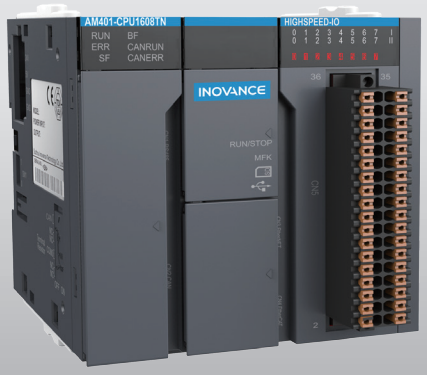
INOVANCE

FORWARD, ALWAYS PROGRESSING







Medium-Sized PLC

Programming Guide (Motion Control)

Industrial Automation | Intelligent Elevator | New Energy Vehicle | Industrial Robot | Rail Transit

Data code 19012378 A00

Preface

Thank you for choosing the medium-sized programmable logic controller (PLC) and expansion modules developed by Inovance and using InoProShop.

- Intended Audience

This guide is intended for technicians who configure, program, and commission motion control functions through the medium-sized PLC (including AM400/AM500/AM600/AC700/AP700/AC800). Readers of this guide are supposed to have a basic understanding of automation and PLC.

Before using the software, read this guide carefully and perform operations correctly with due attention to safety.

- Terms and Abbreviations

Term/Abbr.	Description
InoProShop	Programming software for the medium-sized PLC
Gateway	Dedicated communication service for the medium-sized PLC in this guide
PLC	Programmable logic controller

- Revision History

Date	Version	Change Description
Sept 2024	A00	First release

Contents

Preface 1
1. Overview of the PLCopen Specification 7
2. Composition of the Motion Control Application System 9
3. Composition of the Motion Control Program 11
3.1 User Program Structure 11
3.1.1 User Program Composition 11
3.1.2 Task Type 11
3.1.3 Benefits of a User Program Consisting of Multiple POUs 12
3.1.4 How to Achieve Both Logic Control and Motion Control in User Program 13
3.2 Writing and Commissioning a Simple User Program 13
3.2.1 Creating a Project 14
3.2.2 Writing POUs for Function Processing 15
3.2.3 Setting Motor Parameters 17
3.2.4 Writing Marquee Control Logic 19
3.2.5 Associating a Variable with the Hardware Output Port 19
3.2.6 Troubleshooting User Program Compilation 20
3.2.7 Monitoring the Running of the User Program 21
3.2.8 Summary of Typical Steps of Writing a Motion Control Project 21
4. Execution Mechanism of the Motion Control Program 25
4.1 Task and Configuration in the User Project 25
4.2 Dataflow Analysis of the EtherCAT Bus Network 27
4.3 Data Process for Communication with Servo Slaves 29
4.3.1 Control Information Process 30
4.3.2 CiA402 Data Object Dictionary and Common Objects for Servo Drives 33
4.3.3 Configuration of Servo Axis Motor Parameters 46
4.3.4 EtherCAT Network Status Initialization and Management 49
4.3.5 Servo Axis and I/O Port Control Data Refresh 51
4.4 Timing of MC Data Transmission 52
4.5 Processing Mechanism for Executing MC Function Blocks 53
4.5.1 Cyclic Synchronous Position Control Mode for Servo Motion Commands 53
4.5.2 Data Structure of the Servo Axis 54
4.5.3 Servo Axis Status and Transition Rules 55
4.5.4 Execution Logic of the MC Function Block 56
4.5.5 Data Interaction Between POUs of Tasks of Different Priorities 58
5. Application Programming of User Program 61
5.1. MC Programming For Single-axis MC Positioning 61
5.1.1 Notes for MC Application Programming 61
5.1.2 MC Function Blocks Commonly Used for Single-Axis Control 61
5.1.3 MC Commands and PDO/SDO Configuration 62

5.2 Motion Control Programming for Multi-axis Cam Synchronization 63
5.2.1 Main Function Blocks For Cam Running 64
5.2.2 Master and Slave Axes in Relative Position Mode 67
5.2.3 Master Axis in Absolute Position Mode and Slave Axis in Relative Position Mode 67
5.2.4 Master Axis in Relative Position Mode and Slave Axis in Absolute Position Mode 68
5.3 Cyclic Mode Characteristics of the Cam Table 68
5.3.1 Offset for CamIn Operation 69
5.3.2 Calculation of Master Axis Scaling During Cam Running 69
5.3.3 Calculation of Slave Axis Scaling During Cam Running 70
5.3.4 Characteristics of and Precautions for Using Offset and Scale in Cam Running 70
5.3.5 MC_CamOut FB for Exiting Cam Running Status 71
5.4 MC_Phasing FB for Cam Master Axis Phase Adjustment 71
5.5 Cam Table Design and Its Data Structure 72
5.5.1 Characteristics of the Cam Table 72
5.5.2 Input Mode of the Cam Table 73
5.5.3 Internal Data Structure and Arrays of the Cam Table 74
6. Common MC Instructions 77
6.1 Single-axis Instructions 77
MC_AccelerationProfile 77
MC_Halt 79
MC_HaltSuperImposed 81
MC_Home 83
MC_MoveAbsolute 85
MC_MoveAdditive 91
MC_MoveRelative 93
MC_MoveSuperImposed 97
MC_MoveVelocity 99
MC_MoveFeed 102
MC_PositionProfile 109
MC_Power 111
MC_ReadActualPosition 113
MC_ReadAxisError 115
MC_ReadBoolParameter 116
MC_ReadStatus 117
MC_ReadParameter 118
MC_Reset 120
MC_Stop 121
MC_VelocityProfile 123
MC_WriteBoolParameter 125
MC_WriteParameter 126
MC_AbortTrigger 127
MC_ReadActualTorque 128

MC_ReadActualVelocity 129
MC_SetPosition 130
MC_TouchProbe 132
SMC_MoveContinuousAbsolute 134
SMC_MoveContinuousRelative 136
MC_Jog 138
SMC_Inch 139
SMC3_PersistPosition 141
SMC3_PersistPositionSingleturn 143
SMC_CheckAxisCommunication 145
SMC_FollowPosition 147
SMC_FollowPositionVelocity 151
SMC_FollowVelocity 153
SMC_FollowSetValues 154
SMC_SetControllerMode 156
SMC_CheckLimits 158
SMC_GetMaxSetAccDec 160
SMC_GetMaxSetVelocity 161
MC_GetTrackingError 163
SMC_InPosition 164
SMC_ReadSetPosition 167
SMC_SetTorque 168
SMC_BacklashCompensation 169
SMC_ChangeGearingRatio 172
SMC_ReadFBError 173
SMC_ClearFBError 175
SMC3_PersistPositionLogical 176
SMC_Homing 178
MC_TorqueControl 182
MC_ImmediateStop 184
MC_ResetFollowingError 186
MC_SetTorqueLimit 188
MC_ReadDigitalInput 190
HMC_Reset 191
SMC_SetSoftwareLimits 193
6.2 Axis Group Instructions (Master/Slave Axis Instructions) 194
SMC_CamRegister 194
SMC_GetCamSlaveSetPosition 197
SMC_GetTappetValue 199
MC_CamTableSelect 201
MC_CamOut 215
MC_GearIn 218

MC_GearOut 220
MC_Phasing 227
SMC_CAMBounds 230
SMC_CAMBounds_Pos 233
SMC_WriteCAM 234
6.3 Other Functional Specifications 235
6.3.1 Instruction Cache 235
6.3.2 Hitting Limit 237
6.3.3 Defaults of Motion Control Function Blocks 238
6.3.4 Curve Reversal Prevention 239
6.4 Axis Group Instructions 240
6.4.1 MC_GroupEnableV2 246
6.4.2 MC_GroupDisableV2 247
6.4.3 MC_GroupHomeV2 248
6.4.4 MC_GroupSetPositionV2 250
6.4.5 MC_GroupReadActualPositionV2 251
6.4.6 MC_GroupReadActualVelocityV2 253
6.4.7 MC_GroupReadActualAccelerationV2 254
6.4.8 MC_GroupStopV2 256
6.4.9 MC_GroupHaltV2 259
6.4.10 MC_GroupSetOverrideV2 262
6.4.11 MC_MoveLinearRelativeV2 264
6.4.12 MC_MoveLinearAbsoluteV2 277
6.4.13 MC_MoveCircularRelativeV2 287
6.4.14 MC_MoveCircularAbsoluteV2 296
6.4.15 MC_GroupResetV2 304
6.4.16 MC_GroupReadErrorV2 306
6.4.17 MC_GroupReadStatusV2 307
6.4.18 MC_GroupInterruptV2 309
6.4.19 MC_GroupContinueV2 311
6.4.20 Axis Group Error Codes 313
5.5.4 Reference and Dynamic Switchover of the Cam Table 320
7. Simulation and Commissioning 322
7.1 Simulation Controller 322
7.2 Simulation Servo Drive 323
Appendix A Homing Modes Supported by IS620N 325
A.1 Description of Homing Modes: 325
Appendix B: CiA402 Common Data Object Cheat Sheet Supported by IS620N 335
Appendix C Error Codes 339



Chapter 1 Overview of the PLCopen Specification

1. Overview of the PLCopen Specification

IEC 61131 is an international standard for general-purpose programmable logic controllers (PLCs). It was initiated by several leading PLC technology companies in Europe as an industry standard. Part 3 of this standard, IEC 61131-3, provides international specifications for PLC programming and has defined standards for six programming languages.

PLCopen is a promotion group based in Europe for IEC 61131-3. It is a global membership organization where several renowned PLC manufacturers have contributed to refining certain technical details. The aim is to achieve programming standardization and eliminate technological differences and barriers among different PLC manufacturers. This enables users to program different brands of PLCs without the need to learn additional programming methods.

In China, the corresponding national standard, GBT15969.3, was released in 1995 and updated in 2005. It serves as the recommended design standard for PLC device manufacturers. There is also a corresponding PLCopen promotion organization in China.

The PLCopen Specification not only provides recommendations for standardizing general logic control instructions, program structures, and keywords in various languages, but also specifies technical specifications for the motion control (MC) function blocks. This includes naming conventions, specific functions, input and output variable definitions, and relevant timing logic, ensuring maximum compatibility and interoperability in user programming technologies.

The medium-sized PLC adopts the CODESYS programming platform from 3S-Smart Software Solutions GmbH, a German company. This platform fully supports the PLCopen Specification, allowing users to refer to numerous standard function libraries. The programming flexibility of high-level languages makes it easy for PLC manufacturers and users to develop proprietary function blocks and instruction libraries. By utilizing existing control programs, they can create industry-specific process packages to improve programming efficiency.

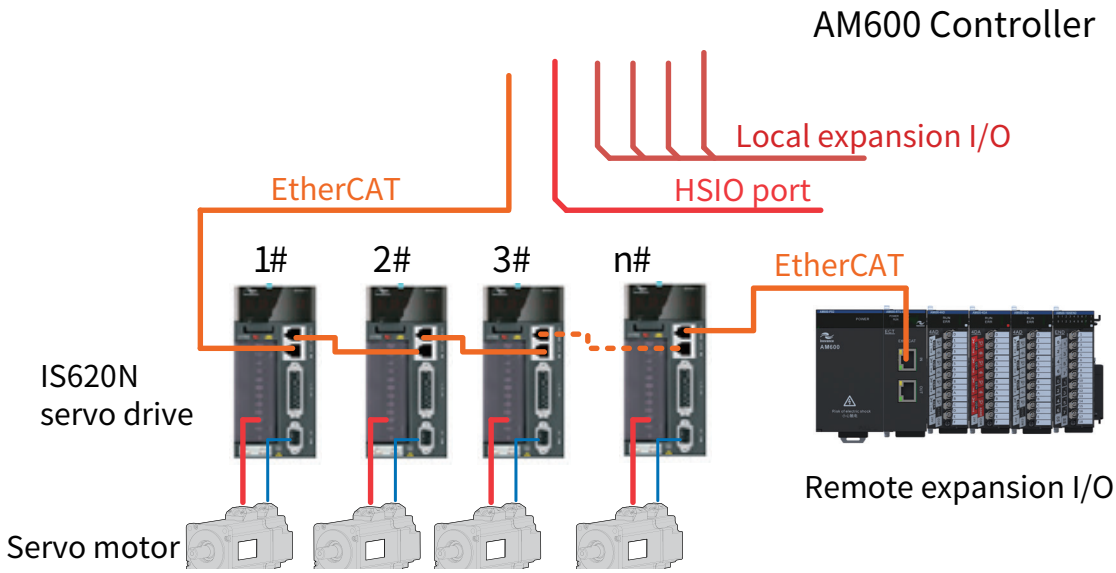


2 Composition of the Motion Control Application System

2. Composition of the Motion Control Application System

The medium-sized PLC is a general-purpose programmable logic controller with the SoftMotion motion control function (CAM/CNC/ROBOT). It controls multiple motion axes through the EtherCAT bus. The following figure shows the typical control bus network, where the IS620N servo is controlled through the bus, and the I/O expansion rack is connected to the CPU module of the medium-sized PLC through the EtherCAT bus.

In the typical motion control network shown below, AM600 is the control master and the servo axes and remote I/O are slaves. The EtherCAT bus is a real-time bus, and the clock of its first slave is used as the reference synchronization clock of the whole network. Therefore, the servo must be installed in the front end of the EtherCAT bus network, that is, the 1# slave of the network must be the servo. The EtherCAT remote module (RTU-ETC) has no internal clock unit, so it is typically installed in the middle or back end of the network requiring motion control.



Motion control (MC) means that the controller commands, through the EtherCAT real-time bus, the servo to run based on software calculations and digital instructions. MC benefits from the high-speed (100 Mbps) and high-frequency (1 ms per communication cycle) interaction of the EtherCAT bus, providing higher accuracy and promptness compared with the traditional pulse control. Correspondingly, MC brings about some programming approaches different from conventional ladder diagram logic control, requiring the use of function blocks that contain more underlying functions.



3. Composition of the Motion Control Program

3. Composition of the Motion Control Program

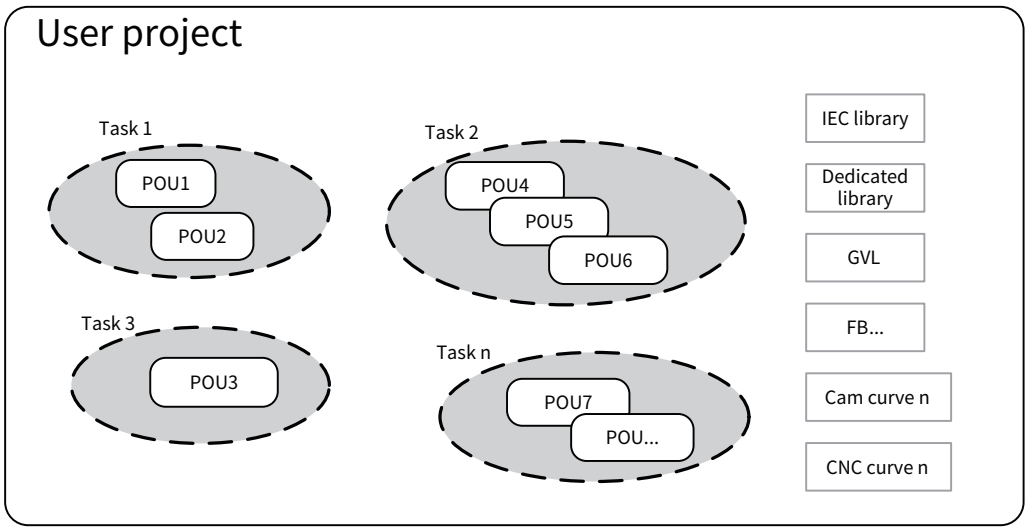
3.1 User Program Structure

The medium-sized PLC is developed based on a multi-tasking operating system, which runs function modules in a multi-tasking mode. A user program can be divided into multiple tasks to be executed separately based on the task priority set by the user.

When writing a user program for the medium-sized PLC, users can divide the program into multiple program organization units based on the type of services processed in the application system and the degree of urgency. In addition, they can specify the execution trigger conditions for each task or the corresponding execution interval (also called execution period) to achieve the optimal control response of the application system.

3.1.1 User Program Composition

As introduced earlier, the multi-tasking mode can be adopted for the medium-sized PLC, that is, several tasks can be executed "at the same time", and each task can have several user program organization units (POUs). The following figure shows the typical composition.



A user project is composed of multiple POUs, which are classified into several task groups based on the POU execution characteristics. Each task group is configured with its own execution characteristics. POUs that are not included in task configuration will not be executed.

The user project also contains some objects supporting the user program, such as the library functions, global variables (GVLs), function blocks (FBs), CAM curves for cam definition, and CNC curves for multi-axis interpolation trajectory definition, as part of the user program.

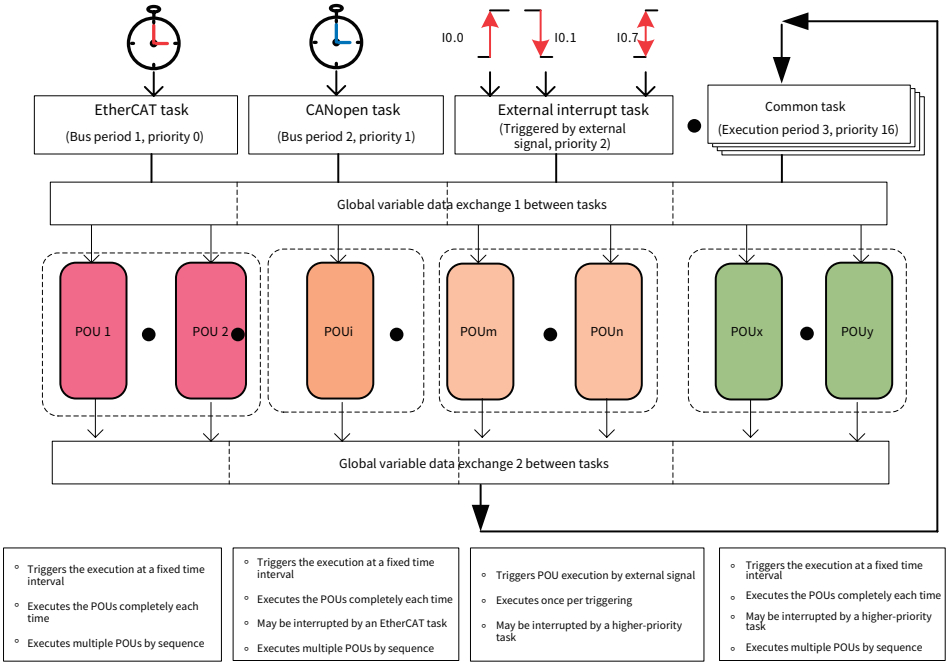
3.1.2 Task Type

Task configuration enables users to divide the user program into several task groups based on the execution requirements. Users can set different execution trigger conditions, execution intervals, and priorities for the task groups.

Common tasks of the medium-sized PLC include the EtherCAT task, CANopen task, HSIO high-speed

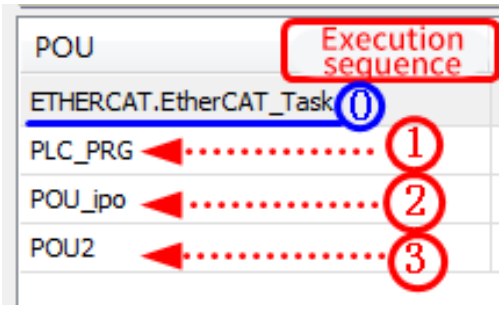
-11-

interrupt task, and main cyclic task. The main body of the user program related to motion control is executed under the EtherCAT task.



The EtherCAT task is one of the most important tasks for the medium-sized PLC, responsible for real-time processing of motion control functions. It operates as a clock interrupt task with a short execution interval and the highest priority. Once specified time conditions are met, the EtherCAT task unconditionally interrupts other tasks and initiates its execution. The interruption continues until all POUs configured under the EtherCAT task have been executed.

Multiple POUs can be specified for a task and these POUs will be executed one by one in the order specified in the task configuration, as shown in the following figure.



The three POUs shown in the figure will be executed in the order of PLC_PRG, POU_ipo, and POU2. The order should be arranged properly, especially when there are global variable update operations and judgment.

There is also a POU named ETHERCAT.EtherCAT_Task, which is executed first by default. It can be considered as a bus communication POU processed by the system by default once the EtherCAT task is executed. It involves the PDO sending and receiving between the master and all slaves, as well as the update to each servo axis data structure.

3.1.3 Benefits of a User Program Consisting of Multiple POUs

Processing programs with different execution periods should be compiled in different POUs. For example, POUs executed based on the EtherCAT period, external interrupt program POUs, and POUs processed based on a 20-ms time period must be written separately.

To improve the readability of a program, you may use different POUs and name them straightforward

-12-

based on the control process sections, operational objects, and physical structural components.

For example, in C programming, you may create an independent POU for a repeatedly called processing program, so that the program can be easily called by your project and reused by other projects.

When multiple programmers collaborate on creating a program, they may write and commission the POUs of their respective process sections, and finally combine the POUs into a user program project.

The programming software InoProShop supports six programming languages. You may choose the most suitable language based on the type of processing logic. Generally, a POU can be written in only one programming language. If multiple programming languages are needed in a project, you may divide the program into multiple POUs.

3.1.4 How to Achieve Both Logic Control and Motion Control in User Program

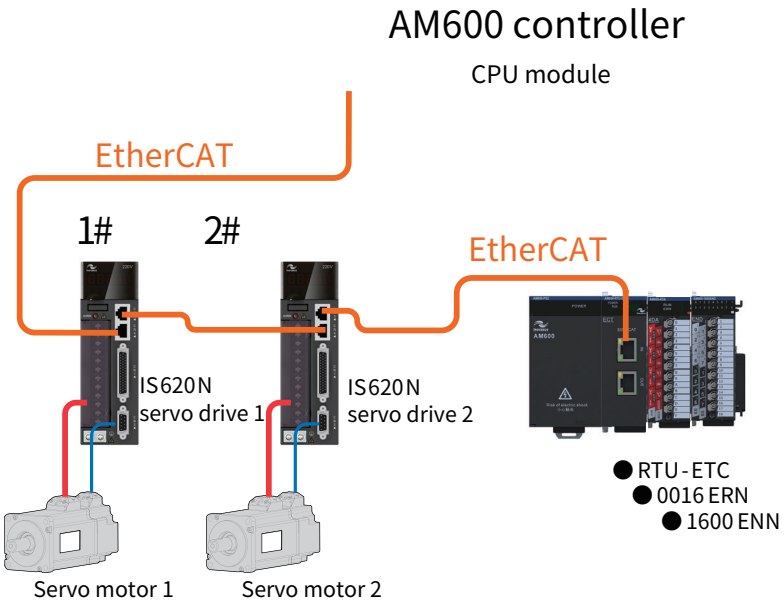
In an application system, synchronous control and trajectory control require higher timeliness compared with logic control. In the AM600 user program, you may group motion control (MC) POUs into the EtherCAT task group, and logic control POUs into general task groups. If specific program variables are declared as global variables, coordinated actions with logic control can be achieved in motion control.

For single-axis MC applications, where the control objects are the servo drive and motor, servo enable, homing, positioning control, velocity control, torque control, and stop and reset are required. For multi-axis synchronous MC applications, such as cam control and trajectory interpolation control, the PLC provides the corresponding MC function blocks to complete these operations. Therefore, function blocks are commonly used control commands in motion control programming, just like the use of prefabricated parts instead of gravel and cement in construction to improve construction efficiency.

A user program can control the execution trigger and termination of the function blocks based on the control logic of the application system. In addition, the user program can determine the execution status of the function blocks and determine whether there is an error. The PLCopen Specification defines the axis state data structures. The controller system establishes a corresponding data structure for each servo axis that has been configured by the user and automatically updates the status of the servo axes in time in each EtherCAT period. The user program can monitor the operation status of the servo axes by accessing the variables of the data structure and use the status variables as the basis for logic control, making it easy to achieve logic control and motion control in a user program.

3.2 Writing and Commissioning a Simple User Program

Before explaining the principle of the programming system and the methods for developing motion control programs, the following uses an example of a basic servo control program to give you a basic understanding of the programming process. In the following example, the application system consists of a CPU module, IS620N servo systems, and AM600-RTU-ECTA and 0016ERN expansion modules.



Assume that we need to write a simple program to achieve the following functions on the AM600 CPU controller:

Enable servo motor 1 to jog.

Every time the command flag is triggered, servo motor 2 runs for two revolutions and then stops, which is used to test whether the system is functioning normally.

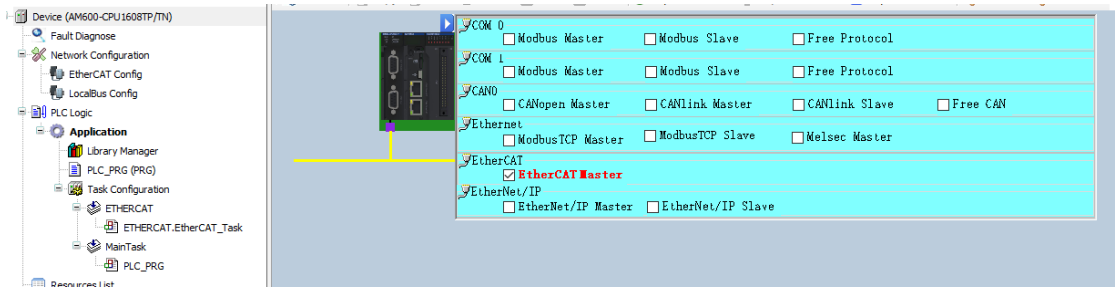
Perform marquee output through the I/O output port on the expansion rack, with the value bits circularly shifted by 1 bit from low to high every 0.5 seconds within the range of 16 bits.

The programming example involves the following approach and steps:

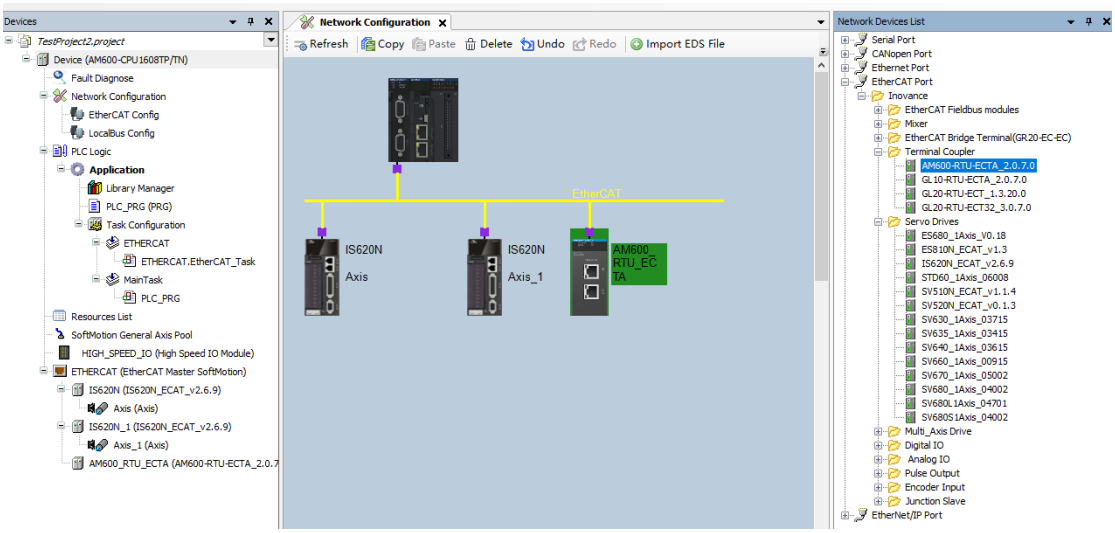
Motion control of the servo needs to be processed in the EtherCAT task period with high timeliness. The marquee control can be processed in the 20-ms task period as timeliness is not critical.

3.2.1 Creating a Project

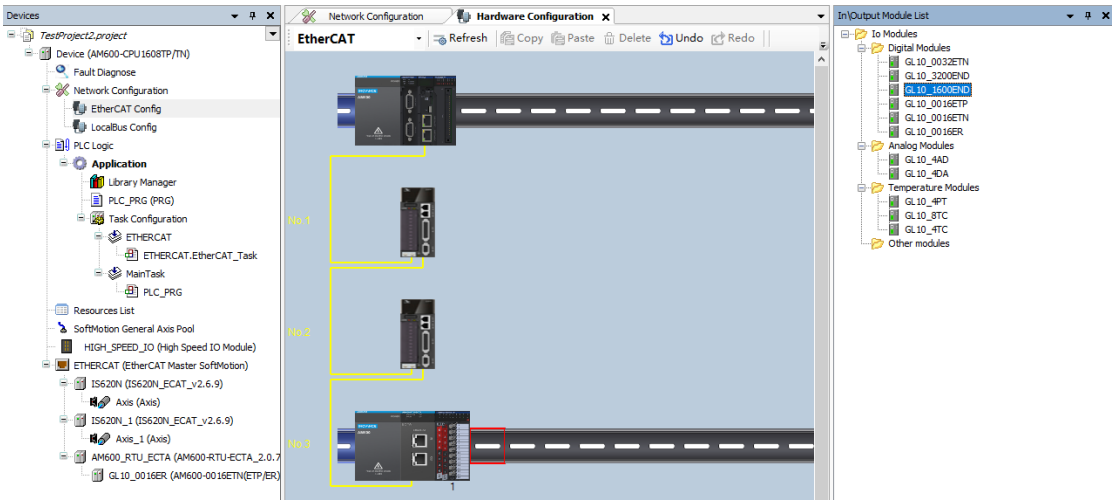
Run InoProShop and create a user project. On the screen shown below, double-click "Network Configuration" in the left pane to add the EtherCAT network bus.



According to the wiring sequence of the devices in the actual system, add two IS620N servos and one AM600-RTU-ECTA remote module (expansion rack) to the network, as shown in the following figure.



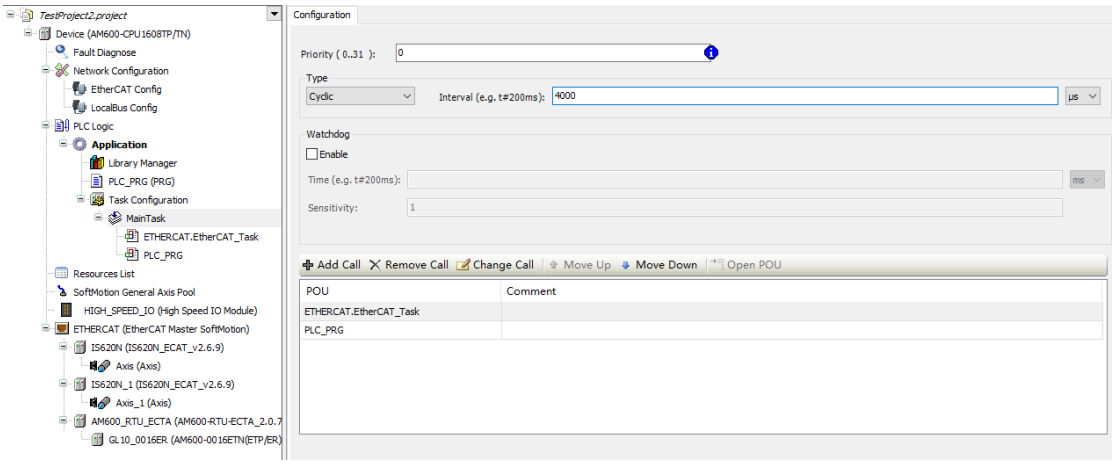
Double-click the RTU-ECT module to enter the expansion rack configuration screen. Add expansion I/O modules according to the actual wiring order, as shown in the following figure.



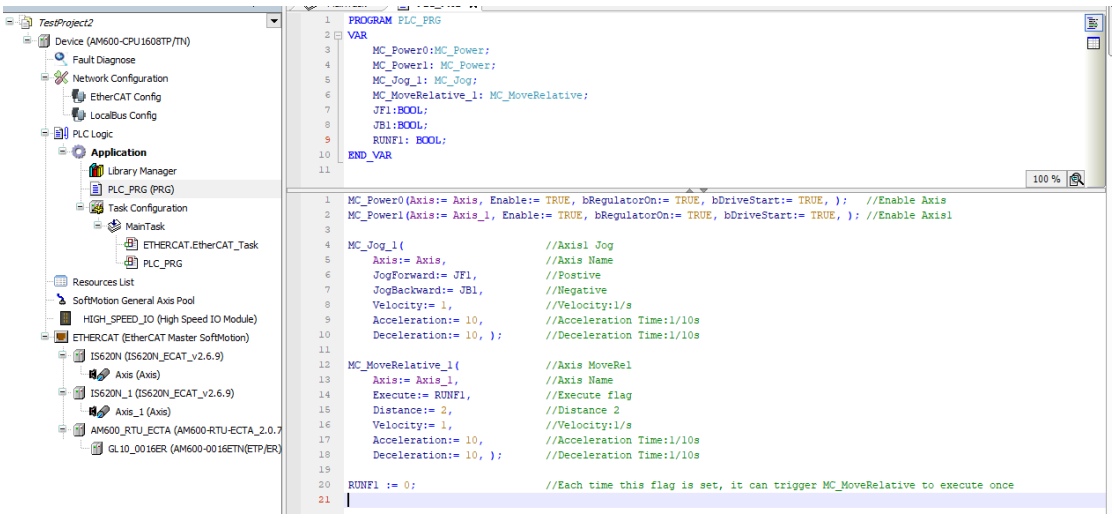
Now, we have completed the hardware configuration in the user project, which is consistent with the wiring in the actual application of AM600.

3.2.2 Writing POUs for Function Processing

Let's take a look at the default task configuration in the InoProShop programming environment. There is a MainTask task by default. Click on it and we can find that it contains a POU named EHERCAT.EtherCAT_Task, indicating that it is an EtherCAT task. There is another POU named PLC_PRG under this task, which was created when the project was created. We can write servo control program code in PLC_PRG.

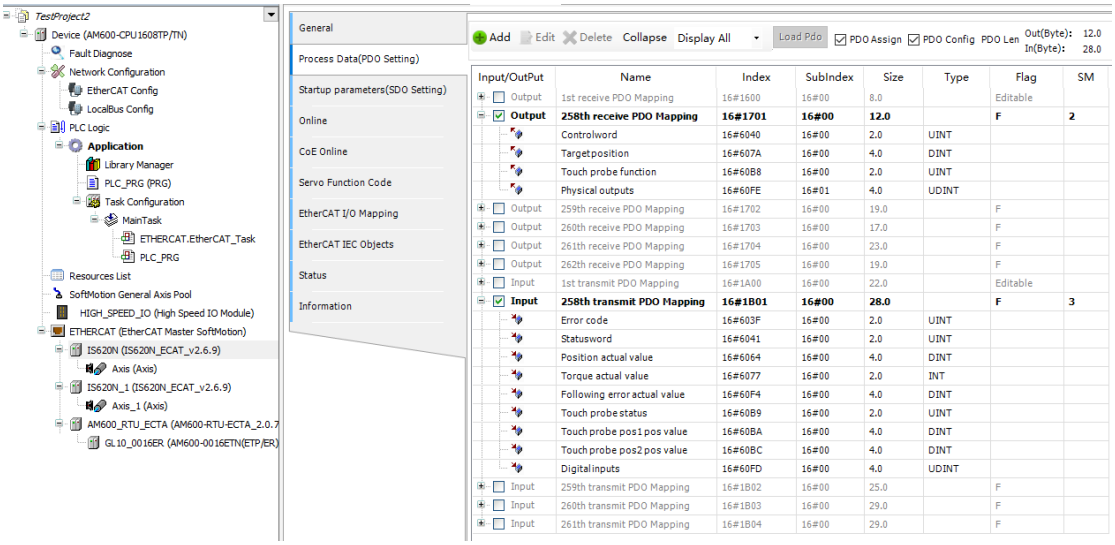


Double-click PLC_PRG in the left pane to enter the POU editing screen. Input the content shown in the following figure. To be concise, variables not used in the MC function block example are not displayed.



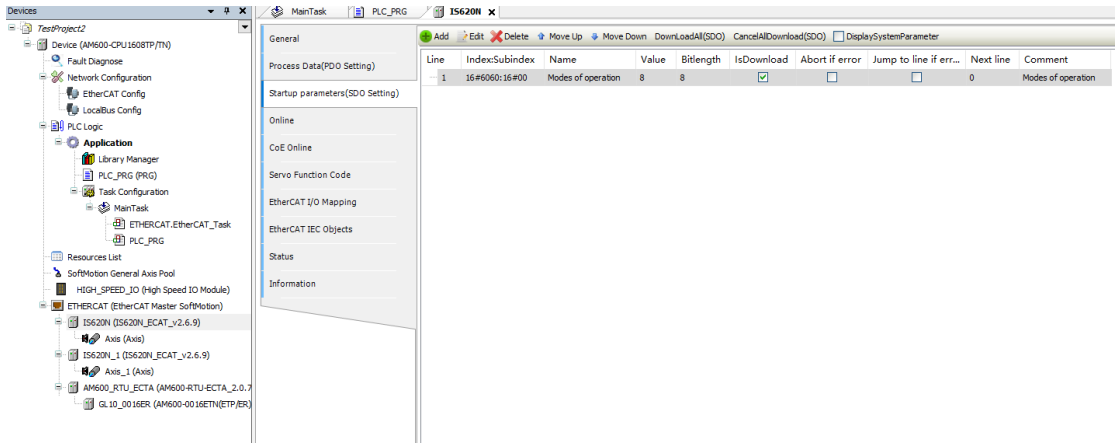
The servo trial run code shown above can make servo 1 jog and servo 2 run for two revolutions each time the RUNF1 flag is set.

To achieve this goal, we need to configure the EtherCAT master communication PDO based on the servo drive. The configuration items checked in the following figure are data for interaction during every EtherCAT communication.



Some items that only require AM600 to rewrite the servo function code are available in the SDO configuration, such as the electronic gear ratio and homing mode. The communication operates function

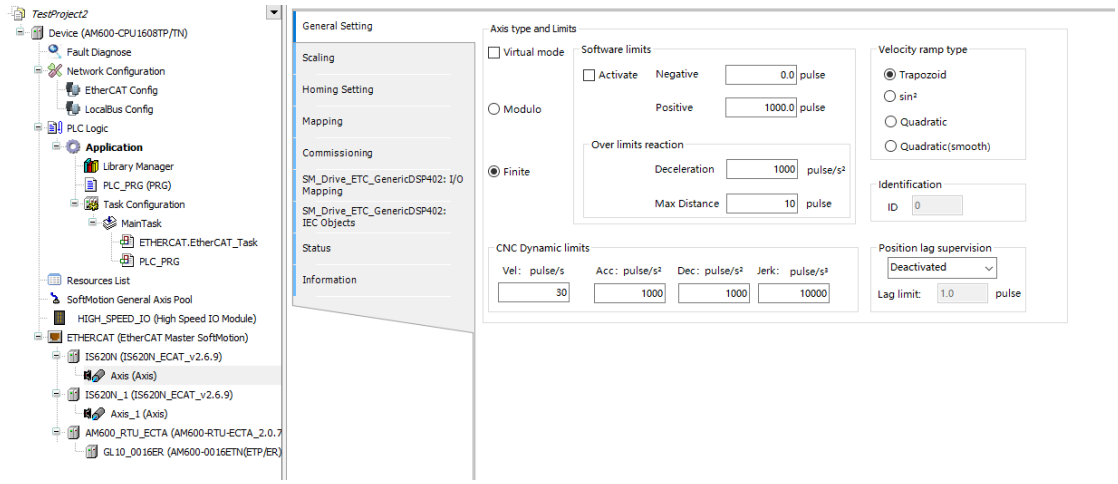
codes of the servo, and only one rewrite operation is carried out after power-on.



In the preceding figure, the servo operation mode is set to "Cyclic Synchronization Position Mode". Simply put, the AM600 controller calculates the position to reach in the next period (TargetPosition) in each EtherCAT task execution and sends it to the servo drive. The servo will complete the movement to the next target point based on the distance/time command.

3.2.3 Setting Motor Parameters

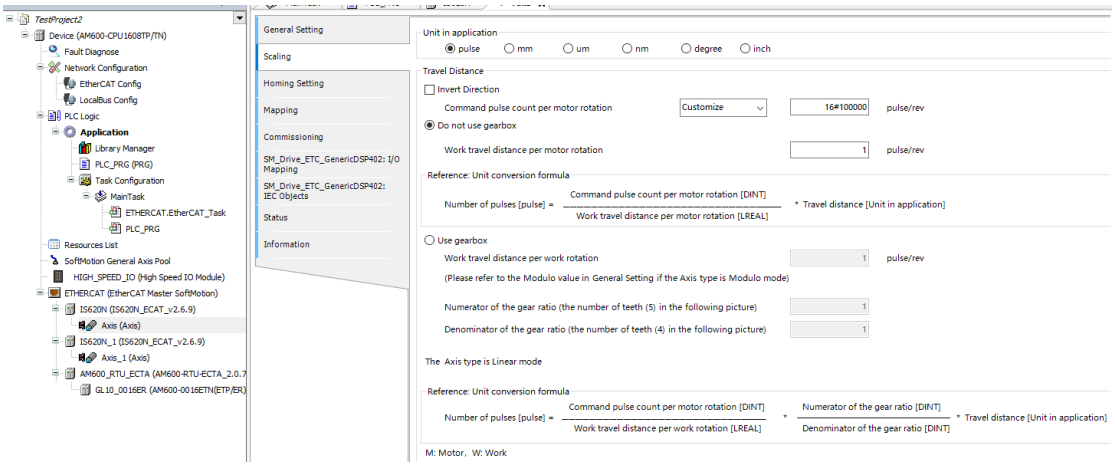
To accurately control the motion position, the controller must accurately calculate the position of the servo motor. Based on the operating and stroke characteristics of the application system, set the "Axis Type and Limitation" parameters for the controller to internally calculate the position based on feedback from the motor encoder. In this way, the controller can get the accurate position and avoid errors caused by the overflow of encoder pulses, as shown in the following figure.



For a screw type reciprocating mechanism with a limited stroke, we often need to know its absolute position within the range of the screw stroke. In this case, select "Linear Mode".

For a unidirectionally revolving axis, the linear mode is prone to position count overflow, resulting in position calculation errors. In this case, select "Cyclic Mode".

The encoder parameters (such as resolution) of the motor and the mechanical reduction ratio of the application system may vary. We need to set them based on the actual situation during programming, as shown in the following figure.

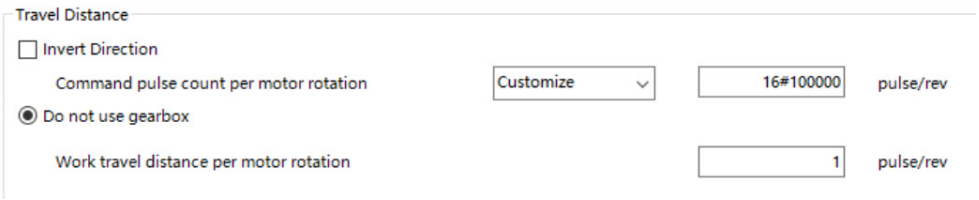


Motors used with the IS620N servo are available in two typical resolutions. For general incremental encoders, the resolution is 20 bits, indicating 1,048,576 pulses per revolution. For absolute encoders, the resolution is 23 bits, indicating 8,388,608 pulses per revolution. In actual operation, the controller sends the number of pulses required for operation to the servo drive by EtherCAT communication to control the servo operation. Therefore, the encoder resolution must be set according to the actual situation, as shown in the preceding figure.

For example, for a 20-bit encoder without a reducer, when the servo is commanded to run for 1 unit, the servo will select 1 revolution (axis motion for 360°).

If you set the "Applied Unit" parameter circled in the preceding figure to 360, when the servo is commanded to run for 1 unit, the servo will select 1/360 revolutions (axis motion for 1°). Similarly, after relevant parameters (commonly known as electronic gear ratios) are set based on the actual mechanical structure, the distance command can be input based on the physical travel distance unit of the application system. This makes the control parameters easy to understand.

Note that the parameters circled in the preceding figure can only be set to integers. The ratio of the parameters in the same row is a valid ratio value, and you can input appropriate integer values on the left and right sides of a row. For example, for a servo motor that drives a screw with a lead of 5.6 mm (that is, the screw slider moves for 5.6 mm when the screw rotates for 1 revolution) through a 4:1 mechanical reduction mechanism, set the parameters as follows:



The dimension of the parameter circled in the figure can be used as the parameter dimension of distance in the MC command.

The servo drive and motor parameters explained above must be set and verified in the corresponding items for both Axis and Axis_1. Otherwise, the desired operation characteristics cannot be achieved.

Example:

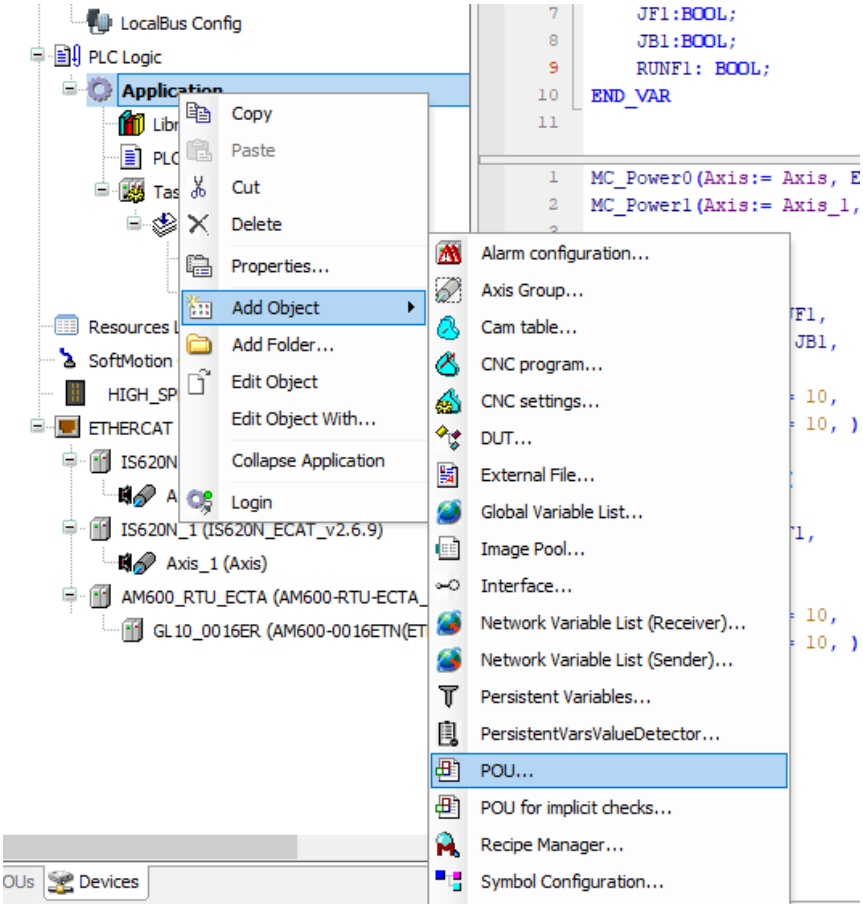
After the motor gear ratio is set, the statement

MC_MoveAbsolute(Axis1:=1, Distance:=80.00);//Command to move to the 80.00 mm position in the coordinates

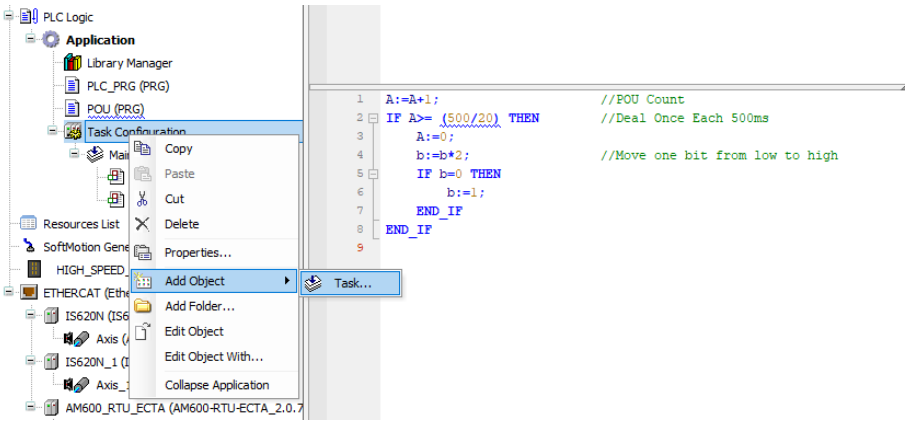
in the user program can make the workpiece move to the 80.00 mm position in the coordinates. The position command unit is the physical coordinate unit of the device, which facilitates commissioning.

3.2.4 Writing Marquee Control Logic

The logic control program of marquees has a lower requirement on timeliness than the motion control of servo axis. It only requires that the DO port changes twice per second. You can set a common task to execute the corresponding POU once every 20 ms to update bit shift. Add a POU first:



Write a program as follows in the POU and add another task:

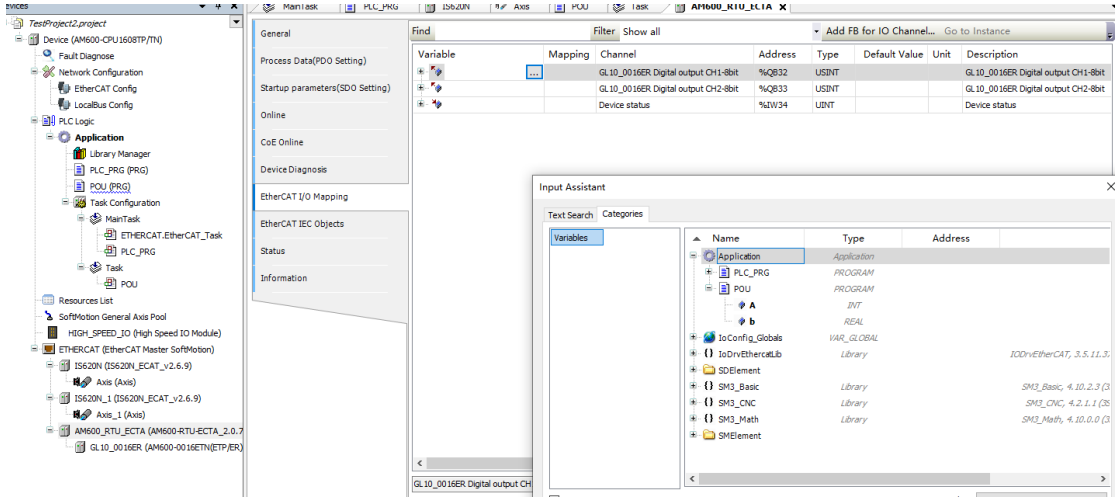


The above program is written in ST. As the POU is executed every 20 ms, we use variable A as the number of times of POU execution, and multiply variable b by two every 25 times of execution (500/20 in the program), that is, shift the binary value by one bit from low to high. Send variable b to the marquee output port, and we can achieve the marquee effect.

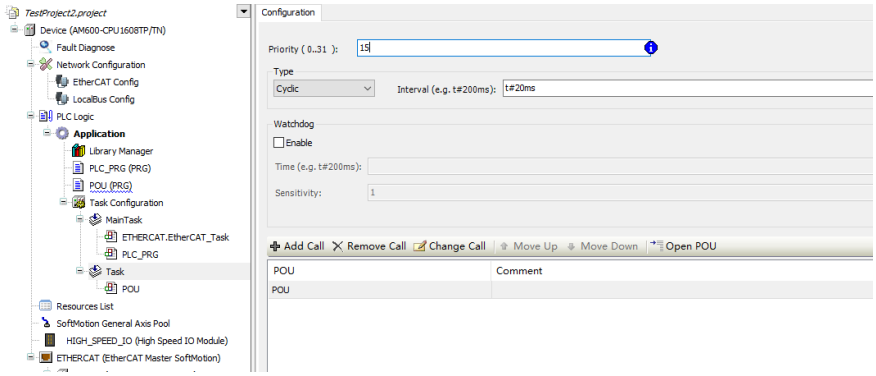
3.2.5 Associating a Variable with the Hardware Output Port

According to the previous requirements, associate variable b in the POU program with the I/O module port in the expansion RTU-ETC rack. Specifically, select the I/O module in the application system

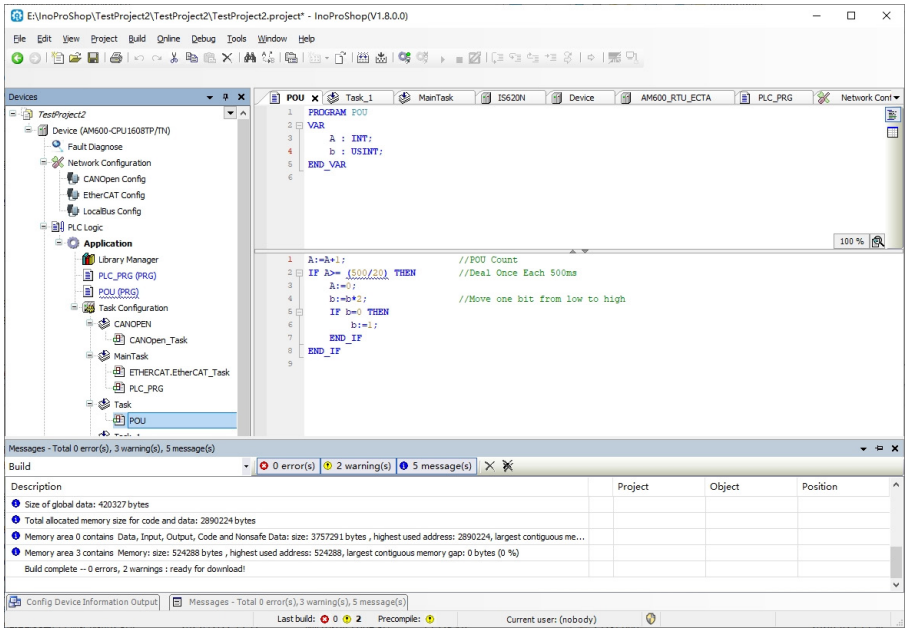
network, select the I/O port, and specify the variable of the POU program in its I/O mapping. User-written program variables are selected in the POU under "Application\", as shown in the following figure.



Assign the marquee POU to the new task (Task) and configure task execution. Set the priority to a routine priority (such as 15) and set the execution interval to 20 ms.



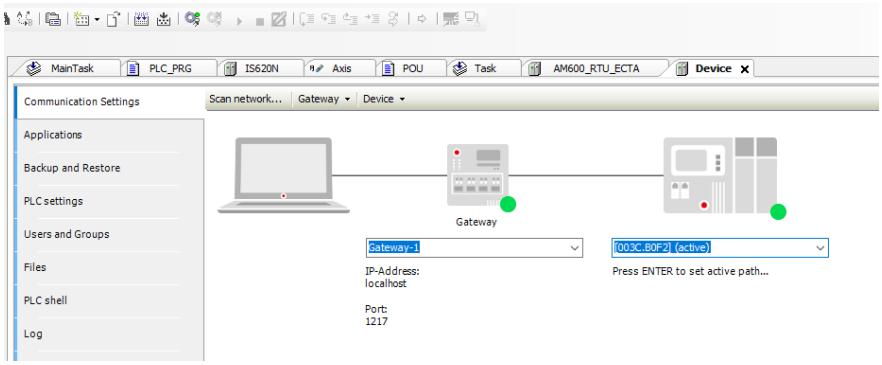
3.2.6 Troubleshooting User Program Compilation



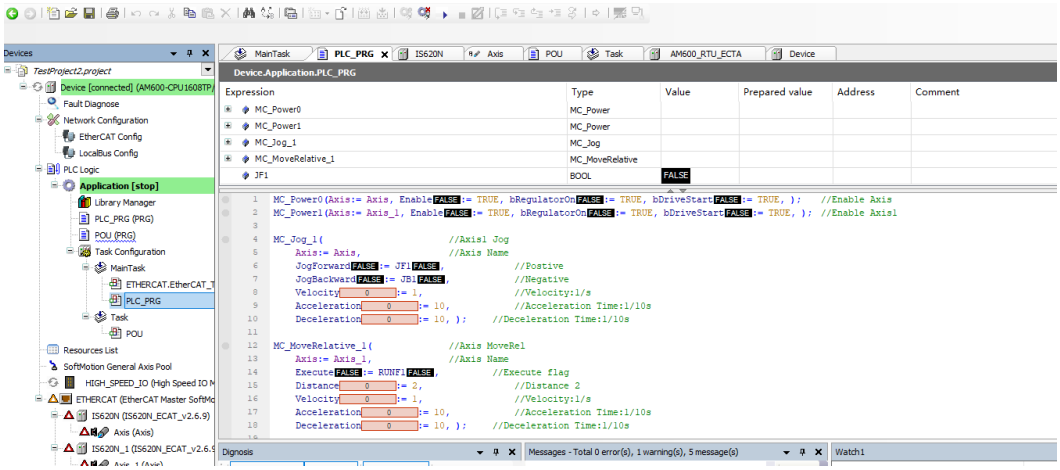
If there are compilation errors, the error type and reason will be displayed, as shown in the preceding figure. After you double-click the error description, the cursor will go to the program editing window for you to make corrections. After dealing with the errors one by one, compile the program until all

compilation errors are rectified.

Finally, download the user program to the AM600 CPU module.

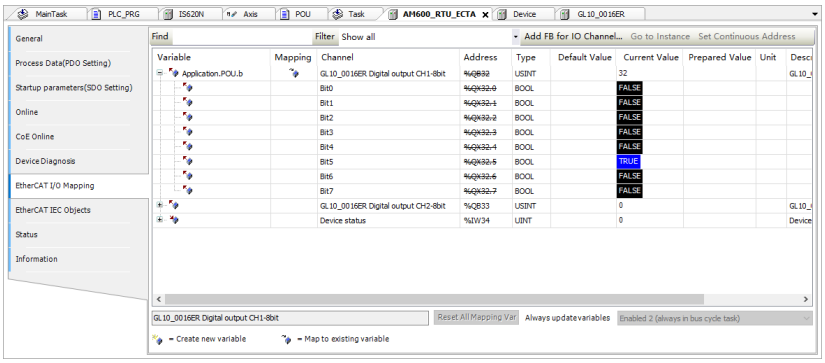


3.2.7 Monitoring the Running of the User Program



In the monitoring screen, you can observe the execution of the program. For example, setting variable JF1 to 1 will make axis 1 jog, and resetting it to 0 will make axis 1 stop. Every time variable RUNF1 is forcibly set to 1, axis 2 will rotate for 2 revolutions.

In the RTU-ETC expansion module screen, you can see the I/O output port is in marquee switching state.



Now, the functions of servo jogging and rotating for 2 revolutions have been realized in programming. A simple programming process is completed.

3.2.8 Summary of Typical Steps of Writing a Motion Control Project

According to the above example, writing a user program with MC function generally involves the following aspects.

Application system hardware configuration: Configure network parameters based on the master controller, expansion module, network type, servo slave, and so on.

User program writing: Based on the control functions to be achieved, write in a POU (such as POU 1) for motion control and in another POU (such as POU 2) for general logic control.

Servo drive parameter configuration: Configure the SDOs and PDOs based on the servo name and servo operation mode in the hardware configuration. Ensure that the required communication objects between the MC function blocks of the user program and the servo are configured in the configuration table.

Servo motor parameter configuration: Accurately configure the encoder resolution of the servo motor and the transmission ratio and axis motion range of the mechanical structure, so that the control object displacement instruction precisely matches the actual displacement.

Task arrangement: According to the timeliness requirement of the control, execute the motion control function POU 1 in the EtherCAT task, and set the interval to 2 ms and the priority to 0. Execute the general logic control POU 2 in a general task, and set the interval to 20 ms and the priority to 16.

Online commissioning: Connect the AM600 controller to the PC through the LAN. Then, power on the device and download the user program for commissioning. If possible, connect the servo drive system to the AM600 controller and then perform commissioning. If no servo system is available, you can set the servo as a virtual axis. If no AM600 controller is available, you can simulate and debug the user program on the PC. Eliminate possible errors in the user program until the expectation is reached.

Memo No. _____

Date / /



4. Execution Mechanism of the Motion Control Program

4. Execution Mechanism of the Motion Control Program

4.1 Task and Configuration in the User Project

As shown in the preceding example, you can set the execution trigger conditions, execution interval, and execution priority for each task group. The medium-sized PLC supports the following task types:

Task Execution Type	Execution Characteristics	Task Example
Cyclic	The POU is executed once at each configured interval.	EtherCAT bus task CANopen bus task Common cyclic tasks
External event	The corresponding POU is executed once when the HSIO status changes or the high-speed counter readings match.	HSIO port status interrupt response task HSIO port counter interrupt response task
Inertial slide (flywheel)	The task is executed circularly and continuously once the execution is started.	Common cyclic tasks
Event	Task execution is triggered once under the preset state 0 1 of the Boolean variable, but not under other state combinations such as 0 0, 1 1, and 1 0.	Soft interrupt handling POU
State	The task is executed circularly under the preset state 1 of the Boolean variable.	Conditional execution task POU

1) Task configuration notes

Set the task type to "Cyclic". "Task Period" refers to the interval for executing this type of task. For general logic control with slow variable changes of the common I/O ports, the task period can be relatively long, for example, 20 ms. For tasks that need to be processed in a timely manner, the task period can be set to a smaller value.

EtherCAT bus communication is a special "cyclic" task and has the top priority. The setpoint of the task period will also be the communication period of the EtherCAT bus, which generally ranges from 1 ms to 4 ms. A smaller setpoint indicates higher precision of motion control. When more axes are to be controlled, a longer period is required. Otherwise, the CPU may be overloaded with calculation.

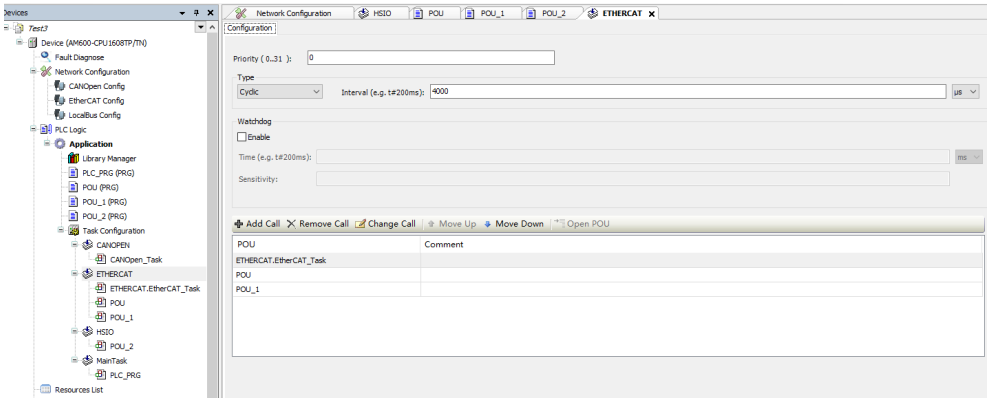
Similarly, CANopen bus communication is another special "cyclic" task and has the second highest priority. The setpoint of the task period will also be the communication period of the CANopen bus, which generally ranges from 4 ms to 8 ms. A smaller setpoint indicates higher precision of motion control. When more axes are to be controlled, the communication duration is longer, and a longer period is required.

Some tasks are executed only when certain statuses are met. For example, task execution is triggered by a status change of the HSIO port, also known as the HSIO interrupt signal, rather than by interval.

For a task configuration, you can set only one execution type, interval, and priority. If you want different execution characteristics, add multiple task configurations.

One task configuration can include multiple POUs, which will be executed at the same interval and in the order that the POUs are added to the task.

As shown in the following figure, there are 4 tasks under "Task Configuration" in InoProShop. Double-click an existing task. You can see the configured parameters of the task in the right window.



You can add a task by selecting "Task Configuration" and right-clicking.

Note that the task with the ETHERCAT.EtherCAT_Task project in the picture above will be an EtherCAT bus task and its task priority should be set to 0.

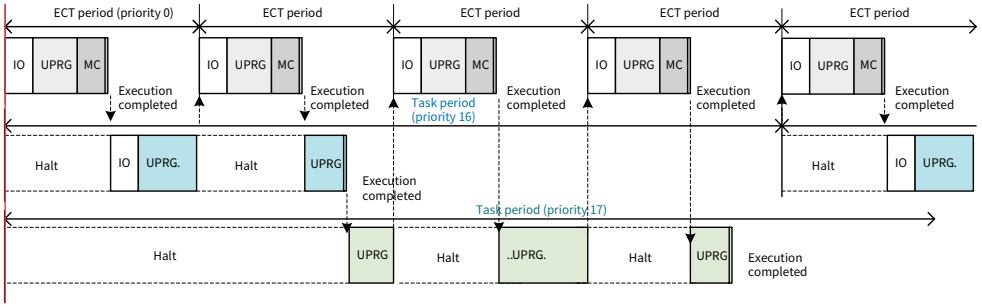
2) Task prioritization

By default, the system software of the medium-sized PLC assigns different priorities for different types of task configurations. This ensures that important tasks, such as motion control, are executed with priority and the controller can be reasonably used in applications that require high-performance MC.

The task priority order is as follows (do not forcibly modify the priority order):

Default Priority	Task Type	Description
0	EtherCAT bus task	Top priority. Only one EtherCAT task is allowed.
1	HSIO interrupt task	Second highest priority. One HSIO task is allowed for each HSIO input port event.
2	CANopen interrupt task	Third highest priority. Only one CANopen task is allowed.
3	ModbusTCP	
4	ModbusRTU	
16	MainPOU	Lowest priority. Up to 4 tasks with different periods are allowed.

A smaller priority setpoint indicates a higher priority. High-priority POUs can interrupt the execution of low-priority POUs, as shown in the following figure, where ECT stands for EtherCAT.



From the preceding figure, we can find that:

When the controller executes tasks, there is a time alignment point that is invisible to users, as shown on the left side of the preceding figure. From this time point, the tasks are executed from top priority to the lowest priority.

The execution of a low-priority task may be interrupted by a high-priority task. After the high-priority task has been executed, the low-priority task interrupted previously will be resumed.

An EtherCAT task has the top priority. When this task is started based on the EtherCAT period, all POUs under the task will be executed once before low-priority tasks are executed.

3) Requirements of execution period setting in task configuration

The medium-sized PLC system software uses a multi-tasking approach to execute "tasks" of the user program and assigns an execution period for each "task". Some global variables may be accessed and modified for different POUs. Therefore, global variables must be synchronized, which can be performed at the "time alignment point" of the task. The periods of cyclic tasks are set in integer multiples.

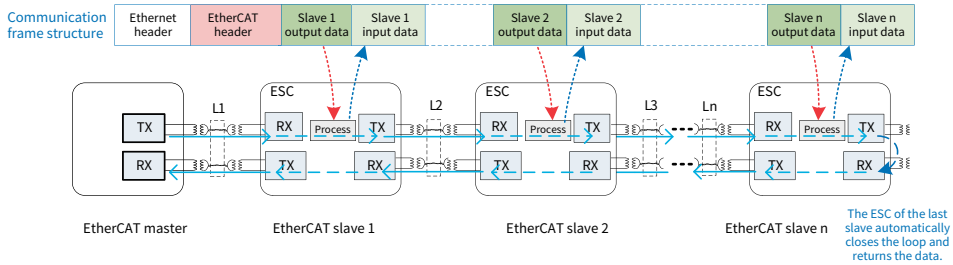
For example, set the EtherCAT period to 1 ms, 2 ms, or 4 ms, the period of a general cyclic task to 20 ms, and the period of a lower-priority cyclic task to 100 ms. Do not set the EtherCAT period to 3 ms, 6 ms, 7 ms, or 9 ms, as this tends to result in a non-integral multiple relationship.

4.2 Dataflow Analysis of the EtherCAT Bus Network

1) Introduction to the EtherCAT bus network

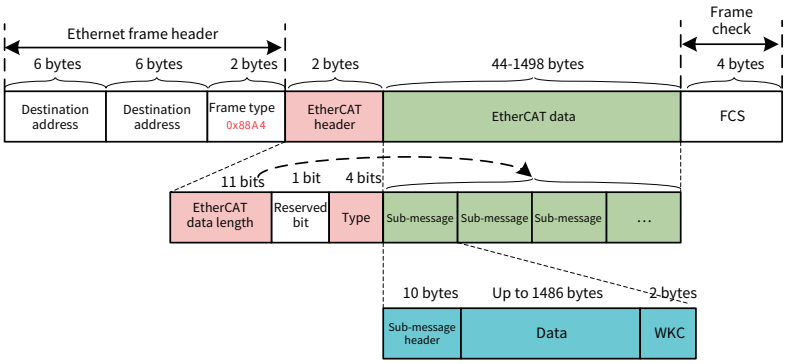
Generally, the EtherCAT bus uses a RJ45 socket and a multi-core Ethernet cable. A Cat5e cable is recommended as it can improve the antijamming ability of the network.

Similar to a general-purpose Ethernet network, the EtherCAT bus network features a communication rate of 100 Mbps, and the cable length of each neighboring slave can be up to 100 meters. The following figure shows the equivalent connection relationships within the network and the communication dataflow.



Different from a general Ethernet, the EtherCAT network allows only one EtherCAT master. In addition, the EtherCAT slave controller (ESC), which is a dedicated network control chip inside the slave, receives communication data in real time in a communication frame and inserts response data into the data frame. This enables the master to access multiple slaves in just one communication frame, which greatly improves communication efficiency.

The communication data frame in the EtherCAT bus uses the UDP/IP frame structure of Ethernet data and frame structure type 0x88A4, except that the data fields in the middle need to be prepared and parsed according to the EtherCAT protocol, as shown in the following figure.



The EtherCAT data fields can be further defined and parsed by "EtherCAT frame" according to a certain protocol. As long as the master and the slave comply with this protocol, data communication can be achieved. Generally, the CANopen Over EtherCAT (CoE) and Sercos Over EtherCAT (SoE) protocols are used, just like the transmission of Modbus protocol frame data (ModbusTCP) over a TCP/IP network.

The medium-sized PLC uses the CoE protocol, which is the DS402 industry standard (also known as CiA402) whose application layer protocol is the CANopen protocol. It is a dedicated protocol for servo motion control, with the following highlights:

1) For high communication efficiency, the master and slave do not use the request-response manner for communication. Instead, in the initialization phase of the bus network, the master gives the slave a list of data items to be sent, such as "PDOs", informing the slave of the data items the master will send and their order (TPDO), as well as the data items the slave is required to send and their order (RPDO). In this way, the slave knows how to parse the master's data frames when receiving them, and can prepare the required response data.

When the master's data frames arrive, the network control chip (ESC) of each slave can obtain the corresponding data segments for the slave's processor to parse according to the configuration table and insert a response data block at the appropriate stage of the EtherCAT communication frame to return to the master.

2) According to the timeliness requirements, the communication data is categorized into "process data (PDO)", which is scheduled to be sent and received cyclically at a high frequency, and "service data (SDO)", which is exchanged only when needed.

3) A servo drive can have as many as hundreds of control command parameters, operation status parameters, and function code setting parameters. The parameters are named in different ways depending on servo drive brands. To ensure the interchangeability of different brands of masters and slaves, the CiA402 protocol provides an "object dictionary (OD)", which defines all function codes, operation commands and their setpoint meaning, as well as operation status parameters and dimensions to be used in servo drives. The CiA402 protocol ensures the universality and interchangeability of products developed by different suppliers so that the products can work with the medium-sized PLC.

4) The configuration of communication objects between the master and slave is a prerequisite for ensuring the successful execution of the MC function blocks. When executing the MC function blocks in the user program, the controller needs to use specific "communication data objects" to send commands to the servo slave and read the slave axis status. Programmers should configure the required data objects in the TPDO and RPDO so that the master controller can control the servo slave.

5) The slave device may not support all the item definitions in the "object dictionary (OD)", but the "device description file (EDS)" from the device manufacturer defines the objects. Programmers need to import the EDS of the slave device in InoProShop to know the supported objects before configuring the device.

6) When writing a user project, users select and configure the TPDO and RPDO tables based on the control needs. During operation, the master will automatically forward the data specified by the data object tables to the corresponding slave through communication. Select only the necessary configuration items to reduce the load of EtherCAT communication and improve the communication efficiency.

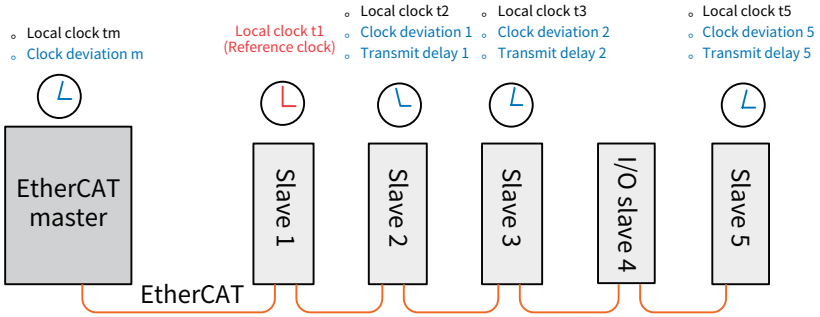
7) The service data object (SDO) configuration items are typically used to initialize function codes of the slave device at the beginning of system operation, and access parameters through function blocks such as MC_SDOread during the operation process. SDO communication features relatively low timeliness and takes up additional EtherCAT communication overhead, and even causes synchronization timeout faults in applications with a high bus loading rate. Therefore, exercise caution when using these configuration items.

After understanding the CiA402 OD and the slave parameter objects commonly used by MC function blocks, you can reasonably configure the PDO and SDO tables.

2) Clock synchronization for the EtherCAT bus

Typically, a network for multi-axis motion control needs to make multiple slaves start or stop moving

synchronously. The EtherCAT network has a distributed clock (DC) mechanism, which allows all the intelligent slaves (such as servo drives and intelligent high-speed expansion modules) to have a consistent clock. Based on the configured synchronization trigger period, each slave will output the data written by the master to the execution unit to achieve synchronization.



As shown in the preceding figure, each intelligent slave has a high-resolution internal local clock (TLocal). In the initialization phase of the EtherCAT bus, the master reads the current time of each slave and takes the local time of the first slave as the "reference clock" of the network. In this way, the master can calculate the "clock offset (Toffset)" of each slave relative to the reference clock and write the clock offset to the corresponding slave for the slave to correct the clock and eliminate the static error.

In addition, in the process of transmitting communication data frames, there are transmission delays due to the hardware network. To resolve this issue, the master sends a specific broadcast frame to make each slave record the moment of data arrival. Then, the master reads the value of the moment recorded by each slave and measures the total delay of returning data frames to accurately calculate the "transmission delay (Tdelay)" of each slave. Afterwards, the master writes the transmission delay time of each slave into the memory of the corresponding slave. With these clock correction values, the slave can get the same clock as the reference clock t1 through calculation according to the formula of TLocal — Toffset — Tdelay.

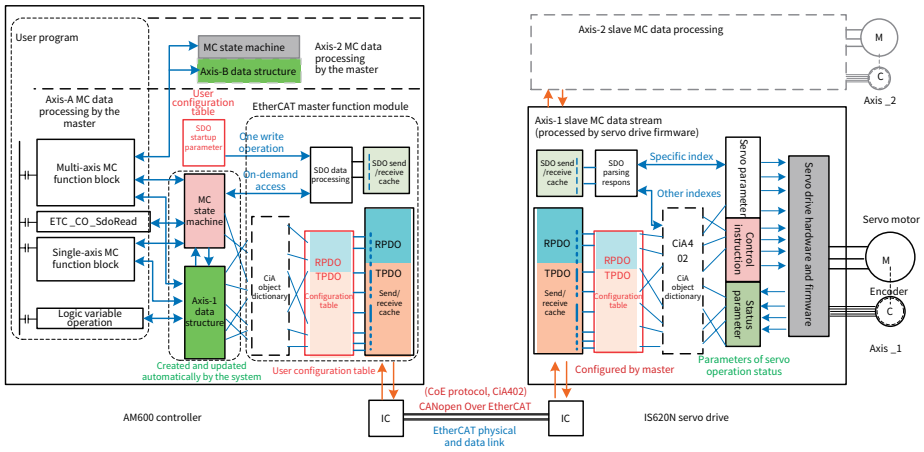
In an EtherCAT network, DC processing can be skipped for I/O slaves that are not sensitive to the DC clock. The EtherCAT master ignores the clock calibration for such I/O slaves during DC calibration.

Each slave ESC chip has a synchronization pulse width register. A synchronization unit, once activated, regularly generates SYNC signals to validate the currently received data. For the servo drive, the received position command is regarded as the target point to start execution.

The DC initialization and calibration of the EtherCAT network slave described above are performed automatically by the EtherCAT master without user intervention. When the EtherCAT bus is ready, it indicates that the DC initialization has been completed. Note that the slave with the internal clock function should be arranged at the front end of the network if possible.

4.3 Data Process for Communication with Servo Slaves

As mentioned earlier, in the EtherCAT communication of the medium-sized PLC, the application layer uses CoE. When the controller executes the MC user program, the communication data between the controller system software and the servo is processed through multiple levels of functional units. The process is shown in the following figure.



The figure shows the AM600 controller on the left side and the IS620N series servo drive and its supporting servo motor on the right side. Network interfaces of the controller and the drive comply with the PLCopen Specification so that both are compatible with third-party devices and can be used interchangeably. In this way, the internal communication data process is applicable to third-party devices complying with the CiA402 protocol. With an understanding of the function and usage of each function block, we know the principle and approach of EtherCAT network-based motion control.

The source of MC instructions is the MC function block in the user program. The object of control operation is the servo axis, and the status of the MC axis is stored in the content of the master controller in the form of "axis data structure" for access by the user program.

4.3.1 Control Information Process

Step 1: Execute the MC function block of the user program and process the command data to be sent.

When executing the user program, the controller executes the MC function block instance such as MC_MoveRelative (Axis_1). Based on the state machine and data structure of the slave (Axis_1) in the memory, the controller:

Checks the current state of the slave axis, and reports an MC execution error if the slave axis is not enabled, is running in torque mode, is running in synchronization mode, is in homing operation, or is generating an alarm.

Sends a command (ControlWord) to make the slave axis run if the slave axis is stopped, or is running in non-synchronized position mode.

Analyzes the current running position (fActPosition), running velocity (fActVelocity), and constraints of the slave axis such as target position, maximum allowable velocity, acceleration, and deceleration, and calculates the required motion position instruction (TargetPosition) for the next operation period.

Waits for the data returned from the slave in the next communication period to analyze the instruction execution of the MC function block. This enables users to know whether the instruction is being executed (Busy), has been executed (Done), has an error (Error), is interrupted by other MC instructions (Aborted), or is waiting for execution (Buffered).

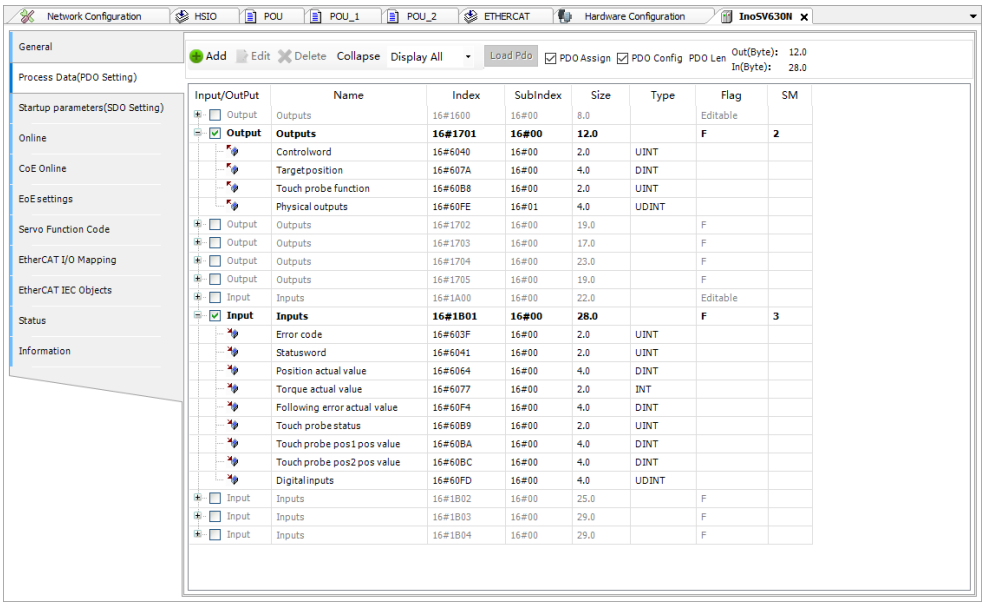
Step 2: Place the control command data to be sent into the EtherCAT transmit cache unit.

The command data (ControlWord and TargetPosition) to be sent to slave Axis_1 is stored in the PDO transmit cache unit. This operation requires that these two parameters ("objects" in the CiA402) are available in the PDO configuration table.

The PDO configuration table stores the "index number" (main index number: sub-index number) of each control parameter to be sent and read by the master.

	Purpose
TPDO configuration table	This table contains a list of objects and attributes that need to be configured by the user during programming based on the content that need to be sent cyclically for slave control. This table is automatically sent by the controller to the slave ESC at the network initialization phase. The controller master determines the size of the transmit cache according to this table and stores the command data to be sent into the transmit cache during operation. The slave parses the received data frames according to this table during operation. The TPDO configuration table can vary by slave.
RPDO configuration table	This table contains a list of objects and attributes that need to be configured by the user during programming based on the content to be responded automatically by the slave. This table is automatically sent to the slave ESC at the network initialization phase. During operation, the slave prepares data according to this table and returns the data to the master by inserting the data into the time slot of the EtherCAT data frame when the master accesses the slave. During operation, the master parses the slave's response data in the returned data frame according to this table. The RPDO configuration table can vary by slave.

The following figure shows the PDO configuration table in InoProShop. The index number and data type of each control parameter are specified by CiA402. The "index number" enables you to look up the parameter and its width type in the "object dictionary (OD)".



At the initialization phase, the master sends the "PDO configuration table" to the slave. The table contains the TPDO, RPDO, and information such as data type and width of each object, providing the basis for the slave to parse data frames.

The TPDO configuration table stores the index number and data width information of each object. The object storage order in the table provides a basis for the system to put data to be sent through MC instructions into the transmit cache unit. As shown in the preceding figure, ControlWord is placed in the first transmit unit, TargetPosition is placed in the second unit, and so on.

According to the RPDO configuration table (such as the nine "objects" in the preceding figure), the slave stores the servo operation status data in the response cache unit based on the index number and order of each object. When the master communication frame accesses the slave, the ESC automatically inserts the data in the cache unit to the appropriate time slot of the data frame and returns it to the master.

The RPDO table also provides a basis for the master to parse response data from the slave.

Step 3: The master control chip sends the data in the transmit cache unit to the slave ESC regularly, and the slave simultaneously sends the response data.

As the master, the controller generates an EtherCAT interrupt command according to the EtherCAT clock period set by the user. After entering the EtherCAT interrupt status, the master initiates EtherCAT communication, sends the data from the PDO transmit cache unit to several slaves through one or several frames, and retrieves response data from slaves in the same communication frame.

Chronologically, the data from the cache unit of the controller is the command data generated from the previous EtherCAT interrupt POU execution.

The slave's response data is not a reply to the master's query, but the current value of the "object" based on the cyclic reply required in the RPDO configuration.

Step 4: The slave receives and parses the data sent from the master.

When the network works normally, the slave ESC receives the communication data frames sent by the master regularly, and automatically stores the data in the communication frames into the local cache.

		ControlWord	TargetPosition	TargetVelocity	TargetTorque	Modes of operation	Touchprobe function	Maxprofile velocity		Error code	Status Word
Master TPDO configuration table	Object Index No.	6040: 00	607A: 00	60FF: 00	6071: 00	6060: 00	60B8: 00	607F: 00	RPDO configuration table	603F: 00	6041: 00
	Type	UINT	DINT	DINT	INT	SINT	UINT	UDINT		UINT	UINT
	Data Width	16bit	32bit	32bit	16bit	8bit	16bit	32bit		16bit	16bit
Slave 1 parses the received data string.		F1A4	5 B32 05C7	01 F2 3201	01 E2	08	04 A0	1008 FA11	(Response data)	4 C87	5 DF1

After receiving a string of PDO data, the processor of the slave extracts the received data string according to the object data type (width) specified in the TPDO table, and stores it into the control command unit based on the parameter attributes represented by the "object index" number for servo operation control.

Based on the object attributes and sequence specified in the RPDO configuration table, the processor of the slave refreshes the response cache unit in the local ESC cyclically with the current operation status and parameters of the servo axis. At an appropriate time slot, the ESC inserts the cached data into the EtherCAT communication frame through a high-speed hardware operation and "sends" it to the master.

Step 5: The master receives and parses the data returned by the slave, updates the axis status parameters, and determines whether execution is complete.

The controller, as the master in the EtherCAT network, sends data frames and at the same time receives communication frames sent back from the slave network in closed loop. From these frames, the controller extracts data strings returned by the slaves, determines the communication status of the network, and analyzes whether the communication operation is successful.

Based on the data received from the slaves, such as Error code, Status Word, and Position Actual Value, the controller system can determine whether the required operation position of the MC function block instance has been reached, and refresh the status of the output variables of the MC function block instance.

In addition, the controller system software updates the data structure of the axis status parameter in time for access by the user program. This is one of the most powerful intrinsic functions of the medium-sized PLC software.

To summarize, this section describes the principle of sending, receiving, and parsing EtherCAT data array packets for the medium-sized PLC. Most of the steps are automatically performed by the system, and users only need to understand the concept of CiA402 objects, master the common "object" types of the servo axis, and select objects for the TPDO and RPDO configuration tables.

4.3.2 CiA402 Data Object Dictionary and Common Objects for Servo Drives

The application layer on the EtherCAT bus of the medium-sized PLC adopts the CANopen Over EtherCAT (CoE) protocol.

CANopen is a common protocol standard, which defines different series of "industry standards" for communication control of different types of devices, as listed below:

CiA401 for I/O modules

CiA402 for servo and motion control

CiA403 for human-machine interfaces

CiA404 for measuring devices and closed-loop control

CiA406 for encoders

CiA408 for proportional hydraulics

The EtherCAT bus communication application layer of the medium-sized PLC adopts the CANopen DS402 (CiA402) protocol, which is the "Servo and Motion Control" Industry Standard of the CANopen protocol. CiA402 is widely used in the motion control based on the CAN bus and EtherCAT bus network. Controllers and servo drives (slave devices) developed by different manufacturers in accordance with CiA402 can work collaboratively or be used interchangeably. This provides users with more options in line with the purpose of the PLCopen Specification.

The core of CiA402 includes the following parts:

Definition of the "object dictionary (OD)" and functional attributes of its "objects", which standardizes the communication data parsing approaches.

Periodic process communication data. The process object configuration is sent first, and then the object parameters are sent periodically according to the configured frame structure.

Occasional data communication, which uses additional communication fields for request-response communication.

The network communication has several operation statuses, which is convenient for the master and slave to perform initialization for communication, diagnose the causes of communication exceptions, and restore network communication.

CiA402 summarizes representative setup parameters, control parameters, and status parameters into "objects" with fixed numbers (index number+sub-index number). A complete object definition table is an "object dictionary".

Similar definition methods are used for other devices, such as BFM area address definition for PLC modules and function code definition for motor drives. All these methods specify different numbers for function parameters, facilitating understanding.

CiA402 object types are divided into the following index number segments by attribute.

Main Index Number Segment	Meaning	Description
0x0000 to 0x1FFF	Protocol type description, manufacturer information, industry standard type description, configuration table description, and so on.	Information is initialized by the manufacturer, and the configuration is done automatically by the system software.
0x2000 to 0x5FFF	Objects and their functional attributes defined by the manufacturer	The manufacturer can design the main index number as the function code of the servo drive, which is used to set the function code parameters and static parameters.
0x6000 to 0x9FFF	Data objects defined by industry standard, used for device control and monitoring	Communication data between the controller and servo for control

0xA000 to 0xFFFF	Reserved	

The preceding table shows that the objects required for motion control are in the index number segment from 0x6000 to 0x9FFF. If you want to modify the servo function code in SDO configuration, pay attention to the index number segment from 0x2000 to 0x5FFF.

To facilitate understanding, we regard the object dictionary as a set of servo drive function code definitions that can be accessed by EtherCAT bus communication.

Data objects commonly used in motion control applications:

Roughly, the controller controls the operation of the servo with the following types of commands:

Commands that control the servo operation status, such as enable, homing, start/stop, and alarm reset

Commands that set the server operation mode, such as position mode, velocity mode, and torque mode

Commands that set the target position, running velocity, and output torque for servo operation

Commands that read the operation information of the servo system, such as operation status, operation mode, position, current velocity, and output torque

Commands that set or modify the function code parameters of the servo system, the operation constraint parameters, and so on

To complete these control operations, users must set several commonly used data objects in the PDO or SDO configuration table during programming. Some data objects can be added based on the functions needed in the user program.

The values of the data objects introduced in this section are used for explaining the function definition of the objects. During actual operation, controller automatically sends the value based on the required control operation.

For the PDO configuration table, users only need to add the data objects required by the controller during operation, and do not need to fill in the specific parameter values or variable names. During compilation, InoProShop automatically associates the variables in the MC function block with the PDOs.

The SDO configuration table is generally used for the controller to initialize the servo function codes (write operation). The write value is a defined constant value. Therefore, the constants must comply with the DS402 specification. Some constants are defined based on the specific internal function codes of the servo drive.

1) Control word 6040h

This object is a command word for the master controller unit to control the operation status of the servo, such as enable, start/stop, and alarm reset. It is the most basic control command word. Therefore, 6040h (control word) is a required item in the PDO configuration table.

Index	6040h
Object name	Control word
Object code	VAR
Access	RW
Data type	UNSIGNED16
Value range	0 to 65535
Default value	0
Access	RW
PDO mapping	Yes
Related mode	All

This object is a command word for the master controller unit to control the operation status of the servo. Its setpoints are clearly defined.

Bit	Name	Description
0	Servo ready	1: Active, 0: Inactive
1	Enable voltage	1: Active, 0: Inactive
2	Quick stop	1: Inactive, 0: Active
3	Servo ON	1: Active, 0: Inactive
4 to 6		Mode-specific
7	Fault reset	Fault reset is applicable to faults and alarms that can be reset. Bit 7 is rising edge-triggered. If bit 7 is kept to 1, other control commands are invalid.
8	Halt	For the halt method in each control mode, see 605Dh.
9 to 10	N/A	Reserved
11 to 15	Manufacturer-specific	Reserved and undefined

Notes:

Assigning values to individual bits of a control word is meaningless. All bits in the control word must work together to form a specific control command.

Bit 0 to bit 3 and bit 7 have the same meanings in different servo modes. Commands must be sent in sequence to guide the servo drive into the expected state according to the CiA402 state machine switching process. Each command corresponds to a specific state.

Meanings of bit 4 to bit 6 are mode-specific. For details, see control commands in different modes.

2) Target position 607Ah

This object is a target position command sent by the master controller for servo operation. The servo runs in profile position (PP) mode in most cases. In MC applications with a medium-sized PLC, the servo runs mostly in cyclic synchronous position (CSP) mode, where the controller commands the servo to run to a target position in the next EtherCAT period. The target position is in the physical dimension set by the user. The target position can be monitored through the axis data structure variable Axis.fSetPosition.

Therefore, the target position object 607Ah is a required item in the PDO configuration table.

Index	607Ah
Name	Target position
Object code	VAR
Data type	INTER32
Access	RW
PDO mapping	Yes
Value range	0x80000000 to 0x7FFFFFFF
Default value	0
Unit	Reference unit
Related mode	PP/CSP
Comment	

This object sets the target position in PP mode and CSP mode.

Bit 6 in 6040h	Description

0	607A is the absolute target position of the current segment. After positioning of the current segment is complete, the position feedback 6064 equals 607A.
1	607A indicates the target increment displacement of the current segment. After positioning of the current segment is complete, the position feedback increment equals 607A.

3) Modes of operation 6060h

The master controller can set the servo operation mode through object 6060h.

Index	6060h
Name	Modes of operation
Object code	VAR
Access	RW
PDO mapping	Yes
Data type	INTEGER8
Value range	0x00 to 0x0A
Default value	0
Related mode	All

This object selects the operation mode of the servo drive.

Value	Servo Mode	Supported by AM600
0x00	N/A	-
0x01	Profile position (PP) mode	-
0x02	N/A	-
0x03	Profile velocity (PV) mode	-
0x04	Profile torque (PT) mode	Supported
0x05	N/A	-
0x06	Homing (HM) mode	Supported
0x07	Interpolated position (IP) mode Not supported	-
0x08	Cyclic synchronous position (CSP) mode	Supported; default mode
0x09	Cyclic synchronous velocity (CSV) mode	-
0x0A	Cyclic synchronous torque (CST) mode	-

Precautions for servo operation mode switchover:

When the servo drive in any state switches over from the PP or CSP mode to another mode, the position references not executed will be abandoned.

When the servo drive in any state switches over from the PV, PT, CSV, or CST mode to another mode, it stops at ramp before entering into that mode.

When the servo drive is running in homing mode, the servo drive cannot switch to another mode. After homing is complete or interrupted (fault or S-ON off), the servo drive can switch to another mode.

When the servo drive in running state switches over from a mode to the cyclic synchronous mode, send the reference at an interval of at least 1 ms; otherwise, reference loss or error will occur.

4) Target velocity 60Ffh (profile velocity)

This velocity command must be set if the servo runs in velocity mode (PV or CSV).

Index	60FFh
Name	Target velocity (profile velocity)
Object code	VAR
Data type	INTER32
Access	RW
PDO mapping	Yes
Value range	0x80000000 to 0x7FFFFFFF
Default value	0x64
Unit	Reference unit/s
Related mode	PV/CSV

5) Target torque 6071h

This command must be set if the servo run in torque mode (PT or CST) in a dimension of a percentage (0.1%) to the motor's rated torque.

Index	6071h
Name	Target torque
Object code	VAR
Data type	INTER16
Access	RW
PDO mapping	Yes
Value range	0xEC78 to 0x1388
Default value	0x0000
Unit	0.1%
Related mode	PT/CST

In target torque settings, 100% (readout value being 1000) corresponds to 1x the rated motor torque.

6) Max profile velocity 607Fh

This object sets the maximum operation velocity of the servo and limits the maximum velocity in PV mode.

In ST or PT mode, this object limits the maximum velocity to avoid motor overspeed that can cause mechanical shock.

This object is invalid in CSP mode.

Index	607Fh
Name	Max. profile velocity
Object code	VAR
Data type	UNSIGNED32
Access	RW
PDO mapping	Yes
Value range	0x00000000 to 0xFFFFFFFF
Default value	0x06400000
Unit	Reference unit/s
Related mode	All

This object sets the maximum running velocity of the servo, which is valid in PV, ST, and PT modes. It limits the maximum running velocity of the servo motor.

For example, when the servo is running in torque mode, if the actual load torque of the motor is less than the torque reference, the motor becomes faster and faster and will be eventually limited at the maximum profile velocity (object 607Fh).

7) Touch probe function 60B8h

This object is used by the master controller to set the touch probe function mode and start/stop of the servo. In the motion control system, the servo probe function detects the servo position signal when a specific DI signal changes. The servo records the servo position in an interrupt mode when the DI signal changes for the host controller to read. This improves the accuracy of system control.

Index	60B8h
Name	Touch probe function
Object code	VAR
Data type	UINTER16
Access	RW
PDO mapping	Yes
Value range	0x0000 to 0xFFFF
Default value	0x0000
Unit	-
Related mode	All

This object defines the functions of touch probe 1 and touch probe 2.

Bit	Description
0	Touch probe 1 enable 0: Disable 1: Enable
1	Touch probe 1 trigger mode 0: Single trigger (trigger first event) 1: Continuous trigger
2	Touch probe 1 trigger signal selection 0: DI8 input signal 1: Z signal
3	N/A
4	Touch probe 1 rising edge 0: Switch off latching at rising edge 1: Enable latching at rising edge
5	Touch probe 1 falling edge 0: Switch off latching at falling edge 1: Enable latching at falling edge
6	N/A
7	N/A
8	Touch probe 2 enable 0: Disable 1: Enable
9	Touch probe 2 trigger mode 0: Single trigger (trigger first event) 1: Continuous trigger
10	Touch probe 2 trigger signal selection 0: DI9 input signal 1: Z signal
11	N/A

12	Touch probe 2 rising edge 0: Switch off latching at rising edge 1: Enable latching at rising edge
13	Touch probe 2 falling edge 0: Switch off latching at falling edge 1: Enable latching at falling edge
14	N/A
15	N/A

The IS620N drive only supports the falling edge of the Z signal.

For absolute encoders, Z signal refers to the zero point of the single-turn position feedback.

8) Servo status word 6041h

This object is used to read the operation status of the servo drive. It is one of the required items in the PDO configuration table.

Index	6041h
Name	Status word
Object code	VAR
Data type	UNSIGNED16
Value range	0 to 65535
Default value	-
Access	RO
PDO mapping	TPDO
Related mode	All

The servo slave feeds the status back to the master through different bits.

Bit	Name	Description
0	Servo ready	1: Active, 0: Inactive
1	S-ON	1: Active, 0: Inactive
2	Servo ON	1: Active, 0: Inactive
3	Fault	1: Active, 0: Inactive
4	Voltage enabled	1: Active, 0: Inactive
5	Quick stop	0: Active, 1: Inactive
6	Switch on disabled	1: Active, 0: Inactive
7	Alarm	1: Active, 0: Inactive
8	Manufacturer-specific	Reserved and undefined
9	Remote control	0: Non-remote control mode. The IS620N series products only support the remote control mode; 1: Remote control mode
10	Target reach	0: The target position or velocity is not reached. 1: The target position or velocity is reached.
11	Internal limit active	0: The position reference or feedback does not reach the software internal position limit. 1: The position reference or feedback reaches the software internal position limit. After the software absolute position limit is activated (see object dictionary 607Dh and 200A-02h), the servo runs with the position limit value as the target position and stops at the limit value.

12 to 13	Mode-specific	Dependent on servo modes
14	N/A	Reserved
15	Homing completed	0: Homing is not performed or complete. 1: Homing is complete and the reference point is found.

Notes:

Reading out a bit separately is meaningless. All bits in the status word constitute servo status feedback together.

Bit 0 to bit 9 have the same meanings in different servo modes. After commands in 6040h are sent in sequence, the servo drive feeds back an acknowledged state.

Meanings of bit 12 to bit 13 are mode-specific. For details, see control commands in different modes.

Bit 10, bit 11, and bit 15 have the same meanings in different servo modes and indicate the servo drive status after a certain mode of operation is implemented.

9) Position actual value 6064h

In most cases, the servo operates in position mode. The controller must monitor the current position of the servo in real time, and the master controller reads the actual current position of the servo through this object. This object is one of the required items in the PDO configuration table.

Index	6064h
Name	Position actual value
Object code	VAR
Data type	INTER32
Access	RO
PDO mapping	TPDO
Value range	-
Default value	-
Unit	Reference unit
Related mode	All

This object indicates user absolute position feedback in real time.

Position feedback 6064h (reference unit) x Position factor 6093h (gear ratio) = Actual position 6063h (pulse unit)

10) Servo error code 603Fh

This object represents the most recent error code or alarm code of the drive. For error codes and their meanings indicated by the low 12 bits, see the IS620N guide. The master controller determines the latest fault code of the servo based on this object. This object is one of the required items in the PDO configuration table.

Index	603Fh
Name	Error code
Object code	VAR
Data type	UINT16
Value range	0 to 65535
Default value	-
Access	RO
PDO mapping	TPDO

Related mode	All

11) Torque actual value 6077h

This object reflects the internal actual torque of the servo drive. The value is given per hundred (0.1%) of rated torque. This object is one of the common items of the PDO configuration table.

Index	6077h
Name	Torque actual value
Object code	VAR
Data type	INTER16
Access	RO
PDO mapping	TPDO
Value range	-
Default value	-
Unit	0.1%
Related mode	All
Comment	-

This object indicates the internal torque of the servo drive.

The value of 100% (readout value of 1000) corresponds to 1x the rated motor torque.

12) Following error actual value 60F4h

This object indicates the deviation, in reference unit, of the current position from the target position. It is used to determine whether a position is reached.

Index	60F4h
Name	Following error actual value
Object code	VAR
Data type	INTER32
Access	RO
PDO mapping	TPDO
Value range	0x80000000 to 0x7FFFFFFF
Default value	-
Unit	Reference unit
Related mode	PP/HM/CSP
Comment	

This object indicates the position deviation (in reference unit). Position deviation 60F4 = Position deviation 200B-36h

13) Touch probe status 60B9h

This object indicates the setting status and trigger status of the servo probe trigger port to help the master controller to read the probe position recording data and determine its validity.

Index	60B9h
Name	Touch probe status
Object code	VAR
Data type	UINTER16
Access	RO
PDO mapping	TPDO

Value range	-
Default value	-
Unit t	-
Related mode	All

This object indicates the status of touch probe 1 and touch probe 2.

Bit	Description
0	Touch probe 1 enable 0: Disabled 1: Enabled
1	Touch probe 1 rising edge value 0: No rising edge value latched 1: Rising edge value latched
2	Touch probe 1 falling edge value 0: No falling edge value latched 1: Falling edge value latched
3	N/A
4	N/A
5	N/A
6	Touch probe 1 trigger signal selection 0: DI8 input signal 1: Z signal
7	Touch probe 1 triggering signal monitoring 0: DI8 is low level 1: DI8 is high level
8	Touch probe 2 enable 0: Disabled 1: Enabled
9	Touch probe 2 rising edge value 0: No rising edge value latched 1: Rising edge value latched
10	Touch probe 2 falling edge value 0: No falling edge value latched 1: Falling edge value latched
11	N/A
12	N/A
13	N/A
14	Touch probe 2 trigger signal selection 0: DI9 input signal 1: Z signal
15	Touch probe 2 triggering signal monitoring 0: DI9 is low level 1: DI9 is high level

14) Touch probe 1 position feedback 60BAh and 60Bbh (Touch Probe Pos1 Value)

Index	60BAh	60BBh
Name	Touch probe 1 rising edge (Touch Probe Pos1 Pos Value)	Touch probe 1 falling edge (Touch Probe Pos1 Neg Value)
Object code	VAR	VAR
Data type	INTER32	INTER32
Access	RO	RO

PDO mapping	TPDO	TPDO
Value range	-	-
Default value	-	-
Unit	Reference unit	Reference unit
Related mode	All	All
Comment	Indicates the position value of the touch probe 1 at rising edge (reference unit).	Indicates the position value of the touch probe 1 at falling edge (reference unit).

15) Touch probe 2 position feedback 60BCh and 60BDh (Touch Probe Pos2 Value)

Index	60BCh	60BDh
Name	Touch probe 2 rising edge (Touch Probe Pos2 Pos Value)	Touch probe 2 falling edge (Touch Probe Pos2 Neg Value)
Object code	VAR	VAR
Data type	INTER32	INTER32
Access	RO	RO
PDO mapping	TPDO	TPDO
Value range	-	-
Default value	-	-
Unit	Reference unit	Reference unit
Related mode	All	All
Comment	Indicates the position value of the touch probe 2 at rising edge (reference unit).	Indicates the position value of the touch probe 2 at falling edge (reference unit).

16) Modes of operation display 6061h

Index	6061h
Name	Modes of operation display
Object code	VAR
Data type	INTEGER8
Access	RO
PDO mapping	TPDO
Value range	-
Default value	-
Related mode	All

The object 6061h indicates the current operation mode of the servo through the following values:

Value	Description
0x00	N/A
0x01	Profile position (PP) mode
0x02	N/A
0x03	Profile velocity (PV) mode
0x04	Profile torque (PT) mode
0x05	N/A

0x06	Homing (HM) mode
0x07	Interpolated position (IP) mode
0x08	Cyclic synchronous position (CSP) mode
0x09	Cyclic synchronous velocity (CSV) mode
0x0A	Cyclic synchronous torque (CST) mode

17) Homing method 6098h

For applications that use relative positioning, a homing operation is required first to allow the servo drive and motion controller to determine the reference homing point for the position.

