OMRON

Machine Automation Controller

# NJ-series

# Motion Control Instructions Reference Manual

**NJ501-1300**
**NJ501-1400**
**NJ501-1500**

sysmac
always in control

W508-E1-01

# Introduction

Thank you for purchasing an NJ-series CPU Unit.
This manual describes the motion control instructions. Please be sure you sufficiently understand the operations and handling procedures, and use the Motion Control Function Module (abbreviated as “MC Function Module”) correctly.
Use this manual together with the user’s manuals for the NJ-series CPU Unit.
When you have finished reading this manual, keep it in a safe location where it will be readily available for future use.

## Intended Audience

This manual is intended for the following personnel, who must also have knowledge of electrical systems (an electrical engineer or the equivalent).

- Personnel in charge of introducing FA systems.
- Personnel in charge of designing FA systems.
- Personnel in charge of installing and maintaining FA systems.
- Personnel in charge of managing FA systems and facilities.

For programming, this manual is intended for personnel who understand the programming language specifications in international standard IEC 61131-3 or Japanese standard JIS B3503.

## Applicable Products

This manual covers the following products.

- NJ-series CPU Units
  - NJ501-1300
  - NJ501-1400
  - NJ501-1500

# Relevant Manuals

There are three manuals that provide basic information on the NJ-series CPU Units: the *NJ-series CPU Unit Hardware User's Manual*, the *NJ-series CPU Unit Software User's Manual* (this manual), and the *NJ-series Instructions Reference Manual*.
Most operations are performed from the Sysmac Studio Automation Software. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for information on the Sysmac Studio.
Other manuals are necessary for specific system configurations and applications.
Read all of the manuals that are relevant to your system configuration and application to make the most of the NJ-series CPU Unit.

| | NJ-series User's Manuals | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Basic information | | | | | | | | |
| | NJ-series CPU Unit Hardware User's Manual | NJ-series CPU Unit Software User's Manual | NJ-series Instructions Reference Manual | NJ-series CPU Unit Motion Control User's Manual | NJ-series CPU Unit Built-in EtherCAT Port User's Manual | NJ-series Motion Control Instructions Reference Manual | NJ-series CPU Unit Built-in EtherNet/IP Port User's Manual | NJ-series Troubleshooting Manual | CJ-series Special Unit Operation Manuals for NJ-series CPU Unit |
| Introduction to NJ-series Controllers | ● | | | | | | | | |
| Setting devices and hardware | | | | | | | | | |
| Using motion control | | | | ● | | | | | |
| Using EtherCAT | ● | | | | ● | | | | |
| Using EtherNet/IP | | | | | | | ● | | |
| Using CJ-series Units | | | | | | | | | ● |
| Software settings | | ● | | | | | | | |
| Using motion control | | | | ● | | | | | |
| Using EtherCAT | | | | | ● | | | | |
| Using EtherNet/IP | | | | | | | ● | | |
| Programming | | ● | ● | | | | | | |
| Using motion control | | | | ● | | ● | | | |
| Using EtherCAT | | | | | ● | | | | |
| Using CJ-series Units | | | | | | | | | ● |
| Programming error processing | | | | | | | | ● | |
| Testing operation and debugging | | ● | | | | | | | |
| Using motion control | | | | ● | | | | | |
| Using EtherCAT | | | | | ● | | | | |
| Using EtherNet/IP | | | | | | | ● | | |
| Troubleshooting and managing errors in an NJ-series Controller | △ Use the relevant manuals for references according to any error that occurs. | △ | | △ | △ | | △ | ● | |
| Maintenance | | | | | | | | | |
| Using EtherCAT | ● | | | | ● | | | | |
| Using EtherNet/IP | | | | | | | ● | | |
| Using CJ-series Units | | | | | | | | | ● |

# Manual Configuration

## NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)

| Section | Description |
|---|---|
| **Section 1**<br>**Introduction** | This section provides an introduction to the NJ-series Controllers and their features, and gives the NJ-series Controller specifications. |
| **Section 2**<br>**System Configuration** | This section describes the system configuration used for NJ-series Controllers. |
| **Section 3**<br>**Configuration Units** | This section describes the parts and functions of the configuration devices in the NJ-series Controller configuration, including the CPU Unit and Configuration Units. |
| **Section 4**<br>**Installation and Wiring** | This section describes where and how to install the CPU Unit and Configuration Units and how to wire them. |
| **Section 5**<br>**Troubleshooting** | This section describes the event codes, error confirmation methods, and corrections for errors that can occur. |
| **Section 6**<br>**Inspection and Maintenance** | This section describes the contents of periodic inspections, the service life of the Battery and Power Supply Units, and replacement methods for the Battery and Power Supply Units. |
| **Appendices** | The appendices provide the specifications of the Basic I/O Units, Unit dimensions, load short-circuit protection detection, line disconnection detection, and measures for EMC Directives. |

## NJ-series CPU Unit Software User's Manual (Cat. No. W501)

| Section | Description |
|---|---|
| **Section 1**<br>**Introduction** | This section provides an introduction to the NJ-series Controllers and their features, and gives the NJ-series Controller specifications. |
| **Section 2**<br>**CPU Unit Operation** | This section describes the variables and control systems of the CPU Unit and CPU Unit status. |
| **Section 3**<br>**I/O Ports, Slave Configuration, and Unit Configuration** | This section describes how to use I/O ports, how to create the slave configuration and unit configuration and how to assign functions. |
| **Section 4**<br>**Controller Setup** | This section describes the initial settings of the function modules. |
| **Section 5**<br>**Designing Tasks** | This section describes the task system and types of tasks. |
| **Section 6**<br>**Programming** | This section describes programming, including the programming languages and the variables and instructions that are used in programming. |
| **Section 7**<br>**Simulation, Transferring Projects to the Physical CPU Unit, and Operation** | This section describes simulation of Controller operation and how to use the results of simulation. |
| **Section 8**<br>**CPU Unit Status** | This section describes CPU Unit status. |
| **Section 9**<br>**CPU Unit Functions** | This section describes the functionality provided by the CPU Unit. |
| **Section 10**<br>**Communications Setup** | This section describes how to go online with the CPU Unit and how to connect to other devices. |
| **Section 11**<br>**Example of Actual Application Procedures** | This section describes the procedures that are used to actually operate an NJ-series Controller. |
| **Section 12**<br>**Troubleshooting** | This section describes the event codes, error confirmation methods, and corrections for errors that can occur. |
| **Appendices** | The appendices provide the CPU Unit specifications, task execution times, system-defined variable lists, data attribute lists, CJ-series Unit memory information, CJ-series Unit memory allocation methods, and data type conversion information. |

