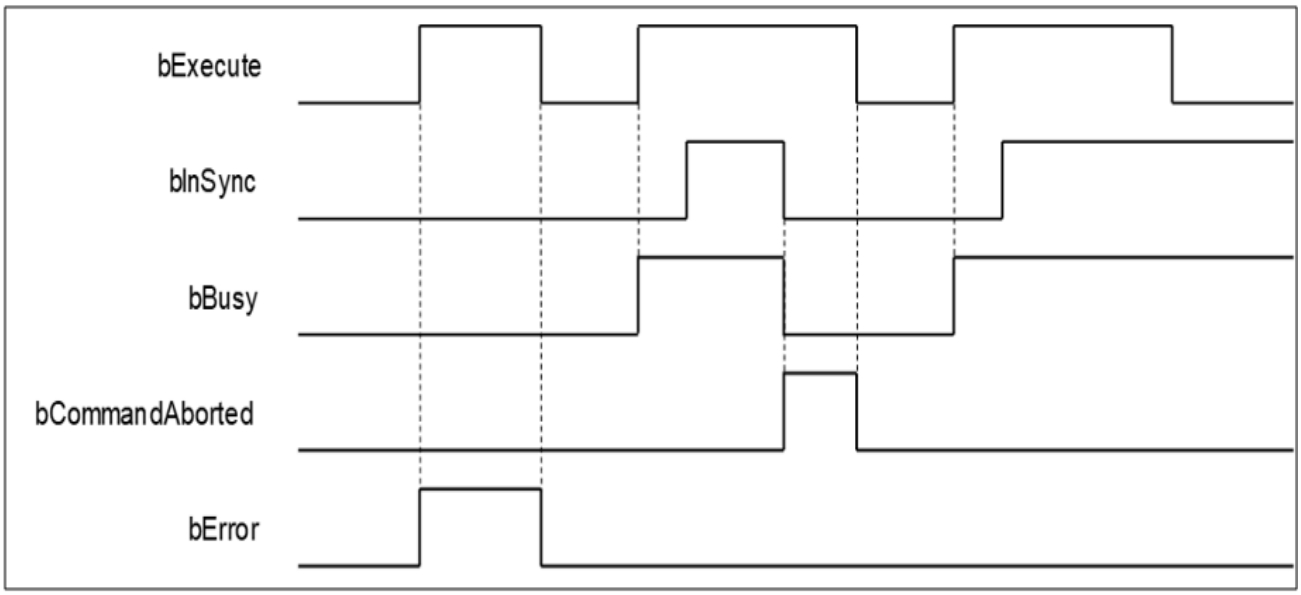
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bInSync	The engaging is complete.	• *bExecute* turns to FALSE. • *bError* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	*bExecute* turns to TRUE.	• *bExecute* turns to FALSE. • *bError* turns to TRUE.
bCommandAborted	The instruction is interrupted by other function blocks.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**

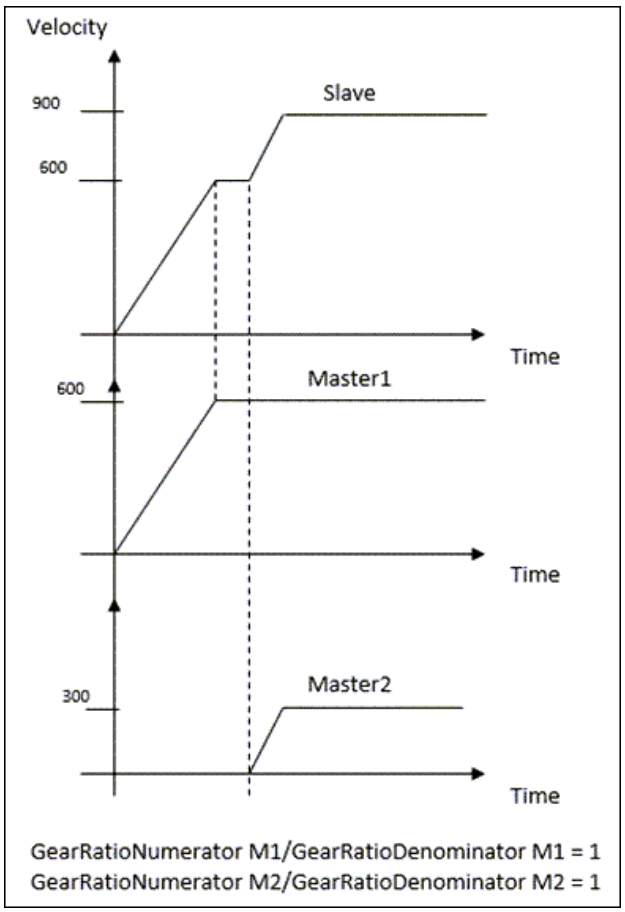


- **Inputs/Outputs**

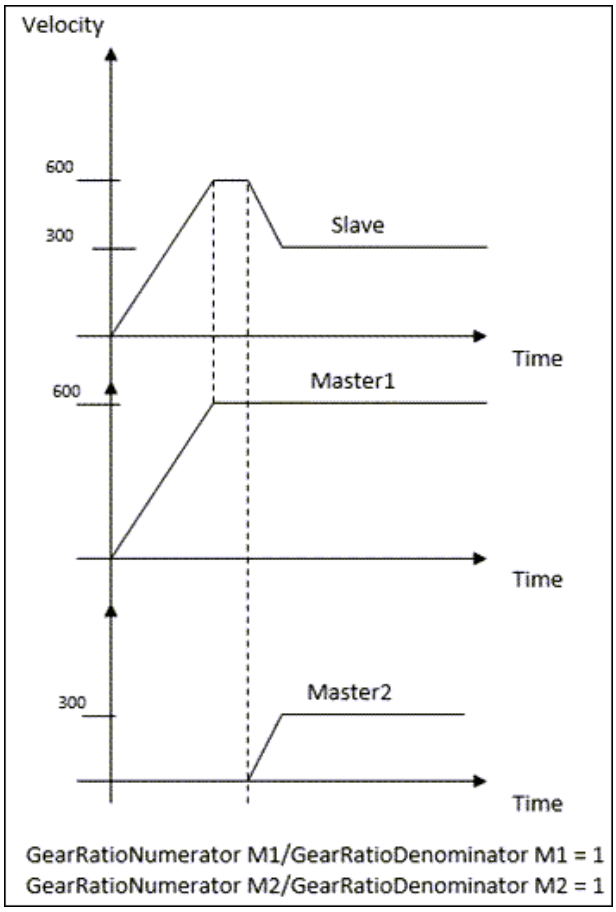
Name	Function	Data Type	Setting Value	Timing to Take Effect
Master1	Master axis 1	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE.
Master2	Master axis 2	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE.
Slave	Slave axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE.

***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 block is only available for DL_MotionControl V1.3.4.0 or later.
  - When the gear relationship is established, the slave axis follows the master axis according to the given velocity ratio to synchronize. The master axis can be a physical axis, a virtual axis, or an external encoder.
  - When the function block is triggered repeatedly, if the input data is invalid, it will run according tot the previous setting. If the data is valid, the new setting will be adopted.
  - *CombineMode*
    - When *CombineMode* is *dmcAddAxes*



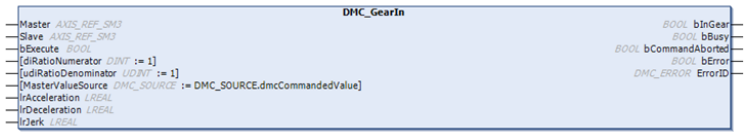
- When *CombineMode* is *dmcSubAxes*



- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Supported Devices**
  - AX-series motion controller

2.2.1.24 DMC_GearIn

DMC_GearIn couples the slave axis to the master axis to move at a fixed velocity ratio.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GearIn		DMC_GearIn_instance(Master:= , Slave:= , bExecute:= , diRatioNumerator:= , udiRatioDenominator:= , MasterValueSource:= , lrAcceleration:= , lrDeceleration:= , lrJerk:= , bInGear=> , bBusy=> , bCommandAborted=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
diRatioNumerator	Gear ratio numerator	DINT	Negative, positive or 0 (1)	*bExecute* turns to TRUE.
udiRatioDenominator	Gear ratio denominator	UDINT	Positive (1)	*bExecute* turns to TRUE.
MasterValueSource	Master axis source	DMC_SOURCE*	0：dmcCommandedValue 1：dmcActualValue (dmcCommandedValue)	*bExecute* turns to TRUE.
Acceleration	Acceleration (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
Deceleration	Deceleration (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.
Jerk	Jerk (Unit: user unit/s^3)	LREAL	Positive (0)	*bExecute* turns to TRUE.

***Note**: DMC_SOURCE: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bInGear	TRUE when the slave and master axes are coupled	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)

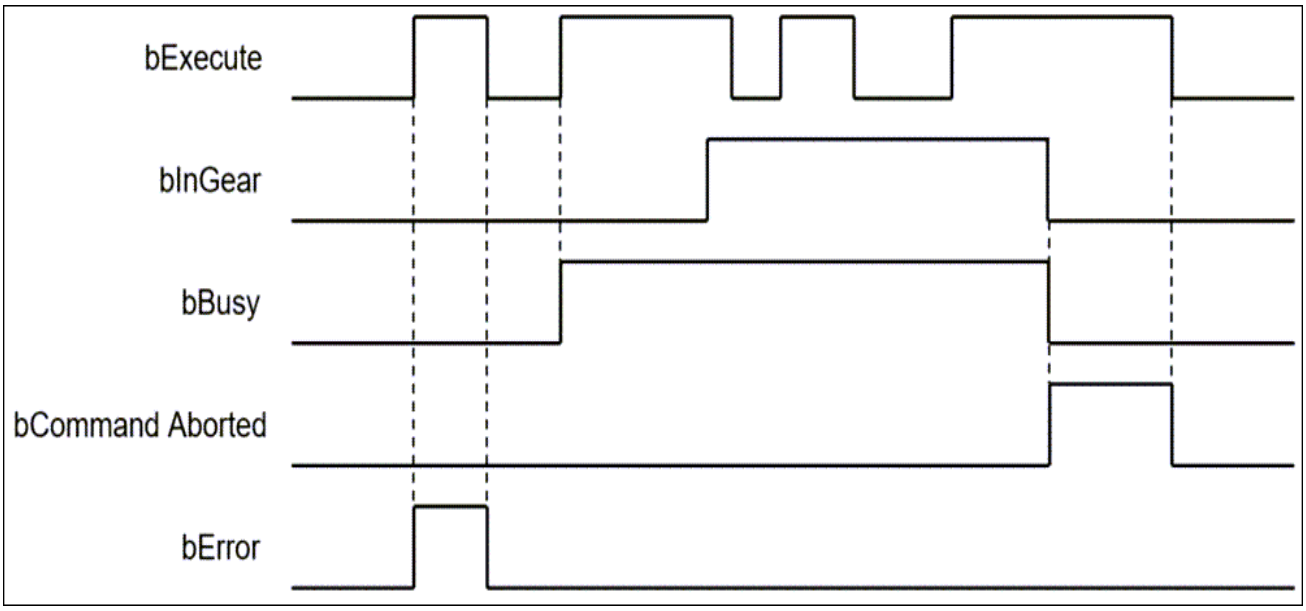
Name	Function	Data Type	Output Range (Default Value)
bCommandAborted	TRUE when the instruction is interrupted	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bInGear	The engaging is complete.	• *bExecute* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bExecute* turns to TRUE.	• *bExecute* turns to FALSE. • *bError* turns to TRUE.
bCommandAborted	The instruction is aborted.	• *bExecute* changes to FALSE • If *bExecute* is FALSE and *bCommandAborted* turns to TRUE, *bCommandAborted* will be TRUE for only one scan cycle and immediately turn to FALSE
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns from TRUE to FALSE. (ErrorID is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**

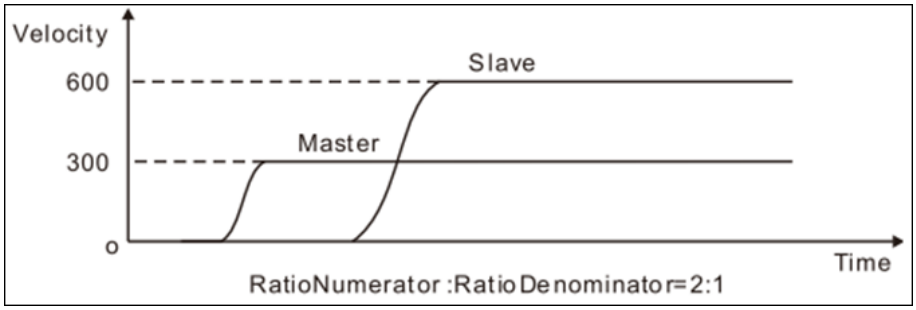


- **Inputs/Outputs**

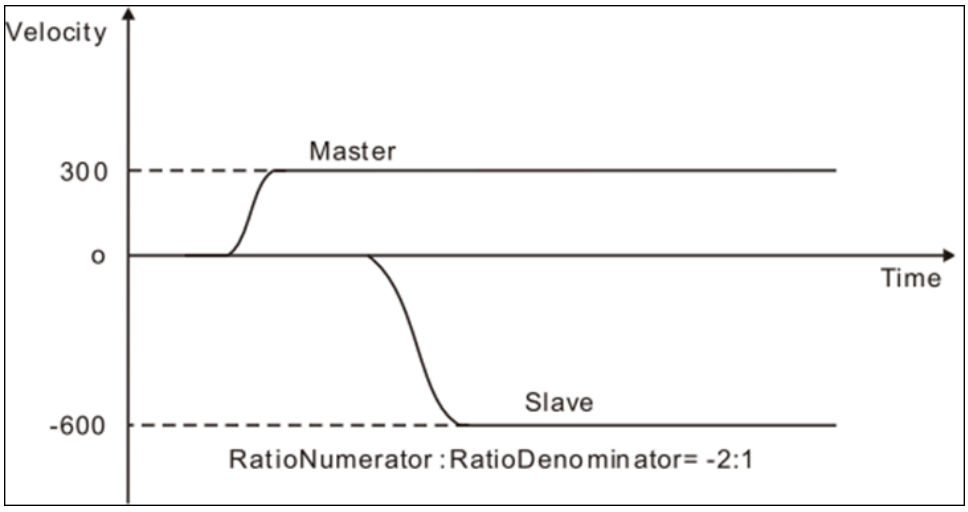
Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	The specified master axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE.
Slave	The specified slave axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE.

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

- **Function**
  - This function block is supported by DL_MotionControl V1.3.4.0 or later.
  - When the function block is triggered repeatedly, if the input data is invalid, it will run according tot the previous setting. If the data is valid, the new setting will be adopted.
  - After the gears are coupled, the slave axis will follow the master axis to move at the given velocity ratio to accomplish the synchronization. The master axis could be physical axis, virtual axis, or external encoder.
  - This function block supports PLC simulation mode in DL_MotionControl V1.4.0.0 or later.
  - RatioNumerator, RatioDenominator
    - When the value of gear ratio is positive, the master and slave axes move towards the same direction.



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



  - Acceleration, Deceleration

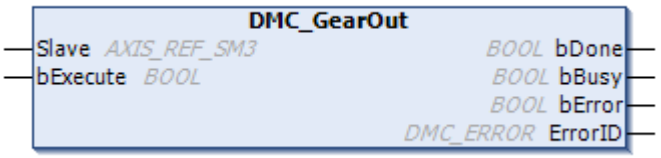
- When running MC_GearIn, the slave axis starts gearing, and if the speed of the slave axis is less than that of the gearing target, the slave axis will accelerate to the gearing target speed according to the given acceleration (Acceleration), and then complete the gearing.
- When running MC_GearIn, the slave axis starts gearing, and if the speed of the slave axis is greater than that of the gearing target, the slave axis will decelerate to the gearing target speed according to the given deceleration (Deceleration), and then complete the gearing.

- Depending on where this function block is placed in the program, the results of the synchronous motion are different. For more information, see *AX-3 Series Operation Manual* section 7.7.2.5 Synchronous Control.

- **Troubleshooting**
  - . If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - Refer to the examples of MC_GearIn.
- **Supported Devices**
  - AX-series motion controller

2.2.1.25 DMC_GearOut

DMC_GearOut disengages the slave axis from the master axis by DMC_GearIn or DMC_CombineAxis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GearOut		DMC_GearOut_instance(Slave:= , bExecute:= , bDone=> , bBusy=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

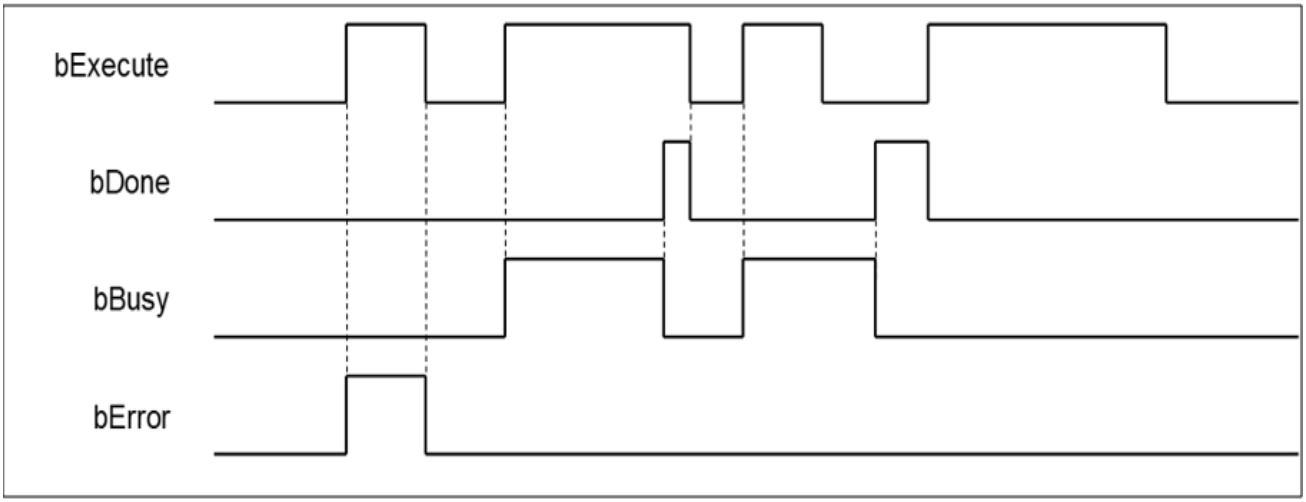
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the gear disengagement is complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The gear disengagement is complete.	• *bExecute* changes to FALSE • *bError* turns to TRUE
bBusy	The instruction is running.	• *bExecute* changes to FALSE • *bError* turns to TRUE
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	*bExecute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



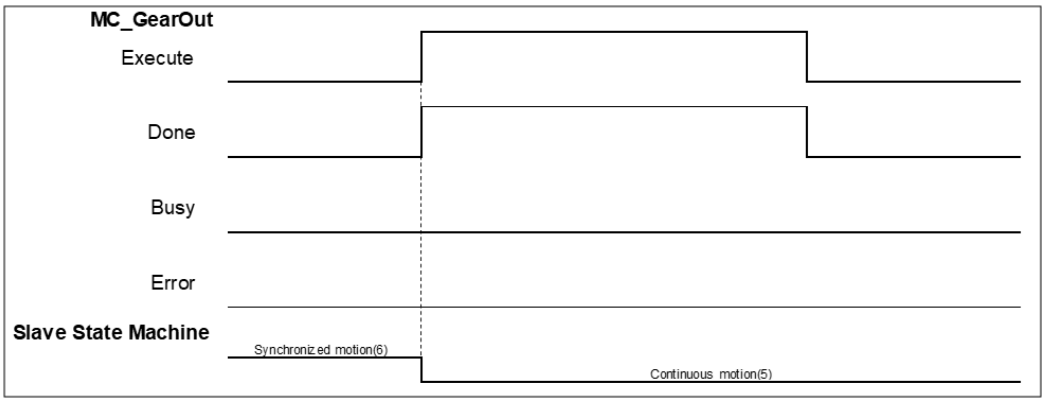
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Slave	The specified slave axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE.

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

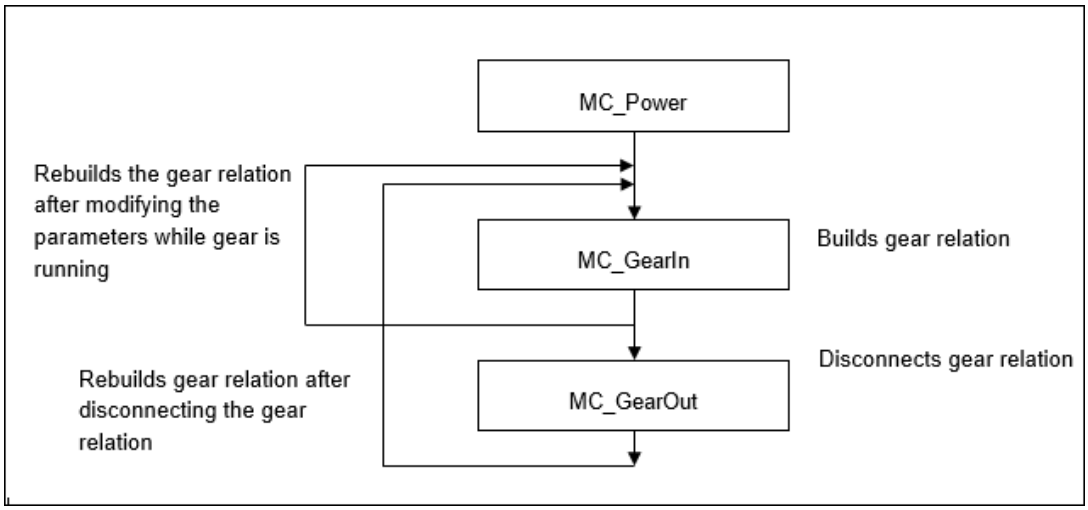
- **Function**

  - This function block is only available for DL_MotionControl V1.3.4.0 or later.
  - This function block supports PLC simulation mode in DL_MotionControl V1.4.0.0 or later.
  - This function block can only be used with DMC_GearIn, DMC_CombineAxis, and DMC_GearInPos.
  - After the gear is disengaged, the slave axis will keep moving at the speed where the gear is disengaged. The axis will be in ContinuousMotion (has nothing to do with the master axis velocity).



  - When the slave axis is out of sync and the velocity is zero, the status will be continuous_motion and remain unchanged.

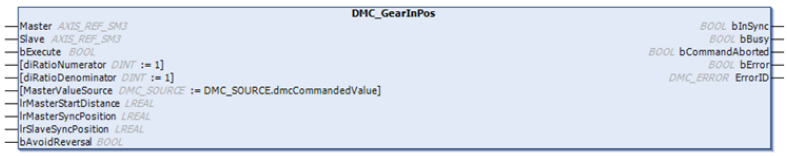
- The sequence for running of the instructions related to electronic gear.



- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - Refer to the examples of MC_GearOut.
- **Supported Devices**
  - AX-series motion controller

2.2.1.26 DMC_GearInPos

DMC_GearInPos couple the slave with the master and slave axis at a specified position.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GearInPos		DMC_GearInPos_instance (Master:= , Slave:= , bExecute:= , diRatioNumerator:= , diRatioDenominator:= , MasterValueSource:= , lrMasterStartDistance:= , lrMasterSyncPosition:= , lrSlaveSyncPosition:= , bAvoidReversal := , bInSync=> , bBusy=> , bCommandAborted=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
diRatioNumerator	Gear ratio numerator between the master and slave axis *1	DINT	Negative, positive or 0 (1)	*bExecute* turns to TRUE and *bBusy* is FALSE.
diRatioDenominator	Gear ratio denominato between the master and slave axis *1	DINT	Negative, positive (1)	*bExecute* turns to TRUE and *bBusy* is FALSE.
MasterValueSource	Master axis source	DMC_SOURCE *2	0: dmcCommandedValue 1: dmcActualValue (dmcCommandedValue)	*bExecute* turns to TRUE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrMasterStartDistance	The distance from the master axis position when the synchronization begins	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrMasterSyncPosition	Master axis synchronization position	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrSlaveSyncPosition	Slave axis synchronization position	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
bAvoidReversal	Reverse is not allowed	BOOL	TRUE/FALSE (FALSE)	*bExecute* turns to TRUE and *bBusy* is FALSE.

***Note**:

1. A negative gear ratio will make the master and slave axes move towards an opposite direction.
2. DMC_SOURCE: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bInSync	TRUE when gearing	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

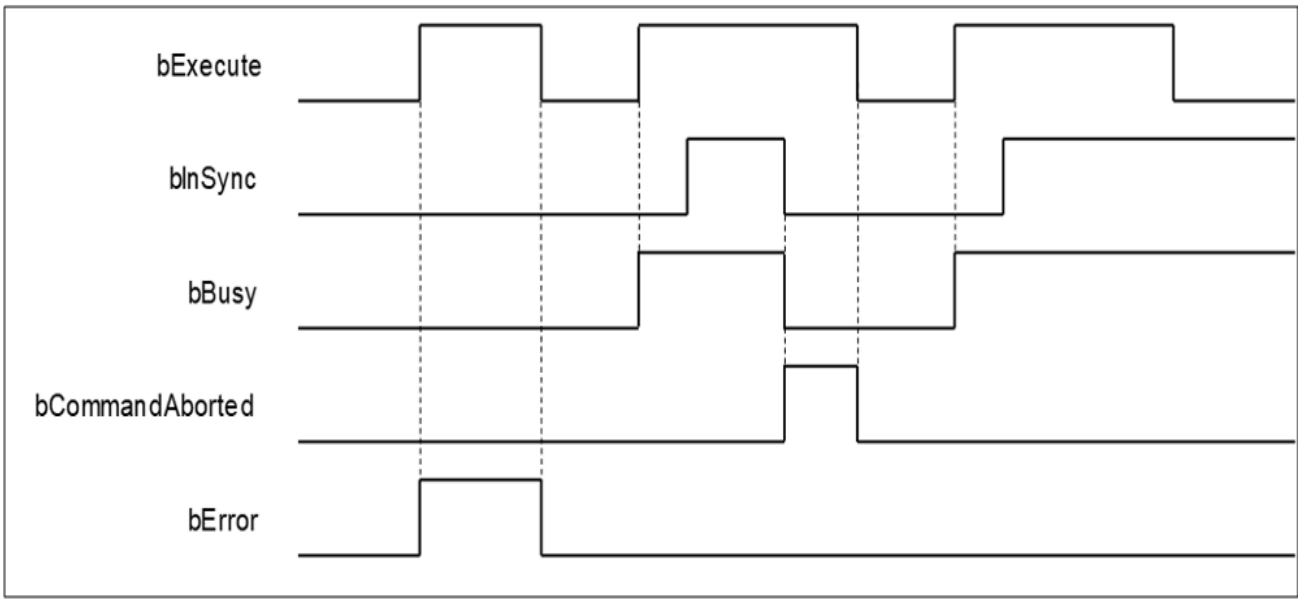
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bInSync	The master and slave axes are synchronized.	• *bCommandAborted* turns to TRUE. • *bError* turns to TRUE.
bBusy	The instruction is running.	• *bCommandAborted* turns to TRUE. • *bError* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bCommandAborted	• DMC_GearOut is run • The instruction is interrupted by another function block.	• *bExecute* changes to FALSE. • If *bExecute* is FALSE and *bCommandAborted* turns to TRUE, *bCommandAborted* will be TRUE for only one scan cycle and immediately turn to FALSE.
bError ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns from TRUE to FALSE (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	The specified master axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE and *bBusy* is FALSE.
Slave	The specified slave axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE and *bBusy* is FALSE.

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

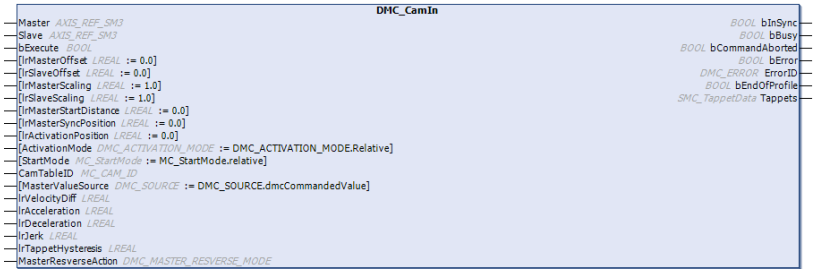
- **Function**
  - This function block is supported by DL_MotionControl V1.3.4.0 or later.
  - When the function block is repeatedly triggered, if the input data is invalid, the function block will continue to run according to the previous setting. If the data is valid, it will run according to the new setting.
  - This function block supports PLC simulation mode in DL_MotionControl V1.4.0.0 or later.
  - You can use DMC_PhasingAbsolute and DMC_PhasingRelative to modify the phase position of the slave axis.

- If the master and slave axes are running in the Finite mode, you need to ensure that the parameters of the synchronization position are set correctly. Assuming the master and slave axes are moving forward, and the master position has exceeded the *StartSync* position when the instruction is running, then the gear will not run correctly. It is recommended that the master and slave axes run in the Modulo mode.
- From the start to the end of synchronization, MC_GearInPos automatically plans the slave axis motion curve based on the following three parameters *StartSync* position, *MasterSyncPosition*, and *SlaveSyncPosition* and the gear ratio. After synchronization, the slave axis follows the master.
- When *lrMasterStartDistance* = 0 or negative, the cam is completed immediately.
- When the master position does not pass the *lrMasterSyncPosition* setpoint, the slave axis will not perform synchronization. If the master position passes the *MasterSyncPosition* setpoint, the slave axis will perform synchronization.
- Depending on where this function block is placed in the program, the results of the synchronous motion are different. For more information, see *AX-3 Series Operation Manual* section 7.7.2.5 Synchronous Control.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - Refer to the example of MC_GearInPos.
- **Supported Devices**
  - AX-series motion controller

2.2.1.27 DMC_CamIn

DMC_CamIn allows the slave axis to follow the master axis based on the specified cam table.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_CamIn		DMC_CamIn_instance (Master:= , Slave:= , bExecute:= , lrMasterOffset:= , lrSlaveOffset:= , lrMasterScaling:= , lrSlaveScaling:= , lrMasterStartDistance:= , lrMasterSyncPosition:= , lrActivationPosition:= , ActivationMode:= , StartMode:= , CamTableID:= , MasterValueSource:= , lrVelocityDiff:= , lrAcceleration:= , lrDeceleration:= , lrJerk:= , TappetHysteresis:= , MasterResverseAction:= , bInSync=> , bBusy=> , bCommandAborted=> , bError=> , ErrorID=> , bEndOfProfile=> , Tappets=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrMasterOffset	Phase offset of the master axis	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrSlaveOffset	Offset of the slave axis	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrMasterScaling	Master axis scaling factor	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrSlaveScaling	Slave axis scaling factor	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrMasterStartDistance	(Reserved)	LREAL	-	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrMasterSyncPosition	(Reserved)	LREAL	-	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrActivationPosition	Specifies the master axis postion when the slave axis performs cam motion.	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
ActivationMode	The absolute or relative relationship between *ActivationPosition* and master axis position.	DMC_ACTIVATION_MODE	0: Relative 1: Absolute 2: PhaseAxis 3: PhaseCAM (Relative)	*bExecute* turns to TRUE and *bBusy* is FALSE.
StartMode	Specifies how the slave axis meshes with the master.	MC_*StartMode*	0: absolute 1: relative 2: ramp_in 3: ramp_in_pos 4: ramp_in_neg (absolute)	*bExecute* turns to TRUE and *bBusy* is FALSE.
CamTableID	Cam table identifier	MC_CAM_ID	MC_CAM_ID[*1]	*bExecute* turns to TRUE and *bBusy* is FALSE.
MasterValueSource	Master axis source	DMC_SOURCE[*2]	0: dmcCommandedValue 1: dmcActualValue (dmcCommandedValue)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrVelocityDiff	Maximum velocity difference when running DMC_CamIn[*3] (Unit: user unit/s)	LREAL	Positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrAcceleration	The acceleration rate when running DMC_CamIn[*3] (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrDeceleration	The deceleration rate when running DMC_CamIn[*3] (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrJerk	The jerk value when running DMC_CamIn[*3] (Unit: user unit/s^3)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
TappetHysteresis	The hysteresis rate of tappet	LREAL	Positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
MasterResverseAction	The master axis is reversed towards the slave axis.	DMC_MASTER_RESVERSE_MODE[*4]	0: KEEP_FOLLWING_MASTER 1: STOP_FOLLOWING_MASTER (KEEP_FOLLWING_MASTER)	*bExecute* turns to TRUE and *bBusy* is FALSE.

***Note**:

1. MC_CAM_ID (Struct): Cam table variables, from output of MC_CAMTableSelect, and input to MC_CamIn.
2. DMC_SOURCE: Enumeration (Enum)
3. This setting is only effective if selecting ramp_in, ramp_in_pos, or ramp_in_neg in *StartMode*.
4. DMC_MASTER_RESVERSE_MODE: Enumeration (Enum)

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when this instruction is interrupted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	Used with function block SMC_GetTappetValue.	SMC_TappetData[*2]	SMC_TappetData
bInSync	TRUE when the master and slave axes are synchronized	BOOL	TRUE/FALSE (FALSE)

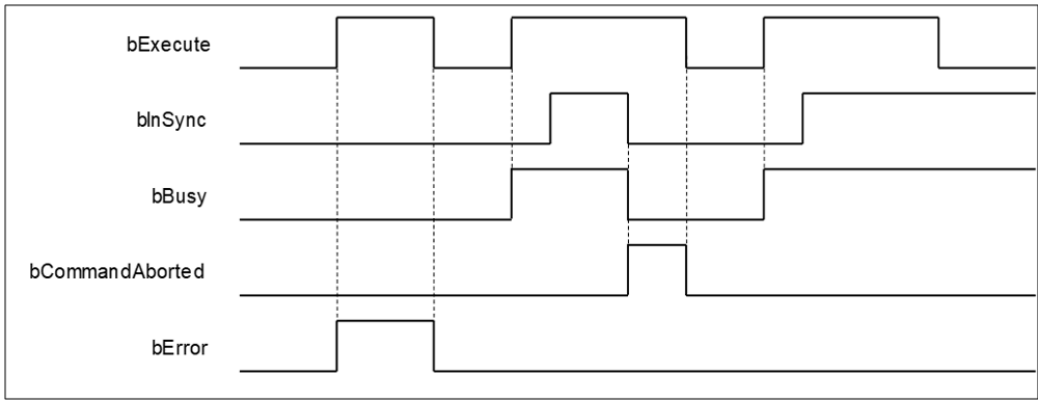
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bInSync	The synchronization is complete.	• *bCommandAborted* is TRUE. • *bError* is TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	The instruction is running.	• *bCommandAborted* is TRUE. • *bError* is TRUE.
bCommandAborted	• DMC_CamOut is run. • The instruction is aborted.	• *bExecute* is FALSE. • If *bExecute* is FALSE and *bCommandAborted* turns to TRUE, *bCommandAborted* will be TRUE for only one period and immediately turn to FALSE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* is FALSE. (Error codes are cleared.)
ErrorID		
EndOfProfile	The end point of the cam profile is complete.	One cycle after *EndOfProfile* turns to TRUE.
Tappets	The cam master axis reaches the tappet position.	The cam master axis leaves the tappet position.

- **Timing Diagram of Output Parameter Changes**



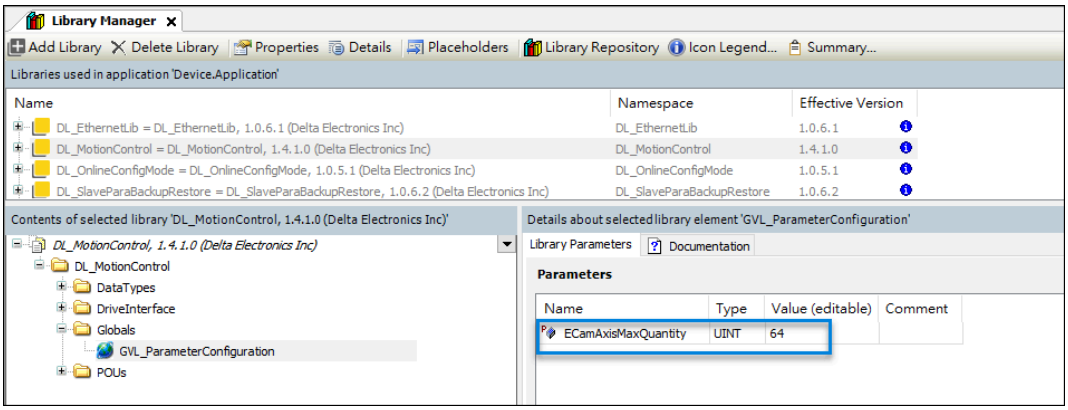
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Slave	The specified slave axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE and *bBusy* is FALSE.
Master	The specified master axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE and *bBusy* is FALSE.

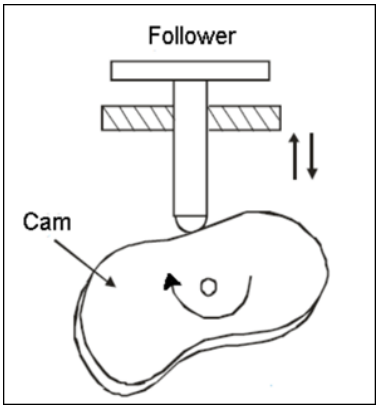
***Note**: AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.

- **Function**
  - This function block is supported by DL_MotionControl V1.3.4.0 or later.
  - When the function block is repeatedly triggered, if the input data is invalid, the function block will continue to run according to the previous setting. If the data is valid, it will run according to the new setting.

- This function block supports PLC simulation mode in DL_MotionControl V1.4.0.0 or later.
- This function block supports the *MasterResverseAction* parameter and virtual axis settings from DL_MotionControl V1.4.1.0.
- It needs to be used with MC_CamTableSelect.
- It can be used with DMC_PhasingAbsolute and DMC_PhasingRelative to modify the phase position of the slave axis.
- When *MasterResverseAction* is set to STOP_FOLLOWING_MASTER, the master and the slave axis is engaged for synchronous motion and when the master axis is reversed, the slave axis stops and disengages from following the master axis. The slave axis is not engaged back to the synchronous motion until the master axis runs forward and passes the position where it is reversed.
- Depending on where this function block is placed in the program, the results of the synchronous motion are different. For more information, see *AX-3 Series Operation Manual* section 7.7.2.5 Synchronous Control.
- When using this function block, one of the axes will be virtual axis. The default number of virtual axes is 64. If the number of the virtual axes is not enough, you can modify it. But if the number is too much, it will affect the performance of the controller.



- E-CAM
  - The traditional mechanical cam mechanism consists of a cam, a follower, and a frame.
    1. The mechanical cam is a machine part with an irregular shape. Generally, it is an input part that transmits motion to the follower at an equal speed, allowing the follower to move according to the set rule through direct contact..
    2. The follower is driven by the mechanical cam and is generally an output part that produces unequal-speed, discontinuous, and irregular motion.
    3. The frame is a mechanical part used to support the mechanical cam and follower.

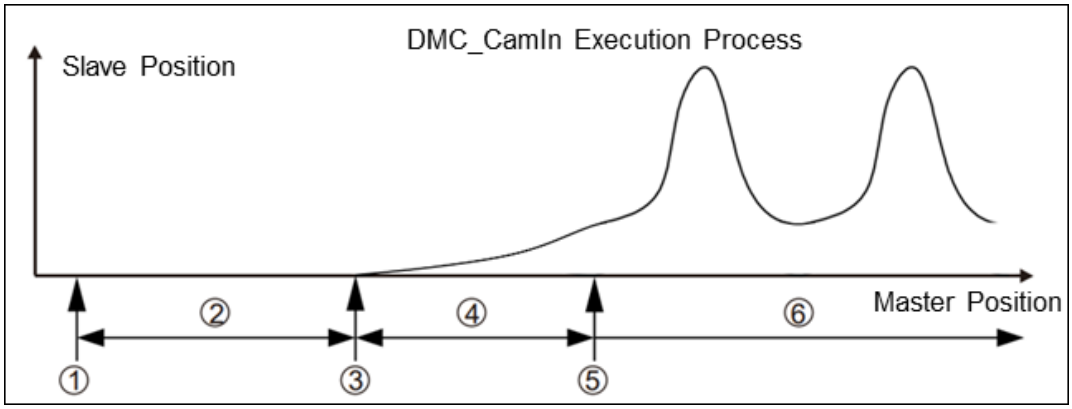


- The electronic cam simulates the mechanical cam through the computer. Compared to the traditional cam, the electronic cam has the following advantages.
  1. User-friendly interface
  2. When different products require different cam curves, software can be used to modify the electronic cam data in the electronic cam table without modifying the mechanism.
  3. Higher acceleration
  4. Smoother operation
  5. Once the cam curve is edited, it needs to be called in the motion control program. The motion control program can call the cam curve by the DMC_CamIn instruction.

◦ DMC_CamIn instruction overview

- DMC_CamIn running process

  The running process of DMC_CamIn is shown in the following figure.



1. Trigger DMC_CamIn to run

   DMC_CamIn starts running, and the slave axis will immediately enter the phase 2.

   Note: If the slave axis is moving, it will stop immediately, which may cause jitter. The input parameters of the DMC_CamIn instruction will be read and locked for use during the running of the instruction.

2. Wait for gearing

   The slave does not move and wait to gear. It waits for the master axis to pass the position specified by the parameter *IrActivationPosition*. The waiting time of the slave axis will vary under different conditions. If the master axis is in the position specified by the parameter *IrActivationPosition* when DMC_CamIn starts running, then the slave axis will immediately begin to gearing. If the master axis never has a chance to reach the position specified by the parameter *IrActivationPosition*, the slave axis will never be able to start gearing, and cam synchronization will never be possible. Parameters *IrActivationPosition* and *IrActivationMode* are used here.

3. The master axis reaches the gearing start position, and the slave axis starts gearing.

   When the master axis passes through the position specified by the parameter *IrActivationPosition*, the slave axis starts gearing. The parameters *IrMasterOffset*, *IrSlaveOffset*, *IrMasterScaling*, and *IrSlaveScaling* will come into play at this point to determine the relationship between the position of the master axis and the slave axis as well as its cam phase.

4. Gearing in progress

The slave axis performs gearing as specified by the parameter *StartMode*. In addition to *StartMode*, the parameters *lrVelocity*, *lrAcceleration*, *lrDeceleration*, *lrJerk*, and *lrMasterSyncPosition* are also used to determine the relative position of the master axis and the speed, acceleration, and deceleration of the slave axis.

5. Gearing is completed, and the master and slave axes are synchronized.

   When the slave axis starts gearing, if the cam phase aligns with the planned cam mechanism, the gearing completes, and the master and slave axes are synchronized with the cam.

   Note: The above figure only represents the master axis position, which is greater than that when the DMC_CamIn instruction starts running. The same can be deduced for the case of equal to and less than.

6. Master and slave axes are acting synchronously.

- Relationship between the master axis position and slave axis position

  - The cam mechanism 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 and slave axis instead of the actual axis position. If the cam mechanism planned is seen as the function CAM, 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.

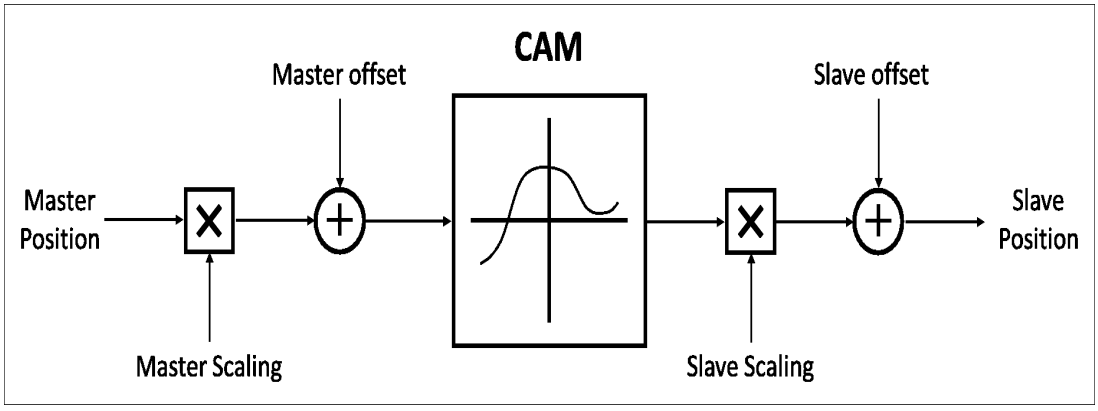
    y = CAM(x)

    x: The master axis cam phase

    y: The slave axis cam phase

  - The cam phase comes from the axis position and there is a conversion, which is related to parameter *MasterAbsolute*, *SlaveAbsolute*, *MasterCompensation*, *SlaveCompensation*, *MasterScaling*, and *SlaveScaling*.

  - The slave axis follows the master axis to perform synchronous cam motion by using the MC_CamIn instruction. In the synchronous cam motion, the relationship between the master axis position and slave axis position is based on the planned cam mechanism (the cam curve or cam table). The process of calculating the slave axis position through the master axis position is illustrated as follows.



  - The following formula is generated from the figure above.

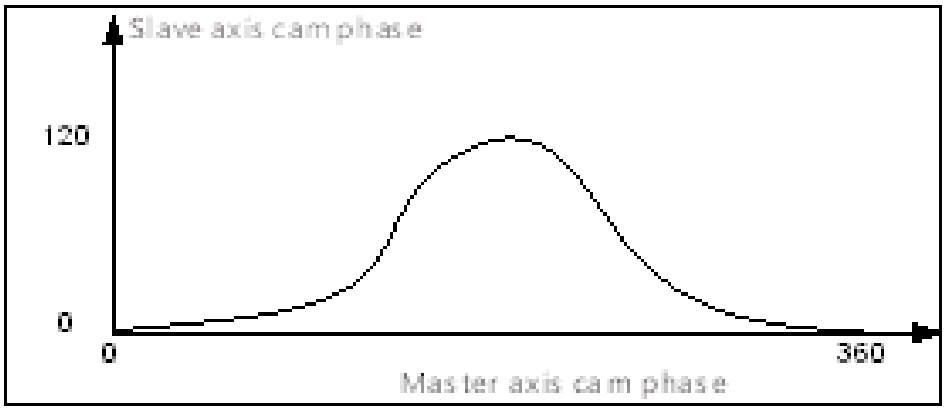
    Position_Slave = *SlaveScaling*×CAM (*MasterScaling*×*MasterPosition*+*MasterCompensation*) + *SlaveCompensation*

- When the master axis is in absolute mode, the master position is the remainder of the current master position divided by modulo. When the master axis is in relative mode, the master position is the start point position (usually 0) of master axis of the corresponding cam curve.

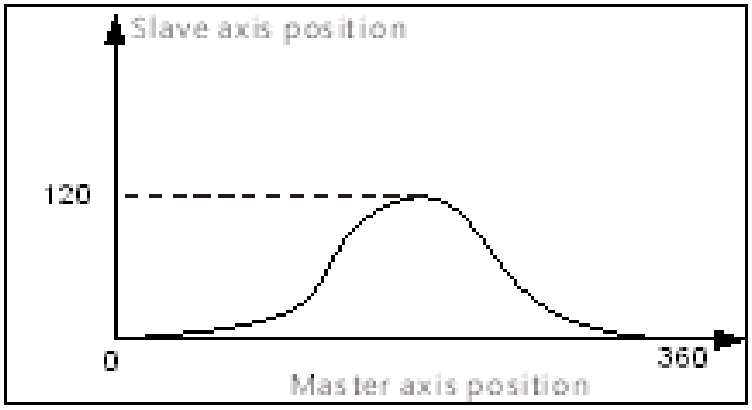
- Offset and scaling (*IrMasterOffset*/*IrMasterScaling*/*IrSlaveOffset*/*IrSlavescaling*)

  - The cam mechanism of the master and slave axis is pre-planned, but when running the cam, the position offset or scaling can be carried out on the pre-planned cam mechanism through the parameters *Offset* and *Scaling*. For example, to process a product with several different sizes, only need to plan one cam mechanism, and then modify the parameter *Offset* and *Scaling* to accommodate different size products. You can specify the scaling factor to scale the master axis phase and slave axis offset of the cam table. The master and slave axis can set the offset and scaling factor respectively.

  - The position offset and scaling of the master and slave axis together determine the cam mechanism that is actually performed, which is illustrated in the following example.

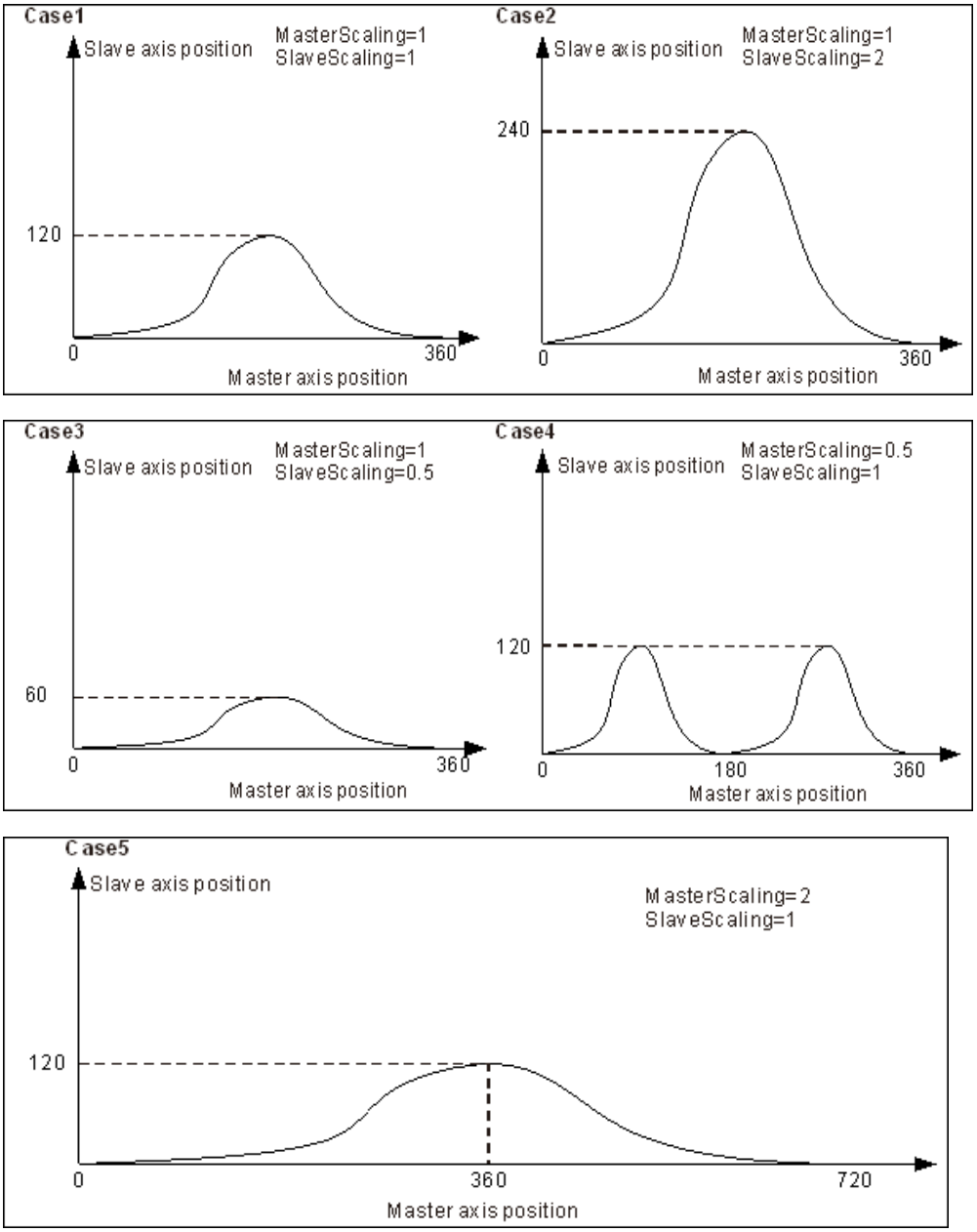


  - If the master and slave axes are in absolute mode, and engaging is performed, the position of the master and slave axes is 0, and when offset and scaling are not used (default value), the actual position correspondence between the master and slave axis during cam running is shown in the following figure.



  - When the position offset or scaling is not the default value, its effect on the actual position of the master and slave axis during cam running is as follows:

1. When the master-slave axis offset is 0, the impact of the master and slave axis scaling on the actual running cam mechanism



Case 1: When the master and slave axis scaling is 1 and the offset is 0, the actual cam mechanism is consistent with the planned.

Case 2: When the master axis scaling is 1, the slave axis scaling is 2, and the master-slave axis offset is 0, the slave axis position corresponding to the master axis position becomes twice the planned one.

Case 3: When the master axis scaling is 1, the slave axis scaling is 0.5, and the master-slave axis offset is 0, the slave axis position corresponding to the master axis position becomes half of the pre-planned one.

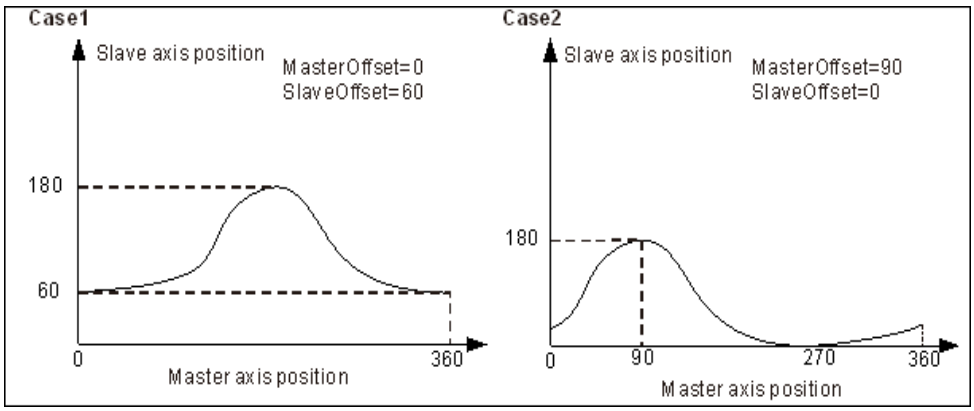
Case 4: When the master axis scaling is 2, the slave axis scaling is 1, and the master-slave axis offset is 0, the master axis position corresponding to the slave axis position becomes twice the planned one. From the cam phase point of view, the cam phase of the master axis is twice as planned, that is, the master axis cam cycle changes from 360 to 180, and the slave axis cam phase does not change.

Case 5: When the master axis scaling is 0.5, the slave axis scaling is 1, and the master-slave axis offset is 0, the master axis position corresponding to the slave axis position becomes half of the planned one. From the cam phase point of view, the cam phase of the master axis is half of the planned, that is, the master axis

cam cycle changes from 360 to 720, and the slave axis cam phase does not change.

2. When the master and slave axis scaling is 1, the effect of the master-slave axis offset on the cam mechanism actually performed.

   The master axis offset is to perform horizontal motion of the actual axis position curve when the cam is running; The slave axis offset is to perform longitudinal motion of the actual axis position curve when the cam is running.



   Case 1: When the master and slave axis scaling is 1, the master axis offset is 0, and the slave axis offset is 60, the slave axis position corresponding to the master axis position adds 60 based on the planned. For example, in the planned cam mechanism, the master axis position 180 is corresponding to the slave axis position 180, but when actually running, the corresponding slave axis position is 240 (240=180+60).

   Case 2: When the master and slave axis scaling is 1, the master axis offset is 90, and the slave axis offset is 0, the master axis position corresponding to the slave axis position adds 90 based on the planned. For example, in the planned cam mechanism, the master axis position 180 is corresponding to the slave axis position 180, but when actually running, the master axis position 90 is corresponding to the slave axis position 180 (180=90+90).

- *ActivationPosition* and *ActivationMode*

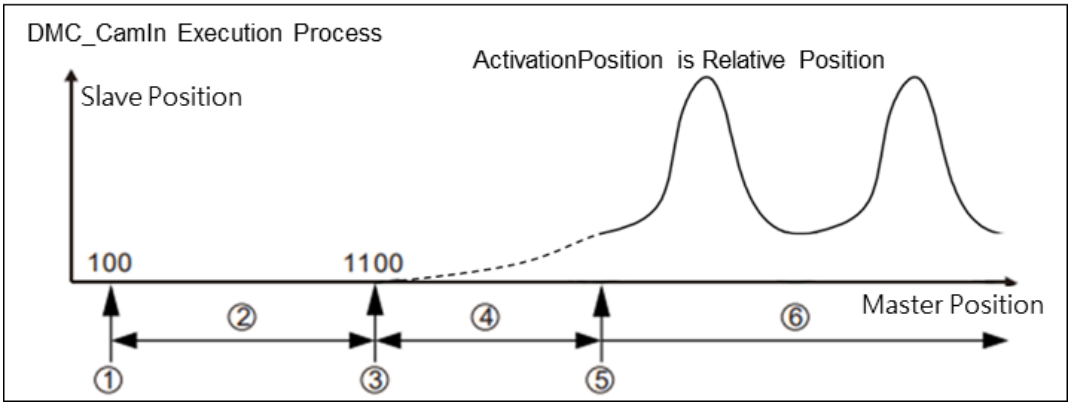
  - *ActivationPosition* and the master axis position when the instruction is triggered determine the position of the master axis when it engages with the slave axis. When the master axis reaches the position (the master axis position in the cam curve coordinate system), the master and slave axis of the cam are synchronized.

  - When the DMC_CamIn instruction is running, *ActivationMode* determines the relative/absolute relationship between *ActivationPosition* and the master axis position when the instruction is triggered.

  - *ActivationMode* = Relative; *ActivationMode* is relative position

    ActivationPosition is relative to the master axis position (Po) when the instruction is triggered. The master axis position (Ps) when the master axis engages with the slave axis is calculated as Ps = Po + *ActivationPosition*

    For example, if the master axis position is 100 when the DMC_CamIn instruction is running and *ActivationPosition* is 1000, the master axis position when engaging starts is 1100 (1100 = 100+1000).

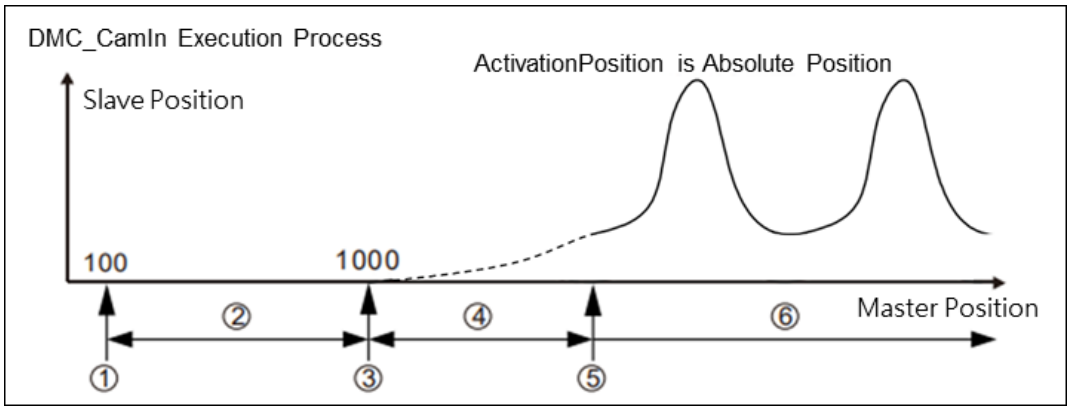


1. Trigger DMC_CamIn to run. The absolute position of the master axis is 100 at this point.
2. Wait for engaging.
3. The master axis reaches the engaging start position (1100), and the slave axis starts engaging.
4. Engaging in progress
5. Engaging completed and the master and slave axes are synchronized.
6. Master and slave axes are acting synchronously.

- *ActivationMode* = Absolute; *ActivationMode* is absolute position

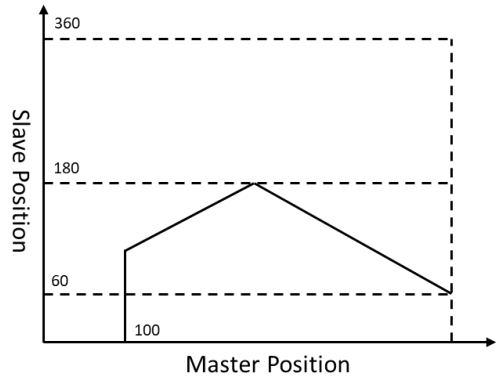
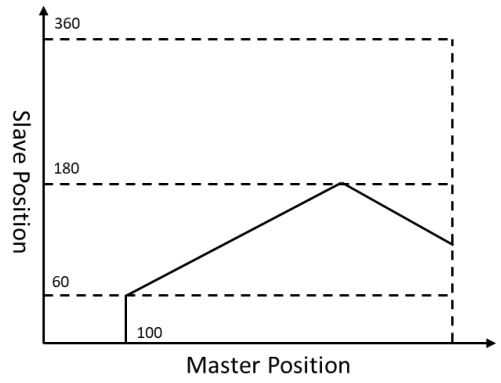
  The relationship between *ActivationPosition* and the master axis position (Po) when the instruction is triggered is absolute. The master axis position (Ps) when the master axis engages with the slave axis is calculated as Ps = *ActivationPosition*.

  For example, if the master axis position is 100 when the DMC_CamIn instruction is running and *ActivationPosition* is 1000, the master axis position when engaging starts is 1000 (1000 = *ActivationPosition*).



1. Trigger DMC_CamIn to run. The absolute position of the master axis is 100 at this point.
2. Wait for engaging.
3. The master axis reaches the engaging start position (1000), and the slave axis starts engaging.
4. Engaging in progress.
5. Engaging completed and the master and slave axes are synchronized.
6. Master and slave axes are acting synchronously.

- The relative and absolute relationship (*MC_CamTableSelect.MasterAbsolute*) between *ActivationPosition* and master axis

	*CamTableSelect.MasterAbsolute* = absolute *ActivationMode* = Absolute *ActivationPosition* = 100 When the master axis is in absolute mode, and when it runs to the position 100 specified in *ActivationMode*, the slave axis will jump directly to the position corresponding to the master axis on the cam table to engage.
	*CamTableSelect.MasterAbsolute* = relative *ActivationMode* = Absolute *ActivationPosition* = 100 When the master axis is in relative mode, and when it runs to the position 100 specified in *ActivationMode*, the slave axis will start engaging from the starting position of the cam table.

- *ActivationMode* = PhaseAxis; *ActivationMode* is absolute axis phase.

  ActivationPosition and the master axis position when the instruction is triggered are the results of the absolute position being calculated to take the remainder according to the modulo.

  The absolute axis phase is periodic. During the master axis is running, its absolute axis may be equal to *ActivationPosition* for many times. The master axis will have both forward and reverse target position when the master axis and the slave axis gears. The relationship between the current position of the master axis and the *ActivationPosition* will also affect the number of mode counts (ModCnt) and the target position of the master axis gear.

  Ps = *ActivationPosition* + [(Current Position / Axis modulo) + ModCnt] x Axis modulo

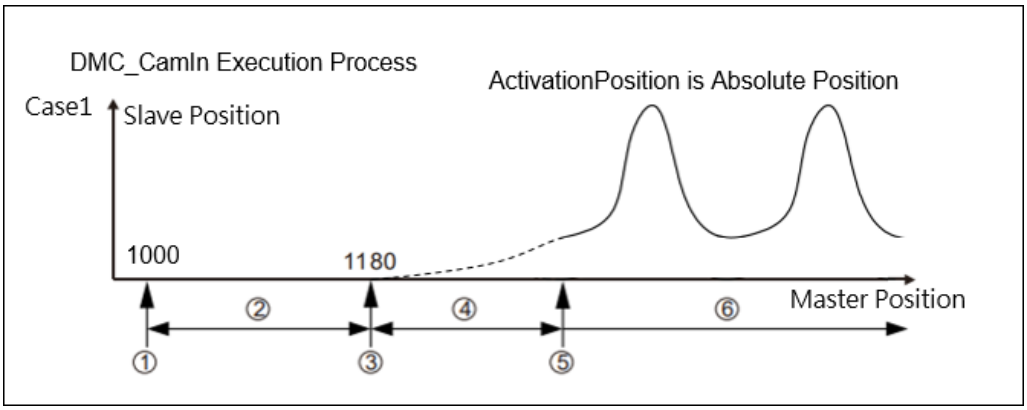
  If Current Position is positive and ahead of *ActivationPosition*, then Ps's ModCnt=1; If Current Position is behind *ActivationPosition*, then Ps's ModCnt=0;

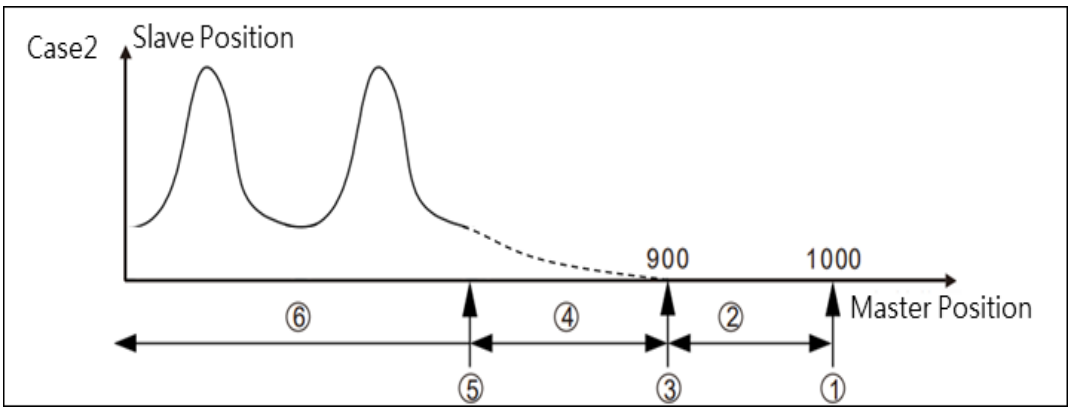
  If Current Position is negtive and ahead of *ActivationPosition*, then Ps's ModCnt=0; If Current Position is behind *ActivationPosition*, then the Ps's ModCnt=1;

  The master axis rotates forward to find the target position (Ps), and ModCnt does not need additional corrected, if look in reverse, the ModCnt will –1.

  For example, the master axis modulo is 400, *ActivationPosition*=100,

  The DMC_CamIn master axis position is 1000 at the start point of the instruction, and since the absolute axis phase of the master axis at the start point of the DMC_CamIn instruction is 200 (200=1000%400), the slave axis does not perform the gearing action. When the master axis position is 1300 (100=1300%400 for the absolute axis phase) or 900 (100=900%400 for the absolute axis phase), the gearing action is performed from the axis (% for remainder).





1. Trigger DMC_CamIn to run. The absolute position of the master axis is 100 at this point.
2. Wait for engaging.
3. The master axis reaches the engaging start position (1300 for Case 1, 900 for Case 2), and the slave axis starts engaging.
4. Engaging in progress
5. Engaging completed and the master and slave axes are synchronized.
6. Master and slave axes are acting synchronously.

Note: When *ActivationPosition* is absolute axis phase, the valid range of the parameter *ActivationPosition* is: 0–modulo (excluding). If the value of *ActivationPosition* is not within the valid range, the running of DMC_CamIn instruction will report an error and the running will fail.

- *ActivationMode* = PhaseCAM; *ActivationMode* is absolute cam phase.

  ActivationPosition and the master axis position when the instruction is triggered are the results of the absolute position being calculated to take the remainder according to its cam cycle.

  The cam axis phase is periodic. During the master axis is running, its absolute axis may be equal to *ActivationPosition* for many times. The master axis will have both forward and reverse target position when the master axis and the slave axis gears. The relationship between the current position of the master axis and *ActivationPosition* will also affect the number of mode counts (ModCnt) and the target position of the master axis gear.

  Ps = *ActivationPosition* + [(Current Position / Cam End) + ModCnt] x Cam End

- ModCnt Calculation
  - Current Position is positive number

	Positive master axis rotation	**Negative master axis rotation**
Current Position > Activation Position	1	0
Current Position < ActivationPosition	0	-1

  - Current Position is negative number

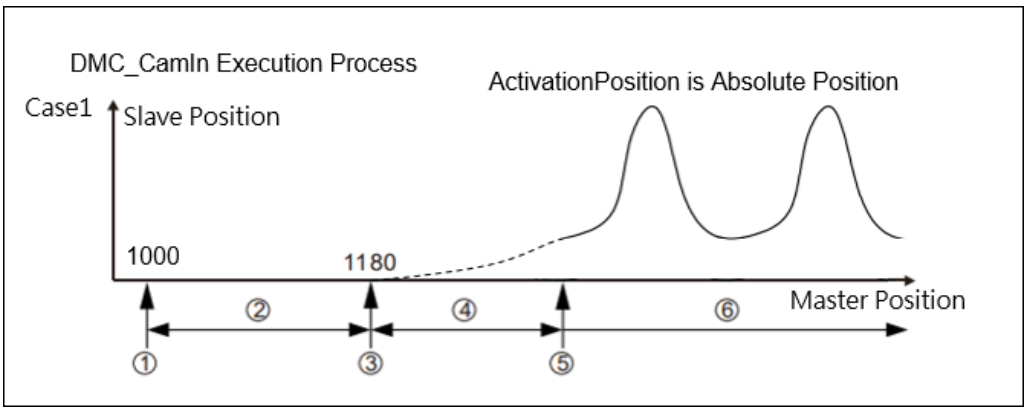
	Positive master axis rotation	**Negative master axis rotation**
Current Position > ActivationPosition	0	–1
Current Position < ActivationPosition	–1	–2

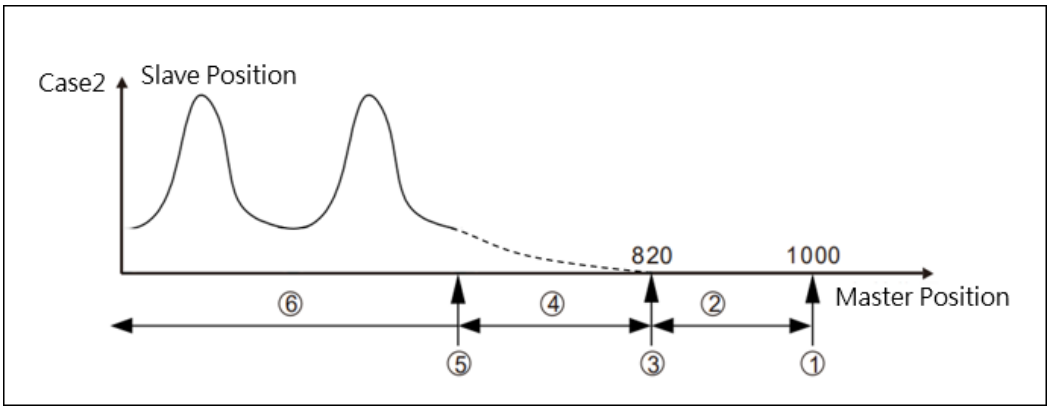
If Current Position is positive and ahead of *ActivationPosition*, then Ps's ModCnt=1; If Current Position is behind the *ActivationPosition*, then the Ps's ModCnt=0;

If Current Position is negtive and ahead of *ActivationPosition*, then Ps's ModCnt=0; If Current Position is behind the *ActivationPosition*, then the Ps's ModCnt=1;

The master axis rotates forward to find the target position (Ps), and ModCnt does not need additional corrected, if look in reverse, ModCnt will –1.

For example: The maximum value for the master axis in the cam table is 360. *ActivationPosition* = 100, the master axis position is 1000 when the DMC_CamIn instruction starts running, and the slave axis will not perform engaging because the absolute cam phase of the master axis is 280 (280 = 1000%360). When the master axis position is 1180 (absolute cam phase is 100 = 1180%360) or 820 (absolute axis phase is 100 = 820%360), the slave axis starts engaging (% represents taking remainder).



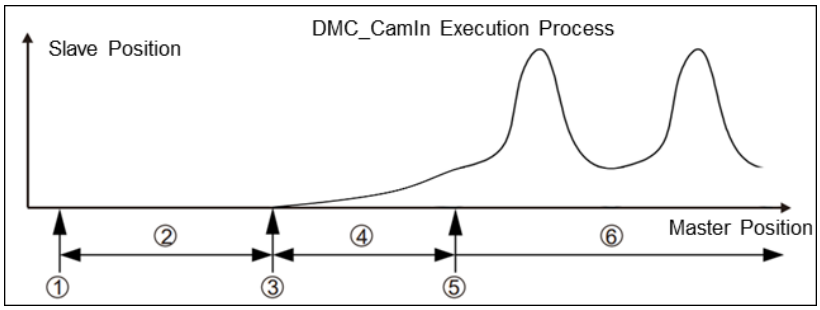


1. Trigger DMC_CamIn to run. The absolute position of the master axis is 1000 at this point (The absolute axis phase is 280).
2. Wait for engaging.
3. The master axis reaches the engaging start position (1180 for Case 1, 820 for Case 2), and the slave axis starts engaging.
4. Engaging in progress.
5. Engaging completed and the master and slave axes are synchronized.
6. Master and slave axes are acting synchronously.

Note: When *ActivationPosition* is absolute axis phase, the valid range of the parameter *ActivationPosition* is: 0–cam cycle (excluding). If the value of *ActivationPosition* is not within the valid range, the running of DMC_CamIn instruction will report an error and the running will fail.

- The relationship between *StartMode* and MasterAbsolute and SlaveAbsolute of CamTableSelect

  During engaging, the motion mode of the slave axis can be set by the parameter *StartMode*. *StartMode* acts in 4, as shown in the following figure.



1. Trigger DMC_CamIn to run.
2. Wait for engaging.
3. The master axis reaches the engaging start position, and the slave axis starts engaging.
4. Engaging in progress
5. Engaging completed and the master and slave axes are synchronized.
6. Master and slave axes are acting synchronously.

   - *StartMode* = 0; Absolute mode: The slave current position is not involved in the Cam calculation as the cam synchronization starts. However, a runout will occurr if the current position of slave axis and its start position from the Cam are not the same.
   - *StartMode* = 1; Relative mode: The cam will change according to the current position of the slave axis. The slave axis position equals to the current position

plus target position. If the slave axis position when engaging is different from the start position plus current position, a runout may occur.

- *StartMode* = 2,3,4; Ramp mode: Add a compensating motion curve to prevent the cam from runout when it starts engaging according to *lrVelocityDiff*, *lrAcceleration*, *lrDeceleration*, and *lrJerk*.

MC_CamTableSelect.MasterAbsolute	Master axis mode
absolute	Absolute
relative	Relative

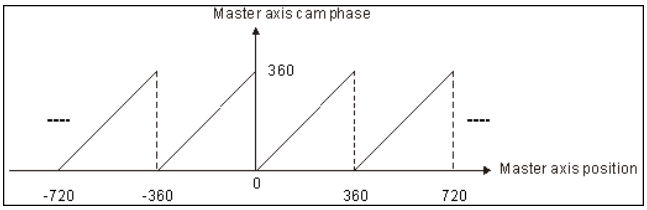
DMC_CamIn .StartMode	MC_CamTableSelect.SlaveAbsolute	Slave axis mode
absolute	TRUE	Absolute
absolute	FALSE	Relative
relative	TRUE	Relative
relative	FALSE	Relative
ramp_in	TRUE	Ramp in Absolute
ramp_in	FALSE	Ramp in Relative
ramp_in_pos	TRUE	Ramp in Positive Absolute
ramp_in_pos	FALSE	Ramp in Positive Relative
ramp_in_neg	TRUE	Ramp in Negative Absolute
ramp_in_neg	FALSE	Ramp in Negative Relative

  - *MC_CamTableSelect.MasterAbsolute* is used to specify the corresponding relationship between the master axis position and its cam phase: absolute when the value is TRUE; relative when the value is FALSE.

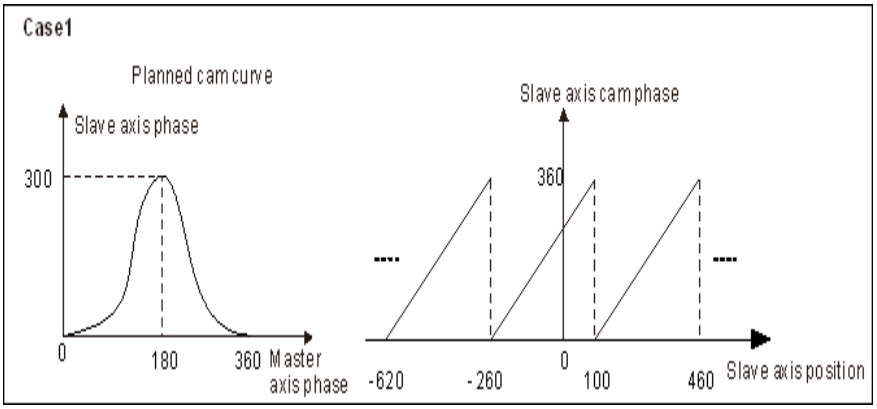
  - *MC_CamTableSelect.Slave Absolute* is used to specify the corresponding relationship between the slave axis position and the cam phase. Specify when engaging starts. The cam phase will be calculated according to this relationship, and the engaging method of the slave axis is related to *DMC_CamIn.StartMode*. Refer to the table above.

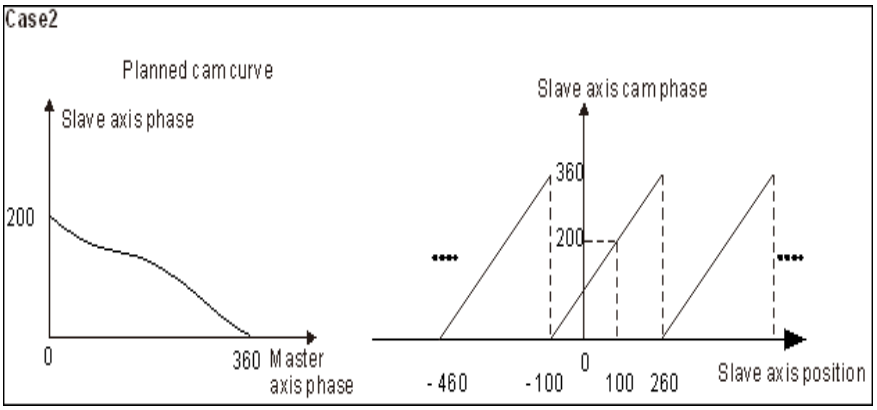
- *MasterAbsolute* = FALSE

  - When *MasterAbsolute* is FALSE, the master axis position is relative to its cam phase. That is, the master axis position corresponds to its cam phase 0 when engaging starts, and then the master axis cam phase will be calculated according to this relationship. For example: The master axis is in relative mode. The maximum value of the master axis in the cam mechanism is 360. The master axis position is 180 when engaging starts, its corresponding cam phase is 0. Axis position 200 corresponds to its cam phase 20 (20 = (200-180)%360), and so on. In this case, the relationship between the master axis position and its cam phase is shown in the following figure.



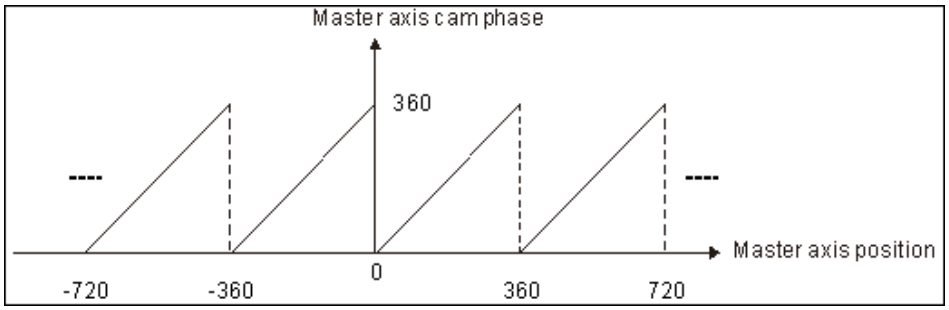
- When *SlaveAbsolute* is FALSE, the slave axis position is relative to its cam phase. That is, when engaging starts, the cam phase of the slave axis and the cam phase of the current master axis match the cam mechanism planned. When the slave axis is in relative mode, the method of determining the cam phase of the slave axis is different from that of the master axis. To determine the cam phase of the slave axis, the cam phase of the slave axis and the cam phase of the current master axis should match the cam mechanism planned. For example: The slave axis is in relative mode. The maximum value of the master axis in the cam mechanism is 360. The master axis position is 100 when engaging starts. If the cam phase of the master axis is 0 at this moment (the slave cam phase is 0 according to the cam mechanism), then the slave axis position 100 corresponds to its cam phase 0, as shown in Case 1 in the following figure. If the slave cam phase is 200 according to the cam mechanism, then the slave position 100 corresponds to its cam phase 200, as shown in Case 2.





- *MasterAbsolute* = TRUE

  When *MasterAbsolute* is TRUE, the relationship between the master axis and its cam phase is absolute. The cam phase of the master axis is the result of taking remainder between the master axis position and the maximum range value of the master axis in the cam mechanism. For example, if the master axis is in absolute mode, and the maximum value of the master axis is 360, when the master axis position is 100, its cam phase is 100 (100=100%360); When the master axis position is 500, its cam phase is 140 (140=500%360), and so on. The relationship between the master axis position and its cam phase is shown in the following figure.



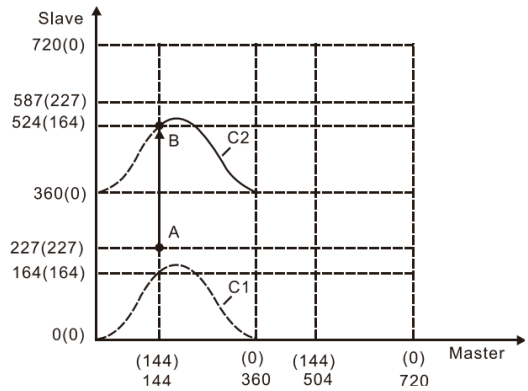
- When *SlaveAbsolute* is TRUE, the relationship between the slave axis and its cam phase is absolute. The cam phase of the slave axis is the result of taking remainder between the slave axis position and the maximum range value of the slave axis in the cam mechanism. When the slave axis is in absolute mode, the relationship between the slave axis position and its cam phase is consistent with that of the master axis.

- When *StartMode* = Absolute or relative

DMC_CamIn.StartMode	MC_CamTableSelect.SlaveAbsolute	Slave axis mode
absolute	TRUE	absolute
absolute	FALSE	relative
relative	TRUE	relative
relative	FALSE	relative

- The slave axis will jump to the engaging point after a cycle, and the point will determine the absolute mode or relative mode according to *StartMode* and MC_CamTableSelect.SlaveAbsolute, as shown in the table above. The mode will engage in a jump way.

- In the following figure, C1 is the planned cam curve. C2 is the actual running cam curve. The coordinates of point A (master, slave axis) are the current position when the DMC_CamIn instruction is triggered, and point B is the actual engaging position of the master and slave axis after jump.



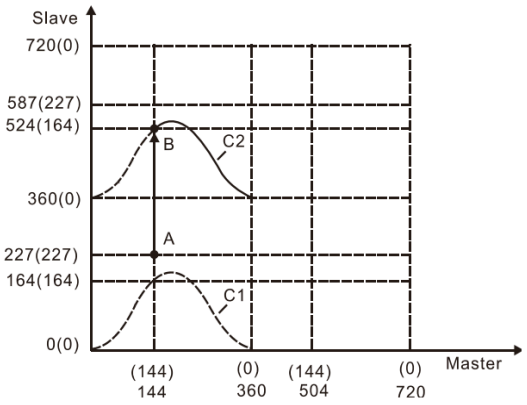
The slave axis moves from point A to point B in relative mode. The slave axis jumps from position 227 to position 524 in one cycle.

- When *StartMode* = Ramp in absolute mode or Ramp in relative mode

DMC_CamIn.StartMode	MC_CamTableSelect.SlaveAbsolute	Slave axis mode
ramp_in	TRUE	Ramp in absolute mode

DMC_CamIn .StartMode	MC_CamTableSelect.SlaveAbsolute	Slave axis mode
ramp_in	FALSE	Ramp in relative mode

- The engaging point will determine the absolute mode or relative mode according to *StartMode* and *MC_CamTableSelect.SlaveAbsolute*, as shown in the table above.
- The motion of the slave axis at this time is affected by the parameters *Velocity*, *Acceleration*, *Deceleration*, and *Jerk*.

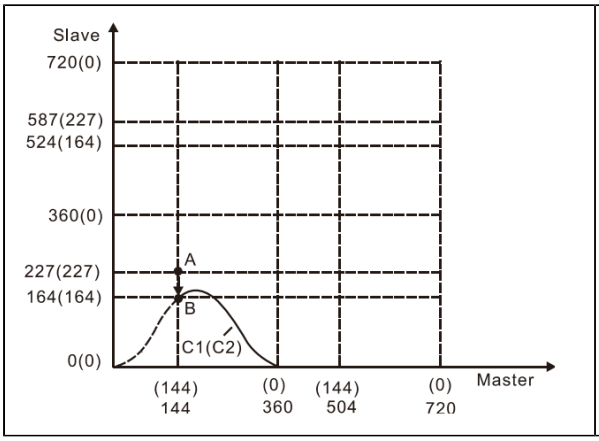


The slave axis moves forward from start point A to engaging point B (297 = (360–227) + (164–0).

- When *StartMode* = Ramp in negative absolute mode or Ramp in negative relative mode

DMC_CamIn .StartMode	MC_CamTableSelect.SlaveAbsolute	Slave axis mode
ramp_in_neg	TRUE	Ramp in negative absolute
ramp_in_neg	FALSE	Ramp in negative relative

- The engaging point will determine the absolute mode or relative mode according to *StartMode* and *MC_CamTableSelect.SlaveAbsolute*, as shown in the table above.
- The slave axis will accelerate in reverse direction to the engaging point.
- The motion of the slave axis at this time is affected by the parameters *Velocity*, *Acceleration*, *Deceleration*, and *Jerk*.

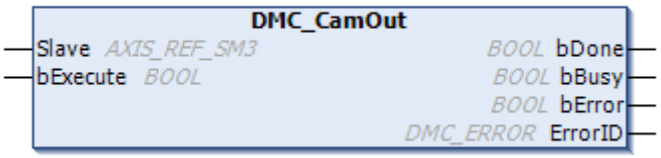


The slave axis moves reversely 63 units from the starting point A to the engaging point B (63 = 227–164).