Index	6098h
Name	Homing method
Object code	VAR
Data type	INTER8
Access	RW
PDO mapping	Yes
Value range	0x00 to 0x23
Default value	0x00
Unit	-
Related mode	HM
Comment	-

The homing method must be set before servo homing can be commanded through EtherCAT communication. This object is used by the master to set the homing method. For details, see the homing method description in the Appendix.

Note: If the IS620N absolute position encoder method is selected, the 35th homing method can be used. The result of the homing operation is that the current position is used as the homing point and the motor does not rotate.

18) Homing speeds 6099h

This object can be used to set the speeds at which the servo drive runs while searching for the home signal. This object has two sub-indexes.

Index	6099h
Name	Homing speeds
Object code	ARR
Data type	UNSIGNED 32
Access	RW
Mapping	Yes
Value range	OD data range
Default value	OD default value
Related mode	HM

The sub-indexes define two speeds used in the homing mode: speed during search for switch and speed during search for zero.

Sub-index	0	1	2
Name	Number of sub-indexes for homing speeds	Speed during search for switch	Speed during search for zero
Data type	UNSIGNED8	UNSIGNED 32	INTER32
Access	RO	RW	RW
PDO mapping	No	Yes	Yes
Value range	2	0x00000000 to 0xFFFFFFFF	0x00000000 to 0xFFFFFFFF

Sub-index	0	1	2
Default value	2	0x001AAAAB	0x0002AAAB
Unit		Reference unit/s	Reference unit/s

Notes:

The first sub-index defines the speed during search for switch. A large value helps prevent the homing timeout fault Er.601.

After finding the switch, the slave decelerates and blocks all home signal changes during deceleration. To prevent the slave from encountering the home signal during deceleration, set the switch position of the deceleration point signal properly to leave sufficient deceleration distance or increase the homing acceleration rate to shorten the deceleration time.

The second sub-index defines the speed during search for zero. Set this sub-index to a small value to avoid overshoot due to stop at a high speed, preventing excessive deviation between the stop position and the preset mechanical home.

19) Homing acceleration 609Ah

This object sets the acceleration rate at which the servo drive runs while searching for the home signal.

Index	609Ah
Name	Homing acceleration
Object code	VAR
Data type	UNSIGNED32
Access	RW
PDO mapping	Yes
Value range	0x00000000 to 0xFFFFFFFF
Default value	0x682AAAAB
Unit	Reference unit/s2
Related mode	HOME

This object sets the acceleration during homing. The setpoint is activated after homing is started.

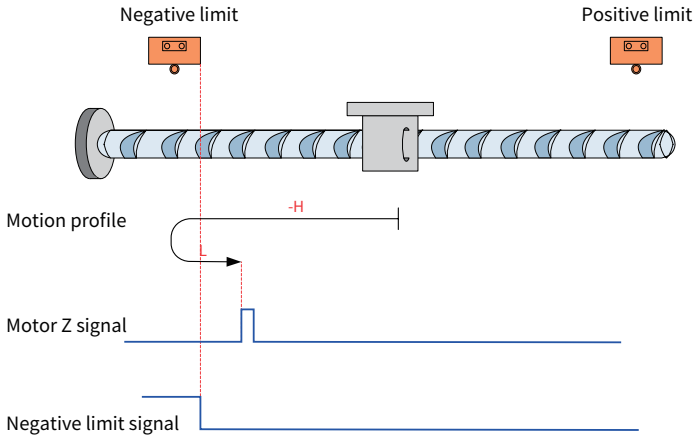
In homing mode, if 605Dh (Halt option code) is set to 2, the servo drive decelerates to stop according to 609Ah.

This object dictionary indicates the position reference (reference unit) increment per second. The setpoint 0 will be forcibly changed to 1.

Example homing method:

6098h = 1

This setting is suitable for applications with the following mechanical structure. There is a limit switch at each end of the slider travel and no zero switch signal, as shown in the following figure.



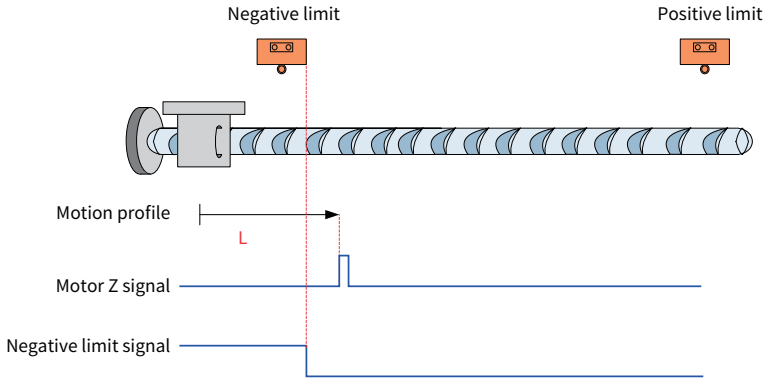
The mechanical home uses motor Z signal, and the deceleration point is the negative limit switch (N-OT).

- Deceleration point signal inactive at start of homing

Note: In the figure, "H" represents 6099-1h (Velocity during search for switch), which is high speed, and "L" represents 6099-2h (Speed during search for zero), which is low speed.

The N-OT signal is inactive initially, and the motor starts homing in negative direction at the high velocity. After reaching the rising edge of the N-OT signal, the motor decelerates and changes to run in positive direction at the low velocity. After reaching the falling edge of the N-OT signal, the motor stops at the first motor Z signal.

- Deceleration point signal active at start of homing



The N-OT signal is active initially, and the motor directly starts homing in positive direction at the low velocity. After reaching the falling edge of the N-OT signal, the motor stops at the first motor Z signal.

According to the functions of the objects above, we can regard the object dictionary as a set of servo drive function code definitions that can be accessed by EtherCAT bus communication, which facilitating understanding.

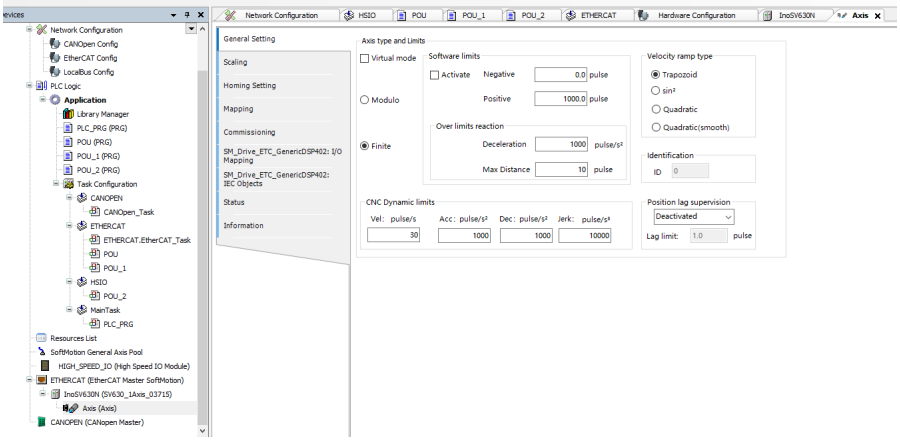
4.3.3 Configuration of Servo Axis Motor Parameters

The motion control action is ultimately achieved through the operation of the servo motor. To make the servo motor run as expected, the controller needs to know the servo motor parameters, characteristic parameters of the mechanical transmission mechanism of the application system, and the operation characteristics desired by the user so that the controller can send appropriate operation position commands. This requires users to set these characteristic parameters for the controller during programming.

The following figure shows how to set the servo motor parameters. Double-click the servo drive under the servo motor. Then, you can set the motor parameters in the right window.

8) On the "Basic Parameters" tab, set the axis location counter mode value. If the servo motor is characterized by round-trip operation, such as the reciprocating operation of the screw, you can select "Linear Mode" (also called multi-turn mode or finite-length mode), which enables the positioning in the absolute position mode when the servo motor rotates for multiple revolutions.

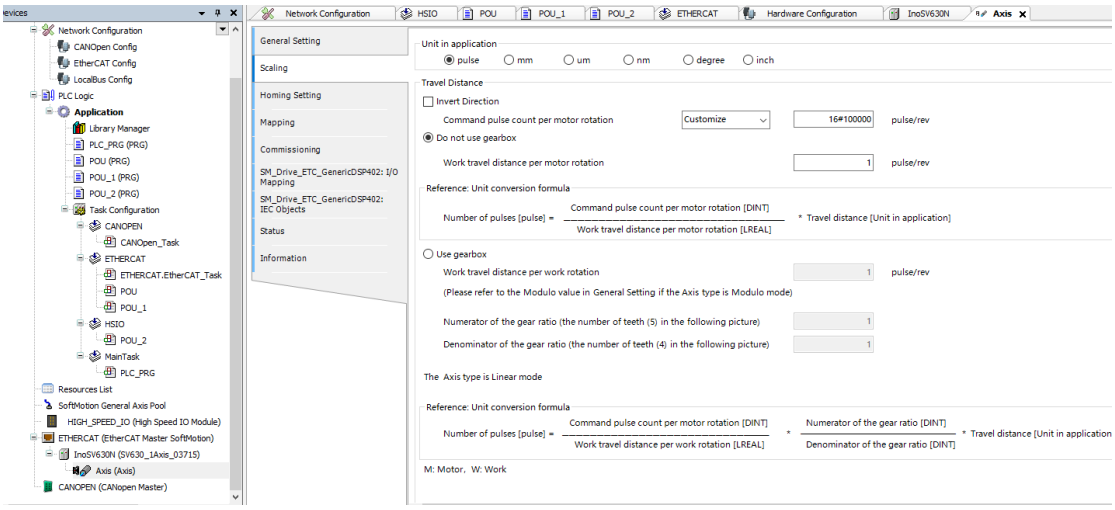
9) If the servo motor runs infinitely in one direction, such as the operation of a flying shear roller, you can select "Cyclic Mode". The position counter starts counting from 0 during each operation period, which avoids the overflow of the position counter.



📖Note that the above setting rules are applicable to both incremental encoder servo motors and absolute encoder servo motors. The above values are not sent to the servo drive. The current position of the motor is accumulated and the span is calculated automatically by AM600 based on the position signal returned from the motor. Therefore, to retain the servo position upon power failure, you need to back up the current position of the axis to the power failure retentive variable in the user program and then restore it to the relevant parameter in time after power-on.

The "Software Limitation" in the preceding figure refers to the travel overlimit protection of the servo motor through the AM600 software, which prevents AM600 from sending overlimit positioning instructions. This is very useful in MC application systems with the absolute position encoder and absolute positioning instructions. The figure also shows options of acceleration and deceleration characteristic curves. Select an option as needed during commissioning to make the mechanical system run smoothly.

10) As the operation position command of the controller is to make the servo run a certain number of pulses, the controller must know the pulse value per revolution of the servo motor encoder, as well as the mechanical parameters such as the reduction ratio of the operating mechanism, screw lead, and pulley circumference. You can configure these parameters on the "Zoom/Mapping" tab page, as shown in the following figure.

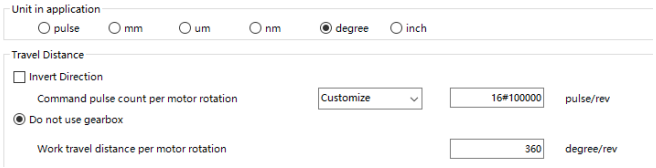


In the preceding figure, item 1 is used to set the number of pulses per revolution, which can be in hexadecimal.

Item 2 is used to set the reduction ratio of the reducer. The ratio 5:1 in the figure means that for every five revolutions of the servo motor axis, the output axis of the reducer will rotate one revolution. If no reducer is used, the reduction ratio is 1:1.

Item 3 is used to set the physical distance that the workpiece runs for every one revolution of the output axis. Examples:

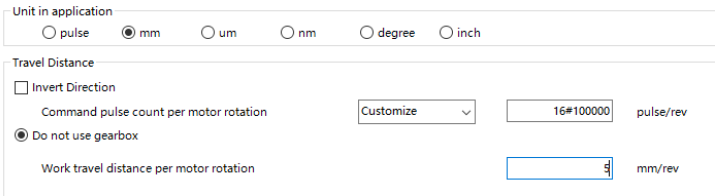
- If you use a flying shear roller, the rotation angle is regarded as the physical distance. If the parameters are configured as follows:



the execution result of instruction MC_MoveRelative(,distance:=1,) is that the mechanism rotates by 1° ;

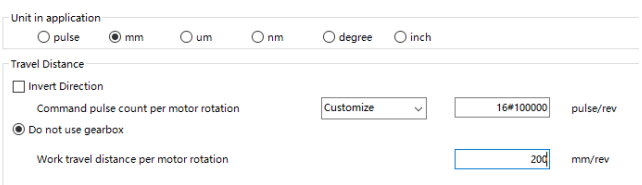
the execution result of instruction MC_MoveRelative(,distance:=360,) is that the mechanism rotates by 360° .

- If you use a screw with a lead of 5 mm, that is, for every 1 revolution of the screw, the slider on the screw moves for 5 mm. If the parameters are configured as follows:



the execution result of instruction MC_MoveRelative(,distance:=1,) is that the slider mechanism travels 1 mm.

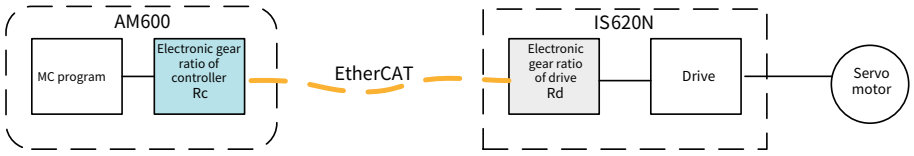
- If you use a synchronous gear with a diameter of 63.7 mm, the synchronous belt moves a distance of 63.7 mm x 3.14 = 200 mm for every 1 revolution. If the parameters are configured as follows:



the execution result of instruction MC_MoveRelative(,distance:=100,) is that the belt mechanism travels 100 mm.

The examples show that we can make the physical unit of the application system be consistent with the unit of the MC operation instruction by configuring items 1 to 3 accurately. This makes the user program instructions clearer, which facilitates variable configuration and reduces errors.

📖Note that the motor parameters are set for the conversion of electronic gear ratios when AM600 sends the final (number of pulses) position instruction. The parameters are not downloaded to the servo drive. The electronic gear ratios set by the function codes in the servo will likewise attenuate the operation instructions. In this way, the actual effect on the servo motor is calculated as Rc x Rd, as shown below:



Therefore, to ensure that the user program has the same performance in all application devices, you need to initialize the function code of the servo electronic gear ratio to the specified parameter value through the SDO operation; otherwise, differences in the operation response will be caused by different settings of the servo function code.

📖Note that, for MC programming, the first step of commissioning is to ensure consistency between the physical unit of the application system and the unit of the MC operation instructions. Otherwise, the programmer cannot determine whether the expected operation effect is achieved, and device damage or personal injury may be caused due to overlimit positions.

4.3.4 EtherCAT Network Status Initialization and Management

20) Initialization and status determination of the EtherCAT network

The AM600 controller starts automatically upon power-on and finishes loading the operating system and user program in about 10 seconds. If the EtherCAT bus is not used in the user program, the controller starts executing the user program after initialization of the bus used by the user program.

If the EtherCAT network is used in the user program, AM600, as the EtherCAT master, initializes the EtherCAT bus in the following steps:

11) Configure the master according to the user's EtherCAT configuration. This takes about 3 seconds.

12) Send a network initialization command to allow the ESC chips of all slaves to start the initialization operation, read the information of slaves in the EtherCAT network one by one, compare the information with the EtherCAT network configuration in the user program, and report an error if there is a discrepancy in the number and order of slaves.

13) Send the SDOs and PDOs to the ESC chip of each slave one by one if the network configuration is normal.

14) Make the network enter the Pro-OP, Safe-OP, and finally OP status.

The above operations are completed automatically by AM600 without user intervention. It takes about 2 seconds to configure each slave. More slaves mean longer network initialization time.

The simplest and most reliable way for the user program to determine whether the network status of the application system is normal is to detect whether MC_Power.status of each servo axis is true. If yes, the network and the servo are ready for normal operation.

21) Communication disconnection and recovery

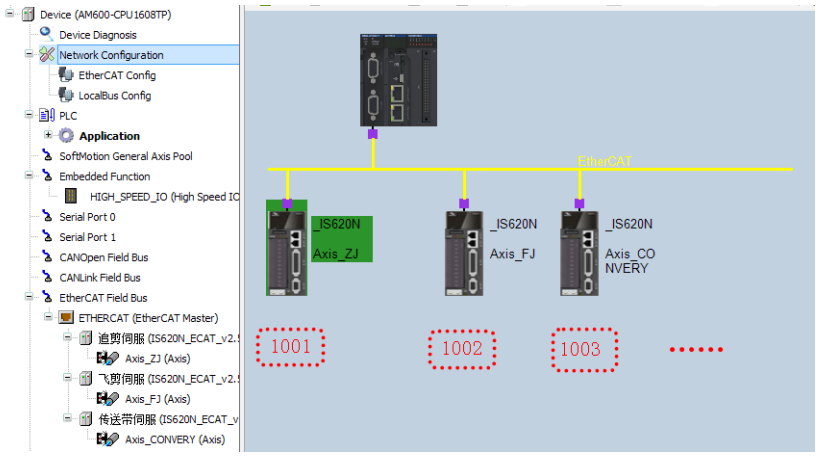
As we know, the prerequisite for an EtherCAT slave to communicate with the master is that the slave ESC enters the Pro-OP, Safe-OP, and finally OP network status after being configured by the master. The

typical internal configuration of the ESC includes the PDO configuration table, which can be obtained by the slave ESC only when the master sends the network configuration. Once the master network enters the OP status, no more configuration information can be sent. Therefore, if the slave is powered up after the EtherCAT network master enters the operation status or is powered up again after it has a power failure during operation, the slave cannot enter the network OP status.

Currently, only restarting the master can restore network operation after an EtherCAT slave has a power failure. For example, toggle the RUN/STOP switch to restart the master. However, this will affect the operation of other slaves.

22) Slave addressing and address settings

During programming, by default, the AM600 master controller automatically performs addressing based on the connection order of network cables for EtherCAT slaves. This addressing method frees users from naming and renaming devices and only requires users to follow the bus network configuration in the user program, making it easy for the master controller to check the network configuration and find hardware connection errors. The following figure shows the rules for AM600 to automatically name the slaves added in the user program.



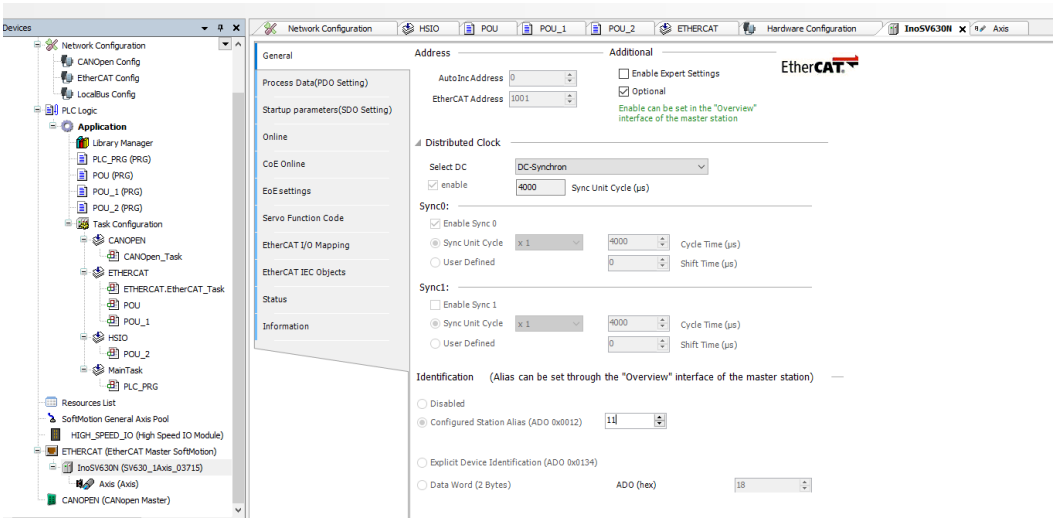
MC_P1: MC_Power;//Declare the instance MC_P1 of MC_Power.

```
MC_P1(Axis:= Axis1,
      Enable:= 1,
      bRegulatorOn:=1,
      bDriveStart:=1,
      Status=> , )//Execute the instance MC_P1 to enable the servo axis Axis1.
```

The slave serial number starts from 1001 and increases by 1 upon addition. During operation, the servos are named based on the connection order of network cables of the servo, and the servo directly connected to AM600 is named 1001. The axis control function in the user program is assigned to the servo with the corresponding serial number. The key point of this addressing method is that the connection order of the EtherCAT network cables must follow the network configuration order in the user program.

However, in some applications where the functions and names of some axes have been clearly defined, the user program of the AM600 master controller must perform addressing based on the pre-defined names. In this case, users must set the addressing method of the network slave to addressing by "slave alias" during programming and set the corresponding "slave alias" in the servo.

15) The following figure shows how to set the addressing method of the slave to addressing by "slave alias".



16) Set the slave alias in the servo slave.

For example, for IS620N, we can set its "slave alias" function code H0C.05 to 11.

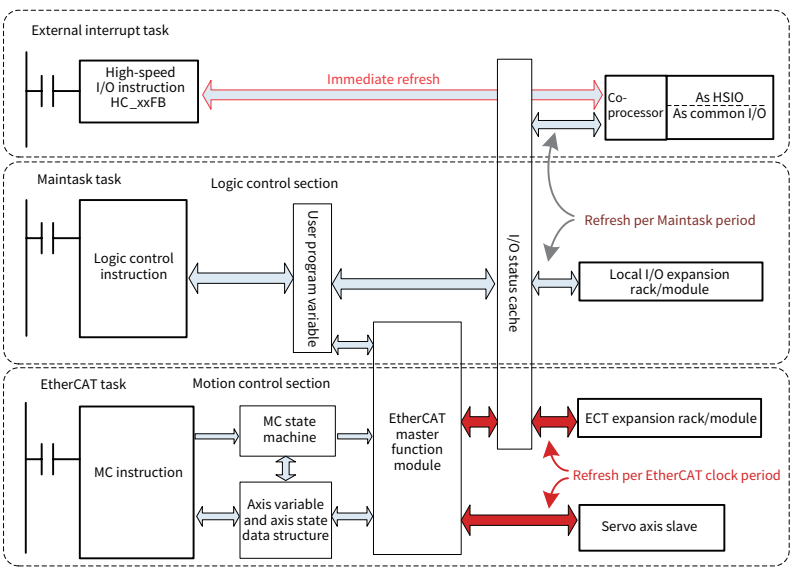
After the user program is configured in this way, regardless of the access position or order of the servo with the alias "11", it is possible to find the servo and assign the servo axis operation function in the user program to the axis.

📖Notes:

- In the user program, one or more axes can be named in this way, as long as the names are not duplicate.
- Currently, the ECT remote expansion module does not support slave setup for addressing by "slave alias".
- If some of the servo axes in the application system are automatically named, the system will first identify the slave with an "alias" and process the rest of the slaves according to the automatic naming rules.

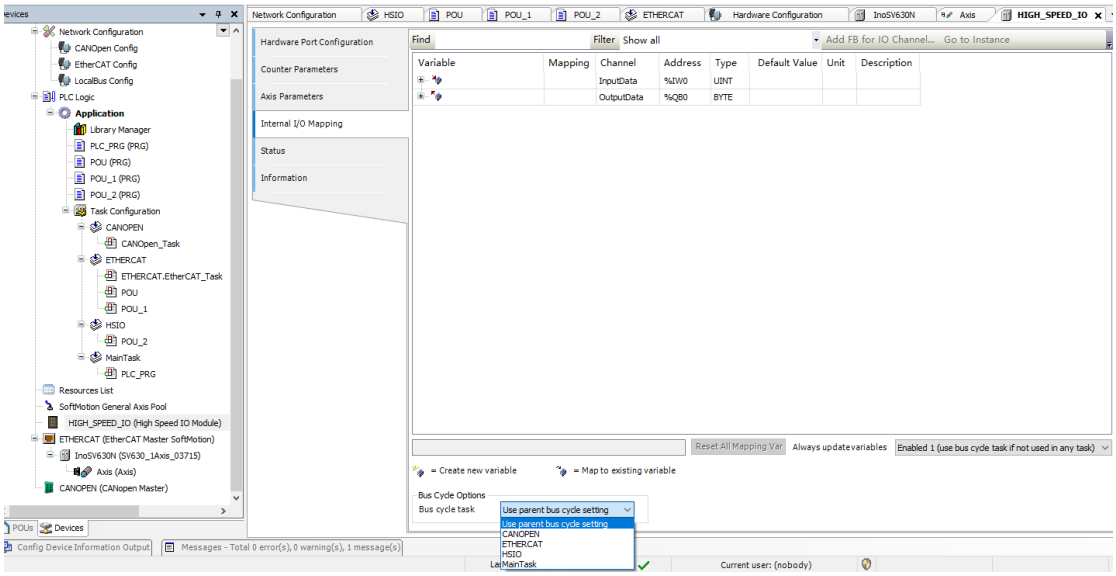
4.3.5 Servo Axis and I/O Port Control Data Refresh

There are three types of I/Os for AM600: HSIO built into the master module, I/O of the main rack expansion module, and I/O of the expansion rack module. The expansion rack is connected to the AM600 master module as an EtherCAT slave, just like the servo axis. The access refresh time of these peripherals has the following characteristics.

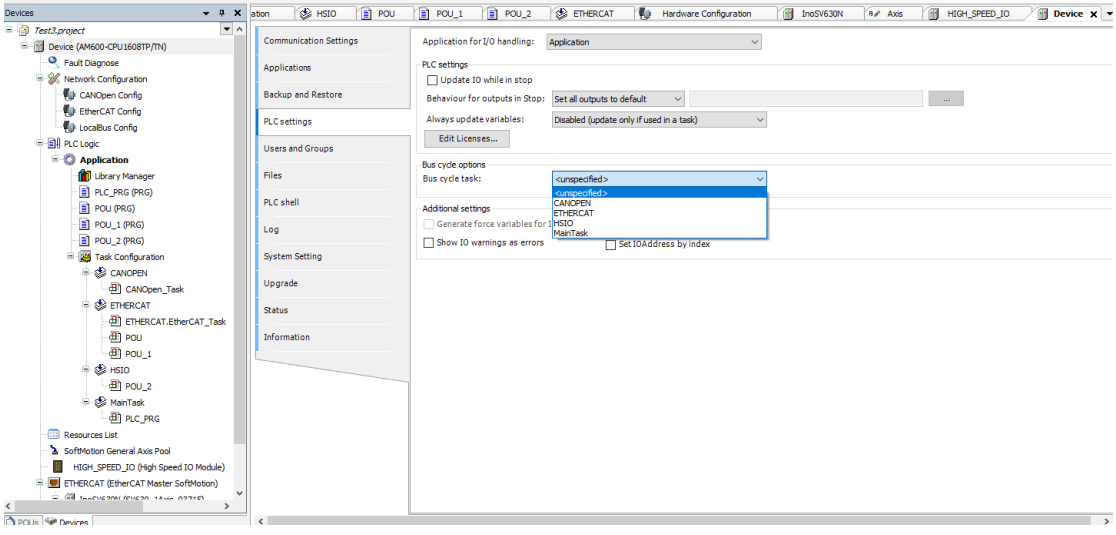


The AM600 controller has a built-in HSIO with 16 inputs and 8 outputs, and it is equipped with an internal co-processor for processing high-speed applications, such as interrupt signals, pulse counting, pulse characteristic measurement and other input signals. It can also carry out control outputs with real-time requirements, such as PTO, PWM, and pulse positioning. When executing Inovance's proprietary high-speed processing function block, it will immediately trigger the execution of the co-processor to update the output timely.

When the HSIO port is configured as a general port, the refresh period of its output refresh port can be set to the general task period, as shown in the following figure.



The AM600 main CPU module is connected to the I/O expansion module on the rack through the "local bus", and the refresh of I/O status is controlled by the CPU module. Its refresh interval is the same as that of the general task period, which can be set in InoProShop.



The AM600 controller is based on the remote I/O of the EtherCAT bus expansion rack. Its I/O communication data is transmitted in the same frame with the servo axis communication, for example, data is transmitted every 1 ms, 2 ms, or 4 ms. However, the logic control POU is generally executed in the general task, and the actual update period of the I/O status is the task period, for example, every 20 ms.

4.4 Timing of MC Data Transmission

The AM600 enters the EtherCAT interrupt according to the EtherCAT period set by the user and executes an entire EtherCAT task. Firstly, the communication operations between the master controller and each

EtherCAT slave are executed, and then all the user-configured POUs under the task are executed. The execution order is the same as the order of POUs in the task configuration table.

The communication operations between the master controller and each EtherCAT slave are as follows.

17) The EtherCAT bus transmit operation is initiated. The data in the TPDO transmit buffer prepared by the system in the previous EtherCAT period is sent to the corresponding slave in order. According to the RPDO configuration, several bytes of slots that are required by the slaves' response data are reserved in the communication frame to fetch the data from each slave.

The data in the TPDO transmit buffer is transmitted in the order of slave connection. The transmitted data contains the data of the general I/Os and the control data of the MC axes.

When there are many slaves and the data length exceeds the allowable length of one communication frame, multiple communication frames will be used.

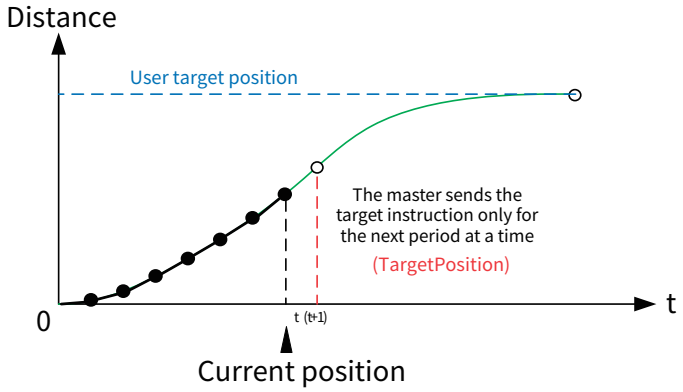
If the user program performs an SDO read/write operation, an SDO send request is sent at the end.

18) The master controller parses the returned frames, takes out the response data of each slave, and analyzes the response data for the MC slave axes. The master also updates the axis status and data structures such as position, velocity, and torque, and determines and updates the execution status indication of the MC function block for access by the user program. At each EtherCAT interruption, the axis parameters read by the user program are the data that has been automatically processed and updated in this section.

4.5 Processing Mechanism for Executing MC Function Blocks

4.5.1 Cyclic Synchronous Position Control Mode for Servo Motion Commands

The "cyclic synchronous position mode" allows the AM600 controller to calculate the required position (TargetPosition) for the next period point by the relevant MC function block during each EtherCAT task execution based on conditions such as the desired position of the slave axis, allowable running velocity, acceleration, and EtherCAT bus period and send it to the servo drive. The servo will move to the next target point according to this distance/time command. In this operation mode, AM600 is responsible for planning and calculating the servo operation position and velocity at each point of time, while the servo only knows the target point to be reached and the running velocity for the next EtherCAT moment.



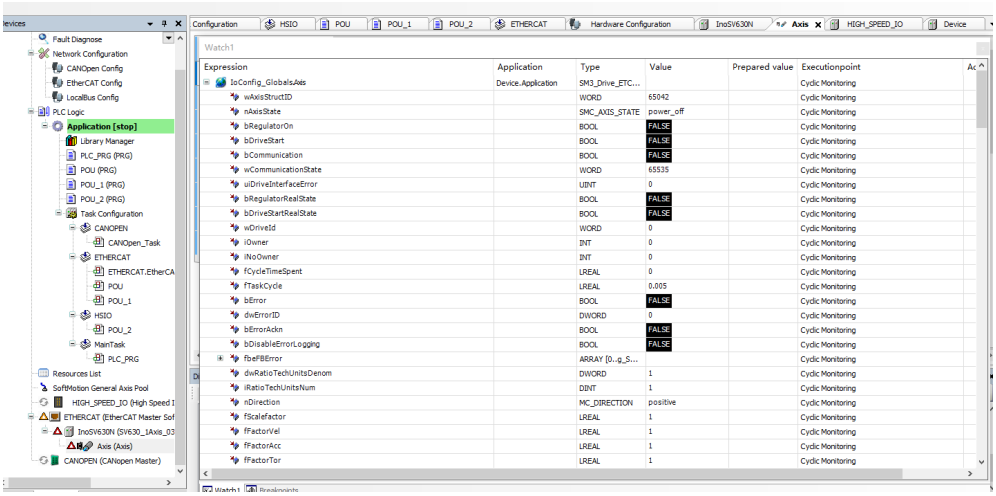
Note that when the servo is running in the "position mode" or "velocity mode", AM600 adopts the "cyclic synchronous position mode" to command the servo to run.

On the other hand, for a running servo axis, there must be an effectively triggered MC function block that continuously monitors the running of the servo axis. If no MC block is running for this axis due to the logic jump of the user program, the servo will stop after this state lasts for several EtherCAT periods, and the controller will generate an alarm for the error.

4.5.2 Data Structure of the Servo Axis

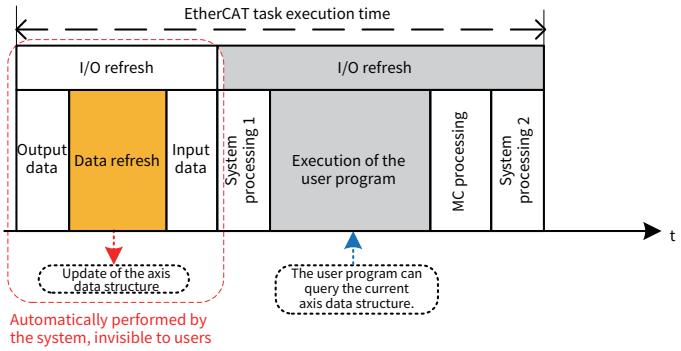
In AM600, the servo slave is managed as a special "axis", and an axis is an important object. In the system software of AM600, the system automatically declares a data structure for each servo axis configured by the user and automatically updates and maintains it in real time during the execution of each EtherCAT interrupt. The user program can access the data structure to learn about the current command value, operation status, operation position, velocity, acceleration rate, torque, and other parameters of the servo axis. There are more than 100 data structure variables in total, which provide a comprehensive description of the axis status.

The following figure shows an example of the Axis monitoring window in the user program. The information comes from the data structure of the axis.



Note the following characteristics of the axis data structure:

- When the user configures a servo axis for the application network, the system automatically declares the data structure, the name of which is the same as the axis name. The variable names and data types in the data structure are defined by the system.
- In the user project shown above, there are three servo axes (Axis, Axis_1, Axis_2), and each axis has the corresponding data structure.
- If the user program uses virtual axes, including encoder axes, the system also declares and maintains an axis data structure for them, only some of the structure variables may change.
- The axis data structure variables are global, that is, they are accessible in all POUs of the user project.
- There is no explicit limit to the number of axes allowed by the system as long as the controller computing power meets the requirements of the application. There is a corresponding number of axis data structures.
- Once the controller has started running, the servo feedback values are automatically updated into this data structure after the controller gets the slave response data during each EtherCAT task operation phase. The variables of this data structure are accessible during the execution of user POUs.



- Axis data variables are specified in the format of "Data structure name.Structure variable name". Generally, the following parameters are used in the data structure:

Axis.nAxisState: Current running state of the axis, which is the state parameter that the servo feeds back to the controller

Axis.fSetPostion: Axis set position, which is sent by the controller to the servo axis

Axis.fActPostion: Actual position of the axis, a status parameter returned by the servo to the controller, in the dimension the same as the command unit set by the user program

Axis.fActVelocity: Actual velocity of the axis, a status parameter returned by the servo to the controller, in the dimension the same as the command unit set by the user program

In the user program, these variables can be used as the basis for motion control calculation and judgment. Some of the variables in the axis structure are command data sent from the controller to the servo axis. In the user program, you can assign values to these variables to control the servo axis. The following ST statement is an example:

Axis.fSetPostion = 500;//The unit of this parameter is the same as the command unit of the command

4.5.3 Servo Axis Status and Transition Rules

AM600 complies with the PLCopen Specification. In a motion control system, the operation status of an axis is divided into several logical statuses. The direct transition of each logical status requires a specific condition or a specified MC instruction. The division enables the axes to be controlled according to the motion modes. The axes can only be in one logical status at one time, and the transition of the logical status must follow the rules. In this way, it avoids operation chaos due to the false triggering of different MCs.

The axis data structure variable (Axis.nAxisState) indicates the current operation status of the axis. Axis.nAxisState is an enumerated variable, with 8 possible statuses:

0: Power_off (Disabled): The axis is not powered on or enabled. You need to execute the MC_Power instruction.

1: Errorstop;---------- First execute the MC_Reset/MC_Power instruction.

2: Stopping;---------- Wait for the stopping operation to be completed.

3: Standstill;---------- The axis has stopped running and is out of synchronization.

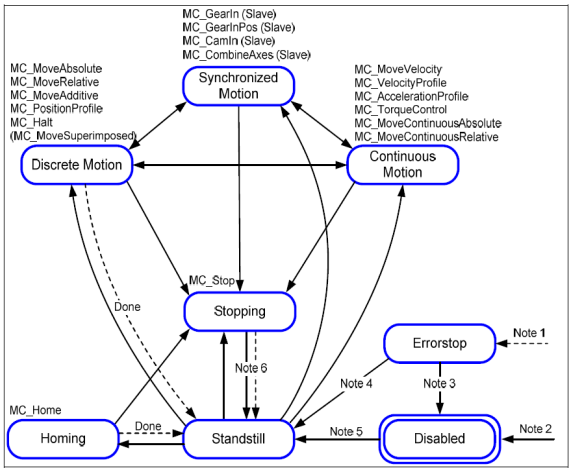
4: Discrete_Motion;--------- The axis is in the state of discrete motion.

5: Continuous_Motion;----- The axis is in continuous motion.

6: Synchronized_Motion;--- The axis is in synchronous motion.

7: Homing;---------- The axis is in homing operation. Wait for the homing operation to be completed.

The following is an axis status transition diagram. The transition of statuses requires specific conditions such as running the MC instruction, or external faults for which the user cannot reset the status. During programming, users must run the relevant instructions according to the logic requirements.



Note 1:	From any state. An error in the axis occurred.
Note 2:	From any state. MC_Power.Enable = FALSE and there is no error in the axis.
Note 3:	MC_Reset AND MC_Power.Status = FALSE
Note 4:	MC_Reset AND MC_Power.Status = TRUE AND MC_Power.Enable = TRUE
Note 5:	MC_Power.Enable = TRUE AND MC_Power.Status = TRUE
Note 6:	MC_Stop.Done = TRUE AND MC_Stop.Execute = FALSE

The MC function block in the figure enables the axis status to transition to the specified status, as shown in the figure:

- In the axis stop state (Standstill, that is, Axis.nAxisState = 3), it is possible to transition to various operation statuses.
- It is possible to transition from multiple statuses to the stop state (Standstill, that is, Axis.nAxisState = 3).
- If the servo axis has an alarm (Errorstop, that is, Axis.nAxisState = 1), you need to run the MC_Reset and MC_Power instructions to make the axis enter the Standstill status before the axis can run again.
- If you do not use the MC instruction to command the axis to move according to the above transition diagram, the axis will not respond, and an alarm message will be generated for the MC function block.

In the user program, sometimes you need to start the subsequent control logic according to the axis status. In this case, the judgment based on Axis.nAxisState is more accurate and reliable than the judgment for the done signal of the MC function block.

Familiarize yourself with the transition conditions shown in the axis status diagram and pay attention to the logic and sequence of MC instructions during programming to make the application stable and reliable.

4.5.4 Execution Logic of the MC Function Block

Axis control commands related to servo slaves are mostly in the form of MC function blocks (also called instructions). Since MC function blocks need to be executed continuously in short intervals and the servo operation response must be monitored in time, MC function blocks for axis motion can only be called for execution in an EtherCAT task. The internal processing of the system is as follows:

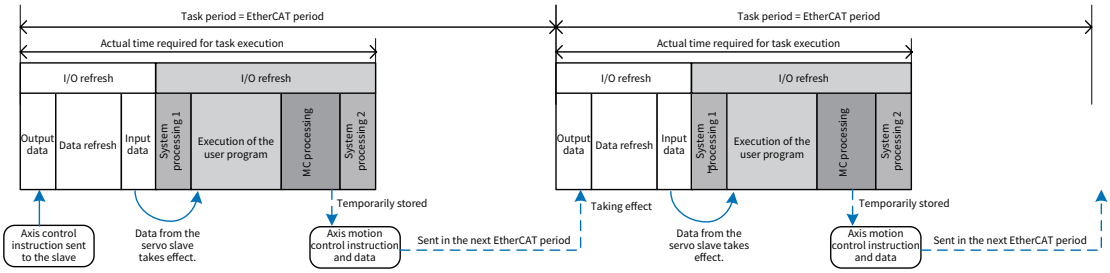
19) Only MC function blocks that have been effectively triggered will be executed. For multiple instances of the same MC function block (for the same axis object), the one that is triggered later will be prioritized.
20) For MC commands for servo axes, the system first checks the validity of the operations in accordance with the axis status transition rule. Then, the system performs operations including the axis status transition and update of the axis target parameters and prepares the axis control command data.
21) The system software for EtherCAT bus control makes the axis control command parameters into

PDO transmit buffer data according to the TPDO configuration table and object dictionary of each servo slave axis set by the user.

22) The system software for EtherCAT bus control will, according to the RPDO configuration table and object dictionary of each servo slave axis set by the user, reserve several bytes of slots required by the slave response data in the EtherCAT frame receive section. Finally, it "assembles" the axis status parameters to be read by the master to the EtherCAT frame transmit buffer and sends them to the slave at the next EtherCAT period.

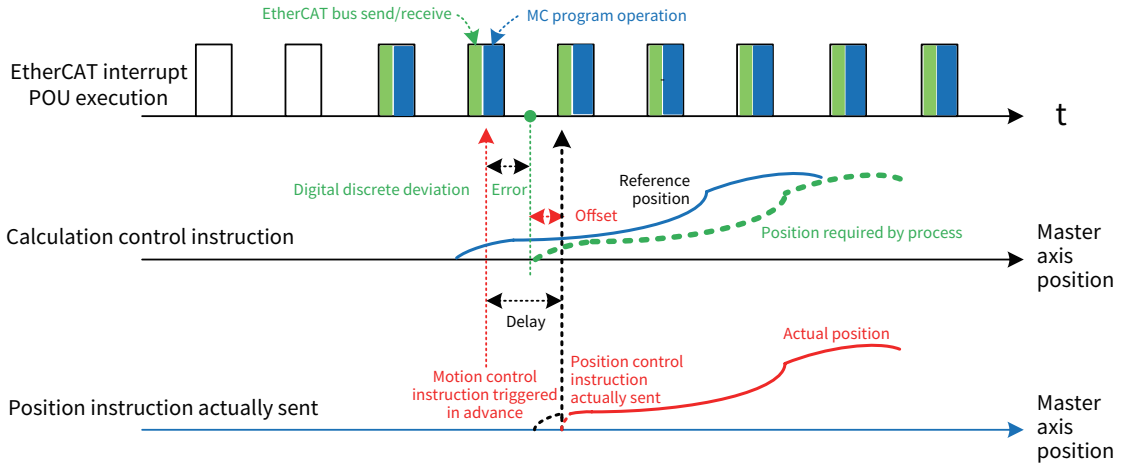
23) The results of the EtherCAT remote I/O operation are stored in the buffer according to the connection order of the slave racks and are sent together with the servo slaves. However, the status of the data in the transmit buffer is updated after the completion of a general task period (task priority of 15 or lower, such as a 20-ms period).

24) The following figure shows the timing for the master controller to execute the user program and send/receive data via EtherCAT:



25) In the MC user program, if the servo system is in operation, an MC function block that has been triggered for execution must be monitoring the servo axis, to avoid the absence of MC monitoring of a running servo axis due to the program logic jump. Using MC_Stop to make it stop is also a kind of monitoring.

It can be found that the axis control commands involved in an EtherCAT task are not sent out during the current POU execution period, and there is a delay of one EtherCAT period. The error caused by such delay must be taken into account in applications requiring precise position and length control, such as the triggering of MC_CamIn for multi-axis synchronous control, as shown in the following figure.



📖Notes:

Measures can be taken during programming to deal with the error caused by the above delay:

Trigger the MC instruction 1 EtherCAT period in advance.

The MC start required by the control process is not necessarily at the beginning of the EtherCAT period, and it can be in the middle of the period. The elimination of this discretization error should be taken into account during programming. The Offset parameter provided by the MC function block can be used for compensation.

26) To eliminate the discretization error, estimate the error caused by this kind of communication

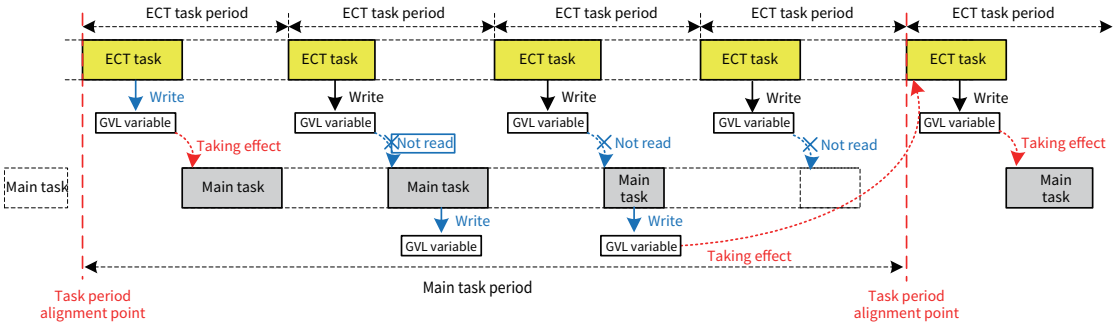
mechanism based on parameters such as the current object's operation position, velocity, and acceleration rate. A smaller EtherCAT bus period is conducive to reducing uncontrollable errors.

4.5.5 Data Interaction Between POUs of Tasks of Different Priorities

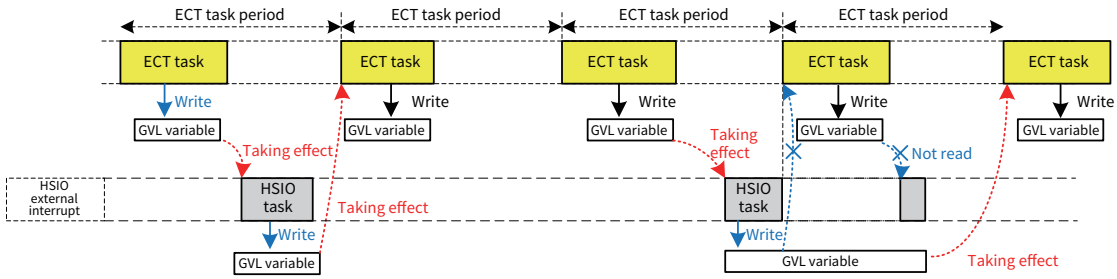
To support variable interaction between multiple POUs, it is necessary to use global variables, that is, you need to declare them in the global variable list (GVL). However, if the POUs are in tasks with different priorities, the data interaction is not in real time, and the result of the data update depends on the task priority, task period, and type of variable access. Pay attention to the following mechanism:

When the user program is executed, for tasks with different priorities and different periods, AM600 internally adopts the rule of start time alignment, that is, there is a common point for calculating the start time of the task period. If the period of one task is an integral multiple of another task, then the tasks will have a regular coincident time point (alignment point), which is often used as the GVL data interaction point.

- Only after a task has been executed, the modifications made by its POU to variables are written to the GVL buffer. Modifications made by low-priority tasks to GVL parameters only take effect at the end of their task period.
- Rewrite operations performed by high-priority POUs to the GVL take effect immediately.
- Low-priority tasks will copy the GVL value from the GVL buffer once before the execution of the first task from the alignment point, as a basis for use during the POU execution of this task. The GVL buffer variables will not be read again during the execution of this task.



- The servo axis data structure is a global variable automatically defined by the system. The system will automatically refresh the data structure every time the ECT task is executed. If the variables of this data structure are to be read in the Main task POU, the readings will be the data updated by the first ECT task after each "task period alignment point". Similarly, if the variables in the data structure needs to modify in the Main task POU, they will be sent to the servo axis in the first ECT task after the next "task period alignment point", with a delay of about one Main task period.
- For event-triggered execution of tasks, such as interrupt task POU execution triggered by HSIO, the latest variable values in the GVL will be used. As shown in the following figure, the updated GVL values between the EtherCAT task and the HSIO interrupt task can be interacted in a timely manner. When the execution of a lower-priority HSIO task is interrupted by an EtherCAT task, the GVL value modified by the HSIO task is valid only when the next EtherCAT task is executed after the HSIO task has been executed.



📖Notes:

It can be found that, in the user program, the period of a general main cyclic task must be an integer multiple of the period of an EtherCAT task. For example, set the period of an EtherCAT task to 2 ms or 4 ms and that of a general main cyclic task to 20 ms; otherwise, an exception will occur during the interaction of GVL parameters.

In tasks with different priorities, if there are modification operations on the same GVL variable, the values may overwrite each other. During programming, it is recommended that only one POU is allowed to perform rewrite or reset operations on a global variable, and the other POUs can only read and refer to it or reset the operation; otherwise, unanticipated errors will occur.



5 Application of the User Program

5. Application Programming of User Program

5.1. MC Programming For Single-axis MC Positioning

5.1.1 Notes for MC Application Programming

The motion control is achieved based on the EtherCAT bus network with the cooperation of the AM600 controller and servo axis such as IS620N. Different from the previous method of hardware output pulse, it uses only software, that is, servo control is achieved by carrying out a calculation and issuing a control command once in each short EtherCAT bus period. Therefore, pay attention to the following points:

- The MC user program is executed based on the EtherCAT task period. MC-related POUs must be configured so that the POUs will be executed under EtherCAT tasks. Most MC function blocks cannot run normally if they are placed in the POUs of low-priority Main tasks.
- The execution of MC function blocks requires the transmission of data objects in communication. Therefore, there must be required configuration items in the PDO configuration table. If the configuration-related data objects are omitted, the servo may not run normally and there will be no error alarm.
- The controller can initialize the settings of the servo function code through the SDO configuration and determine the servo operation mode (generally CSP mode), servo motor encoder mode, and electronic gear ratio, to ensure that the control commands correspond to the physical operation position. The initialization of the servo improves the commissioning efficiency of the device and reduces errors in parts replacement.
- The servo axis control must follow the rules and logic of axis status transition. Use the appropriate MC function block for control based on the current status of the axis and the desired motion.
- The user program uses an instance of the MC function block. An MC instance can only be used for the control of one servo axis. If it is used for the control of several servo axes at the same time, the control will be out of order.
- For a running servo axis, there must be an MC function block that continuously monitors the running of the servo axis. Even using MC_Stop is a kind of monitoring. If the servo axis is not monitored by an MC function block due to the logic jump of the user program, the system will stop and generate an error, which is not easy to detect.
- Pay attention to the safety during commissioning. If the servo system uses an incremental encoder, there must be a homing operation before normal operation. The DI signal input port of the servo drive can access the home position signal. For motion in a limited range (such as a screw), there should be a limit and safety protection signals before commissioning.

5.1.2 MC Function Blocks Commonly Used for Single-Axis Control

An MC function block (FB) is also known as an MC instruction. To be precise, what is used in the user program is the object instance of the MC FB, and the servo axes are controlled through MC object instances. Example:

MC_Power1:MC_Power;//Declare the instance MC_Power1

MC_Power1 (Axis=Axis1)

Single-axis control is generally used for positioning control, that is, the servo motor drives the external mechanism to move to the specified position. Sometimes the servo must run at the specified velocity or torque. In the single-axis control, the following MC function blocks are usually used:

Control Operation	Required MC Command	Description
Servo enabling	MC_Power	Run this instruction to enable the servo axis for subsequent operation control.
Servo jog	MC_Jog	Jogging of the servo motor, commonly used in low-speed test runs to check the device or adjust the position of servo motor.
Relative positioning	MC_MoveRelative	Take the current position as the reference and run for the specified distance.
Relative superposition positioning	MC_MoveAdditive	Move for the specified distance based on the current operation instruction of the servo.
Absolute positioning	MC_MoveAbsolute	Command the servo to move to the specified coordinate point.
Velocity control	MC_MoveVelocity	Command the servo to run at the specified velocity.
Torque control	SMC_SetTorque	Command the servo to run at the specified torque.
Servo halt	MC_Halt	Command the servo to halt. If MC_Movexxx is triggered again, the servo can run again.
Emergency stop	MC_Stop	Command the servo to stop urgently. Only after the stop command is reset and MC_Movexxx is triggered, the servo can run again.
Alarm reset	MC_Reset	When the servo stops with an alarm, run this instruction to reset.
Servo operation mode switchover	MC_ControlMode	Command the servo to select the "Position", "Velocity" or "Torque" mode.
Servo homing	MC_Home	Command the servo to start the homing operation. The home signal of the application system and the limit signals of both sides are connected to the DI port of the servo.
Controller homing	SMC_Homing	The control system starts the homing operation. The home signal of the application system and the limit signals of both sides are connected to the DI port of the controller.

5.1.3 MC Commands and PDO/SDO Configuration

When AM600 executes the servo axis MC commands of the user program, it is necessary to add the information items required for interaction with the servo to the communication PDO/SDO configuration table to achieve the control function.

MC Command	Required TPDO	Required RPDO
MC_Power MC_Halt MC_Stop MC_Reset MC_Home SMC_Homing	ControlWord (control word)	StatusWord (status word) Errorcode (error code)
MC_Jog	TargetPosition (target position)	Position actual value (current axis position) Following error actual value (current following error)
MC_MoveRelative		
MC_MoveAdditive		
MC_MoveAbsolute		
MC_MoveVelocity	Target velocity (target velocity) Max profile velocity (max. Profile velocity)	
SMC_SetTorque	Target torque (target torque)	Torque actual value (current torque)

MC_ControlMode	Modes of operation (operation mode)	16#6060=8: Cyclic Synchronous Position (CSP) mode 16#6060=9: Velocity mode 16#6060=10: Torque mode

The above TPDO and RPDO are basic configuration items required to perform single-axis control.

In MC control, the servo usually runs in position mode. In EtherCAT bus-based applications, the servo runs in Cyclic Synchronous Position mode. Therefore, set the servo operation mode to CSP in the SDO configuration during programming. For example, the following items are generally initialized in SDO for IS620N:

Initialization Operation for Servo	Required SDO	Description
Set to Cyclic Synchronous Position mode.	Modes of operation (16#6060)	Set to 8.
Setting electronic gear ratio	16#6091-1: 16#6091-2:	1:1 is recommended. (Function code setting is not recommended.)
Setting motor encoder type	16#0201 (IS620N function code)	0: Incremental; 1: Absolute finite length; 2: Absolute infinite length
Setting maximum allowable velocity	Max profile velocity (16#607F)	Applicable to velocity mode and torque mode.
Setting maximum allowable torque	Max torque (16#6072)	Applicable to velocity mode and torque mode.
Setting homing mode	Homing method (16#6098)	
Setting homing velocity	Homing speeds (16#6099)	
Touch probe function	Touchprobe function (16#60B8)	

5.2 Motion Control Programming for Multi-axis Cam Synchronization

Cam motion adopts the concept of relative motion between the mechanical cam and the tappet. Based on the specific nonlinear relationship for the relative position, the controller makes the servo slave axis follow the master axis for continuous and synchronous motion to meet the required motion characteristics of the device. Cam motion is extensively used in applications requiring the synchronization function, such as fixed-length cutting, chasing shear control, flying shear control, and multi-color overprint.

The master-slave axis position relationship of the electronic cam curve is shown below. The horizontal axis indicates the position of the master axis, and the vertical axis indicates the position of the slave axis.



AM600 adopts software to achieve cam motion control, that is, it uses the digital "cam table" to replace the mechanical cam, which is also called electronic cam control. Compared to the mechanical cam, cam motion control has the following features:

- Easy creation of cam shapes: Cam tables, cam curves, or arrays are used to describe cams.
- Diverse cam shapes: Multiple cam tables are available and can be switched dynamically during operation.

- Easy modification of cam shapes: Cam table key points can be modified dynamically during operation.
- Multiple cam slave axes: Multiple cam slave axes are supported.
- Cam tappet: Multiple cam tappets and multiple setting intervals are supported.
- Cam clutch: During cam operation, the user program can make it enter and exit the cam operation.
- Features of electronic cam: Support for virtual master axis, phase shift, and output superposition

The cam operation of AM600 is carried out by software only. In cam running status, the next target point of the slave axis is calculated every time the EtherCAT task is executed, thus providing higher functional flexibility than hardware cam operation.

Three elements of electronic cam control are as follows:

27) Master axis: A reference axis used for synchronous control

28) Slave axis: A servo axis that follows the master axis according to the desired non-linear characteristics based on the position of the master axis

29) Cam table: A data table or cam curve that describes the relative position, range, and periodicity of the master axis and slave axis

During programming, users need to design the cam table to specify the master axis and slave axis. Then, trigger the cam at an appropriate moment during running so that the slave axis can enter the cam running status.

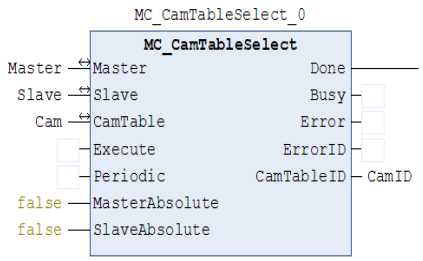
The basic instruction function blocks of electronic cam control are as follows:

Control Operation	Required MC Command	Description
Select cam table	MC_CamTableSelect	Run this command to associate the master axis, slave axis, and cam table.
Enter cam running	MC_CamIn	Make the slave axis enter cam running
Exit cam running	MC_CamOut	Make the slave axis exit cam running
Correct cam phasing	MC_Phasing	Modify master axis phasing

5.2.1 Main Function Blocks For Cam Running

1) MC_CamTableSelect FB for Cam Table Selection

This function block is used to associate the master axis, slave axis, and cam table, and to set the period of cam running as well as the position mode (absolute position or relative position) of the master and slave axes. This command is a management-type command, that is, once this command is triggered and executed once, the relevant master and slave axes can run according to this characteristic. To modify the cam table or change the master or slave axis, you need to trigger the execution of this function block again.



MC_CamTableSelect(
Master:= , //Cam master axis
Slave:= , //Cam slave axis
CamTable:= , //Cam table
Execute:= , //Command-triggered variable, rising edge-triggered
Periodic:= , //Set the cam periodicity
MasterAbsolute:= ,//Master axis position mode
SlaveAbsolute:= , //Slave axis position mode
Done=> , //
Busy=> ,
Error=> ,
ErrorID=> ,
CamTableID=>);
	MC_CamIn(
	Master:= , //Cam master axis
	Slave:= , //Cam slave axis
	Execute:= ,//Execution-triggered, rising edge-triggered
	MasterOffset:= ,//Master axis position offset
	SlaveOffset:= , //Slave axis position offset
	MasterScaling:= ,//Master axis scaling ratio
	SlaveScaling:= , //Slave axis scaling ratio
	StartMode:= , //Slave axis trigger position mode
	CamTableID:= , //Cam table pointer
	VelocityDiff:= , //Velocity deviation
	Acceleration:= , //Acceleration rate
	Deceleration:= , //Deceleration rate
	Jerk:= , //Jerk
	TappetHysteresis:= , //Tappet hysteresis
	InSync=> , //Synchronization indication
	Busy=> , //Running
	CommandAborted=> , //Command aborted
	Error=> , //Error
	ErrorID=> , // Error ID
	EndOfProfile=> , //Executed at end of cam
	Tappets=> ,// Tappet active
	);

The input and output variables of this function block are as follows:

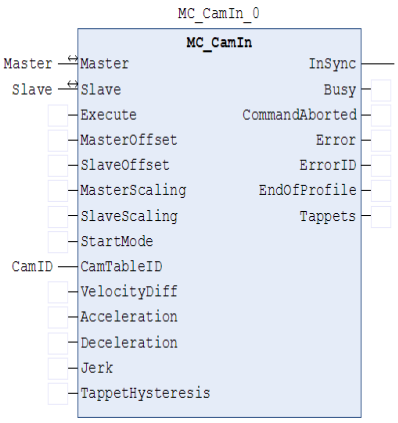
MC_CamTableSelect

This function block is part of the library SM3_Basic.library. It is designed for selecting the cam tables by setting connections to relevant tables.

	Type	Initialization	Description
VAR_IN_OUT			
Master	AXIS_REF		Reference to master axis, see AXIS_REF_SM3
Slave	AXIS_REF		Reference to slave axis, see
CamTable	MC_CAM_REF		Reference to cam description, see MC_CAM_REF
VAR_INPUT			
Execute	BOOL	FALSE	A rising edge of this input value will start the execution of the function block
Periodic	BOOL	TRUE	TRUE = periodic, FALSE = non periodic
MasterAbsolute	BOOL	TRUE	TRUE = absolute, FALSE = relative coordinates
SlaveAbsolute	BOOL	TRUE	TRUE = absolute, FALSE = relative coordinates
VAR_OUTPUT			
Done	BOOL	FALSE	TRUE, if preselection has been done
Busy	BOOL	FALSE	TRUE, if execution of function block is not finished
Error	BOOL	FALSE	Signals that an error has occurred within the function block
ErrorID	SMC_ERROR	0	Error identification, see SMC_Error
CamTableID	MC_CAM_ID		Identifier of cam table, to be used for function block MC_CamID

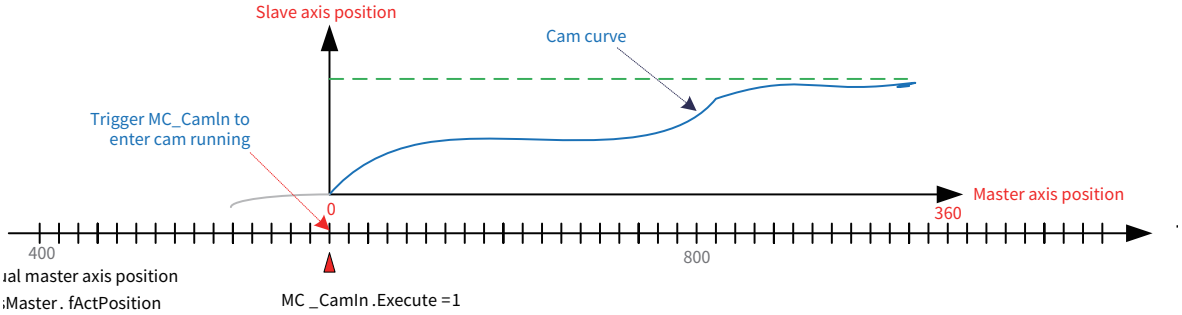
2) MC_CamIn function block for cam running

After the MC_CamTableSelect function block is run, users can run this function block to make the slave axis enter the cam running status (Synchronized_Motion, that is, Axis.nAxisState = 6). The system executes this function block once every time it enters an EtherCAT task to calculate the next target point of the slave axis based on the current position of the master axis and the cam table. If the MC_CamTableSelect function block is not run beforehand, an error will be reported if this function block is triggered.



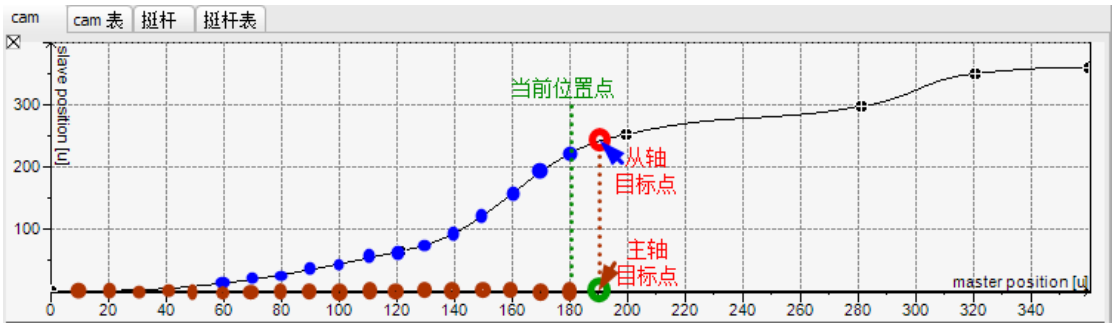

```
MC_CamOut(
    Slave:= , //Cam slave axis
    Execute:= ,//Trigger variable, rising edge-triggered
    Done=> , //Execution completed
    Busy=> , //Execution in progress
    Error=> , //Error
    ErrorID=> //Error ID
);
```

This instruction makes the cam slave axis enter into the state of synchronous operation with the cam master axis. It controls the cam slave axis to adjust to the corresponding target point according to the current position in the master axis and the position correspondence of the cam table. The execution of this instruction has no impact on the master axis.

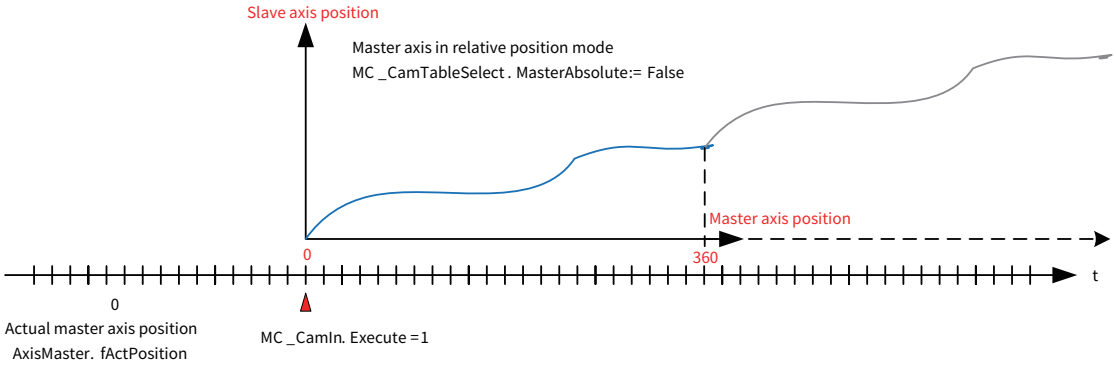


Once MC_CamIn is triggered, the slave axis moves by following the position of the master axis based on the position correspondence in the cam table. Note that the position correspondence, not the velocity correspondence, must be followed.

After entering cam running, the system will parse the CAM cam table for every EtherCAT interrupt, calculate the next target point of the slave axis based on the current position of the master axis, and send the next target position to the slave axis to make it run.



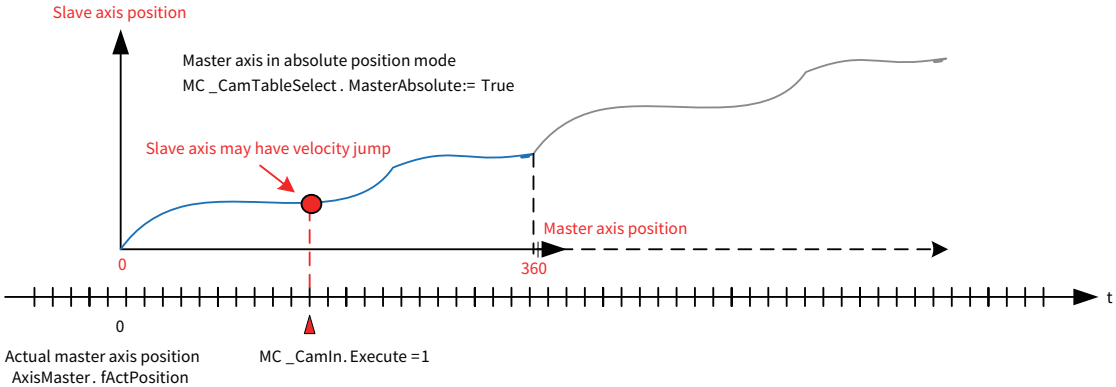
5.2.2 Master and Slave Axes in Relative Position Mode



When the master axis is in relative position mode, the cam operation module will use the current position as the start point X=0 of the master axis when entering the cam status.

When the slave axis is in relative position mode, the cam operation module will use the current position as the start point Y0 of the slave axis when entering the cam status, based on which the cam output results thereafter will be superimposed.

5.2.3 Master Axis in Absolute Position Mode and Slave Axis in Relative Position Mode

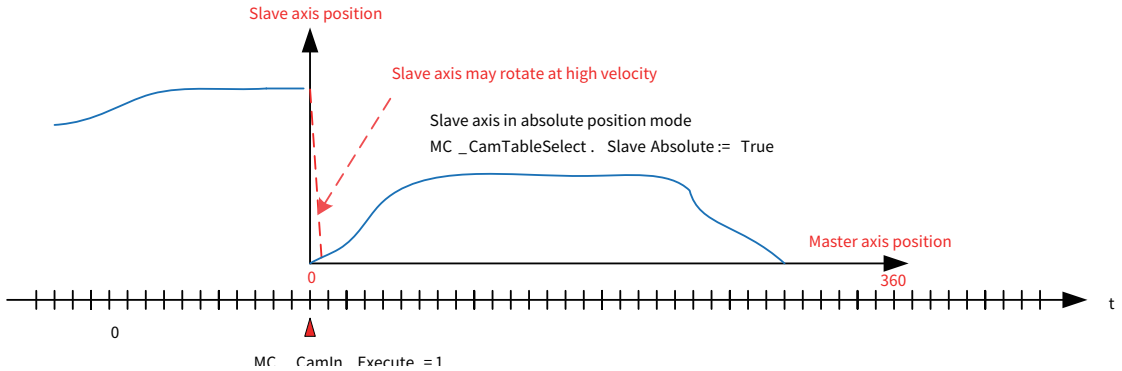


When the master axis is in absolute position mode, the cam operation module will calculate the slave axis position based on the current position of the master axis when entering the cam status. Therefore, pay attention to the following points:

- High-speed rotation during slave axis position adjustment in cam running will cause impact or damage to the device.
- If the current position is beyond the value range of the cam table, the slave axis will not move and an alarm will be generated.
- If the cam table is in cyclic mode, the next cam period starts when the execution of the current period is

completed.

5.2.4 Master Axis in Relative Position Mode and Slave Axis in Absolute Position Mode



When the slave axis is in absolute position mode, it will move to the position required by the cam when entering cam running. If the deviation is large, automatic adjustment of high-speed motion will occur.

Take measures according to the application characteristics:

- For devices requiring the alignment operation, such as the revolving knife for fixed-length cutting, the absolute position must be adopted for the cam slave axis. During programming, perform the homing operation for the revolving knife before its first rotary cutting action.
- Reasonably set the master axis position range for the cam table to avoid reverse cam position adjustment at the beginning of the next period.
- Run SMC_GetCamSlaveSetPosition to set the slave axis position of the cam entry point to the current coordinates of the slave axis.
- For applications supporting relative position mode, use relative position mode:

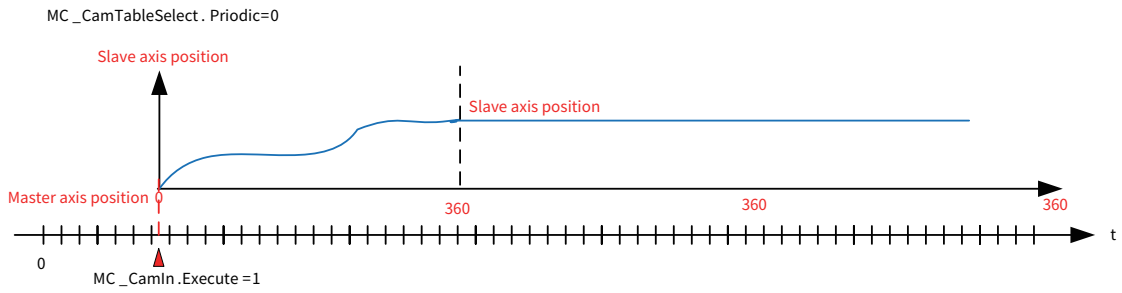
MC_CamTableSelect.SlaveAbsolute = False; or MC_CamIn.StartMode = 1; (relative mode)

Notes:

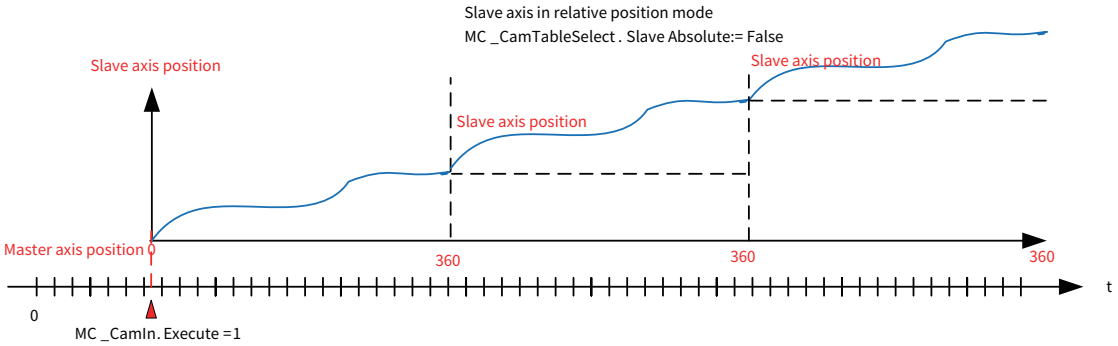
When the slave axis is set to absolute mode of "finite length", the controller will choose a closer direction when making homing adjustment if it is possible to turn left or right for the homing operation. When designing the range of the cam table, make sure that it does not exceed the actual range of operation required; otherwise, instantaneous high-speed rotation adjustment of the servo slave axis will occur, resulting in a mechanical shock.

5.3 Cyclic Mode Characteristics of the Cam Table

The following figure shows the result of single-period cam running. When the cam table is set to single-period mode (Periodic = 0), the slave axis exits the cam running status after one cam table period is completed.

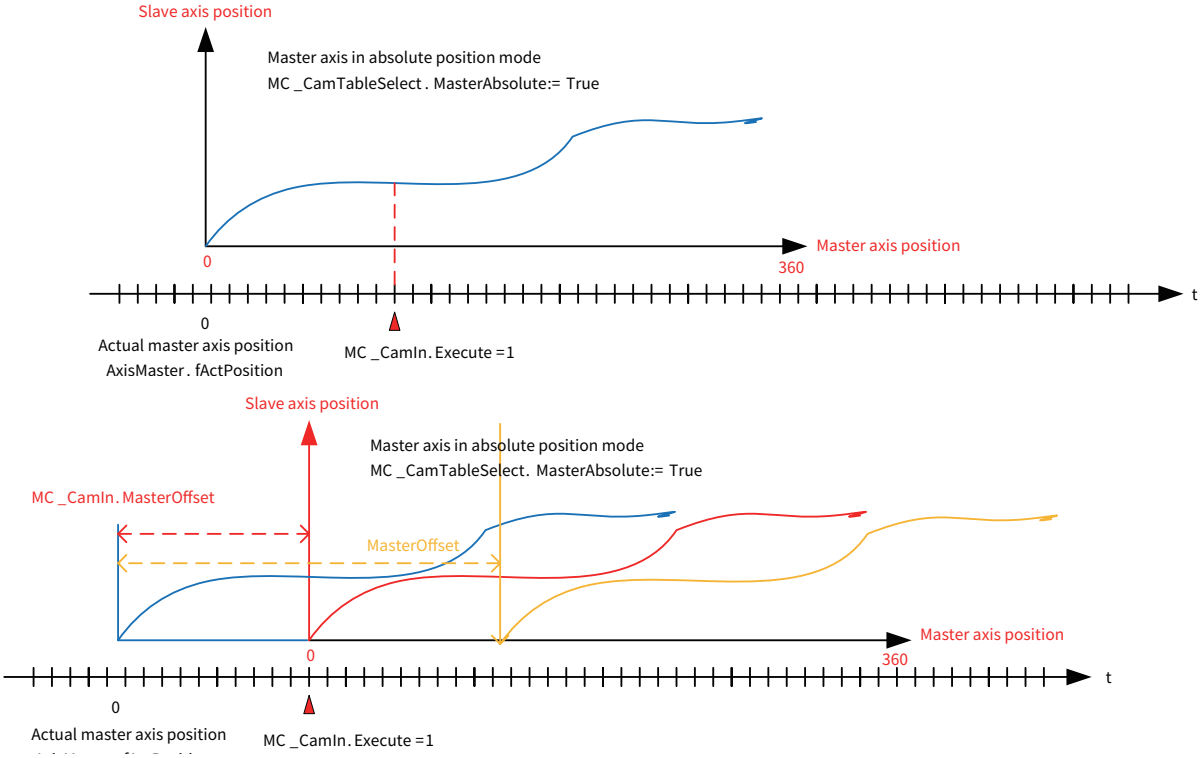


When the cam table is set to cyclic mode (Periodic = 1), the slave axis starts to run the next cam period after running for one cam table period, until a user program commands it to exit the cam running status, as shown below:



The preceding figure shows the result of cyclic cam running. Once the master axis position range in the cam table is completed, the motion of the next cam period starts automatically.

5.3.1 Offset for CamIn Operation



By setting an offset for the cam master axis, you can modify the start point of the cam slave axis. Based on the current actual position of the master axis, calculate the offset value, which can start at point 0 of the cam table:

MasterOffset = 0 - AxisMaster.fActPosition

5.3.2 Calculation of Master Axis Scaling During Cam Running

By default, MasterScaling is set to 1 in the system. If the user program modifies this variable:



After setting MasterScaling for the cam master axis, you can perform linear scaling of the master axis position so that the position correspondence with the cam table meets the requirement.

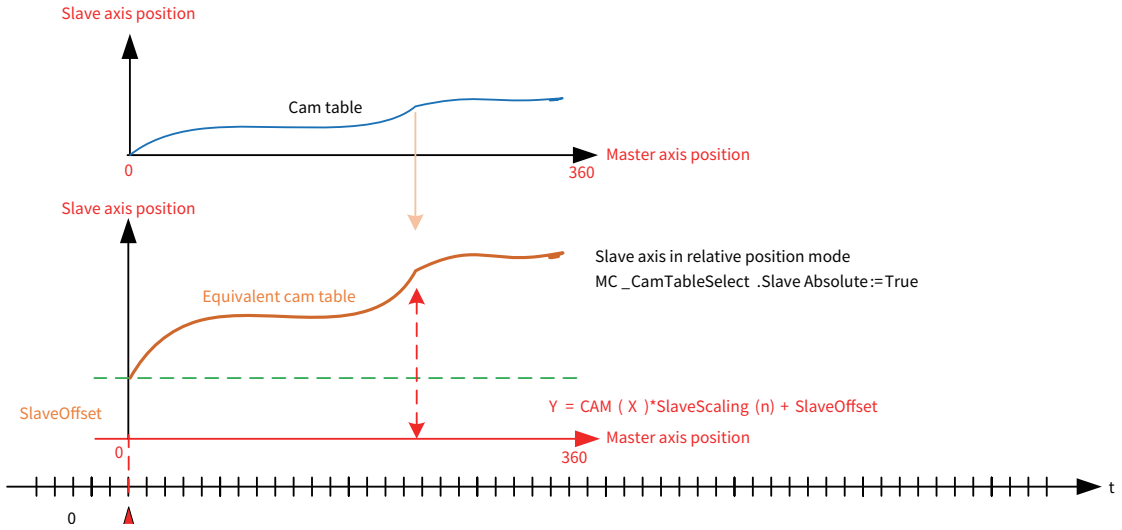
If the offset value of the master axis is considered, the calculated position of the master axis (X) in the cam table is:

X = MasterPosition x MasterScaling(n) + MasterOffset

This parameter can be used for dimensional fine-tuning of processing workpieces.

5.3.3 Calculation of Slave Axis Scaling During Cam Running

By default, SlaveScaling is set to 1 in the system. If the user program modifies this variable:



After setting SlaveScaling for the cam slave axis, you can perform linear scaling of the slave axis position so that the output of cam control meets the requirement on the slave axis motion position.

If the offset value of the slave axis is considered, the output position of the cam slave axis (Y) is:

Y = CAM(X) x SlaveScaling(n) + SlaveOffset

This parameter can be used for dimensional fine-tuning of processing workpieces.

5.3.4 Characteristics of and Precautions for Using Offset and Scale in Cam Running

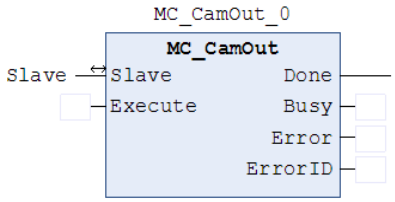
It is recommended to adopt the relative mode for the master axis position mode and the slave axis position mode, unless otherwise required by the application system. This can make the programming simple and lower the possibility of a mechanical shock.

The master axis start and stop ranges, offset, scale and other values of the cam table can make up for the design deviation of the cam table. It is recommended to use the default settings to facilitate commissioning and maintenance and reduce possible running errors.

When the cam table period is executed/aborted or after the cam table is switched, the system will clear the values of the offset and scale in the memory and restore the default values when MC_CamIn is executed again.

5.3.5 MC_CamOut FB for Exiting Cam Running Status

When the slave axis is in cam running status, triggering the execution of this function block can make the slave axis exit the cam running status and enter the continuous running status (Continuios_Motion, that is, Axis.nAxisState = 5). The execution of this instruction has no impact on the master axis.



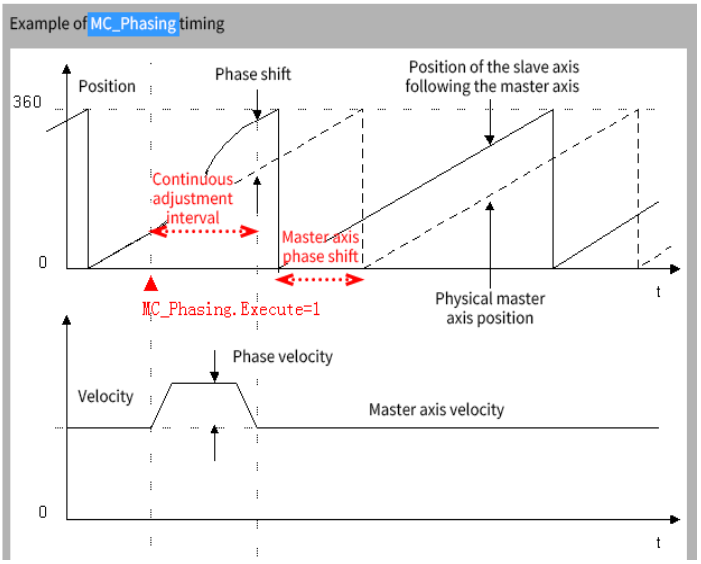

```
VAR i: INT;
CAM: MC_CAM_REF:= (
   byType:=2, (* non-equidistant *)
   byVarType:=2, (* UINT *)
   nElements:=128,
   xStart:=0,
   xEnd:=360);
Table: SMC_CAMTable_UINT_128_2:= (
    fEditorMasterMin:= 0, fEditorMasterMax:= 360,
    fTableMasterMin:= 0, fTableMasterMax:= 6000,
    fEditorSlaveMin:= 0, fEditorSlaveMax:= 360,
    fTableSlaveMin:= 0, fTableSlaveMax:= 6000);
END_VAR
```

Note: When you execute this instruction, the slave axis exits the cam running status. However, it will continue to run at the velocity the same as that when it exited the cam status, just as the driven gear continues to run by inertia even after being disconnected from the mechanical clutch. In this case, another MC function block is required to take over the motion control of the slave axis, such as MC_Movexxx, MC_Halt, and MC_Stop.

5.4 MC_Phasing FB for Cam Master Axis Phase Adjustment

In the cam synchronous operation of some devices, sometimes it is necessary to correct the relative phase between the cam master axis and the slave axis. In this case, you can use the MC_Phasing instruction.

This instruction modifies the calculation result of the cam slave axis position. You can set the velocity and acceleration constraints generated by the phase adjustment. During the adjustment, the running velocity and position of the slave axis remain continuous. After the adjustment, the phase difference is kept in continuous operation.



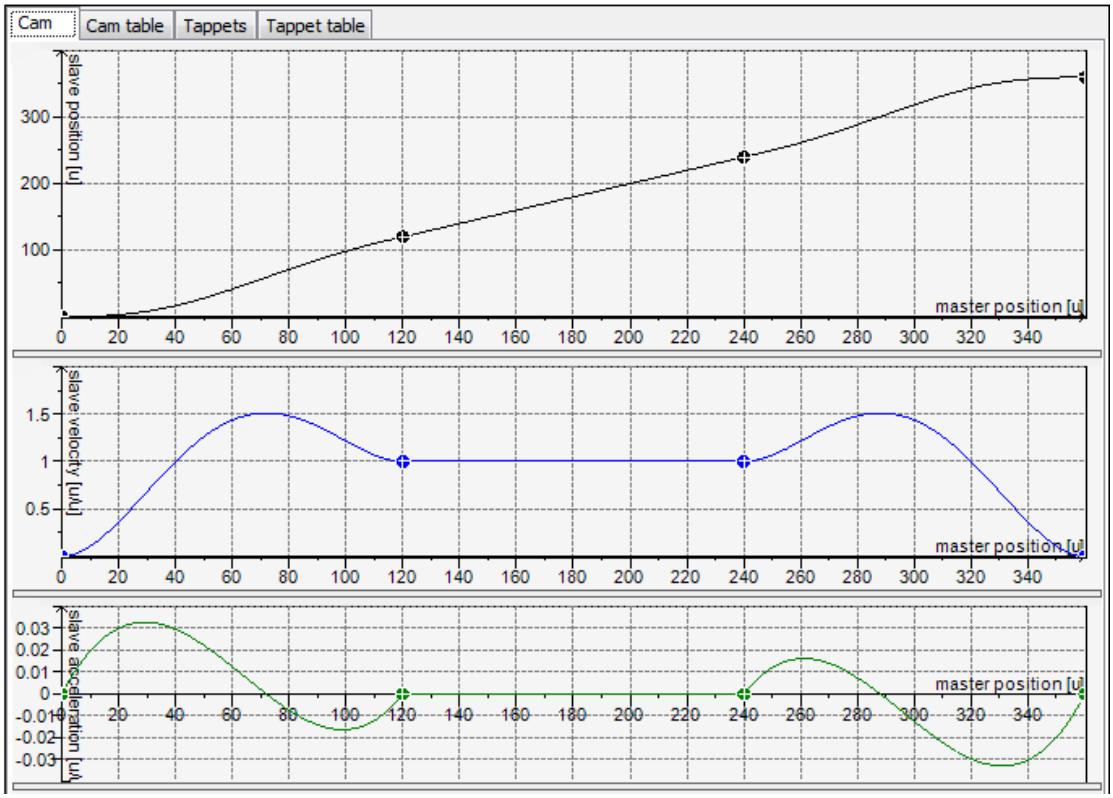
This instruction can be used to adjust the position of the color mark in the sheared workpiece segment in synchronous control.

5.5 Cam Table Design and Its Data Structure

The cam table is one of the items for writing a user program for cam running. It determines the characteristics of cam running and can adopt the graphical or tabular form.

5.5.1 Characteristics of the Cam Table

The following figure shows the cam table in graphical form. The horizontal axis indicates the master axis position, and the axis length is the travel of cam running. There are four coordinate curves, and the vertical axes indicate the slave axis position, the slave axis velocity, the slave axis acceleration rate, and the slave axis jerk, respectively. We tend to focus on the position curve and velocity curve during programming commissioning, and also the acceleration rate curve during the smoothness commissioning.



The cam curves have the following characteristics:

- In the master-slave position curve coordinates, the vertical axis indicates the allowable motion range of the slave axis. For the other three curves, the vertical axes indicate the velocity ratio and acceleration ratio between the slave axis and the master axis.
- The cam curve is monotonic in the vertical direction, that is, each coordinate of the master axis can only correspond to a unique coordinate of the slave axis. During cam execution, the coordinates of the master axis move in the ascending direction.
- The cam curve can have several key points. The line type between two key points can be set as a straight line or a quintic curve. The system will make the best optimization for each quintic curve to minimize the abrupt changes in velocity and acceleration rate.
- The start and end coordinates of the horizontal axis (master axis) start from 0 and end at 360 by default. Users can modify them based on the actual physical travel.

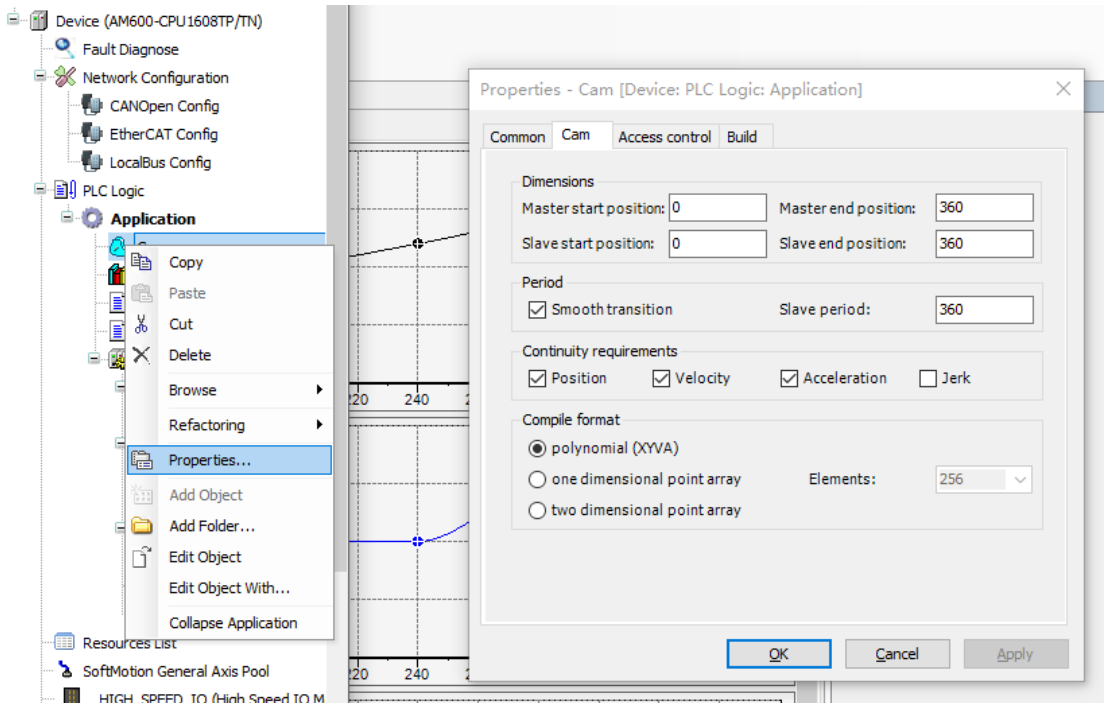
5.5.2 Input Mode of the Cam Table

30) When users want to create a cam table, the system will automatically set up the simplest cam curve, based on which users can make modifications as required.

31) Users can increase or decrease the number of key points and modify the coordinates of the key points of the cam curve.

32) Users can modify the line type between any two neighboring key points, which can be quintic curve or straight line.

33) By default, the system uses a quintic curve to link two neighboring key points in the cam curve, ensuring the continuity of the velocity during operation and reducing the mechanical shock.

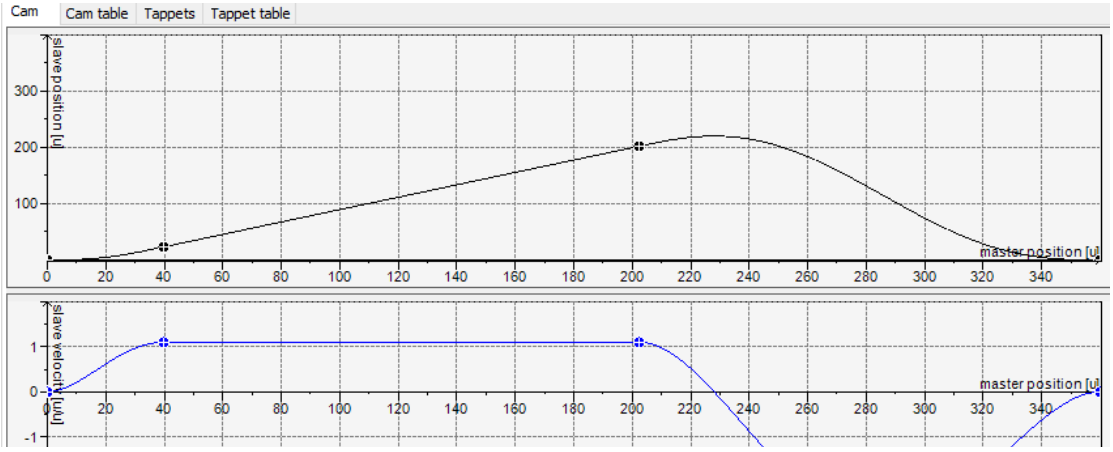


The key points in the cam curve are related to the mechanical motion requirements of the control object. Example:

34) For chasing shear applications, it is recommended that the coordinate range of the master axis correspond to the physical travel of the running interval for easy analysis.

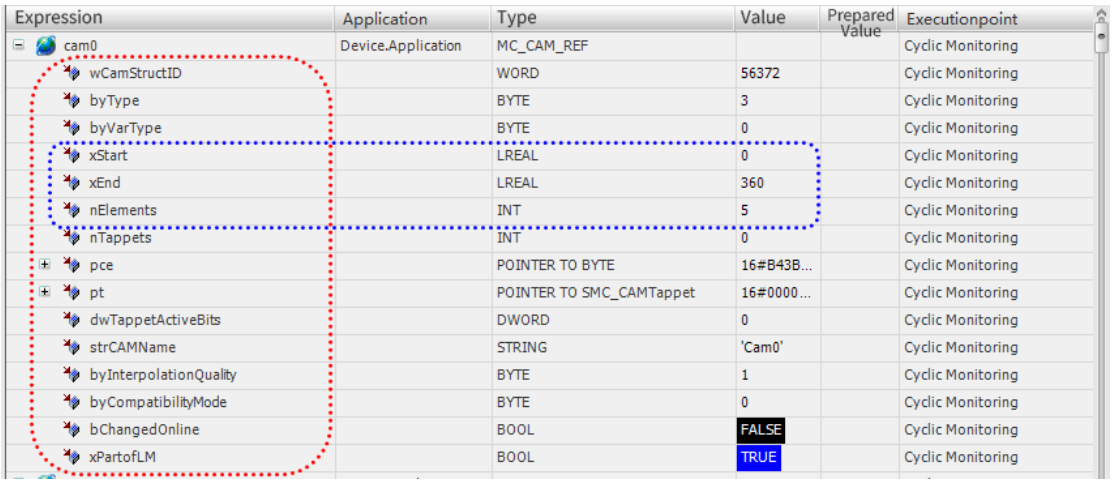
35) The key points include the start and end points of the round trip of the slave axis, the start position point of the synchronous operation interval, and the position point for out of synchronization.

36) For proportional synchronization intervals, the line segments of the cam curve should be straight lines. The line type for other intervals should be quintic curve.



5.5.3 Internal Data Structure and Arrays of the Cam Table

For each cam table in InoPro, there is a data structure describing the characteristic data of the cam table. The following figure shows the data structure describing the cam table "CAM0". Pay attention to the names of variables in the structure:



Expression	Application	Type	Value	Prepared Value	Executionpoint
cam0	Device.Application	MC_CAM_REF			Cyclic Monitoring
wCamStructID		WORD	56372		Cyclic Monitoring
byType		BYTE	3		Cyclic Monitoring
byVarType		BYTE	0		Cyclic Monitoring
xStart		LREAL	0		Cyclic Monitoring
xEnd		LREAL	360		Cyclic Monitoring
nElements		INT	5		Cyclic Monitoring
nTappets		INT	0		Cyclic Monitoring
pce		POINTER TO BYTE	16#B43B...		Cyclic Monitoring
pt		POINTER TO SMC_CAMTappet	16#0000...		Cyclic Monitoring
dwTappetActiveBits		DWORD	0		Cyclic Monitoring
strCAMName		STRING	'Cam0'		Cyclic Monitoring
byInterpolationQuality		BYTE	1		Cyclic Monitoring
byCompatibilityMode		BYTE	0		Cyclic Monitoring
bChangedOnline		BOOL	FALSE		Cyclic Monitoring
xPartofLM		BOOL	TRUE		Cyclic Monitoring

InoPro has an internal data structure that describes the characteristics of the cam table. Users can also write a cam table manually, as shown below:
Although it is not necessary to manually write a cam table, we can modify the desired cam characteristic data by accessing the data structure.

Note: When declaring the cam table CAM0, the system automatically declares the CAM0 data structure of the global variable type by default, along with the CAM0_A[i] array.

For example, to modify the number or coordinates of key points of the cam table CAM0 in the user program:

```
CAM0. nElements:=20; //Change the number of key points to 20
  CAM0. xEnd:=500; //Change the end point of the master axis to 500. //For example, modify the coordinates of 2 key points in the user program:
    CAM0_A[3].dx:=30;
    CAM0_A[3].dy:=45;
    CAM0_A[3].dv:=1;
    CAM0_A[3].da:=0;

    CAM0_A[4].dx:=60;
    CAM0_A[4].dy:=75;
    CAM0_A[4].dv:=1;
    CAM0_A[4].da:=0;
```

Method of modifying the cam table online

"Online modification of the cam curve" refers to the modification of the key point coordinates of the cam curve based on the control characteristics during the execution of the program written by the user.

The modification generally involves the key point coordinates. Users can also modify the number of key points, the distance range of the master axis, and so on.

Note: Modify the cam table before entering cam running instead of during running; otherwise, unanticipated movement results may be caused.

Applications requiring modification of the cam table:

37) In general, OEM customers use cam tables that have been verified by commissioning.

38) If there are several processing objects or modes, multiple cam tables can be preset for automatic switchover according to users' needs.

39) Some devices may require a wider range of adaptability. For example, if the packaging device requires an applicable packaging length in the range of 10 cm to 25 cm, and the corresponding running velocity must be automatically changed, it may be necessary to modify the cam table online.



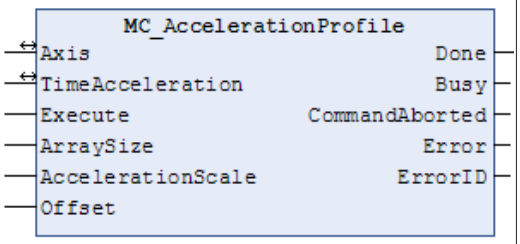
6. Common MC Instructions

6. Common MC Instructions

6.1 Single-axis Instructions

MC_AccelerationProfile

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ AccelerationProfile	Acceleration profile instruction		MC_AccelerationProfile(Axis:= , TimeAcceleration:= , Execute:= , ArraySize:= , AccelerationScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
TimeAcceleration	Axis acceleration time and acceleration description	MC_TA_REF	-	-	Axis acceleration time and acceleration data description; acceleration data consists of multiple sets of data

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
ArraySize	Dynamic array	INT	Value Range	0	The number of arrays used in the operation profile
AccelerationScale	Integration factor	LREAL	"Positive" + "0"	1	Scale factor of the acceleration or deceleration in MC_TA_REF
Offset	Offset	LREAL	-	0	Overall offset of the acceleration or deceleration

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction execution completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of axis instruction is completed
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed

CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ ERROR	0	Output an error code when an error occurs

3) Function Description

- This function block is a profile motion model of time period and acceleration/deceleration. The operation mode is Discrete Motion. It runs based on the data set by the user for the TimeAcceleration variable.
- This function block can run in Standstill, Continuous Motion, Synchronized Motion, or Discrete Motion status. The status during instruction running is Discrete Motion. It cannot run in other statuses.
- The function block is started at the rising edge of Execute. The velocity of this instruction is superimposed based on the previous one when it is run repeatedly in Discrete Motion, which tends to cause system failure.
- TimeAcceleration is of the MC_TA_REF data type.

  MC_TA_REF description:

Member	Type	Initial Value	Description
Number_of_pairs	INT	0	Number of profile path segments
IsAbsolute	BOOL	TRUE	Absolute motion (TRUE) and relative motion option
MC_TA_Array	ARRAY[1..N] OF SMC_TA	-	Array of time and acceleration values

SMC_TA description:

Member	Type	Initial Value	Description
delta_time	TIME	TIME#0ms	Time of acceleration segment
Acceleration	LREAL	0	Current acceleration value

Note: The set acceleration is reflected in the change of velocity. All the acceleration changes are reflected in an S-curve. Therefore,

the acceleration data for the final result of (A+B)/2 (A: Start acceleration; B: End acceleration) is reflected in the final velocity.

4) Timing Diagram

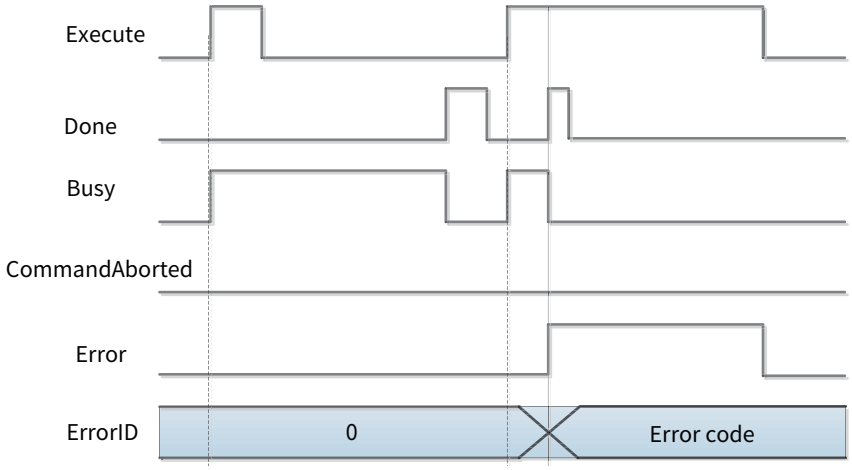
The condition MC_TA_Array has been set by other means.

The instruction can run only when the axis is in Standstill status.

Execute of the function block must have a rising edge condition.

Done of the function block indicates that the execution of the instruction is completed.

Busy of the function block indicates that the execution of the instruction is in progress.



5) Error Description

The error occurs as the instruction is not started in the axis status of Standstill or there is a parameter error in the instruction system. An axis error must be cleared before the start of the operation.

Note: For details on the error code, see “ “Appendix C Error Codes” .

MC_Halt

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_Halt	Axis stop instruction	MC_Halt MC_Halt Axis Execute Deceleration Jerk Done Busy CommandAborted Error ErrorID	MC_Halt(Axis:= , Execute:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
Deceleration	Deceleration	LREAL	"Positive" + "0"	0	Deceleration of the function block (u/s^2)
Jerk	Jerk	LREAL	"Positive" + "0"	0	Specify the jerk [reference unit/s^3]

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description

Done	Instruction execution completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of axis instruction is completed
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
CommandAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ ERROR	0	Output an error code when an error occurs

3) Function Description

- This function block stops the motion of one axis under normal operation. The execution of this instruction can be terminated when another axis instruction is run again.
- This function block can run only in the Motion status and cannot run in any other status.
- The function block starts at the rising edge of Execute.
- The function block is in the Discrete Motion status during instruction execution and in the Standstill status after the completion of the instruction.
- When the motion instruction is aborted by MC_Halt and MC_Stop, users can adjust the acceleration by setting the axis variable bAvoidReversalOnHaltStop to TRUE. This avoids the negative direction of velocity during the stopping process. When the instruction is halted or stopped, if the velocity value at the breakpoint is small, the acceleration rate is large, and the jerk is small, the velocity may not be able to reduce to 0 directly, but reduces to 0, reversely accelerates, and finally reversely decelerates to 0. As the acceleration rate is large and the jerk is small, the velocity value at which the acceleration is reduced to 0 by the current maximum jerk is greater than the velocity value at the breakpoint. Therefore, the velocity direction must be reversed so that the velocity can be reduced to 0 at the same time as the acceleration. This phenomenon occurs in quadratic_ramp and quadratic_smooth_ramp.

4) Timing Diagram

The instruction can be run only when the axis is in the Motion status.

Execute of the function block must have a rising edge condition.