## NJ-series CPU Unit Motion Control User's Manual (Cat. No. W507)

| Section | Description |
|---|---|
| **Section 1 Introduction to the Motion Control Function Module** | This section describes the features, system configuration, and application flow for the Motion Control Function Module. |
| **Section 2 Motion Control Configuration and Principles** | This section outlines the internal structure of the CPU Unit and describes the configuration and principles of the MC Function Module. |
| **Section 3 Configuring Axes and Axes Groups** | This section describes the concept of axes and axes groups, the settings for axes that are required for the MC Test Run operations to function on the Sysmac Studio, and the instructions for creating and configuring axes and axes groups using the Sysmac Studio. |
| **Section 4 Checking Wiring from the Sysmac Studio** | This section describes the MC Test Run operations of the Sysmac Studio. You can use the MC Test Run operations to monitor sensor signals, check Servomotor wiring, and more, all without any programming. |
| **Section 5 Motion Control Parameters** | This section provides information on the axis parameters and axes group parameters that are used for motion control. |
| **Section 6 Motion Control Programming** | This section provides the specifications of a motion control program and the operating procedures that are required up through actual program development. |
| **Section 7 Manual Operation** | This section describes manual operation when the MC Function Module is used together with an OMRON G5-series Servo Drive. |
| **Section 8 Homing** | This section describes homing. |
| **Section 9 Motion Control Functions** | This section describes the motion control functions that are used when connected to OMRON G5-series Servo Drives with built-in EtherCAT communications. |
| **Section 10 Sample Programming** | This section describes basic application methods for homing, error monitoring, and other functions, and provides programming samples for absolute positioning, cam operation, and other axis operations. |
| **Section 11 Troubleshooting** | This section describes the items to check when problems occur in the MC Function Module. It includes error diagnosis and countermeasures for error indications, and error diagnosis and countermeasures for operating conditions. |
| **Appendices** | The appendices describe settings and connection methods for OMRON G5-series Servo Drive objects. |

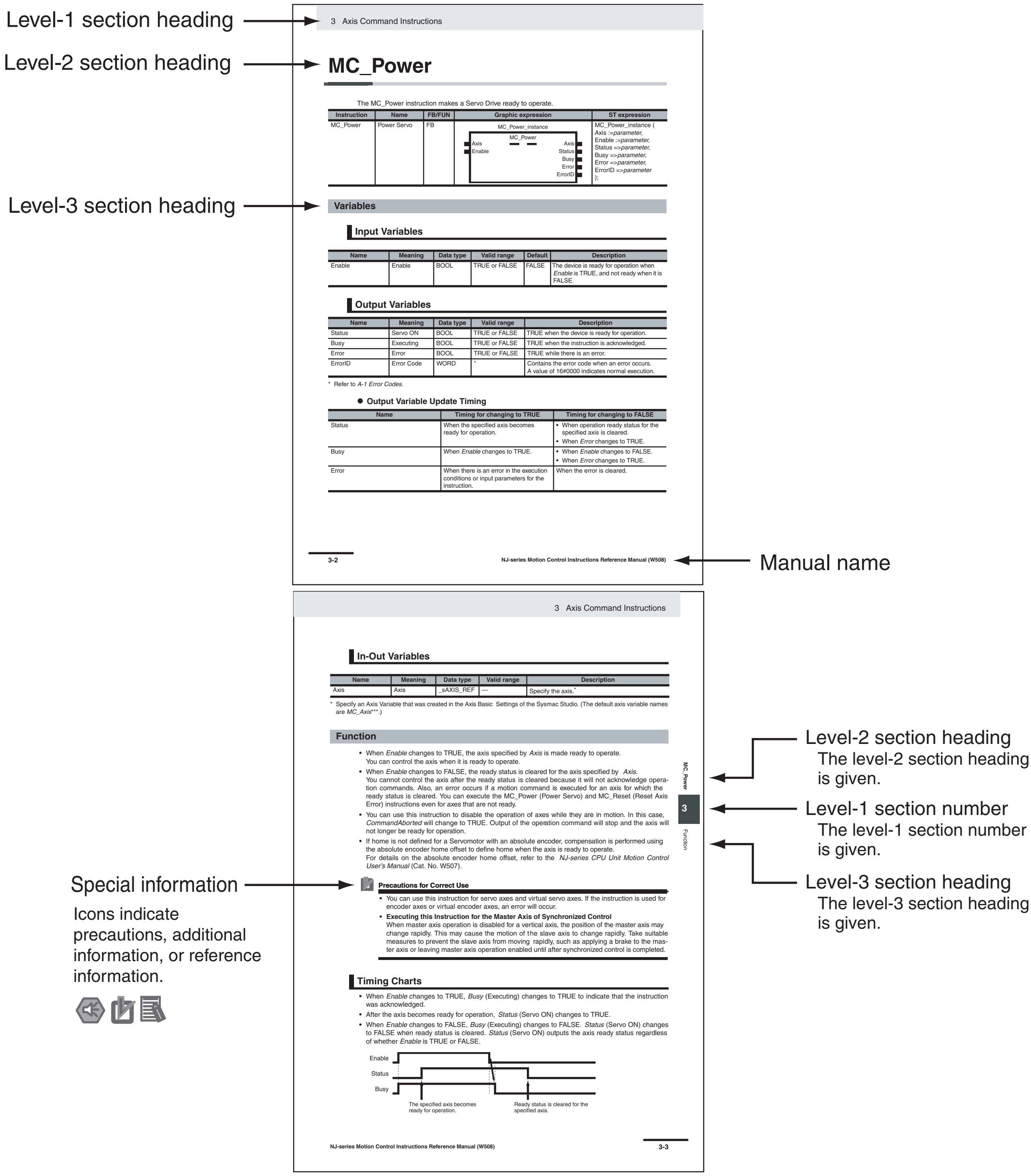
## NJ-series Motion Control Instructions Reference Manual (Cat. No. W508) (This Manual)

| Section | Description |
|---|---|
| **Section 1 Introduction to Motion Control Instructions** | This section gives an introduction to motion control instructions supported by NJ-series CPU Units. |
| **Section 2 Variables and Instructions** | This section describes the variables and instructions for the Motion Control Function Module. |
| **Section 3 Axis Command Instructions** | This section describes the instructions that are used to perform single-axis control for the MC Function Module. |
| **Section 4 Axes Group Instructions** | This section describes the instructions to perform multi-axes coordinated control for the MC Function Module. |
| **Section 5 Common Command Instructions** | This section describes the instructions that are used for both axes and axes groups. |
| **Appendices** | The appendices describe the error codes that are generated by the instructions. |

# Manual Structure

## Page Structure

The following page structure is used in this manual.




**Note:** This page is for illustration only. It does not represent a specific page in this manual.

## Special Information

Special information in this manual is classified as follows:

**Precautions for Safe Use**

Precautions on what to do and what not to do to ensure safe usage of the product.

**Precautions for Correct Use**

Precautions on what to do and what not to do to ensure proper operation and performance.

**Additional Information**

Additional information to read as required.
This information is provided to increase understanding or make operation easier.