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - Refer to the examples of MC_CamIn.
- **Supported Devices**
  - AX-series motion controller

2.2.1.28 DMC_CamOut

DMC_CamOut is used to disengage the slave axis from the master axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_CamOut		DMC_CamOut_instance(Slave:= , bExecute:= , bDone=> , bBusy=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

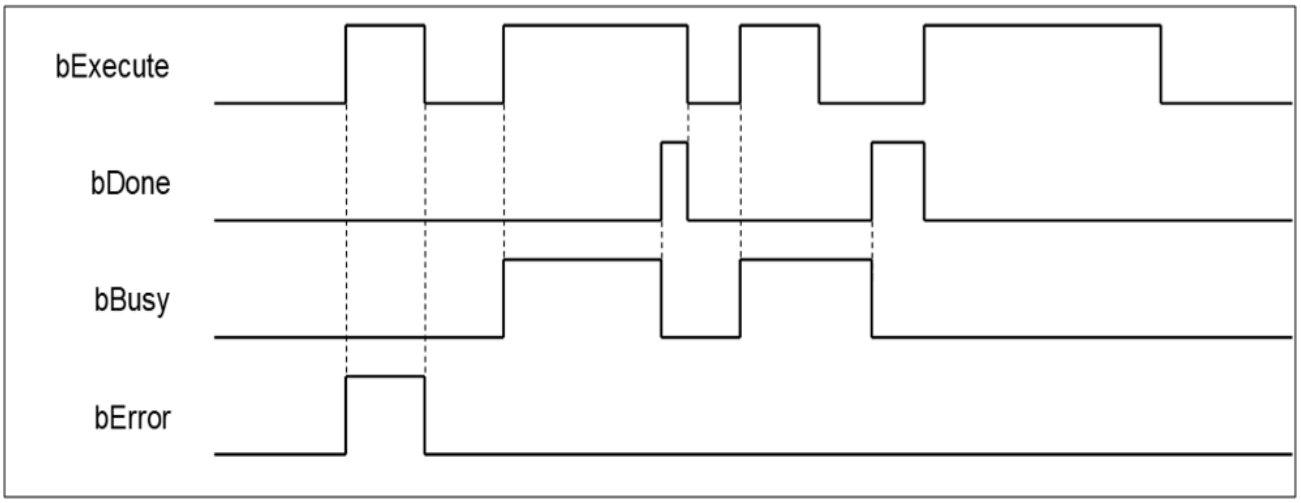
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the master and slave axes are disengaged	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The instruction DMC_CamOut is complete.	• *bExecute* is FALSE • *bError* is TRUE
bBusy	The instruction is running.	• *bExecute* is FALSE • *bError* is TRUE
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* is FALSE. (Clear the error code recorded in *ErrorID*.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

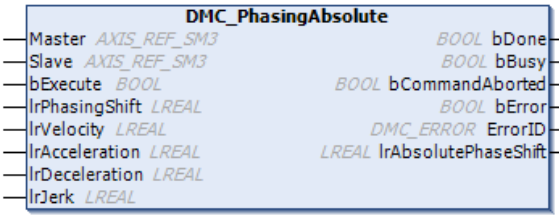
Name	Function	Data Type	Setting Value	Timing to Take Effect
Slave	The specified slave axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE.

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

- **Function**
  - This function block is supported by DL_MotionControl V1.3.4.0 or later.
  - This function block supports PLC simulation mode in DL_MotionControl V1.4.0.0 or later.
  - This function block can only be used with DMC_CamIn.
  - You can use this function block with DMC_PhasingAbsolute and DMC_PhasingRelative to modify the phase position of the slave axis.
  - When the slave axis is decoupled from the master axis, it maintains the velocity while decoupling and the slave state is ContinuousMotion. (irrelevant to the velocity of the slave axis)
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - Refer to the examples of MC_CamOut.
- **Supported Devices**
  - AX-series motion controller

2.2.1.29 DMC_PhasingAbsolute

DMC_PhasingAbsolute controls the master axis phase compensation according to the specified absolute phase compensation value.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_PhasingAbsolute		DMC_PhasingAbsolute_instance(Master :=, Slave:=, bExecute:=, lrPhasingShift:=, lrVelocity:=, lrAcceleration:=, lrDeceleration:=, lrJerk:=, bDone =>, bBusy =>, bCommandAborted =>, bError =>, ErrorID =>, lrAbsolutePhaseShift =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrPhaseShift	Phase shift between the master and slave axis*	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrVelocity	The max velocity of the phase shift (Unit: user unit/s)	LREAL	Positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrAcceleration	The max acceleration of the phase shift (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrDeceleration	The max deceleration of the phase shift (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrJerk	The max jerk value of the phase shift (Unit: user unit/s^3)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.

***Note**: If positive value, 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)
bDone	TRUE when the phase compensation is complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)

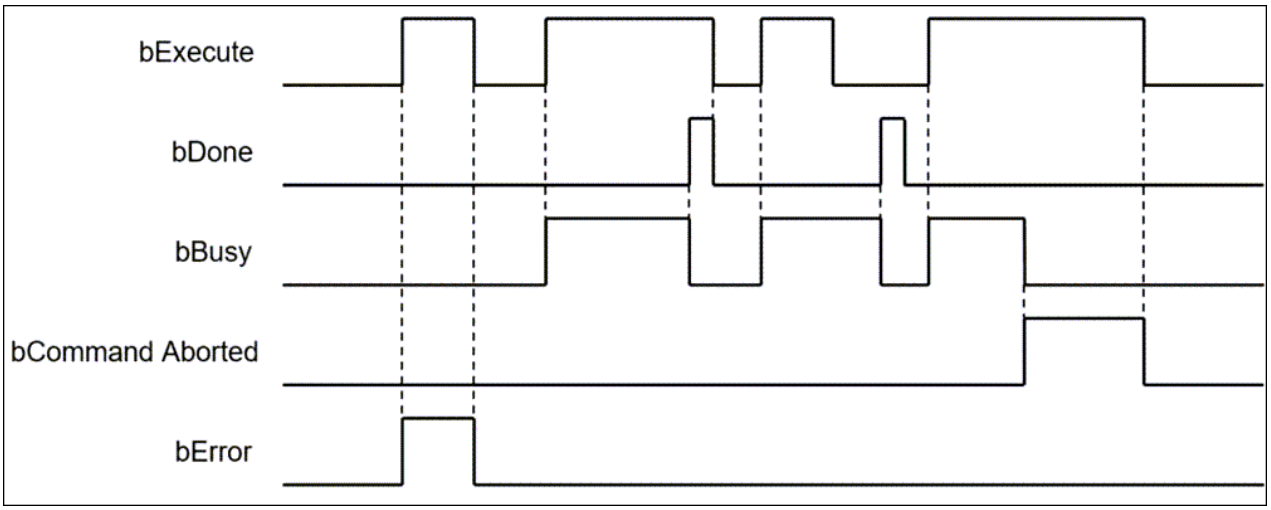
Name	Function	Data Type	Output Range (Default Value)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The phase compensation is complete.	• *bExecute* changes to FALSE. • *bError* turns to TRUE.
bBusy	The instruction is running.	• *bExecute* changes to FALSE. • *bError* turns to TRUE.
bCommandAborted	This instruction is interrupted by another instruction.	• *bExecute* changes to FALSE. • If *bExecute* is FALSE and *bCommandAborted* turns to TRUE, *bCommandAborted* will be TRUE for only one cycle and then immediately turn to FALSE.
bError ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns from TRUE to FALSE. (Clear the error code recorded in *ErrorID*.)

- **Timing Diagram of Output Parameter Changes**

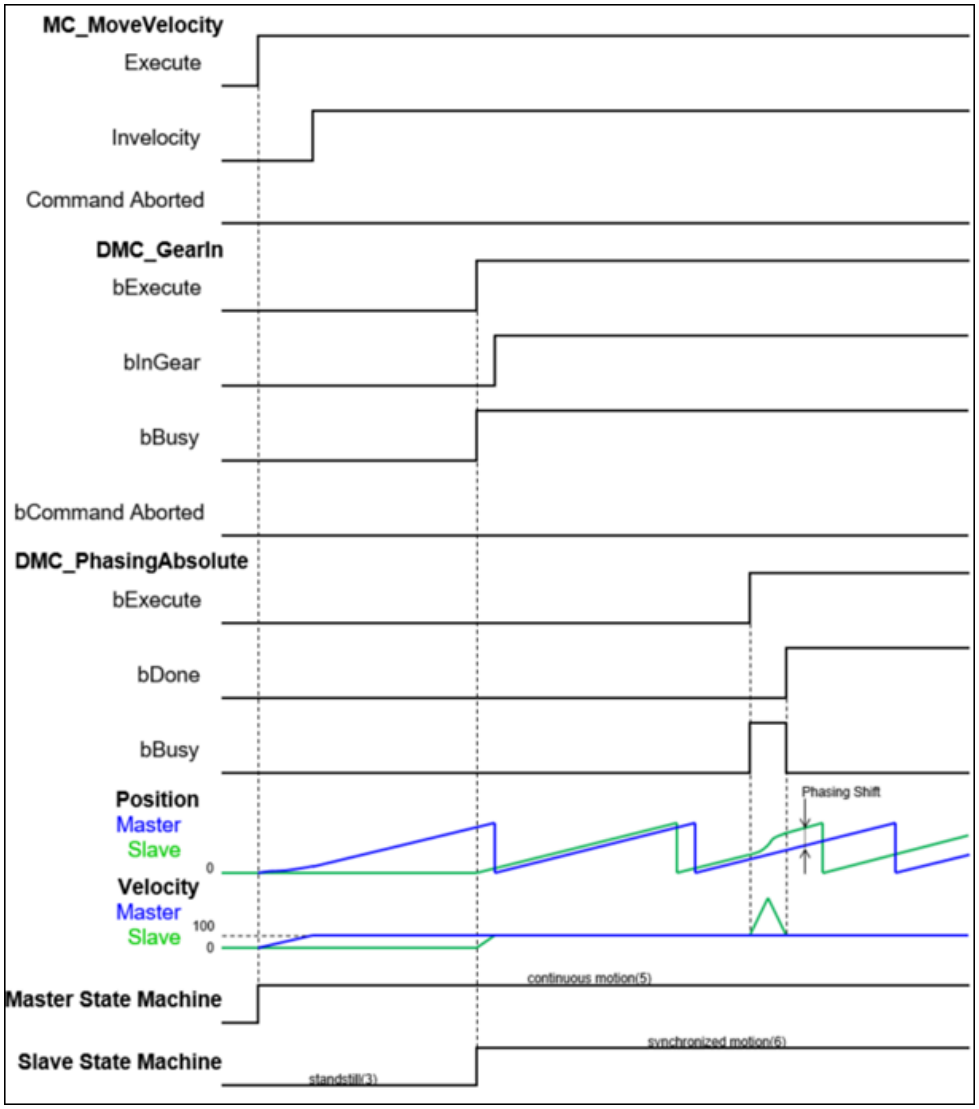


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	The specified the master axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE.
Slave	The specified the slave axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE.

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

- **Function**
  - This function block is supported by DL_MotionControl V1.3.4.0 or later.
  - This function block is not available in PLC simulation mode.
  - This function block can only be used with DMC_GearIn, DMC_CamIn, DMC_GearInPos, and DMC_CombineAxis.
  - When *bExecute* is turning to TRUE, the master-slave axis relation is established. The slave axis will shift the phase by planning a smooth curve. If the *PhaseShift* value is positive, the slave axis is lagging behind the master axis, and the negative means the slave axis is ahead of the master axis.
  - The position of the master axis remains unchanged while only the phase shift of the slave axis is performed.

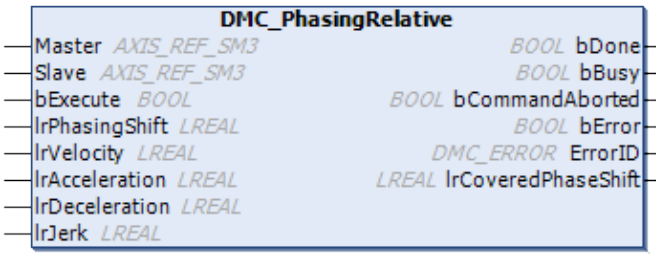


- DMC_PhasingAbsolute can only be used when the state is Synchronized motion.
- When DMC_PhasingAbsolute is running, the state of the slave axis will remain in Synchronized motion.
- Place DMC_PhasingAbsolute before DMC_CamIn to ensure that the phase compensation is correct.
- Depending on where this function block is placed in the program, the results of the synchronous motion are different. For more information, see *AX-3 Series Operation Manual* section 7.7.2.5 Synchronous Control.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - Refer to the examples of DMC_PhasingRelative.
- **Supported Devices**
  - AX-series motion controller

2.2.1.30 DMC_PhasingRelative

DMC_PhasingRelative controls the master axis phase compensation according to the specified relative phase compensation value.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_PhasingRelative		DMC_PhasingRelative_instance(Master :=, Slave:=, bExecute:=, lrPhasingShift:=, lrVelocity:=, lrAcceleration:=, lrDeceleration:=, lrJerk:=, bDone =>, bBusy =>, bCommandAborted =>, bError =>, ErrorID =>, lrCoveredPhaseShift =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrPhaseShift	Phase shift amount between the master and slave axis*	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrVelocity	The max velocity of the phase shift (Unit: user unit/s)	LREAL	Positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrAcceleration	The max acceleration of the phase shift (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrDeceleration	The max deceleration of the phase shift (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrJerk	The max jerk value of the phase shift (Unit: user unit/s^3)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.

***Note**: If positive value, 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)
bDone	TRUE when the phase compensation is complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)

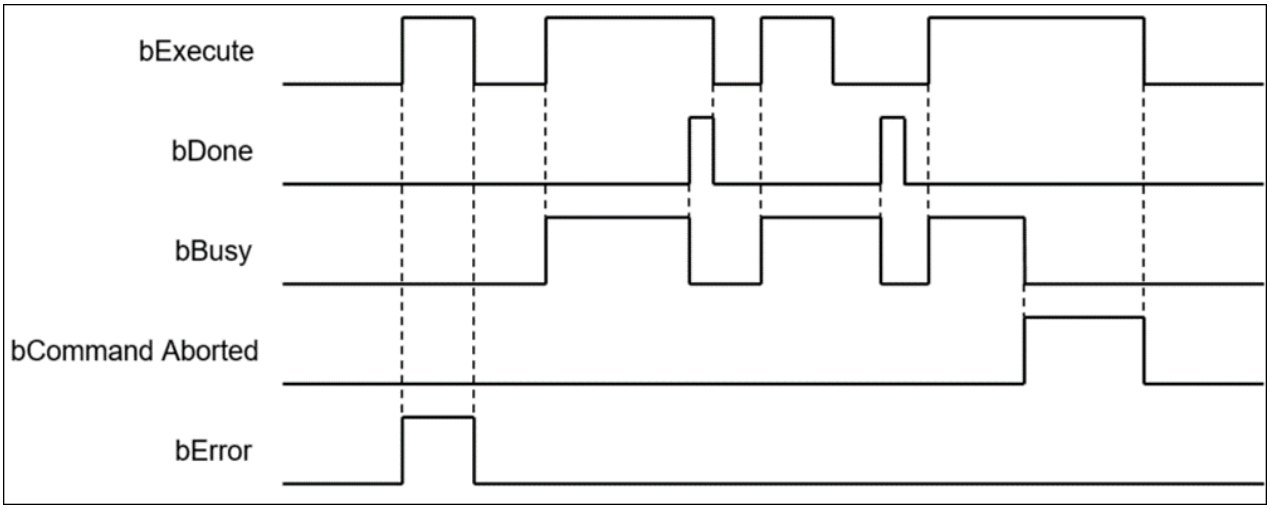
Name	Function	Data Type	Output Range (Default Value)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The phase shift of the master and slave axis reaches the target.	• *bExecute* changes to FALSE. • *bError* turns to TRUE.
bBusy	*bExecute* changes to TRUE.	• *bExecute* changes to FALSE. • *bError* turns to TRUE.
bCommandAborted	• This instruction is interrupted by another instruction. • This instruction is interrupted by MC_Stop.	• *bExecute* changes to FALSE. • If *bExecute* is FALSE and *bCommandAborted* turns to TRUE, *bCommandAborted* will be TRUE for only one cycle and then immediately turn to FALSE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	When *bExecute* turns from TRUE to FALSE. (Clear the error code recorded in *ErrorID*.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**

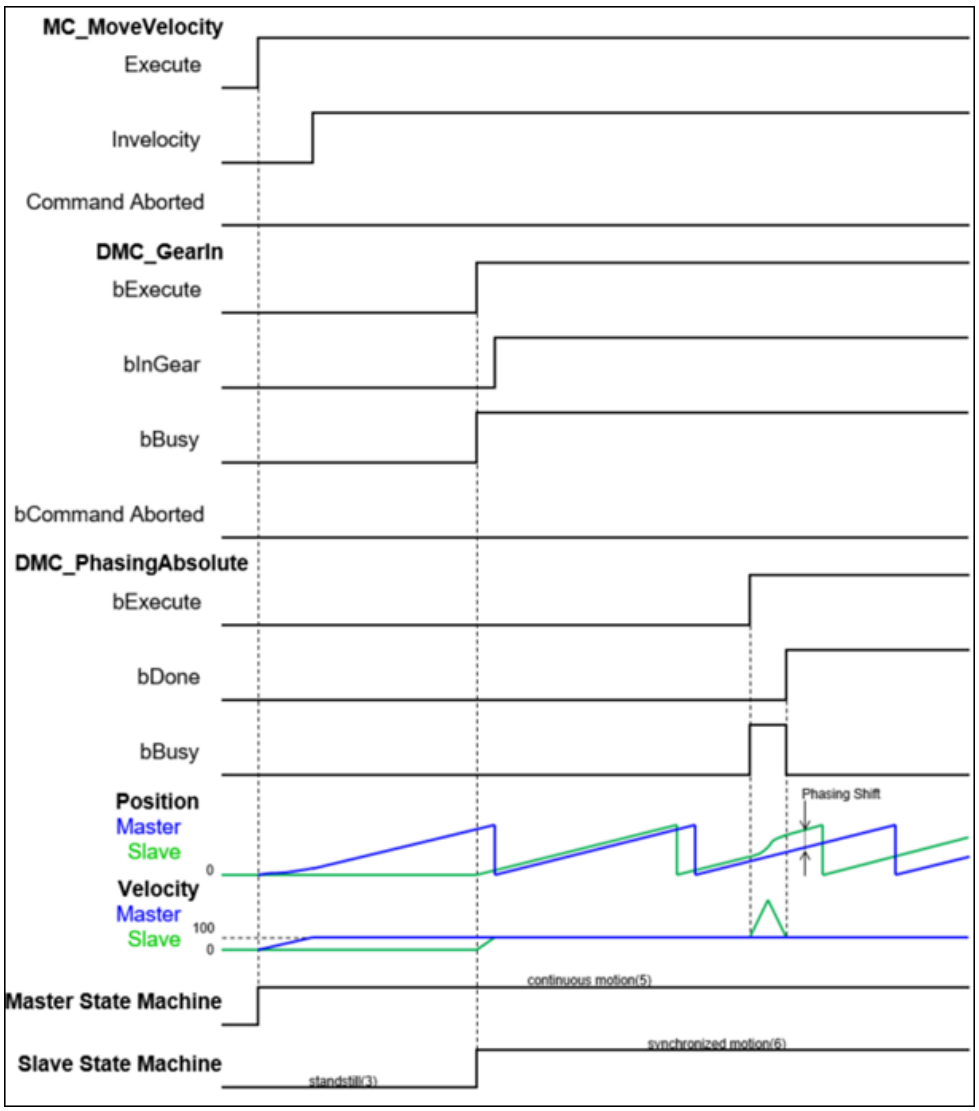


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	The specified master axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE.
Slave	The specified slave axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE.

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

- **Function**
  - This function block supported by DL_MotionControl V1.3.4.0 or later.
  - This function block is not available in PLC simulation mode.
  - This function block can only be used with DMC_GearIn, DMC_CamIn, DMC_GearInPos, and DMC_CombineAxis.
  - When *bExecute* is turning to TRUE, the master-slave axis relation is established. The slave axis will shift the phase by planning a smooth curve. If the *PhaseShift* value is positive, the slave axis is lagging behind the master axis, and the negative means the slave axis is ahead of the master axis.
  - The position of master axis remains unchanged while DMC_PhasingAbsolute acts on the slave axis.



  - DMC_PhasingRelative can only be used when the state is under Synchronized motion.

- When DMC_PhasingRelative is running, the state of slave axis will remain at Synchronized motion.
- Place DMC_PhasingRelative before DMC_CamIn to ensure that the phase compensation is correct.
- Depending on where this function block is placed in the program, the results of the synchronous motion are different. For more information, see *AX-3 Series Operation Manual* section 7.7.2.5 Synchronous Control.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

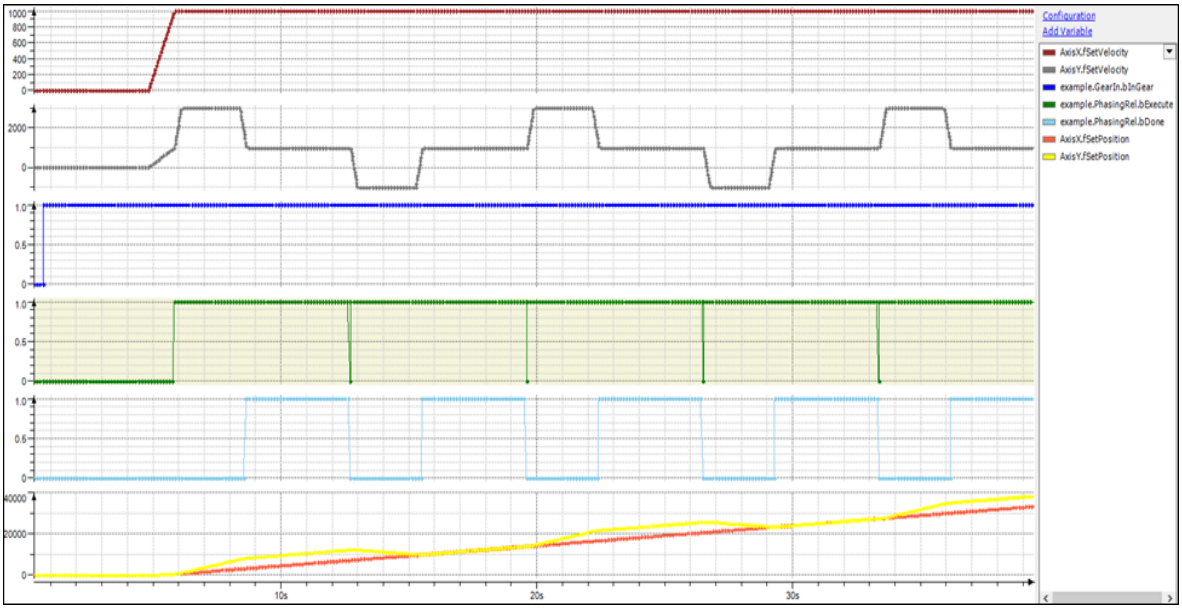
  This example illustrates the motion mode and track of DMC_PhasingRelative during running.

  - Variable declare

```
PROGRAM example_phasingREL
VAR
PowerX: MC_Power;
PowerY: MC_Power;
MoveVelocity: MC_MoveVelocity;
GearIn: DMC_GearIn;
PhasingRel: DMC_PhasingRelative;
stage: UINT := 0;
TaskCnt: UINT := 0;
END_VAR
Program
PowerX(Axis := AxisX);
PowerY(Axis := AxisY);
MoveVelocity(Axis := AxisX);
GearIn(Master := AxisX, Slave := AxisY);
PhasingRel(Master := AxisX, Slave := AxisY);
CASE stage OF
0 :
PowerX.bDriveStart := TRUE;
PowerX.bRegulatorOn := TRUE;
PowerX.Enable := TRUE;
PowerY.bDriveStart := TRUE;
PowerY.bRegulatorOn := TRUE;
PowerY.Enable := TRUE;
IF (PowerX.Status AND PowerY.Status) THEN
stage := 1;
END_IF
1 :
GearIn.MasterValueSource := DMC_SOURCE.dmcCommandedValue;
GearIn.RatioNumerator := 1;
GearIn.RatioDenominator := 1;
GearIn.lrAcceleration := 4000;
GearIn.lrDeceleration := 4000;
GearIn.lrJerk := 10;
GearIn.bExecute := TRUE;
IF (GearIn.bInGear) THEN
TaskCnt := TaskCnt + 1;
IF TaskCnt = 1000 THEN
stage := 2;
END_IF
END_IF
2 :
MoveVelocity.Velocity := 1000;
MoveVelocity.Acceleration := 1000;
MoveVelocity.Deceleration := 1000;
```

```
MoveVelocity.Jerk := 10;
MoveVelocity.Direction := MC_DIRECTION.current;
MoveVelocity.Execute := TRUE;
IF (MoveVelocity.InVelocity) THEN
stage := 3;
END_IF
3 :
PhasingRel.bExecute := FALSE;
stage := 4;
TaskCnt := 0;
4 :
PhasingRel.lrPhasingShift := 5000;
PhasingRel.lrVelocity := 2000;
PhasingRel.lrAcceleration := 8000;
PhasingRel.lrDeceleration := 8000;
PhasingRel.lrJerk := 100000;
PhasingRel.bExecute := TRUE;
IF (PhasingRel.bDone) THEN
TaskCnt := TaskCnt + 1;
IF TaskCnt = 1000 THEN
stage := 5;
END_IF
END_IF
5 :
PhasingRel.bExecute := FALSE;
stage := 6;
TaskCnt := 0;
6 :
PhasingRel.lrPhasingShift := -5000;
PhasingRel.lrVelocity := 2000;
PhasingRel.lrAcceleration := 8000;
PhasingRel.lrDeceleration := 8000;
PhasingRel.lrJerk := 100000;
PhasingRel.bExecute := TRUE;
IF (PhasingRel.bDone) THEN
TaskCnt := TaskCnt + 1;
IF TaskCnt = 1000 THEN
stage := 3;
END_IF
END_IF
END_CASE
```

- Timing Diagram



1. Run DMC_GearIn, establish a master-slave relationship, and then use MC_MoveVelocity to make the master axis run at the same speed.
2. When *bEexcute* is TRUE, the slave axis phase offset starts according to the value of *PhaseShift* of DMC_PhasingRelative.
3. When the slave axis reaches the offset set earlier, *bDone* stops, and *bBusy* resets.

- **Supported Devices**
  - AX-series motion controller

2.2.2 Administrative Motion Control Instructions

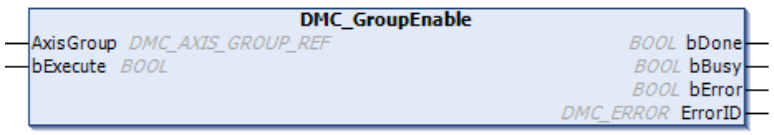
Administrative instructions generally refer to the running of the instruction to set the driver or read the relevant information without causing the actual motion of the motor. The function blocks used in this section are from the library "DL_MotionControl*" and the function blocks used can be synchronized 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.

2.2.2.1 DMC_GroupEnable

DMC_GroupEnable switches the axis group state from GroupDisable to GroupStandby.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_Group Enable		DMC_GroupEnable_instance(AxisGroup: = , bExecute: = , bDone=> , bBusy=> , bError=> , ErrorID=>) ;

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

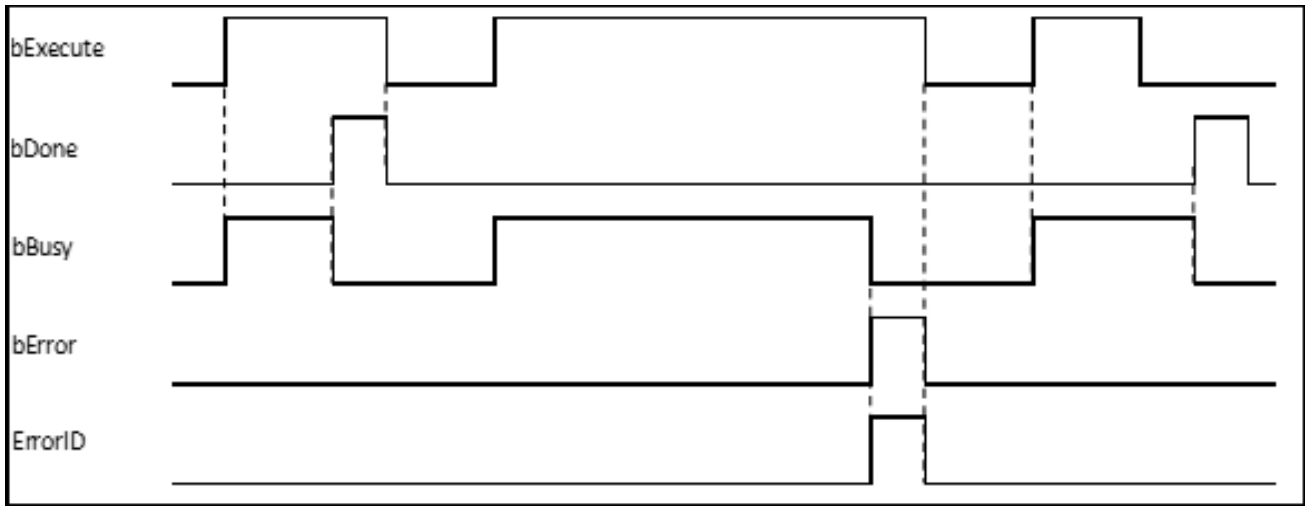
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the instruction is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The instruction is completed.	• *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	The instruction is running.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError (ErrorID)	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	*bExecute* turns from TRUE to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



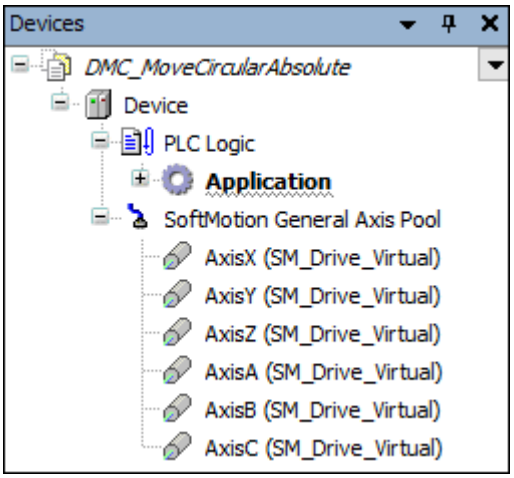
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*bExecute* turns to TRUE and *bBusy* is FALSE.

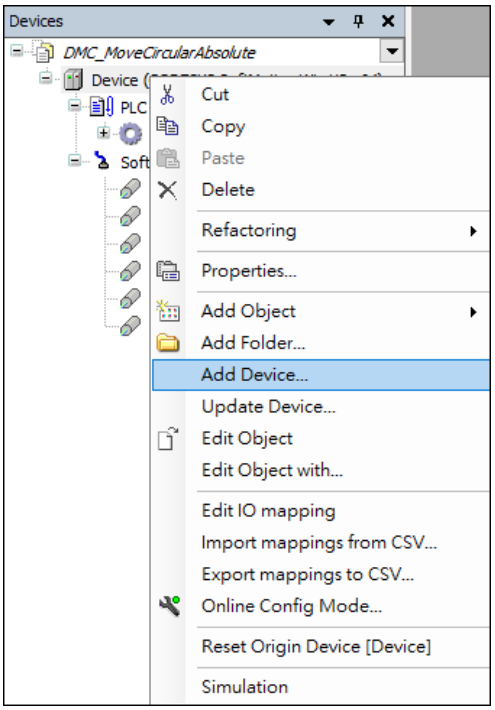
***Note**: DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**

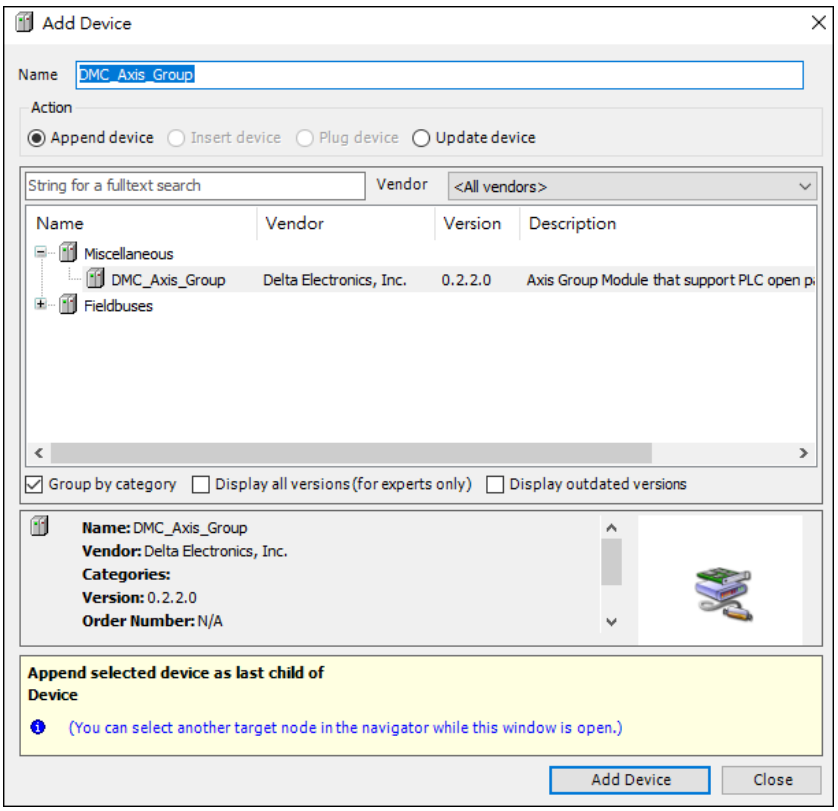
1. Add axes to **SoftMotion General Axis Pool** in the project. In this example, six virtual axes have been added, i.e. AxisX, AxisY, AxisZ, AxisA, AxisB, and AxisC.



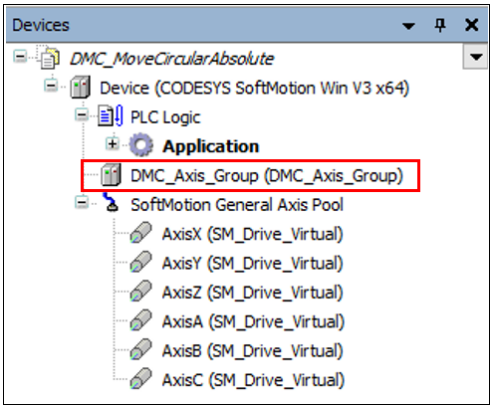
2. Right-click **Device** in the project and then select **Add Device**.



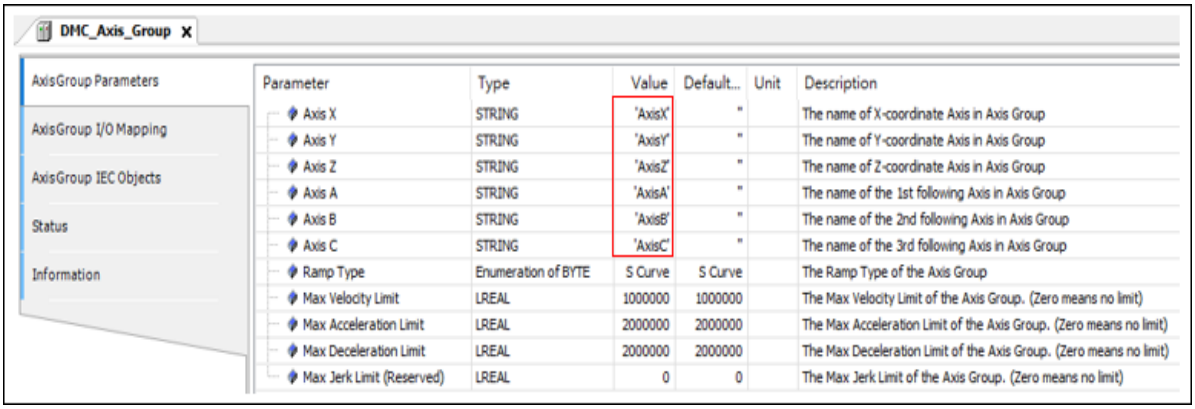
3. Find **DMC_ Axis_Group** and then click **Add Device**.



4. **DMC_Axis_Group (DMC_Axis_Group)**shows in the **Device** pane. The axis group is added.



5. Click **DMC_Axis_Group** setting page and then select **AxisGroup Parameters**. In the **Parameter** column, AxisX–AxisC represent axes 1–6 in the axis group. Enter the **Value** field of the Axis X–Axis C with the names of "AxisX"–"AxisC". The axis group in this example uses 6 axes, AxisX, AxisY, AxisZ, AxisA, AxisB, and AxisC.



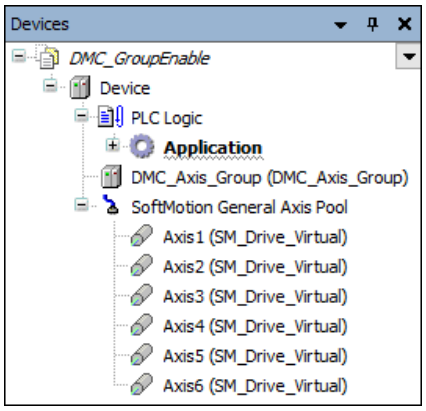
Parameter	Type	Value	Default...	Unit	Description
Axis X	STRING	'AxisX'	''		The name of X-coordinate Axis in Axis Group
Axis Y	STRING	'AxisY'	''		The name of Y-coordinate Axis in Axis Group
Axis Z	STRING	'AxisZ'	''		The name of Z-coordinate Axis in Axis Group
Axis A	STRING	'AxisA'	''		The name of the 1st following Axis in Axis Group
Axis B	STRING	'AxisB'	''		The name of the 2nd following Axis in Axis Group
Axis C	STRING	'AxisC'	''		The name of the 3rd following Axis in Axis Group
Ramp Type	Enumeration of BYTE	S Curve	S Curve		The Ramp Type of the Axis Group
Max Velocity Limit	LREAL	1000000	1000000		The Max Velocity Limit of the Axis Group. (Zero means no limit)
Max Acceleration Limit	LREAL	2000000	2000000		The Max Acceleration Limit of the Axis Group. (Zero means no limit)
Max Deceleration Limit	LREAL	2000000	2000000		The Max Deceleration Limit of the Axis Group. (Zero means no limit)
Max Jerk Limit (Reserved)	LREAL	0	0		The Max Jerk Limit of the Axis Group. (Zero means no limit)

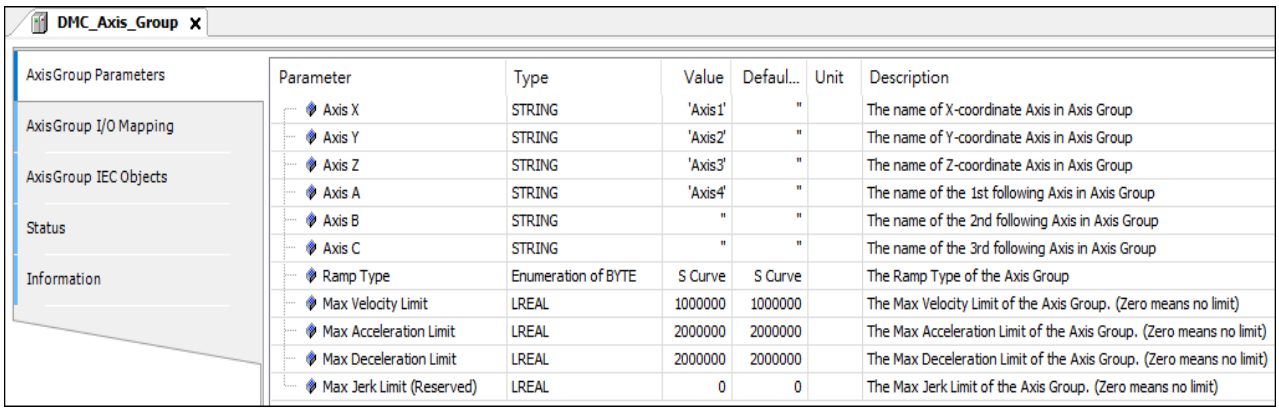
6. AxisX–AxisC in **AxisGroup Parameters** represent the axes 1–6 in the axis group respectively. They are 6 dimensions of space, i.e. coordinate axes X, Y, Z, A, B and C. If you do not set the coordinate axes, you don't need to fill in the values.
7. If the **Value** field for axes in **AxisGroup Parameters** is not set with the names of axes, no error will occur when DMC_GroupEnable is enabled. 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.
8. If there are duplicate or invalide axis names in the **Value** field in the **AxisGroup Parameters** tab, an error will occur when DMC_GroupEnable is running.
9. At least one axis is specified in the **Value** field for axes in the **AxisGroup Parameters** tab. Otherwise, an error will occur when DMC_GroupEnable is running.
10. Whether the specified single axis is in the standstill state or not will not be judged when DMC_GroupEnable is running. After the axis group is created, the state of the axis group will be switched from GroupDisable to GroupStandby, and axes in the axis group will keep the current state.
11. If axes in the group are in the ErrorSTOP state, the axis group state will change from GroupDisable->GroupStandby->GroupErrorStop.
12. For more details of axis states, refer to Axis State Transitions in *AX-3 Series Operation Manual*.

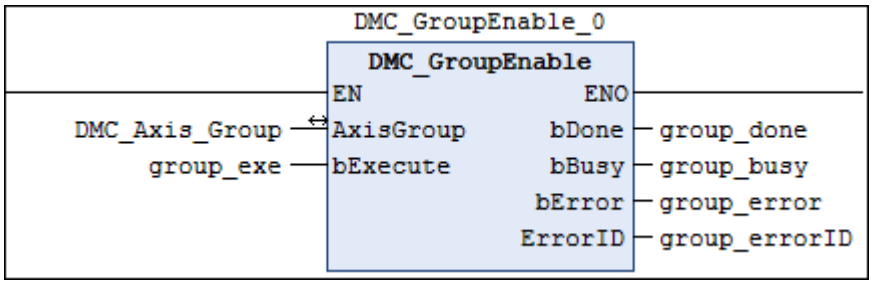
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding troubleshooting, refer to **Appendix** of this manual.

- **Example**

  This example refers to the limitation of the number of axes for multi-axis motion and the demand for axes in the actual application. It uses DMC_GroupEnable to switch the axis group state from GroupDisable to GroupStandby for the upcoming axis group motion.





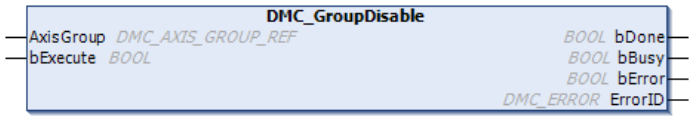


1. Before the absolute interpolation of Axis1– Axis4 is performed, create Axis1–Axis4, add them to the axis group DMC_Axis_Group, and then enter Axis1–Axis4 in the **Value** field for Parameter AxisX–AxisA.
2. Use DMC_GroupEnable to create the axis group before Axis 1–Axis 4 performs the absolute interpolation 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 keep the current state.
4. When DMC_GroupEnable is run after the axis group is created, no error occurs, and the axes enter the Standstill state. Then, the axis group DMC_Axis_Group can be used for the interpolation of simultaneous motion.

- **Supported Devices**
  - AX-series motion controller

2.2.2.2 DMC_GroupDisable

DMC_GroupDisable sets the state of an axis group to GroupDisable.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupDisable		DMC_GroupDisable_instance(AxisGroup: = , bExecute: = , bDone=> , bBusy=> , bError=> , ErrorID=>) ;

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

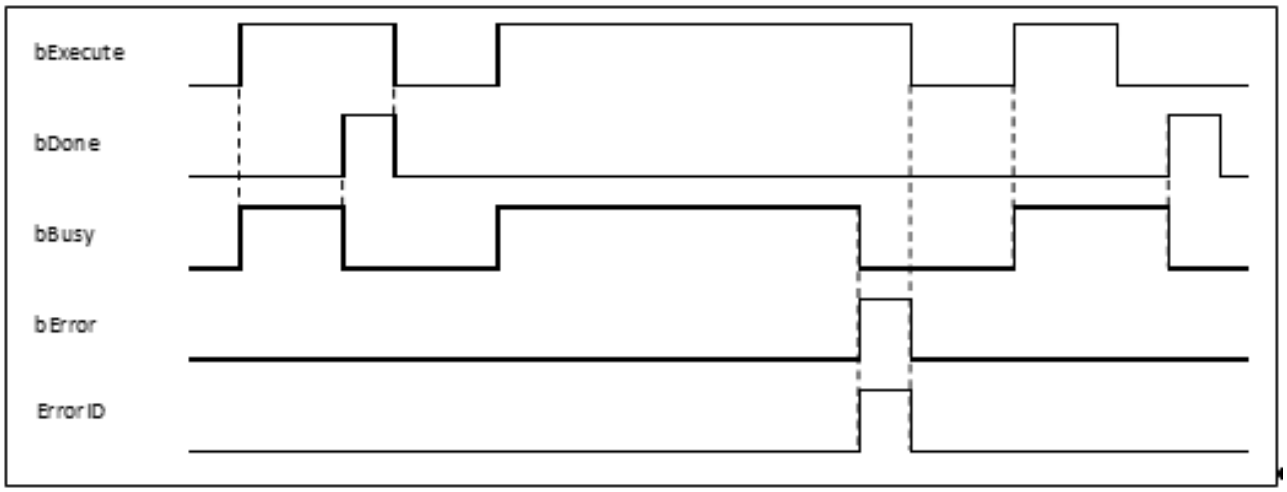
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the instruction is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The instruction is complete.	• *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	*bExecute* changes to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError (ErrorID)	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	*bExecute* turns from TRUE to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*bExecute* turns to TRUE and *bBusy* is FALSE.

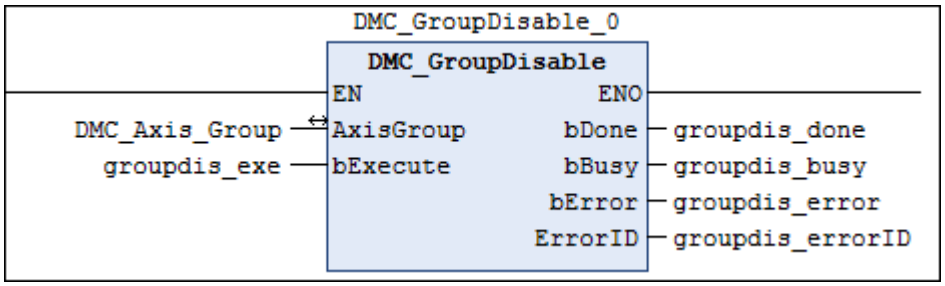
***Note**: DMC_AXIS_GROUP_REF (FB): All function blocks for an axis group contain this variable, which works as the starting program for function blocks.

- **Function**
  - When this instruction is 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 of axis states, refer to Axis State Transitions in *AX-3 Series Operation Manual*.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and troubleshooting, refer to **Appendix** of this manual.

- **Example**

Switch the axis group state from GroupStandby to GroupDisable.



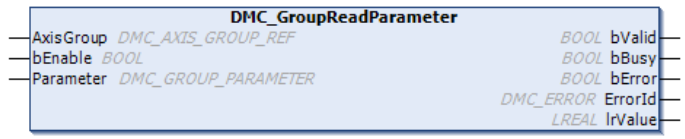
  - This instruction sets the group axis DMC_Axis_Group specified by AxisGroup to the GroupDisable state.

- DMC_GroupDisable will run when groupdis_exe (*bExecute*) changes to TRUE. If groupdis_done (*bDone*) changes to TRUE, DMC_Axis_Group axis group is set to GroupDisable state.

- **Supported Devices**
  - AX-series motion controller

2.2.2.3 DMC_GroupReadParameter

DMC_GroupReadParameter reads axis group parameters.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupReadParameter		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Parameter	The parameter to be set	DMC_GROUP_PARAMETER*		When the function block *bEnable* is TRUE, the setting 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	Maximum velocity limit
18	PARAM_MAX_ACCELERATION_LIMIT	Maximum acceleration limit
19	PARAM_MAX_DECELERATION_LIMIT	Maximum deceleration limit
21*1	PARAM_PLANNING_PRIORITY	Velocity ramp planning priority
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	Radius correction range

***Note**:

1. DL_MotionControl V1.2.0.0 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 running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*1	DMC_ERROR (DMC_NO_ERROR)
lrValue	Read parameter value	LREAL*2	Positive, negative , or 0 (0)

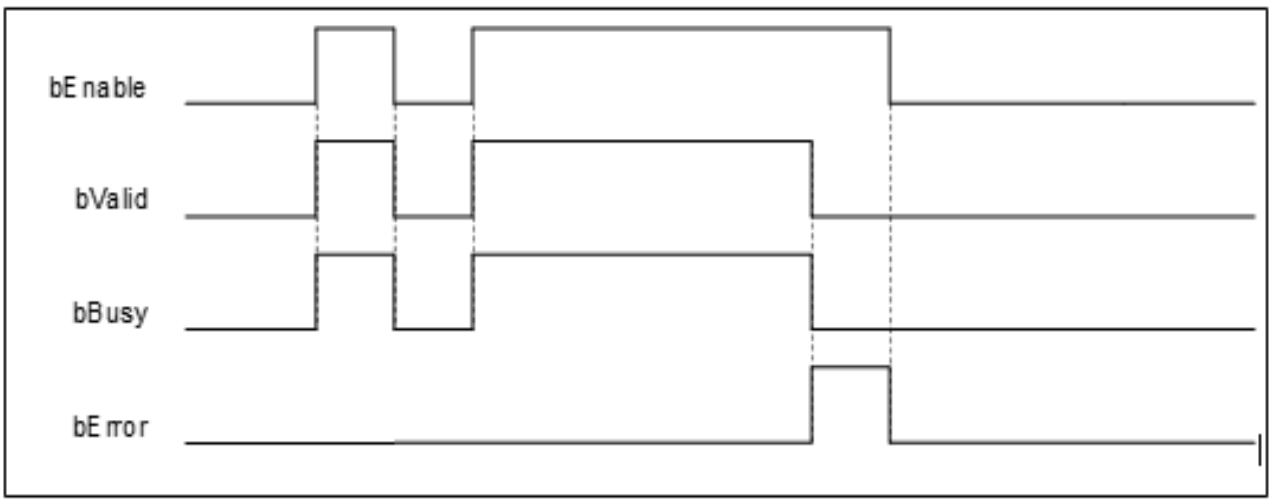
***Note**:

1. DMC_ERROR: Enumeration (Enum)
2. No matter what data type of the original parameter type is (including ENUM), the parameter read will be expressed as LREAL.

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bValid	*bEnable* turns to TRUE, and the output pin *lrValue* is valid.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bEnable* turns to TRUE.	• *bValid* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID*)	*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 of Output Parameter Changes**



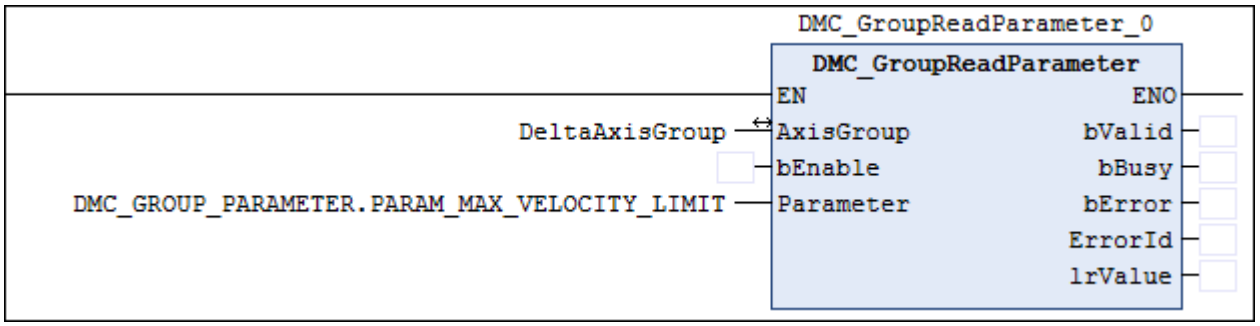
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*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 ENUM type, the parameter read will be its corresponding number.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding troubleshooting, refer to **Appendix** for error code descriptions.
- **Example**

  This example shows how to use DMC_GroupReadParameter to read axis group parameters.



- **Supported Devices**
  - AX-series motion controller

2.2.2.4 DMC_GroupWriteParameter

DMC_GroupWriteParameter writes axis group parameters.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupWriteParameter		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Parameter	The parameter to be written	DMC_GROUP_PARAMETER*2	DMC_GROUP_PARAMETER (PARAM_RAMP_TYPE)	When the function block *bExecute* is TRUE, the setting 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 of *Parameter* will be updated.

***Note**:

1. No matter what data 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	Maximum velocity limit
18	PARAM_MAX_ACCELERATION_LIMIT	Maximum acceleration limit
19	PARAM_MAX_DECELERATION_LIMIT	Maximum deceleration limit
21*	PARAM_PLANNING_PRIORITY	Velocity ramp planning priority
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

Setting Value	Name	Function
28	PARAM_RADIUS_CORRECTION_PERCENTAGE	Radius correction range

***Note**: DL_MotionControl V1.2.0.0 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
	0: Velocity 1: Acceleration (1)	0: Velocity first 1: Acceleration first

Note: From DL_MotionControl library V1.3.5.0 or later, the default value is 1(Acceleration).

- **Outputs**

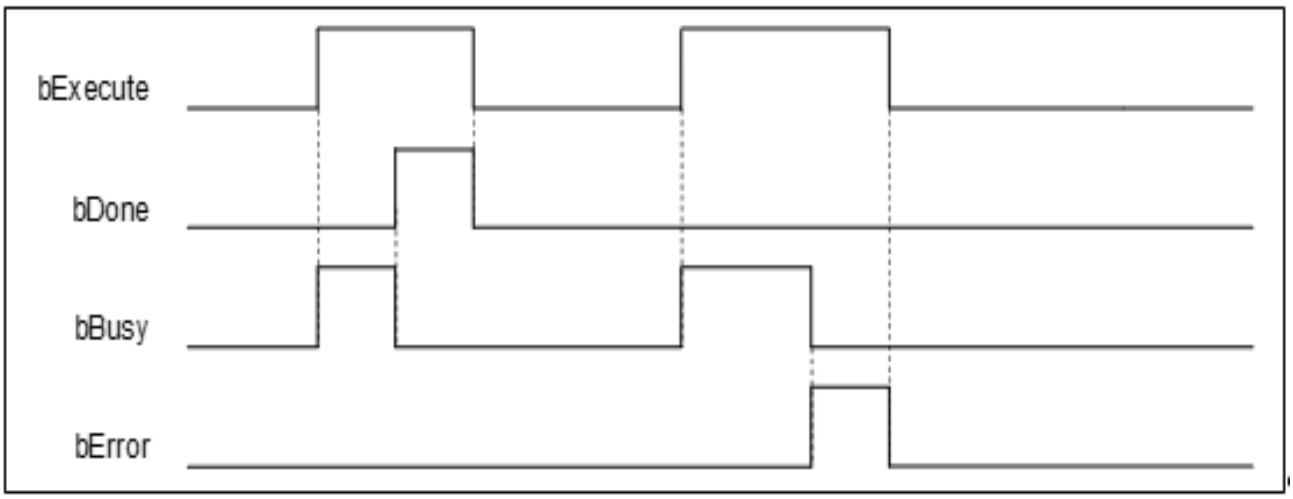
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the parameter is written	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The parameter is written.	• *bExcute* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bExexcute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID*)	*bEexcute* turns to FALSE. (Clear the error code recorded in *ErrorID*.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

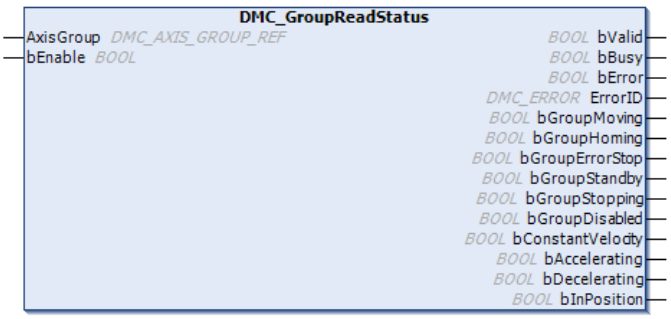
Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*bExecute* turns to TRUE, and *bBusy* is FALSE.

***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 written is ENUM type, set *IrValue* as the corresponding number.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and troubleshooting, refer to **Appendix** for error code descriptions.
- **Example**
  - Refer to the programming example of *DMC_GroupReadParameter*.
- **Supported Devices**
  - AX-series motion controller

2.2.2.5 DMC_GroupReadStatus

DMC_GroupReadStatus reads the state of an axis group.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupReadStatus		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	Runs the instruction when *bExecute* 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 running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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)

Name	Function	Data Type	Output Range (Default Value)
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)

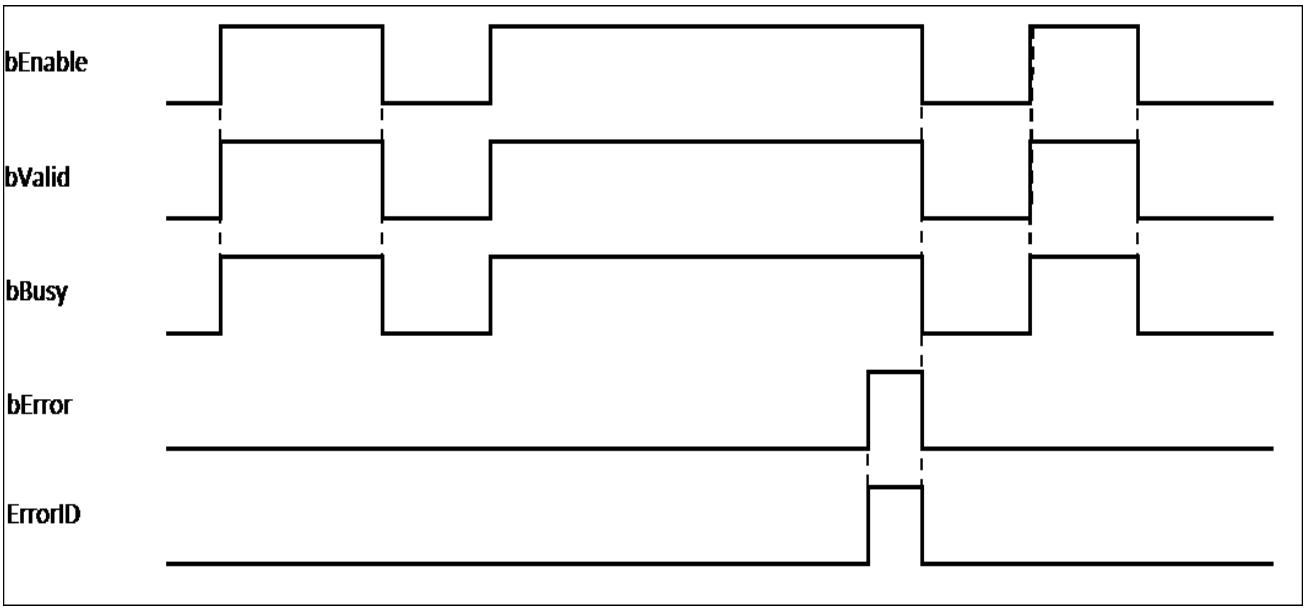
- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bValid	*bEnable* is TRUE and other axis group state outputs are valid.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bEnable* changes to TRUE.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bError (ErrorID)	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	• *bEnable* turns from TRUE to FALSE (Error Code is cleared.)
bGroupMoving	*bEnable* is TRUE and the output keeps updating its value.	• *bEnable* is TRUE and the output keeps updating its value. • *bEnable* turns to FALSE. • *bError* turns to TRUE.
bGroupHoming	*bEnable* is TRUE and the output keeps updating its value.	• *bEnable* is TRUE and the output keeps updating its value. • *bEnable* turns to FALSE. • *bError* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bGroupErrorStop	*bEnable* is TRUE and the output keeps updating its value.	• *bEnable* is TRUE and the output keeps updating its value. • *bEnable* turns to FALSE. • *bError* turns to TRUE.
bGroupStandby	*bEnable* is TRUE and the output keeps updating its value,	• *bEnable* is TRUE and the output keeps updating its value, • *bEnable* turns to FALSE, • *bError* turns to TRUE,
bGroupStopping	*bEnable* is TRUE and the output keeps updating its value.	• *bEnable* is TRUE and the output keeps updating its value. • *bEnable* turns to FALSE. • *bError* turns to TRUE.
bGroupDisabled	*bEnable* is TRUE and the output keeps updating its value.	• *bEnable* is TRUE and the output keeps updating its value. • *bEnable* turns to FALSE. • *bError* turns to TRUE.
bConstantVelocity	*bEnable* is TRUE and the output keeps updating its value.	• *bEnable* is TRUE and the output keeps updating its value. • *bEnable* turns to FALSE. • *bError* turns to TRUE.
bAccelerating	*bEnable* is TRUE and the output keeps updating its value.	• *bEnable* is TRUE and the output keeps updating its value. • *bEnable* turns to FALSE. • *bError* turns to TRUE.
bDecelerating	*bEnable* is TRUE and the output keeps updating its value.	• *bEnable* is TRUE and the output keeps updating its value. • *bEnable* turns to FALSE. • *bError* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bInPosition	*bEnable* is TRUE and the output keeps updating its value.	• *bEnable* is TRUE and the output keeps updating its value. • *bEnable* turns to FALSE. • *bError* turns to TRUE.

- **Timing Diagram of Output Parameter Changes**



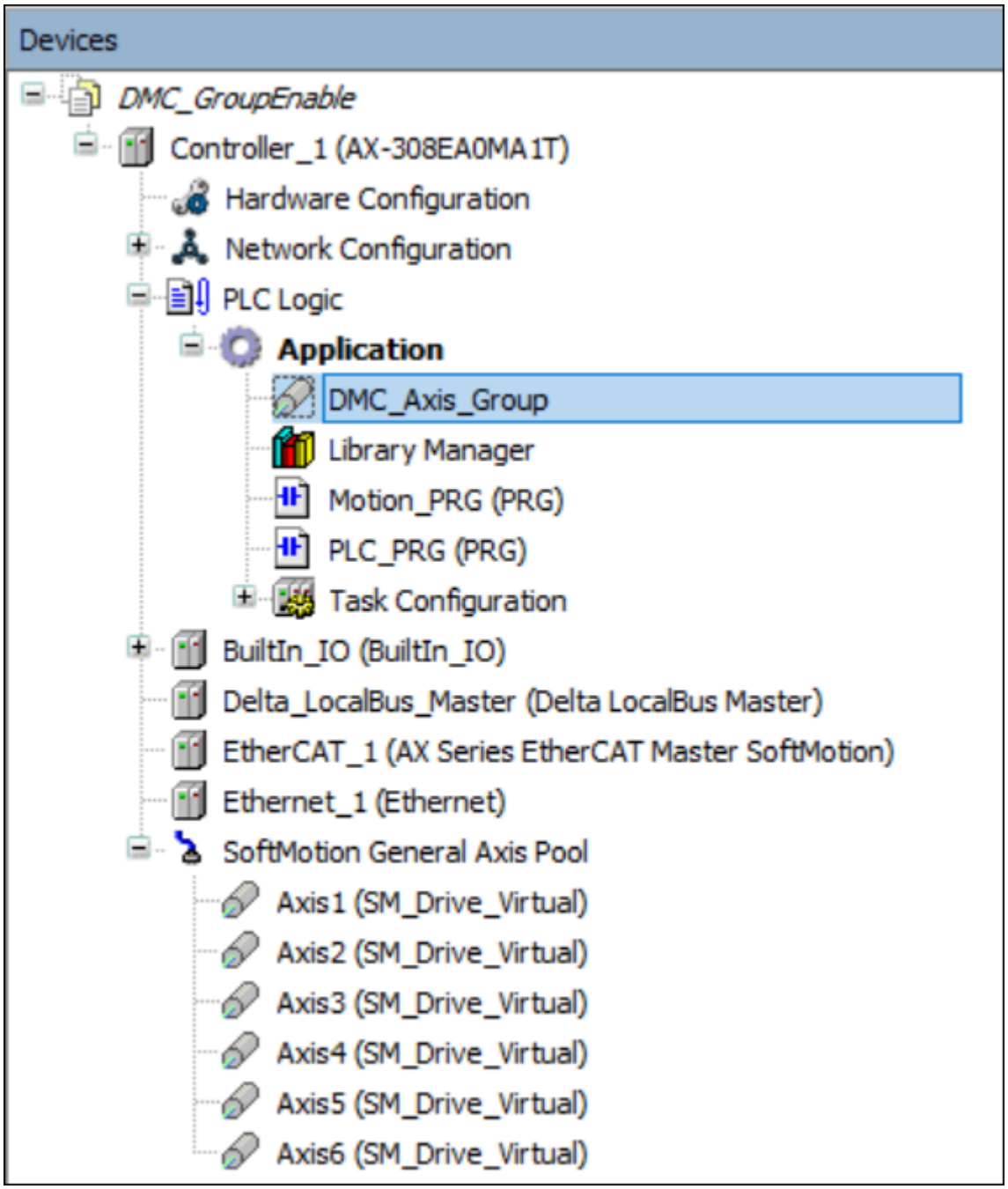
- **Inputs/Outputs**

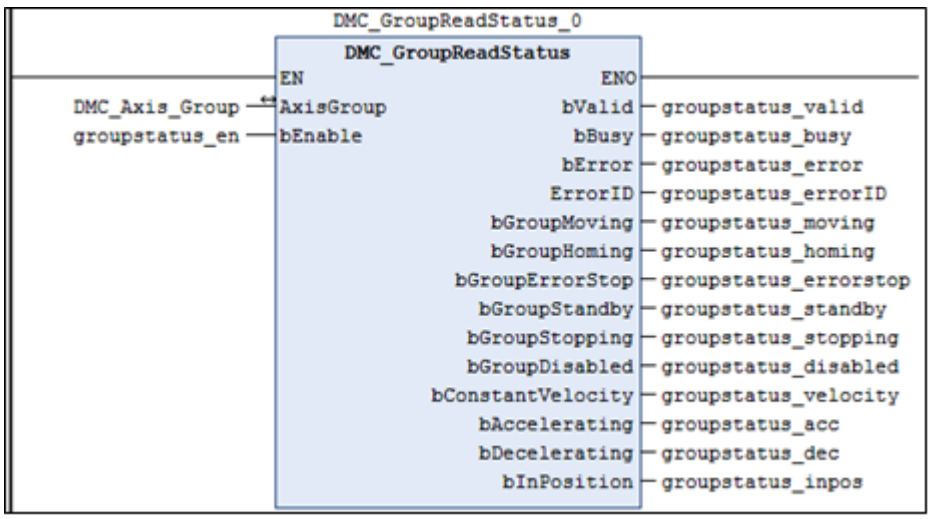
Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*bEnable* turns to TRUE and *bBusy* is FALSE.

***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**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and corresponding troubleshooting, refer to **Appendix** of this manual.
- **Example**

  The example shows how to use DMC_GroupReadStatus to read the current state of the specified axis group.


Devices
DMC_GroupEnable
Controller_1 (AX-308EA0MA1T)
Hardware Configuration
Network Configuration
PLC Logic
Application
DMC_Axis_Group
Library Manager
Motion_PRG (PRG)
PLC_PRG (PRG)
Task Configuration
BuiltIn_IO (BuiltIn_IO)
Delta_LocalBus_Master (Delta LocalBus Master)
EtherCAT_1 (AX Series EtherCAT Master SoftMotion)
Ethernet_1 (Ethernet)
SoftMotion General Axis Pool
Axis1 (SM_Drive_Virtual)
Axis2 (SM_Drive_Virtual)
Axis3 (SM_Drive_Virtual)
Axis4 (SM_Drive_Virtual)
Axis5 (SM_Drive_Virtual)
Axis6 (SM_Drive_Virtual)


DMC_GroupReadStatus_0
DMC_GroupReadStatus
EN
ENO
DMC_Axis_Group
AxisGroup
bValid
groupstatus_valid
groupstatus_en
bEnable
bBusy
groupstatus_busy
bError
groupstatus_error
ErrorID
groupstatus_errorID
bGroupMoving
groupstatus_moving
bGroupHoming
groupstatus_homing
bGroupErrorStop
groupstatus_errorstop
bGroupStandby
groupstatus_standby
bGroupStopping
groupstatus_stopping
bGroupDisabled
groupstatus_disabled
bConstantVelocity
groupstatus_velocity
bAccelerating
groupstatus_acc
bDecelerating
groupstatus_dec
bInPosition
groupstatus_inpos

1. Add **DMC_Axis_Group** in **Device**.
2. 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.

- **Supported Devices**
  - AX-series motion controller

2.2.2.6 DMC_GroupReadError

DMC_GroupReadError reads axis group errors.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupReadError		DMC_GroupReadError_instance(AxisGroup: = , bEnable: = , bValid=> , bBusy=> , bError=> , ErrorID=> , GroupErrorID=>) ;

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	Runs the instruction 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 running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
GroupErrorID	When the axis group is in ErrorStop state, the output shows an error code for the current axis group. Refer to **Appendix** for error code descriptions.	DMC_ERROR	DMC_ERROR (DMC_NO_ERROR)

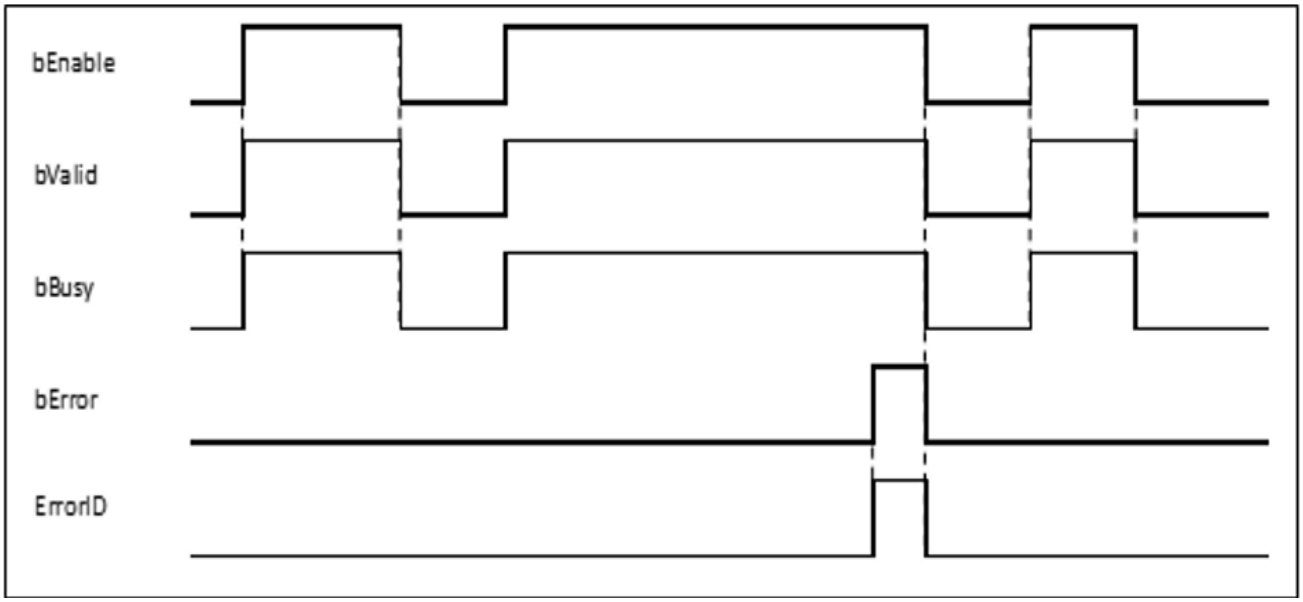
***Note**: DMC_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bValid	*bEnable* is TRUE and the output value is valid.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bEnable* is TRUE.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bError (ErrorID)	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID* and axis group error code is recorded in *GroupErrorID*)	*bEnable* turns from TRUE to FALSE. (Both error codes are cleared)
GroupErrorID	*bEnable* is TRUE and the output keeps updating.	*bEnable* is TRUE and the output keeps updating.

- **Timing Diagram of Output Parameter Changes**



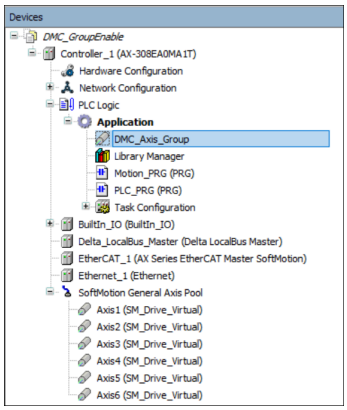
- **Inputs/Outputs**

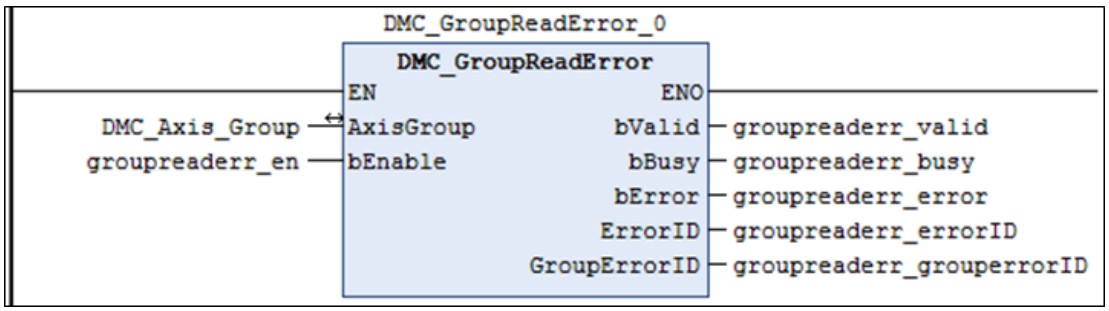
Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified the axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*bEnable* turns to TRUE and *bBusy* is FALSE

***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**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and troubleshooting, refer to **Appendix** in this manual.
- **Example**

  The example shows how to use DMC_GroupReadError to read an axis group error after the axis group is created.





1. Add **DMC_Axis_Group** in **Device**.
2. When groupreaderr_en (*bEnable*) and groupreaderr_valid (*bValid*) changes to TRUE, DMC_GroupReadError reads the state of the axis group **DMC_Axis_Group** via its output.

- **Supported Devices**
  - AX-series motion controller

2.2.2.7 DMC_GroupReset

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupReset		DMC_GroupReset_instance(AxisGroup: = , bExecute: = , bDone=> , bBusy=> , bError=> , ErrorID=>) ;

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

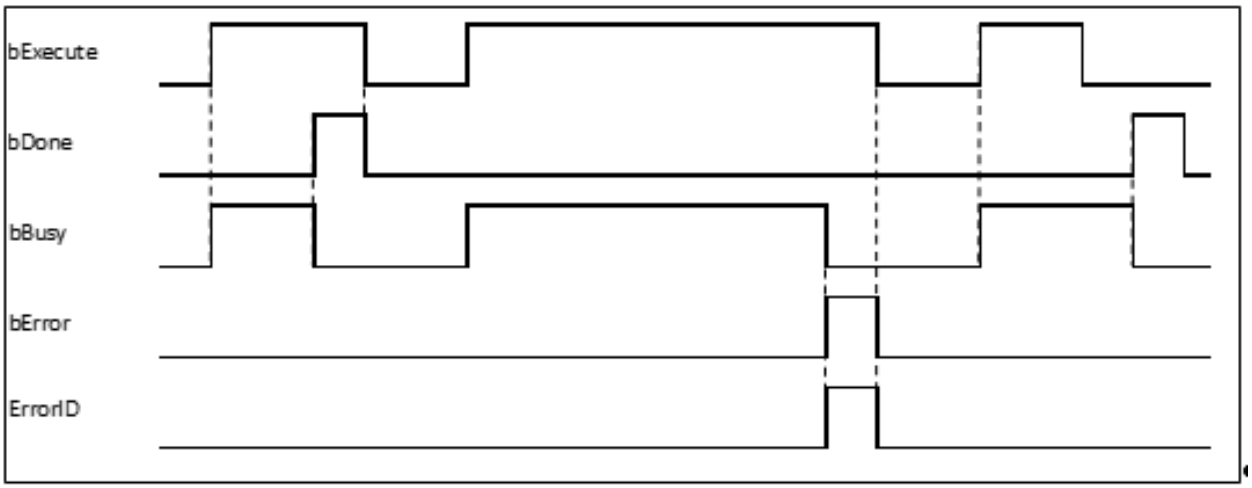
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the instruction is complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs in running of the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The axis group errors is cleared.	• *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	*bExecute* changes to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError (ErrorID)	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	*bExecute* turns from TRUE to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

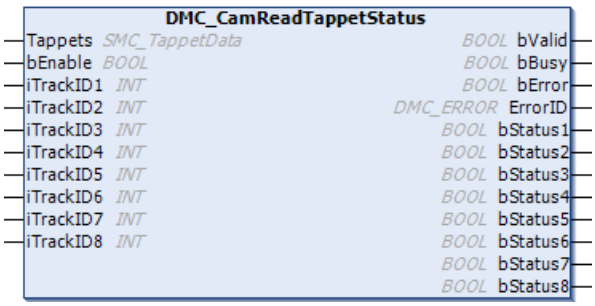
Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*bExecute* turns to TRUE and *bBusy* is FALSE.

***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, the axis group motion can be performed.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and troubleshooting, refer to **Appendix** of this manual.
- **Supported Devices**
  - AX-series motion controller

2.2.2.8 DMC_CamReadTappetStatus

DMC_CamReadTappetStatus reads the status of multiple tappets.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_CamReadTappetStatus		DMC_CamReadTappetStatus_instance(Tappets : =, bEnable : =, iTrackID1 : =, iTrackID2 : =, iTrackID3 : =, iTrackID4 : =, iTrackID5 : =, iTrackID6 : =, iTrackID7 : =, iTrackID8 : =, bValid =>, bBusy =>, bError =>, ErrorID =>, bStatus1 =>, bStatus2 =>, bStatus3 =>, bStatus4 =>, bStatus5 =>,

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (TRUE)	-
iTrackID1	The tappet number	INT	0–512 (0) *	*bEnable* is TRUE.
iTrackID2	The tappet number	INT	0–512 (0)	*bEnable* is TRUE.
iTrackID3	The tappet number	INT	0–512 (0)	*bEnable* is TRUE.
iTrackID4	The tappet number	INT	0–512 (0)	*bEnable* is TRUE.
iTrackID5	The tappet number	INT	0–512 (0)	*bEnable* is TRUE.
iTrackID6	The tappet number	INT	0–512 (0)	*bEnable* is TRUE.
iTrackID7	The tappet number	INT	0–512 (0)	*bEnable* is TRUE.
iTrackID8	The tappet number	INT	0–512 (0)	*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 running	BOOL	TRUE/FALSE (FALSE)

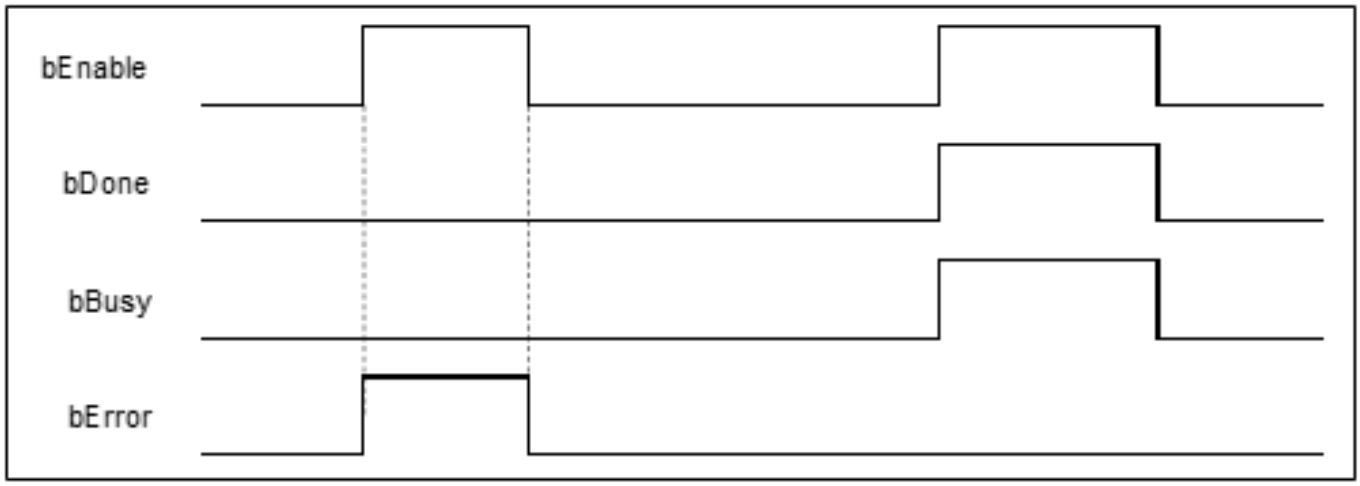
Name	Function	Data Type	Output Range (Default Value)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
bStatus1	The status of the tappet specified by iTrackID1	BOOL	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)
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	*bEnable* turns to TRUE.	• *bError* turns to TRUE. • *bEnable* turns to FALSE.
bBusy	*bEnable* turns to TRUE.	*bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* turns to FALSE.
ErrorID		
bStatus1	The status of the specified tappet is TRUE.	The status of the specified tappet is FALSE.
bStatus2	The status of the specified tappet is TRUE.	The status of the specified tappet is FALSE.
bStatus3	The status of the specified tappet is TRUE.	The status of the specified tappet is FALSE.
bStatus4	The status of the specified tappet is TRUE.	The status of the specified tappet is FALSE.
bStatus5	The status of the specified tappet is TRUE.	The status of the specified tappet is FALSE.
bStatus6	The status of the specified tappet is TRUE.	The status of the specified tappet is FALSE.
bStatus7	The status of the specified tappet is TRUE.	The status of the specified tappet is FALSE.
bStatus8	The status of the specified tappet is TRUE.	The status of the specified tappet is FALSE.

- **Timing Diagram of Output Parameter Changes**



- **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

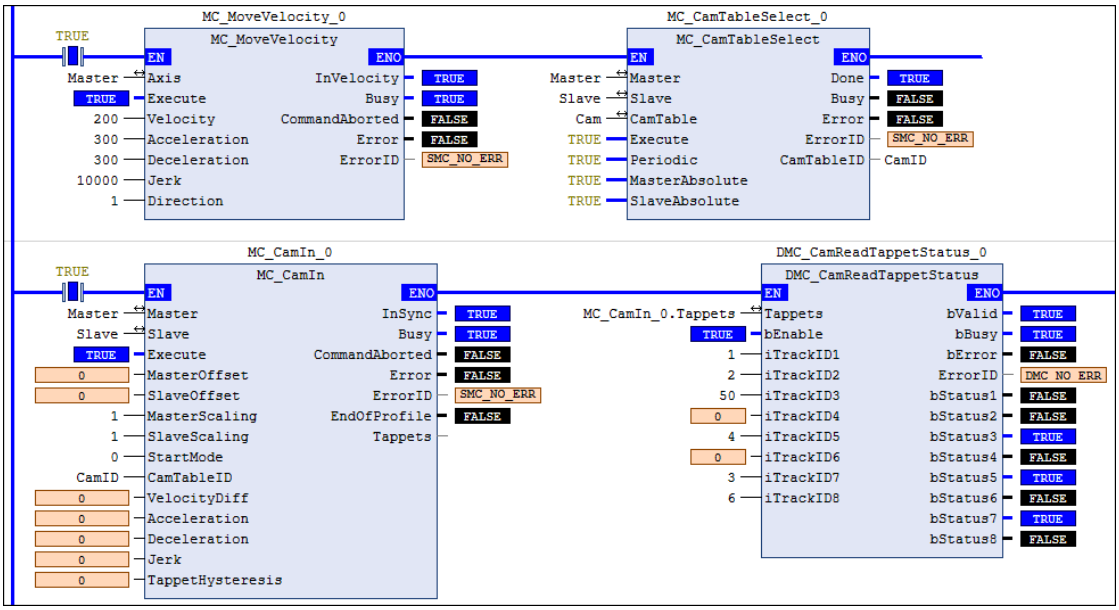
***Note**: SMC_TappetData (STRUCT): the internal interface between MC_CamIn and SMC_GetTappetValue for tappet data transmission.

Name	Function	Data Type	Setting Value (Default Value)
ctt	The direction through which the tappet point will pass before the action occurs.	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	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	The delay time before the tappet changes to ON in the TAPPETACTION_time mode	DWORD	Positive or 0 (0)
dwDuration	The duration for the tappet is ON in the TAPPETACTION_time mode.	DWORD	Positive or 0 (0)
iGroupID	The track ID of the tappet	INT	Positive, negative, or 0 (0)
x	The tappet position	LREAL	Positive, negative, or 0 (0)
dwActive	The internal variable	DWORD	Positive or 0 (0)

- **Function**
  - The instruction allows you 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**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  The example explains how CamReadTappetStatus works.

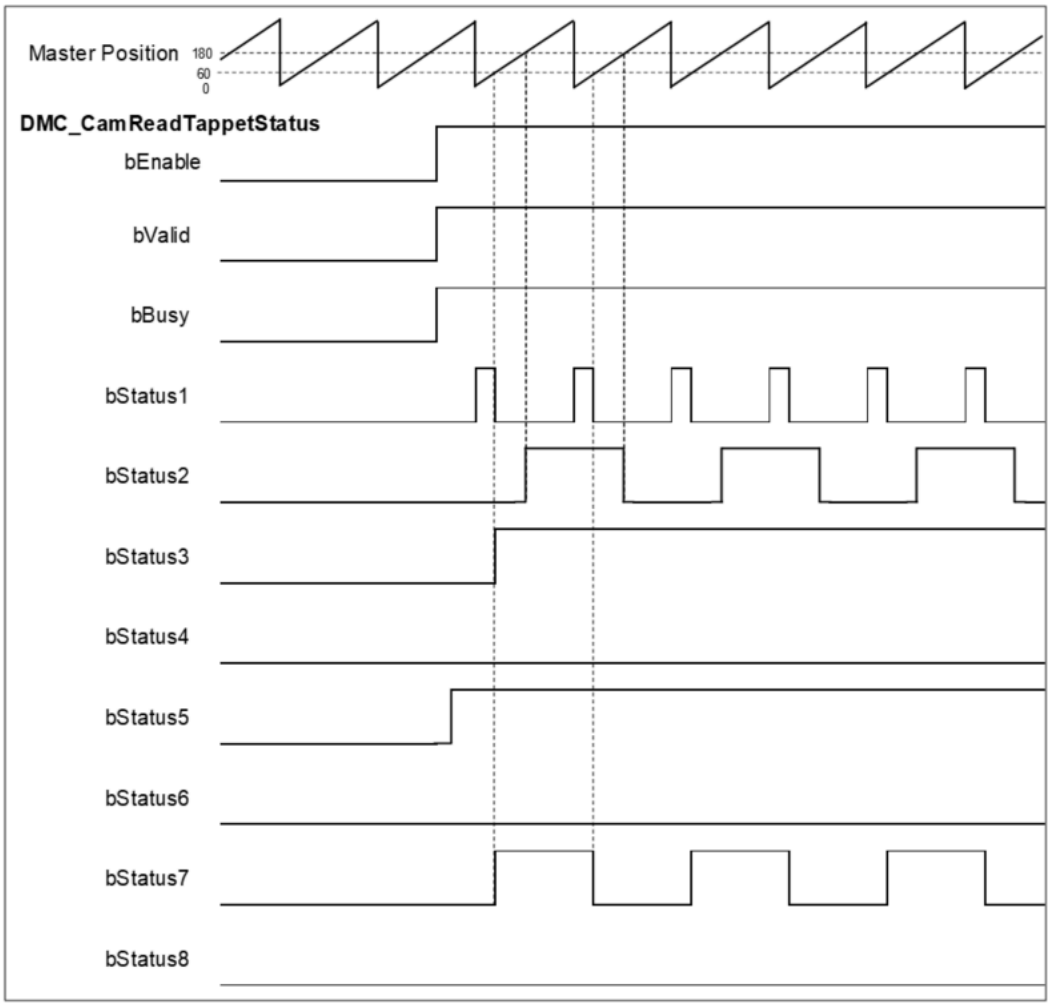
  - Input 1 for iTrackID1, 2 for iTrackID2, 50 for iTrackID3, 4 for iTrackID4, 3 for iTrackID7. No tappet numbers are provided for iTrackID4, iTrackID6, and iTrackID8, so they will not take action.