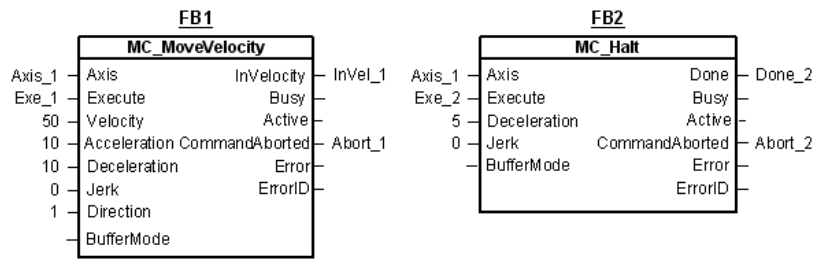
Done of the function block indicates that the execution of the instruction is completed.

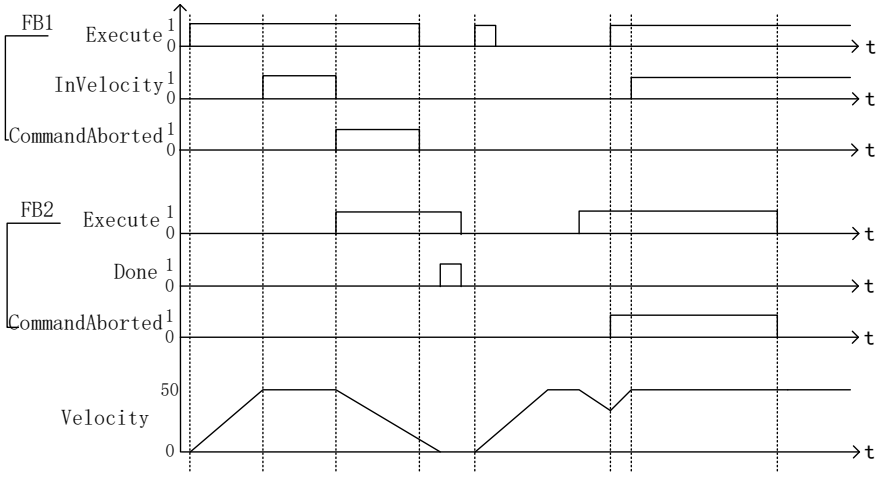
Busy of the function block indicates that the execution of the instruction is in progress.

CommandAborted of the function block indicates that the instruction is aborted by other motion control instructions, in which case the flag bit is TRUE;

Programming example: Changes in the flag bits of the MC_MoveVelocity instruction and MC_Halt instruction in different timing operations;

The processing of CommandAborted is described in the following timing diagram.





5) Error Description

The error occurs as the instruction is not started in the axis status of Standstill or there is a parameter error in the instruction system. An axis error must be cleared before the start of the operation.

Note: For details on the error code, see “ “Appendix C Error Codes” .

MC_HaltSuperImposed

This function requires all superimposed motions of an axis to be halted, and the basic motion will not be interrupted.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_Halt Superimposed	Superimposed motion halted	MC_HaltSuperImposed_0 MC_HaltSuperImposed EN ENO Axis Done Execute Busy Deceleration CommandAborted Jerk Error ErrorID	MC_HaltSuperImposed_0(Axis:= Axis, Execute:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF	-	-	Reference to the axis, that is, an instance of AXIS_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Start	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
Deceleration	Deceleration	LREAL	-	0	Deceleration rate in the deceleration phase [u/s^2]
Jerk	Jerk	LREAL	-	0	Slope change of the axis acceleration and deceleration [u/s^3]

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is completed
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
CommandAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ ● This instruction suspends all superimposed motions of an axis without interrupting the basic motion.

◆ ● This instruction can be triggered repeatedly.

◆ ● This instruction supports multi-trigger, which must be after the basic motion. If the multi-trigger occurs before the basic motion instruction, the basic motion instruction will report an error and cause the axis to be disconnected.

◆ This instruction must be triggered after the superimposed instruction. If it is triggered before the superimposed instruction, this instruction will report an error. If there is no superimposed instruction being executed in the system, triggering this instruction will result in an error. If the superimposed instruction runs separately, that is, if it runs as a relative motion instruction, triggering this instruction will directly set it to "done" without affecting the superimposed instruction.

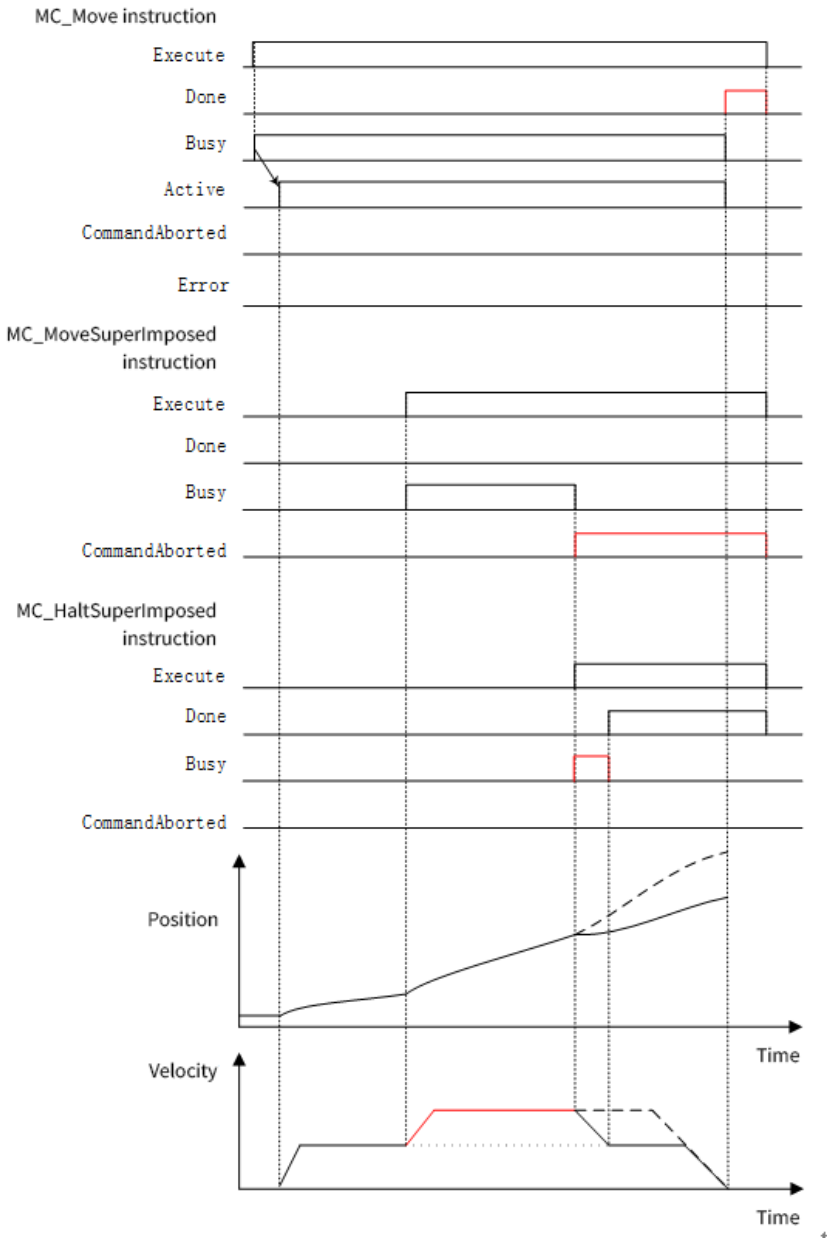
◆ This instruction cannot be triggered separately.

◆ If the superimposed instruction is triggered during the operation of this instruction, the superimposed instruction will be aborted directly.

◆ ● This instruction does not allow multiple instructions to share the instance name; otherwise, the abrupt change of position will cause the axis to be disconnected.

4) Timing Diagram

After the MC_Move instruction is activated, the superimposed instruction is triggered. The timing diagram when this instruction is activated in the superimposed state is shown below.



MC_Home

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_Home	Axis homing instruction	MC_Home MC_Home Axis Execute Position Done Busy CommandAborted Error ErrorID	MC_Home(Axis:= Axis, Execute:= , Position:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
Position	Axis reached position	LREAL	Value Range	0	Indicate the homing position of the axis position

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction execution completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of axis instruction is completed
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ ERROR	0	Output an error code when an error occurs

3) Function Description

◆ This function block performs a homing operation. The position data is the zero point position of the axis.

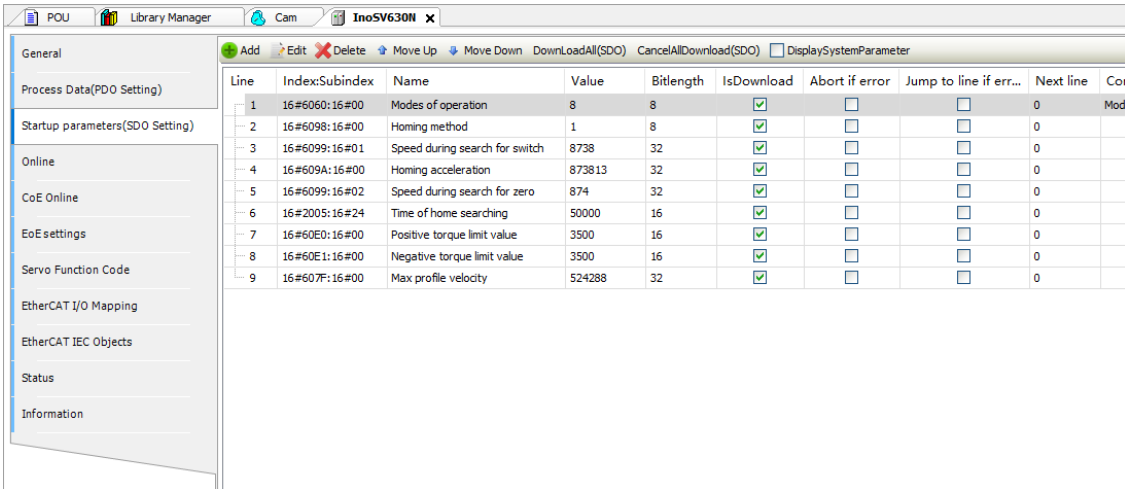
◆ The operation status of this function block is Standstill, and the status during instruction running is Homing. It cannot run in other statuses.

◆ The function block starts at the rising edge of Execute.

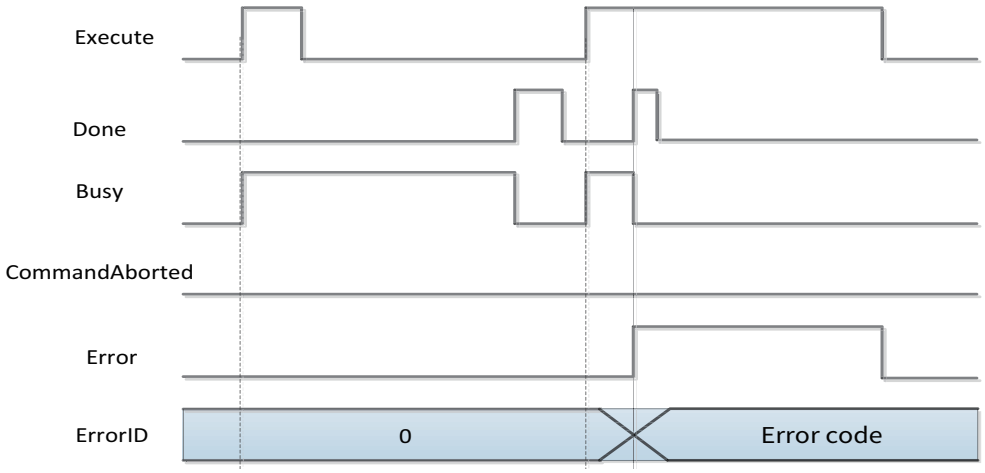
◆ Instructions for setting Inovance servo: When performing the homing operation for each servo axis, users must set the homing mode of servo parameters. The setting mode allows manual setting of the servo function code. The corresponding function code can also be configured through the startup parameter of the AM600 slave. The following indexes and sub-indexes must be set if the communication mode is adopted.

Item	Index	Sub-index	Description
Homing mode	0x6098		Select the specific parameters to be set according to the servo guide.
Velocity during search for home	0x6099	0x01	Generally, the defined velocity is high to reduce the homing time
Speed during search for zero	0x6099	0x02	Generally, the defined velocity is low
Homing acceleration and deceleration	0x609A		Acceleration/deceleration change during homing
Homing timeout	0x2005	0x24	If the homing time exceeds the set time, the system reports Err.601

Example of AM600 setting:



4) Timing Diagram



MC_MoveAbsolute

Specify the target position in absolute coordinates for positioning.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ MoveAbsolute	Instruction for absolute axis position control	MC_MoveAbsolute_0 MC_MoveAbsolute EN Axis Execute Position Velocity Acceleration Deceleration Jerk Direction BufferMode ENO Done Busy Active CommandAborted Error ErrorID	MC_MoveAbsolute_0(Axis:= Axis, Execute:= , Position:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Direction:= , BufferMode:= , Done=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
Position	Axis reached position	LREAL	Value Range	0	This position is the absolute position of the axis
Velocity	Running velocity	LREAL	Value Range	0	Maximum velocity of the axis to the target position
Acceleration	Acceleration	LREAL	Value Range	0	Acceleration rate for velocity increase
Deceleration	Deceleration	LREAL	Value Range	0	Deceleration rate for velocity decrease
Jerk	Jerk	LREAL	Value Range	0	Slope change of the curve acceleration/ deceleration
Direction	Polarity	MC_ DIRECTION	Negative, shortest Positive, current, fastest	shortest	Negative: Move in negative direction. Shortest: Select direction based on shortest path. Positive: Move in positive direction. Current: Move in current direction. Fastest: Automatically select the fastest direction. (This function axis is effective in rotation mode.)
BufferMode	Buffer Mode	MC_BUFFER_ MODE	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	0	Specify the action to be taken when multiple instances initiate a motion instruction. 0: Aborting 1: Buffered 2: Blend at the low velocity 3: Blend at the previous velocity 4: Blend at the next velocity 5: Blend at the high velocity

◆ Output Variable

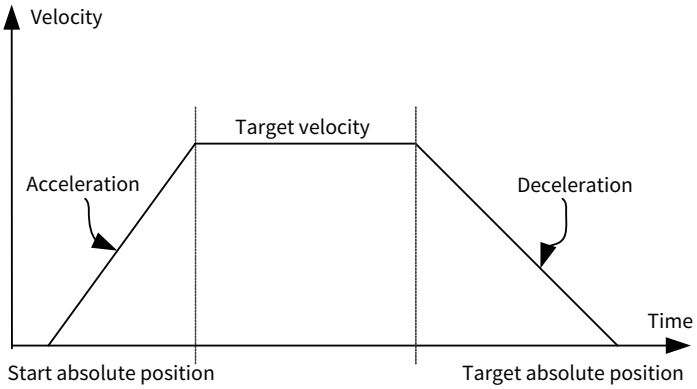
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction execution completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of axis instruction is completed
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Active	Control	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the function block has control on the axis
CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

◆ This function block is an instruction for absolute axis positioning. The position data is the absolute position of the axis.

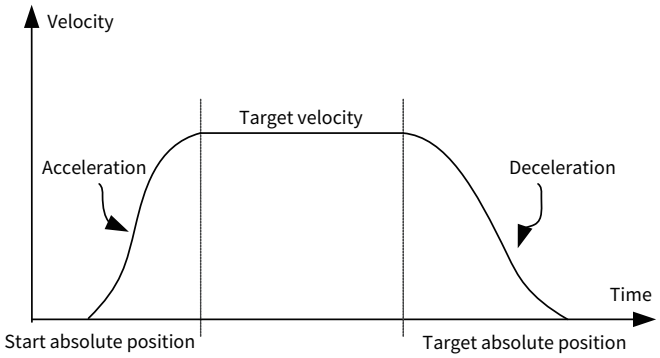
- The operation status of this function block is Standstill, and the status during instruction running is Discrete Motion. A complete running process must control the different motion statuses of the axis.
- The motion is started at the rising edge of Execute. This instruction can be rising edge-triggered repeatedly in Discrete Motion to refresh the latest position data each time.
- If Acceleration or Deceleration is zero, the instruction execution will be abnormal. However, the state of the axis is Discrete Motion.
- Trapezoid acceleration/deceleration action

  There is data for Velocity, Acceleration and Deceleration. Jerk is 0.



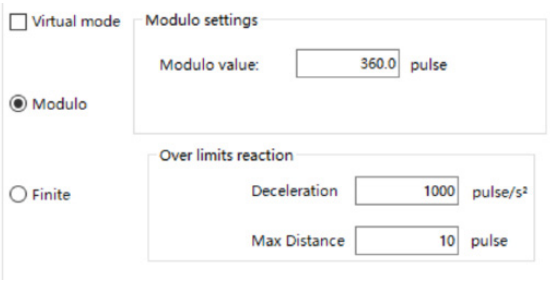
- S-curve acceleration/deceleration action

  There is data for Velocity, Acceleration, Deceleration and Jerk.

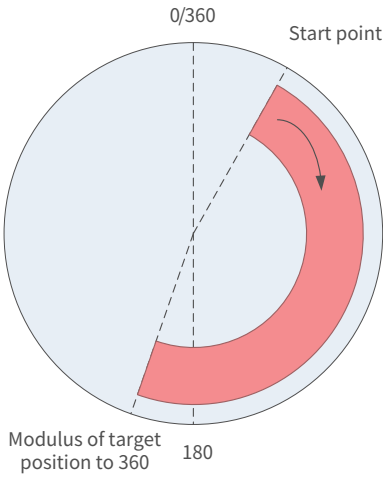


- Absolute positioning of axis in cyclic mode

  1) The axis rotation period is set to 360 and the direction is set to Positive.



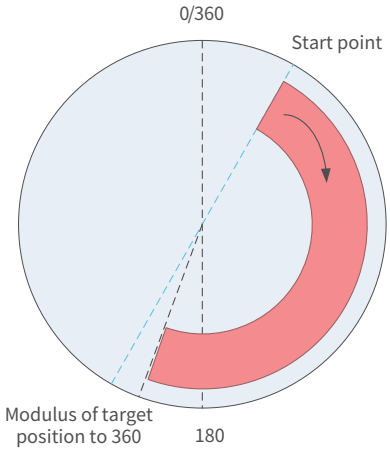
When the modulus value of Position to 360 (taking the remainder of Position/360, for example, if Position is 380, then the modulus value to 360 is 20; if Position is 350, then the modulus value to 360 is 350) is greater than the start absolute position, then the axis moves in positive direction for such distance: Modulus value of Position to 360 - Start absolute position.



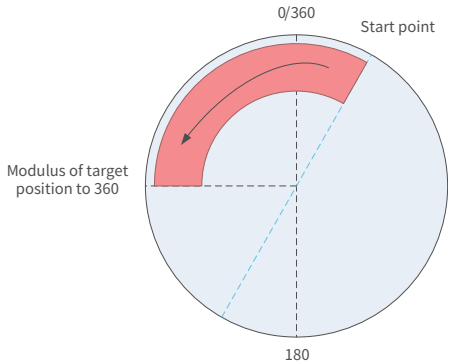
When the modulus value of Position to 360 (taking the remainder of Position/360, for example, if Position is 380, then the modulus value to 360 is 20) is smaller than the start absolute position, then the axis moves in positive direction for such a distance: 360 - Start absolute position + Modulus value of Position to 360.

2) The axis rotation period is set to 360, and the direction is set to Shortest or Fastest. The modulus of Position to 360 is XPosition.

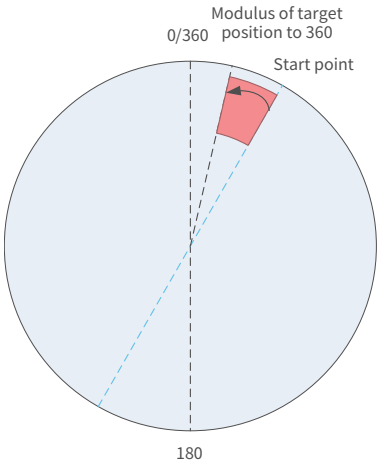
If 0 =< Xposition - Start absolute position < 180, then the axis moves in positive direction for such a distance: Xposition - Start absolute position.



If 180 < XPosition - Start absolute position, then the axis moves in negative direction for such a distance: 360 - XPosition + Start absolute position.

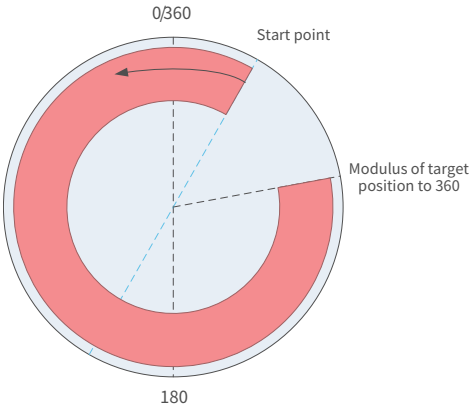


If XPosition < Start absolute position, then the axis moves in negative direction for such a distance: Start absolute position - XPosition.



3) The axis rotation period is set to 360, and the direction is set to Shortest or Negative. The modulus of Position to 360 is XPosition.

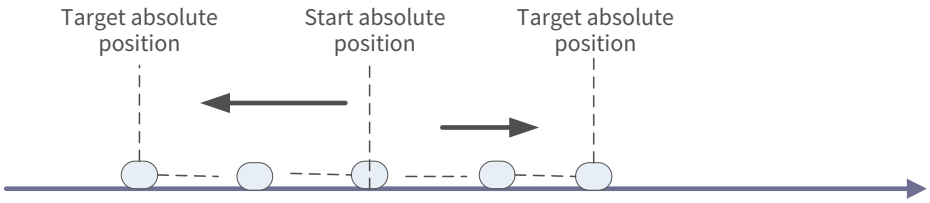
The axis moves in negative direction for such a distance: Start absolute position + 360 - XPosition.



◆ Absolute positioning of axis in linear mode

If Target absolute position > Start position, then the axis moves in positive for such a distance: Target absolute position - Start position. If Target position < Start position,

then the axis moves in negative direction for such a distance: Start position - Target position. The running direction set in linear mode does not determine the running direction of the axis.



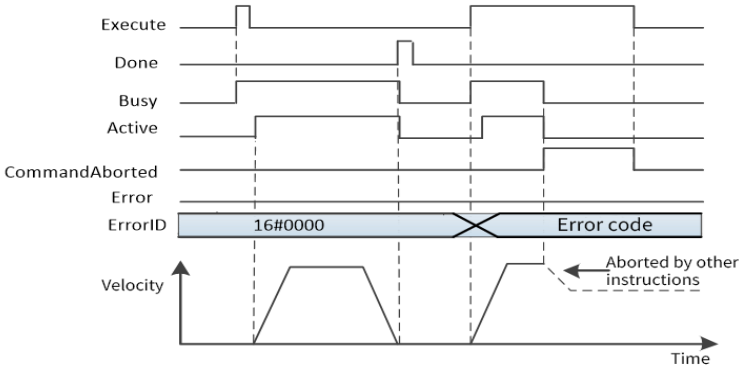
4) Precautions

* When Direction is set to "2: Current", motion is performed in the direction of the instruction of the previous motion. Therefore, depending on the combination of instructions, the direction of the instruction may be different from that of the input of the previous motion instruction.
* If the relative displacement (difference between the input absolute displacement and the current displacement) is not 0 but Velocity (target velocity) is 0, the instruction cannot run normally.
* If both the relative displacement (difference between the input absolute displacement and the current displacement) and Velocity (target velocity) are not 0, if the input variable Acceleration, Deceleration, or Jerk is 0, the default initial value is assigned. For details, see “6.3.3 Defaults of Motion Control Function Blocks” on page 238.

When both the relative displacement (difference between the input absolute displacement and the current displacement) and Velocity (target velocity) are 0, the instruction is set to Done.

5) Timing Diagram

- The value of Busy changes to TRUE when Execute is started. The value of Active changes to TRUE in the next period.
- The value of Done changes to TRUE when the positioning is completed at Position.
- When this instruction is aborted by another instruction, the value of CommandAborted changes to TRUE and those of Busy and Active change to FALSE.



- Motion re-execution instruction

The motion of this instruction can be changed by changing the input variables in the positioning motion and setting Execute to TRUE again.

The input variables that can be changed for the motion re-execution instruction include Position, Velocity, Acceleration, and Deceleration.

- Start of this instruction during the execution of other instructions

When this instruction is started for the currently executing instruction, it will be switched or cached to this instruction.

The action when multiple instances of this instruction are started is determined by BufferMode.

Buffer Mode	Description
Aborting	Immediately aborts the currently executing instruction and switches to this instruction. If the direction of axis motion is reversed due to instruction switching, reverse running is performed after the velocity is decelerated to zero.
Buffered	The function block is started immediately after the last instruction motion is terminated. No blending is performed here. When the end conditions (such as Done, InVelocity, InEndVelocity, InGear, InSync, EndOfProfile) are reached, the new motion starts at the velocity of the previous motion. If the previous motion was MC_MoveAbsolute or MC_MoveRelative, the new motion will start in static state.
Blending	Starts at the velocity (relay velocity) at which the currently executing instruction reaches the target position, and continuously makes the cached instruction take motions. Change the motion of the currently executing instruction, ensuring that the target position is reached at the relay velocity. There are four ways to specify the relay velocity:

	Blend at the low velocity (BlendingLow)	The function block is started immediately after the last instruction motion is terminated. The axis does not stop between motions but passes through the end position of the first motion at the lower velocity of the two motion instructions.
	Blend at the previous velocity (BlendingPrevious)	The function block is started immediately after the last instruction motion is terminated. The axis does not stop between motions but passes through the end position of the first motion at the velocity of the first motion instruction.
	Blend at the next velocity (BlendingNext)	The function block is started immediately after the last instruction motion is terminated. The axis does not stop between motions but passes through the end position of the first motion at the velocity of the second motion instructions.
	Blend at the high velocity (BlendingHigh)	The function block is started immediately after the last instruction motion is terminated. The axis does stop between motions but passes through the end position of the first motion at the higher velocity of the two motion instructions.

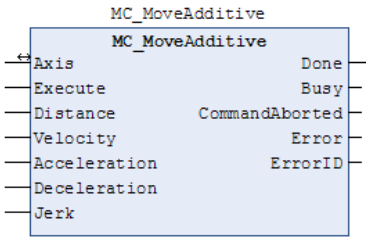
◆ Motion re-execution instruction

When starting motion instructions for multiple instances by using this instruction, users can choose the aborting, buffered, or blending mode.

MC_MoveAdditive

The axis is superimposed with the data specified by Distance based on the original instruction position, which is used for online position superimposition for the motion axis control process. In Discrete Motion status, this instruction can add the MC_MoveAddtive execution process at any time. In Continuous Motion status, it can only be in a certain section of the instruction execution. In Standstill state, it is equivalent to the MC_MoveRelative instruction.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ MoveAdditive	Superimposed absolute motion instruction		MC_MoveAdditive(Axis:= , Execute:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
Distance	Axis reached position	LREAL	Value Range	0	This data is the superimposed position data.
Velocity	Running velocity	LREAL	Value Range	0	Maximum velocity of the axis to the target position
Acceleration	Acceleration	LREAL	Value Range	0	Acceleration rate for velocity increase

Deceleration	Deceleration	LREAL	Value Range	0	Deceleration rate for velocity decrease
Jerk	Jerk	LREAL	Value Range	0	Slope change of the curve acceleration/ deceleration

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction execution completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of axis instruction is completed
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ ERROR	0	Output an error code when an error occurs

3) Function Description

◆ This function block commands a controlled motion over a specified relative distance in addition to the most recent commanded position.

◆ When this function block is in Discrete Motion status, it causes CommandAbort of other instructions to be set.

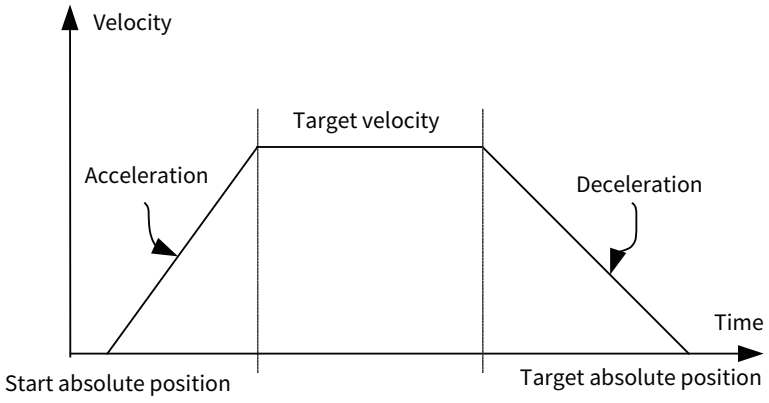
◆ In Standstill state, it can run independently to achieve relative positioning.

◆ If Acceleration or Deceleration is zero, the instruction execution will be abnormal. However, the state of the axis is Discrete Motion.

◆ The function block starts at the rising edge of Execute.

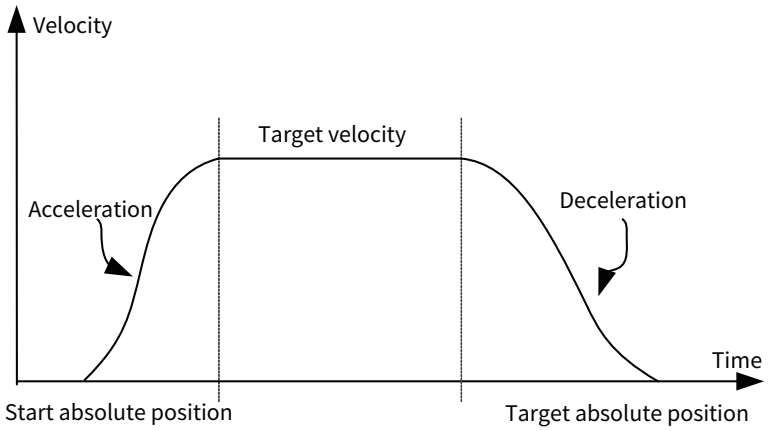
Trapezoid acceleration/deceleration action

There is data for Velocity, Acceleration and Deceleration. Jerk is 0.



S-curve acceleration/deceleration action

There is data for Velocity, Acceleration, Deceleration and Jerk.



4) Timing Diagram

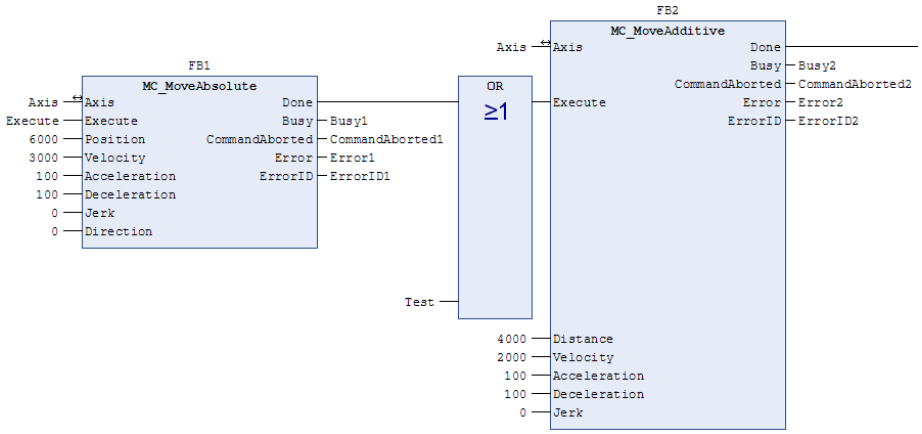
The instruction can run only when the axis is in Standstill status.

Execute of the function block must have a rising edge condition.

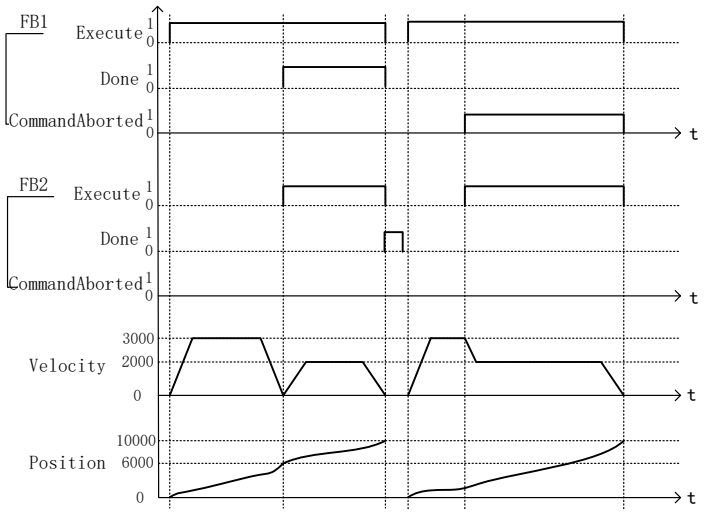
Done of the function block indicates that the execution of the instruction is completed.

Busy of the function block indicates that the execution of the instruction is in progress.

◆ Example



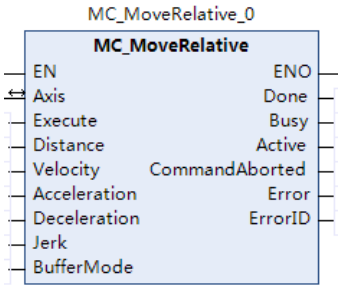
◆ Timing operation description:



MC_MoveRelative

This function block specifies the motion distance from the current position to perform positioning.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_MoveRelative	Axis relative positioning instruction		MC_MoveRelative_0(Axis:= Axis, Execute:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , BufferMode:= , Done=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
Distance	Relative position of the motion	LREAL	Value Range	0	This data is the relative position of the motion.
Velocity	Running velocity	LREAL	Value Range	0	Maximum velocity of the axis to the target position
Acceleration	Acceleration	LREAL	Value Range	0	Acceleration rate for velocity increase
Deceleration	Deceleration	LREAL	Value Range	0	Deceleration rate for velocity decrease
Jerk	Jerk	LREAL	Value Range	0	Slope change of the curve acceleration/deceleration
BufferMode	Buffer Mode	MC_BUFFER_MODE	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	0	Specify the action to be taken when multiple instances initiate a motion instruction. 0: Aborting 1: Buffered 2: Blend at the low velocity 3: Blend at the previous velocity 4: Blend at the next velocity 5: Blend at the high velocity

◆ Output Variable

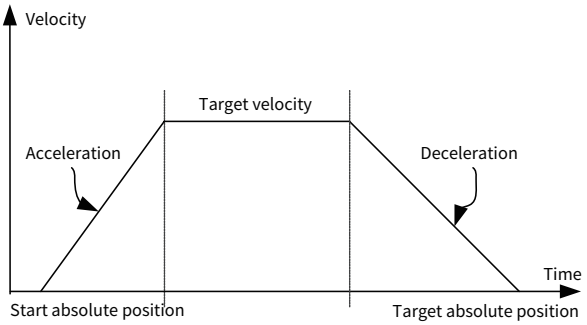
Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction execution completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of axis instruction is completed

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Active	Control	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the function block has control on the axis
CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

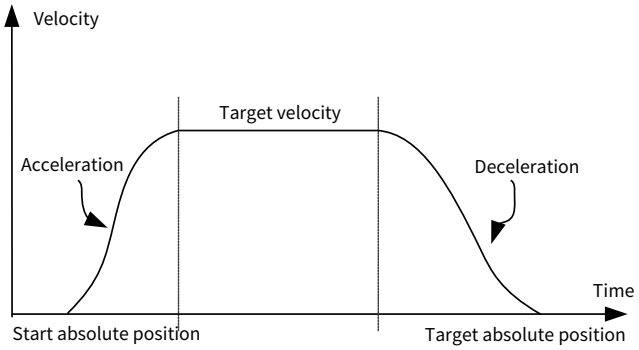
- This function block specifies the motion distance from the current position of the instruction to perform positioning.
- The operation status of this function block is Standstill, and the status during instruction running is Discrete Motion.
- The motion is started at the rising edge of Execute. This instruction can be rising edge-triggered repeatedly to refresh the latest position data each time.
- You can specify input variables Velocity, Acceleration, Deceleration and Jerk.
- Trapezoid acceleration/deceleration action

  There is data for Velocity, Acceleration and Deceleration. Jerk is 0.



- S-curve acceleration/deceleration action

  There is data for Velocity, Acceleration, Deceleration and Jerk.



4) Precautions

* If the input variable Distance (relative displacement) is not 0 but Velocity (target velocity) is 0, the instruction cannot run normally.

* If both the input variable Distance (relative displacement) and Velocity (target velocity) are not 0, and the input variable Acceleration, Deceleration, or Jerk is 0, the default initial value is assigned.

For details, see “6.3.3 Defaults of Motion Control Function Blocks” on page 238.

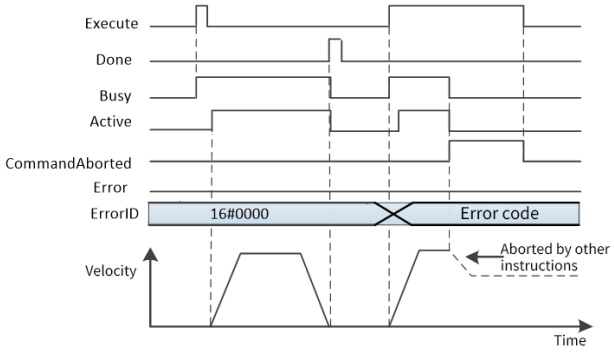
* If the input variables Distance and Velocity are 0, the instruction is set to Done.

5) Timing Diagram

The value of Busy changes to TRUE when Execute is started. The value of Active changes to TRUE in the next period.

When the Distance is reached and positioning is completed, the value of Done changes to TRUE.

When this instruction is aborted by another instruction, the value of CommandAborted changes to TRUE and those of Busy and Active change to FALSE.



◆ Motion re-execution instruction

The motion of this instruction can be changed by changing the input variables in the positioning motion and setting Execute to TRUE again.

The input variables that can be changed for the motion re-execution instruction include Distance, Velocity, Acceleration, and Deceleration.

The start point of Distance for re-execution is the current position of the instruction.

◆ Start of this instruction during the execution of other instructions

When this instruction is started for the currently executing instruction, it will be switched or cached to this instruction.

The action when multiple instances of this instruction are started is determined by BufferMode.

Buffer Mode	Description
Aborting	Immediately aborts the currently executing instruction and switches to this instruction. If the direction of axis motion is reversed due to instruction switching, reverse running is performed after the velocity is decelerated to zero.
Buffered	The function block is started immediately after the last instruction motion is terminated. No blending is performed here. When the end conditions (such as Done, InVelocity, InEndVelocity, InGear, InSync, EndOfProfile) are reached, the new motion starts at the velocity of the previous motion. If the previous motion was MC_MoveAbsolute or MC_MoveRelative, the new motion will start in static state.
Blending	Starts at the velocity (relay velocity) at which the currently executing instruction reaches the target position, and continuously makes the cached instruction take motions. Change the motion of the currently executing instruction, ensuring that the target position is reached at the relay velocity. There are four ways to specify the relay velocity:

	Blend at the low velocity (BlendingLow)	The function block is started immediately after the last instruction motion is terminated. The axis does not stop between motions but passes through the end position of the first motion at the lower velocity of the two motion instructions.
	Blend at the previous velocity (BlendingPrevious)	The function block is started immediately after the last instruction motion is terminated. The axis does not stop between motions but passes through the end position of the first motion at the velocity of the first motion instruction.
	Blend at the next velocity (BlendingNext)	The function block is started immediately after the last instruction motion is terminated. The axis does not stop between motions but passes through the end position of the first motion at the velocity of the second motion instructions.
	Blend at the high velocity (BlendingHigh)	The function block is started immediately after the last instruction motion is terminated. The axis does stop between motions but passes through the end position of the first motion at the higher velocity of the two motion instructions.

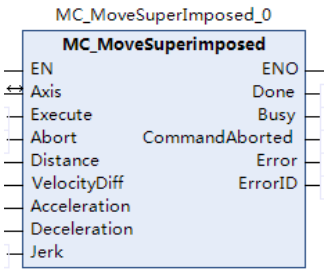
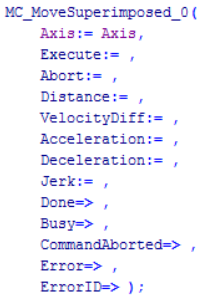
◆ Start of other instructions during the execution of this instruction

When starting motion instructions for multiple instances by using this instruction, users can choose the aborting, buffered, or blending mode.

MC_MoveSuperImposed

This function block commands a controlled motion of a specified velocity and position in addition to the existing velocity and position. It has no impact on the original instruction execution time model.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ MoveSuperImposed	Superimposed relative motion instruction		


```
MC_MoveSuperimposed_0(
    Axis:= Axis,
    Execute:= ,
    Abort:= ,
    Distance:= ,
    VelocityDiff:= ,
    Acceleration:= ,
    Deceleration:= ,
    Jerk:= ,
    Done=> ,
    Busy=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorID=> );
```

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
Abort	Abort condition	BOOL	TRUE, FALSE	FALSE	Abort the ongoing motion and reset all outputs
Distance	Axis reached position	LREAL	Value Range	0	This data is the superimposed position data.
VelocityDiff	Superimposed velocity	LREAL	Value Range	0	Superimposed velocity for axis running
Acceleration	Acceleration	LREAL	Value Range	0	Acceleration rate for velocity increase
Deceleration	Deceleration	LREAL	Value Range	0	Deceleration rate for velocity decrease

Jerk	Jerk	LREAL	Value Range	0	Slope change of the curve acceleration/ deceleration

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction execution completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of axis instruction is completed
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ ERROR	0	Output an error code when an error occurs

3) Function Description

◆ This function block superimposes the position (Distance) and velocity (VelocityDiff) on other instructions.

◆ In motion mode, MC_MoveSuperImposed can be superimposed on any other instruction.

◆ MC_MoveSuperImposed can be aborted by another MC_MoveSuperImposed.

◆ In the Standstill status, the MC_MoveSuperimposed function block acts like MC_MoveRelative.

◆ The function block starts at the rising edge of Execute.

◆ This function block is an instruction for absolute axis positioning. The position data is the absolute position of the axis.

◆ The function block starts at the rising edge of Execute.

4) Precautions

If an instance of MC_MoveSuperImposed is active and another instance of the MC_MoveSuperImposed type is called, the second instance reports an error. If an instance of MC_MoveSuperImposed is active and is started again at a new rising edge of Execute (possibly with a different input), the active superimposed motion will be aborted and replaced by a new superimposed motion, while the original motion control function block remains active.

The Abort pin functions to abort the superimposed motion. Triggering Abort clears the superimposed motion that has been executed. If the superimposed position is large, it will cause a sudden change in the desired position, which leads to a servo error. Therefore, exercise with caution.

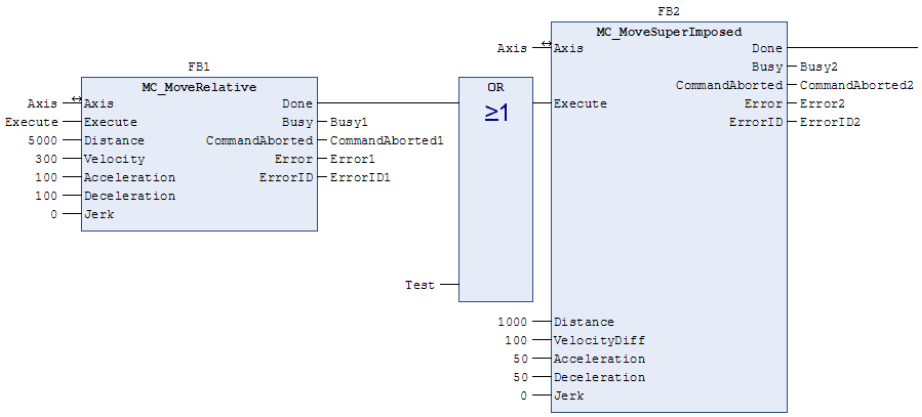
5) Timing Diagram

Execute of the function block must have a rising edge condition.

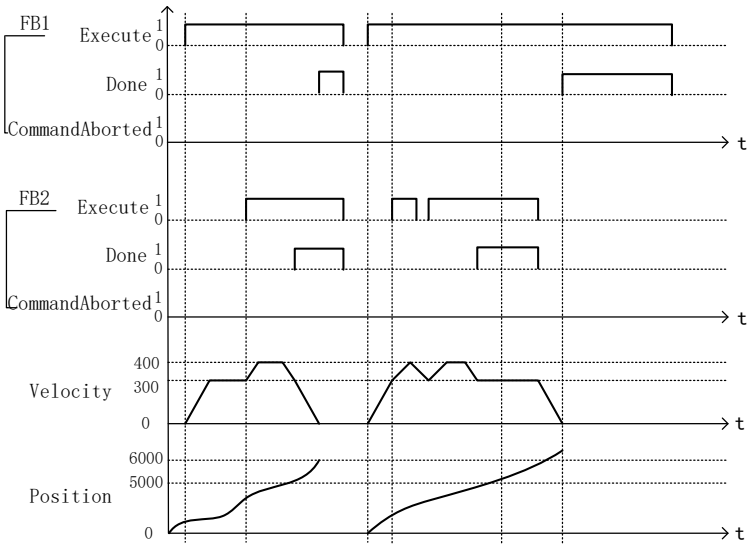
Done of the function block indicates that the execution of the instruction is completed.

Busy of the function block indicates that the execution of the instruction is in progress.

◆ Example



◆ Timing operation description:



MC_MoveVelocity

This function block can achieve axis velocity control in the drive CSV mode and CSP mode. After the axis is enabled, the running velocity can be set through the input pin Velocity, and this instruction will run when triggered by the rising edge.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_MoveVelocity	Velocity control instruction	MC_MoveVelocity_0 MC_MoveVelocity EN, Axis, Execute, Velocity, Acceleration, Deceleration, Jerk, Direction, BufferMode ENO, InVelocity, Busy, Active, CommandAborted, Error, ErrorID	MC_MoveVelocity_0(Axis:= Axis, Execute:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Direction:= , BufferMode:= , InVelocity=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
Velocity	Velocity reference	LREAL	Value Range	0	This data is the velocity value for this instruction.
Acceleration	Acceleration	LREAL	Value Range	0	Acceleration rate for velocity increase
Deceleration	Deceleration	LREAL	Value Range	0	Deceleration rate for velocity decrease
Jerk	Jerk	LREAL	Value Range	0	Slope change of the curve acceleration/deceleration
Direction	Running direction	MC_Direction	Positive, negative, current	current	Instruction operation in running direction
BufferMode	Buffer Mode	MC_BUFFER_MODE	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	0	Specify the action to be taken when multiple instances initiate a motion instruction. 0: Aborting 1: Buffered 2: Blend at the low velocity 3: Blend at the previous velocity 4: Blend at the next velocity 5: Blend at the high velocity

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
InVelocity	Flag of reaching the set velocity	BOOL	TRUE, FALSE	FALSE	Set to true when the set velocity is reached
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Active	Control	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the function block has control on the axis
CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

◆ This function block performs analog velocity control based on position control.

◆ The velocity control action starts at the rising edge of Execute.

◆ The motion direction is specified by Direction. When Direction is set to "Positive", it moves in positive direction. When Direction is set to "Negative", it moves in negative direction. When Direction is set to "Current", the motion differs depending on whether the axis is stopped. When the axis is stopped, the axis moves in the direction of the last motion. When the power is turned on or restarted, the axis moves in positive direction. When this instruction is started during the process of axis motion by activating the motion instruction in multiple instances, the axis moves in the direction of the current motion.

4) Precautions

When Direction is set to "MC_Direction", the axis moves in the direction of the previous instruction. Therefore, depending on the combination of instructions, the direction of the instruction may be

different from that of the input of the previous motion instruction.

When the input variable Velocity is 0, the default initial value will be assigned if Acceleration, Deceleration or Jerk is 0. For details, see "6.3.3 Defaults of Motion Control Function Blocks" on page 238.

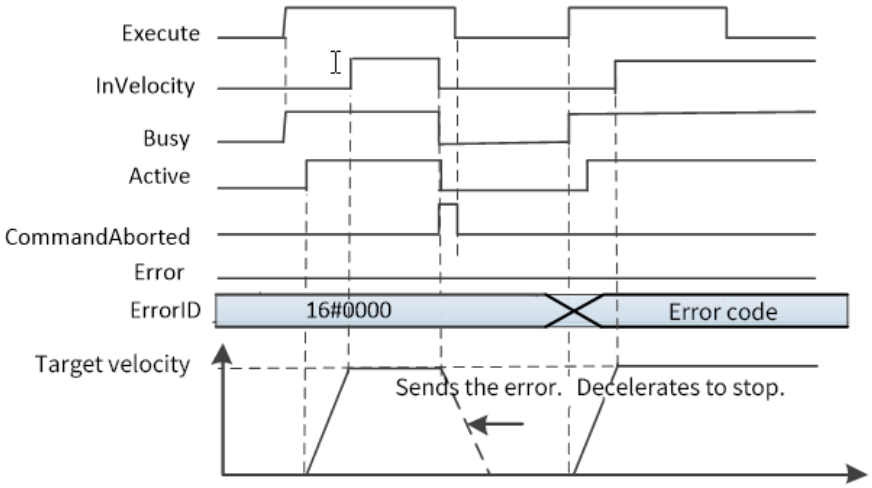
5) Timing Diagram

The value of Busy changes to TRUE when Execute is started. The value of Active changes to TRUE in the next period.

InVelocity changes to TRUE when Velocity is reached.

When this instruction is aborted by another instruction, the value of CommandAborted changes to TRUE and those of Busy, Active, and InVelocity change to FALSE.

◆ Example



◆ Start of this instruction during the execution of other instructions

When this instruction is started for the currently executing instruction, it will be switched or cached to this instruction.

The action when multiple instances of this instruction are started is determined by BufferMode.

Buffer Mode	Description
Aborting	Immediately aborts the currently executing instruction and switches to this instruction. If the direction of axis motion is reversed due to instruction switching, reverse running is performed after the velocity is decelerated to zero.
Buffered	The function block is started immediately after the last instruction motion is terminated. No blending is performed here. When the end conditions (such as Done, InVelocity, InEndVelocity, InGear, InSync, EndOfProfile) are reached, the new motion starts at the velocity of the previous motion. If the previous motion was MC_MoveAbsolute or MC_MoveRelative, the new motion will start in static state.
Blending	Starts at the velocity (relay velocity) at which the currently executing instruction reaches the target position, and continuously makes the cached instruction take motions. Change the motion of the currently executing instruction, ensuring that the target position is reached at the relay velocity. There are four ways to specify the relay velocity:

	Blend at the low velocity (BlendingLow)	The function block is started immediately after the last instruction motion is terminated. The axis does not stop between motions but passes through the end position of the first motion at the lower velocity of the two motion instructions.
	Blend at the previous velocity (BlendingPrevious)	The function block is started immediately after the last instruction motion is terminated. The axis does not stop between motions but passes through the end position of the first motion at the velocity of the first motion instruction.
	Blend at the next velocity (BlendingNext)	The function block is started immediately after the last instruction motion is terminated. The axis does not stop between motions but passes through the end position of the first motion at the velocity of the second motion instructions.
	Blend at the high velocity (BlendingHigh)	The function block is started immediately after the last instruction motion is terminated. The axis does stop between motions but passes through the end position of the first motion at the higher velocity of the two motion instructions.

◆ Start of other instructions during the execution of this instruction

Only when Aborting or Buffered is selected for BufferMode of other instructions, the MC instruction can be started by using multiple instances of other instructions during the execution of this instruction.

When Buffered is selected, if the output variable InVelocity of this instruction changes to TRUE, the motion of starting the instruction for multiple instances is executed.

MC_MoveFeed

This function block executes the specified distance positioning from the position where the external device triggers the interrupt input. The interrupt feed can be used for absolute positioning, relative positioning and velocity control.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_MoveFeed	Interrupt positioning instruction	MC_MoveFeed_0 MC_MoveFeed EN, Axis, TriggerInput, Execute, WindowOnly, FirstPosition, LastPosition, ReferenceType, Position, Velocity, Acceleration, Deceleration, Jerk, Direction, MoveMode, FeedDistance, FeedVelocity, BufferMode, ErrorDetect ENO, Done, InFeed, Busy, Active, CommandAborted, Error, ErrorID	MC_MoveFeed_0(Axis:= Axis, TriggerInput:= Trigger, Execute:= , WindowOnly:= , FirstPosition:= , LastPosition:= , ReferenceType:= , Position:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Direction:= , MoveMode:= , FeedDistance:= , FeedVelocity:= , BufferMode:= , ErrorDetect:= , Done=> , InFeed=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF	-	-	Reference to the axis, that is, an instance of AXIS_REF
TriggerInput	Trigger signal	TRIGGER_REF	-	-	Associated attributes such as trigger signal or trigger attribute

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Start	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
WindowOnly	Window valid	BOOL	TRUE, FALSE	FALSE	Enable or disable the window
FirstPosition	Start position	LREAL	Value Range	0	Specify the latch enable position [u]
LastPosition	End position	LREAL	Value Range	0	Specify the latch disable position [u]
ReferenceType	Position type	MC_ REFERENCE_ TYPE	1: FeedBack	1	By default, the latch position is the actual position.
Position	Target position	LREAL	Positive number, negative number, 0	0	When MoveMode is [0: Absolute value positioning], specify the target position in absolute coordinates. When MoveMode is [1: Relative value positioning], specify the movement distance. When MoveMode is [2: Velocity control], it is not required to specify the position. Unit: [u]
Velocity	Target velocity	LREAL	Positive number	0	Specify the target velocity [u/s]
Acceleration	Acceleration	LREAL	Positive number	0	Specify the acceleration rate [u/s^2]
Deceleration	Deceleration	LREAL	Positive number	0	Specify the deceleration rate [u/s^2]
Jerk	Jerk	LREAL	Positive number	0	Specify the jerk [$u/^3$]
Direction	Direction	MC_ DIRECTION	-1: Negative 0: Shortest 1: Positive 2: Current 3: Fastest	Positive	Select the direction
MoveMode	Motion mode	MC_MOVE_ MODE	0: Absolute 1: Relative 2: Velocity	0	Select the motion mode
FeedDistance	Feed distance	LREAL	Positive number, negative number, 0	0	Movement distance after the input interrupt feed Specify a positive value to feed in the same direction as axis movement before the interrupt input, and a negative value to feed in opposite direction [u].
FeedVelocity	Feedrate	LREAL	Positive number	0	Movement velocity after the input interrupt feed [u/s]
BufferMode	Buffer mode	MC_BUFFER_ MODE	0: Aborting 1: Buffered	0	Specify the action to be taken when there are multiple motion instructions
ErrorDecect	Error detection	BOOL	TRUE, FALSE	FALSE	Specify whether to detect an error when there is no interrupt source input. TRUE: The Error signal is set to TRUE if no interrupt signal is detected after the position specified by Position is reached. FALSE: The Done signal is set if no interrupt signal is detected after the position specified by Position is reached.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is completed
InFeed	Latch input	BOOL	TRUE, FALSE	FALSE	Set to TRUE in standard transfer after latch input is received
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Active	Control	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the function block has control on the axis
CommanAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	DWORD	-	0	Output an error code when an error occurs

◆ TRIGGER_REF Description

Structure	Element	Data Type	Default	Description
TRIGGER_REF	iTriggerNumber	INT	0	Set to TRUE when the instruction is completed
	bFastLaching	BOOL	TRUE	-
	bInput	BOOL	FALSE	-

[Note]: The specification of the servo touch probe must be distinguished from that of the instruction touch probe. Take IS620N as an example.

As the external DI trigger signals, DI8 with function 38 and DI9 with function 39 must be respectively used for touch probe 1 and touch probe 2. The following part takes DI8 as an example to describe how to perform the setting.

[Requirement]: Touch probe 1 rising edge, continuous latching

4) Set the DI8 function: set 0x2003-11 to 38.

5) Set the DI8 logic in 0x2003-12.

DI8 Logic 2003-12h Setpoint	Description
0: Active low	The drive forcibly changes it to falling edge active.
1: Active high	The drive forcibly changes it to rising edge-triggered.
2: Rising edge-triggered	Rising edge-triggered
3: Falling edge-triggered	Falling edge-triggered
4: Edge change-triggered	Rising/Falling edge-triggered

Set 0x2003-12 to 1 or 2 in this example.

For details, see the IS620N Series Servo Design and Maintenance User Guide.

3) Function Description

◆ ● Movement is performed at the rising edge of Execute according to MoveMode (move by absolute value, move by relative value, or velocity control).

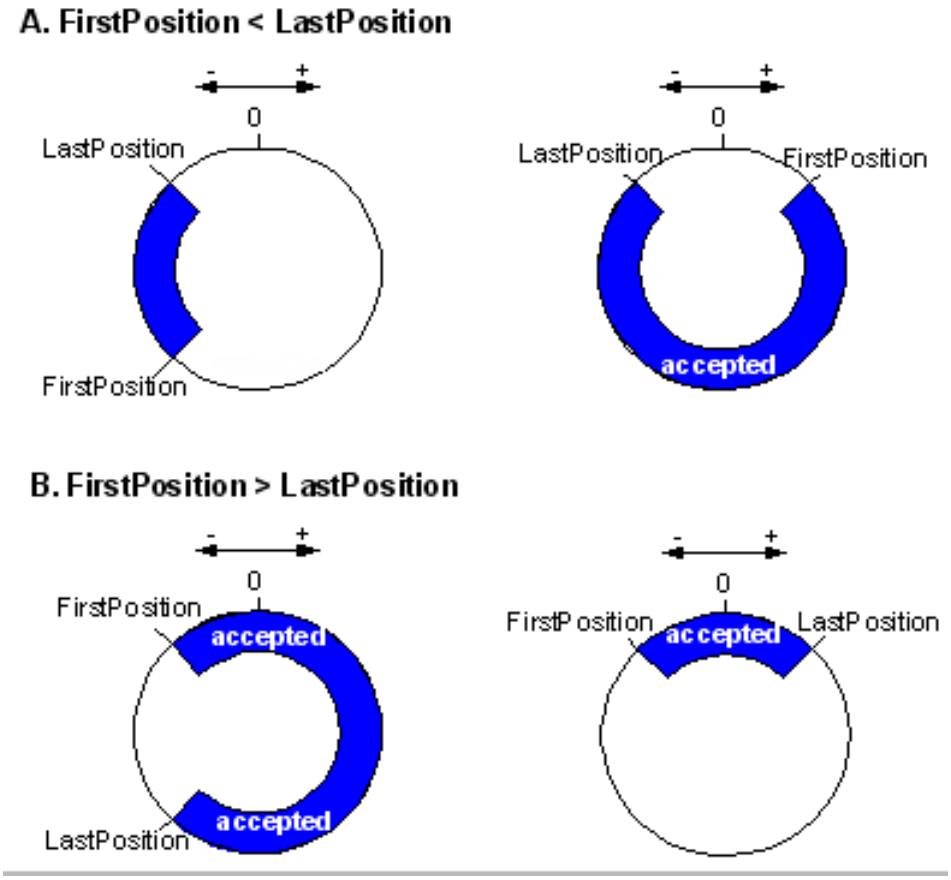
◆ ● When move by absolute value is selected, set the target position through Position. When move by relative value is selected, set the target distance through Position.

◆ ● Regardless of the movement mode, the movement is performed at Velocity (target velocity).

◆ ● During movement, relative positioning is performed at the rising edge of external input (interrupt input). The movement distance is specified by FeedDistance, starting from the feedback position at FeedVelocity.

- ◆ ● Interrupt standard transfer is performed by using the instruction of move by absolute value or relative value. The motion will be stopped at the target position if an interrupt signal is not input before the target position is reached.
- ◆ ● When an interrupt mask is used, set WindowOnly (window valid) to TRUE and specify FirstPosition (start position) and LastPosition (end position). Interrupt standard positioning is performed by feeding back the initial interrupt signal that occurs from FirstPosition to LastPosition.
- ◆ ● A brief description of the three motion modes is given. The MoveFeed parameter is used to filter the first segment of motion. There are three motion modes: absolute positioning, relative positioning, and target velocity motion. Absolute positioning and relative positioning can be done by lowering the Done setting of the instruction without triggering the instruction. In velocity mode, movement will be performed at the target velocity.
- ◆ ● The values of the motion parameters of the instruction, that is, the shared acceleration rate, deceleration rate, and jerk, must be clearly described. These parameters are shared by the first segment of motion and the feed motion, and the target velocity cannot be 0.
- ◆ ● When the window function WindowsOnly of the probe is not set, and FirstPosition and LastPosition are set arbitrarily, the instruction is not affected. Touch probe triggering will not be restricted by position. Triggering the touch probe anywhere will enable the instruction to enter the feed motion. When WindowOnly is set, the instruction will determine the value of FirstPosition and LastPosition. In linear mode, FirstPosition should be less than or equal to LastPosition. The final judgment of the touch probe position is FirstPosition ⩽ Touch probe position ⩽ LastPosition. If FirstPosition > LastPosition, the instruction is processed in the same way as an error reported for an abnormal parameter. In rotary axis mode, if FirstPosition ⩽ LastPosition, the judgment position of the window is the clockwise interval from FirstPosition to LastPosition of the same period (including LastPosition and FirstPosition). If FirstPosition > LastPosition, the judgment position of the window is the clockwise interval from FirstPosition to LastPosition of the same period (excluding LastPosition and FirstPosition). In particular,

  when LastPosition and FirstPosition exceed the position of one rotation period, an error is reported.

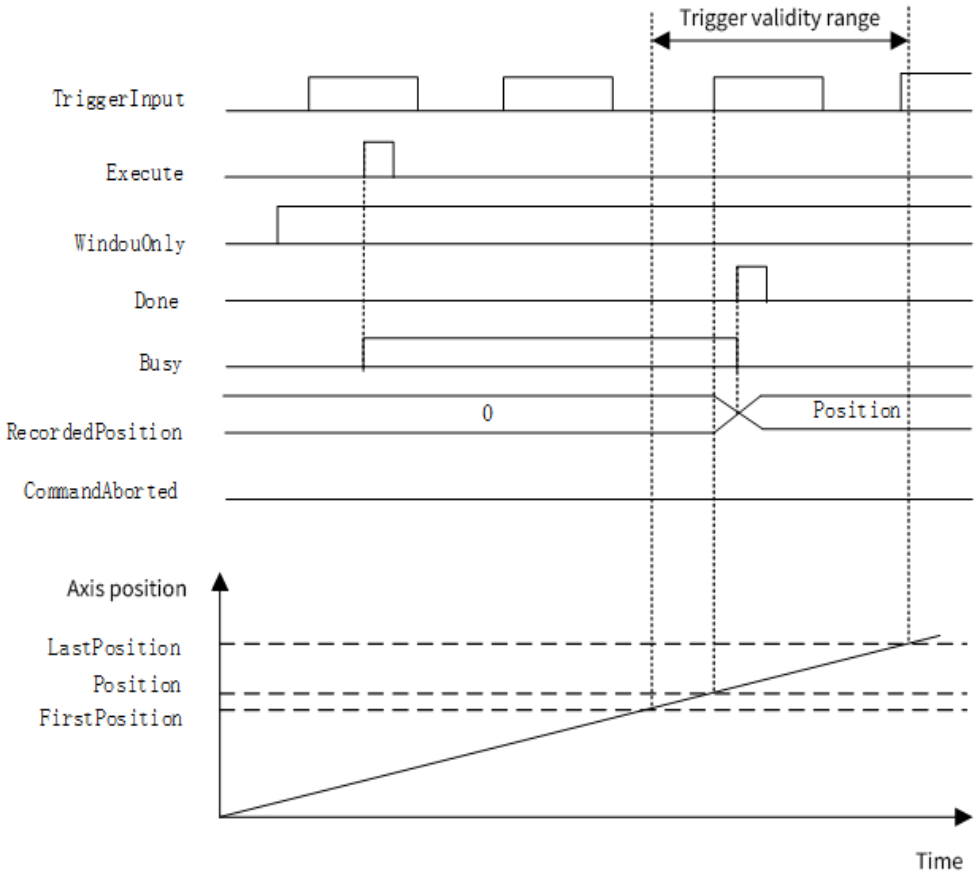


- ● For absolute positioning in rotation mode, check whether the direction belongs to 5 directions that are set. For the velocity mode, check whether 3 directions are set: Positive, Negative, and Current. If not, the Direction parameter reports an error.
- ● The error detection function determines whether to report an error if the touch probe interrupt has not been triggered after the movement has reached the target position. If not triggered, the FB reports an error without affecting the execution of subsequent buf instructions.
- ● This instruction cannot be triggered repeatedly. Otherwise, the instruction reports an error, and the error of occupied touch probe caused by repeated triggering can only be canceled by the MC_ AbortTrigger instruction.
- ● The channel of the current instruction is occupied by TouchProbe, which triggers the instruction. The MoveFeed instruction does not occupy the probe channel when it is in the buffer.
- ● If it is detected during MoveFeed execution that the touch probe is occupied, the error of touch probe occupied will be reported.
- ● For the ECAT axis, if 60b8/60b9/60ba/60bc PDO is not configured, then an error is reported.
- ● For drive mode, operation is not allowed in virtual axis mode.
- ● The axis cannot be executed in error status.
- ● Simultaneous triggering of different interrupt positioning instances in the same touch probe channel will invalidate the touch probe (including different triggering schemes for the same touch probe channel).
- ● Note: If the velocity at the moment of triggering the feed motion is large and the feed distance is small, the desired feed distance may be smaller than the current set position, that is, reverse running will occur when the desired feed position is reached.

4) Timing Diagram

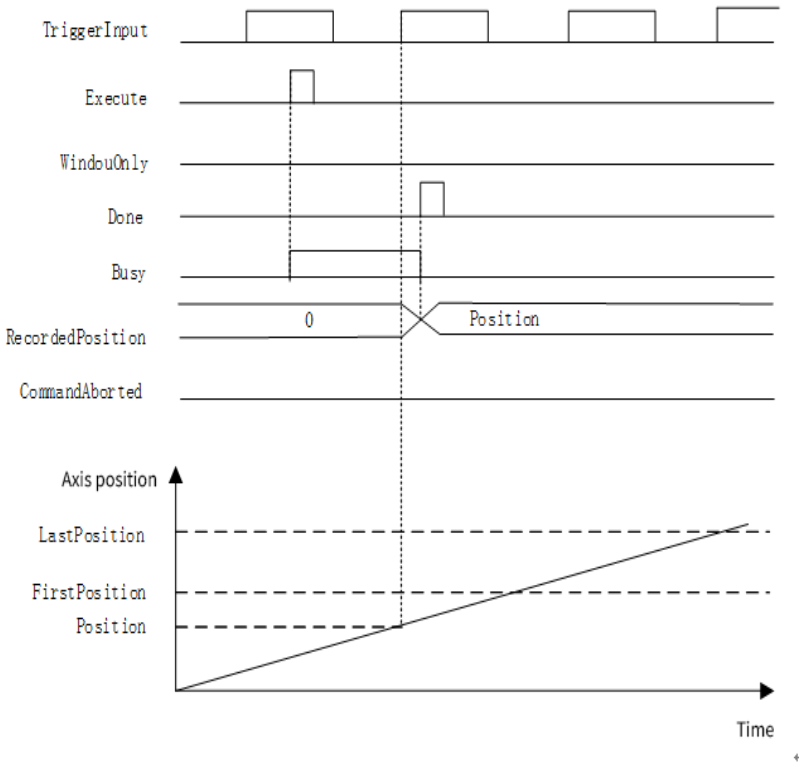
◆ When WindowOnly is set to Enable

The trigger input is detected only within the window to obtain the axis position.

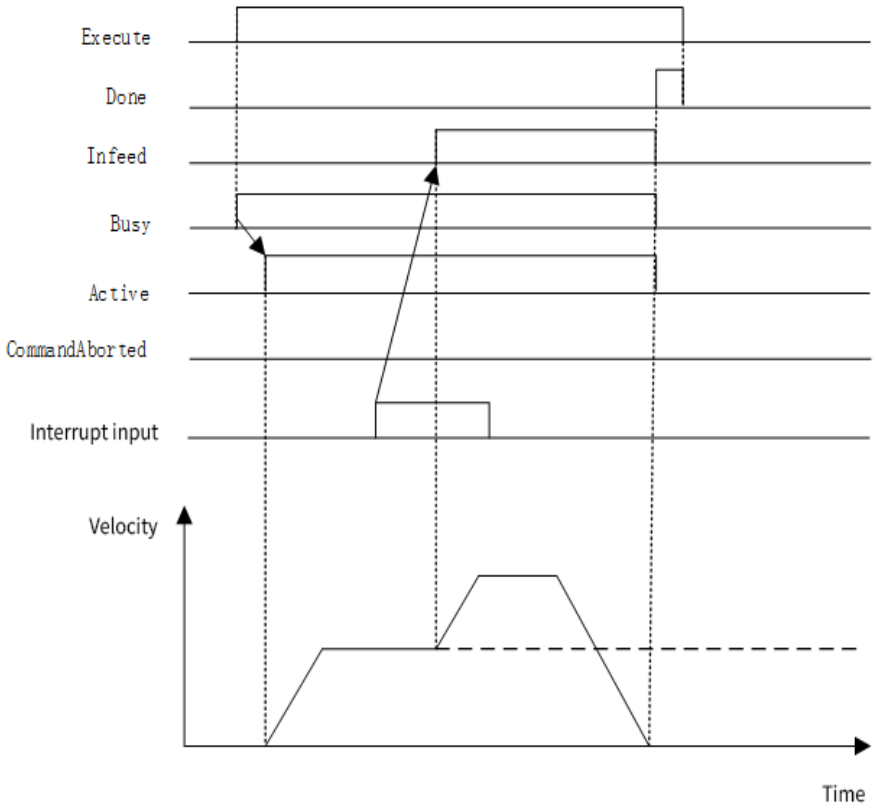


◆ When WindowOnly is set to Disable

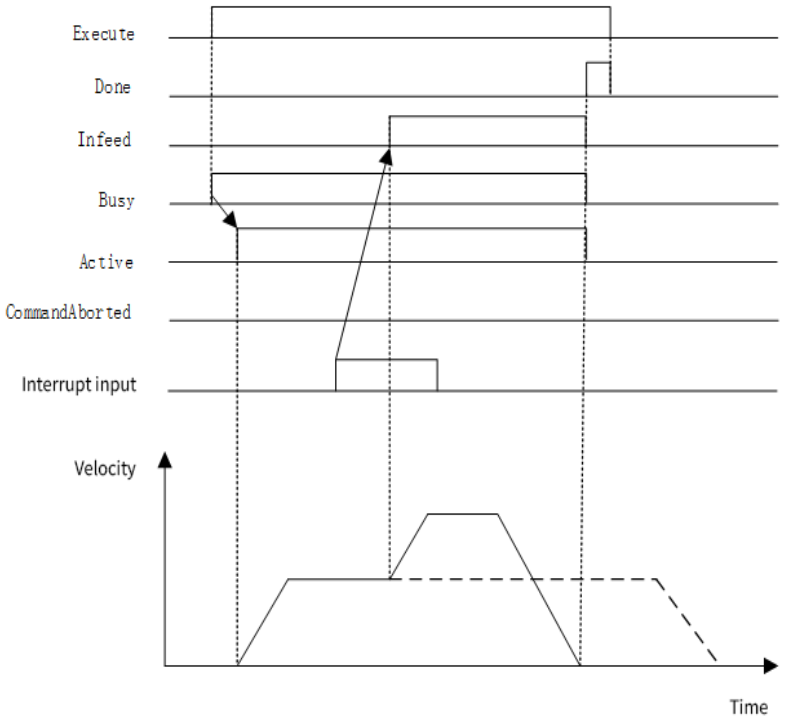
The axis position at the time of the initial trigger after Execute changes to TRUE is used as the reference position for the standard distance.



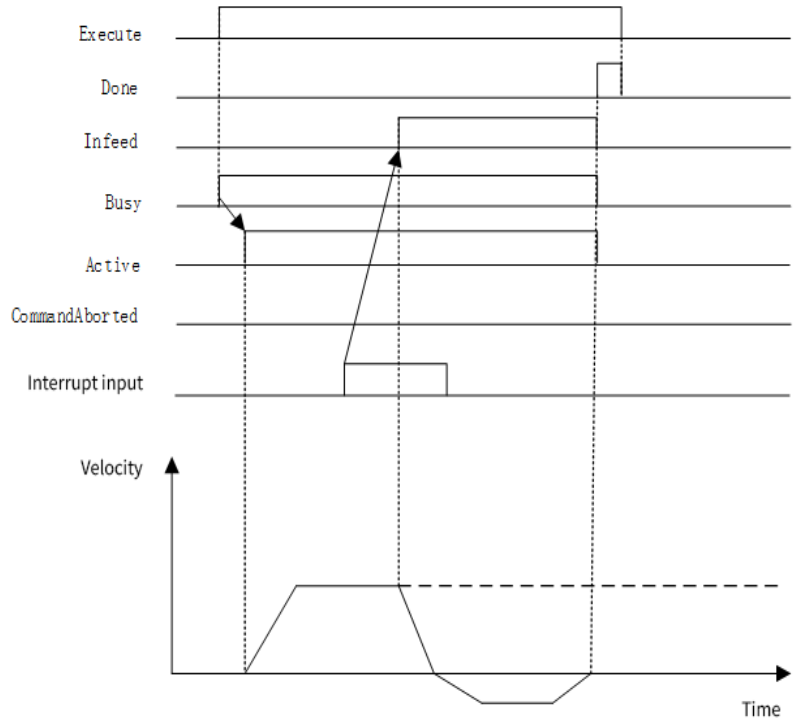
◆ When MoveMode is set to Absolute or Relative



◆ When MoveMode is set to Velocity

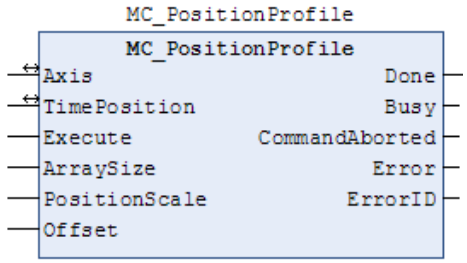


◆ When the standard position is reversed after an interruption



MC_PositionProfile

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ PositionProfile	Position profile instruction		MC_PositionProfile(Axis:= , TimePosition:= , Execute:= , ArraySize:= , PositionScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
TimePosition	Axis position operation time and position description	MC_TP_REF	-	-	Axis position operation time and position data description; data consists of multiple sets of data

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
ArraySize	Dynamic array	INT	Value Range	0	The number of arrays used in the operation profile
PositionScale	Integration factor	LREAL	"Positive" + "0"	1	Position scale factor in MC_TP_REF
Offset	Offset	LREAL	-	0	Overall offset of the position

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction execution completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of axis instruction is completed
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ ERROR	0	Output an error code when an error occurs

3) Function Description

◆ This function block is a profile motion model of time period and position. The operation mode is

Discrete Motion. It runs based on the data set by the user for the TimePosition variable.

- The operation status of this function block is Standstill, and the status during instruction running is Discrete Motion. It cannot run in other statuses.
- The function block is started at the rising edge of Execute. This instruction is repeated in Discrete Motion.
- TimePosition is of the MC_TP_REF data type.

MC_TP_REF description:

Member	Type	Initial Value	Description
Number_of_pairs	INT	0	Number of profile path segments
IsAbsolute	BOOL	TRUE	Absolute motion (TRUE) and relative motion option
MC_TP_Array	ARRAY[1..N] OF SMC_TP	-	Array of time and position

SMC_TP description:

Member	Type	Initial Value	Description
delta_time	TIME	TIME#0ms	Time of position segment
position	LREAL	0	Current position value

Note: When there is a change in the velocity, the corresponding adjustment is made based on the set position data in an S-curve.

4) Timing Diagram

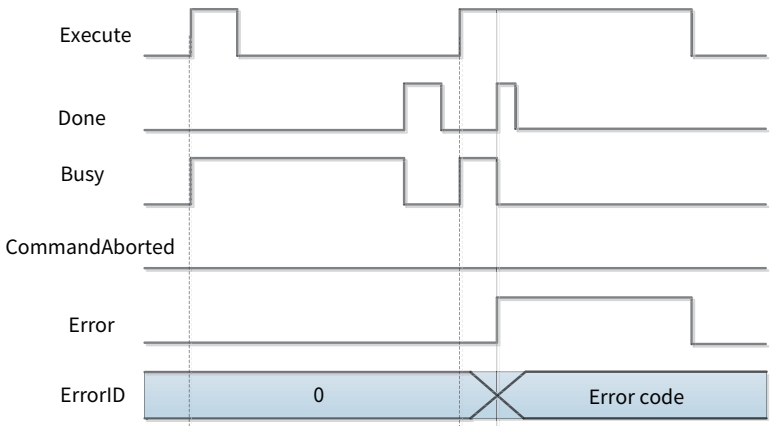
The position profile instruction can run only when the condition MC_TP_Array has been set by other means.

The instruction can run only when the axis is in Standstill status.

Execute of the function block must have a rising edge condition.

Done of the function block indicates that the execution of the instruction is completed.

Busy of the function block indicates that the execution of the instruction is in progress.



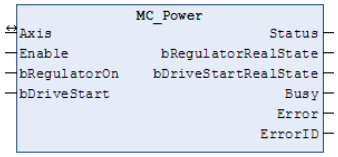
5) Error Description

The error occurs as the instruction is not started in the axis status of Standstill or there is a parameter error in the instruction system. An axis error must be cleared before the start of the operation.

Note: For details on the error code, see “ “Appendix C Error Codes” .

MC_Power

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_Power	Axis enable instruction		MC_Power(Axis:= , Enable:= , bRegulatorOn:= , bDriveStart:= , Status=> , bRegulatorRealState=> , bDriveStartRealState=> , Busy=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	The function block starts processing when set to TRUE
bRegulatorOn	Enable state	BOOL	TRUE, FALSE	FALSE	The axis is enabled when set to TRUE
bDriveStart	Enable the drive	BOOL	TRUE, FALSE	FALSE	Set to TRUE to disable emergency stop of the function block

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Status	Ready for motion	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the axis is ready for motion
bRegulatorRealState	Axis enable signal state	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis enable is active
bDriveStartRealState	Drive enabled	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the axis is not interrupted by the quick stop mechanism
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the processing of the function block is not completed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ ERROR	0	Output an error code when an error occurs

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

3) Function Description

◆ Other inputs will be processed by the function block only when the input Enable is set to TRUE.

◆ If the MC_Power function block has been called and bRegulatorOn is set to FALSE, the function block will set the relevant axis state (nAxisState) to the power_off, indicating that the drive is not ready for motion.

◆ If the MC_Power function block has been called and bRegulatorOn is set to TRUE, the function block will set the relevant axis state (nAxisState) to Standstill if no error has occurred in the axis. If an error has

occurred, the corresponding error state will be output.

- If Enable, bRegulatorOn and bDriveStart are set to TRUE but the output Status remains FALSE after a certain period of time, then the output Error will be set. This may happen if a hardware issue arises when the axis is enabled.

If the enable signal is lost (usually in operating mode), nAxisState of the relevant axis will be set to ErrorStop.

4) Timing Diagram

Setting Enable to TRUE, bRegulatorOn to TRUE and bDriveStart to TRUE makes Busy become TRUE, the axis enters the ON state and Status becomes TRUE, respectively.



5) Error Description

Do not write a program to start other instances of MC_Power in the axis that is executing MC_Power. In principle, only one MC_Power instruction can be set for each axis.

If MC_Power of another instance is started in the axis where MC_Power is being executed, MC_Power that is executed later will be executed preferentially.

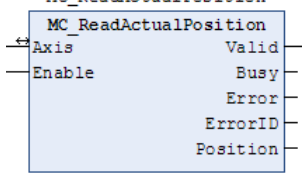
Note: For details on the error code, see “ “Appendix C Error Codes” .

MC_ReadActualPosition

This instruction reads the actual position at which the drive is running and saves it in a variable unit defined by itself.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression

MC_ ReadActualPosition	Instruction for reading actual position	MC_ReadActualPosition  	MC_ReadActualPosition(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Position=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Execution condition	BOOL	TRUE, FALSE	FALSE	Read the current position of the servo if set to TRUE

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Position data obtainable	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the drive position can be obtained correctly
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs
Position	Axis position obtained	LREAL	Axis position	0	Axis position data obtained by instruction

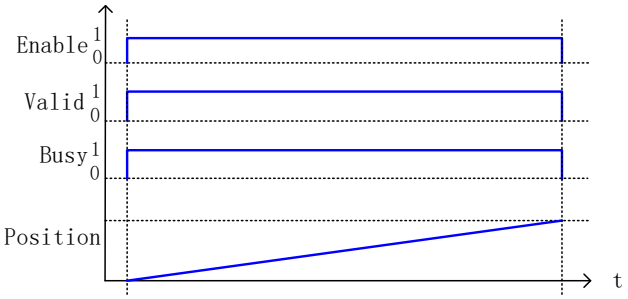
3) Function Description

This instruction reads the actual position of the drive. It is active at high level of Enable and can be executed many times without affecting each other.

4) Timing Diagram

Enable of the function block must be set to TRUE.
Valid of the function block indicates that the value of Position obtained is valid.
Busy of the function block indicates that the execution of the instruction is in progress.

Timing operation description:



MC_ReadAxisError

This instruction reads the axis error and saves it in a variable unit defined by itself.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ReadAxisError	Instruction for reading axis error	MC_ReadAxisError Axis Enable Valid Busy Error ErrorID AxisError AxisErrorID SWEndSwitchActive	MC_ReadAxisError(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , AxisError=> , AxisErrorID=> , SWEndSwitchActive=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Execution condition	BOOL	TRUE, FALSE	FALSE	Read the current position of the servo if set to TRUE

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Flag of error data obtainable	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the error data of the axis can be obtained
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ ERROR	0	Output an error code when an error occurs
AxisError	Axis error flag	BOOL	TRUE, FALSE	FALSE	Set the corresponding flag when an axis error is read
AxisErrorID	Axis error code	DWORD	-	0	An error code is obtained
SWEndSwitchActive	Soft limit switch active	BOOL	TRUE, FALSE	FALSE	Check the status of the soft limit switch during reading

3) Function Description

This function block reads the error code of the drive. It is active at high level of Enable and can be executed many times without affecting each other.

4) Timing Diagram

Enable of the function block must be set to TRUE.

Valid of the function block indicates that the values of AxisError and AxisErrorID obtained are valid.

Busy of the function block indicates that the execution of the instruction is in progress.

MC_ReadBoolParameter

This instruction reads the bit parameter of the drive axis and saves it in a variable unit defined by itself.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ ReadBoolParameter	Instruction for reading axis bit parameters	MC_ReadBoolParameter Axis, Enable, ParameterNumber Valid, Busy, Error, ErrorID, Value	MC_ReadBoolParameter(Axis:= , Enable:= , ParameterNumber:= , Valid=> , Busy=> , Error=> , ErrorID=> , Value=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Execution condition	BOOL	TRUE, FALSE	FALSE	Read the current position of the servo if set to TRUE
ParameterNumber	Serial number of axis parameter	DINT	-	0	Obtain the index, sub-index and serial number of axis parameters

Note: ParameterNumber (DINT) = -DWORD_TO_DINT(SHL(USINT_TO_DOWRD(usiDataLength), 24) (length of data in object dictionary)

+ SHL(UINT_TO_DWORD(uiIndex), 8) (index in object dictionary -16 bits)

+ usisubIndex(sub-index in object dictionary - 8 bits))

usiDataLength: Fill in bytes; 16#01 for 1 byte, 16#02 for 2 bytes, 16#04 for 4 bytes, and so on

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Position data obtainable	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the drive position can be obtained correctly
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed

Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs
Value	Axis position status obtained	BOOL	TRUE, FALSE	FALSE	Axis position status obtained by instruction

3) Function Description

This instruction reads the bit data status of the drive. It is active at high level of Enable and can be executed many times without affecting each other.

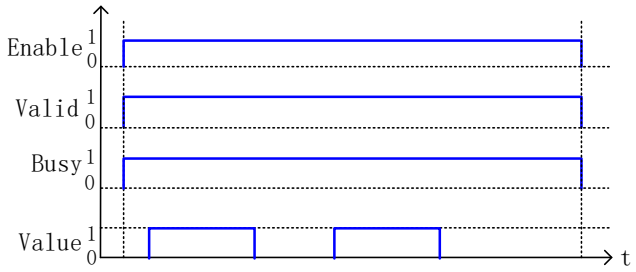
4) Timing Diagram

Enable of the function block must be set to TRUE.

Valid of the function block indicates that the bit status data obtained is valid.

Busy of the function block indicates that the execution of the instruction is in progress.

◆ Timing operation description:



MC_ReadStatus

This instruction reads the status data of the axis and saves it in a variable unit defined by itself.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ReadStatus	Instruction for reading axis status	MC_ReadStatus MC_ReadStatus Axis Enable Valid Busy Error ErrorID Disabled Errorstop Stopping StandStill DiscreteMotion ContinuousMotion SynchronizedMotion Homing ConstantVelocity Accelerating Decelerating FBErrorOccured	MC_ReadStatus(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Disabled=> , Errorstop=> , Stopping=> , StandStill=> , DiscreteMotion=> , ContinuousMotion=> , SynchronizedMotion=> , Homing=> , ConstantVelocity=> , Accelerating=> , Decelerating=> , FBErrorOccured=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Execution condition	BOOL	TRUE, FALSE	FALSE	Read the current position of the servo if set to TRUE

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Flag of error data obtainable	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the error data of the axis can be obtained
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ ERROR	0	Output an error code when an error occurs
Disabled	Axis disabled	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis is disabled
Errorstop	Axis error	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the axis is in error state
Stopping	Axis in stopping process	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the axis is in stopping process
StandStill	Axis in standard state	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis is in the standard (operational) state
DiscreteMotion	Axis in discrete motion	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis is in discrete motion
ContinuousMotion	Axis in continuous motion	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the axis is in continuous motion
SynchronizedMotion	Axis in synchronous motion	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the axis is in synchronous motion
Homing	Axis in homing state	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the axis is in homing state
ConstantVelocity	Axis running velocity reached	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis reaches the running velocity
Accelerating	Axis acceleration	BOOL	TRUE, FALSE	FALSE	Set to TRUE during axis acceleration
Decelerating	Axis deceleration	BOOL	TRUE, FALSE	FALSE	Set to TRUE during axis deceleration
FBErrorOccured	Axis FB error occurrence	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an axis FB error occurs

3) Function Description

◆ This instruction reads the axis status. It is active at high level of Enable and can be executed many times without affecting each other.

◆ Enable of the function block must be set to TRUE.

◆ Valid of the function block indicates the data of the status flags can be read.

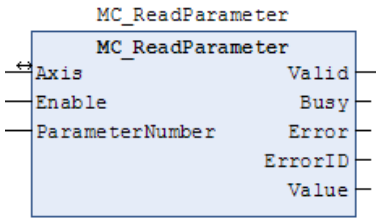
◆ Busy of the function block indicates that the execution of the instruction is in progress.

MC_ReadParameter

This instruction reads parameters of the drive axis and saves it in the variable unit defined by itself.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression

MC_ ReadParameter	Instruction for reading axis parameters		MC_ReadParameter(Axis:= , Enable:= , ParameterNumber:= , Valid=> , Busy=> , Error=> , ErrorID=> , Value=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Execution condition	BOOL	TRUE, FALSE	FALSE	Read the current position of the servo if set to TRUE
ParameterNumber	Serial number of axis parameter	DINT	-	0	Obtain the index, sub-index and serial number of axis parameters

Note: ParameterNumber (DINT) = -DWORD_TO_DINT(SHL(USINT_TO_DOWRD(usiDataLength), 24) (length of data in object dictionary)

+ SHL(UINT_TO_DWORD(uiIndex), 8) (index in object dictionary -16 bits)

+ usisubIndex(sub-index in object dictionary - 8 bits)

usiDataLength: Fill in bytes; 16#01 for 1 byte, 16#02 for 2 bytes, 16#04 for 4 bytes, and so on

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Position data obtainable	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the drive position can be obtained correctly
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs
Value	Axis parameters obtained	LREAL	-	0	Axis parameter obtained by instruction

3) Function Description

This instruction reads the bit data status of the drive. It is active at high level of Enable and can be executed many times without affecting each other.

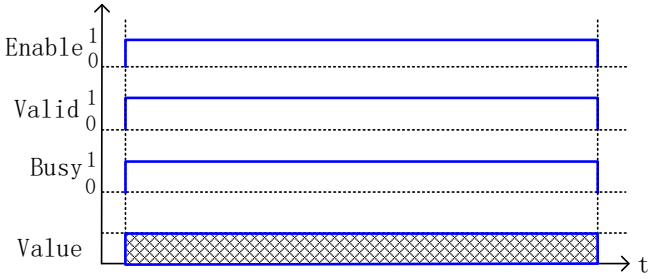
4) Timing Diagram

Enable of the function block must be set to TRUE.

Valid of the function block indicates that the bit status data obtained is valid.

Busy of the function block indicates that the execution of the instruction is in progress.

Timing operation description:



MC_Reset

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_Reset	Instruction for resetting axis error state	MC_Reset MC_Reset Axis Execute Done Busy Error ErrorID	MC_Reset(Axis:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction execution completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of axis instruction is completed
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ERROR	0	Output an error code when an error occurs

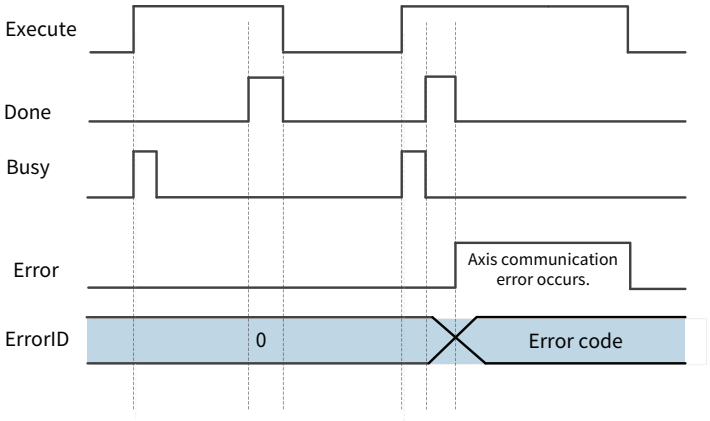
3) Function Description

◆ This function block changes the axis status from Errorstop to Standstill when the axis is in normal communication, that is, changes the abnormal status to the normal status.

◆ When the axis Errorstop cannot be reset and Axis.bCommunication is FLASE, the communication between master and slave must be re-established.

Note that the Busy flag bit in the instruction is connected for a very short period of time.

4) **Timing Diagram**



MC_Stop

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_Stop	Instruction for stopping an axis	MC_Stop MC_Stop Axis Execute Deceleration Jerk Done Busy Error ErrorID	MC_Stop(Axis:= , Execute:= , Deceleration:= , Jerk:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
Deceleration	Deceleration	LREAL	"Positive" + "0"	0	Deceleration of the function block (u/s^2)
Jerk	Jerk	LREAL	"Positive" + "0"	0	Specify the jerk [reference unit/s^3]

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction execution completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of axis instruction is completed
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

- This function block stops the motion of an axis under normal operation. When the axis is in the Stopping status, any instruction for this axis is invalid.
- When the axis is in Stopping status, Execute is FALSE, Done is True, and the axis status changes to Standstill.
- This function block can run only in the Motion status and cannot run in any other status.
- The function block starts at the rising edge of Execute.
- When Busy indicating valid execution of MC_Stop is valid, starting MC_Stop again will make the system enter the Errorstop status.
- In Halt or Stop status, the axis variable bAvoidReversalOnHaltStop can be used to adjust the acceleration to avoid velocity reversal. For details, see the MC_Halt instruction.

4) Timing Diagram

The instruction can be run only when the axis is in the Motion status.

Execute of the function block must have a rising edge condition.

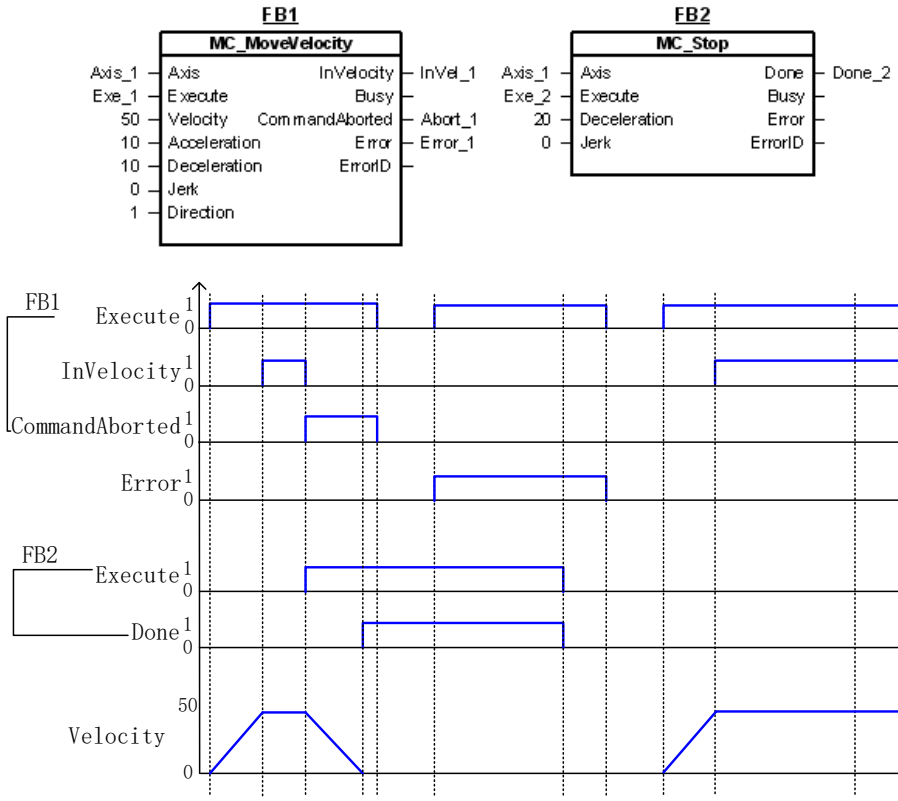
Done of the function block indicates that the execution of the instruction is completed.

Busy of the function block indicates that the execution of the instruction is in progress.

CommandAborted of the function block indicates that the instruction is aborted by other motion control instructions, in which case the flag bit is TRUE.

Programming example: Changes in the flag bits of the MC_MoveVelocity instruction and MC_Stop instruction in different timing operations.

The processing of CommandAborted is described in the following timing diagram.



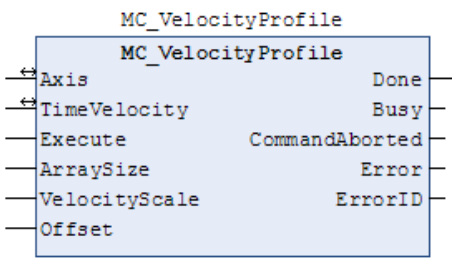
5) Error Description

When MC_Stop is run repeatedly, the error flag Error is True, and ErrorID is SMC_MS_AXI.

Note: For details on the error code, see “ “Appendix C Error Codes” .

MC_VelocityProfile

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ VelocityProfile	Velocity profile instruction		MC_VelocityProfile(Axis:= , TimeVelocity:= , Execute:= , ArraySize:= , VelocityScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
TimeVelocity	Axis velocity operation time and velocity description	MC_TV_REF	-	-	Axis velocity operation time and velocity data description, consisting of multiple sets of data.

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
ArraySize	Dynamic array	INT	Value Range	0	The number of arrays used in the operation profile
VelocityScale	Velocity factor	LREAL	"Positive", "0"	1	Velocity scale factor
Offset	Offset	LREAL	-	0	Overall offset of the velocity

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction execution completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is completed
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ ERROR	0	Output an error code when an error occurs

3) Function Description

- This function block is a profile motion model of time period and velocity. The operation mode is Continuous Motion. It runs based on the data set by the user for the TimeVelocity variable.
- This function block can run in Standstill, Continuous Motion, Synchronized Motion, or Discrete Motion status. The status during instruction running is Discrete Motion. It cannot run in other statuses.
- The function block is started at the rising edge of Execute. This instruction is repeated in Discrete Motion.
- TimeVelocity is of the MC_TV_REF data type.

MC_TV_REF description:

Member	Type	Initial Value	Description
Number_of_pairs	INT	0	Number of profile path segments
IsAbsolute	BOOL	TRUE	Absolute motion (TRUE) and relative motion option
MC_TV_Array	ARRAY[1..N] OF SMC_TV	-	Array of time and velocity

SMC_TV description:

Member	Type	Initial Value	Description
delta_time	TIME	TIME#0ms	Time of the velocity segment
Velocity	LREAL	0	Currently recorded velocity

Note: The whole velocity process is S-curve acceleration and deceleration. The velocity of each profile section is superimposed. When the instruction is repeatedly executed, the velocity is also superimposed. Avoid overspeed during instruction execution. In the case of repeated operation, the axis status must be reset to Standstill.

4) Timing Diagram

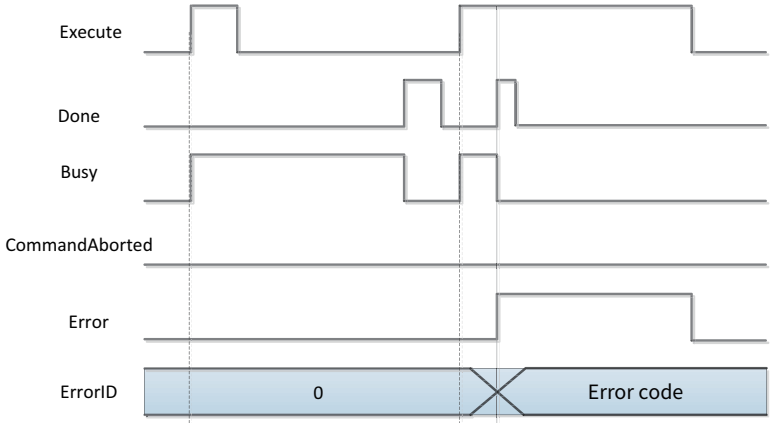
The position profile curve instruction can run only when the condition MC_TV_Array has been set by other means.

The instruction can run only when the axis is in Standstill status.

Execute of the function block must have a rising edge condition.

Done of the function block indicates that the execution of the instruction is completed.

Busy of the function block indicates that the execution of the instruction is in progress.



5) Error Description

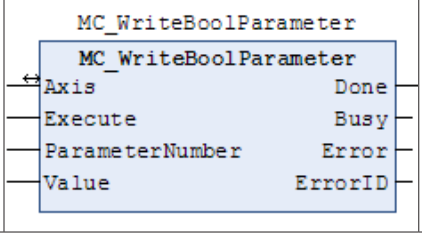
The error occurs as the instruction is not started in the axis status of Standstill or there is a parameter error in the instruction system. An axis error must be cleared before the start of the operation.

Note: For details on the error code, see “ “Appendix C Error Codes” .

MC_WriteBoolParameter

This instruction sets the bit parameter of the drive axis.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ WriteBoolParameter	Instruction for setting bit parameters		MC_WriteBoolParameter(Axis:= , Execute:= , ParameterNumber:= , Value:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Drive a setup operation for a rising edge operation
ParameterNumber	Serial number of axis parameter	DINT	-	0	Obtain the index, sub-index and serial number of axis parameters
Value	Value	BOOL	TRUE, FALSE	FALSE	Set the bit parameter value

Note: ParameterNumber (DINT) = -DWORD_TO_DINT(SHL(USINT_TO_DOWRD(usiDataLength), 24) (length of data in object dictionary)

+ SHL(UINT_TO_DWORD(uiIndex), 8) (index in object dictionary -16 bits)

+ usisubIndex(sub-index in object dictionary - 8 bits)

usiDataLength: Fill in bytes; 16#01 for 1 byte, 16#02 for 2 bytes, 16#04 for 4 bytes, and so on

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Setup operation successful	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the setup operation is successful
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

This instruction sets the bit parameter of the axis. It is started at the rising edge of Execute and can be

executed many times without affecting each other.

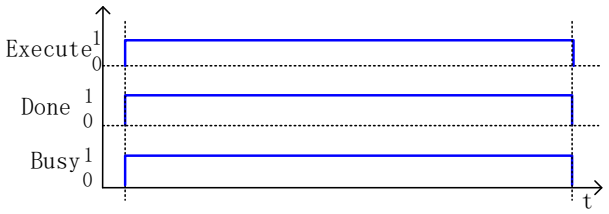
4) Timing Diagram

Execute of the function block must have a rising edge condition.

Done of the function block indicates that the setup operation is successful.

Busy of the function block indicates that the execution of the instruction is in progress.

◆ Timing operation description:



MC_WriteParameter

This instruction block modifies the parameters of the drive axis and saves them in the variable unit defined by itself.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ WriteParameter	Instruction for setting axis parameters	MC_WriteParameter MC_WriteParameter Axis Done Execute Busy ParameterNumber Error Value ErrorID	MC_WriteParameter(Axis:= , Execute:= , ParameterNumber:= , Value:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Drive a setup operation for a rising edge operation
ParameterNumber	Serial number of axis parameter	DINT	-	0	Obtain the index, sub-index and serial number of axis parameters
Value	Value	LREAL	-	-	Set the bit parameter value

Note: ParameterNumber (DINT) = -DWORD_TO_DINT(SHL(USINT_TO_DOWRD(usiDataLength), 24) (length of data in object dictionary)

+ SHL(UINT_TO_DWORD(uiIndex), 8) (index in object dictionary -16 bits)
+ usisubIndex(sub-index in object dictionary - 8 bits)

usiDataLength: Fill in bytes; 16#01 for 1 byte, 16#02 for 2 bytes, 16#04 for 4 bytes, and so on

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Setup operation successful	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the setup operation is successful
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ ERROR	0	Output an error code when an error occurs

3) Function Description

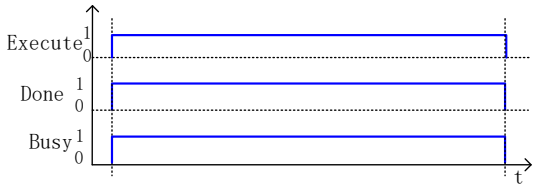
This instruction sets the bit parameter of the axis. It is started at the rising edge of Execute and can be executed many times without affecting each other.

4) Timing Diagram

Execute of the function block must have a rising edge condition.

Done of the function block indicates that the setup operation is successful.

Busy of the function block indicates that the execution of the instruction is in progress.



MC_AbortTrigger

This function block aborts event association related to the input latch, which is used in conjunction with MC_Touchprobe.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_AbortTrigger	Instruction for aborting event association	MC_AbortTrigger MC_AbortTrigger Axis — Done TriggerInput — Busy Execute — Error ErrorID	MC_AbortTrigger(Axis:= , TriggerInput:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
TruggerInput	Trigger signal	TRIGGER_REF	-	-	Description of trigger signals and trigger attributes

◆ TRIGGER_REF description:

Structure	Element	Data Type	Initial Value	Description

TRIGGER_REF	iTriggerNumber	INT	-1	Specify which of the functions is latched in drive mode. 0: Touch probe 1 latching at rising edge 1: Touch probe 1 latching at falling edge 2: Touch probe 2 latching at rising edge 3: Touch probe 2 latching at falling edge For details, see the IS620N Series Servo Design and Maintenance User Guide.
	bFastLatching	BOOL	TRUE	Specify the mode of latch trigger: TRUE: Drive mode FALSE: Controller mode
	bInput	BOOL	-	When bFastLatching = FLASE, it is triggered by controller input signal.
	bActive	BOOL	-	Active signal triggered

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Drive a setup operation for a rising edge operation

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Setup operation successful	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the setup operation is successful
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

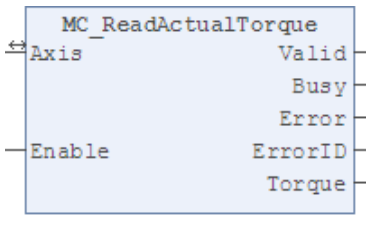
The MC_AbortTrigger function block aborts the association of a trigger signal or attribute with the related trigger instruction.
Execute of the function block must have a rising edge condition.
Done of the function block indicates that the setup operation is successful.