**Note** References are provided to more detailed or related information.

# Sections in this Manual

# CONTENTS

# Section 3 Axis Command Instructions

## Section 4 Axes Group Instructions

## Section 5 Common Command Instructions



## Appendices



## Index

# Read and Understand this Manual

Please read and understand this manual before using the product. Please consult your OMRON representative if you have any questions or comments.

## Warranty and Limitations of Liability

### WARRANTY

OMRON's exclusive warranty is that the products are free from defects in materials and workmanship for a period of one year (or other period if specified) from date of sale by OMRON.

OMRON MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, REGARDING NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR PARTICULAR PURPOSE OF THE PRODUCTS. ANY BUYER OR USER ACKNOWLEDGES THAT THE BUYER OR USER ALONE HAS DETERMINED THAT THE PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE. OMRON DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED.

### LIMITATIONS OF LIABILITY

OMRON SHALL NOT BE RESPONSIBLE FOR SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR COMMERCIAL LOSS IN ANY WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED ON CONTRACT, WARRANTY, NEGLIGENCE, OR STRICT LIABILITY.

In no event shall the responsibility of OMRON for any act exceed the individual price of the product on which liability is asserted.

IN NO EVENT SHALL OMRON BE RESPONSIBLE FOR WARRANTY, REPAIR, OR OTHER CLAIMS REGARDING THE PRODUCTS UNLESS OMRON'S ANALYSIS CONFIRMS THAT THE PRODUCTS WERE PROPERLY HANDLED, STORED, INSTALLED, AND MAINTAINED AND NOT SUBJECT TO CONTAMINATION, ABUSE, MISUSE, OR INAPPROPRIATE MODIFICATION OR REPAIR.

# Application Considerations

## SUITABILITY FOR USE

OMRON shall not be responsible for conformity with any standards, codes, or regulations that apply to the combination of products in the customer's application or use of the products.

At the customer's request, OMRON will provide applicable third party certification documents identifying ratings and limitations of use that apply to the products. This information by itself is not sufficient for a complete determination of the suitability of the products in combination with the end product, machine, system, or other application or use.

The following are some examples of applications for which particular attention must be given. This is not intended to be an exhaustive list of all possible uses of the products, nor is it intended to imply that the uses listed may be suitable for the products:

- Outdoor use, uses involving potential chemical contamination or electrical interference, or conditions or uses not described in this manual.
- Nuclear energy control systems, combustion systems, railroad systems, aviation systems, medical equipment, amusement machines, vehicles, safety equipment, and installations subject to separate industry or government regulations.
- Systems, machines, and equipment that could present a risk to life or property.

Please know and observe all prohibitions of use applicable to the products.

NEVER USE THE PRODUCTS FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE OMRON PRODUCTS ARE PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

## PROGRAMMABLE PRODUCTS

OMRON shall not be responsible for the user's programming of a programmable product, or any consequence thereof.

# Disclaimers

## CHANGE IN SPECIFICATIONS

Product specifications and accessories may be changed at any time based on improvements and other reasons.

It is our practice to change model numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the products may be changed without any notice. When in doubt, special model numbers may be assigned to fix or establish key specifications for your application on your request. Please consult with your OMRON representative at any time to confirm actual specifications of purchased products.

## DIMENSIONS AND WEIGHTS

Dimensions and weights are nominal and are not to be used for manufacturing purposes, even when tolerances are shown.

## PERFORMANCE DATA

Performance data given in this manual is provided as a guide for the user in determining suitability and does not constitute a warranty. It may represent the result of OMRON's test conditions, and the users must correlate it to actual application requirements. Actual performance is subject to the OMRON Warranty and Limitations of Liability.

## ERRORS AND OMISSIONS

The information in this manual has been carefully checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical, or proofreading errors, or omissions.

# Safety Precautions

Refer to the following manuals for safety precautions.

- NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)
- NJ-series CPU Unit Software User's Manual (Cat. No. W501)

# Precautions for Safe Use

Refer to the following manuals for precautions for correct use.

- NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)
- NJ-series CPU Unit Software User's Manual (Cat. No. W501)

# Precautions for Correct Use

Refer to the following manuals for precautions for safe use.

- NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)
- NJ-series CPU Unit Software User's Manual (Cat. No. W501)

# Regulations and Standards

## Conformance to EC Directives

### Applicable Directives

- EMC Directives
- Low Voltage Directive

### Concepts

#### EMC Directive

OMRON devices that comply with EC Directives also conform to the related EMC standards so that they can be more easily built into other devices or the overall machine. The actual products have been checked for conformity to EMC standards.*
Whether the products conform to the standards in the system used by the customer, however, must be checked by the customer. EMC-related performance of the OMRON devices that comply with EC Directives will vary depending on the configuration, wiring, and other conditions of the equipment or control panel on which the OMRON devices are installed. The customer must, therefore, perform the final check to confirm that devices and the overall machine conform to EMC standards.

* Applicable EMC (Electromagnetic Compatibility) standards are as follows:
EMS (Electromagnetic Susceptibility): EN 61131-2 and EN 61000-6-2
EMI (Electromagnetic Interference): EN 61131-2 and EN 61000-6-4 (Radiated emission: 10-m regulations)

#### Low Voltage Directive

Always ensure that devices operating at voltages of 50 to 1,000 VAC and 75 to 1,500 VDC meet the required safety standards. The applicable directive is EN 61131-2.

#### Conformance to EC Directives

The NJ-series Controllers comply with EC Directives. To ensure that the machine or device in which the NJ-series Controller is used complies with EC Directives, the Controller must be installed as follows:

- The NJ-series Controller must be installed within a control panel.
- You must use reinforced insulation or double insulation for the DC power supplies connected to DC Power Supply Units and I/O Units.
- NJ-series Controllers that comply with EC Directives also conform to the Common Emission Standard (EN 61000-6-4). Radiated emission characteristics (10-m regulations) may vary depending on the configuration of the control panel used, other devices connected to the control panel, wiring, and other conditions.
  You must therefore confirm that the overall machine or equipment complies with EC Directives.

## Conformance to Shipbuilding Standards

The NJ-series Controllers comply with the following shipbuilding standards. Applicability to the shipbuilding standards is based on certain usage conditions. It may not be possible to use the product in some locations. Contact your OMRON representative before attempting to use a Controller on a ship.

### Usage Conditions for NK and LR Shipbuilding Standards

- The NJ-series Controller must be installed within a control panel.
- Gaps in the door to the control panel must be completely filled or covered with gaskets or other material.
- The following noise filter must be connected to the power supply line.

**Noise Filter**

| Manufacturer | Model |
|---|---|
| Cosel Co., Ltd. | TAH-06-683 |

## Trademarks

- Sysmac and SYSMAC are trademarks or registered trademarks of OMRON Corporation in Japan and other countries for OMRON factory automation products.
- Windows, Windows 98, Windows XP, Windows Vista, and Windows 7 are registered trademarks of Microsoft Corporation in the USA and other countries.
- EtherCAT® is a registered trademark of Beckhoff Automation GmbH for their patented technology.
- The SD logo is a trademark of SD-3C, LLC.