  - 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

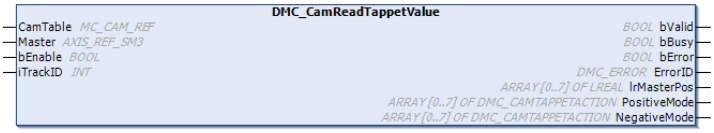


- When *bEnable* changes to TRUE, DMC_CamReadTappetStatus starts to update the status of tappets.
- Take the second output point (*bStatus2*) for example. The corresponding tappet ID is 2, and its action is reversed when the position reaches 180.
- When *bEnable* changes to FALSE, the outputs keep the current status of tappets.

- **Supported Devices**
  - AX-series motion controller

2.2.2.9 DMC_CamReadTappetValue

DMC_CamReadTappetValue reads the data of a single tappet.

FB/ FC	Instruction	Graphic Expression	ST Language
FB	DMC_CamReadTappetValue		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	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (TRUE)	-
iTrackID	The ID of the Track to be read	INT	1–512 (0)	*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 running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR[*1]	DMC_ERROR (DMC_NO_ERROR)
lrMasterPos	Specify the tappet position relates to the master axis (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)

Name	Function	Data Type	Output Range (Default Value)
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)

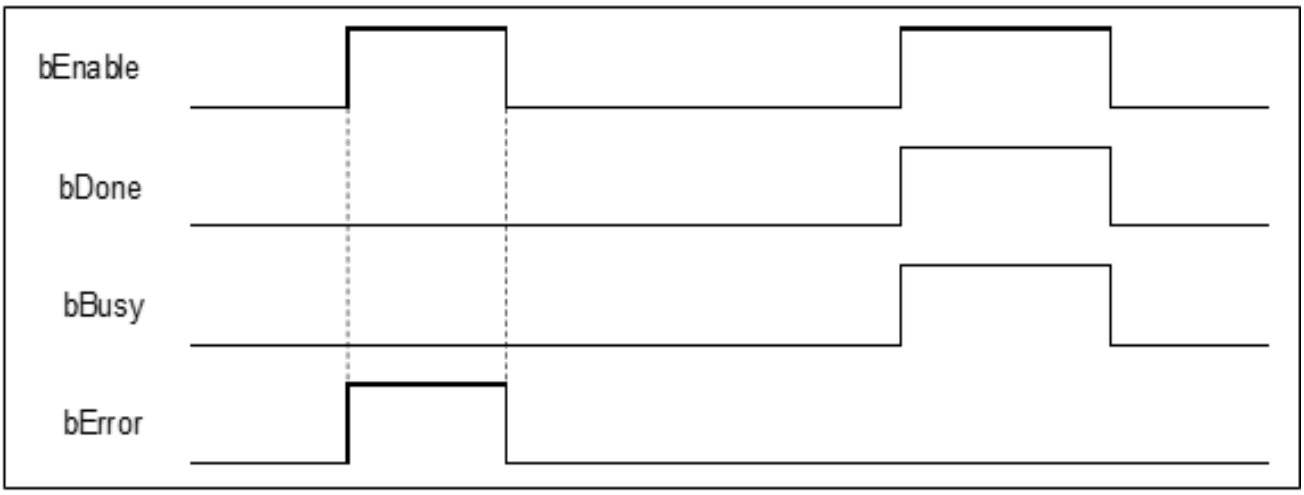
***Note**:

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 *IrMasterPos* is set to 0 and *PositiveMode* and *NegativeMode* are set to TAPPETACTION_none.

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bValid	*bEnable* turns to TRUE.	• *bError* turns to TRUE. • *bEnable* turns to FALSE.
bBusy	*bEnable* turns to TRUE.	*bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* turns to FALSE.
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **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

Name	Function	Data Type	Setting Value	Timing to Take Effect
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 parameter.

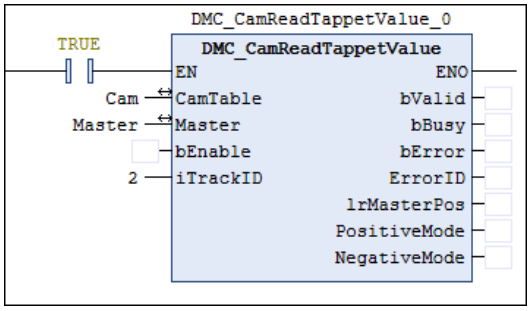
- **Function**
  - A tappet table can be set in the Cam table. Multiple tappet points can be set for one Track. The function block can read up to 8 tappet switching points in the same tappet path.
  - The tappet data contains the master axis position corresponding to the tappet point, the positive passing mode and the negative passing mode. The modes include TAPPETACTION_none, TAPPETACTION_on, TAPPETACTION_off, TAPPETACTION_inv and TAPPETACTION_time
  - The first point of the same position has data only for negative and the second point has data only for positive. In this case, the two points will be displayed as one point.
  - The following table lists the description of each mode.

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**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

  The example explains how DMC_CamReadTappetValue works by reading the tappet point 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

- **Supported Devices**
  - AX-series motion controller

2.2.2.10 DMC_CamWriteTappetValue

DMC_CamWriteTappetValue modifies the tappet data in an existing tappet track.

FB/ FC	Instruction	Graphic Expression	ST Language
FB	DMC_ CamWriteTappetValue		DMC_CamWriteTappetValue_instance(CamTable : =, bExecute : =, iTrackID: =, lrMasterPosition : =, PositiveMode : =, NegativeMode : =, bDone =>, bBusy =>, bError =>, ErrorID =>) ;

- **Inputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (TRUE)	-
iTrackID	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	The mode for the tappet point when it has passed in the positive 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
NegativeMode	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	TRUE when the output is valid	BOOL	TRUE/FALSE (FALSE)

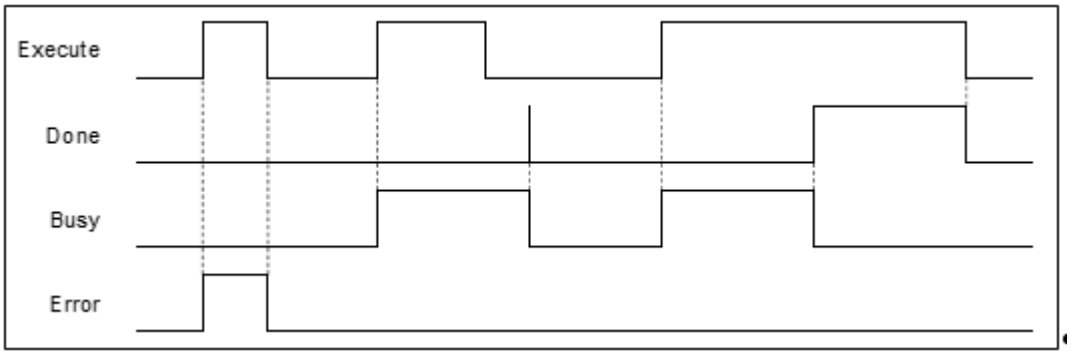
Name	Function	Data Type	Output Range (Default Value)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

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

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The instruction is complete.	• *bError* turns to TRUE. • *bExecute* turns to FALSE.
bBusy	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns from TRUE to FALSE.

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF*	MC_CAM_REF	*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 turned on 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.

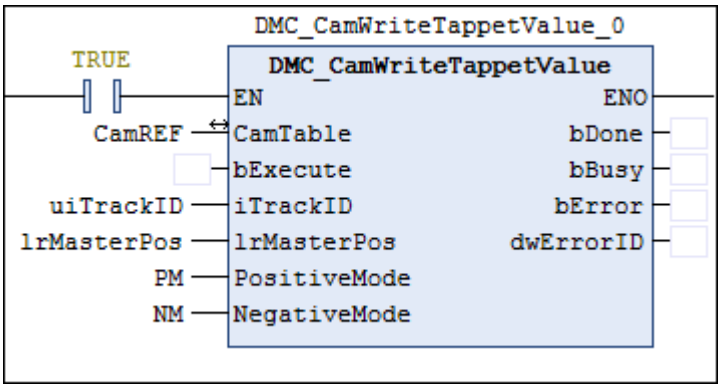
***Note**: When the mode is set to TAPPETACTION_time in this function block, the tappet switch will turn off after being ON for a fixed 100 ms.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  The example explains how DMC_CamWriteTappetValue works 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 running the function block

Track ID	Master axis position	Direction	Passing 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

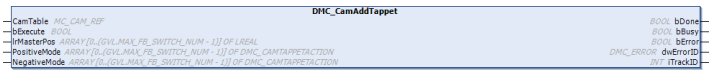
- The tappet table after running the function block

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

- **Supported Devices**
  - AX-series motion controller

2.2.2.11 DMC_CamAddTappet

DMC_CamAddTappet adds a new tappet track at the end of the tappet table.

FB/ FC	Instruction	Graphic Expression	ST Language
FB	DMC_CamAddTappet		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	Runs the instruction 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)	*bExecute* turns from FALSE to TRUE.
PositiveMode	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)	*bExecute* turns from FALSE to TRUE.
NegativeMode	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)	*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 running	BOOL	TRUE/FALSE (FALSE)

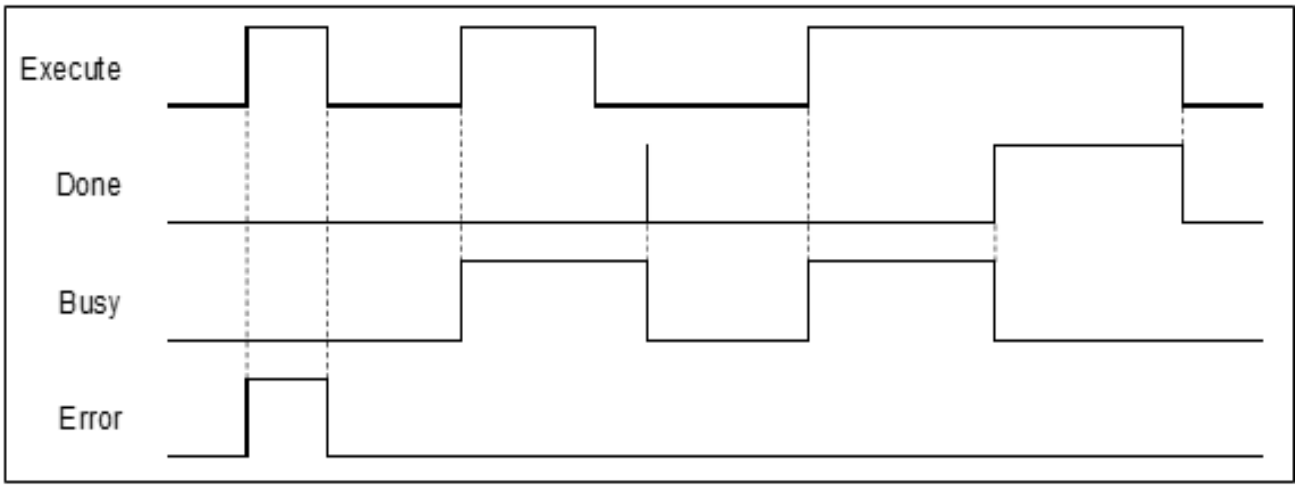
Name	Function	Data Type	Output Range (Default Value)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
iTrackID	The Track ID of the new tappet track	INT	1–512

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

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The instruction is completed.	• *bError* is TRUE. • *bExecute* turns to FALSE.
bBusy	*bExecute* is TRUE.	• *bDone* is TRUE. • *bError* is TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns from TRUE to FALSE.

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF	MC_CAM_REF*	*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.
  - Tappet path
    - One tappet path includes one track ID, one tappet switch (Boolean signal), and multiple tappet points.
  - Tappet
    - One tappet point includes the Track ID of the tappet path where the tappet point is, the master axis position corresponding to the tappet point, the positive passing mode and the negative passing mode.

      The following table list the description of each mode.

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 turned on 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.

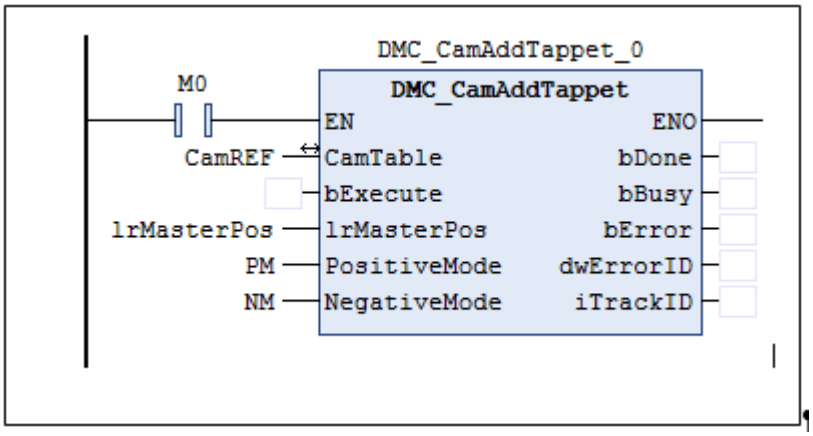
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  The example explains how DMC_CamAddTappetValue works 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 running the function block.

Track ID	Master axis position	Direction	Mode
1	100	Negative	TAPPETACTION_off
1	100	Positive	TAPPETACTION_on

Track ID	Master axis position	Direction	Mode
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

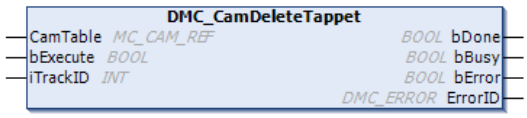
- See the tappet table after running the function block.

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

- **Supported Devices**
  - AX-series motion controller

2.2.2.12 DMC_CamDeleteTappet

DMC_CamDeleteTappet deletes the specified tappet track.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_CamDeleteTappet		DMC_CamDeleteTappet_instance(CamTable : =, bExecute : =, iTrackID : =, bDone =>, bBusy =>, bError =>, ErrorID =>) ;

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (TRUE)	-
iTrackID	The ID of the track to be deleted	INT	1–512 (0)	*bExecute* is TRUE.

- **Outputs**

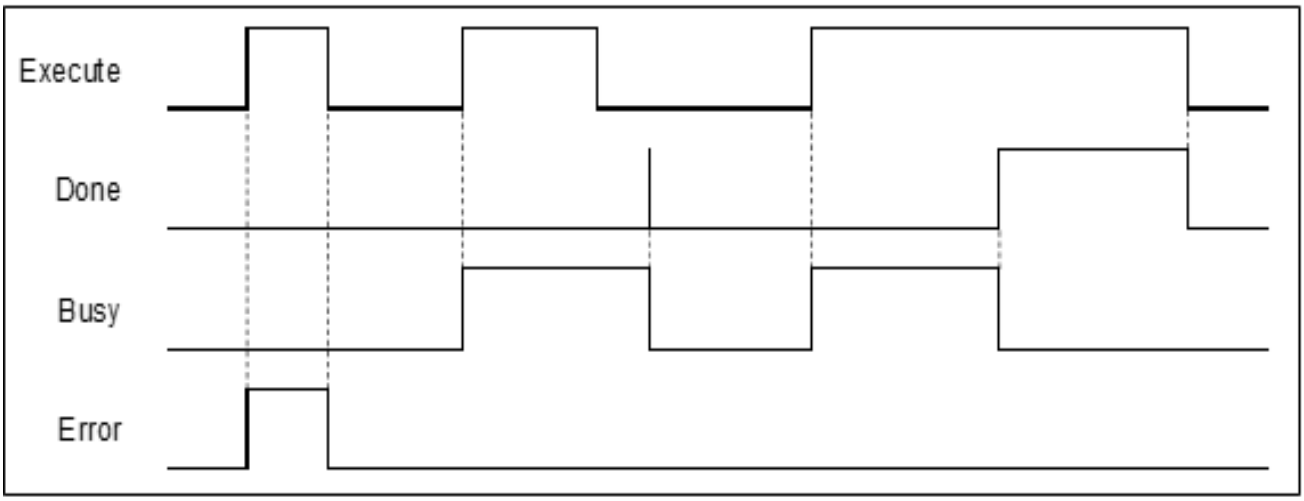
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the instruction is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The instruction is complete.	• *bError* is TRUE. • *bExecute* turns to FALSE.
bBusy	*bExecute* is TRUE.	• *bDone* is TRUE. • *bError* is TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns from TRUE to FALSE.
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF	MC_CAM_REF*	*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 turned on 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.

Mode	Function	Action
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.

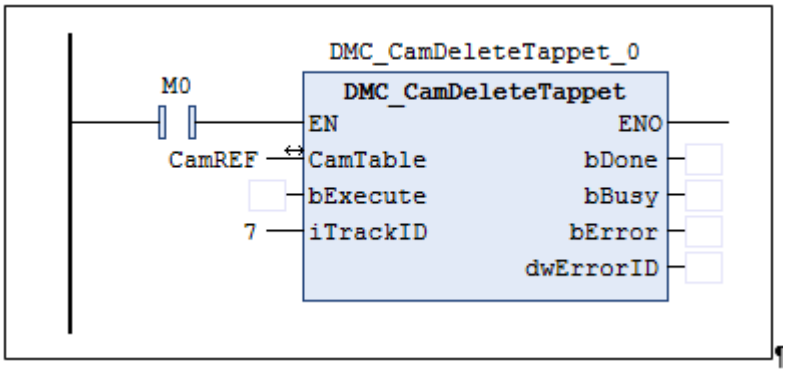
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  The example explains how DMC_CamDeleteTappet works 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 running the function block.

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
7	7000	Positive	TAPPETACTION_inv
2	0	Negative	TAPPETACTION_off

Track ID	Master axis position	Direction	Mode
2	0	Positive	TAPPETACTION_on

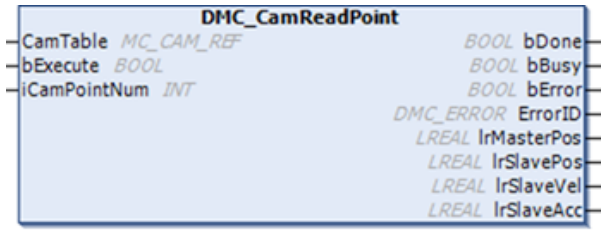
- See the tappet table after running the function block

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

- **Supported Devices**
  - AX-series motion controller

2.2.2.13 DMC_CamReadPoint

DMC_CamReadPoint reads the data of a single cam point.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_CamReadPoint		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (TRUE)	-
iCamPointNum	The number of the cam point to be read	INT	0–256 (0)	When *bExecute* is TRUE

- **Outputs**

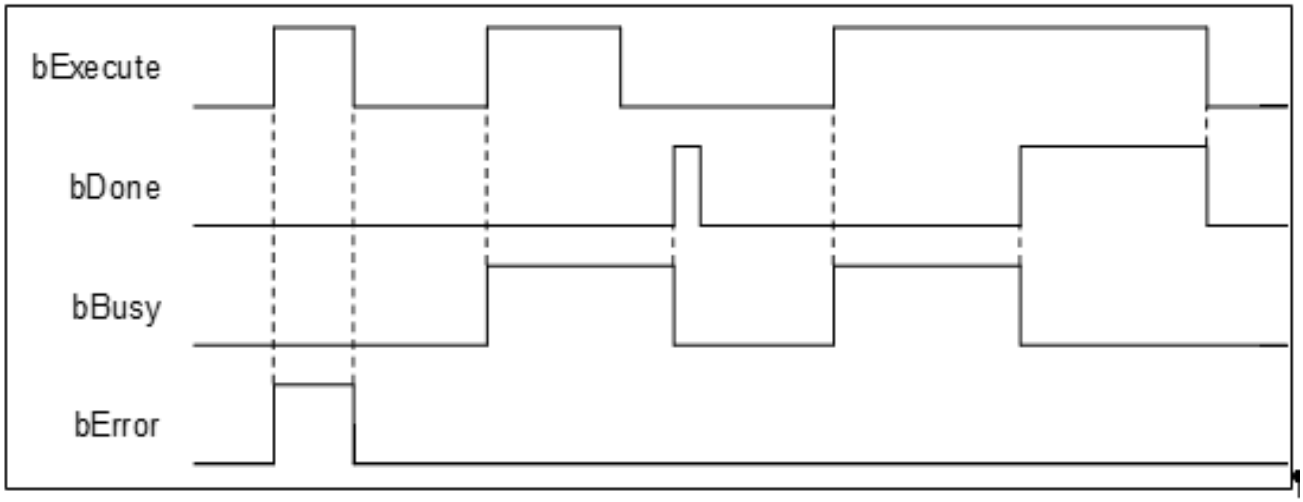
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the instruction is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	*bExecute* changes to TRUE.	• *bError* turns to TRUE. • *bExecute* turns to FALSE.
bBusy	*bExecute* changes to TRUE.	*bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE.

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

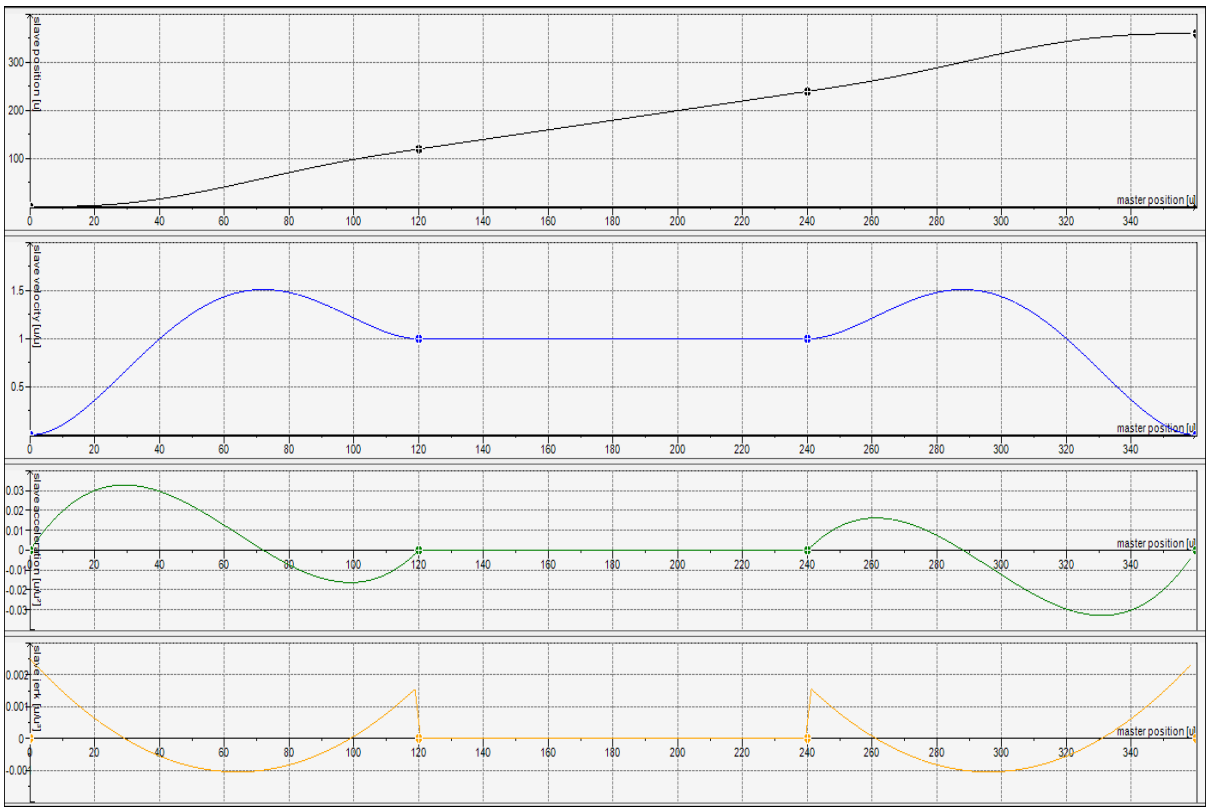
Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF	MC_CAM_REF*	*bExecute* changes to TRUE.

***Note**: MC_CAM_REF (FB): The basic CAM.

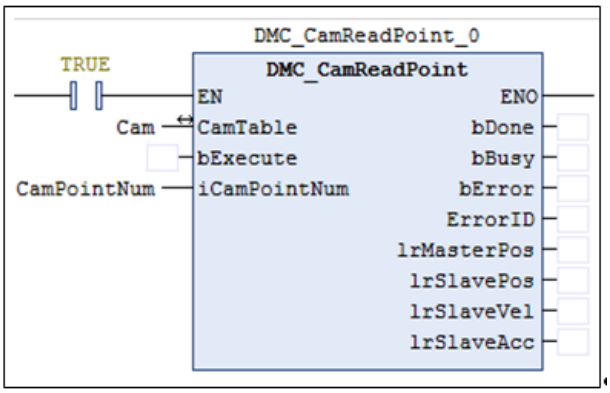
- **Function**
  - CamTable sets the cam table is to read. *iCamPointNum* sets the number of the cam point to read. The output parameter *lrMasterPos* shows the corresponding master position of the cam point. *lrSlavePos* shows the corresponding slave position of the cam point, i.e., the cam point coordinate position. *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.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For error codes and troubleshooting, refer to **Appendix** for error code descriptions.
- **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					



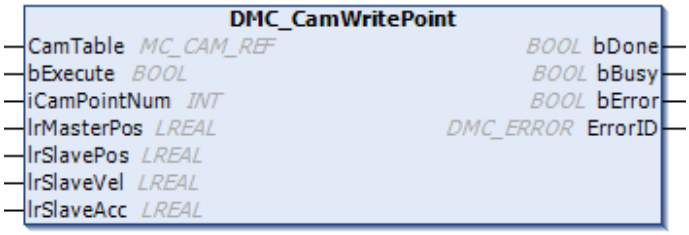
2. Input **Cam** for the specified CamTable and **2** for **CamPointNum**, the number of the cam point to be read.
3. Set *DMC_CamReadPoint_0.bExecute* to TRUE.
4. *DMC_CamReadPoint_0.lrMasterPos*: 240, *DMC_CamReadPoint_0.lrSlavePos*: 240, *DMC_CamReadPoint_0.lrSlaveVel*: 1, and *DMC_CamReadPoint_0.lrSlaveAcc*: 0 can be read while the instruction is waiting until *DMC_CamReadPoint_0.bDone* changes from FALSE to TRUE.



- **Supported Devices**
  - AX-series motion controller

2.2.2.14 DMC_CamWritePoint

DMC_CamWritePoint writes the data of a single cam point.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_CamWritePoint		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (TRUE)	-
iCamPointNum	The number of the cam point to be written.	INT	0–256 (0)	*bExecute* is TRUE.
lrMasterPos	The position of the cam master axis	LREAL	Positive, negative, or 0 (0)	*bExecute* is TRUE.
lrSlavePos	The position of the cam slave axis	LREAL	Positive, negative, or 0 (0)	*bExecute* is TRUE.
lrSlaveVel	The velocity of the cam slave axis	LREAL	Positive, negative, or 0 (0)	*bExecute* is TRUE.
lrSlaveAcc	The acceleration of the cam slave axis	LREAL	Positive, negative, or 0 (0)	*bExecute* is TRUE.

- **Outputs**

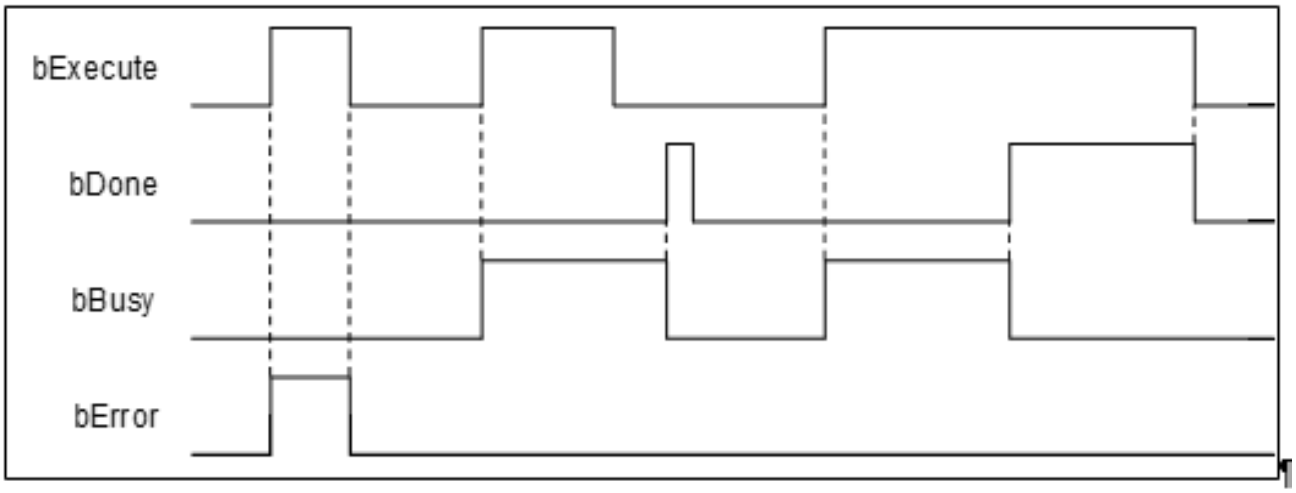
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the instruction is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	*bExecute* changes to TRUE.	• *bError* turns to TRUE. • *bExecute* turns to FALSE.
bBusy	*bExecute* changes to TRUE.	*bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE.

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF	MC_CAM_REF*	*bExecute* changes to TRUE.

***Note**: MC_CAM_REF (FB): The basic CAM.

- **Function**

  - The master axis position reduction function is supported by DL_MotionControl V1.3.4.0 or later.
  - The CamTable sets the cam table to write data. *iCamPointNum* sets the cam point number to be written. *lrMasterPos* and *lrSlavePos* respectively write 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, if the cam table does not need to access specific information (e.g.,*lrSlaveVel* or *lrSlaveAcc*), the cam operation will not be affected even if the input parameters are set.
  - 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 start or end 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 not change. MC_CamTableSelect must be restarted in order to run the cam table with the changed boundary.

- When the end cam data point is modified, and the master position written in IrMasterPos is smaller than the original cam table range, after DMC_CAMWritePoint is executed, MC_CamTableSelect must be re-executed before the master position exceeds the modified position. Otherwise, the slave axis will report the error “SMC_CI_MASTER_OUT_OF_SCALE”, and the cam synchronization stops.

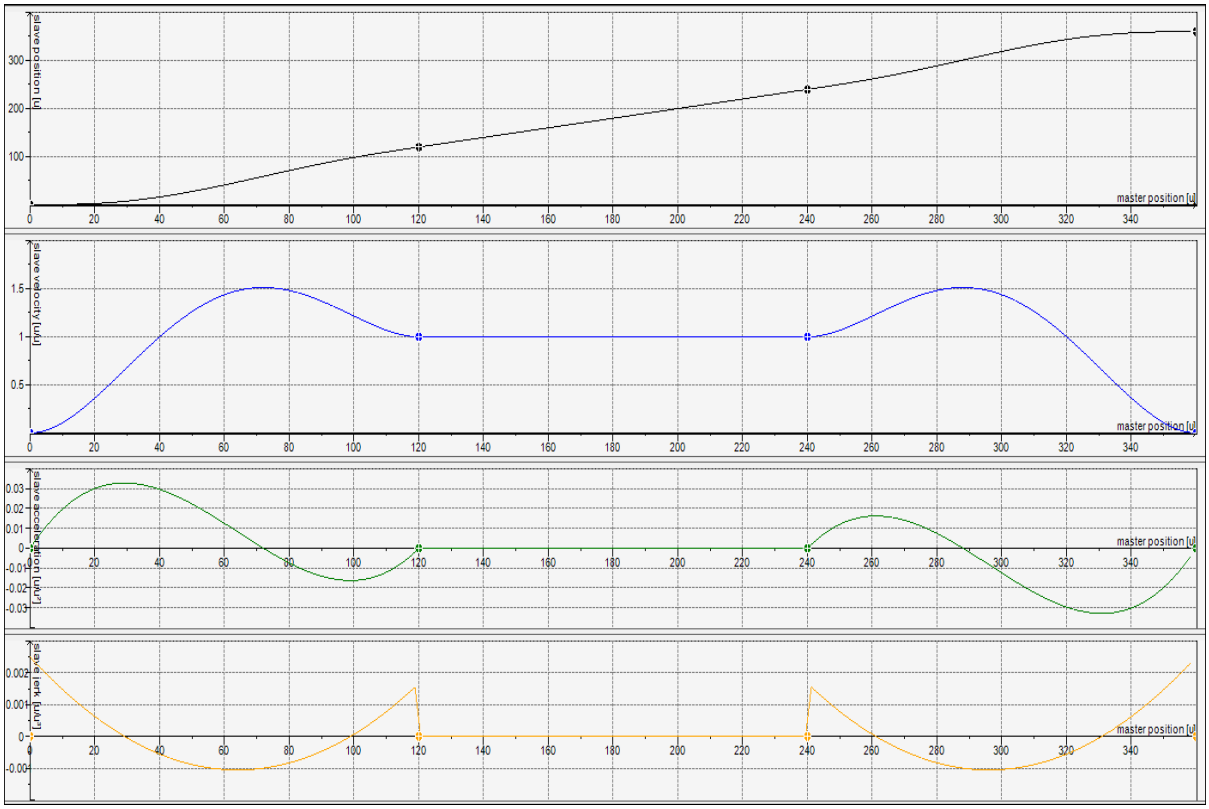
- **Troubleshooting**

  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

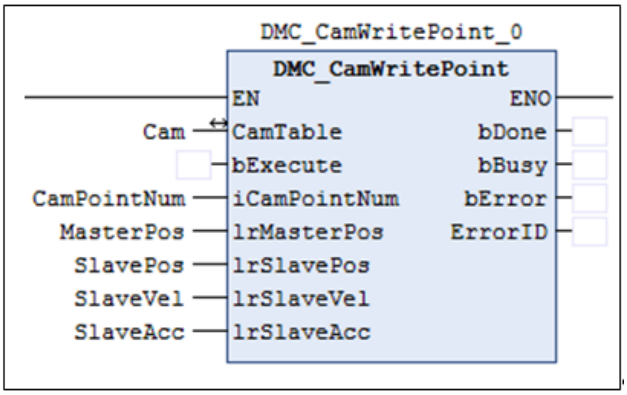
- **Example**

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. Enter **Cam** for *CamTable*, and set the data point number to be written CamPointNum = 2.
3. Enter300 for **MasterPos**, 250 for **SlavePos**, 2 for **SlaveVel**, and 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.



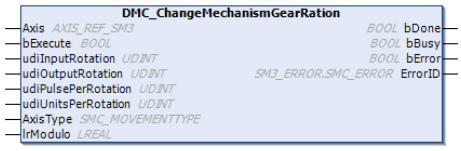
6. After the writing is finished, the actual values for the cam table "Cam" are shown as below.

	X	Y	V	A
0	0	0	0	0
1	120	120	1	0
2	**300**	**250**	**2**	**4**
3	360	360	0	0

- **Supported Devices**
  - AX-series motion controller

2.2.2.15 DMC_ChangeMechanismGearRation

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	ST Language
FB	DMC_ChangeMechanismGearRation		DMC_ChangeMechanismGearRation_instance(Axis :=, bExecute:=, udiInputRotation:=, udiOutputRotation:=, udiPulsePerRotation:=, lrUnitsPerRotation:=, AxisType:=, lrModulo:=, MotorDirection:=, bDone=>, bBusy=>, bError=>, ErrorID=>,);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
udiInputRotation	Input of the gearbox.	UDINT	Positive (0)	*bExecute* is TRUE and *bBusy* is FALSE.
lrOutputRotation	Output of the gearbox.	LREAL	Positive (0)	*bExecute* is TRUE and *bBusy* is FALSE.
udiPulsePerRotation	The number of pulses per rotation of the input end of the gearbox (Unit: pulses/ rotation)	UDINT	Positive (0)	*bExecute* is TRUE and *bBusy* is FALSE.
lrUnitsPerRotation*3	The number of units the terminal actuator travels per rotation of the output end of the gearbox	UDINT	Positive (0)	*bExecute* is TRUE and *bBusy* is FALSE.
AxisType	Axis type	SMC_MOVEMENTTYPE*1	0: rotary 1: linear (rotary)	*bExecute* is TRUE and *bBusy* is FALSE.
lrModulo	The number of units per rotation of the rotation axis.	LREAL	Positive (0)	*bExecute* is TRUE and *bBusy* is FALSE.

***Note**:

1. SMC_MOVEMENTTYPE: Enumeration (ENUM)
2. DMC _ MotorDir: Enumeration (Enum)
3. In DL_MotionControl version 1.0.2.0 and later, the *udiUnitsPerRotation* parameter changes to *lrUnitsPerRotation*.
4. DL_MotionControl version 1.1.0.0 and later supports the *MotorDirection* parameter.

- **Outputs**

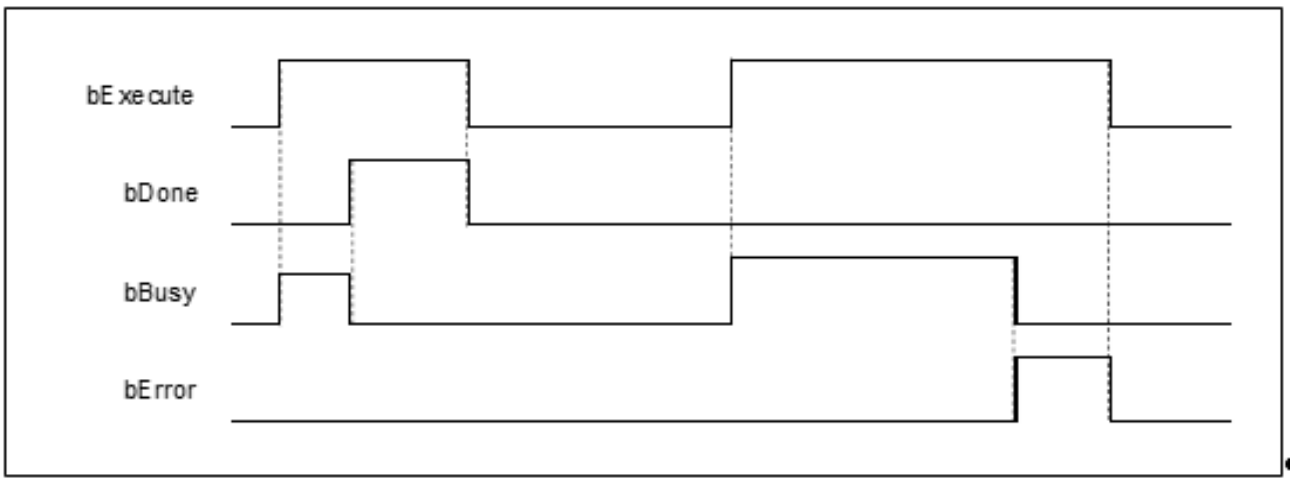
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the instruction is complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The instruction is completed.	• *bExecute* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bExecute* turns to TRUE.	• *bExecute* turns to FALSE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	*bExecute* turns from TRUE to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**

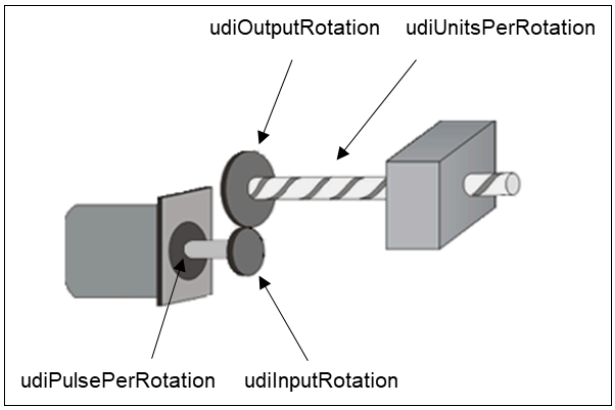


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*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 *lrUnitsPerRotation* is the number of terminal actuator travel units per rotation of the output of the gearbox.
  - This function block is available for the encoder axis.

- **Troubleshooting**

  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

The example shows the behavior of DMC_ChangeMechanismGearRation.

  - Relevant parameters setting:

    udiOutputRotation : *udiInputRotation*= 2 : 1

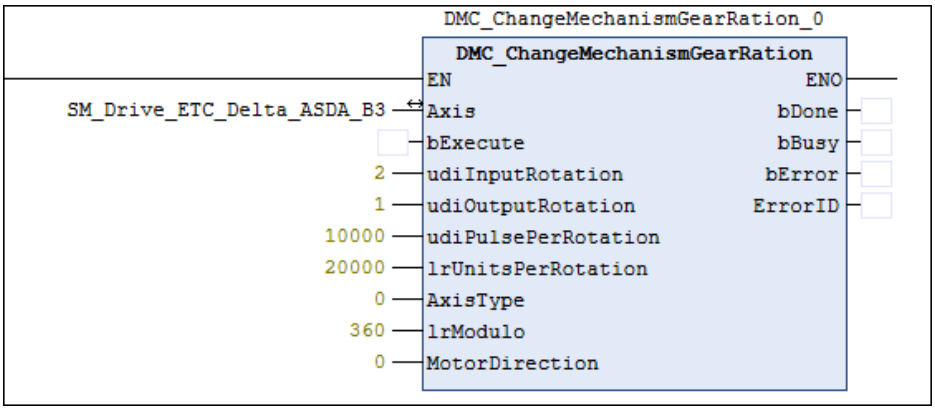
    udiPulsePerRotation: 10000 pulses

    udiUnitsPerRotation: 20000 us

    AxisType: 0 (rotary axis)

    lrModulo: 360

- Function block setting

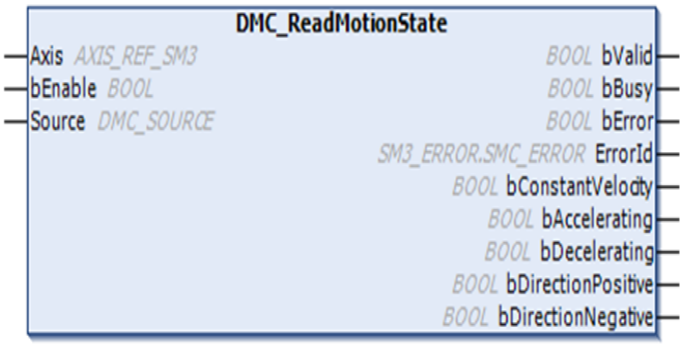


- **Supported Devices**

  - AX-series motion controller

2.2.2.16 DMC_ReadMotionState

DMC_ReadMotionState reads the behavior state of the moving axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_ReadMotionState		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	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Source	Data source. Commanded: the source is the instruction value Actual: the source is the actual value of the axis	DMC_SOURCE*	0: dmcCommandedValue (0)	*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 running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
bConstantVelocity	The current velocity is constant	BOOL	TRUE/FALSE (FALSE)
bAccelerating	The absolute value of the current velocity is increasing	BOOL	TRUE/FALSE (FALSE)
bDecelerating	The absolute value of the current velocity is decreasing	BOOL	TRUE/FALSE (FALSE)
bDirectionPositive	The current position is increasing	BOOL	TRUE/FALSE (FALSE)

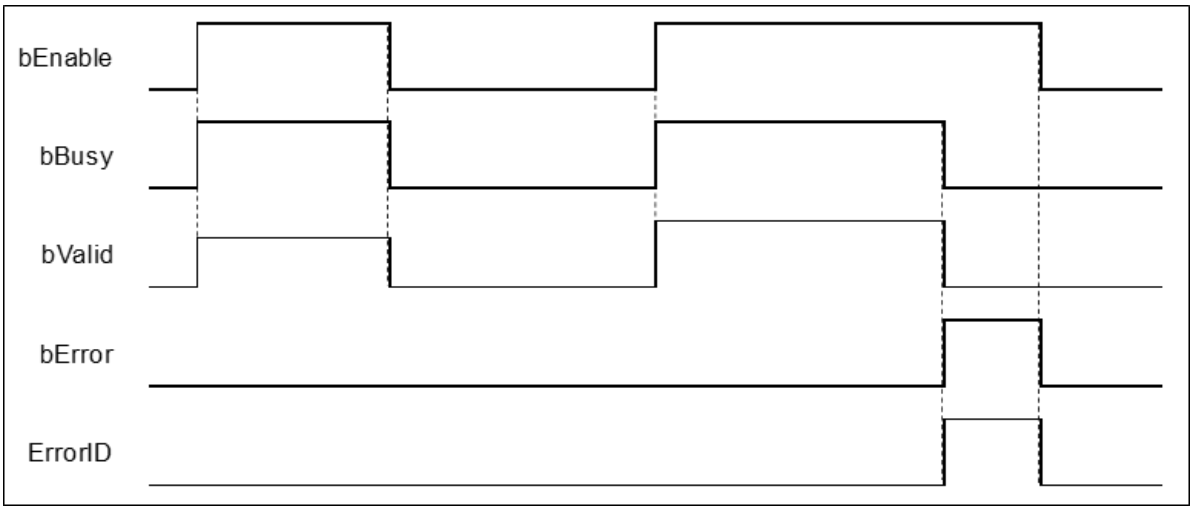
Name	Function	Data Type	Output Range (Default Value)
bDirectionNegative	The current position is decreasing	BOOL	TRUE/FALSE (FALSE)

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

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bValid	*bEnable* is TRUE and the axis motion state can be read,	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bEnable* is TRUE.	• When *bEnable* turns to FALSE • When *bError* turns to TRUE
bError ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error Code is recorded.)	*bEnable* turns to FALSE. (Clear the error code recorded in *ErrorID*.)
bConstantVelocity	The current velocity is constant	*bEnable* is TRUE but the velocity is not constant.
bAccelerating	The absolute value of the current velocity is increasing.	*bEnable* is TRUE but the velocity does not increase.
bDecelerating	The absolute value of the current velocity is decreasing.	*bEnable* is TRUE but the velocity does not decrease
bDirectionPositive	The current position is increasing.	• *bEnable* is TRUE and the direction of motion is not positive. • *bEnable* is TRUE and the axis does not move any more.
bDirectionNegative	The current position is decreasing.	• *bEnable* is TRUE and the direction of motion is not negative. • *bEnable* is TRUE and the axis does not move any more.

- **Timing Diagram of Output Parameter Changes**



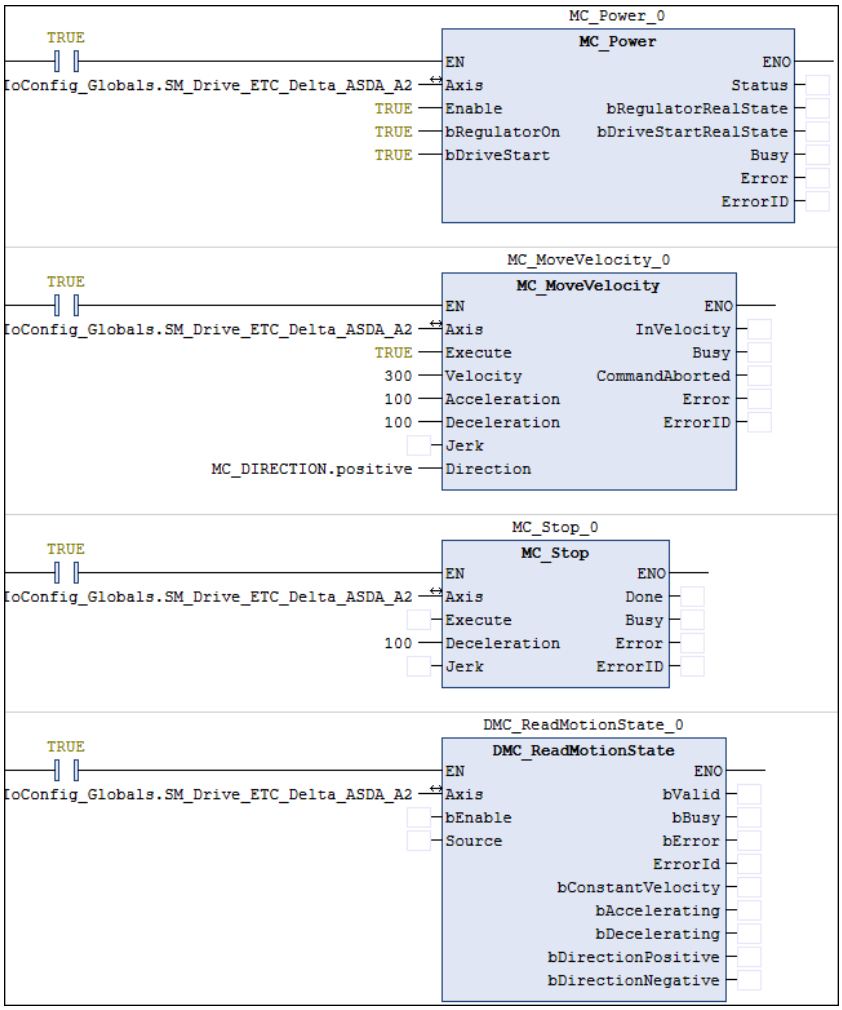
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bEnable* is TRUE and *bBusy* is FALSE.

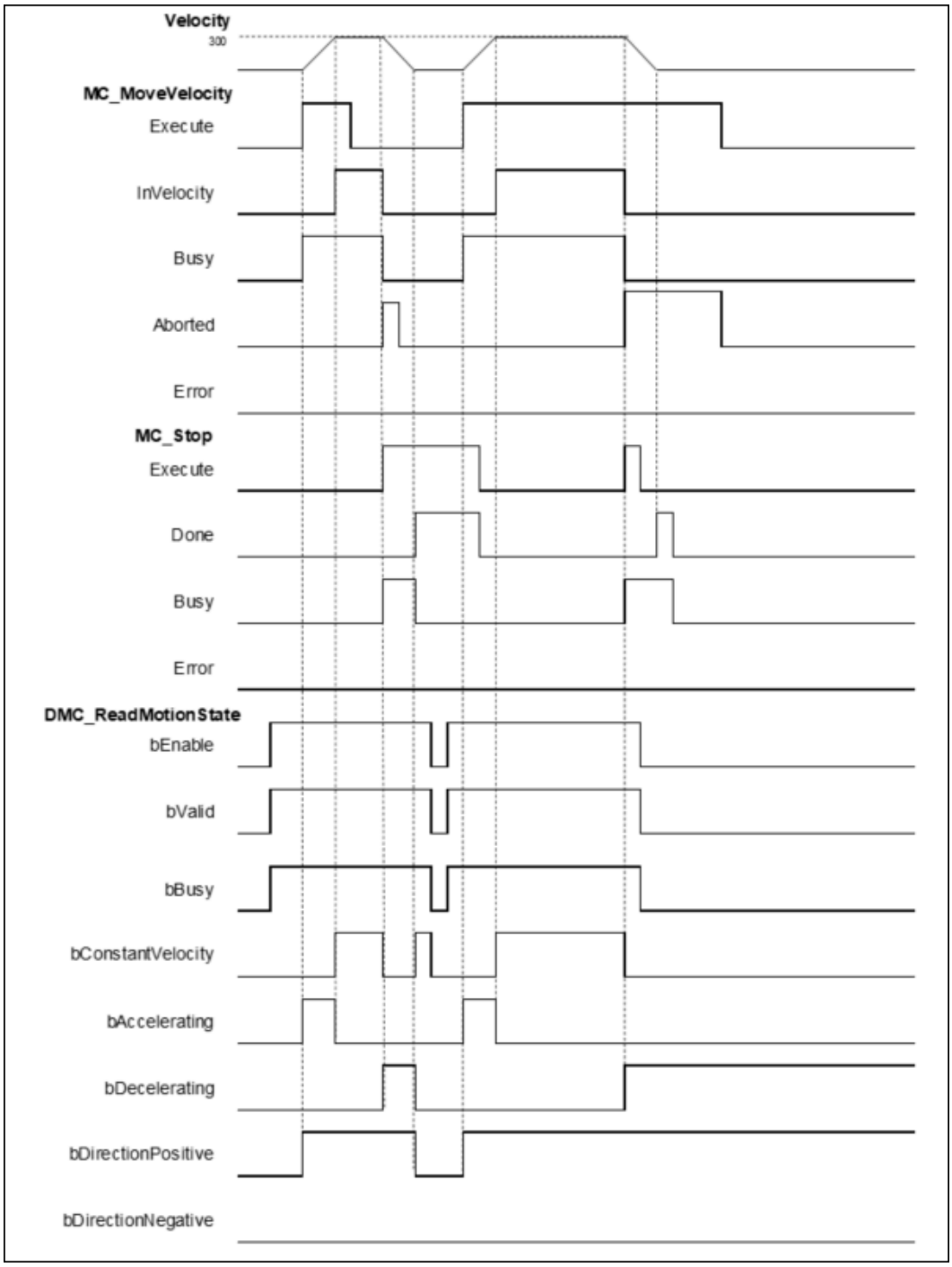
***Note**: AXIS_REF_SM3 (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**
  - DMC_ReadMotionState reads the behavior state of the moving axis. (i.e. acceleration/deceleration, constant velocity, positive/negative direction of motion).
  - When the velocity is 0, the output *bConstantVelocity* changes to TRUE.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  The example shows the motion behavior that MC_ReadMotionState reads MC_MoveVelocity and MC_Stop.


MC_Power_0
MC_Power
TRUE
EN
ENO
oConfig_Globals.SM_Drive_ETC_Delta_ASDA_A2
Axis
Status
TRUE
Enable
bRegulatorRealState
TRUE
bRegulatorOn
bDriveStartRealState
TRUE
bDriveStart
Busy
Error
ErrorID
MC_MoveVelocity_0
MC_MoveVelocity
TRUE
EN
ENO
oConfig_Globals.SM_Drive_ETC_Delta_ASDA_A2
Axis
InVelocity
TRUE
Execute
Busy
300
Velocity
CommandAborted
100
Acceleration
Error
100
Deceleration
ErrorID
Jerk
MC_DIRECTION.positive
Direction
MC_Stop_0
MC_Stop
TRUE
EN
ENO
oConfig_Globals.SM_Drive_ETC_Delta_ASDA_A2
Axis
Done
Execute
Busy
100
Deceleration
Error
Jerk
ErrorID
DMC_ReadMotionState_0
DMC_ReadMotionState
TRUE
EN
ENO
oConfig_Globals.SM_Drive_ETC_Delta_ASDA_A2
Axis
bValid
bEnable
bBusy
Source
bError
ErrorId
bConstantVelocity
bAccelerating
bDecelerating
bDirectionPositive
bDirectionNegative

- Timing Diagram



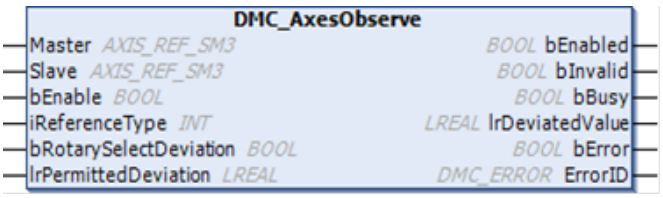
1. When *bValid* and *bBusy* change to TRUE after DMC_ReadMotionState has started, the motion state can be read.
2. The axis starts to accelerate until the target velocity after MC_MoveVelocity has started. When *bAccelerating* and *bDirectionPositive* change to TRUE, it indicates that the axis is accelerating in the positive direction.
3. When the axis reaches the target velocity, the instruction keeps 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.

- **Supported Devices**
  - AX-series motion controller

2.2.2.17 DMC_AxesObserve

DMC_AxesObserve monitors the deviation between the master axis position and slave axis position. The function block will output the error when the deviation exceeds the set value,

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_AxesObserve		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	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
iReferenceType	Position type	INT	0: Command position 1: Actual position (0)	*bEnable* is TRUE.
bRotarySelect Deviation	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)	*bEnable* is TRUE.
lrPermitted Deviation	The permitted deviation between the two axes	LREAL	Positive or 0 (0)	*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 running	BOOL	TRUE/FALSE (FALSE)
lrDeviatedValue	The error value between the two axes	LREAL	Positive, negative, or 0 (0)

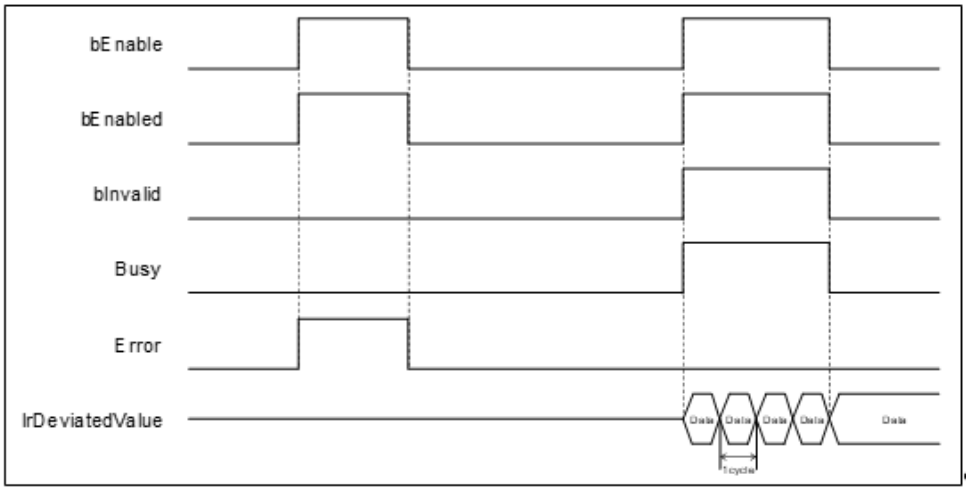
Name	Function	Data Type	Output Range (Default Value)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bEnabled	*bEnable* turns to TRUE.	• *bError* turns to TRUE. • *bEnable* turns to FALSE.
bInvalid	• *bEnable* turns to TRUE. • The deviation between the two axes exceeds the allowed value.	• *bError* turns to TRUE. • *bEnable* turns to FALSE.
bBusy	*bEnable* turns to TRUE.	• *bError* turns to TRUE. • *bEnable* turns to FALSE.
lrDeviatedValue	*bEnable* turns to TRUE.	*bEnable* turns to FALSE, stop updating data.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* turns to FALSE.
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	The specified master axis	AXIS_REF_SM3	AXIS_REF_SM3*	*bEnable* turns to TRUE.
Slave	The specified 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.

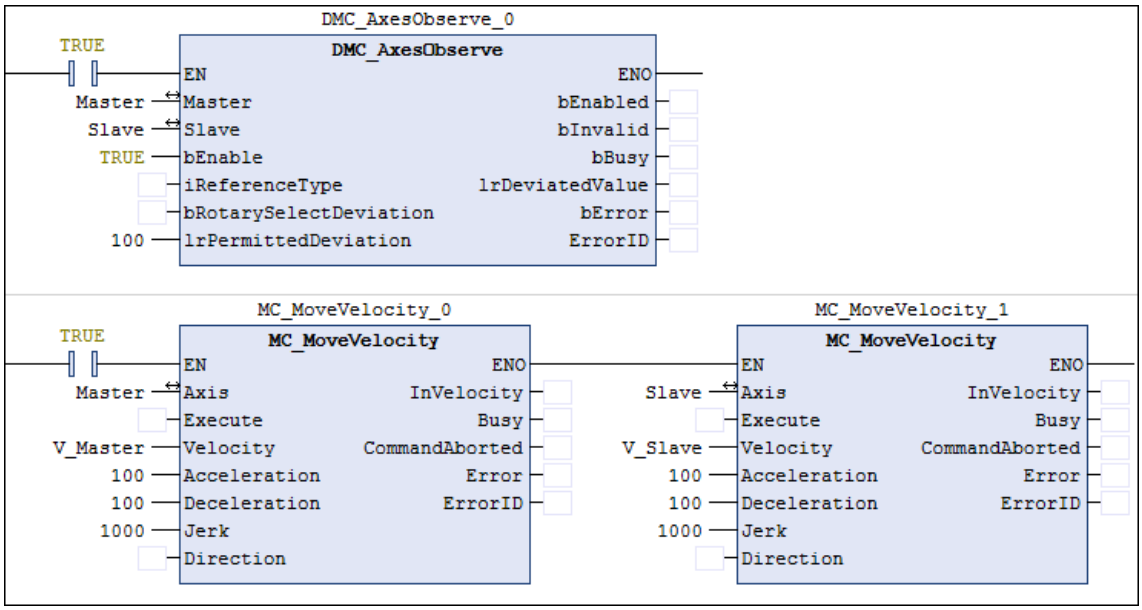
- **Function**
  - DMC_AxesObserve checks if the deviation between the master axis position and slave axis position exceeds the set value.

    When │*lrDeviatedValue*│ is > *lrPermittedDeviation* and *bInvalid* turns to TRUE.
  - It is recommended to set the same mode for the master and slave axes. If one of them is a rotary axis and the other is a linear axis, the deviation will be calculated as a linear axis.
  - When both the master axis and slave axis are rotary axes, but their distances per rotation are different, the deviation will be calculated in the 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 while TRUE means to read the longer direction.
  - The following is the calculation method of *lrDeviatedValue*.

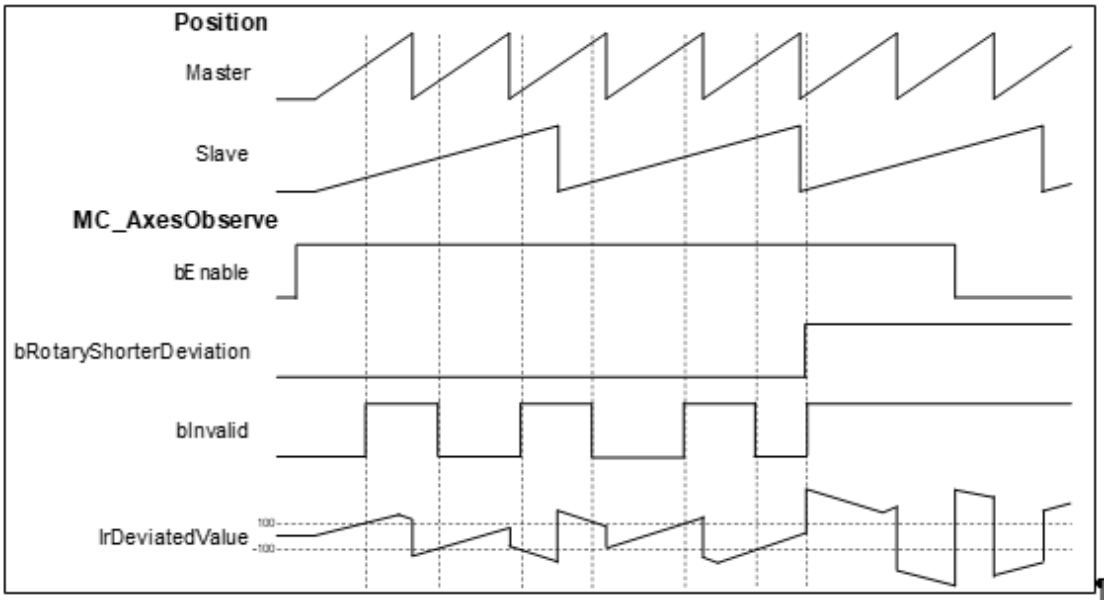
Master axis mode	Slave axis mode	Calculation method
Linear axis	Linear axis	*lrDeviatedValue* (Error between axes) = Master axis position–Slave axis position
Rotary axis	Linear axis	
Linear axis	Rotary axis	
Rotary axis	Rotary axis	*lrDeviatedValue* (Error between axes) = Master axis position–Slave axis 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 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 position of the master axis < the current position of the slave axis, the sign of *lrDeviatedValue* (Error between axes) is negative (–).

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **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 velocity of the master axis is faster than the slave axis. 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 set value 100 of *lrPermittedDeviation*, *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. Since the *lrPermittedDeviation* is set to 100, *bInvalid* must be TRUE.

- **Supported Devices**
  - AX-series motion controller

2.2.2.18 DMC_PositionLag

DMC_PositionLag sets the range of the lag error and check whether the position lag is exceeded.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_PositionLag		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	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
eStopMode	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)	*bEnable* and *bBusy* are TRUE.
fMaxPositionLag	The maximum lag error value	LREAL	Positive or 0 (0)	*bEnable* and *bBusy* are TRUE.
fSetActTimeLagCycles	The lag cycle between command value and actual value	LREAL	Positive or 0 (3)	*bEnable* and *bBusy* are TRUE.

- **Outputs**

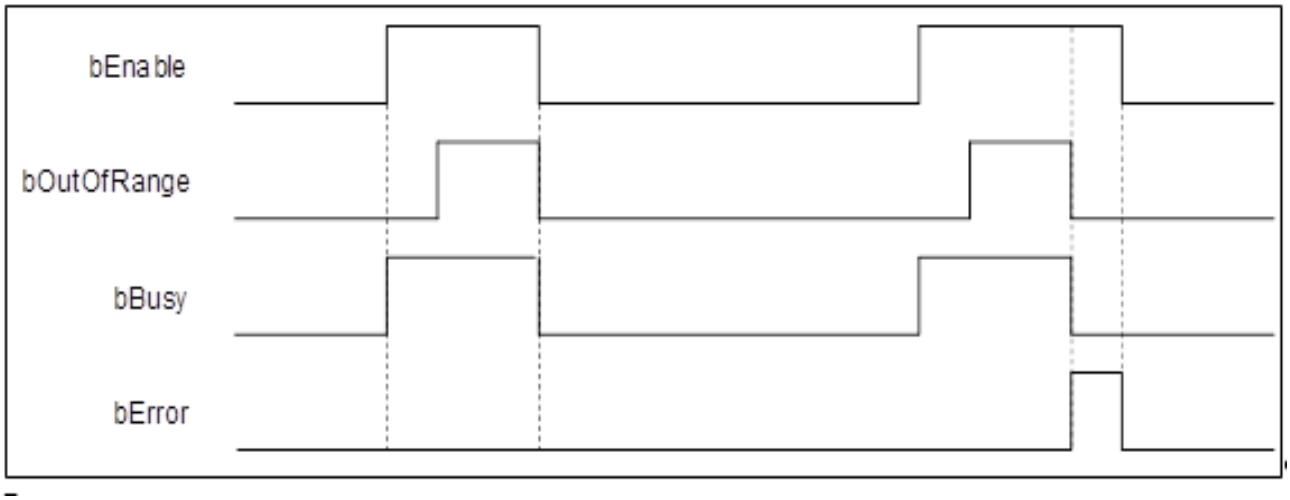
Name	Function	Data Type	Output Range (Default Value)
bOutOfRange	TRUE when LagTime exceeds the setting value	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
lrPosLag	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	LagTime exceeds the setting value.	*bEnable* turns to FALSE.
bBusy	The instruction is running	*bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* turns to FALSE. (Error code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*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.

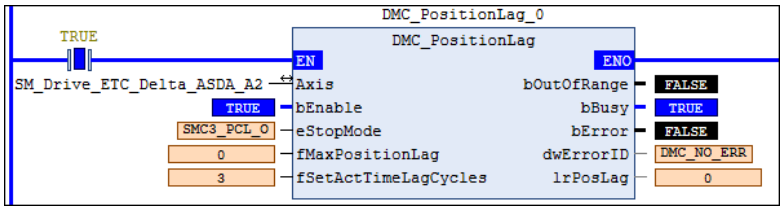
- SMC3_PCL_HALT: When *LagError* is out of the allowed range, the axis parameter *bDriveStart* changes to FALSE.
- SMC3_PCL_ENABLE: When *LagError* is out of the allowed range and the axis stops, there are no change for *bRegulatorOn* and *bDriveStart* states.

- Calculation and Judgement of *LagError*
  - ｜(*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*.

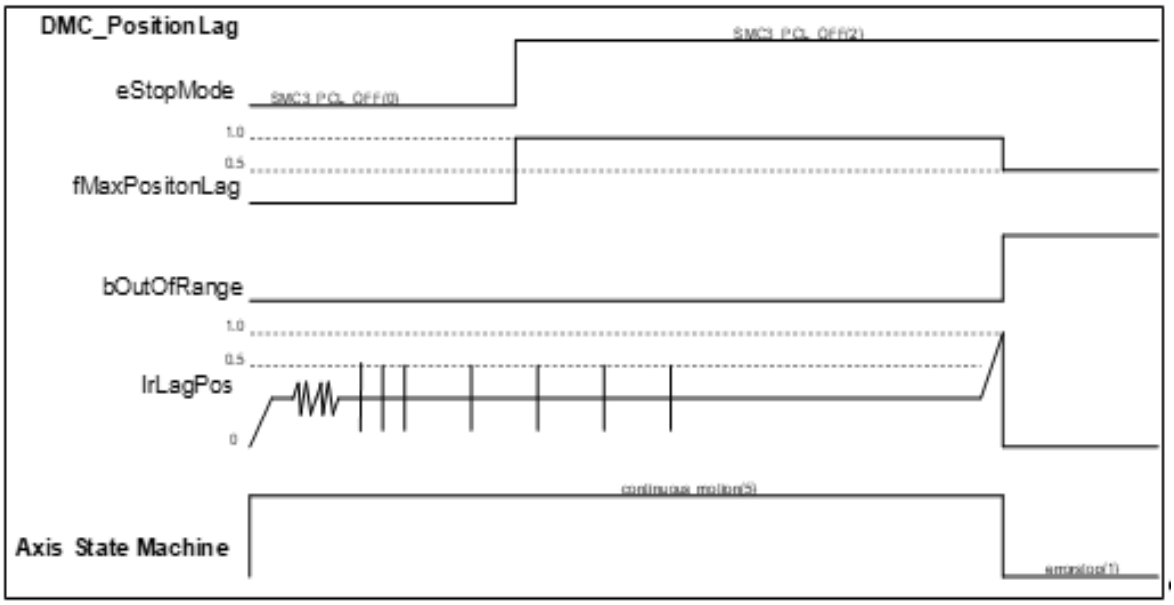
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **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.

- **Supported Devices**
  - AX-series motion controller

2.2.2.19 DMC_SetTorqueLimit

DMC_SetTorqueLimit sets the maximum torque of an axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_SetTorqueLimit		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	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrMaxTorque	The maximum rated torque (Unit: Nm)	LREAL	Positive or 0 (0)	*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 running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR(DMC_NO_ERROR)

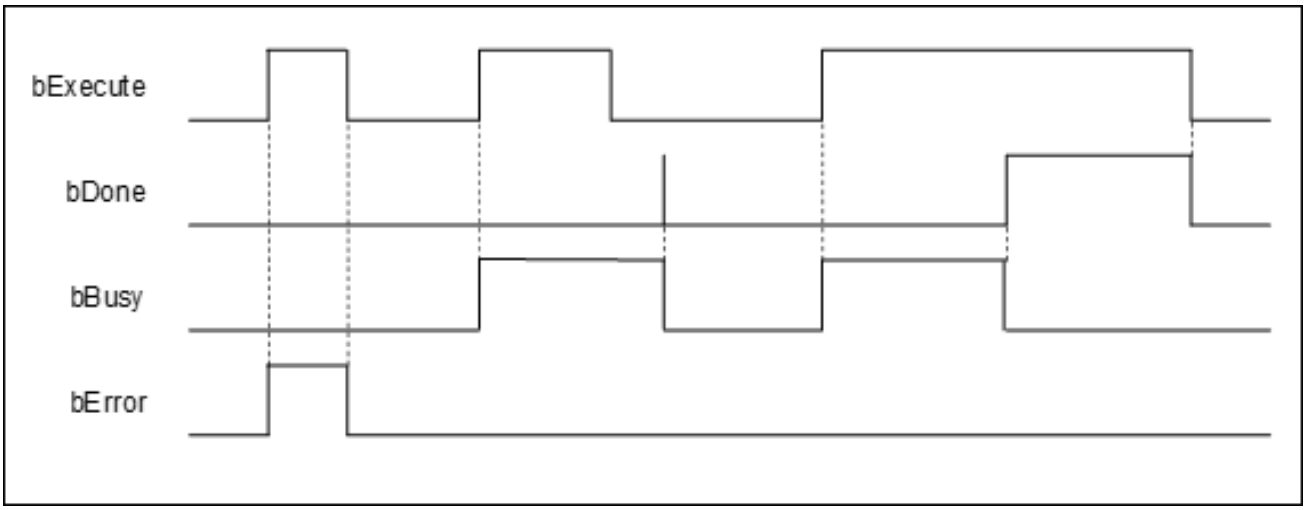
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The setting is completed.	• *bExecute* turns to FALSE. • *bDone* will change to FALSE after remaining TRUE for one period when *bExecute* is FALSE but *bDone* changes to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	*bExecute* turns to TRUE.	• When *bDone* turns to TRUE • When *bError* turns to TRUE
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns from TRUE to FALSE. (Error code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



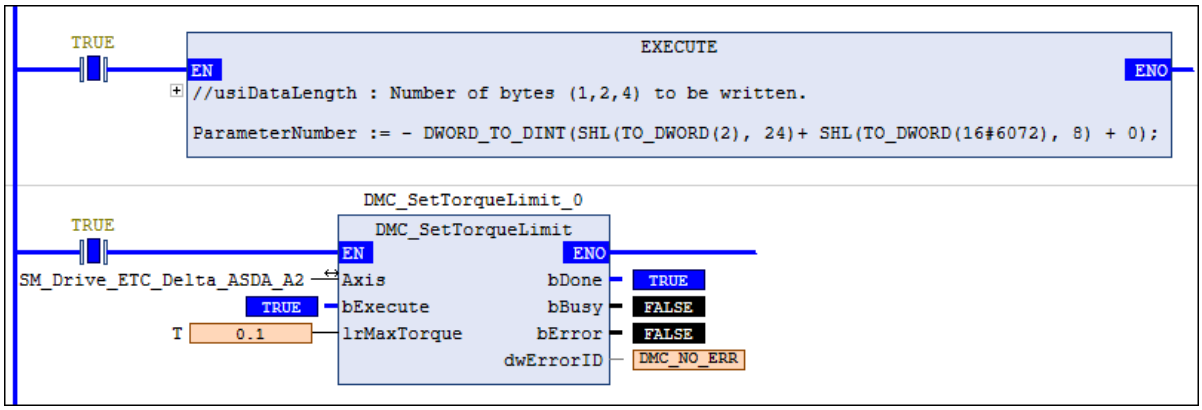
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bExecute* turns to TRUE.

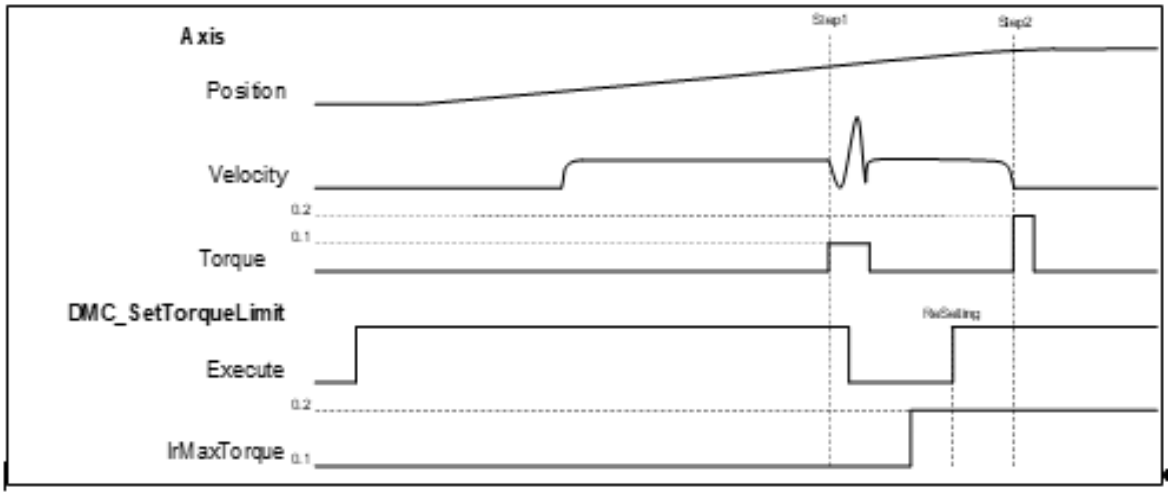
***Note**: AXIS_REF_SM3 (FB): The interface is built in every function block and works as the starting program of the function block.

- **Function**
  - This function block is not available in PLC simulation mode.
  - After the maximum torque is set, the motor's torque will be limited to the set value to protect the motor from damage caused by excessive torque when encountering large resistance.
  - DMC_SetTorqueLimit can be used to set the maximum torque of an axis in units of Nm.
  - DMC_SetTorqueLimit can be used only in CSP or CSV mode.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  The example illustrates how to use DMC_SetTorqueLimit.



- Timing Diagram

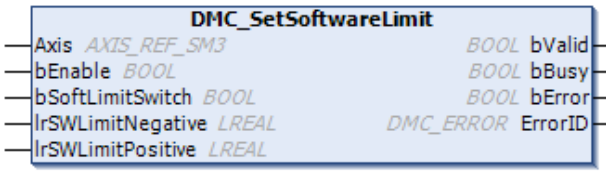


1. Set the maximum rated torque of the axis to 0.1Nm before starting the motion. Then, the instruction runs at a constant velocity.
2. Use the external force to stop the axis (Step 1) while running the instruction. The actual torque of the axis reaches 0.1 Nm and then stop the external force.
3. Set the maximum rated torque to 0.2 Nm and use the external force to stop (Step2). The actual torque of the axis reaches 0.2 Nm. The Following Error occurs in the axis, and Servo ON is disabled.

- **Supported Devices**
  - AX-series motion controller

2.2.2.20 DMC_SetSoftwareLimit

DMC_SetSoftwareLimit enables, disables, and sets the upper and lower software limits.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_SetSoftwareLimit		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	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
bSoftLimitSwitch	Turn on or turn off software limits.	BOOL	TRUE/FALSE (FALSE)	*bEnable* and *bBusy* are TRUE.
lrSWLimitNegative	Negative software limit (User unit)	LREAL	Positive, negative, or 0	*bEnable* and *bBusy* are TRUE.
lrSWLimitPositive	Positive software limit (User unit)	LREAL	Positive, negative, or 0	*bEnable* and *bBusy* are TRUE.

- **Outputs**

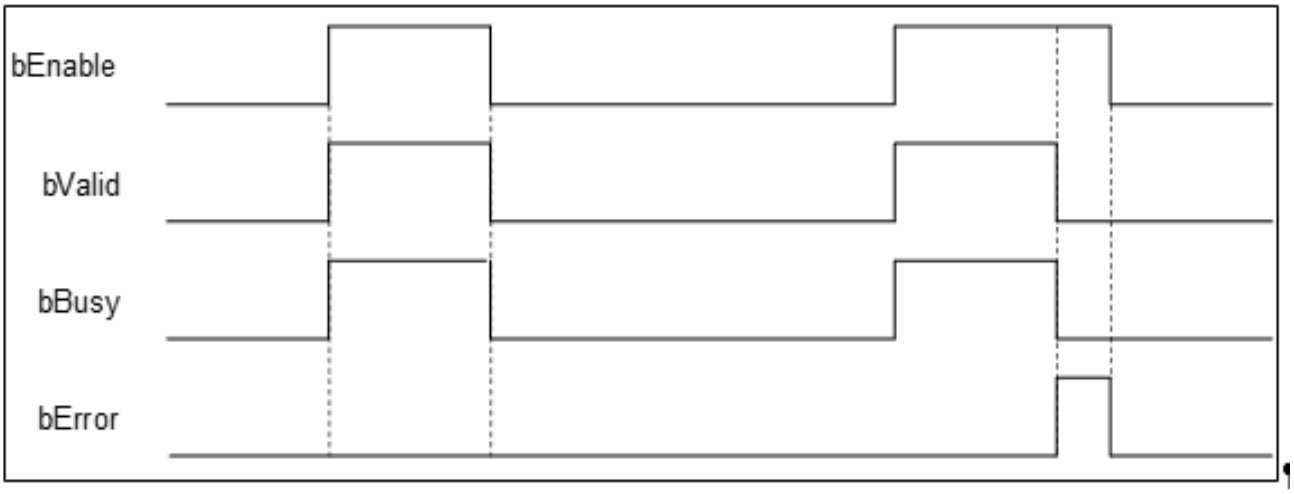
Name	Function	Data Type	Output Range (Default Value)
bValid	TRUE when the control over software limit parameters is valid	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	When a command error occurs, record the error code. For the detailed description of the error code, refer to **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	The control over software limit parameters is valid.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bEnable* turns to TRUE.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* is FALSE. (Error code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



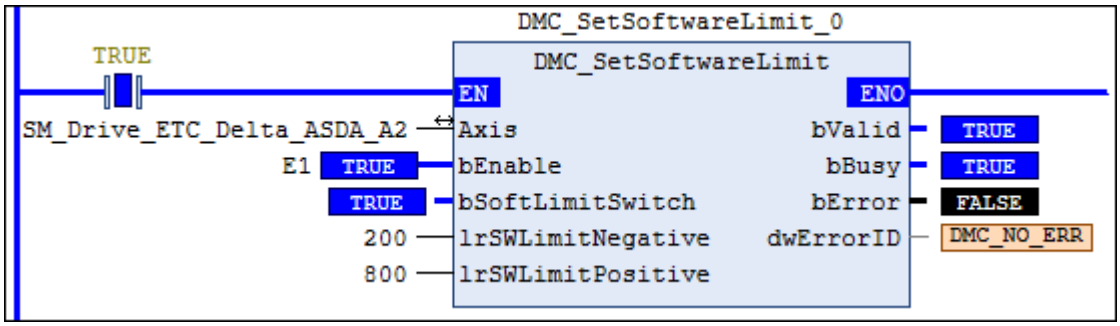
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	When *bEnable* turns to TRUE

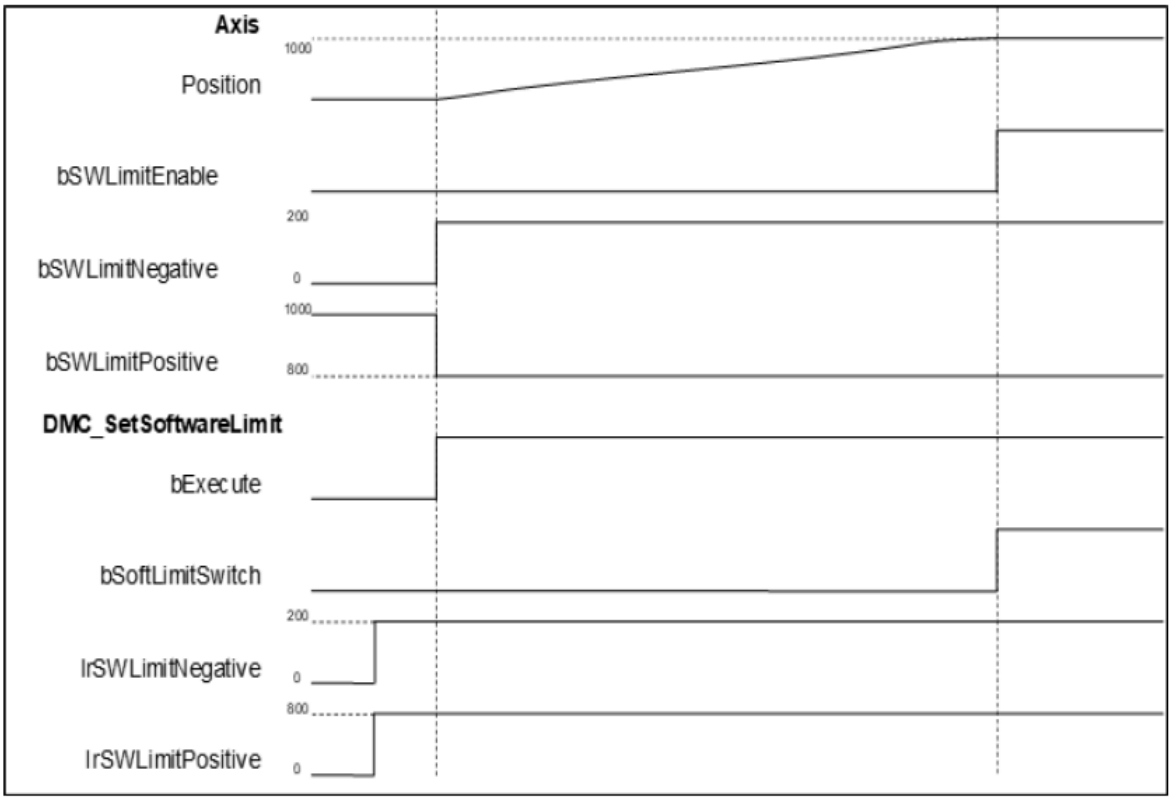
***Note**: AXIS_REF_SM3 (FB): The interface is built in every function block and works as the starting program of the function block.

- **Function**
  - After running DMC_SetSoftwareLimit, the writing and monitoring of *bSoftLimitSwitch*, *fSWLimitNegative*, and *fSWLimitPositive* axis parameters will be conducted according to the settings of *bSoftLimitSwitch*, *lrSWLimitNegative*, and *lrSWLimitPositive* of the instruction.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  The example illustrates how to use DMC_SetSoftwareLimit to set the software limits.



- Timing Diagram

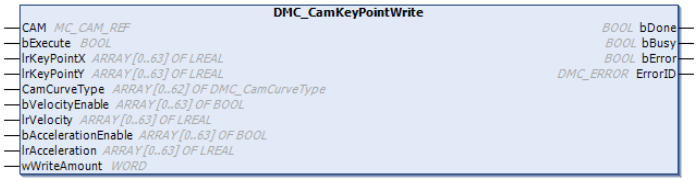


1. After DMC_SetSoftwareLimit starts, the axis parameters writing is conducted based on the input parameters of the instruction.
2. When *bSoftLimitSwitch* is TRUE, the axis stopping starts as the axis position is out of the software limits.

- **Supported Devices**
  - AX-series motion controller

2.2.2.21 DMC_CamKeyPointWrite

DMC_CamKeyPointWrite writes key cam points based on the selected 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	ST Language
FB	DMC_CamKeyPointWrite		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	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrKeyPointX	The master axis positions of key points (Unit: user unit)	LREAL [0..63]	Negative, Positive, or 0 (0)	*bExecute* is TRUE and the output *bBusy* is FALSE.
lrKeyPointY	The slave axis positions of key points (Unit: user unit)	LREAL [0..63]	Negative, Positive, or 0 (0)	*bExecute* is TRUE and the output *bBusy* is FALSE.
CamCurveType	The cam curve type between key cam points	DMC_CamCurveType [0..62]*	0: Line 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)	*bExecute* is TRUE and the output *bBusy* is FALSE.
bVelocityEnable	The velocity settings of key points	BOOL [0..63]	Negative, Positive, or 0 (0)	*bExecute* is TRUE and the output *bBusy* is FALSE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrVelocity	The velocity of key cam points	LREAL [0..63]	Negative, Positive, or 0 (0)	*bExecute* is TRUE and the output *bBusy* is FALSE.
bAccelerationEnable	Enables acceleration settings of key points	BOOL [0..63]	Negative, Positive, or 0 (0)	*bExecute* is TRUE and the output *bBusy* is FALSE.
lrAcceleration	Acceleration rates of key cam points	LREAL [0..63]	Negative, Positive, or 0 (0)	*bExecute* is TRUE and the output *bBusy* is FALSE.
wWriteAmount	The amount of key cam points	WORD	2–64 (2)	*bExecute* is TRUE and the output *bBusy* is FALSE.