Busy of the function block indicates that the execution of the instruction is in progress.

MC_ReadActualTorque

This instruction reads the actual torque value of the drive and saves it in the variable unit defined by itself.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ ReadActualTorque	Instruction for reading the current torque value		MC_ReadActualTorque0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Torque=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Execution condition	BOOL	TRUE, FALSE	FALSE	Read the current position of the servo if set to TRUE

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Current torque value obtainable	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the drive torque can be obtained correctly
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ERROR	0	Output an error code when an error occurs
Torque	Current torque value obtained	LREAL	Torque value	0	Current torque data obtained by instruction

3) Function Description

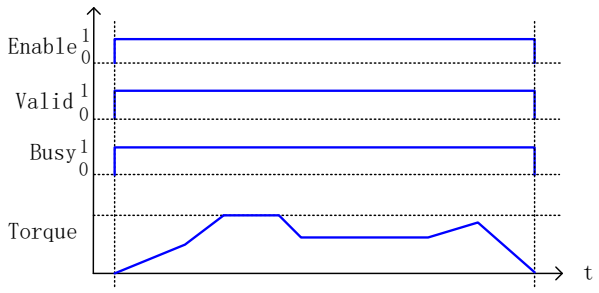
This instruction reads the actual torque value of the drive. It is active at high level of Enable and can be executed many times without affecting each other.

4) Timing Diagram

Enable of the function block must be set to TRUE.

Valid of the function block indicates that the value of Torque obtained is valid.

Busy of the function block indicates that the execution of the instruction is in progress.

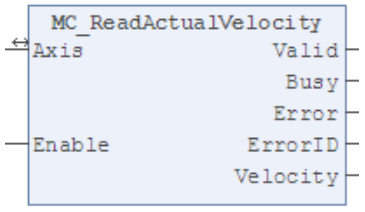


MC_ReadActualVelocity

This instruction reads the actual velocity value of the drive and saves it in the variable unit defined by itself.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression

MC_ ReadActualVelocity	Instruction for reading the current velocity		MC_ReadActualVelocity0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Velocity=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Execution condition	BOOL	TRUE, FALSE	FALSE	Read the current axis velocity when set to TRUE

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Current velocity value obtainable	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the drive velocity can be obtained correctly
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ ERROR	0	Output an error code when an error occurs
Velocity	Current velocity value obtained	LREAL	Velocity	0	Current velocity data obtained by instruction

3) Function Description

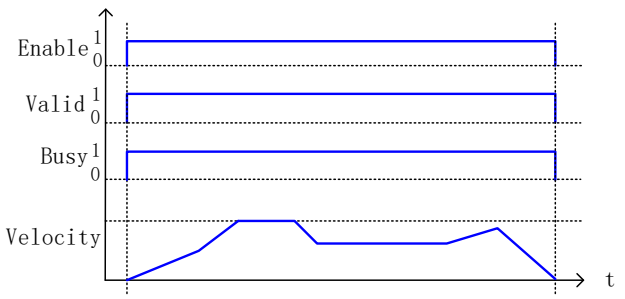
This instruction reads the actual velocity value of the drive. It is active at high level of Enable and can be executed many times without affecting each other.

4) Timing Diagram

Enable of the function block must be set to TRUE.

Valid of the function block indicates that the value of Velocity obtained is valid.

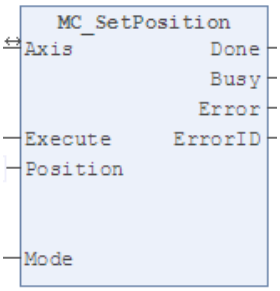
Busy of the function block indicates that the execution of the instruction is in progress.



MC_SetPosition

This instruction sets the axis position parameter to shift the coordinate system of an axis without any movement caused.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ SetPosition	Instruction for setting axis position parameters		MC_SetPosition0(Axis:= , Execute:= , Position:= , Mode:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Drive a setup operation for a rising edge operation
Position	Axis position data	LREAL	-	0	Position data
Mode	Value	BOOL	TRUE, FALSE	FALSE	Position mode; TRUE: Relative position; FALSE: Absolute position

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Setup operation successful	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the setup operation is successful
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

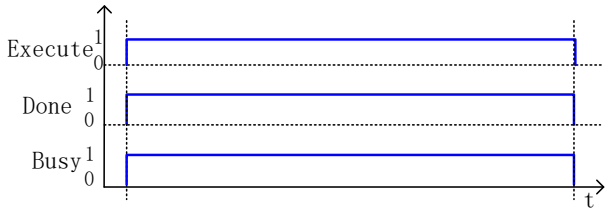
This instruction sets the axis position parameter to shift the coordinate system of an axis without any movement caused. It is started at the rising edge of Execute. This instruction can be executed many times without affecting each other.

4) Timing Diagram

Execute of the function block must have a rising edge condition.

Done of the function block indicates that the setup operation is successful.

Busy of the function block indicates that the execution of the instruction is in progress.



MC_TouchProbe

The instruction saves the position data of the current axis when triggered by an external signal.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_TouchProbe	Instruction for enabling external locking	MC_TouchProbe Axis TriggerInput Execute WindowOnly FirstPosition LastPosition Done Busy Error ErrorID RecordedPosition CommandAborted	MC_TouchProbe(Axis:= , TriggerInput:= , Execute:= , WindowOnly:= , FirstPosition:= , LastPosition:= , Done=> , Busy=> , Error=> , ErrorID=> , RecordedPosition=> , CommandAborted=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
TruggerInput	Trigger signal	TRIGGER_REF	-	-	Associated attributes such as trigger signal or trigger attribute

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Drive a setup operation for a rising edge operation
WindowOnly	Trigger window	BOOL	TRUE, FALSE	FALSE	
FirstPosition	Trigger start position	LREAL	-	0	Specify the start position for receiving trigger events
LastPosition	Trigger end position	LREAL	-	0	Specify the end position for receiving trigger events

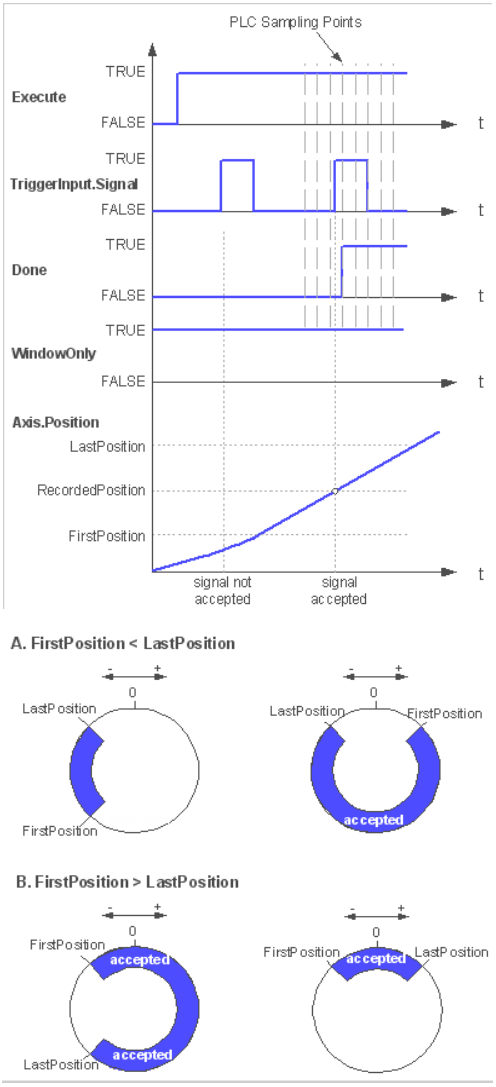
◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Setup operation successful	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the setup operation is successful
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ERROR	0	Output an error code when an error occurs
RecordedPosition	Trigger recording position	LREAL	-	0	Position where trigger event occurred

Output Variable	Name	Data Type	Value Range	Initial Value	Description
CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted

3) Function Description

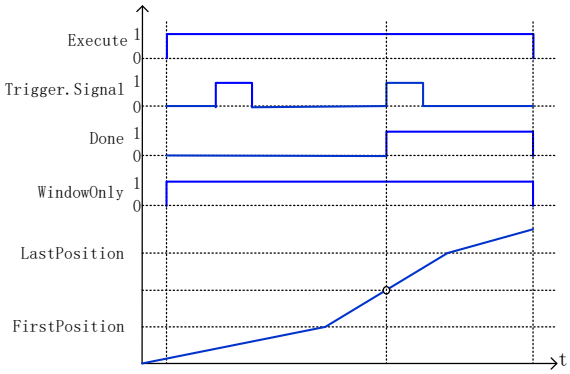
- This instruction records the current position of the running axis when triggered by signal TruggerInput.
- Execute is rising edge-triggered.
- When the drive is latched: The drive records the position in the controller after collecting the latching signal.



4) Timing Diagram

Execute of the function block must have a rising edge condition.

Done of the function block indicates that the setup operation is successful.



The position is a cyclic counting unit. The position for triggering signals can be reused.

SMC_MoveContinuousAbsolute

This function block commands the axis to move continuously to an absolute position (specified by Distance) ending with the specified velocity (EndVelocity).

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_MoveContinousAbsolute	Instruction for continuous control of the absolute axis position	SMC_MoveContinuousAbsolute_0 SMC_MoveContinuousAbsolute EN, Axis, Execute, Position, Velocity, EndVelocity, EndVelocityDirection, Acceleration, Deceleration, Jerk, Direction, AdaptEndVelToAvoidOvershoot ENO, InEndVelocity, PositionReached, Busy, CommandAborted, Error, ErrorID	SMC_MoveContinuousAbsolute_0(Axis:= Axis, Execute:= , Position:= , Velocity:= , EndVelocity:= , EndVelocityDirection:= , Acceleration:= , Deceleration:= , Jerk:= , Direction:= , AdaptEndVelToAvoidOvershoot:= , InEndVelocity=> , PositionReached=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
Position	Absolute position of the motion	LREAL	Value Range	0	This data is the absolute position of the motion.
Velocity	Running velocity	LREAL	Value Range	0	Maximum velocity of the axis to the target position
EndVelocity	Motion ending velocity	LREAL	Value Range	0	Running velocity after the instruction is executed
EndVelocity Direction	Direction of the ending velocity	MC_ Direction	Positive, negative, current	Current	Available: positive, negative, current; Not available: shortest, fastest
Acceleration	Acceleration	LREAL	Value Range	0	Acceleration rate for velocity increase
Deceleration	Deceleration	LREAL	Value Range	0	Deceleration rate for velocity decrease

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Jerk	Jerk	LREAL	Value Range	0	Slope change of the curve acceleration/deceleration
Direction	Running direction	Shortest	Value Range	Shortest	Options for linear/circular axes: positive, negative; Options for rotary/circular axes: positive, negative, current, shortest, fastest
AdaptEndVel ToAvoidOvershoot	End velocity adjustment flag	BOOL	TRUE, FALSE	FALSE	TRUE: Adjust the ending velocity to avoid overshoot; FALSE: No processing

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
InEndVelocity	Instruction position reached	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction position is reached
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

◆ This function block is an instruction for absolute axis positioning. The distance data is the absolute position of the axis.

◆ The operation status of this function block is Standstill, and the status during instruction running is Discrete Motion. A complete running process must control the different motion statuses of the axis.

◆ The motion is started at the rising edge of Execute. This instruction can be rising edge-triggered repeatedly in Discrete Motion to refresh the latest position data each time.

◆ If Acceleration or Deceleration is zero, the instruction execution will be abnormal. However, the state of the axis is Discrete Motion.

◆ If AdaptEndVelToAvoidOvershoot is TRUE, the valid ending velocity will be adjusted to avoid overshoot. In some cases, a given EndVelocity may cause position overshoot. For example, if the remaining distance is too short to reach the ending velocity from the current velocity and acceleration, the axis may rotate in negative direction, resulting in position overshoot. In another situation, the ending velocity is in negative direction of the movement distance. To reach the target position at this velocity, the axis will first exceed the distance, and then reversely accelerate to the ending velocity, which will result in position overshoot. To avoid overshoot:

If the remaining distance is too short to reach the desired EndVelocity, focus more on reaching EndVelocity, regardless of the target position constraints. In this way, the actual displacement will be greater than the target position, that is, the target position will be reached first (DistanceTravelled = TRUE), and then the target velocity will be reached (InTenVelocity = TRUE).

If the direction of the ending velocity is opposite to the motion direction, set the ending velocity corresponding to the target position to zero and then run from zero to the desired ending velocity. Note that the motion direction is not determined by the current velocity and acceleration but by the sign of

the distance to the target position.

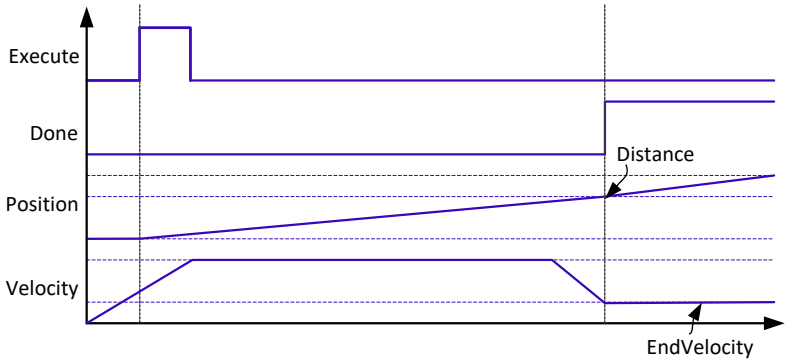
4) Timing Diagram

The instruction can run only when the axis is in Standstill status.

Execute of the function block must have a rising edge condition.

Done of the function block indicates that the execution of the instruction is completed.

Busy of the function block indicates that the execution of the instruction is in progress.



SMC_MoveContinuousRelative

This function block commands the axis to move continuously to a relative position (specified by Distance) ending with the specified velocity (EndVelocity).

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_MoveContinuousRelative	Axis relative positioning instruction	SMC_MoveContinuousRelative_0 SMC_MoveContinuousRelative EN, Axis, Execute, Distance, Velocity, EndVelocity, EndVelocityDirection, Acceleration, Deceleration, Jerk, AdaptEndVelToAvoidOvershoot ENO, InEndVelocity, DistanceTravelled, Busy, CommandAborted, Error, ErrorID	SMC_MoveContinuousRelative_0(Axis:= Axis, Execute:= , Distance:= , Velocity:= , EndVelocity:= , EndVelocityDirection:= , Acceleration:= , Deceleration:= , Jerk:= , AdaptEndVelToAvoidOvershoot:= , InEndVelocity=> , DistanceTravelled=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Execution condition	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
Distance	Relative position of the motion	LREAL	Value Range	0	This data is the relative position of the motion.
Velocity	Running velocity	LREAL	Value Range	0	Maximum velocity of the axis to the target position

Input Variable	Name	Data Type	Value Range	Initial Value	Description
EndVelocity	Motion ending velocity	LREAL	Value Range	0	Running velocity after the instruction is executed
EndVelocityDirection	Direction of the ending velocity	MC_Direction	Positive, negative, current	Current	Available: positive, negative, current; Not available: shortest, fastest
Acceleration	Acceleration	LREAL	Value Range	0	Acceleration rate for velocity increase
Deceleration	Deceleration	LREAL	Value Range	0	Deceleration rate for velocity decrease
Jerk	Jerk	LREAL	Value Range	0	Slope change of the curve acceleration/deceleration
AdaptEndVelTo AvoidOvershoot	End velocity adjustment flag	BOOL	TRUE, FALSE	FALSE	TRUE: Adjust the ending velocity to avoid overshoot; FALSE: No processing

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
InEndVelocity	Instruction position reached	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction position is reached
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ ERROR	0	Output an error code when an error occurs

3) Function Description

◆ The operation status of this function block is Standstill, and the status during instruction running is Discrete Motion. During the instruction execution, pay attention to the operation status of the axis, to avoid interrupting the execution of other instructions of the axis or being interrupted by other instructions of the axis.

◆ The motion is started at the rising edge of Execute. This instruction can be rising edge-triggered repeatedly in Discrete Motion to refresh the latest position data each time.

◆ If Acceleration or Deceleration is zero, the instruction execution will be abnormal. However, the state of the axis is Discrete Motion.

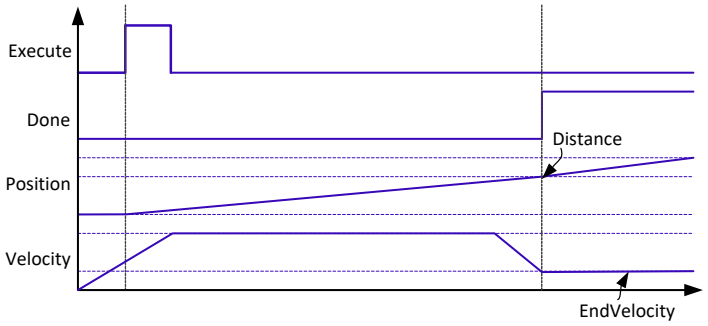
◆ If AdaptEndVelToAvoidOvershoot is TRUE, the valid ending velocity will be adjusted to avoid overshoot.

4) Timing Diagram

Execute of the function block must have a rising edge condition.

Done of the function block indicates that the execution of the instruction is completed.

Busy of the function block indicates that the execution of the instruction is in progress.



MC_Jog

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_Jog	Axis jog instruction	MC_Jog Axis JogForward JogBackward Velocity Acceleration Deceleration Jerk Busy CommandAborted Error ErrorId	MC_Jog(Axis:= , JogForward:= , JogBackward:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Busy=> , CommandAborted=> , Error=> , ErrorId=>);

2) Related Variables

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
JogForward	Valid in positive direction	BOOL	TRUE, FALSE	FALSE	Set to TRUE to start moving in positive direction; set to FALSE to stop moving in positive direction
JogBackward	Valid in negative direction	BOOL	TRUE, FALSE	FALSE	Set to TRUE to start moving in negative direction; set to FALSE to stop moving in negative direction
Velocity	Target velocity	LREAL	Positive number or 0	0	Specify the target velocity Unit: [instruction unit/s]
Acceleration	Acceleration	LREAL	Positive number or 0	0	Specify the acceleration rate Unit: [instruction unit/s]
Deceleration	Deceleration	LREAL	Positive number or 0	0	Specify the deceleration rate Unit: [instruction unit/s]

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
CommandAborted	Abortion of execution	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

3) Function Description

- This instruction performs jogging at the specified Velocity (target velocity).
- To jog in positive direction, set JogForward to TRUE. To jog in negative direction, set JogBackward to TRUE.
- If both JogForward and JogBackward are set to TRUE, no motion will occur.

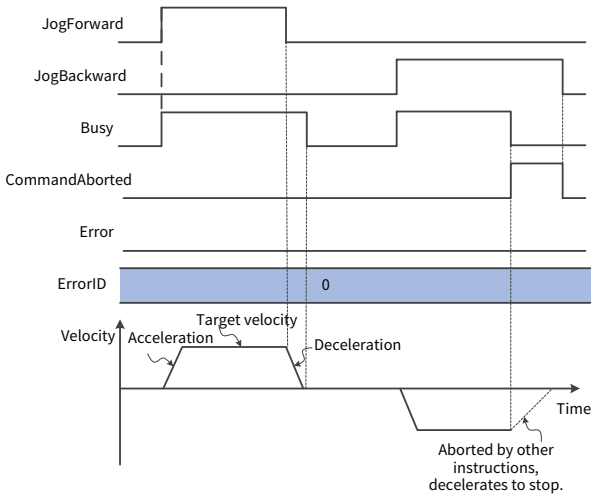
If the set velocity of the MC_Jog instruction exceeds the maximum jogging velocity in the axis parameter, the maximum jogging velocity will be followed.

4) Timing Diagram

Busy will change to TRUE while JogForward or JogBackward is activated.

Busy will change to FALSE while deceleration starts at the falling edge of JogForward or JogBackward and the axis stops.

When this instruction is aborted by another instruction, the value of CommandAborted changes to TRUE and that of Busy changes to FALSE.

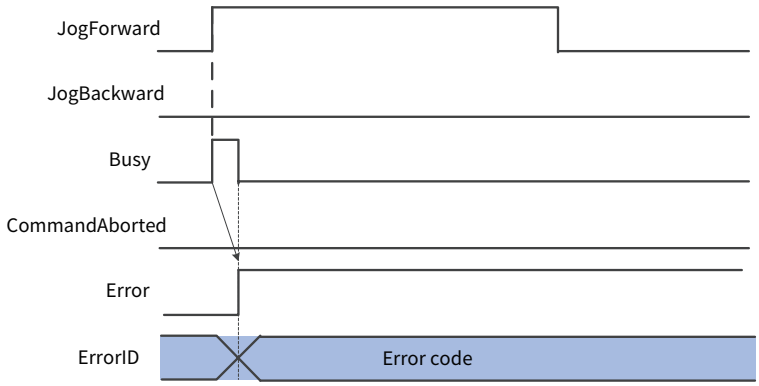


5) Error Description

If an error occurs during the execution of this instruction, Error changes to TRUE and the axis stops.

You can check the output value of ErrorID (error code) for the cause of error.

- Timing diagram when an exception occurs

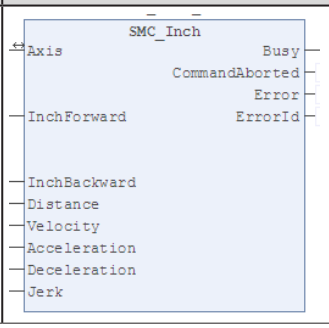


For details on the error code, see “ “Appendix C Error Codes” .

SMC_Inch

This instruction performs single-step motion control of axes through the program.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_Inch	Axis relative positioning instruction		SMC_Inch0(Axis:= , InchForward:= , InchBackward:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Busy=> , CommandAborted=> , Error=> , ErrorId=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
InchForward	Execution in positive direction	BOOL	TRUE, FALSE	FALSE	If InchForward is TRUE, the axis will move at the given velocity (Velocity, Acceleration, or Deceleration) in positive direction until the distance is reached. To start the motion again, the input must be specified as FALSE and then TRUE. If InchForward is set to FALSE before it reaches position, then the axis will immediately decelerate to 0 and Busy will be set to FALSE. If InchBackward is set to TRUE in the simulation, no motion will occur.
InchBackward	Backward execution	BOOL	TRUE, FALSE	FALSE	If InchBackward is TRUE, the axis will move at the given velocity (Velocity, Acceleration, or Deceleration) in positive direction to the specified position. To start another motion, the input must be set to FALSE and then TRUE. If InchForward is set to TRUE at the same time, no axis motion will occur.
Distance	Movement distance	LREAL	Value Range	0	This data is the movement distance
Velocity	Running velocity	LREAL	Value Range	0	Maximum velocity of the axis to the target position
Acceleration	Acceleration	LREAL	Value Range	0	Acceleration rate for velocity increase
Deceleration	Deceleration	LREAL	Value Range	0	Deceleration rate for velocity decrease

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Busy	Instruction execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is being executed
CommandAbort	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ ERROR	See SMC_ ERROR	0	Output an error code when an error occurs

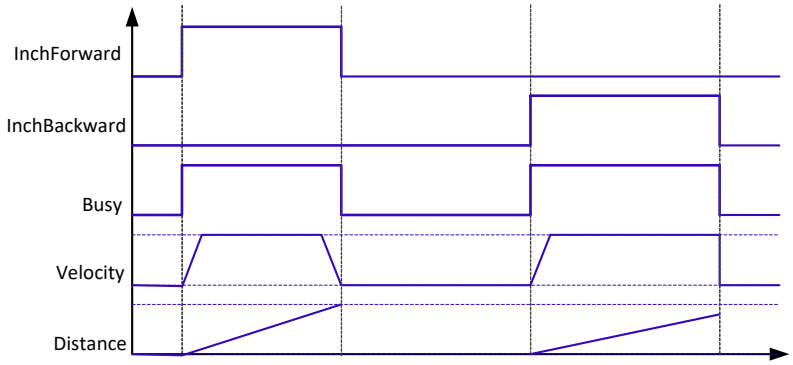
3) Function Description

- The operation status of this function block is Standstill, and the status during instruction running is Discrete Motion. During the instruction execution, pay attention to the operation status of the axis, to avoid interrupting the execution of other instructions of the axis or being interrupted by other instructions of the axis.
- If Acceleration or Deceleration is zero, the instruction execution will be abnormal. However, the state of the axis is Discrete Motion.

4) Timing Diagram

InchForward/InchBackward of the function block must have a TRUE/FALSE condition.

Busy of the function block indicates that the execution of the instruction is in progress.



SMC3_PersistPosition

This instruction keeps recording the position of the absolute encoder of the real axis (after the controller is powered off and restarted again, the position value recorded before power-off will be restored). If the servo motor uses an absolute encoder, this function block can be used in conjunction with it.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC3_ PersistPosition	Instruction for keeping the axis position	SMC3_PersistPosition0 SMC3_PersistPosition Axis PersistentData bEnable bPositionRestored bPositionStored bBusy bError eErrorID eRestoringDiag	SMC3_PersistPosition0(Axis:= , PersistentData:= , bEnable:= , bPositionRestored=> , bPositionStored=> , bBusy=> , bError=> , eErrorID=> , eRestoringDiag=>);

2) Related Variables

- Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
PersistentData	Retentive data	SMC3_PersistPosition_ Data	-	-	Power-down retentive data structure for storing position information

- Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bEnable	Executed	BOOL	TRUE, FALSE	FALSE	The function block is executed if set to TRUE and not executed if set to FALSE. To restore the last stored position during initialization, this value must be set to TRUE from application startup.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bPositionRestored	Position restoring	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the position is restored upon axis restart
bPositionStored	Position saving	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the position is stored after an FB call
bBusy	FB execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when FB execution is not completed.
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
eErrorID	Error code	SMC_ERROR	-	SMC_NO_ERROR	Output an error code when an error occurs
eRestoringDiag	Restoration diagnostics	SMC3_Persist-PositionDiag		SMC3_ PersistPositionDiag. SMC3_PPD_ RESTORING_OK	Diagnostic information in position restoration SMC3_PPD_RESTORING_OK: Position successfully restored; SMC3_PPD_AXIS_PROP_CHANGED: Axis parameters have been changed and the position could not be restored; SMC3_PPD_DATA_STORED_DURING_ WRITING: The function block copies data from the axis parameter data structure instead of from PersistentData. Possible cause: Non-synchronized retentive variable, controller crash

3) Function Description

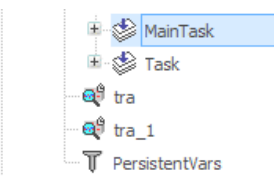
◆ If bEnable is TRUE upon PLC restart, then bPositionRestroed outputs TRUE.

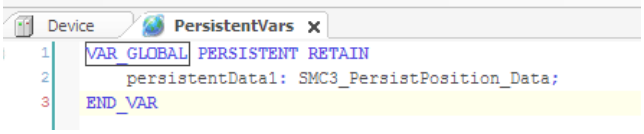
◆ Virtual and logical axes are not supported.

◆ It is hereby declared that the actual position of the axis in AM600 is: Offset + Encoder feedback position (instruction unit Plus) x Scale. The position recorded by the absolute encoder after power-off is the instruction unit value. Therefore, to restore the "actual position" before power-off upon PLC restart, use this function block and configure SMC3_PersistPosition_Data as a retentive variable.

Usage (when the real axis encoder is a multi-turn absolute encoder):

6) SMC3_PersistPosition_Data declared in PersistentVars





7) Called in the PLC main task (EthCat task)

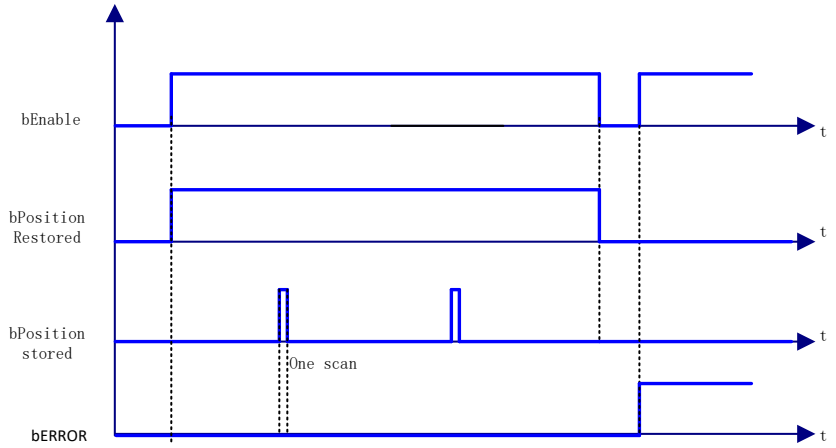
◆ Declaration section:

```
VAR
    SMC3_PersistPosition_1:SMC3_PersistPosition;
END_VAR
```

Program section:

```
SMC3_PersistPosition_1(Axis:=X_Axis , PersistentData:=persistentData1 ,bEnable:=TRUE );
```

4) Timing Diagram



5) Error Description

If the input axis is a virtual or logical one, an error will be output. An axis error will result in an error output.

Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC3_PersistPositionSingleturn

This instruction keeps recording the position of the absolute encoder (single-turn) of the real axis (after the controller is powered off and restarted again, the position value recorded before power-off will be restored). If the servo motor uses a single-turn absolute encoder, use this function block in conjunction with it.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC3_ PersistPositionSingleturn	Instruction for keeping the axis position	SMC3_PersistPositionSingleturn_0 SMC3_PersistPositionSingleturn Axis → bPositionRestored PersistentData → bPositionStored bBusy bError bEnable → eErrorID usiNumberOfAbsoluteBits → eRestoringDiag	SMC3_PersistPositionSingleturn_0(Axis:= , PersistentData:= , bEnable:= , usiNumberOfAbsoluteBits:= , bPositionRestored=> , bPositionStored=> , bBusy=> , bError=> , eErrorID=> , eRestoringDiag=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
PersistentData	Retentive data	SMC3_PersistPositionSingletrun_Data	-	-	Power-down retentive data structure for storing position information

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bEnable	Executed	BOOL	TRUE, FALSE	FALSE	The function block is executed if set to TRUE and not executed if set to FALSE. To restore the last stored position during initialization, this value must be set to TRUE from application startup.
usiNumberofAbsoluteBitesusiNumberofAbsoluteBites	Bit	UINT	-	16	Specify the bits of absolute encoder (such as 20-bit and 24-bit)

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bPositionRestored	Position restoring	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the position is restored upon axis restart
bPositionStored	Position saving	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the position is stored after an FB call
bBusy	FB execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when FB execution is not completed.
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
eErrorID	Error code	SMC_ERROR	-	SMC_NO_ERROR	Output an error code when an error occurs
eRestoringDiag	Restoration diagnostics	SMC3_PersistPositionDiag	-	SMC3_PersistPositionDiag.SMC3_PPD_RESTORING_OK	Diagnostic information in position restoration SMC3_PPD_RESTORING_OK: Position successfully restored; SMC3_PPD_AXIS_PROP_CHANGED: Axis parameters have been changed and the position could not be restored; SMC3_PPD_DATA_STORED_DURING_WRITING: The function block copies data from the axis parameter data structure instead of from PersistentData. Possible cause: Non-synchronized retentive variable, controller crash

3) Function Description

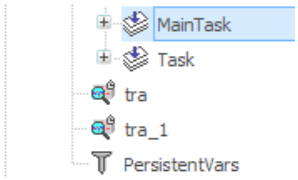
◆ If bEnable is TRUE upon PLC restart, then bPositionRestroed outputs TRUE.

◆ Virtual and logical axes are not supported.

◆ To restore the "actual position" before power-off upon PLC restart, use this function block and configure SMC3_PersistPositionSingleTurn_Data as a retentive variable.

Usage (when the real axis encoder is a multi-turn absolute encoder):

8) SMC3_PersistPositionSingleTurn_Data declared in PersistentVars




```
VAR_GLOBAL PERSISTENT RETAIN
    persistentData2: SMC3_PersistPositionSingleTurn_Data;
END_VAR
```

9) Called in the PLC main task (EthCat task)

◆ Declaration section:

VAR

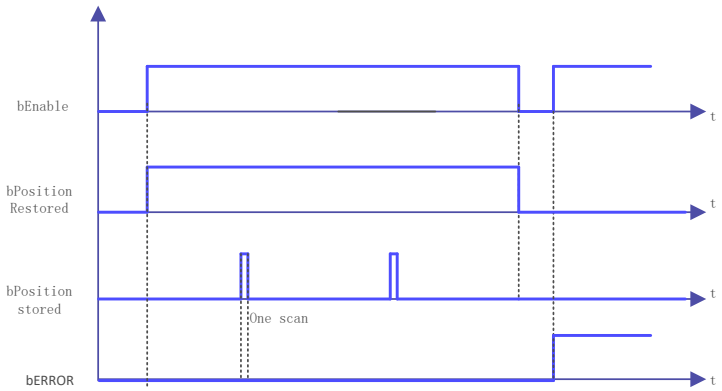
SMC3_PersistPosition_2: SMC3_PersistPositionSingleTurn_Data;

END_VAR

◆ Program section:

```
SMC3_PersistPosition_2(Axis:=Y_Axis , PersistentData:=persistentData2 ,bEnable:=TRUE );
```

4) Timing Diagram



5) Error Description

If the input axis is a virtual or logical one, an error will be output. An axis error will result in an error output.

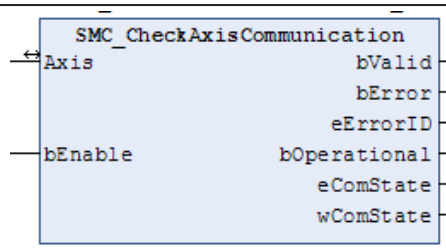
Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC_CheckAxisCommunication

This instruction checks the current communication status of the drive.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression

SMC_ CheckLimits	Axis limit check instruction		SMC_CheckAxisCommunication0(Axis:= , bEnable:= , bValid=> , bError=> , eErrorID=> , bOperational=> , eComState=> , wComState=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bEnable	Executed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when a check is in progress

◆ Output Variable

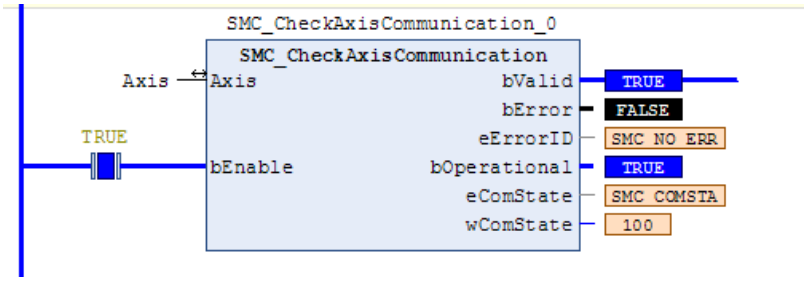
Output Variable	Name	Data Type	Value Range	Initial Value	Description
bValid	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when instruction execution is valid
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
eErrorID	Error code	SMC_ERROR			See SMC_Error
bOperational	Communication normal	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the communication is normal (code 100) and operations can be performed on the axis Set to FALSE when the communication is abnormal and operations cannot be performed on the axis
eComState	Communication state	SMC_COMMUNI-CATIONSTATE	-	-	Including: SMC_COMSTATE_NOT_STARTED: Communication not started SMC_COMSTATE_VARIABLE_INITIALIZATION: Communication variable initialization SMC_COMSTATE_BASE_COM_INITIALIZATION: Base port initialization SMC_COMSTATE_DRIVE_INITIALIZATION: Communication drive initialization SMC_COMSTATE_DRIVE_WAITING_FOR_SYNC: Warning for synchronization SMC_COMSTATE_INITIALIZATION_DONE: Initialization done SMC_COMSTATE_OPERATIONAL: Normal communication SMC_COMSTATE_REINITIALIZATION: Communication re-initialization SMC_COMSTATE_ERROR: Communication error SMC_COMSTATE_UNKNOWN: Unknown communication state
wComState	Communication code	WORD	-	-	Same value as Axis.wCommunicationState in the input/output axis structure variable. Code indicating the current communication state. See AXIS_REF_SM3 parameter 1013.

3) Function Description

When bEnable is TRUE, no error occurs, and bValid outputs TRUE, this instruction checks the axis communication state.

When bValid outputs TRUE, this instruction checks the axis communication state. When eComState outputs SMC_COMSTATE_OPERATIONAL, bOperational outputs TRUE.

4) Sample Program



5) Error Description

At the rising edge of bExecute:

An error is output if there is an axis error.

An error is output if the axis input is invalid.

Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC_FollowPosition

This instruction sets the axis position without performing any check. This instruction is different from MC_MoveAbsolute in that after the bExecute rising edge signal arrives, it will give the axis position instruction in each task period regardless of the axis status. (Users can use this instruction to write cam functions instead of using instructions such as MC_CamIn.)

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_ FollowPosition	Axis position reference instruction	SMC_FollowPosition Axis bBusy bCommandAborted bError bExecute iErrorID fSetPosition	SMC_FollowPosition_0(Axis:= , bExecute:= , fSetPosition:=SET_POSITION , bBusy=> , bCommandAborted=> , bError=> , iErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bExecute	Executed	BOOL	TRUE, FALSE	FALSE	Execute the FB at the rising edge
fSetPosition	Set position	LREAL	-	0	Axis position setting

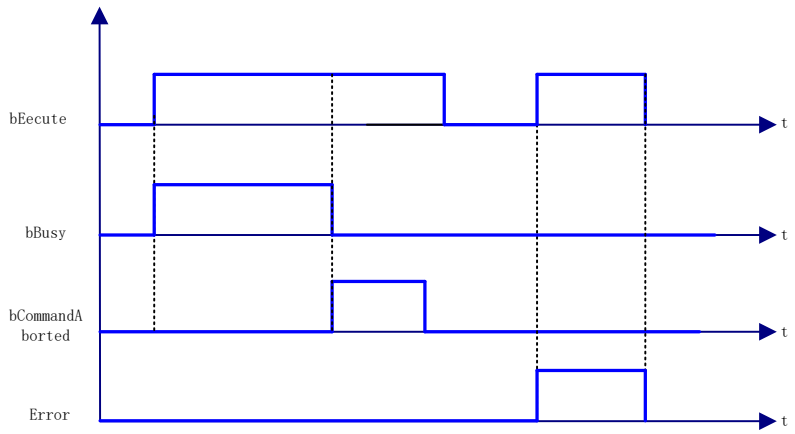
◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed (At this time, the axis is in the synchronized status, which is the same as the axis status during the execution of the cam MC_CamIn instruction). The bBusy status can be cleared with the MC_Camout instruction.
bCommandAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis is interrupted by other control instructions
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
iErrorID	Error code	SMC_ERROR	-	-	See SMC_Error

3) Function Description

- After SMC_FollowPosition is started at the rising edge of bExecute, the axis will send position instruction to the axis in each task period.
- When the bBusy signal is received, the axis is in Synchronized Motion status, which is the same as the slave axis when the MC_CamIn instruction takes effect. The status can be cleared with the MC_CamOut instruction.
- The axis velocity is calculated by the increment of the position difference between two task periods: Velocity = $\Delta L / \Delta t$. ΔL is the difference between fSetVelocity of this task period and fSetVelocity of the previous task period. Δt is the scanning time.
- When the bExecute signal is TRUE, bBusy changes from TRUE to FALSE when this instruction is interrupted by another instruction.

4) Timing Diagram



5) Error Description

At the rising edge of bExecute:

An error is output when the Axis variable is connected to a non-AXIS_REF_SM3 type structure variable.

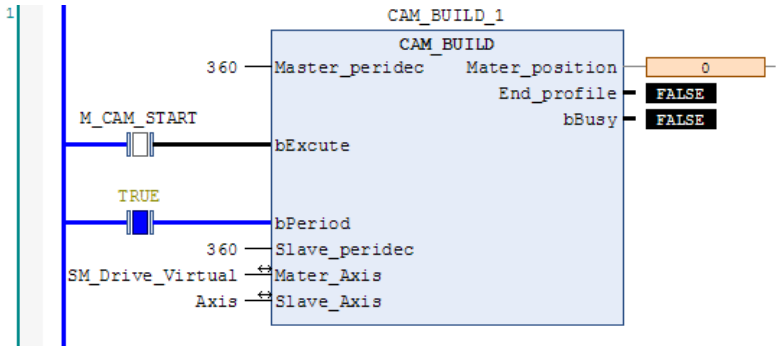
An error is output when the axis is disabled.

An error is output when there is an axis error during instruction execution.

Note: For details on the error code, see “ “Appendix C Error Codes” .

6) Sample Description

Use SMC_FollowPosition to achieve the electronic cam function.



FB variable definition section:

```
FUNCTION_BLOCK CAM_BUILD
VAR_INPUT//Input variable definition
    Master_peridec:REAL; //Master axis period
    bExcute:BOOL; //Instruction execution
    bPeriod:BOOL; //Cyclic cam execution; false: single-period execution
    Slave_peridec:REAL; //Slave period
END_VAR
VAR_OUTPUT; //Output variable definition
    Mater_position:LREAL;//Master axis position (the position of the master axis calculated after the start of instruction execution)
    End_profile:BOOL; //Flag bit of curve output completed
    bBusy:BOOL; //Execution in progress
END_VAR
VAR//FB intermediate variable definition
    SMC_FollowPosition_0: SMC_FollowPosition;
    SET_POSITION: LREAL;
    SET_POSITIONOLD: LREAL;
    Mater_positionOLD:LREAL;
    bExcute_old:BOOL;
    INC:LREAL;
    Y:LREAL;
    X5:LREAL;
    X4:LREAL;
    X3:LREAL;
    X2:LREAL;
    X1:LREAL;
    MC_Stop0: MC_Stop;
    STOP:BOOL;
    COUNTNUM:DINT;
    SET_INC:LREAL;
    YOLD:LREAL;
    SMC_FollowPositionVelocity_0: SMC_FollowPositionVelocity;
    K:REAL;
    K_OUT:REAL;
    MC_CamOut_0: MC_CamOut;
END_VAR
VAR_IN_OUT// Input and output variable definition
    Mater_Axis:AXIS_REF_SM3;
    Slave_Axis:AXIS_REF_SM3;
END_VAR
```

Program section:

```
IF bExcute AND NOT bExcute_old THEN //Rising edge initialization parameter
        Mater_position:=0;
        Mater_positionOLD:=Mater_Axis.fActPosition;
        End_profile:=FALSE;
        SET_POSITION:=Slave_Axis.fActPosition;
        SET_POSITIONOLD:=Slave_Axis.fActPosition;
        COUNTNUM:=0;
        YOLD:=0;
        K:=0;
ELSE
  IF bExcute_old THEN
         INC:=Mater_Axis.fActPosition-Mater_positionOLD;//Increment of master axis task period
         IF INC<0 THEN //Master axis code position exceeds zero point (when axis mode is set to modulo)
                  INC:=Mater_Axis.fActPosition-Mater_positionOLD+Mater_Axis.fPositionPeriod;
         END_IF
        Mater_position:=INC+Mater_position;//Current position of master axis
        Mater_positionOLD:=Mater_Axis.fActPosition;
         //********Curve judgment completed*********//
         IF Mater_position>=Master_peridec THEN
                  End_profile:=TRUE;
          ELSE
                  End_profile:=FALSE;
         END_IF
         IF bPeriod THEN
          IF Mater_position>=Master_peridec THEN
                  Mater_position:=Mater_position-Master_peridec;
          END_IF
         END_IF
  END_IF
END_IF
IF bExcute_old THEN
X1:=(Mater_position/Master_peridec);
         X2:=X1*X1;
         X3:=X2*X1;
         X4:=X3*X1;
         X5:=X4*X1;
         Y:=(6*X5-15*X4+10*X3)*Slave_peridec;//Slave axis position, curve
         K:=(30*X4-60*X3+30*X2)*Slave_peridec/Master_peridec;//Curve slope
         SET_INC:=Y-YOLD;
        IF  SET_INC<0 THEN
                  SET_INC:=Slave_peridec-YOLD+Y;
         END_IF
         YOLD:=Y;
         IF bPeriod THEN
                          SET_POSITION:=SET_POSITION+SET_INC;
    ELSE
                  IF End_profile THEN
       SET_POSITION:=SET_POSITIONOLD+Slave_peridec;
                          ELSE
                          SET_POSITION:=SET_POSITION+SET_INC;
          END_IF
```

```
        END_IF
        IF SET_POSITION>=Slave_Axis.fPositionPeriod THEN
                SET_POSITION:=SET_POSITION-Slave_Axis.fPositionPeriod;
        END_IF
END_IF
SMC_FollowPosition_0(
        Axis:=Slave_Axis,
        bExecute:=bExcute,
        fSetPosition:=SET_POSITION ,
        bBusy=>bBusy ,
        bCommandAborted=> ,
        bError=> ,
        iErrorID=> );
 IF NOT bExcute AND bExcute_old THEN
        STOP:=TRUE;
 END_IF
MC_CamOut_0(
        Slave:=Slave_Axis,
        Execute:= STOP,
        Done=> ,
        Busy=> ,
        Error=> ,
        ErrorID=> );
MC_Stop0(
        Axis:=Slave_Axis,
        Execute:= MC_CamOut_0.Done OR MC_CamOut_0.Error ,
        Deceleration:=20000 ,
        Jerk:= 20000,
        Done=> ,
        Busy=> ,
        Error=> ,
        ErrorID=> );
        IF MC_Stop0.Done OR MC_Stop0.Error THEN
                STOP:=FALSE;
        END_IF
        IF NOTbExcute_old THEN
                End_profile:=FALSE;
        END_IF
bExcute_old:=bExcute;
```

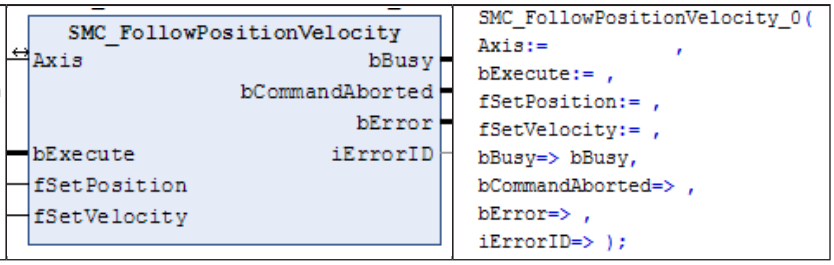
SMC_FollowPositionVelocity

The usage and function of this instruction are the same as SMC_FollowPosition. This instruction additionally provides velocity setting.

Note: The velocity setting must adapt to the position setting change. Velocity setting = First order derivative of the difference in position settings between task periods with respect to time. For example, if the position setting is the same between two periods, the velocity must be set to 0; otherwise, it will cause the motor to vibrate violently.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression

SMC_ FollowPositionVelocity	Axis position and velocity reference instruction	

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_ REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bExecute	Executed	BOOL	TRUE, FALSE	FALSE	Execute the FB at the rising edge
fSetPosition	Set position	LREAL	-	0	Axis position setting
fSetVelocity	Set velocity	LREAL	-	0	Axis velocity setting

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed (At this time, the axis is in the synchronized status, which is the same as the axis status during the execution of the cam MC_CamIn instruction). The bBusy status can be cleared with the MC_ Camout instruction.
bCommandAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis is interrupted by other control instructions
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
iErrorID	Error code	SMC_ERROR	-	-	See SMC_Error

3) Function Description

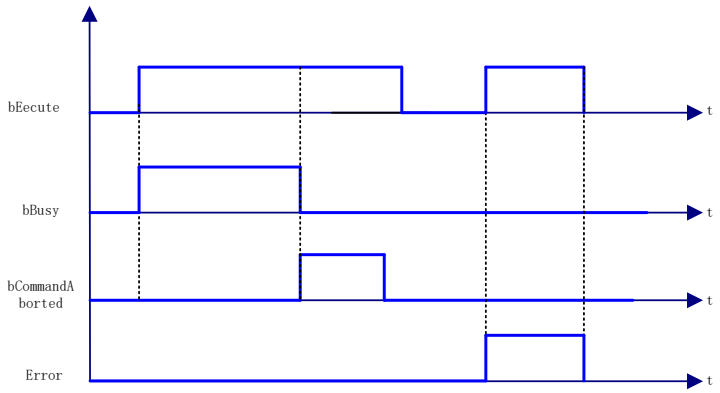
◆ After SMC_FollowPositionVelocity is started at the rising edge of bExecute, the axis will send the position and velocity setting instruction to the axis in each task period.

◆ When the bBusy signal is received, the axis is in Synchronized Motion status, which is the same as the slave axis when the MC_CamIn instruction takes effect. The status can be cleared with the MC_CamOut instruction.

◆ The velocity setting must adapt to the position setting change. fSetVelocity = $\Delta L / \Delta t \cdot \Delta L$ is the difference between fSetVelocity of this task period and fSetVelocity of the previous task period. Δt is the scanning time.

◆ When the bExecute signal is TRUE, bBusy changes from TRUE to FALSE when this instruction is interrupted by another instruction.

4) Timing Diagram



5) Error Description

At the rising edge of bExecute:

An error is output when the Axis variable is connected to a non-AXIS_REF_SM3 type structure variable.

An error is output when the axis is disabled.

An error is output when there is an axis error during instruction execution.

Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC_FollowVelocity

This instruction sets the axis velocity without performing any check. This instruction is different from MC_MoveVelocity in that after the execution of the rising edge model, it will give the axis velocity instruction in each task period. (The MC_MoveVelocity instruction must be refreshed to take effect after the velocity is changed.)

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_ FollowVelocity	Axis velocity reference instruction	SMC_FollowVelocity Axis bExecute fSetVelocity bBusy bCommandAborted bError iErrorID	SMC_FollowVelocity_0(Axis:= , bExecute:= , fSetVelocity:= , bBusy=> , bCommandAborted=> , bError=> , iErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_ REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bExecute	Executed	BOOL	TRUE, FALSE	FALSE	Execute the FB at the rising edge
fSetVelocity	Set velocity	LREAL	-	0	Axis velocity setting

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed (At this time, the axis is in the synchronized status, which is the same as the axis status during the execution of the cam MC_CamIn instruction). The bBusy status can be cleared with the MC_Camout instruction.
bCommandAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis is interrupted by another control instruction (when bExecute is True)
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
iErrorID	Error code	SMC_ERROR	-	-	See SMC_Error

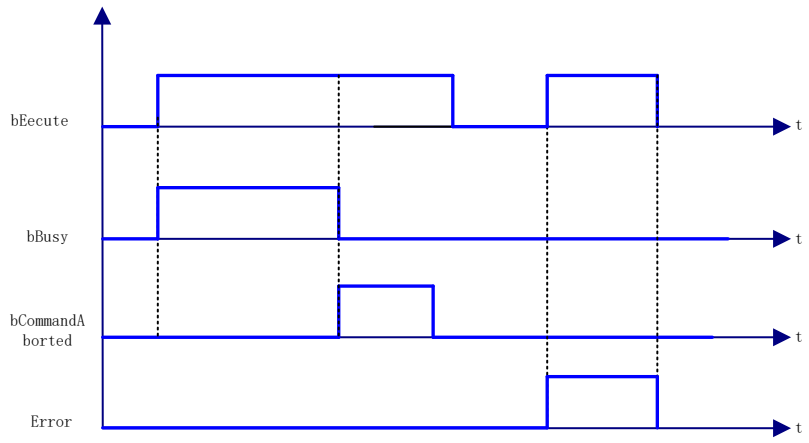
3) Function Description

◆ After SMC_FollowVelocity is started at the rising edge of bExecute, the axis will send the velocity instruction to the axis in each task period.

◆ When the bBusy signal is received, the axis is in Synchronized Motion status, which is the same as the slave axis when the MC_CamIn instruction takes effect. The status can be cleared with the MC_CamOut instruction.

◆ When the bExecute signal is TRUE, bBusy changes from TRUE to FALSE when this instruction is interrupted by another instruction.

4) Timing Diagram



5) Error Description

At the rising edge of bExecute:

An error is output when the Axis variable is connected to a non-AXIS_REF_SM3 type structure variable.

An error is output when the axis is disabled.

An error is output when there is an axis error during instruction execution.

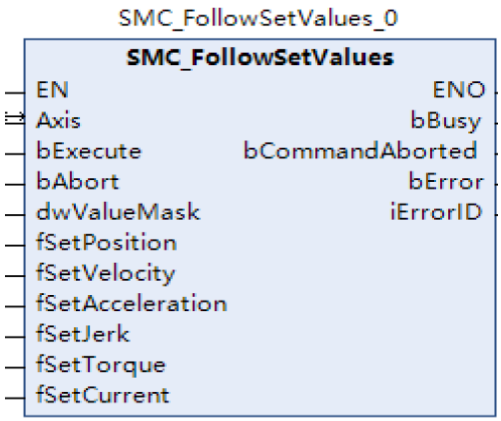
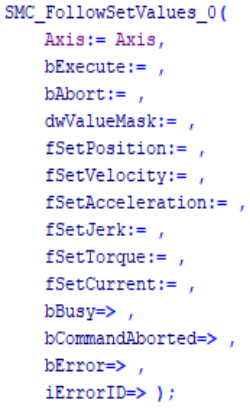
Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC_FollowSetValues

Like other SMC_Follow functions, it directly sends an instruction to the axis. However, this instruction not only includes other SMC_Follow functions, but also includes acceleration, current, torque and

other control signals. Therefore, it can be regarded as a comprehensive instruction. Users can select the desired through the DwValueMask value.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_ FollowSetValues	Axis-related instruction reference		


```
SMC_FollowSetValues_0(
    Axis:= Axis,
    bExecute:= ,
    bAbort:= ,
    dwValueMask:= ,
    fSetPosition:= ,
    fSetVelocity:= ,
    fSetAcceleration:= ,
    fSetJerk:= ,
    fSetTorque:= ,
    fSetCurrent:= ,
    bBusy=> ,
    bCommandAborted=> ,
    bError=> ,
    iErrorID=> );
```

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF	-	-	Reference to the axis, that is, an instance of AXIS_ REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bExecute	Start	BOOL	TRUE, FALSE	FALSE	Execute the FB at the rising edge
bAbort	Abort condition	BOOL	TRUE, FALSE	FALSE	Abort the ongoing motion and reset all outputs
dwValueMask	Control management	DWORD	-	0	Bit 0: TRUE: fSetPosition active; FALSE: Ignore Bit 1: TRUE: fSetVelocity active; FALSE: Ignore Bit 2: TRUE: fSetAcceleration active; FALSE: Ignore Bit 3: TRUE: fSetJerk active; FALSE: Ignore Bit 4: TRUE: fSetTorque active; FALSE: Ignore Bit 5: TRUE: fSetCurrent active; FALSE: Ignore
fSetPosition	Set position	LREAL	-	0	Axis position setting (calibrated unit)
fSetVelocity	Set velocity	LREAL	-	0	Axis velocity setting (calibrated unit)
fSetAcceleration	Set acceleration	LREAL	-	0	Axis acceleration setting (calibrated unit/s2)
fSetJerk	Set jerk	LREAL	-	0	Axis jerk setting (calibrated unit/s3)
fSetTorque	Torque reference	LREAL	-	0	Axis torque setting (NM/N)
fSetCurrent	Set current	LREAL	-	0	Axis current setting (A)

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
bCommandAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs

Output Variable	Name	Data Type	Value Range	Initial Value	Description
iErrorID	Error code	SMC_ERROR	-	0	Output an error code when an error occurs

3) Function Description

- After SMC_FollowSetValues is started at the rising edge of bExecute, it will send the selected parameter instruction to the axis in each task period.
- ● When the bBusy signal is received, the axis is in Synchronized Motion status, which is the same as the slave axis when the MC_CamIn instruction takes effect. The status can be cleared with the MC_CamOut instruction.
- ● When the bExecute signal is TRUE, bBusy changes from TRUE to FALSE when this instruction is interrupted by another instruction.
- ● The control parameter can be selected through the value of DwValueMask. For example, if DwValueMask is 1, it sends the position for each task period, with the same function as the SMC_FollowPosition instruction. If DwValueMask is 2, it is an instruction output for the velocity. If DwValueMask is 3, it is an instruction output for the position and velocity. If DwValueMask is 7, it is an instruction output for the position, velocity, acceleration, and so on.

4) Timing Diagram

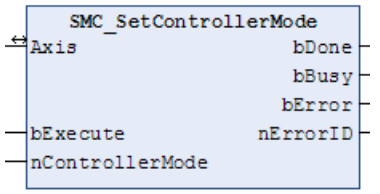
```
SMC_FollowSetValues_0(
    Axis:= Axis,
    bExecute:= ,
    bAbort:= ,
    dwValueMask:= ,
    fSetPosition:= ,
    fSetVelocity:= ,
    fSetAcceleration:= ,
    fSetJerk:= ,
    fSetTorque:= ,
    fSetCurrent:= ,
    bBusy=> ,
    bCommandAborted=> ,
    bError=> ,
    iErrorID=> );
```

SMC_SetControllerMode

SMC_SetControllerMode sets the current operation mode of the servo. By default, synchronous cyclic position control is adopted. For details on the control mode setting, see IS620N Series Servo Design and Maintenance User Guide.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression

SMC_ SetControllerMode	Instruction for setting axis control mode		SMC_SetControllerMode0(Axis:= , bExecute:= , nControllerMode:= , bDone=> , bBusy=> , bError=> , nErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

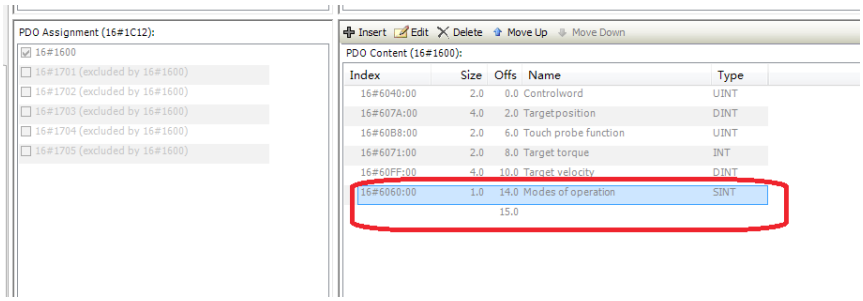
Input Variable	Name	Data Type	Value Range	Initial Value	Description
bExecute	Executed	BOOL	TRUE, FALSE	FALSE	Execute the FB at the rising edge
nControllerMode	Control mode	SMC_CONTROLLER_ MODE	-	SMC_ Position	Axis control mode 1: SMC_torque: Torque control mode 2: SMC_Velocity: Velocity control mode 3: SMC_Position: Position control mode 4: SMC_Current: Current control mode

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bDone	Mode setting completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when mode setting is completed
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
iErrorID	Error code	SMC_ERROR	-	-	See SMC_Error

3) Function Description

◆ After being started at the rising edge of bExecute, SMC_SetControllerMode sets the servo drive control mode, which can also be set through the value of Axis.out.byModesofOpreation (requiring the addition of the Object Dictionary 6060h in the process data).



◆ Conditions for using the function block:

1. The axis must meet these control conditions, for example, a virtual axis is not allowed to use this function block.

2. The synchronization period supported by each mode must be consistent (see 7.3.3 "Communication

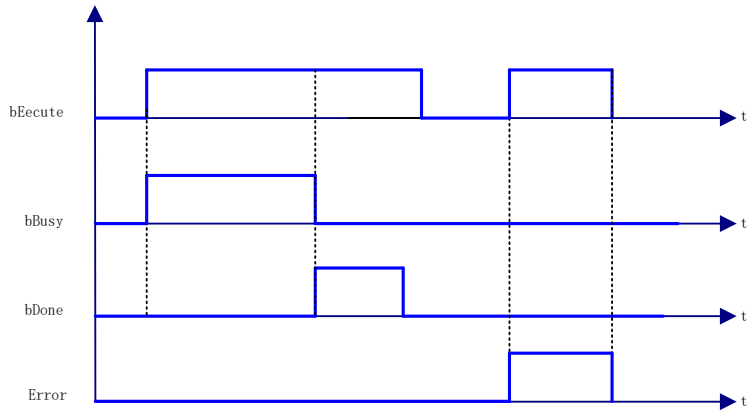
Period Supported by Each Mode" of the IS620N Series Servo Design and Maintenance User Guide).

3. The axis must be in a state other than Errorstop, Stopping, or Homing during instruction execution; otherwise, an error will occur.

- If the axis has not changed to the set control mode after the instruction has been executed for 1000 task periods, the instruction will report an error and bError will change from FALSE to TRUE.
- When the control mode of the axis changes from low level to high level (torque -> velocity, torque -> position, velocity -> position), the function block calculates the set value of the high level mode. For example, when there is a change from torque mode to position mode, the function block will compensate for the time lag between the actual and set values by superimposing an expected position distance on the actual position of the current axis (calculated based on the current velocity and the time offset during the task period).
- After the instruction is executed, when the actual control mode of the axis changes to the set control mode and the bDone signal is triggered, the axis will still run during the time between the trigger of the instruction and the bDone signal. In addition, the function block will calculate the appropriate set value according to the set control mode during this period. However, if the bDone signal is triggered and there is no other control instruction to continue to set the value for the axis, the axis will stop immediately and report an error. Therefore, it is necessary to use the rising edge of the bDone signal to trigger instructions such as MC_Halt, MC_MoveVelocity or MC_MoveAbsolute to smooth the control axis.

Note that when the control mode changes to torque control, a torque control instruction (such as MC_TorqueControl and SMC_SetTorque) is required to smooth the control axis.

4) Timing Diagram



5) Error Description

At the rising edge of bExecute:

The axis is invalid.

The axis status is invalid.

The axis does not meet the control mode.

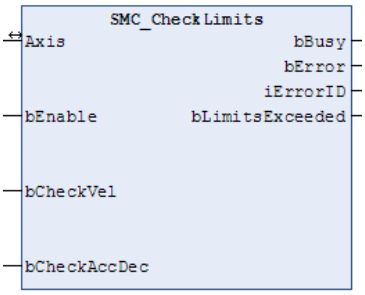
An error is output if there is an axis error.

Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC_CheckLimits

This instruction checks whether the current drive setpoint exceeds the maximum value configured by the controller.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_ CheckLimits	Axis limit check instruction		SMC_CheckLimits0(Axis:= , bEnable:= , bCheckVel:= , bCheckAccDec:= , bBusy=> , bError=> , iErrorID=> , bLimitsExceeded=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bEnable	Executed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when a check is in progress
bCheckVel	Velocity check	BOOL	TRUE, FALSE	FALSE	A velocity check will be performed if set to TRUE. No velocity check will be performed if set to FALSE.
bCheckAccDec	Acceleration/ deceleration check	BOOL	TRUE, FALSE	FALSE	An acceleration/deceleration check will be performed if set to TRUE. No acceleration/ deceleration check will be performed if set to FALSE.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	An axis check will be performed if set to TRUE. No axis check will be performed if set to FALSE.
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
iErrorID	Error code	SMC_ERROR	-	-	See SMC_Error
bLimitsExceeded	Limit output check	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current set velocity or acceleration/deceleration rate exceeds Axis.fSWMaxVelocity, Axis.fSWMaxAcceleration, or Axis.fSWMaxDeceleration.

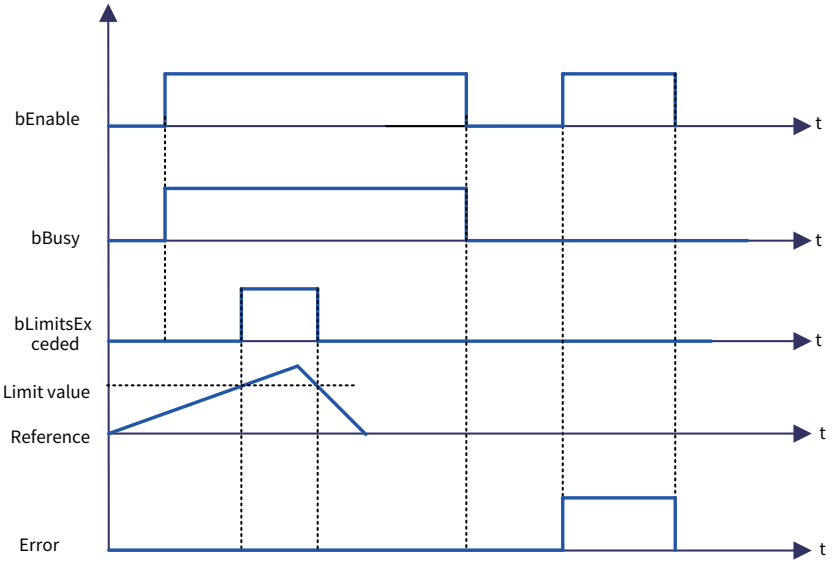
3) Function Description

◆ When bEnable is TRUE and bBusy outputs TRUE, an axis velocity check or acceleration check will be performed.

◆ If the set velocity or acceleration/deceleration rate of the current axis exceeds the set value of Axis. fSWMaxVelocity, Axis.fSWMaxAcceleration, or Axis.fSWMaxDeceleration, the bLimitsExceeded signal outputs TRUE.

Note: This instruction only checks that the current instruction velocity or acceleration/deceleration exceeds the set limit, and it does not stop the axis.

4) Timing Diagram



5) Error Description

At the rising edge of bExecute:

An error is output if there is an axis error.

An error is output if the axis input is invalid.

Note: For details on the error code, see “ “Appendix C Error Codes”.

SMC_GetMaxSetAccDec

This instruction reads the maximum acceleration/deceleration rate of the axis.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_GetMaxSetAccDec	Maximum acceleration/deceleration rate of the axis	SMC_GetMaxSetAccDec Axis → bValid bBusy fMaxAcceleration bEnable → dwTimeAtMax dwTimeStamp	SMC_GetMaxSetAccDec_0(Axis:= , bEnable:= , dwTimeStamp:= , bValid=> , bBusy=> , fMaxAcceleration=> , dwTimeAtMax=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bEnable	Executed	BOOL	TRUE, FALSE	FALSE	Execute the read operation if set to TRUE

dwTimeStamp		Dword	-	-	Optional timestamp input, which can be used to find out what happens at the maximum value.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bValid	Enable	BOOL	TRUE, FALSE	FALSE	Set to TRUE when instruction execution is valid
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
fMaxAcceleration	Maximum acceleration/ deceleration value	LREAL	-	0	Maximum acceleration/deceleration value (positive means acceleration, negative means deceleration, and the maximum absolute acceleration/deceleration value is the final value)
dwTimeAtMax	Timestamp corresponding to the maximum value	Dword	-	0	dwTimeStamp value corresponding to the maximum acceleration/deceleration rate (For example, when the acceleration continues to increase, the value follows dwTimeStamp, and the fMaxAcceleration value is updated. Once the acceleration rate reaches the maximum value, the maximum value of fMaxAcceleration is recorded, and dwTimeStamp corresponding to the maximum value is also recorded.)

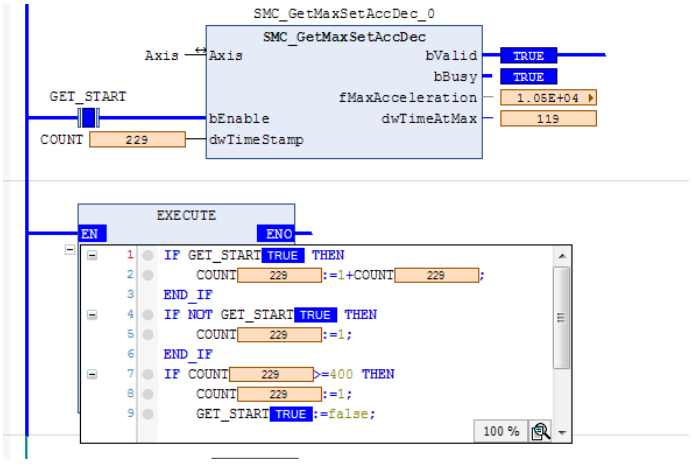
3) Function Description

◆ When bEnable is TRUE, no error occurs, and bValid outputs TRUE, a maximum acceleration/deceleration check will be performed for the axis.

◆ When the absolute value of acceleration/deceleration is larger than the previously recorded value, fMaxAcceleration and dwTimeAtMax will be updated.

◆ The value of dwTimeAtMax is the value of dwTimeStamp when the maximum acceleration/deceleration rate occurs. Therefore, dwTimeStamp must be set to a variable value, such as a cumulative value changing with the task period or a fixed time period. (See the sample program.)

4) Sample Program

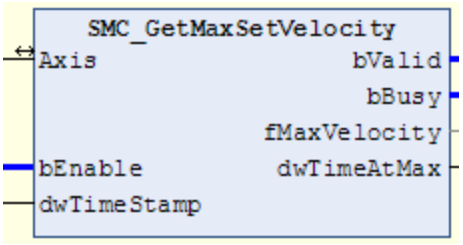


SMC_GetMaxSetVelocity

This instruction reads the maximum velocity of the axis.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression

SMC_GetMax SetVelocity	Maximum acceleration/ deceleration rate of the axis		SMC_GetMaxSetVelocity_0(Axis:= , bEnable:= , dwTimeStamp:= , bValid=> , bBusy=> , fMaxVelocity=> , dwTimeAtMax=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_ REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bEnable	Executed	BOOL	TRUE, FALSE	FALSE	Execute the read operation if set to TRUE
dwTimeStamp	-	Dword	-	-	Optional timestamp input, which can be used to find out what happens at the maximum value.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bValid	Enable	BOOL	TRUE, FALSE	FALSE	Set to TRUE when instruction execution is valid
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
fMaxVelocity	Maximum acceleration value	LREAL	-	0	Maximum velocity value (positive means forward, negative means backward, and the maximum absolute value is the final value)
dwTimeAtMax	Timestamp corresponding to the maximum value	Dword	-	0	dwTimeStamp value corresponding to the maximum velocity (For example, when the velocity continues to increase, the value follows dwTimeStamp, and the fMaxVelocity value is updated. Once the velocity reaches the maximum value, the maximum value of fMaxVelocity is recorded, and dwTimeStamp corresponding to the maximum value is also recorded.)

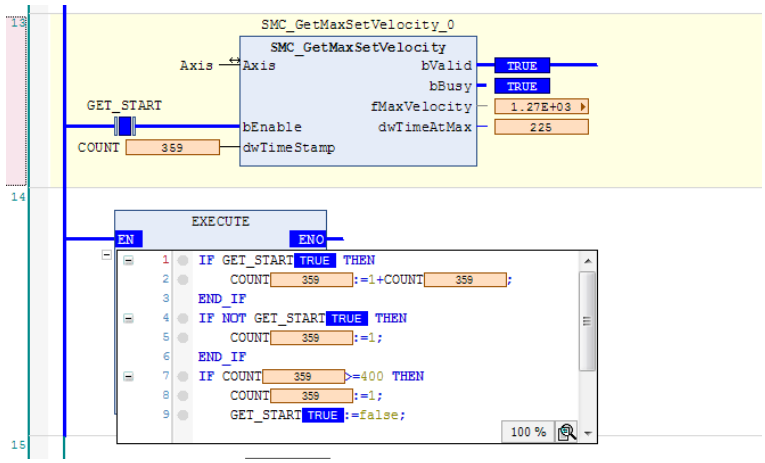
3) Function Description

◆ When bEnable is TRUE, no error occurs, and bValid outputs TRUE, a maximum acceleration/deceleration check will be performed for the axis.

◆ When the absolute value of velocity is larger than the previously recorded value, fMaxVelocity and dwTimeAtMax will be updated.

◆ The value of dwTimeAtMax is the value of dwTimeStamp when the maximum velocity occurs. Therefore, dwTimeStamp must be set to a variable value. For example, set to a count value changing with the task period or a fixed time period. (See the sample program.)

4) Sample Program



MC_GetTrackingError

This instruction measures the current or maximum lag error (difference between the instruction position and actual axis position) for dead-zone time compensation.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_ GeTrackingError	Instruction for reading the axis lag deviation	SMC_GetTrackingError Axis → bValid bBusy fActTrackingError bEnable → fMaxTrackingError byDeadTimeCycles → dwTimeAtMax dwTimeStamp	SMC_GetTrackingError(Axis:= , bEnable:= , byDeadTimeCycles:= , dwTimeStamp:= , bValid=> , bBusy=> , fActTrackingError=> , fMaxTrackingError=> , dwTimeAtMax=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_ REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bEnable	Executed	BOOL	TRUE, FALSE	FALSE	Execute the read operation if set to TRUE
byDeadTimeCycles	-	Byte	-	2	Number of dead-zone periods, for which a lag check is performed after bEnable trigger lags for certain dwTimeStamp values
dwTimeStamp	-	Dword	-	-	Optional timestamp input, which can be used to find out what happens at the maximum value.

◆ Output Variable

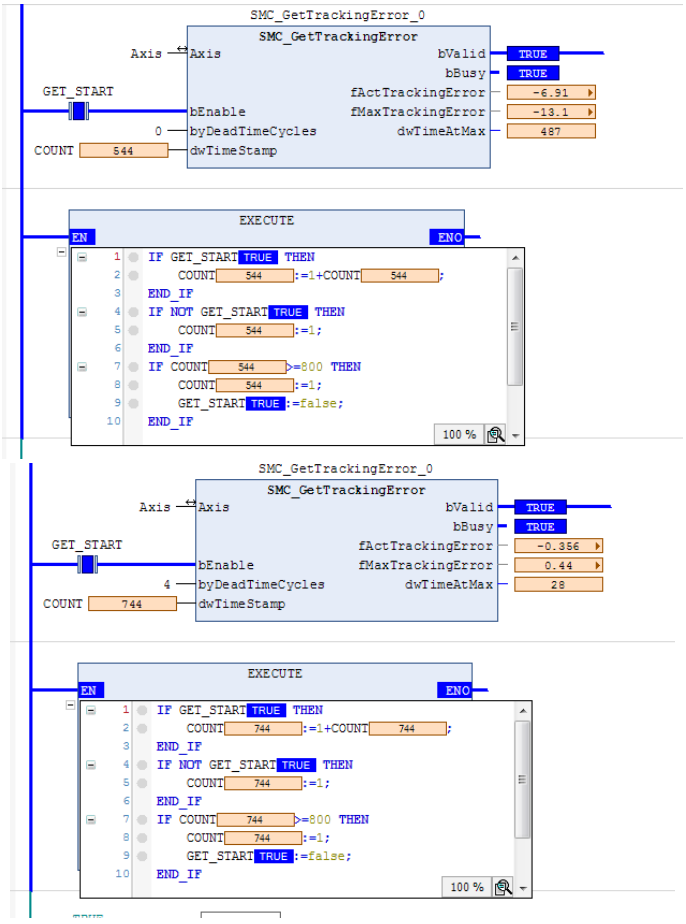
Output Variable	Name	Data Type	Value Range	Initial Value	Description
bValid	Enable	BOOL	TRUE, FALSE	FALSE	Set to TRUE when instruction execution is valid
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
fActTrackingError	Current lag	LREAL	-	0	Current deviation detection related to the value of byDeaTimeCycles

fMaxTrackingError	Maximum lag	LREAL	-	0	Current deviation value (deviation of the instruction position from the feedback position)
dwTimeAtMax	Timestamp corresponding to the maximum value	Dword	-	0	Maximum deviation value (positive means lag, negative means lead, and the maximum absolute value is the final value) Note: This value is affected by the value of byDeaTimeCycles.

3) Function Description

- When bEnable is TRUE and bValid outputs TRUE, axis lag deviation detection will be performed.
- When the absolute value of the deviation is larger than the previously recorded value, fMaxTrackingError and dwTimeAtMax will be updated.
- The value of dwTimeAtMax is the value of dwTimeStamp when the maximum deviation occurs. Therefore, dwTimeStamp must be set to a variable value. For example, set to a count value changing with the task period or a fixed time period. (See the sample program.)

4) Sample Program

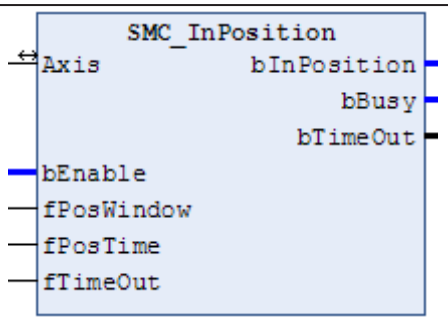


SMC_InPosition

This instruction monitors the deviation of the set position value of the current axis from the actual value and determines whether the axis is within the required deviation range based on the set deviation window.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression

SMC_InPosition	Axis deviation monitoring instruction		SMC_InPosition0(Axis:=Axis , bEnable:= , fPosWindow:= , fPosTime:= , fTimeOut:= , bInPosition=> , bBusy=> , bTimeOut=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bEnable	Executed	BOOL	TRUE, FALSE	FALSE	Execute the read operation if set to TRUE
fPosWindow	Deviation window	LREAL	-	0	Set the window for deviation monitoring. If fPosWindow > Distance (deviation between the instruction position and the feedback position), then output bInPosition as TRUE according to fPosTime.
fPosTime	Trigger time	LREAL	-	0	Deviation time within the window, used to trigger bInPosition Unit: s (seconds)
fPosTiOut	Timeout period	LREAL	-	0	Deviation timeout Unit: s (seconds)

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bInPosition	Normal deviation	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the deviation is within the set window
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
bTimeOut	Timeout	LREAL	TRUE, FALSE	FALSE	Current deviation detection related to the value of byDeaTimeCycles

3) Function Description

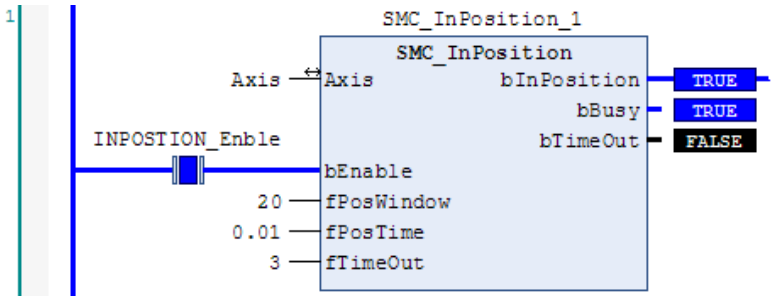
◆ When bEnable is TRUE, once the detected deviation is smaller than the set window fPosWindow for fPosTime, bInPosition changes to TRUE. Once the detected deviation is larger than the set window, bInPosition immediately changes to FALSE. Note: fPosTime must be set reasonably; otherwise it will cause bTimeOut trigger (for example, for a cam with a period of 2 seconds, the time for determining continuous deviation not exceeding the set window time is 1.5 seconds. If fPosTime is larger than 1.5 seconds, bInPosition will not be triggered).

◆ When bEnable is TRUE, the bBusy outputs TRUE.

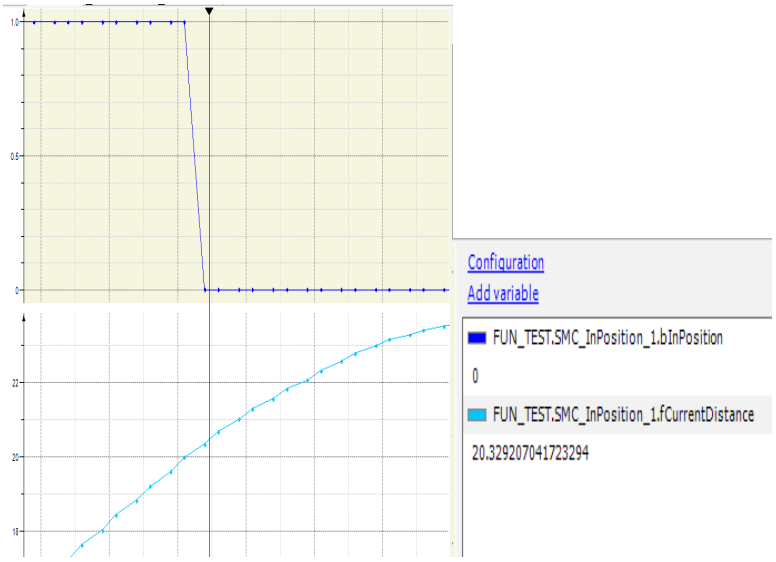
◆ The deviation value can monitor fCurrentDistance in the SMC_InPosition structure.

When bEnable is TRUE, if bInPosition does not change to TRUE even after the set time of fPosTime, then bTimeOut changes to TRUE.

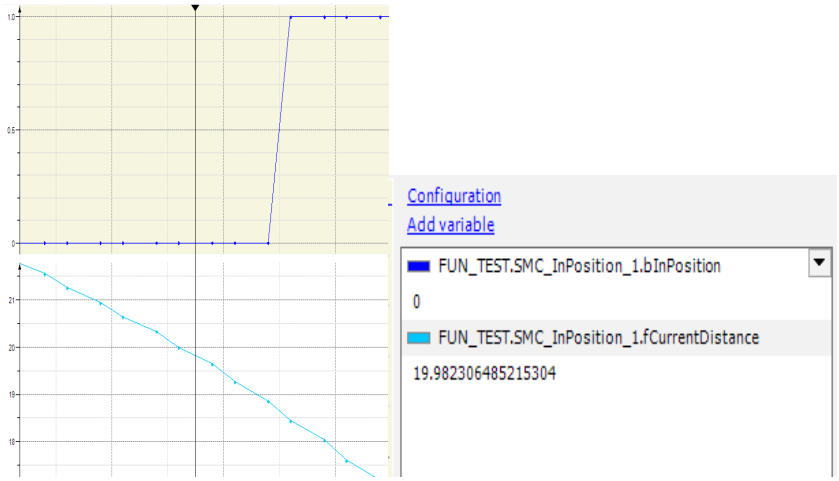
◆ Timing Diagram Sample Program



◆ Sample Program

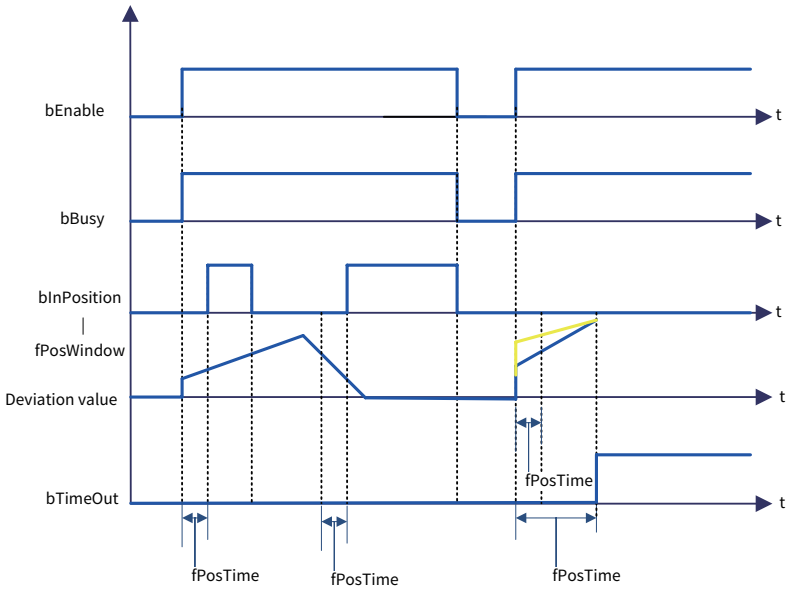


bInPosition immediately changes from TRUE to FALSE after the set window is exceeded.



After 4 task periods (2.5 ms) within the set window, bInPosition becomes TRUE, which matches the program setting of 0.01s.

4) Timing Diagram



SMC_ReadSetPosition

This instruction reads the instruction position of the axis (converted user unit).

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_ ReadSetPosition	Instruction for reading the instruction position	SMC_ReadSetPosition Axis Enable Valid Busy Error ErrorID Position	SMC_ReadSetPosition0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Position=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_ REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Executed	BOOL	TRUE, FALSE	FALSE	Execute the read operation if set to TRUE

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Enable	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the read value is valid
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	-	-	See SMC_Error

Position	Instruction position	LREAL	-	0	Instruction position of current task period

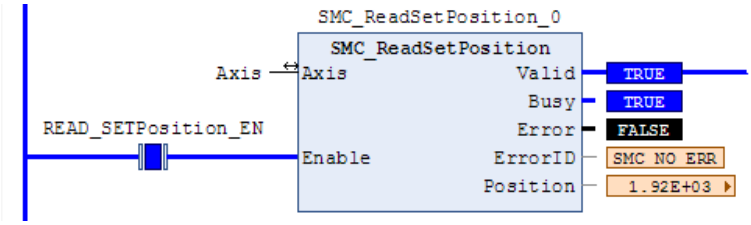
3) Function Description

When Enable is TRUE, Valid is output if no error occurs, and Busy outputs TURE.

The output of Position is the value of Axis.fSetPosition.

When Enable becomes FALSE, Valid and Busy output FALSE. Position retains the value at the moment before the value changes to FALSE.

4) Timing Diagram Sample Program



5) Error Description

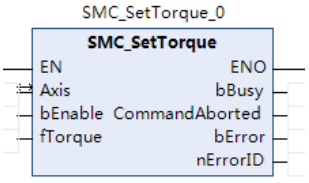
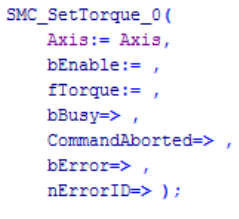
At the rising edge of bExecute: An error is output if there is an axis error. An error is output if the axis input is invalid.

Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC_SetTorque

This instruction sets the axis torque (valid in torque control mode).

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_SetTorque	Torque setting instruction		


```
SMC_SetTorque_0(
    Axis:= Axis,
    bEnable:= ,
    fTorque:= ,
    bBusy=> ,
    CommandAborted=> ,
    bError=> ,
    nErrorID=> );
```

2) Related Variables

Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bEnable	Executed	BOOL	TRUE, FALSE	FALSE	Set axis torque at the rising edge
fTorque	Torque setting	LREAL	-	0	Target torque value

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
CommandAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted.
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	-	-	See SMC_Error

3) Function Description

◆ If there is no error at the rising edge of bEnable, bBusy outputs TURE.

◆ This instruction only sets the torque value for the axis and does not perform torque control. The axis control mode is valid in the torque control mode.

◆ The torque setting instruction can only run in synchronous torque mode. Before enabling this instruction, switch the control mode to synchronous torque mode by using the SMC_SetControllermode system.

◆ The actual torque of the drive is limited by the maximum positive and negative torque set in the configuration.

◆ To stop the execution of this instruction, use the MC_Stop (forced stop) or MC_ImmediateStop (emergency stop) instruction. After the instruction is stopped, the drive switches to the synchronous position mode.

4) Error Description

At the rising edge of bExecute:

An error is output if there is an axis error. An error is output if the axis input is invalid.

An error is output when there is an axis control mode error, and the error code is SMC_ST_WRONG_CONTROLLER_MODE.

Note: For details on the error code, see “ “Appendix C Error Codes” .

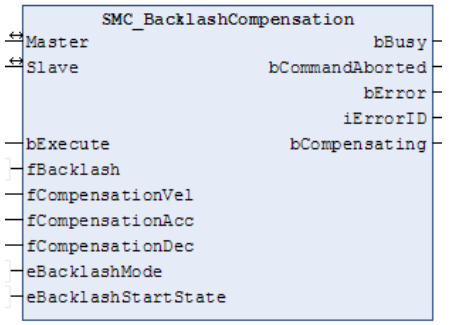
SMC_BacklashCompensation

This instruction compensates for the gap between master and slave axes. For example, when the virtual axis is the master axis and the slave axis is the synchronous mirror of the virtual axis in the belt transmission, there is a backlash between the position of the slave axis and the master axis due to external reasons. In this case, this instruction can be used to compensate for this backlash.

The function of this instruction is similar to that of the phase shift instruction (MC_Phasing), where the phase depends on the running direction of the master axis.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression

SMC_ BacklashCom pensation	Backlash compensation instruction		SMC_BacklashCompensation0(Master:= , Slave:= , bExecute:= , fBacklash:= , fCompensationVel:= , fCompensationAcc:= , fCompensationDec:= , eBacklashMode:= , eBacklashStartState:= , bBusy=> , bCommandAborted=> , bError=> , iErrorID=> , bCompensating=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Master	Master axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_ REF_SM3
Slave	Slave axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_ REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bExecute	Executed	BOOL	TRUE, FALSE	FALSE	Set the offset at the rising edge
fBacklash		LREAL	-	0	Compensate for the backlash
fCompensationVel		LREAL	-	0	Velocity at compensation
fCompensationAcc		LREAL	-	0	Acceleration rate at compensation
fCompensationDec		LREAL	-	0	Deceleration rate at compensation
eBacklashMode		SMC_ BACKLASH_ MODE	-	SMC_BL_ AUTO	Compensation mode: SMC_BL_AUTO: The running direction of the master axis determines the compensation direction. SMC_BL_POSITIVE: Positive compensation, independent of the running direction of the master axis SMC_BL_NEGATIVE: Negative compensation, independent of the running direction of the master axis SMC_BL_OFF: No compensation
eBacklashStartState		SMC_ BACKLASH_ STARTSTATE	-	SMC_BL_ START_ NEGATIVE	Describe the operating state of the axis during instruction execution. SMC_BL_START_NEGATIVE: The slave axis moves under negative traction. No compensation is required for motion in negative direction. Once the motion direction changes to positive, compensation must be established as twice the value of fBacklash. SMC_BL_START_POSITIVE: The slave axis moves under positive traction. No compensation is required for motion in positive motion. Once the motion direction changes to negative, compensation must be established as twice the value of fBacklash. SMC_BL_START_NONE: Motion in positive or negative direction will generate distance compensation equaling the value of fBacklash.

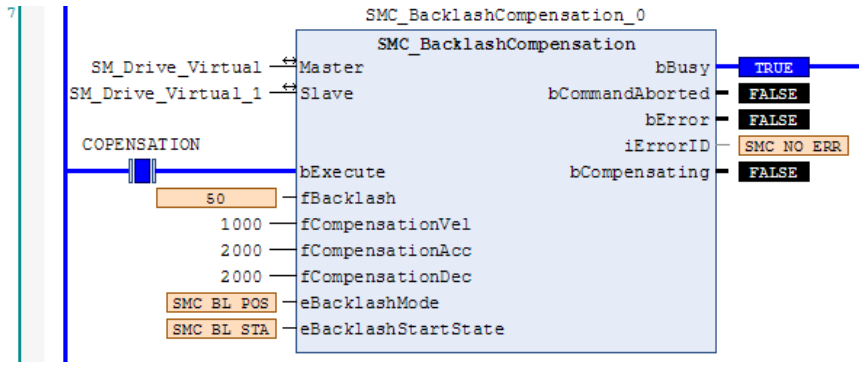
◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
bCommandAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is interrupted by other control instructions
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
iErrorID	Error code	SMC_ERROR	-	-	See SMC_Error
bCompsating	Compensation in progress	BOOL	TRUE, FALSE	FALSE	

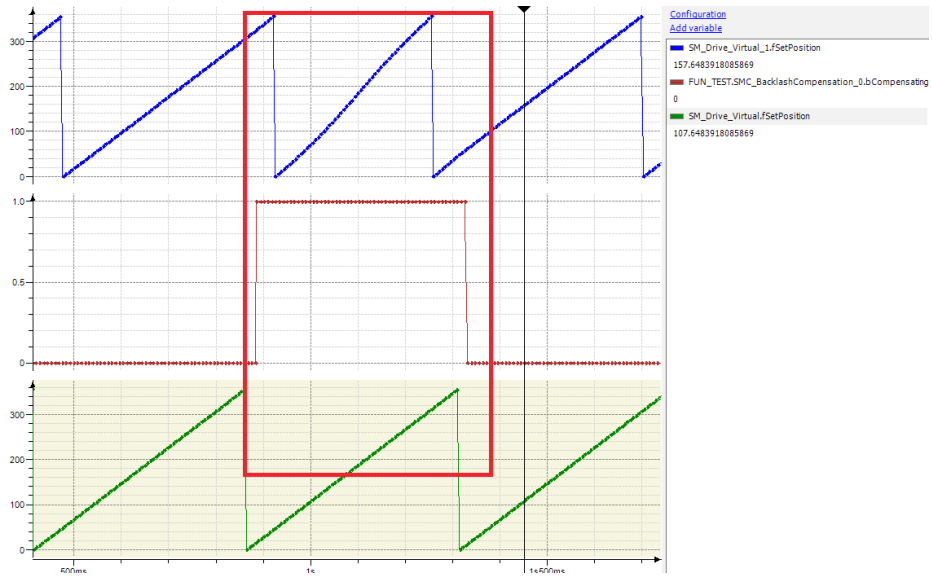
3) Function Description

- If there is no error at the rising edge of bExecute, bBusy and bCompsating output TRUE. After compensation is completed, bCompsating outputs FALSE.
- Working mode: eBacklashMode - compensation direction is "Positive", eBacklashStartState is "Positive", and fBacklash is a positive value. Before the bBusy signal arrives, the master and slave axes should be in the same position if possible; otherwise, the slave axis will be synchronized with the master axis phase after the bExecute rising edge signal arrives. If the bBusy signal is already available, refreshing the bExecute rising edge should observe:

4) Timing Diagram Sample Program



- Sample Program



5) Error Description

At the rising edge of bExecute:

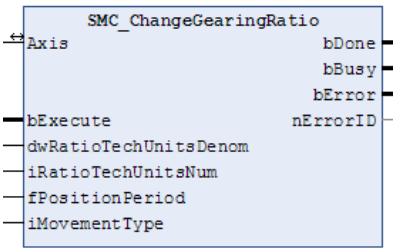
An error is output if there is an axis error. An error is output if the axis input is invalid.

Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC_ChangeGearingRatio

This instruction changes the electronic gearing ratio (ratio of pulse to user unit) and drive type set by the user. Note: After this function block is executed, the axis must be restarted by SMC3_ReinitDrive to ensure that the setup variables can be initialized correctly.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_ ChangeGearingRatio	Instruction for changing gear ratio		SMC_ChangeGearingRatio0(Axis:= , bExecute:= , dwRatioTechUnitsDenom:= , iRatioTechUnitsNum:= , fPositionPeriod:= , iMovementType:= , bDone=> , bBusy=> , bError=> , nErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_ SM3. The gear ratio of this axis will be changed

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bExecute	Executed	BOOL	TRUE, FALSE	FALSE	Execute the FB at the rising edge
dwRatioTechUnitsDenom	-	DWORD	-	0	Pulse unit converted to application units (eg: mm)
iRatioTechUnitsNum	-	DINT	-	0	dwRatioTechUnitsDenom value corresponding to the desired application unit
fPositionPeriod	-	LREAL	-	-	Position cycle period (modulus value), valid only for rotary motors
iMovementType	-	INT	-	-	0: Modulo axis; 1: Finite axis

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bDone	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when execution setup is completed
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
nErrorID	Error code	SMC_ERROR	-	-	See SMC_Error

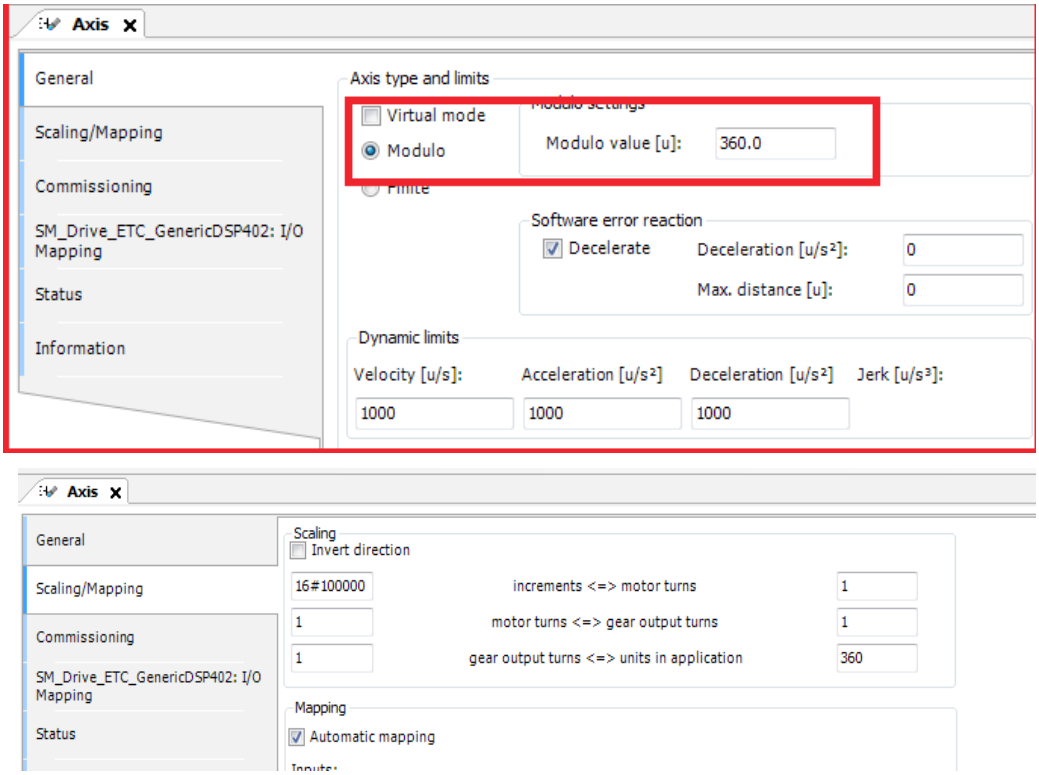
3) Function Description

◆ At the rising edge of bExecute, if there is no error, bBusy outputs TURE. At completion, bDone outputs TRUE, and bBusy outputs FALSE.

For example, for a 20-bit encoder servo motor with a 10:1 reduction ratio, if the lead screw is driven

(10 mm pitch), the motor needs to rotate 10 turns and the screw moves a distance of 10 mm. Set dwRatioTechUnitsDenom to 1048576*10 and iRatioTechUnitsNum to 10.

◆ The function block is used to dynamically modify the highlighted part of the program shown below:



4) Error Description

At the rising edge of bExecute:

An error is output if there is an axis error.

An error is output when the input value is invalid, with the error code SMC_CGR_ZERO_VALUES.

An error is output when the axis is in instruction control, with error code SMC_CGR_DRIVE_POWERED.

An error is output when the input modulus value is invalid (eg: < 0), with error code SMC_CGR_INVALID_POSPERIOD.

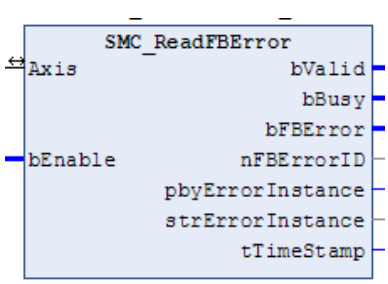
Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC_ReadFBError

This instruction reads the function block error.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression

SMC_ ReadFBError	Instruction for reading function block errors		SMC_ReadFBError(Axis:= , bEnable:= , bValid=> , bBusy=> , bFBError=> , nFBErrorID=> , pbyErrorInstance=> , strErrorInstance=> , tTimeStamp=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_ REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bEnable	Executed	BOOL	TRUE, FALSE	FALSE	Execute the read operation if set to TRUE

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bValid	Enable	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the read value is valid
bBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
bFBError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an FB error occurs
nFBErrorID	Error code	SMC_ERROR	-	-	See SMC_Error
pbyErrorInstance	-	-	-	-	Function block error at output point
strErrorInstance	-	-	-	-	Point to the error function block (program, subprogram, function block)
tTimeStamp	-	TIME	-	-	Timestamp when the error occurred

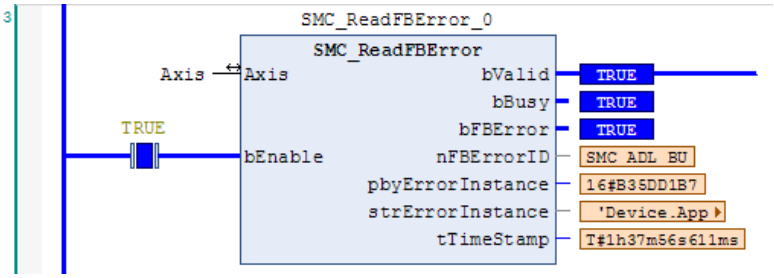
3) Function Description

◆ When Enable is TRUE, Valid is output if no error occurs, and Busy outputs TURE.

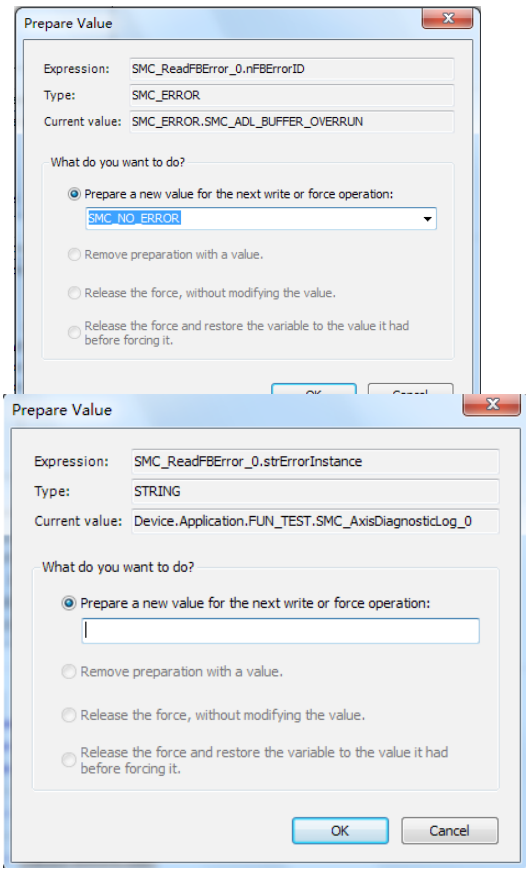
◆ If there is a function block alarm, bFBError outputs TRUE.

◆ When Enable becomes FALSE, Valid and Busy output FALSE.

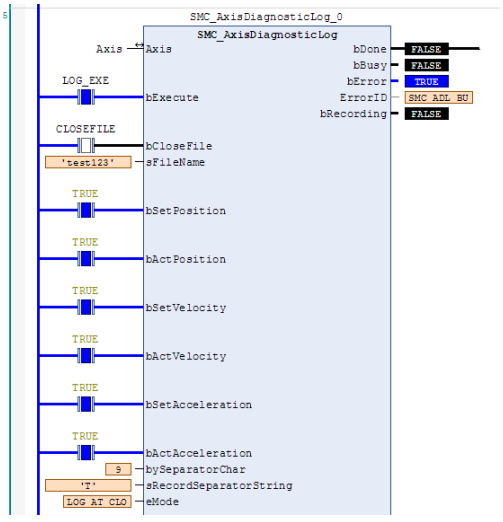
4) Timing Diagram Sample Program



◆ Sample Program



◆ Error ID



Function block where the error occurred

5) Error Description

At the rising edge of bExecute:

An error is output if there is an axis error.

An error is output if the axis input is invalid.

Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC_ClearFBError

This instruction clears the FB error.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_ ClearFBError	Instruction for clearing the FB error		TEST:=SMC_ ClearFBError(pDrive:=ADR(Axis));

2) Related Variables

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
pDrive	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

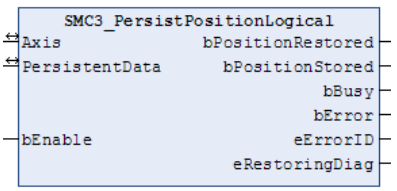
◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
SMC_ClearFBError	Error clearing	BOOL	TRUE, FALSE	FALSE	Clear the error f set to TRUE

SMC3_PersistPositionLogical

This instruction keeps recording the position of the logical axis (right-click at the real or virtual axis and click "Add Device" to select the logical axis to be added). After the controller is powered off and restarted, the recorded value of the position before power-off will be restored.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC3_ PersistPosition Logical	Instruction for keeping the axis position		SMC3_PersistPositionLogical0(Axis:= , PersistentData:= , bEnable:= , bPositionRestored=> , bPositionStored=> , bBusy=> , bError=> , eErrorID=> , eRestoringDiag=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_LOGICAL_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
PersistentData	Retentive data	SMC3_ PersistPositionLogical_ Data	-	-	Power-down retentive data structure for storing position information

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description

bEnable	Executed	BOOL	TRUE, FALSE	FALSE	The function block is executed if set to TRUE and not executed if set to FALSE. To restore the last stored position during initialization, this value must be set to TRUE from application startup.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bPosition Restored	Position restoring	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the position is restored upon axis restart
bPosition Stored	Position saving	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the position is stored after an FB call
bBusy	FB execution in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when FB execution is not completed.
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
eErrorID	Error code	SMC_ERROR	-	SMC_NO_ERROR	Output an error code when an error occurs
eRestoring-Diag	Restoration diagnostics	SMC3_ PersistPositionDiag	-	SMC3_ PersistPositionDiag. SMC3_PPD_ RESTORING_OK	Diagnostic information in position restoration SMC3_PPD_RESTORING_OK: Position successfully restored; SMC3_PPD_AXIS_PROP_CHANGED: Axis parameters have been changed and the position could not be restored; SMC3_PPD_DATA_STORED_DURING_ WRITING: The function block copies data from the axis parameter data structure instead of from PersistentData. Possible cause: Non-synchronized retentive variable, controller crash

3) Function Description

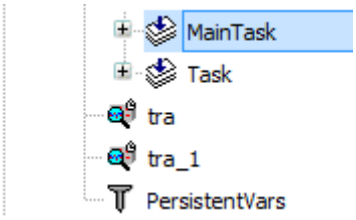
◆ If bEnable is TRUE upon PLC restart, then bPositionRestroed outputs TRUE.

Virtual and real axes are not supported.

◆ To restore the "position" before power-off upon PLC restart, use this function block and configure SMC3_PersistPositionLogical_Data as a retentive variable.

◆ Usage (when the real axis encoder is a multi-turn absolute encoder):

SMC3_PersistPositionLogical_Data declared in PersistentVars




```
VAR_GLOBAL PERSISTENT RETAIN
    persistentData3: SMC3_PersistPositionLogical_Data;
END_VAR
```

Called in the PLC main task (EthCat task)

◆ Declaration section:

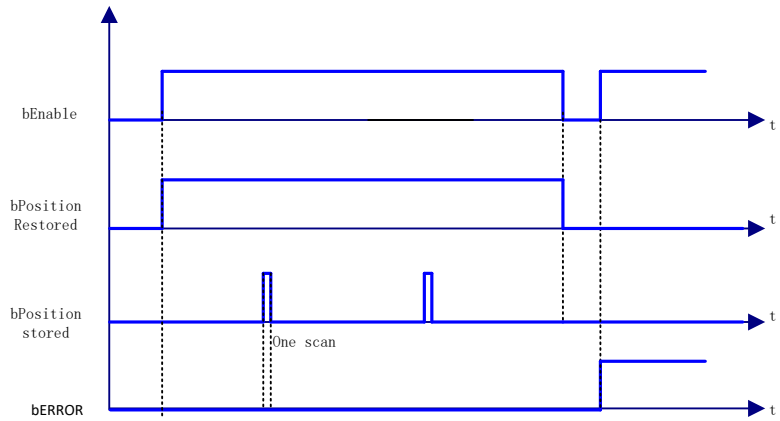
VAR

```
    SMC3_PersistPosition_3:SMC3_PersistPositionLogical;
END_VAR
```

- Program section:

```
2  SMC3_PersistPosition_1(Axis:=X_Axis , PersistentData:=persistentData1 ,bEnable:=TRUE );
```

- Timing Diagram



4) Error Description

If the input axis is a virtual or real one, an error will be output. An axis error will result in an error output.

Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC_Homing

This axis homing instruction is different from MC_Home. For MC_Home, the homing method is set at the axis configuration. For this instruction, the homing method is controlled by the controller.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_ Homing	Axis homing instruction	SMC_Homing Axis bExecute fHomePosition fVelocitySlow fVelocityFast fAcceleration fDeceleration fJerk nDirection bReferenceSwitch fSignalDelay nHomingMode bReturnToZero bIndexOccured fIndexPosition bIgnoreHWLimit bDone bBusy bCommandAborted bError nErrorID bStartLatchingIndex	SMC_Homing0(Axis:= , bExecute:= , fHomePosition:= , fVelocitySlow:= , fVelocityFast:= , fAcceleration:= , fDeceleration:= , fJerk:= , nDirection:= , bReferenceSwitch:= , fSignalDelay:= , nHomingMode:= , bReturnToZero:= , bIndexOccured:= , fIndexPosition:= , bIgnoreHWLimit:= , bDone=> , bBusy=> , bCommandAborted=> , bError=> , nErrorID=> , bStartLatchingIndex=>);

2) Related Variables

- Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description

Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bExecute	Executed	BOOL	TRUE, FALSE	FALSE	The function block is executed if set to TRUE and not executed if set to FALSE.
fHomePosition	Home position setting	LREAL	-	0	Home position setting after homing, the unit is a user-calibrated one
fVelocitySlow	Low velocity	LREAL	-	0	Low velocity reference after the reference switch
fVelocityFast	Fast	LREAL	-	0	High velocity reference after the reference switch
fAcceleration	Acceleration	LREAL	-	0	Acceleration setting
fDeceleration	Deceleration	LREAL	-	0	Deceleration setting
fJerk	Acceleration derivative	LREAL	-	0	Jerk in [u/s3]
nDirection	Homing direction	MC_DIRECTION	-	Negative	Star direction of homing. See MC_DIRECTION
bReferenceSwitch	Reference switch	BOOL	TRUE, FALSE	FALSE	Reference switch connection TRUE: Trigger reference switch FALSE: Close reference switch
fSignalDelay	Delay	LREAL	-	0	Transfer time of the reference switch, to compensate for dead-zone time Unit: second
nHomingMode	Homing mode	SMC_HOMING_MODE	-	-	See SMC_HOMING_MODE.
bReturnTozero	Return to zero position	BOOL	TRUE, FALSE	FALSE	TRUE: The axis moves to the zero position after homing is complete (Note: If fHomePosition=10, the axis position becomes 10 after homing is complete. If bReturnTozero is TRUE, the axis moves for 10 units in negative direction to the zero position after homing is complete.)
bIndexOccured		BOOL	TRUE, FALSE	FALSE	TRUE: Flag pulse recording, which is valid when homing mode is FAST_BSLOW_I_S_STOP or FAST_SLOW_I_S_STOP
fIndexPosition		LREAL	-	0	Position recorded at the time of flag pulse
bIgnoreHWLimit	Ignore hardware limit	BOOL	TRUE, FALSE	FALSE	If its value is TRUE, set hardware limit switch to FALSE. If a physical switch is used as both a hardware limit switch and a reference switch, then the hardware control will be set to FALSE.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bDone		BOOL	TRUE, FALSE	FALSE	Set to TRUE when homing is complete
bBusy		BOOL	TRUE, FALSE	FALSE	Set to TRUE when function block is in effect
bCommandAborted		BOOL	TRUE, FALSE	FALSE	Set to TRUE when the function block is interrupted by another instruction
Error		BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID		SMC_ERROR	-	0	Error code, enumerated variable See SMC_Error for specific error code.
bStartLatchingIndex		BOOL	TRUE, FALSE	FALSE	Generated by bIndexOccured and fIndexPosition

◆ Homing Mode (SMC_HOMING_MODE)

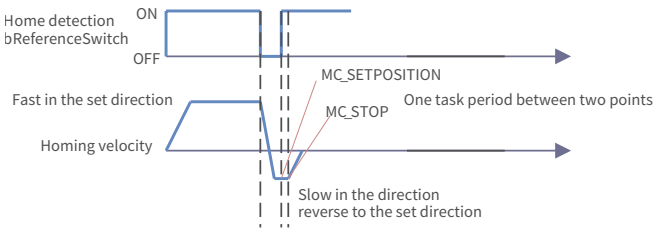
Enumeration Name	Type	Initial Value	Description
FAST_BSLOW_S_STOP	SMC_HOMING_MODE	0	The axis moves toward the home switch rapidly in the set direction. After touching the home switch, the axis leaves the home switch slowly in negative direction. After that, execute MC_setPosition to set the current position to the fHomePosition setpoint, and then execute MC_stop.
FAST_BSLOW_STOP_S	SMC_HOMING_MOD	1	The axis moves toward the home switch rapidly in the set direction. After touching the home switch, the axis leaves the home switch slowly in negative direction. After that, execute MC_stop to stop the axis, and then execute MC_setPosition to set the current position to the fHomePosition setpoint.
FAST_BSLOW_I_S_STOP	SMC_HOMING_MOD	2	The axis moves toward the home switch rapidly in the set direction. After touching the home switch, the axis leaves the home switch slowly in negative direction. When the bIndexOccured signal arrives, execute MC_setPosition first and then MC_stop.
FAST_SLOW_S_STOP	SMC_HOMING_MOD	4	The axis moves toward the home switch rapidly in the set direction. After touching the home switch, the axis leaves the home switch slowly. After that, execute MC_setPosition to set the current position to the fHomePosition setpoint and then execute MC_stop.
FAST_SLOW_STOP_S	SMC_HOMING_MOD	5	The axis moves toward the home switch rapidly in the set direction. After touching the home switch, the axis leaves the home switch slowly. After that, execute MC_stop, and then execute MC_setPosition to set the current position to the fHomePosition setpoint.
FAST_SLOW_I_S_STOP	SMC_HOMING_MOD	6	The axis moves toward the home switch rapidly in the set direction. After touching the home switch, the axis leaves the home switch slowly in negative direction. When the bIndexOccured signal arrives, execute MC_setPosition first and then MC_stop.

3) Function Description

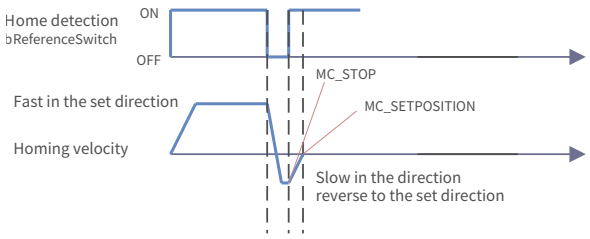
- After SMC_HOMING is started at the rising edge of bExecute, the axis starts moving at fVelocityFast and in the direction defined by nDirection until bReferenceSwitch changes to FALSE. The axis will then slowly stop and leave the reference switch at fVelocitySlow in negative direction. Homing is complete after bReferenceSwitch changes to TRUE.
- The state of bReferenceSwitch is ON->OFF->ON after the homing instruction is enabled. Homing is complete at the rising edge of OFF->ON. Set the reference position.
- Reference position = fHomePostion + ((fSignalDelay x 1000 + 1 DC clock period)/1000) x fVelocitySlow. It compensates for the set bReferenceSwitch sampling delay and one communication period displacement delay.
- If bReturnToZero = TRUE, bReferenceSwitch will, at the rising edge of state OFF->ON, set the reference position to: fHomePostion + ((fSignalDelay x 1000+1 DC clock period)/1000) x fVelocitySlow. Then, the axis moves to the zero position at fVelocityFast.

Note: After the Done signal, the axis position is set to fHomePosition. The timing of the setting is related to nHomingMode. (For details, see SMC_HOMING_MODE.) The following figures show different homing modes:

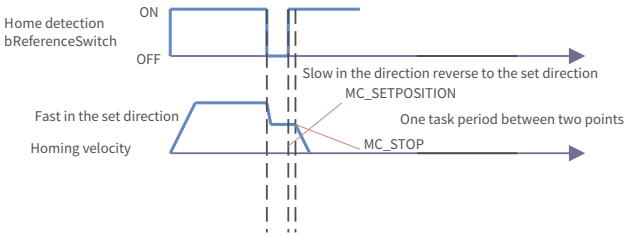
10) Homing mode "0"



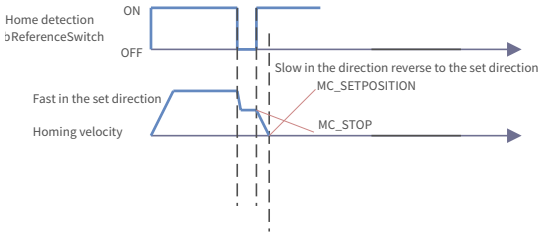
11) Homing mode "1"



12) Homing mode "4"

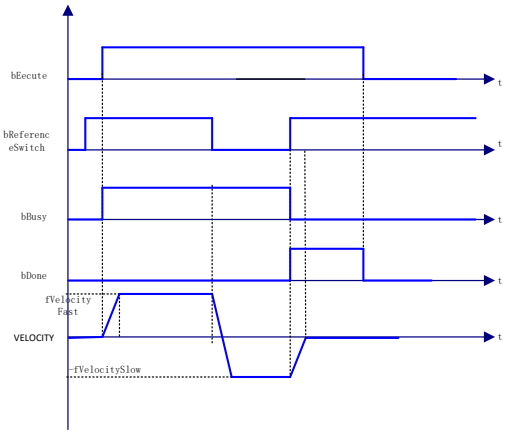


13) Homing mode "5"

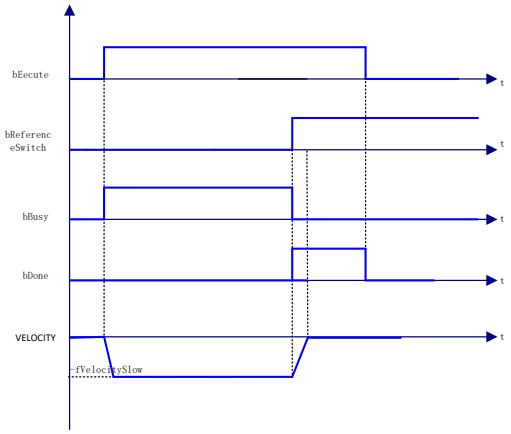


4) Timing Diagram

14) When bReferenceSwitch is TRUE during instruction execution



15) When bReferenceSwitch is FALSE during instruction execution



5) Error Description

There is an error in the input axis type.
There is an axis error.
The axis is disabled.
The velocity or acceleration is invalid.
Note: For details on the error code, see “ “Appendix C Error Codes” .

MC_TorqueControl

This instruction performs torque control by using the torque control mode of the servo drive.

1) Instruction Format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_ TorqueControl	Torque Control instruction	FB	MC_TorqueControl_0 MC_TorqueControl EN — ENO Axis — InTorque Execute — Busy Torque — CommandAborted TorqueRamp — Error Velocity — ErrorID	MC_TorqueControl_0(Axis:= Axis, Execute:= , Torque:= , TorqueRamp:= , Velocity:= , InTorque=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Start	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge

Torque	Target torque	LREAL	The positive torque cannot be larger than fMaxPositiveTorque of the axis structure. The absolute value of the negative torque cannot be larger than fMaxNegativeTorque of the axis structure.	0	Specify the target torque to be output to the servo drive in units of [0.1%], at the ratio "100.0%" of rated torque. Unit: [%/s] If 100 is entered, the target torque is the rated torque.
TorqueRamp	Torque slope	LREAL	Positive or 0	0	Specify the ratio for converting the current value to the target torque. The larger the value, the faster the target torque is reached [%/s]. The value 0 means the output is just the target torque.
Velocity	Velocity	LREAL	Positive or 0	0	Max. running velocity

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
InTorque	Target torque reached	BOOL	TRUE, FALSE	FALSE	TRUE: Target torque is reached Note: This flag is continuously refreshed during instruction execution.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Command Aborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ Specify the torque instruction value directly to control the output torque of the servo motor.

◆ The target torque is specified in units of [0.1%]. For the specified value, the first decimal place is valid, and other decimal places are discarded.

◆ The actual torque of the drive is limited by the maximum positive and negative torque set in the configuration.

◆ To stop the execution of this instruction, use the MC_Stop (forced stop) or MC_ImmediateStop (emergency stop) instruction. After the instruction is stopped, the drive switches to the synchronous position mode.

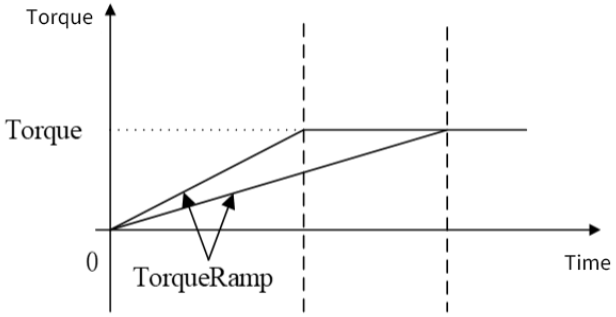
◆ This instruction achieves torque control by using the torque control mode of the servo drive. The axis is in the Continuous Motion status during instruction execution.

◆ Velocity is always a positive value. The direction depends on the torque and load.

◆ The torque instruction requires the drive to map the desired torque (0x6071) and the maximum profile velocity (0x607f); otherwise, an error will be reported.

◆ TorqueRamp specifies the slope from the currently specified instruction torque to the output target torque.

Examples are shown below:



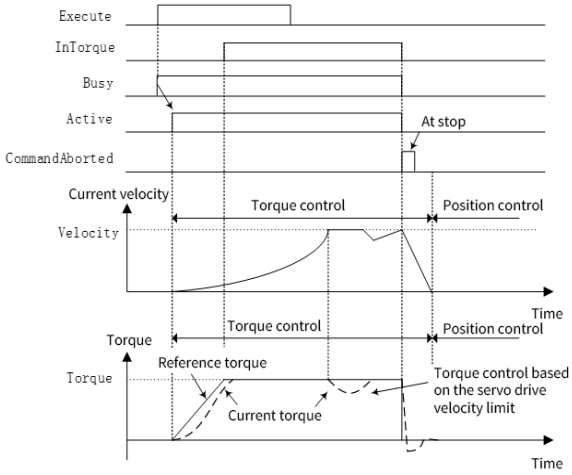
As shown above, the larger the TorqueRamp, the faster the target torque Torque is reached.

4) Precautions

The torque control instruction can only run in synchronous torque mode. Before enabling this instruction, switch the control mode to synchronous torque mode by using the SMC_SetControllermode system.

5) Timing Diagram

Start this instruction and then stop it.



MC_ImmediateStop

This instruction stops the axis according to the stopping mode specified by StopMode, regardless of the axis status.

1) Instruction Format

Instruction	Name	FB/ FC	LD Expression	ST Expression
MC_ImmediateStop	Immediate stop instruction	FB	MC_ImmediateStop_0 MC_ImmediateStop EN ENO Axis Done Execute Busy StopMode Error ErrorID	MC_ImmediateStop_0(Axis:= Axis, Execute:= , StopMode:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related Variables

Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Start	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
StopMode	Stop	MC_STOP_MODE	0/1	0	0: The instruction velocity is reduced from the current velocity to 0. 1: Immediately stop and switch the servo to OFF.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE upon instruction completion
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ This instruction can be executed when the axis is in any status. For example, this instruction can be used to stop the axis immediately even if it decelerates to stop due to an exception.

◆ When ErrorStop is TRUE, the MC_Stop instruction cannot be executed, but the MC_ImmediateStop instruction can be executed.

◆ After the instruction is executed, the motion is stopped immediately as specified by StopMode. The instruction in the action changes to the CommandAborted status.

◆ If the axis is in disabled state, execution completion is returned directly.

◆ If the axis is a non-control one, an error will be reported.

◆ If the servo is set to OFF, the axis can be enabled only after MC_Reset is executed if an emergency stop occurs.

◆ When MC_ImmediateStop is triggered in torque control mode, the control mode will change to position mode first, and then an emergency stop will be performed.

4) Precautions

◆ Axis in Stopping status

In the following conditions, the axis status is Stopping:

The axis is decelerated to stop by the MC_Stop instruction.

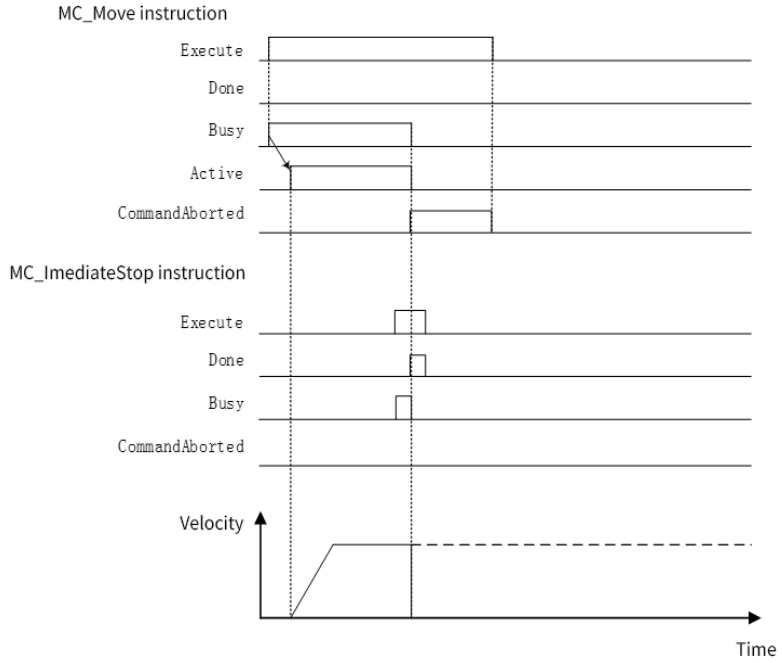
The MC_ResetFollowingError instruction is being executed.

When this instruction is started, Error of the above instruction in execution changes to TRUE.

5) Timing Diagram

The value of Busy changes to TRUE when Execute is started.

When processing of the immediate stop instruction is complete, Done changes to TREU.



MC_ResetFollowingError

This instruction resets the deviation between the current instruction position and the feedback position.

1) Instruction Format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_Reset FollowintError	Deviation reset instruction	FB	MC_ResetFollowingError_0 MC_ResetFollowingError EN ENO Axis Done Execute Busy CommandAborted Error ErrorID	MC_ResetFollowingError_0(Axis:= Axis, Execute:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Start	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge

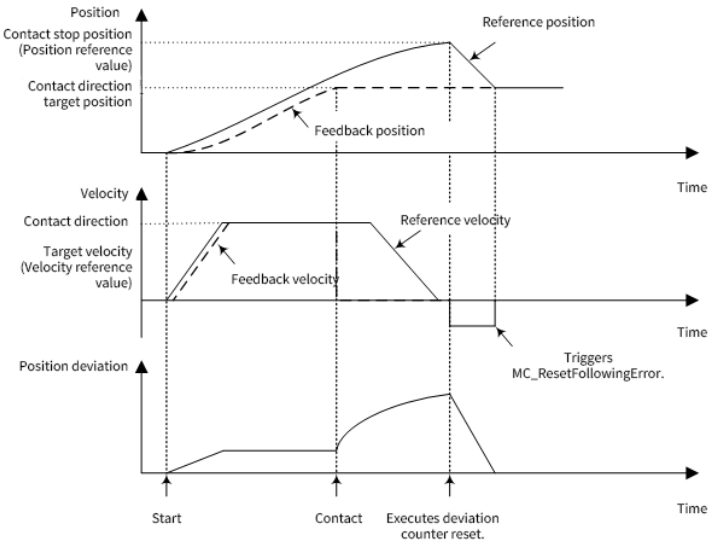
◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE upon instruction completion

Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
CommandAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the current instruction is aborted
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

- This instruction sets the deviation between the current instruction position and the feedback position of the MC function module to "0" in the cyclic synchronous position mode.
- When the rising edge of Execute is detected, the feedback position at that time is given to the instruction as the new target position.
- As shown in the figure below, when this instruction is started during a contact action in which a position deviation occurs, the position instruction is issued in negative direction so that the position deviation is "0". For an instruction with a position deviation, CommandAborted becomes TRUE and the instruction is aborted.



- When the position deviation is set to "0", a position instruction is issued by using the maximum velocity set in the axis parameter. The maximum acceleration and maximum deceleration are not applicable.
- The velocity at which the deviation reset instruction is executed is the largest among the current velocity, the velocity value set in the background axis dynamic parameter, and the velocity set in the default axis parameter.
- When the instruction is completed by reaching the new target position, Done changes to TRUE.
- If the axis is in an error state, the instruction will not be executed and an error will be returned.
- If the axis is a non-control one, an error will be reported.
- Notes on the triggering condition of the instruction and the axis status upon instruction triggering: The instruction cannot be called when the axis is in the Error, Homing, Down-enable or Stopping status. In addition, if the instruction is in the following error reset, triggering the instruction will also report an error. After the instruction is triggered, the axis is in Stopping status. After instruction execution is complete, the axis status changes to Standstill.
- Notes on repeated triggering of an instruction and multi-triggering: Repeated triggering of the instruction will report an error indicating that the axis is in error reset.

- Notes on the relationship between this instruction and the Stop instruction: The Stop instruction cannot be executed during a reset, and the reset is not allowed during the execution of the Stop instruction; otherwise, an error will be reported.
- Notes on starting the SetPosition instruction during the instruction execution: It is not allowed to start the SetPosition instruction during the reset; otherwise, there may be a position jump that causes an excessive position deviation.
- Notes on interrupting this instruction by another instruction: It is not allowed to interrupt this instruction during a reset, except for an emergency stop.
- Acceleration overrun is not checked during instruction execution.

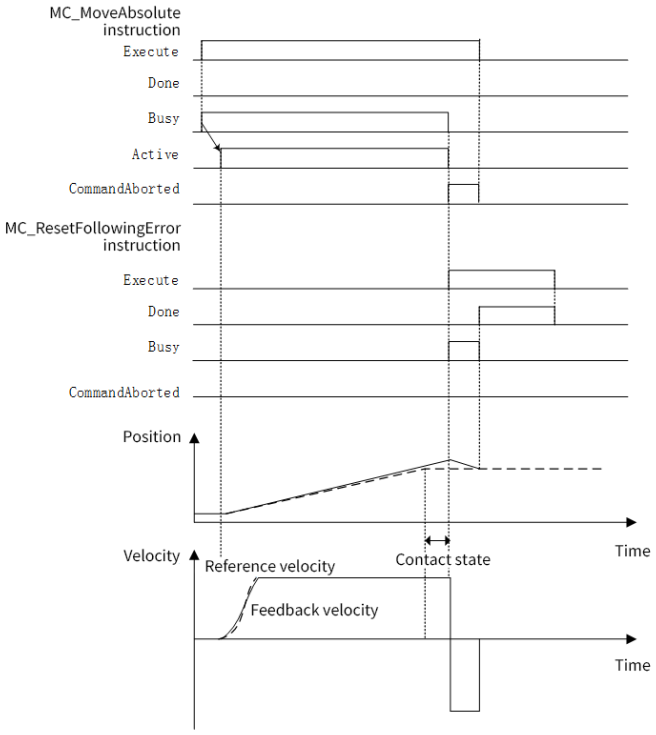
4) Precautions

* Please start this instruction at a low axis velocity. This instruction assigns the instruction value in the opposite direction to the previous instruction (contact direction). Therefore, if this instruction is started at a high axis velocity, it may cause a shock to the machine.

* This instruction issues a position instruction in the direction opposite to the motion in which the position deviation occurs. However, it is not applicable to "Motion during reversal" in the axis parameter.

5) Timing Diagram

The following figure shows the timing of starting this instruction in the contact status after the MC_MoveAbsolute instruction is started.

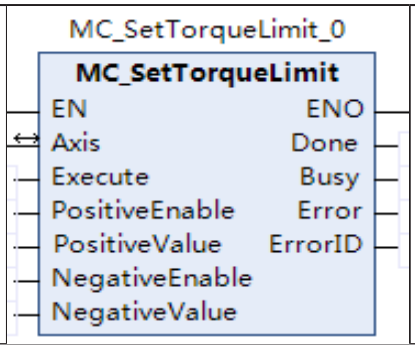
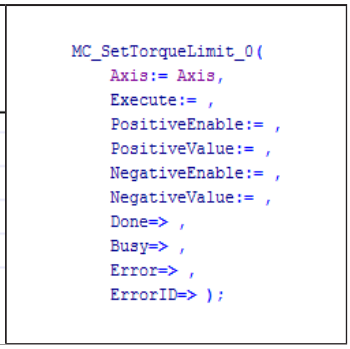


MC_SetTorqueLimit

This instruction limits the servo drive output through the torque limit function of the servo drive.

1) Instruction Format

Instruction	Name	FB/FC	LD Expression	ST Expression

MC_Set TorqueLimit	Torque limit instruction	FB		

2) Related Variables

Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Start	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
PositiveEnable	Valid in positive direction	BOOL	TRUE, FALSE	FALSE	TRUE: Enable the positive torque limit. FALSE: Disable the positive torque limit.
PositiveValue	Value of positive torque limit	LREAL	Positive number	300	Set the torque limit in positive direction in increments of 1%. (The actual servo increment is 0.1%. For details, see the servo guide.) If the value exceeds "Maximum positive torque limit" in the axis parameter, the positive torque will be the "Maximum positive torque limit". If "0" or "Negative" is specified, the motion will be taken based on the value "0".
NegativeEnable	Valid in negative direction	BOOL	TRUE, FALSE	FALSE	TRUE: Enable the negative torque limit. FALSE: Disable negative torque limit.
NegativeValue	Value of negative torque limit	LREAL	Positive number	300	Set the torque limit in negative direction in increments of 1%. (The actual servo increment is 0.1%. For details, see the servo guide.) If the value exceeds "Maximum negative torque limit" in the axis parameter, the negative torque will be the "Maximum negative torque limit". If "0" or "Negative" is specified, the motion will be taken based on the value "0".

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of axis instruction is complete
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

- If PositiveEnable is set to TRUE when Execute (TRUE) is triggered, the limit will be performed based on PositiveValue. If NegativeEnable is set to TRUE, the limit will be performed based on NegativeValue.
- If PositiveEnable is set to FALSE, set "Upper limit of positive torque" of the axis parameter in the servo drive. Similarly, if NegativeEnable is set to FALSE, set "Upper limit of negative torque" of the axis parameter in the servo drive.
- When Execute is set to FALSE in this instruction, set "Upper limit of positive torque" and "Upper limit of negative torque" in the servo drive and set Busy to FALSE.
- When the values of PositiveValue and NegativeValue are set to a number less than or equal to 0, the motion will be taken based on the value "0".
- The torque limit can be set in units of 1% relative to the motor torque. For the specified value, the first decimal place is valid.

4) Precautions

Currently, this instruction can only take effect when 0x60e0 and 0x60e1 are not configured. If PDO object dictionary 0x60e0 and 0x60e1 are configured, the written value will be refreshed by the default value (0).

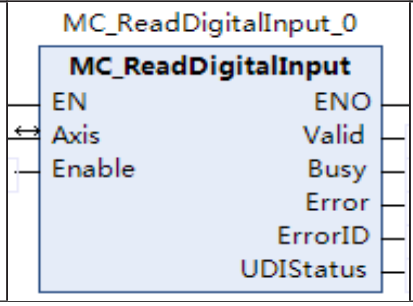
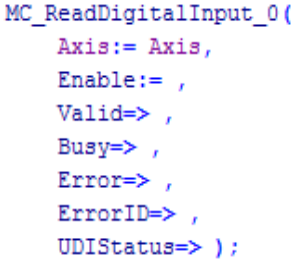
5) Timing Diagram

Omitted.

MC_ReadDigitalInput

This instruction reads digital inputs.

1) Instruction Format

Instruction	Name	FB/FC	LD Expression	ST Expression
MC_ ReadDigitalInput	Digital input read	FB		


```
MC_ReadDigitalInput_0(
    Axis:= Axis,
    Enable:= ,
    Valid=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> ,
    UDIStatus=> );
```

2) Related Variables

Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

- Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Start	BOOL	TRUE, FALSE	FALSE	TRUE: Execute the function block. FALSE: Do not execute the function block.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Active state	BOOL	TRUE, FALSE	FALSE	TRUE: The function block has a valid output.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	DWORD	-	0	Output an error code when an error occurs
UDIStatus	Input terminal status	UDINT	-	0	DI terminal state. The standard format compliant with CiA402 is defined as follows: Bit 0: Negative limit signal; Bit 1: Positive limit signal; Bit 2: Home signal; Bit 3-31: Custom

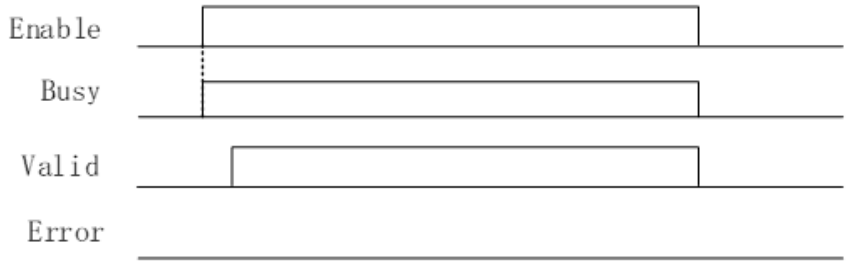
3) Function Description

- ◆ The instruction is active high: The short pulse on the digital input may end before the next function block period occurs.
- ◆ This instruction reads the status of the axis digital input terminals. It applies to EtherCAT bus axes and does not support the virtual axis mode.
- ◆ When Enable = ON, the Valid signal is valid (if the value 0x60fd is read successfully in the requested EtherCAT bus axis).
- ◆ An error is returned in the following conditions:
  The axis number does not exist.
- ◆ Axis initialization fails.
- ◆ The axis type is incorrect.

4) Precautions

This instruction can read digital input values regardless of whether the PDO with digital input (16#60fd) is configured.

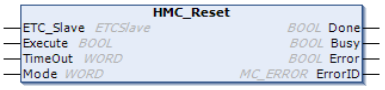
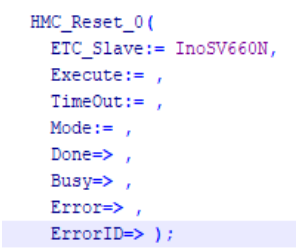
5) Timing Diagram



HMC_Reset

This instruction resets drive communication faults and axis faults.

1) Instruction Format

Instruction	Name	FB/FC	LD Expression	ST Expression
HMC_Reset	Fault reset	FB		


```
HMC_Reset_0(
  ETC_Slave:= InoSV660N,
  Execute:= ,
  TimeOut:= ,
  Mode:= ,
  Done=> ,
  Busy=> ,
  Error=> ,
  ErrorID=> );
```

2) Related Variables

Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
ETC_Slave	Axis	ETCSlave	-	-	

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Start	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
TimeOut	Timeout time	WORD	>=3000	10000	Timeout time = Task period x TimeOut. The timeout time is at least 3s.
Mode	Mode	WORD	0 or 1	0	0: Fast reset 1: Slow reset with DC synchronization, reset time > 20s

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Execution completed	BOOL	TRUE, FALSE	FALSE	TRUE: Reset completed
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE: Reset in progress
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ • This instruction resets EtherCAT slaves and CIA402 axes, such as servo drive, AC drive, and EtherCAT remote I/O modules.

◆ It automatically recognizes the number and status of CIA402 axes under EtherCAT slave devices, and resets the axis in Errorstop state for the state machine. It facilitates axis reset for multidrive devices, such as IS810, SV820, GR10_4MPE of Inovance.

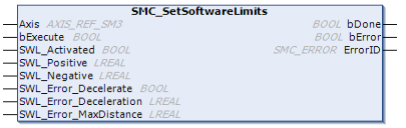
4) Precautions

Generally, fast reset is adopted, that is, Mode is set to 0. If Mode is set to 1, slow reset is adopted, and online reset provides the DC function. For an EtherCAT slave device that is powered on again, for example, partial servo failure on the bus, restart after a power failure, and online access to the master, this mode must be adopted; otherwise, an unpredictable error may occur.

SMC_SetSoftwareLimits

This instruction sets software limits for the host controller.

1) Instruction Format

Instruction	Name	FB/FC	LD Expression	ST Expression
SMC_ SetSoftwareLimits	Instruction for setting software limits	FB		MC_SetTorqueLimit_0(Axis:= Axis, Execute:= , PositiveEnable:= , PositiveValue:= , NegativeEnable:= , NegativeValue:= , Done=> , Busy=> ,

2) Related Variables

Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Axis	Axis	AXIS_REF_SM3	-	-	

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bExecute	Start	BOOL	TRUE, FALSE	FALSE	Start the motion at the rising edge
SWL_Activated	Software limit active	BOOL	TRUE, FALSE		TRUE: Activate software limit
SWL_Positive	Positive limit	LREAL	-	0	Positive limit
SWL_Negative	Negative limit	LREAL	-	0	Negative limit
SWL_Error_ Decelerate	Decelerate configuration	BOOL	TRUE, FALSE		Invalid setting. By default, deceleration is required when the limit is encountered.
SWL_Error_ Deceleration	Deceleration for overlimit error response	LREAL	-	0	Deceleration for positive limit
SWL_Error_ MaxDistance	Maximum deceleration distance	LREAL	-	0	Maximum deceleration distance, only used in positive limit

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Execution completed	BOOL	TRUE, FALSE	FALSE	TRUE: Reset completed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

This instruction sets the positive and negative position limits for the host controller and the response to a software limit error.

4) Precautions

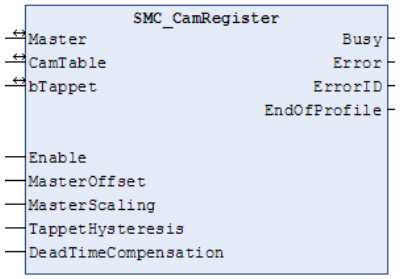
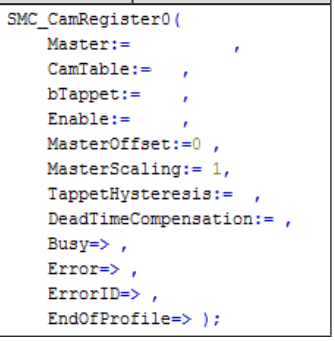
The deceleration rate of the overlimit response is the largest one among the three parameters: "fSwLimitDeceleration" (function block parameter "SWL_Error_Deceleration"), background dynamic limit parameter "fSWMaxDeceleration", and the deceleration calculated by stopping at the maximum deceleration distance "fSWErrorDistance".

6.2 Axis Group Instructions (Master/Slave Axis Instructions)

SMC_CamRegister

This instruction performs cam tappet control (cam switch). Tappet control can be achieved with this function block by configuring the tappet table without editing the master/slave axis curve during cam editing.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_ CamRegister	Cam tappet control instruction		


```
SMC_CamRegister0(
    Master:=          ,
    CamTable:=    ,
    bTappet:=     ,
    Enable:=      ,
    MasterOffset:=0 ,
    MasterScaling:= 1,
    TappetHysteresis:=  ,
    DeadTimeCompensation:= ,
    Busy=> ,
    Error=> ,
    ErrorID=> ,
    EndOfProfile=> );
```

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Master	Master axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
CamTable	Cam table	MC_CAM_REF	-	-	Reference to an electronic cam, that is, an instance of electronic cam
bTappet	Tappet output	ARRAY [1..MAX_ NUM_TAPPETS] OF BOOL	-	-	Output of the tappet point

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Executed	BOOL	TRUE, FALSE	FALSE	The function block is executed if set to TRUE and not executed if set to FALSE.
Masteroffset	Master axis offset	LREAL	-	0	Master axis offset
MasterScaling	Master axis scale	LREAL	-	1	Linear scaling factor of master axis
TappetHysteresis	Tappet damping	LREAL	-	0	Tappet control damping factor
DeadTime Compensation	Dead-zone time compensation	LREAL	-	0	The tappet output is linearly compensated according to the current velocity of the master axis. The value can be positive or negative. Unit: s

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE: Executing function block
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	-	SMC_NO_ERROR	Output an error code when an error occurs
EndofProfile	Profile period completed	BOOL	TRUE, FALSE	FALSE	TRUE: The master axis position is greater than or equal to the set period.

3) Function Description

◆ When the Enable signal is TRUE, and there is no error output, Busy outputs TRUE, indicating that tappet control is executed.

◆ This function block is irrelevant to the slave axis in the electronic cam. Only the master axis period and tappet table need to be configured.

◆ bTappet is a one-dimensional Boolean structure (MAX_NUM_TAPPETS=512), and bTappet[i] corresponds to the output of the ith tappet point.

◆ The unit of DeadTimeCompensation is second. When it is set to a positive value, the tappet signal will be output in advance; when it is set to a negative value, the tappet signal will be output with lag.

For example, if it is set to 0.02 seconds, and Ethcat task period is set to 4 ms, then the tappet outputs the tappet value at the master axis set position calculated by this formula: P - V*0.02 (V: linear velocity of the master axis; P: tappet output position). If it is set to -0.02 seconds, the tappet signal will be output with a lag of five periods after the master axis set position is greater than or equal to P.

◆ Example of using this function block:

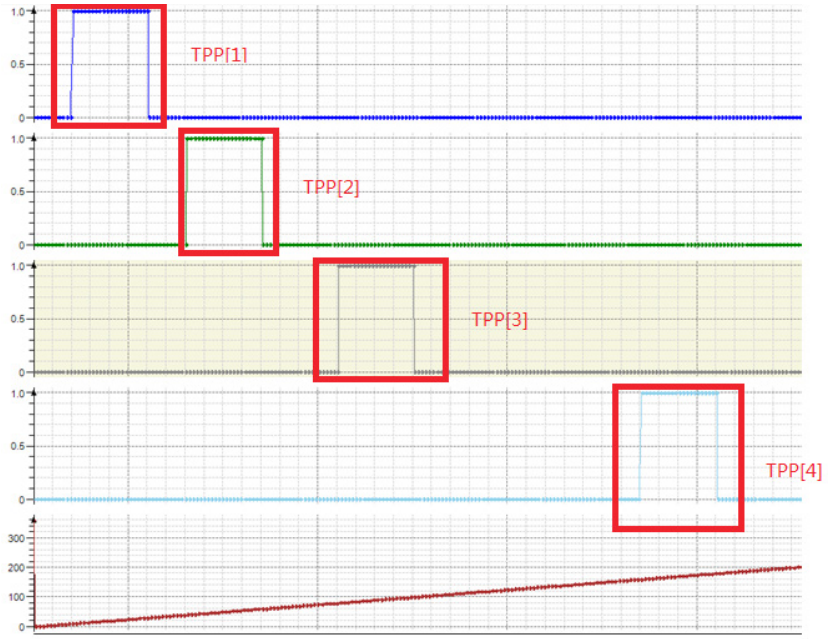
Variable declaration:

```
VAR
        TPP:ARRAY[1..MAX_NUM_TAPPETS] OF BOOL;
        SMC_CamRegister0: SMC_CamRegister;
END_VAR
Program section:
SMC_CamRegister0(
        Master:=Virtual_X ,
        CamTable:=Cam,
        bTappet:=TPP ,
        Enable:=TRUE ,
        MasterOffset:=0 ,
        MasterScaling:= 1,
        TappetHysteresis:= 0,
        DeadTimeCompensation:=0 ,
        Busy=> ,
        Error=> ,
        ErrorID=> ,
        EndOfProfile=> );
Cam editing:
```

	Track ID	X	positive pass	negative pass
	1			
		10	switch OFF	none
		30	switch ON	none
	2			
		40	switch OFF	none
		60	switch ON	none
	3			
		80	switch OFF	none
		100	switch ON	none
	4			
		160	switch OFF	none

Start axis Virtual_X:

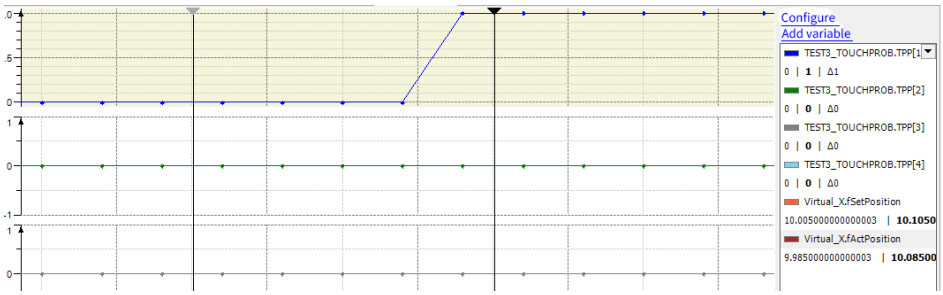
Monitoring curve:



When the dead-zone time compensation is set to -0.02 seconds

```
SMC_CamRegister0(
        Master:=Virtual_X ,
        CamTable:=Cam,
        bTappet:=TPP ,
        Enable:=TRUE ,
        MasterOffset:=0 ,
        MasterScaling:= 1,
        TappetHysteresis:= 0,
        DeadTimeCompensation:=-0.02 ,
        Busy=> ,
        Error=> ,
        ErrorID=> ,
        EndOfProfile=> );
```

The tappet output lags five task periods (each task period is 4 ms):



4) Error Description

There is an axis error, the axis is disabled, or the offset value or the scale value exceeds the master axis range.

Note: For details on the error code, see “ “Appendix C Error Codes”.

SMC_GetCamSlaveSetPosition

This instruction reads the slave axis position, velocity and acceleration information of the cam table.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_GetCamSlaveSetPosition	Instruction for obtaining the cam slave axis position	SMC_GetCamSlaveSetPosition Master — fStartPosition Slave — fStartVelocity fStartAcceleration Busy Enable — Error MasterOffset — ErrorID SlaveOffset MasterScaling SlaveScaling CamTableID	SMC_GetCamSlaveSetPosition0(Master:= , Slave:= , Enable:= , MasterOffset:= , SlaveOffset:= , MasterScaling:= , SlaveScaling:= , CamTableID:= fStartPosition=> , fStartVelocity=> , fStartAcceleration=> , Busy=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Master	Master axis	AXIS_REF	-	-	Reference to the axis
Slave	Slave axis	AXIS_REF	-	-	Reference to the axis

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Executed	BOOL	TRUE, FALSE	FALSE	The function block is executed if set to TRUE and not executed if set to FALSE.
Masteroffset	Master axis offset	LREAL	-	0	Master axis offset of the cam table
Slaveoffset	Slave axis offset	LREAL	-	0	Slave axis offset of the cam table
MasterScaling	Master axis scaling	LREAL	-	1	Master axis scaling factor of the cam table
SlaveScaling	Slave axis scaling	LREAL	-	1	Slave axis scaling factor of the cam table
CamTableID	Cam ID	MC_CAM_ID	-	-	Cam table ID

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
fStartPosition	Slave axis position	LREAL	-	0	Slave axis position obtained based on the cam table and the current master axis information
fStartVelocity	Slave axis velocity	LREAL	-	0	Slave axis velocity obtained based on the cam table and the current master axis information
fStartAcceleration	Slave axis acceleration	LREAL	-	0	Slave axis acceleration rate obtained based on the cam table and the current master axis information
busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE: Executing the function block
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	-	SMC_NO_ERROR	Output an error code when an error occurs

3) Function Description

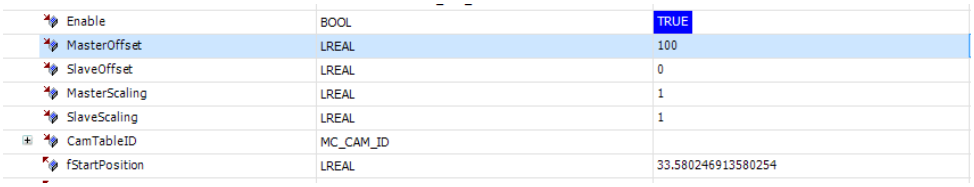
Output value calculated by this instruction: Y = (Cam ((Cam start master axis position of the cam + Masteroffset) x MasterScaling) + Slaveoffset) x SlaveScaling. Cam is the cam table function. For example, if the cam start master axis position is 0, the master/slave scaling ratio is 1, Masteroffset is 100, and Slaveoffset is 0, then the output of the function block is the slave axis position corresponding to the cam table at 100.

This function block can read the slave position as long as the cam table is built successfully. It has no requirement on whether the master and slave axes are running.

Example:

```
Declaration:
SMC_GetCamSlaveSetPosition0: SMC_GetCamSlaveSetPosition;
ENABLE: BOOL;
    MC_CamTableSelect0: MC_CamTableSelect;
Program:
MC_CamTableSelect0(
    Master:=Virtual_X ,
    Slave:=Virtual_Y ,
    CamTable:=Cam ,
    Execute:= ,
    Periodic:=TRUE ,
    MasterAbsolute:=0 ,
    SlaveAbsolute:=0 ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> ,
    CamTableID=> );
SMC_GetCamSlaveSetPosition0(
    Master:= Virtual_X,
    Slave:= Virtual_Y,
    Enable:=ENABLE ,
    MasterOffset:= 100,
    SlaveOffset:=0 ,
    MasterScaling:=1 ,
    SlaveScaling:= 1,
    CamTableID:=MC_CamTableSelect0.CamTableID,
```

fStartPosition=> ,
fStartVelocity=> ,
fStartAcceleration=> ,
Busy=> ,
Error=> ,
ErrorID=>);



4) Error Description

The instruction error is output when Error is TRUE.

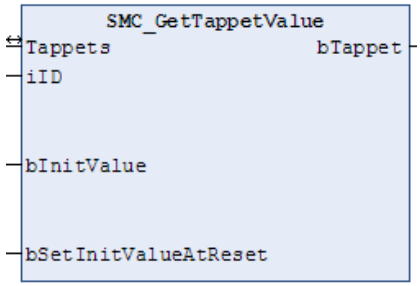
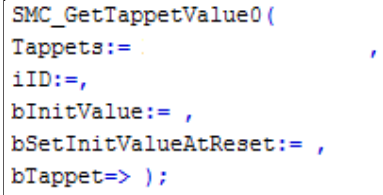
See ErrorID and SMC_ERROR to determine the cause of the error.

Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC_GetTappetValue

This instruction obtains the current tappet output value when used in conjunction with the MC_CamIn instruction.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_ GetTappetValue	Instruction for obtaining the tappet output value		

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Tappets	Tappet	SMC_TappetData	-	-	Reference to the tappet

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
iID	Tappet group ID	INT	-	0	Group ID of the tappet
bInitValue	Initial Value	BOOL	-	-	Initialization value of the tappet when the function block is called for the first time

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bSetInitValueAtReset		BOOL	-	-	TRUE: The tappet output value will be initialized to bInitValue when the MC_CamIn function block is restarted. FALSE: The current tappet output value will be kept when the MC_CamIn function block is restarted.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bTappet	Tappet output	BOOL	-	FALSE	Tappet value

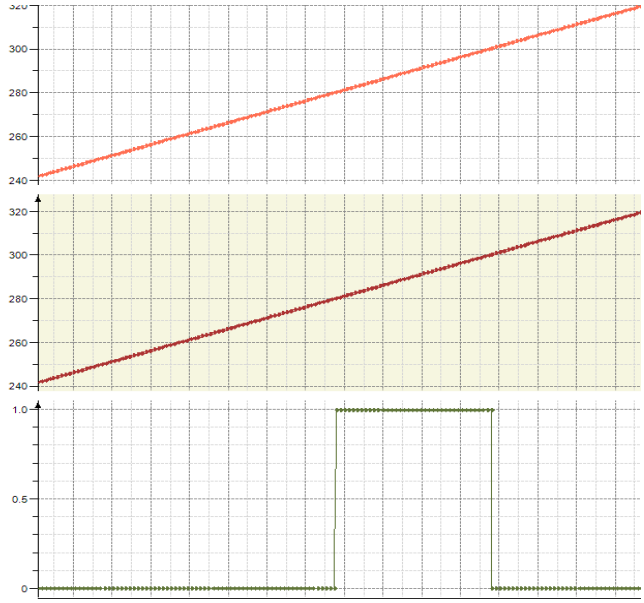
3) Function Description

◆ This function block must be used in conjunction with the MC_CamIn instruction.

◆ This function block reads the tappet output value as does the SMC_CamRegister function. Due to a conflict between the two, only one of these instructions can be used in a cam tappet table.

Example of use:

```
MC_CamIn0(
        Master:=Virtual_X ,
        Slave:= Virtual_Y,
        Execute:= ,
        MasterOffset:= 0,
        SlaveOffset:= 0,
        MasterScaling:=1 ,
        SlaveScaling:= 1,
        StartMode:= 1,
        CamTableID:= MC_CamTableSelect0.CamTableID,
        VelocityDiff:= ,
        Acceleration:= ,
        Deceleration:= ,
        Jerk:= ,
        TappetHysteresis:= ,
        InSync=> ,
        Busy=> ,
        CommandAborted=> ,
        Error=> ,
        ErrorID=> ,
        EndOfProfile=> ,
        Tappets=> );
 SMC_GetTappetValue0(
        Tappets:= MC_CamIn0.Tappets,
        iID:=2,
        bInitValue:= false,
        bSetInitValueAtReset:=true ,
        bTappet=> );
```



4) Error Description

There is an axis error.

The axis is disabled.

CamTable ID has no mapping object.

Note: For details on the error code, see “ “Appendix C Error Codes” .

MC_CamTableSelect

This instruction specifies the cam table. It must be used in conjunction with the MC_CamIn instruction.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_ CamTableSelect	Instruction for specifying the cam table	MC_CamTableSelect Master Slave CamTable Execute Periodic MasterAbsolute SlaveAbsolute Done Busy Error ErrorID CamTableID	MC_CamTableSelect0(Master:= Slave:= CamTable:=), Execute:= , Periodic:= , MasterAbsolute:= , SlaveAbsolute:= , Done=> , Busy=> , Error=> , ErrorID=> , CamTableID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Master	Master axis	AXIS_REF_SM3	-	-	Reference to the master axis, that is, an instance of AXIS_ REF_SM3
Slave	Slave axis	AXIS_REF	-	-	Reference to the slave axis, an instance of AXIS_REF_SM3

CamTable	Table selection	MC_CAM_REF	-	-	Reference to the cam table description, that is, an instance of MC_CAM_REF

Note:

The master and slave axes cannot be specified as the same axis; otherwise, an error will be output. The cam table corresponding to CamTable must be correctly edited; otherwise, an instruction error will be reported. The master and slave axes can be real or virtual axes.

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Executed	BOOL	TRUE, FALSE	FALSE	Execute the instruction at the rising edge
Periodic	Periodic Mode	BOOL	TRUE, FALSE	FALSE	Specify whether to execute the cam table periodically or only once TRUE: Periodically False: Once
MasterAbsolute	Absolute mode of master axis	BOOL	TRUE, FALSE	FALSE	Specify whether the following distance coordinate system of the master axis is based on absolute or relative position. 1: Absolute position, 0: Relative position
SlaveAbsolute	Absolute mode of slave absolute	BOOL	TRUE, FALSE	FALSE	Specify whether the current instruction position of the slave axis is the absolute or relative value of the cam table output with StartMode in the MC_CamIn instruction. Absolute: Cam table output value corresponding to the current master axis position Relative: Cam table output value superimposed by the slave axis position at the start of the instruction 1: Absolute position, 0: Relative position

Note:

Improper selection of MasterAbsolute and SlaveAbsolute may cause the electronic cam output to jump. Therefore, determine the cam curve operating mode before setting the variables.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the selection is completed
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the selection is in progress
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs
CamTableID	Valid cam ID	MC_CAM_ID	-	-	Select the valid Cam_ID, which is used together with CamTableID in MC_CamIn instruction.

Note:

When an error occurs, see SMC_ERROR in Help based on ErrorID.

3) Function Description

◆ This instruction specifies the cam table required for the electronic cam to run. Therefore, edit the cam table online or through the cam editor before using this instruction.

◆ At the rising edge of Execute, the specified cam table is executed, and the specified cam table can be refreshed after a cam table update.

- When the Done signal outputs TRUE, the output variable CamTableID is generated and becomes valid.
- During instruction execution, the Busy signal outputs TRUE. When the Done signal outputs TRUE, the Busy signal outputs FALSE.
- For the specific functions of MasterAbsolute, SlaveAbsolute, and Periodic, see the MC_CamIn instruction.

4) Error Description

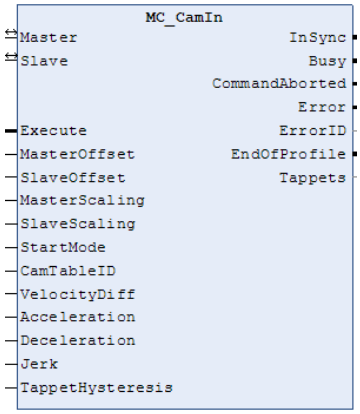
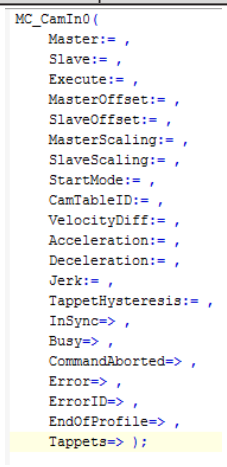
- The master and slave axes cannot be specified as the same axis; otherwise, an error will be output.
- The cam table corresponding to CamTable must be edited correctly; otherwise, an error will be output.

Note: For details on the error code, see "Appendix C Error Codes".

MC_Camin

This instruction uses the specified cam table to start executing the electronic cam action. The offset value, scaling ratio and working mode of the master and slave axes can be specified according to the application requirements.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_CamIn	Start cam operation		


```
MC_CamIn0(
    Master:= ,
    Slave:= ,
    Execute:= ,
    MasterOffset:= ,
    SlaveOffset:= ,
    MasterScaling:= ,
    SlaveScaling:= ,
    StartMode:= ,
    CamTableID:= ,
    VelocityDiff:= ,
    Acceleration:= ,
    Deceleration:= ,
    Jerk:= ,
    TappetHysteresis:= ,
    InSync=> ,
    Busy=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorID=> ,
    EndOfProfile=> ,
    Tappets=> );
```

2) Related Variables

- Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Master	Master axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
Slave	Slave axis	AXIS_REF	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

Note:

The master and slave axes cannot be specified as the same axis; otherwise, an error will be output.

- Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Cam function execution	BOOL	TRUE, FALSE	FALSE	Execute the electronic cam at the rising edge
MasterOffset	Master offset	LREAL	Negative value, positive value, 0	0	Move the phase of the master axis based on the specified offset value
SlaveOffset	Slave axis offset	LREAL	Negative value, positive value, 0	0	Move the phase of the slave axis based on the specified offset value

Input Variable	Name	Data Type	Value Range	Initial Value	Description
MasterScaling	Pre-compiling ratio of master axis	LREAL	> 0.0	1	Zoom in/out the phase of the master axis based on the specified ratio
SlaveScaling	Pre-compiling ratio of slave axis	LREAL	> 0.0	1	Zoom in/out the phase of the slave axis based on the specified ratio
StartMode	Slave axis output mode relative to cam	MC_ StartMode	0 to 4	absolute	0: Absolute position:1: Relative position 2: ramp_in; 3: ramp_in_pos; 4: ramp_in_neg
CamTableID	Table ID	MC_CAM_ ID	> 0	-	Define the use of the cam table. It is used in conjunction with the output point CamTableID of MC_CamTableSelect
VelocityDiff	Coupling velocity	LREAL	> 0.0	0	Maximum velocity different from ramp_in
Acceleration	Acceleration	LREAL	> 0.0	0	Acceleration at ramp_in
Deceleration	Deceleration	LREAL	> 0.0	0	Deceleration at ramp_in
Jerk	Jerk	LREAL	> 0.0	0	Acceleration at ramp_in
TappetHysteresis	Tappet factor Range	LREAL	> 0.0	0	Tappet damping factor

Note:

The master and slave axes cannot be specified as the same axis; otherwise, an error will be output.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
InSync	Cam valid	BOOL	TRUE, FALSE	FALSE	InSync is set after the cam relationship is established between the master and slave axes, and is reset when the execution condition of the instruction is OFF.
Busy	Synchronous operation in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE at the rising edge of Execute, which indicates that the cam relationship is in coupling and needs to be reset with the Cam_out instruction. The instruction execution condition reset cannot reset this status.
CommandAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Output TRUE when the slave axis is interrupted by other control instructions
Error	Error	BOOL	TRUE, FALSE	FALSE	When an error is detected, the Error bit is set. The Error bit is reset when the execution condition of the instruction is OFF.
ErrorID	Error code	SMC_ERROR	See SMC_ ERROR	0	Output an error code when an error occurs
EndOfProfile	Curve completed	BOOL	-	FALSE	If Periodic is 0 (non-periodic) during the execution of MC_CamTableSelect, the EndOfProfile bit is set when the cam curve is executed once and is reset when the execution condition of the instruction is OFF.
Tappets		SMC_ TappetData	-	-	Associate a cam tappet that can be read by the MC_ GetTappetValue instruction.

3) Function Description

◆ This instruction is started at the rising edge of Execute if there is no axis error and the cam table is selected correctly.

◆ In a cam system, to call a cam curve, first call the MC_CamTableSelect instruction to select the cam table, and then execute MC_CamIn. To replace the cam curve, call the MC_CamTableSelect instruction to re-select the cam table.

◆ It is necessary to use the Camout instruction to cancel the cam coupling relationship between the

master axis and the slave axis.

- During the execution of this instruction, if the slave axis of this instruction executes other motion instructions, the cam relationship between the slave axis and the master axis will be canceled, and CommandAborted will output TRUE.

4) Instruction Details

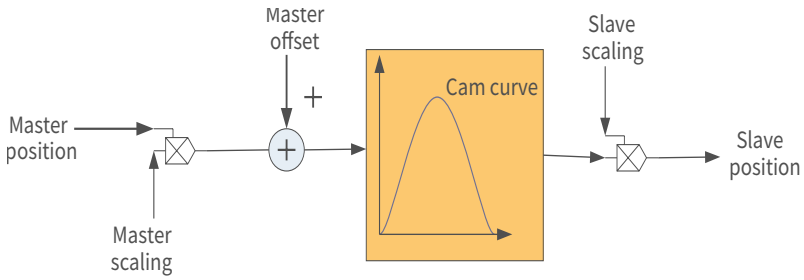
The following describes the instruction in details:

- Instruction Execution Condition

This instruction can be started in the status of master axis stopping, position control, velocity control, or synchronous control.

Note: The cam slave position setpoint should be within the software limit; otherwise, the instruction will be incorrectly output.

- The contact point in the cam curve is calculated as follows:



According to the above diagram, the calculation formula is as follows:

Position_Slave = SlaveScaling x CAM (MasterScaling x MasterPosition + MasterOffset) + SlaveOffset

The master and slave positions in this formula are related to the cam function curve and do not represent the actual physical axis positions.

The relationship between the master/slave axis positions and the master/slave real axis positions is described in detail.

Note: The master and slave positions are required for the cam function curve and are not the master and slave real axis positions.

- Relationship of Periodical Mode to EndOfProfile:

The periodical mode determines whether or not the electronic cam will be performed again after the master axis reaches the termination position.

Non-periodical mode: Periodic is set to FALSE for the MC_CamTableSelect instruction.

In non-periodical mode, EndofProfile outputs TRUE when the cam is completed and outputs FALSE when the execution input is FALSE. In this case, the cam is executed for only one master axis period.

Note: The master axis period refers to the range between the start point of the master axis of the electronic cam to the end point.

16) Periodical mode: Periodic is set to TRUE for the MC_CamTableSelect instruction.



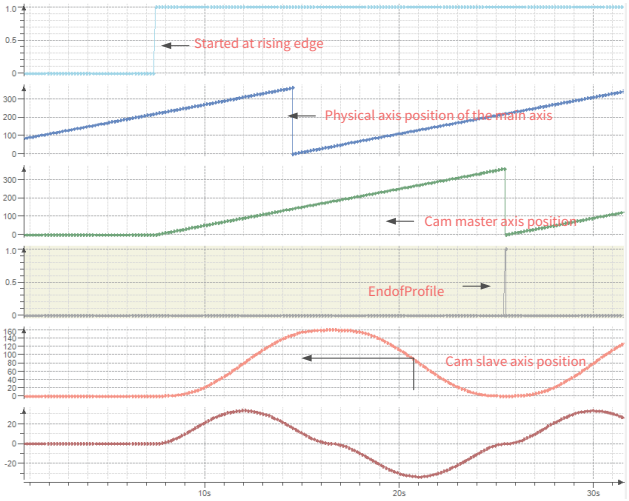
In this case, the cam will be continuously executed for the next period after the completion of one master axis period, and the TRUE output of the EndofProfile signal only lasts for one task period.

Notes:

When the cam master axis position is larger than or equal to the cam end position, the EndofProfile signal outputs TRUE, and the cam master axis position is updated to the sum of the cam start position and the portion exceeding the end position. For example, the start position of the electronic cam master axis is 0, the end position is 360, the master/slave scaling ratio is set to 1, the master/slave offset is set to 0, the task period is 2 ms, and the master axis velocity is 100. When the cam master axis position of a task period is 359.99, then EndofProfile of the next period outputs True and the master axis position is: 359.99 + 100 x 0.002 - 360 = 0.19.

It is recommended to keep a smooth transition between the start and end positions of the cam curve in periodical mode; otherwise, a position jump will be generated.

For example, if the start velocity is 0 and the end velocity is not 0, it will cause the master axis to jump at the end of the period and the beginning of a new period.



The relationship between StartMode and the absolute/relative mode of the master and slave axes in MC_CamTableSlect:

Absolute mode: At the start of a new electronic cam period, the calculation of the electronic cam is independent of the current slave axis position. If the start position of the slave axis relative to the master axis is different from the end position of the slave axis relative to the master axis, a jump will be caused.

Relative mode: The new electronic cam will change according to the current slave position. That is, the position of the slave axis at the end of the last electronic cam period will be added up by the current electronic cam motion as a "slave axis offset". However, if the slave axis position corresponding to the

master axis start position is not 0 in the electronic cam definition, a jump will be caused.

Ramp input: Add a compensating motion (obtained based on the limit value VelocityDiff, acceleration and deceleration) to prevent potential jumps at the start of the electronic cam. Thus, as long as the slave axis rotates, the positive ramp input provides only positive compensation, while the reverse ramp input only provides reverse compensation. For a linearly moving slave axis, the compensation direction can be achieved automatically, that is, the positive ramp input and the negative ramp input can be interpreted in terms of ramp inputs).

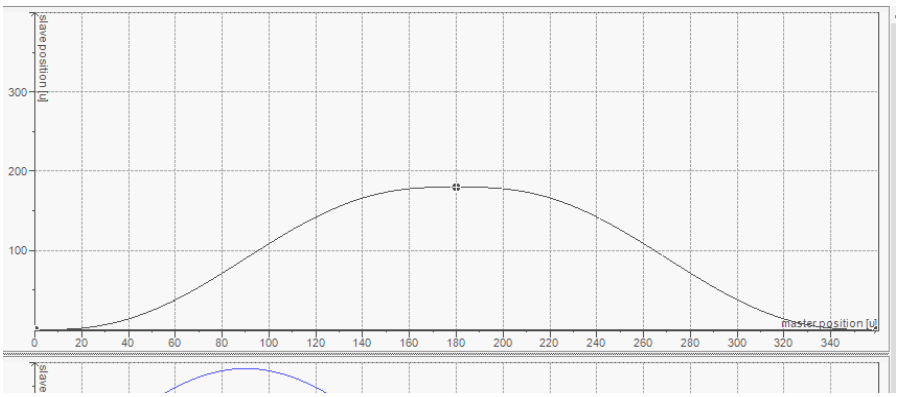
The relationship is shown in the following table:

MC_CamTableSelect.MasterAbsolute	Master axis mode
absolute	Absolute mode
relative	Relative mode

MC_CamIn.StartMode	MC_CamTableSelect.SlaveAbsolute	Slave axis mode
absolute	TRUE	Absolute mode
absolute	FALSE	Relative mode
relative	TRUE	Relative mode
relative	FALSE	Relative mode
ramp_in	TRUE	Absolute ramp-in
ramp_in	FALSE	Relative ramp-in
ramp_in_pos	TRUE	Absolute ramp-in in positive direction
ramp_in_pos	FALSE	Relative ramp-in in positive direction
ramp_in_neg	TRUE	Absolute ramp-in in negative direction
ramp_in_neg	FALSE	Relative ramp-in in negative direction

The relationship is described below:

Cam master axis range (0-360), cam slave axis range (0-180), periodical mode, master/slave offset (0), master/slave scaling ratio (1) The designed cam table is shown below:

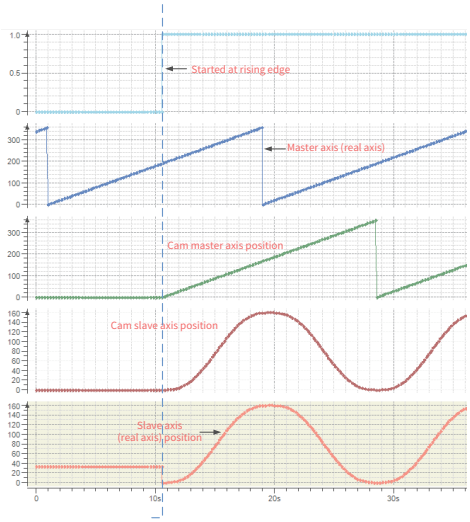


17) StartMode = 0 (absolute mode)

When MasterAbsolute of the MC_CamTableSlect instruction is set to FALSE and SlaveAbsolute is set to TRUE, then the master axis works in relative mode and the slave axis works in absolute mode. When the cam is started at the rising edge of Execute, the cam master axis starts from the "Start position" (0) of the cam table, and the cam slave axis calculates the output according to the cam table meshing formula mentioned above. The instruction position of the slave real axis is equal to the meshing calculation

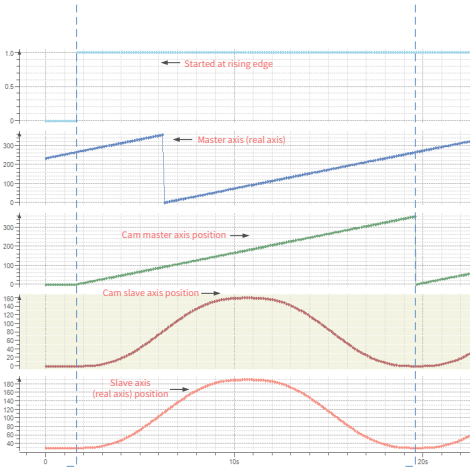
output value. For example, if the start position of the cam slave axis is 0, and the position of the slave real axis is 20 when the cam is started, then a jump will be caused when the position of the slave real axis from the start is commanded to be 0.

Note: In this case, if the start position of the slave axis (real axis) is not at the cam slave axis start position, then a jump will be caused.



When both MasterAbsolute and SlaveAbsolute of the MC_CamTableSlect instruction are set to FALSE, then the master and slave axes work in relative mode. When the cam is started at the rising edge of Execute, the cam master axis starts from the "Start position" (0) of the cam table, and the cam slave axis calculates the output according to the cam table meshing formula mentioned above. The instruction position of the slave real axis is equal to the sum of the meshing calculation output value (cam slave axis position) and the slave real axis position at startup.

For example, if the slave real axis position at cam startup is 20, and the slave axis start position of the cam table is 0, then the slave real axis position at cam startup is commanded to 20. The subsequent value is the sum of 20 and the cam table calculation value, and the peak value is 200, which is the sum of 20 and the maximum cam table calculation value (180 in this case).



5) Error Description

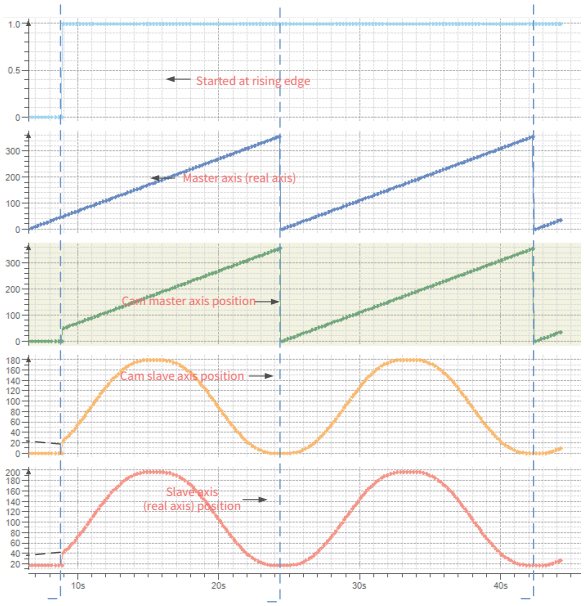
- The setting information of this instruction does not match that of the Camslect instruction.
- The axis is disabled.

When MasterAbsolute of the MC_CamTableSlect instruction is set to TRUE and SlaveAbsolute is set to FALSE, then the master axis works in absolute mode and the slave axis works in relative mode. At the rising edge of Execute, the cam master axis starts from the current "master real axis position" upon cam startup. Slave real axis position instruction = Cam table meshing calculation value (cam slave axis

position) + Slave axis position at startup

Note: 1. In this case, if the start position of the master axis (real axis) is not at the cam master axis start position, then a jump will be caused.

2. The master axis position must be within the position range of the cam master axis.

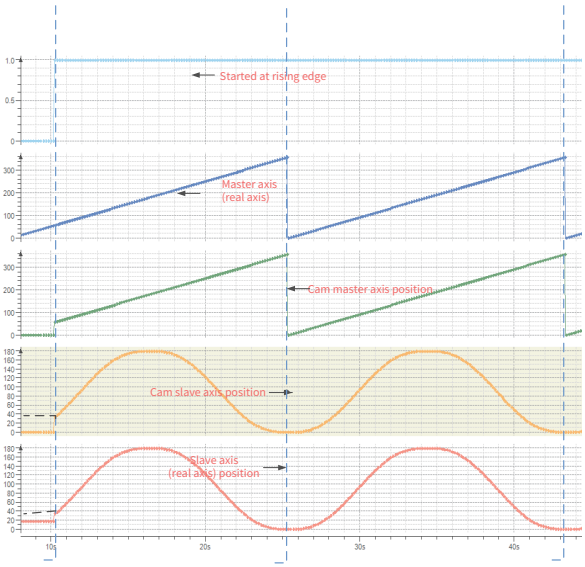


When both MasterAbsolute and SlaveAbsolute of the MC_CamTableSlect instruction are set to TRUE, then the master and slave axes work in absolute mode. At the rising edge of Execute, the cam master axis starts from the current "master real axis position" upon cam startup. Slave real axis position instruction = Cam table meshing calculation value (cam slave axis position)

Notes:

1. In this case, if the start position of the master axis (real axis) is not at the cam master axis start position and the slave axis position is not at the cam slave axis start position, then a jump will be caused.

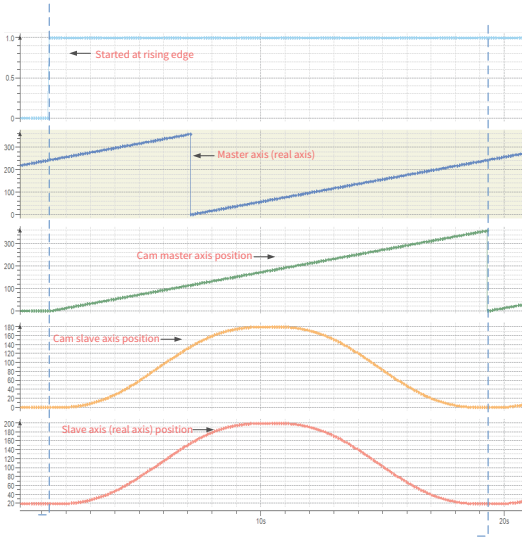
2. The master axis position must be within the position range of the cam master axis.



2) StartMode = 1 (relative mode)

When MasterAbsolute of the MC_CamTableSlect instruction is set to FALSE and SlaveAbsolute is set to TRUE or FALSE, then the master and slave axes work in relative mode. At the rising edge of Execute, the cam master axis starts from the "cam table start position" upon cam startup. Slave real axis position instruction = Cam table meshing calculation value + Cam table meshing calculation value (cam slave axis

position)

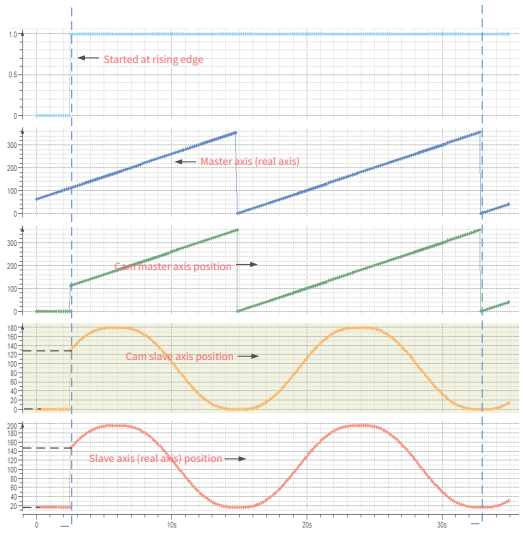


When MasterAbsolute of the MC_CamTableSlect instruction is set to TRUE and SlaveAbsolute is set to TRUE or FALSE, then the master axis works in absolute mode and the slave axis works in relative mode. At the rising edge of Execute, the cam master axis starts from the "current master axis position" upon cam startup. Slave real axis position instruction = Slave axis position upon startup + Cam table meshing calculation value (cam slave axis position)

Notes:

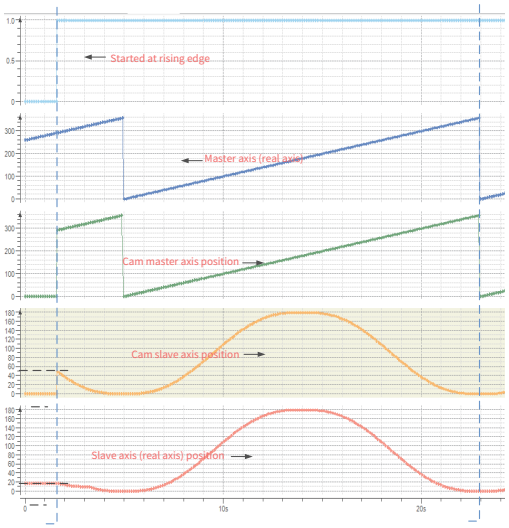
1. In this case, if the start position of the master axis (real axis) is not at the cam master axis start position, then a jump will be caused.

2. The master axis position must be within the position range of the cam master axis.

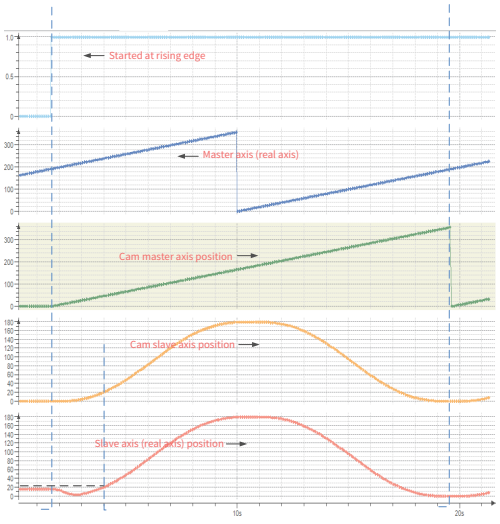


3) StartMode = 2 (ramp-in mode)

When both MasterAbsolute and SlaveAbsolute of the MC_CamTableSlect instruction are set to TRUE, then the master and slave axes work in absolute mode. At the rising edge of Execute, the cam master axis starts from the "current master axis position" upon cam startup. The slave axis adds a compensation motion through the set VelocityDiff, Acceleration, and Deceleration to avoid the potential jump during switching. Slave real axis position instruction = Cam table meshing calculation value (cam slave axis position) + f(VelocityDiff, Acceleration, Deceleration)

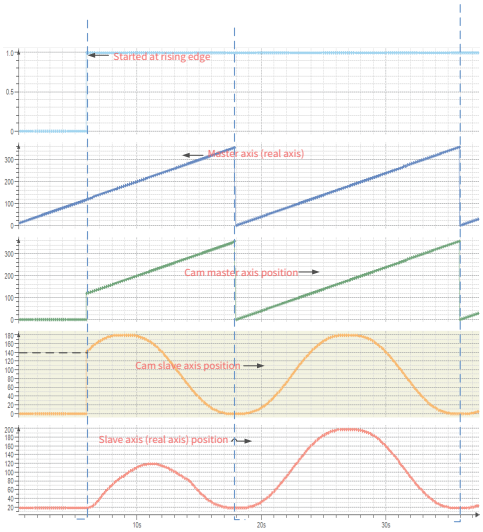


When MasterAbsolute of the MC_CamTableSlect instruction is set to FALSE and SlaveAbsolute is set to TRUE, then the master axis works in relative mode and the slave axis works in absolute mode. At the rising edge of Execute, the cam master axis starts from the "cam master axis start position" upon cam startup. The slave axis adds a compensation motion through the set VelocityDiff, Acceleration, and Deceleration to avoid the potential jump during insertion. Slave real axis position instruction = Cam table meshing calculation value (cam slave axis position) + f(VelocityDiff, Acceleration, Deceleration)



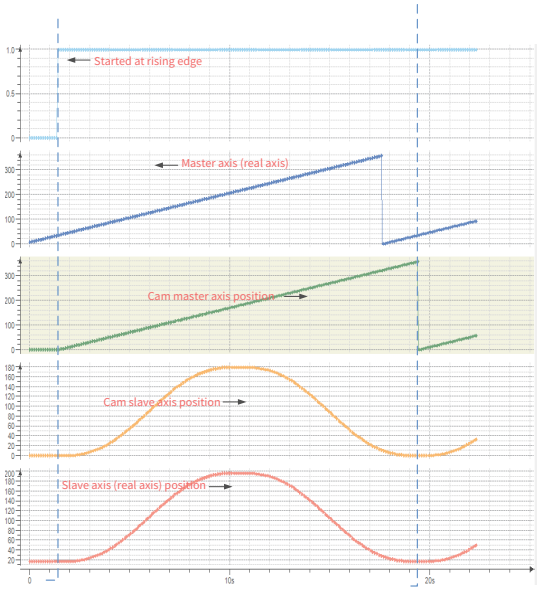
When MasterAbsolute of the MC_CamTableSlect instruction is set to TRUE and SlaveAbsolute is set to FALSE, then the master axis works in absolute mode and the slave axis works in relative mode. At the rising edge of Execute, the cam master axis starts from the "current master axis position" upon cam startup. The slave axis adds a compensation motion through the set VelocityDiff, Acceleration, and Deceleration to avoid the potential jump during switching. Slave real axis position instruction = Slave axis current position + Cam table meshing calculation value (cam slave axis position) + f(VelocityDiff, Acceleration, Deceleration)

Note: In this mode, the cam curve during the first master axis period may vary considerably from the designed curve.



When both MasterAbsolute and SlaveAbsolute of the MC_CamTableSlect instruction are set to FALSE, then the master and slave axes work in relative mode. At the rising edge of Execute, the cam master axis starts from the "cam master axis start position" upon cam startup. The slave axis adds a compensation motion through the set VelocityDiff, Acceleration, and Deceleration to avoid the potential jump during insertion. Slave real axis position instruction = Slave axis current position + Cam table meshing calculation value (cam slave axis position) + f(VelocityDiff, Acceleration, Deceleration)

Note: In this mode, the cam curve during the first master axis period may vary considerably from the designed curve.



4) StartMode = 3, 4 (ramp_in_pos, ramp_in_neg)

When the slave axis works in "rotary mode", compensation is performed only in positive direction of the axis for ramp_in_pos and in negative direction for ramp_in_neg. When the axis works in linear mode, the compensation direction is automatically adjusted for ramp_in_pos, ramp_in_neg, and ramp_in, that is, if the axis is set to linear mode, it works in the same way for ramp_in_pos, ramp_in_neg, and ramp_in.

Scaling ratio, master/slave axis offset:

According to the cam meshing formula: Input variables MasterOffset and MasterScaling change the master axis position according to the following formula, and the electronic cams will calculate based on the changed position X:

X = MasterScaling x MasterPosition + MasterOffset

Therefore, the electronic cam will run at the high velocity if the value of MasterScaling is larger than 1 and run at the low velocity if the value is smaller than 1.

The SlaveOffset parameter makes the electronic cam move longitudinally (in the direction of the slave axis),

and the SlaveScaling parameter stretches the electronic cam in the direction of the slave axis. According to the following formula, the electronic cam is stretched in the first step

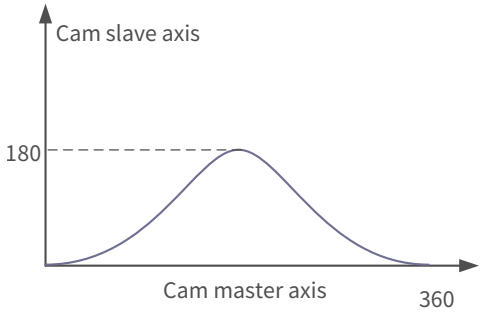
and then moves:

Y = SlaveScaling x CAM (X) + SlaveOffset

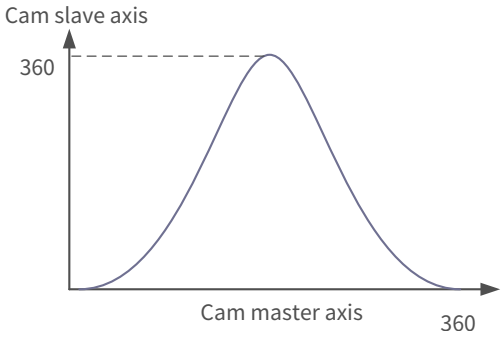
If SlaveScaling > 1, the electronic cam will be stretched and the motion range of the slave axis will be increased. If SlaveScaling < 1,

the motion range of the slave axis will be decreased.

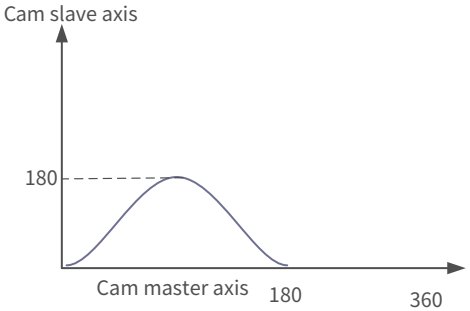
When MasterScaling = 1.0, SlaveScaling = 1.0, MasterOffset = 0, and SlaveOffset = 0, the cam curve is the planned one, as shown below:



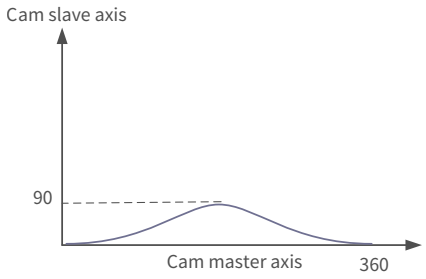
When MasterScaling = 1.0, SlaveScaling = 2.0, MasterOffset = 0, and SlaveOffset = 0, the cam curve is shown below:



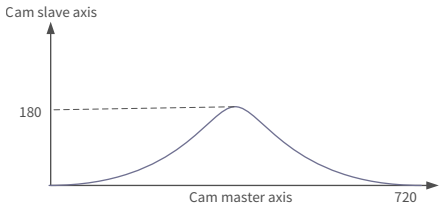
When MasterScaling = 2.0, SlaveScaling = 1.0, MasterOffset = 0, and SlaveOffset = 0, the cam curve is shown below:



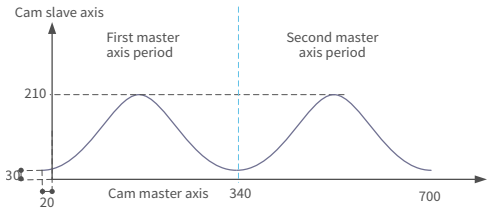
When MasterScaling = 1.0, SlaveScaling = 0.5, MasterOffset = 0, and SlaveOffset = 0, the cam curve is shown below:



When MasterScaling = 0.5, SlaveScaling = 1, MasterOffset = 0, and SlaveOffset = 0, the cam curve is shown below:



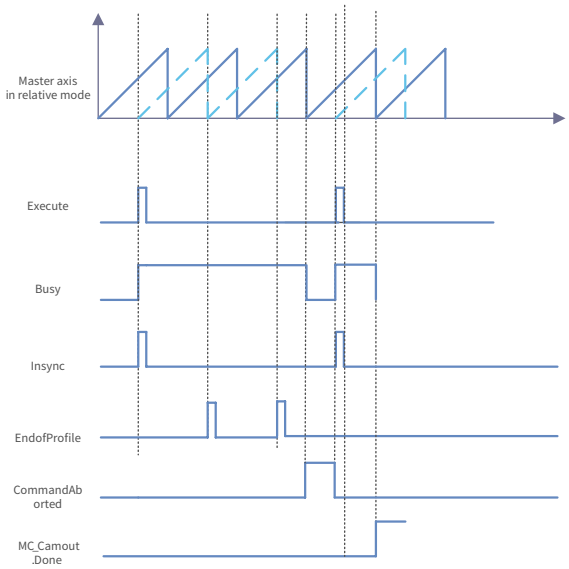
When MasterScaling = 1, SlaveScaling = 1, MasterOffset = 20, and SlaveOffset = 30, the cam curve is shown below:



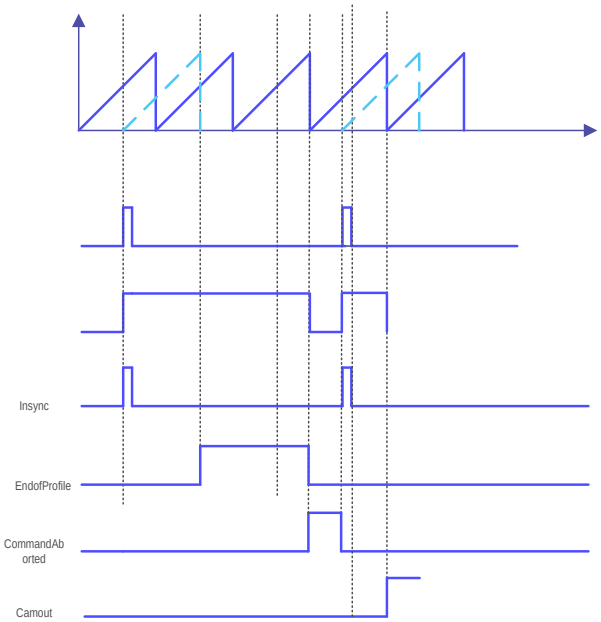
6) Timing Diagram:

The timing in periodical mode (MC_CamTableSelect.Periodic set to TRUE) is shown below:

Note: The MC_Camout instruction only cancels the cam coupling relationship between master and slave axes. If the slave axis velocity is not 0 at the time of cancellation, the slave axis will not automatically decelerate to 0. In this case, it must be used in conjunction with the MC_STOP instruction.



The timing in non-periodical mode (MC_CamTableSelect.Periodic set to FALSE) is shown below:



◆ Electronic cam restart:

The two electronic cams can be switched at any time, but some conditions must be considered: In the electronic cam editor, the slave axis position is defined as the calculated output of the electronic cam function, which is calculated based on a master axis positions within the master axis range. Therefore, this can be expressed by the following formula: SlavePosition = CAM(MasterPosition). Since the actual period of the master axis drive is generally different from the master axis range defined by the electronic cam, the master axis position must be scaled to the domain defined by the function to enable the correct input of the electronic cam function: SlavePosition = CAM(MasterScale x MasterPosition + MasterOffset). Similarly, if an electronic cam starts in absolute mode and produces an upward jump, the function output (virtual slave position) will also be corrected proportionally: SlavePosition = SlaveScale x CAM(MasterPosition) + SlaveOffset. In the worst case, both proportional corrections must be applied. Therefore, the slave position (SlavePosition) is calculated based on a more complex formula:

Slaveposition = SlaveScale x CAM(MasterScale x Masterposition + MasterOffset) + SlaveOffset

At the end of each electronic cam period, the scale and offset can be changed to obtain more appropriate parameters. However, restarting the MC_CamIn module of the electronic cam will delete its memory, including the scale and offset values. As a result, the defined electronic cam function will be adapted to different slave axis values. For this reason, it is recommended to restart MC_CamIn-FB only when another different electronic cam needs to be processed.

Note: See the motion control function part for electronic cam switching.
See the motion control function part for electronic cam samples.
See the motion control function part for the tappet function.

7) Error Description

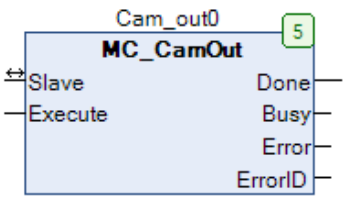
When an abnormality is detected when this instruction is activated, Error becomes TRUE.

You can check the output value of ErrorID. For details on the error code, see “ “Appendix C Error Codes”.

MC_CamOut

This instruction cancels cam coupling of the slave axis. Note: After this instruction is executed, the slave axis will continue to run at the same velocity as before decoupling. Therefore, this instruction must be used in conjunction with an instruction such as MC_Stop.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_CamOut	Instruction for canceling cam coupling		MC_CamOut(Slave:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input and output

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Slave	Slave axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Instruction execution	BOOL	-	-	Execute the instruction at the rising edge

◆ Output

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completed	BOOL	TRUE, FALSE	FALSE	Completion of cam decoupling of the master axis
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Executing instruction
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

◆ This instruction cancels cam coupling of the slave axis.

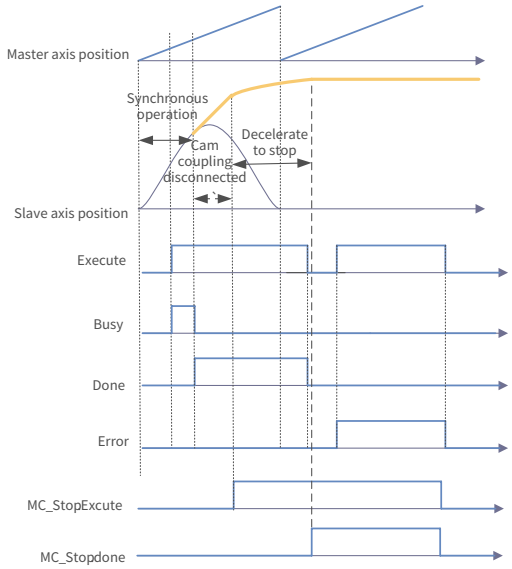
◆ At the rising edge of Execute, the cam coupling of the slave axis will be canceled.

◆ The slave axis may not stop after the cam coupling relationship is canceled.

◆ If the slave axis velocity is not 0 before this instruction is executed, the cam coupling relationship will be canceled after the completion of the DONE signal. However, the slave axis will still run at the original velocity.

◆ If this instruction is executed when the slave axis has no cam coupling relationship, an error will be output.

4) Timing Diagram

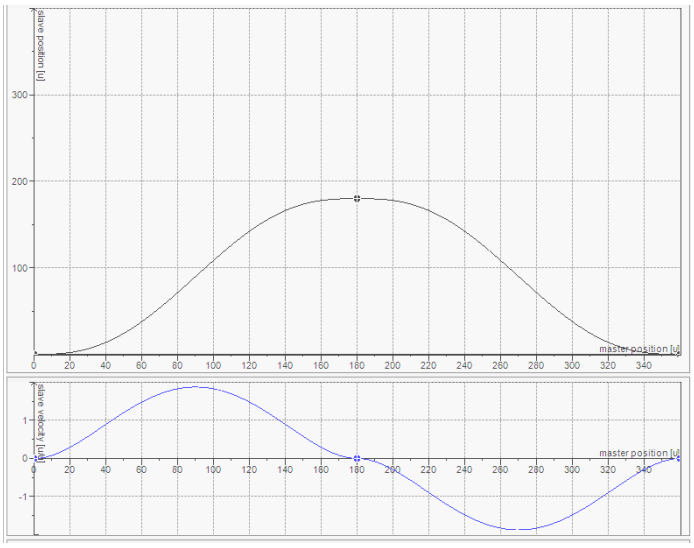


5) Usage Example

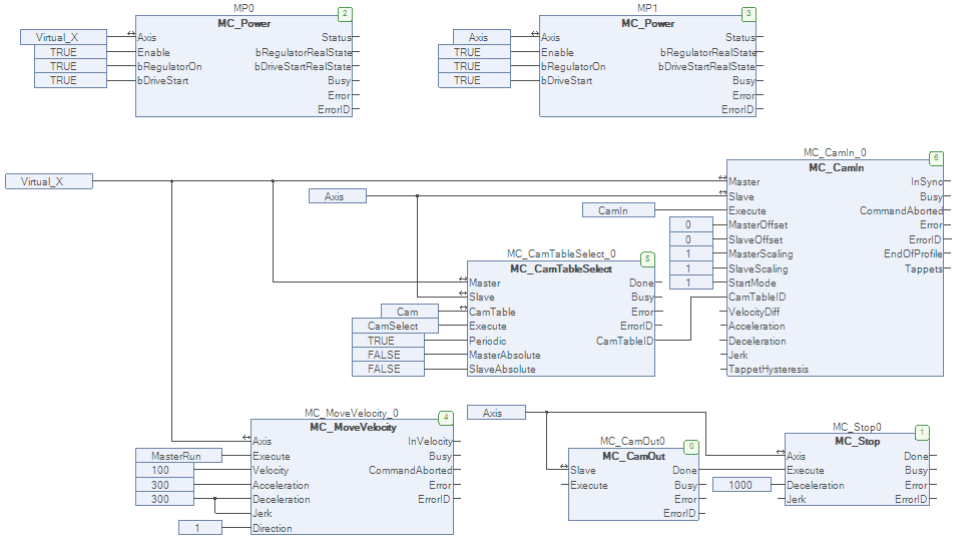
This example applies the cam-related instruction. It describes the axis motion status when a cam relationship is created, run, and canceled.

Create the following cam table in the cam editor:

	X	Y	V	A	J	Segm...	min(P...	max(P...	max(|V...	max(|A...
	0	0	0	0	0					
						Poly5	0	180	1.8749...	0.0320...
	180	180	0	0	0					
						Poly5	0	180	1.8749...	0.0320...
	360	0	0	0	0					



Program:



The master and slave axes are automatically enabled after power-on. If MasterRun is set to TRUE, the master axis runs at the velocity of 100.

Set CamSelect to TRUE, select the cam table and set CamIn to TRUE to start the electronic cam.

To disconnect the electronic cam, set MC_CamOut0.Execute to TRUE.

Notes:

See the motion control function part for online modification of the cam table.

6) Error Description

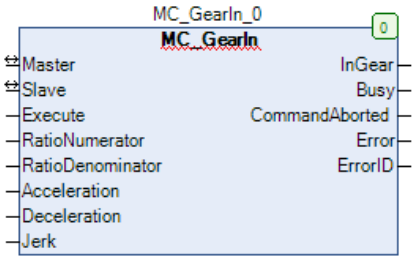
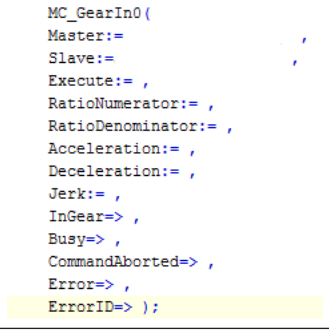
If an error occurs during instruction execution, Error outputs TRUE.

ERRORID can be checked. For details on the SMC_ERROR error code, see “ “Appendix C Error Codes”

MC_GearIn

This instruction sets the gear ratio between the slave axis and the master axis to perform electronic gearing.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_GearIn	Electronic gear function block		


```
MC_GearIn0(
Master:=                    ,
Slave:=                     ,
Execute:= ,
RatioNumerator:= ,
RatioDenominator:= ,
Acceleration:= ,
Deceleration:= ,
Jerk:= ,
InGear=> ,
Busy=> ,
CommandAborted=> ,
Error=> ,
ErrorID=> );
```

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description

Master	Master axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
Slave	Slave axis	AXIS_REF	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Executed	BOOL	TRUE, FALSE	FALSE	Execute the instruction at the rising edge
RatioNumerator	Gear ratio numerator	DINT	Positive, negative-	1 —	Gear ratio numerator
RatioDenominator	Gear ratio denominator	UDINT	Positive number	1	Gear ratio denominator
Acceleration	Acceleration	LREAL	Positive number	-	Specify an acceleration rate
Deceleration	Deceleration	LREAL	Positive number	-	Specify a deceleration rate
Jerk	Jerk	LREAL	Positive or 0	-	Jerk

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
InGear	Gear ratio reached	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the slave axis reaches the target velocity
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
CommandAborted	Aborting	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is interrupted by other control instructions
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

◆ Specify the object axis through Slave and specify RatioNumerator, RatioDenominator, ReferenceType, Acceleration, and Deceleration to perform gearing.

◆ The instruction position, feedback position, and latest instruction position can be specified for the master axis (Master).

◆ At the rising edge of Execute, the electronic gearing action starts.

◆ After the start of the action, the slave performs acceleration and deceleration with the target velocity obtained by multiplying the master axis velocity by the gear ratio.

◆ To cancel coupling after running the electronic gear, execute the GearOut instruction.

◆ This instruction is a velocity electronic gear, and the loss of synchronization distance caused by the acceleration will not be automatically compensated.

◆ When the Busy signal is TRUE during instruction execution, if the target velocity of the slave axis is not reached, the new rising edge of Execute will not affect it.

◆ When the Busy signal is TRUE during instruction execution, if the target velocity of the slave axis is reached, the new rising edge of Execute will not affect it.

◆ When the target velocity is reached, InGear is TRUE. Slave axis movement amount = Master axis movement amount x RatioNumerator/RatioDenominator.

◆ If the master axis velocity changes in real time, use this instruction with caution.

4) Precautions

* Do not use the MC_SetPosition instruction during the execution of this instruction; otherwise, an

accident may be caused by rapid motor operation.

* Before using the MC_SetPosition (current position change) instruction for the master axis, cancel the relationship between the master axis and the slave axis.

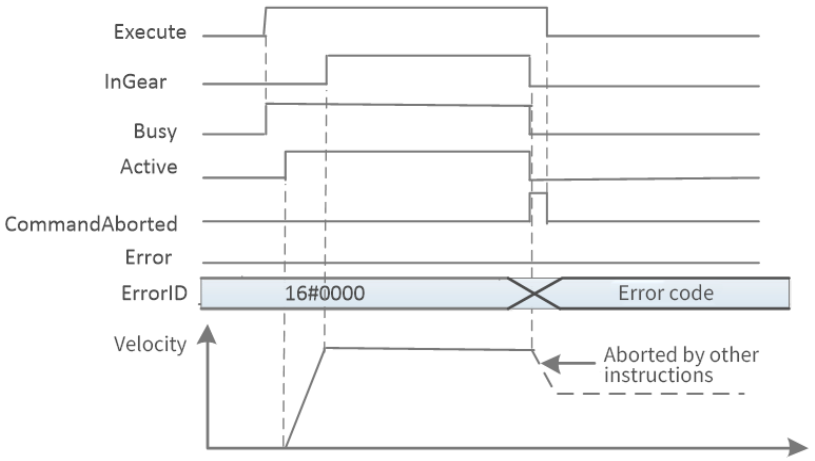
5) Timing Diagram:

The value of Busy changes to TRUE when Execute is started. The value of Active changes to TRUE in the next period.

When the target velocity is reached, InGear changes to TRUE.

If this instruction is aborted by another instruction, CommandAborted becomes TRUE, and Busy, Active, and InGear become FALSE.

To end the electronic gearing action midway, use the MC_GearOut or the MC_Stop instruction.



◆ Start of this instruction during the execution of other instructions

When this instruction is started for the currently executing instruction, it will be switched or cached to this instruction.

The action when multiple instances of this instruction are started is determined by BufferMode.

Buffer Mode	Description
Aborting	Immediately aborts the currently executing instruction and switches to this instruction. If the direction of axis motion is reversed due to instruction switching, reverse running is performed after the velocity is decelerated to zero.
Buffered	The function block is started immediately after the last instruction motion is terminated. No blending is performed here. When the end conditions (such as Done, InVelocity, InEndVelocity, InGear, InSync, EndOfProfile) are reached, the new motion starts at the velocity of the previous motion. If the previous motion was MC_MoveAbsolute or MC_MoveRelative, the new motion will start in static state.

◆ Start of other instructions during the execution of this instruction

Multiple instances of the instruction can be executed in an interrupted manner for the slave axis.

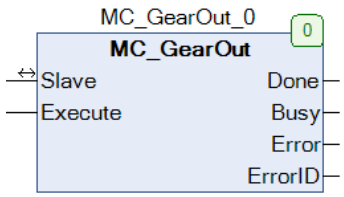
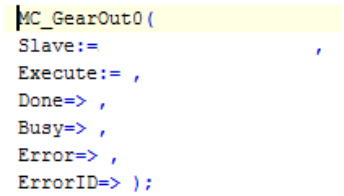
In this case, stop the gear operation and start executing multiple instances.