Other company names and product names in this document are the trademarks or registered trademarks of their respective companies.

## Software Licenses and Copyrights

This product incorporates certain third party software. The license and copyright information associated with this software is available at http://www.fa.omron.co.jp/nj_info_e/.

# Unit Versions

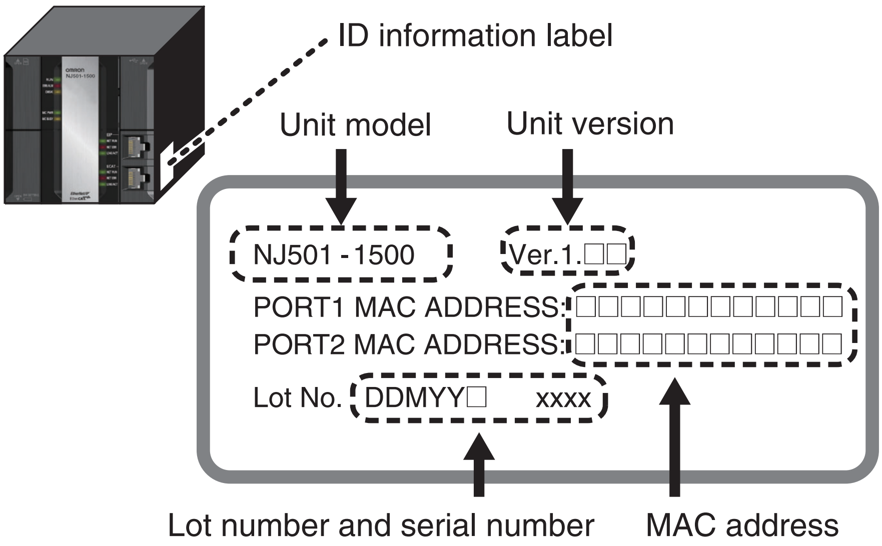
## Unit Versions

A “unit version” has been introduced to manage CPU Units in the NJ Series according to differences in functionality accompanying Unit upgrades.

### Notation of Unit Versions on Products

The unit version is given on the ID information label of the products for which unit versions are managed, as shown below.

Example for NJ-series NJ501-□□□□ CPU Unit:




The following information is provided on the ID information label.

| Item | Description |
|---|---|
| Unit model | Gives the model of the Unit. |
| Unit version | Gives the unit version of the Unit. |
| Lot number and serial number | Gives the lot number and serial number of the Unit.<br>DDMYY: Lot number, □: For use by OMRON, xxxx: Serial number<br>“M” gives the month (1 to 9: January to September, X: October, Y: November, Z: December) |
| MAC address | Gives the MAC address of the built-in port on the Unit. |

### Confirming Unit Versions with Sysmac Studio

You can use the Unit Production Information on the Sysmac Studio to check the unit version of the CPU Unit, CJ-series Special I/O Units, CJ-series CPU Bus Units, and EtherCAT slaves. The unit versions of CJ-series Basic I/O Units cannot be checked from the Sysmac Studio.

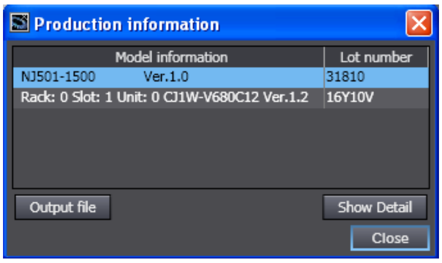
#### ● CPU Unit and CJ-series Units

1. Double-click **CPU/Expansion Racks** under **Configurations and Setup** in the Multiview Explorer. Or, right-click **CPU/Expansion Racks** under **Configurations and Setup** and select ***Edit*** from the menu.

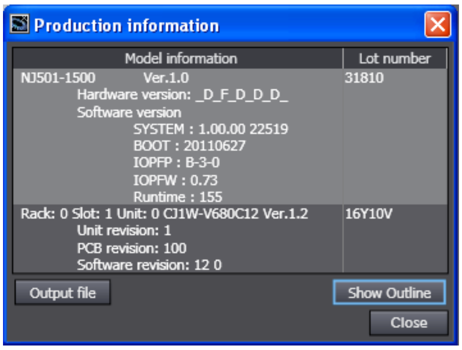
   The Unit Editor is displayed for the Controller Configurations and Setup layer.

**2** Right-click any open space in the Unit Editor and select ***Production Information***.

The Production Information Dialog Box is displayed.




Simple Display




Detailed Display

In this example, “Ver.1.0” is displayed next to the unit model.

The following items are displayed.

| CPU Unit | CJ-series Units |
|---|---|
| Unit model | Unit model |
| Unit version | Unit version |
| Lot number | Lot number |
| | Rack number, slot number, and unit number |

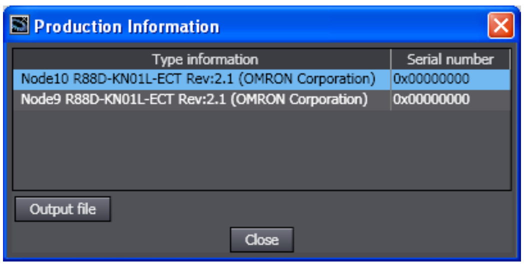
## ● EtherCAT Slaves

**1** Double-click **EtherCAT** under **Configurations and Setup** in the Multiview Explorer. Or, right-click **EtherCAT** under **Configurations and Setup** and select ***Edit*** from the menu.

The EtherCAT Configuration Tab Page is displayed for the Controller Configurations and Setup layer.

**2** Right-click the master in the EtherCAT Configurations Editing Pane and select **Display Production Information**.

The Production Information Dialog Box is displayed.




The following items are displayed.
Node address
Type information*
Serial number

* If the model number cannot be determined (such as when there is no ESI file), the vendor ID, product code, and revision number are displayed.