***Note**: DMC_CamCurveType: Enumeration (ENUM)

- **Outputs**

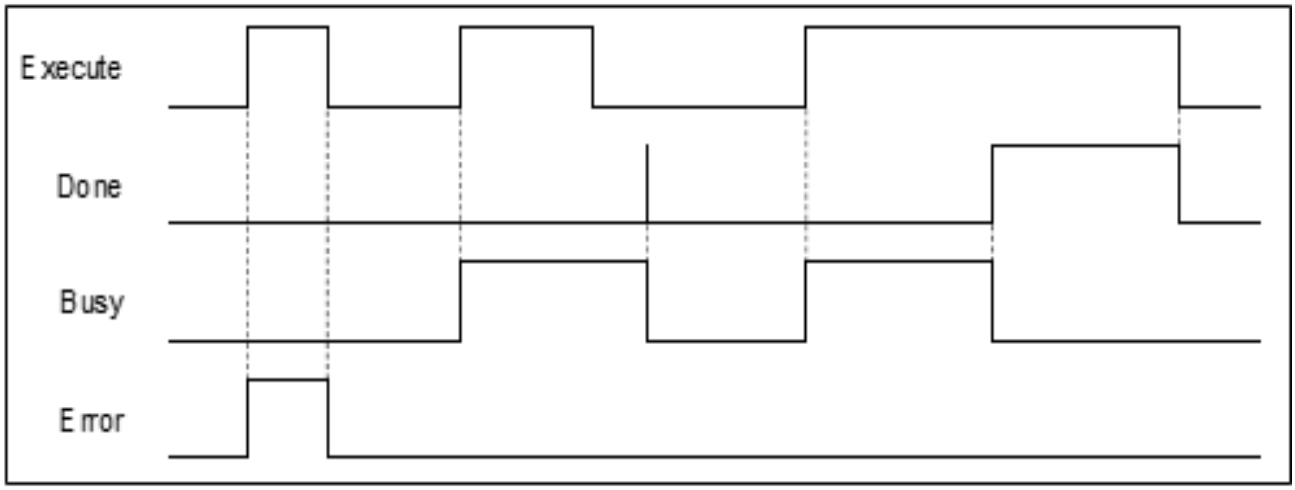
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the instruction is complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	When a command error occurs, 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	The instruction is complete.	• *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	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns from TRUE to FALSE (Error code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF*	MC_CAM_REF	*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.

  - The following lists the CamCurveType and descriptions.

CamCurveType	Description
Line (0)	It is used when the velocity must be maintained constantly. It it better to use in low-speed motion because it is easy to have a large force at the beginning and end of a straight line (the acceleration between the beginning and end of the line tends to be infinite), which is particularly visible in the high-speed motion.
Quadratic_Parabola (1)	It it used when the acceleration must be maintained constantly. It it better to use in lower-speed motion because 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.
Poly5 (2)	It is used to set the velocity and acceleration boundaries of the start point and the end point, or select to automatically continue the velocity and acceleration boundary value of the previous or next segment (via disabling *bVelocityEnable* / *bAccelerationEnable*)
Basic_Sine (3)	It is used when 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

CamCurveType	Description
	shock and vibration. So the curve type is suitable for applications in the medium and low speed operation.
Inclined_Sine (4)	It is used when 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 (5)	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 (6)	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 is smoother. The curve type is applied for high-speed operation.
Cubic_Spline_Nature (7)	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 (8)	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 (9)	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.
Harmonic2_Direct (10)	This curve consists of two different harmonic motions. One is a quarter of the amplitude and the other is twice the frequency. This curve has zero acceleration at the beginning of the stroke, eliminating the high shock and high vibration of a single simple harmonic curve, but there will be maximum acceleration at the end of the stroke, and it is recommended to use Harmonic2_ Inverse or Poly5 as the next engaged motion curve.
Harmonic2_Inverse (11)	This curve consists of two different harmonic motions. One is a quarter of the amplitude and the other is twice the frequency. This curve has zero acceleration at the end of the stroke, eliminating the high shock and high vibration of a single simple harmonic curve, but there will be maximum acceleration at the beginning of the stroke, and it is recommended to use Harmonic2_ Direct or Poly5 as the next engaged motion curve.

- Velocity Enable / Acceleration Enable
  - You can enable and 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 *IrVelocity* and *IrAcceleration* 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
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
10	Harmonic2_Direct (10)	Not possible	Not possible	0	Automatically calculated
11	Harmonic2_Inverse (11)	Not possible	Not possible	0	Automatically calculated

***Note**:

1. Not possible: The set value is invalid; Possible: The set 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;

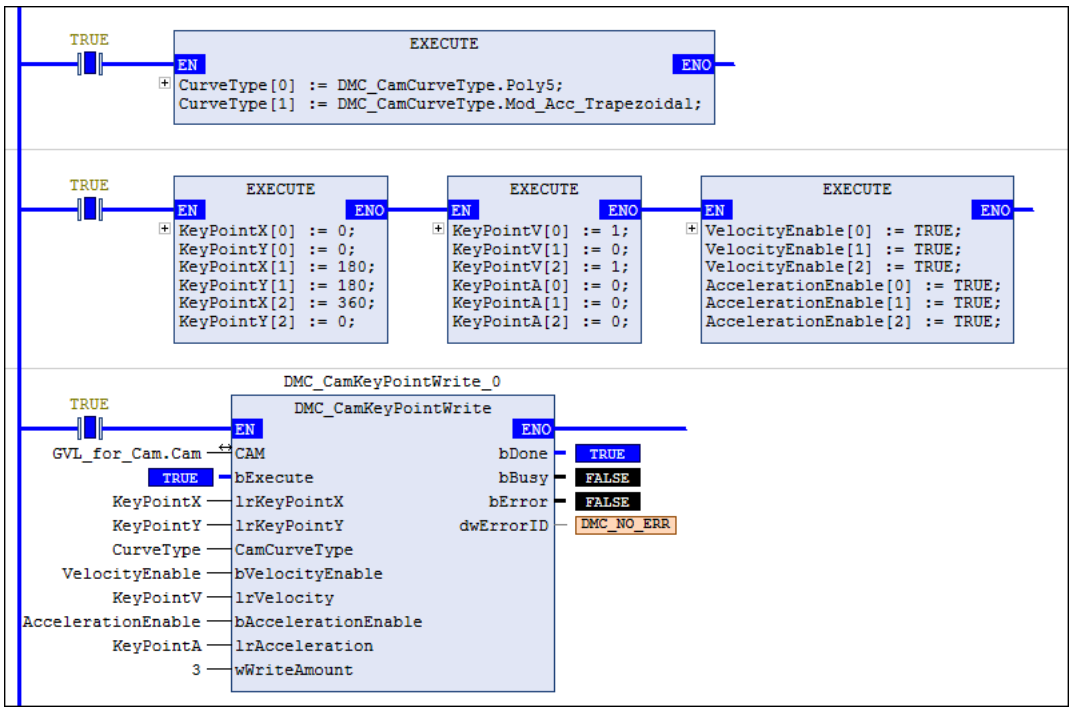
CamCurve_Type[2] : = Cubic_Spline_Nature;

- 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

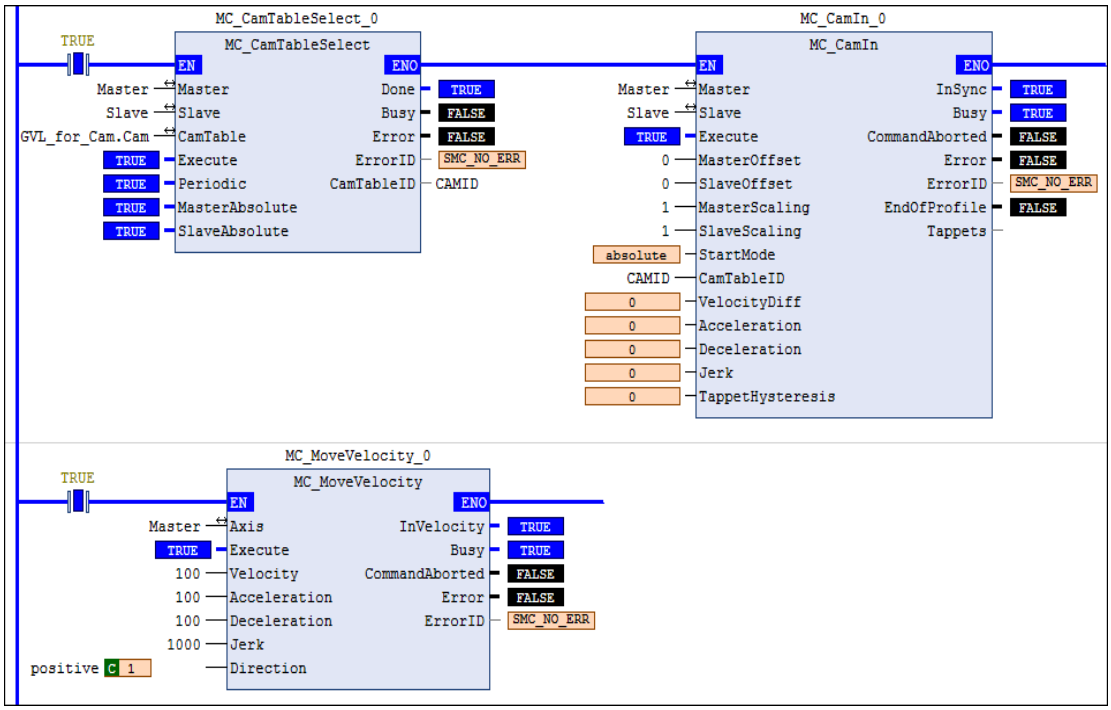
Curve Type	Description
Line (0)	Used in the situation where the velocity-constant motion is maintained. There will be a large force on the start point and end point of a straight line (the accelerations for the start point and end point of the line segment approaches infinity), which is very obvious 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 and it is zero at the middle point. So 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.

Curve Type	Description
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.

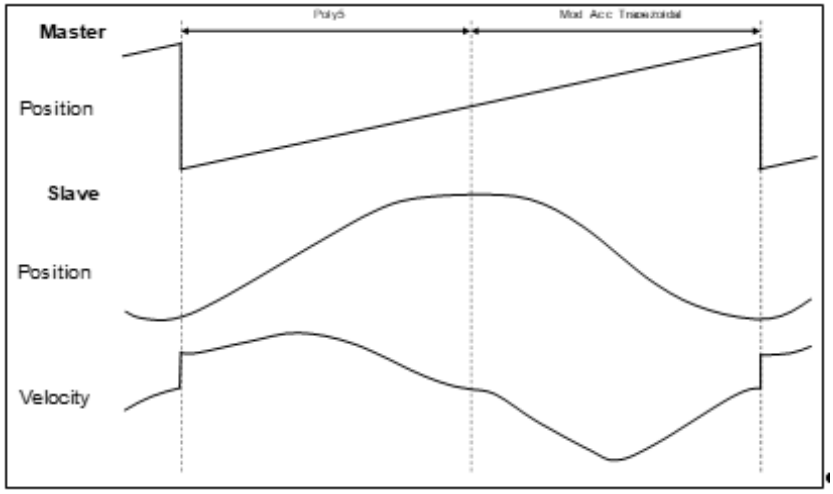
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - **Example 1**: The example illustrates the synchronized motion based on the cam table generated from DMC_ DMC_CamKeyPointWrite.



- 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.

- **Example 2**: The example illustrates the special applications of cubic interpolation curves.

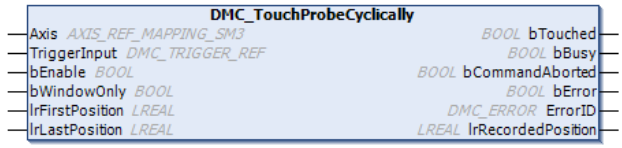
```
CamCurve_Type[0] := Line;
CamCurve_Type[1] := Cubic_Spline_Nature;   } Case 1.
CamCurve_Type[2] := Cubic_Spline_Nature;   }
CamCurve_Type[3] := Line;
CamCurve_Type[4] := Cubic_Spline_Nature;   }
CamCurve_Type[5] := Cubic_Spline_Nature;   } Case 2.
CamCurve_Type[6] := Cubic_Spline_Nature;   }
CamCurve_Type[7] := 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;
```

1. If you want to plan a cubic curve with three key points, select the curve types with the same boundaries.
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.
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.

- **Supported Devices**
  - AX-series motion controller

2.2.2.22 DMC_TouchProbeCyclically

DMC_TouchProbeCyclically continuously records the captured position of an axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_TouchProbeCyclically		DMC_TouchProbeCyclically_instance(Axis : =, TriggerInput : =, bEnable : =, bWindowOnly : =, lrFirstPosition : =, lrLastPosition : =, bTouched =>, bBusy =>, bCommandAborted =>, bError =>, ErrorID =>, lrRecordedPosition =>);

- **Inputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
bEnable	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
bWindowOnly	The Window range setting.	BOOL	TRUE/FALSE (FALSE)	*bEnable* and *bBusy* are TRUE.
lrFirstPosition	The start position of Window (Unit: user unit)	LREAL	Negative, Positive, or 0 (0)	*bEnable* and *bBusy* are TRUE.
lrLastPosition	The end position of Window (Unit: user unit)	LREAL	Negative, Positive, or 0 (0)	*bEnable* and *bBusy* are TRUE.

- **Outputs**

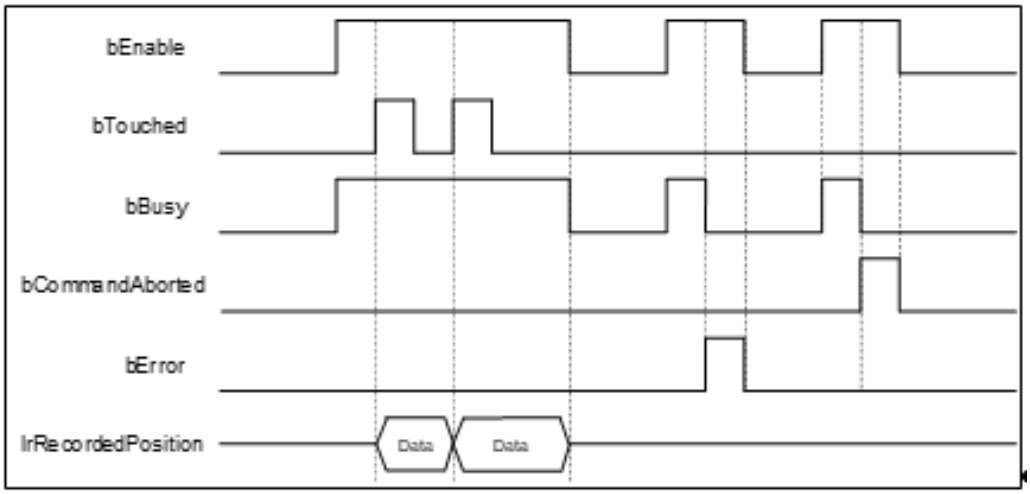
Name	Function	Data Type	Output Range (Default Value)
bTouched	TRUE when the trigger signal is TRUE and axis position recording is complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
lrRecordedPosition	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	The trigger signal is TRUE and axis position recording is completed.	• *bEnable* turns to FALSE. • *bCommandAborted* turns to TRUE. • *bError* turns to TRUE. • After a period when *bEnable* turns to TRUE.
bBusy	The instruction is running.	• *bEnable* turns to FALSE. • *bCommandAborted* turns to TRUE. • *bError* turns to TRUE.
bCommand Aborted	The instruction is interrupted by another instruction.	*bEnable* turns to FALSE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* changes from TRUE to FALSE.
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*1	AXIS_REF_SM3 (Must be specified)	-
TriggerInput	Trigger signal	DMC_TRIGGER_REF*2	TRIGGER_REF (-1)	*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)

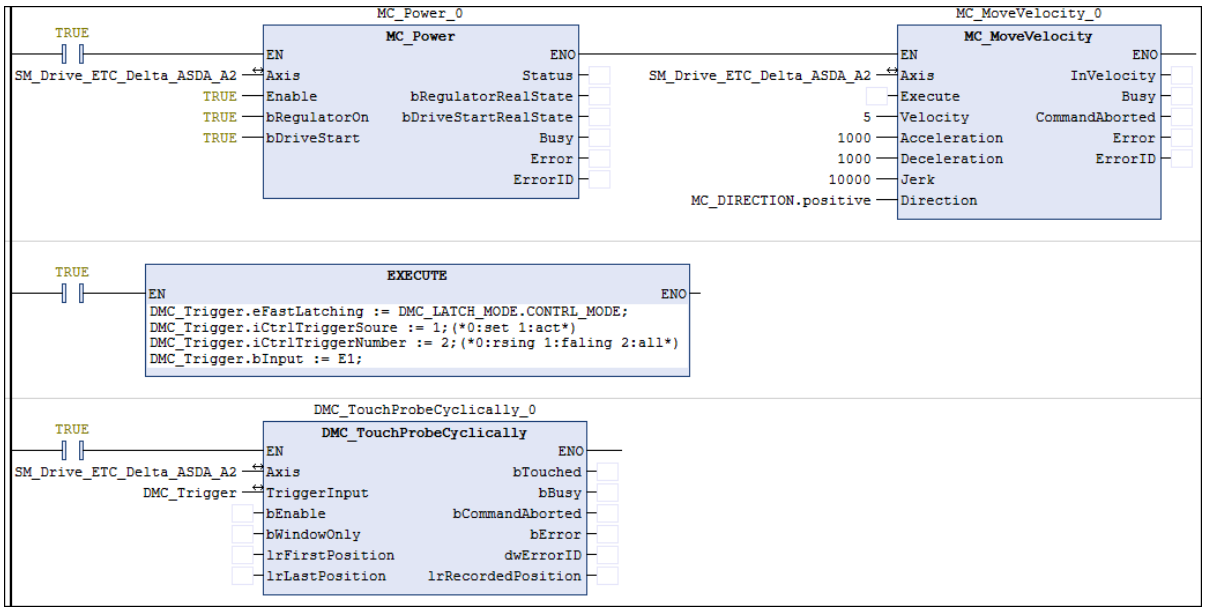
Name	Function	Data Type	Setting Value (Default Value)
iTriggerNumber	The trigger channel	INT	0: Touch Probe 1 1: Touch Probe 2 (-1)
eFastLatching	The trigger signal	DMC_LATCH_MODE	0: DRIVE_MODE 1: CONTRL_MODE (DRIVE_MODE)
bInput	The trigger signal source when the controller is triggered	BOOL	The trigger 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)
iDrvTriggerSource (Reserved)	Drive capture source settings	INT	0: Motor encoders 1: CN5
lrNumerator (Reserved)	Drive mode CN5 numerator gear ratio	LREAL	Positive
lrDenominator (Reserved)	Drive mode CN5 denominator gear ratio	LREAL	Positive
bCN5ModuleTypeEnable (Reserved)	Drive mode CN5 linear axis/ modulo axis	BOOL	TRUE: Modulo axis FALSE: Linear axis
lrCN5ModuloValue (Reserved)	Drive mode CN5 modulus value setting	LREAL	Positive

- **Function**
  - This function block is not available in PLC simulation mode.
  - When the trigger signal (*eFastLatching*) is DRIVE_MODE, the position is provided by the servo and iCtrlTriggerSource will not work. iCtrlTriggerSource works only for CONTRL_MODE.
  - When DMC_TouchProbeCyclically is used, the Touch Probe Function (60B8h) cannot be configured to PDO. If you configure it to PDO, the function block will report an error while running.
  - DMC_TouchProbeCyclically cannot be used with MC_TouchProbe together. If MC_TouchProbe is already run, an error will occur when DMC_TouchProbeCyclically runs. DMC_TouchProbeCyclically will also report an error if MC_TouchProbe is run during DMC_TouchProbeCyclically running.
  - 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.

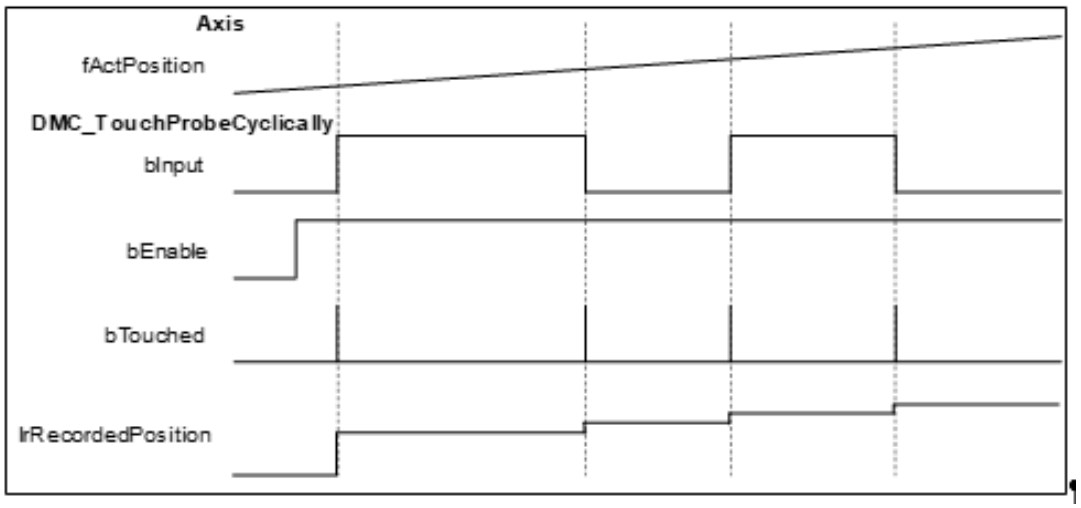
- *bInput* of *TriggerInput* is the trigger signal source under CONTRL_MODE, and it is an invalid parameter under DRIVE_MODE mode.
- When DRIVE_MODE is used, TouchProbe1 and TouchProbe2 can be started respectively in two independent DMC_TouchProbeCyclically instructions.
- The position capture error occurs if the axis runs out of position range. To avoid this, ensure the function block must be in the enable status before the axis moves out of the range.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**
  - **Example 1**: This example illustrates how to use DMC_TouchProbeCyclically in the CONTRL_MODE mode.

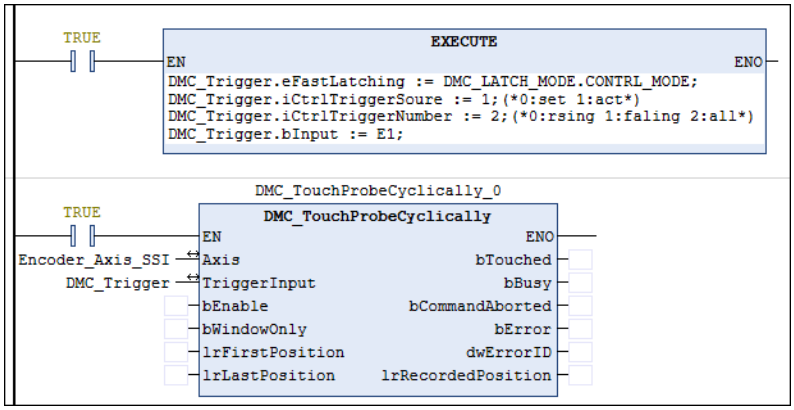


    - Timing Diagram

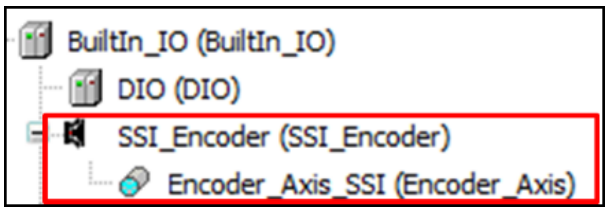


- DMC_TouchProbeCyclically specifies CONTRL_MODE and uses the rising edge or falling edge as the trigger signal with the actual position of the axis (*fActPosition*) as the reference position.
- When the signal source *bInput* of *TriggerInput* is triggered in CONTRL_MODE mode and the state of bInput changes, the function block will record the actual position of the current axis, and *bTouched* will remain TRUE for one period.

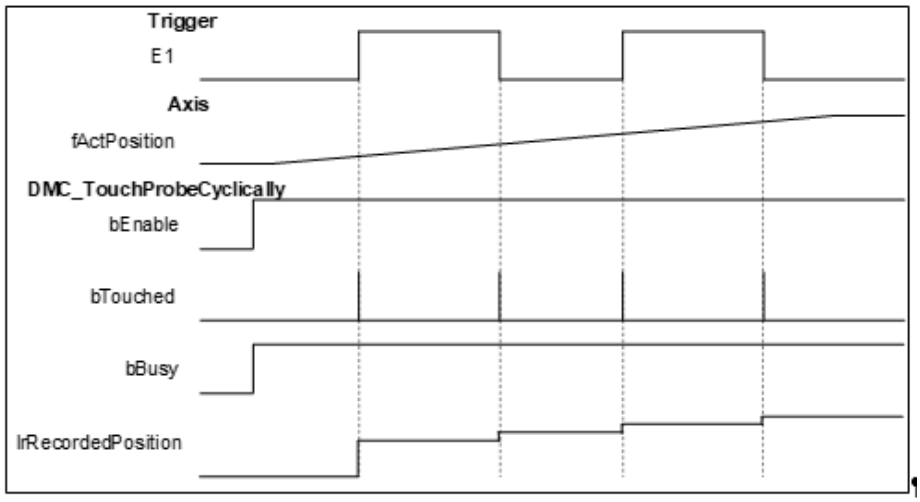
◦ **Example 2**: The example illustrates how to use DMC_TouchProbeCyclically by using SSI Encoder as the signal source in CONTRL_MODE.



- Project tree setting

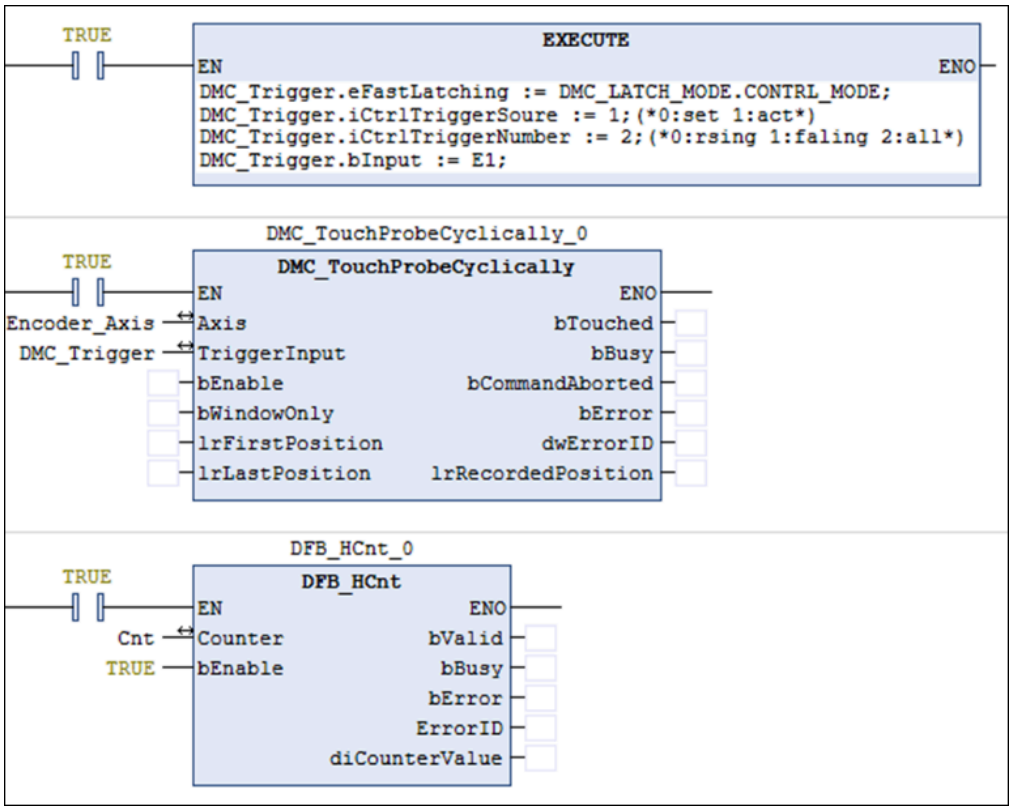


- Timing Diagram

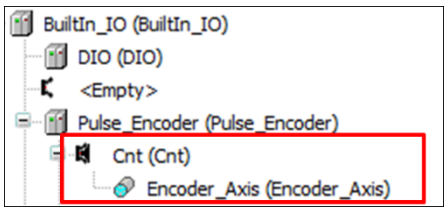


1. Select SSI Encoder as the signal source for DMC_TouchProbeCyclically.
2. Add an SSI Encoder to the device tree and then connect the SSI Encoder to the AX-308 module. For wiring, refer to 2.2.4 CPU Module Input and Output Terminals in *AX-3 Series Operation Manual*.
3. When *bInput* of *TriggerInput* is triggered, DMC_TouchProbeCyclically will record the position of the current SSI Encoder.

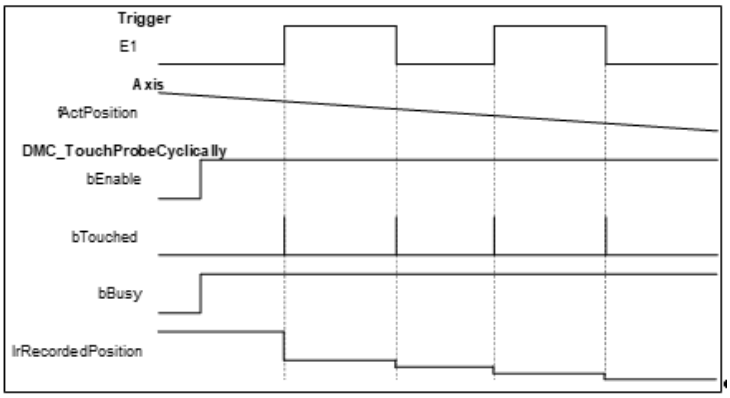
◦ **Example 3**: The example illustrates how to use DMC_TouchProbeCyclically with Pulse Encoder as the signal source in CONTRL_MODE mode.



- Project tree setting

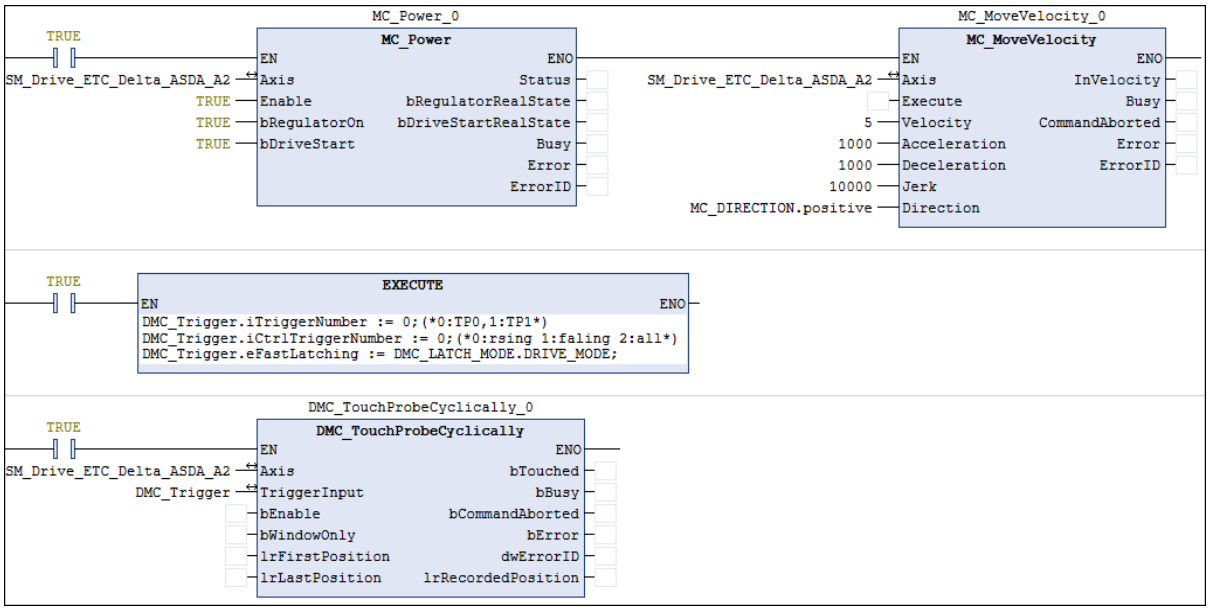


- Timing Diagram

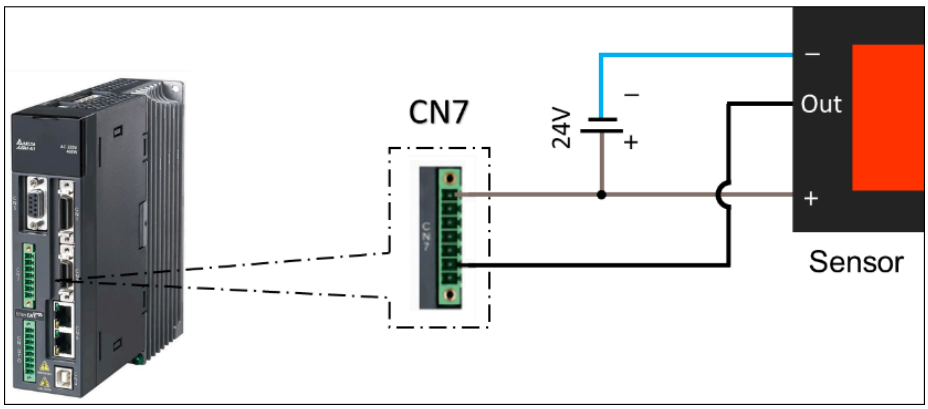


1. Select Pulse Encoder as the signal source for DMC_TouchProbeCyclically.
2. Add a Count to the device tree (here is Count 1), and then add DFB_HCnt to the program to read the value of the Pulse Encoder. At last, connect the Pulse Encoder. For wiring, refer to Section 2.2.4 CPU Module Input and Output Terminals in *AX-3 Series Operation Manual*.
3. When *bInput* of *TriggerInput* is triggered, DMC_TouchProbeCyclically will record the position of the current Pulse Encoder.

- **Example 4**: The example illustrates how to use DMC_TouchProbeCyclically in DRIVE_MODE.

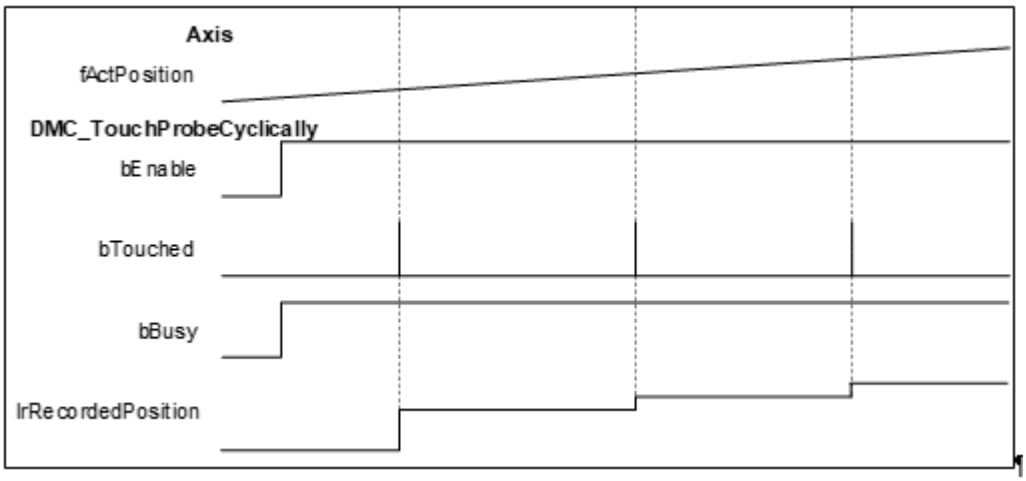


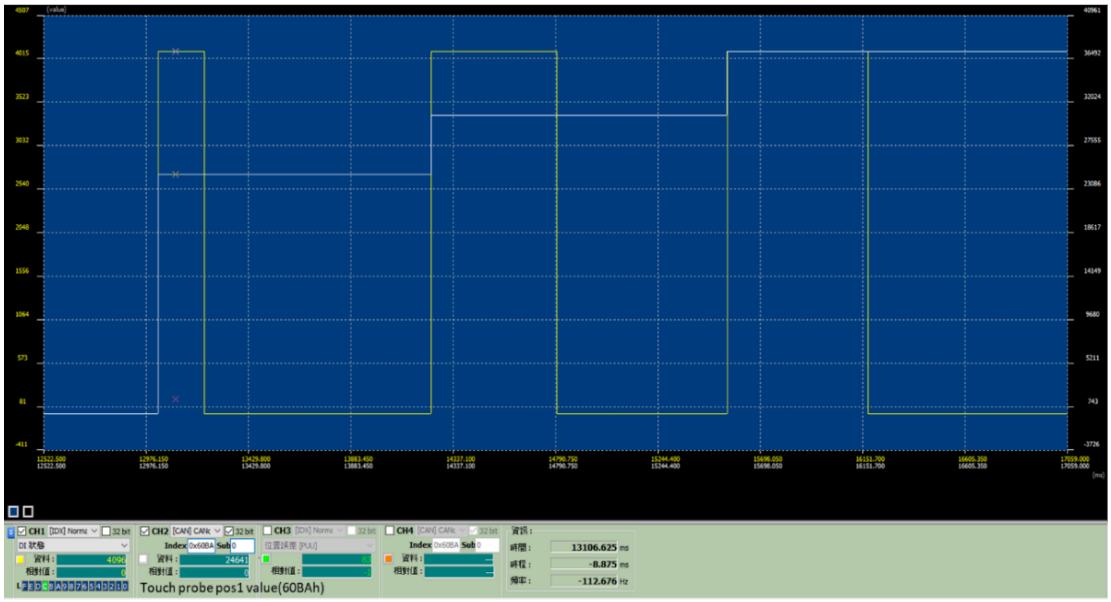
- Wiring figure



  - The trigger signal comes from DI13 of extension DIs of the servo drive's CN7.
  - 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



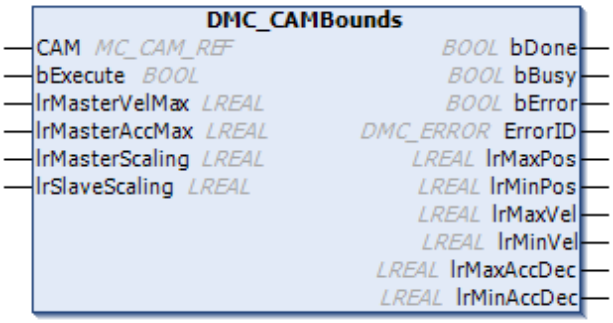


- DMC_TouchProbeCyclically specifies DRIVE_MODE with TouchProbe 1 which is triggered by the rising edge signal.
- When the switch trigger occurs, the drive records the current position, sends it back to the controller and stores it in the function block *lrRecordedPosition*. *bTouched* will remain TRUE for one period.
- In DRIVE_MODE, the drive continuously records the current position in real time. Therefore, the recorded position will be earlier than the actual feedback position of the controller.

- **Supported Devices**
  - AX-series motion controller

2.2.2.23 DMC_CAMBounds

DMC_CAMBounds inputs the maximum velocity and acceleration of the master axis in the cam table to predict the maximum and minimum values of the slave axis' position, velocity, and acceleration when it is following the master axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_CAMBounds		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	Runs the instruction 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)	*bExecute* and *bBusy* are TRUE.
lrMasterAccMax	Expected maximum acceleration of the master axis (User unit/sec^2)	LREAL	Positive or 0 (1)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrMasterScaling	The scaling factor of the master axis	LREAL	Positive (1)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrSlaveScaling	The scaling factor of the slave axis	LREAL	Positive (1)	*bExecute* turns to TRUE and *bBusy* is FALSE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the calculation is complete	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)

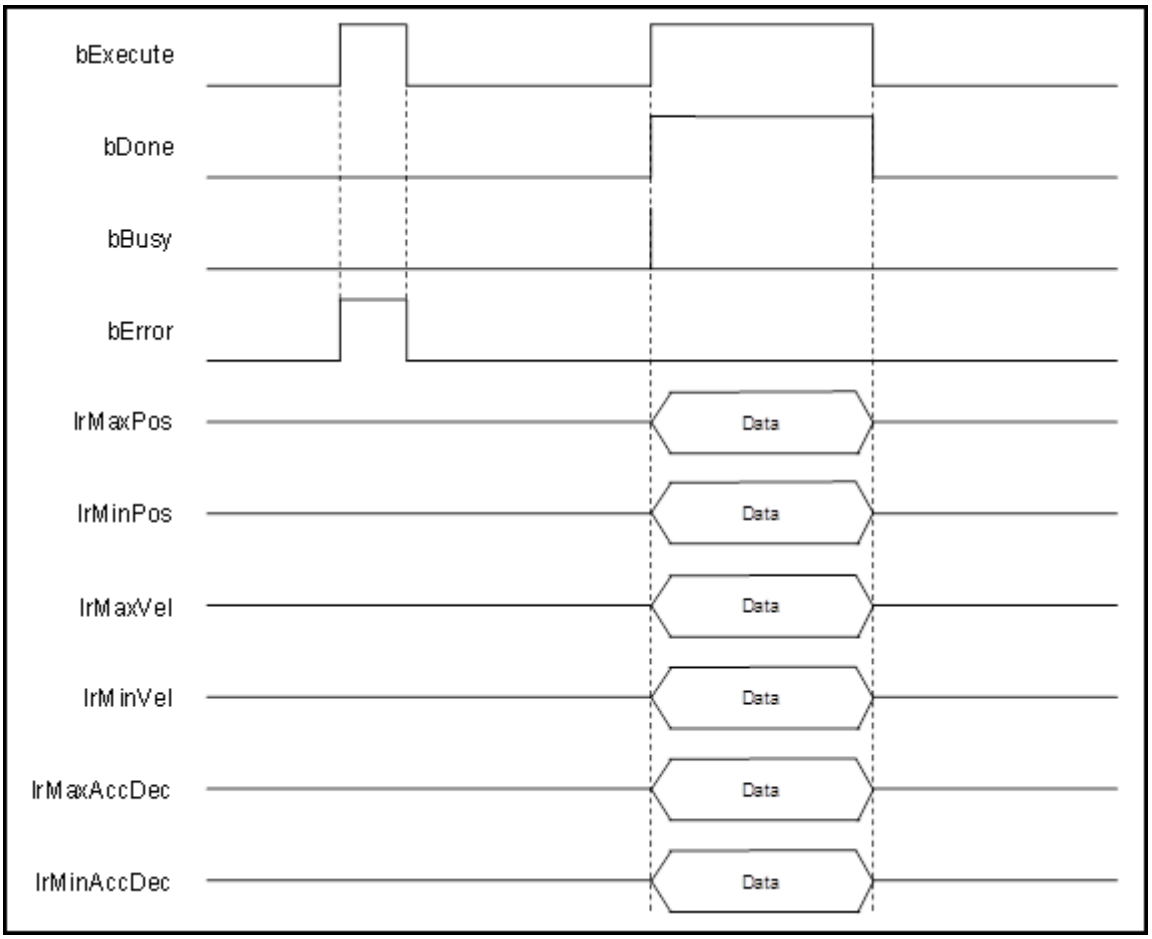
Name	Function	Data Type	Output Range (Default Value)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
lrMaxPos	The maximum position of the slave axis (User unit)	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	The trigger signal is TRUE and axis position recording is complete.	• *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	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE.
ErrorID		
dMaxPos	Update values after calculation completes.	The value goes to zero when *bExecute* turns to FALSE.
dMinPos		
dMaxVel		
dMinVel		
dMaxAccDec		
dMinAccDec		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

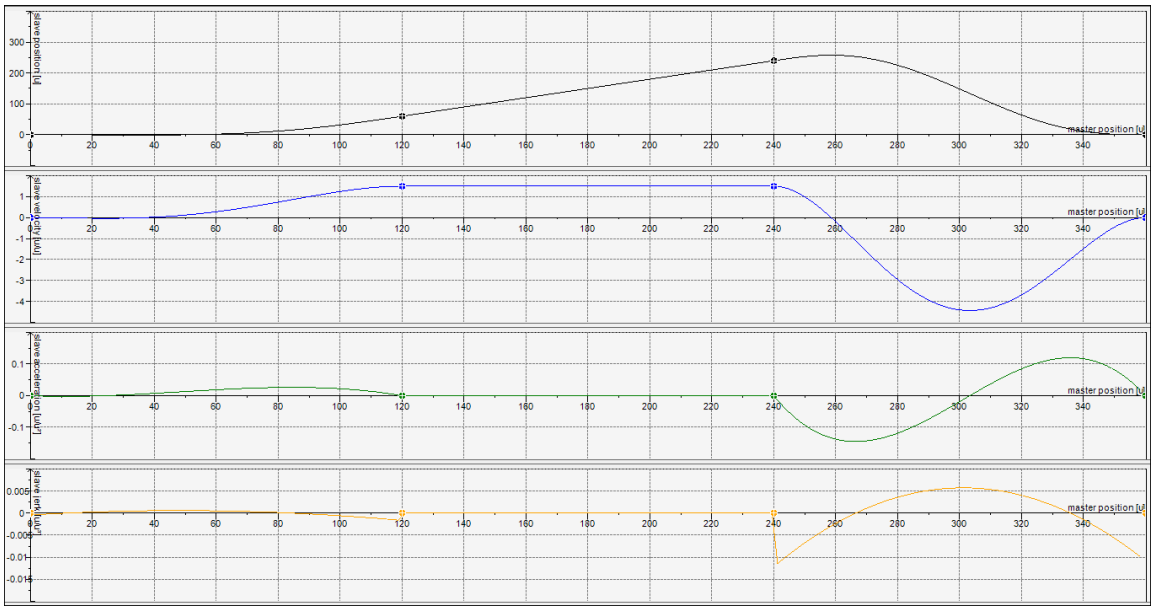
Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF*	MC_CAM_REF	When *bExecute* is TRUE

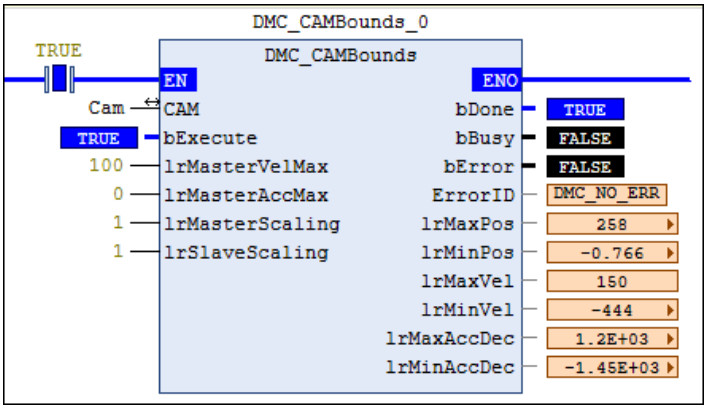
***Note**: MC_CAM_REF (FB): Basic cam

- **Function**
  - DMC_CAMBounds is used to check whether the user-defined cam table curve is correct. Use the maximum velocity, acceleration, and deceleration 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**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

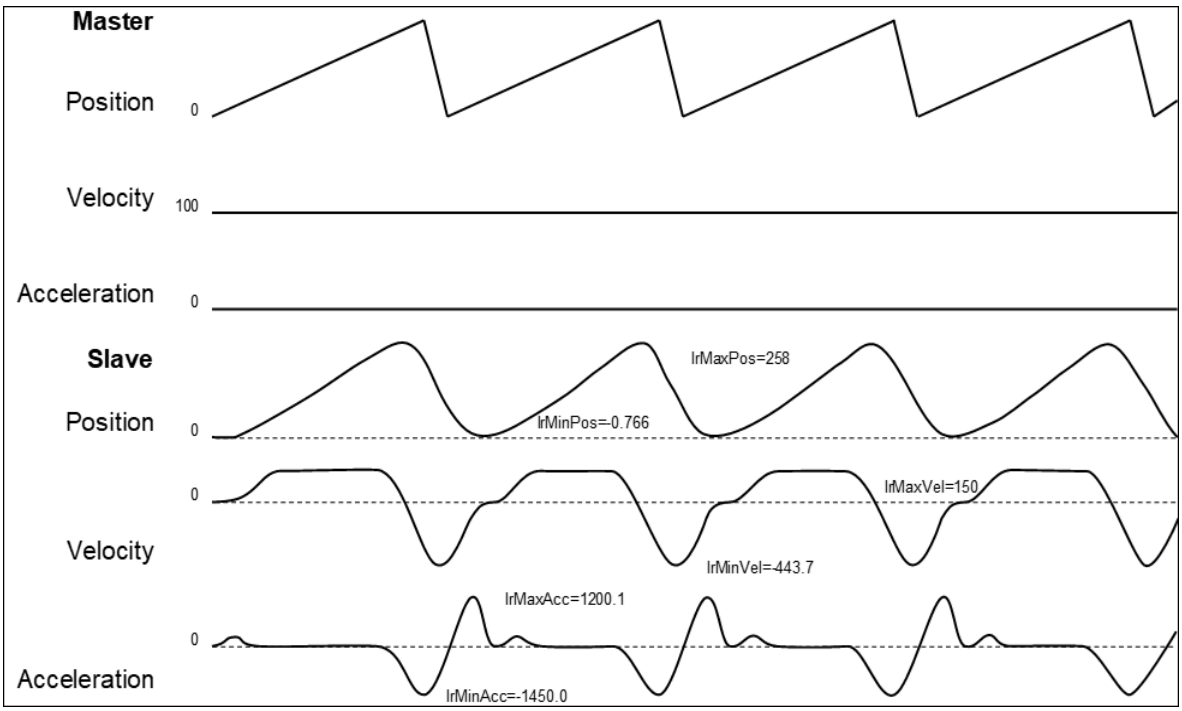
  The example shows the use of DMC_CAMBounds.

- Cam table:





- Timing Diagram

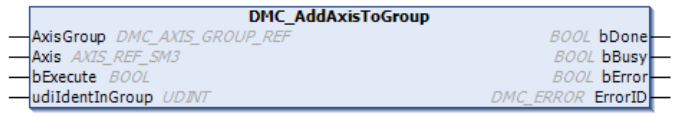


From the oscillogram, you can see that the maximum and minimum positions, velocity, and acceleration of the slave axis are consistent with the output of the DMC_CAMBounds.

- **Supported Devices**
  - AX-series motion controller

2.2.2.24 DMC_AddAxisToGroup

DMC_AddAxisToGroup adds a single axis to the axis group.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_AddAxisToGroup		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
udiIdentInGroup	Where the axis to be added in the axis group	UDINT	1–6 (1)	When *bExecute* isTRUE, the parameters of *udiIdentInGroup* are updated.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the single axis is added	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

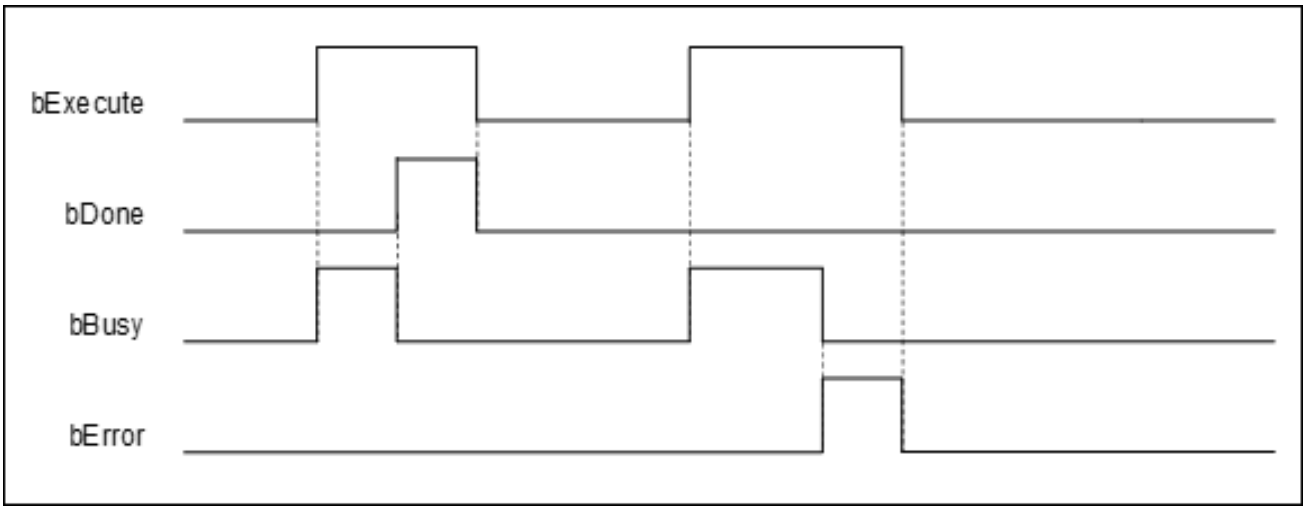
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The single axis is added.	• *bExecute* turns to FALSE. • *bError* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEexcute* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

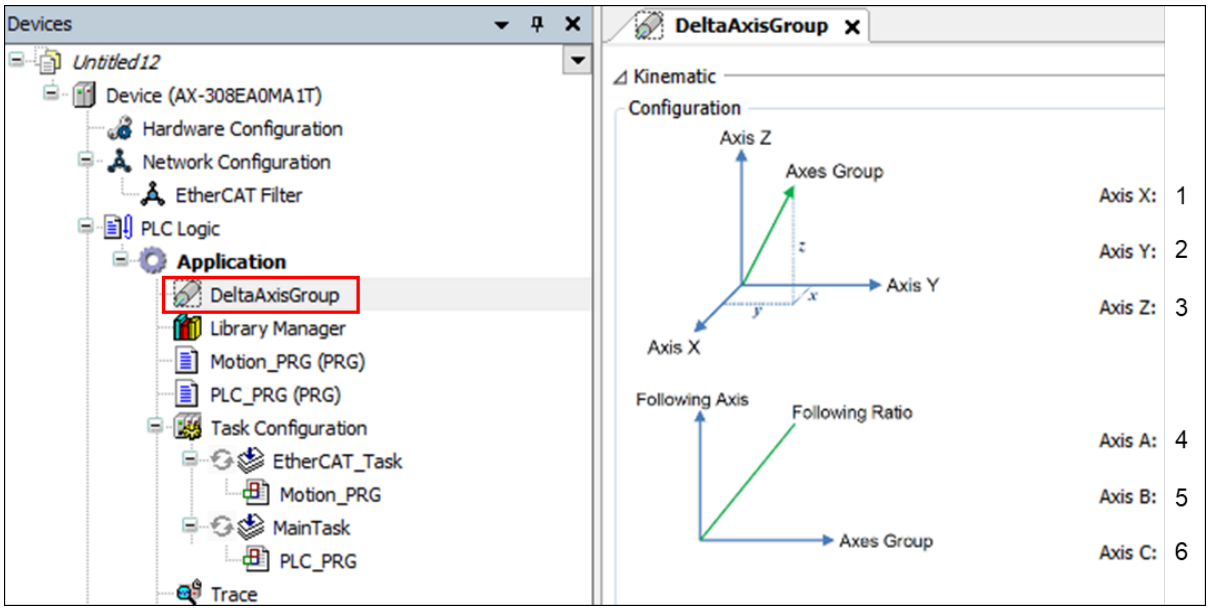
Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*1	DMC_AXIS_GROUP_REF	*bExecute* turns to TRUE and *bBusy* is FALSE
Axis	The specified axis	AXIS_REF_SM3*2	AXIS_REF_SM3	*bExecute* turns to TRUE and *bBusy* is FALSE

***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 supported by DL_MotionControl V1.2.0.0 or later.
  - Adds the axis to the axis group.
  - If an axis already exists in *udildentInGroup*, it will be overwritten.
  - The axis group state must be Disabled to run this function block.

- The *udiIdentInGroup* settings for DIADesigner-AX axis group is as follows:



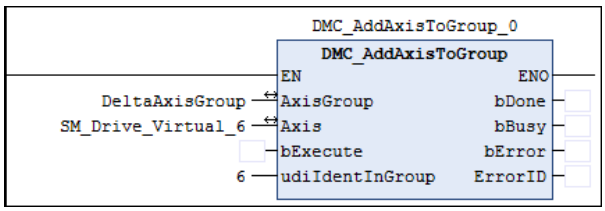
- **Troubleshooting**

  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

  - For the error codes and troubleshooting methods, refer to **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 axis position 6 in the axis group.

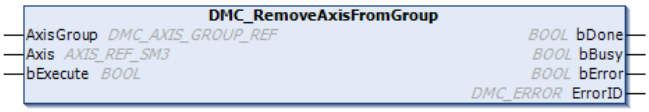
  - When *DMC_RemoveAxisFromGroup.bDone* is TRUE, the axis has been configured at the specified position in the axis group.

- **Supported Devices**

  - AX-series motion controller

2.2.2.25 DMC_RemoveAxisFromGroup

The DMC_RemoveAxisFromGroup removes a single axis from an axis group.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_RemoveAxisFromGroup		DMC_RemoveAxisFromGroup_instance(AxisGroup: = , Axis: = , bExecute: = , bDone=> , bBusy=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

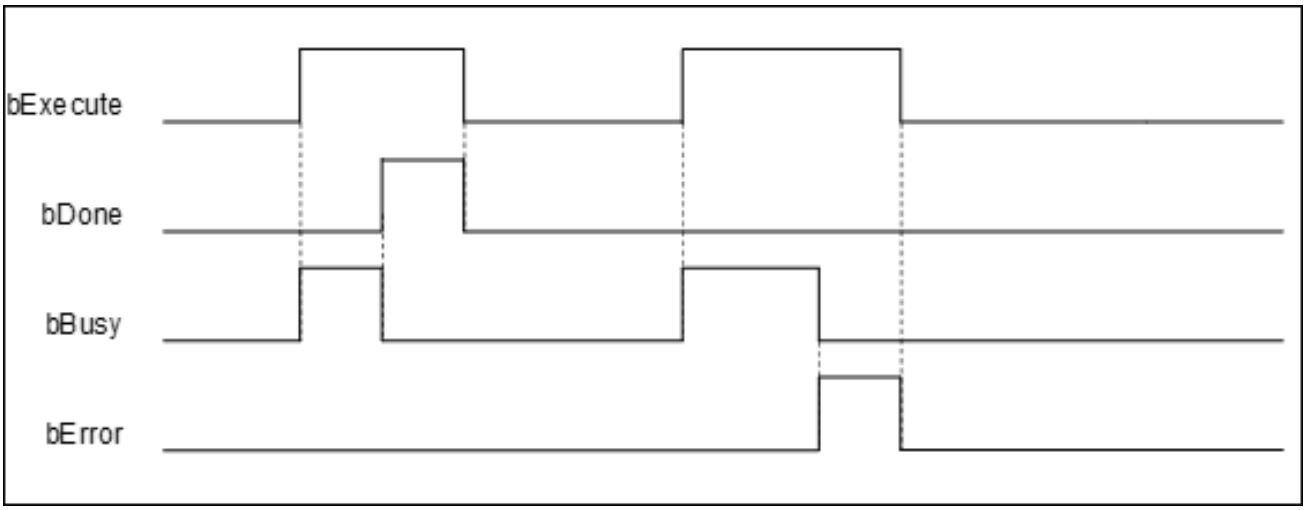
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the single axis is removed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The single axis is removed.	• *bExecute* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*1	DMC_AXIS_GROUP_REF	When *bExecute* turns to TRUE and *bBusy* is FALSE
Axis	The specified 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.
  - 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 while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For the error codes and troubleshooting methods, refer to **Appendix** of this manual.
- **Example**

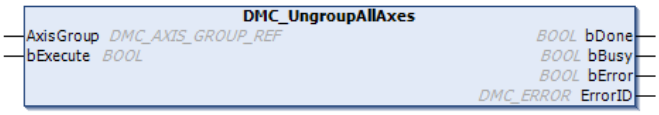
This example shows how to use DMC_RemoveAxisFromGroup to remove an axis from an axis group.

- 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.

- **Supported Devices**
  - AX-series motion controller

2.2.2.26 DMC_UngroupAllAxes

DMC_UngroupAllAxes removes all axes in the axis group.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_UngroupAllAxes		DMC_UngroupAllAxes_instance(AxisGroup: = , bExecute: = , bDone=> , bBusy=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

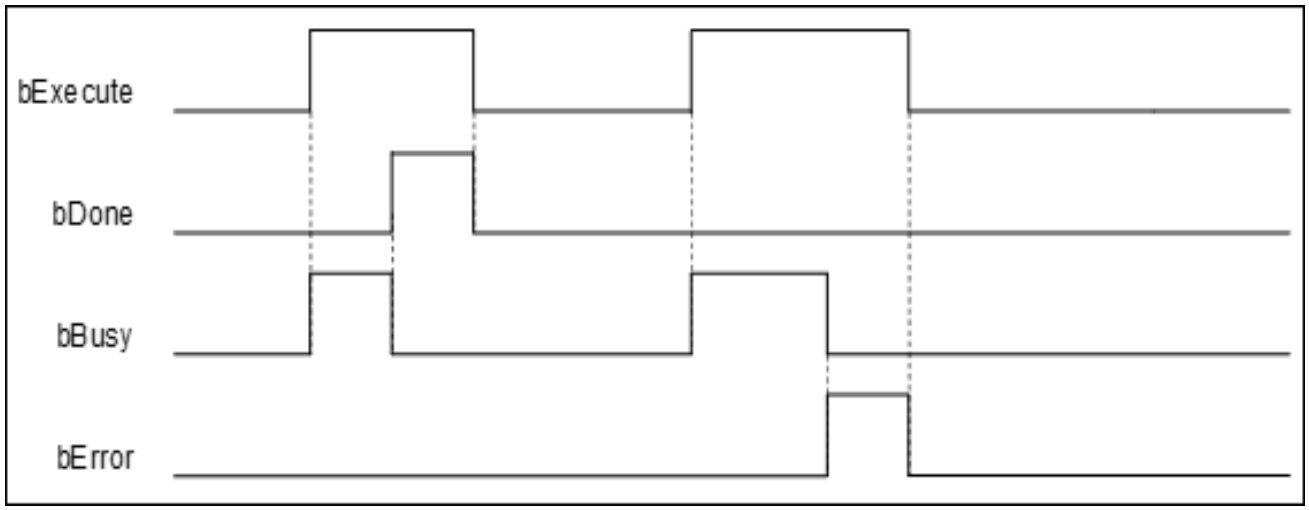
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the single axis is removed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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
bBusy	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEexcute* turns to FALSE. (Error Code is cleared.)
ErrorID		
bDone	The axes are removed.	• *bExecute* turns to FALSE. • *bError* turns to TRUE.

- **Timing Diagram of Output Parameter Changes**



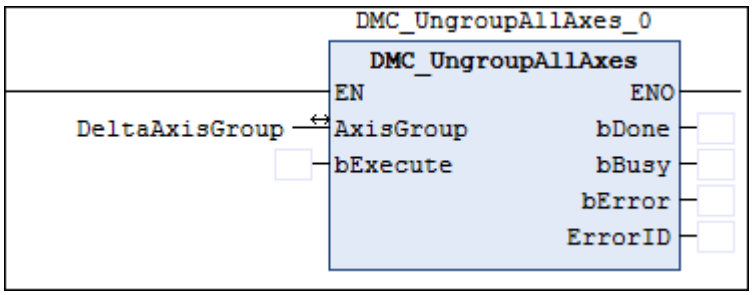
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*1	DMC_AXIS_GROUP_REF	*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 while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For the error codes and 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.

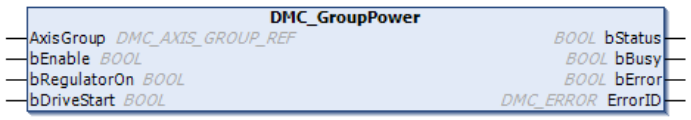


  - Enter the name of the single axis that you want to remove, and then run *DMC_UngroupAllAxes.bExecute*. When *bDone* turns to TRUE, all axes in the axis group are removed.

- **Supported Devices**
  - AX-series motion controller

2.2.2.27 DMC_GroupPower

DMC_GroupPower enables, disables, or quick stop axes in an axis group.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupPower		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	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
bRegulatorOn	Enables the power	BOOL	TRUE/FALSE (FALSE)	Only when *bEnable*=TRUE
bDriveStart	Disables the quickstop mechanism.	BOOL	TRUE/FALSE (FALSE)	Only when *bEnable*=TRUE

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bStatus	TRUE when all axes in the axis group are enabled	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

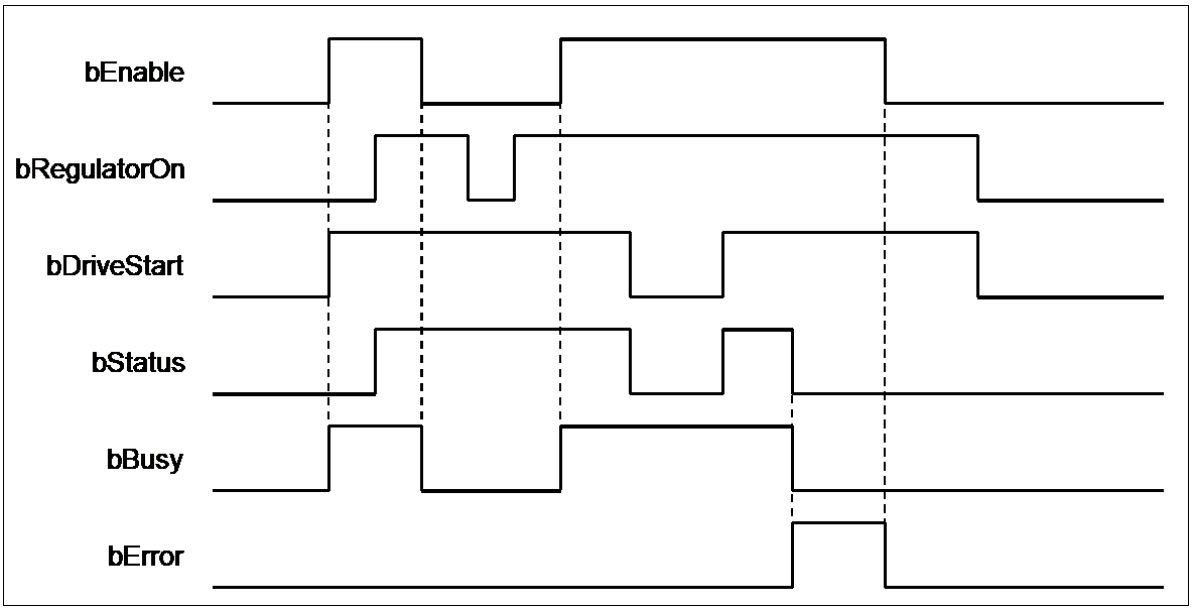
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bStatus	*bRegulatorRealState* and *bDriveStartRealState* turn to TRUE and all axes in the axis group are turned on and enter a movable state.	• *bRegulatorRealState* and *bDriveStartRealState* turn to FALSE. • *bError* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	*bEnable* turns to TRUE.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



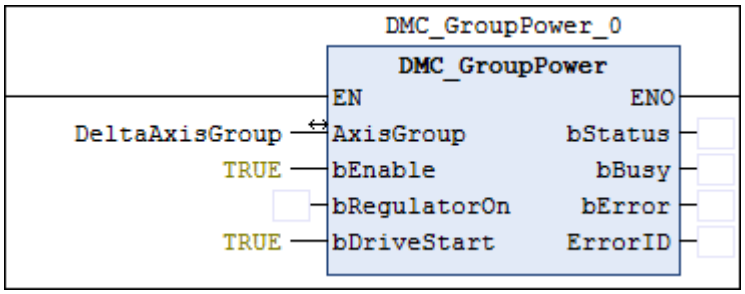
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*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 supported by 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. If the axis group status was GroupDisabled, it remained GroupDisabled after using DMC_GroupPower.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For the error codes and troubleshooting methods, refer to **Appendix** of this manual.
- **Example**

This example shows how to use DMC_GroupPower to enable all 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 axes in the axis group can be enabled.

- **Supported Devices**
  - AX-series motion controller

2.2.2.28 DMC_GroupSetOverride

DMC_GroupSetOverride changes the velocity of the axis group motion by override control factor.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GroupSetOverride		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 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, *lrVelFactor* will be updated.
lrAccFactor	Override control acceleration and deceleration factor (Reserved)	LREAL	0.0–1.0 (1.0)	When *bEnable* is TRUE, *lrAccFactor* will be updated.
lrJerkFactor	Override control jerk factor (Reserved)	LREAL	0.0–1.0 (1.0)	When *bEnable* is TRUE, *lrJerkFactor* will be updated.

- **Outputs**

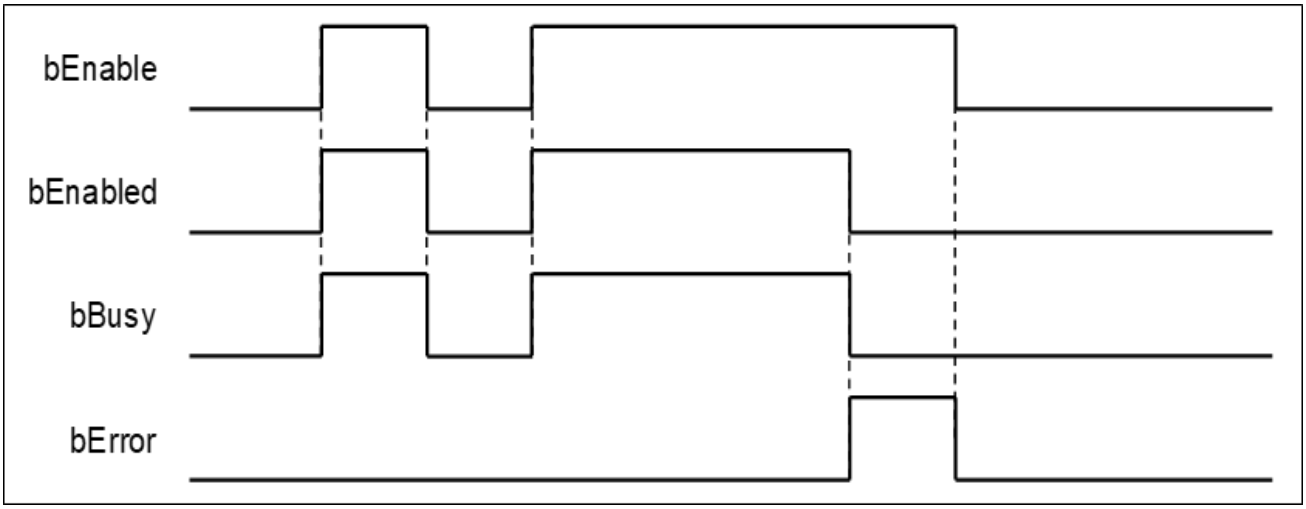
Name	Function	Data Type	Output Range (Default Value)
bEnabled	TRUE when the factor is successfully set	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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
bEnabled	*bEnable* turns to TRUE and the factor is successfully set.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bEnable* turns to TRUE.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
AxisGroup	The specified axis group	DMC_AXIS_GROUP_REF*	DMC_AXIS_GROUP_REF	*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 supported by 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 *lrVelFactor* is 0, the current axis group motion will stop, but the axis group status will not change. After switching to a non-zero value, the motion will be continued.
  - The factor of this function block does not effect the DMC_GroupStop and the deceleration and stop of ErrorStop.

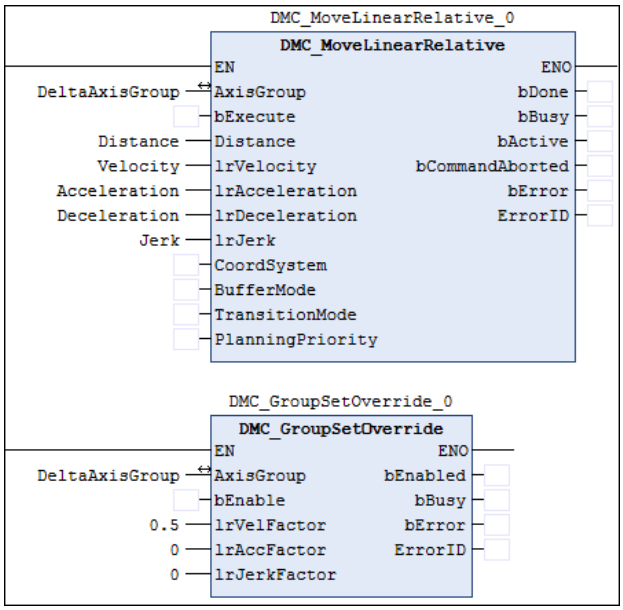
- The acceleration and deceleration velocity of this function block are based on the current motion command of the axis group.
  - If DMC_MoveLinearAbsolote acceleration and deceleration is set to 100, the override control will perform the acceleration and deceleration changes at 100.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For the error codes and troubleshooting methods, refer to **Appendix** of this manual.

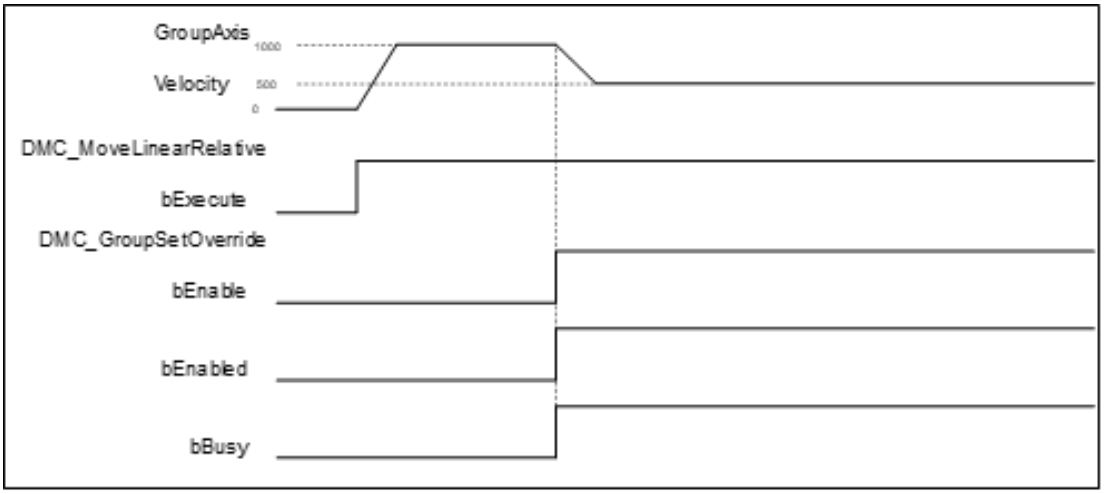
- **Example**

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.

- **Supported Devices**
  - AX-series motion controller

2.2.2.29 DMC_GetCamSlaveData

DMC_GetCamSlaveData gets the slave axis data in a cam table according to the master axis position set.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GetCamSlaveData		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	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrCamPos	Cam master axis position (User Unit)	LREAL	Positive or 0 (0)	*bEnable* changes from FALSE to TRUE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	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)

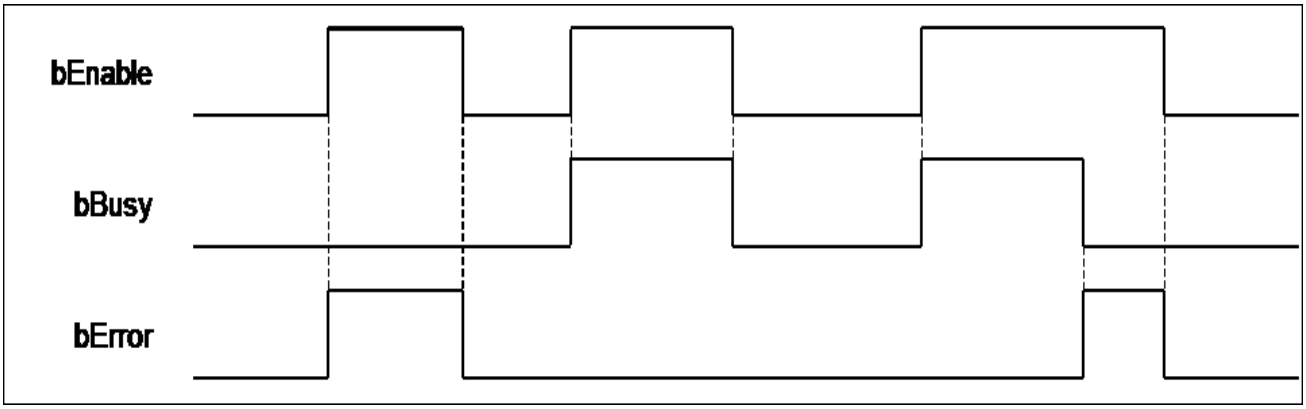
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	*bEnable* turns to TRUE.	*bError* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	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 of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF*	MC_CAM_REF	*bEnable* turns to TRUE.

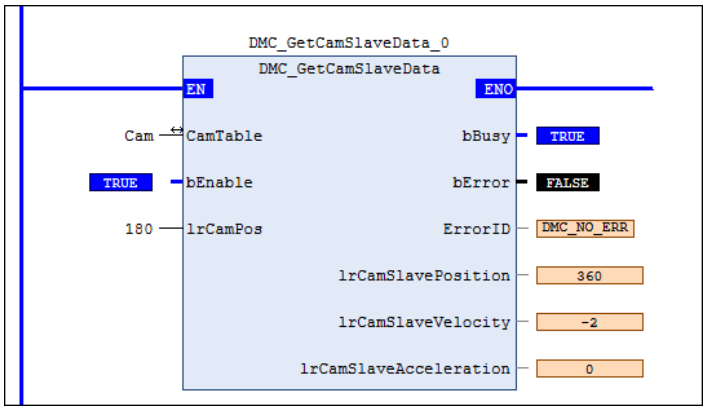
***Note**: MC_CAM_REF (FB): User-defined cam table parameters.

- **Function**
  - This function is supported by DL_MotionControl V1.2.3.0 or later.
  - By entering the master position in the 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 type is a one-dimensional table of slave positions or a two-dimensional table of related master and slave positions, 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**

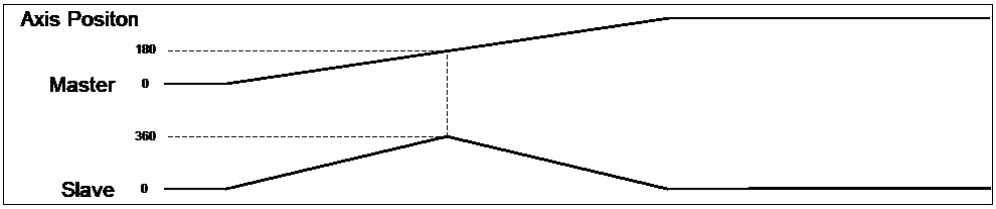
- If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- For the error codes and troubleshooting methods, refer to **Appendix** of this manual.

- **Example**

  This example shows how to use DMC_GetCamSlaveData function block to get the master axis position and the corresponding position of the slave axis in the cam table.



  - Timing Diagram



    - The figure above shows the changes of the master and slave axis when the cam table runs for a cycle.
    - Input 180 for DMC_GetCamSlaveData.lrCamPos (master axis position is 180). When the master axis runs to 180, the slave axis position will be 360.

- **Supported Devices**

  - AX-series motion controller

2.2.2.30 DMC_GetDeltaServoDriveError

DMC_GetDeltaServoDriveError only reads the current errors of Delta's servo panel.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GetDeltaServoDriveError		DMC_GetDeltaServoDriveError _instance(Axis:= , bEnable:= , bValid=> , bBusy=> , bError=> , ErrorID=> , ServoDriveError=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bValid	TRUE when read normally	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
ServoDriveError	Show the Delta servo panel errors	DWORD	DWORD (0)

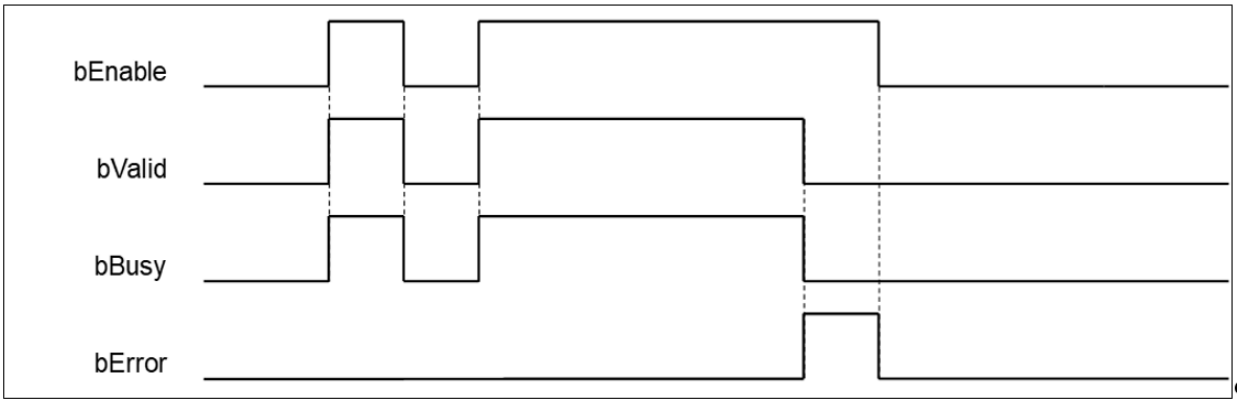
***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bValid	*bEnable* turns to TRUE.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bEnable* turns to TRUE.	• When *bEnable* turns to FALSE • When *bError* turns to TRUE
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* turns to FALSE. (Error Code is cleared.)
ErrorID		

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
ServoDriveError	Update information when *bEnable* is TRUE.	Will not update information when *bEnable* is FALSE.

- **Timing Diagram of Output Parameter Changes**



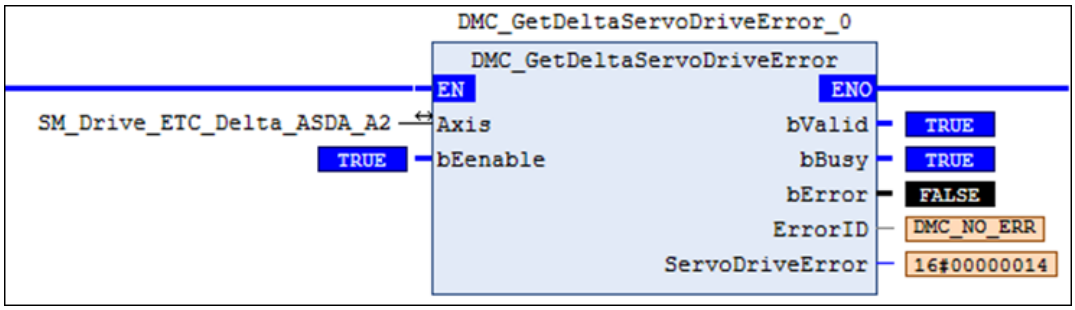
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*bEnable* turns to TRUE and *bBusy* is FALSE.

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

- **Function**
  - This function block is supported by DL_MotionControl V1.3.4.0 or later.
  - Can only read the panel alarm code of Delta servo.
  - The alarm code of the ASDA panel is displayed in hexadecimal.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For the error codes and troubleshooting methods, refer to the **Appendix** of this manual.
- **Example**

This example shows how to read error codes on the servo panel by DMC_GetDeltaServoDriveError.

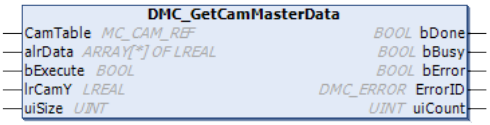


  - When the servo encounters the negative limit of the hardware, the panel reports AL014 and the function block reads 0x014.

- **Supported Devices**
  - AX-series motion controller

2.2.2.31 DMC_GetCamMasterData

DMC_GetCamMasterData gets the master axis data in a cam table according to the slave axis position set.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_GetCam MastereData		DMC_GetCamMasterData_instance(CamTable:= , alrData:= , bExecute:= , lrCamY:= , uiSize:= , bDone=> , bBusy=> , bError=> , ErrorID=> , uiCount=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrCamY	Cam slave axis position (User Unit)	LREAL	Positive (0)	*bExecute* changes from FALSE to TRUE and *bBusy* is FALSE.
uiSize	The number of the corresponding master axis positions	UINT	Positive (0)	*bExecute* changes from FALSE to TRUE and *bBusy* is FALSE

- **Outputs**

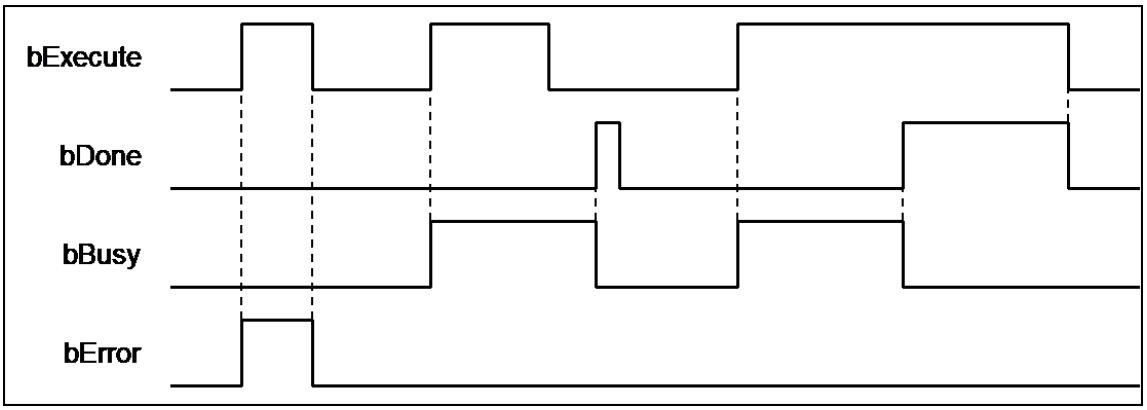
Name	Function	Data Type	Output Range (Default Value)
bDone	The trigger signal is TRUE and the axis position is recorded.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)
uiCount	The actual number of the master axis positions	UINT	UINT positive

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The number of corresponding or all master axis positions is obtained.	• *bExecute* turns to FALSE • *bError* turns to TRUE
bBusy	*bExecute* turns to TRUE.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		
ServoDriveError	Update information when *bExecute* is TRUE.	*bExecute* turns to FALSE.

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified cam table	MC_CAM_REF*	MC_CAM_REF (cannot be empty)	*bExecute* turns to TRUE.
alrData	The corresponding master axis position	ARRAY[*] OF LREAL	ARRAY[*] OF LREAL (cannot be empty)	*bExecute* turns to TRUE.