It is not allowed to execute multiple instances of the instruction in a non-interruptive manner.

MC_GearOut

This instruction aborts the MC_GearIn and MC_GearInPos instructions in execution.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_GearOut	Instruction for canceling electronic gear coupling		


```
MC_GearOut0(
Slave:=                    ,
Execute:= ,
Done=> ,
Busy=> ,
Error=> ,
ErrorID=> );
```

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Slave	Slave axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Executed	BOOL	TRUE, FALSE	FALSE	Execute the instruction at the rising edge

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the electronic gear coupling between the slave axis and the main axis is canceled
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

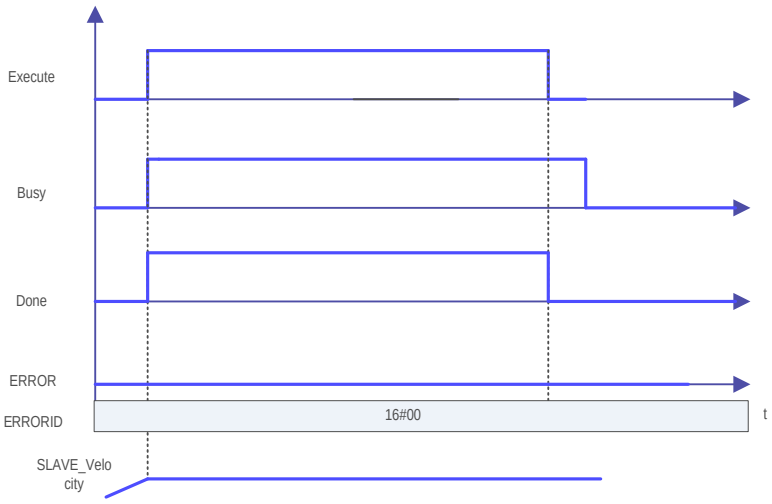
◆ At the rising edge of Execute, execute the action of removing the electronic gearing.

◆ If Execute is TRUE and ERROR is FALSE, then Busy and Done output TRUE.

◆ The slave axis velocity is the same as that one before removal. Therefore, it is necessary to stop the slave axis with the MC_Stop instruction.

◆ At the falling edge of Execute, Done is FALSE.

◆ The MC_Stop instruction resets the Busy signal.



4) Error Description

An error in the parameter setting can cause an alarm.

An alarm will be caused if the axis is not disabled.

Note: For details on the error code, see “ “Appendix C Error Codes” .

MC_GearInPos

This instruction sets the electronic gear ratio between the master axis and the slave axis to perform electronic gearing.

Specify the master axis position, slave axis position, and distance at which synchronization of the master axis starts, and insert the electronic gear based on these values.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
MC_GearInPos	Instruction for specifying the position to insert the electronic gear coupling	MC_GearInPos_0 **MC_GearInPos** EN → ENO Master → StartSync Slave → InSync Execute → Busy RatioNumerator → Active RatioDenominator → CommandAborted MasterSyncPosition → Error SlaveSyncPosition → ErrorID MasterStartDistance BufferMode AvoidReversal	MC_GearInPos(Master:= SM_Drive_Virtual, Slave:= Axis, Execute:= , RatioNumerator:= , RatioDenominator:= , MasterSyncPosition:= , SlaveSyncPosition:= , MasterStartDistance:= , BufferMode:= , AvoidReversal:= , StartSync=> , InSync=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Master	Master axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
Slave	Slave axis	AXIS_REF	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Instruction execution	BOOL	TRUE, FALSE	FALSE	Execute the instruction at the rising edge
RatioNumerator	Gear ratio numerator	DINT	-	1 —	Numerator of master-slave velocity ratio
RatioDenominator	Gear ratio denominator	DINT	-	1	Denominator of master-slave velocity ratio
MasterSyncPosition	Position of the master axis	LREAL	-	-	Master axis position at the time of master-slave gear ratio coupling
SlaveSyncPosition	Sync position of the slave axis	LREAL	-	-	Slave axis position at master-slave gear ratio coupling
MasterStartDistance	Master axis position for sync execution	LREAL	-	-	The slave axis calculates a smooth curve based on this position value and the values MasterSyncPosition and SlaveSyncPosition so that the slave axis is synchronized with the master axis gear at SlaveSyncPosition. The master axis range for the curve is [MasterStartDistance, MasterSyncPosition].
BufferMode	Buffer mode	MC_ BUFFER_MODE	Aborting =0 Buffered=1 BlendingPrevious=3	0	Set to FALSE if reverse running is performed when the physical position of the slave axis is overrun. Set to TRUE if reverse running is not allowed physically or a danger will be caused. It applies only to modulo axes. If reverse running cannot be avoided, the axis will stop with an error.
AvoidReversal	Reverse running inhibited	BOOL	TRUE, FALSE	FALSE	Set to FALSE if reverse running is performed when the physical position of the slave axis is overrun. Set to TRUE if reverse running is not allowed physically or danger will be caused. It applies only to modulo axes. If reverse running cannot be avoided, the axis will stop with an error.

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
StartSync	Coupling processing started	BOOL	TRUE, FALSE	FALSE	Set to TRUE if the electronic gear coupling processing is started
InSync	Coupling in progress	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the electronic gear coupling processing is completed and master-slave gear ratio coupling is in progress.
Busy	Instruction in execution	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
Active	Controlling	BOOL	TRUE, FALSE	FALSE	Set to TRUE when control is being performed
CommandAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Interrupted by other control instructions

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

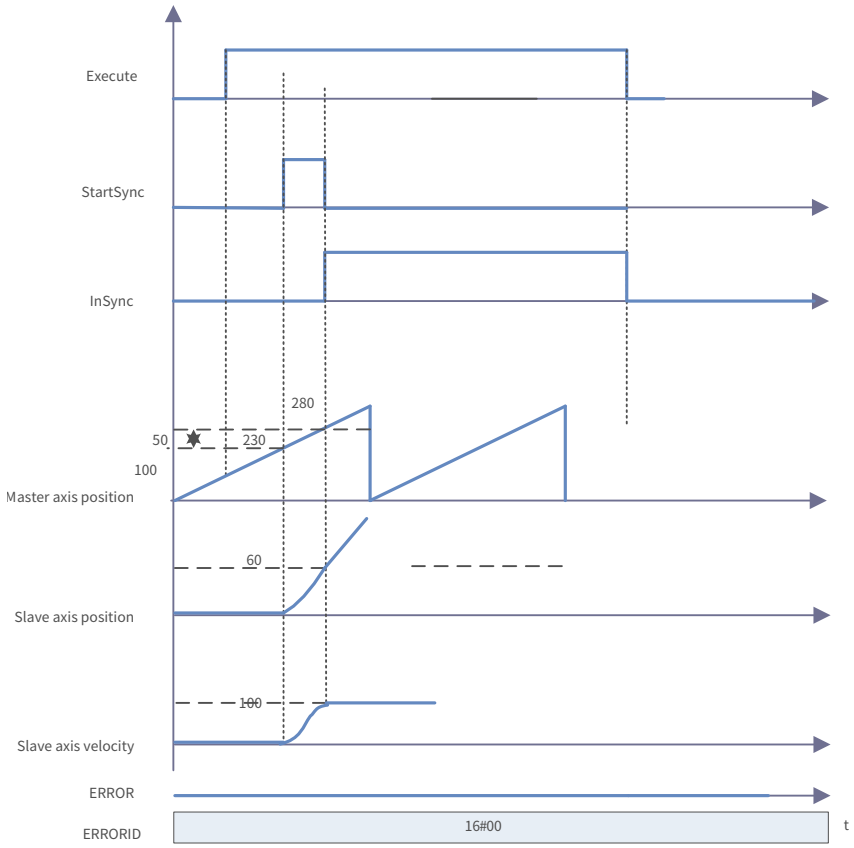
- Specify the object axis through Slave and specify RatioNumerator, RatioDenominator, ReferenceType, Acceleration, and Deceleration to perform gearing.
- The instruction position, feedback position, and latest instruction position can be specified for the master axis (Master).
- Start the instruction at the rising edge of Execute.
- After the start of the action, the slave performs acceleration and deceleration with the target velocity obtained by multiplying the master axis velocity by the gear ratio.
- The whole synchronization process of this function block is essentially an electronic cam, in which the slave axis follows the master axis during the synchronization interval. Based on the range of the master axis (MasterSyncPosition-MasterStartDistance, MasterSyncPosition), the range of the slave axis (Current position, SlaveSyncPosition), as well as the set gear ratio, the instruction automatically designs a cam curve, and the slave axis follows the master axis to complete the cam action during the synchronization.
- Note: If the master and slave axes are working in linear mode, ensure that the above parameters are set properly; otherwise, the gearing action cannot be carried out correctly. Therefore, it is recommended that the master and slave axes work in cyclic mode when this instruction is used.

  For example, when the master and slave axes work in linear mode and both move in positive direction, if Master axis position > MasterSyncPosition - MasterStartDistance or Slave axis position > SlaveSyncPosition when this instruction is executed, then the electronic gearing action cannot be inserted.

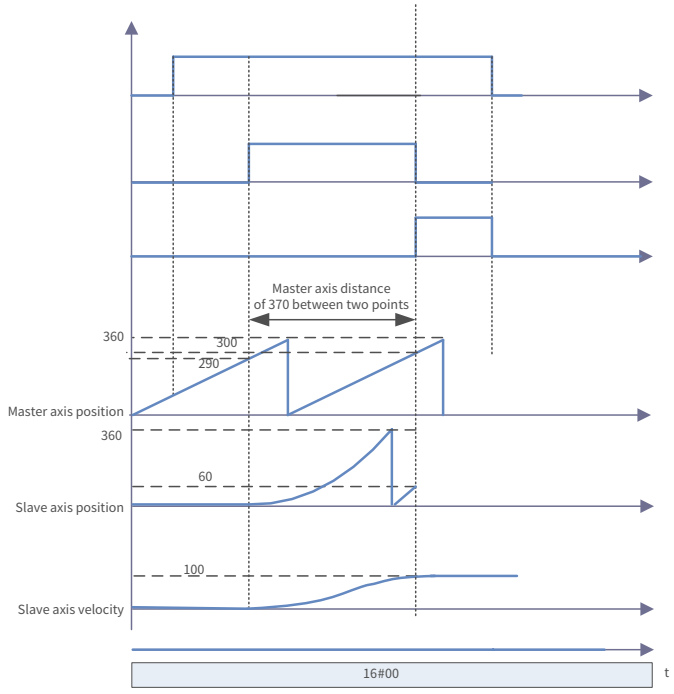
  Sample timing diagrams for different parameters are given below:

  When the master and slave axes work in cyclic mode (360):

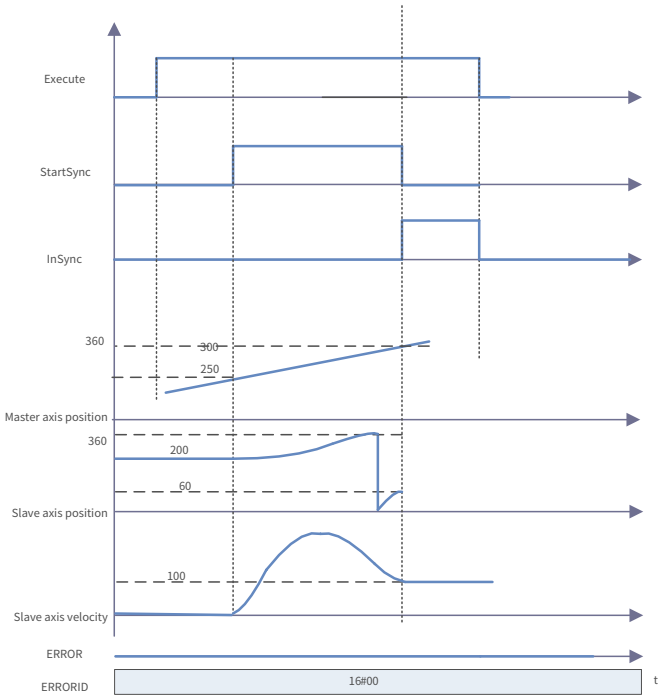
  18) MasterSyncPosition = 280, MasterStartDistance = 50, SlaveSyncPosition = 60, Master axis velocity = 50, AvoidReversal = FALSE



19) MasterSyncPosition = 300, MasterStartDistance = 370, SlaveSyncPosition = 60, Master axis velocity = 50, AvoidReversal = FALSE



20) MasterSyncPosition = 300, MasterStartDistance = 50, SlaveSyncPosition = 60, Master axis velocity = 50, AvoidReversal = FALSE, Slave axis start position > 60



The target velocity will be reached when the synchronization is completed (InSync = TRUE). Slave axis movement amount = Master axis movement amount x RatioNumerator/RatioDenominator

AvoidReversal: If the slave axis is a modulo axis and the master axis velocity (in a multiple relationship with the gear ratio) is not relative to the slave axis velocity, then MC_GearInPos tries to avoid reverse running of the slave axis. It tries to "stretch" the motion of the slave axis by adding 5 slave periods. If this "stretching" is invalid, then an error occurs and the slave axis stops. If the slave axis velocity is related to the major axis velocity (which is a multiple of the gear ratio), then an error occurs and the axis stops. If the slave axis is a linear one, an error is generated at the rising edge of Execute.

4) Precautions

Before using the MC_SetPosition (current position change) instruction for the master axis, cancel the relationship between the master axis and the slave axis.

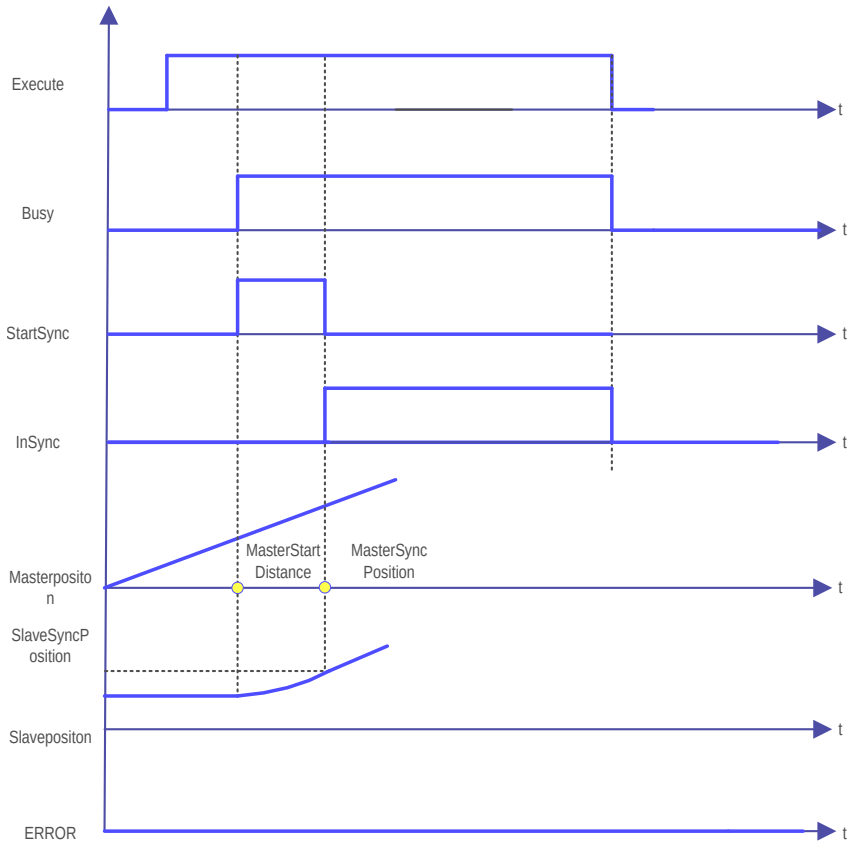
5) Timing Diagram

At the rising edge of Execute, the electronic gearing action starts.

The value of Busy changes to TRUE when Execute is started. After the start of the action, the gearing action is started by Active and StartSync.

When MasterSyncPosition and SlaveSyncPosition are reached, InSync changes to TRUE.

When this instruction is aborted by another instruction, the value of CommandAborted changes to TRUE and those of Busy, Active, StartSync, and InSync change to FALSE.



◆ Motion re-execution instruction

This instruction cannot be re-executed.

◆ Start of this instruction during the execution of other instructions

When this instruction is started for the currently executing instruction, it will be switched or cached to this instruction.

The action when multiple instances of this instruction are started is determined by BufferMode.

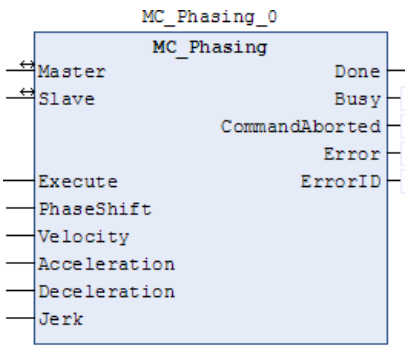
Buffer Mode	Description
Aborting	Immediately aborts the currently executing instruction and switches to this instruction. If the direction of axis motion is reversed due to instruction switching, reverse running is performed after the velocity is decelerated to zero.
Buffered	The function block is started immediately after the last instruction motion is terminated. No blending is performed here. When the end conditions (such as Done, InVelocity, InEndVelocity, InGear, InSync, EndOfProfile) are reached, the new motion starts at the velocity of the previous motion. If the previous motion was MC_MoveAbsolute or MC_MoveRelative, the new motion will start in static state.

MC_Phasing

This instruction specifies the phase shift between the master axis and the slave axis.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression

MC_Phasing	Main-slave axis phase shift		MC_Phasing0(Master:= , Slave:= , Execute:= , PhaseShift:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
Master	Master axis	AXIS_REF_SM3	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3
Slave	Slave axis	AXIS_REF	-	-	Reference to the axis, that is, an instance of AXIS_REF_SM3

◆ Input Variable

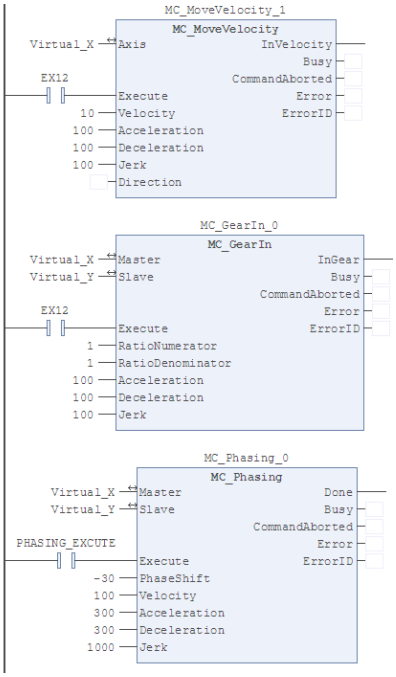
Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Instruction execution	BOOL	TRUE, FALSE	FALSE	Execute the instruction at the rising edge
PhaseShift	Master-slave phase shift	LREAL	-	0	A positive number indicates that the slave axis lags.
Velocity	Velocity	LREAL	-	0	Maximum velocity when the phase shift is executed
Acceleration	Acceleration	LREAL	-	0	Maximum acceleration rate when the phase shift is executed
Deceleration	Deceleration	LREAL	-	0	Maximum deceleration rate when the phase shift is executed.
Jerk	Second derivative of velocity	LREAL	-	0	Maximum jerk when performing phase shift is executed

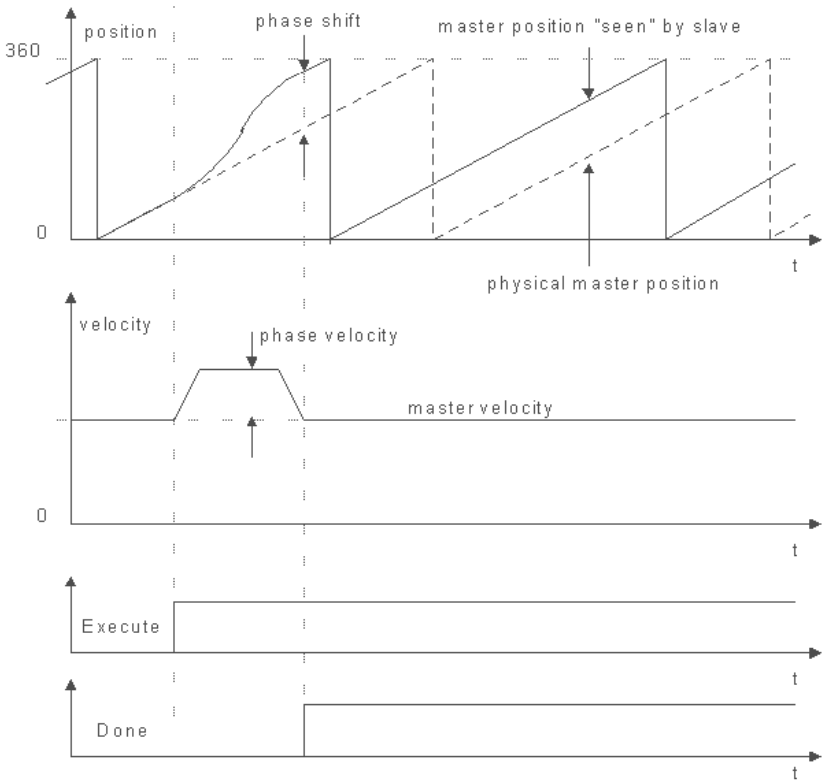
◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the phase shift is completed
Busy	Instruction in execution	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
CommandAborted	Instruction aborted	BOOL	TRUE, FALSE	FALSE	Interrupted by other control instructions
Error	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

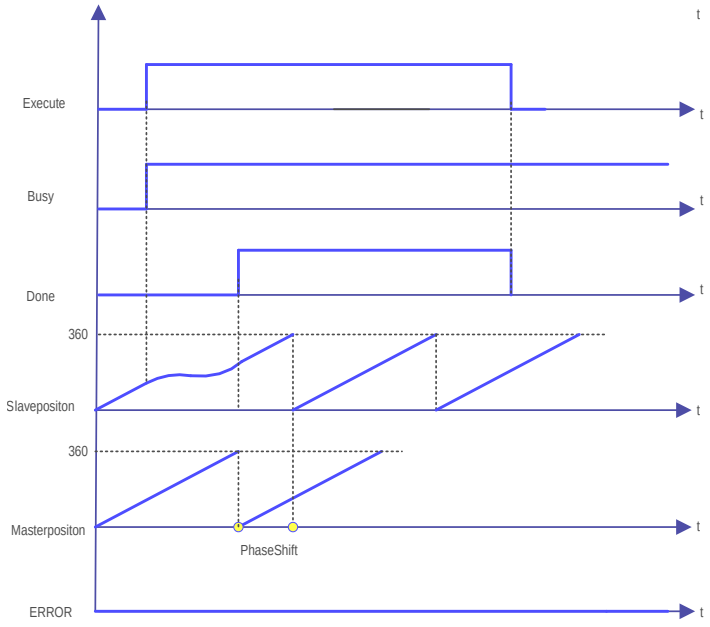
- This instruction executes phase shift at the rising edge of Execute. The slave axis automatically calculates a smooth curve to complete phase shift from the slave axis to the master axis, which is specified by PhaseShift. A positive value indicates that the slave axis lags behind the master axis.
- The Done signal outputs TRUE after phase shift is completed.
- The master-slave phase shift is compensated based on the value of PhaseShift, Velocity, Acceleration and Deceleration.
- When the phase shift between the master axis and the slave axis reaches PhaseShift, the Done signal is output.
- When the instruction is executed, the instruction position and feedback position of the master axis remain unchanged, and the slave axis is adjusted. The phase shift between the slave axis and the master axis is the value of PhaseShift.
- The final result of this instruction is the phase shift between the set values of the axes. Therefore, the actual feedback value of the real axis may not be the same as the final shift.
- This instruction is used in conjunction with the MC_GearIn instruction as follows: The master axis is Virtual_x, and the slave axis is Virtual_y. At the rising edge of EX12, master axis velocity control and master-slave electronic gearing are performed, and then the phase shift is performed. In addition, it can be used in conjunction with an electronic cam. The slave axis acts as an "electronic cam master axis" to achieve the phase shift effect of an electronic cam master axis.





4) Timing Diagram

When the master and slave axes move in 360 cycles, adjustment is performed at the rising edge of the Execute signal. After the adjustment, phase shift between the slave axis and the master axis is the value of PhaseShift.



5) Error Description

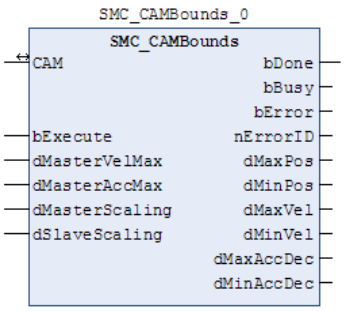
- If the Error outputs TRUE when the instruction is started, an error occurs.
- Check ErrorID and check SMC_ERROR in the help to determine the alarm information. For details on the error code, see “ “Appendix C Error Codes” .

SMC_CAMBounds

This function block calculates the maximum position, velocity, and acceleration rate of the slave axis when the slave axis is cam-coupled to the master axis.

The master axis moves under the input maximum velocity and acceleration/deceleration limits. This instruction can be used to check the correctness of the curves for cam table designs, provided that the maximum acceleration/deceleration rate and velocity are known.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_CAMBounds	Cam upper and lower limits		SMC_CAMBounds_0(CAM:= , bExecute:= , dMasterVelMax:= , dMasterAccMax:= , dMasterScaling:= , dSlaveScaling:= , bDone=> , bBusy=> , bError=> , nErrorID=> , dMaxPos=> , dMinPos=> , dMaxVel=> , dMinVel=> , dMaxAccDec=> , dMinAccDec=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
CAM	Cam	MC_CAM_REF	-	-	Reference to the cam, that is, an instance of MC_CAM_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bExecute	Instruction execution	BOOL	TRUE, FALSE	FALSE	Execute the instruction at the rising edge
dMasterVelMax	Maximum velocity	LREAL	-	1	Maximum master axis velocity in absolute mode
dMasterAccMax	Maximum acceleration	LREAL	-	0	Maximum master axis acceleration in absolute mode
dMasterScaling	Scaling factor	LREAL	-	1	Scaling factor in master axis cam application
dSlaveScaling	Scaling factor	LREAL	-	1	Scaling factor in slave axis cam application

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bDone	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the calculation is completed
bBusy	Instruction in execution	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
nErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs
dMaxPos	Maximum position	LREAL	-	0	Calculate the maximum position of the slave axis according to the cam table
dMinPos	Minimum position	LREAL	-	0	Calculate the minimum position of the slave axis according to the cam table.
dMaxVel	Maximum velocity	LREAL	-	0	Calculate the maximum velocity

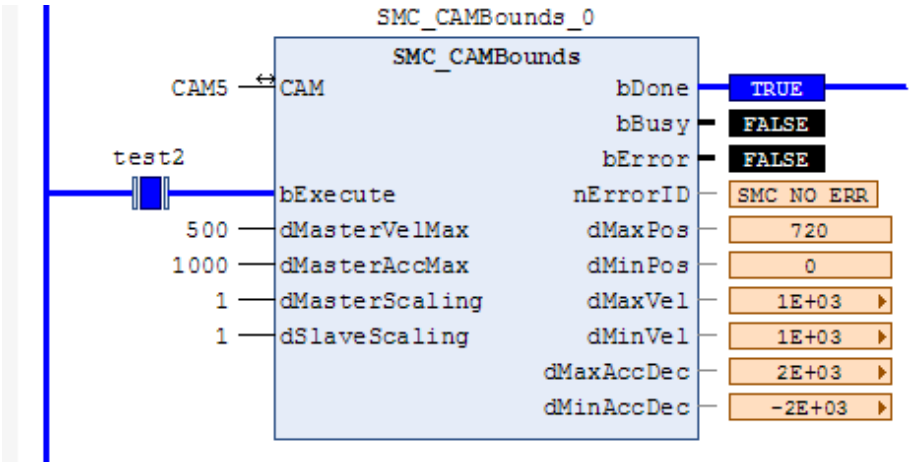
Output Variable	Name	Data Type	Value Range	Initial Value	Description
dMinVel	Minimum velocity	LREAL	-	0	Calculate the minimum velocity
dMaxAccDec	Maximum acceleration	LREAL	-	0	Calculate the maximum acceleration
dMinAccDec	Minimum acceleration	LREAL	-	0	Calculate the minimum acceleration

3) Function Description

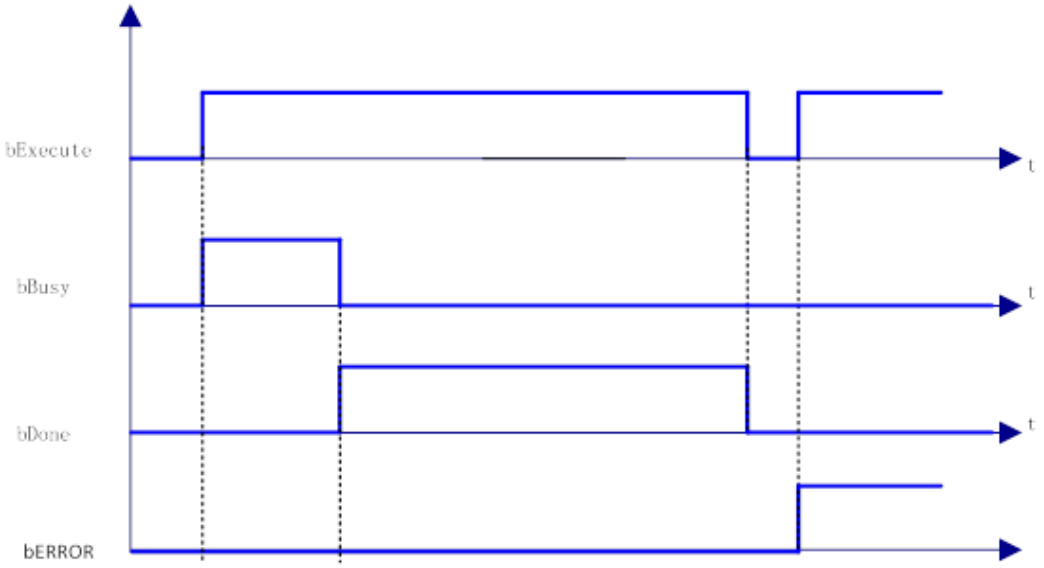
- At the rising edge of bExecute, the "maximum position", "minimum position" and other values of the slave axis are calculated based on dMasterVelMax, dMasterAccMax, dMasterScaling, and dSlaveScaling as well as the cam table data. For example, if the master axis has a period of 360 and the cam table is a straight line with a slope of 2, the result of the calculation is shown in the figure below:

  This instruction can be used when the master axis works in absolute mode, the master axis is set to cyclic mode, or the modulus value is set to the master axis period.

  The cam table is XYVA, which is valid in polynomial mode and not valid for 1D or 2D arrays.



4) Timing Diagram



5) Error Description

The cam table format is not polynomial mode.

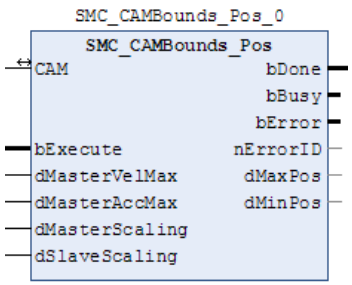
The MC_CAM_REF set value of the cam table does not match the actual cam table.

Note: For details on the error code, see “ “Appendix C Error Codes” .

SMC_CAMBounds_Pos

This function block calculates the maximum and maximum positions of the slave axis when the slave axis is cam-coupled to the master axis. This function block does not calculate the maximum acceleration. Its other functions are the same as those of the SMC_CAMBounds instruction.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_CAMBounds_Pos	Upper and lower cam position limits		SMC_CAMBounds_Pos0(CAM:= , bExecute:= , dMasterVelMax:= , dMasterAccMax:= , dMasterScaling:= , dSlaveScaling:= , bDone=> , bBusy=> , bError=> , nErrorID=> , dMaxPos=> , dMinPos=>);

2) Related Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
CAM	Cam	MC_CAM_REF	-	-	Reference to the cam, that is, an instance of MC_CAM_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
bExecute	Instruction execution	BOOL	TRUE, FALSE	FALSE	Execute the instruction at the rising edge
dMasterVelMax	Maximum velocity	LREAL	-	1	Maximum master axis velocity in absolute mode
dMasterAccMax	Maximum acceleration	LREAL	-	0	Maximum master axis acceleration in absolute mode
dMasterScaling	Scaling factor	LREAL	-	1	Scaling factor in master axis cam application
dSlaveScaling	Scaling factor	LREAL	-	1	Scaling factor in slave axis cam application

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bDone	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the calculation is completed
bBusy	Instruction in execution	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is being executed
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
nErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs
dMaxPos	Maximum position	LREAL	-	0	Calculate the maximum position of the slave axis according to the cam table

dMinPos	Minimum position	LREAL	-	0	Calculate the minimum position of the slave axis according to the cam table.

3) Function Description

- At the rising edge of bExecute, the "maximum position" and "minimum position" of the slave axis are calculated based on dMasterVelMax, dMasterAccMax, dMasterScaling, and dSlaveScaling as well as the cam table data.
- This instruction can be used when the master axis works in absolute mode, the master axis is set to cyclic mode, or the modulus value is set to the master axis period.
- The cam table is XYVA, which is valid in polynomial mode and not valid for 1D or 2D arrays.

4) Error Description

The cam table format is not polynomial mode. The MC_CAM_REF set value of the cam table does not match the actual cam table.

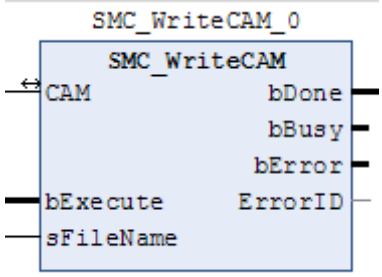
Note: For details on the error code, see " Appendix C Error Codes" .

SMC_WriteCAM

This instruction stores the edited cam table as a file when the program is running. It allows the cam table to be used by instructions such as MC_CamIn. For details on the content of the generated file, see "Cam Format".

This instruction can be used in conjunction with SMC_ReadCAM.

1) Instruction Format

Instruction	Name	Graphic Expression	ST Expression
SMC_WriteCAM	Cam upper and lower limits		

2) Related Variables

- Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
CAM	Cam	MC_CAM_REF	-	-	Reference to the cam, that is, an instance of MC_CAM_REF

- Input Variable

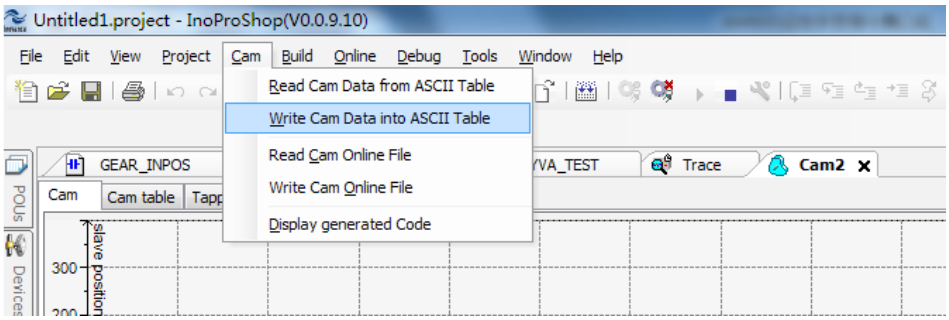
Input Variable	Name	Data Type	Value Range	Initial Value	Description
bExecute	Instruction execution	BOOL	TRUE, FALSE	FALSE	Execute the instruction at the rising edge
sFileName	Document Name	STRING	-	''	File name in ASCII format containing a cam description. For details, see "Cam Format" in Help.

- Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
bDone	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the cam has been written to the file
bBusy	Instruction in execution	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction execution is not completed
bError	Error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
nErrorID	Error code	SMC_ERROR	See SMC_ERROR	0	Output an error code when an error occurs

3) Function Description

- This instruction is executed at the rising edge of bExecute. The cam information of the "Cam" connection is stored in the file connected by the name "sFileName".
- When the storage is complete, the bDone signal outputs TRUE.
- The stored cam table information is limited by the hardware memory.
- Note: This function is executed while the program is running. The cam table information can also be manually stored offline.



4) Error Description

- This instruction can only complete the cam table of XYVA polynomial mode. For 1D or 2D mode, an error will be output.
- The file name connected by "sFileName" does not exist or the information is wrong.

Note: For details on the error code, see " Appendix C Error Codes" .

6.3 Other Functional Specifications

6.3.1 Instruction Cache

1) Aborting and buffered mode

Some function blocks (FBs) have a "BufferMode" input, which enables the FB to work in buffered or non-buffered (default) mode. The difference between the modes lies in when the motion is started.

- Non-buffered mode: The motion instruction takes effect immediately, even if it interrupts another motion. The buffering area for the instruction movement is deleted.
- Buffered mode: The motion instruction waits until the current function block sets its output to Done, InPosition, or InVelocity. Buffered mode is also used to define the velocity curve during motion blending.

Some buffer modes are shown below:

Buffer Mode	Description
Aborting	Default mode without buffering. The function block is started immediately and aborts the active motion. This instruction takes effect immediately for the axis.
Buffered	The function block is started immediately after the last instruction motion is terminated. No blending is performed here. When the end conditions (such as Done, InVelocity, InEndVelocity, InGear, InSync, EndOfProfile) are reached, the new motion starts at the velocity of the previous motion. If the previous motion was MC_MoveAbsolute or MC_ MoveRelative, the new motion will start in static state.
Blend at the low velocity (BlendingLow)	The function block is started immediately after the last instruction motion is terminated. The axis does not stop between motions but passes through the end position of the first motion at the lower velocity of the two motion instructions.
Blend at the previous velocity (BlendingPrevious)	The function block is started immediately after the last instruction motion is terminated. The axis does not stop between motions but passes through the end position of the first motion at the velocity of the first motion instruction.
Blend at the next velocity (BlendingNext)	The function block is started immediately after the last instruction motion is terminated. The axis does not stop between motions but passes through the end position of the first motion at the velocity of the second motion instructions.
Blend at the high velocity (BlendingHigh)	The function block is started immediately after the last instruction motion is terminated. The axis does stop between motions but passes through the end position of the first motion at the higher velocity of the two motion instructions.

2) Impact of buffering modes on defined function blocks

Function Block	Defined as a Buffering/ Blending Instruction	Follow a Buffering/ Blending Instruction	Signal for Activating the Buffering/Blending FB
MC_Power	No	No	-
MC_Home	No	No	-
MC_Stop	No	No	-
MC_Halt	No	No	-
MC_MoveAbsolute	Yes	Yes	Done
MC_MoveRelative			
MC_MoveAdditive	No	No	-
MC_MoveSuperimposed	No	No	-
MC_MoveVelocity	Yes	Yes (Buffered only)	InVelocity
SMC_ MoveContinuousAbsolute	No	Yes (Buffered only)	InEndVelocity
SMC_MoveContinuousRelative			
MC_PositionProfile	No	No	-
MC_VelocityProfile			
MC_AccelerationProfile			
MC_CamIn	No	Yes, also if periodic (only Buffered)	EndOfProfile
MC_CamOut	No	Yes (Buffered only)	Done
MC_GearIn	Yes (BlendingPrevious only)	Yes (Buffered only)	InGear
MC_GearOut	No	Yes (Buffered only)	Done
MC_GearInPos	No	Yes (Buffered only)	InSync
SMC_FollowPosition	No	No	-
SMC_FollowVelocity			
SMC_FollowPositionVelocity			
SMC_FollowSetValues			
SMC_SetTorque	No	Yes	-

Function Block	Defined as a Buffering/ Blending Instruction	Follow a Buffering/ Blending Instruction	Signal for Activating the Buffering/Blending FB
MC_Phasing	No	No	-
MC_Jog	No	Yes (Buffered only)	Busy
SMC_Inch		Yes (Buffered only)	Busy
SMC_BacklashCompensation	No	No	-

3) Execution order of cached function blocks

In buffered motion or blending motion mode, the FB instances of the next instruction motion must not be executed earlier than the FB instance of the previous instruction motion (the execution order in the main program). If this rule is violated, a new error SMC_MOVING_WITHOUT_ACTIVE_MOVEMENT will be reported and the axis will switch to the Errorstop status.

4) Specific features of the mixed state

The buffering mode does not change the drive position characteristics. Rules of its valid blending velocity are as follows:

- If the blending velocity cannot be reached (without position overshoot), the valid blending velocity is the next velocity that can be reached (without overshoot).

[Note]: The valid blending velocity can be higher or lower than the blending velocity.

- If the second motion instruction starts in a direction opposite to that of the first motion instruction, the valid blending velocity is set to 0. This prevents the position from going beyond its target position in the direction of the first motion.
- If the path of the second motion is too short to decelerate from blending velocity to 0, the valid blending velocity will be adjusted. It is set to the maximum velocity that is allowed for safe braking to a standstill status on the path of the second motion.
- In the case of a rotary axis, the result of the input direction of MC_MoveAbsolute is not affected by blending to the second motion. This means that the target position of the first motion is always in the same modulus period, regardless of whether it follows the blending motion.
- In the case of a rotary axis and a second motion of the MC_MoveAbsolute type, the blending velocity does not affect the modulus period of the target position of the second motion instruction when Direction = Fastest. This means that the target of the same period will be selected, regardless of whether the second motion instruction uses buffered or blending mode.

5) Precautions for buffering mode

- An instruction with a buffering area cannot be repeatedly triggered in the buffering area (during execution in the non-aborting status). It can be repeatedly triggered if not in the buffering area (in aborting status, when the instruction is complete, or in non-active status in Busy mode). That is, there can only be one buffer instance with the Buffered/Blending function block.
- During the execution of the buffer instruction, if the motion parameters are modified, there is no impact on the original instruction. If the parameters are modified and re-triggered, only the modification to aborting mode is supported.

6.3.2 Hitting Limit

21) Determination of hitting limit; processing rule for hitting the negative limit in positive direction: For

positive motion, only the positive limit is judged, and an error is reported when the motion goes beyond the positive limit. For negative motion, only the negative limit is judged, and an error is reported when the motion goes beyond the negative limit.

22) Rules of changes in the status bit, axis status, and instruction output flag bit of hitting limit: When the axis starts to decelerate upon software limit, it directly switches into the ErrorStop status and the instruction enters the Error status.

23) Processing rule for hitting the limit switch: The instruction determines whether it will cross the limit during the movement. If the trajectory of the current instruction will decelerate upon software limit only after crossing the limit, then it will interrupt the current controlled instruction within the limit and finally stop at the limit boundary through the deceleration parameter set in the background.

24) Stopping rule for hitting the software limit: If fSwLimitDeceleration is smaller than fSWMaxDeceleration, it will stop based on fSWMaxDeceleration. If the distance of stopping based on the maximum deceleration rate is larger than fSWErrorDistance, it will stop based on fSWErrorDistance. The current velocity and position to the limit are calculated based on the software maximum deceleration (fSWMaxDeceleration), software limit maximum deceleration (fSwLimitDeceleration) and deceleration distance (fSWErrorDistance). The parameter for deceleration will be the one that allows the minimum deceleration distance (maximum deceleration rate), among the maximum software limit deceleration, software limit deceleration and the maximum software limit deceleration. The software limit deceleration process is a T-curve, that is, after deceleration upon soft limit is triggered, the acceleration rate jumps directly to the software limit deceleration rate/maximum software limit deceleration rate/deceleration rate calculated based on the maximum software limit deceleration distance.

25) If the initial position is beyond or on the limit and the axis moves in the direction of the limit, the axis processing logic and function block output flag bit change as follows: the axis directly enters the ErrorStop status and the instruction is set to Error.

26) Added an option to make the axis that hits the limit not enter the ErrorStop status

For the above item 2/5, when the axis enters the software limit, it will switch to the ErrorStop status and the instruction enters the Error status. The option Axis.bSWLimitNotErrorStopEnable is added so that:

If Axis.bSWLimitNotErrorStopEnable is TRUE, when the axis starts to decelerate upon software limit, there is an option of not reporting an error, that is, the axis will be in non-ErrorStop status, making it possible to reverse the movement away from the limit when a new instruction is triggered. If Axis. bSWLimitNotErrorStopEnable is FALSE, the function of the above item 2 is kept.

Similarly for item 5, if Axis.bSWLimitNotErrorStopEnable is TRUE, when the initial position is beyond or on the limit and the axis moves in the direction of the limit, there is an option of not reporting an error, that is, the axis will be in non-ErrorStop status, making it possible to reverse the movement away from the limit when a new instruction is triggered. If Axis.bSWLimitNotErrorStopEnable is FALSE, the function of the above item 5 is kept.

6.3.3 Defaults of Motion Control Function Blocks

The default values of the motion variable input limits, that is, the velocity limit, acceleration limit, deceleration limit and jerk limit, for any current motion instruction are all 0. However, during the execution of the instruction, none of the limit input values can be 0. When the input values are not assigned, the default values will be adopted. In this case, if the instruction is triggered directly, an error will be reported.

Users can avoid the error that is reported when the limit value of the motion variable input of an instruction is 0 or less than 0. That is, if the parameter Velocity, Acceleration, Deceleration or Jerk for the motion control instruction exceeds the value range, a new default value will be used, which is in the

structure stDynamicDefault.

stDynamicDefault description:

Structure	Element	Data Type	Default value	Description
stDynamicDefault	fDefaultVelocity	LREAL	10	New default value of velocity
	fDefaultAcceleration	LREAL	100	New default value of acceleration rate
	fDefaultDeceleration	LREAL	100	New default value of deceleration rate
	fDefaultJerk	LREAL	10000	New default value of jerk

The following instructions are involved:

Instruction Involved	Velocity	Acceleration	Deceleration	Jerk
MC_MoveAbsolute	Velocity < 0	Acceleration rate ⩽ 0	Deceleration rate ⩽ 0	Jerk ⩽ 0
MC_MoveAdditive	Velocity < 0	Acceleration rate ⩽ 0	Deceleration rate ⩽ 0	Jerk ⩽ 0
MC_MoveRelative	Velocity < 0	Acceleration rate ⩽ 0	Deceleration rate ⩽ 0	Jerk ⩽ 0
MC_MoveSuperImposed	Velocity < 0	Acceleration rate ⩽ 0	Deceleration rate ⩽ 0	Jerk ⩽ 0
MC_MoveVelocity	Velocity < 0	Acceleration rate ⩽ 0	Deceleration rate ⩽ 0	Jerk ⩽ 0
SMC_MoveContinuousAbsolute	Velocity < 0	Acceleration rate ⩽ 0	Deceleration rate ⩽ 0	Jerk ⩽ 0
SMC_MoveContinuousRelative	Velocity < 0	Acceleration rate ⩽ 0	Deceleration rate ⩽ 0	Jerk ⩽ 0
MC_Jog	Velocity < 0	Acceleration rate ⩽ 0	Deceleration rate ⩽ 0	Jerk ⩽ 0
SMC_Inch	Velocity < 0	Acceleration rate ⩽ 0	Deceleration rate ⩽ 0	Jerk ⩽ 0
MC_Halt	-	-	Deceleration rate ⩽ 0	Jerk ⩽ 0
MC_Stop	-	-	Deceleration rate ⩽ 0	Jerk ⩽ 0

The default values will be assigned to the input variables of the above instructions within the ranges shown in the table. However, there are two special cases:

- If the velocity is 0 in the position-related instruction but the desired displacement is not 0, an error will be reported.
- If the velocity at the end of continuous motion is smaller than 0, the end velocity is set to 0.

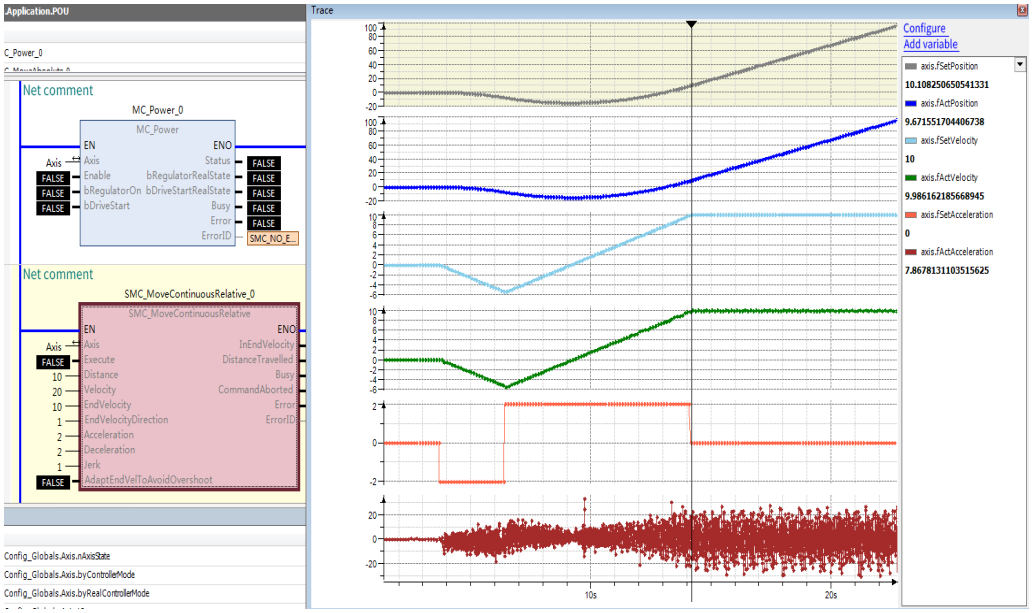
6.3.4 Curve Reversal Prevention

- If Axis.bCurveInvertedEnable is TRUE, the current motion is interrupted by another motion, and the displacement generated in the transition process between the velocity at the interruption and the target velocity is larger than the relative displacement of the motion to the target position, the deceleration rate will be automatically adjusted to avoid curve reversal.
- If Axis.bCurveInvertedEnable is FALSE, the original reversal phenomenon will be maintained.

When the target displacement is very small and the difference between the breakpoint velocity and the target velocity is large, the velocity demand may not be satisfied at the target displacement because the deceleration rate is too small, resulting in velocity reversal. The reversal is not acceptable in many situations. Examples are shown below:

The parameters of continuous relative motion are as follows: initial velocity (breakpoint velocity) = 0, end velocity = 10, target displacement when the end velocity is reached = 10 Based on this velocity and acceleration/deceleration rate, the running displacement when the end velocity is reached in the fastest

manner is $\frac{v^2}{2*2}=\frac{10^2}{4}=25$, which is greater than the target displacement of 10. In this case, there will be a reversal, as shown in the figure:

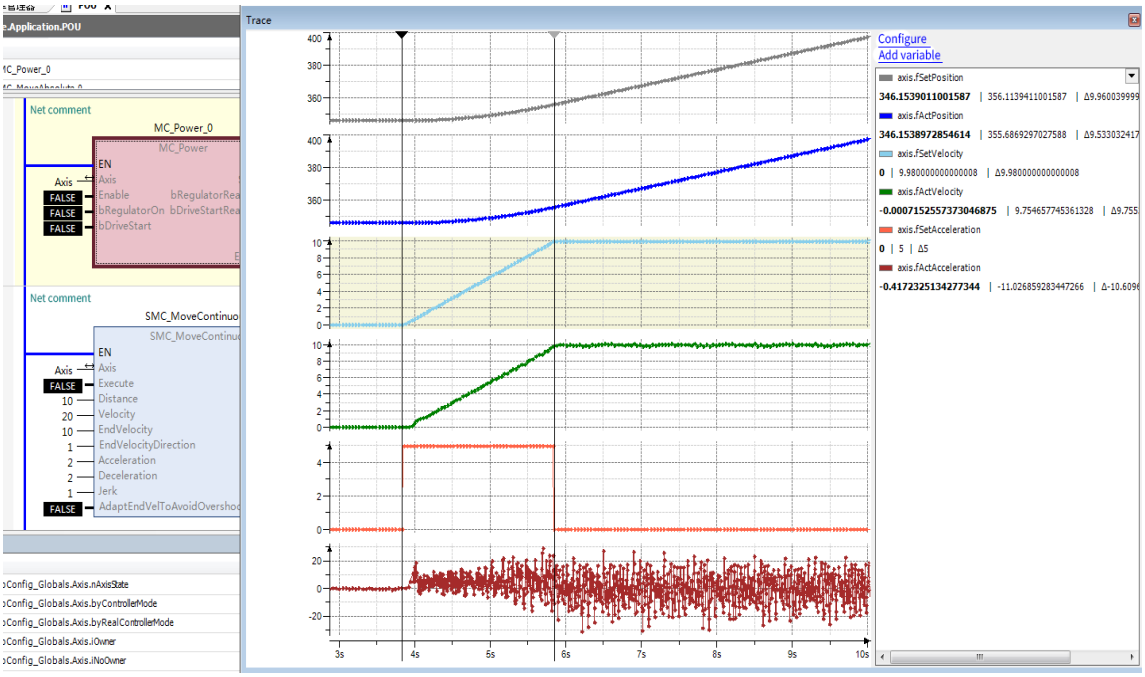


That is, if the straight drop displacement of the initial and end velocity at the breakpoint is larger than the total relative displacement, then there will be velocity reversal after the interruption.

To avoid the reversal, consider optimization within the algorithm to automatically change the deceleration rate and take the smallest acceleration/deceleration value.

Examples are shown below:

Similar to the above case, it is known that the straight drop displacement is larger than the total relative displacement. If bCurveInvertedEnable is set to TRUE, the acceleration rate will be automatically adjusted as follows: $\frac{v^2}{2*A}=\frac{10^2}{2*A}=10$. As $A=5$, the velocity reversal is avoided, as shown in the figure.

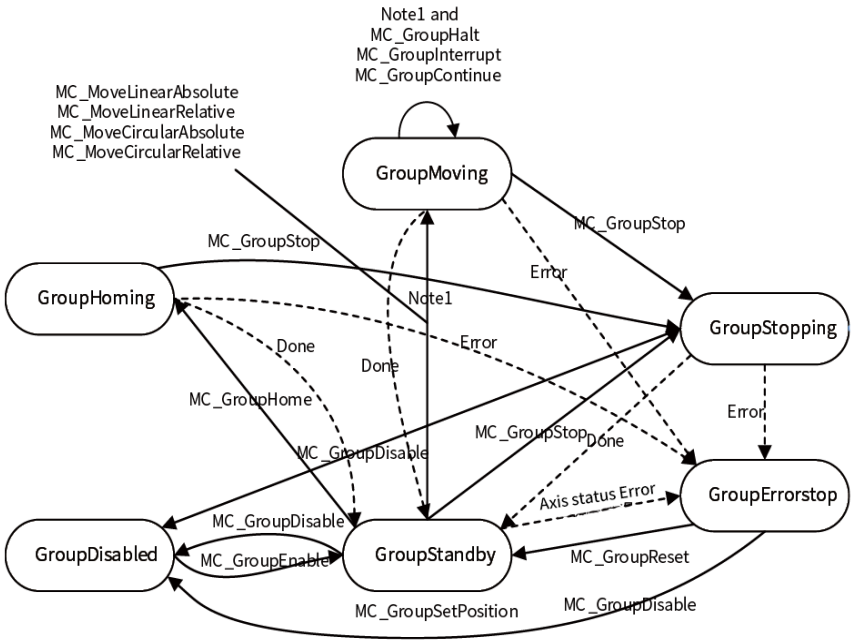


6.4 Axis Group Instructions

1) Instruction List

Instruction Category	Name	Function
Axis Group Instructions	MC_GroupEnableV2	Enabling an axis group
	MC_GroupDisableV2	Disabling an axis group
	MC_ GroupHomeV2	Performing axis group homing
	MC_GroupSetPositionV2	Setting the current position of an axis group
	MC_GroupReadActualPositionV2	Reading the actual position of an axis group
	MC_GroupReadActualVelocityV2	Reading the actual velocity of an axis group
	MC_GroupReadActualAccelerationV2	Reading the actual acceleration rate of an axis group
	MC_GroupStopV2	Stopping an axis group immediately
	MC_GroupHaltV2	Stopping an axis group
	MC_GroupSetOverrideV2	Setting the axis group ratio
	MC_MoveLinearRelativeV2	Linear relative motion
	MC_MoveLinearAbsoluteV2	Linear absolute motion
	MC_MoveCircularRelativeV2	Arc relative motion
	MC_MoveCircularAbsoluteV2	Arc absolute motion
	MC_GroupResetV2	Resetting an axis group
	MC_GroupReadErrorV2	Reading the axis group error message
	MC_GroupReadStatusV2	Reading the axis group status
	MC_GroupInterruptV2	Pausing an axis group
	MC_GroupContinueV2	Restarting an axis group

2) Axis group state machine



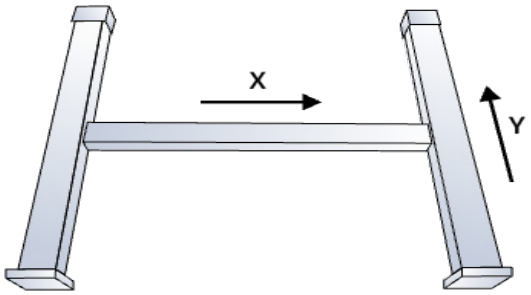
- Before switching the axis group state from GroupDisabled to GroupStandby, make sure all axes in the axis group are mounted to the axis group, and all axes in the axis group have been enabled successfully without error messages.
- If an axis in the axis group has an error message when the axis group is in the GroupStandby state, the axis group switches to the GroupErrorStop state.

- To switch the axis group state from GroupErrorStop to GroupStandby, you need to call the MC_GroupReset instruction to clear errors of the axis group and errors of axes in the axis group. If the errors still exist, state switching is not allowed.
- To switch the axis group state from GroupStandby to GroupMoving or GroupHoming state, make sure each configured axis in the axis group is enabled and is in the Synchronized_Motion or Standstill state. Otherwise, an error will be generated when you call the axis group motion instruction. However, it does not affect the axis group state machine.
- When the axis group is in the GroupStopping state, you need to reset the output variable Done of the MC_GroupStop instruction. When the input variable Execute is reset, the axis group switches to the GroupStandby state.
- After the axis group is enabled, the axes in the axis group cannot use single-axis instructions (except the single-axis immediate stop instruction).
- The MC_GroupSetPosition instruction for setting the current position of an axis group can only be called when the axis group is in the GroupStandby or GroupMoving state and the motion is idle.
- The MC_GroupDisable instruction can be used to switch the axis group state from GroupErrorstop to GroupDisabled.
- The GroupErrorstop state does not affect the state of a single axis in an axis group.

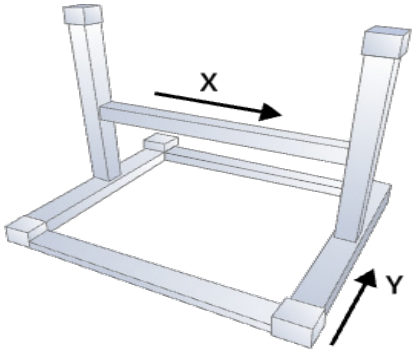
3) Axis group types

- Diagrams of axis group types

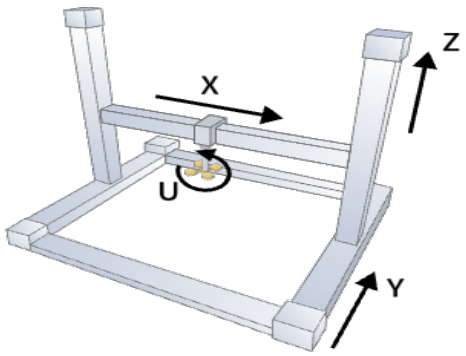
27) XY type



28) XYZ type

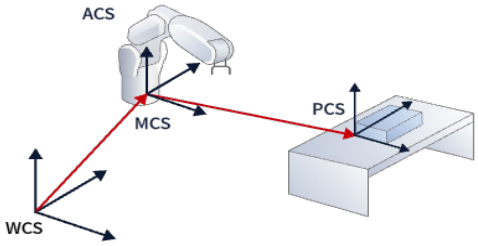


29) XYZU type



- Diagram of axis group coordinate system

The core function of an axis group is coordinate motion. This section specifies the coordinate systems associated with the axis group motion. The coordinate systems (CoordSystem) involved in the axis group are shown below:



According to the characteristics of axis group motion, the coordinate system types include Axis Coordinate System, Machine Coordinate System, Part Coordinate System, and World Coordinate System, as listed in the table below:

Coordinate System	Meaning	Description
Axis Coordinate System	Axis Coordinate System (ACS)	Axis-related, indicating coordinates of each joint
Machine Coordinate System	Machine Coordinate System (MCS)	Axis group base coordinate system, axis group relative to the World Coordinate System
Part Coordinate System	Part Coordinate System (PCS)	Workspace
World Coordinate System	World Coordinate System (WCS)	Base space common to all Machine Coordinate Systems

4) Variable

AXES_GROUP_REF: Axis group structure

Variable	Type	Description
iId	E_GROUP_ID	Axis group type identifier
iNo	DWORD	Axis group number
bDisableErrorLogging	BOOL	Print log information
fbeFBError	ARRAY[0..5] OF MC_FBERROR	Instruction error message

Variable	Type	Description
bError	BOOL	Error state
iErrorID	DWORD	Error ID
eState	E_AXES_GROUP_STATE	Axis group state machine
bIdle	BOOL	Idle state identifier
stCmd	ARRAY [1..2] OF S_PNT	[ACS/PCS] coordinate position, velocity, and acceleration rate of the current instruction
stFrontCmd	ARRAY [1..2] OF S_PNT	[ACS/PCS] coordinate position, velocity, and acceleration rate of the instruction in the previous period
stActual	ARRAY [1..2] OF S_PNT	Current real-time [ACS/PCS] coordinate position, velocity, and acceleration rate
stFrontActual	ARRAY [1..2] OF S_PNT	Real-time [ACS/PCS] coordinate position, velocity, and acceleration rate of the last period
fActualPathVel	LREAL	Main dimension subspace resultant velocity of the real-time TCP point (PCS)
fActualPathAcc	LREAL	Main dimension subspace resultant acceleration of the real-time TCP point (PCS)

E_AXES_GROUP_STATE: Axis group state machine enumeration type

Variable	Value	Description
GROUP_ERRORSTOP	0	Axis group stopped due to an error
GROUP_DISABLED	1	Axis group disabled
GROUP_STOPPING	2	Axis group stopped immediately
GROUP_STANDBY	3	Axis group standby
GROUP_MOVING	4	Axis group in motion
GROUP_HOMING	5	Motion homing

E_COORD_TYPE: Curve profile type enumeration type

Variable	Value	Description
TRAPEZOIDAL_CURVE	0	Trapezoidal profile
CUBIC_CURVE	1	Cubic 7-segment S-curve
SEVENTH_CYRVE	2	Septuple 7-segment S-curve

E_ERROR_STOP: Stop mode enumeration type

Variable	Value	Description
T_ERROR_STOPPED	0	Error stopped
T_ERROR_STOPPING	1	Stopping error
T_ERROR_STOP	2	Error stop activated

E_UNIT_SPACE: User main dimension enumeration type

Variable	Value	Description
UNIT_TRAJECT	0	Position as user main dimension
UNIT_ORIENT	1	Orientation as user main dimension
UNIT_PERCENT	2	Uniaxial scale factor as user main dimension orientation

MC_OUTMODE: Detachment mode enumeration type

Variable	Value	Description
mcDecelerationStop	0	Position as user main dimension
mcConstantVelocity	1	Orientation as user main dimension
mcImmediateStop	2	Uniaxial scale factor as user main dimension orientation

MC_BUFFER_MODE_2: Axis group buffer mode enumeration type

Variable	Value	Description
MC_ABORTING	0	Aborting mode
MC_BUFFEREN	1	Buffer mode
MC_BLEND_LOW	2	Blend at the low velocity
MC_BLEND_PRE	3	Blend at the previous velocity
MC_BLEND_NEXT	4	Blend at the next velocity
MC_BLEND_HIGH	5	Blend at the high velocity

MC_TRANSITION_MODE_V2: Transition mode enumeration type

Variable	Value	Description
TM_NONE	0	No insertion of transition contour lines
TM_START_VELOCITY	1	Transition with the set starting velocity
TM_CORNER_DISTANCE	3	Transition with the set corner distance
TM_MAX_CORNER_DEVIATION	4	Transition with the set maximum corner deviation
TM_DEFINED_VELOCITY	10	Transition with the maximum velocity of the current instruction
TM_RESET_ABORT	11	Reset aborting transition mode
TM_DEC_ABORT	12	Deceleration aborting transition mode

MC_CIRCMODE_V2: Arc mode enumeration type

Variable	Value	Description
BORDER_ARC	0	Three-point arc
CENTER_ARC	1	Center arc
RADIUS_ARC	2	Radius arc

MC_COORDSYSTEM_2: Coordinate system enumeration type

Variable	Value	Description
ACS	0	Axis Coordinate System
PCS	1	Machine Coordinate System

E_STOP_MODE_V2: Stop mode enumeration type

Variable	Value	Description
DEC_STOP	0	Decelerate to stop
IMMED_STOP	1	Stop immediately

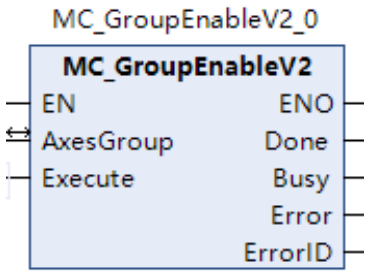
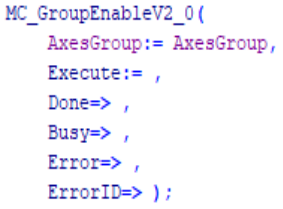
MC_CIRC_PATHCHOICE_V2: Axis group type library

Variable	Value	Description
SHORTPART	0	Short arc path
LONGPART	1	Long arc path

6.4.1 MC_GroupEnableV2

This instruction enables an axis group by switching its state from GroupDisabled to GroupStandby.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_ GroupEnableV2	Enabling an axis group		


```
MC_GroupEnableV2_0(
    AxesGroup:= AxesGroup,
    Execute:= ,
    Done=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
```

2) Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ This instruction switches the axis group state specified by the input/output variable AxesGroup from GroupDisabled to GroupStandby.

◆ When the axis group is in the GroupStandby state, other axis group instructions can be executed.

◆ This instruction switches the state of a single axis (Axis) in the axis group specified by the input/output variable AxesGroup to SYNCHRONIZED_MOTION.

4) Precautions

◆ This instruction reports an error when there is an unconfigured single axis (Axis) in the axis group specified by AxesGroup. Please configure a single axis (Axis) for the axis group specified by AxesGroup as required.

- When this instruction is enabled, the single axis (Axis) in the axis group specified by AxesGroup of this instruction must be in the off (PowerOff) or standby (StandStill) state.
- If a single axis (Axis) in the axis group specified by AxesGroup of this instruction is in the ErrorStop state when this instruction is enabled, the error code of the axis will be output.
- A single axis (Axis) can be added to different axis groups (AxesGroup), but only one axis group (AxesGroup) can be enabled. If an axis group (AxesGroup) configured with this axis is already enabled, other axis groups (AxesGroups) configured with this axis cannot be enabled until the axis group (AxesGroup) is disabled by using the MC_GroupDiable instruction.
- If the output variable Done changes to TRUE, the axis group specified by AxesGroup is enabled. In this case, it is not allowed to execute a single-axis instruction for the single axis in the axis group (AxesGroup).

5) Timing Diagram

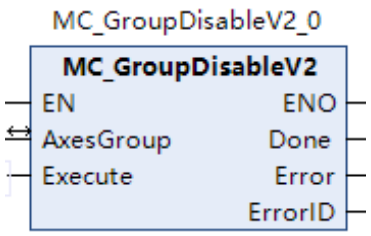
6) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.2 MC_GroupDisableV2

This instruction disables an axis group by switching its state to GroupDisabled.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_ GroupDisableV2	Disabling an axis group		MC_GroupDisableV2_0(AxesGroup:= AxesGroup, Execute:= , Done=> , Error=> , ErrorID=>);

2) Variables

- Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

- Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge

- Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description

Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

- This instruction disables the axis group specified by the input/output variable AxesGroup by changing its state to GroupDisabled.
- If this instruction is executed when the axis group (AxesGroup) is in the GroupDisabled state, the output variable Done is set to TRUE immediately.
- When the axis group (AxesGroup) is in the GroupDisabled state, a single-axis instruction can be executed for the single axis (Axis) in the axis group (AxesGroup).
- This instruction does not affect the state of the single axis (Axis) in the axis group specified by the input/output variable AxesGroup. If the axis group is in the GroupStandby state, the single axis in the axis group will be switched to the Standstill state.

4) Precautions

- This instruction can be executed when the axis group (AxesGroup) is in the GroupStandby, GroupDisabled, or GroupErrorStop state and cannot be used in other states. Otherwise, an error will be reported.
- After this instruction is executed, a single-axis instruction can be executed for the axis in the axis group (AxesGroup).

5) Timing Diagram

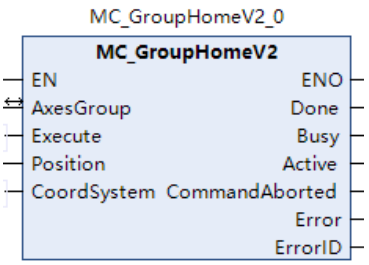
6) Error description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.3 MC_GroupHomeV2

This instruction enables an axis group to return to the mechanical zero position.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_ GroupHomeV2	Performing axis group homing		MC_GroupHomeV2_0(AxesGroup:= AxesGroup, Execute:= , Position:= , CoordSystem:= , Done=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Variables

- Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge
Position	Position	ARRAY[0···gc_iAxis_Num-1] OF LREAL	-	[0,0,0···]	Absolute coordinate of the mechanical zero position
CoordSystem	Coordinate system	MC_COORDSYSTEM_V2COORDSYSTEM_V2	ACS	ACS	Select the coordinate system type

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Active	Axis group under control of the instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Command Aborted	Instruction execution aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ This instruction is started at the rising edge of Execute and enables the axis group (AxesGroup) to return to the mechanical zero position.

◆ This instruction defines the homing sequence for each axis in the axis group (AxesGroup).

· No descending order is allowed in the defined homing sequence, and the homing number of the first axis must be 1.

· For example, if the axis group type is XYZA, the homing sequence of the axis group can be defined as “1, 1, 1, 1”, “1, 2, 3, 4”, “1, 2, 2, 2”, “1, 2, 2, 3”, or “1, 2, 3, 3”.

· Currently, the homing sequence can be defined in the axis group interface and the axis group structure.

◆ This instruction cannot be triggered again.

4) Precautions

◆ This instruction can be executed only when the axis group (AxesGroup) is in the GroupStandby state. It will report an error when being executed in other states.

◆ This instruction can be interrupted only by the MC_GroupStop instruction.

◆ This instruction supports homing only under the ACS.

◆ Currently, only servo homing is supported. Other homing modes will be developed later.

5) Timing Diagram

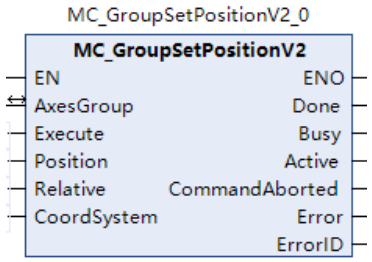
6) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.4 MC_GroupSetPositionV2

This instruction sets the current position of each axis in the axis group.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_ GroupSetPositionV2	Setting the current position of an axis group		MC_GroupSetPositionV2_0(AxesGroup:= AxesGroup, Execute:= , Position:= , Relative:= , CoordSystem:= , Done=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge
Position	Position	ARRAY[0···gc_iAxis_Num-1] OF LREAL	-	[0,0···]	Position [u] in the corresponding coordinate type
Relative	Enabling relative coordinates	BOOL	TRUE, FALSE	FALSE	RELATIVE=TRUE; ABSOLUTE=FALSE
CoordSystem	Coordinate system type	MC_COORDSYSTEM_V2	ACS, PCS	ACS	Select the coordinate system type

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received

Active	Axis group under control of the instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Command Aborted	Instruction execution aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

- This instruction specifies the current position of each axis in the axis group by setting the input/output variable AxesGroup.
- You can set the current position of each axis (Axis) in the axis group (AxesGroup) under the ACS or PCS and update the axis offset or part coordinate system without changing the part coordinate system type. You can also set the position in the dynamic coordinate system.
- Relative (enabling relative coordinates)

  You can select how to set the position by setting the input variable Relative of this instruction.

  Relative=FALSE: Absolute coordinate system. After this instruction is executed, the actual position of each axis (Axis) in the axis group (AxesGroup) equals the value set by the input parameter Position of this instruction.

  Relative=TRUE: Relative coordinate system. The actual position of each axis (Axis) in the axis group (AxesGroup) after this instruction is executed equals the sum of the value set by the input parameter Position of this instruction and the actual position of each axis (Axis) in the axis group (AxesGroup) before this instruction is executed.

4) Precautions

- When this instruction is executed, the axis group specified by the input/output variable AxesGroup must be in the GroupStandby or GroupMoving state and the value of Idle must be TRUE. There is no requirement on the state of axes (Axis) in the axis group (AxesGroup).
- Setting the position under the ACS will affect the position of the axis group under the PCS. Setting the position under the PCS will not affect the position of the axis group under the ACS.

5) Timing Diagram

6) Error Description

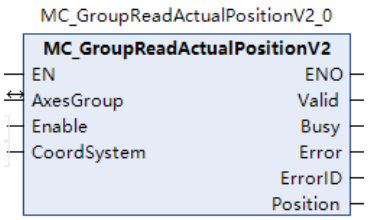
Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.5 MC_GroupReadActualPositionV2

This instruction reads the actual position of an axis group in the selected coordinate system.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression

MC_GroupRead ActualPositionV2	Reading the actual position of an axis group		MC_GroupReadActualPositionV2_0(AxesGroup:= AxesGroup, Enable:= , CoordSystem:= , Valid=> , Busy=> , Error=> , ErrorID=> , Position=>);

2) Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Instruction execution	BOOL	TRUE, FALSE	FALSE	TRUE: Execute the instruction; FALSE: Do not execute the instruction
CoordSystem	Coordinate system type	MC_COORDSYSTEM_V2	ACS, PCS	ACS	Select the coordinate system type

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Valid instruction output	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs
Position	Position	ARRAY[1…N]OF LREAL	-	[0,0,…]	Return the current position [u] of the axis group
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ When Valid is TRUE, this instruction obtains the actual position of all single axes (Axis) in the axis group specified by the input/output variable AxesGroup of this instruction in the selected coordinate type (CoordSystem) based on the control cycle.

4) Precautions

◆ When this instruction is executed, the axis group specified by the input/output variable AxesGroup must not be in the GroupDisabled state. There is no requirement on the state of axes (Axis) in the axis group (AxesGroup).

5) Timing Diagram

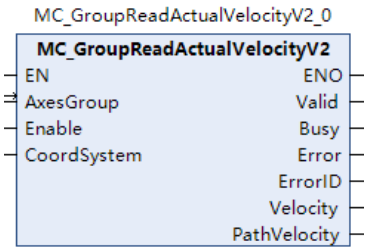
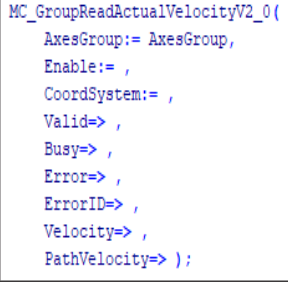
6) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.6 MC_GroupReadActualVelocityV2

This instruction reads the actual velocity of an axis group in the selected coordinate system.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_GroupRead ActualVelocityV2	Reading the actual velocity of an axis group		


```
MC_GroupReadActualVelocityV2_0(
    AxesGroup:= AxesGroup,
    Enable:= ,
    CoordSystem:= ,
    Valid=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> ,
    Velocity=> ,
    PathVelocity=> );
```

2) Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Instruction execution	BOOL	TRUE, FALSE	FALSE	TRUE: Execute the instruction; FALSE: Do not execute the instruction
CoordSystem	Coordinate system type	MC_COORDSYSTEM_V2	ACS, PCS	ACS	Select the coordinate system type

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Valid instruction output	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs
Velocity	Velocity component	ARRAY[0···gc_ iAxis_Num-1] OF LREAL	-	[0,0,···.]	Return the velocity component [u] of the axis group in the corresponding coordinate system
PathVelocity	Resultant velocity	LREAL	-	0	Resultant velocity of the axis group TCP main space axes group

3) Function Description

◆ When Valid is TRUE, the instruction obtains the actual velocity (velocity component) of all

single axes (Axis) in the axis group specified by the input/output variable AxesGroup of this instruction in the selected coordinate type (CoordSystem) and the resultant velocity of the axis group under the PCS based on the control cycle.

4) Precautions

- When this instruction is executed, the axis group specified by the input/output variable AxesGroup must not be in the GroupDisabled state. There is no requirement on the state of axes (Axis) in the axis group (AxesGroup).
- The output variable Velocity is the component velocity in the given coordinate system. The position dimension for the axis group (AxesGroup) to output the position scale of the coordinate component of axes (Axis) in each coordinate system corresponds to the type of logical axes in each coordinate system set by AxesGroup.
- The output variable PathVelocity (resultant velocity) is the resultant velocity of the main dimension space path. When the PCS or ACS is selected, the resultant velocity (position and attitude) of the user main dimension subspace of the axis group TCP in the part coordinate system will be returned.

Position as main dimension: Output the resultant linear velocity of the position subspace TCP in the user part coordinate system;

Attitude as main dimension: Output the resultant angular velocity of the attitude subspace TCP in the user part coordinate system;

Ratio as main dimension: Output the TCP resultant velocity in the user part coordinate system (dimensionless).

5) Timing Diagram

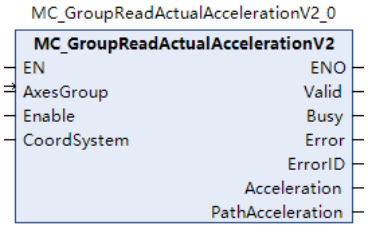
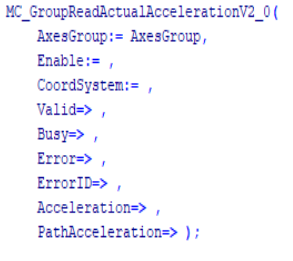
6) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.7 MC_GroupReadActualAccelerationV2

This instruction reads the actual acceleration rate of an axis group in the selected coordinate system.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_GroupRead ActualAccelerationV2	Reading the actual acceleration rate of an axis group		


```
MC_GroupReadActualAccelerationV2_0(
    AxesGroup:= AxesGroup,
    Enable:= ,
    CoordSystem:= ,
    Valid=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> ,
    Acceleration=> ,
    PathAcceleration=> );
```

2) Variables

- Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

- Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Instruction execution	BOOL	TRUE, FALSE	FALSE	TRUE: Execute the instruction; FALSE: Do not execute the instruction
CoordSystem	Coordinate system type	MC_COORDSYSTEM_V2	ACS, PCS	ACS	Select the coordinate system type

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Valid instruction output	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs
Acceleration	Acceleration rate component	ARRAY[0···gc_iAxis_Num-1] OF LREAL	-	[0,0,···]	Return the acceleration rate component of the axis group in the corresponding coordinate system
Path Acceleration	Resultant acceleration rate	LREAL	-	0	Return the resultant acceleration rate of the axis group TCP in the main dimension space

3) Function Description

◆ When Valid is TRUE, the instruction obtains the actual acceleration rate (acceleration rate component) of all single axes (Axis) in the axis group specified by the input/output variable AxesGroup of this instruction in the selected coordinate type (CoordSystem) and the resultant acceleration rate of the axis group under the PCS based on the control cycle.

4) Precautions

◆ When this instruction is executed, the axis group specified by the input/output variable AxesGroup must not be in the GroupDisabled state. There is no requirement on the state of axes (Axis) in the axis group (AxesGroup).

◆ The output variable Acceleration is the acceleration rate component in the given coordinate system. The position dimension for the axis group (AxesGroup) to output the position scale of the coordinate component of axes (Axis) in each coordinate system corresponds to the type of logical axes in each coordinate system set by AxesGroup.

◆ The output variable PathAcceleration (resultant acceleration rate) is the resultant acceleration rate of the main dimension space path. When the PCS or ACS is selected, the resultant acceleration rate (position and attitude) of the user main dimension subspace of the axis group TCP in the part coordinate system will be returned.

Position as main dimension: Output the resultant linear acceleration rate of the position subspace TCP in the user part coordinate system;

Attitude as main dimension: Output the resultant angular accelerate rate of the attitude subspace TCP in the user part coordinate system;

Ratio as main dimension: Output the TCP resultant acceleration rate in the user part coordinate system (dimensionless).

5) Timing Diagram

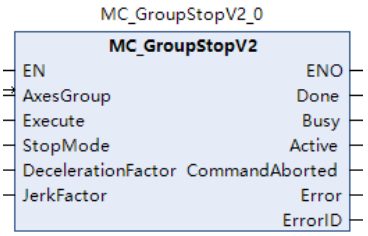
6) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.8 MC_GroupStopV2

This instruction makes the motion stop immediately. It is generally used for stopping an axis group under an abnormal situation, and the current motion cannot be recovered.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_GroupStopV2	Stopping an axis group immediately		MC_GroupStopV2_0(AxesGroup:= AxesGroup, Execute:= , StopMode:= , DecelerationFactor:= , JerkFactor:= , Done=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge
StopMode	Stop mode	E_STOP_MODE_V2	0: DecStop 1: ImmedStop	0	DecStop: Decelerate to stop ImmedStop: Stop immediately
Deceleration Factor	Deceleration factor	LREAL	[0.01,1.0]	0.5	Ratio relative to the maximum deceleration of the axis group
JerkFactor	Jerk factor	LREAL	[0.0,1.0]	0.5	Ratio relative to the maximum jerk of the axis group

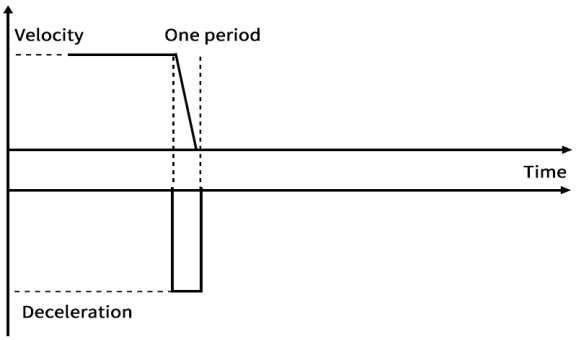
◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Active	Axis group under control of the instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control

Command Aborted	Instruction execution aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

- This instruction is started at the rising edge of Execute. It stops the axis group (AxesGroup) immediately in case of abnormal operation.
- StopMode
- This instruction sets the emergency stop method for abnormal operation of the axis group (AxesGroup) by inputting the variable StopMode.
- ① StopMode =0 (DecStop): Each axis (Axis) of the axis group (AxesGroup) decelerates and stops according to the parameter set in the instruction. When deceleration and stopping operations are completed, Done will be set to TRUE, and the axis group state will be switched to GroupStopping.
- ② StopMode = 1 (ImmedStop): The velocity of each axis (Axis) of the axis group (AxesGroup) will be reduced to 0 in one cycle. Done will be set to TRUE, and the axis group state will be switched to GroupStopping.



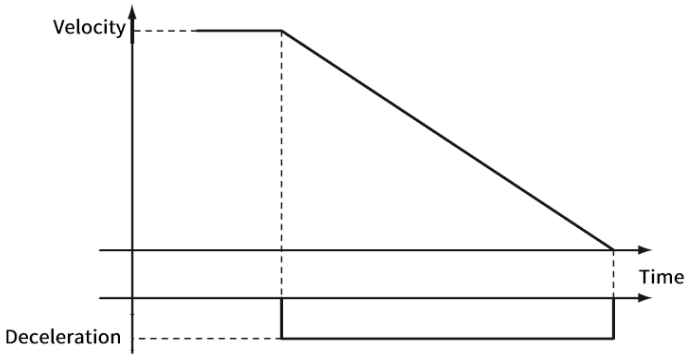
◆ DecelerationFactor and JerkFactor

· The deceleration rate and jerk during the execution of this instruction are determined by the input variables DecelerationFactor and JerkFactor, respectively.

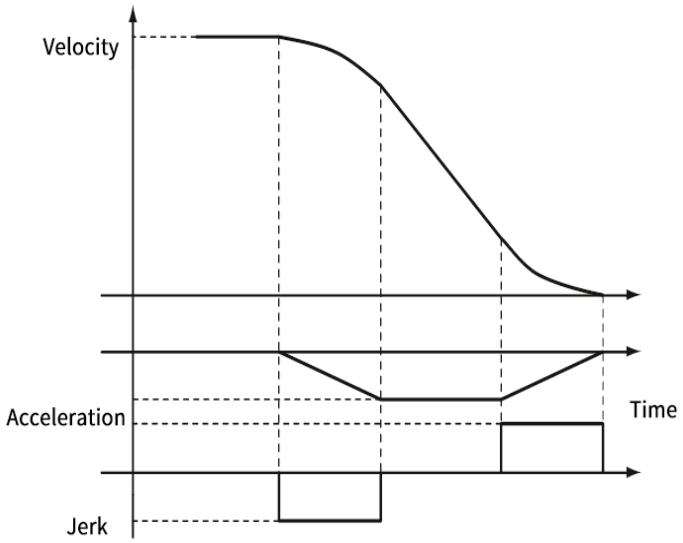
· StopMode = 0 (DecStop): Each axis (Axis) of the axis group (AxesGroup) decelerates and stops according to the product of its software-limited maximum axis acceleration rate and DecelerationFactor as well as the product of its software-limited maximum axis jerk and JerkFactor.

· JerkFactor

① JerkFactor = 0: Velocity profile and deceleration profile



② JerkFactor ≠ 0: Velocity profile, deceleration profile, and jerk profile



◆ StopMode is set to DecStop default.

◆ StopMode = ImmedStop: DecelerationFactor and JerkFactor are invalid.

◆ When this instruction is executed, the axis group state can only be GroupStandby or GroupMoving. It will report an error if executed in other states.

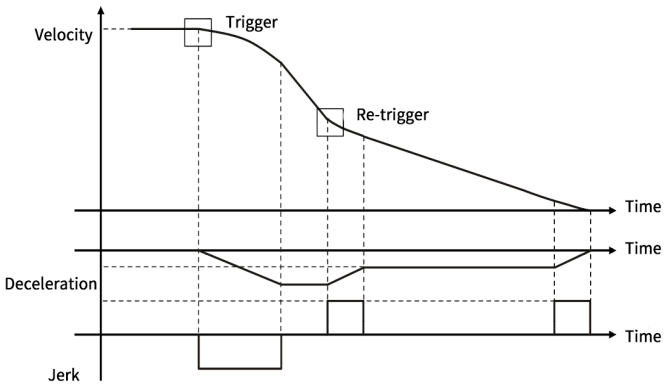
◆ This instruction can interrupt the motion FB being executed by the axis group (AxesGroup) and set the axis group state to GroupStopping. In this state, it is forbidden to start any motion function instruction. Otherwise, the corresponding motion instruction will report an error.

◆ If this instruction is executed during the homing process of the axis group (AxesGroup), StopMode, DecelerationFactor, and JerkFactor are invalid. The deceleration and stopping operations may not be performed along the current path, and the axis group (AxesGroup) will decelerate and stop in the same way as a servo stops.

◆ When the axis group (AxesGroup) is in the GroupStopping state, this instruction cannot be started. Otherwise, an error will be reported.

◆ Repeated instruction triggering

· This instruction can be triggered again. If you trigger this instruction again after modifying the parameters, the instruction will be executed based on the modified parameters. If the modified parameters are not reasonable, the instruction will report an error, the velocity of the axis group (AxesGroup) will be reduced to 0 immediately, and the axis group status will change to GroupErrorStop.



4) Precautions

- The trajectory profile type of the axis group (AxesGroup) is set as required. If the profile type is trapezoidal, the input variable JerkFactor is invalid, and the instruction performs the deceleration and stopping operations based on DecelerationFactor. If the profile type is cubic or septuple, the instruction performs the deceleration and stopping operations based on DecelerationFactor and JerkFactor.
- Whether the profile type of each axis (Axis) in the axis group (AxesGroup) is trapezoidal or S-shaped does not affect the planning method of the axis group (AxesGroup).

5) Timing Diagram

6) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.9 MC_GroupHaltV2

This instruction stops the motion of an axis group. It is generally used under normal conditions.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_GroupHaltV2	Stopping an axis group	MC_GroupHaltV2_0 MC_GroupHaltV2 EN → ENO AxesGroup → Done Execute → Busy DecelerationFactor → Active JerkFactor → CommandAborted Error ErrorID	MC_GroupHaltV2_0(AxesGroup:= AxesGroup, Execute:= , DecelerationFactor:= , JerkFactor:= , Done=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Variables

- Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

- Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge
Deceleration Factor	Deceleration factor	LREAL	[0.01,1.0]	0.5	Ratio relative to the maximum deceleration of the axis group
JerkFactor	Acceleration factor	LREAL	[0.0,1.0]	0.5	Ratio relative to the maximum jerk of the axis group

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Active	Axis group under control of the instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Command Aborted	Instruction execution aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ This instruction is started at the rising edge of Execute. It stops the axis group (AxesGroup) during normal operation.

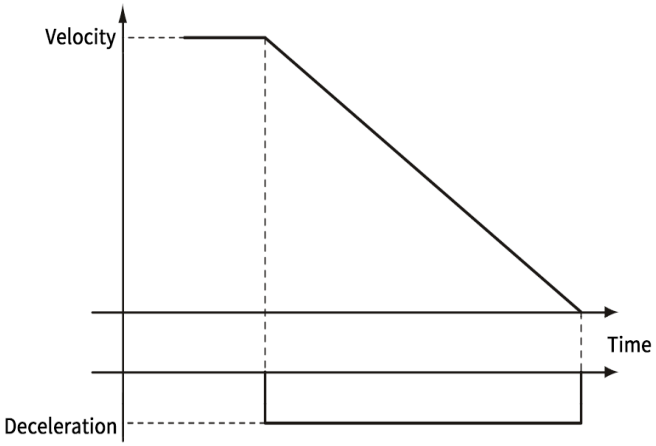
◆ DecelerationFactor and JerkFactor

· The deceleration rate and jerk during the execution of this instruction are determined by the input variables DecelerationFactor and JerkFactor, respectively.

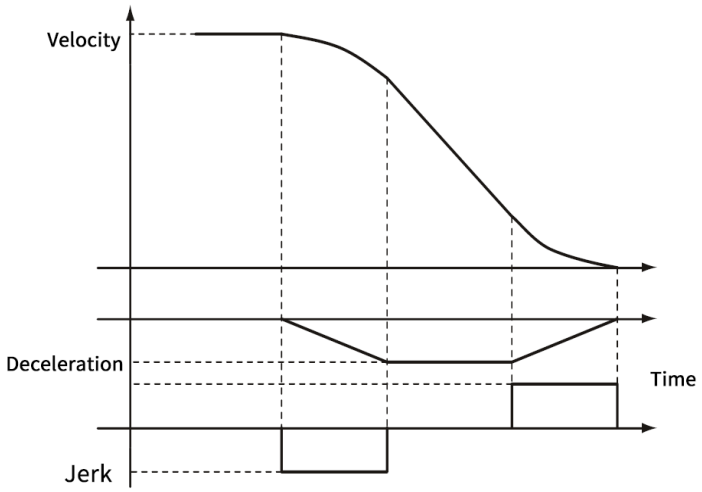
Each axis (Axis) of the axis group (AxesGroup) decelerates and stops according to the product of its software-limited maximum axis acceleration rate and DecelerationFactor as well as the product of its software-limited maximum axis jerk and JerkFactor.

· JerkFactor

① JerkFactor = 0: Velocity profile and deceleration profile



② JerkFactor ≠ 0: Velocity profile, deceleration profile, and jerk profile



◆ When this instruction is executed, the axis group state can only be GroupStandby or GroupMoving. It will report an error if executed in other states.

◆ This instruction cannot be triggered again.

◆ This instruction can interrupt the instruction that makes the current axis group (AxesGroup) move. When the axis group motion instruction is interrupted, the axis group (AxesGroup) keeps the GroupMoving state. After its resultant velocity in the current motion space (workspace/axis space) becomes 0, Done of the instruction is set to TRUE, and the axis group state switches to GroupStandby or retains the GroupMoving state. The state changes as follows:

· If the current motion is in a static coordinate system, this instruction switches the axis group from GroupMoving to GroupStandby.

· If the current motion is in the user PCS (LIN/CIRC type) under a dynamic coordinate system, this instruction keeps the axis group in the GroupMoving state.

· If the current motion is in a dynamic coordinate system and also in the joint axis coordinate space (DIRECT type), this instruction switches the axis group state to GroupStandby.

◆ This instruction can be interrupted by other motion instructions.

◆ This instruction cannot be executed when the axis group (AxesGroup) is in the homing process. Otherwise, an error will be reported.

◆ This instruction cannot be executed when the axis group is in the GroupStopping state. Otherwise, an error will be reported.

◆ This instruction can be interrupted by other MC_GroupHalt instructions, which can interrupt each other sequentially. That is, the MC_GroupHalt instruction that starts later interrupts the one that starts earlier, and the deceleration and stopping operations will be performed according to the parameters of the MC_GroupHalt instruction that starts later.

4) Precautions

- ◆ The trajectory profile type of the axis group (AxesGroup) is set as required. If the profile type is trapezoidal, the input variable JerkFactor is invalid, and the instruction performs the deceleration and stopping operations based on DecelerationFactor. If the profile type is cubic or septuple, the instruction performs the deceleration and stopping operations based on DecelerationFactor and JerkFactor.
- ◆ Whether the profile type of each axis (Axis) in the axis group (AxesGroup) is trapezoidal or S-shaped does not affect the planning method of the axis group (AxesGroup).

5) Timing Diagram

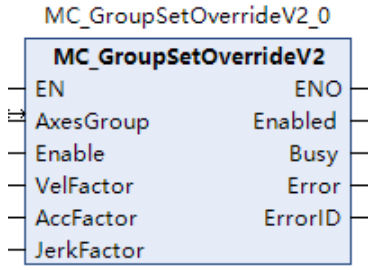
6) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.10 MC_GroupSetOverrideV2

This instruction sets the ratio of the axis group motion in the coordinate system. The ratio parameter applies, as a factor, to the velocity, acceleration rate, deceleration rate, and jerk operation phases of each function block in the axis group concurrently.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_GroupSetOverrideV2	Setting the axis group ratio		MC_GroupSetOverrideV2_0(AxesGroup:= AxesGroup, Enable:= , VelFactor:= , AccFactor:= , JerkFactor:= , Enabled=> , Busy=> , Error=> , ErrorID=>);

2) Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Instruction execution	BOOL	TRUE, FALSE	FALSE	TRUE: Execute the instruction; FALSE: Do not execute the instruction
VelFactor	Velocity ratio factor	LREAL	[0.0, 100.0]	100.0	Percentage of speed ratio
AccFactor	Acceleration factor	LREAL	[1.0, 100.0]	100.0	Ratio relative to the maximum acceleration rate of the axis group
JerkFactor	Jerk factor	LREAL	[0.0, 100.0]	100.0	Ratio relative to the maximum jerk of the axis group

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Enabled	Instruction executed successfully	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executed successfully
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

This instruction is started at the rising edge of Enable. It sets the ratio of the axis group (AxesGroup) motion in the coordinate system.

· The default value of VelFactor, AccFactor, and JerkFactor is 100.0. The value of VelFactor and JerkFactor can also be 0.0.

· The unit of the ratio factor is [%]. “100” indicates 100% and “50” indicates 50%.

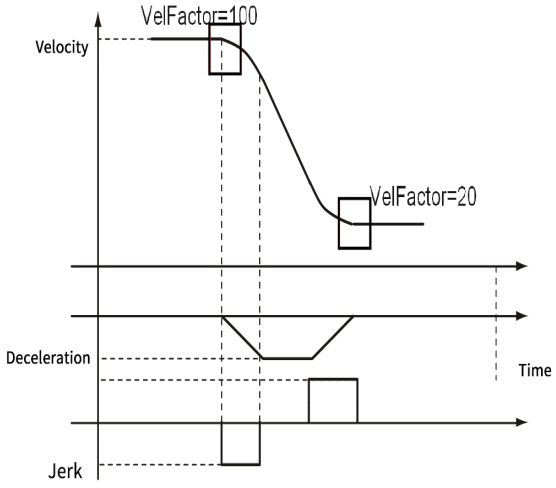
· When Enable is FALSE, the value of VelFactor, AccFactor, and JerkFactor will be restored to the default value.

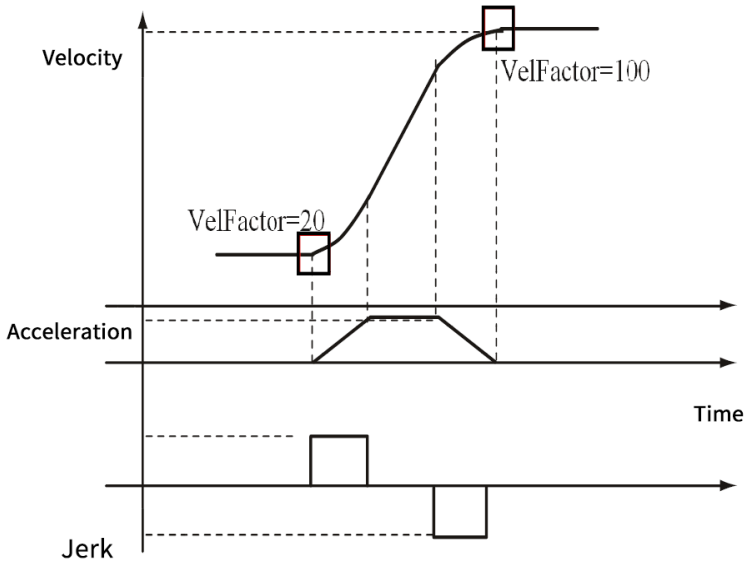
· When VelFactor is 0, the running velocity of the axis group (AxesGroup) changes to 0, but the axis group state does not switch to GroupStandby, that is, the axis group (AxesGroup) keeps running at the velocity of 0.

· This instruction only affects the velocity at which the axis group (AxesGroup) is running without interrupting its current motion.

· VelFactor can respond in real time and act on the current motion. If the instruction output Enabled becomes TRUE, the velocity ratio factor has been set successfully. AccFactor and JerkFactor reflect the ratio adjustment response speed and are only effective during the change of motion ratio.

Note: When the axis group state is grouphoming, groupsgtopping, or grouperrorstop, the ratio instruction will report an error. However, the actual ratio will continue to take effect as long as the axis group state restores to normal and the Enable input is TRUE.





During velocity ratio adjustment, the deceleration (acceleration) rate equals the product of the software-limited maximum deceleration (acceleration) rate of the axis group and AccFactor. The jerk equals the product of the software-limited maximum jerk of the axis group and JerkFactor.

◆ The trajectory profile type of the axis group (AxesGroup) is set as required. If the profile type is trapezoidal, JerkFactor does not take effect.

4) Precautions

- ◆ This instruction can only be started when the axis group (AxesGroup) is in the GroupMoving or GroupStandby state. Otherwise, an error will be reported.
- ◆ This instruction cannot be started when the axis group (AxesGroup) is controlled by the MC_GroupStop, MC_GroupHalt, or MC_GroupInterrupt instruction. Otherwise, an error will be reported.

5) Timing Diagram

6) Error Description

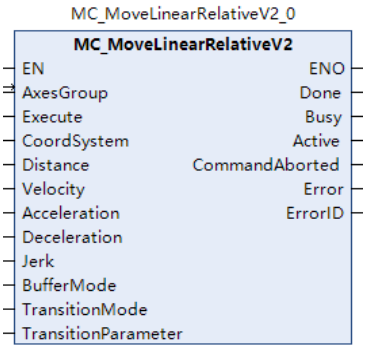
Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.11 MC_MoveLinearRelativeV2

This instruction enables the linear interpolation motion of the axis group from the current TCP point to the relative position point under the given coordinate system.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression

MC_Move LinearRelativeV2	Linear relative motion		MC_MoveLinearRelativeV2_0(AxesGroup:= AxesGroup, Execute:= , CoordSystem:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , BufferMode:= , TransitionMode:= , TransitionParameter:= , Done=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge
Coord System	Coordinate type	MC_COORDSYSTEM_V2	ACS, PCS	PCS	Currently, only PCS is supported
Distance	Relative position	ARRAY[0…gc_iAxis_Num-1] OF LREAL	-	[0,0,…]	Displacement component in each dimension in the given coordinate system
Velocity	Velocity	LREAL	> 0	-	Maximum main velocity, not necessarily reachable
Acceleration	Acceleration rate	LREAL	> 0	-	Maximum main acceleration rate, not necessarily reachable
Deceleration	Deceleration rate	LREAL	> 0	-	Maximum main deceleration rate, not necessarily reachable
Jerk	Jerk	LREAL	>=0	-	Maximum main jerk, not necessarily reachable
BufferMode	Buffer mode	MC_BUFFER_MODE_V2	MC_ABORTING=0; MC_BUFFERED=1; MC_BLEND_LOW=2; MC_BLEND_PRE=3; MC_BLEND_NEXT=4; MC_BLEND_HIGH=5	MC_ABORTING	Aborting, buffered, blend

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Transition Mode	Transition mode	MC_TRANSITION_MODE_V2	TMNone=0; TMStartVelocity=1; TMCornerDistance=3; TMMaxCornerDeviation=4; TMDefinedVelocity=10; TM_RESET_ABORT=11; TM_DEC_ABORT=12	TMNone	Define the transition trajectory characteristics
Transition Parameter	Transition parameter	LREAL	Velocity [0.0,1.0] Distance [0.0,+ ∞)	1	The transition parameters vary with the transition mode

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Active	Axis group under control of the instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Command Aborted	Instruction execution aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ This instruction is started at the rising edge of Execute and performs linear relative motion of 2 to 4 axes.

◆ You can specify input variables CoordSystem, Distance, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, and TransitionParameter.

◆ The target position of this instruction is the sum of the position (starting point) of the axis group (AxesGroup) at the start of the instruction and the input variable Distance (relative position).

◆ CoordSystem

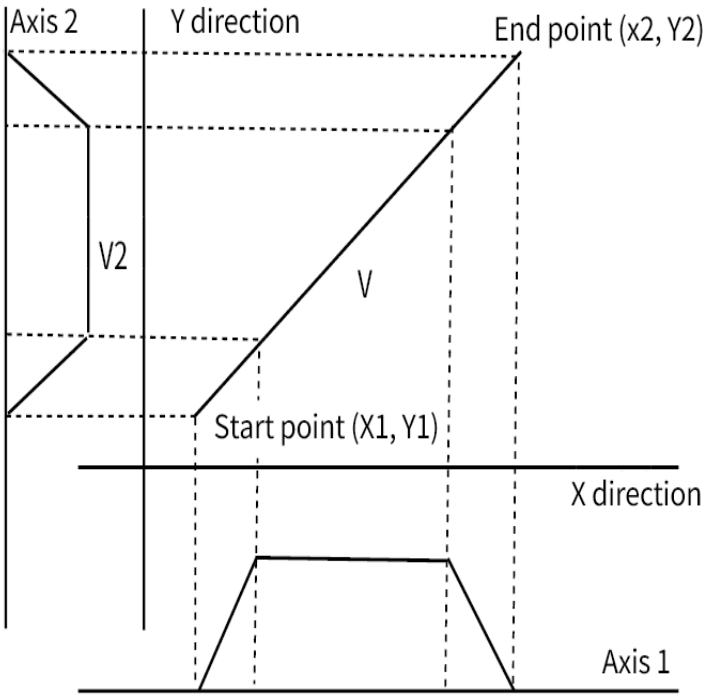
· Specify the coordinate system for linear relative motion. Currently, only linear relative motion under the PCS is supported.

◆ Velocity, Acceleration, Deceleration, and Jerk

· If the velocity of any axis (Axis) in the axis group (AxesGroup) exceeds the software-limited maximum velocity of the axis, an error stop will be performed.