## Unit Version Notation

In this manual, unit versions are specified as shown in the following table.

| Product nameplate | Notation in this manual | Remarks |
|---|---|---|
| “Ver.1.0” or later to the right of the lot number | Unit version 1.0 or later | Unless unit versions are specified, the information in this manual applies to all unit versions. |

# Related Manuals

The following manuals are related to the NJ-series Controllers. Use these manuals for reference.

| Manual name | Cat. No. | Model numbers | Application | Description |
|---|---|---|---|---|
| NJ-series CPU Unit Hardware User's Manual | W500 | NJ501-□□□□ | Learning the basic specifications of the NJ-series CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided. | An introduction to the entire NJ-series system is provided along with the following information on a Controller built with an NJ501 CPU Unit.<br>• Features and system configuration<br>• Introduction<br>• Part names and functions<br>• General specifications<br>• Installation and wiring<br>• Maintenance and inspection<br>Use this manual together with the *NJ-series CPU Unit Software User's Manual* (Cat. No. W501). |
| NJ-series CPU Unit Software User's Manual | W501 | NJ501-□□□□ | Learning how to program and set up an NJ-series CPU Unit. Mainly software information is provided. | The following information is provided on a Controller built with an NJ501 CPU Unit.<br>• CPU Unit operation<br>• CPU Unit features<br>• Initial settings<br>• Programming based on IEC 61131-3 language specifications<br>Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500). |
| NJ-series CPU Unit Motion Control User's Manual | W507 | NJ501-□□□□ | Learning about motion control settings and programming concepts. | The settings and operation of the CPU Unit and programming concepts for motion control are described. Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ-series CPU Unit Software User's Manual* (Cat. No. W501). |
| NJ-series Instructions Reference Manual | W502 | NJ501-□□□□ | Learning about the specifications of the instruction set that is provided by OMRON. | The instructions in the instruction set (IEC 61131-3 specifications) are described. When programming, use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ-series CPU Unit Software User's Manual* (Cat. No. W501). |
| NJ-series Motion Control Instructions Reference Manual | W508 | NJ501-□□□□ | Learning about the specifications of the motion control instructions that are provided by OMRON. | The motion control instructions are described. When programming, use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500), NJ-series CPU Unit Software User's Manual (Cat. No. W501) and *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507). |
| CJ-series Special Unit Manuals for NJ-series CPU Unit | W490<br>W498<br>W499<br>W491<br>Z317<br>W492<br>W494<br>W497 | CJ1W-□□□□ | Learning how to use CJ-series Units with an NJ-series CPU Unit. | The methods and precautions for using CJ-series Units with an NJ501 CPU Unit are described, including access methods and programming interfaces. Manuals are available for the following Units.<br>Analog I/O Units, Insulated-type Analog I/O Units, Temperature Control Units, ID Sensor Units, High-speed Counter Units, Serial Communications Units, and DeviceNet Units.<br>Use these manuals together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ-series CPU Unit Software User's Manual* (Cat. No. W501). |

| Manual name | Cat. No. | Model numbers | Application | Description |
|---|---|---|---|---|
| NJ-series CPU Unit Built-in EtherCAT Port User's Manual | W505 | NJ501-□□□□ | Using the built-in EtherCAT port on an NJ-series CPU Unit. | Information on the built-in EtherCAT port is provided. This manual provides an introduction and provides information on the configuration, features, and setup.<br>Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ-series CPU Unit Software User's Manual* (Cat. No. W501). |
| NJ-series CPU Unit Built-in EtherNet/IP Port User's Manual | W506 | NJ501-□□□□ | Using the built-in EtherNet/IP port on an NJ-series CPU Unit. | Information on the built-in EtherNet/IP port is provided. Information is provided on the basic setup, tag data links, and other features.<br>Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ-series CPU Unit Software User's Manual* (Cat. No. W501). |
| NJ-series Troubleshooting Manual | W503 | NJ501-□□□□ | Learning about the errors that may be detected in an NJ-series Controller. | Concepts on managing errors that may be detected in an NJ-series Controller and information on individual errors are described.<br>Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ-series CPU Unit Software User's Manual* (Cat. No. W501). |
| Sysmac Studio Version 1 Operation Manual | W504 | SYSMAC-SE2□□□ | Learning about the operating procedures and functions of the Sysmac Studio. | Describes the operating procedures of the Sysmac Studio. |
| CX-Integrator CS/CJ/CP/NSJ-series Network Configuration Tool Operation Manual | W464 | | Learning how to configure networks (data links, routing tables, Communications Unit settings, etc.). | Describes operating procedures for the CX-Integrator. |
| CX-Designer User's Manual | V099 | | Learning to create screen data for NS-series Programmable Terminals. | Describes operating procedures for the CX-Designer. |
| CX-Protocol Operation Manual | W344 | | Creating data transfer protocols for general-purpose devices connected to CJ-series Serial Communications Units. | Describes operating procedures for the CX-Protocol. |

# Revision History

A manual revision code appears as a suffix to the catalog number on the front and back covers of the manual.

**Cat. No. W508-E1-01**

Revision code

| Revision code | Date | Revised content |
|---|---|---|
| 01 | July 2011 | Original production |

# 1

# Introduction to Motion Control Instructions

This section gives an introduction to motion control instructions supported by NJ-series CPU Units.

# 1-1 Motion Control Instructions

Motion control instructions are used in the user program to execute motion controls for an NJ-series Controller. These instructions are defined as function blocks.

The motion control instructions of the MC Function Module are based on the technical specifications of function blocks for PLCopen motion control.

There are two types of motion control instructions: PLCopen-defined instructions and instructions that are unique to the MC Function Module. This section provides an overview of the PLCopen motion control function blocks and motion control instructions.

For details on motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Function Blocks for PLCopen Motion Control

PLCopen standardizes motion control function blocks to define a program interface for the languages specified in IEC 61131-3 (JIS B 3503). Single-axis positioning, electronic cams, and multi-axis coordinated control are defined along with basic procedures for executing instructions.

By using PLCopen motion control function blocks, programming can be more easily reused without hardware dependence. Costs for training and support are also reduced.

**Additional Information**

**PLCopen**

PLCopen is a promotion body for IEC 61131-3 that has its headquarters in Europe and a worldwide membership structure. IEC 61131-3 is an international standard for PLC programming. PLCopen Japan is the promotion committee for the Japanese market and consists of members that have concerns related to the Japanese market.

- The website of headquarters of PLCopen in Europe is http://www.plcopen.org/.

## Overview of Motion Control Instructions

This section describes items defined in the technical specifications of function blocks for PLCopen motion control and provides an overview of their application in the MC Function Module.

### Types of Motion Control Instructions

The following table list the different types of motion control instructions.

| Classification | Type | Functional group | Description |
|---|---|---|---|
| Instructions for common commands | Common administration instructions | Cam tables | These instructions are used to control the common status of the MC Function Module, and to manipulate and monitor data. |
| | | Parameters | |
| Instructions for axis commands | Single-axis motion instructions | Single-axis position control | These instructions move single axes. |
| | | Single-axis velocity control | |
| | | Single-axis torque control | |
| | | Single-axis synchronized control | |
| | | Single-axis manual operation | |
| | Single-axis administration instructions | Auxiliary functions for single-axis control | This instructions control or monitor axis status. |
| Instructions for axes group commands | Multi-axis motion instructions | Multi-axes coordinated control | These instructions perform coordinated movement of an axes group. |
| | Multi-axis administration instructions | Auxiliary functions for multi-axes coordinated control | These instructions control or monitor axes group status. |

### State Transitions

State transitions are defined for axes, axes groups, and instruction execution. For details on the state and state transitions of the MC Function Module, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### Execution and Status of Motion Control Instructions

Variables that start instruction execution or that indicate the execution status are defined as common rules for the instructions. There are two input variables that start instruction execution: *Execute* and *Enable*. The output variables that indicate the execution status of an instruction include *Busy*, *Done*, *CommandAborted*, and *Error*.