***Note**: MC_CAM_REF (FB): User-defined cam table parameters.

- **Function**
  - This function is supported by DL_MotionControl V1.4.1.0 or later.
  - The number of the master axis positions will be stored in the input/output parameter *alrData*, and the size can be defined within the limits of *uiSize* matrix.
  - When entering a position that is not in the cam table for *lrCamY*, this function block will report an error.

- *uiCount* is the number of the master axis positions based on the slave axis position, and this number will not exceed the number set by *uiSize*.
  - For example, *uiCount* = 2, if *alrData* is [0,100,0,0], it means that the master axis position that meets the slave axis position is 0 and 100 respectively.
- This function block is not recommended for use in EtherCAT Tasks.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
  - For the error codes and corresponding troubleshooting methods, refer to **Appendix** of this manual.

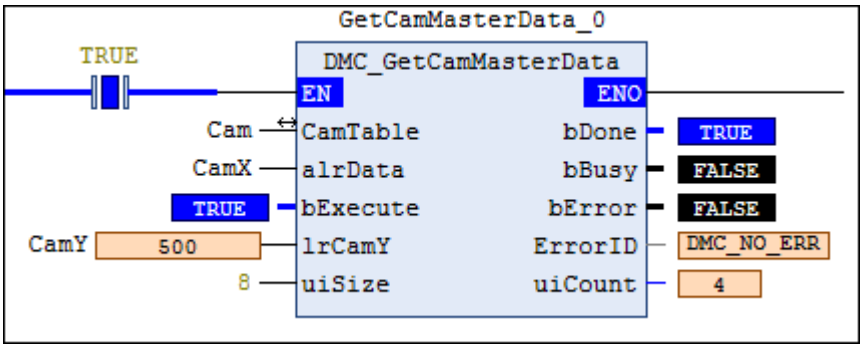
- **Example**

  This example shows how to use DMC_GetCamMasterData function block to get the master axis position in the cam table.

  - XYVA-Cam table

X	Y	V	A	J	Segm...
0	0	-5	5	0	
					Poly5
120	120	-5	-4.5	0	
					Poly5
240	240	1	0	0	
					Poly5
360	360	-5	5	0	

  - Function block input



CamX	ARRAY [0..9] OF LREAL	
CamX[0]	LREAL	24.337053593750284
CamX[1]	LREAL	47.314518074784132
CamX[2]	LREAL	265.583928099868
CamX[3]	LREAL	352.88814760002651
CamX[4]	LREAL	0
CamX[5]	LREAL	0
CamX[6]	LREAL	0
CamX[7]	LREAL	0
CamX[8]	LREAL	0
CamX[9]	LREAL	0

- Input/output alrData: Use the variable *CamX* for *alrData*. The data type of the variable is ARRAY[0..9] OF LREAL.
- When the slave axis position is 500, the number of mater axis positions is 4 (24.337, 47.314, 265.583, 352.888).
- If *uiSize*=3, the last master axis position (352.888) will not be obtained.

- **Supported Devices**
  - AX-series motion controller

2.3 DL_MotionControlLight

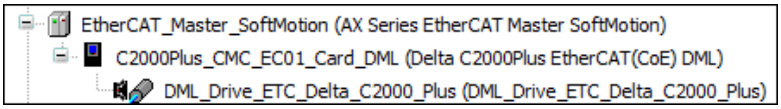
2.3.1 Positioning Axis and Velocity Axis Instructions

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.

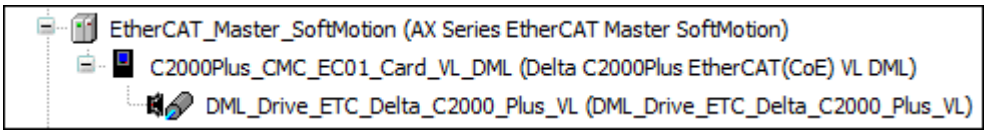
Software

The following is the AX-series controllers' motion control structure.

- **Positioning Axis**



- **Velocity Axis**



The following table describes the relationship between the EtherCAT axis type and the motion module of CiA 402.

Axis Type	CiA 402 Mode
Velocity Axis	VL, TQ
Positioning Axis	PP, PV, TQ, HM, VL

Note: The software interface is from DIADesiger-AX version 1.6.1. For more information about the mode instructions, see *AX-3 Series Operation Manual* Section 7.7.

The following table describes instructions of the supported axis type.

Instruction	Positioning Axis	Velocity Axis
MC_Power_DML	V	V
MC_Stop_DML	V	V
MC_Reset_DML	V	V
MC_Halt_DML	V	
MC_Home_DML	V	
MC_MoveAbsolute_DML	V	
MC_MoveRelative_DML	V	
MC_MoveVelocity_DML	V	
MC_WriteBoolParameter_DML	V	V
MC_ReadBoolParameter_DML	V	V
MC_WriteParameter_DML	V	V
MC_ReadParameter_DML	V	V
MC_ReadStatus_DML	V	V
MC_TorqueControl_DML*1	V	V
MC_ChangeAxisConfig_DML	V	
MC_ReinitDrive_DML	V	
MC_VelocityControl_DML*2	V	V

Note:

1. MS300, MH300, and MX300 are not supported.
2. ASDA-A2-E, ASDA-A3-E, ASDA-B3-E, ASDA-E3 and ASDA-W3 are not supported.

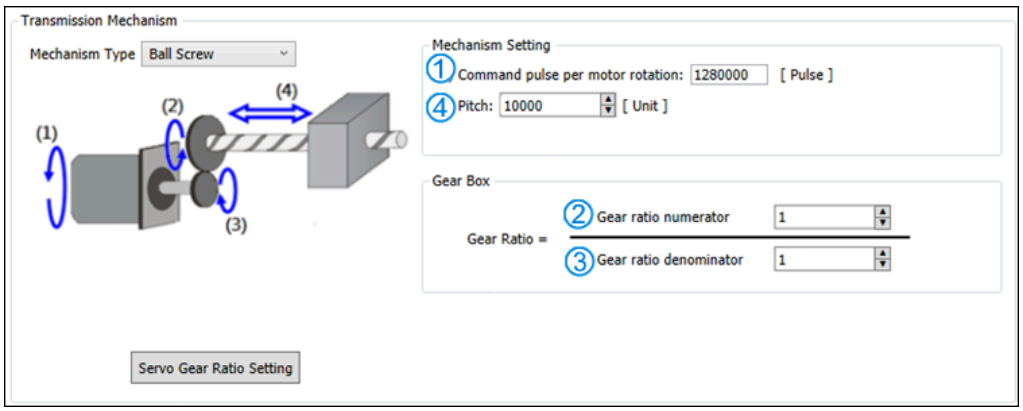
The following tables lists all the devices that support the positioning axis or/and velocity axis.

Device name	Positioning axis	Velocity axis
C2000	V	V
C2000+	V	V
MH300		V
MS300		V
A2-E	V	V
A3-E	V	V
B3-E	V	V
E3-E	V	V
R1-EC5621	V	V
W3	V	V
MX300		V

- **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

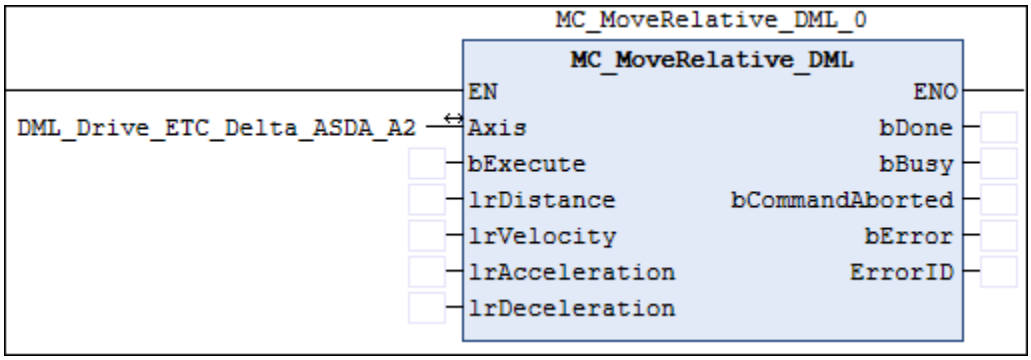


  - Take the above figure as a calculation example

    Factor = (①/②) / (③/④)

    - Positioning axis

      The maximum speed and maximum acceleration and deceleration range of the function block are calculated as follows:



Min. lrVelocity = (minimum revolutions of servo drive / 60) * (servo motor one-round resolution / Factor)

Max. lrVelocity = (rated revolutions of servo drive) / (servo motor one-round resolution / Factor)Min. lrAcceleration = ((3000 / 60)*1000) * (servo motor one-round resolution / Factor) / slowest acceleration time for servo drives

Max. lrAcceleration = ((3000 / 60)*1000) * (servo motor one-round resolution / Factor) / fatest acceleration time for servo drives

Min. lrDeceleration = ((3000 / 60)*1000) * (servo motor one-round resolution / Factor) / slowest deceleration time for servo drives

Max. lrDeceleration = ((3000 / 60)*1000) * (servo motor one-round resolution / Factor) / fatest deceleration time for servo drives

- Speed axis

  Min. lrVelocity = minimum revolutions of frequency converter / (Factor / Encoder) * 60)

  Max. lrVelocity = maximum revolutions of frequency converter / (Factor / Encoder) * 60)

- 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.

- **Positioning axis version and supported device**

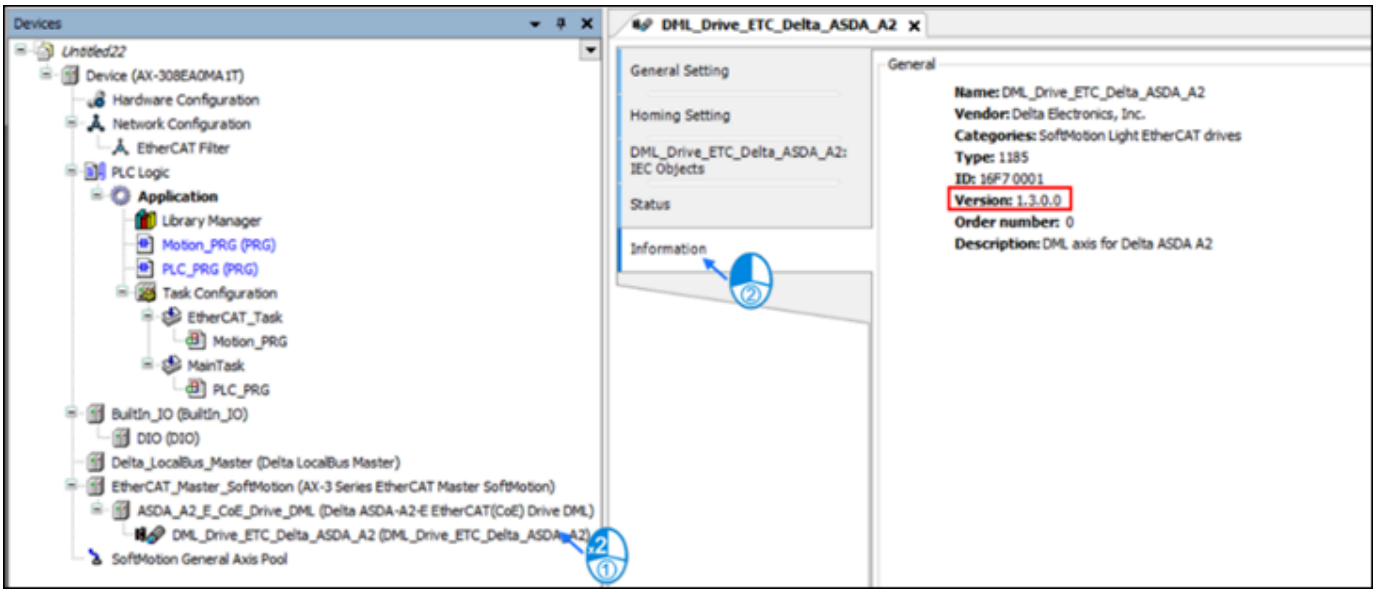
DML Axis DDF Version	Supported device							SoftMotion Version	Library Manager	
	C2000+CH20	A2	A3	B3	E3	W3	R1-EC5621		DML_Drive_ETC_Delta_XX*	DL_MotionControl Light
1.0.0.0		V	V	V				4.6.1.0	1.0.14	1.0.0.13
1.0.0.5		V		V			V	4.6.1.0	1.0.14	1.0.0.13

DML Axis DDF Version	Supported device							SoftMotion Version	Library Manager	
	C2000+ CH20	A2	A3	B3	E3	W3	R1-EC5621		DML_Drive_ETC_Delta_XX*	DL_MotionControl Light
1.0.0.6			V					4.6.1.0	1.0.14	1.0.0.13
1.0.0.14		V	V	V			V	4.6.1.0	1.0.14	1.0.0.13
1.1.0.0	V	V	V	V			V	4.6.1.0	1.1.0.0	1.1.0.0
1.3.0.0	V	V	V	V			V	4.6.1.0 4.10.0.0	1.3.0.0	1.1.0.0 1.3.0.0
1.3.2.0		V	V	V					1.3.0.0	1.1.0.0 1.3.0.0
1.4.0.0	V	V					V	4.6.1.0 4.10.0.0 4.16.0.0	1.4.0.0	1.4.0.0
1.4.0.1			V	V	V	V			1.4.0.0	1.4.0.0
1.4.1.2	V		V						1.4.2.0	1.4.0.0
1.4.2.0	V	V	V	V	V	V	V		1.4.2.0	1.4.2.0

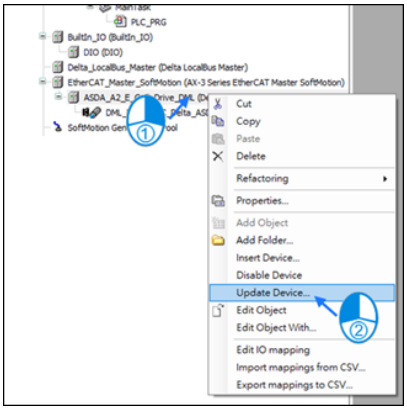
- **Velocity axis version and supported device**

DML Axis DDF Version	Supported device			SoftMotion Version	Library Manager	
	CH2000 C2000+	MH300 MS300	MX300		DML_Drive_ETC_Delta_XX*	DL_MotionControl Light
1.1.0.0	V	V		4.6.1.0	1.1.0.0	1.1.0.0
1.3.0.0	V	V		4.6.1.0 4.10.0.0	1.3.0.0	1.1.0.0 1.3.0.0
1.4.0.0	V			4.6.1.0 4.10.0.0 4.16.0.0	1.4.0.0	1.1.0.0 1.3.0.0
1.4.0.1		V			1.4.0.1	1.4.0.0
1.4.1.2	V	V			1.4.1.2	1.4.0.1
1.4.1.4			V		1.4.1.4	1.4.1.2
1.4.2.0	V	V	V		1.4.2.0	1.4.2.0

*__Note__: Double-click the third-level device, and then select **Information**. You can query the DML_Drive_ETC_Delta_XX version, which is determined by DML DDF. You can upgrade the version by right-clicking the device and select **Update Device**.



To upgrade DML DDF



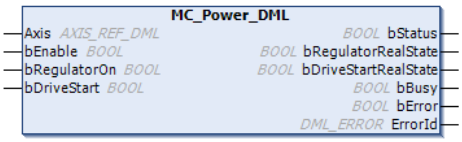
- **Devices that support DML**

Device	Firmware version
C2000+	V3.08 or later
C2000+ EtherCAT communication card	V4.00 or later
CH2000	V2.07 or later
CH2000 EtherCAT communication card	V3.00 or later
MH300 EtherCAT communication card	V1.04 or later
MS300 EtherCAT communication card	V1.04 or later
A2-E	V1.650 or later
A3-E	V2.1106 or later
B3-E	V2.1806 or later
E3-E	V1.0206 or later
R1-EC5621	V2.04 or later
W3	V1.0006 or later
MX300	V1.00 or later

- This library is not available in PLC simulation mode.

2.3.1.1 MC_Power_DML

MC_Power_DML enables, disables and quick stops the axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Power_DML		MC_Power_DML_instance(Axis : =, bEnable : =, bRegulatorOn: =, bDriveStart : =, bStatus =>, bRegulatorRealState =>, bDriveStartRealState =>, bBusy =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
bRegulatorOn	Power ON	BOOL	TRUE/FALSE (FALSE)	Only valid when *bEnable* is TRUE
bDriveStart	Disable the quickstop mechanism.	BOOL	TRUE/FALSE (FALSE)	Only valid when *bEnable* is TRUE

- **Outputs**

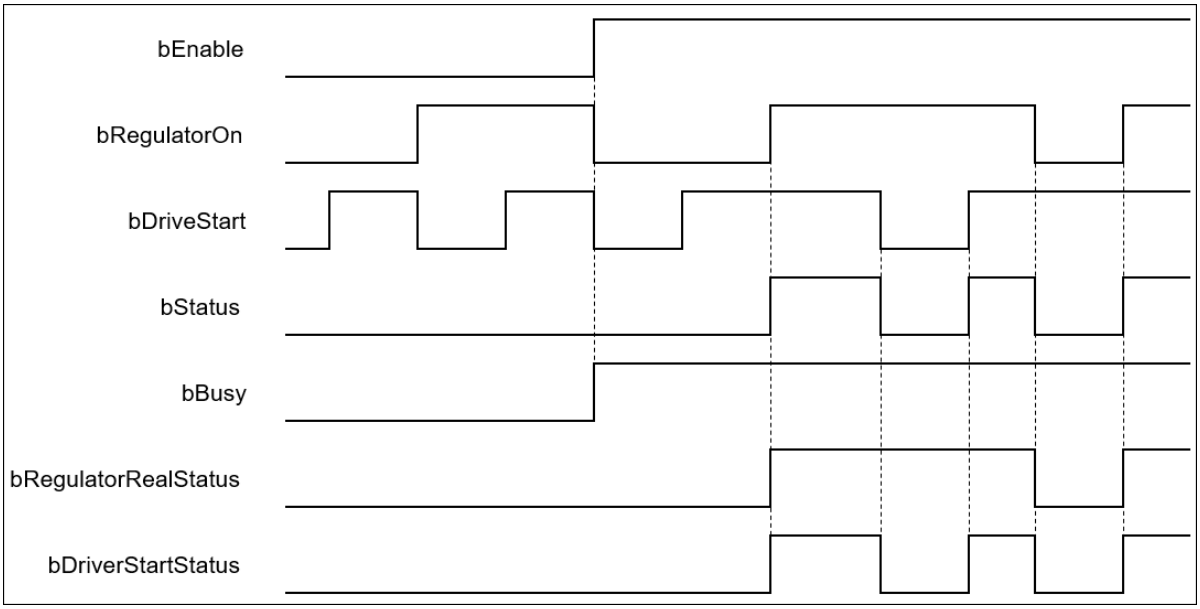
Name	Function	Data Type	Output Range (Default Value)
bStatus	TRUE when the specified axis can move	BOOL	TRUE/FALSE (FALSE)
bRegulatorRealState	TRUE when the power is ON	BOOL	TRUE/FALSE (FALSE)
bDriveStartRealState	TRUE when the quick stop mechanism can be used	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	*bEnable* is TRUE and *bRegulatorRealState* and *bDriveStartRealState* turn to TRUE.	• bEnable is TRUE and *bRegulatorRealState* or *bDriveStartRealState* turns to FALSE. • *bError* turns to TRUE.
bRegulatorRealState	*bEnable* and *bRegulatorRealState* are TRUE.	• *bEnable* is TRUE and *bRegulatorRealState* turns to FALSE. • *bError* turns to TRUE.
bDriveStartRealState	*bEnable*, *bRegulatorRealState*, and *bDriveStartRealState* are TRUE.	• *bEnable* is TRUE, and *bRegulatorRealState* or *bDriveStartRealState* turns to FALSE. • *bError* turns to TRUE.
bBusy	*bEnable* turns to TRUE.	• *bEnable* turns to FALSE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	Error Code is cleared.
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

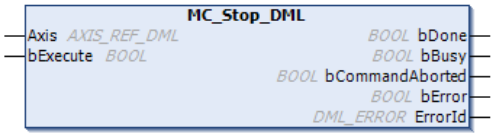
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*bEnable* turns to TRUE.

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

- **Function**
  - *bRegulatorOn* and *bDriveStart* will take effect only when *bEnable* is TRUE.
  - When *bEnable*, *bRegulatorOn*, and *bDriveStart* are TRUE, *bStatus* turns to TRUE and *nAxisState* (state machine) turns to Standstill.
  - When *bEnable* and *bRegulatorOn* are TRUE and then *bDriveStart* is set to FALSE, *nAxisState* (state machine) turns to Stopping.
  - When *bEnable* and *bDriveStart* are TRUE and then *bRegulatorOn* is set to FALSE, *nAxisState* (state machine) turns to Disabled.
  - When the axis state machine is under the Standstill status, Delta servo ASDA-xx-E Series runs MC_Stop_DML, and *bStatus* of MC_Power_DML turns to FALSE.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the programming example of the MC_Power function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.2 MC_Stop_DML

MC_Stop_DML decelerates the axis to a stop.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Stop_DML		MC_Stop_DML_instance(Axis : =, bExecute : =, bDone =>, bBusy =>, bCommandAborted=>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the velocity reaches 0	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

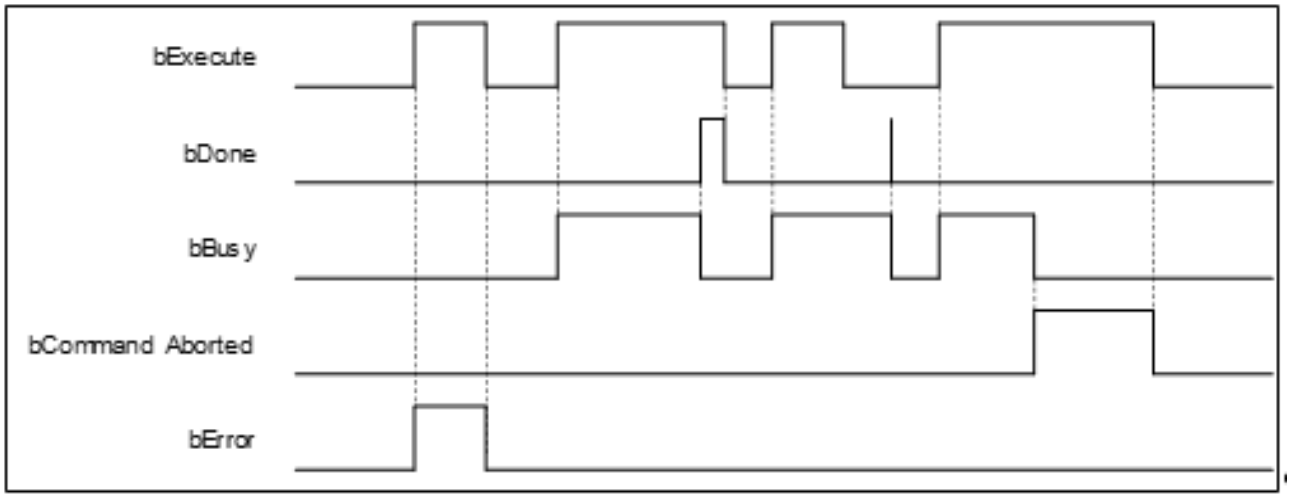
***Note**: DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The axis decelerates to a stop or the velocity is 0.	• *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 turn to FALSE.
bCommandAborted	The axis state switches to Disabled during instruction running.	• *bExecute* turns to FALSE. • If *bExecute* is FALSE and *bCommandAborted* is TRUE,

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
		bCommandAborted will immediately change to FALSE after maintaining a TRUE state for a scan cycle.
bBusy	*bExecute* turns to TRUE and the instruction is running	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*bExecute* turns to TRUE and *bBusy* is FALSE

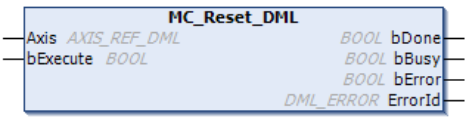
*__Note__: AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**
  - MC_Stop_DML is used to stop the axis in motion, and the state machine enters the stopping status.
  - 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**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**
  - For the example, refer to the programming example of the MC_Stop function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.3 MC_Reset_DML

MC_Reset_DML clears axis errors.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Reset_DML		MC_Reset_DML_instance(Axis : =, bExecute : =, bDone =>, bBusy =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when axis error clearing is completed and the axis enters Standstill or Disabled status	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

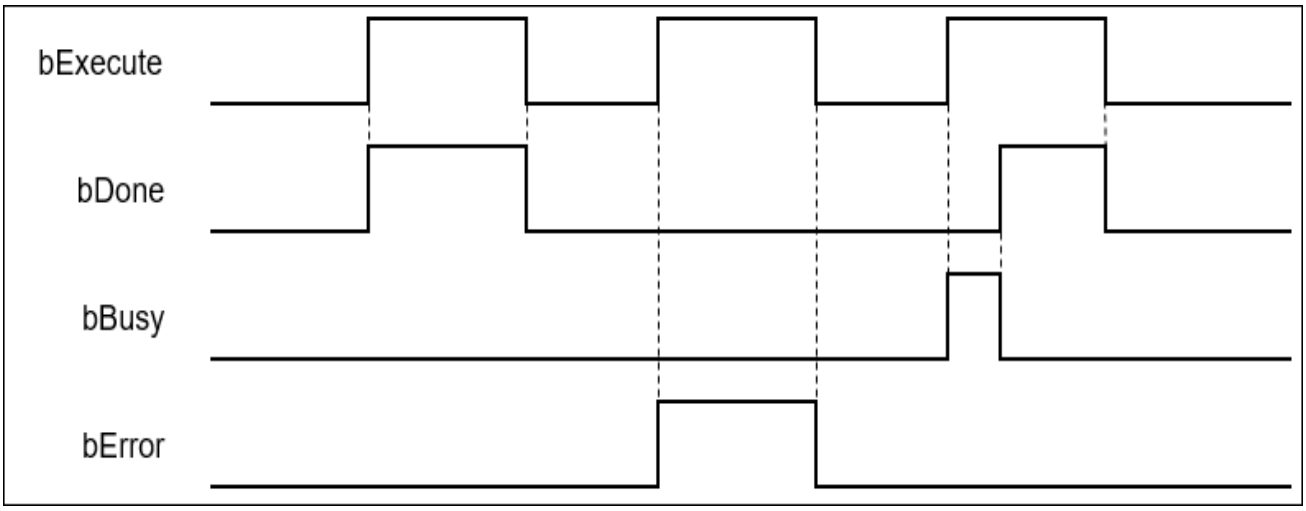
***Note**: DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	Axis error clearing is completed.	• *bExecute* turns to FALSE. • If *bExecute* is FALSE and *bDone* turns to TRUE, *bDone* will be TRUE for one period and then immediately turn to FALSE.
bBusy	The instruction is running.	• *bError* turns to TRUE • *bDone* turns to TRUE

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*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 status to a running status.

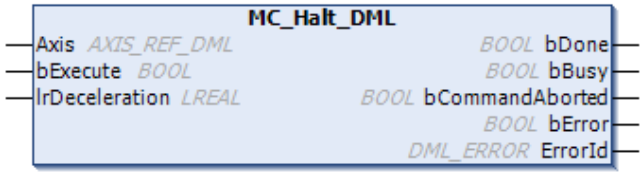
    When *MC_Power_DML.bEnable* is TRUE, the axis changes from Errorstop to Standstill.

    When *MC_Power_DML.bEnable* is FALSE, the axis 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 will return to the Standstill or Disabled status.
  - 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**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the programming example of the MC_Reset function block.
  - In this function block, input the positioning axis to the Axis parameter.

- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.4 MC_Halt_DML

MC_Halt_DML halts an axis in a controllable way.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Halt_DML		MC_Halt_DML_instance(Axis : =, bExecute : =, lrDeceleration : =, bDone =>, bBusy =>, bCommandAborted =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrDeceleration	Deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the axis stops and the velocity is 0	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

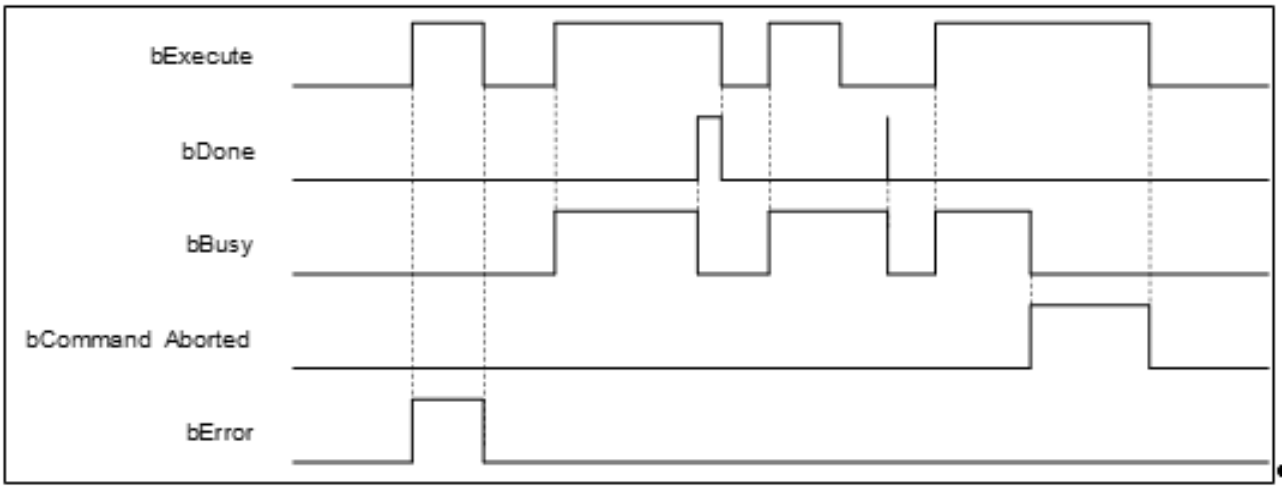
***Note**: DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The axis decelerates to a stop.	• *bExecute* turns to FALSE. • If *bExecute* is FALSE and *bDone* turns to TRUE, *bDone* will be TRUE for one period and then immediately turn to FALSE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	*bExecute* turns to TRUE and the instruction is run	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bCommandAborted	This instruction is interrupted by another function block	• *bExecute* turns to FALSE • If *bExecute* is FALSE and *bCommandAborted* turns to TRUE, *bCommandAborted* will be TRUE for one period and immediately turn to FALSE.
bError ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*bExecute* turns to TRUE and *bBusy* is FALSE.

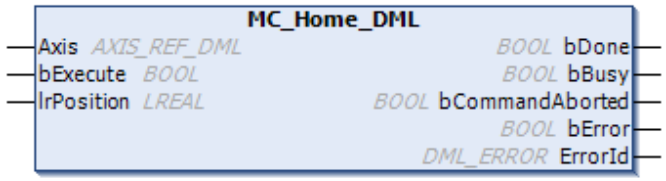
***Note**: AXIS_REF_DML (FB): All function blocks contain this variable, which works as the starting program for function blocks.

- **Function**
  - Different from MC_Stop_DML, MC_Halt_DML can be interrupted by other motion function blocks when stopping a motion.
  - When MC_Halt_DML is running, the axis will enter the discrete_motion status. When the velocity of the axis reaches zero, the axis will enter the Standstill status.
  - When using the ASDA-A2-E, ASDA-A3-E, and ASDA-B3-E slaves, the axis stops immediately when the MC_Halt_DML stops in the PP Mode.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the programming example of the MC_Halt function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.5 MC_Home_DML

MC_Home_DML controls the axis to perform the homing operation.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Home_DML		MC_Home_DML_instance(Axis : =, bExecute: =, lrPosition: =, bDone =>, bBusy =>, bCommandAborted =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrPosition	The absolute home position (Unit: user unit)	LREAL	Positive, negative, or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.

- **Outputs**

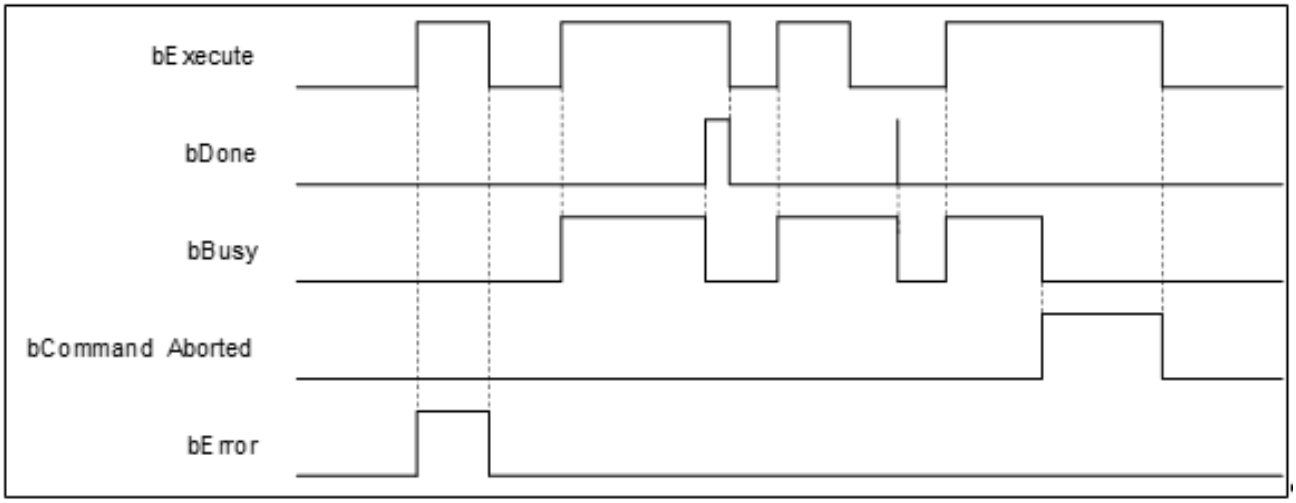
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when homing is completed and the axis is in Standstill status	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	Homing is complete.	• *bExecute* turns to FALSE. • If *bExecute* is FALSE and *bDone* turns to TRUE, *bDone* will be TRUE for one period and then immediately turn to FALSE.
bBusy	*bExecute* turns to TRUE and the instruction is run.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bCommandAborted	• This instruction is interrupted by another instruction. • The instruction is interrupted by MC_Stop_DML	• *bExecute* turns to FALSE. • If *bExecute* is FALSE and *bCommandAborted* turns to TRUE, *bCommandAborted* will be TRUE for one period and immediately turn to FALSE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	When Error Code is cleared
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

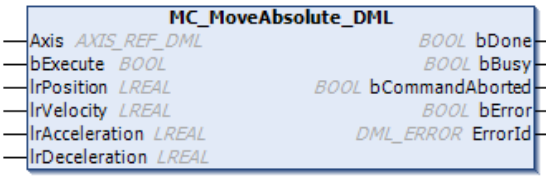
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*bExecute* turns to TRUE and *bBusy* is FALSE.

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

- **Function**
  - This function block is run only when the axis is in the Standstill status. The axis is in the Homing status during execution. The function block cannot be run when the axis is in other status.
  - Position is the absolute position when homing is complete.
  - The home mode can be selected from the axis parameter page.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the programming example of the MC_Home function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.6 MC_MoveAbsolute_DML

MC_MoveAbsolute_DML controls the axis to move to the absolute target position based on the specified motion behavior.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_MoveAbsolute_DML		MC_MoveAbsolute_DML_instance(Axis : =, bExecute : =, lrPosition : =, lrVelocity : =, lrAcceleration : =, lrDeceleration : =, bDone =>, bBusy =>, bCommandAborted =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrPosition	Absolute target position (Unit: user unit)	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrVelocity	Target velocity (Unit: user unit/s)	LREAL	Positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrAcceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrDeceleration	Deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the absolute target position is reached	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)

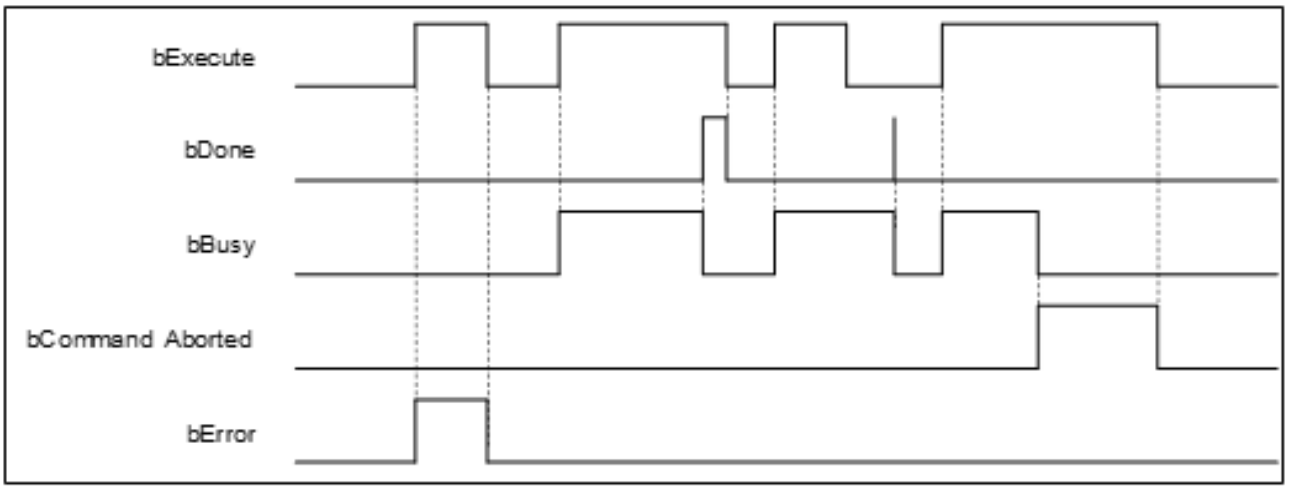
Name	Function	Data Type	Output Range (Default Value)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

***Note**: DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The absolute target position is reached.	• *bExecute* turns to FALSE. • If *bExecute* is FALSE and *bDone* turns to TRUE, *bDone* will be TRUE for one period and then immediately turn to FALSE.
bBusy	*bExecute* turns to TRUE and the instruction is running.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bCommandAborted	• This instruction is interrupted by another instruction • The instruction is interrupted by MC_Stop_DML.	• *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 ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

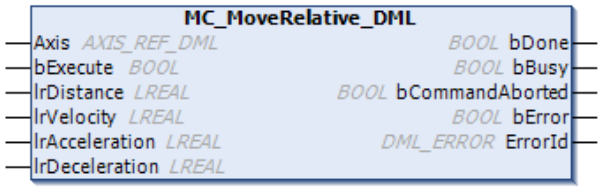
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*bExecute* turns to TRUE and *bBusy* is FALSE.

***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**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the programming example of the MC_MoveAbsolute function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.7 MC_MoveRelative_DML

MC_MoveRelative_DML controls the axis to move to the relative target position according to the specified motion behavior.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_MoveRelative_DML		MC_MoveRelative_DML_instance(Axis : =, bExecute : =, lrDistance : =, lrVelocity : =, lrAcceleration : =, lrDeceleration : =, bDone =>, bBusy =>, bCommandAborted =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrDistance	Relative distance to be moved (Unit: user unit)	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrVelocity	Target velocity (Unit: user unit/s)	LREAL	Positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrAcceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrDeceleration	Deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the relative distance is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)

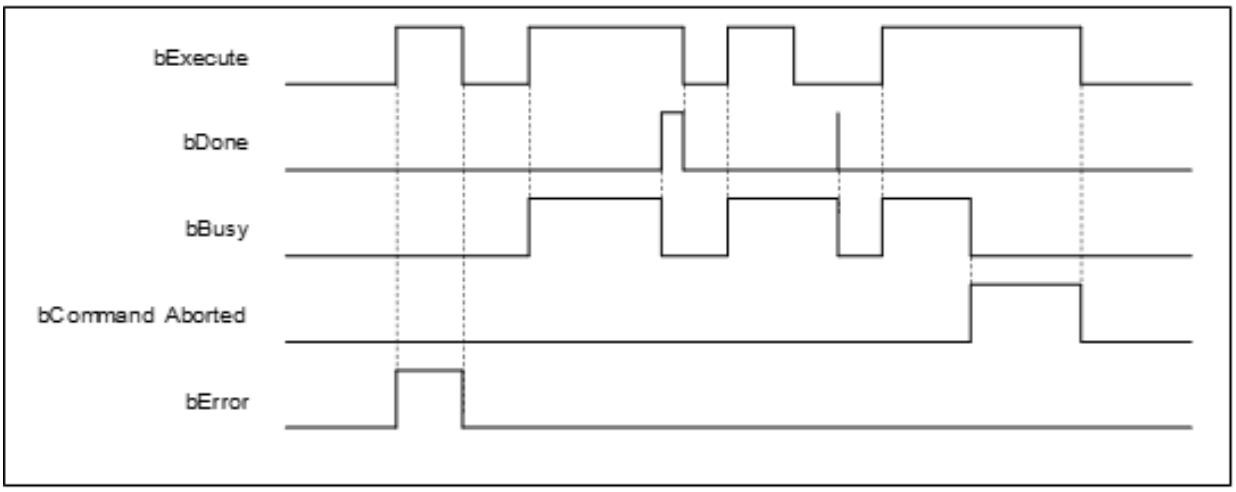
Name	Function	Data Type	Output Range (Default Value)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

***Note**: DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The relative positioning is complete.	• *bExecute* turns to FALSE. • If *bExecute* is FALSE and *bDone* turns to TRUE, *bDone* will be TRUE for one period and immediately turn to FALSE.
bBusy	*bExecute* turns to TRUE and the instruction is running.	• *bDone* turns to TRUE. • *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bCommandAborted	• This instruction is interrupted by another instruction. • The instruction is interrupted by MC_Stop_DML.	• *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 ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

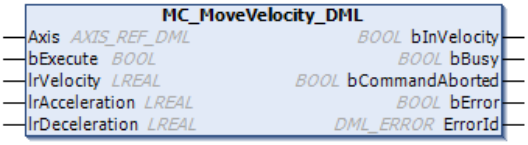
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*bExecute* turns to TRUE and *bBusy* is FALSE.

***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**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the programming example of the MC_MoveRelative function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.8 MC_MoveVelocity_DML

MC_MoveVelocity_DML controls the axis in the position mode to move at a constant velocity according to the specified motion mode and velocity.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_MoveVelocity_DML		MC_MoveVelocity_DML_instance(Axis : =, bExecute : =, lrVelocity : =, lrAcceleration : =, lrDeceleration : =, bInVelocity =>, bBusy =>, bCommandAborted =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrVelocity	Target velocity (Unit: user unit/s)	LREAL	Negative, positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrAcceleration	Acceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrDeceleration	Deceleration rate (Unit: user unit/s^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.

- **Outputs**

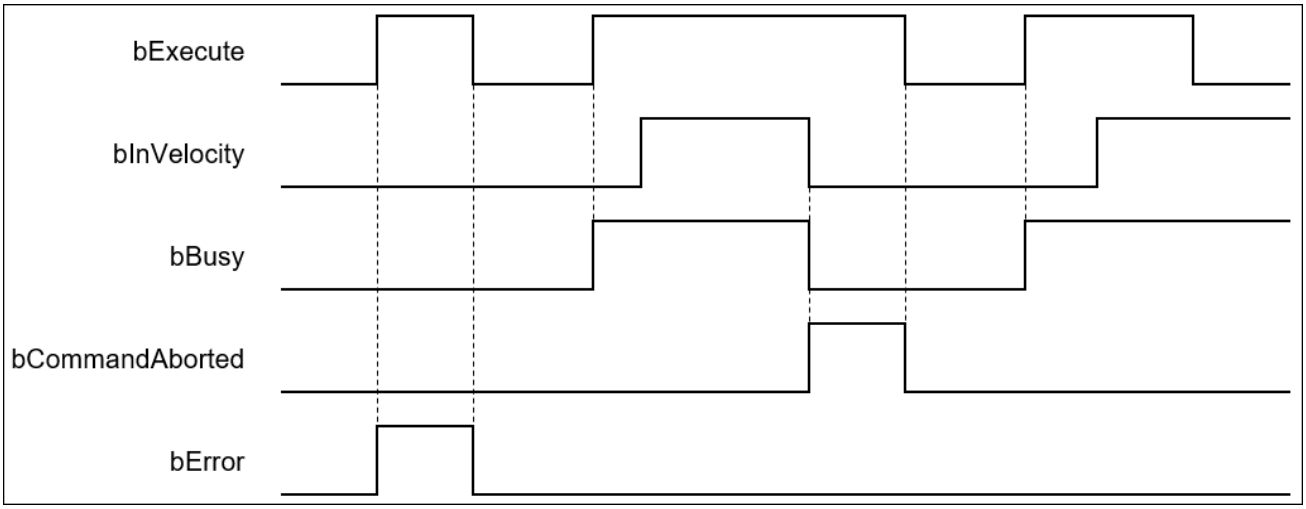
Name	Function	Data Type	Output Range (Default Value)
bInVelocity	TRUE when the target velocity is reached	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The target velocity is reached.	• *bCommandAborted* turns to TRUE. • *bExecute* turns to TRUE again and *lrVelocity* value is changed.
bBusy	*bExecute* turns to TRUE and the instruction is run.	• *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bCommandAborted	• This instruction is interrupted by another instruction. • The instruction is interrupted by MC_Stop_DML.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*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*).
  - MC_MoveVelocity_DML can be interrupted by another motion instruction.

- When the instruction is interrupted by another instruction, the output *bInVelocity* changes to FALSE, and the output *bCommandAborted* changes to TRUE.
- When *bExecute* of MC_MoveVelocity_DML changes 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 *bExecute* 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. *bInVelocity* will be TRUE until it is interrupted by another instruction.
- When the 0x60FF (Target Velocity) is configured in the PDO, if the input speed exceeds the value of the 0x60FF range, the motor will not run. For detailed range values, refer to the slave manual.
- Enter positive and negative values for *lrVelocity* to set the direction of the motor.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the programming example of the MC_MoveVelocity function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.9 MC_WriteBoolParameter_DML

MC_WriteBoolParameter_DML writes a Boolean value to the specified parameter.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_WriteBoolParameter_DML		MC_WriteBoolParameter_instance(Axis : =, bExecute : =, diParameterNumber : =, bValue : =, bDone =>, bBusy =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
diParameterNumber	The number of the axis parameter.	DINT	Positive, negative, or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
bValue	The Boolean value of the parameter to write	BOOL	TRUE/FALSE (FALSE)	*bExecute* turns to TRUE and *bBusy* is FALSE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the parameter writing is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

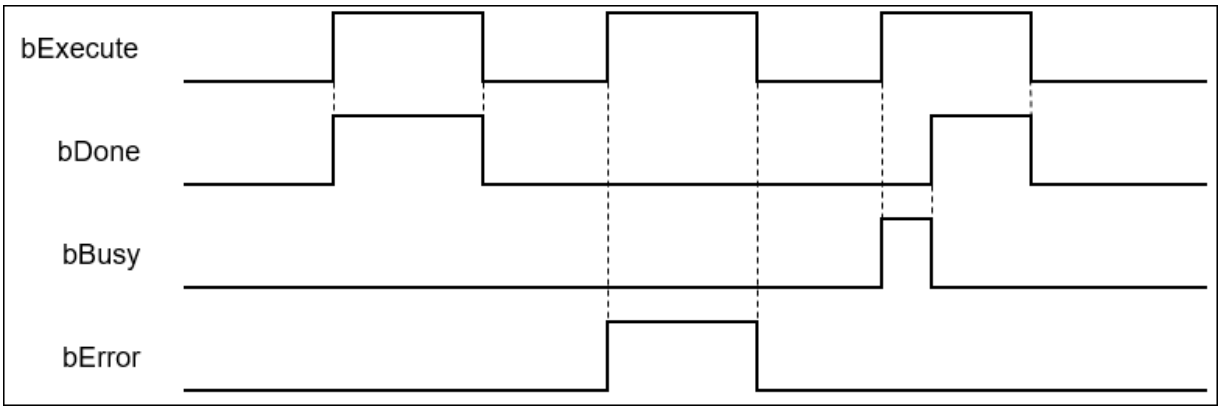
***Note**: DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	When the parameter writing is complete	*bExecute* turns from TRUE to FALSE.
bBusy	• *bExecute* turns to TRUE and the instruction is running. • Writing parameter is in progress.	• *bDone* turns to TRUE. • *bError* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*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**
  - 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.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the programming example of the MC_WriteParameter function block.
  - In this function block, input the positioning axis to the Axis parameter.

- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.10 MC_ReadBoolParameter_DML

MC_ReadBoolParameter_DML reads the Boolean value of a specified parameter.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_ReadBoolParameter_DML		MC_ReadBoolParameter_DML_instance(Axis : =, bEnable : =, diParameterNumber : =, bValid =>, bBusy =>, bError =>, ErrorID =>, bValue =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
diParameterNumber	The number of the axis parameter	DINT	Positive, negative, or 0 (0)	*bEnable* turns to TRUE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bValid	TRUE when the read parameter value is available	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)
bValue	The read parameter value	BOOL	TRUE/FALSE (FALSE)

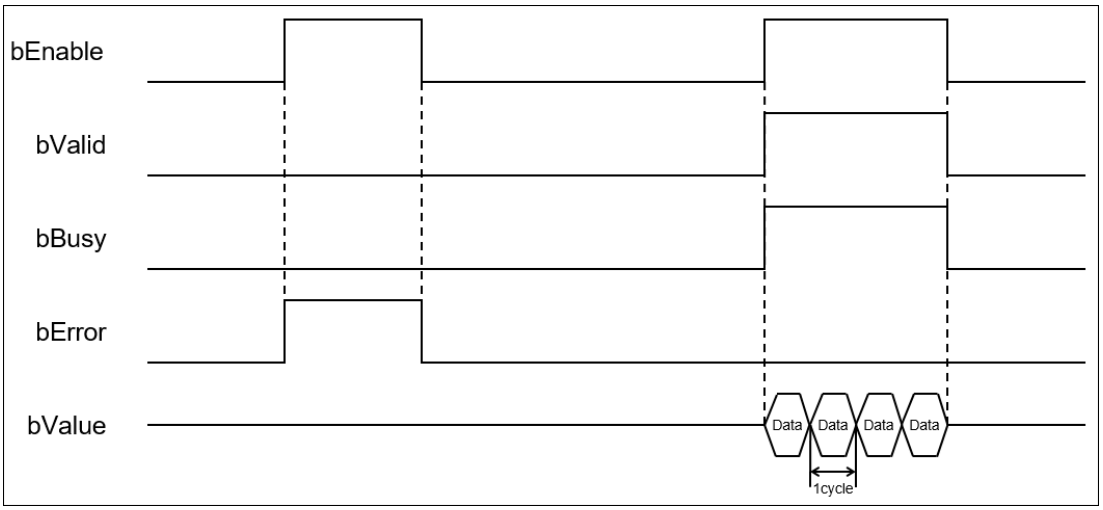
***Note**: DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bValid	• *bEnable* turns to TRUE. • The parameter to be read is available.	• *bEnable* turns from TRUE to FALSE. • *bError* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	• *bEnable* turns to TRUE and the instruction is running. • The parameter to be read is available.	• *bEnable* turns from TRUE to FALSE. • *bError* turns to TRUE
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	When *bEnable* is FALSE. (Error Code is cleared.)
ErrorID		
bValue	Updates continuously when *bValid* is TRUE.	Update stops when *bValid* is FALSE.

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*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.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the programming example of the MC_ReadParameter function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.11 MC_WriteParameter_DML

MC_WriteParameter_DML writes a value in the parameter.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_WriteParameter_DML		MC_WriteParameter_DML_instance(Axis : =, bExecute : =, diParameterNumber : =, lrValue : =, bDone =>, bBusy =>, bError =>, ErrorID =>)

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
diParameter Number	The number of the axis parameter	DINT	Positive, negative, or 0 (0)	*bExecute* is TRUE and *bBusy* is FALSE.
lrValue	The parameter value to write	LREAL	Positive, negative, or 0 (0)	*bExecute* is TRUE and *bBusy* is FALSE.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the parameter writing is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)

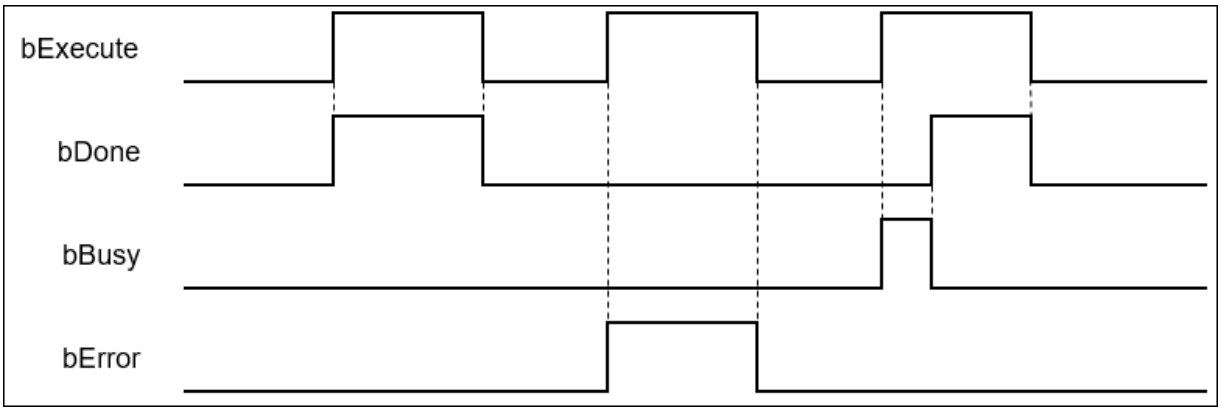
***Note**: DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The parameter writing is complete.	*bExecute* turns from TRUE to FALSE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	• *bExecute* turns to TRUE and the instruction is run. • Parameter writing is in progress.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*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**

  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.

- **Troubleshooting**

  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**
  - For the example, refer to the programming example of the MC_WriteParameter function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.12 MC_ReadParameter_DML

MC_ ReadParameter_DML reads the value from a parameter.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_ReadParameter_DML		MC_ReadParameter_DML_instance(Axis : =, bEnable : =, diParameterNumber : =, bValid =>, bBusy =>, bError =>, ErrorID =>, lrValue =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bEnable	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
diParameterNumber	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 Value)
bValid	TRUE when the read parameter value is available	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*	DML_ERROR (DML_NO_ERROR)
lrValue	The read parameter value	LREAL	Positive, negative, or 0 (0)

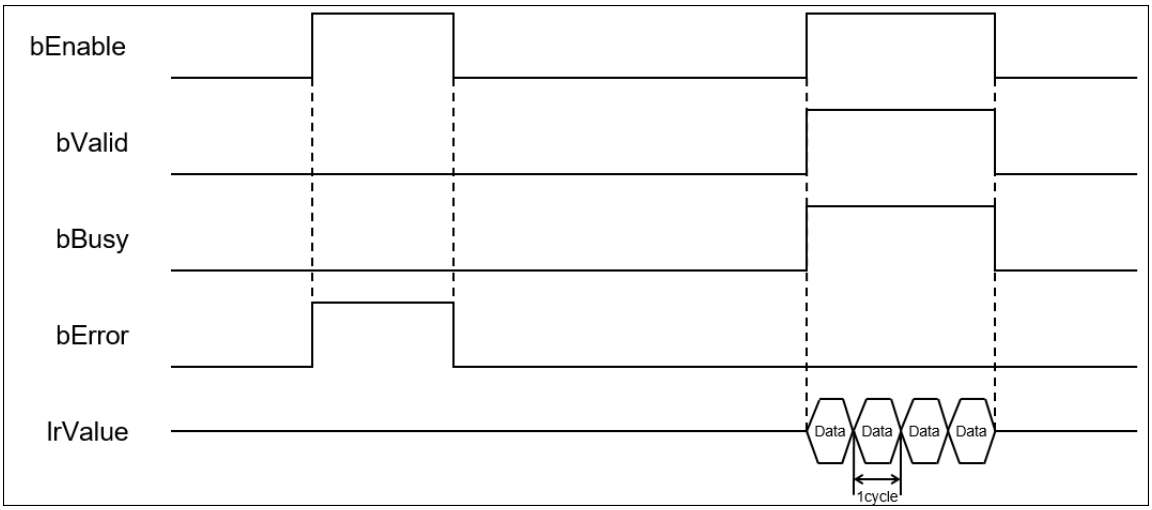
***Note**: DML_ERROR: Enumeration (ENUM)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bValid	• *bEnable* turns to TRUE. • The read parameter value is available.	• *bEnable* turns from TRUE to FALSE. • *bError* turns to TRUE.

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	• *bEnable* turns to TRUE and the instruction is run. • The read parameter value is available.	• *bEnable* turns from TRUE to FALSE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* is FALSE. (Error Code is cleared.)
ErrorID		
lrValue	Update continuously when *bValid* is TRUE.	Update stops when *bValid* is FALSE.

- **Timing Diagram of Output Parameter Changes**



***Note**:

1. Data = Parameter values
2. 1 cycle = One task cycle

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*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**

  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.

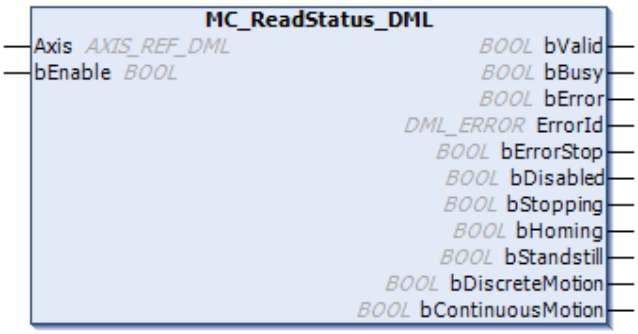
    See the reference formula as follows.

    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.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the programming example of the MC_ReadParameter function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.13 MC_ReadStatus_DML

MC_ReadStatus_DML reads the status of an axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_ReadStatus_DML		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 Value)	Timing to Take Effect
bEnable	Runs the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bValid	TRUE when the axis status at the output is available	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*1	DML_ERROR (DML_NO_ERROR)
bErrorStop	To know details on the axis status 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	• *bEnable* turns to TRUE. • The axis status at the output is available.	• *bEnable* turns from TRUE to FALSE. • *bError* turns to TRUE.
bBusy	*bEnable* turns to TRUE and the instruction is running.	• *bEnable* turns from TRUE to FALSE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bEnable* is FALSE. (Error Code is cleared.)
ErrorID		
bDisabled	The axis is in Disabled status.	The axis is not in Disabled status.
bErrorstop	The axis is in Errorstop status.	The axis is not in Errorstop status.
bStopping	The axis is in Stopping status.	The axis is not in Stopping status.
bStandStill	The axis is in StandStill status.	The axis is not in StandStill status.
bDiscreteMotion	The axis is in Discrete Motion status.	The axis is not in Discrete Motion status.
bContinuousMotion	The axis is in Continuous Motion status.	The axis is not in Continuous Motion status.
bHoming	The axis is in Homing status.	The axis is not in Homing status.

- **Inputs/Outputs**

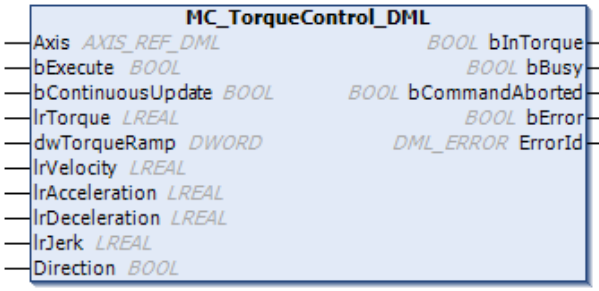
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*bEnable* turns to TRUE.

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

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the programming example of the MC_ReadStatus function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.14 MC_TorqueControl_DML

MC_TorqueControl_DML controls the servo drive to perform torque control according to its torque control mode.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_TorqueControl		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 Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
bContinuousUpdate	Continuously updates the target torque when *bContinuousupdate* is TRUE	BOOL	TRUE/FALSE (FALSE)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrTorque	The target torque (Unit: N.m)	LREAL	Positive, negative, or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
dwTorqueRamp	The change rate of the torque from current torque to target torque (Unit: ms) *	DWORD	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrVelocity	The maximum velocity	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrAcceleration	Reserved	LREAL	-	-
lrDeceleration	Reserved	LREAL	-	-
lrJerk	Reserved	LREAL	-	-
Direction	Reserved	BOOL	-	-

***Note**: Here taking ASDA-A2 as an example with the unit of microsecond. For other servo models, refer to 0x6087 in the object dictionary.

- **Outputs**

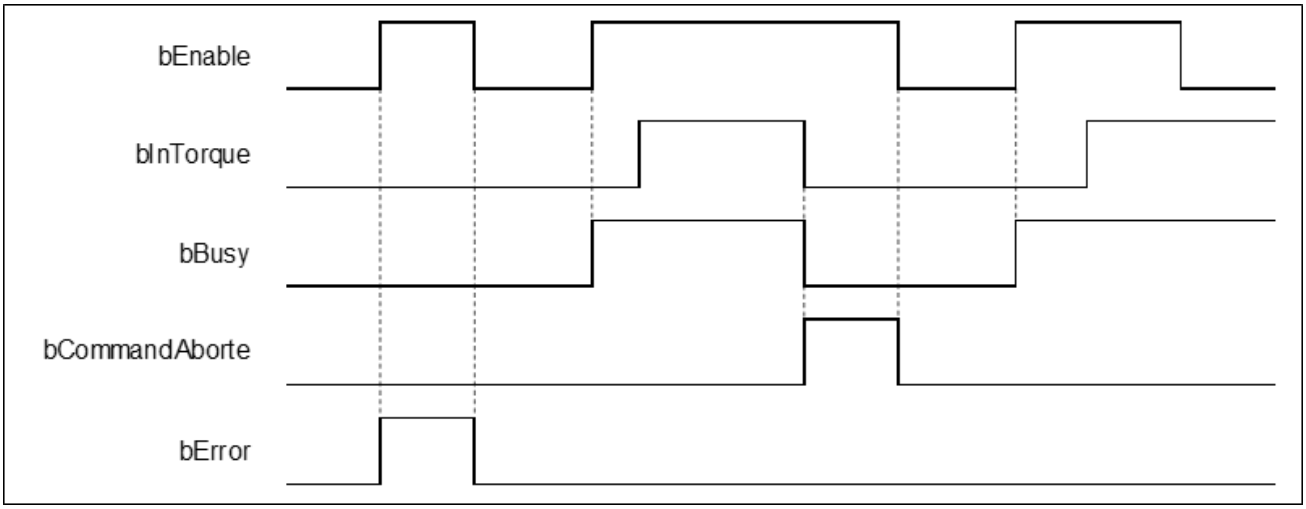
Name	Function	Data Type	Output Range (Default Value)
bInTorque	TRUE when the target torque is reached	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bCommandAborted	TRUE when the instruction is aborted	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	*bExecute* turns to TRUE and the axis state is available.	• *bError* turns to TRUE. • *bCommandAborted* turns to TRUE. • *bExecute* turns to TRUE again and *lrTorque* value changes.
bBusy	*bExecute* turns to TRUE and the instruction is running.	• *bError* turns to TRUE. • *bCommandAborted* turns to TRUE.
bCommandAborted	The instruction is aborted.	• *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	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*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, 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 runs for a velocity-constant motion according to the specified target velocity (*lrVelocity*), acceleration rate (*lrAcceleration*), and deceleration rate (*lrDeceleration*).
  - MC_TorqueControl_DML can be interrupted by another motion instruction.
  - When MC_TorqueControl_DML is interrupted by another motion instruction, the output *bInTorque* will turn to FALSE and the output *bCommandAborted* will turn 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 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 changed to the new torque value.
  - When *bExecute* turns to FALSE after running the instruction and the target torque is reached, *bInTorque* of MC_TorqueControl_DML turns to TRUE. *bInTorque* will remain TRUE until it is interrupted 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**

  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**
  - For the example, refer to the programming example of the DMC_TorqueControl function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.15 MC_ChangeAxisConfig_DML

MC_ChangeAxisConfig_DML modifies basic axis settings, including user unit to pulse number ratio, axis type and user units per rotary axis rotation.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_ChangeAxisConfig_DML		MC_ChangeAxisConfig_DML_instance(Axis : =, bExecute : =, dwRatioTechUnitsDenom : =, iRatioTechUnitsNum : =, fModuloPeriodU : =, fMovementType : =, bDone =>, bBusy =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
dwRatioTechUnitsDenom	Electronic gear ratio denominator (Pulse number)	DWORD	Positive or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
iRatioTechUnits Num	Electronic gear ratio numerator (User units)	DINT	Positive, negative, or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
fModuloPeriodU	Maximum position of the rotary axis	LREAL	Positive, negative, or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
fMovementType	Linear axis/rotary axis	SML_MOVEMENTTYPE	0: SML_MT_MODULO 1: SML_MT_FINITE	*bExecute* turns to TRUE and *bBusy* is FALSE.

- **Outputs**

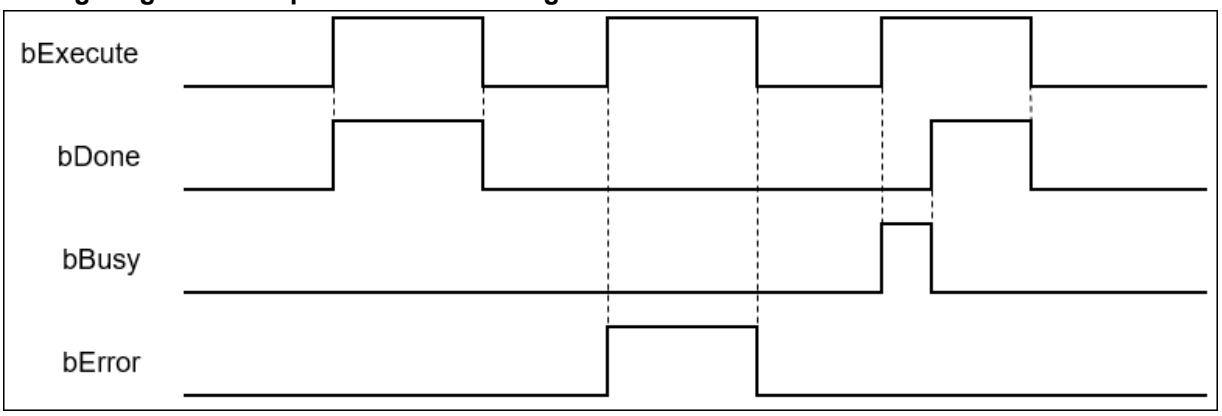
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the parameter writing is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The parameter writing is completed.	*bExecute* turns from TRUE to FALSE.
bBusy	• *bExecute* turns to TRUE and the instruction is running. • Writing parameter is in progress.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



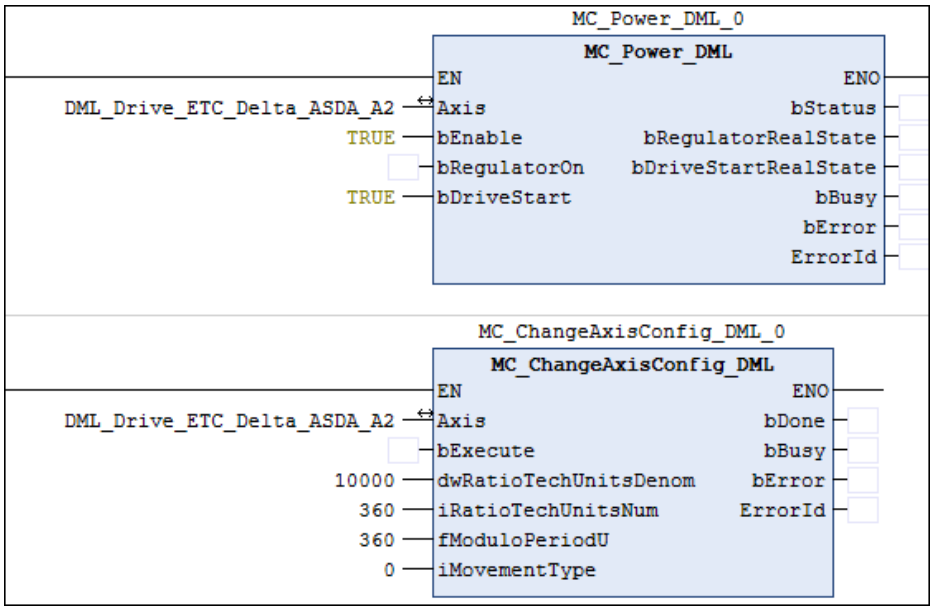
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*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_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**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  This example explains how MC_ChangeAxisConfig_DML is used to modify axis parameters.

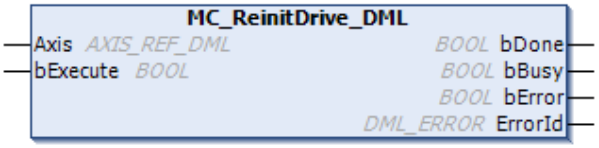


- Set the gear ratio denominator and gear ratio numerator (10000: 360), maximum position of the rotary axis (360) and axis type (0). Change *bRegulatorOn* of MC_Power to FALSE before running MC_ChangeAxisConfig.

- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.16 MC_ReinitDrive_DML

MC_ReinitDrive_DML reinitializes the axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_ReinitDrive_DML		MC_ReinitDrive_DML_instance(Axis : =, bExecute : =, bDone =>, bBusy =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-

- **Outputs**

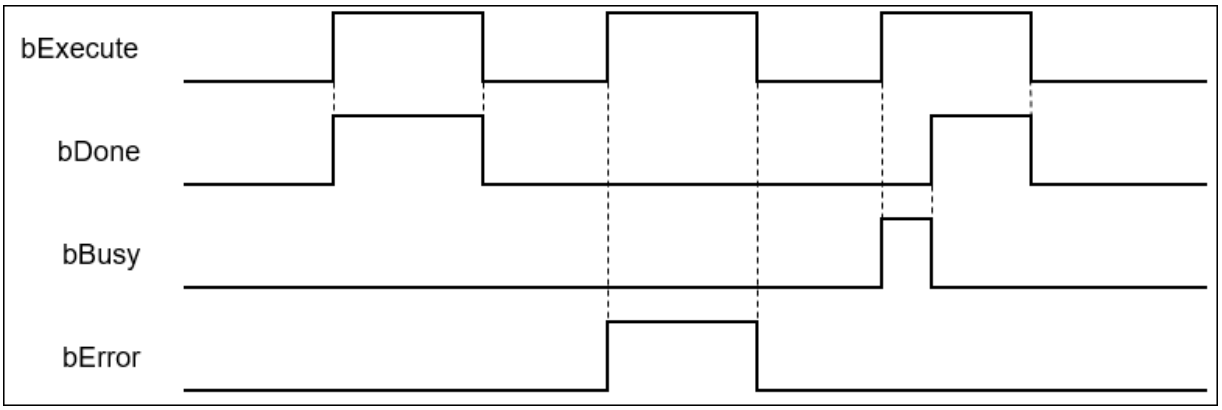
Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when initialization is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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	The initialization is complete.	*bExecute* turns from TRUE to FALSE.
bBusy	• *bExecute* turns to TRUE and the instruction is running. • The initialization is in progress.	• *bDone* turns to TRUE. • *bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*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.

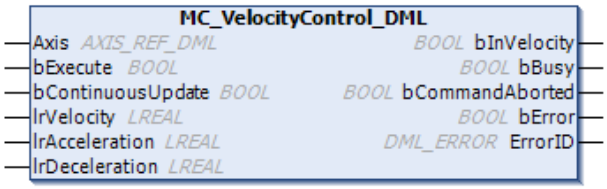
- **Function**
  - MC_ReinitDrive_DML is used to initialize the drive that has an error or lost synchronization.
  - Resetting the network cannot clear errors in the positioning axis itself,such as left and right limits, emergency stops.
  - The function block resets EtherCAT network of the positioning axis and keeps the servo in its previous status. 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**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.1.17 MC_VelocityControl_DML

MC_VelocityControl_DML controls the specified axis to move evenly according to the specified motion mode and speed in the velocity mode (VL).

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_VelocityControl_DML		MC_VelocityControl_DML_instance(Axis : =, bExecute : =, bContinuousUpdate : =, lrVelocity : =, lrAcceleration : =, lrDeceleration : =, bInVelocity =>, bBusy =>, bCommandAborted =>, bError =>, ErrorID =>)

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
bContinuousUpdate*	When *bContinuousUpdate* is TRUE, the target speed will be updated continuously.	BOOL	TRUE/FALSE (FALSE)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrVelocity	Target speed (user unit/sec)	LREAL	Negative, positive, or 0 (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrAcceleration	Acceleration (user unit/sec^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrDeceleration	Deceleration (user unit/sec^2)	LREAL	Positive (0)	*bExecute* turns to TRUE and *bBusy* is FALSE.

***Note**: When *bContinuousUpdate* is activated, the speed, acceleration and deceleration are modified immediately.

- **Outputs**

Name	Function	Data Type	Output Range (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 aborted	BOOL	TRUE/FALSE (FALSE)

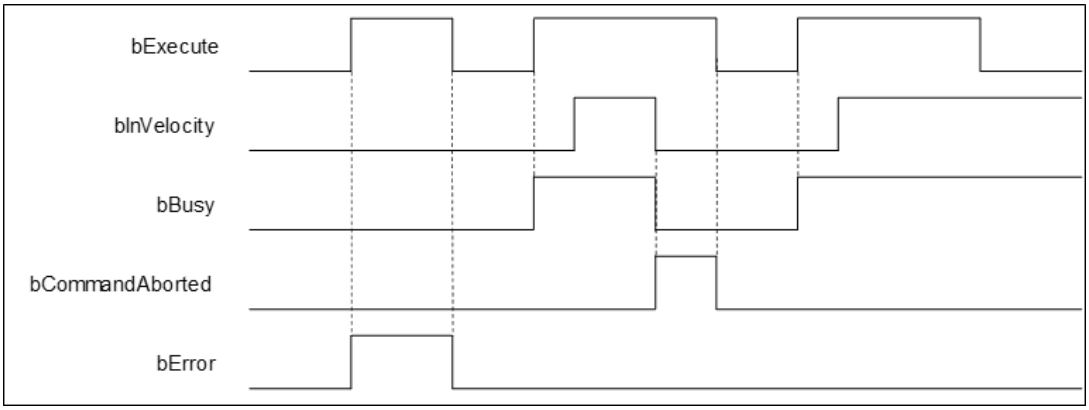
Name	Function	Data Type	Output Range (Default Value)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DMC_ERROR*	DMC_ERROR (DMC_NO_ERROR)

***Note**: DML_ERROR: Enumeration (ENUM)

- ◦ **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bInVelocity	The axis speed reaches the target speed.	• *bCommandAborted* turns to TRUE. • *bContinuousUpdate* is TRUE and write a new value to *IrVelocity*. • *bError* turns to TRUE.
bBusy	*bEnable* turns to TRUE.	• When *bError* turns to TRUE. • When *bCommandAborted* turns to TRUE.
bCommandAborted	• The function block is interrupted by another function block. • The function block is interrupted by MC_Stop.	• *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 ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID*).	*bExecute* turns to FALSE. (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*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.
  - When the *bContinuousUpdate* input parameter of the function block is TRUE and a new target velocity is assigned, the axis speed is adjusted to the new speed.
  - Enter positive and negative values for *lrVelocity* to set the rotation direction of the motor.
- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the programming example of the, refer to the DMC_VelocityControl function block.
  - In this function block, input the positioning axis to the Axis parameter.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.2 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*.

The instructions in this section can only be used for the MH300 frequency converter with the CMM-EC02 communication card.

- Environment preparation

  Library: DL_MotionControlLight V1.2.0.0 or later

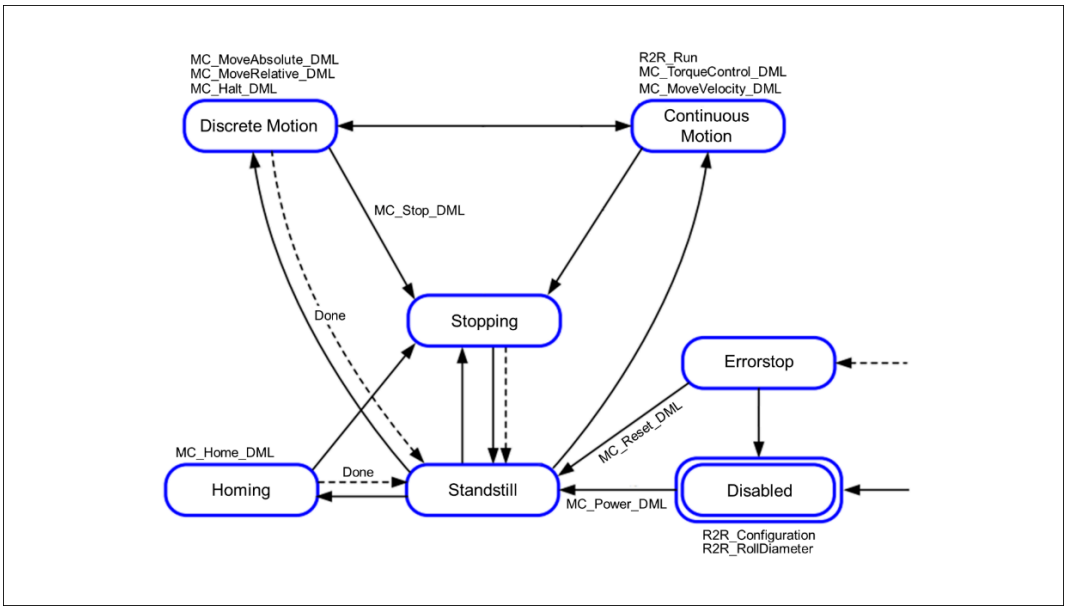
  MH300 firmware version: V2.00 or later

  EtherCAT communication card CMM-EC02 firmware version: V37124 or later

- Introduction to tension control status

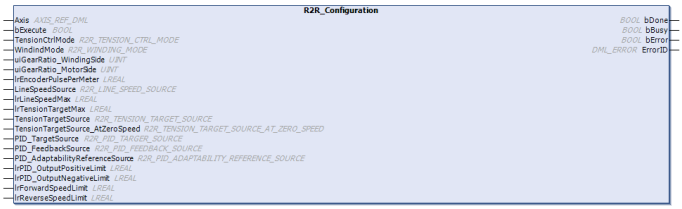
  When running R2R_Configuration and R2R_RollDiameter function blocks, the axis status should be Disabled. When running R2R_Run function block, the axis status will change from Disabled to Standstill to Continuous Motion. Otherwise R2R_Run will switch to Disabled when the function block is turned off.

  - Positioning axis status:



2.3.2.1 R2R_Configuration

R2R_Configuration configures the tension control.

FB/FC	Instruction	Graphic Expression	ST Language
FB	R2R_Configuration		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	Runs the instruction 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)	*bExecute* turns to TRUE and *bBusy* is FALSE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
			2: TensionCloseLoop_TorqueMode 3: TensionOpenLoop_TorqueMode (TensionCloseLoop_SpeedMode)	
WindindMode	Winding mode	R2R_WINDING_MODE*2	0: Rewind 1: Unwind (Rewind)	*bExecute* turns to TRUE and *bBusy* is FALSE.
uiGearRatio_WindingSide	Winding side mechanical gear A	UINT	1–65535 (100)	*bExecute* turns to TRUE and *bBusy* is FALSE.
uiGearRatio_MotorSide	Motor side mechanical gear B	UINT	1–65535 (100)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrEncoderPulsePerMeter	The number of pulses per meter (Pulse/m)	LREAL	0–6000 (0)	*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)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrLineSpeedMax	Maximum line speed (m/mm)	LREAL	0.0–3000.0 (1000.0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrTensionTargetMax	Maximum tension value (N)	LREAL	0–65535 (0)	*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)	*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)	*bExecute* turns to TRUE and *bBusy* is FALSE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
PID_TargetSource	PID target source	R2R_PID_TARGER_SOURCE*6	0: R2R_Run_lrPID_TargetValue 1: AVI 2: ACI (R2R_Run_lrPID_TargetValue)	*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)	*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)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrPID_OutputPositiveLimit	Tension PID positive output limit (%)	LREAL	0–655.35 (20.0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrPID_OutputNegativeLimit	Tension PID negative output limit (%)	LREAL	0–655.35 (1.0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrForwardSpeedLimit	Torque mode forward speed limit (%)	LREAL	0–120 (10)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrReverseSpeedLimit	Torque mode reverse speed limit (%)	LREAL	0–120 (10)	*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	Output Range (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 within the instruction	BOOL	TRUE/FALSE (FALSE)

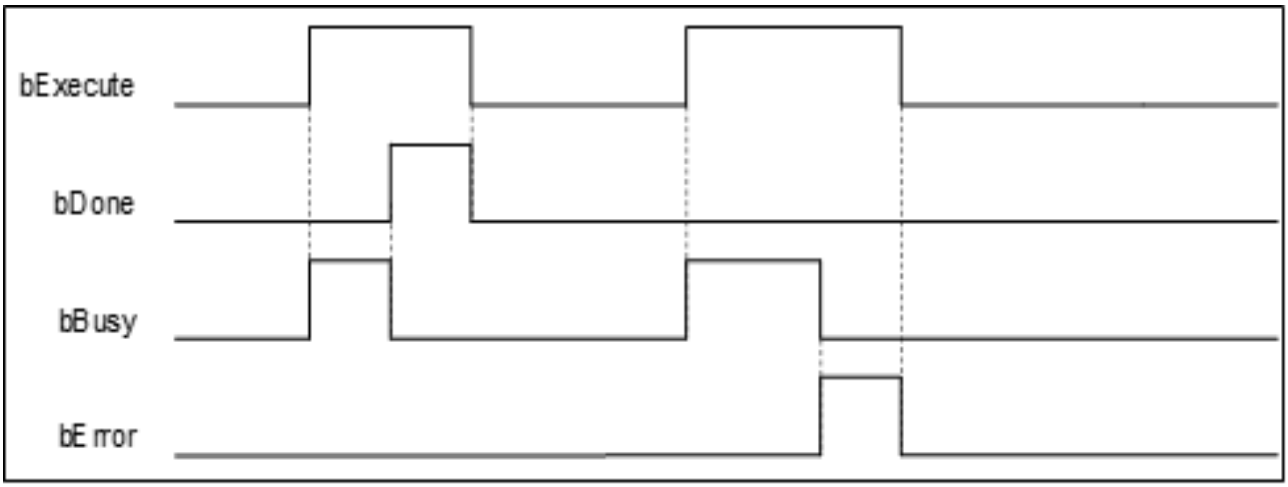
Name	Function	Data Type	Output Range (Default Value)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*	DML _ERROR (DML_NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The motion restarts.	• *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	*bExecute* turns to TRUE.	• *bDone* turns to TRUE • *bError* turns to TRUE
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID*)	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified 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 supported by DL_ MotionControlLight V1.2.0.0 or later.
  - This instruction is used to set the parameters for tension control. Use this function block to configure settings before running tension control.
  - There are 4 modes of TensionCtrlMode. The following describes each mode architecture.
    - TensionCloseLoop_SpeedMode (tension closed-loop control, speed mode)

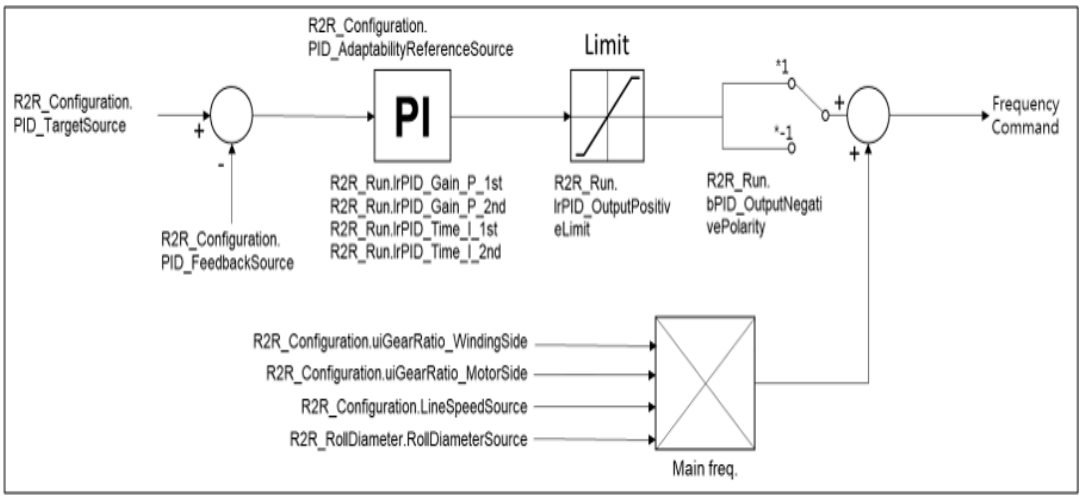
      In this mode, the driver is set to speed mode for tension control.

      Tension control main frequency: $f(Hz) = \frac{V}{\pi D} \cdot \frac{A}{B}$

      V: Line speed (m/min.)

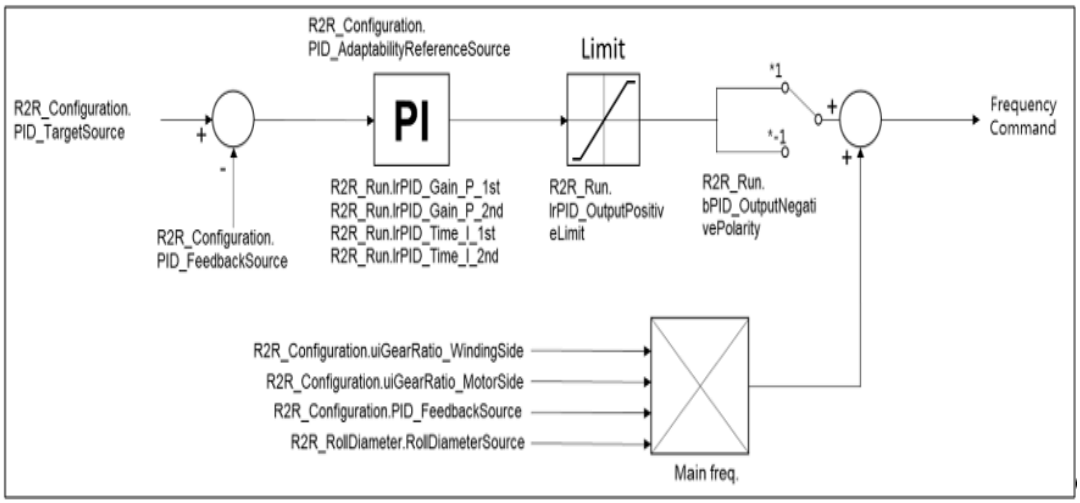
      D: Reel diameter (m)

      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.