· The trajectory profile type of the axis group (AxesGroup) is set as required. If the profile type is trapezoidal, Jerk does not take effect and is regarded as 0. (The profile type of the axis group is specified in the axis group data structure).

· The linear motion is decomposed to each axis when linear relative motion is performed.



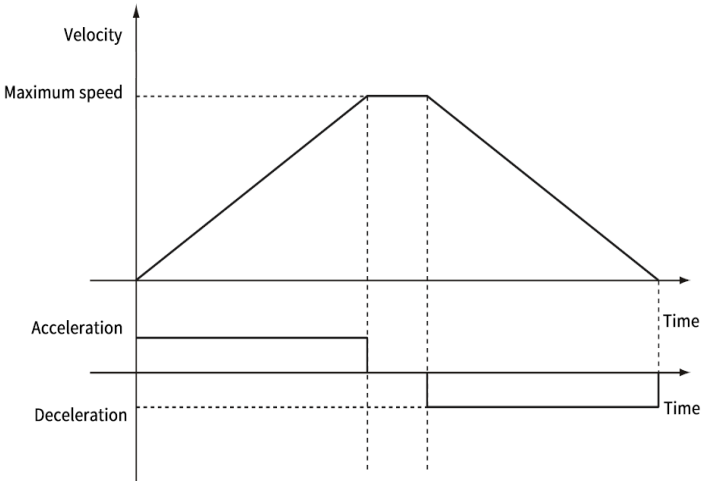
V: Target velocity for the axis group to perform linear motion

V1: Velocity decomposed to axis 1 V1=(X2-X1)*V/ √ ([(X2-X1)]^2+[(Y2-Y1)]^2)

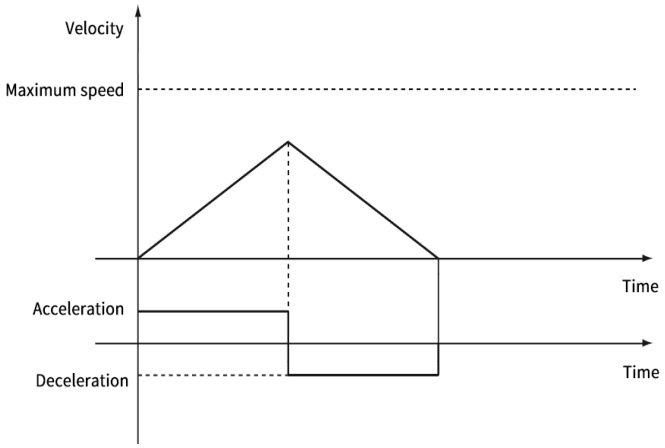
V2: Velocity decomposed to axis 2 V2=(Y2-Y1)*V/ √ ([(X2-X1)]^2+[(Y2-Y1)]^2)

· The curves of velocity, acceleration rate, deceleration rate, and jerk with the conditions Jerk = 0 and Jerk ≠ 0 are shown below.

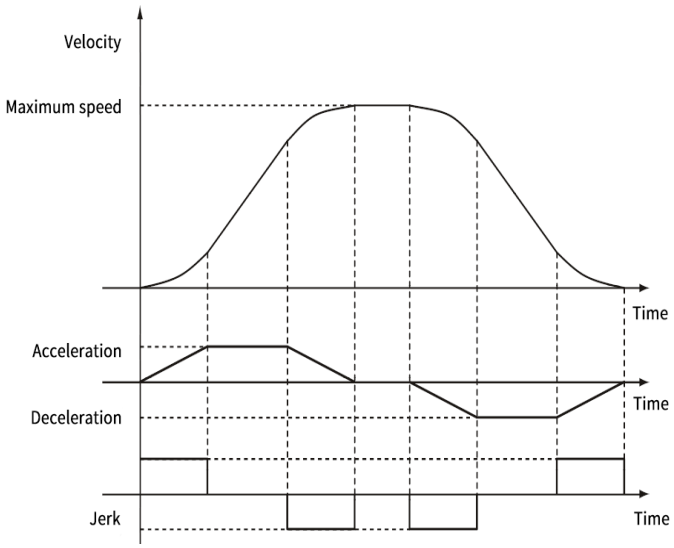
Jerk = 0, maximum interpolation velocity reachable



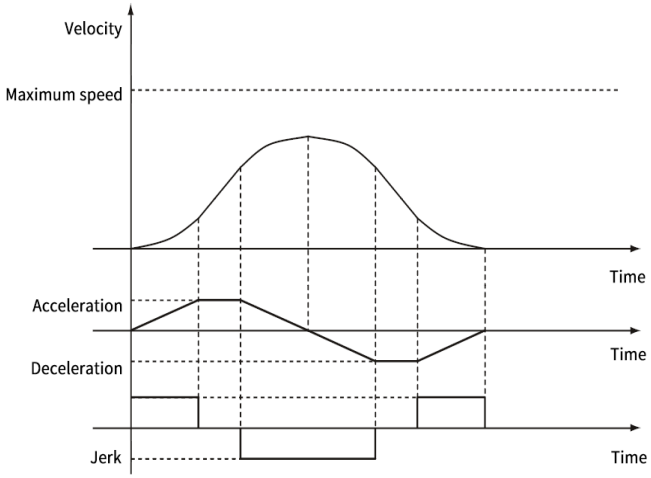
Jerk = 0, maximum interpolation velocity unreachable due to short motion distance



Jerk ≠ 0, maximum interpolation velocity reachable (Non-trapezoidal profile)



JerkFactor ≠ 0, maximum interpolation velocity unreachable due to short motion distance (Non-trapezoidal profile)



◆ Repeated instruction triggering

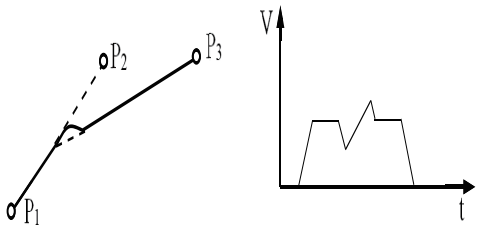
This instruction cannot be triggered again.

◆ Starting multiple motion instructions

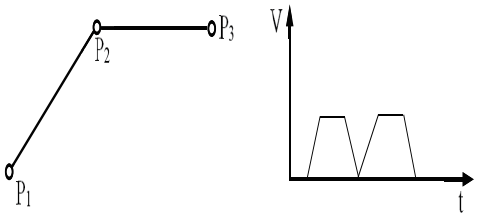
When this instruction is started for the currently executing instruction, it will be switched to this instruction or this instruction will be buffered. Up to 8 instructions can be buffered for each axis group. The motion of this instruction is specified by BufferMode when multiple instructions are started.

◆ BufferMode

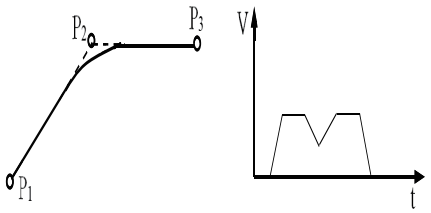
Aborting (MC_ABORTING): Interrupt the current instruction and start to execute this instruction.



Buffered (MC_BUFFERED): This instruction is buffered and will be started after the current instruction is executed.



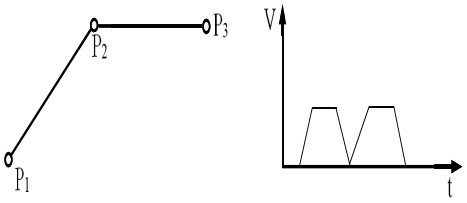
Blend (MC_BLEND): Blend the velocity of two instructions in the specified way. This method is specified in TransitionMode.



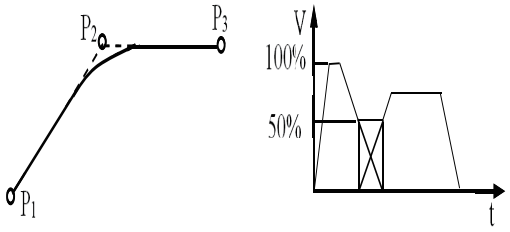
◆ TransitionMode

The blend method of the velocity of two instructions when the input variable TransitionMode of this instruction specifies BufferMode as blend.

· TMNone: No insertion of transition curve If TMNone is adopted as the blend mode, the result is the same as the transition mode between the velocity of the two instructions in buffered mode.



TMStartVelocity: Blend at the previous velocity Blending is performed for the transition based on a certain ratio (0,1] of the actual maximum velocity of the current motion instruction. Any transition between motion instructions is valid.

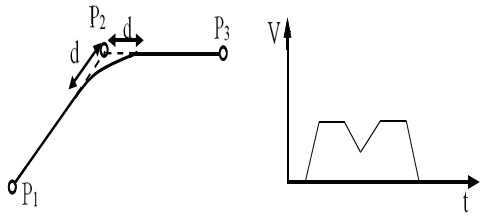


TMCornerDistance: Corner transition mode In Cartesian space, specify the distance from the previous instruction to the end under the main dimension space. Use this distance to open a transition trajectory to link the neighboring trajectory. The distance here is in a broad sense and is related to the main dimension type of the axis group.

① Position as main dimension: The parameter is the displacement from the transition point to the end point.

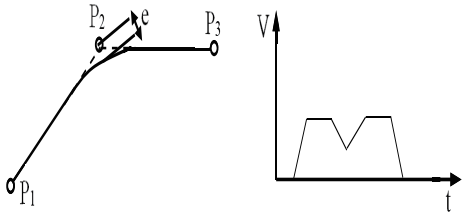
② Attitude as main dimension: The parameter is the attitude from the transition point to the end point.

③ Ratio as main dimension: The parameter is the ratio of the resultant distance from the transition point to the end point to the total resultant displacement (0-1).



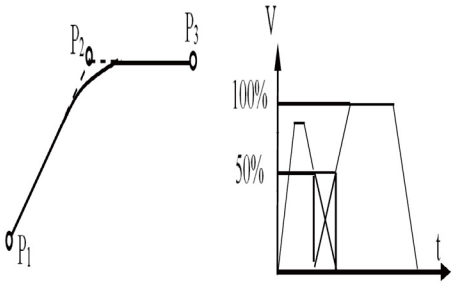
· TMMaxCornerDeviation: Maximum corner position deviation transition mode

In Cartesian space, set the maximum position deviation distance for the trajectory transition of the neighboring instructions in the position subspace. Two neighboring trajectories are linked by constructing the corresponding transition trajectory. In this way, the maximum position deviation of the transition trajectory is always within a certain range. Only the position deviation is involved here, which is not related to the main dimension. Both instructions are valid when they are in Cartesian space (LIN/CIRC). Otherwise, the instructions are processed in no-transition mode.



· TMDefinedVelocity: Blend at the next velocity

Blending is performed for the transition based on a certain ratio (0,1] of the actual maximum velocity of the next instruction of the current motion. Any transition between motion instructions is valid.



4) Precautions

- This instruction only supports linear relative motion under the PCS.
- When the current motion instruction is interrupted, the linear relative motion immediately starts to plan a new instruction from the interrupted position. In the workspace, a transition trajectory is constructed for deceleration and reverse reset along the velocity direction of the interrupted position, which ensures a smooth link with the latter section of the trajectory. Here, only the velocity smoothing is guaranteed, and the acceleration smoothing is not guaranteed.
- This instruction can only be started when the axis group is in the GroupMoving or GroupStandby state. If it is started in other states, an error will be reported.
- The execution of this instruction can be interrupted by other motion instructions as well as the MC_GroupStop and MC_GroupHalt instructions.
- After the execution of this instruction, the axis group (AxesGroup) state switches from GroupMoving to GroupStandby or retains the GroupMoving state. The state changes as follows:

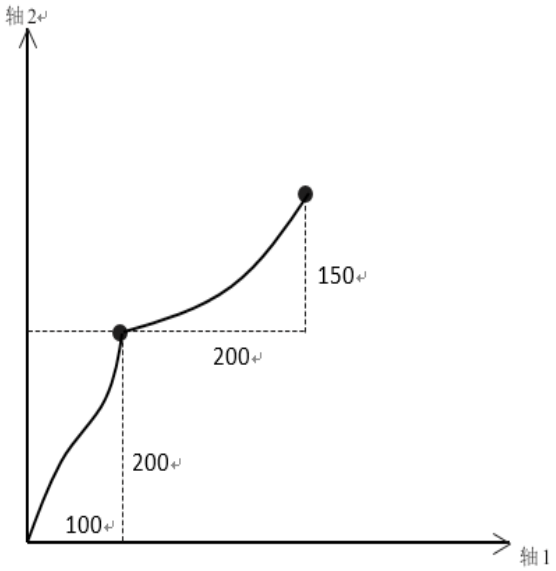
If the current motion is in a static coordinate system, the axes group state switches from GroupMoving to GroupStandby. If the current motion is in a dynamic coordinate system, the axis group remains in the GroupMoving state.

5) Program Example

Take a 2-axis group as an example to describe the linear relative motion.

Starting point: (0,0); first relative displacement: (100,200); second relative displacement: (200,150)

You need to execute two linear relative motion instructions in sequence, and the second instruction is in buffered mode (MC_BUFFERED).



· Steps:

① Determine the axis group (AxesGroup) type and the axes (Axis) in the axis group in InoProShop.

② Execute the MC_Power instruction to enable axes (Axis) in the axis group (AxesGroup).

③ Execute the MC_Home instruction to enable axes (Axis) in the axis group (AxesGroup) to perform the homing operation.

Or execute the MC_GroupHome instruction after the execution of the MC_GroupEnable instruction to enable axes (Axis) in the axis group (AxesGroup) to perform the homing operation.

④ Execute the MC_GroupEnable instruction to enable the axis group (AxesGroup).

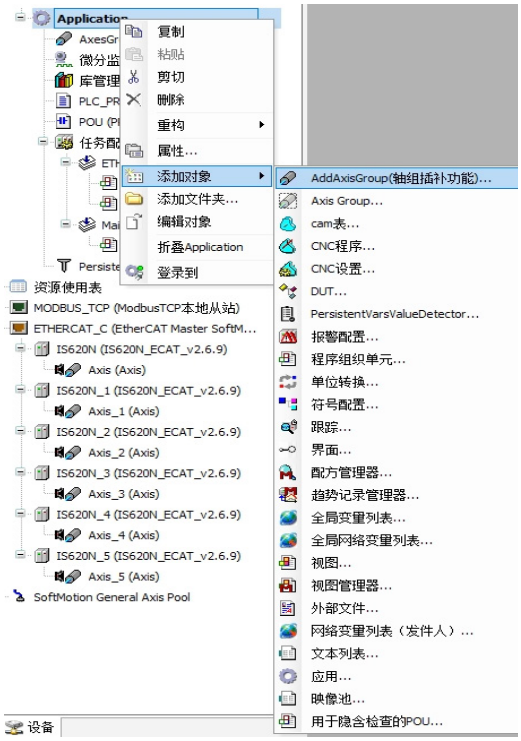
⑤ Set the parameters of the first point-to-point relative motion instruction.

⑥ Set the parameters of the second point-to-point relative motion instruction.

⑦ Start two point-to-point relative motion instructions.

· Adding and setting the axis group

1. Right-click “Application” in the device UI and select “Add Object” > “AddAxisGroup”.



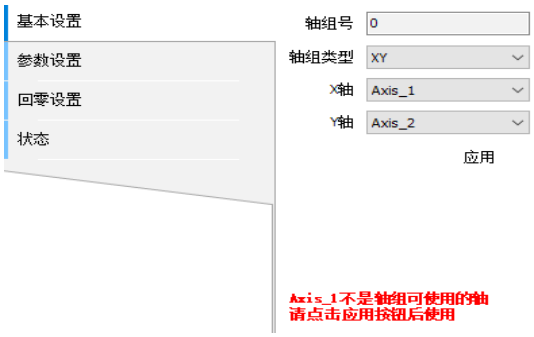
2. After adding an axis group, double-click the axis group to be set and enter the setting UI.



Note 1: The profile type of the axis group should be specified in the axis group structure. By default, the profile type is trapezoidal, and the jerk settings do not take effect.

Note 2: The type of a single axis in the axis group should be AXIS_REF_ETC_INOVANCE. If the selected type does not match, you need to click “Commit” to make it take effect, which will trigger full loading. If the single axis is released from the axis group and its axis type does not match that when it was added to the axis group, you also need to click “Commit” to make it take effect, which will trigger full loading.

Note 3: Currently, the type of a single axis in the axis group can only be 402 (real axis only).

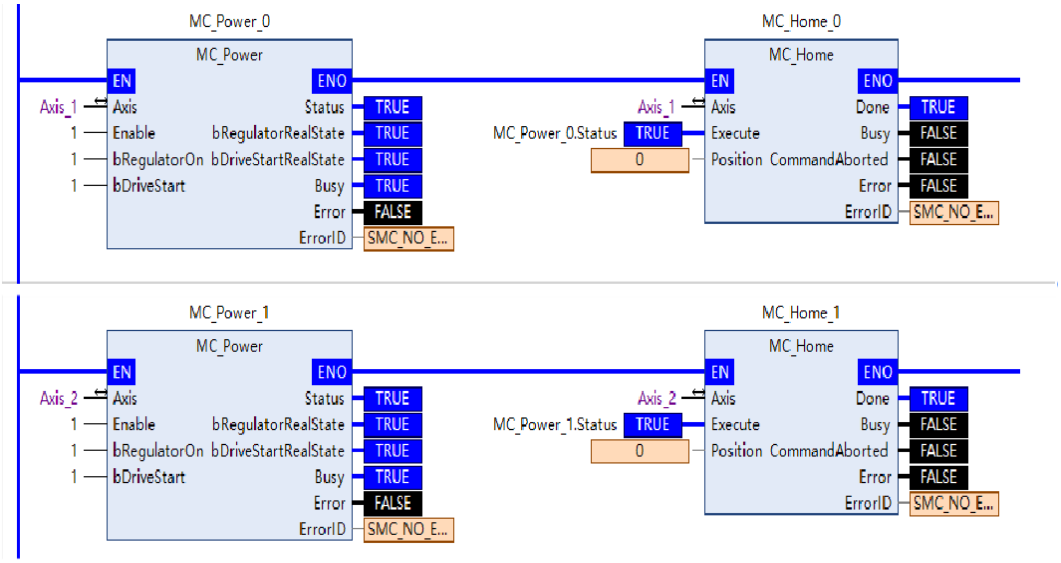


3. Double-click the interface of the single axis selected in the axis group, and then set the parameters. In “BasicSettings”, set the axis type (linear or rotary). (Note: Rotary axes cannot be added to the axis group. Otherwise, an error will be reported.) Set limits for the axis velocity, acceleration rate, deceleration rate, and jerk, and set positive and negative limits. Set the profile type of the axis (Note: The profile type of the axis is valid only in the motion instruction of the axis. If the axis is added to an axis group, its profile type is specified by the axis group).

· Ladder diagram

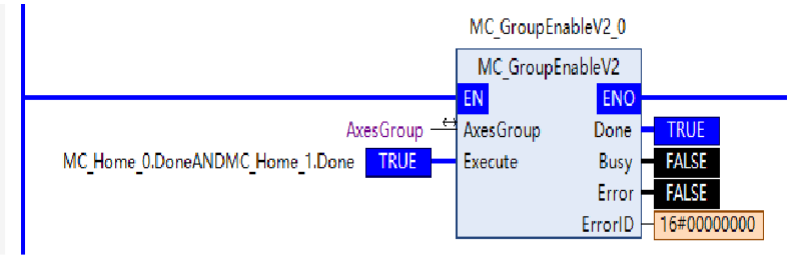
① MC_Power_0 and MC_Power_1 are instantiations of the MC_Power (axis enable) instruction and they enable Axis_1 and Axis_2 respectively. When MC_Power_0.Status and MC_Power_1.Status are set to TURE, the two axes have been enabled. In this case, the axis homing instruction is executed.

② MC_Home_0 and MC_Home_1 are instantiations of the MC_Home instruction and they enable Axis_1 (axis) and Axis_2 (axis) to return to the mechanical zero position respectively.



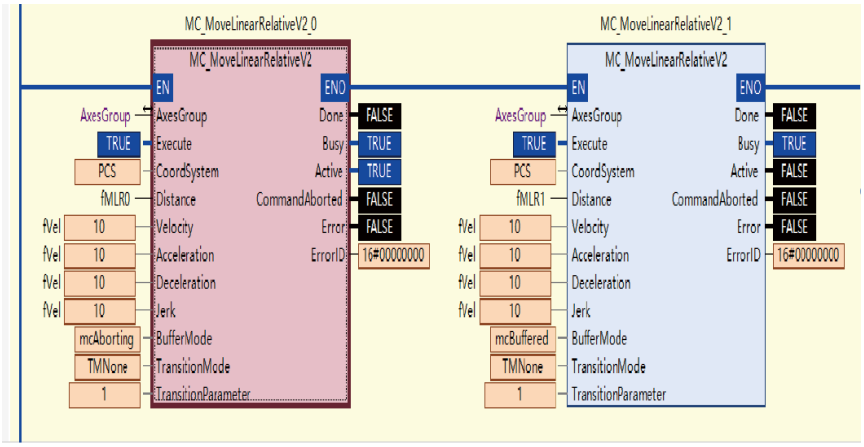
③ MC_Home_0.Done and MC_Home_1.Done indicate that Axis_1 and Axis_2 have completed the homing instruction respectively. Only after both axes have completed the homing instruction, the MC_GroupEnable instruction will be executed for the axis group.

MC_GroupEnable_0 is the instantiation of the MC_GroupEnable instruction.

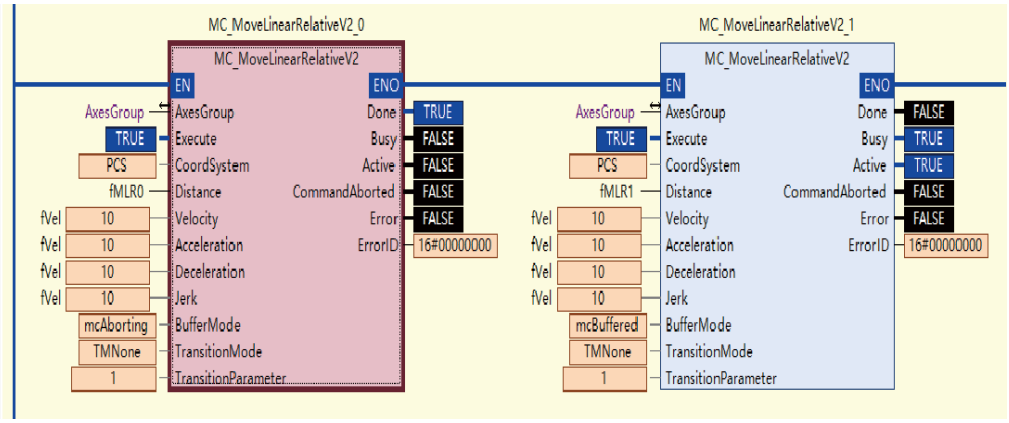


After the axis group is enabled, the linear relative motion begins. MC_MoveLinearRelative_0 and MC_

MoveLinearRelative_1 are instantiations of the MC_MoveLinearRelative instruction. The input pin parameter and buffer mode settings are as follows. The two instructions are triggered simultaneously.



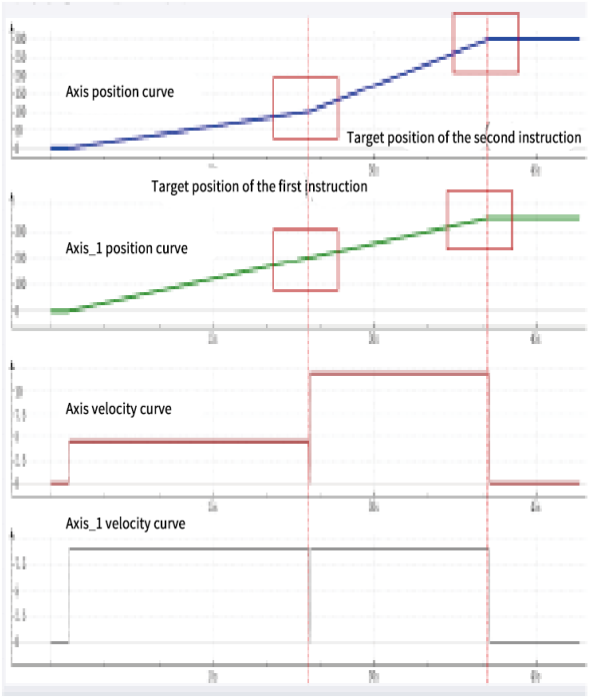
The first instruction will perform motions first, with its MC_MoveLinearRelative_0. Busy and MC_MoveLinearRelative_0. Active both set to TRUE. The second instruction will be in the standby state with only MC_MoveLinearRelative_1. Busy set to TRUE. The second instruction MC_MoveLinearRelative_1. Active will not be set until the first instruction MC_MoveLinearRelative_0.Done is set to TRUE.



④ The position parameter settings for each of the two instructions are shown below.

fMLR0	ARRAY [0..(gc_iAxis_Num - 1)] OF LREAL	
fMLR0[0]	LREAL	100
fMLR0[1]	LREAL	200
fMLR0[2]	LREAL	0
fMLR0[3]	LREAL	0
fMLR1	ARRAY [0..(gc_iAxis_Num - 1)] OF LREAL	
fMLR1[0]	LREAL	200
fMLR1[1]	LREAL	150
fMLR1[2]	LREAL	0
fMLR1[3]	LREAL	0

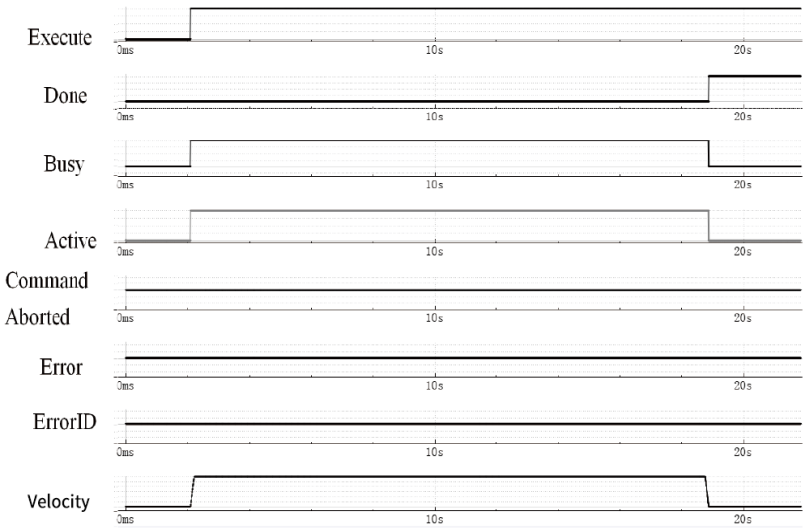
⑤ The position profiles and velocity profiles of axes in the axis group are shown below.



6) Timing Diagram

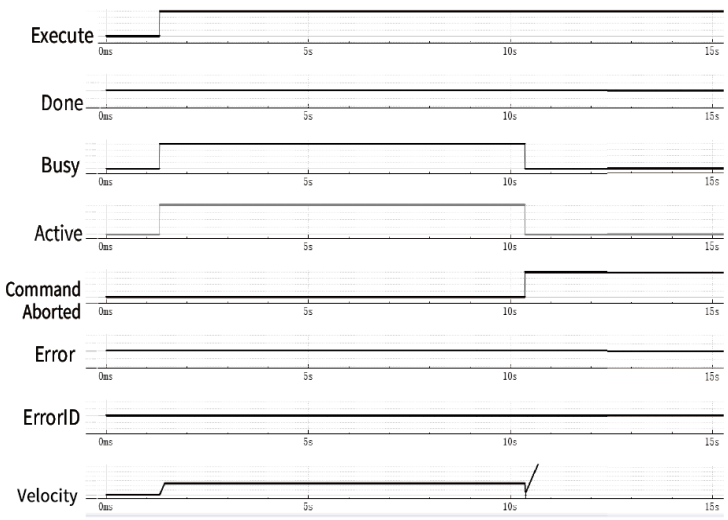
① This instruction operates normally.

Busy and Active change to TRUE when Execute is started. When the target position is reached, the axis group velocity changes to 0, Done becomes TRUE, and Busy and Active become FALSE.

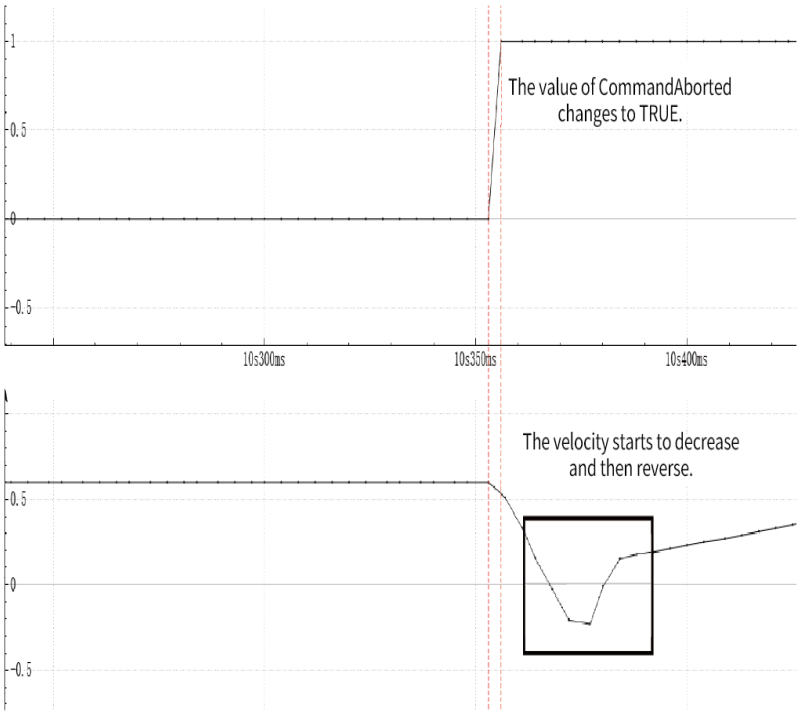


② This instruction is interrupted by another instruction.

Busy and Active change to TRUE when Execute is started. When this instruction is interrupted by another instruction, CommandAborted becomes TRUE and Busy and Active become FALSE.

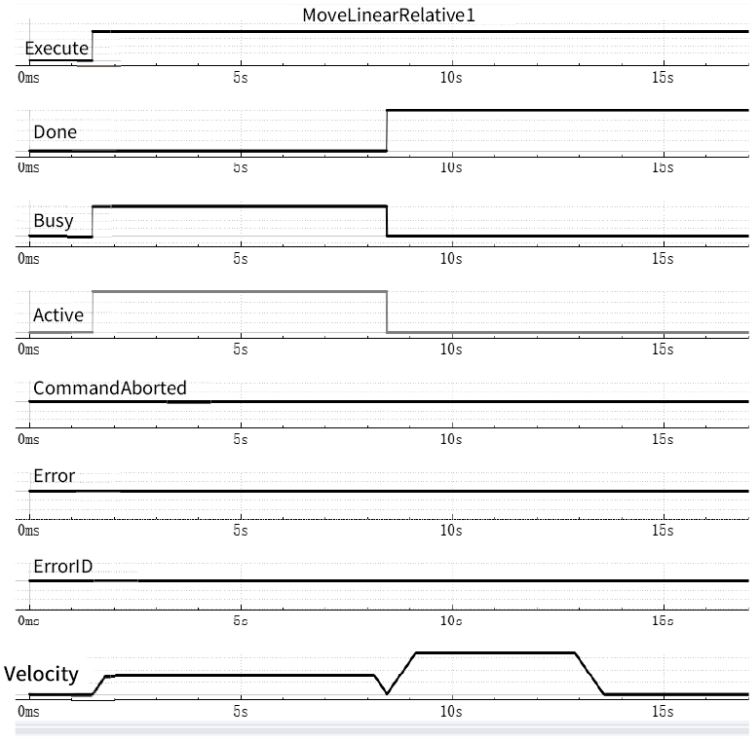


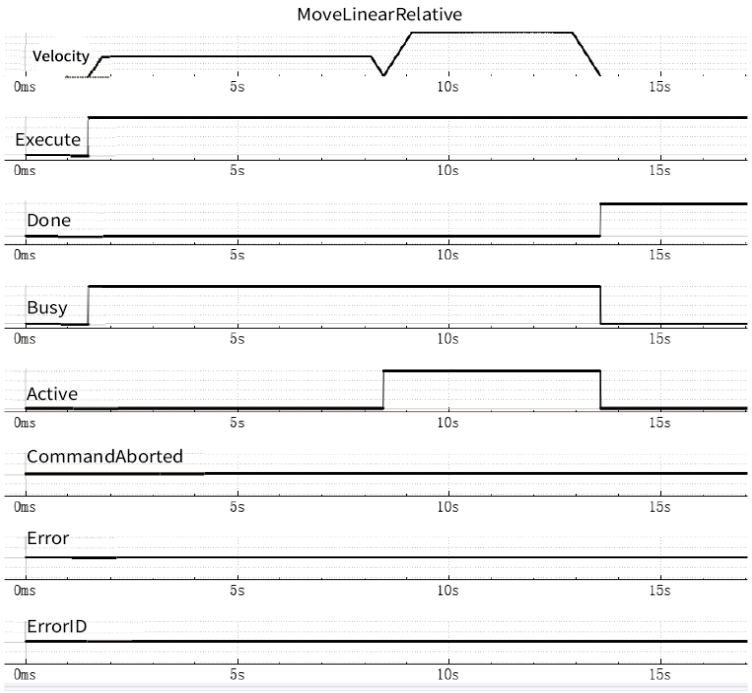
As mentioned in the notes, when the current motion instruction is interrupted, the linear relative motion immediately starts to plan a new instruction from the interrupted position. In the joint space, a transition trajectory is constructed for deceleration and reverse reset along the velocity direction of the interrupted position, which ensures a smooth link with the latter section of the trajectory.



③ The current instruction is followed by a buffered instruction.

Execute of MoveLinearRelative1 and MoveLinearRelative2 are started at the same time. Busy and Active of MoveLinearRelative1 change to TRUE and Busy of MoveLinearRelative2 changes to TRUE. The axis group moves according to the parameters of MoveLinearRelative1 until Done of MoveLinearRelative1 becomes TRUE. In the meantime, Busy and Active of MoveLinearRelative1 become FALSE and Active of MoveLinearRelative2 becomes TRUE. In this case, the axis group moves according to the parameters of MoveLinearRelative2. When the target position of MoveLinearRelative2 is reached, the velocity of the axis group becomes 0, Done of MoveLinearRelative2 becomes TRUE, and Busy and Active become FALSE.





7) Error description

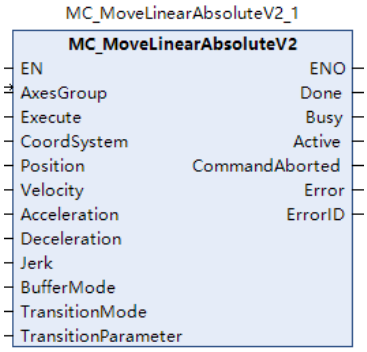
Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.12 MC_MoveLinearAbsoluteV2

This instruction enables the linear interpolation motion of the axis group from the current TCP point to the absolute position point under the given coordinate system.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression

MC_Move LinearAbsoluteV2	Linear absolute motion		MC_MoveLinearAbsoluteV2_1(AxesGroup:= AxesGroup, Execute:= , CoordSystem:= , Position:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , BufferMode:= , TransitionMode:= , TransitionParameter:= , Done=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge
Coord System	Coordinate type	MC_COORDSYSTEM_V2	ACS, PCS	PCS	Currently, only PCS is supported
Position	Position	ARRAY[0…gc_iAxis_Num-1] OF LREAL	-	0	Absolute position coordinates of each dimension in the given coordinate system
Velocity	Velocity	LREAL	> 0	-	Maximum main velocity, not necessarily reachable
Acceleration	Acceleration rate	LREAL	> 0	-	Maximum main acceleration rate, not necessarily reachable
Deceleration	Deceleration rate	LREAL	> 0	-	Maximum main deceleration rate, not necessarily reachable
Jerk	Jerk	LREAL	>=0	-	Maximum main jerk, not necessarily reachable
BufferMode	Buffer mode	MC_BUFFER_MODE_V2	MC_ABORTING=0; MC_BUFFERED=1; MC_BLEND_LOW=2; MC_BLEND_PRE=3; MC_BLEND_NEXT=4; MC_BLEND_HIGH=5	MC_ABORTING	Aborting, buffered, blend

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Transition Mode	Transition mode	MC_TRANSITION_MODE_V2	TMNone=0; TMStartVelocity=1; TMCornerDistance=3; TMMaxCornerDeviation=4; TMDefinedVelocity=10; TM_RESET_ABORT=11; TM_DEC_ABORT=12	TMNone	Define the transition trajectory characteristics
Transition Parameter	Transition parameter	LREAL	Velocity [0.0,1.0] Distance [0.0,+ ∞)	1	The transition parameters vary with the transition mode

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Active	Axis group under control of the instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Command Aborted	Instruction execution aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ This instruction is started at the rising edge of Execute and performs linear absolute motion from axis 2 to axis 4.

◆ You can specify input variables CoordSystem, Position, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, and TransitionParameter.

◆ CoordSystem

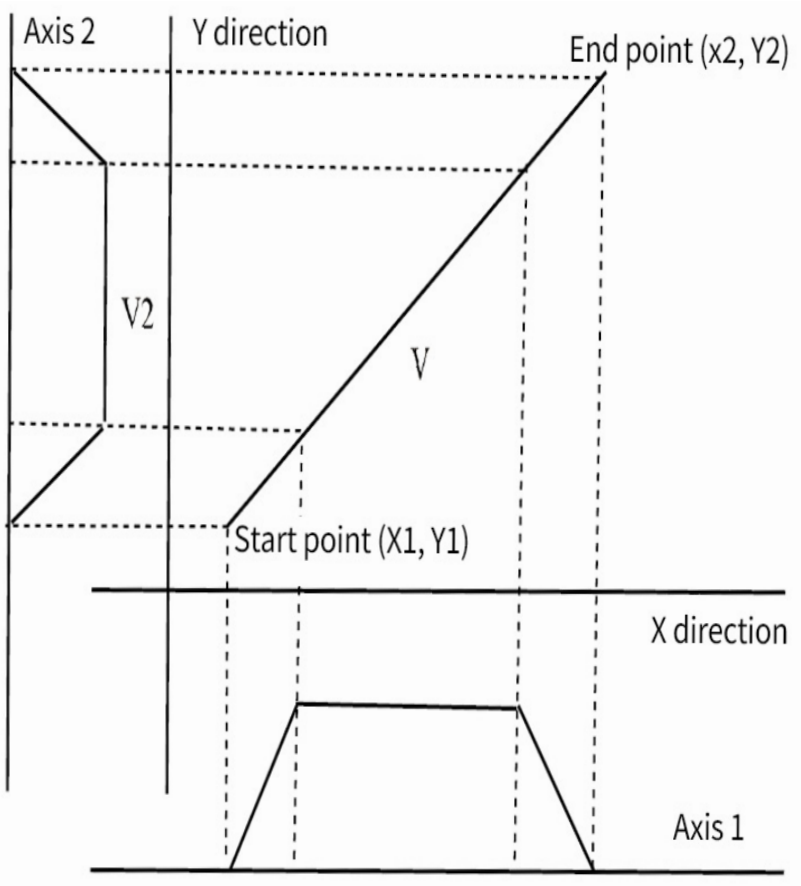
· Specify the coordinate system for linear absolute motion.

Velocity, Acceleration, Deceleration, and Jerk

· If the velocity of any axis (Axis) in the axis group (AxesGroup) exceeds the software-limited maximum velocity of the axis, an error stop will be performed.

· The trajectory profile type of the axis group (AxesGroup) is set as required. If the profile type is trapezoidal, Jerk does not take effect and is regarded as 0. (The profile type of the axis group is specified in the axis group data structure)

· The linear motion is decomposed to each axis when linear absolute motion is performed.



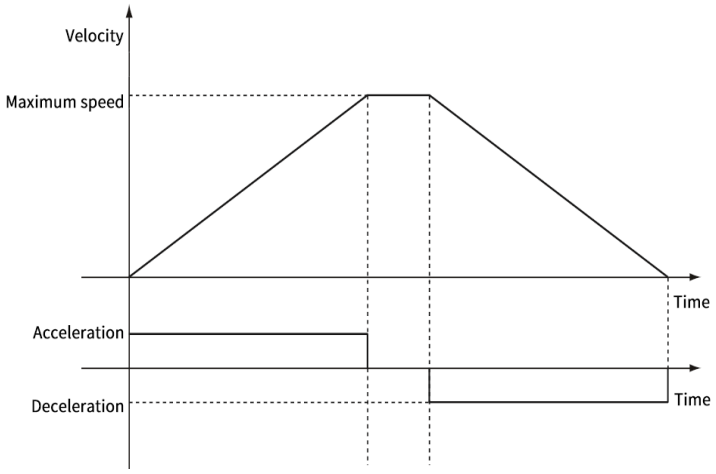
V: Target velocity for the axis group to perform linear motion

V1: Velocity decomposed to axis 1 V1=(X2-X1)*V/ √ ([(X2-X1)]^2+[(Y2-Y1)]^2)

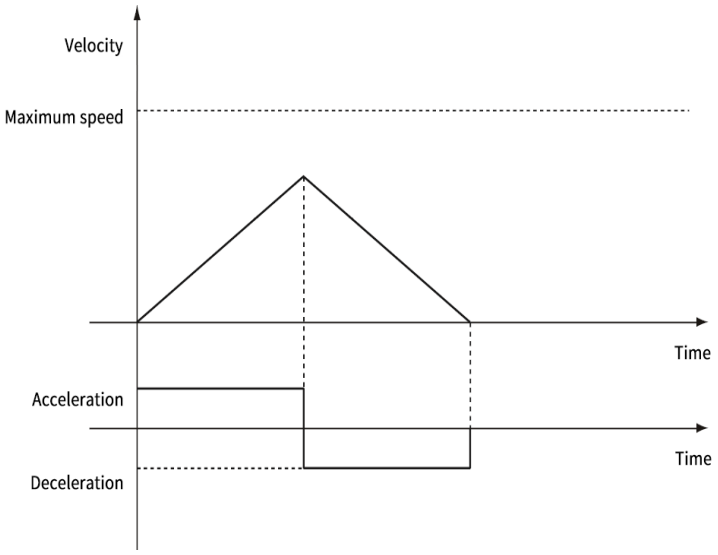
V2: Velocity decomposed to axis 2 V2=(Y2-Y1)*V/ √ ([(X2-X1)]^2+[(Y2-Y1)]^2)

· The curves of velocity, acceleration rate, deceleration rate, and jerk with the conditions Jerk = 0 and Jerk ≠ 0 are shown below.

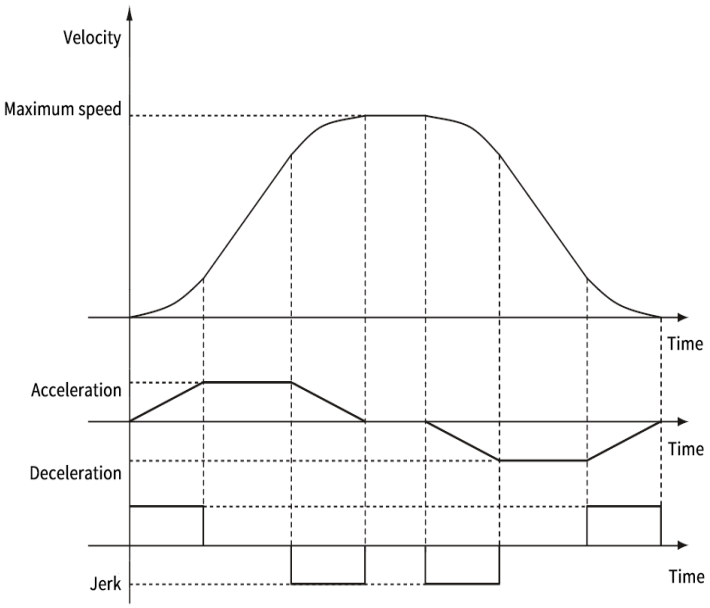
JerkFactor = 0, maximum interpolation velocity reachable



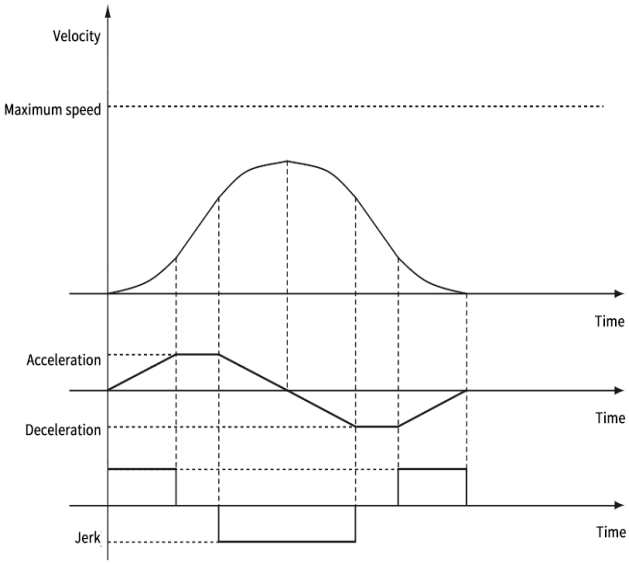
JerkFactor = 0, maximum interpolation velocity unreachable due to short motion distance



JerkFactor ≠ 0, maximum interpolation velocity reachable (Non-trapezoidal profile)



JerkFactor ≠ 0, maximum interpolation velocity unreachable due to short motion distance (Non-trapezoidal profile)



◆ Repeated instruction triggering

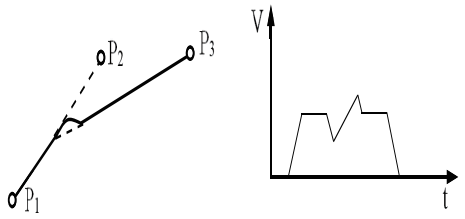
· This instruction cannot be triggered again.

◆ Starting multiple motion instructions

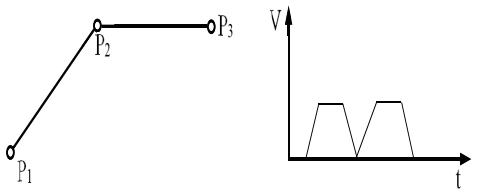
· When this instruction is started for the currently executing instruction, it will be switched to this instruction or this instruction will be buffered. Up to 8 instructions can be buffered for each axis group. The motion of this instruction is specified by BufferMode when multiple instructions are started.

◆ BufferMode

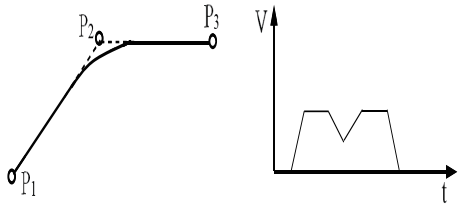
· Aborting (MC_ABORTING): Interrupt the current instruction and start to execute this instruction.



· Buffered (MC_BUFFERED): This instruction is buffered and will be started after the current instruction is executed.



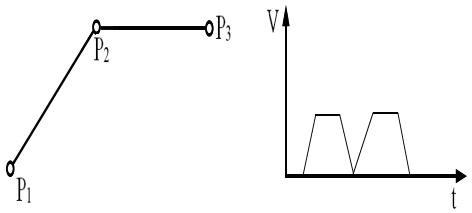
· Blend (MC_BLEND): Blend the velocity of two instructions in the specified way. This method is specified in TransitionMode.



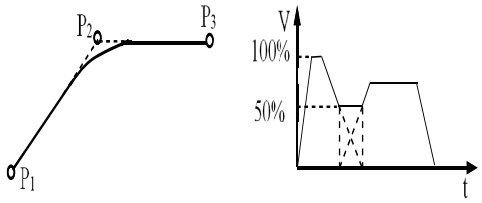
TransitionMode

The blend method of the velocity of two instructions when the input variable TransitionMode of this instruction specifies BufferMode as blend.

· TMNone: No insertion of transition curve If TMNone is adopted as the blend mode, the result is the same as the transition mode between the velocity of the two instructions in buffered mode.



· TMStartVelocity: Blend at the previous velocity Blending is performed for the transition based on a certain ratio (0,1] of the actual maximum velocity of the current motion instruction. Any transition between motion instructions is valid.

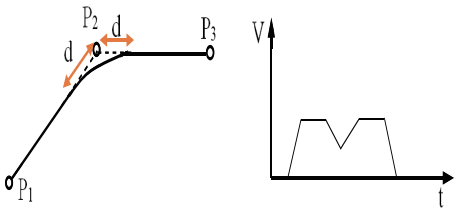


· TMCornerDistance: Corner transition mode In Cartesian space, specify the distance from the previous instruction to the end under the main dimension space. Use this distance to open a transition trajectory to link the neighboring trajectory. The distance here is in a broad sense and is related to the main dimension type of the axis group.

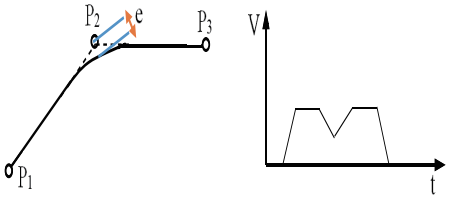
① Position as main dimension: The parameter is the displacement from the transition point to the end point.

② Attitude as main dimension: The parameter is the attitude from the transition point to the end point.

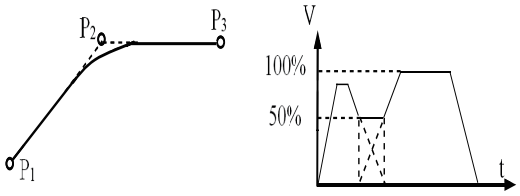
③ Ratio as main dimension: The parameter is the ratio of the resultant distance from the transition point to the end point to the total resultant displacement (0-1).



· TMMaxCornerDeviation: Maximum corner position deviation transition mode In Cartesian space, set the maximum position deviation distance for the trajectory transition of the neighboring instructions in the position subspace. Two neighboring trajectories are linked by constructing the corresponding transition trajectory. In this way, the maximum position deviation of the transition trajectory is always within a certain range. Only the position deviation is involved here, which is not related to the main dimension. Both instructions are valid when they are in Cartesian space (LIN/CIRC). Otherwise, the instructions are processed in no-transition mode.



· TMDefinedVelocity: Blend at the next velocity Blending is performed for the transition based on a certain ratio (0,1] of the actual maximum velocity of the next instruction of the current motion. Any transition between motion instructions is valid.



4) Precautions

- This instruction only supports linear absolute motion under the PCS.
- When the current motion instruction is interrupted, the linear absolute motion immediately starts to plan a new instruction from the interrupted position. In the workspace, a transition trajectory is constructed for deceleration and reverse reset along the velocity direction of the interrupted position, which ensures a smooth link with the latter section of the trajectory. Here, only the velocity smoothing is guaranteed, and the acceleration smoothing is not guaranteed.
- This instruction can only be started when the axis group is in the GroupMoving or GroupStandby state. If it is started in other states, an error will be reported.
- The execution of this instruction can be interrupted by other motion instructions as well as the MC_GroupStop and MC_GroupHalt instructions.
- After the execution of this instruction, the axis group (AxesGroup) state switches from GroupMoving to GroupStandby or retains the GroupMoving state. The state changes as follows:
- If the current motion is in a static coordinate system, the axes group state switches from GroupMoving to GroupStandby. If the current motion is in a dynamic coordinate system, the axis group remains in the GroupMoving state.

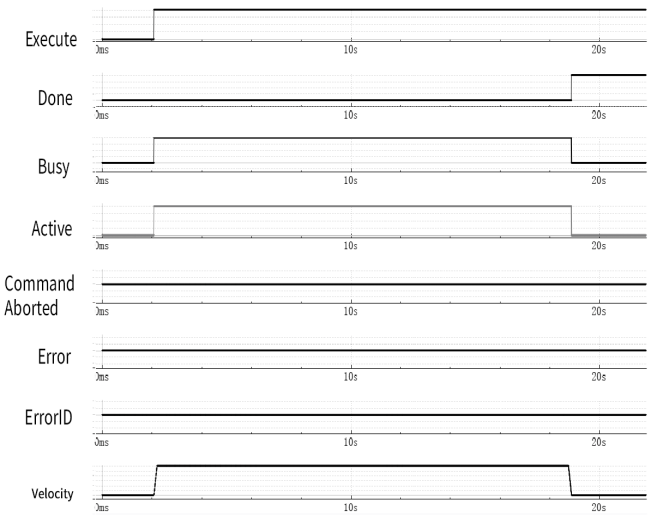
5) Program example

Refer to the MC_MoveLinearRelative instruction for the program example of this instruction. Motion instructions of the axis group (AxesGroup) can be used together.

6) Timing Diagram

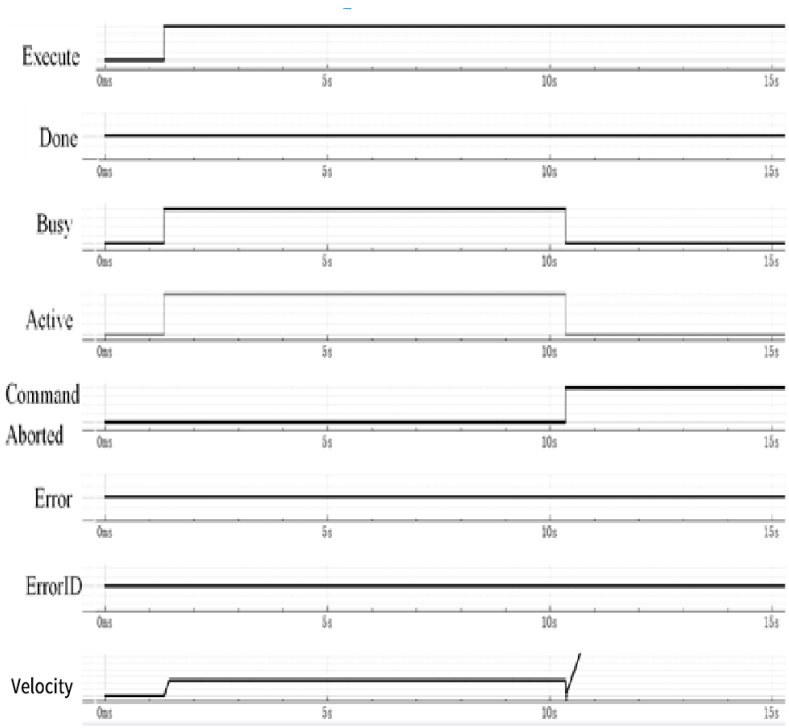
① This instruction operates normally.

Busy and Active change to TRUE when Execute is started. When the target position is reached, the axis group velocity changes to 0, Done becomes TRUE, and Busy and Active become FALSE.

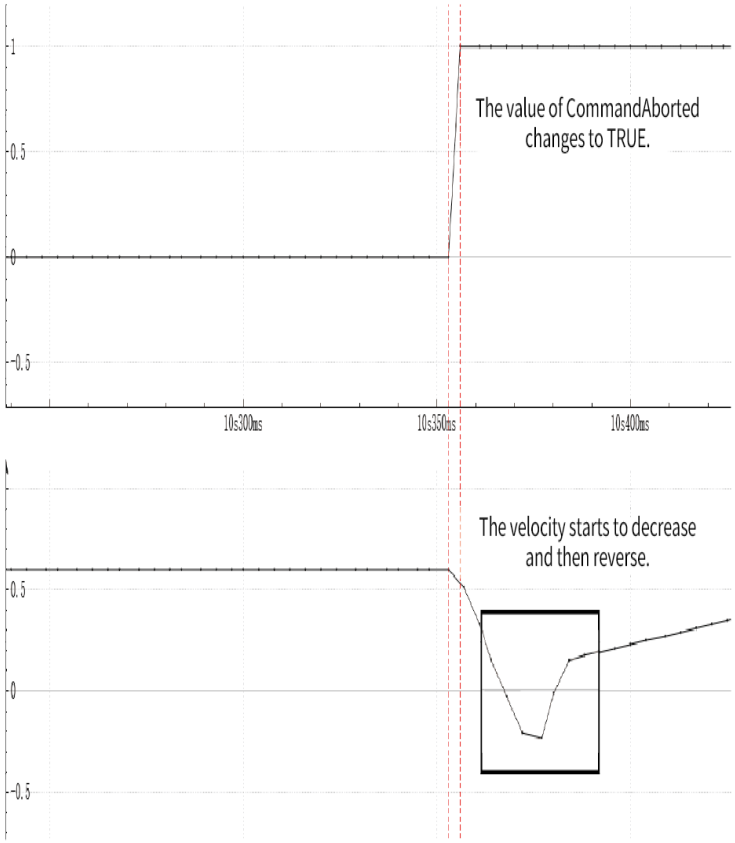


② This instruction is interrupted by another instruction.

Busy and Active change to TRUE when Execute is started. When this instruction is interrupted by another instruction, CommandAborted becomes TRUE and Busy and Active become FALSE.

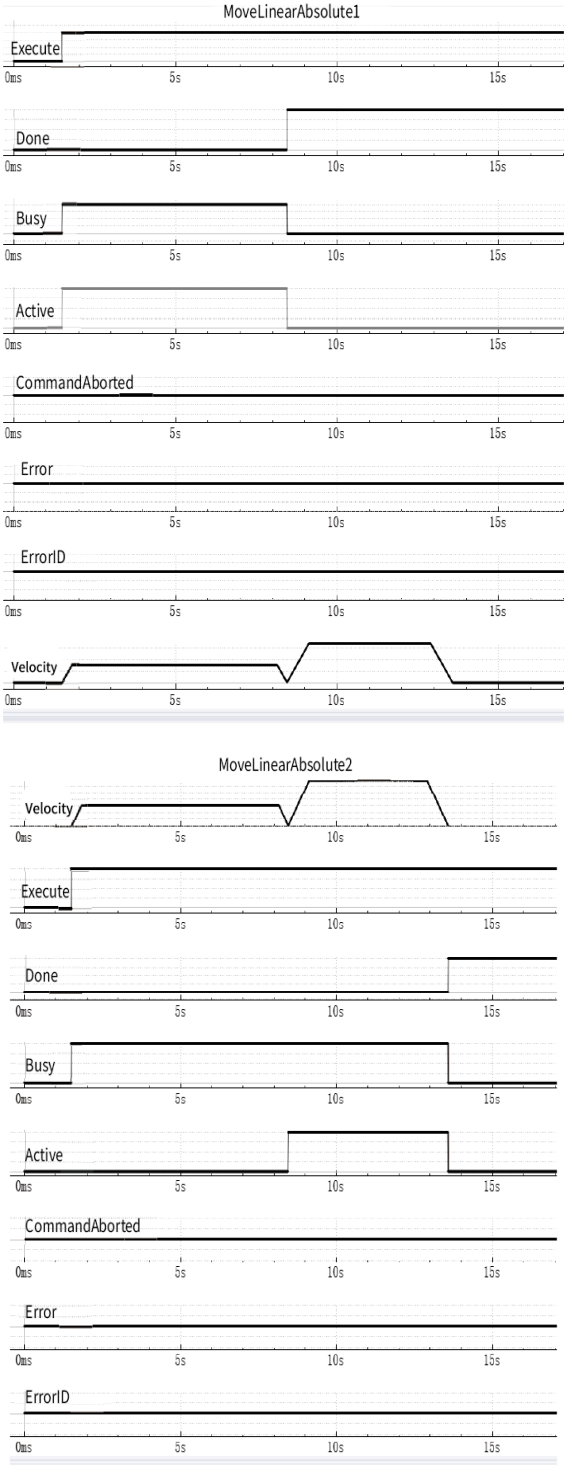


As mentioned in the notes, when the current motion instruction is interrupted, the point-to-point absolute motion immediately starts to plan a new instruction from the interrupted position. In the joint space, a transition trajectory is constructed for deceleration and reverse reset along the velocity direction of the interrupted position, which ensures a smooth link with the latter section of the trajectory.



④ The current instruction is followed by a buffered instruction.

Execute of MoveLinearAbsolute1 and MoveLinearAbsolute2 are started at the same time. Busy and Active of MoveLinearAbsolute1 change to TRUE and Busy of MoveLinearAbsolute2 changes to TRUE. The axis group moves according to the parameters of MoveLinearAbsolute1 until Done of MoveLinearAbsolute1 becomes TRUE. In the meantime, Busy and Active of MoveLinearAbsolute1 become FALSE and Active of MoveLinearAbsolute2 becomes TRUE. In this case, the axis group moves according to the parameters of MoveLinearAbsolute2. When the target position of MoveLinearAbsolute2 is reached, the velocity of the axis group becomes 0, Done of MoveLinearAbsolute2 becomes TRUE, and Busy and Active become FALSE.



7) Error Description

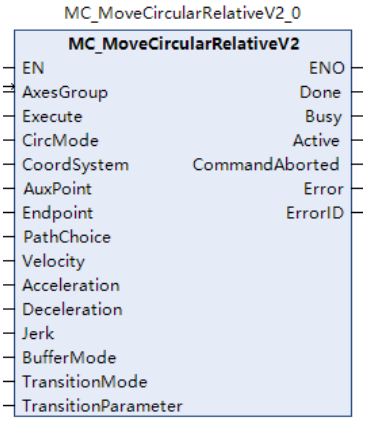
Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.13 MC_MoveCircularRelativeV2

This enables the arc interpolation motion of the axis group from the current TCP point to the relative position point under the given coordinate system.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression

MC_Move CircularRelativeV2	Arc relative motion		MC_MoveCircularRelativeV2_0(AxesGroup:= AxesGroup, Execute:= , CircMode:= , CoordSystem:= , AuxPoint:= , Endpoint:= , PathChoice:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , BufferMode:= , TransitionMode:= , TransitionParameter:= , Done=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge
CircMode	Arc mode	MC_CIRCMODE_V2	BORDER_ARC_TYPE =0; ENTER_ARC_TYPE =1; RADIUS_ARC_TYPE =2	0	0: Three-point arc; 1: Center arc; 2: Radius arc.
CoordSystem	Coordinate type	MC_COORDSYSTEM_V2	ACS, PCS	PCS	Currently, only PCS is supported
AuxPoint	Auxiliary point	ARRAY[0…gc_iAxis_Num-1] OF LREAL	-	[0,0,…]	Position of the auxiliary point relative to the starting point for each dimension in the given coordinate system
EndPoint	End point	ARRAY[0…gc_iAxis_Num-1] OF LREAL	-	[0,0,…]	Position of the end point relative to the starting point for each dimension in the given coordinate system
PathChoice	Path choice	MC_CIRC_PATHCHOICE_V2	SHORT_ARC=0; LONG_ARC=1	SHORT_ARC	Clockwise and anti-clockwise paths (center arc); Long or short path (radius arc)
Velocity	Velocity	LREAL	> 0	-	Maximum main jerk, not necessarily reachable
Acceleration	Acceleration rate	LREAL	> 0	-	Maximum linear acceleration rate, not necessarily reachable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Deceleration	Deceleration rate	LREAL	> 0	-	Maximum linear deceleration rate, not necessarily reachable
Jerk	Jerk	LREAL	>=0	-	Maximum linear jerk, not necessarily reachable
BufferMode	Buffer mode	MC_BUFFER_MODE_V2	MC_ABORTING=0; MC_BUFFERED=1; MC_BLEND_LOW=2; MC_BLEND_PRE=3; MC_BLEND_NEXT=4; MC_BLEND_HIGH=5	MC_ABORTING	Aborting, buffered, blend
Transition Mode	Transition mode	MC_TRANSITION_MODE_V2	>TMNone=0; TMStartVelocity=1; TMCornerDistance=3; TMMaxCornerDeviation=4; TMDefinedVelocity=10; TM_RESET_ABORT=11; TM_DEC_ABORT=12	TMNone	Define the transition trajectory characteristics and type
Transition Parameter	Transition parameter	LREAL	Velocity [0.0,1.0] Distance [0.0,+ ∞)	1	The transition parameters vary with the transition mode

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Active	Axis group under control of the instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Command Aborted	Instruction execution aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

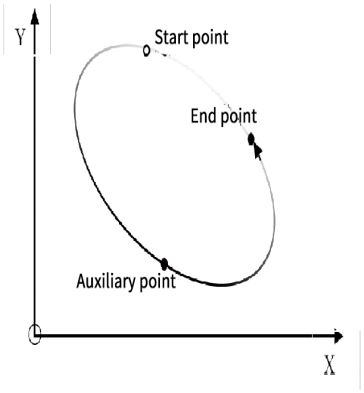
3) Function Description

◆ This instruction is started at the rising edge of Execute and performs arc relative motion from axis 2 to axis 4.

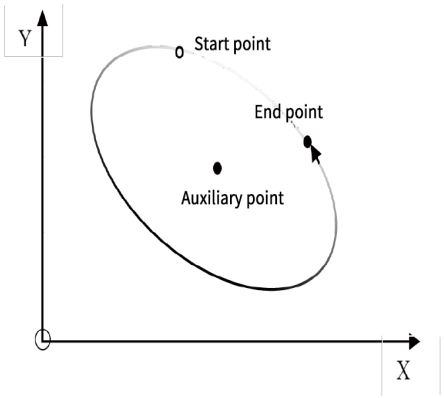
◆ You can specify input variables CircMode, CoordSystem, AuxPoint, EndPoint, PathChoice, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, and TransitionParameter.

◆ CircMode

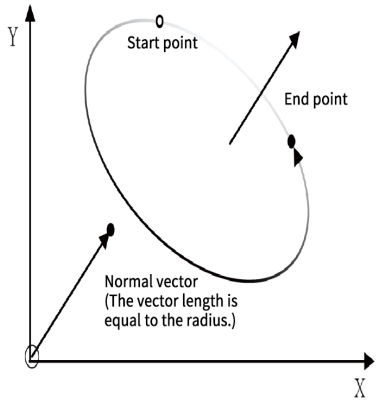
· BORDER_ARC_TYPE (three-point arc): An arc consists of three points: start point (current position of the axis group), auxiliary point (AuxPoint), and end point (EndPoint). If the three points form a straight line or two of them coincide, the trajectory is a straight line connecting the start point and the end point. If the three points do not form a straight line, the trajectory is an arc passing through the three points. The range of the arc angle is (0, 2π). The figure shows the composition of a three-point arc in space, with axis Z facing off-screen.



· ENTER_ARC_TYPE (center arc): An arc is defined by using the current position of the axis group (AxesGroup) as the start point, AuxPoint as the center and EndPoint as the end point. PathChoice specifies whether the arc is long or short. The arc angle range is less than 2π and not equal to π. When the three points form a straight line, the trajectory is a straight line connecting the start and end points. If the distance between the start point and the end point to the center of the circle deviates too much, an error will be reported. If there is a small deviation (<=0.1 mm), the center of the circle will be corrected automatically. The figure shows the composition of a center arc in space, with axis Z facing off-screen.



· RADIUS_ARC_TYPE (radius arc): The radius of the arc is determined by the length of the AuxPoint vector, and the arc normal vector is determined by the vector direction. PathChoice determines whether the arc is long or short. The arc angle range is less than 2π. When the start and end points are not perpendicular to the normal vector, an error will be reported if the deviation angle is too large. If the deviation is small (<10°), the direction of the normal vector will be corrected according to the parameter. If the distance between the start point and the end point is greater than twice the normal vector modulus, the arc reports an error. If the normal vector modulus is less than the calculation accuracy 1e06, an error will be reported.



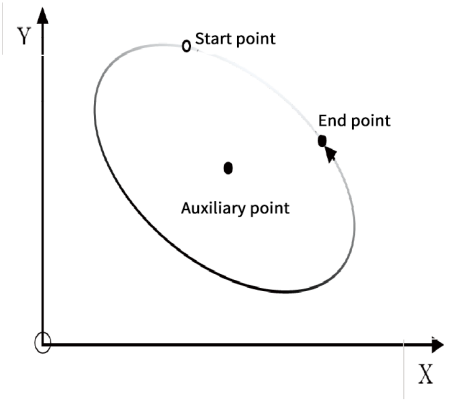
◆ CoordSystem

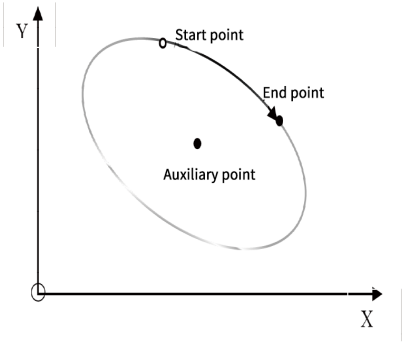
· Specify the coordinate system for arc relative motion. Currently, only arc relative motion under the PCS is supported.

◆ PathChoice

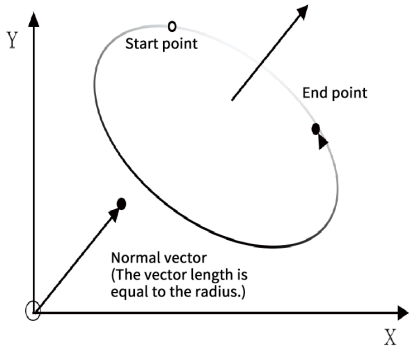
Depending on the CircMode, there are different choices for the arc direction and length.

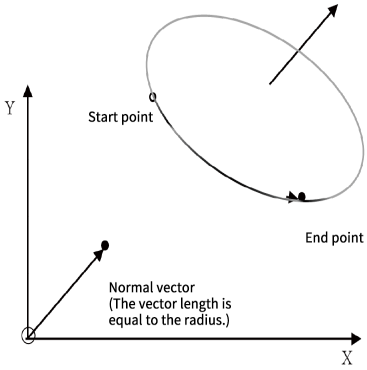
· ENTER_ARC_TYPE (center arc): You can set a clockwise or counterclockwise path.





· RADIUS_ARC_TYPE (radius arc): You can set a long or short path.





◆ Velocity, Acceleration, Deceleration, and Jerk

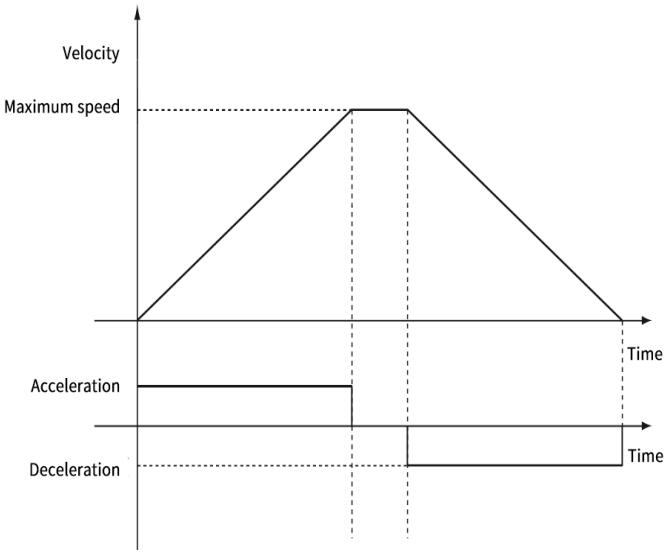
· If the velocity of any axis (Axis) in the axis group (AxesGroup) exceeds the product of the software-

limited maximum velocity of the axis and the warning specification ratio factor, an error stop will be performed.

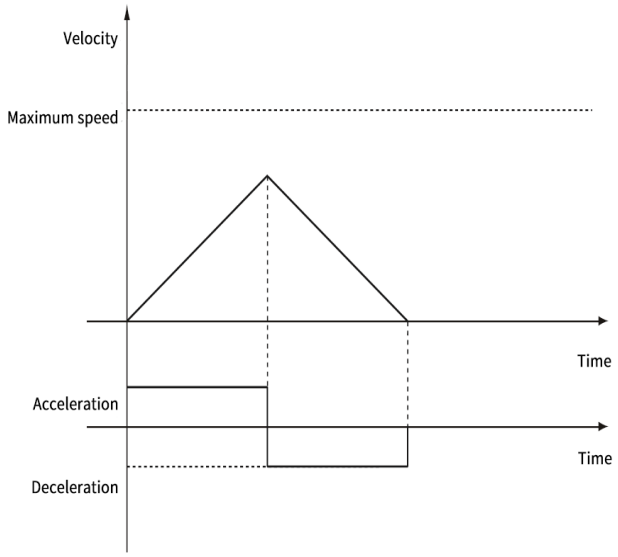
· The trajectory profile type of the axis group (AxesGroup) is set as required. If the profile type is trapezoidal, Jerk does not take effect and is regarded as 0. (The profile type of the axis group is specified in the axis group data structure)

· The curves of velocity, acceleration rate, deceleration rate, and jerk with the conditions Jerk = 0 and Jerk ≠ 0 are shown below.

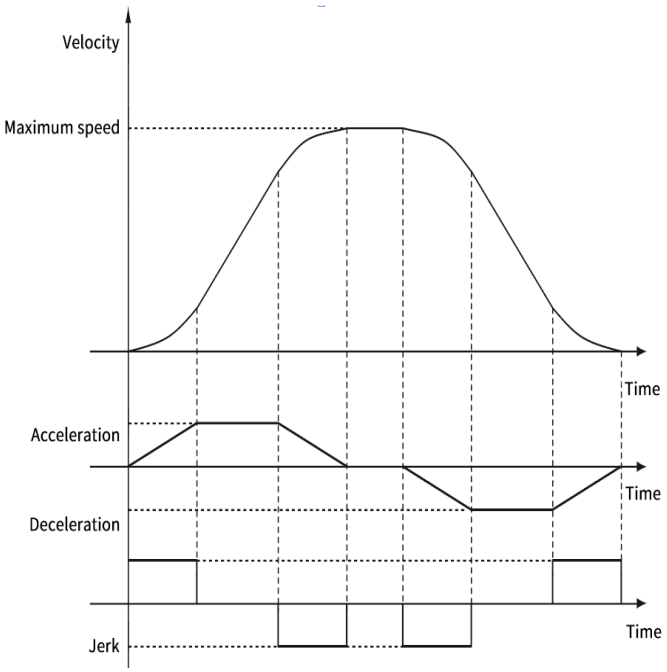
Jerk = 0, maximum interpolation velocity reachable



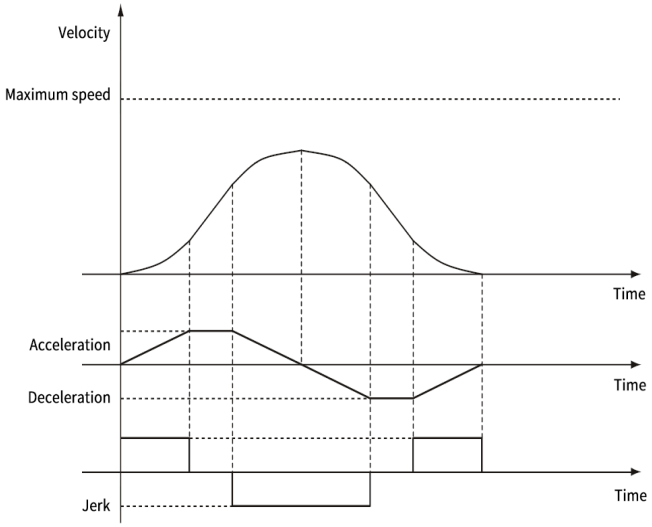
Jerk = 0, maximum interpolation velocity unreachable due to short motion distance



Jerk ≠ 0, maximum interpolation velocity reachable (Non-trapezoidal profile)



Jerk ≠ 0, maximum interpolation velocity unreachable due to short motion distance (Non-trapezoidal profile)



◆ Repeated instruction triggering

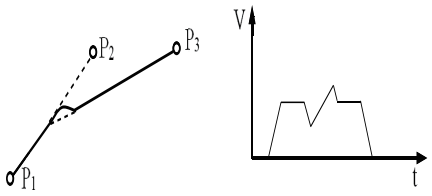
· This instruction cannot be triggered again.

◆ Starting multiple motion instructions

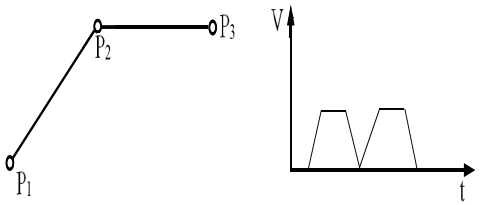
· When this instruction is started for the currently executing instruction, it will be switched to this instruction or this instruction will be buffered. Up to 8 instructions can be buffered for each axis group. The motion of this instruction is specified by BufferMode when multiple instructions are started.

◆ BufferMode

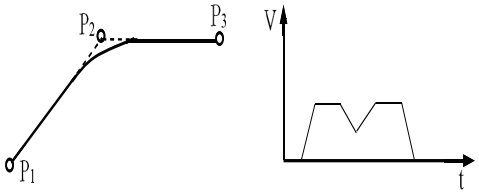
· Aborting (MC_ABORTING): Interrupt the current instruction and start to execute this instruction.



· Buffered (MC_BUFFERED): This instruction is buffered and will be started after the current instruction is executed.



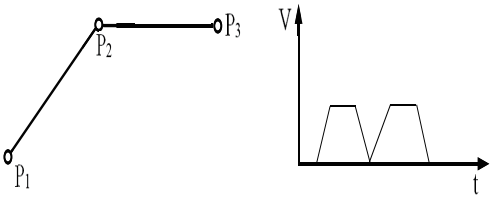
· Blend (MC_BLEND): Blend the velocity of two instructions in the specified way. This method is specified in TransitionMode.



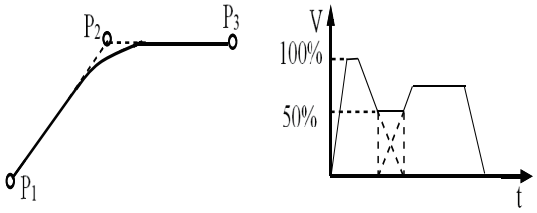
◆ TransitionMode

The blend method of the velocity of two instructions when the input variable TransitionMode of this instruction specifies BufferMode as blend.

· TMNone: No insertion of transition curve If TMNone is adopted as the blend mode, the result is the same as the transition mode between the velocity of the two instructions in buffered mode.



· TMStartVelocity: Blend at the previous velocity Blending is performed for the transition based on a certain ratio (0,1] of the actual maximum velocity of the current motion instruction. Any transition between motion instructions is valid.

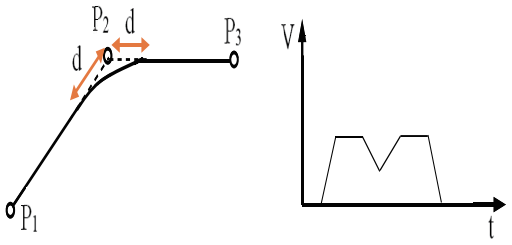


· TMCornerDistance: Corner transition mode In Cartesian space, specify the distance from the previous instruction to the end under the main dimension space. Use this distance to open a transition trajectory to link the neighboring trajectory. The distance here is in a broad sense and is related to the main dimension type of the axis group.

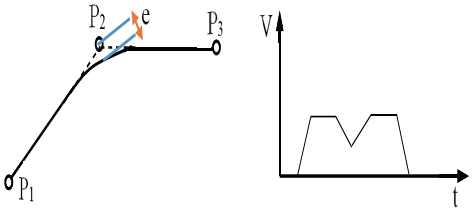
① Position as main dimension: The parameter is the displacement from the transition point to the end point.

② Attitude as main dimension: The parameter is the attitude from the transition point to the end point.

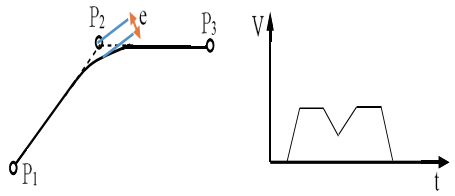
③ Ratio as main dimension: The parameter is the ratio of the resultant distance from the transition point to the end point to the total resultant displacement (0-1).



· TMMaxCornerDeviation: Maximum corner position deviation transition mode In Cartesian space, set the maximum position deviation distance for the trajectory transition of the neighboring instructions in the position subspace. Two neighboring trajectories are linked by constructing the corresponding transition trajectory. In this way, the maximum position deviation of the transition trajectory is always within a certain range. Only the position deviation is involved here, which is not related to the main dimension. Both instructions are valid when they are in Cartesian space (LIN/CIRC). Otherwise, the instructions are processed in no-transition mode.



· TMDefinedVelocity: Blend at the next velocity Blending is performed for the transition based on a certain ratio (0,1] of the actual maximum velocity of the next instruction of the current motion. Any transition between motion instructions is valid.



4) Precautions

- This instruction only supports arc relative motion under the PCS.
- When the current motion instruction is interrupted, the arc relative motion immediately starts to plan a new instruction from the interrupted position. In the workspace, a transition trajectory is constructed for deceleration and reverse reset along the velocity direction of the interrupted position, which ensures a smooth link with the latter section of the trajectory. Here, only the velocity smoothing is guaranteed, and the acceleration smoothing is not guaranteed.
- This instruction can only be started when the axis group is in the GroupMoving or GroupStandby state. If it is started in other states, an error will be reported.
- The execution of this instruction can be interrupted by other motion instructions as well as the MC_GroupStop and MC_GroupHalt instructions.
- After the execution of this instruction, the axis group (AxesGroup) state switches from GroupMoving to GroupStandby or retains the GroupMoving state. The state changes as follows:
- If the current motion is in a static coordinate system, the axes group state switches from GroupMoving to GroupStandby. If the current motion is in a dynamic coordinate system, the axis group remains in the GroupMoving state.

5) Program example

Refer to the MC_MoveLinearRelative instruction for the program example of this instruction. Motion instructions of the axis group (AxesGroup) can be used together.

6) Timing Diagram

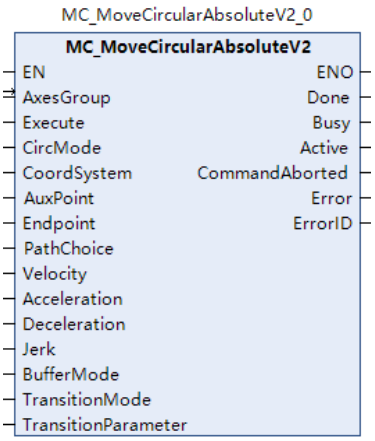
7) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.14 MC_MoveCircularAbsoluteV2

This instruction removes all axes from an axis group by disassociating the mapping of all configured axes.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_Move CircularAbsoluteV2	Arc absolute motion		MC_MoveCircularAbsoluteV2_0(AxesGroup:= AxesGroup, Execute:= , CircMode:= , CoordSystem:= , AuxPoint:= , Endpoint:= , PathChoice:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , BufferMode:= , TransitionMode:= , TransitionParameter:= , Done=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge
CircMode	Arc mode	MC_CIRCMODE_V2	BORDER_ARC_TYPE =0; ENTER_ARC_TYPE =1; RADIUS_ARC_TYPE =2	0	0: Three-point arc; 1: Center arc; 2: Radius arc.
CoordSystem	Coordinate type	MC_COORDSYSTEM_V2	ACS, PCS	PCS	Currently, only PCS is supported
AuxPoint	Auxiliary point	ARRAY[0···gc_iAxis_Num-1] OF LREAL	-	[0,0,···]	Position of the auxiliary point relative to the starting point for each dimension in the given coordinate system
EndPoint	End point	ARRAY[0···gc_iAxis_Num-1] OF LREAL	-	[0,0,···]	Position of the end point relative to the starting point for each dimension in the given coordinate system
PathChoice	Path choice	MC_CIRC_PATHCHOICE_V2	SHORT_ARC=0; LONG_ARC=1	SHORT_ARC	Clockwise and anti-clockwise paths (center arc); Long or short path (radius arc)
Velocity	Velocity	LREAL	> 0	-	Maximum linear velocity, not necessarily reachable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Acceleration	Acceleration rate	LREAL	> 0	-	Maximum linear acceleration rate, not necessarily reachable
Deceleration	Deceleration rate	LREAL	> 0	-	Maximum linear deceleration rate, not necessarily reachable
Jerk	Jerk	LREAL	>=0	-	Maximum linear jerk, not necessarily reachable
BufferMode	Buffer mode	MC_BUFFER_MODE_V2	MC_ABORTING=0; MC_BUFFERED=1; MC_BLEND_LOW=2; MC_BLEND_PRE=3; MC_BLEND_NEXT=4; MC_BLEND_HIGH=5	MC_ABORTING	Aborting, buffered, blend
Transition Mode	Transition mode	MC_TRANSITION_MODE_V2	TMNone=0; TMStartVelocity=1; TMCornerDistance=3; TMMaxCornerDeviation=4; TMDefinedVelocity=10; TM_RESET_ABORT=11; TM_DEC_ABORT=12 TMMaxCornerDeviation=4; TMDefinedVelocity=10; TM_RESET_ABORT=11; TM_DEC_ABORT=12	TMNone	Define the transition trajectory characteristics and type
Transition Parameter	Transition parameter	LREAL	Velocity [0.0,1.0] Distance [0.0,+ ∞)	1	The transition parameters vary with the transition mode

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Active	Axis group under control of the instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Command Aborted	Instruction execution aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

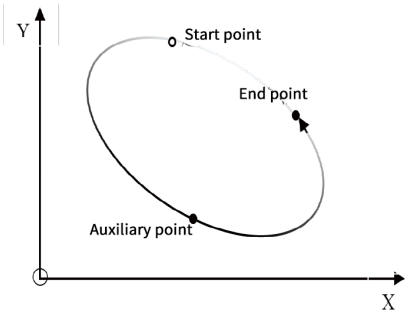
3) Function Description

◆ This instruction is started at the rising edge of Execute and performs arc absolute motion from axis 2 to axis 4.

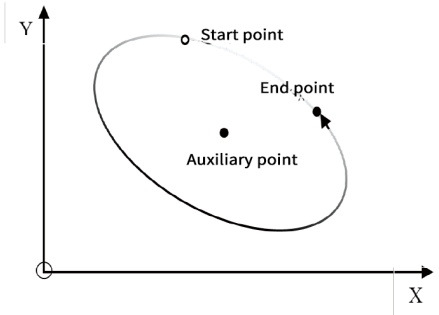
◆ You can specify input variables CircMode, CoordSystem, AuxPoint, EndPoint, PathChoice, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, and TransitionParameter.

◆ CircMode

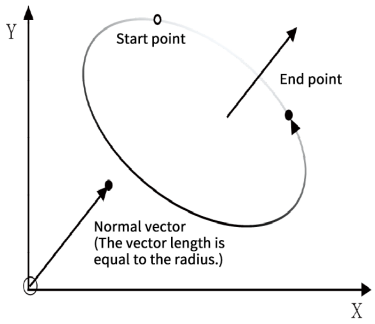
· BORDER_ARC_TYPE (three-point arc): An arc consists of three points: start point (current position of the axis group), auxiliary point (AuxPoint), and end point (EndPoint). If the three points form a straight line or two of them coincide, the trajectory is a straight line connecting the start point and the end point. If the three points do not form a straight line, the trajectory is an arc passing through the three points. The range of the arc angle is (0, 2π). The figure shows the composition of a three-point arc in space, with axis Z facing off-screen.



· ENTER_ARC_TYPE (center arc): An arc is defined by using the current position of the axis group (AxesGroup) as the start point, AuxPoint as the center and EndPoint as the end point. PathChoice specifies whether the arc is long or short. The arc angle range is less than 2π and not equal to π. When the three points form a straight line, the trajectory is a straight line connecting the start and end points. If the distance between the start point and the end point to the center of the circle deviates too much, an error will be reported. If there is a small deviation (<=0.1 mm), the center of the circle will be corrected automatically. The figure shows the composition of a center arc in space, with axis Z facing off-screen.



· RADIUS_ARC_TYPE (radius arc): The radius of the arc is determined by the length of the AuxPoint vector, and the arc normal vector is determined by the vector direction. PathChoice determines whether the arc is long or short. The arc angle range is less than 2π. When the start and end points are not perpendicular to the normal vector, an error will be reported if the deviation angle is too large. If the deviation is small (<10°), the direction of the normal vector will be corrected according to the parameter. If the distance between the start point and the end point is greater than twice the normal vector modulus, the arc reports an error. If the normal vector modulus is less than the calculation accuracy 1e06, an error will be reported.



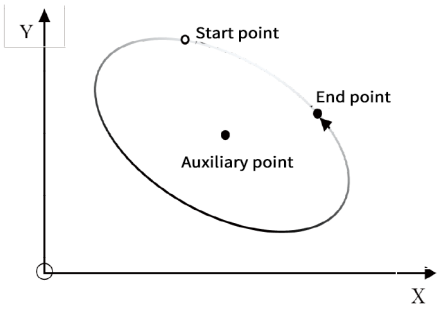
◆ CoordSystem

· Specify the coordinate system for arc relative motion. Currently, only arc relative motion under the PCS is supported.

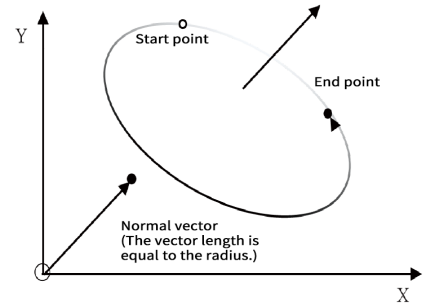
◆ PathChoice

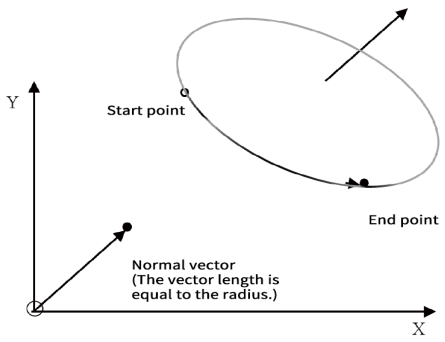
Depending on the CircMode, there are different choices for the arc direction and length.

· ENTER_ARC_TYPE (center arc): You can set a clockwise or counterclockwise path.



· RADIUS_ARC_TYPE (radius arc): You can set a long or short path.





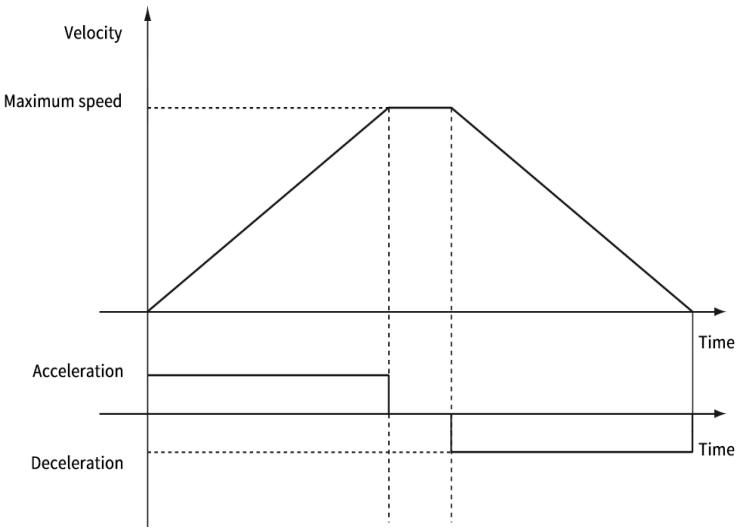
◆ Velocity, Acceleration, Deceleration, and Jerk

· If the velocity of any axis (Axis) in the axis group (AxesGroup) exceeds the software-limited maximum velocity of the axis, an error stop will be performed.

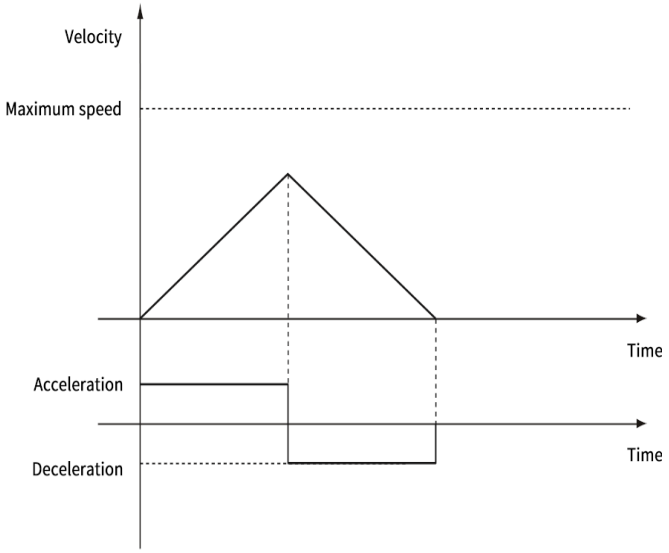
· The trajectory profile type of the axis group (AxesGroup) is set as required. If the profile type is trapezoidal, Jerk does not take effect and is regarded as 0. (The profile type of the axis group is specified in the axis group data structure)

· The curves of velocity, acceleration rate, deceleration rate, and jerk with the conditions Jerk = 0 and Jerk ≠ 0 are shown below.

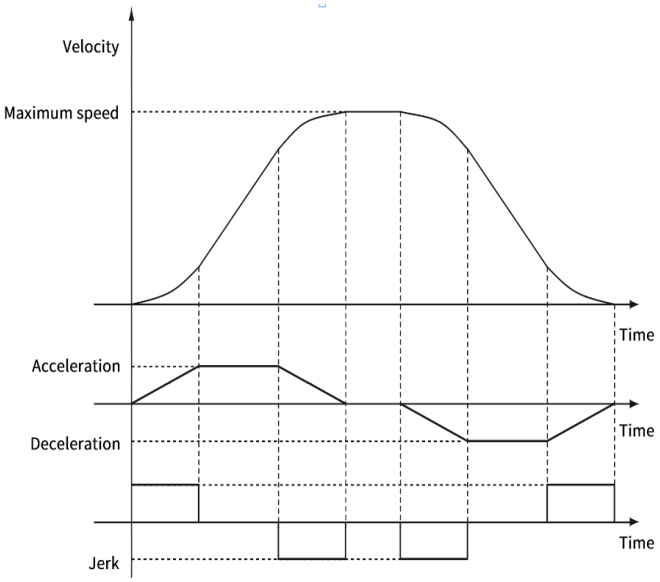
Jerk = 0, maximum interpolation velocity reachable



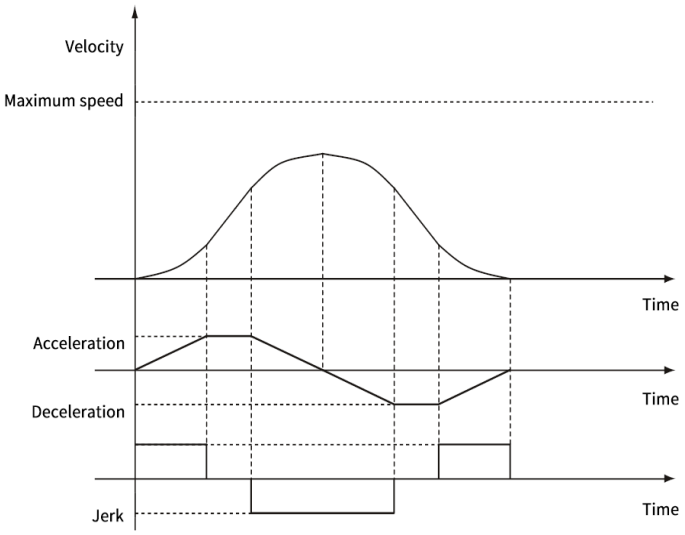
Jerk = 0, maximum interpolation velocity unreachable due to short motion distance



Jerk ≠ 0, maximum interpolation velocity reachable (Non-trapezoidal profile)



Jerk ≠ 0, maximum interpolation velocity unreachable due to short motion distance (Non-trapezoidal profile)



◆ Repeated instruction triggering

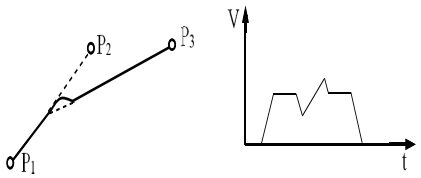
· This instruction cannot be triggered again.

◆ Starting multiple motion instructions

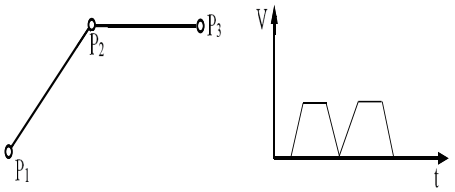
· When this instruction is started for the currently executing instruction, it will be switched to this instruction or this instruction will be buffered. Up to 8 instructions can be buffered for each axis group. The motion of this instruction is specified by BufferMode when multiple instructions are started.

◆ BufferMode

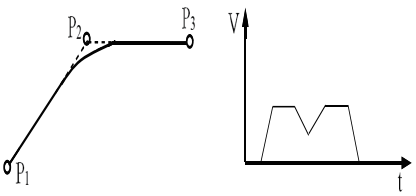
· Aborting (MC_ABORTING): Interrupt the current instruction and start to execute this instruction.



· Buffered (MC_BUFFERED): This instruction is buffered and will be started after the current instruction is executed.



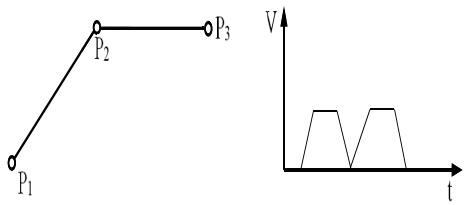
· Blend (MC_BLEND): Blend the velocity of two instructions in the specified way. This method is specified in TransitionMode.



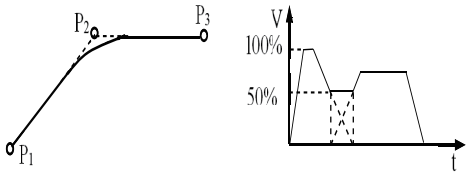
◆ TransitionMode

The blend method of the velocity of two instructions when the input variable TransitionMode of this instruction specifies BufferMode as blend.

· TMNone: No insertion of transition curve If TMNone is adopted as the blend mode, the result is the same as the transition mode between the velocity of the two instructions in buffered mode.



· TMStartVelocity: Blend at the previous velocity Blending is performed for the transition based on a certain ratio (0,1] of the actual maximum velocity of the current motion instruction. Any transition between motion instructions is valid.

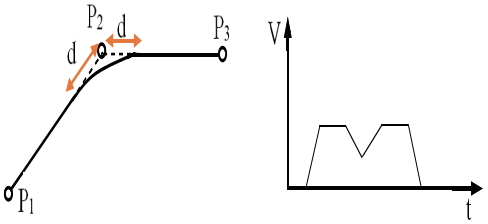


· TMCornerDistance: Corner transition mode In Cartesian space, specify the distance from the previous instruction to the end under the main dimension space. Use this distance to open a transition trajectory to link the neighboring trajectory. The distance here is in a broad sense and is related to the main dimension type of the axis group.

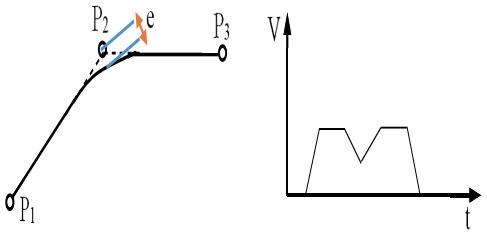
① Position as main dimension: The parameter is the displacement from the transition point to the end point.

② Attitude as main dimension: The parameter is the attitude from the transition point to the end point.

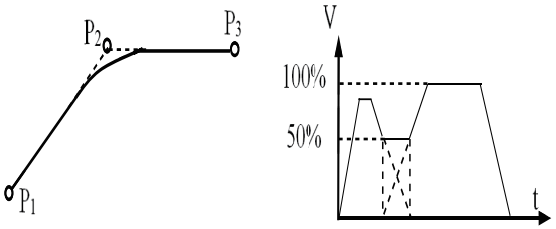
③ Ratio as main dimension: The parameter is the ratio of the resultant distance from the transition point to the end point to the total resultant displacement (0-1).



· TMMaxCornerDeviation: Maximum corner position deviation transition mode In Cartesian space, set the maximum position deviation distance for the trajectory transition of the neighboring instructions in the position subspace. Two neighboring trajectories are linked by constructing the corresponding transition trajectory. In this way, the maximum position deviation of the transition trajectory is always within a certain range. Only the position deviation is involved here, which is not related to the main dimension. Both instructions are valid when they are in Cartesian space (LIN/CIRC). Otherwise, the instructions are processed in no-transition mode.



· TMDefinedVelocity: Blend at the next velocity Blending is performed for the transition based on a certain ratio (0,1] of the actual maximum velocity of the next instruction of the current motion. Any transition between motion instructions is valid.



4) Precautions

- This instruction only supports arc absolute motion under the PCS.
- When the current motion instruction is interrupted, the arc absolute motion immediately starts to plan a new instruction from the interrupted position. In the workspace, a transition trajectory is constructed for deceleration and reverse reset along the velocity direction of the interrupted position, which ensures a smooth link with the latter section of the trajectory. Here, only the velocity smoothing is guaranteed, and the acceleration smoothing is not guaranteed.
- This instruction can only be started when the axis group is in the GroupMoving or GroupStandby state. If it is started in other states, an error will be reported.
- The execution of this instruction can be interrupted by other motion instructions as well as the MC_GroupStop and MC_GroupHalt instructions.
- After the execution of this instruction, the axis group (AxesGroup) state switches from GroupMoving to GroupStandby or retains the GroupMoving state. The state changes as follows:
- If the current motion is in a static coordinate system, the axes group state switches from GroupMoving to GroupStandby. If the current motion is in a dynamic coordinate system, the axis group remains in the GroupMoving state.

5) Program example

Refer to the MC_MoveLinearRelative instruction for the program example of this instruction. Motion instructions of the axis group (AxesGroup) can be used together.

6) Timing Diagram

7) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

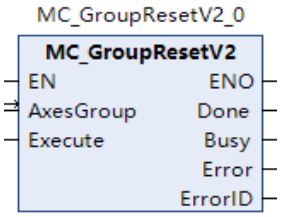
6.4.15 MC_GroupResetV2

This instruction resets an internal error of an axis group to make the axis group state switch from GroupErrorStop to GroupStandby. It also performs the axis error reset operation for all axes in the axis

group.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression

MC_GroupResetV2	Resetting an axis group		`MC_GroupResetV2_0(`
			`AxesGroup:= AxesGroup,`
			`Execute:= ,`
			`Done=> ,`
			`Busy=> ,`
			`Error=> ,`
			`ErrorID=> );`

2) Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ This instruction is started at the rising edge of Execute. It performs the reset operation for errors of an axis group in the GroupErrorStop state and the axis errors in an axis group.

◆ Starting other instructions during the execution of this instruction

· If you execute multiple instantiated MC_GroupReset instructions for the same axis group (AxesGroup), each axis group reset instruction will be executed in the order in which the instructions are triggered.

· If you execute the MC_Reset instruction for an axis (Axis) in the specified axis group (AxesGroup) when executing the MC_GroupReset instruction for the axis group (AxesGroup), the two instructions will be executed sequentially in the order in which they are triggered.

4) Precautions

◆ If you execute this instruction when an axis group communication error occurs or the axis group

(AxesGroup) is in the GroupErrorStop state but the stopping action has not been completed, the axis group reset operation fails and an error will be reported.

- If an axis (Axis) in the axis group (AxesGroup) has an error, execute this instruction to reset all axes (Axis). An error will be reported if the reset times out. After the reset is completed, initialize the axis group (AxesGroup) again, and update the coordinate system (ACS/PCS). The axis group state will switch from GroupErrorStop to GroupStandby.
- Do not execute this instruction in the GroupDisabled state. Otherwise, an error will be reported. If you execute this instruction in a non-GroupErrorStop state, Done is set to TRUE immediately and the axis group remains unchanged.