For detailed specifications of the MC Function Module, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

**Precautions for Correct Use**

The timing in the timing charts that are given in this manual may not necessarily be the same as the timing displayed for data traces on the Sysmac Studio.
Refer to the *NJ-series CPU Unit Software User's Manual* (Cat. No. W501) for details on data tracing.

## Error Processing

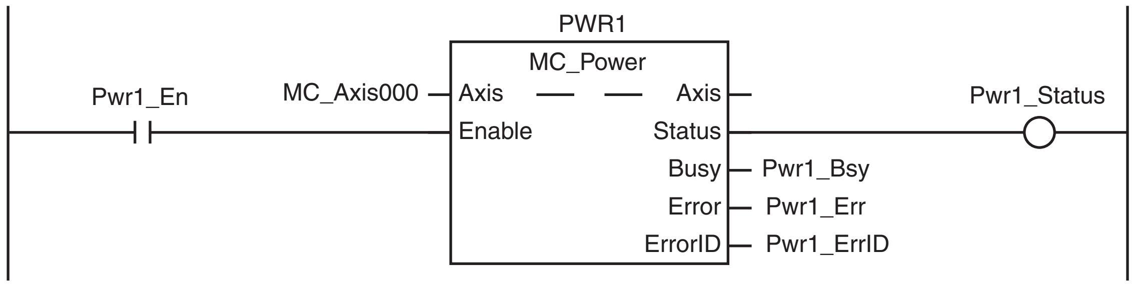
You execute motion control instructions to implement motion control with the MC Function Module. When motion control instructions are executed, input parameters and instruction processing are checked for errors. If an error occurs in an instruction, the *Error* output variable from the instruction changes to TRUE and an error code is output to *ErrorID* output variable.

There are two ways that you can use to program processing of errors for motion control instructions.

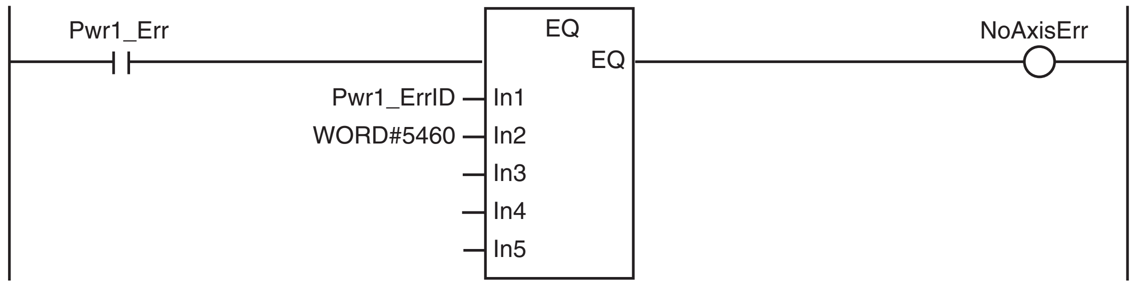
### ● Error Processing for Individual Instructions

You can use the *Error* and *ErrorID* output variables from the instruction to process errors that occur for each instruction.
The following example shows how to determine if an Illegal Axis Specification occurs for the instruction with the instance name PWR1. The instructions are programmed so that error processing is executed if *NoAxisErr* changes to TRUE.

Turning ON the Servo



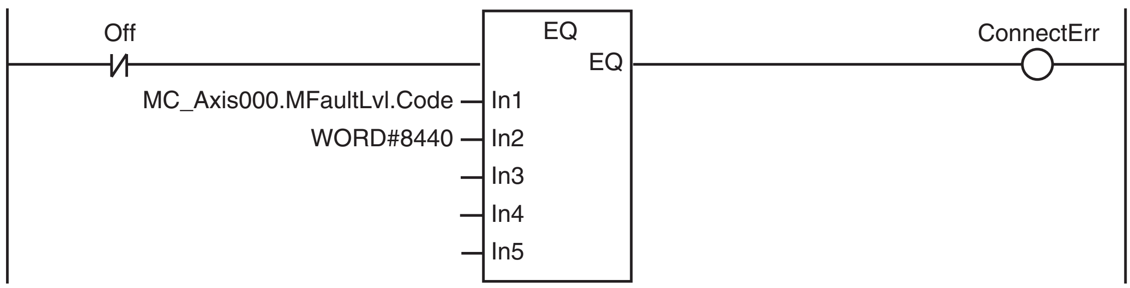

Checking to See If the Specified Axis Exists




### ● Error Processing for Different Types of Errors

You can use the error status that is provided by the system-defined variables for motion control to process each type of error separately.
The following example shows how to determine if a Slave Communications Error occurs for the axis that is called MC_Axis000. The instructions are programmed so that error processing is executed if *ConnectErr* changes to TRUE.

Checking for Communications Errors between the CPU Unit and Servo Drive



## Changing Input Variables during Execution of Motion Control Instructions (Restarting Instructions)

If the values of the input variables to an instruction instance are changed while the motion control instruction is under execution and then *Execute* is changed to TRUE again, operation will follow the new values.

For details on re-execution of MC Function Module instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Multi-execution of Instructions with *BufferMode*

A different instruction instance can be executed during axis motion. You can specify when a motion starts by setting an input variable called *BufferMode*. The following Buffer Modes are supported for *BufferMode*.

- Aborting: Abort (*Aborting*)
- Buffered: Standby (*Buffered*)
- Blending Low: Blending with the low velocity (*BlendingLow*)
- Blending Previous: Blending with the previous velocity (*BlendingPrevious*)
- Blending Next: Blending with the next velocity (*BlendingNext*)
- Blending High: Blending with the high velocity (*BlendingHigh*)

In Aborting Mode, other motions are aborted and the function block is executed immediately. In other modes, the next instruction waits until an output variable such as *Done* or *InVelocity* from the currently executed instruction changes to TRUE. For *Buffered*, the next instruction is executed after the current instruction is executed and *Done* changes to TRUE. For the blending modes, two instruction motions are executed consecutively without pausing. The transition velocity between the two motions is selected from four modes.

For the MC Function Module, *BufferMode* is also referred to as multi-execution of instructions.

For details on multi-execution of instructions for the MC Function Module, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Structures Used for Motion Control

Information required for motion control are defined as structures in PLCopen technical materials. Data type names and basic aspects are defined, but the contents of the structures are not defined.