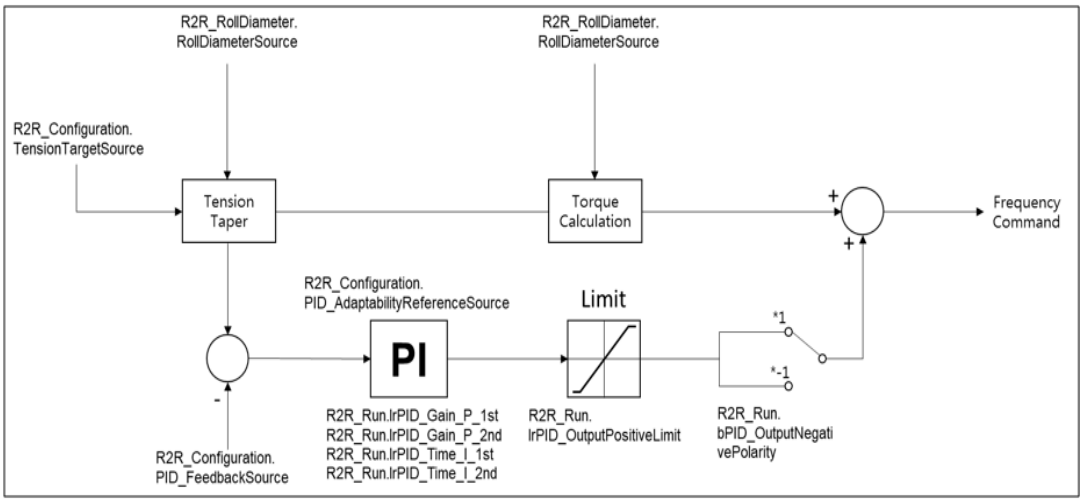
    - TensionCloseLoop_TorqueMode (Tension closed-loop control, torque mode)

      In this mode, the driver is set to torque mode for tension control.

      F: Tension (N)

      D: Reel diameter (m)

      $\text{Torque}(N-m) = \frac{F \cdot D}{2}$



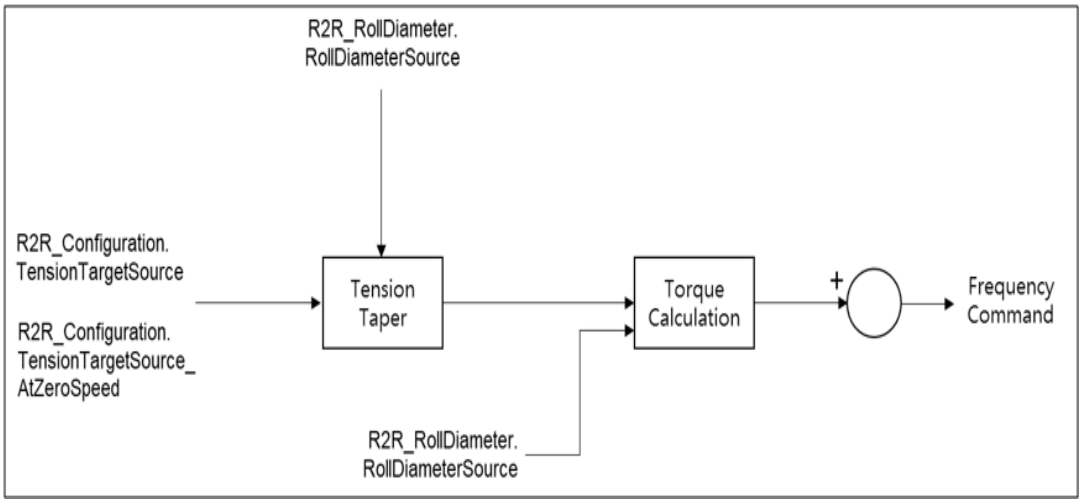
- TensionOpenLoop_TorqueMode (Tension open-loop control, torque mode)

  In this mode, the driver is set to torque mode for tension control.

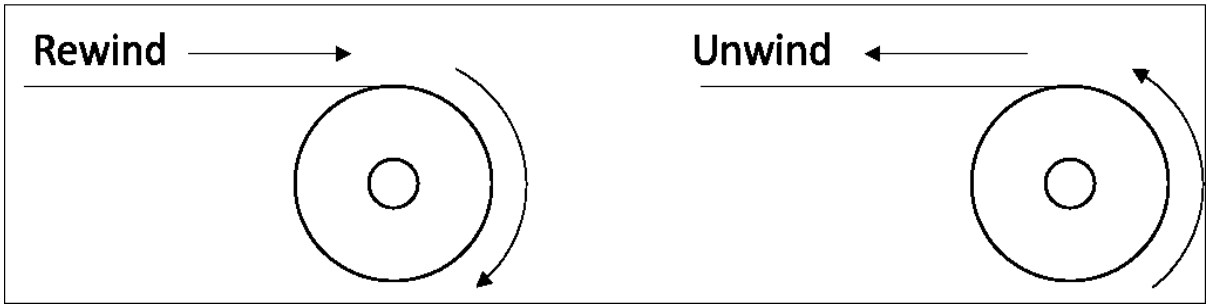
  F: Tension (N)

  D: Reel diameter (m)

  $$\text{Torque}(\text{N-m}) = \frac{F \cdot D}{2}$$

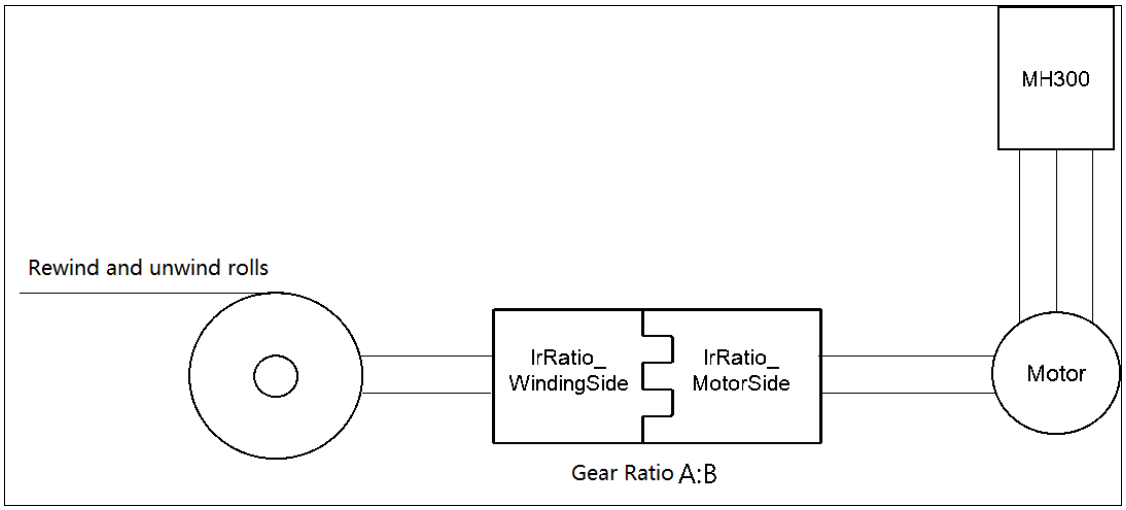


- WindingMode has Rewind and Uwind modes.



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.

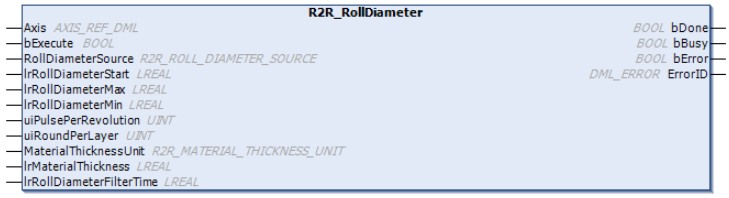


- Select **PG_CARD** in the *LineSpeedSource* input parameter. You must install the EMM-PG01x accessory card on the MH300 driver. Refer to *MH300 Series User Manual* Chapter 8 for details.
- MH300 driver supports the following signal inputs. Refer to MH300 Series User Manual Chapter 6 for the hardware configuration of signals.
  - DFM_DMC: Pulse voltage as the 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 must be in the Disabled status to run this function block.
- This instruction is only supported by Delta MH300 with EtherCAT (CMM-EC02) communication card.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**
  - For the example, refer to the R2R_Run function block.
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.2.2 R2R_RollDiameter

R2R_RollDiameter sets the roll diameter.

FB/FC	Instruction	Graphic Expression	ST Language
FB	R2R_RollDiameter		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
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	*bExecute* turns to TRUE and *bBusy* is FALSE.

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
			8: ThicknessIntegrate_Winding_ Encorder_MI67 9: ThicknessIntegrate_Winding_ CloseSW_MI7 (LineSpeed)	
lrRollDiameterStart	Current roll diameter (mm)	LREAL	1.0–6000.0 (6000.0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrRollDiameterMax	Maximum roll diameter (mm)	LREAL	1.0–6000.0 (6000.0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrRollDiameterMin	Empty roll diameter (mm)	LREAL	1.0–6000.0 (1.0)	*bExecute* turns to TRUE and *bBusy* is FALSE.
uiPulsePerRevolution	Number of pulses per revolution (Pulse/lr)	UINT	1–60000 (1)	*bExecute* turns to TRUE and *bBusy* is FALSE.
uiRoundPerLayer	Number of rounds per layer (round/layer)	UINT	1–10000 (1)	*bExecute* turns to TRUE and *bBusy* is FALSE.
MaterialThicknessUnit	Material thickness multiple	R2R_MATERIAL_THICKNESS_GAIN[*2]	0: millimeter 1: centimeter (millimeter)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrMaterialThickness	Material thickness (mm)	LREAL	0.001–65.0 (0.001)	*bExecute* turns to TRUE and *bBusy* is FALSE.
lrRollDiameterFilterTime	Roll diameter filter time (s)	LREAL	0–100.0 (1.0)	*bExecute* turns 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	Output Range (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 within the instruction	BOOL	TRUE/FALSE (FALSE)

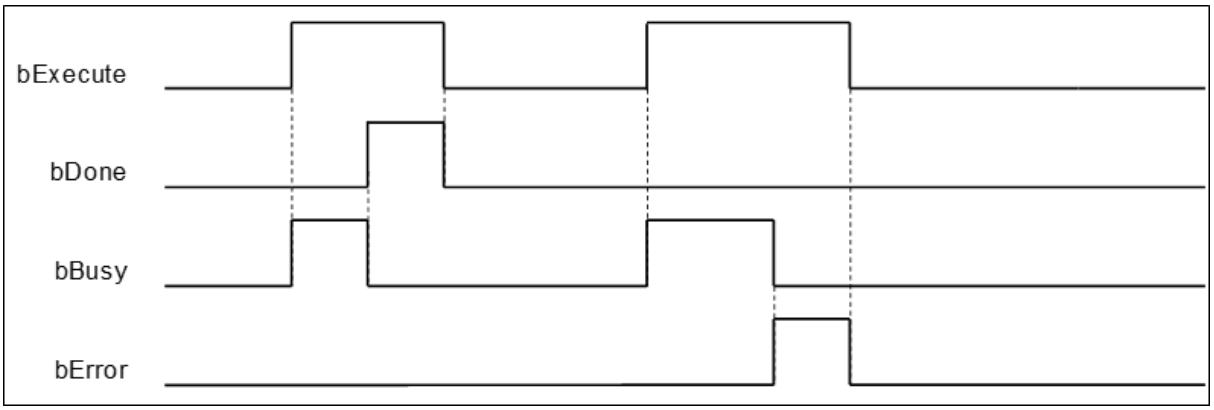
Name	Function	Data Type	Output Range (Default Value)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*	DML _ERROR (DML _NO_ERROR)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	When the motion 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 ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID*)	When *bExecute* turns to FALSE (Error Code is cleared.)

- **Timing Diagram of Output Parameter Changes**



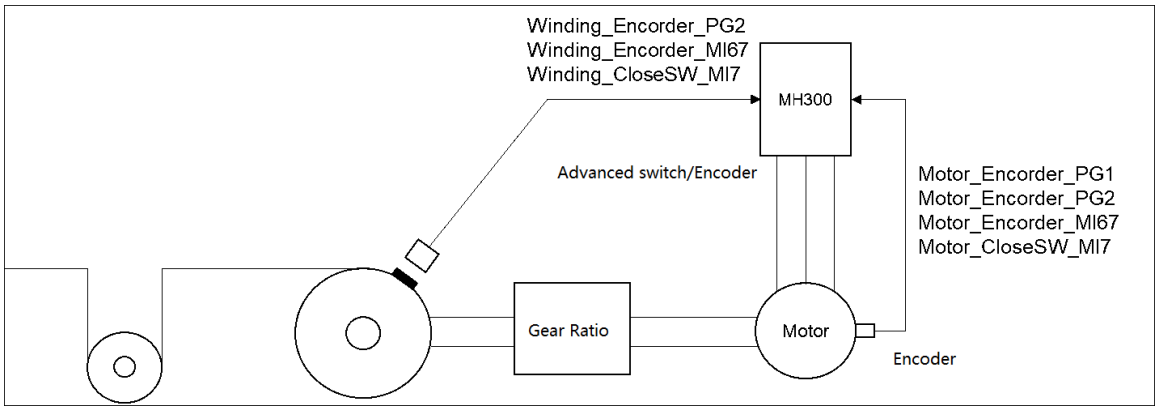
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified 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.

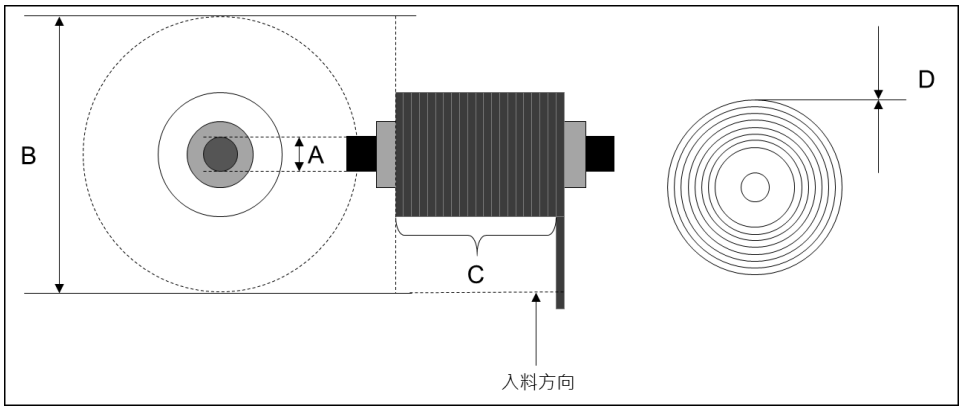
- **Function**
  - This function is only available with DL_ MotionControlLight V1.2.0.0 or later.
  - This instruction is an industry function block, which can set the relevant parameters of the roll diameter.

- 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:

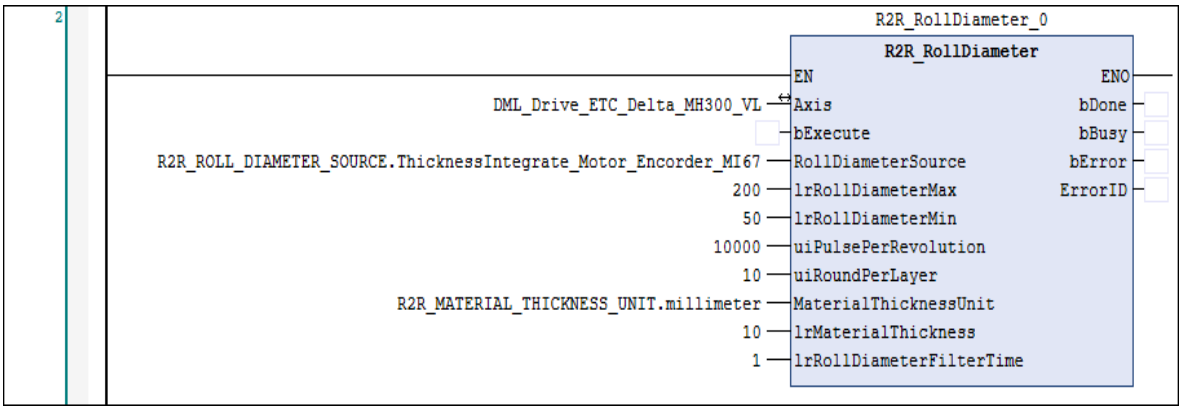


- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

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

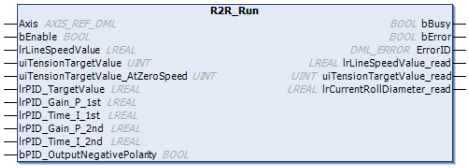


- Set the corresponding parameters, and then run this function. When *bDone* of the R2R_RollDiameter_0 is TRUE, parameters have been written to the driver.

- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.3.2.3 R2R_Run

R2R_Run activates the tension control function.

FB/FC	Instruction	Graphic Expression	ST Language
FB	R2R_Run		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	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
lrLineSpeedValue	Current line speed (m/min)	LREAL	0.0–3000.0(0)	*bEnable* turns to TRUE.
uiTensionTargetValue	Tension command value (N)	UINT	0–65535 (0)	*bEnable* turns to TRUE.
uiTensionTargetValue _AtZeroSpeed	Zero speed tension value (N)	UINT	0–65535 (0)	*bEnable* turns to TRUE.
lrPID_TargetValue	PID target value (%)	LREAL	0–100 (50.0)	*bEnable* turns to TRUE.
lrPID_Gain_P_1st	Tension PID P gain 1(%)	LREAL	0.0–1000.0 (50.0)	*bEnable* turns to TRUE.
lrPID_Time_I_1st	Tension PID I integration time 1	LREAL	0.0–500.0 (1.0)	*bEnable* turns to TRUE.
lrPID_Gain_P_2nd	Tension PID P gain 2(%)	LREAL	0.0–1000.0 (50.0)	*bEnable* turns to TRUE.
lrPID_Time_I_2nd	Tension PID I integration time 2	LREAL	0.0–500.0 (1.0)	.*bEnable* turns to TRUE.
bPID_OutputNegativePolarity*	Tension PID output status selection	BOOL	TRUE/FALSE (FALSE)	*bEnable* turns to TRUE.

***Note**: When *bPID_OutputNegativePolarity* is FALSE, the PID output is positive.

- **Outputs**

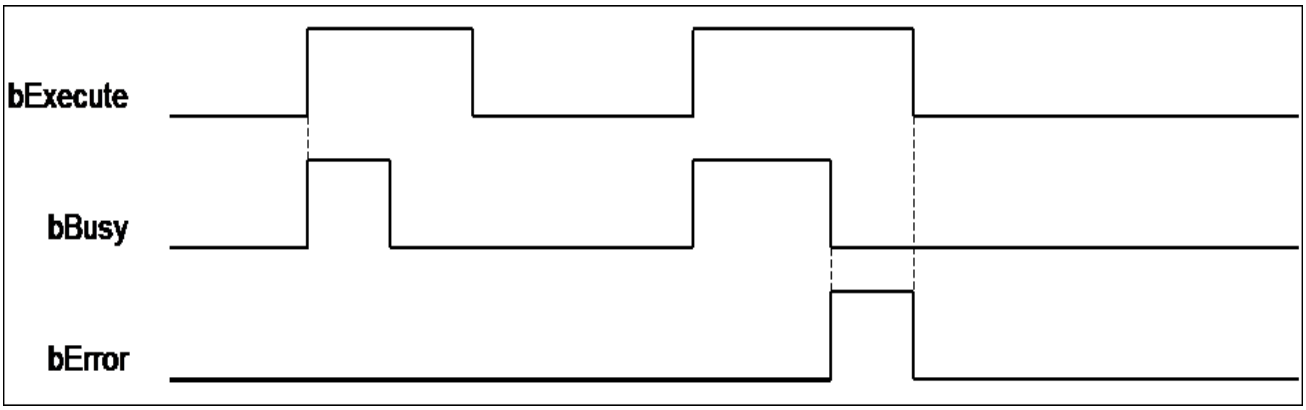
Name	Function	Data Type	Output Range (Default Value)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	DML_ERROR*	DML _ERROR (DML _NO_ERROR)
lrLineSpeedValue_read	Read the current line speed.	LREAL	Positive number or 0 (0)
uiTensionTargetValue_read	Read the current roll diameter.	UINT	Positive number or 0 (0)
lrCurrentRollDiameter_read	Read the tension command value.	LREAL	Positive number or 0 (0)

***Note**: DMC_ERROR: Enumeration (Enum)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bBusy	*bExecute* turns to TRUE.	*bError* turns to TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect. (Error code is recorded in *ErrorID*).	*bExecute* turns to FALSE. (Error Code is cleared.)
ErrorID		
lrLineSpeedValue_read	Continuously update when *bBusy* is TRUE	Stop updating when *bBusy* is FALSE.
uiTensionTargetValue_read	Continuously update when *bBusy* is TRUE	Stop updating when *bBusy* is FALSE.
lrCurrentRollDiameter_read	Continuously update when *bBusy* is TRUE	Stop updating when *bBusy* is FALSE.

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

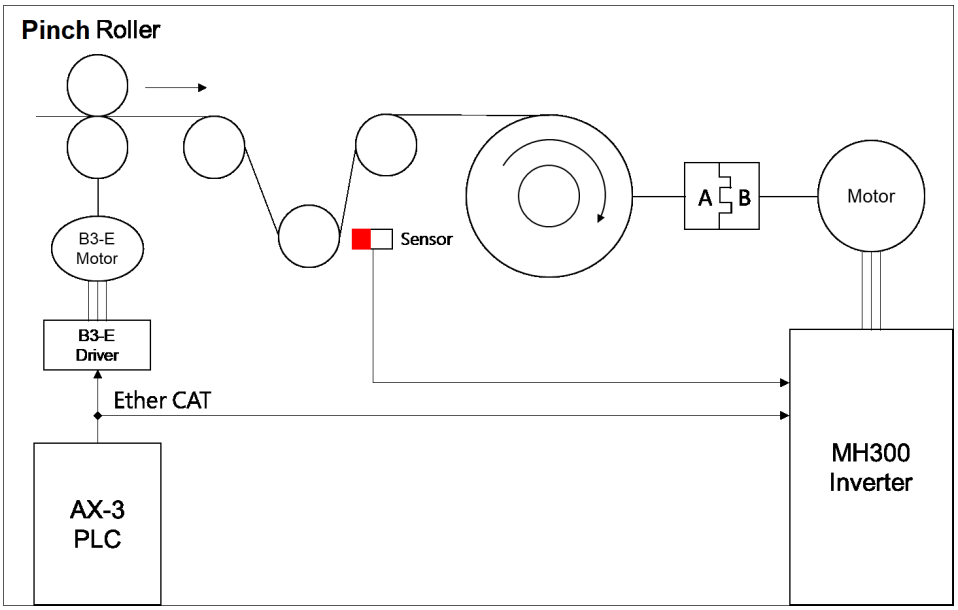
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_DML*	AXIS_REF_DML	*bExecute* turns to TRUE and *bBusy* is FALSE.

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

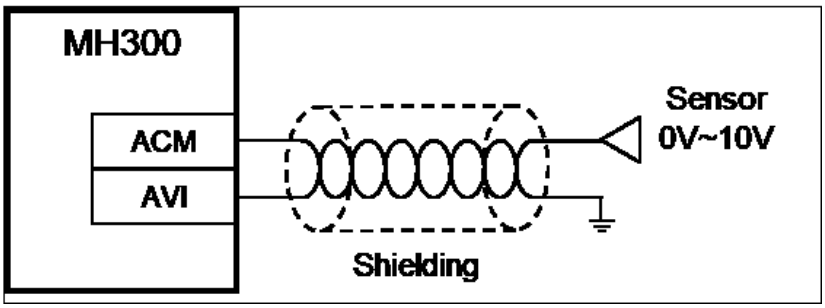
- **Function**
  - This function is supported by DL_ MotionControlLight V1.2.0.0 or later.
  - This instruction is an industry function block that activates tension control.
  - If you perform this function, you need to change the axis status to Standstill, and if the function is successfully run, the 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 *bPID_OutputNegativePolarity* can be given back by tension according to the different needs of customers. Refer to the following table 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**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **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_Speed Mode	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_lrLineSpeedVa lue	Set the line speed source to the function block R2R_Run.lrLineSpeedVal ue.
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

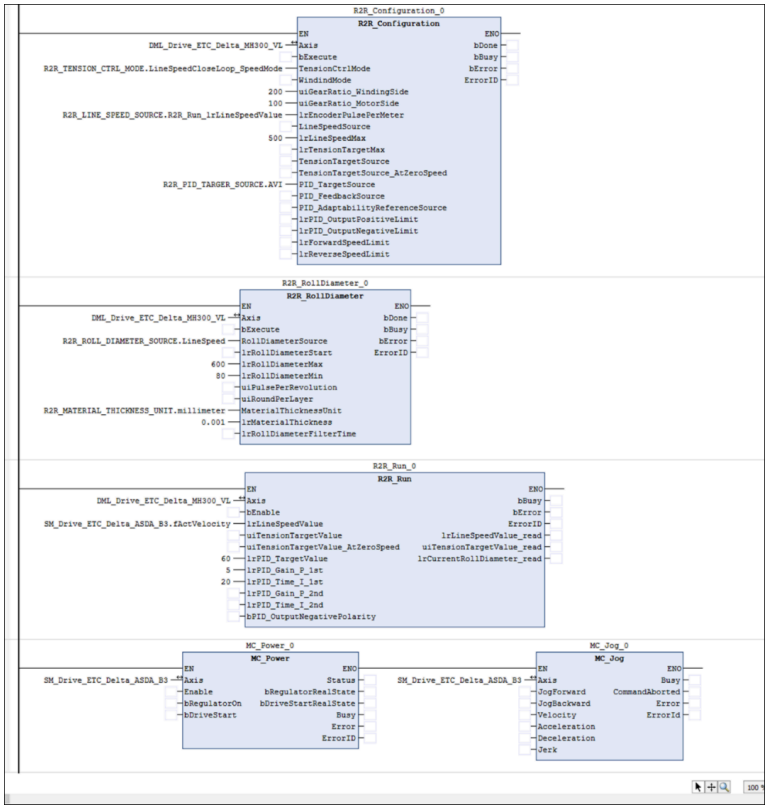
- R2R_Run Parameters

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

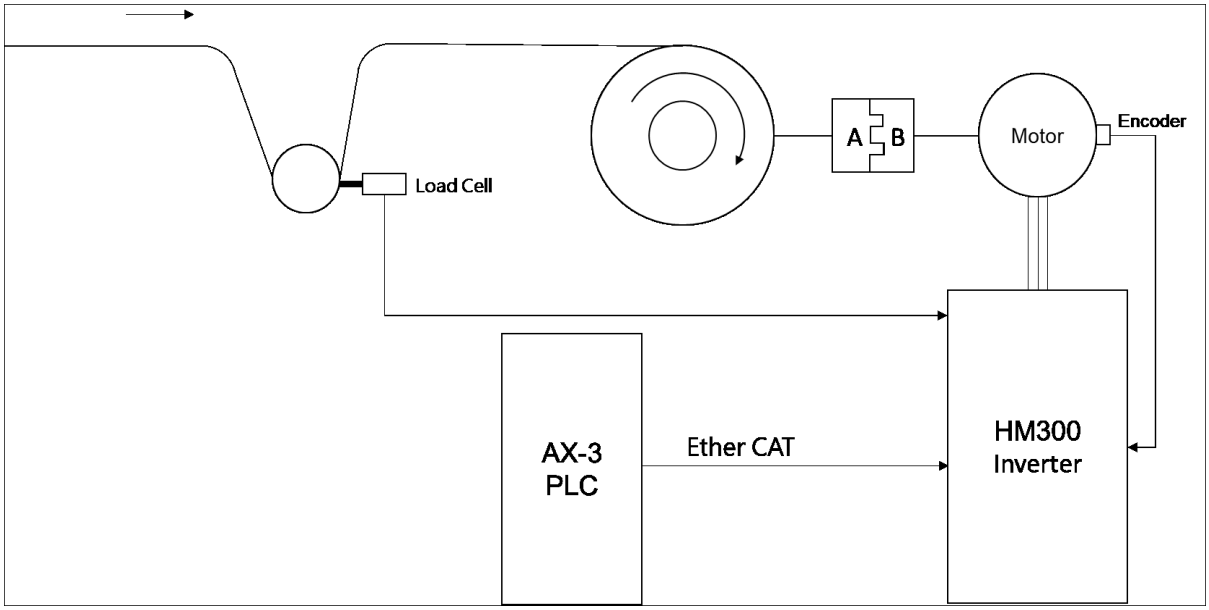
Name	Value	Description
03-28	0	Select AVI terminal input
03-29	1	Select ACI terminal input

- Tension closed-loop speed mode program

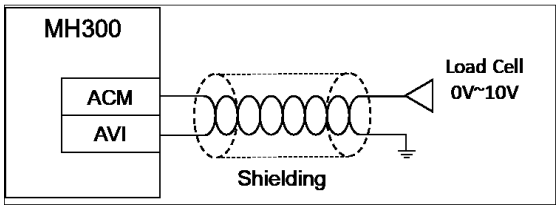


  - Set the R2R_Configuration, R2R_RollDiameter, and R2R_Run parameters according to the above table.
  - Starting sequence: start the R2R_Configuration, R2R_RollDiameter, and then start the R2R_Run.
  - After the tension function is activated, start the feeding axis, and the line speed source of MH300 runs according to the line speed fed back by the feeding axis.

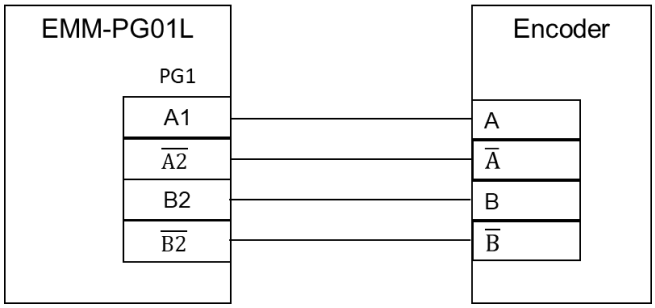
- **Example 2**: This example shows how to use the TensionCloseLoop_TorqueMode (Tension closed-loop, torque mode) mode. Refer to the following device architecture.



- Load Cell Wiring Diagram.



- EMM-PG01L Wiring Diagram



- R2R_Configuration Parameters

Name	Value	Description
TensionCtrlMode	TensionCloseLoop_Torque Mode	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)

Name	Value	Description
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

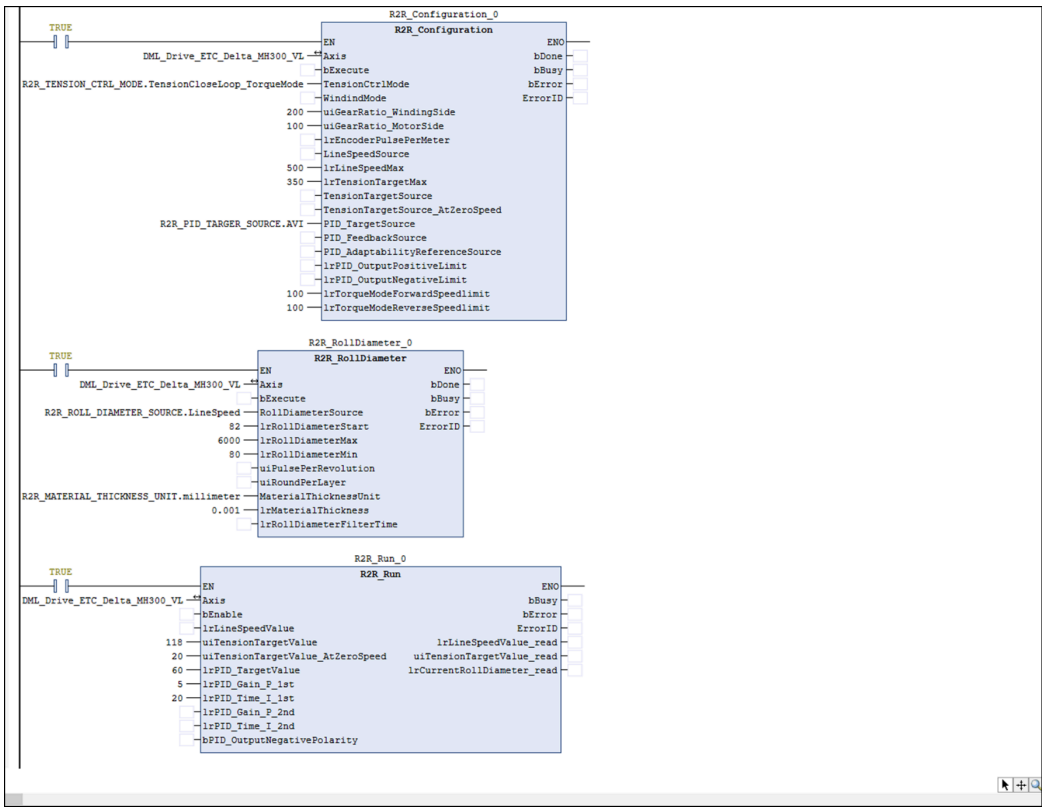
- Encoder Settings

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

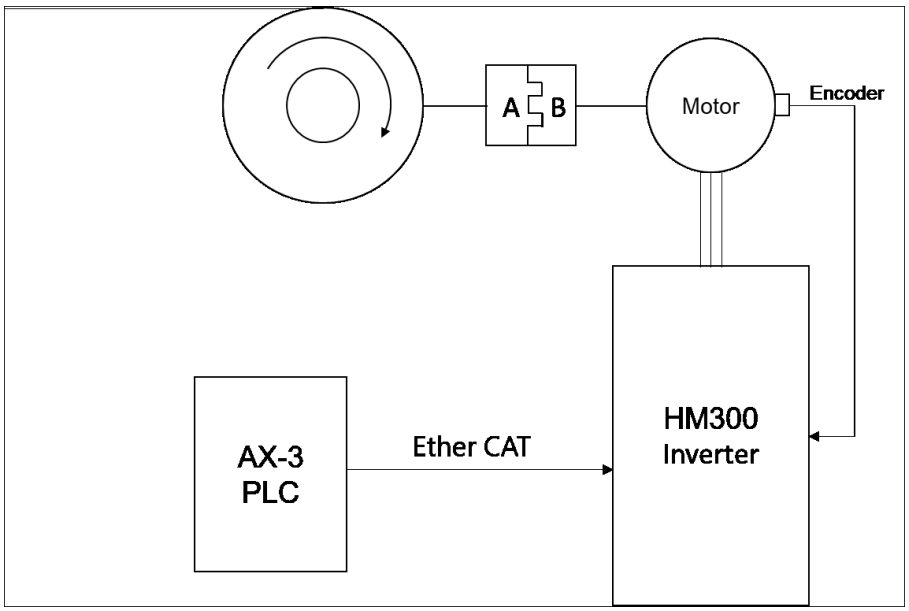
Name	Value	Description
03-28	0	Select AVI terminal input.
10-16	0	Pulse input type setting
10-17	1	Electronic gears A
10-18	1	Electronic gears B

- Tension closed-loop torque mode program

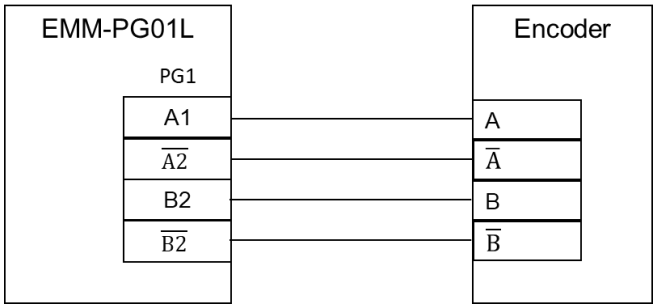


- Set the R2R_Configuration, R2R_RollDiameter, and R2R_Run parameters according to the above table.
- Starting sequence: start the R2R_Configuration, R2R_RollDiameter, and then start the R2R_Run.
- After the tension function is activated, it will run according to the tension value of the function block.

◦ **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)

Name	Value	Description
TensionTargetSource	R2R_Run_uiTensionTarget Value	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_AtZero Speed	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

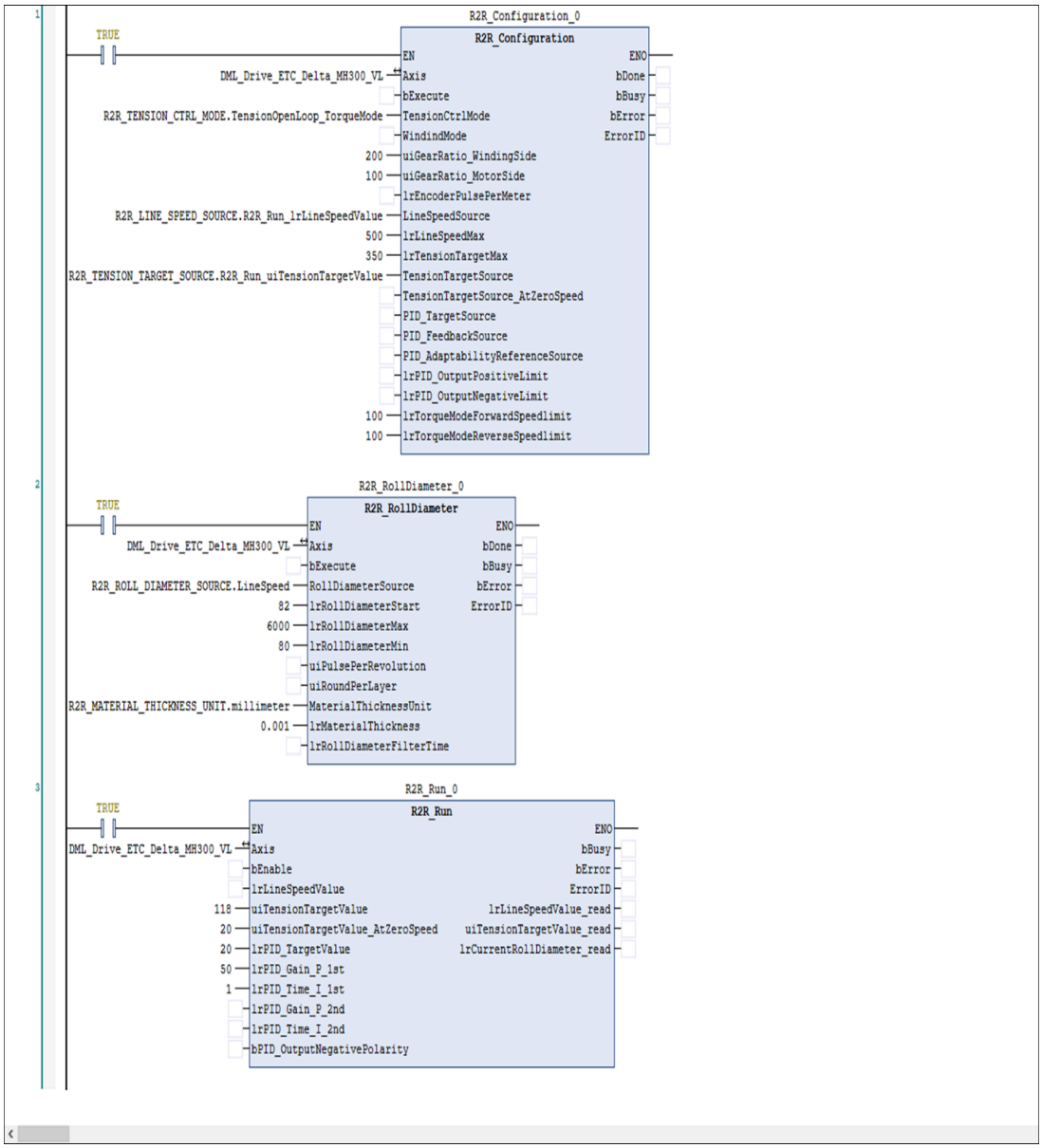
- Encoder Settings

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.
- The tension function will run according to the tension value of the block when it is activated, and the line speed can be set to match the speed of the unloading axis through R2R_Run.lrLinSpeedValue

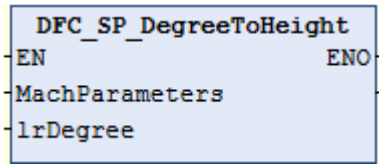
- **Supported Devices**
  - AX-series motion controller, AX-series basic motion controller

2.4 DL_ServoPress_AX

2.4.1 Servo Press Instruction

2.4.1.1 DFC_SP_DegreeToHeight

DFC_SP_DegreeToHeight is used in the servo press industry. It calculates the slide position height by inputting crank radius, connecting rod length, and crankshaft angle. During motion control, the current slide height can be calculated from the current crank angle, crank radius, 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 Value)	Timing to Take Effect
MachParameters	Servo press parameters	Reference DMC_SP_MACHINE_PARAMETERS[*2]	-	-
lrDegree	The degree of the slave axis	LREAL[*1]	Positive (0)	*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)	LREAL	Positive
lrLLength	Connecting Rod (unit: mm)	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's Stroke per minutes (SPM).	LREAL	Positive
lrGearRatio	Reducer gear ratio	LREAL	Positive
lrRPMmotor	Max RPM of the motor	LREAL	Positive

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
lrHeight	Slider height	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 of Output Parameter Changes**

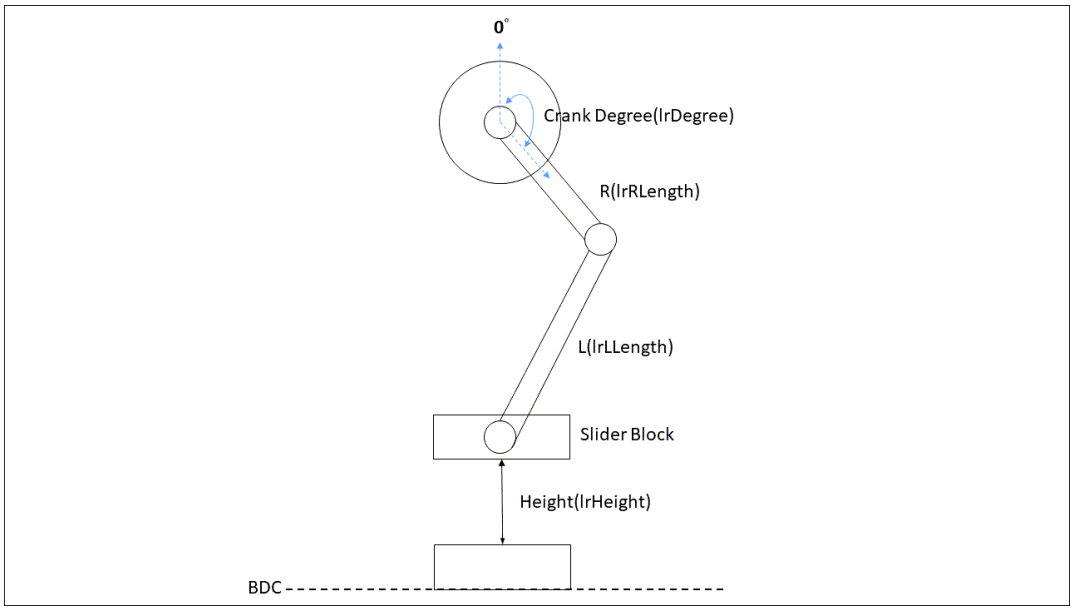
-

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

- **Function**

  - Input *MachParameters* and *lrDegree*. The function block will calculate the slider height (*lrHeight*).
  - If invalid values are set, *lrHeight* will output 0.
  - Schematic diagram:

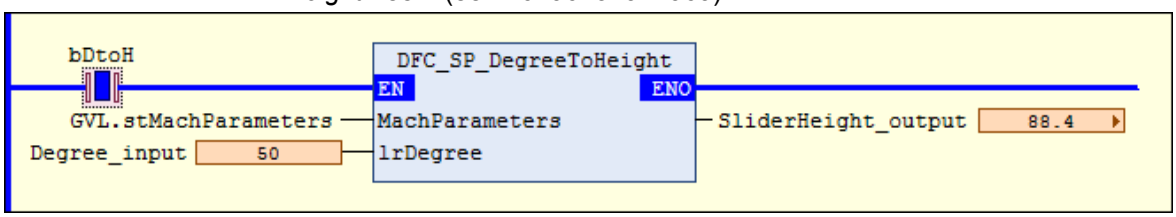


- **Troubleshooting**

  -

- **Example**

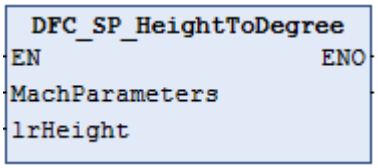
  Set *MachParameters.lrRLength* = 50mm, *MachParameters.lrLLength* = 120mm, and *lrDegree* = 50. Then the function will return *lrHeight* =88.4 (88.416296732977969).



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

2.4.1.2 DFC_SP_HeightToDegreeDFC_SP_HeightToDegree

DFC_SP_HeightToDegree is used in the servo press industry. It calculates the crank (0~180°) by inputting the parameters of crankshaft radius, connecting rod length, and slide height. If the slide has passed the Bottom Dead Center (BDC), the function block automatically mirrors the angle (by returning 360.0 minus the calculated value).When an invalid value (ex: less than 0) is input, the angle cannot be calculated, and 0 will be returned.

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 Value)	Timing to Take Effect
MachParameters	Servo press parameters	ReferenceDMC_SP_MACHIME_PARAMETERS*3	-	-
lrHeight	Slider height	LREAL*1	Positive*2	-

***Note1**: LREAL Range:

Positive values: Between –1.7976931348623157E+308 and –4.9406564584124654E-324

Zero: 0

***Note2**: *lrHeight* valid range: 0– (2 * rfMachParameters.lrRLength)

***Note3**: DMC_SP_MACHINE_PARAMETERS

Name	Function	Data Type	Setting Value
lrRLength	Crankshaft length (unit: mm)	LREAL	Positive
lrLLength	Connecting rod length (unit: mm)	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's SPM.	LREAL	Positive
lrGearRatio	Reducer 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 of Output Parameter Changes**

  -

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

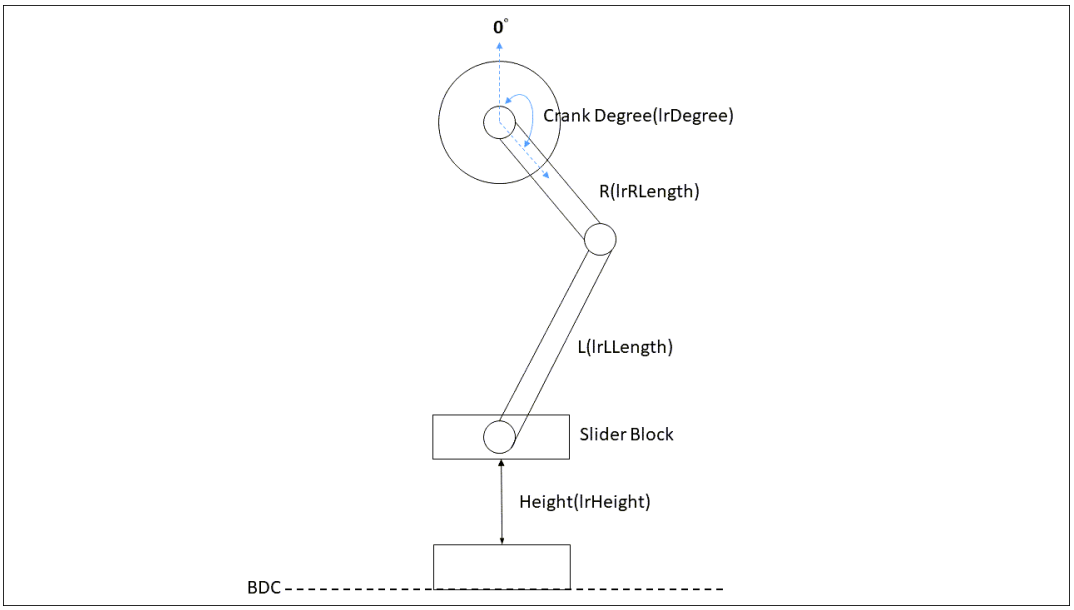
- **Function**

- Input *MachParameters*, and *lrHeight*. The function will calculate *lrDegree*.

- This function is used in servo press function blocks related to generating key points, such as DMC_SP_CamCrankCurve, DMC_SP_CamLinkCurve, and DMC_SP_CamCoinCurve.

- Input valid MachParameters.lrRLength, MachParameters.lrLLegnth, and *lrHeight*, then the function block will calculate *lrDegree*. If invalid values are set, the function will return 0.

- Schematic diagram:

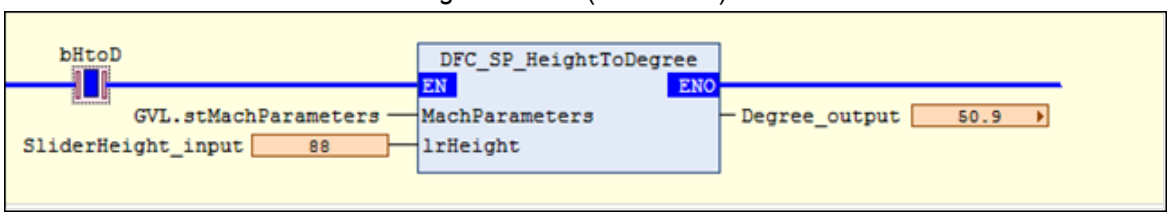


- **Troubleshooting**

  -

- **Example**

  Set up MachParameters.lrRLength = 50.0mm, MachParameters.lrLLength = 120.0mm, and *lrHeight* = 88.=mm. The function will return *lrDegree* = 50.9°(50.859…..).



- **Supported Devices**

  - AX-308E, AX-8

2.4.1.3 DMC_SP_CamCoinCurve

DMC_SP_CamCoinCurveervo is used in the servo press industry. It generates cam key point data by inputting the impression curve algorithm, and generates an electronic cam table through DMC_SP_CamCurveWrite function block. You can use the electronic cam table for servo press processing.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DMC_SP_Cam CoinCruve	DMC_SP_CamCoinCurve EN ENO bExecute bDone MachParameters bBusy lrHeightStart bError lrHeightEnd dwErrorID lrT1percent lrMasterPoint lrT2percent lrSlavePoint lrT3sec CamCurveType lrT4percent bVelEnable lrVelValue bAccEnable lrAccValue wWriteAmount	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 Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	FALSE/TRUE (FALSE)	-
MachParameters	Servo press 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] 0–100	*bBusy*=FALSE & *bExecute* is at rising-edge

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrT2percent	T2 percentage of time	LREAL*1	Positive (0) [Range] 0–100	*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	Crankshaft length (unit: mm)	LREAL	Positive
lrLLength	Connecting rod length (unit: mm)	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's SPM. (six times *lrSPMsys* is equal to the unit deg/sec)	LREAL	Positive
lrGearRatio	Reducer 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 finishes generating coin key-points	BOOL	FALSE/TRUE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	FALSE/TRUE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	FALSE/TRUE (FALSE)
dwErrorID	Error Code of the error	DWORD	16#00000000–16#FFFFFFFF (16#00000000)
lrMasterPoint	Master position of the key point	[0..63] LREAL Array	0.0–360.0 (0.0)
lrSlavePoint	Slave position of the key point	[0..63] LREAL Array	0.0–360.0 (0.0)
CamCurveType	Curve type between key points	[0..62] DMC_CamCurveType Array	0–9*2 (0)
bVelEnable	Enables the key point velocity configurations	[0..63] BOOL Array	TRUE/FALSE (FALSE)

Name	Function	Data Type	Output Range (Default Value)
lrVelValue	Velocity value	[0..63] LREAL Array	LREAL[*1] (0)
bAccEnable	Enables the key point acceleration configurations	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrAccValue	Acceleration value	[0..63] LREAL Array	LREAL[*1] (0)
wWriteAmount	Key point position value	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)

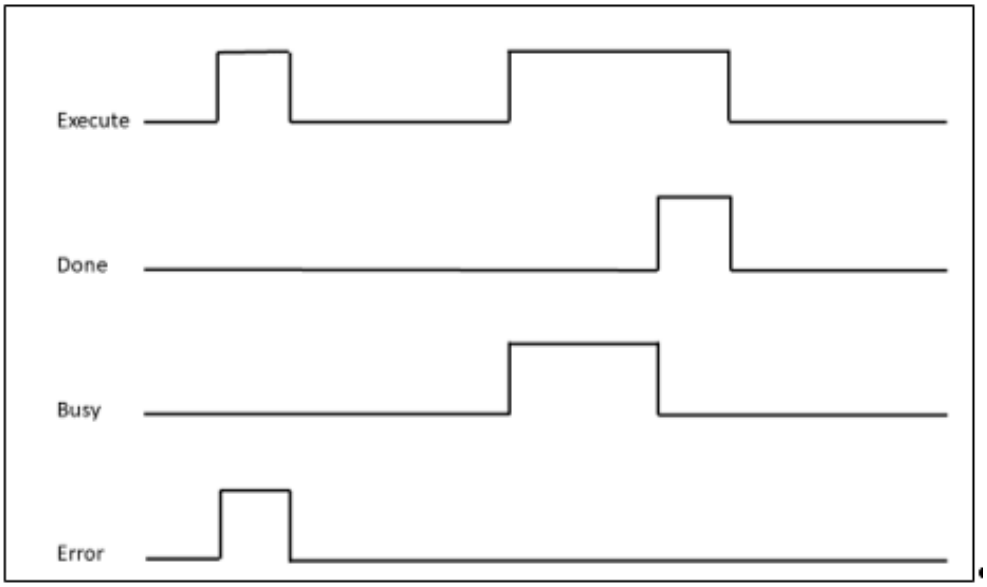
Cubic_Spline_Clamp (8)

Cubic_Spline(9)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The function block finishes curve key points generation.	• *bExecute* is FALSE. • *bExecute* is FALSE but *bDone* turns to TRUE, *bDone* will remain TRUE for one scan cycle and then change to FALSE
bBusy	*bExecute* turns to TRUE	• *bDone* is TRUE. • *bError* is TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* is FALSE.

- **Timing Diagram of Output Parameter Changes**

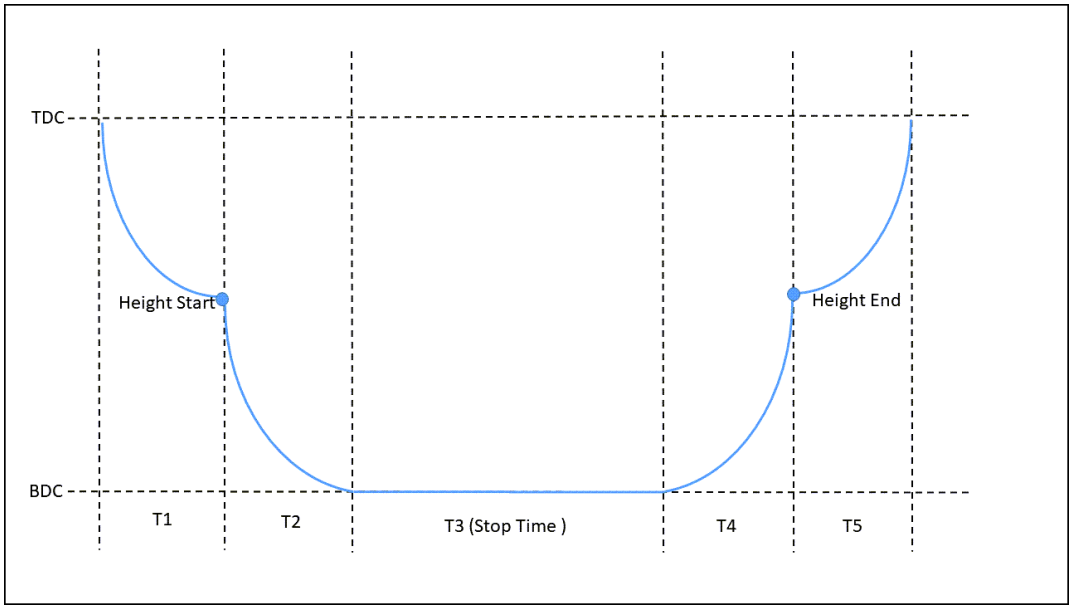


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

- **Function**

  - To generate the servo press cam key point information, the data must be input into the DMC_SP_CamCurveWrite function block.
  - After the function block finishes executing, use DMC_SP_CamCurveWrite to generate and write ECAM table.
  - Press curve setting diagram:

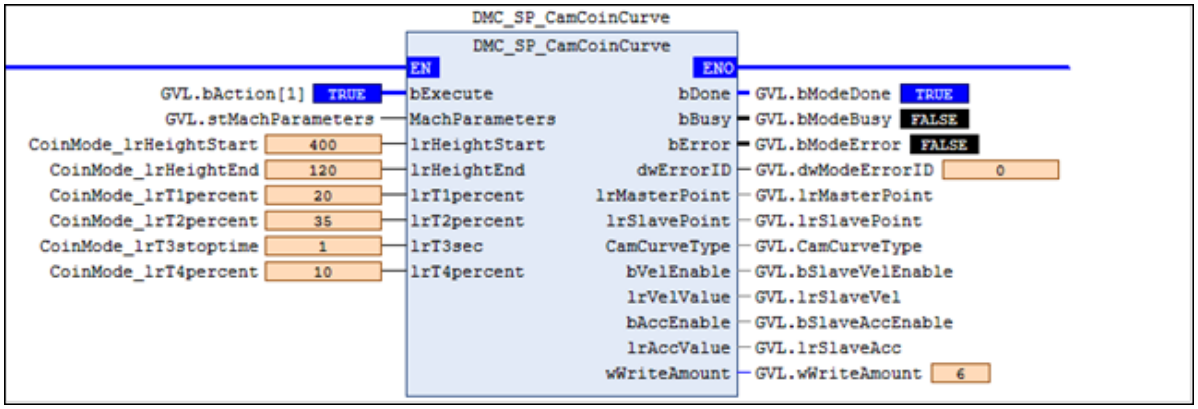


- **Troubleshooting**

  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *dwErrorID* (Error Code) to check the problem.

Error Code	Description	Contents	Corrective Action
0	No Error	-	-
1	Invalid parameters	Invalid electromechanical parameters were set	Check whether these values are within the range: • MachParameters.lrRLength • MachParameters.lrLLength • MachParameters.lrSPMsys
2	Invalid slider height	Incorrect range of slider height was set	Check whether 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 whether the values are within the range: • lrT1percent • lrT2percent • lrT3sec • lrT4percent

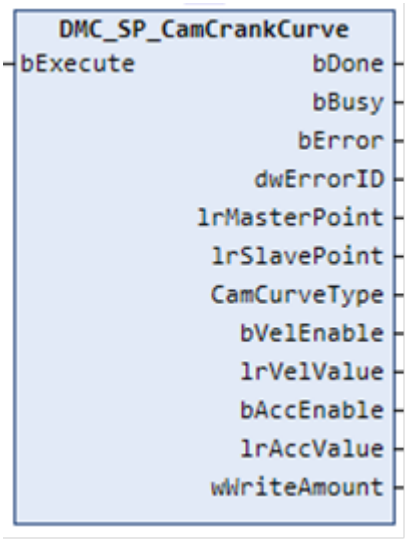
- **Example**
  - Set MachParameters.lrLlength=900, MachParameters lrRlength=300, *lrHightStart*=400, *lrHightEnd*=120, *lrT1_percent*=20, *rT2_percent*=35, *lrT3_StopTime*=1, *lrT4_percent*=10.
  - Set *bExecute* from FALSE to TRUE to trigger curve key-points calculation. Once calculation completes, *bDone* will be TRUE, and generate related key-points array.



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

2.4.1.4 DMC_SP_CamCrankCurve

DMC_SP_CamCrankCurve is used in the servo press industry. It generates cam key point information by inputting the crankshaft curve algorithm and generates an electronic cam table through DMC_SP_CamCurveWrite FB. You 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 Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	FALSE/TRUE (FALSE)	-

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when FB finishes generating key-points position	BOOL	FALSE/TRUE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	FALSE/TRUE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	FALSE/TRUE (FALSE)
dwErrorID	Error Code of the error	DWORD	DWORD (0)
lrMasterPoint	Master positions	[0..63] LREAL Array	0.0–360.0 (0.0)
lrSlavePoint	Slave positions	[0..63] LREAL Array	0.0–360.0 (0.0)
CamCurveType	Curve type between key points	[0..62] DMC_CamCurveType Array	0–9[*2](0)
bVelEnable	Enables the key point velocity configurations	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrVelValue	Velocity value	[0..63] LREAL Array	LREAL[*1](0)

Name	Function	Data Type	Output Range (Default Value)
bAccEnable	Enables the key point acceleration configurations	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrAccValue	Acceleration value	[0..63] LREAL Array	LREAL[*1](0)
wWriteAmount	Key point position value	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)

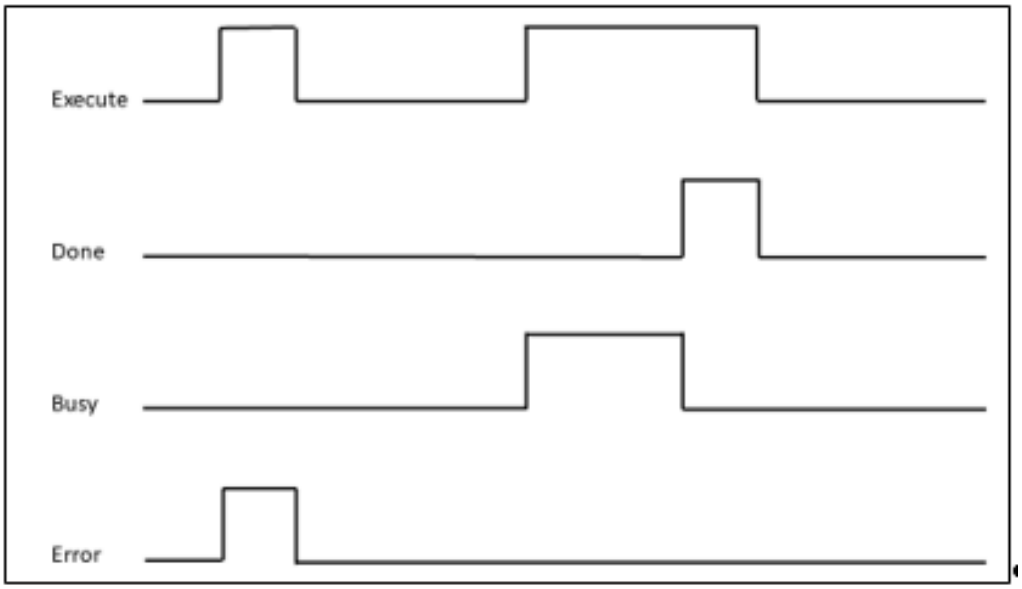
Cubic_Spline_Clamp (8)

Cubic_Spline (9)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The function block finishes curve key points generation	• *bExecute* is FALSE. • *bExecute* is FALSE but *bDone* turns to TRUE, *bDone* will remain TRUE for one scan cycle and then change to FALSE.
bBusy	*bExecute* is TRUE.	• *bDone* is TRUE. • *bError* is TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* is FALSE.

- **Timing Diagram of Output Parameter Changes**

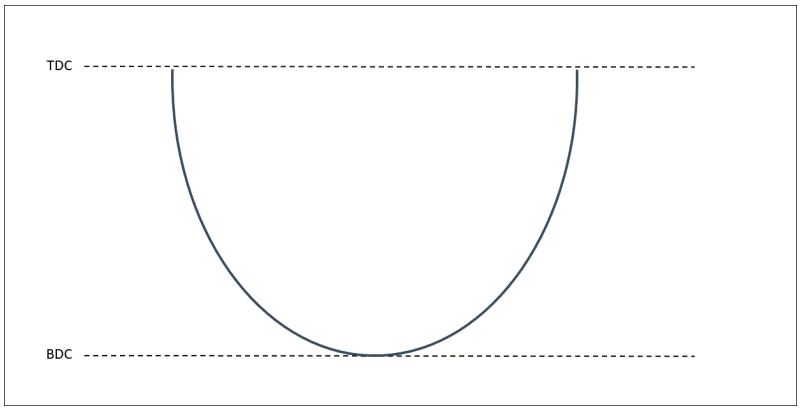


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

- **Function**

  - To generate the key point data for DMC_CamKeyPointWrite FB, the slider move from TDC to BDC.



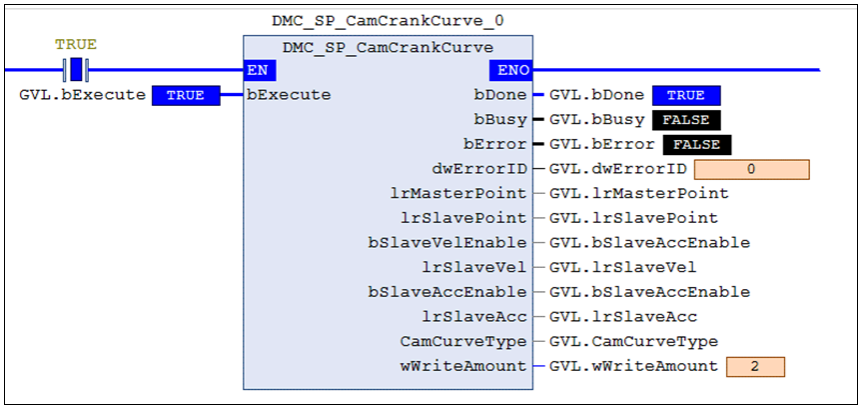
- **Troubleshooting**

Error Code	Description	Contents	Corrective Action
0	No Error	-	-

- **Example**

  1. Input *refMachParameters.lrRLength* and *refMachParameters.lrLLength*.
  2. Set *bExecute* to TRUE.

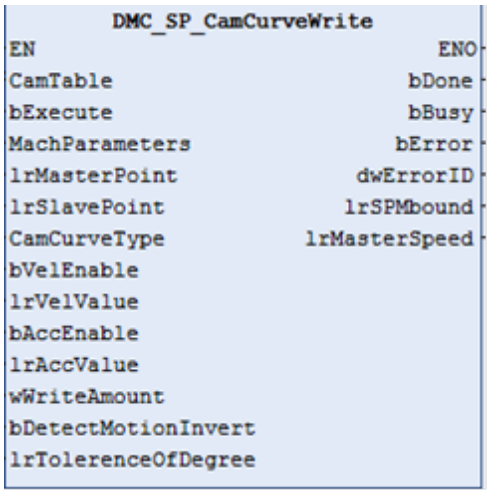
3. Wait for *bDone* to change to TRUE. The FB will generate key points data.



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

2.4.1.5 DMC_SP_CamCurveWrite

DMC_SP_CamCurveWrite is used in the servo press industry. It generates an electronic cam table by inputting the key point data and calculates the SPM upper limit value and the Master Axis speed according to the 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 Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	FALSE/TRUE (FALSE)	Start to run with rising-edge signal, only one time.
MachParameters	Servo press parameters	ReferenceDMC_SP_MACHINE_PARAMETERS[*3]	-	*bBusy* = FALSE & *bExecute* is at rising-edge
lrMasterPoint	Key points of master axis	[0..63] LREAL Array	0–360 (0.0)	*bBusy* = FALSE & *bExecute* is at rising-edge
lrSlavePoint	Key points of slave axis	[0..63] LREAL Array	0–360 (0.0)	*bBusy* = FALSE & *bExecute* is at rising-edge
CamCurveType	Curve type setup	DMC_CamCuryveType Array[0..62]	0–9[*2] (0)	*bBusy* = FALSE & *bExecute* is at rising-edge
bVelEnable	Enables the key point velocity configurations	WORD Array [0..63]	FALSE/TRUE (FALSE)	*bBusy* = FALSE & *bExecute* is at rising-edge

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrVelValue	Velocity value	[0..63] LREAL Array	LREAL*1 (0)	*bBusy* = FALSE & *bExecute* is at rising-edge
bAccEnable	Enables the key point acceleration configurations	WORD Array [0..63]	BOOL (FALSE)	*bBusy* = FALSE & *bExecute* is at rising-edge
lrAccValue	Acceleration value	[0..63] LREAL Array	LREAL*1 (0)	*bBusy* = FALSE & *bExecute* is at rising-edge
wWriteAmount	Key point position value	WORD	2–64 (0)	*bBusy* = FALSE & *bExecute* is at rising-edge
bDetectMotionInvert	Enables detecting motion invert	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	Crankshaft length (unit: mm)	LREAL	Positive
lrLLength	Connecting rod length(unit: mm)	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's SPM. (six times *lrSPMsys* is equal to the unit deg/sec)	LREAL	Positive

Name	Function	Data Type	Setting Value
lrGearRatio	Reducer gear ratio	LREAL	Positive
lrRPMmotor	Max RPM of the motor	LREAL	Positive

***Note4**: When *bDetectMotionInvert* is set to FALSE, *lrTolerenceOfDegree* will not work.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TURE when FB finishes writing ECAM table and calculating ECAM velocity upper bound value	BOOL	FALSE/TRUE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	FALSE/TRUE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	FALSE/TRUE (FALSE)
dwErrorID	Error Code of the errors	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

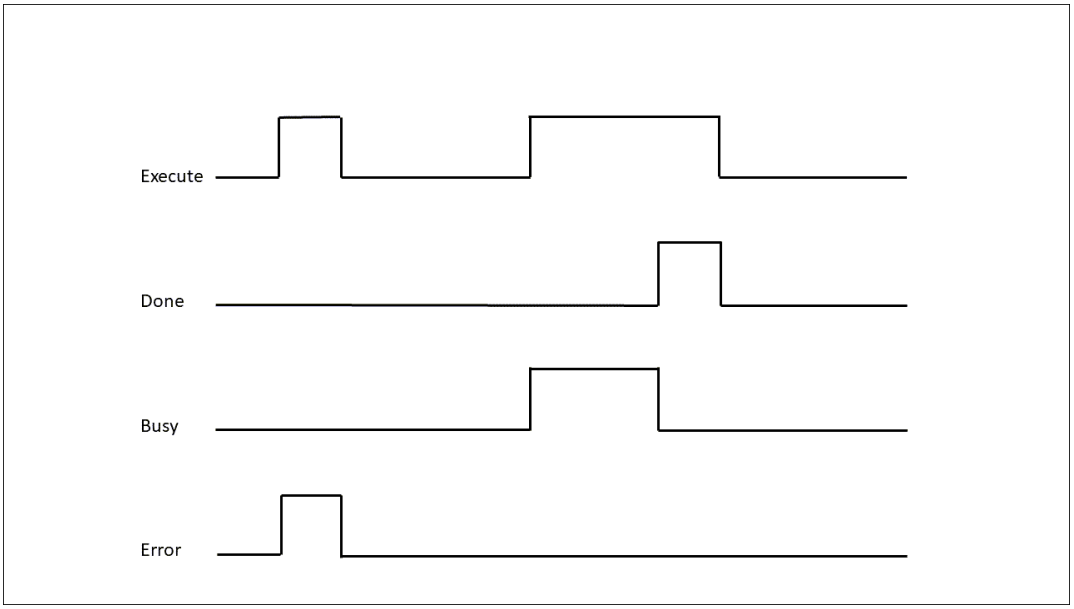
Negative values: 4.9406564584124654E-324 to 1.7976931348623157E+308

***Note2**: Valid *MachParameters.lrSPMsys* should not exceed the *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	The function block finishes curve key points generation.	• *bExecute* is FALSE. • *bExecute* is FALSE but *bDone* turns to TRUE, *bDone* will remain TRUE for one scan cycle and then change to FALSE.
bBusy	*bExecute* is on the rising edge	• *bDone* is TRUE. • *bError* is TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* is FALSE.

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
CamTable	The specified 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 the CAM table ID.
  - DMC_SP_CamCurveWrite will also calculate the upper limit of the SPM value. If system SPM value exceeds the limit, the error message will be shown. You must adjust the system SPM accordingly.
  - If set *bDetectMotionInvert* to TRUE and configure*lrTolerenceOfDegree*, the DMC_SP_CamCurveWrite function block can detect curve reversal in the motion.

- **Troubleshooting**
  - If an error occurs while running the instruction, *bError* will change to TRUE. You can refer to *dwErrorID* (Error Code) to check the problem.

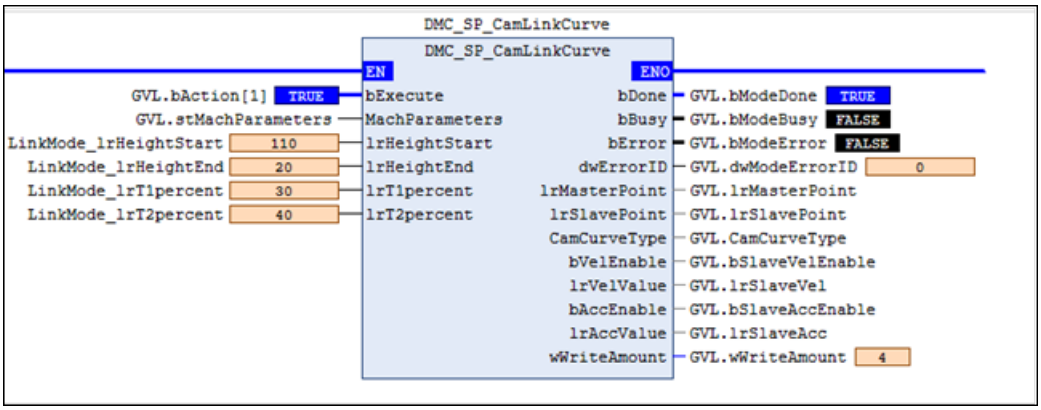
Error Code	Description	Contents	Corrective Action
0	No Error	-	-
1	Invalid parameters	Invalid parameters were set	Check whether these values are valid: • MachParameters.lrGearRatio • MachParameters.lrRPMmotor • MachParameters.lrSPMsys
4	Exceeds supported amount of key points	wWriteAmount is out of range	Check whether the values are in the valid range: • wWriteAmount

Error Code	Description	Contents	Corrective Action
5	SPM exceeds the motor upper limits	On-demand master axis SPM is over motor upper bound	Check whether the values are in the valid range: • MachParameters.lrSPMsys
6	CAM Data Type is not supported	CAM Data Type is not supported	Check whether the CAM data type is supported: • CamTable.byType • CamTable.byVarType
7	Reversal tolerance exceeds the limits	Tolerance of reversal degree is out of range	Check whether the values are in the valid range: • lrTolerenceOfDegree
8	Curve reversal motion detected	One or more reversal motion are found	To adjust the input values: • lrVelValue • lrAccValue • lrTolerenceOfDegree
Errors raised by internal referenced function blocks			
0x188B5	wWriteAmount out of range	WriteAmount input error	Check and correct the input value of WriteAmount and then run the function block.
0x188B6	Invalid master value of key points	Invalid master position	Correct the input of master position and then restart the function block .
0x188B7	Invalid acceleration values of key points	Invalid acceleration	Correct the acceleration input value of master position and then restart the function block.
0x188B8	Invalid acceleration settings	Invalid acceleration setting	Correct the velocity, acceleration and curve type inputs and then restart the function block.
0x188B9	The curve type is not supported	Invalid curve type setting	Correct the curve type and then restart the function block.
0x188BA	There is no boundary condition or wrong boundary condition	Spine has no boundary.	Check whether there’s boundary condition (Nature or Clamp) for the selected curve's previous and subsequent segments, and ensure that the boundary conditions at both ends are consistent. Then, restart the function block.
0x188BB	The cam table data is written by other function	Failed to write to CAM.	Check if the cam table has been rewritten by other function blocks, Restart the function block after the rewriting have completed.

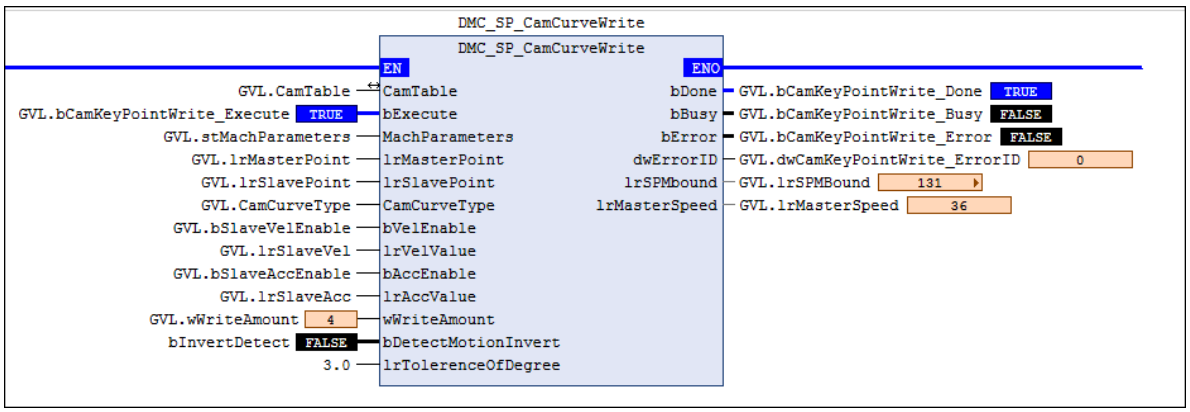
- **Example**

1. Generate ECAM table.

   Select DMC_SP_CamLinkCurve FB (or other function block, such as DMC_SP_CamPulse1Curve) to generate Key-Point Data.

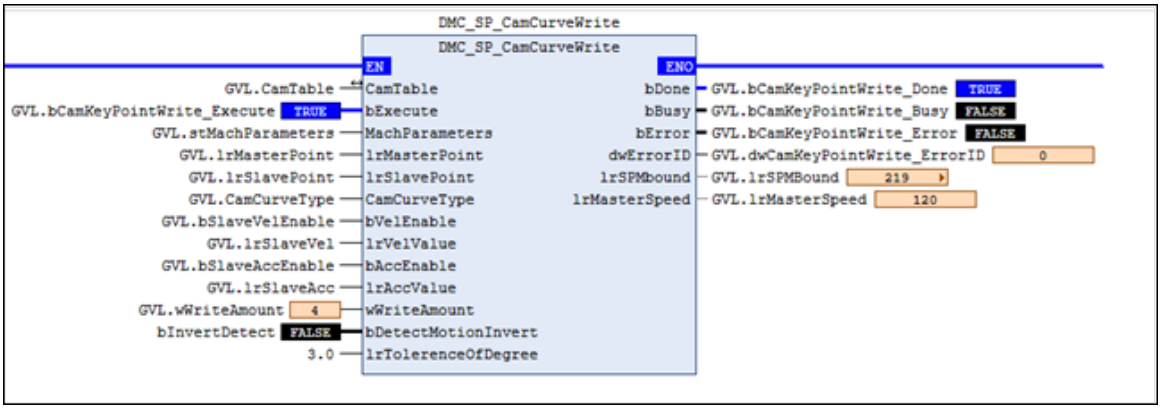


2. After the key-points data is generated, execute DMC_SP_CamCurveWrite to write the specific ECAM table.



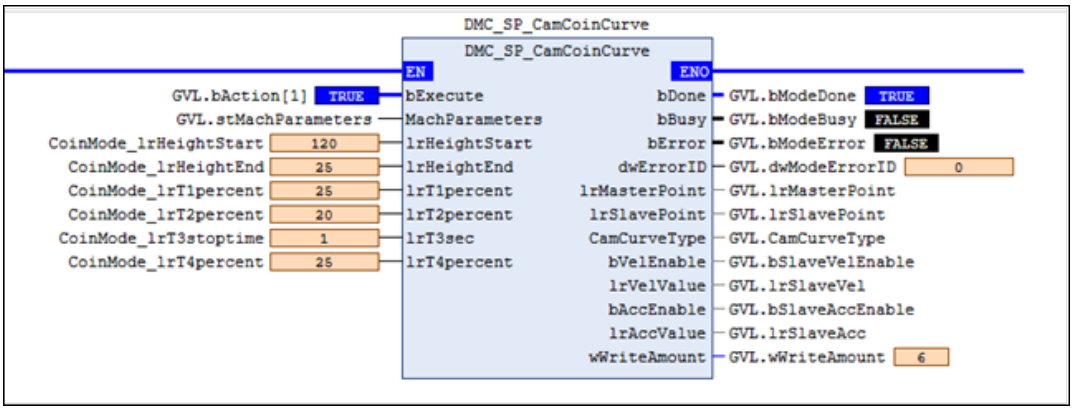
3. Calculate the upper limite of SPM. Check whether SPM exceeds the SPM limit.

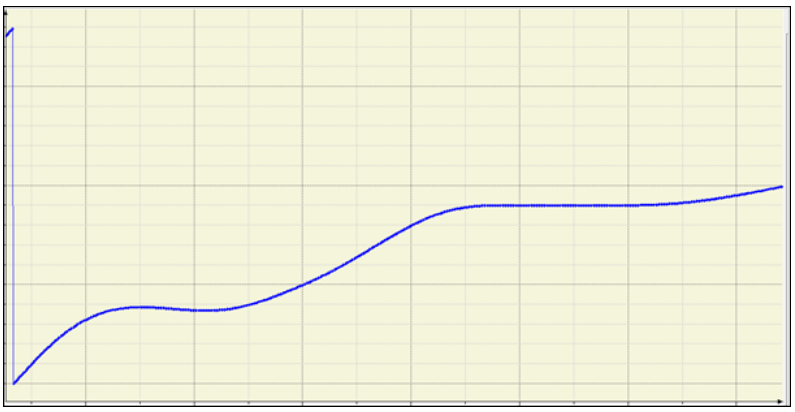
   Set Motor RPM=300, SPM = 350, GearRatio = 1.0, After DMC_SP_CamxxxCurve FB generate key points, DMC_SP_CamCurveWrite will calculate the upper limit of SPM. If the input SPM exceeds the SPM limit, the function block will show an error code. In this case, adjust SPM and run DMC_SP_CamCurveWrite again.



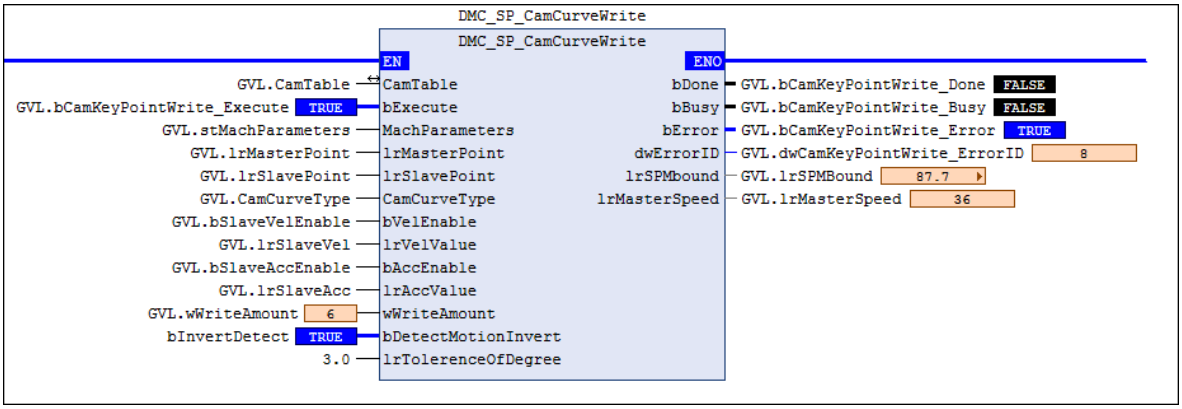
4. Curve Invert Detect

   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.





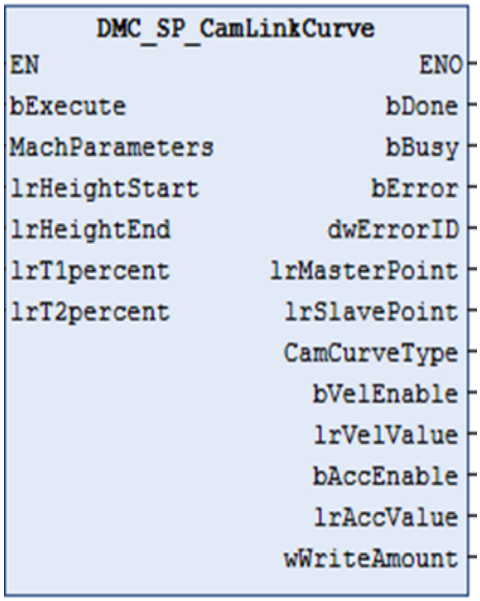
5. Set *bInvertDetect* to TRUE, and set *lrTolerenceOfDegree* to 3.0 and run DMC_SP_CamCurveWrite again. *bError* will be TRUE, and *ErrorID* will show 8.



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

2.4.1.6 DMC_SP_CamLinkCurve

DMC_SP_CamLinkCurve is used in the servo press industry. It generates cam key point data according to the extended curve algorithm and generates an electronic cam table through DMC_SP_CamCurveWrite FB. You can use the generated electronic cam table for the servo press processing.

FB/FC	Instruction	Graphic Expression	ST Language
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 Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	FALSE/TRUE (FALSE)	-
MachParameters	Servo press 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– 100	*bBusy*=FALSE & *bExecute* is at rising-edge
lrT2percent	T2 percentage of time	LREAL*1	Positive (0) [Range] 0–100	*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	Crankshaft length (unit: mm)	LREAL	Positive
lrLLength	Connectting rod (unit: mm)	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's SPM. (six times *lrSPMsys* is equal to the unit deg/sec)	LREAL	Positive
lrGearRatio	Reducer gear ratio	LREAL	Positive
lrRPMmotor	Max RPM of the motor	LREAL	Positive

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TURE when the function block finishes generating coin key-points	BOOL	FALSE/TRUE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	FALSE/TRUE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	FALSE/TRUE (FALSE)
dwErrorID	Error Code of the error	DWORD	16#00000000–16#FFFFFFFF (16#00000000)
lrMasterPoint	Master position of the key point	[0..63] LREAL Array	0.0–360.0 (0.0)
lrSlavePoint	Slave position of the key point	[0..63] LREAL Array	0.0–360.0 (0.0)
CamCurveType	Curve type between key points	[0..62] DMC_CamCurveType Array	0–9[*2](0)
bVelEnable	Enables the key point velocity configurations	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrVelValue	Velocity value	[0..63] LREAL Array	LREAL[*1](0)
bAccEnable	Enables the key point acceleration configurations	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrAccValue	Acceleration value	[0..63] LREAL Array	LREAL[*1](0)
wWriteAmount	Key Point write amount	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)

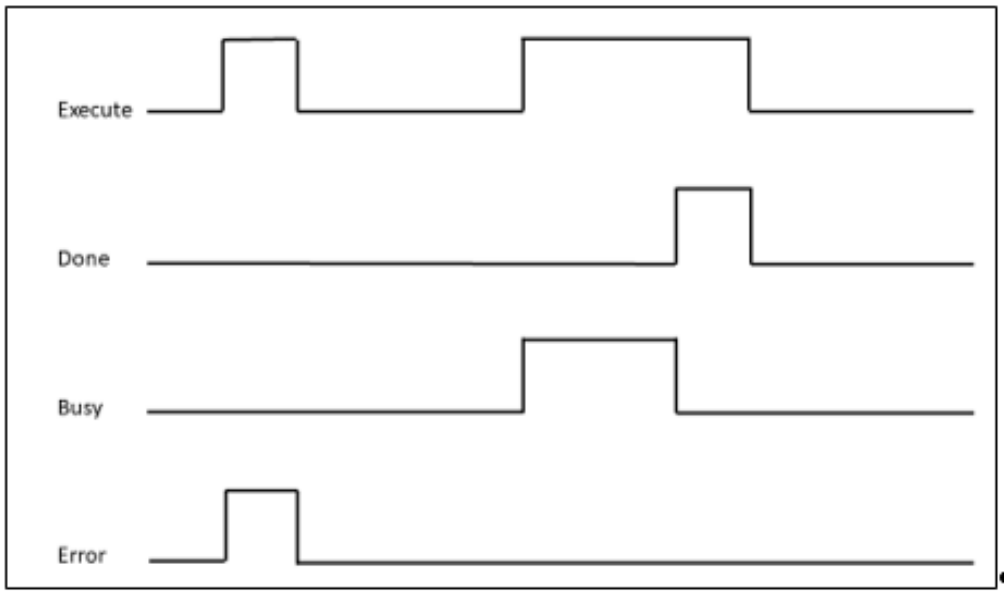
Cubic_Spline_Clamp (8)

Cubic_Spline(9)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The function block finishes curve key points generation.	• *bExecute* is FALSE. • *bExecute* is FALSE but *bDone* turns to TRUE, *bDone* will remain TRUE for one scan cycle and then change to FALSE.
bBusy	*bExecute* is at rising-edge.	• *bDone* is TRUE. • *bError* is TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* is FALSE.