5) Timing Diagram

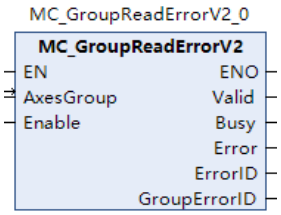
6) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.16 MC_GroupReadErrorV2

This instruction reads the error message of the specified axis group (AxesGroup) and describes the general axis group errors that are not related to the axis group instruction error.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_Group ReadErrorV2	Reading the axis group error message		MC_GroupReadErrorV2_0(AxesGroup:= AxesGroup, Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , GroupErrorID=>);

2) Variables

- Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

- Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Instruction execution	BOOL	TRUE, FALSE	FALSE	TRUE: Execute the instruction; FALSE: Do not execute the instruction

- Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Valid instruction output	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control

Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs
GroupErrorID	Fault code	DWORD	-	0	Axis group error code

3) Function Description

◆ When Valid is TRUE, the error message of the axis group specified by the input/output variable AxesGroup of this instruction is obtained. If there is an axis group error, GroupErrorID outputs the corresponding error code. If there is no error in the axis group, GroupErrorID outputs 0.

4) Precautions

- ◆ When this instruction is executed, the axis group specified by the input/output variable AxesGroup must not be in the GroupDisabled state. There is no requirement on the state of axes (Axis) in the axis group (AxesGroup).
- ◆ The output variable ErrorID of this instruction is the error code corresponding to the error message when an execution error of this instruction occurs. GroupErrorID is the error code corresponding to the error message of the axis group (AxesGroup) read during the execution of this instruction.

5) Timing Diagram

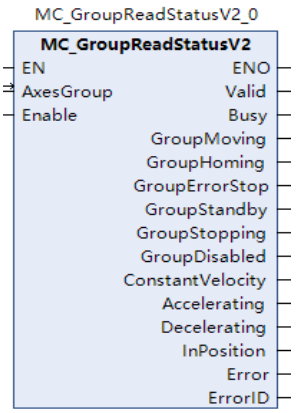
6) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.17 MC_GroupReadStatusV2

This instruction reads the state of the specified axis group.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_GroupReadStatusV2	Obtaining the axis group state		MC_GroupReadStatusV2_0(AxesGroup:= AxesGroup, Enable:= , Valid=> , Busy=> , GroupMoving=> , GroupHoming=> , GroupErrorStop=> , GroupStandby=> , GroupStopping=> , GroupDisabled=> , ConstantVelocity=> , Accelerating=> , Decelerating=> , InPosition=> , Error=> , ErrorID=>);

2) Variables

- ◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Enable	Instruction execution	BOOL	TRUE, FALSE	FALSE	TRUE: Execute the instruction; FALSE: Do not execute the instruction

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Valid	Valid instruction output	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
GroupMoving	Axis group in motion	BOOL	TRUE, FALSE	FALSE	Axis group state machine
GroupHoming	Performing axis group homing	BOOL	TRUE, FALSE	FALSE	Axis group state machine
GroupErrorStop	Axis group stopped due to an error	BOOL	TRUE, FALSE	FALSE	Axis group state machine
GroupStandby	Axis group standby	BOOL	TRUE, FALSE	FALSE	Axis group state machine
GroupStopping	Axis group stopping	BOOL	TRUE, FALSE	FALSE	Axis group state machine
GroupDisabled	Disabling an axis group	BOOL	TRUE, FALSE	FALSE	Axis group state machine
ConstantVelocity	Constant speed	BOOL	TRUE, FALSE	FALSE	Move at constant velocity on the instruction path
Accelerating	Acceleration rate	BOOL	TRUE, FALSE	FALSE	Accelerate on the instruction path
Decelerating	Deceleration rate	BOOL	TRUE, FALSE	FALSE	Decelerate on the instruction path
InPosition	Target position reached	BOOL	TRUE, FALSE	FALSE	Target position reached
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ When Valid is TRUE, this instruction obtains the state of the axis group specified by input/output variable AxesGroup of this instruction.

4) Precautions

◆ When this instruction is executed, there is no requirement on the state of the axis group specified by input/output variable AxesGroup and the state of axes (Axis) in the axis group (AxesGroup)

5) Timing Diagram

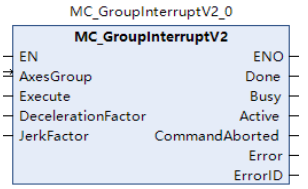
6) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.18 MC_GroupInterruptV2

This instruction pauses an axis group along the current motion path of the axis group, which does not change the current motion state of the axis group.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_Group InterruptV2	Pausing an axis group		MC_GroupInterruptV2_0(AxesGroup:= AxesGroup, Execute:= , DecelerationFactor:= , JerkFactor:= , Done=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Variables

◆ Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

◆ Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge
Deceleration Factor	Deceleration factor	LREAL	(0.0,1.0]	0.3	Ratio relative to the maximum deceleration of the axis group
JerkFactor	Jerk factor	LREAL	[0.0,1.0]	0.3	Ratio relative to the maximum jerk of the axis group

◆ Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Active	Axis group under control of the instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Command Aborted	Instruction execution aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

◆ This instruction is started at the rising edge of Execute. It pauses an axis group (AxesGroup) along the

current motion path of the axis group, which does not change the current motion state of the axis group.

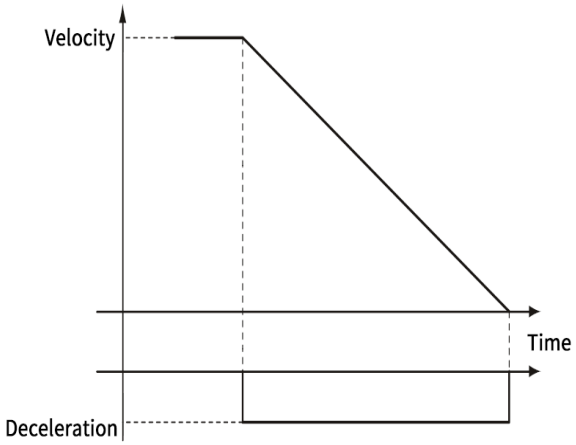
- This instruction does not make the axis group (AxesGroup) pause immediately upon the execution but makes it decelerate based on Deceleration and JerkFactor. After this instruction is started, the current motion instruction of the axis group will not be interrupted. Busy of this instruction is always set. After Active is reset, the axis group maintains the original state.
- To resume the motion paused by this instruction, execute the MC_GroupContinue instruction.
- This instruction can be interrupted by the MC_GroupContinue, MC_GroupHalt, MC_GroupStop, or another MC_GroupInterrupt instruction, as well as by a new axis group (AxesGroup) motion instruction.
- DecelerationFactor and JerkFactor

· The deceleration rate and jerk during the execution of this instruction are determined by the input variables DecelerationFactor and JerkFactor, respectively.

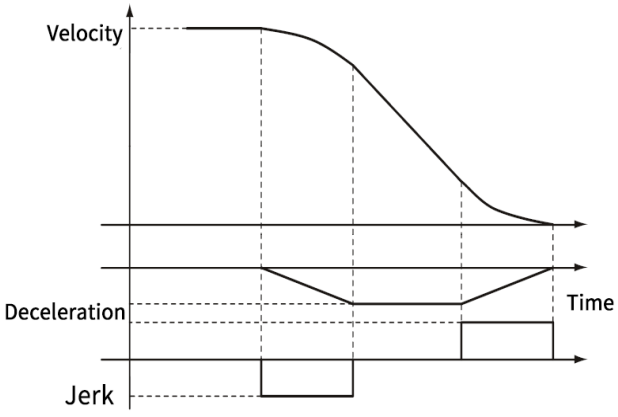
· Each axis (Axis) of the axis group (AxesGroup) pauses according to the product of its software-limited maximum axis acceleration rate and DecelerationFactor as well as the product of its software-limited maximum axis jerk and JerkFactor.

· JerkFactor

① JerkFactor = 0: Velocity profile and deceleration profile



② JerkFactor ≠ 0: Velocity profile, deceleration profile, and jerk profile



- Repeated instruction triggering

· This instruction cannot be triggered again.

4) Precautions

- This instruction cannot be executed when the axis group (AxesGroup) is GroupStandby, GroupErrorstop,

GroupDisabled, or GroupStopping state. Otherwise, an error will be reported.

- This instruction cannot be executed when the axis group (AxesGroup) is controlled by the MC_GroupHalt and in the homing process. Otherwise, an error will be reported.
- The trajectory profile type of the axis group (AxesGroup) is set as required. If the profile type is trapezoidal, the input variable JerkFactor is invalid, and the instruction performs the pause operation based on DecelerationFactor. If the profile type is cubic or septuple, the instruction performs the pause operation based on DecelerationFactor and JerkFactor.
- Whether the profile type of each axis (Axis) in the axis group (AxesGroup) is trapezoidal or S-shaped does not affect the planning method of the axis group (AxesGroup).

5) Timing Diagram

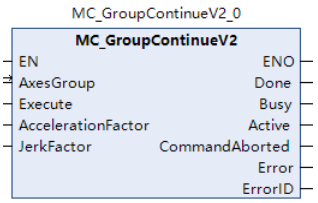
6) Error Description

Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.19 MC_GroupContinueV2

This instruction restarts an axis group and allows the motion that is currently paused or in the process of being paused to return to the original motion path.

1) Instruction Format

Instruction	Name	LD Expression	ST Expression
MC_Group ContinueV2	Restarting an axis group		MC_GroupContinueV2_0(AxesGroup:= AxesGroup, Execute:= , AccelerationFactor:= , JerkFactor:= , Done=> , Busy=> , Active=> , CommandAborted=> , Error=> , ErrorID=>);

2) Variables

- Input/Output Variable

Input/Output Variable	Name	Data Type	Value Range	Initial Value	Description
AxesGroup	Axis group	AXES_GROUP_REF	-	-	Reference to the axis group, that is, an instance of AXES_GROUP_REF

- Input Variable

Input Variable	Name	Data Type	Value Range	Initial Value	Description
Execute	Starting the instruction	BOOL	TRUE, FALSE	FALSE	Started at the rising edge
Acceleration Factor	Acceleration factor	LREAL	(0.0,1.0]	0.3	Ratio relative to the maximum deceleration of the axis group
JerkFactor	Jerk factor	LREAL	[0.0,1.0]	0.3	Ratio relative to the maximum jerk of the axis group

- Output Variable

Output Variable	Name	Data Type	Value Range	Initial Value	Description
Done	Instruction completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the execution of instruction is complete
Busy	Executing instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received
Active	Axis group under control of the instruction	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is under control
Command Aborted	Instruction execution aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted
Error	Instruction error	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs
ErrorID	Fault code	DWORD	-	0	Output an error code when an error occurs

3) Function Description

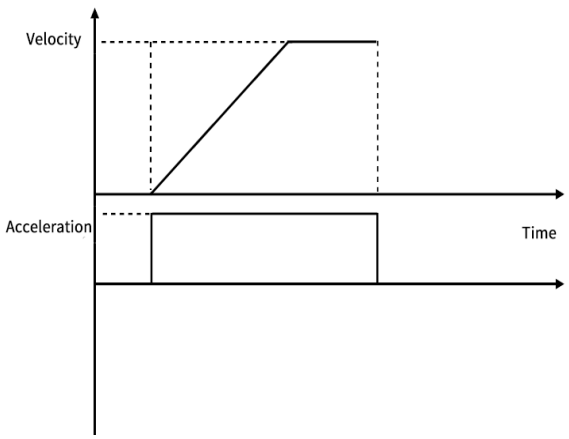
- This instruction is started at the rising edge of Execute. It restarts an axis group and allows the motion that is currently paused or in the process of being paused to return to the original motion path.
- After this instruction is started, Active of the currently paused motion instruction becomes TRUE when Busy and Active of this instruction are set. The axis group accelerates based on the parameters of this instruction until it recovers the velocity at the moment of being paused, and Done of this instruction is set to TRUE.
- This instruction is used in pairs with the MC_GroupInterrupt instruction.
- This instruction can be interrupted by the MC_GroupInterrupt, MC_GroupHalt, MC_GroupStop, or another MC_GroupContinue instruction, as well as by a new axis group (AxesGroup) motion instruction.
- AccelerationFactor, JerkFactor

  · The acceleration rate and jerk during the execution of this instruction are determined by the input variables AccelerationFactor and JerkFactor, respectively.

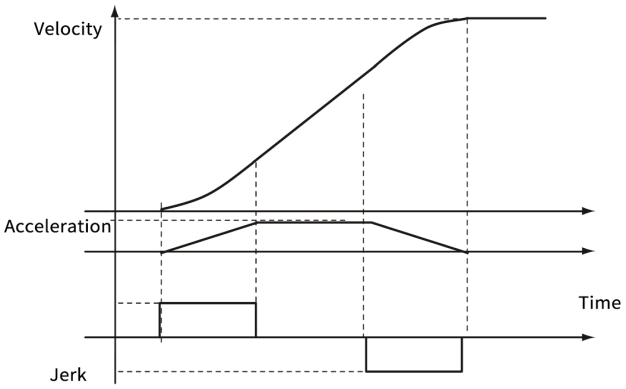
  · Each axis (Axis) of the axis group (AxesGroup) restarts according to the product of its software-limited maximum axis acceleration rate and AccelerationFactor as well as the product of its software-limited maximum axis jerk and JerkFactor.

  · JerkFactor

  ① JerkFactor = 0: Velocity profile and acceleration profile



  ② JerkFactor ≠ 0: Velocity profile, acceleration rate profile, and jerk profile



- Repeated instruction triggering
  - This instruction cannot be triggered again.

4) Precautions

- This instruction cannot be executed when the axis group (AxesGroup) is GroupStandby, GroupErrorstop, GroupDisabled, or GroupStopping state. Otherwise, an error will be reported.
- This instruction cannot be executed when the axis group (AxesGroup) is controlled by the MC_GroupHalt and in the homing process. Otherwise, an error will be reported.
- The trajectory profile type of the axis group (AxesGroup) is set as required. If the profile type is trapezoidal, the input variable JerkFactor is invalid, and the instruction performs the restart operation based on AccelerationFactor. If the profile type is cubic or septuple, the instruction performs the restart operation based on AccelerationFactor and JerkFactor.
- Whether the profile type of each axis (Axis) in the axis group (AxesGroup) is trapezoidal or S-shaped does not affect the planning method of the axis group (AxesGroup).

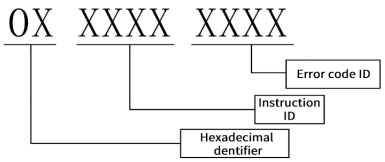
5) Timing Diagram

6) Error Description

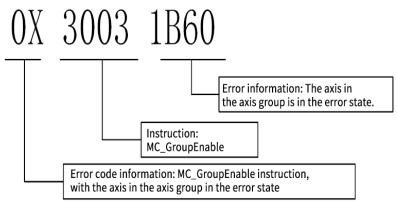
Refer to the error code for the error description, cause, and solution when an error occurs.

6.4.20 Axis Group Error Codes

The fault code for the relevant function is shown below:



0x30031B60 is used as an example for illustration:



- Instruction codes

Command Code (Hexadecimal)	Instruction	Instruction Function
3003	MC_GroupEnable	Enabling an axis group
3004	MC_GroupDisable	Disabling an axis group
3005	MC_GroupHome	Performing axis group homing
3006	MC_GroupSetPosition	Setting the current position of an axis group
3007	MC_GroupReadActualPosition	Reading the actual position of an axis group
3008	MC_GroupReadActualVelocity	Reading the actual velocity of an axis group
3009	MC_GroupReadActualAcceleration	Reading the actual acceleration rate of an axis group
300A	MC_GroupStop	Stopping an axis group immediately
300B	MC_GroupHalt	Stopping an axis group
300C	MC_GroupSetOverride	Setting the axis group ratio
300F	MC_MoveCircularRelative	Arc relative motion
3010	MC_MoveCircularAbsolute	Arc absolute motion
3011	MC_MoveLinearRelative	Linear relative motion
3012	MC_MoveLinearAbsolute	Linear absolute motion
3013	MC_GroupReset	Resetting an axis group
3014	MC_GroupReadError	Reading the axis group error message
3015	MC_GroupReadStatus	Reading the axis group status
3018	MC_GroupInterrupt	Pausing an axis group
3019	MC_GroupContinue	Restarting an axis group

◆ Fault Code

Fault Code (Dec.)	Fault Code (Hex.)	Error Message	Cause	Solution
7001	1B59	Axis type of the axis group is not defined	The axis type of the axis group controlled by this instruction is not defined.	Select the axis type for the single axis in the axis group as required.
7002	1B5A	Axis group cannot be executed in invalid state	The axis group for setting the ratio is in the GROUP_DISABLED or GROUP_HOMING state.	If the axis group for setting the ratio is in the GROUP_DISABLED state, use the MC_GroupEnable instruction to switch the axis group state from the axis group GROUP_DISABLED to GROUP_STANDBY. Then, execute this instruction.
7003	1B5B	Axis group disabled	The axis group controlled by this instruction is in the GROUP_DISABLED state. This instruction can be triggered and executed only when the axis group is in the GROUP_MOVING or GROUP_STANDBY state.	Use the MC_GroupEnable instruction to switch the axis group state to GROUP_STANDBY and then execute this instruction.
7004	1B5C	Axis group in stopping state	The axis group controlled by this instruction is in the GROUP_STOPPING state. This instruction can be triggered and executed only when the axis group is in the GROUP_MOVING or GROUP_STANDBY state.	1. Use the MC_GroupDisable instruction to switch the axis group state from GROUP_STOPPING to GROUP_DISABLED. Then, use the MC_GroupEnable instruction to switch the axis group state to GROUP_STANDBY and then execute this instruction. 2. Check whether the output variable Done of the MC_GroupStop instruction, which makes the axis group change to the GROUP_STOPPING state, is set. If yes, reset the input variable Execute of the MC_GroupStop instruction. When the axis group is in the GROUP_STANDBY state, execute this instruction again.
7005	1B5D	Axis group in homing state	The axis group controlled by this instruction is in the GROUP_HOMING state. This instruction can be triggered and executed only when the axis group is in the GROUP_MOVING or GROUP_STANDBY state.	Wait until the axis group completes the homing instruction and the axis group is in the GROUP_STANDBY state before executing this instruction.
7006	1B5E	Axis group stopped due to an error	The axis group controlled by this instruction is in the GROUP_ERRORSTOP state. This instruction can be triggered and executed only when the axis group is in the GROUP_MOVING or GROUP_STANDBY state.	Execute the MC_GroupReset to switch the axis group state to GROUP_STANDBY and then execute this instruction.

Fault Code (Dec.)	Fault Code (Hex.)	Error Message	Cause	Solution
7007	1B5F	Axis group stopping due to an error	The axis group to be reset has a single axis that enters the AXIS_ERRORSTOP state due to the MC_ImmediateStop instruction.	1. Check whether the axis group to be reset has a single axis that enters the AXIS_ERRORSTOP state due to the MC_ImmediateStop instruction. If yes, execute the MC_Reset instruction to reset the single axis and then execute the MC_GroupReset instruction. 2. Check whether several instantiations of the MC_GroupReset instruction are executed concurrently for the axis group to be reset. If yes, delete the extra instantiation of the MC_GroupReset instruction and execute the MC_GroupReset instruction again.
7008	1B60	Axis in axis group in error state	The current axis group has a single axis that is in the AXIS_POWEROFF state.	Check whether the state of an axis in the current axis group is in the AXIS_POWEROFF state. Execute the MC_Reset instruction to switch the axis state from AXIS_POWEROFF state to AXIS_STANDSTILL or AXIS_POWER_OFF, and then execute this instruction again.
7009	1B61	The period of a single axis in the axis group is not set correctly.	The axis group has an axis of which the task period is less than or equal to 0.	Correct the setting of the task period parameter for the axes in the axis group.
7010	1B62	Axis group controlled by stop instruction	The axis group is executing the MC_GroupHalt instruction.	Wait until the execution of the MC_GroupHalt instruction is completed for the axis group before executing this instruction.
7011	1B63	Axis group controlled by emergency stop instruction	The axis group is executing the MC_GroupStop instruction.	Wait until the execution of the MC_GroupStop instruction is completed for the axis group before executing this instruction.
7012	1B64	Axis group controlled by interrupt instruction	The axis group is controlled by the MC_GroupInterrupt instruction.	Execute the MC_GroupContinue instruction to switch the instruction type of the axis group from GROUP_INTERRUPT to GROUP_SETOVERRIDE, and then execute this instruction.
7016	1B68	Axis not configured in the axis group	-	-
7017	1B69	Axis coordinates exceed positive limit	1. For relative motion, the input variable Distance of this instruction causes the axis group to exceed its positive limit when this instruction is executed. 2. For absolute motion, the input variable Position of this instruction causes the axis group to exceed its positive limit when this instruction is executed.	1. Based on the positive limit, select the appropriate moving distance for relative motion of the axis group before executing this instruction. 2. Based on the positive limit, select the appropriate target position for absolute motion of the axis group before executing this instruction.
7018	1B6A	Axis coordinates exceed negative limit	1. For relative motion, the input variable Distance of this instruction causes the axis group to exceed its negative limit when this instruction is executed. 2. For absolute motion, the input variable Position of this instruction causes the axis group to exceed its negative limit when this instruction is executed.	1. Based on the negative limit, select the appropriate moving distance for relative motion of the axis group before executing this instruction. 2. Based on the negative limit, select the appropriate target position for absolute motion of the axis group before executing this instruction.
7019	1B6B	Axis coordinate velocity exceeds limit	The absolute velocity of a single axis in an axis group under the joint coordinate system (ACS) is greater than the absolute value of the product of the warning planning ratio factor and the axis planning velocity.	Adjust the motion velocity parameter of the axis group as required.
7020	1B6C	Axis group in state of switching part coordinate system	Some function blocks cannot be triggered while the axis group is switching the part coordinate system.	Trigger this instruction again after the switchover is complete.
7021	1B6D	Axis group under dynamic part coordinate system	The axis group is under a dynamic part coordinate system, and the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction has been executed.	The MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction cannot be executed when the axis group is under a dynamic part coordinate system.
7022	1B6E	Axis group in motion	It is prohibited to set the part coordinate system or trigger continuous trajectory function blocks or jog function blocks when the axis group is in the process of interpolation motion.	It is forbidden to trigger the relevant function blocks during the motion of the axis group. Trigger this function block after the interpolation movement is complete.
7023	1B6F	Coupling drive ratio set incorrectly	The coupling factor settings should be reversible. Otherwise, an error will be reported when the axis group is enabled or reset.	Check and modify the coupling factor and drive ratio settings to ensure rationality.
7024	1B70	A single axis in axis group is not enabled	A single axis in the axis group is not enabled when the axis group is in motion or under a dynamic coordinate system.	Check the enabling state of the single axis in the axis group to meet the requirements of the instruction. Then, execute this inspection again.
7026	1B72	Axis in axis group in stopped state	A single axis in the axis group is in the stopped state when the axis group is in motion or under a dynamic coordinate system.	Check the enabling state of the single axis in the axis group to meet the requirements of the instruction. Then, execute this inspection again.

Fault Code (Dec.)	Fault Code (Hex.)	Error Message	Cause	Solution
7040	1B80	Axis group velocity beyond lower limit	1. The value range of input variable VelFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction is (0.0,1.0]. 2. The value of input variable Velocity of the instruction is less than or equal to 0.	1. Correct the value of input variable VelFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of input variable Velocity of the instruction.
7041	1B81	Axis group velocity beyond upper limit	1. The value range of input variable VelFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction is (0.0,1.0]. 2. The value of input variable Velocity of the instruction exceeds the upper limit of the system motion parameter.	1. Correct the value of input variable VelFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of input variable Velocity of the instruction.
7042	1B82	Axis group acceleration rate beyond lower limit	1. The value range of input variable AccFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction is (0.0,1.0]. 2. The value of input variable Acceleration of the instruction is less than or equal to 0.	1. Correct the value of input variable AccFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of input variable Acceleration of the instruction.
7043	1B83	Axis group acceleration rate beyond upper limit	1. The value range of input variable AccFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction is (0.0,1.0]. 2. The value of input variable Acceleration of the instruction exceeds the upper limit of the system motion parameter.	1. Correct the value of input variable AccFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of input variable Acceleration of the instruction.
7044	1B84	Axis group deceleration rate beyond lower limit	1. The value range of input variable AccFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction is (0.0,1.0]. 2. The value of input variable Deceleration of the instruction is less than or equal to 0.	1. Correct the value of input variable AccFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of input variable Deceleration of the instruction.
7045	1B85	Axis group deceleration rate beyond upper limit	1. The value range of input variable AccFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction is (0.0,1.0]. 2. The value of input variable Deceleration of the instruction exceeds the upper limit of the system motion parameter.	1. Correct the value of input variable AccFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of input variable Deceleration of the instruction.
7046	1B86	Axis group jerk beyond lower limit	1. The value range of input variable JerkFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction is [0.0,1.0]. 2. The value of input variable Jerk of the instruction is less than or equal to 0.	1. Correct the value of input variable JerkFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of input variable Jerk of the instruction.
7047	1B87	Axis group jerk beyond upper limit	1. The value range of input variable JerkFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction is [0.0,1.0]. 2. The value of input variable Jerk of the instruction exceeds the upper limit of the system motion parameter.	1. Correct the value of input variable JerkFactor of the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of input variable Jerk of the instruction.
7048	1B88	Axis group velocity factor beyond lower limit	The value range of input variable VelFactor of the MC_GroupSetOverride instruction is [0,100].	Correct the value of input variable VelFactor of the instruction.
7049	1B89	Axis group velocity factor beyond upper limit	The value range of input variable VelFactor of the MC_GroupSetOverride instruction is [0,100].	Correct the value of input variable VelFactor of the instruction.
7050	1B8A	Axis group acceleration factor beyond lower limit	The value range of input variable AccFactor of the MC_GroupSetOverride instruction is [1,100].	Correct the value of input variable AccFactor of the instruction.
7051	1B8B	Axis group acceleration factor beyond upper limit	The value range of input variable AccFactor of the MC_GroupSetOverride instruction is [1,100].	Correct the value of input variable AccFactor of the instruction.
7052	1B8C	Axis group jerk factor beyond lower limit	The value range of input variable JerkFactor of the MC_GroupSetOverride instruction is [0,100].	Correct the value of input variable JerkFactor of the instruction.
7053	1B8D	Axis group jerk factor beyond upper limit	The value range of input variable JerkFactor of the MC_GroupSetOverride instruction is [0,100].	Correct the value of input variable JerkFactor of the instruction.
7054	1B8E	Axis group transition parameter beyond lower limit	The value of input variable TransitionParameter of the instruction varies with the input variable TransitionMode. If TransitionMode is related to Velocity, the value range of TransitionParameter is [0, 1]. If TransitionMode is related to Distance, the value range of TransitionParameter is [0, + ∞).	Correct the value of input variable TransitionParameter of the instruction.
7055	1B8F	Axis group transition parameter beyond upper limit	The value of input variable TransitionParameter of the instruction varies with the input variable TransitionMode. If TransitionMode is related to Velocity, the value range of TransitionParameter is [0, 1]. If TransitionMode is related to Distance, the value range of TransitionParameter is [0, + ∞).	Correct the value of input variable TransitionParameter of the instruction.

Fault Code (Dec.)	Fault Code (Hex.)	Error Message	Cause	Solution
7056	1B90	Axis group buffer mode is not defined	The value of input variable BufferMode of the instruction is out of range.	Select a buffer mode as required to make the axis group move.
7057	1B91	Axis group transition mode is not defined	The value of input variable TransitionMode of the instruction is out of range.	Select a transition mode as required to make the axis group move.
7058	1B92	Axis group transition mode not allowed	1. If the input variable BufferMode of the MC_MoveXXXRelative or MC_MoveXXXAbsolute instruction is Blend, the input variable TransitionMode of this instruction cannot be TMResetAbort or TMDecAbort. 2. If the input variable BufferMode of the MC_MoveXXXRelative or MC_MoveXXXAbsolute Mandatory is Aborting, the input variable TransitionMode of this instruction must be TMResetAbort, TMDecAbort, or TMnone.	Select a transition mode as required to make the axis group move.
7059	1B93	Axis group arc mode is not defined	The value of input variable CircMode of the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction is out of range.	Select an arc mode to make the axes group perform arc motion as required.
7060	1B94	Axis group arc path not defined	The value of input variable PathChoice of the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction is out of range.	Select a path as required to make the axis group perform arc motion.
7061	1B95	Failed to reset buffered motion	The instruction is triggered again after the buffer mode of the buffer instruction to MC_ABORTING.	Do not trigger the instruction again after the buffer mode of the buffer instruction to MC_ABORTING.
7063	1B97	Kinematic model does not exist	The kinematic model of the axis group controlled by the instruction does not exist.	Select the axis group type as required.
7065	1B99	Full instruction buffer area of the axis group	Up to 8 instructions can be buffered for an axis group.	Wait until the instructions currently being executed and the buffered instructions of the axis group have been executed before executing other buffered motion instructions.
7068	1B9C	Incorrect communication status of a single axis mounted to axis group	A single axis in the axis group to be enabled has an incorrect communication status.	Check the communication status of each axis in the axis group.
7081	1BA9	Axis group fails to decelerate and quits motion after motion is interrupted	The deceleration time is longer than the motion time of the next instruction when the current instruction is interrupted by the next instruction.	At this time, the axis group is in an error stop state. Execute the axis group reset instruction and then perform the corresponding operation.
7083	1BAB	Axis group path planning abnormally interrupted and pulse stopped	EN/ENO in LD language is adopted. The axis group motion instruction is blocked by EN during execution.	Check whether EN/ENO in LD language is adopted and the axis group motion instruction is blocked by EN during execution.
7091	1BB3	Kinematic model does not exist	The kinematic model of the axis group controlled by the instruction does not exist.	On the IDE interface, choose "PLC" > "Motion Control" > "Axis Group" > Name of axis group for arc relative motion > "Basic Settings" > "Axis Group Type". Select the axis group type as required.
7092	1BB4	Unknown motion instruction	The motion instruction is undefined.	Check whether the motion instruction is correct.
7093	1BB5	Cannot set dynamic coordinate system based on the same axis group	The following axis group is the same as the main axis group.	When setting dynamic coordinates, ensure that the following axis group is not the same as the main axis group.
7100	1BBC	Identifier of the axis to be mounted to the axis group beyond the set range	The identifier of the axis to be added exceeds the maximum number of physical axes allowed.	Check whether the identifier of the axis to be added is appropriate.
7101	1BBD	Axis mounted to the axis group already assigned to an axis group	The axis to be added is already assigned to an axis group.	The axis to be added is already assigned to an axis group and does not need to be added again.
7102	1BBE	Axis object already mounted to current axis needs to be unmounted and then mounted again, axis already added to axis identifier	1. The axis to be added is already mounted to another axis group. 2. The axis is already added to the axis identifier and cannot be added to the axis identifier again.	1. Check whether the axis to be added is correct. 2. Remove the axis to be added from the other axis group, and then execute this instruction.
7103	1BBF	Axis to be removed no longer exists	The axis to be removed is not in the axis group.	The axis to be removed is not in the axis group and does not need to be removed again.
7106	1BC2	The number of single axes in the axis group does not match the axis group type	The number of single axes in the axis group to be enabled does not match the axis group type.	Check whether the single axes in the axis group are removed because of the use of the MC_UngroupAllAxes or MC_RemoveAxisFromGroup instruction. If yes, execute the MC_AddAxisToGroup instruction to add the single axes to the axis group.

Fault Code (Dec.)	Fault Code (Hex.)	Error Message	Cause	Solution
7107	1BC3	Axis already enabled by an axis group	A single axis in the axis group to be enabled is enabled by another axis group.	Check whether any of the axes in the axis group to be enabled are mounted to another enabled axis group. If yes, disable the enabled axis group and then execute this instruction.
7108	1BC4	-	-	-
7109	1BC5	Axis group in stopping state	The axis group is in the GROUP_STOPPING state.	The axis group is in the axis group stopping state. In this case, the MC_GroupHalt instruction cannot be started.
7110	1BC6	Axis state machine in the axis group not satisfied	The axis to be added by the MC_AddAxisToGroup instruction is not in the StandStill state.	Switch the state of the axis to be added to StandStill or PowerOff.
7111	1BC7	A single axis is rotary and cannot be added to an axis group.	1. The axis to be added by the MC_AddAxisToGroup instruction is rotary. 2. The axis to be added by the MC_AddAxisToGroup instruction is an encoder axis. 3. The conveyor axis of the MC_TrackConveyorBelt instruction is rotary. 4. A single axis in the axis group to be enabled by the MC_GroupEnable instruction is rotary. 5. The turnplate axis of the MC_TrackRotaryTable instruction is rotary. 6. The datum axis of the MC_ SetCoordinateTransform instruction is rotary.	Check whether a single axis in the axis group with an instruction error is rotary or whether the axis of the instruction is rotary. If yes, select the appropriate linear axis to execute the instruction.
7112	1BC8	Instruction not allowed when the single axis is in non-CSP or non-CSV mode	1. The axis to be added by the MC_AddAxisToGroup instruction is in non-CSP or non-CSV mode. 2. A single axis in the axis group to be enabled by the MC_GroupEnable instruction is in non-CSP or non-CSV mode.	1. Execute the MC_SetControllerMode instruction to switch the control mode of the axis to CSV or CSP, and then execute the MC_AddAxisToGroup instruction. 2. Execute the MC_SetControllerMode instruction to switch the control mode of the axis to CSV or CSP, and then execute the MC_GroupEnable instruction.
7113	1BC9	Position not periodically refreshed	The configured axis in the axis group space of the MCGroupRead, ActualPosition, MC_GroupReadActualVelocity, or MCGroupReadActualAcceleration instruction is missing, or there is an error in positive kinematics calculation.	1. Configure the axis group again. 2. Check whether the kinematics parameters of the axis group are correct.
7114	1BCA	Axis group object does not exist	The axis group object of the error instruction does not exist.	Check whether the axis group corresponding to the error instruction is configured on the IDE interface. If not, configure the axis group and then execute the corresponding instruction.
7115	1BCB	Axis object is empty	The axis to be added by the MC_AddAxisToGroup instruction does not exist.	The axis to be added by the MC_AddAxisToGroup instruction does not exist. Check whether the axis is configured on the IDE interface. If not, configure this axis and then execute the instruction again.
7116	1BCC	Axis group communication status abnormal	The status of the servo slave to which the single axis mounted to the axis group of the error instruction belongs is abnormal.	1. Check whether the slave device to which the single axis is mounted is a servo drive. 2. Check whether the servo has an alarm message. 3. Check whether the network cable connected to the slave is normal. 4. Replace the network cable. 5. Restart the servo drive.
7117	1BCD	COORDSYSTEM parameter input abnormal	1. The MC_GroupHome instruction only supports homing under the ACS. 2. The MC_GroupSetPosition, MC_ GroupReadActualPosition, MC_ GroupReadActualVelocity, MC_ GroupReadActualAcceleration, MC_ MoveDirectRelative, MC_MoveDirectAbsolute, and MC_MoveLinearAbsolute instructions can only be executed under the ACS or PCS. 3. Currently, the MC_MoveCircularRelative and MC_ MoveCircularAbsolute instructions only support the PCS mode.	Select the appropriate coordinate system parameters according to the error instruction.
7119	1BCF	Axis group type not defined	-	-

Fault Code (Dec.)	Fault Code (Hex.)	Error Message	Cause	Solution
7120	1BD0	Axis group motion cannot be triggered again	This instruction is started again when being executed by the axis group.	1. Check whether multiple identical instantiated instructions are executed in the PLC program. If yes, delete the repeated instantiated instructions and execute this instruction again. 2. This instruction cannot be executed again when being executed by the axis group.
7121	1BD1	Single axis in the axis group is not enabled	A single axis in the axis group of this instruction is not enabled.	Execute the MC_Power instruction to switch the state of a single axis that is in the off state to the standby state, and then execute this instruction again.
7123	1BD3	Axis group homing start point sequence error	The homing sequence of the first axis in the axis group must be 1.	The homing sequence of the first axis in the axis group must be 1.
7124	1BD4	Axis group homing sequence out of range	The homing sequence of an axis in the axis group exceeds the number of axes in the axis group or is smaller than 0.	Correct the homing sequence of the axis group to make it greater than 0 and smaller than or equal to the number of axes in the group.
7125	1BD5	Axis group homing method is incorrect	The homing method of the axis group is not servo homing or host controller homing.	Modify the homing method of the single axis in the axis group as servo homing or host controller homing.
7126	1BD6	Axis group homing sequence is not ascending	The homing sequence of the axis group must be ascending, which can be defined as "1, 1, 1, 1", "1, 2, 3, 4", "1, 2, 2, 2", "1, 2, 2, 3", or "1, 2, 3, 3".	Correct the homing sequence of the axis group to meet the ascending sequence requirement.
7128	1BD8	Limit triggered for the single axis in the axis group, and the planning velocity direction of such a single axis is the same as the limit triggering direction	The limit is triggered for the single axis in the axis group, and the planning velocity direction of such single axis is the same as the limit triggering direction.	Check whether the single axis in the axis group is configured with a limit. If yes, modify the positive and negative limit ranges or uncheck the limit as required.
7129	1BD9	Axis group object switching during axis group operation is not allowed	Another axis group starts the MC_GroupStop instruction again when the instruction is executed by an axis group.	Check whether multiple axis groups execute the same instantiated instruction. If yes, create unrepeated instantiated instructions for each axis group and execute the corresponding instruction again.
7130	1BDA	Point overlapping	The start point and end point of the instruction overlap. It is used for internal determination, and not displayed externally.	-
7131	1BDB	Distance between the start point and end point is too long (larger than the given diameter)	When the input variable CircMode of the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction is RADUS_ARC_TYPE, the improper value of the input variable EndPoint of this instruction causes a long distance between the start point and end point.	Before executing the instruction, select an end point as required to make the axis group perform arc motion.
7132	1BDC	Distance between the start end and end point is too short	When the input variable CircMode of the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction is RADUS_ARC_TYPE, the improper value of the input variable EndPoint of this instruction causes a short distance between the start point and the end point.	Before executing the instruction, select an end point as required to make the axis group perform arc motion.
7133	1BDD	Radius is too short	When the input variable CircMode of the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction is RADUS_ARC_TYPE, the arc radius is too short.	Before executing the instruction, select an end point as required to make the axis group perform arc motion.
7134	1BDE	Normal vector is not perpendicular (error reported if deviation is more than 10 degrees)	When the input variable CircMode of the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction is RADUS_ARC_TYPE, the normal vector is not perpendicular.	Before executing the instruction, select an end point as required to make the axis group perform arc motion.
7135	1BDF	Distance from start point to center of circle is not equal to that from end point to center of circle	When the input variable CircMode of the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction is ENTER_ARC_TYPE, the distance from the start point to the center of circle is not equal to that from the end point to the center of circle.	Before executing the instruction, select an end point as required to make the axis group perform arc motion.
7800	1E78	SM3_Basic library version does not match	The version of SM3_Basic library does not match that of the new Inovance axis library.	Upgrade the SoftMotion version to 1.4.2.0 or later and the SM3_Basic library version to 4.10.4.0 or later.
7810	1E82	Absolute motor mode homing offset beyond limit	The absolute motor mode homing offset is beyond the limit.	Ensure that the absolute motor axis homing offset is less than 2^31-1.

5.5.4 Reference and Dynamic Switchover of the Cam Table

The cam table is stored in an array within the controller. It can be pointed to by a specific MC_CAM_REF variable type, such as the following declaration:

Cam table p: MC_CAM_REF;

Users can assign a value to this variable, which can also be regarded as making it point to a specific cam table:

Cam table p:= Cam0; //Point to the required cam table

```
Cam table p:  MC_CAM_REF;  //Cam table pointer;
TableID:  uint;  //Cam table selection command, which can be set by HMI;
Case TableID of
0:  Cam table p: = Cam table A;
1:  Cam table p: = Cam table B;
2:  Cam table p: = Cam table C;
End_case

MC_CamTableSelect_0(            //Cam relationship
            Master:=    Virtual master axis,
            Slave:=                 Cam slave axis,
            CamTable:=              Cam table p,
            Execute:=   ReSelect,   //Cam table selection, rising edge-triggered
            Periodic:=  TRUE,
            MasterAbsolute:=FALSE,
            SlaveAbsolute:= FALSE);
```

In the above programming example, users can use the value of the MC_CAM_REF variable to achieve the switchover of multiple cam tables.

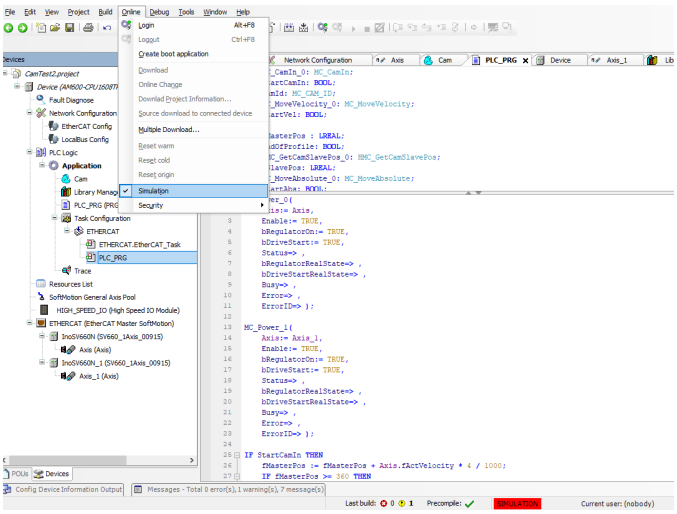


Chapter 7 Simulation and Commissioning

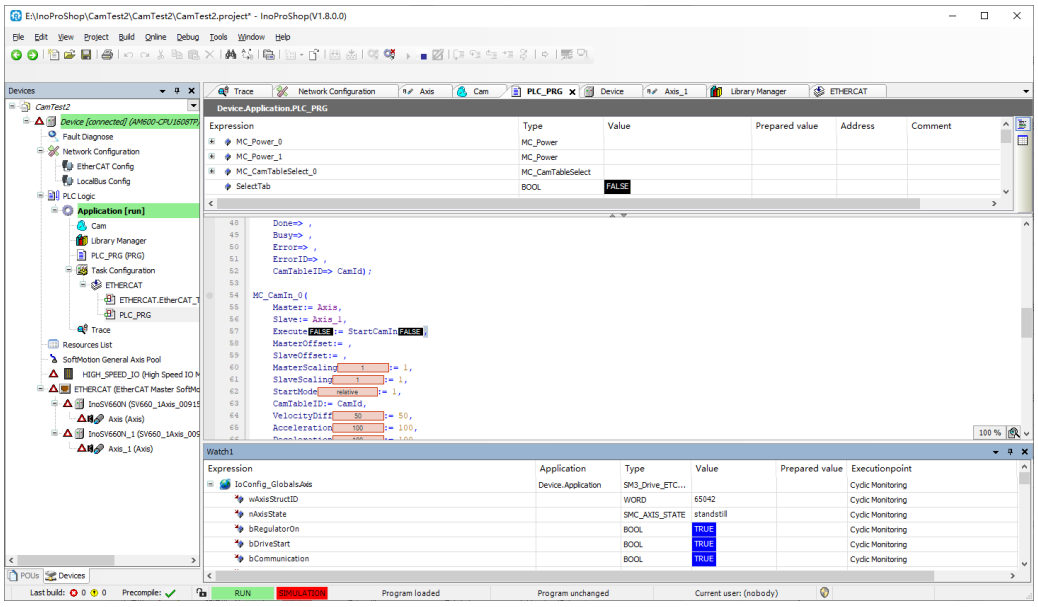
7. Simulation and Commissioning

7.1 Simulation Controller

If no AM600 controller hardware is available for user program commissioning, users can use the simulation function of InoProShop to debug the logic of the user program. The following figure shows how to enable the simulation function. In the simulation state, there is a reminder of simulation state in red at the bottom of the programming software.



In the simulation state, you can also compile the user program and "log in" to the controller. By loading the user program into the PC simulator, you can monitor the user program, forcibly modify the parameters, and observe the execution result of the user program as if you have connected to the controller, as shown in the following figure:

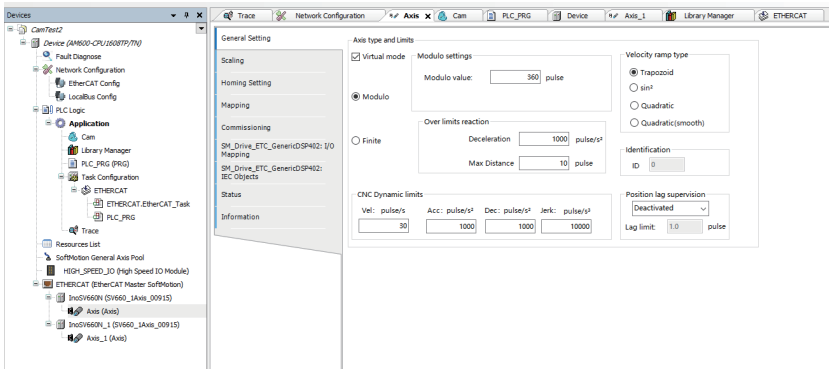


Although it is not possible to simulate the operation of the network bus, users can observe the execution logic of the program and check the execution result after the data structure parameters of the servo axis are forcibly modified.

The steps of the simulated monitoring and commissioning program are the same as those in the scenario with AM600. After "logging in", users can click "Run" or "Stop" to execute or stop the user program. Before modifying the user program, users need to "log out".

7.2 Simulation Servo Drive

If the AM600 controller is available but no servo drive is available or the servo drives are insufficient during MC application commissioning, users can use the "virtual axis" instead of the servo drive axis, as shown in the following figure:



Check the "Virtual Axis Mode" option in the figure. During controller commissioning, the servo drive axis will be simulated. If a physical servo is available, you can uncheck this option.

During programming commissioning, if the number of connected servo axes is different from the number configured in the user program, the system will generate an alarm and the commissioning will fail. After virtual axis is connected, the system will not generate an alarm, but will run by simulating the servo through the software. You can visualize the "running" state of the axis to check correctness of MC program.

Virtual axes are also axes. Although it is a "virtual axis", the axis status operation logic must be designed in accordance with the PLCopen Specification. For example, run MC_Power before the operation. After an error occurs, MC_Reset must be run. This can help debug and eliminate the logic errors in the user program.

If a physical servo axis is connected, just uncheck "Virtual Axis Mode" option shown in preceding figure.



Chapter A Appendix

Appendix A Homing Modes Supported by IS620N

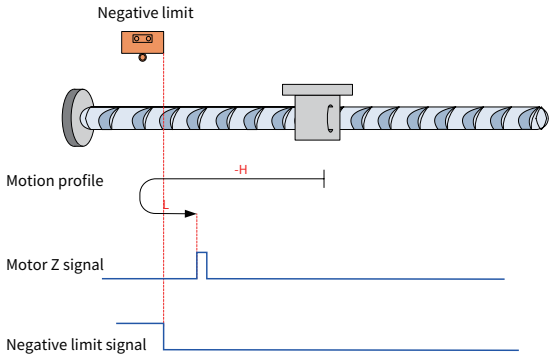
A.1 Description of Homing Modes:

1) 6098h = 1

Mechanical home: motor Z signal

Deceleration point: negative limit switch

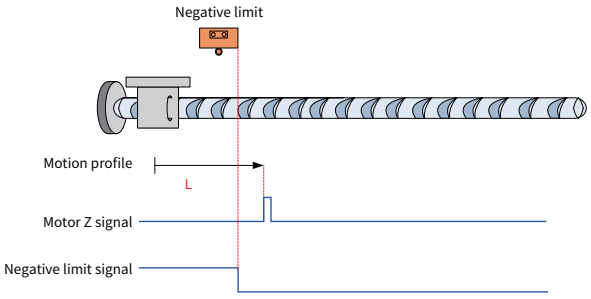
◆ Deceleration point signal inactive at start of homing



Note: In the figure, "H" represents 6099-1h (Speed during search for switch), which is high speed, and "L" represents 6099-2h (Speed during search for zero), which is low speed.

The N-OT signal is inactive initially, and the motor starts homing in negative direction at the high velocity. After reaching the rising edge of the N-OT signal, the motor decelerates and changes to run in positive direction at the low velocity. After reaching the falling edge of the N-OT signal, the motor stops at the first motor Z signal.

◆ Deceleration point signal active at start of homing



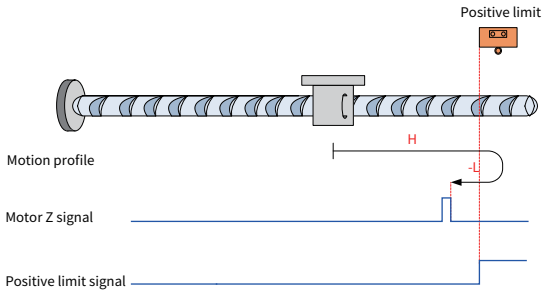
The N-OT signal is active initially, and the motor directly starts homing in positive direction at the low velocity. After reaching the falling edge of the N-OT signal, the motor stops at the first motor Z signal.

2) 6098h = 2

Home: Z signal

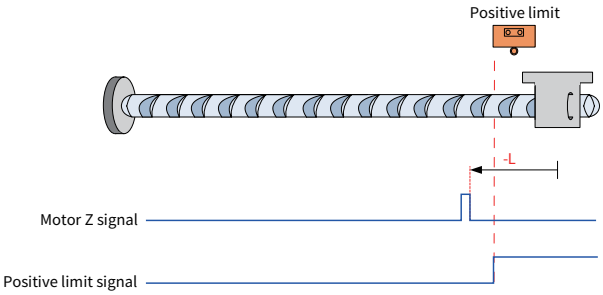
Deceleration point: positive limit switch

◆ Deceleration point signal inactive at start of homing



The P-OT signal is inactive initially, and the motor starts homing in positive direction at the high velocity. After reaching the rising edge of the P-OT signal, the motor decelerates and changes to run in negative direction at the low velocity. After reaching the falling edge of the P-OT signal, the motor stops at the first motor Z signal.

◆ Deceleration point signal active at start of homing



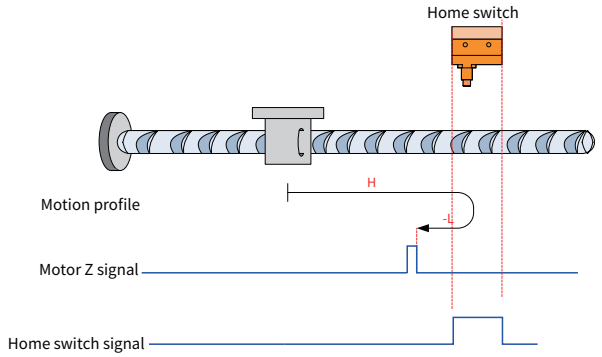
The P-OT signal is active initially, and the motor directly starts homing in negative direction at the low velocity. After reaching the falling edge of the P-OT signal, the motor stops at the first motor Z signal.

3) 6098h = 3

Home: Z signal

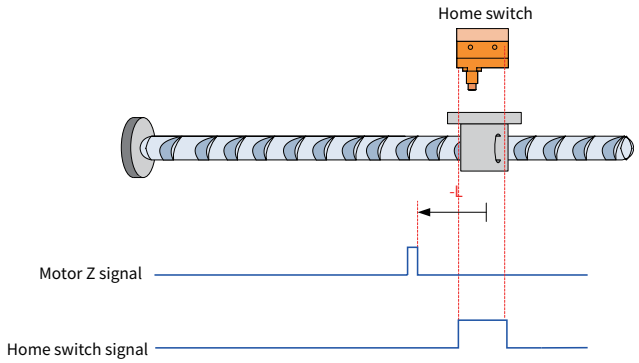
Deceleration point: home switch (HW)

◆ Deceleration point signal inactive at start of homing



The HW signal is inactive initially, and the motor starts homing in positive direction at the high velocity. After reaching the rising edge of the HW signal, the motor decelerates and changes to run in negative direction at the low velocity. After reaching the falling edge of the HW signal, the motor stops at the first motor Z signal.

◆ Deceleration point signal active at start of homing



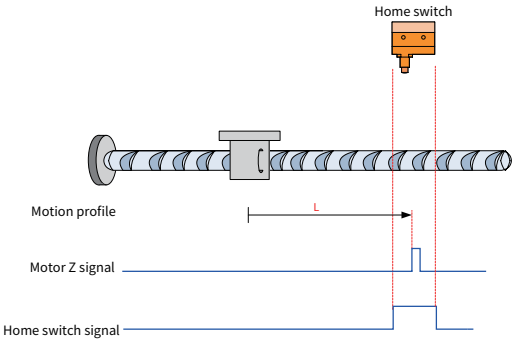
The HW signal is active initially, and the motor directly starts homing in negative direction at the low velocity. After reaching the falling edge of the HW signal, the motor stops at the first motor Z signal.

4) 6098 = 4

Home: Z signal

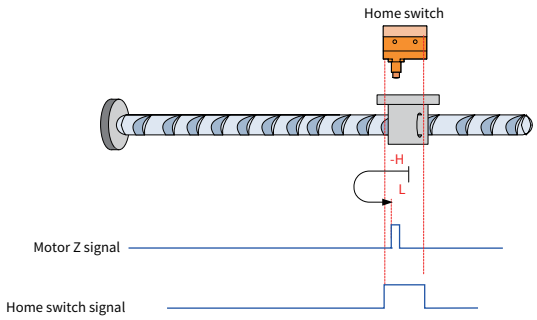
Deceleration point: home switch (HW)

◆ Deceleration point signal inactive at start of homing



The HW signal is inactive initially, and the motor directly starts homing in positive direction at the low velocity. After reaching the rising edge of the HW signal, the motor stops at the first motor Z signal.

◆ Deceleration point signal active at start of homing



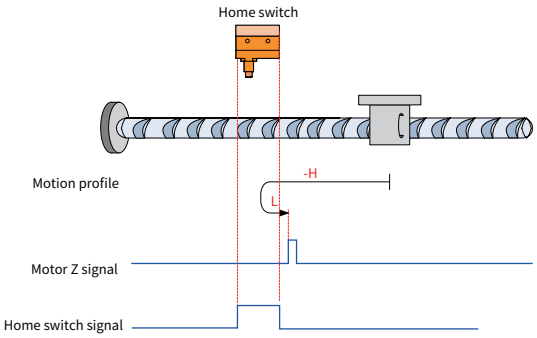
The HW signal is active initially, and the motor starts homing in negative direction at the high velocity. After reaching the falling edge of the HW signal, the motor decelerates and changes to run in negative direction at the low velocity. After reaching the rising edge of the HW signal, the motor stops at the first motor Z signal.

5) 6098h = 5

Home: Z signal

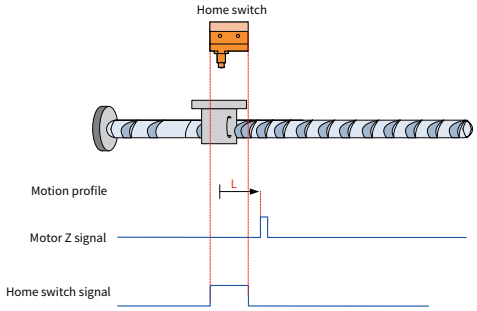
Deceleration point: home switch (HW)

◆ Deceleration point signal inactive at start of homing



The HW signal is inactive initially. The motor starts homing in negative direction at the high velocity. After reaching the rising edge of the HW signal, the motor decelerates and changes to run in positive direction at the low velocity. After reaching the falling edge of the HW signal, the motor stops at the first motor Z signal.

◆ Deceleration point signal active at start of homing



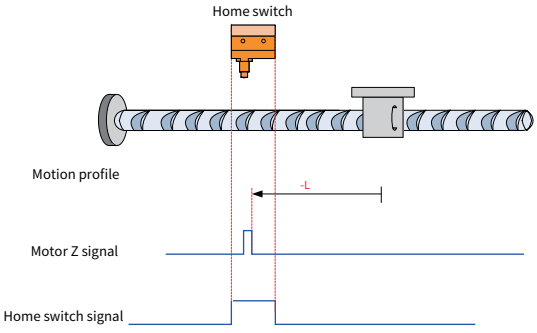
The HW signal is active initially, and the motor directly starts homing in positive direction at the low velocity. After reaching the falling edge of the HW signal, the motor stops at the first motor Z signal.

6) 6098 = 6

Home: Z signal

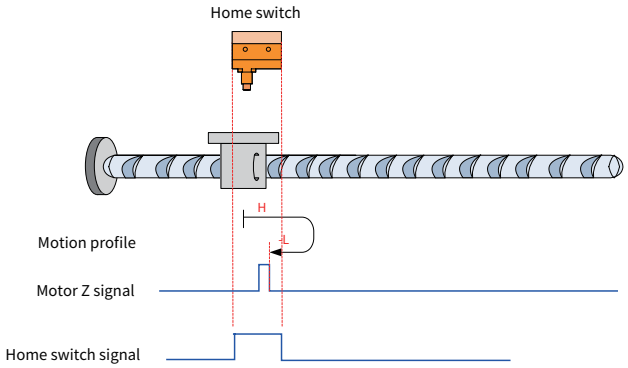
Deceleration point: home switch (HW)

◆ Deceleration point signal inactive at start of homing



The HW signal is inactive initially, and the motor directly starts homing in negative direction at the low velocity. After reaching the rising edge of the HW signal, the motor stops at the first motor Z signal.

◆ Deceleration point signal active at start of homing



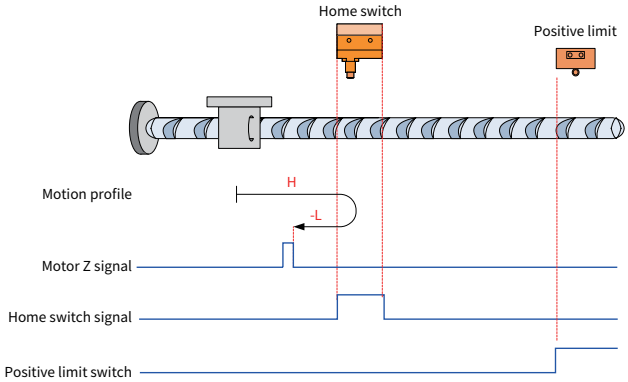
The HW signal is active initially, and the motor starts homing in positive direction at the high velocity. After reaching the falling edge of the HW signal, the motor decelerates and changes to run in negative direction at the low velocity. After reaching the rising edge of the HW signal, the motor stops at the first motor Z signal.

7) 6098 = 7

Home: Z signal

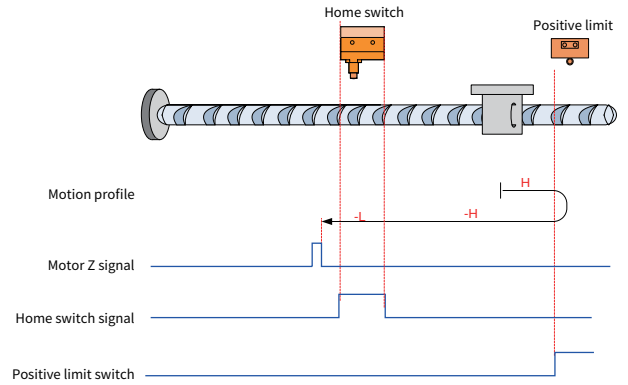
Deceleration point: home switch (HW)

- Deceleration point signal inactive at start of homing, not reaching positive limit switch



The HW signal is inactive initially, and the motor starts homing in positive direction at the high velocity. If the motor does not reach the limit switch, it decelerates and changes to run in negative direction at the low velocity after reaching the rising edge of the HW signal. After reaching the falling edge of the HW signal, the motor stops at the first motor Z signal.

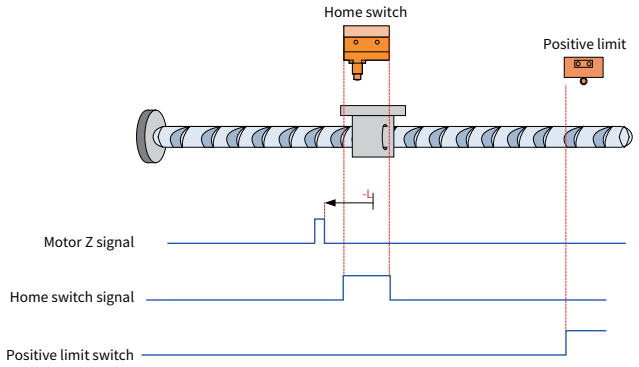
- Deceleration point signal inactive at start of homing, reaching the positive limit switch



The HW signal is inactive initially and the motor starts homing in positive direction at the high velocity. If the motor reaches the limit switch, the motor automatically runs in negative direction at the high velocity. After reaching the rising edge of the HW signal, the motor decelerates and continues to run in negative direction at the low velocity. After reaching the falling edge of the HW signal, the motor stops at

the first motor Z signal.

◆ Deceleration point signal active at start of homing



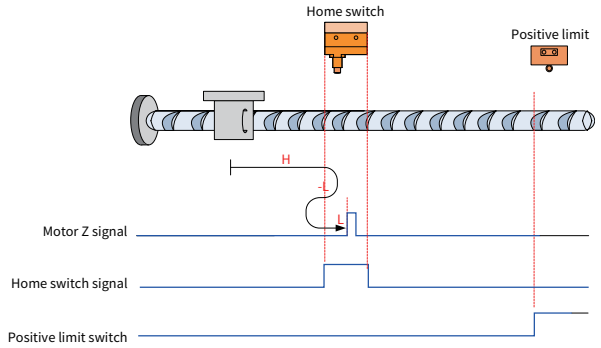
The HW signal is active initially, and the motor directly starts homing in negative direction at the low velocity. After reaching the falling edge of the HW signal, the motor stops at the first motor Z signal.

8) 6098 = 8

Home: Z signal

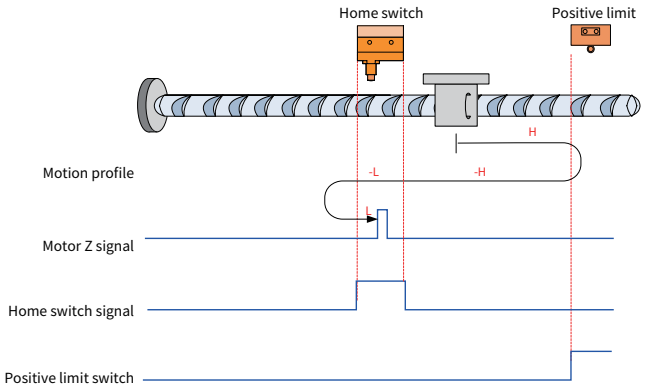
Deceleration point: home switch (HW)

◆ Deceleration point signal inactive at start of homing, not reaching positive limit switch



The HW signal is inactive initially, and the motor starts homing in positive direction at the high velocity. If the motor does not reach the limit switch, it decelerates and changes to run in negative direction at the low velocity after reaching the rising edge of the HW signal. After reaching the falling edge of the HW signal, the motor changes to run in positive direction at the low velocity, and stops at the first motor Z signal.

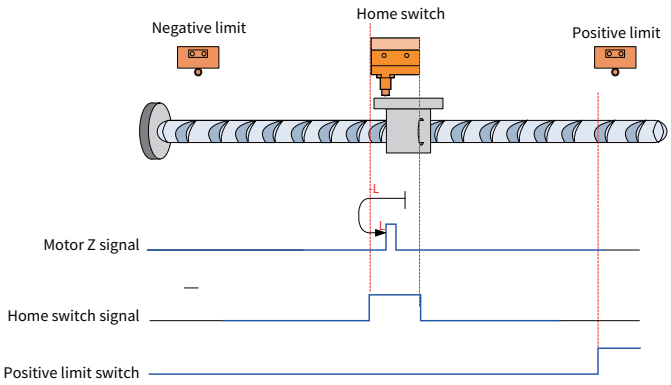
◆ Deceleration point signal inactive at start of homing, reaching the positive limit switch



The HW signal is inactive initially, and the motor starts homing in positive direction at the high velocity. If the motor reaches the limit switch, it automatically changes to run in negative direction at the high velocity. After reaching the rising edge of the HW signal, the motor decelerates and continues to

run in negative direction at the low velocity. After reaching the falling edge of the HW signal, the motor changes to run in positive direction at the low velocity, and stops at the first motor Z signal.

◆ Deceleration point signal active at start of homing



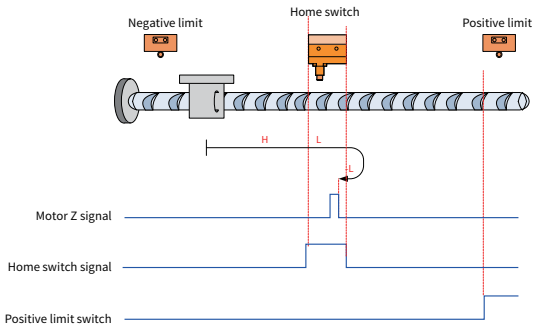
The HW signal is active initially, and the motor directly starts homing in negative direction at the low velocity. After reaching the falling edge of the HW signal, the motor changes to run in positive direction at the low velocity. After reaching the rising edge of the HW signal, the motor stops at the first motor Z signal.

9) 6098 = 9

Home: Z signal

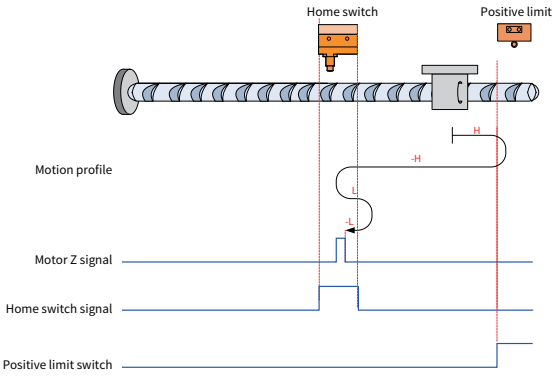
Deceleration point: home switch (HW)

◆ Deceleration point signal inactive at start of homing, not reaching positive limit switch



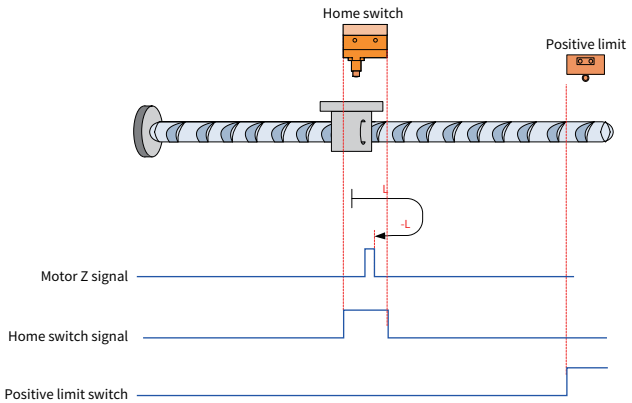
The HW signal is inactive initially, and the motor starts homing in positive direction at the high velocity. If the motor does not reach the limit switch, it decelerates and changes to run in positive direction at the low velocity after reaching the rising edge of the HW signal. After reaching the falling edge of the HW signal, the motor changes to run in negative direction at the low velocity, and stops at the first motor Z signal.

◆ Deceleration point signal inactive at start of homing, reaching the positive limit switch



The HW signal is inactive initially, and the motor starts homing in positive direction at the high velocity. If the motor reaches the limit switch, it automatically changes to run in negative direction at the high velocity. After reaching the rising edge of the HW signal, the motor decelerates and resumes running in positive direction at the low velocity. After reaching the falling edge of the HW signal, the motor changes to run in negative direction at the low velocity, and stops at the first motor Z signal.

◆ Deceleration point signal active at start of homing



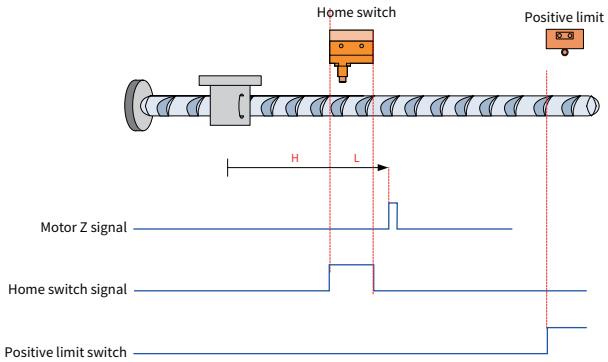
The HW signal is active initially, and the motor directly starts homing in positive direction at the low velocity. After reaching the falling edge of the HW signal, the motor changes to run in negative direction at the low velocity. After reaching the rising edge of the HW signal, the motor stops at the first motor Z signal.

10) 6098 = 10

Home: Z signal

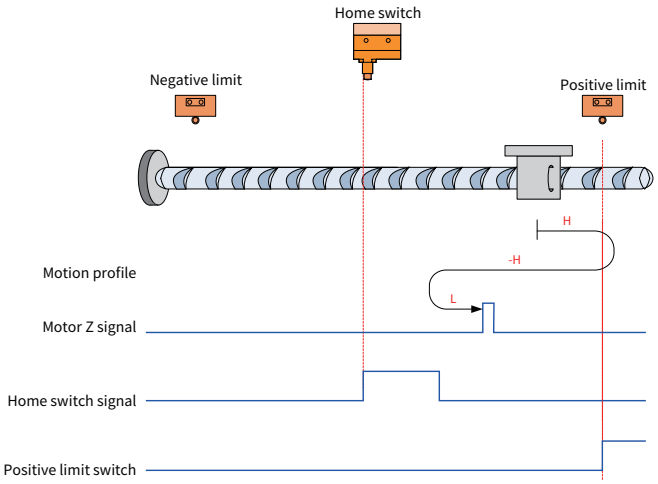
Deceleration point: home switch (HW)

◆ Deceleration point signal inactive at start of homing, not reaching positive limit switch



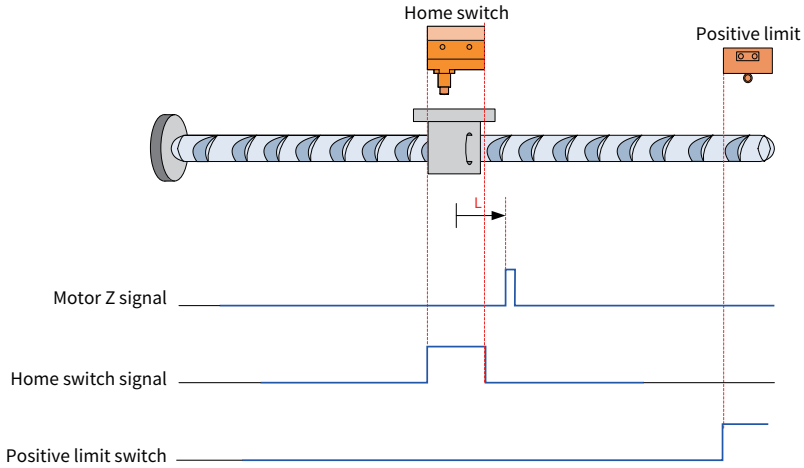
The HW signal is inactive initially, and the motor starts homing in positive direction at the high velocity. If the motor does not reach the limit switch, it decelerates and continues to run in positive direction at the low velocity after reaching the rising edge of the HW signal. After reaching the falling edge of the HW signal, the motor continues to run in positive direction at the low velocity, and stops at the first motor Z signal.

◆ Deceleration point signal inactive at start of homing, reaching the positive limit switch



The HW signal is inactive initially, and the motor starts homing in positive direction at the high velocity. If the motor reaches the limit switch, it automatically changes to run in negative direction at the high velocity. After reaching the rising edge of the HW signal, the motor decelerates and resumes running in positive direction at the low velocity. After reaching the falling edge of the HW signal, the motor stops at the first motor Z signal.

◆ Deceleration point signal active at start of homing



The HW signal is active initially, and the motor directly starts homing in positive direction at the low velocity. After reaching the falling edge of the HW signal, the motor stops at the first motor Z signal.

11) 6098h = 11&12&13&14

Similar to the profile when 6098 = 7 to 10, opposite in the initial running direction only

12) 6098h = 17 to 30

Same profiles as 6098 = 1 to 14, without the step of searching for motor Z signal. The motor stops immediately at receiving the following home signal.

Homing mode 6098	Home signal
17	N-OT falling edge
18	P-OT falling edge
19	HW falling edge
20	HW rising edge
21	HW falling edge

22	HW rising edge
23	HW falling edge
24	HW rising edge
25	HW rising edge
26	HW falling edge
27	HW falling edge
28	HW rising edge
29	HW rising edge
30	HW falling edge

13) 6098h = 31 to 32

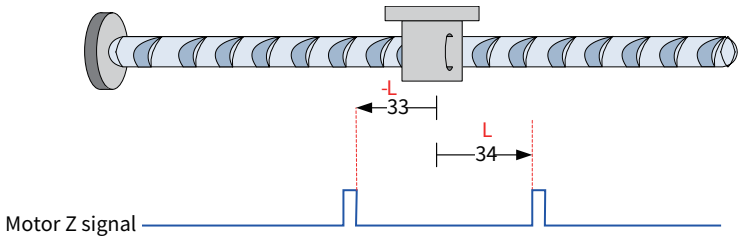
This mode is not defined in the standard 402 protocol. It can be used for expansion purpose.

14) 6098h = 33 and 34

Home: Z signal

Deceleration point: None

- Homing mode 33: The motor runs in negative direction at the low velocity, and stops at the first motor Z signal.
- Homing mode 34: The motor runs in positive direction at the low velocity, and stops at the first motor Z signal.



15) 6098h = 35

Homing mode 35: The current position is the home. After the homing signal is triggered (6040 control word: 0x0F→0x1F), the current position 6064 = 607C.

Appendix B: CiA402 Common Data Object Cheat Sheet Supported by IS620N

Index (HEX)	Sub-index (HEX)	Name	Access	Size	Unit	Setting Range	Default Value	PDO Mapping
603F	00	Error code	RO	UINT16	-			TPDO
This object gives the most recent error code or alarm code of the drive. The corresponding lowest 12 bits indicate the fault code. For the code definitions, see the IS620 guide. Use 200B:22 or 23 to view up to 10 latest fault log codes.								
6040	00	Control word	RW	UINT16	-	0 to 65535	0	RPDO
Status guidance after servo power-up, instruction control in each servo mode								
6041	00	Status word	RO	UINT16	-			TPDO
Indicate the servo drive running status.								
605A	00	Quick stop option code	RW	INT8		0 to 7	2	-
0 to 7 Select the drive quick stop mode								
605D	00	Halt option code	RW	INT8		1 to 3	1	-
Select the drive halt mode								
6060	00	Modes of operation	RW	INT8	-	0 to 10	0	RPDO
1 Profile Position (PP) mode 3 Profile Velocity (PV) mode 4 Profile Torque (PT) mode 6 Homing Mode (HM) 8 Cyclic Synchronous Position (CSP) mode 9 Cyclic Synchronous Velocity (CSV) mode 10 Cyclic Synchronous Torque (CST) mode								
6061	00	Servo operation mode display	RO	INT8	-			TPDO
Actual operation mode								
6062	00	Position reference	RO	INT32	Reference unit			TPDO
Position instruction value during each position loop period time, reference unit								
6063	00	Position feedback	RO	INT32	Encoder unit			TPDO
The current position of the motor as fed back from the motor encoder								
6064	00	Position feedback	RO	INT32	Reference unit			TPDO
Position feedback value after inverse gear ratio calculation 6063 = 6064 x Gear ratio								
6065	00	Following error window	RW	UINT32	Reference unit	0 to 232-1	3145728	RPDO
When the position deviation 60F4 is greater than ±6065, the drive reports an excessive position deviation (Er.B00) error. If bit 13 of 6041 is 1 in PP mode, this fault can be reset.								
6067	00	Position window	RW	UINT32	Reference unit	0 to 65535	7	RPDO
When the position deviation 60F4 is less than this value, and the time reaches 6068, the servo drive considers that the position is reached, and sets status word 6041 bit 10 = 1. When either condition is not met, the position window is invalid.								
6068	00	Position window time	RW	UINT16	ms	0 to 65535	0	RPDO
When the position deviation 60F4 is less than this value, and the time reaches 6068, the servo drive considers that the position is reached, and sets status word 6041 bit 10 = 1. When either condition is not met, the position window is invalid.								
606C	00	Actual velocity	RO	INT32	Reference unit/s			TPDO
The object indicates the position feedback per second (reference unit).								
606D	00	Velocity window	RW	UINT32	rpm	0 to 65535	10	RPDO
When the difference between the motor velocity feedback and the velocity instruction is within ±606D, and the time reaches 606E, the servo drive considers that the velocity is reached, and sets status word 6041 bit 10 = 1. When either condition is not met, the velocity window is invalid.								
606E	00	Velocity window time	RW	UINT16	ms	0 to 65535	0	RPDO
When the difference between the velocity feedback and the velocity instruction is within ±606D, and the time reaches 606E, the servo drive considers that the velocity is reached, and sets status word 6041 bit 10 = 1. When either condition is not met, the velocity window is invalid.								

Index (HEX)	Sub-index (HEX)	Name	Access	Size	Unit	Setting Range	Default Value	PDO Mapping
6071	00	Target torque	RW	INT16	0.1%	–5000 to 5000	0	RPDO
Target torque setting in torque mode								
6072	00	Max. torque	RW	UINT16	0.1%	0 to 5000	0	RPDO
Maximum torque limit								
6074	00	Torque reference	RO	INT16	0.1%	–5000 to 5000	0	TPDO
Torque output instruction after internal calculation of the drive								
6077	00	Actual torque	RO	INT16	0.1%	–5000 to 5000	0	TPDO
Feedback torque value acquired by the drive								
607A	00	Target position	RW	INT32	Reference unit	-231-(231-1)	0	RPDO
Servo target position in profile position mode and cyclic synchronous position mode								
607C	00	Home offset	RW	INT32	Reference unit	-231-(231-1)	0	RPDO
Position of the mechanical home offset from the mechanical zero								
607D	Software absolute position limit							
	00	Highest sub-index supported	RO	UINT8	-	2	2	-
	01	Min. position limit	RW	INT32	User position unit	-231-(231-1)	-231	RPDO
	02	Max. position limit	RW	INT32	User position unit	-231-(231-1)	231-1	RPDO
After homing is complete, set the minimum and maximum position limits for running by combining with 607C. Position commands exceeding the limit will stop when the limit is reached.								
607E	00	Polarity	RW	UINT8	-	0 to 255	0	RPDO
Bit 7 - Position reference polarity: 0: Keep original polarity; 1: Reverse polarity Bit 6 - Velocity reference polarity: 0: Keep original polarity; 1: Reverse polarity Bit 5 - Torque reference polarity: 0: Keep original polarity; 1: Reverse polarity								
607F	00	Maximum velocity	RW	UINT32	Reference unit/s	0 to 232-1	104857600	RPDO
Maximum velocity limit Setting: 607F = Maximum allowable motor velocity (rpm) x Encoder resolution/60								
6081	00	Profile velocity	RW	UINT32	User velocity unit	0 to 232-1	0	RPDO
Setting of the uniform motor running velocity for the displacement in profile position mode								
6083	00	Profile acceleration	RW	UINT32	Reference unit/s2	1 to 232-1	1747626667	RPDO
Acceleration in PP, CSV, or PV mode Default value: 1747626667; Reference unit: /s2, indicating that the motor accelerates from 0 rpm to 1000 rpm in 10 ms.								
6084	00	Profile deceleration	RW	UINT32	Reference unit/s2	1 to 232-1	1747626667	RPDO
Deceleration in PP, CSV, or PV mode Default value: 1747626667; Reference unit: /s2, indicating that the motor decelerates from 1000 rpm to 0 rpm in 10 ms.								
6085	00	Deceleration rate for quick stop	RW	UINT32	User acceleration unit	1 to 232-1	1747626667	RPDO
Acceleration of deceleration section if 605A = 2 when a quick stop command (bit 2 of 6040 = 0) is issued by the host controller. Default value: 1747626667; Reference unit: /s2, indicating that the motor accelerates from 0 rpm to 1000 rpm in 10 ms.								
6086	00	Motion profile type	RW	INT16	-	0	0	RPDO
Set the motor running curve in profile position mode. Currently, only linear motion is supported.								
6087	00	Torque slope	RW	UINT32	0.1%/s	0	0xFFFFFFFF	RPDO
Set the torque command increment per second in profile torque mode.								

Index (HEX)	Sub-index (HEX)	Name	Access	Size	Unit	Setting Range	Default Value	PDO Mapping
6091	Gear ratio							
	00	Highest sub-index supported	RO	UINT8		2	2	
	01	Motor revolutions	RW	UINT32	-	0 to 232-1	1	RPDO
	02	Shaft revolutions	RW	UINT32	-	1-232-1	1	RPDO
Establish the proportionality between the encoder unit and the reference unit.								
6098	00	Homing method	RW	INT8	-	0-35	0	RPDO
Support 35 homing methods specified by DS402 protocol.								
6099	01	High velocity value of searching for the deceleration point signal	RW	UINT32	Reference unit/s	0 to 232-1	1747626	RPDO
	02	Low speed during search for zero	RW	UINT32	Reference unit/s	0 to 232-1	174762	RPDO
609A	00	Homing acceleration	RW	UINT32	Reference unit/s2	1 to 232-1	1747	RPDO
Acceleration in variable velocity section in homing mode. Default value: 1747; Reference unit: /s2, indicating that the motor accelerates from 0 rpm to 1000 rpm in 10 ms.								
60B0h	00	Position offset	RW	INT32	Reference unit	-231-(231-1)	0	RPDO
60B1h	00	Velocity offset	RW	INT32	Reference unit/s	-231-(231-1)	0	RPDO
60B2h	00	Torque offset	RW	INT32	0.1%	–5000 to 5000	0	RPDO
60B8h	00	Touch probe function	RW	UINT16	-	0 to 65535	0	RPDO
60B9h	00	Probe status	RW	UINT16	-	0 to 65535	0	RPDO
60BAh	00	Touch probe 1 rising edge	RW	INT32	Reference unit	-231-(231-1)	0	RPDO
60BBh	00	Touch probe 1 falling edge	RW	INT32	Reference unit	-231-(231-1)	0	RPDO
60BCh	00	Touch probe 2 rising edge	RW	INT32	Reference unit	-231-(231-1)	0	RPDO
60BDh	00	Touch probe 2 falling edge	RW	INT32	Reference unit	-231-(231-1)	0	RPDO
60E0h	00	Positive torque limit	RW	UINT16	0.1%	0 to 5000	2000	RPDO
60E1h	00	Negative torque limit	RW	UINT16	0.1%	0 to 5000	2000	RPDO
60E3h	00	Supported homing method	RW	UINT16	-	-	-	-
60E6h	00	Position calculation method	RW	UINT16	-	0 to 1	0	-
60F4h	00	Position deviation	RO	INT32	Reference unit	-231-(231-1)	0	TPDO
Position deviation, reference unit								
60FC	00	Position reference	RO	INT32	Encoder unit	-231-(231-1)	0	TPDO
Position reference, encoder unit								
60FDh	00	DI status	RO	UINT32	-	0 to 232-1	0	RPDO
60FEh	00	DO status	RO	UINT32	-	0 to 232-1	0	RPDO
60FFh	00	Target velocity	RW	INT32	Reference unit/s	-231-(231-1)	0	RPDO
Velocity reference setting in Synchronous Cyclic Velocity mode								
6502	00	Supported drive modes	RO	UINT32			0000 03ADhex	TPDO
Display the modes supported by the drive.								

Index (HEX)	Sub-index (HEX)	Name	Access	Size	Unit	Setting Range	Default Value	PDO Mapping
6091	Gear ratio							
	00	Highest sub-index supported	RO	UINT8		2	2	
	01	Motor revolutions	RW	UINT32	-	0 to 232-1	1	RPDO
	02	Shaft revolutions	RW	UINT32	-	1-232-1	1	RPDO
Establish the proportionality between the encoder unit and the reference unit.								
6098	00	Homing method	RW	INT8	-	0-35	0	RPDO
Support 35 homing methods specified by DS402 protocol.								
6099	01	High velocity value of searching for the deceleration point signal	RW	UINT32	Reference unit/s	0 to 232-1	1747626	RPDO
	02	Low speed during search for zero	RW	UINT32	Reference unit/s	0 to 232-1	174762	RPDO
609A	00	Homing acceleration	RW	UINT32	Reference unit/s2	1 to 232-1	1747	RPDO
Acceleration in variable velocity section in homing mode. Default value: 1747; Reference unit: /s2, indicating that the motor accelerates from 0 rpm to 1000 rpm in 10 ms.								
60B0h	00	Position offset	RW	INT32	Reference unit	-231-(231-1)	0	RPDO
60B1h	00	Velocity offset	RW	INT32	Reference unit/s	-231-(231-1)	0	RPDO
60B2h	00	Torque offset	RW	INT32	0.1%	–5000 to 5000	0	RPDO
60B8h	00	Touch probe function	RW	UINT16	-	0 to 65535	0	RPDO
60B9h	00	Probe status	RW	UINT16	-	0 to 65535	0	RPDO
60BAh	00	Touch probe 1 rising edge	RW	INT32	Reference unit	-231-(231-1)	0	RPDO
60BBh	00	Touch probe 1 falling edge	RW	INT32	Reference unit	-231-(231-1)	0	RPDO
60BCh	00	Touch probe 2 rising edge	RW	INT32	Reference unit	-231-(231-1)	0	RPDO
60BDh	00	Touch probe 2 falling edge	RW	INT32	Reference unit	-231-(231-1)	0	RPDO
60E0h	00	Positive torque limit	RW	UINT16	0.1%	0 to 5000	2000	RPDO
60E1h	00	Negative torque limit	RW	UINT16	0.1%	0 to 5000	2000	RPDO
60E3h	00	Supported homing method	RW	UINT16	-	-	-	-
60E6h	00	Position calculation method	RW	UINT16	-	0 to 1	0	-
60F4h	00	Position deviation	RO	INT32	Reference unit	-231-(231-1)	0	TPDO
Position deviation, reference unit								
60FC	00	Position reference	RO	INT32	Encoder unit	-231-(231-1)	0	TPDO
Position reference, encoder unit								
60FDh	00	DI status	RO	UINT32	-	0 to 232-1	0	RPDO
60FEh	00	DO status	RO	UINT32	-	0 to 232-1	0	RPDO
60FFh	00	Target velocity	RW	INT32	Reference unit/s	-231-(231-1)	0	RPDO
Velocity reference setting in Synchronous Cyclic Velocity mode								
6502	00	Supported drive modes	RO	UINT32			0000 03ADhex	TPDO
Display the modes supported by the drive.								

Appendix C Error Codes

SMC_ERROR: Records the error ID returned by the motion control function block.

Error code	Generation Source	Variable	Error Cause
0	All	SMC_NO_ERROR	No error
1	Drive interface	SMC_DI_GENERAL_COMMUNICATION_ERROR	Communication error (such as a broken Sercos ring)
2	Drive interface	SMC_DI_AXIS_ERROR	Axis error
3	Drive interface	SMC_DI_FIELDBUS_LOST_SYNCRONICITY	Loss of bus DC synchronization
10	Drive interface	SMC_DI_SWLIMITS_EXCEEDED	Software limit switch is activated After bSWLimitEnable is enabled, the current position of the axis is not within the range of fSWLimitPositive and fSWLimitNegative range
11	Drive interface	SMC_DI_HWLIMITS_EXCEEDED	Hardware limit switch is activated
13	Drive interface	SMC_DI_HALT_OR_QUICKSTOP_NOT_SUPPORTED	Drive status stopped or quick stop is not supported
14	Drive interface	SMC_DI_VOLTAGE_DISABLED	Drive is not enabled
15	Drive interface	SMC_DI_IRREGULAR_ACTPOSITION	Position format currently given by drive is incorrect. Check communication
16	Drive interface	SMC_DI_POSITIONLAGERROR	Position lag error. Set and current positions exceed limit
17	Drive interface	SMC_DI_HOMING_ERROR	Drive homing error
20	All modules created by motion control	SMC_REGULATOR_OR_START_NOT_SET	Controller is not enabled or brake is closed
21	Axis in wrong control mode	SMC_WRONG_CONTROLLER_MODE	Axis is not in correct control mode
30	Drive interface	SMC_FB_WASNT_CALLED_DURING_MOTION	Modules created by motion control are not called until the end of motion
31	All modules	SMC_AXIS_IS_NO_AXIS_REF	The given AXIS_REF variable is not of the AXIS_REF type
32	Axis is in incorrect control mode	SMC_AXIS_REF_CHANGED_DURING_OPERATION	Return value of the AXIS_REF variable is processed before the module is activated
33	Drive interface	SMC_FB_ACTIVE_AXIS_DIABLED	Axis is not activated during movement (MC_Power. bRegulatorOn)
34	All modules created by motion control	SMC_AXIS_NOT_READY_FOR_MOTION	Axis cannot process the current instruction in the current status
35	All modules created by motion control	SMC_AXIS_ERROR_DURING_MOTION	Axis error during motion
40	Virtual drive	SMC_VD_MAX_VELOCITY_EXCEEDED	Maximum velocity (fMaxVelocity) reached
41	Virtual drive	SMC_VD_MAX_ACCELERATION_EXCEEDED	Maximum acceleration (fMaxAcceleration) reached
42	Virtual drive	SMC_VD_MAX_DECELERATION_EXCEEDED	Maximum deceleration (fMaxDeceleration) reached
50	SMC_Homing	SMC_3SH_INVALID_VELACC_VALUES	Invalid velocity or acceleration value
51	SMC_Homing	SMC_3SH_MODE_NEEDS_HWLIMIT	End limit switch required for module (safety use)
70	SMC_SetControllerMode	SMC_SCM_NOT_SUPPORTED	Mode is not supported
71	SMC_SetControllerMode	SMC_SCM_AXIS_IN_WRONG_STATE	Control mode used in current mode is not supported
75	SMC_SetTorque	SMC_ST_WRONG_CONTROLLER_MODE	Axis is in an incorrect control mode; this function block needs to be enabled in torque mode
80	SMC_ResetAxisGroup	SMC_RAG_ERROR_DURING_STARTUP	Error during axis group startup
90	SMC_ChangeGearingRatio	SMC_CGR_ZERO_VALUES	Incorrect variable
91	SMC_ChangeGearingRatio	SMC_CGR_DRIVE_POWERED	Transmission ratio cannot be changed in drive control mode
92	SMC_ChangeGearingRatio	SMC_CGR_INVALID_POSPERIOD	Incorrect position period (<=0)
110	MC_Power	SMC_P_FTASKCYCLE_EMPTY	Axis contains no information during the scan period (fTaskCycle = 0)
120	MC_Reset	SMC_R_NO_ERROR_TO_RESET	Axis has no error reset
121	MC_Reset	SMC_R_DRIVE_DOESNT_ANSWER	Axis did not perform an error reset
122	MC_Reset	SMC_R_ERROR_NOT_RESETTABLE	Error cannot be reset
123	MC_Reset	SMC_R_DRIVE_DOESNT_ANSWER_IN_TIME	No response to communication with the axis
130	MC_ReadParameter, MC_ReadBoolParameter	SMC_RP_PARAM_UNKNOWN	Parameter number position

Error code	Generation Source	Variable	Error Cause
131	MC_ReadParameter, MC_ ReadBoolParameter	SMC_RP_REQUESTING_ERROR	An error occurred during the parameter transfer to drive. See error in function block example ReadDriveParameter (SM_ DriveBasic.lib)
140	MC_WriteParameter, MC_ WriteBoolParameter	SMC_WP_PARAM_INVALID	Parameter number position or write operation is not allowed
141	MC_WriteParameter, MC_ WriteBoolParameter	SMC_WP_SENDING_ERROR	See error in module example WriteDriveParameter (Drive_ Basic.lib)
170	MC_Home	SMC_H_AXIS_WASNT_STANDSTILL	Axis not in standard state
171	MC_Home	SMC_H_AXIS_DIDNT_START_HOMING	An error occurred during homing execution
172	MC_Home	SMC_H_AXIS_DIDNT_ANSWER	Communication error
173	MC_Home	SMC_H_ERROR_WHEN_STOPPING	Homing stopped due to an error. Check whether deceleration is set.
180	MC_Stop	SMC_MS_UNKNOWN_STOPPING_ERROR	An unknown error occurred during stopping
181	MC_Stop	SMC_MS_INVALID_ACCDEC_VALUES	Invalid velocity or acceleration values
182	MC_Stop	SMC_MS_DIRECTION_NOT_APPLICABLE	Direction = shortest unavailable
183	MC_Stop	SMC_MS_AXIS_IN_ERRORSTOP	Axis in stopping error status. Stopping cannot be processed.
184	MC_Stop	SMC_BLOCKING_MC_STOP_WASNT_CALLED	An instance of MC_Stop, which locks the axis (Execute = TRUE), cannot be called. Call MC_Stop (Execute = FALSE).
201	MC_MoveAbsolute	SMC_MA_INVALID_VELACC_VALUES	Invalid velocity or acceleration values
202	MC_MoveAbsolute	SMC_MA_INVALID_DIRECTION	Direction error
226	MC_MoveRelative	SMC_MR_INVALID_VELACC_VALUES	Invalid velocity or acceleration values
227	MC_MoveRelative	SMC_MR_INVALID_DIRECTION	Direction error
251	MC_MoveAdditive	SMC_MAD_INVALID_VELACC_VALUES	Invalid velocity or acceleration values
252	MC_MoveAdditive	SMC_MAD_INVALID_DIRECTION	Direction error
276	MC_MoveSuperImposed	SMC_MSI_INVALID_VELACC_VALUES	Invalid velocity or acceleration values
277	MC_MoveSuperImposed	SMC_MSI_INVALID_DIRECTION	Direction error
301	MC_MoveVelocity	SMC_MV_INVALID_ACCDEC_VALUES	Invalid velocity or acceleration values
302	MC_MoveVelocity	SMC_MV_DIRECTION_NOT_APPLICABLE	Direction=shortest/fastest not supported
325	MC_PositionProfile	SMC_PP_ARRAYSIZE	Wrong alignment size
326	MC_PositionProfile	SMC_PP_STEP0MS	Step time = t#0s
350	MC_VelocityProfile	SMC_VP_ARRAYSIZE	Wrong alignment size
351	MC_VelocityProfile	SMC_VP_STEP0MS	Step time = t#0s
375	MC_AccelerationProfile	SMC_AP_ARRAYSIZE	Wrong alignment size
376	MC_AccelerationProfile	SMC_AP_STEP0MS	Step time = t#0s
400	MC_TouchProbe	SMC_TP_TRIGGEROCCUPIED	Trigger condition has been activated
401	MC_TouchProbe	SMC_TP_COULDNT_SET_WINDOW	Window function is not supported by the drive interface
402	MC_TouchProbe	SMC_TP_COMM_ERROR	Communication error
410	MC_AbortTrigger	SMC_AT_TRIGGERNOTOCCUPIED	Trigger condition has been terminated
426	SMC_ MoveContinuousRelative	SMC_MCR_INVALID_VELACC_VALUES	Invalid velocity or acceleration values
427	SMC_ MoveContinuousRelative	SMC_MCR_INVALID_DIRECTION	Direction error
451	SMC_ MoveContinuousAbsolute	SMC_MCA_INVALID_VELACC_VALUES	Invalid velocity or acceleration values
452	SMC_ MoveContinuousAbsolute	SMC_MCA_INVALID_DIRECTION	Direction error
453	SMC_ MoveContinuousAbsolute	SMC_MCA_DIRECTION_NOT_APPLICABLE	Direction= fastest unavailable
600	SMC_CamRegister	SMC_CR_NO_TAPPETS_IN_CAM	Cam does not contain any tappet
601	SMC_CamRegister	SMC_CR_TOO_MANY_TAPPETS	Tappet group ID reaches MAX_NUM_TAPPETS
602	SMC_CamRegister	SMC_CR_MORE_THAN_32_ACCESSES	More than 32 interfaces in a CAM_REF
625	MC_CamIN	SMC_CI_NO_CAM_SELECTED	No cam is selected
626	MC_CamIN	SMC_CI_MASTER_OUT_OF_SCALE	Master axis out of range
627	MC_CamIN	SMC_CI_RAMPIN_NEEDS_VELACC_VALUES	Velocity and acceleration must be precisely specified for ramp_in function block
628	MC_CamIN	SMC_CI_SCALING_INCORRECT	Incorrect scale variables fEditor/TableMasterMin/Max

Error code	Generation Source	Variable	Error Cause
640	SMC_CAMBounds, SMC_CamBounds_Pos	SMC_CB_NOT_IMPLEMENTED	Function block given in cam format is not supported
675	MC_GearIn	SMC_GI_RATIO_DENOM	RatioDenominator = 0
676	MC_GearIn	SMC_GI_INVALID_ACC	Invalid acceleration
677	MC_GearIn	SMC_GI_INVALID_DEC	Invalid acceleration
725	MC_Phase	SMC_PH_INVALID_VELACCDEC	Invalid velocity/acceleration/deceleration
726	MC_Phase	SMC_PH_ROTARYAXIS_PERIOD0	Rotary axis fPositionPeriod = 0
750	All modules using MC_CAM_REF as input	SMC_NO_CAM_REF_TYPE	The given cam is not of type MC_CAM_REF
751	MC_CamTableSelect	SMC_CAM_TABLE_DOES_NOT_COVER_MASTER_SCALE	If the data retrieved from CamTable is not the master axis area (xStart and xEnd) obtained by data transformation
775	MC_GearInPos	SMC_GIP_MASTER_DIRECTION_CHANGE	The master axis changes its rotational direction during slave coupling.
800	SMC_BacklashCompensation	SMC_BC_BL_TOO_BIG	Excessive gear return ratio (fBacklash) (> position period/2)
1000	Function block requiring CNC license	SMC_NO_LICENSE	Target has no CNC license
1001	SMC_Interpolator	SMC_INT_VEL_ZERO	Path cannot be processed because velocity = 0
1002	SMC_Interpolator	SMC_INT_NO_STOP_AT_END	Previous path object Vel_End > 0
1003	SMC_Interpolator	SMC_INT_DATA_UNDERRUN	Warning: GEOINFO list is processed in DataIn, but the list is not set at the end. Reason: Forgetting to set EndOfList in DataIn or SMC_Interpolator is faster than path compilation module
1004	SMC_Interpolator	SMC_INT_VEL_NONZERO_AT_STOP	Stop velocity > 0
1005	SMC_Interpolator	SMC_INT_TOO_MANY_RECURSIONS	Excessive use of SMC_Interpolator, SoftMotion call error
1006	SMC_Interpolator	SMC_INT_NO_CHECKVELOCITIES	Input-OutQueue DataIn is not used as a final processing module for SMC_CHeckVelocities
1007	SMC_Interpolator	SMC_INT_PATH_EXCEEDED	Internal/Numeric error
1008	SMC_Interpolator	SMC_INT_VEL_ACC_DEC_ZERO	Velocity, acceleration or deceleration is empty or too low
1009	SMC_Interpolator	SMC_INT_DWIPOTIME_ZERO	FB call dwIpoTime = 0
1050	SMC_Interpolator2Dir	SMC_INT2DIR_BUFFER_TOO_SMALL	Data buffer too small
1051	SMC_Interpolator2Dir	SMC_INT2DIR_PATH_FITS_NOT_IN_QUEUE	Path is not fully contained in the queue
1100	SMC_CheckVelocities	SMC_CV_ACC_DEC_VEL_NONPOSITIVE	Velocity, deceleration or acceleration value is not in positive direction
1120	SMC_Controlaxisbypos	SMC_CA_INVALID_ACCDEC_VALUES	Variables fGapVelocity/fGapAcceleration/ fGapDeceleration are not positive values
1200	SMC_NCDecoder	SMC_DEC_ACC_TOO_LITTLE	Acceleration value is not allowed
1201	SMC_NCDecoder	SMC_DEC_RET_TOO_LITTLE	Deceleration value is not allowed
1202	SMC_NCDecoder	SMC_DEC_OUTQUEUE_RAN_EMPTY	Data below Queue is read and is empty
1203	SMC_NCDecoder	SMC_DEC_JUMP_TO_UNKNOWN_LINE	The line number jumped cannot be executed because of an unknown line number
1204	SMC_NCDecoder	SMC_DEC_INVALID_SYNTAX	Syntax error
1205	SMC_NCDecoder	SMC_DEC_3DMODE_OBJECT_NOT_SUPPORTED	These objects do not support 3D mode
1300	SMC_GCodeViewer	SMC_GCV_BUFFER_TOO_SMALL	Buffer too small
1301	SMC_GCodeViewer	SMC_GCV_BUFFER_WRONG_TYPE	Buffer element type error
1302	SMC_GCodeViewer	SMC_GCV_UNKNOWN_IPO_LINE	Current interpolation line cannot be found
1500	All function blocks using SMC_CNC_REF	SMC_NO_CNC_REF_TYPE	The given CNC program is not of the SMC_CNC_REF type
1501	All function blocks using SMC_OUTQUEUE	SMC_NO_OUTQUEUE_TYPE	The given OutQueue is not of the SMC_OUTQUEUE type
1600	CNC function block	SMC_3D_MODE_NOT_SUPPORTED	This function block is only available in 2D path
2000	SMC_ReadNCFile	SMC_RNCF_FILE_DOESNT_EXIST	File does not exist
2001	SMC_ReadNCFile	SMC_RNCF_NO_BUFFER	No buffer allocation
2002	SMC_ReadNCFile	SMC_RNCF_BUFFER_TOO_SMALL	Buffer too small
2003	SMC_ReadNCFile	SMC_RNCF_DATA_UNDERRUN	Low buffer data in buffer area is read and is empty
2004	SMC_ReadNCFile	SMC_RNCF_VAR_COULDNT_BE_REPLACED	Placeholder variable cannot be replaced
2005	SMC_ReadNCFile	SMC_RNCF_NOT_VARLIST	Input pvl cannot point to SMC_VARLIST object

Error code	Generation Source	Variable	Error Cause
2050	SMC_ReadNCQueue	SMC_RNCQ_FILE_DOESNT_EXIST	File cannot be opened
2051	SMC_ReadNCQueue	SMC_RNCQ_NO_BUFFER	No buffer definition
2052	SMC_ReadNCQueue	SMC_RNCQ_BUFFER_TOO_SMALL	Buffer too small
2053	SMC_ReadNCQueue	SMC_RNCQ_UNEXPECTED_EOF	Unknown end of file
2100	SMC_AxisDiagnosticLog	SMC_ADL_FILE_CANNOT_BE_OPENED	File cannot be opened
2101	SMC_AxisDiagnosticLog	SMC_ADL_BUFFER_OVERRUN	Buffering out of range; WriteToFile must be called more often
2200	SMC_ReadCAM	SMC_RCAM_FILE_DOESNT_EXIST	File cannot be opened
2201	SMC_ReadCAM	SMC_RCAM_TOO_MUCH_DATA	Too much data saved to cam
2202	SMC_ReadCAM	SMC_RCAM_WRONG_COMPILE_TYPE	Wrong compile mode
2203	SMC_ReadCAM	SMC_RCAM_WRONG_VERSION	File version error
2204	SMC_ReadCAM	SMC_RCAM_UNEXPECTED_EOF	Unknown end of file
3001	SMC_ WriteDriveParamsToFile	SMC_WDPF_CHANNEL_OCCUPIED	SMC_WDPF_TIMEOUT_PREPARING_LIST
3002	SMC_ WriteDriveParamsToFile	SMC_WDPF_CANNOT_CREATE_FILE	File cannot be created
3003	SMC_ WriteDriveParamsToFile	SMC_WDPF_ERROR_WHEN_READING_ PARAMS	Error in reading file parameters
3004	SMC_ WriteDriveParamsToFile	SMC_WDPF_TIMEOUT_PREPARING_LIST	Time error in preparing parameter list
5000	SMC_Encoder	SMC_ENC_DENOM_ZERO	Conversion factor (dwRatioTechUnitsDenom) of the decoder reference is 0.
5001	SMC_Encoder	SMC_ENC_AXISUSEDBYOTHERFB	Other modules are handling the decoder axis.
5002	Drive interface	SMC_ENC_FILTER_DEPTH_INVALID	Invalid filter

Copyright © Shenzhen Inovance Technology Co., Ltd.

19012378A00

Shenzhen Inovance Technology Co., Ltd.

www.inovance.com

Add.: Inovance Headquarters Tower, High-tech Industrial Park,
Guanlan Street, Longhua New District, Shenzhen
Tel: (0755) 2979 9595 Fax: (0755) 2961 9897

Suzhou Inovance Technology Co., Ltd.

www.inovance.com

Add.: No. 16 Youxiang Road, Yuexi Town,
Wuzhong District, Suzhou 215104, P.R. China
Tel: (0512) 6637 6666 Fax: (0512) 6285 6720