The main data types defined in PLCopen and the data types used in the MC Function Module are shown in the following table.

| Data type | | Definition |
|---|---|---|
| PLCopen | MC Function Module | |
| AXIS_REF | _sAXIS_REF | This is a structure that contains information on the corresponding axis. |
| AXES_GROUP_REF | _sGROUP_REF | This is a structure that contains information on the corresponding axes group. |
| TRIGGER_REF | _sTRIGGER_REF | This is a structure that contains information on trigger inputs.<br>• Trigger specifications<br>• Detection pattern information (positive, negative, both, edge, level, pattern recognition, etc.) |
| INPUT_REF | --- | This is a structure that contains information relating to the input specifications. It may include virtual data. This data type is not used by the MC Function Module. |
| OUTPUT_REF | --- | This is a structure relating to physical outputs. This data type is not used by the MC Function Module. |

As shown in the above table, the MC Function Module uses some data types that are defined by PLCopen and some that are defined specifically for the MC Function Module.

Refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for definitions of the data types and structures that are handled by the MC Motion Module.

## Precautions for Master and Auxiliary Axes in Synchronized Control

Precautions that are related to sudden changes in velocity and conditions that lead to errors are given below for master and auxiliary axes in synchronized control.

### Sudden Changes in Velocity

When the velocity of the master or auxiliary axis changes suddenly when synchronized motion is started or during synchronized motion, the motion of the slave axis can change suddenly and sometimes place an excessive load on the machine. Take suitable precautions in the following cases because the velocity of the master or auxiliary axis may change suddenly.

- When one of the following four instructions is executed for the master or auxiliary axis:

  MC_ImmediateStop instruction

  MC_SetPosition instruction

  MC_ResetFollowingError instruction

  MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control) instruction

  To ensure that the slave axis does not move suddenly, set suitable input parameters and execution timing for the above instructions or execute them after synchronized control has been released.
- When the immediate stop input signal or limit stop input signal changes to TRUE for the master or auxiliary axis
- When the Servo turns OFF for the master or auxiliary axis

  When the Servo is turned OFF when the master or auxiliary axis is a vertical axis, the position of the axis may change suddenly. Take suitable measures to prevent the slave axis from moving suddenly, such as applying a brake to the master or auxiliary axis or turning OFF the Servo after synchronized control has been released.
- When you change the control mode of the Servo Drive

  Take suitable precautions for changes in the velocity when an instruction is executed. Set suitable input parameters for the instruction.

### Conditions That Lead to Errors

When any of the following four conditions occurs for the master or auxiliary axis when synchronized motion is started or during synchronized motion, a Master Axis Position Read Error or Auxiliary Axis Position Read Error occurs for the slave axis.
The *CommandAborted* output variable from the synchronized control instruction changes to TRUE at the same time.

- EtherCAT process data communications are not established.
- An EtherCAT Slave Communications Error occurs while EtherCAT communications are not established.
- An Absolute Encoder Current Position Calculation Failed error occurs.
- The slave is disconnected.

The following occur if multi-execution of instructions is used for the synchronized control instruction for the slave axis.

- Even if the master or auxiliary axis is in one of the four conditions given above, multi-execution of instructions is acknowledged normally and the instruction is buffered.
- The motion for the buffered instruction is started as normal if none of the above four conditions exist.

**Additional Information**

If the MC_Home instruction is executed for the master or auxiliary axis or if the MC_Power instruction is executed for an axis that uses an absolute encoder, the slave ignores the changes in position of the master or auxiliary axis. Therefore, the slave axis does not move suddenly when defining home.

# 1-2 Basic Information on Motion Control Instructions

This section describes basic specifications and restrictions for programming with motion control instructions for the MC Function Module built into the CPU Unit. For details on motion control instructions, refer to *SECTION 3 Axis Command Instructions*, *Section 4 Axes Group Instructions*, and *Section 5 Common Command Instructions*.

## Motion Control Instruction Names

All motion control instructions for the MC Function Module begin with "MC_".
To see whether an instruction is defined by PLCopen or whether it is an instruction defined for the MC Function Module itself, refer to *2-1 Variables*.

## Languages for Motion Control Instructions

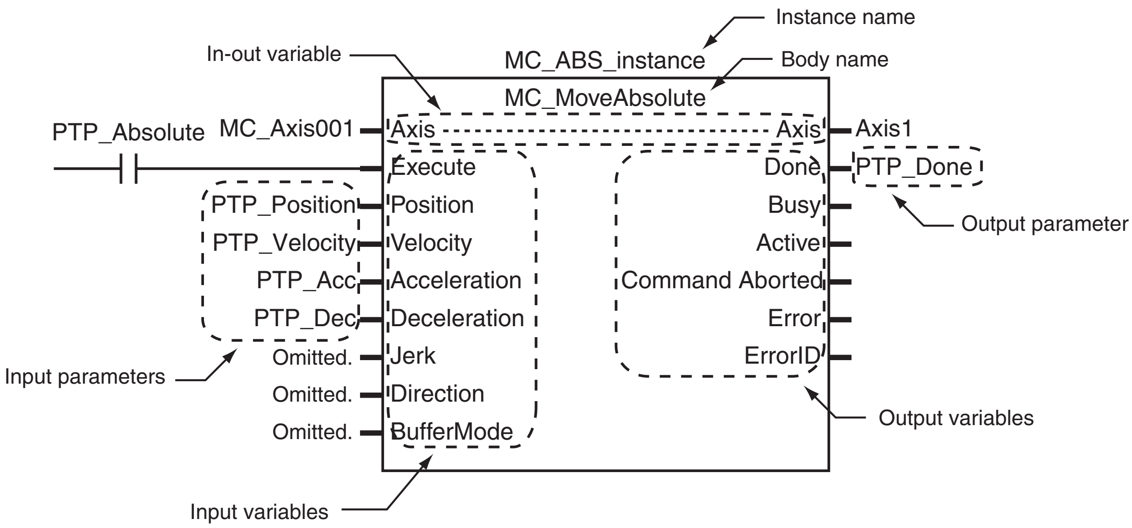
Motion control instructions of the MC Function Module can be used in the following programming languages.

- Ladder diagrams (LD)
- Structured text (ST)

### Ladder Diagrams (LD)

Instruction instances of motion control instructions are located in ladder diagrams. The instruction instances can be named.

The following example shows the MC_MoveAbsolute (Absolute Positioning) instruction.




- The axis variable name of the Servo Drive to control is specified with the in-out variable *Axis*.
- Motion conditions, such as the target position or target velocity, are specified with input variables.
- The status of the instruction or the status of the Servo Drive is output with output variables.
- If input parameters are omitted, input variables are set to default values.
- When you program in a ladder diagram, insert an input between the *Execute* or *Enable* input variable and the left bus bar. If the instruction is connected directly to the left bus bar without an input, an error occurs when the program is built.