- **Timing Diagram of Output Parameter Changes**

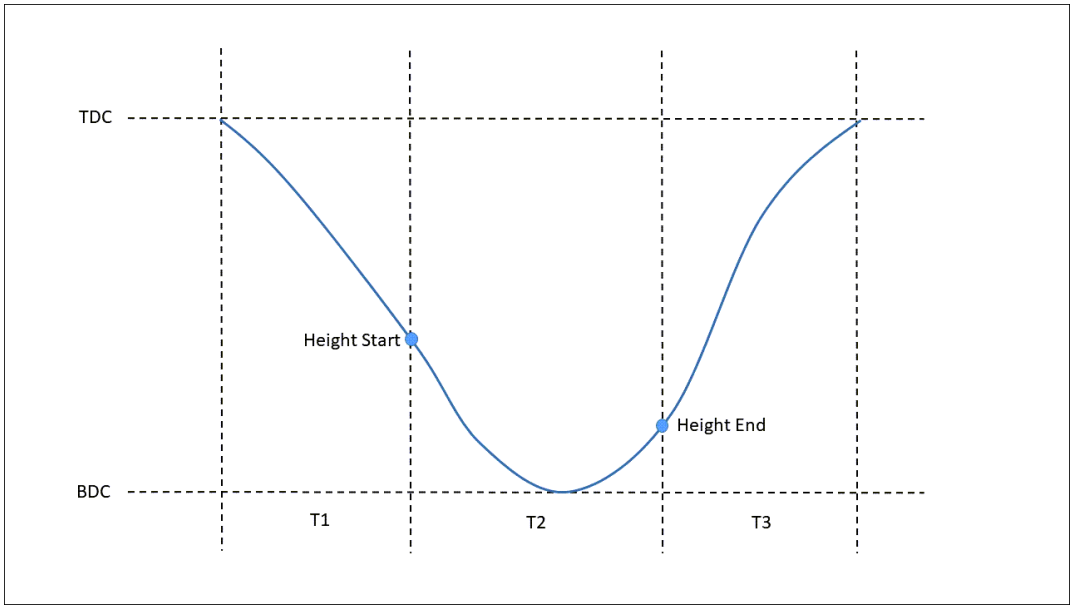


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

- **Function**

  1. Input T1, T2, and function block will calculate the T3 time percent.
  2. The Slider will move from TDC to Height Start at T1 time percent.
  3. The Slider will move to Height End at T2 time percent.
  4. Return to TDC at T3 time percent.



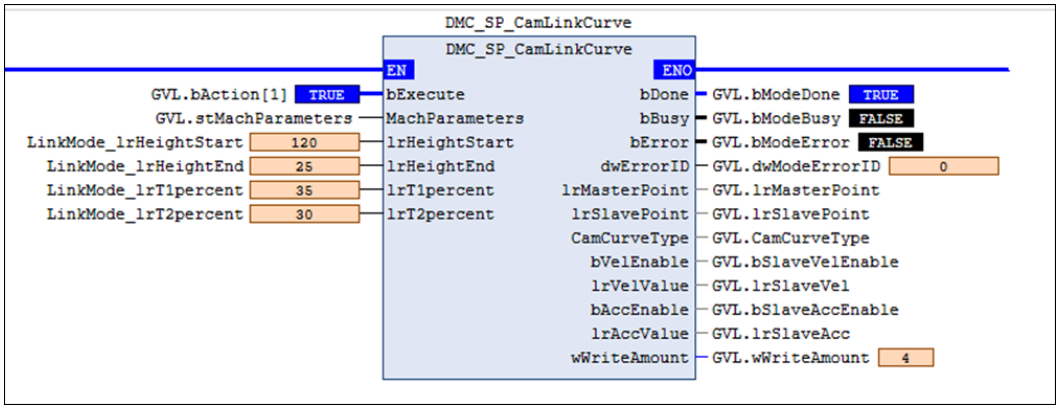
- **Troubleshooting**

Error Code	Description	Contents	Corrective Action
0	No Error	-	-
1	Invalid parameters	Invalid mechanical parameters were set	Check whether the values are in the valid range • MachParameters.lrRLength • MachParameters.lrLLength
2	Invalid slider height	Incorrect range of slider height was set	Check whether the values are in the valid range • lrHeightStart • lrHeightEnd
3	Invalid time percentage	Invalid time percentage was set. (0% or sum of time percentage exceeds 100%)	Check whether the values are in the valid range • lrT1percent • lrT2percent

- **Example**

  1. Input *MachParameters.lrRLength* and *MachParameters.lrLLength* .
  2. input *lrHeightStart* and *lrHeightEnd*.
  3. Input T1 time percent and T2 time percent.
  4. Set *bExecute* to TRUE.

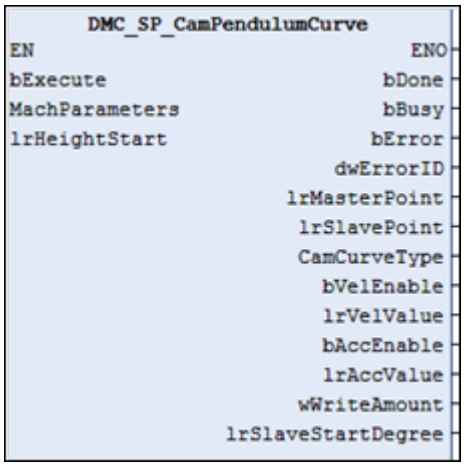
5. When *bDone* to change to TRUE, the key-points data is generated.



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

2.4.1.7 DMC_SP_CamPendulumCurve

DMC_SP_CamPendulumCurve is used in the servo press industry. It generates cam key point information according to the pendulum curve algorithm, and generates an electronic cam table through DMC_SP_CamCurveWrite FB. You 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 Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	FALSE/TRUE (FALSE)	Start to run with the rising edge signal.
MachParameters	Servo press 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–(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	Crankshaft length (unit: mm)	LREAL	Positive
lrLLength	Connecting rod (unit: mm)	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master-axis's SPM. (six times *lrSPMsys* is equal to the unit deg/sec)	LREAL	Positive

Name	Function	Data Type	Setting Value
lrGearRatio	Reducer gear ratio	LREAL	Positive
lrRPMmotor	Max RPM of the motor	LREAL	Positive

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when the function block completed generating key points	BOOL	FALSE/TRUE (FALSE)
bBusy	TURE when the function block is running	BOOL	FALSE/TRUE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	FALSE/TRUE (FALSE)
dwErrorID	Error code of the error	DWORD	16#00000000–16#FFFFFFFF (16#00000000)
lrMasterPoint	Master position of the key point	[0..63] LREAL Array	0.0–360.0 (0.0)
lrSlavePoint	Slave position of the key point	[0..63] LREAL Array	0.0–360.0 (0.0)
CamCurveType	Curve type between key points	[0..62] DMC_CamCurveType Array	0–9[*2] (0)
bVelEnable	Enables the key point velocity configurations	[0..63] BOOL Array	FALSE/TRUE (FALSE)
lrVelValue	Velocity value	[0..63] LREAL Array	LREAL (0)
bAccEnable	Enables the key point acceleration configurations	[0..63] BOOL Array	FALSE/TRUE (FALSE)
lrAccValue	Acceleration value	[0..63] LREAL Array	LREAL (0)
wWriteAmount	Key Point write amount	WORD	0–64 (0)
lrSlaveStartDegree	Based on *MachParameters.lrRLength*, *MachParameters.lrLLength*, and *lrHeightStart* to calculate the degree 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)

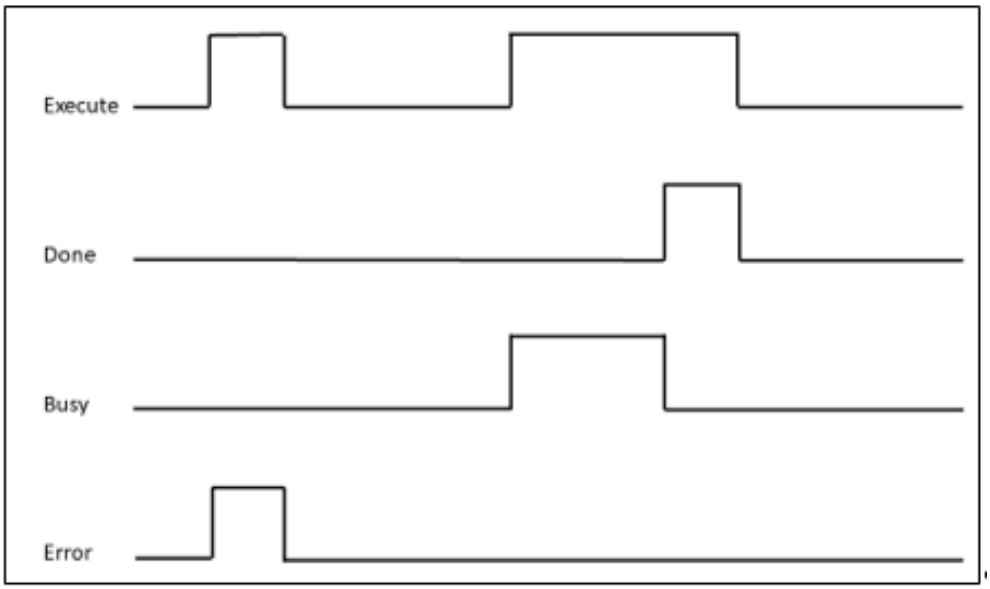
Cubic_Spline_Clamp (8)

Cubic_Spline (9)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The function block finishes curve key points generation.	• *bExecute* is FALSEx. • *bExecute* is FALSE but *bDone* turns to TRUE, *bDone* will remain TRUE for one scan cycle and then change to FALSE.
bBusy	*bExecute* is TRUE.	• *bDone* is TRUE. • When *bError* is TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* is FALSE.

- **Timing Diagram of Output Parameter Changes**

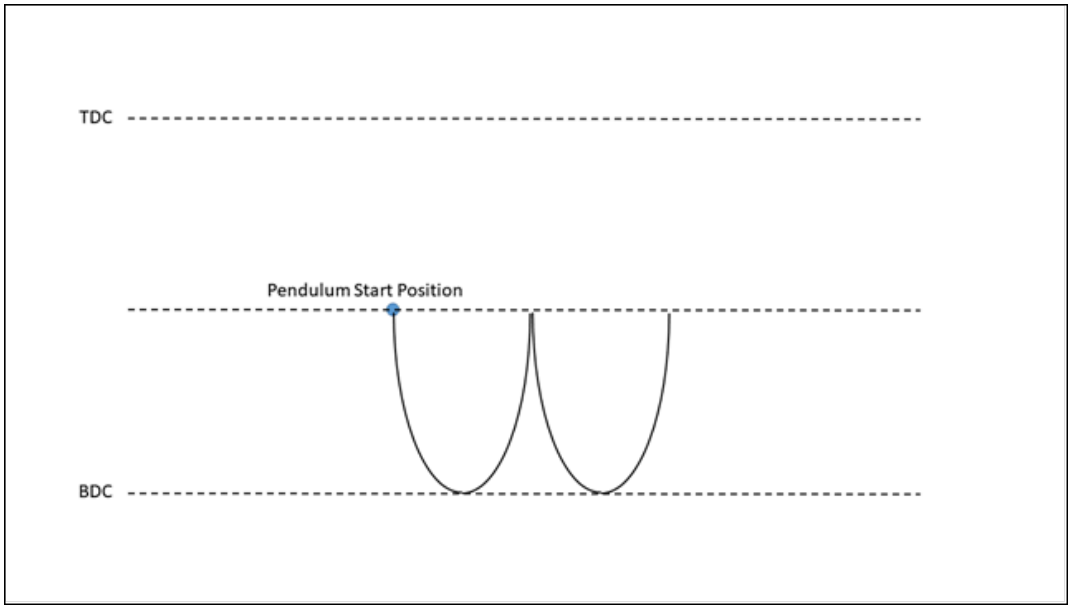


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

- **Function**
  - After starting the motion, the slider will move from Start Position to End Position.
  - During the execution, move back and forth between the starting point and the BDC.

- Return to TDC at last.
- Run DMC_SP_CamCurveWrite to generate and write ECAM table after this function block is done.
- Pendunlum curve diagram:

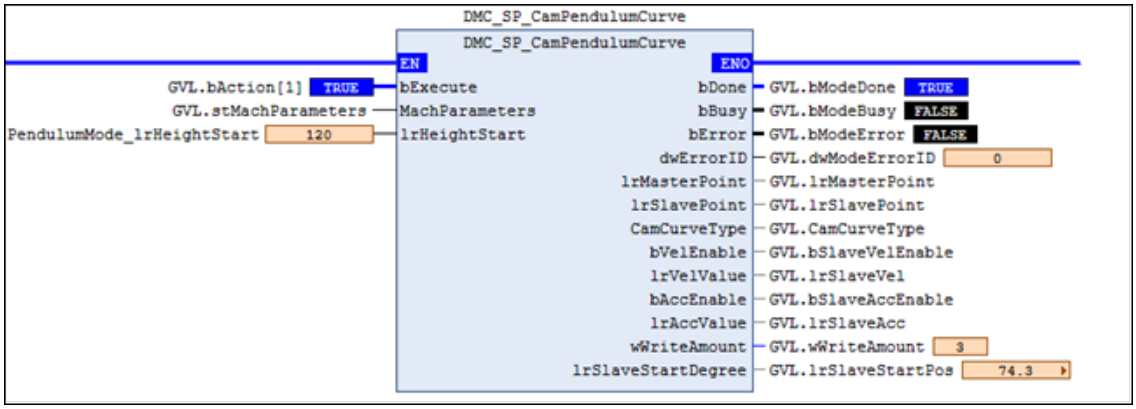


- **Troubleshooting**

Error Code	Description	Contents	Corrective Action
0	No Error	-	-
1	Invalid parameters	Invalid electromechanical parameters were set	Check whether the values are within the valid range • MachParameters.lrRLength • MachParameters.lrLLength
2	Invalid slider height	Incorrect range of slide height was set	Check whether the values are within the valid range • lrHeightStart

- **Example**

  1. Set *fMachParameters.lrRLength*=90.0mm, *MachParameters.lrLLength*=670mm, and *lrHeightStart*=120.0mm.
  2. Set *bExecute* from FALSE to TRUE to trigger curve key-points calculation. Once the calculation completes, *bDone* will be TRUE, and generates related key points arrays.

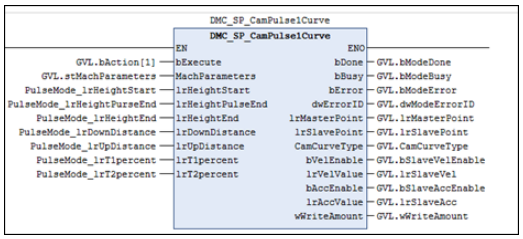


- **Supported Devices**

  - AX-308E, AX-8

2.4.1.8 DMC_SP_CamPulse1Curve

DMC_SP_CamPulse1Curve is used in the servo press industry. It generates cam key point information according to the pulse 1 curve algorithm, and generates an electronic cam table through DMC_SP_CamCurveWrite function block.

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* to 5–10 degree for Pulse1 Curve to run DMC_SP_CamCurveWrite. Base on 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 Value (Default Value)	Timing to Take Effect
bExecute	Runs the instruction when *bExecute* changes from FALSE to TRUE.	BOOL	FALSE/TRUE (FALSE)	Start to run with the rising-edge signal, only one time.
lrRLength	Crankshaft length	LREAL[*1]	Positive (0)	*bBusy* = FALSE & *bExecute* is at rising-edge
lrLLength	Connecting rod length	LREAL[*1]	Positive (0) [Range]lrLLength > 2 * lrRLength	*bBusy* = FALSE & *bExecute* is at rising-edge
MachParameters	Servo press machine electromechanical parameters	Refer to DMC_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

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
lrHeightPulseEnd	Pulse Process End Position	LREAL*1	[Range] 0–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–lrHeightStart	*bBusy* = FALSE & *bExecute* is at rising-edge
lrUpDistance	The distance above the slider	LREAL*1	Positive (0) [Range] 0–lrDownDistance	*bBusy* = FALSE & *bExecute* is at rising-edge
lrT1percent	T1 time proportion	LREAL*1	Positive (0) [Range] 0–100	*bBusy* = FALSE & *bExecute* is at rising-edge
lrT2percent	T2 time proportion	LREAL*1	Positive (0) [Range] 0–100	*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	Value
lrLLength	Connecting rod length	LREAL	lrLLength > (2 * lrRLength)
lrSPMsys	Virtual master axis's SPM	LREAL	Positive
lrGearRatio	Reducer gear ratio	LREAL	Positive
lrRPMmotor	Max RPM of the motor	LREAL	Positive
lrRLength	Crankshaft length	LREAL	Positive

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
bDone	TRUE when FB finishes generating key-points	BOOL	FALSE/TRUE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	FALSE/TRUE (FALSE)
bError	TRUE when an error occurs within the instruction	BOOL	FALSE/TRUE (FALSE)
dwErrorID	Error code when running FB	DWORD	16#00000000–16#FFFFFFFF (16#00000000)

Name	Function	Data Type	Output Range (Default Value)
lrMasterPoint	Master positions	[0..63] LREAL Array	0.0–360.0 (0.0)
lrSlavePoint	Slave positions	[0..63] LREAL Array	0.0–360.0 (0.0)
CamCurveType	Key point to key point curve type	[0..62] DMC_CamCurveType Array*1	0–9*2 (0)
bVelEnable	Enables the key point velocity configurations	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrVelValue	Velocity value	[0..63] LREAL Array	LREAL*1 (0)
bAccEnable	Enables the key point acceleration configurations	[0..63] BOOL Array	TRUE/FALSE (FALSE)
lrAccValue	Acceleration value	[0..63] LREAL Array	LREAL*1 (0)
wWriteAmount	Key Point write amount	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)

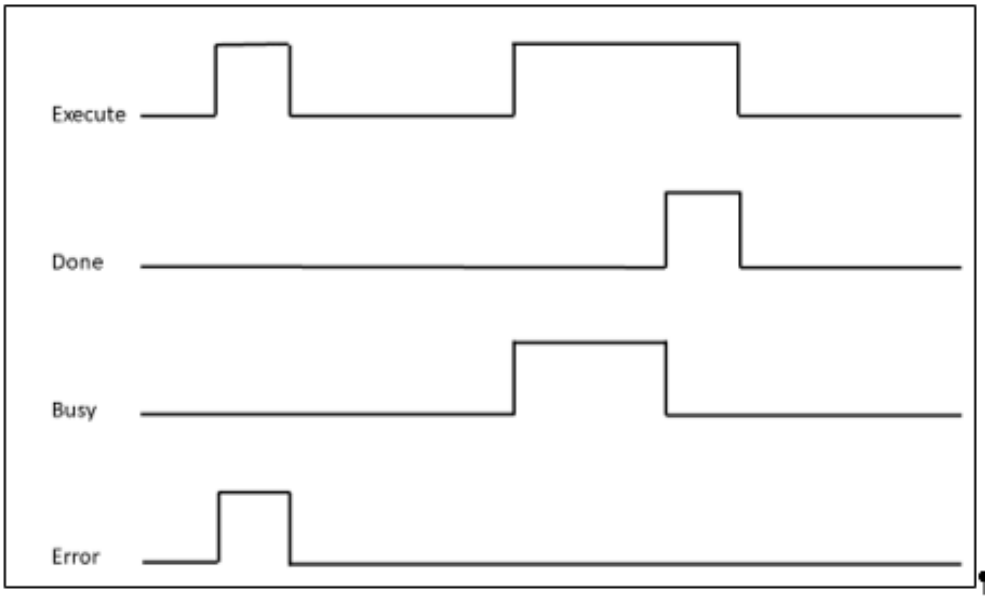
Cubic_Spline_Clamp (8)

Cubic_Spline (9)

- **Output Update Timing**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
bDone	The function block finishes curve key points generation.	• *bExecute* is FALSE. • *bExecute* is FALSE but *bDone* turns to TRUE, *bDone* will remain TRUE for one scan cycle and then change to FALSE.
bBusy	*bExecute* is TRUE.	• *bDone* is TRUE. • *bError* is TRUE.
bError	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*bExecute* is FALSE.

- **Timing Diagram of Output Parameter Changes**

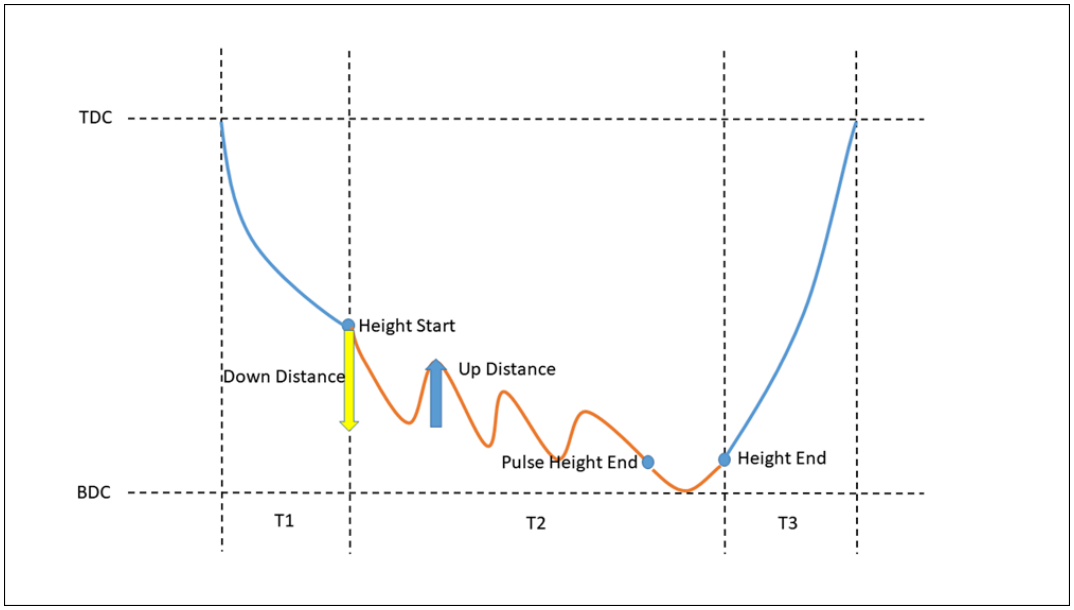


- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
-	-	-	-	-

- **Function**

  - Generate the key point data for DMC_SP_CamCurveWrite function block.
  - Input T1 and T2 time percent. The function block will calculate T3 time percent.
  - At the T1 time ratio, the slider will move from TDC to the starting position of the processing
  - At the T2 time percentage, the slider will move to the end position of the pulse peak.
  - The slider will return to the end position.
  - At the T3 time ratio, return to TDC.



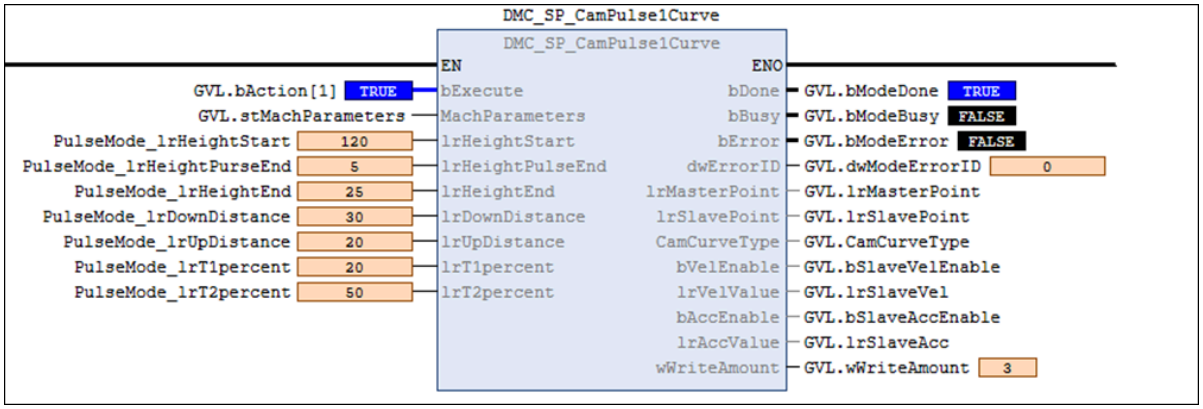
- **Troubleshooting**

Error Code	Description	Contents	Corrective Action
0	No Error	-	-
1	Invalid machine parameters	Invalid electromechanical parameters were set	Check whether these values are within the valid range • *MachParameters.lrRLength* • *MachParameters.lrLLength*
2	Invalid slider height	Incorrect range of slider height was set	Check whether the values are within the valid range • *lrHeightStart* • *lrHeightPulseEnd* • *lrHeightEnd* • *lrDownDistance* • *lrUpDistance*
3	Invalid time percentage	Invalid time percentage was set. (0% or sum of time percentage exceeds 100%)	Check whether the values are within the valid 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*

- **Example**

1. Input Crank Length and Link Length (Link > 2* Crank).
2. Input Height Start, Height End, and Height Pulse End (Height Start > Height PulseEnd).
3. Input slider down and up distance (Down > Up).
4. Input T1percent and T2percent.
5. Set *bExecute* to TRUE.
6. When*bDone* to change to TRUE, the function block will generate the key point data.

7. Transfer the key point data to DMC_SP_CamCurveWrite to generate CamTable.



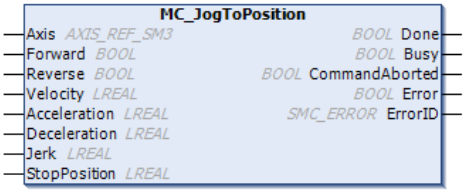
- **Supported Devices**
  - AX-308E, AX-8

2.5 DL_PLCopenPart3

2.5.1 Motion Control Instructions

2.5.1.1 MC_JogToPosition

MC_JogToPosition controls the axis to move at constant velocity in the specified direction (Forward or Reverse). When Forward or Reverse changes from TRUE to FALSE, the axis decelerates to stop at the specified target position.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_JogToPosition		MC_JogToPosition_instance(Axis :=, Forward :=, Reverse :=, Velocity :=, Acceleration :=, Deceleration :=, Jerk :=, StopPosition :=, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>)

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Forward	Run the instruction when *Forward* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Reverse	The instruction will run when *Reverse* turns from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Velocity	Target velocity (user unit/s)	LREAL	Positive number or 0 (0)	*Forward* or *Reverse* = TRUE.
Acceleration	Acceleration (user unit/s^2)	LREAL	Positive number or 0 (0)	*Forward* or *Reverse* = TRUE.
Deceleration	Deceleration (user unit/s^2)	LREAL	Positive number or 0 (0)	*Forward* or *Reverse* = TRUE.
Jerk	Jerk (user unit/s^3)	LREAL	Positive number or 0 (0)	*Forward* or *Reverse* = TRUE.
StopPosition	Final stop position in a rotary axis system	LREAL	Negative, positive number or 0 (0)	*Forward* or *Reverse* = TRUE.

Note: Either *Forward* or *Reverse*. Both changing to TRUE will not work.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Done	TRUE when the master axis is desynchronized with the slave axis	BOOL	TRUE/FALSE (FALSE)

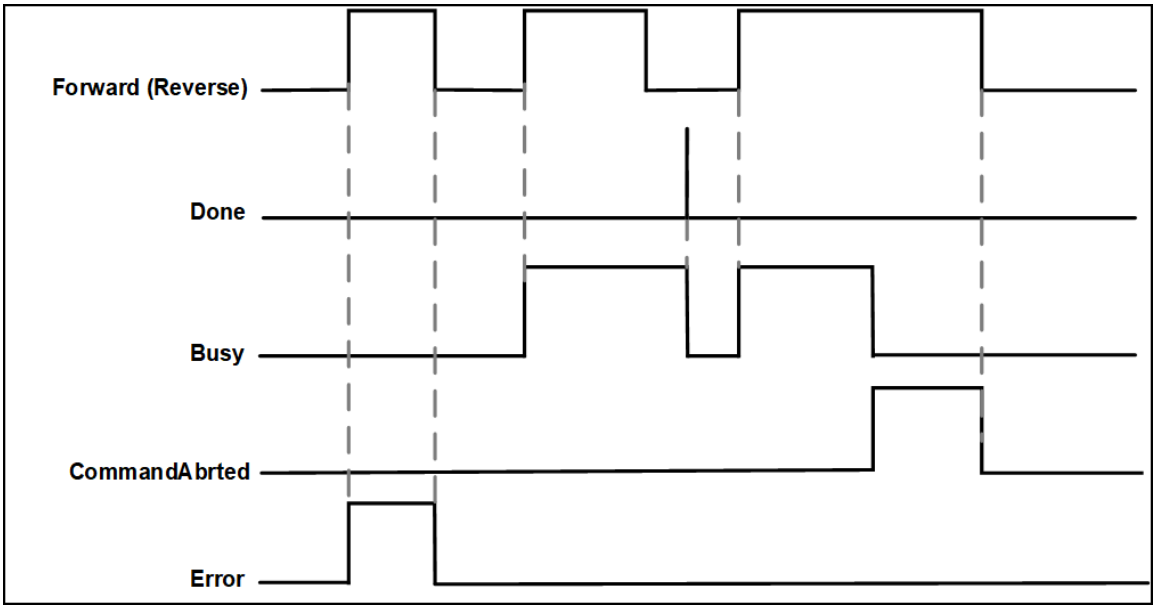
Name	Function	Data Type	Output Range (Default Value)
Busy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

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

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Done	The axis reaches the specified position.	• *Forward* or *Reverse* is FALSE. • *Done* is TRUE for only one scan cycle and immediately turns to FALSE.
Busy	*Forward* or *Reverse* = TRUE	• *Error* is TRUE • *CommandAborted* is TRUE
CommandAborted	• The function block is interrupted by another function block • The function block is interrupted by MC_Stop	*Forward* or *Reverse* is FALSE
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Forward* or *Reverse* is FALSE (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Forward* or *Reverse* is TRUE and *Busy* is FALSE.

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

- **Function**
  - This function block is used to control rotary axes.
  - When both *Forward* and *Reverse* are FALSE, only one can be changed to TRUE. They cannot be set to TRUE at the same time.
  - When *Forward* is TRUE, set *Forward* to FALSE or *Reverse* to TRUE. The axis will decelerate to stop at the target position.
  - When *Reverse* is TRUE, set *Reverse* to FALSE or *Forward* to TRUE. The axis will decelerate to stop at the target position.
  - Actual stop position = take the remainder of StopPosition / Axis.fPositionPeriod, for example:

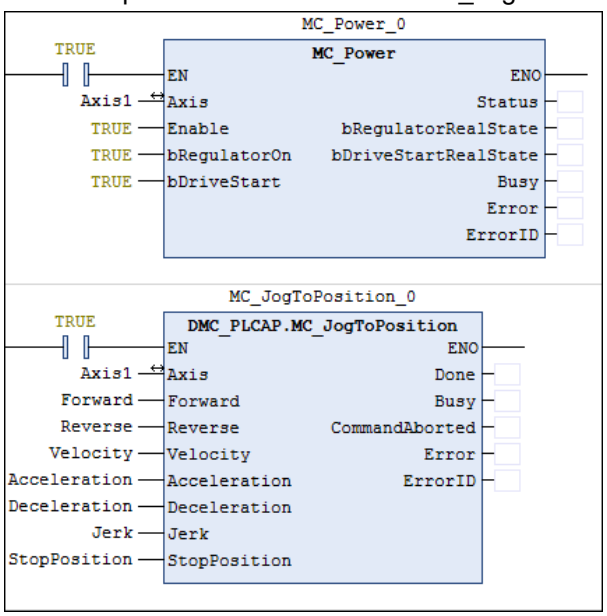
    Suppose Axis.fPositionPeriod is 3600

StopPosition	4000	3000	3600	−300	−3000	−3600	−7000
Actual stop position	400	3000	0	3300	600	0	200

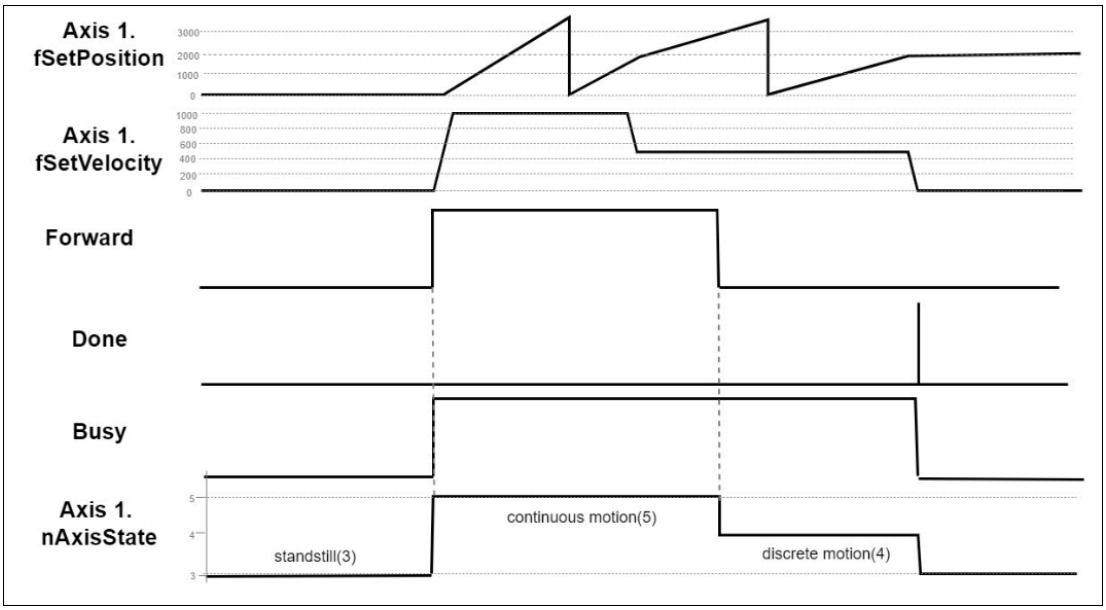
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

The example illustrates how to run MC_JogToPosition and the motion path.



- Timing Diagram

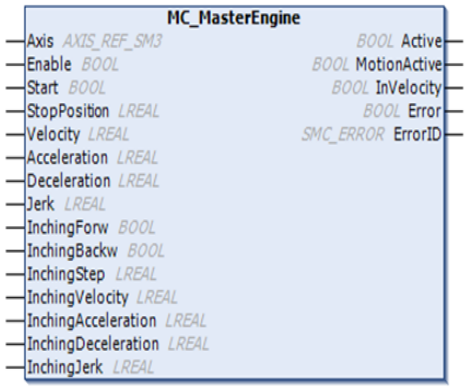


  - When *Forward* is TRUE, this function block moves forward at a constant velocity according to the set *Velocity*, *Acceleration*, *Deceleration*, and *Jerk* set. During the motion, *Busy* is TRUE, and the state machine becomes Continuous motion.
  - When *Forward* is TRUE, change the target velocity to 500, and the axis speed will be adjusted to the new speed according to the set *Acceleration*, *Deceleration*, and *Jerk*.
  - When *Forward* is FALSE, the axis will start to move towards the absolute position 2,000, and the state machine changes to Discrete motion during the movement. When the absolute position 2,000 is reached, *Done* turns to TRUE (for one period), and *Busy* turns to FALSE.

- **Supported Devices**
  - AX-series motion controller

2.5.1.2 MC_MasterEngine

MC_MasterEngine controls the virtual master axis in packaging machine. It enables constant axis motion at specified velocity and movement patterns, with stopping at target positions. Additionally, it provides inching movement to the designated direction.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_MasterEngine		MC_ MasterEngine _instance(Axis :=, Enable :=, Start :=, StopPosition :=, Velocity :=, Acceleration :=, Deceleration :=, Jerk :=, InchingForw :=, InchingBackw :=, InchingStep :=, InchingVelocity :=, InchingAcceleration :=, InchingDeceleration :=, InchingJerk :=, Active => MotionActice => InVelocity => Error =>, ErrorID =>)

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	Run the instruction when *Enable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Start	Start to move at constant speed.	BOOL	TRUE/FALSE (FALSE)	*Enable* = TRUE
StopPosition	Stop position	LREAL	Negative, positive number or 0 (0)	*Enable* = TRUE and *Start* = TRUE
Velocity	Target speed (user unit/s)	LREAL	Positive number (0)	*Enable* = TRUE and *Start* = TRUE
Acceleration	Acceleration (user unit/s^2)	LREAL	Positive number (0)	*Enable* = TRUE and *Start* = TRUE
Deceleration	Deceleration (user unit/s^2)	LREAL	Positive number (0)	*Enable* = TRUE and *Start* = TRUE
Jerk	Jerk (user unit/s^3)	LREAL	Positive number (0)	*Enable* = TRUE and *Start* = TRUE
InchingForw	Inching in the forward direction	BOOL	TRUE/FALSE (FALSE)	*Enable* = TRUE and *InchingBackw* = FALSE

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
InchingBackw	Inching in the backward direction	BOOL	TRUE/FALSE (FALSE)	*Enable* = TRUE and *InchingForw* = FALSE
InchingStep	Maximum distance for inching	LREAL	Positive Number or 0 (0)	*InchingForw* = TRUE or *InchingBackw* = TRUE
InchingVelocity	Velocity for inching (user unit/s)	LREAL	Positive number (0)	*InchingForw* = TRUE or *InchingBackw* = TRUE
InchingAcceleration	Acceleration for inching (user unit/s^2)	LREAL	Positive number (0)	*InchingForw* = TRUE or *InchingBackw* = TRUE
InchingDeceleration	Deceleration for inching (user unit/s^2)	LREAL	Positive number (0)	*InchingForw* = TRUE or *InchingBackw* = TRUE
InchingJerk	Jerk for inching (user unit/s^3)	LREAL	Positive number (0)	*InchingForw* = TRUE or *InchingBackw* = TRUE

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
Active	TRUE when *Enable* is TRUE	BOOL	TRUE/FALSE (FALSE)
MotionActice	TRUE when the uniform or inching motion is triggered	BOOL	TRUE/FALSE (FALSE)
InVelocity	TRUE when the set Velocity is reached	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

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

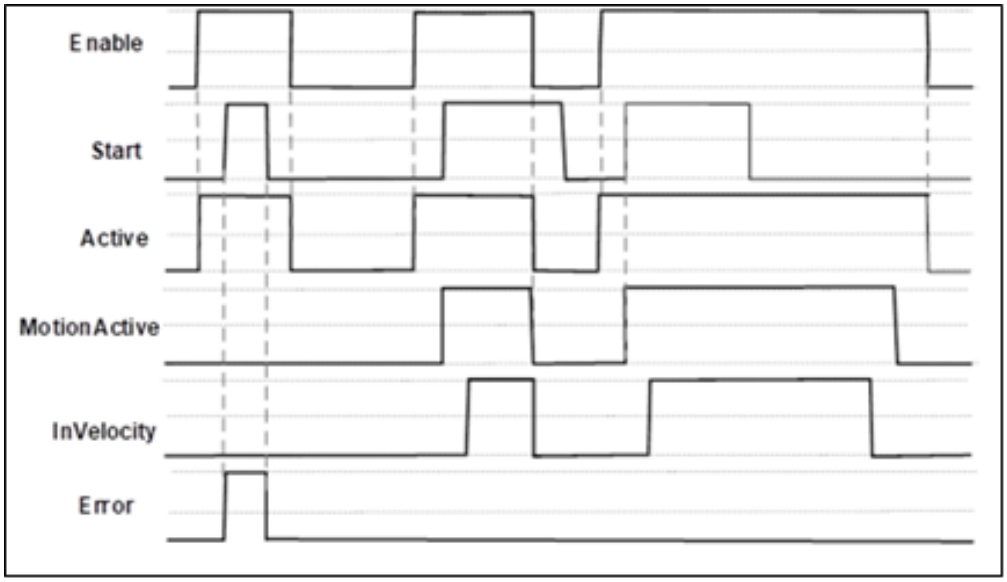
- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Active	*Enable* is TRUE.	*Enable* is FALSE
MotionActice	*Enable* is TRUE and *Start*, *InchingForw*, or *InchingBackw* is TRUE.	• *Error* is TRUE. • *Enable* is TRUE and *Start*, *InchingForw*, or *InchingBackw* is FALSE.
InVelocity	The set *Velocity* is reached.	*Enable* is TRUE and *Velocity* is given a new value.
Error		*Forward* or *Reverse* is FALSE. (Error Code is cleared.)

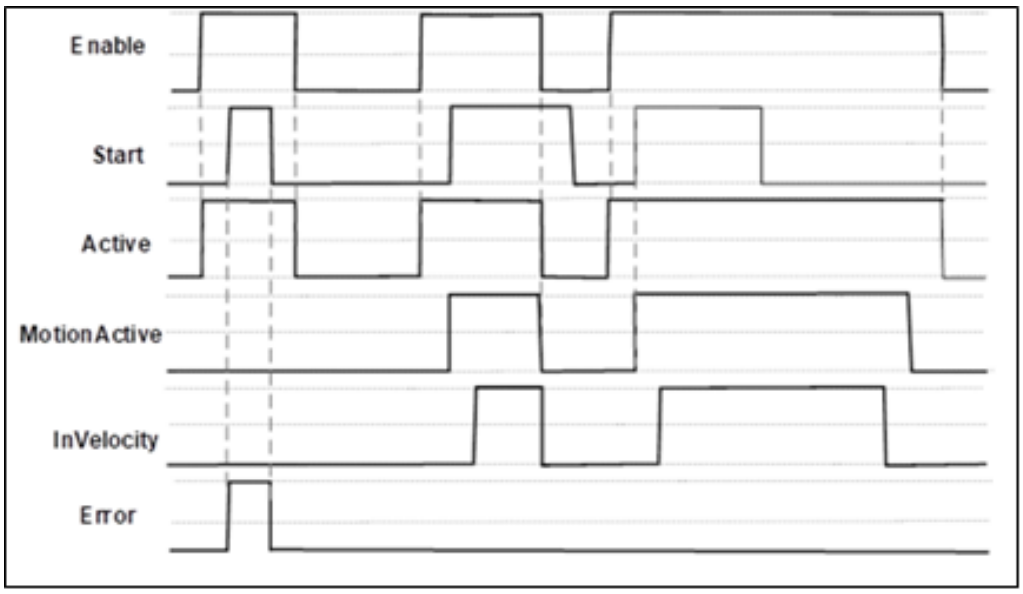
Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
ErrorID	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	

- **Timing Diagram of Output Parameter Changes**

Uniform motion



Inching motion



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Forward* or *Reverse* is TRUE and *Busy* is FALSE.

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

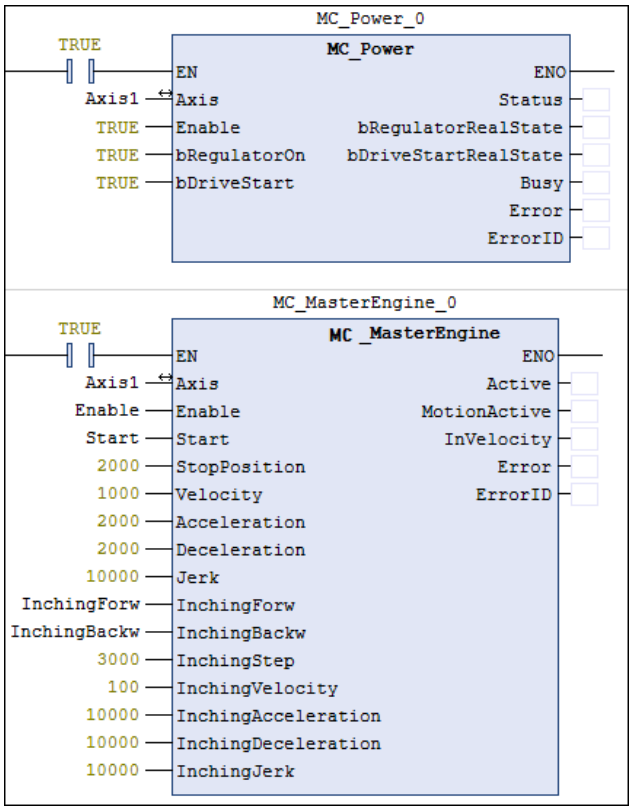
- **Function**
  - To perform a content velocity motion, the axis must be a rotary axis.
  - If *Start* = TRUE, the axis will move at a constant velocity. *Velocity* > = 0 moves forward. *Velocity* < 0 moves backward. When *Start* is set to FALSE, the axis will slow down and stop at *StopPosition*.
  - If *InchingForw* is TRUE, the axis will move the *InchingStep* distance in the forward direction, and if *InchingForw* is set to FALSE during the motion, the axis will slow down and stop early.
  - If *InchingBackw* is TRUE, the axis will move the *InchingStep* distance in the backward direction, and if *InchingBackw* is set to FALSE during the motion, the axis will decelerate and stop early.
  - If both *InchingForw* and *InchingBackw* are FALSE, only either of them can be set to TRUE. If both are set to TRUE, it will not work.
  - If *InchingForw* is TRUE, then turns to FALSE or set *InchingBackw* to TRUE, the axis will decelerate to stop.
  - If *InchingBackw* is TRUE, then turns to FALSE or sets *InchingForw* to TRUE, the axis will decelerate to stop.
  - Actual stop position = take the remainder of StopPosition / Axis.fPositionPeriod, for example:

    Suppose *Axis.fPositionPeriod* is 3600.

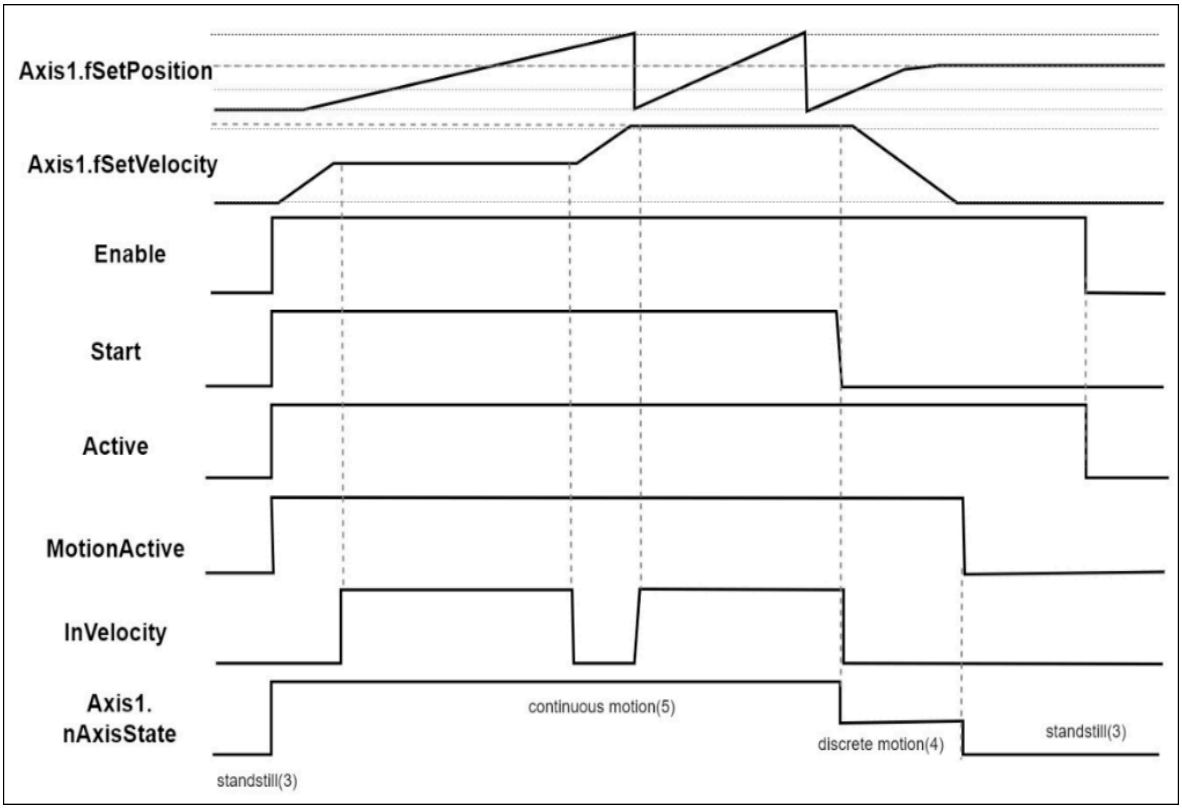
StopPosition	4000	3000	3600	−300	−3000	−3600	−7000
Actual stop position	400	3000	0	3300	600	0	200

- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.
- **Example**

  The example illustrates how to run MC_MasterEngine and the motion path.



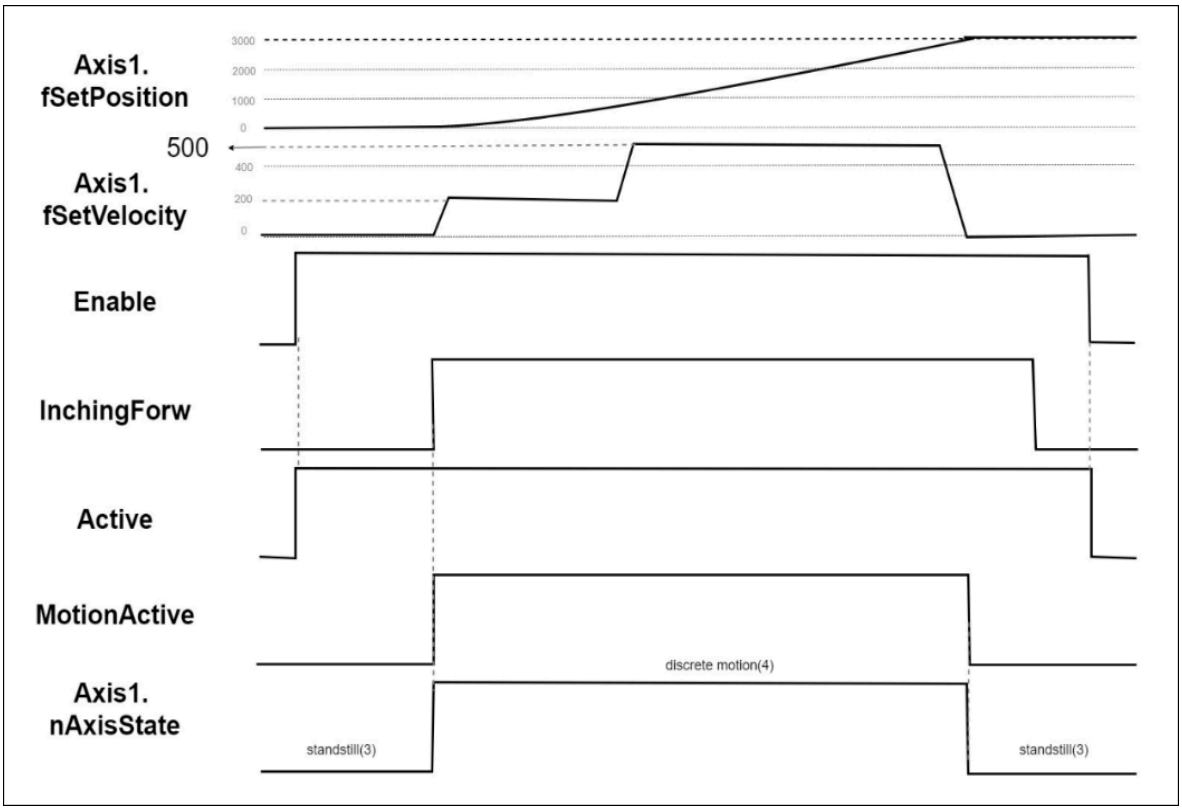
- Timing Diagram (uniform motion)



- When *Start* is TRUE, this function block moves in forward at a constant speed according to the *Velocity*, *Acceleration*, *Deceleration*, and *Jerk* set by the user. During the motion, *MotionActive* is TRUE, and the state machine changes Continuous motion.
- When the target velocity is reached, *InVelocity* is TRUE.

- When *Start* is TRUE, change the velocity to 2,000, and the axis speed will be adjusted to the new speed of 2,000 according to the *Acceleration*, *Deceleration*, and *Jerk* set by the user.
- When *Start* is FALSE, the axis will start to move towards the absolute position 2,000, and the state machine changes to Discrete motion during the movement. When the absolute position 2,000 is reached, *MotionActive* turns to FALSE.

◦ Timing Diagram (inching motion)



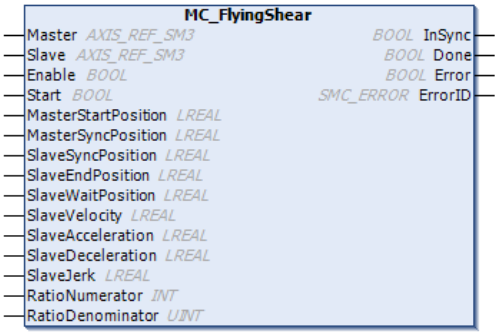
- When *InchingForw* is TRUE, this function block performs a forward inching movement according to the set *InchingVelocity*, *InchingAcceleration*, *InchingDeceleration*, and *InchingJerk*. During the motion, *MotionActive* is TRUE and the state machine is Discrete motion. When the inching distance (*InchingStep*) is reached, *MotionActive* becomes FALSE.
- When the target velocity is reached, *InVelocity* will not be TRUE because the motion state is Discrete motion.
- When *InchingForw* is TRUE, change the velocity to 500, and the axis speed will be adjusted to the new speed 500 according to the setting value of *InchingVelocity*, InchingAcceleration, *InchingDeceleration*, and *InchingJerk*.

- **Supported Devices**

◦ AX-series motion controller

2.5.1.3 MC_FlyingShear

MC_FlyingShear controls a slave axis to perform flying shear (synchronized cut-to-length) motion, where the slave axis tracks and cuts material in sync with a continuously moving master axis.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_FlyingShear		MC_FlyingShear _instance(Master :=, Slave :=, Enable :=, Start :=, MasterStartPosition :=, MasterSyncPosition :=, SlaveSyncPosition :=, SlaveEndPosition :=, SlaveWaitPosition :=, SlaveVelocity :=, SlaveAcceleration :=, SlaveDeceleration :=, SlaveJerk :=, RatioNumerator :=, RatioDenominator :=, InSync:=, Done => Error =>, ErrorID =>)

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	Run the instruction when *Enable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Start	Start the flying shear	BOOL	TRUE/Fals (FALSE)	Only when *Enable* = TRUE
MasterStartPosition	Master axis start position	LREAL	Positive number or 0 (0)	Only when *Enable* = TRUE and *Start* = TRUE
MasterSyncPosition	Master axis position where synchronized motion starts	LREAL	Positive number or 0 (0)	Only when *Enable* = TRUE and *Start* = TRUE
SlaveSyncPosition	Slave axis position where synchronized motion starts	LREAL	Positive number or 0 (0)	Only when *Enable* = TRUE and *Start* = TRUE
SlaveEndPosition	Slave position where synchronized motion ends	LREAL	Positive number or 0 (0)	Only when *Enable* = TRUE and *Start* = TRUE

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
SlaveWaitPosition	Slave position where slave axis waits	LREAL	Positive number or 0 (0)	Only when *Enable* = TRUE and *Start* = TRUE
SlaveVelocity	Slave velocity (user unit/s)	LREAL	Positive number (0)	Only when *Enable* = TRUE
SlaveAcceleration	Acceleration (user unit/s^2)	LREAL	Positive number (0)	Only when *Enable* = TRUE
SlaveDeceleration	Deceleration (user unit/s^2)	LREAL	Positive number (0)	Only when *Enable* = TRUE
SlaveJerk	Jerk (user unit/s^3)	LREAL	Positive number (0)	Only when *Enable* = TRUE
RatioNumerator	Velocity ratio when synchronized-Numerator	INT	Positive number (0)	Only when *Enable* = TRUE
RatioDenominator	Velocity ratio when synchronized-Denominator	UINT	Positive number (0)	Only when *Enable* = TRUE

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
InSync	TRUE when synchronized motion is in progress	BOOL	TRUE/FALSE (FALSE)
Done	TRUE when the slave axis has reached waiting position	BOOL	TRUE/FALSE (FALSE)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for descriptions.	SMC_ERROR*	SMC_ERROR (SMC_NO_ERROR)

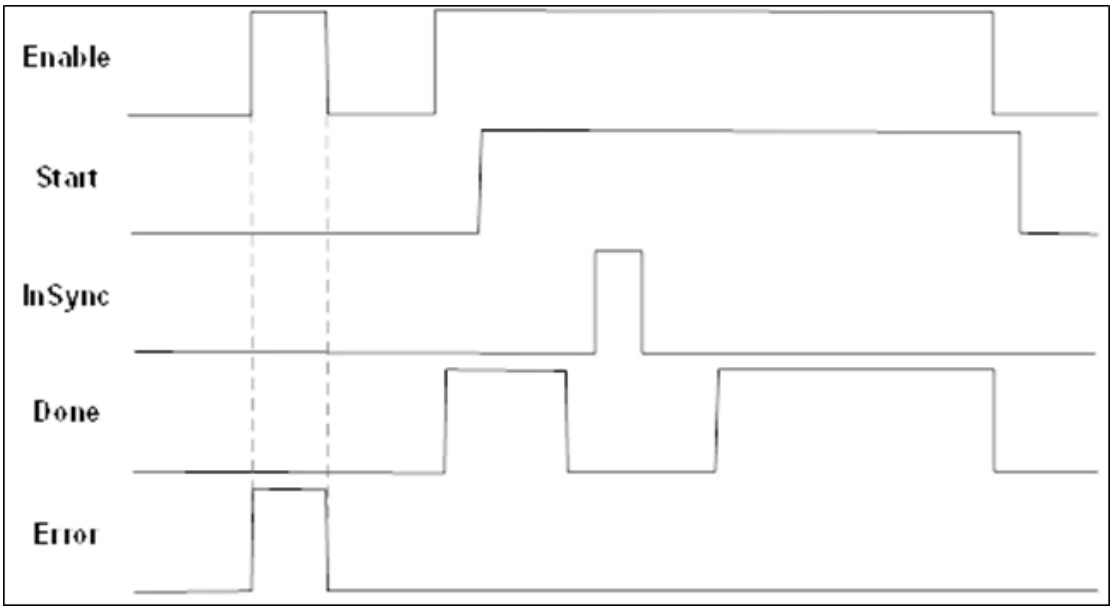
***Note**: SMC_ERROR: Enumeration (Enum)

- **Output Updating Time**

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
InSync	The synchronized motion is in progress.	• When *Error* is TRUE • When synchronization stops

Name	Timing for Shifting to TRUE	Timing for Shifting to FALSE
Done	The slave axis has reached waiting position.	*Enable* is TRUE and *Start*, *InchingForw*, or *InchingBackw* is FALSE
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Forward* or *Reverse* is FALSE (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value	Timing to Take Effect
Master	The specified master axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Enable* is TRUE.
Slave	The specified slave axis	AXIS_REF_SM3*	AXIS_REF_SM3	*Enable* is TRUE.

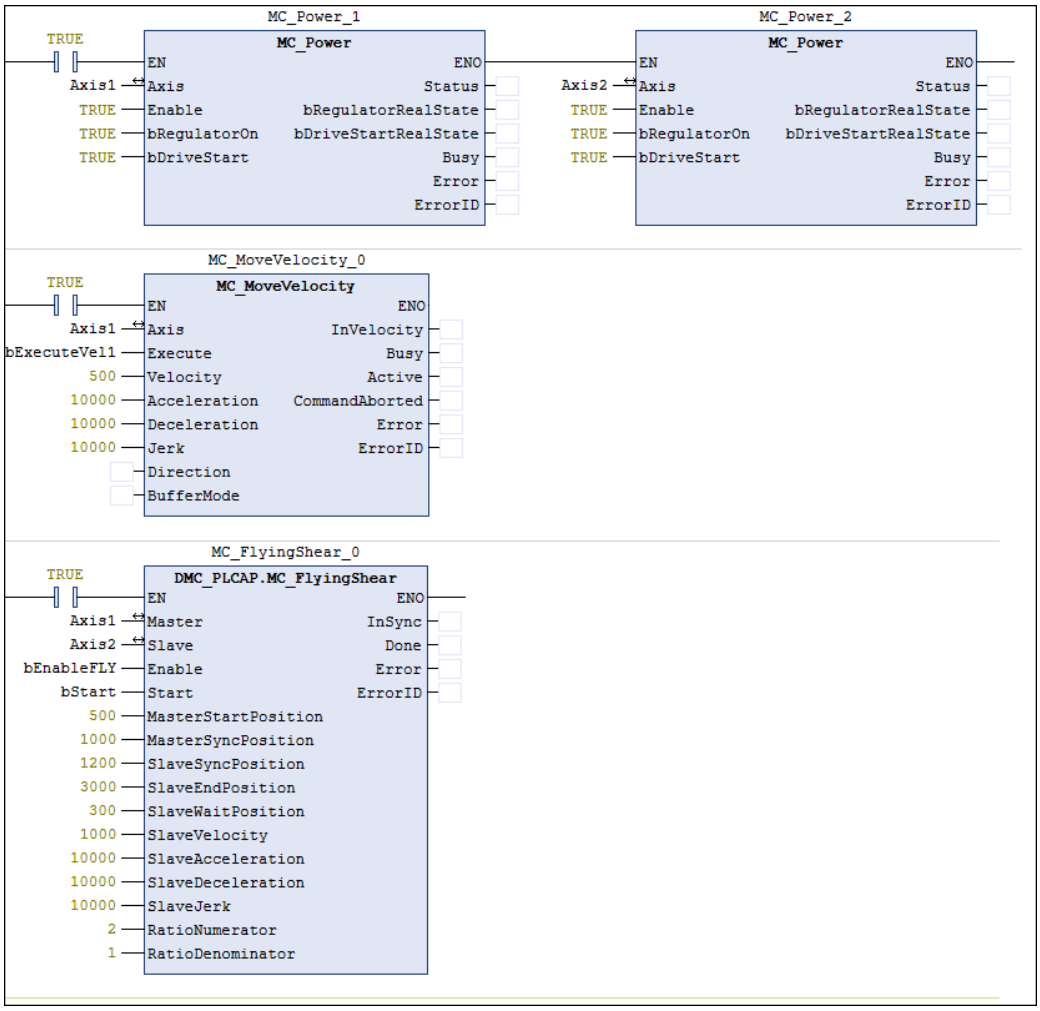
***Note**: AXIS_REF_SM3 (FB): Every function block contains this variable, which works as the starting program for the function block.

- **Function**
  - The master axis must be a rotary axis.
  - The parameter limits are as follows: *RatioDenominator* cannot be 0; *RatioNumerator* needs to be greater than 0;

    MasterStartPosition cannot be less than 0; *MasterStartPosition* cannot be greater than *MasterSyncPosition*
  - When *Enable* is at the rising edge, the slave axis will perform absolute positioning to *SlaveWaitPosition* according to the set *SlaveVelocity*, *SlaveAcceleration*, *SlaveDeceleration*, and *SlaveJerk*.
- **Troubleshooting**
  - If an error occurs while running the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

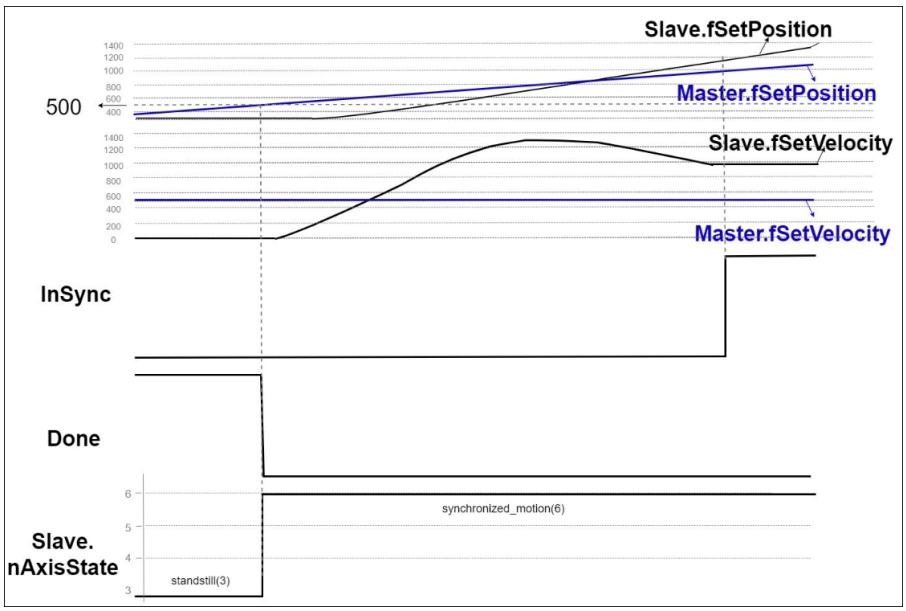
  The example illustrates how to run MC_FlyingShear and the motion path.



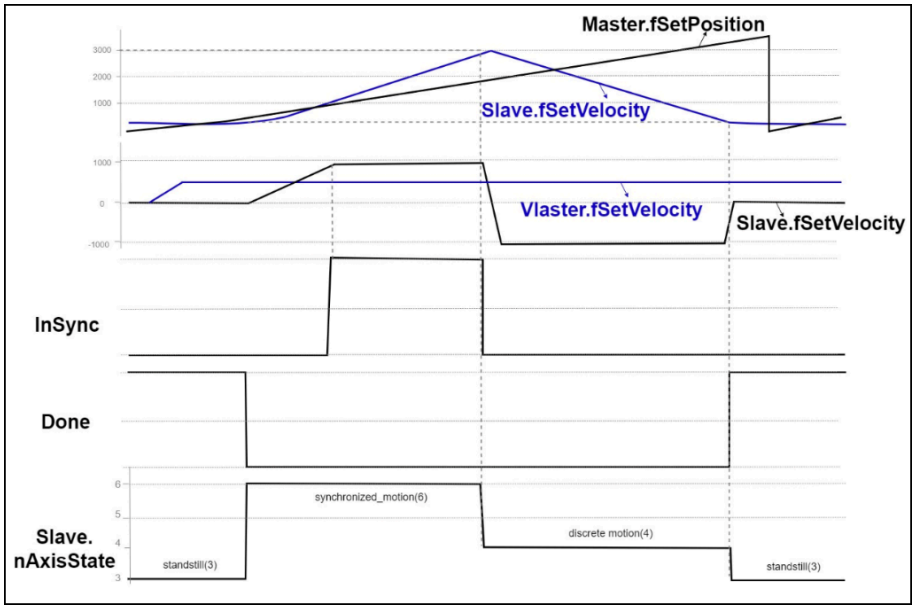
  - Timing Diagram

    - After *Enable* is TRUE, and the rising edge of *Start* is triggered, and when the master axis passes *MasterStartPosition*, the slave axis will start to follow the master axis, and the slave axis state will change to Synchronized Motion.

    - As shown in the figure below, the velocity starts to synchronize when the master axis passes *MasterSyncPosition* (1000). Slave position = *SlaveSyncPosition* (1200); Slave axis velocity = Master axis velocity * *RatioNumerator* / *RatioDenominator*; *InSync* = TRUE



- When the slave axis passes *SlaveEndPosition* (3000), the slave axis will return to *SlaveWaitPosition* (300), and the slave axis state will change from Synchronized Motion to Discrete Motion.



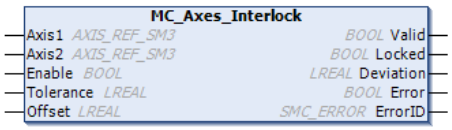
- When the master axis passes *MasterStartPosition*, and *Start* becomes TRUE again, the synchronization will be repeated.

- **Supported Devices**
  - AX-series motion controller

2.5.2 Administrative Motion Control Instructions

2.5.2.1 MC_Axes_Interlock

MC_Axes_Interlockfunction monitors two axes to ensure their positions remain within a specified tolerance. If the positional deviation exceeds the limit, an error will occur.

FB/FC	Instruction	Graphic Expression	ST Language
FB	MC_Axes_Interlock		MC_Axes_Interlock_instance(Axis1 :=, Axis2 :=, Enable :=, Tolerance :=, Offset :=, Valid =>, Locked =>, Deviation =>, Error =>, ErrorID =>)

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)	Timing to Take Effect
Enable	Run the instruction when *Enable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (FALSE)	-
Tolerance	The maximum position deviation tolerance	LREAL	Positive number or 0 (0)	*Enable* = TRUE
Offset	The offset to be ignored when comparing the position of two axes.	LREAL	Negative, positive number or 0 (0)	*Enable* = TRUE

Note: Position deviation of the two axes = Axis1 position − Offset − Axis2 position

- **Outputs**

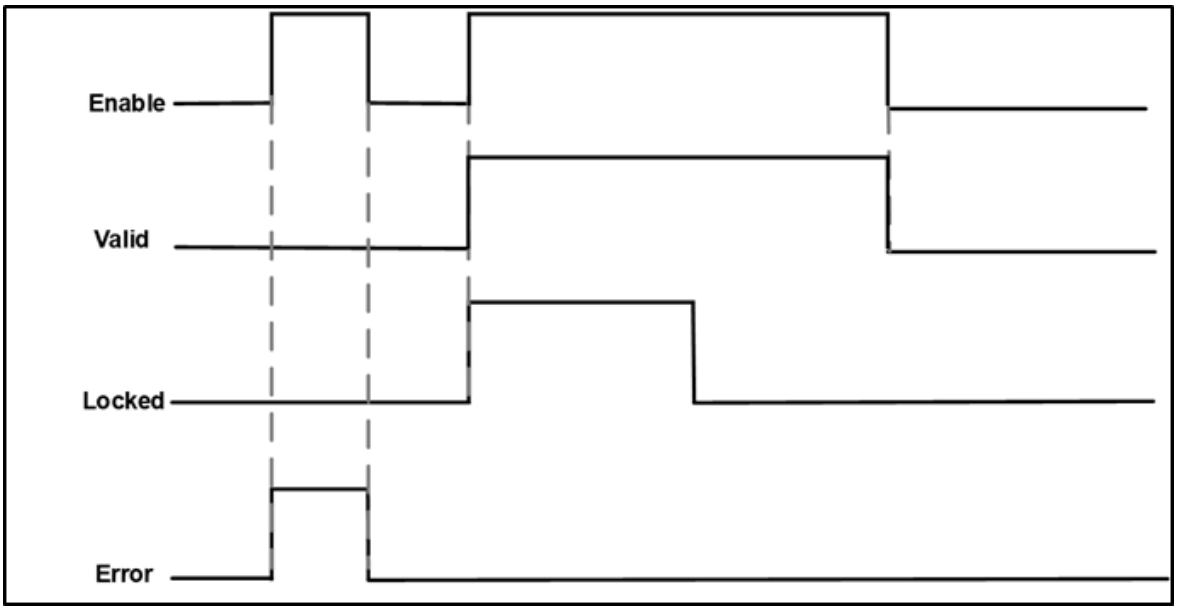
Name	Function	Data Type	Output Range (Default Value)
Valid	Indicates that the function block is enabled and there are no internal errors.	BOOL	TRUE/FALSE (FALSE)
Locked	Indicates that both axes are within the specified position tolerance and that neither axis has an error	BOOL	TRUE/FALSE (FALSE)
Deviation	The positional deviation between the two axes	LREAL	Negative, positive number or 0 (0)
Error	TRUE when an error occurs within the instruction	BOOL	TRUE/FALSE (FALSE)
ErrorID	Records the error code. Refer to **Appendix** for 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
Valid	*Enable* is TRUE.	*Enable* is FALSE.
Locked	Both axes are within the specified position *Tolerance* .	Both axes are out of the specified position *Tolerance* .
Deviation	*Enable* is TRUE.	When *Enable* is FALSE, stop updating
Error	An error occurs during the execution of the instruction or the input value of the instruction is incorrect.	*Enable* is FALSE. (Error Code is cleared.)
ErrorID		

- **Timing Diagram of Output Parameter Changes**



- **Inputs/Outputs**

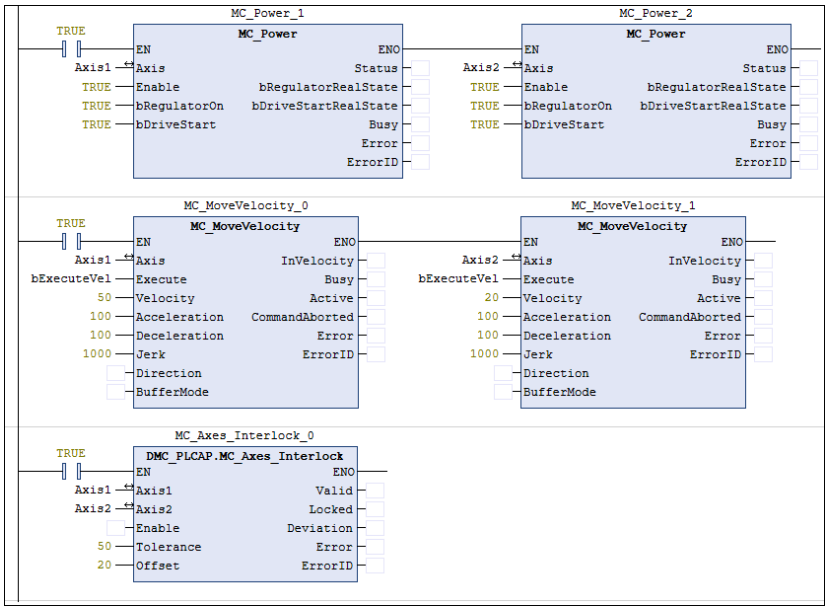
Name	Function	Data Type	Setting Value	Timing to Take Effect
Axis1	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	When *Forward* or *Reverse* is TRUE and *Busy* is FALSE
Axis2	The specified axis	AXIS_REF_SM3*	AXIS_REF_SM3	When *Forward* or *Reverse* is TRUE and *Busy* is FALSE

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

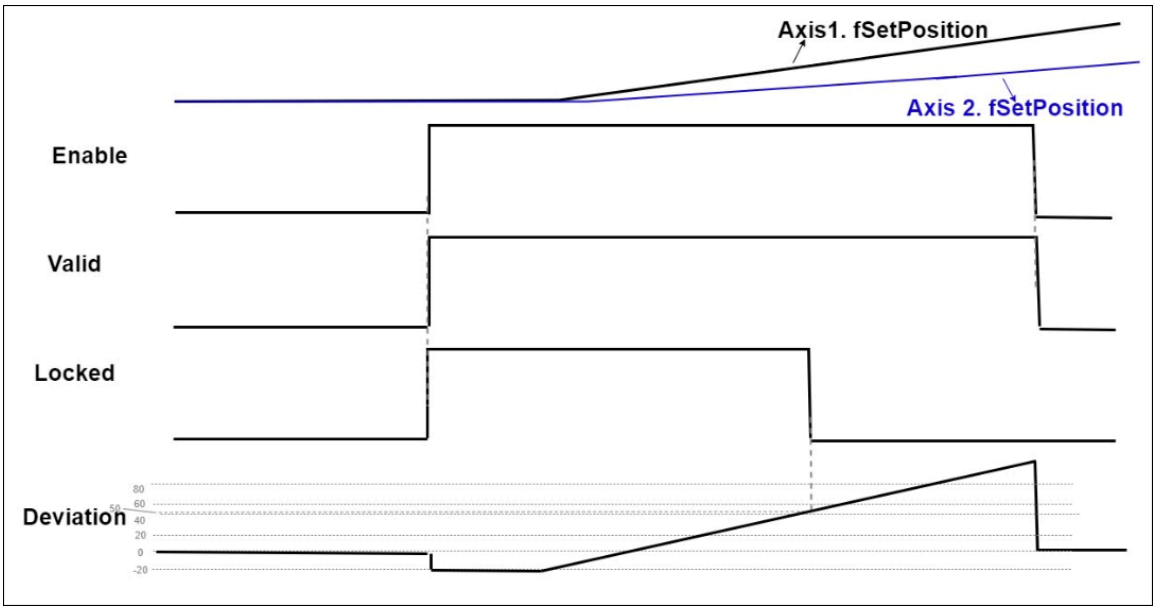
- **Function**
  - The type of Axis1 and Axis2 must be the same. (Both should be linear or rotary)
  - Calculate the position deviation when *Enable* = TRUE. If the absolute value of the deviation is less than or equal to *Tolerance*, *Locked* = TRUE.
  - Deviation = Axis1 Position − *Offset* − Axis2 Position
- **Troubleshooting**
  - If an error occurs during the running of the instruction, *Error* will change to TRUE. You can refer to *ErrorID* (Error Code) to check the problem.

- **Example**

  This example illustrates how to use MC_Axes_Interlock to monitor the deviation of the two axes.



  - Timing Diagram



    - When *Enable* is TRUE, this function block starts to monitor the deviation and *Valid* = TRUE.
    - When *Enable* is TRUE, the initial deviation of the two axes = 0 − 20 − 0 = −20 due to *Offset* = 20.
    - When *Enable* is TRUE, *Valid* = TRUE.

      If the deviation is < = 50, then *Locked* = TRUE; If >50, then *Locked* = FALSE.
    - When *Enable* is TRUE, *Deviation* will be updated.
    - When *Enable* is FALSE, *Valid* = FALSE.

- **Supported Devices**

  - AX-series motion controller

A. Appendix

A.1 Instruction Tables and Indexes

A.1.1 By Function

- **Single-axis Motion Control Instruction (Synchronous axis)**

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 motion 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.
	MC_JogToPosition	MC_JogToPosition controls the specified axis to move at the same speed in the specified direction.
	MC_MasterEngine	MC_MasterEngine controls the virtual master axis in packaging machine.
	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 motion of the target, the position where the interrupt occurs is used as the starting point to perform the positioning motion.
	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.

Categories	Name	Description
	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.
	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 engaging between master and slave axis.
	MC_GearIn	The function block MC_GearIn activates a linear master-slave engaging.
	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.
	MC_FlyingShear	MC_FlyingShear performs a defined synchronized motion between a continuously running master axis and a slave axis.
	DMC_MoveLink	DMC_MoveLink enables the slave axis to follow the master axis for synchronous positioning motion.
	DMC_CombineAxis	DMC_CombineAxis allows the slave axis to follow the master axie at a fixed proportion speed.
	DMC_GearIn	DMC_GearIn allows the slave axis to follow the master axis at a fixed proportion speed.
	DMC_GearOut	DMC_GearOut is used to detach the slave axis that moves synchronously with the master axis by DMC_GearIn or DMC_CombineAxis.
	DMC_GearInPos	DMC_GearInPos establishes a gear synchronization mechanism between the master and slave axis at the specified location.
	DMC_CamIn	DMC_CamIn allows the slave axis to follow the master axis based on the specified cam table.
	DMC_CamOut	DMC_CamOut is used to disengage the slave axis that uses DMC_CamIn to synchronize with the master axis.
	DMC_PhasingAbsolute	DMC_PhasingAbsolute controls the master axis phase compensation according to the specified absolute phase compensation value.

Categories	Name	Description
	DMC_PhasingRelative	DMC_PhasingRelative controls the master axis phase compensation according to the specified relative phase compensation value.
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.
	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.
	MC_Axes_Interlock	MC_Axes_Interlock monitors two axes which are mechanically linked to ensure that the position of both axes is within specified tolerances.
	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.

Categories	Name	Description
	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.
	DMC_GetCamSlaveData	Input the axis position for DMC_GetCamSlaveData to get information about the slave axis of the specified cam table.
	DMC_GetDeltaServoDriveError	DMC_GetDeltaServoDriveError can only read the current errors of Delta's servo panel.

- **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.

Categories	Name	Description
	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.
	DMC_MoveDirectAbsolute	DMC_MoveDirectAbsolute controls axis groups to move to the absolute position in the specified coordinate system. Each axis is calculated independently during the motion, and the motion path is not calculated.
	DMC_MoveDirectRelative	DMC_MoveDirectRelative controls axis groups to move to the relative position in the specified coordinate system. Each axis is calculated independently during the motion, and the motion path is not calculated.
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.

Categories	Name	Description
	DMC_GroupPower	DMC_GroupPower controls the enablement, shutdown and immediate stop of all axes in the axis group.
	DMC_GroupSetOverride	DMC_GroupSetOverride changes the velocity of the axis group motion by override control factor.

A.1.2 By Device

- The supported model types are listed as follows.

Function Block	Device Type			
	AX-3	AX-5	AX-C	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	●	●	●	●

Function Block	Device Type			
	AX-3	AX-5	AX-C	AX-8
MC_DigitalCamSwitch	●	●	●	●
MC_Home_DML	●	●	●	●
MC_Stop_DML	●	●	●	●
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	●	●	●	●
R2R_Configuration	●	●	●	●
R2R_RollDiameter	●	●	●	●
R2R_Run	●	●	●	●
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	●	●	●	●

Function Block	Device Type			
	AX-3	AX-5	AX-C	AX-8
DMC_GroupDisable	●	●	●	●
DMC_GroupReadParameter	●	●	●	●
DMC_GroupWriteParameter	●	●	●	●
DMC_GroupReadStatus	●	●	●	●
DMC_GroupReadError	●	●	●	●
DMC_GroupReset	●	●	●	●
DMC_GroupInterrupt	●	●	●	●
DMC_GroupContinue	●	●	●	●
DMC_CamReadTappetStatus	●	●	●	●
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_GroupPower	●	●	●	●
DMC_MoveDirectAbsolute	●	●	●	●
DMC_MoveDirectRelative	●	●	●	●
DMC_MoveModulo	●	●	●	●
DMC_Home_E	●	●	●	●
SMC_BacklashCompensation	●	●	●	●

Function Block	Device Type			
	AX-3	AX-5	AX-C	AX-8
MC_VelocityControl_DML	●	●	●	●
DMC_GroupSetOverride	●	●	●	●
DMC_GetCamSlaveData	●	●	●	●
DMC_GetDeltaServoDriveError	●	●	●	●
DMC_CombineAxis	●	●	●	●
DMC_GearIn	●	●	●	●
DMC_GearOut	●	●	●	●
DMC_GearInPos	●	●	●	●
DMC_CamIn	●	●	●	●
DMC_CamOut	●	●	●	●
DMC_PhasingAbsolute	●	●	●	●
DMC_PhasingRelative	●	●	●	●
MC_JogToPosition	●	●	●	●
MC_MasterEngine	●	●	●	●
MC_FlyingShear	●	●	●	●
MC_Axes_Interlock	●	●	●	●

A.1.3 By Letter

- A
  - MC_AbortTrigger
  - MC_AccelerationProfile
  - MC_Axes_Interlock
  - 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
  - DMC_CombineAxis
  - DMC_CamIn
  - DMC_CamOut
- D
  - MC_DigitalCamSwitch
- F
  - MC_FlyingShear
- 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_GroupPower
DMC_GroupSetOverride
DMC_GetCamSlaveData
DMC_GetDeltaServoDriveError
DMC_GearIn
DMC_GearOut
DMC_GearInPos

- 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_MasterEngine
  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_MoveDirectAbsolute

DMC_MoveDirectRelative

DMC_MoveModulo

- P

MC_Phasing

MC_PositionProfile

MC_Power

MC_Power_DML

DMC_PositionLag

DMC_PhasingAbsolute

DMC_PhasingRelative

- 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

R2R_Configuration

R2R_RollDiameter

R2R_Run

- S

MC_SetPosition

MC_SetOverride

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 Pin: 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 Pin: StartMode
SMC_CAMTAPPETTYPE	0: TAPPET_pos 1: TAPPET_all 2: TAPPET_neg	0: When pass in a positive direction 1: When pass in both directions 2: When pass in a negative direction	Function block: MC_CamIn.Tappets.pTaps Pin: ctt
SMC_CAM TAPPETACTION	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 Pin: cta
MC_TAPPETMODE	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 Pin: 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 Pin: BufferMode
DMC_GROUP_TRANSITION_MODE	0: None 10: Overlap 11: Single_Axis	0: The previous & next instructions follow BufferMode setting during	Function block: DMC_MoveLinearAbsolute DMC_MoveLinearRelative

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
		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. 11: The previous & next instructions follow BufferMode setting during blending,	DMC_MoveCircularAbsolute DMC_MoveCircularRelative DMC_MoveDirectAbsolute DMC_MoveDirectRelative Pin: TransitionMode
DMC_CIRC_PLANE	0: XY_plane 1: YZ_plane 2: ZX_plane 3: ARBITRARY_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. 3: The circle is drew in the space.	Function block: DMC_MoveCircularAbsolute DMC_MoveCircularRelative Pin: 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 Pin: CircMode
DMC_CIRC_PATHCHOICE	0: CLOCKWISE 1: COUNTER_CLOCKWISE	0: Clockwise 1: Counterclockwise	Function block: DMC_MoveCircularAbsolute DMC_MoveCircularRelative Pin: 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 Pin: 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 Pin: RampType
DMC_GROUP_PARAMETER	16: PARAM_RAMP_TYPE 17:PARAM_MAX_VELOCITY_LIMIT	16: Velocity ramp type 17: Max. velocity limit	Function block: DMC_GroupReadParameter

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
	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	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	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: TensionCtrlMod
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	Function Block: R2R_Configuration Pin: LineSpeedSource

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
		5: Pulse input via MI6 / MI7 terminals	
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
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	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	Function block: R2R_RollDiameter Pin: RollDiameterSource

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
	9: ThicknessIntegrate_Winding_CloseSW_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	
R2R_MATERIAL_THICKNESS_GAIN	0: millimeter 1: centimeter	0: mm unit 1: cm unit	Function block: R2R_RollDiameter Pin: MaterialThicknessGain
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
SMC_MOVEMENTTYPE	0: rotary 1: linear	1: Rotary axis 2: Linear axis	Function block: DMC_ ChangeMechanismGearRation Pin: AxisType
DMC_MotorDir	0: ReverseOFF 1: ReverseOn	1: Forward 2: Reverse	Function block:

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
			DMC_ ChangeMechanismGear Ration Pin: MotorDirection
DMC_ DISABLEMODE	0: ZERO_TORQUE 1: CURRENT_TORQUE	1: Clear the torque command 2: Retain the torque command	Function block: DMC_TorqueControl

- The Data Types listed below are Structure type.

Data Type	Function Block	Definition
AXIS_REF_SM3*	Applied to the MC_ / DMC_ function block.	This structure contains all the required data and parameters for axis motion.
DMC_AXIS_GROUP_REF*	Applied to the DMC_Group function block.	This structure contains all the required data and parameters for group motion.
AXIS_REF_VIRTUAL_SM3	Applied to the 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 Operation Manual* for more details of structural type parameters.

A.3 Error Codes and Troubleshooting

A.3.1 For Synchronous Axis

When an error occurs, you can troubleshoot errors through error codes and the corresponding indicators. Refer to *AX-3 Series Operation 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 run 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 run MC_Reset.
0x00003	SMC_DI_FIELDBUS_LOST_SYNCRONICITY	Loss of synchronicity	1. Run SMC3_ReinitDrive. If the error occurs frequently, refer to Task max cycle time and increase EtherCAT DC time. 2. Check whether the EtherCAT network cable is damaged.
0x0000A	SMC_DI_SWLIMITS_EXCEEDED	Software limit errors	Run MC_Reset and run reversely away from the limit.
0x0000B	SMC_DI_HWLIMITS_EXCEEDED	Hardware limit errors	Run MC_Reset.
0x0000C	SMC_DI_LINEAR_AXIS_OUTOFRANGE	Incremental position of a linear axis is out of range	Run MC_Reset
0x0000D	SMC_DI_HALT_OR_QUICKSTOP_NOT_SUPPORTED	Not support Halt or Quickstop state of drivers	Run MC_Reset
0x00010	SMC_DI_POSITIONLAGERROR	Excessive position error	Run MC_Reset
0x00011	SMC_DI_HOMING_ERROR	Homing error occurs	Run MC_Reset
0x00014	SMC_REGULATOR_OR_START_NOT_SET	The motion FB cannot be run under the current axis state.	Turn on the servo and run MC_Reset, then Re-run the motion FB.
0x00015	SMC_WRONG_CONTROLLER_MODE	The axis is under wrong controller mode.	Run 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.
0x0001B	SMC_AXIS_IS_NOT_OPERATIONAL	MC_Power is not used for the axes.	Confirm the order in which the instructions are run.

Error code	Description	Contents	Corrective Action
0x0001E	SMC_FB_WASNT_CALLED_DURING_MOTION	Function blocks cannot be called in motion state	Run 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.	Run MC_Reset and do not change the axis input of the function block.
0x00021	SMC_FB_ACTIVE_AXIS_DISABLED	Run servo off while axis is in motion.	Power on the servo and run MC_Reset.
0x00022	SMC_AXIS_NOT_READY_FOR_MOTION	The motion instruction cannot be run under the current axis state.	Axis is in an uncontrolled state. Confirm whether it is powered on or with errors. Start the axis or run MC_Reset_DML 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 run 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. 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.

Error code	Description	Contents	Corrective Action
0x00050	SMC_RAG_ERROR_DURING_STARTUP	Error occurs when the axis group is activated.	Make sure the configuration is normal and re-run 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, must be within 8–32 bits.	The value input of *usiNumberOfAbsoluteBits* of SMC3_PersistPositionSingleturn is incorrect. Enter a valid 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.
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 run 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_CHANN

Error code	Description	Contents	Corrective Action
			ELS 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-run the FB after the axis enters standstill state.
0x000AB	SMC_H_AXIS_DIDNT_START_HOMING	Errors occur while homing.	Before running 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 running 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-run the FB after the axis leaving ErrorStop state.
0x000B5	SMC_MS_INVALID_ACCDEC_VALUES	Invalid setting value of velocity or acceleration	Re-run the FB after correcting the input value of *Deceleration*.
0x000B7	SMC_MS_AXIS_IN_ERRORSTOP	Drive in operating state Errorstop.	Re-run the FB after the axis leaving ErrorStop state.
0x000B8	SMC_BLOCKING_MC_STOP_WASNT_CALLED	The axis is locked and MC_Stop cannot be called.	Check the inputs of MC_Stop and run the FB again.
0x000B9	SMC_MS_AXIS_ALREADY_STOPPING	A stop cannot be aborted while executing MC_Stop.	Re-run the FB after the axis leaving Stopping state.
0x000C9	SMC_MA_INVALID_VELACC_VALUES	Invalid values of *Velocity* or *Acceleration*	Re-run 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-run the FB after correcting the input values of *Velocity*, *Acceleration*, *Deceleration*, and *Jerk*.

Error code	Description	Contents	Corrective Action
0x000FB	SMC_MAD_INVALID_VELACC_VALUES	Invalid values of *Velocity* or *Acceleration*	Re-run 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-run 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-run the FB after the running of first MC_MoveSuperimposed is completed.
0x0012D	SMC_MV_INVALID_ACCDEC_VALUES	Invalid value of *Velocity* or *Acceleration*	Re-run 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	Enter the correct *ArraySize*, then re-run the function blocks.
0x00146	SMC_PP_STEP0MS	Delta_time is not allowed to be set to 0.	Enter the correct Delta_Time, then re-run the function blocks.
0x0015E	SMC_VP_ARRAYSIZE	Incorrect array size	Enter the correct *ArraySize*, then re-run the function blocks.
0x0015F	SMC_VP_STEP0MS	Delta_time is not allowed to be set to 0.	Enter the correct Delta_Time, then re-run the function blocks.
0x00177	SMC_AP_ARRAYSIZE	Incorrect array size	Enter the correct *ArraySize*, then re-run the function blocks.
0x00178	SMC_AP_STEP0MS	Delta_time is not allowed to be set to 0.	Enter the correct Delta_Time, then re-run 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 and re-run the function block.
0x0019A	SMC_AT_TRIGGERNOTOCCUPIED	Triggering has been reset.	Check whether MC_TouchProbe is running and the axis position has not been captured and 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*,

Error code	Description	Contents	Corrective Action
			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.
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_ChangeDynamicLimits can only be called in standstill or power_off state.	Check whether 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 fMax*Jerk*, 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. Run 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.

Error code	Description	Contents	Corrective Action
0x00273	SMC_CI_RAMPIN_NEEDS_VELACC_VALUES	The value of *Velocity* and acceleration must be set in the function of ramp_in.	1. Run 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*, or *ramp_in_n*eg, the input values of *VelocityDiff*, *Acceleration*, and *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. Run 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 minimum values in *fEditor* and *fTable* while using a cam table not in "XYVA" format. Then you can re-run the function block.
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. Run 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.

Error code	Description	Contents	Corrective Action
0x002A5	SMC_GI_INVALID_DEC	Invalid value of deceleration	1. Run 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 (Turn on/Disable) is changed without the permission.	Run 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. Run 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.	Run MC_Reset to make the axis back to standstill state and Re-run 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.
0x0079E	SMC_INVALID_PARAMETER	1. The MovementType of the two axes is inconsistent. 2. If both axes are rotary axes, the rotation period of the two axes is inconsistent. 3. *Tolerance* or *InchingStep* is negative.	Check the parameter settings of the instruction.
0x01771	SMC_INVALID_AXIS_TYPE	The axis must be rotary.	Check the axis settings.
0x186A0	DMC_TPC_INVALID_PDO_MAPPING	PDO mapping error	Do not configure Touch probe function (60B8h) in PDO.
0x186A1	DMC_TPC_TRIGGEROCCUPIED	Trigger has been created.	Do not run 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-run the FB after modification.
0x186A5	DMC_TPC_ETC_CO_TIMEOUT	Communication error	SDO time outs. Check whether 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.
0x186A9	DMC_TPC_SLAVE_NOT_SUPOORT	In CN5 mode, ETCSlave is not the ASDA-A3	Confirm the slave model.

Error code	Description	Contents	Corrective Action
0x186B3	DMC_MF_INVALID_ACCDEC_VALUES	Invalid velocity or acceleration value	After correcting the *Velocity*, *Acceleration*, *Deceleration*, input values, run the function block again.
0x186B4	DMC_MF_AXIS_NOT_READY_FOR_MOTION	Axis status cannot run motion control instruction	The axis is in a state that cannot be controlled. Confirm whether the target axis is powered on or in an error state, and enable the axis or run MC_Reset on the axis.
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 run MC_Reset.
0x186B6	DMC_MF_REGULATOR_OR_START_NOT_SET	Axis status cannot run motion control instruction	After starting the servo, run MC_Reset and Re-run 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_WINDOW	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_INVALID_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 negatives; restart the function block after correction.
0x186C5	DMC_ML_AXIS_NOT_READY_FOR_MOTION	Axis status cannot run motion control instruction	The axis is in a state that cannot be controlled. Confirm whether the target axis is powered on or in an error state, and enable the axis or run MC_Reset on the axis.
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 run MC_Reset.
0x186C7	DMC_ML_REGULATOR_OR_START_NOT_SET	Axis status cannot run motion control instruction	After starting the servo, run MC_Reset and Re-run the motion function block.

Error code	Description	Contents	Corrective Action
0x186C8	DMC_ML_TP_TRIGGEROCCUPIED	There are other MC_TouchProbes executing	After making sure that no other MC_TouchProbe is running 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 whether the Cam table has no data.
0x186D5	DMC_CB_CAM_DATATYPE_NOT_SUPPORT	Cam table format error	Check whether 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-run 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-run the FB after correcting Track ID.
0x187D3	DMC_CRTV_NO_TAPPETID	The track ID to read does not exist.	Re-run the FB after checking the tappet inputs.
0x187D4	DMC_CRTV_NO_TAPPETS_IN_CAM	There’s no tappets set in the cam table.	Re-run the FB after adding new tappets.
0x187DA	DMC_CWTV_INVALID_TAPPETID	Invalid Track ID	Re-run the FB after correcting Track ID.
0x187DB	DMC_CWTV_INVALID_MASTER_POS	Invalid master position	Correct the input of master position, then Re-run 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-run the FB after correcting Track ID.
0x187DE	DMC_CWTV_TAPPET_NUM_EXCEED_LIMIT	The number of tappets exceeds the limit.	Re-run 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-run the FB.
0x187E4	DMC_CAT_INVALID_MASTER_POS	The user-defined master position is out of range.	Re-run 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-run 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 set to

Error code	Description	Contents	Corrective Action
			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-run the FB.
0x187ED	DMC_CDT_NO_TAPPETS_IN_CAM	There's no tappet in the tappet table.	Re-run 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.
0x187EF	DMC_CDT_TAPPETID_NOT_FOUND	The Track ID for deletion still exists in the Tappet table	Correct the Track ID and then restart the function block
0x187F4	DMC_CRP_INVALID_POINTNUM	Invalid point number	Check whether the point number of specified data is more than the point number of cam data. Re-run the FB after modification.
0x187FA	DMC_CWP_INVALID_POINTNUM	Invalid point number	Check whether the point number of specified data is more than the point number of cam data. Re-run the FB after modification.
0x187FB	DMC_CWP_INVALID_MASTERPOS	Invalid master position	1. Check whether the master position of data point to be modified exceeds the master position of the front and back point. Re-run the FB after modification. 2. If you want to modify the cam table in bulk, it is recommended to start modifying from the last point of the master axis to avoid 0x187FB error codes. (For example, the point to be modified in the cam table is 200, then you need to modify it from point 200 back to point 1)
0x18801	DMC_TC_INVALID_VALUES	Invalid value	Confirm pin input parameter value. Re-run 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.

Error code	Description	Contents	Corrective Action
0x18803	DMC_TC_SDO_RW_FAIL	Wrong communication	SDO read & write failed. Reply to the servo communication, and run 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 running 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 run motion control instruction.	After starting servo, run MC_Reset, and Re-run motion function block.
0x18809	DMC_TC_INVALID_PDO_MAPPING	Slave does not configure the related OD on PDO.	1. Confirm the PDO configuration. 2. Confirm PLC is not in the simulation mode.
0x1880A	DMC_TC_TORQUE_RAMP_VALUE_RANG_EXCEEDED	The Torque Ramp input is out of range.	Confirm the input parameters of the function block.
0x1880B	DMC_TC_VELOCITY_VALUE_RANG_EXCEEDED	The Max Profile *Velocity* input is out of range.	Confirm the input parameters of the function block.
0x1880C	DMC_TC_WRONG_AXIS_TYPE	Wrong axis type	Confirm that the function block axis is the EtherCAT axis.
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 running	Re-run the function block.
0x18814	DMC_VC_INVALID_ACCDEC_VALUES	Wrong value	Confirm pin input parameter value. Re-run the FB after modification.
0x18815	DMC_VC_DIRECTION_NOT_APPLICABLE	Wrong value	Confirm pin input parameter value. Re-run the FB after modification.
0x18816	DMC_VC_AXIS_NOT_READY_FOR_MOTION	Wrong axis state	Power on servo, and re-run the function block.

Error code	Description	Contents	Corrective Action
0x18817	DMC_VC_AXIS_ERROR_DURING_MOTION	Axis error	Confirm servo error information. Refer to Servo manual for error elimination, and run MC_Reset.
0x18818	DMC_VC_REGULATOR_OR_START_NOT_SET	Axis error	Power on servo, run MC_Rest, and Re-run motion function block.
0x18819	DMC_VC_WRONG_CONTROLLER_MODE	Axis is in the wrong controller mode.	Function block does not support running in the current mode. To run this function block, run 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.
0x1881F	DMC_CMGR_RAG_ERROR_DURING_STARTUP	Communication error	Confirm if the bus configuration is normal, and Re-run DMC_ChangeMechanismGearRation.
0x18820	DMC_CMGR_RAG_ERROR_AXIS_NOT_INITIALIZED	Axis initializing	EtherCAT Master cannot run DMC_ChangeMechanismGearRation during Initialization.
0x1882E	DMC_GM_NO_ERROR_TO_RESET	There’s no error to be reset.	Re-run 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 run the reset action.	After the communication status of the axis is back to normal, Re-run 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

Error code	Description	Contents	Corrective Action
			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-run 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-run 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.	Run 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.
0x1883A	DMC_GM_DIRECT_AXIS_MAPPING_ERROR	A non-0 displacement is specified for a non-existent axis in the Direct motion instruction.	Run MC_GroupReset to return the axis group to GroupStandby state. And check the parameter setting of the axis group and the position of the axis group motion instruction, and confirm that each axis in the axis group with displacement has the correct designated single axis.
0x1883B	DMC_GM_JOG_AXIS_MAPPING_ERROR	A non-0 displacement is specified for an axis that does not exist in the jog motion instruction.	Run MC_GroupReset to return the axis group to GroupStandby state. And check the parameter setting of the axis group and the position of the axis group motion instruction, and confirm that each axis in the axis group with displacement 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 motion.	Run 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

Error code	Description	Contents	Corrective Action
			displacement has been mapped to an appointed axis.
0x18840	DMC_GM_HELIX_AXIS_MAPPING_ERROR	Command a non-zero displacement to an axis, which does not exist, in a helical motion.	Run 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. Run 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. Run 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. Run 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.

Error code	Description	Contents	Corrective Action
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. Run 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 (DMC_AXIS_GROUP_REF.bContinueDataWriten), then run 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, run 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. Run 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

Error code	Description	Contents	Corrective Action
			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. Run 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. Run 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 motion is not supported.	1. Run 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.
0x1885C	DMC_GM_AXIS_GROUP_VELOCITY_EXCEED_LIMIT	The velocity of axis group exceeds the limit set in the parameter setting.	1. Run MC_GroupReset to make the group state back to GroupStandby. 2. Make sure the group velocity does not exceed

Error code	Description	Contents	Corrective Action
			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. Run 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. Run 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. Run 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. Run 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. Run MC_GroupReset to make the group state back to GroupStandby. 2. Make sure the parameters set for the motion instruction are

Error code	Description	Contents	Corrective Action
			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.
0x18863	DMC_GM_INVALID_COORD_SYSTEM	This motion instruction does not support this coordinate system.	1. Change to the supported coordinate system. 2. Re-run the function block.
0x18864	DMC_GM_KIN_INVALID_PARAMETERS	The kinematics parameters of the axis group are not set correctly.	1. Run MC_GroupReset to make the group state back to GroupStandby. 2. Confirm the kinematics parameters. 3. Re-run the function block.
0x18865	DMC_GM_KIN_INVALID_CONSTELLATION	The cartesian coordinate of the points in the axis group motion path exceed the working area of the axis group.	1. Run MC_GroupReset to make the group state back to GroupStandby. 2. Make sure the cartesian coordinate of the points in the axis group motion path does not exceed the working area of the axis group. 3. Re-run the function block.
0x18866	DMC_GM_KIN_NOT_INITIALIZED	The axis group does not set kinematics transformation.	1. Run MC_GroupReset to make the group state back to GroupStandby. 2. Make sure the axis group has set kinematics transformation. 3. Re-run the function block.
0x18867	DMC_GM_KIN_CONFIGS_DIFFER	The kinematics configuration of all points in the motion path of the axis group is inconsistent.	1. Run MC_GroupReset to make the group state back to GroupStandby. 2. Make sure that the kinematics configuration of all points in the motion

Error code	Description	Contents	Corrective Action
			path of the axis group should be consistent. 3. Re-run the function block.
0x18868	DMC_GM_KIN_SINGULAR_CONFIGURATION	Set the kinematics configuration as singular configuration.	1. Make sure that the kinematics configurations are set correctly. 2. Re-run the function block.
0x18869	DMC_GM_DYN_TRACKING_MUTUAL_DEPENDENCY	The axis group and the axis group it is tracking cannot form a loop.	1. Make sure the axis group and the axis group it is tracking form a loop. 2. Re-run the function block.
0x1886A	DMC_GM_DYN_TRACKING_DEPENDENCY_IN_DIFFERENT_TASK	The axis group is not under the same task as the axis group it is tracking.	1. Make sure the axis group is under the same task as the axis group it is tracking. 2. Re-run the function block.
0x1886B	DMC_GM_DYN_TRACKING_PCS_STILL_IN_USE	While dynamic tracking is in progress, the PCS used cannot be modified.	1. Stop tracking or wait for tracking to complete. 2. Re-run the function block.
0x1886C	DMC_GM_DYN_TRACKING_INVALID_BUFFER_MODE	Dynamic tracking does not support this BufferMode.	1. Make sure the Buffer Mode used is supported by tracking. 2. Re-run the function block.
0x1886D	DMC_GM_DYN_TRACKING_OPERATION_NOT_SUPPORTED	Dynamic tracking does not support this operation.	Dynamic tracking does not support this operation.
0x1886E	DMC_GM_INVALID_INPUT	The value of the function block input parameter is invalid.	1. Make sure that the values of the function block input parameters are valid. 2. Re-run the function block.
0x1886F	DMC_GM_INVALID_DYNAMIC_FACTOR	Invalid velocity/acceleration/jerk factor values.	1. Make sure that the value of the Factor parameter is valid. 2. Re-run the function block.

Error code	Description	Contents	Corrective Action
0x18870	DMC_GM_INVALID_DYNLIMITS	Invalid velocity/acceleration/deceleration/jerk values	1. Make sure that the values of velocity/acceleration/deceleration/jerk are valid. 2. Re-run the function block.
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 turned on axis group.	1. Make sure the specified axis does not exist in other turned on 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.

Error code	Description	Contents	Corrective Action
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.
0x18887	DMC_GM_KINEMATICS_AXIS_MAPPING_ERROR	The axis mapping settings for the axis group do not correspond to the axis mapping settings required by the configuration.	1. Adjust the axis mapping settings for the axis group. 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 turned on.	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.
0x188B6	DMC_CKPW_INVALID_MASTERPOS	Invalid master position	Re-run the FB after correcting the input of master position.
0x188B7	DMC_CKPW_INVALID_ ACC	Invalid acceleration	Re-run the FB after correcting the acceleration input value of master position.
0x188B8	DMC_CKPW_INVALID_ACC_SETTING	Invalid acceleration setting	Re-run the FB after determining the velocity, acceleration and curve type.

Error code	Description	Contents	Corrective Action
0x188B9	DMC_CKPW_INVALID_CURVE_TYPE_SETTING	Invalid curve type setting	The input curve type is not supported. Re-run 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-run the FB.
0x188BB	DMC_CKPW_CAM_IS_WRITING_BY_OTHER_FUNCTION	Failure to write CAM.	Check whether the cam table you're currently using is being written by other FBs, then wait for the writing completed before you re-run the FB.
0x188C5	DMC_HP_INVALID_HOME_SPEED	Invalid home speed value	Set **Homing speed during search for switch** and **Homing speed during search for z phase pulse** to a non-zero value on the **PoAxis Configuration** page.
0x188C6	DMC_HP_INVALID_HOME_ACC_DEC	Invalid home acceleration or deceleration value	Set **Homing Acceleration** to a non-zero value on on the **PoAxis Configuration** page or stop the axis immediately.
0x188C7	DMC_HP_INVALID_HOME_POSITION	Invalid value of home position	Set *IrPosiotion* to be in the rotary range of pulse axis. [0–PulseAxis.ModuloValue]
0x188C8	DMC_HP_AXIS_NOT_PULSEAXIS	The input variable type is not set to be PulseAxis_REF.	After selecting **Pulse Axis X** in **Hardware IO Configuration**, enter the **IEC Object** variable to the input parameter Axis of the function block DMC_Home_P.
0x188C9	DMC_HP_HOMING_METHOD_RESERVED	Homing method is not supported by current version.	Check whether the homing method is supported by the version you're currently using. refer to the specification document for changing the mode.
0x188CA	DMC_HP_HOMING_MOVEMENT_HW_LIMIT	Positive or negative limit signal is activated and axis cannot perform homing in this circumstances.	Check whether the hardware limit signals you're using are 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_NOT_STANDSTILL	Axis state is not Standstill.	Confirm that DMC_Home_P is running when the axis state is Standstill.

Error code	Description	Contents	Corrective Action
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.
0x188DF	DMC_GI_RATIO_DENOM	Wrong input parameter	Gear ratio denominator cannot be 0.
0x188E0	DMC_GI_INVALID_ACC	Wrong input parameter	Set valid acceleration.
0x188E1	DMC_GI_INVALID_DEC	Wrong input parameter	Set valid deceleration.
0x188E3	DMC_GI_INVALID_JERK	Wrong input parameter	Set valid jerk.
0x188E6	DMC_GI_MASTER_VALUE_SOURCE_NOT_EXIST	Wrong input parameter	Set valid master source.
0x188E7	DMC_GI_MASTER_AND_SLAVE_ARE_THE_SAME	Wrong axis	The slave and master axis should not be the same.
0x188F4	DMC_CA_INVALID_ACC	Wrong input parameter	Set valid acceleration.
0x188F5	DMC_CA_INVALID_DEC	Wrong input parameter	Set valid deceleration.
0x188F7	DMC_CA_INVALID_JERK	Wrong input parameter	Set valid jerk.
0x188FA	DMC_CA_MASTER_VALUE_SOURCE_NOT_EXIST	Wrong axis	Set valid master source.
0x188FB	DMC_CA_COMBINE_MODE_SETTING_ERROR	Wrong mode set	Set the existed combination mode.
0x188FC	DMC_CA_MASTER1_AND_SLAVE_ARE_THE_SAME	Wrong axis	Set the master axis 1 different from the slave axis.
0x188FD	DMC_CA_MASTER2_AND_SLAVE_ARE_THE_SAME	Wrong axis	Set the master axis 2 different from the slave axis.
0x1891B	DMC_CI_NO_CAM_SELECTED	Wrong CamTable	Set valid CamTable.
0x1891D	DMC_CI_RAMPIN_NEEDS_VELACC_VALUES	Wrong input parameter	Set valid velocity.
0x1891E	DMC_CI_SCALING_INCORRECT	Wrong input parameter	Set valid scaling.
0x18922	DMC_CI_MASTER_VALUE_SOURCE_NOT_EXIST	Wrong axis	Set valid master source.
0x18923	DMC_CI_ACTIVATION_MODE_SETTING_ERROR	Wrong Activation mode	Set valid Activation mode.
0x18924	DMC_CI_INVALID_ACC	Wrong input parameter	Set valid acceleration.
0x18925	DMC_CI_INVALID_DEC	Wrong input parameter	Set valid deceleration.
0x18926	DMC_CI_INVALID_JERK	Wrong input parameter	Set valid jerk.
0x18927	DMC_CI_MASTER_AND_SLAVE_ARE_THE_SAME	Wrong axis	The slave and master axis should not be the same.
0x18929	DMC_CI_ACTIVATION_POS_OUT_OF_RANG	Wrong input parameter	Set valid Activation mode.

Error code	Description	Contents	Corrective Action
0x18943	DMC_PA_INVALID_VELACC_VALUES	Wrong input parameter	Set valid velocity.
0x18947	DMC_PA_INVALID_ACC	Wrong input parameter	Set valid acceleration.
0x18948	DMC_PA_INVALID_DEC	Wrong input parameter	Set valid deceleration.
0x18949	DMC_PA_INVALID_JERK	Wrong input parameter	Set valid jerk.
0x1894A	DMC_PA_MASTER_AND_SLAVE_ARE_THE_SAME	Wrong axis	The slave and master axis should not be the same.
0x18957	DMC_PR_INVALID_VELACC_VALUES	Wrong input parameter	Set valid velocity.
0x18959	DMC_PR_SYNC_MOTIOM_RUNTIME_NOT_READY	Does not support this method	Does not support Codesys MC_CamIn and MC_GearIn
0x1895B	DMC_PR_INVALID_ACC	Wrong input parameter	Set valid acceleration.
0x1895C	DMC_PR_INVALID_DEC	Wrong input parameter	Set valid deceleration.
0x1895D	DMC_PR_INVALID_JERK	Wrong input parameter	Set valid jerk.
0x1895E	DMC_PR_MASTER_AND_SLAVE_ARE_THE_SAME	Wrong axis	The slave and master axis should not be the same.
0x1896C	DMC_STL_WP_PARAM_INVALID	Invalid parameter	The input parameter is too large. Re-run 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 whether 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 whether the servo you’re currently using meets Cia402, otherwise the function block cannot run.
0x1896F	DMC_STL_WP_PARAM_CONVERSION_ERROR	Parameter conversion error	No such error should occur while matching ASDA-A2-E to use. Check whether the servo you’re currently using meets Cia402, otherwise the function block cannot 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-run 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-run the FB.

Error code	Description	Contents	Corrective Action
0x1898A	DMC_PL_INVALID_POSITIONLAG	Invalid MaxPositionLag input	The input value of *fMaxPositionLag* is negative, correct the value before re-run the FB.
0x1898B	DMC_PL_INVALID_LAGCYCIES	Invalid SetActTimeLagCycles input	The input value of *SetActTimeLagCycles* is negative. Correct the value before re-run the FB.
0x18996	DMC_MVSBP_INVALID_DIRECTION	Invalid direction	Only positive and negative direction are allowed, correct the direction of motion before re-run the FB.
0x18997	DMC_MVSBP_INVALID_PHASE	Invalid phase input.	RoundPhase/StopPhase input error. Correct the input parameters before re-run the FB.
0x18998	DMC_MVSBP_AXIS_NOT_READY_FOR_MOTION	Slave axis is not ready for motion.	The slave is not under control. Check whether the target axis is powered on or in error, then enable the axis or run 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 run 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, run MC_Reset before re-run the FB.
0x1899B	DMC_MVSBP_INVALID_ACCDEC_VALUES	Invalid velocity, acceleration, deceleration, and jerk	After correcting the parameter, re-run the function block.
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. Turn on the axis or MC_Reset the axis according to the situation.

Error code	Description	Contents	Corrective Action
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 whether *lrModulo* is set to the correct range.
0x189DA	DMC_MM_INVALID_POS_VALUES	lrPosition input error	Check whether *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 whether the input value is out of range.
0x18A1F	DMC_GCMD_NO_MASTER_POINT_MATCH	No corresponding master axis postion	Check whether the input and output parameters are correct.

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 Series Operation 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	DML_NO_ERROR	No error messages	-
0x00001	DML_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	DML_DI_AXIS_ERROR	Axis error	Confirm Slave error information and eliminate the error, and then Re-run MC_Reset_DML.
0x00015	DML_WRONG_OPMODE	Wrong control mode	Function block does not support running in the current mode. To run this function block, run SMC_SetControllerMode first to switch the axis to the appropriate mode.
0x00022	DML_AXIS_NOT_READY_FOR_MOTION	The slave state cannot run 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	DML_MA_MR_MODULO_ACT_POS_NOT_MAPPED	PDO lacks the essential parameter.	Configure Actual Position (16#6064) in PDO.
0x00024	DML_MV_INVALID_VELACCDEC_VALUES	Invalid velocity, acceleration, or 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	DML_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	DML_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	DML_CGR_MODULOPERIOD_NOT_INTEGRAL	Module period is not an integer.	After modifying the *fModuloPeriodU* parameter, re-run the function block.
0x0005E	DML_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.

Error code	Description	Contents	Corrective Action
0x0005F	DML_CGR_MODULOPERIOD_NON_POSITIVE	Module period cannot be a negative.	After modifying the *fPositionPeriod* parameter, re-run the function block.
0x00060	DML_CGR_MODULOPERIOD_TOO_SMALL	Module period is too small.	After modifying the *fPositionPeriod* parameter, re-run the function block.
0x00061	DML_CGR_MODULOPERIOD_TOO_LARGE	Module period is too large.	After modifying the *fPositionPeriod* parameter, re-run the function block.
0x00078	DML_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	DML_R_ERROR_NOT_RESETTABLE	Error, non-resettable	Confirm whether the Slave error has been eliminated. After error disappeared, restart MC_Reset_DML.
0x00083	DML_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	DML_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	DML_WP_SENDING_ERROR	Slave has no corresponding OD, or writing 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.
0x0008E	DML_WP_TMT_PARAM_CONVERSION_ERROR	Conversion error of the axis parameter to servo OD. Unknown SoftMotionLight parameter	The written parameter does not exist.
0x000AA	DML_H_AXIS_WASNT_STANDSTILL	Axis is not at the Standstill state.	Make axis enter the Standstill state, and re-run the function block.
0x000B7	DML_MS_AXIS_IN_ERRORSTOP	Driver is at the Errorstop state. Cannot run 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 run motion control instructions.	After confirming the axis at the state that can run motion

Error code	Description	Contents	Corrective Action
			instructions, re-run the function block.
0x186A2	DML_MA_INVALID_VALUES	The input parameter value is invalid.	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.
0x186A5	DML_MA_R2R_ENABLED	This function cannot be run while R2R is running.	After the R2R function is completed, run the function block again.
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 run motion instructions, re-run the function block.
0x186AC	DML_MR_INVALID_VALUES	The input parameter value is invalid.	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 run motion control instructions.	After confirming the axis at the state that can run motion instructions, re-run the function block.
0x186B6	DML_MV_INVALID_VALUES	The input parameter value is invalid.	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

Error code	Description	Contents	Corrective Action
			Object, and then re-run the function block.
0x186BF	DML_TC_AXIS_NOT_READY_FOR_MOTION	The axis state cannot run 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 value is invalid.	Confirm pin input parameter value. Re-run 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.
0x186C9	DML_VC_AXIS_NOT_READY_FOR_MOTION	The axis state cannot run motion control instructions.	After confirming the axis at the state that can run motion instructions, re-run the function block.
0x186CA	DML_VC_INVALID_VALUES	The input parameter value is invalid.	Confirm pin input parameter value. Re-run 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 run 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 value is invalid.	Confirm pin input parameter value. Re-run 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

Error code	Description	Contents	Corrective Action
			Object, and then re-run the function block.
0x186DD	DML_MS_AXIS_NOT_READY_FOR_MOTION	The axis state cannot run 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.
0x186FA	DML_P_R2R_ENABLED	This function cannot be run while R2R is running.	After the R2R function is completed, run the function block again.
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

Error code	Description	Contents	Corrective Action
0x18A94	DML_R2R_CIG_UNSUPPORTED_PID_FEEDBACK_SOURCE	PID feedback source not supported.	PID feedback source not supported
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	Unsupported line speed source.	Unsupported line speed source
0x18A97	DML_R2R_CIG_UNSUPPORTED_LINE_SPEED_SOURCE	Unsupported line speed source.	Re-enter the supported source.
0x18A98	DML_R2R_CIG_UNSUPPORTED_WINDING_MODE	Unsupported winding mode.	Re-enter the supported winding mode.
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.
0x18AB2	DML_R2R_RD_MATERIAL_THICKNESS_IS_OVER_RANGE	The roll material thickness is out of range.	Enter an appropriate value.
0x18AB3	DML_R2R_RD_ROLL_DIAMETER_FILTER_TIME_IS_OVER_RANGE	Roll diameter calculation filter time is out of range.	Enter an appropriate value.

Error code	Description	Contents	Corrective Action
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 running in the 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 to leave the continuous motion state.

A.3.3 For SM3_Drive_ETC Library

The following table lists the errors, error codes, and the solutions of the SM3_Drive_ETC library.

Error code	Description	Contents	Corrective Action
0x00000	SMC3_ETC_CO_NO_ERROR	No error	-
0x00001	SMC3_ETC_CO_FIRST_ERROR	Parameter error	Check whether the parameters uiIndex, usiSubIndex, and usiDataLength are valid and within the slave range.
0x00002	SMC3_ETC_CO_OTHER_ERROR	EtherCAT Master not found	Check the master state.
0x00003	SMC3_ETC_CO_DATA_OVERFLOW	Communication error	The SDO transferred is too large. Modify it and run the function block again.
0x00004	SMC3_ETC_CO_TIMEOUT	Communication error	SDO Time Out. Check whether the servo has its OD.

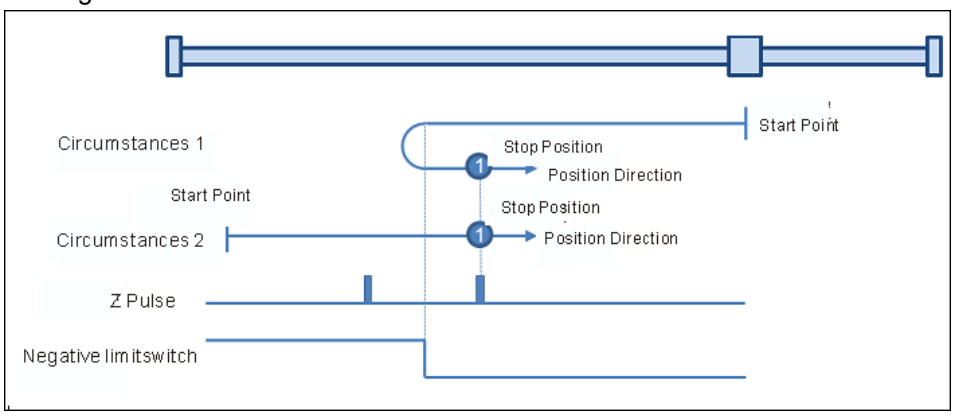
A.4 Explanation of DMC_Home_P

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.

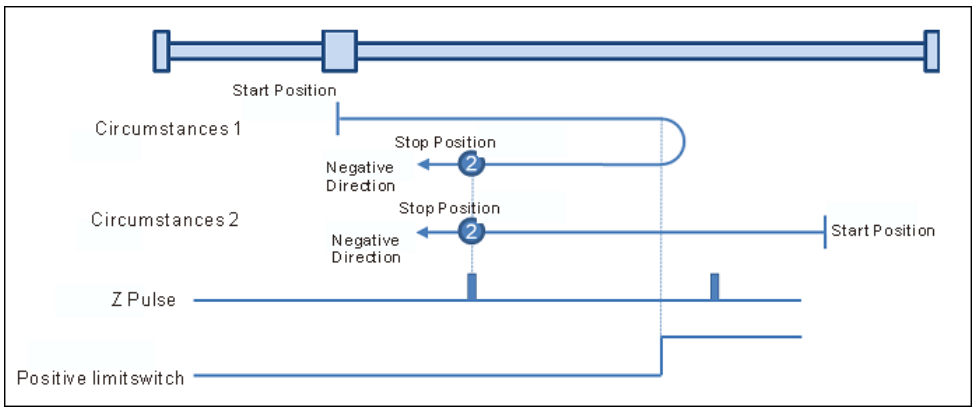
  Circumstance 2: MC_Home instruction is run when the negative limit switch is ON and the axis moves in the positive direction at the second-phase speed. Where the first Z pulse is met is the home position when the negative limit switch is OFF.



- **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.

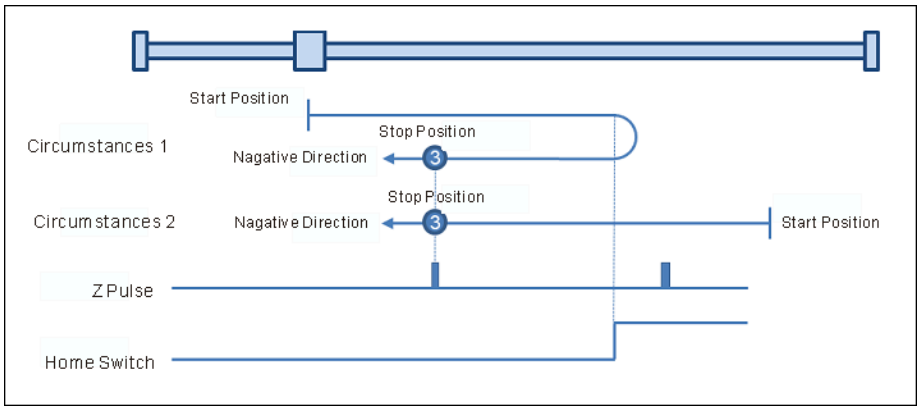
  Circumstance 2: MC_Home instruction is run when the positive limit switch is ON and the axis moves in the negative direction at the second-phase speed. Where the first Z pulse is met is the home position while the positive limit switch is OFF.



- **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.

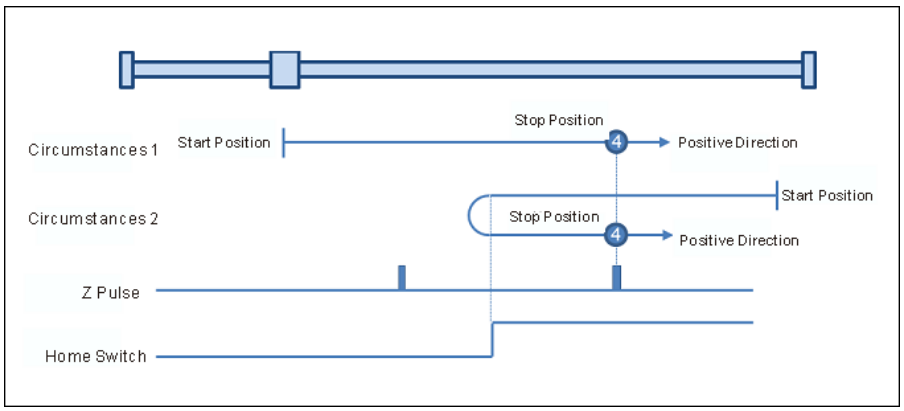
Circumstance 2: When the home switch is ON, MC_Home instruction is run and the axis directly moves in the negative direction at the second-phase speed. Where the first Z pulse is met is the home position while the home switch is OFF.



- **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.

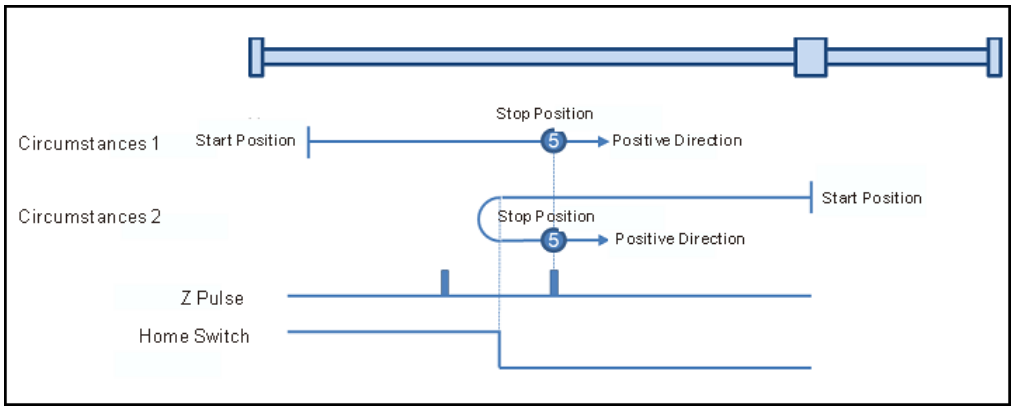
  Circumstance 2: When the home switch is ON, MC_Home instruction is run and the axis moves in the negative direction at the second-phase speed. When the axis encounters that the home switch is OFF, the motion direction changes and the axis moves at the second-phase speed. Where the first Z pulse is met 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.

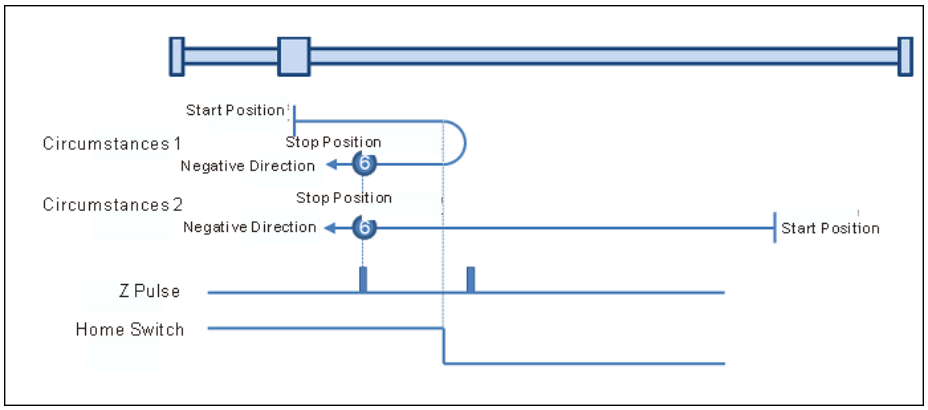
  Circumstance 2: When the home switch is OFF, MC_Home instruction is run and the axis moves in the negative direction at the first-phase speed. When the home switch is ON, the motion direction changes and the axis moves at the second-phase speed. Where the first Z pulse is met is the home position when 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.

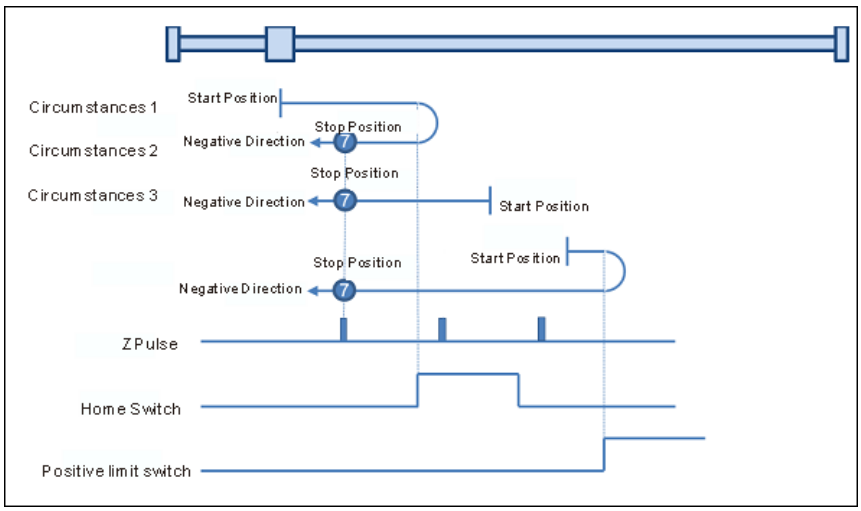


- **Mode 7**: **Homing which depending on the home switch, positive limit switch and Z pulse**

  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.

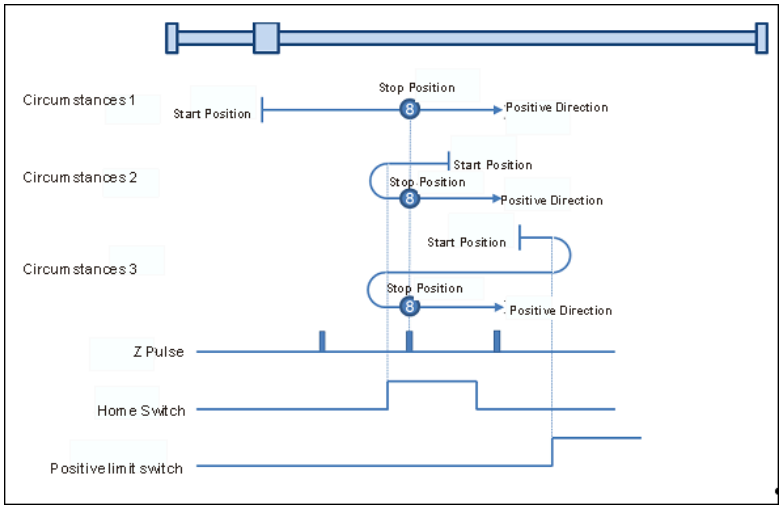


- **Mode 8**: **Homing depending on the home switch, positive limit switch and Z pulse.**

  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.

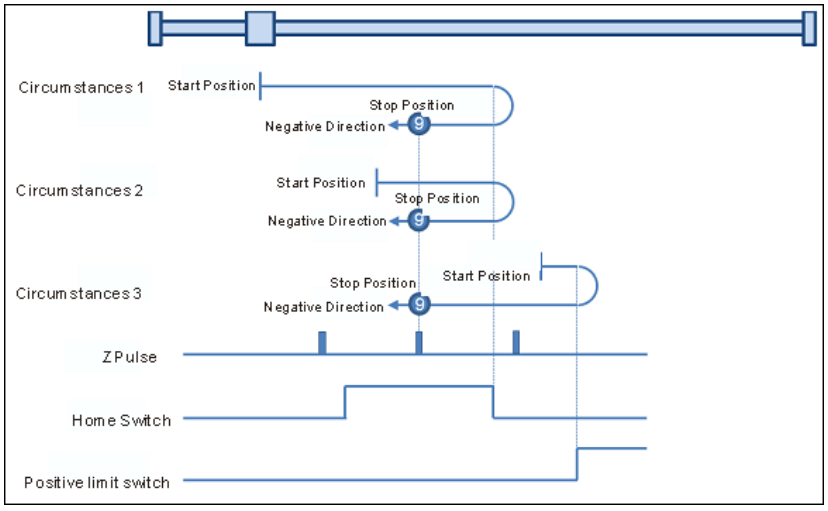


- **Mode 9**: **Homing depending on the home switch, positive limit switch and Z pulse**

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.

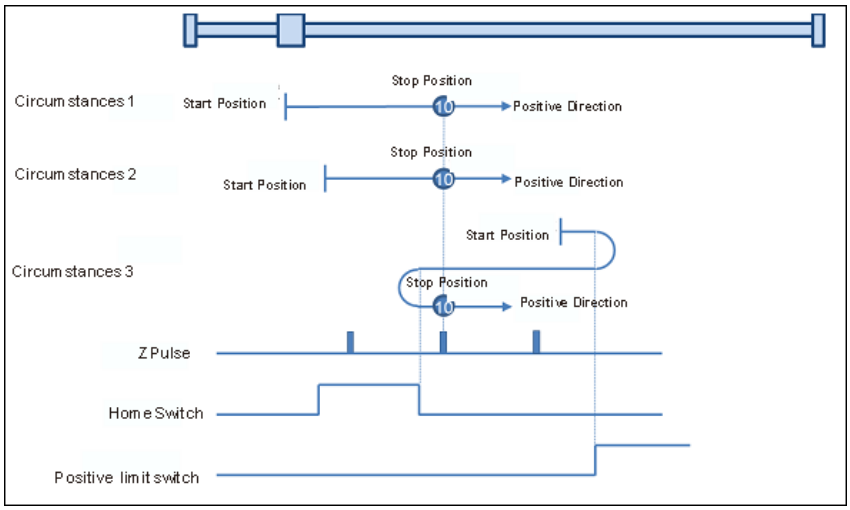


- **Mode 10**: **Homing depending on the home switch, positive limit switch and Z pulse.**

  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

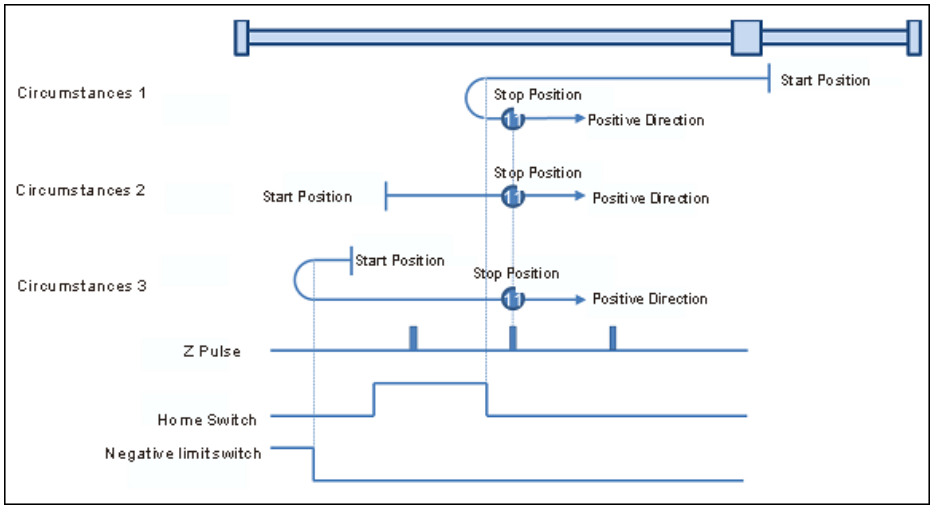
- **Mode 11**:

  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.

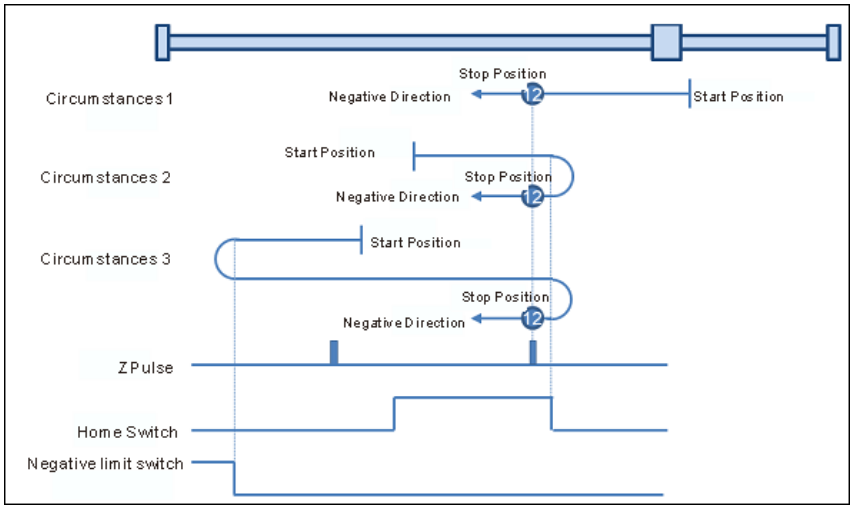


- **Mode 12**: **Homing depending on the home switch, negative limit switch and Z pulse**

  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.

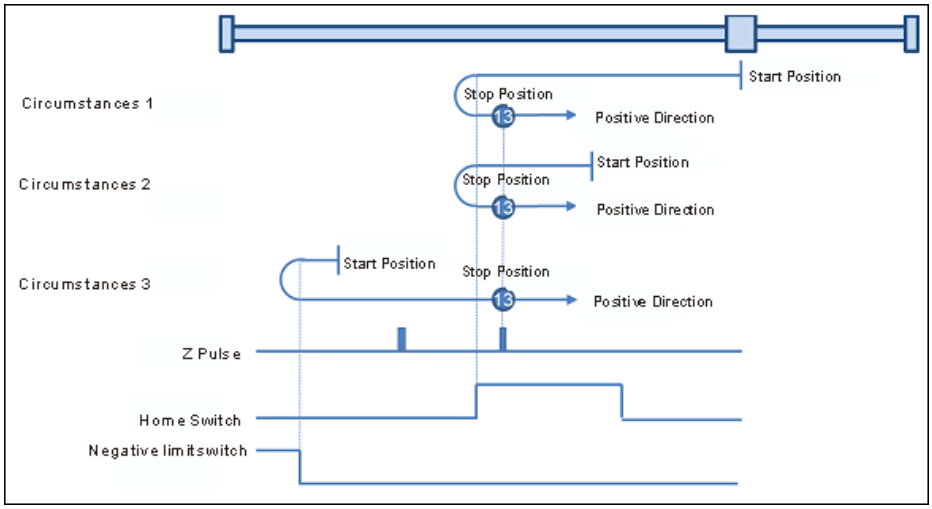


- **Mode 13**: **Homing depending on the home switch, negative limit switch and Z pulse**

  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.

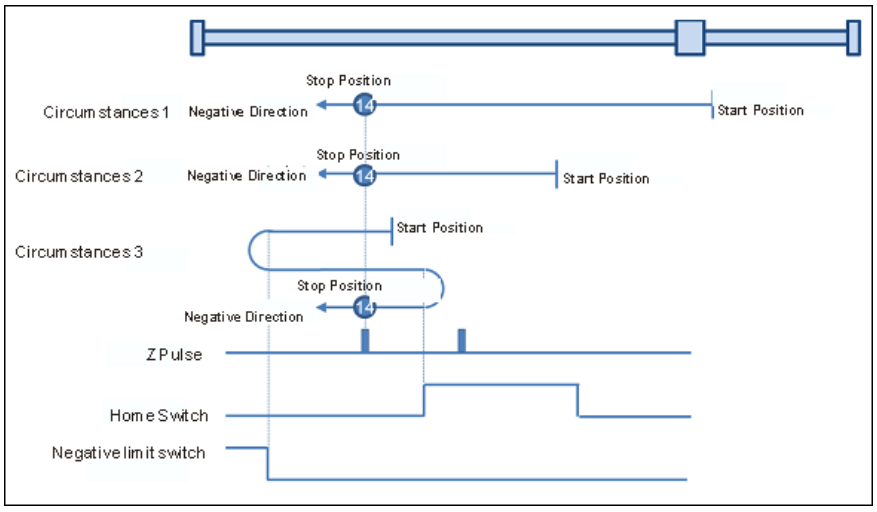


- **Mode 14**: **Homing depending on the home switch, negative limit switch and Z pulse**

  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.

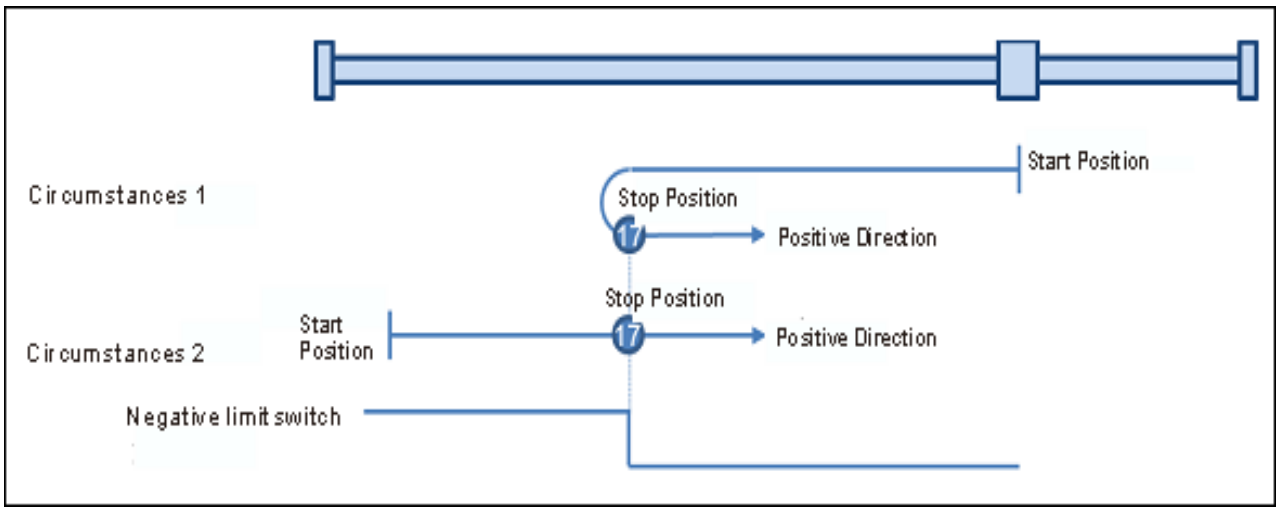
Mode 17–mode 30 Homing which has nothing to do with Z pulse

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.

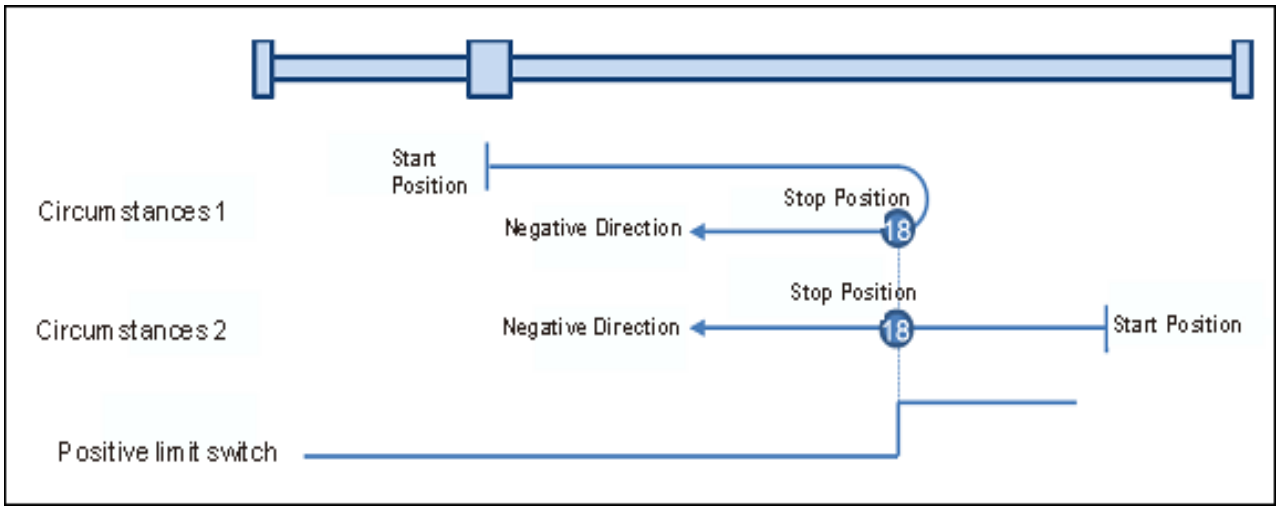
  Circumstance 2: MC_Home instruction is run when the negative limit switch is ON and the axis moves in the positive direction at the second-phase speed. Where the servo is the home position when the negative limit switch is OFF.



- **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.

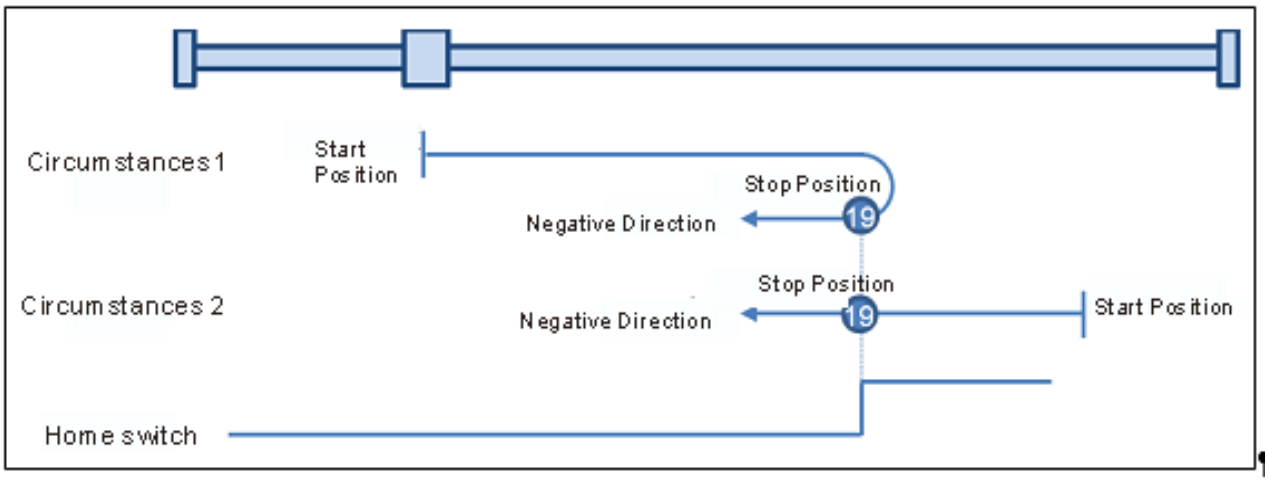
  Circumstance 2: MC_Home instruction is run when the positive limit switch is ON and the axis moves in the negative direction at the second-phase speed. Where the servo is the home position while the positive limit switch is OFF.



- **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.

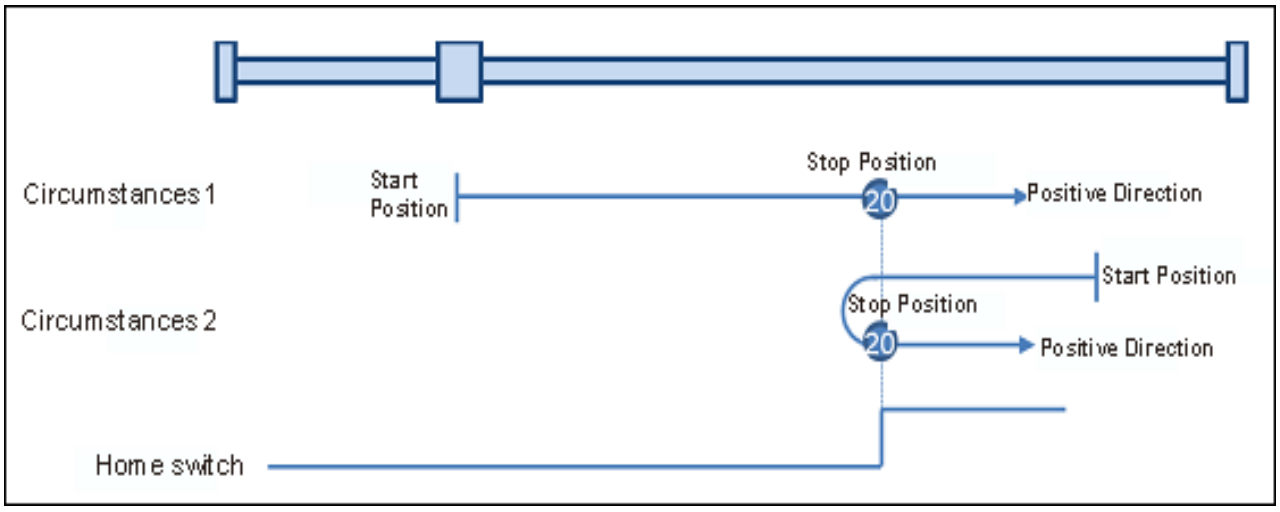
Circumstance 2: MC_Home instruction is run and the axis directly moves in the negative direction at the second-phase speed while the home switch is ON. And where the axis stands is the home position at the moment when the home switch becomes OFF.



- **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.

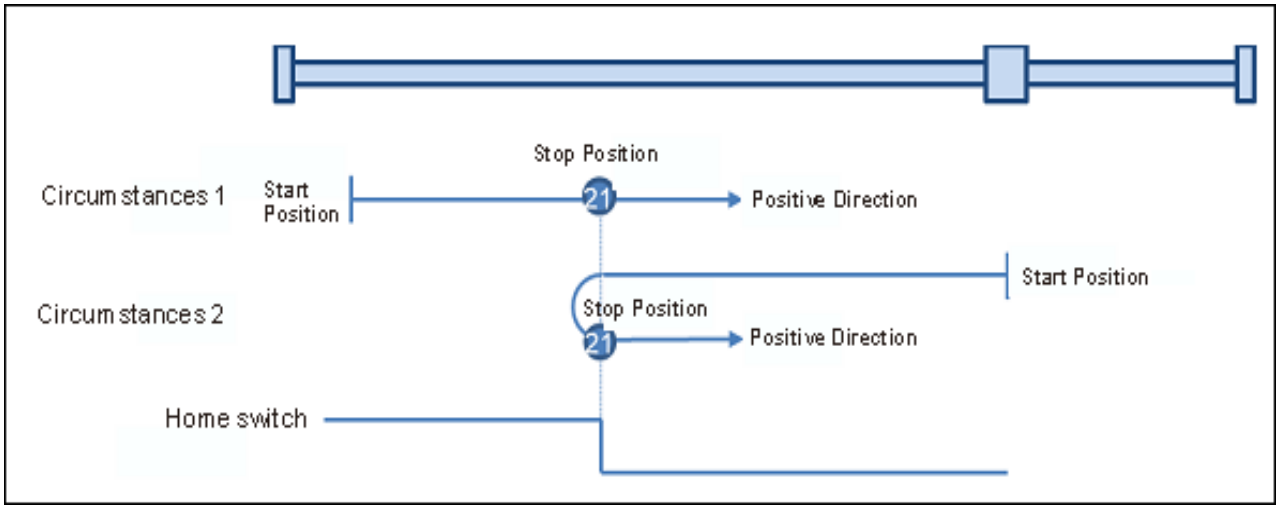


- **Mode 21**: **Homing which depends on the home switch, similar to mode 5, but has nothing to do with Z pulse.**

  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-

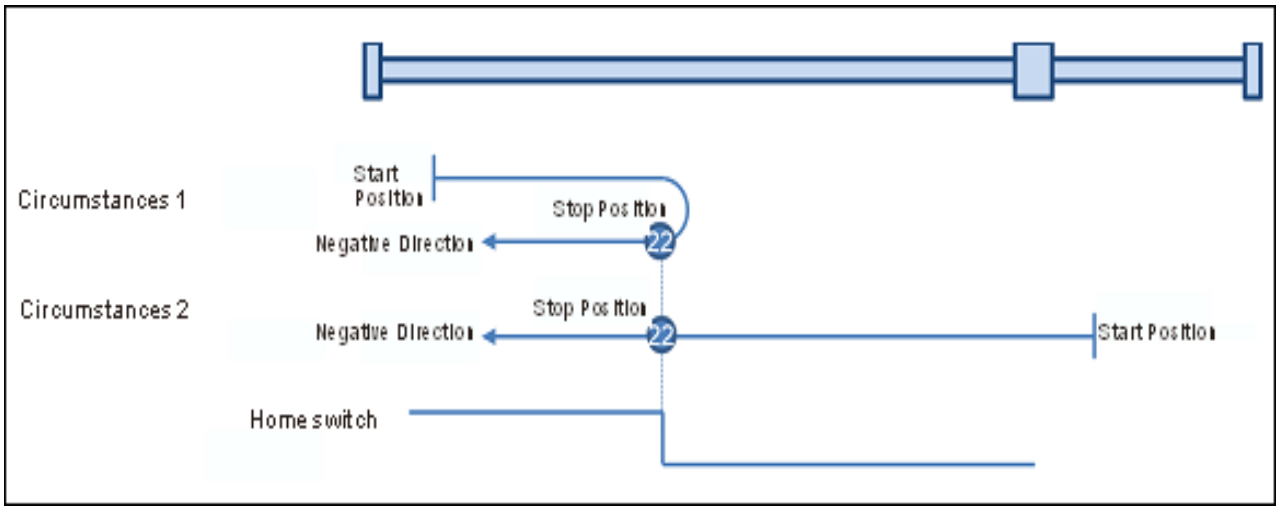
phase speed once the home switch becomes ON. And where the axis stands is the home position at the moment the home switch becomes OFF.



- **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.



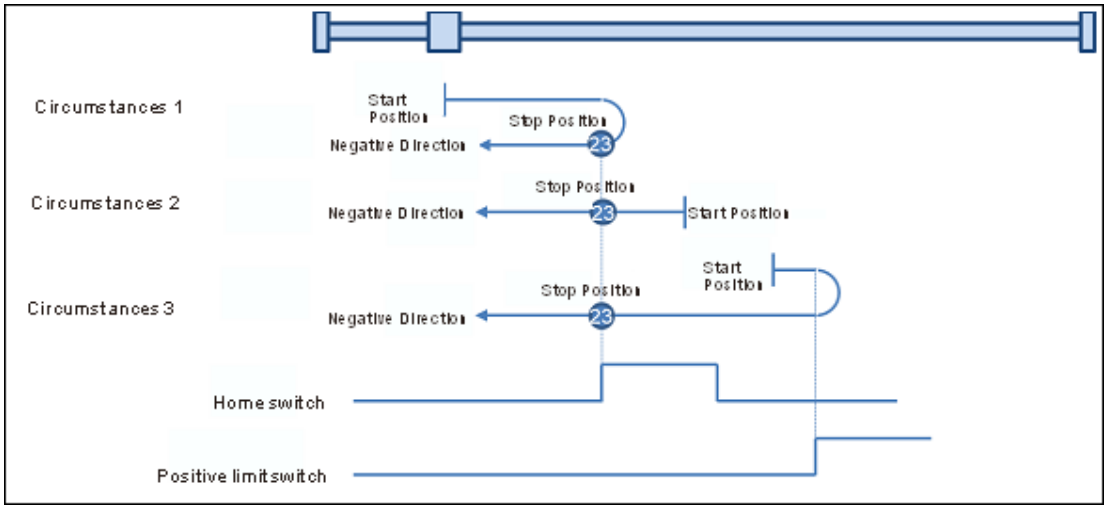
- **Mode 23**: **Homing which depends on the home switch and positive limit switch, similar to mode 7, but has nothing to do with Z pulse.**

  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.

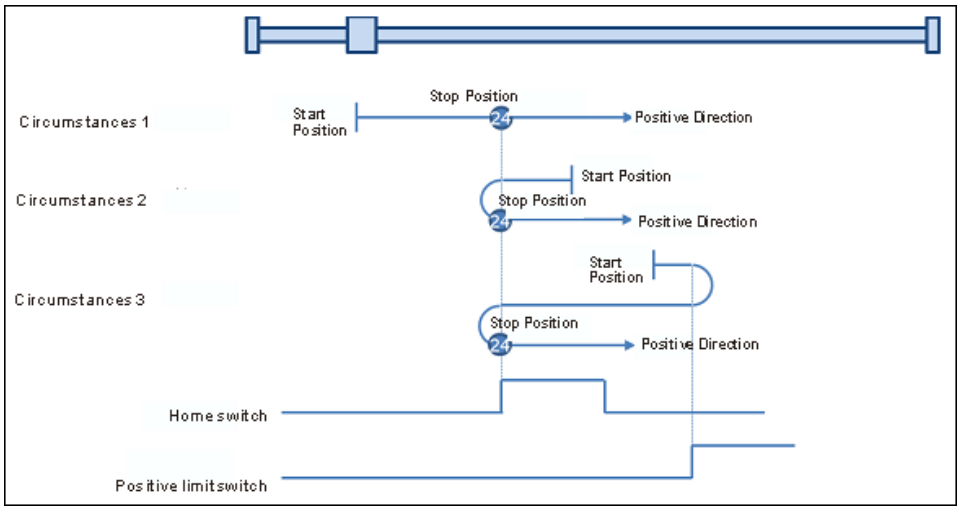


- **Mode 24**: **Homing which depends on the home switch and positive limit switch, similar to mode 8, but has nothing to do with Z pulse.**

  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.



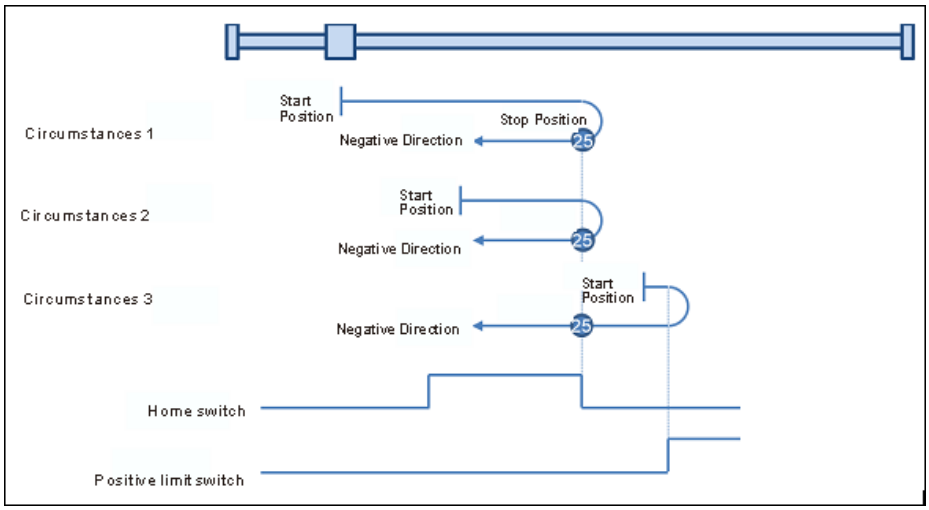
- **Mode 25**: **Homing which depends on the home switch and positive limit switch, similar to mode 9, but has nothing to do with Z pulse.**

  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.

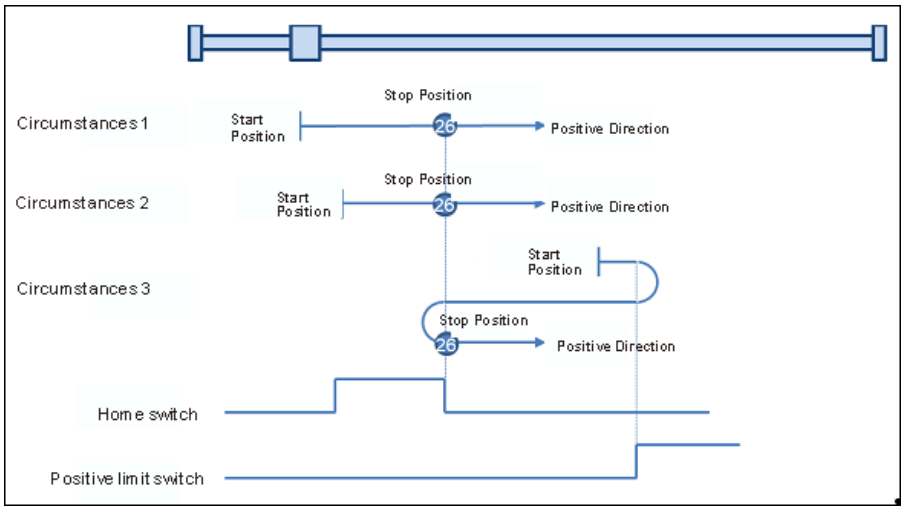


- **Mode 26**: **Homing which depends on the home switch and positive limit switch, similar to mode 10, but has nothing to do with Z pulse.**

  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.



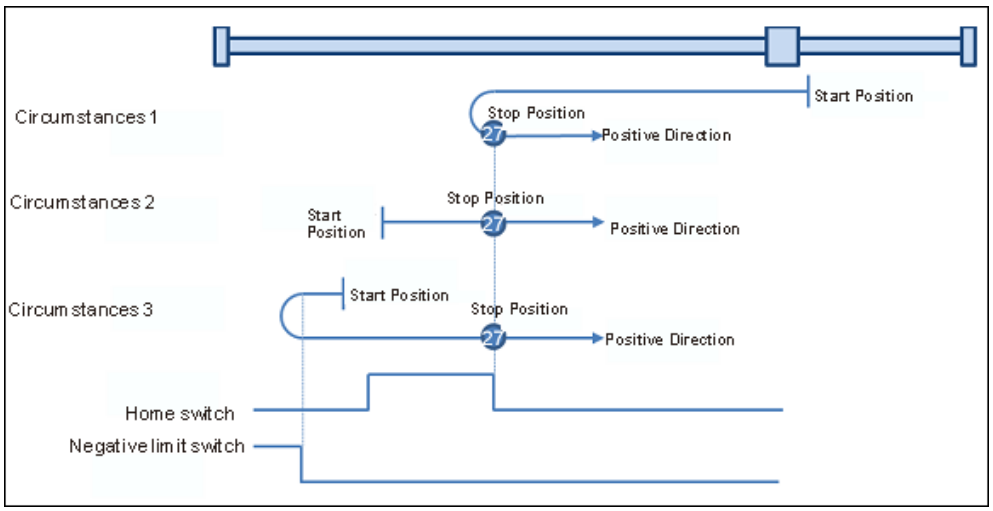
- **Mode 27**: **Homing which depends on the home switch and negative limit switch, similar to mode 11, but has nothing to do with Z pulse.**

  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.

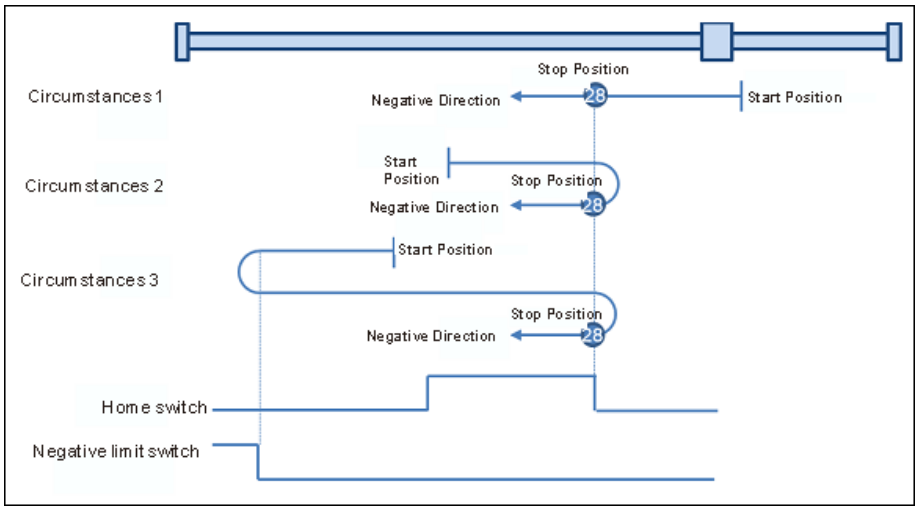


- **Mode 28**: **Homing which depends on the home switch and negative limit switch, similar to mode 12, but has nothing to do with Z pulse.**

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.

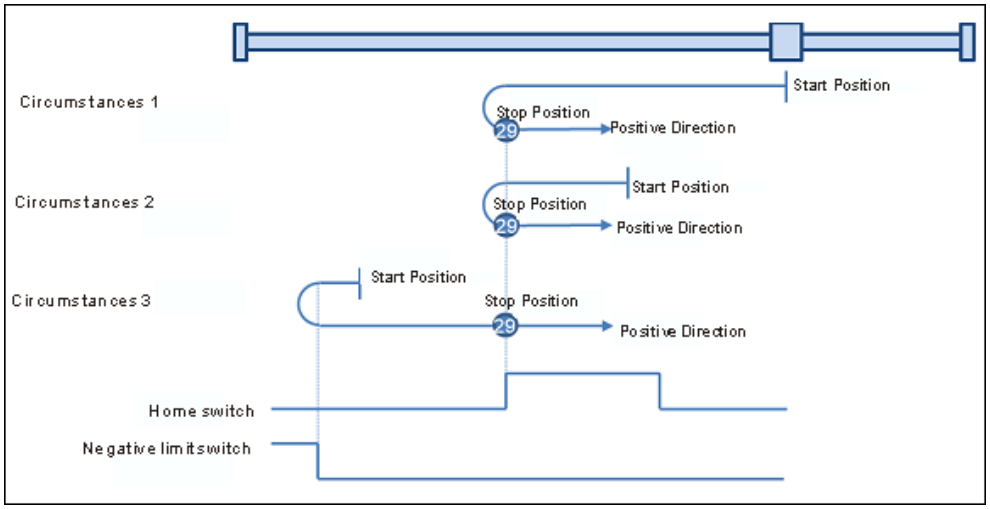


- **Mode 29**: **Homing which depends on the home switch and negative limit switch, similar to mode 13, but has nothing to do with Z pulse.**

  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.



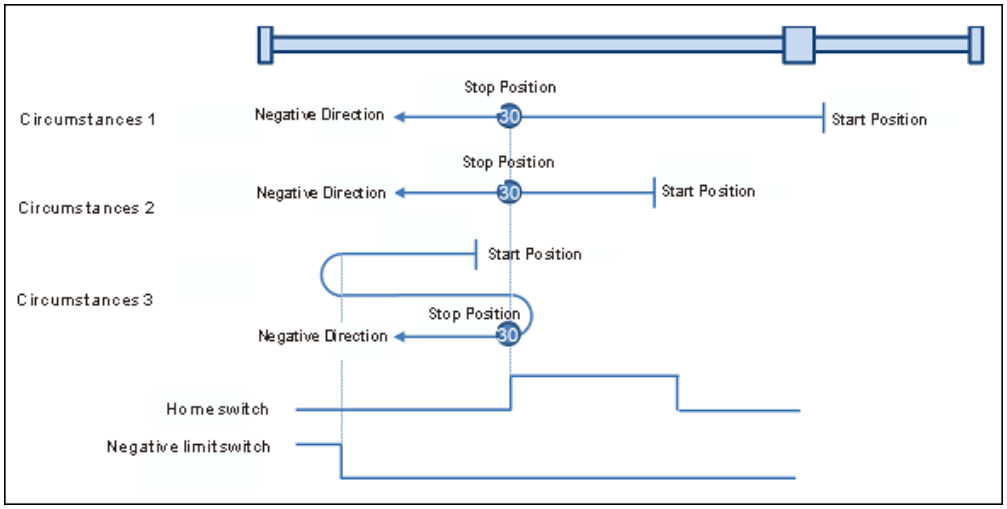
- **Mode 30**: **Homing which depends on the home switch and negative limit switch, similar to mode 14, but has nothing to do with Z pulse.**

  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.



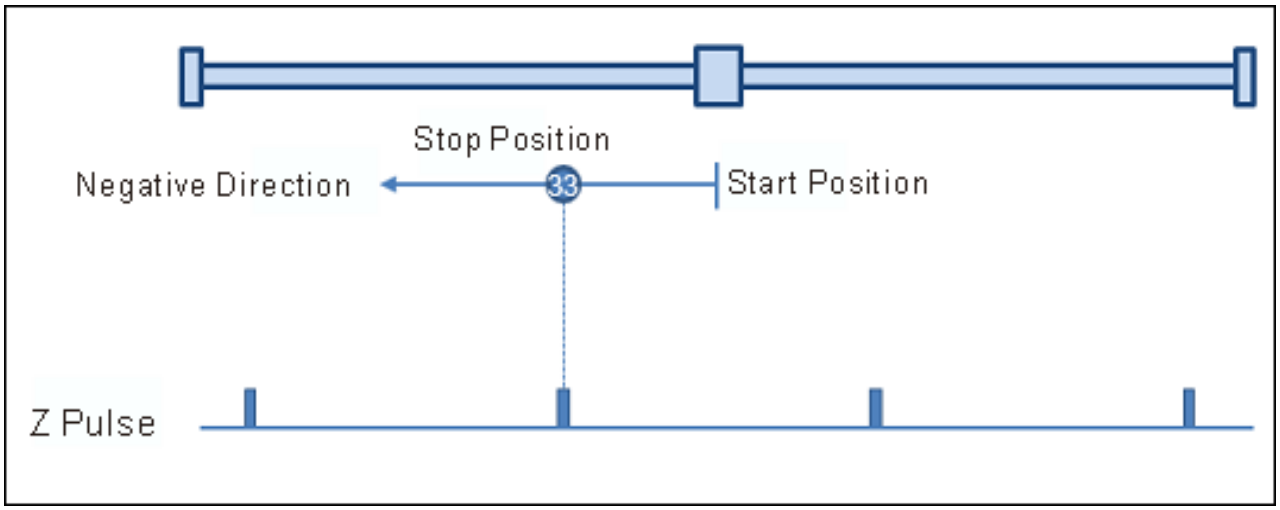
- **Mode 31 and Mode 32**: **Reserved**

  Mode 31 and mode 32 Reserved for future homing.

Mode 33, Mode 34: Homing which only depends on Z pulse

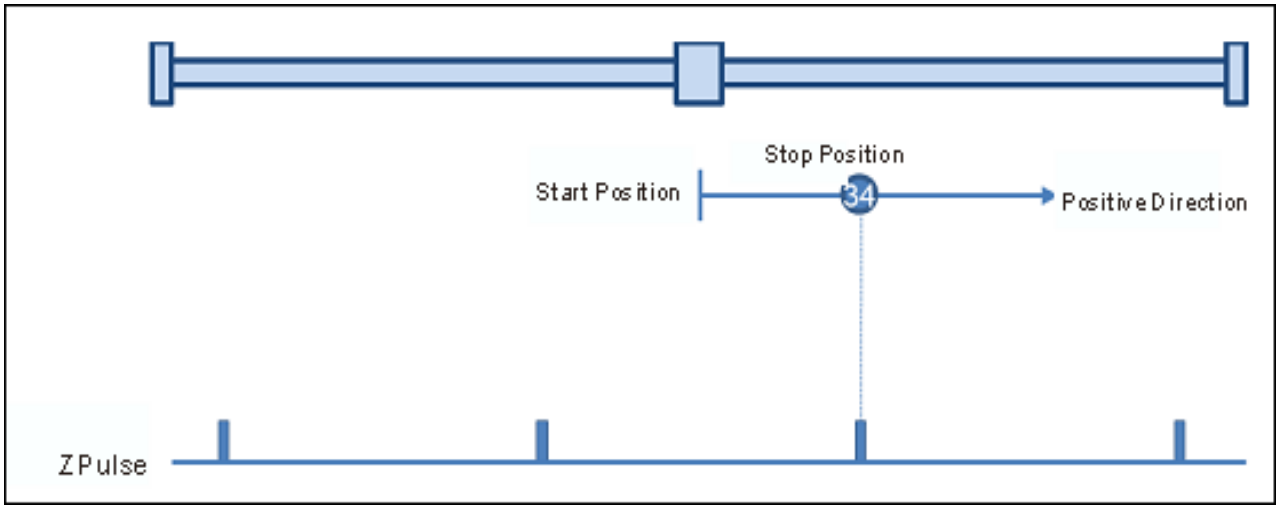
- **Mode 33**: **Homing depending on Z pulse (Negative direction)**

  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.

Smarter. Greener. Together.

Industrial Automation Headquarters

Delta Electronics, Inc.
Taoyuan Technology Center
No.18, Xinglong Rd., Taoyuan District,
Taoyuan City 330477, 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.
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.
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: +66-2709-2827

Delta Electronics (Australia) Pty Ltd.
Unit 2, Building A, 18-24 Ricketts Road,
Mount Waverley, Victoria 3149 Australia
Mail: IA.au@deltaww.com
TEL: +61-1300-335-823 / +61-3-9543-3720

Americas

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

Delta Electronics Brazil Ltd.
Estrada Velha Rio-São Paulo, 5300 Eugênio de
Melo - São José dos Campos CEP: 12247-004 - SP - Brazil
TEL: +55-12-3932-2300 / FAX: +55-12-3932-237

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

EMEA

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

Delta Electronics (Netherlands) B.V.
Automotive Campus 260, 5708 JZ Helmond, The Netherlands
Mail: Sales.IA.Benelux@deltaww.com
TEL: +31(0)40 800 3900

Delta Electronics (Netherlands) B.V.
Coesterweg 45, D-59494 Soest, Germany
Mail: Sales.IA.DACH@deltaww.com
TEL: +49 2921 987 238

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

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

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

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

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

Eltek Dubai (Eltek MEA DMCC)
OFFICE 2504, 25th Floor, Saba Tower 1,
Jumeirah Lakes Towers, Dubai, UAE
Mail: Sales.IA.MEA@deltaww.com
TEL: +971(0)4 2690148

DIAS-Manual-0022-EN
2023/05/11