## Structured Text (ST)

The instruction instance name is specified. Instruction variables are written from upper left to lower left, then upper right to lower right. The following example shows MC_MoveAbsolute (Absolute Positioning).

```
MC_ABS_instance(
    Axis := MC_Axis001 ,
    Execute := PTP_Absolute ,
    Position := PTP_Position ,
    Velocity := PTP_Velocity ,
    Acceleration := PTP_Acc ,
    Deceleration := PTP_Dec ,
    Jerk := PTP_Jerk ,
    Direction := _mcNoDirection ,
    BufferMode := _mcAborting ,
    Axis => MC_Axis001 ,
    Done => PTP_Done
    ) ;
```

# Motion Control Instruction Locations

This section describes the tasks in which motion control instructions can be located, and the differences in operation that can occur for different locations in the user program.

## Task Types

Motion control instructions can be used in the primary periodic task and in a periodic task with an execution priority of 16 (called a priority 16 periodic task).

| Task type | Applicable? |
|---|---|
| Primary periodic task | OK |
| Periodic task (execution priority: 16) | OK |
| Periodic task (execution priority: 17) | No |
| Periodic task (execution priority: 18) | No |

You can place motion control instructions both in the primary periodic task and in the priority-16 periodic task.

## Function Block Definitions

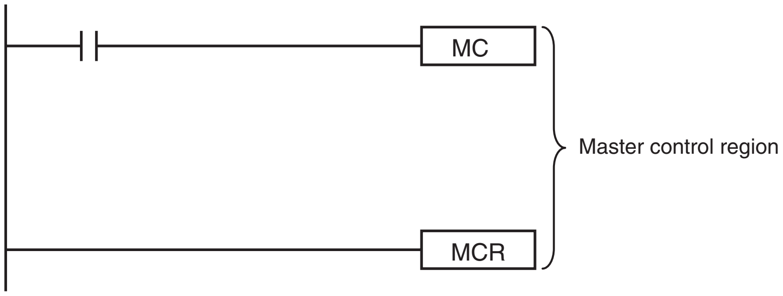
You can also use motion control instructions in user-defined function block definitions.

**Additional Information**

Design efficiency is improved through program structuring, and program visibility is improved if a process with multiple operations is treated as a single function block.

## Master Control Regions

The area in a ladder diagram between the Master Control Start instruction (MC) and the Master Control End instruction (MCR) is the master control region.

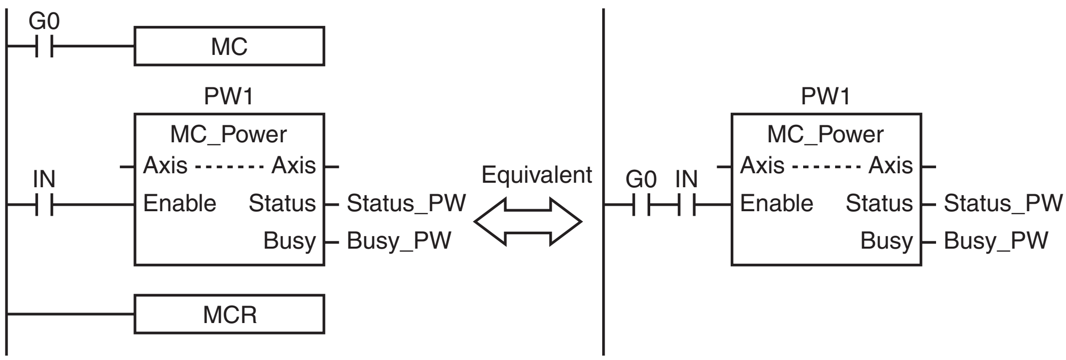


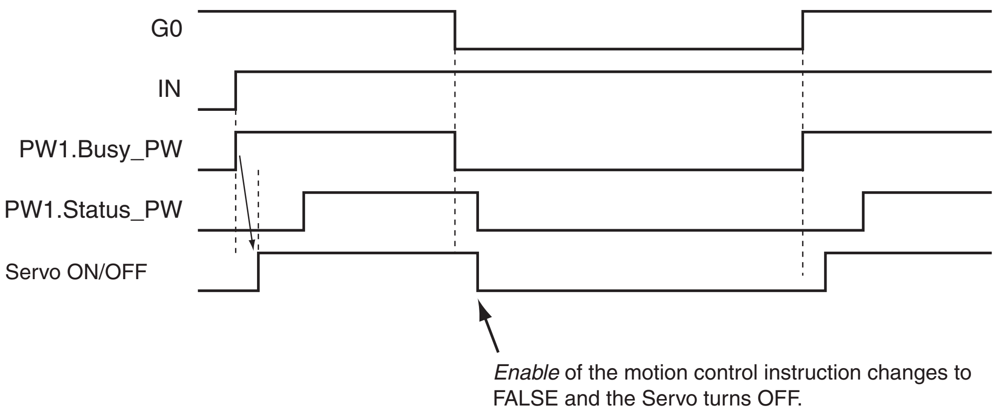

If a motion control instruction is located in the master control region, and the MC input condition is FALSE, the following will occur.

- Motion control instructions for which input variable *Enable* or *Execute* is connected directly to the left bus bar are executed with a FALSE value for *Enable* or *Execute*.
- Inline ST sections are executed normally.
- The values of the output parameters are updated as normal even when the *Enable* or *Execute* input variables to the motion control instructions are FALSE.

### ● Enable-type Motion Control Instructions

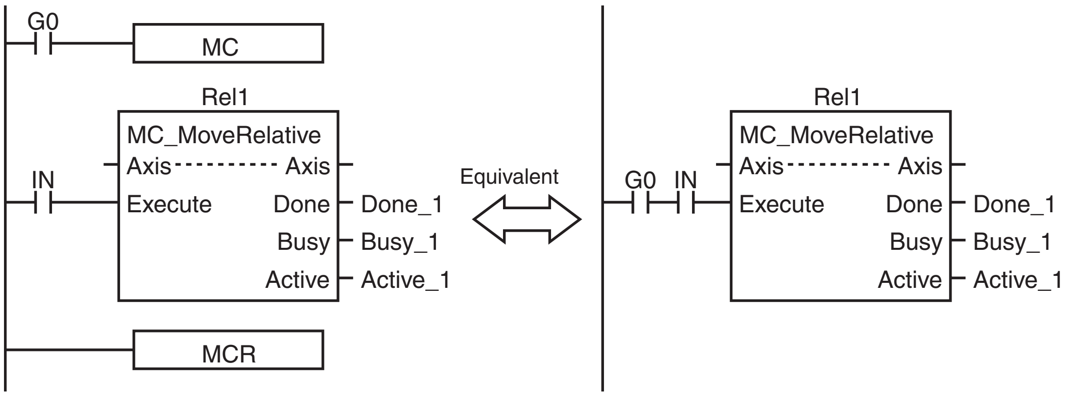
- Instructions located in master control regions are equivalent to the programming shown on the right in the following figure.
- When G0 is TRUE, MC_Power is executed normally.
- When G0 is FALSE, MC_Power is executed as if the *Enable* input variable was FALSE.

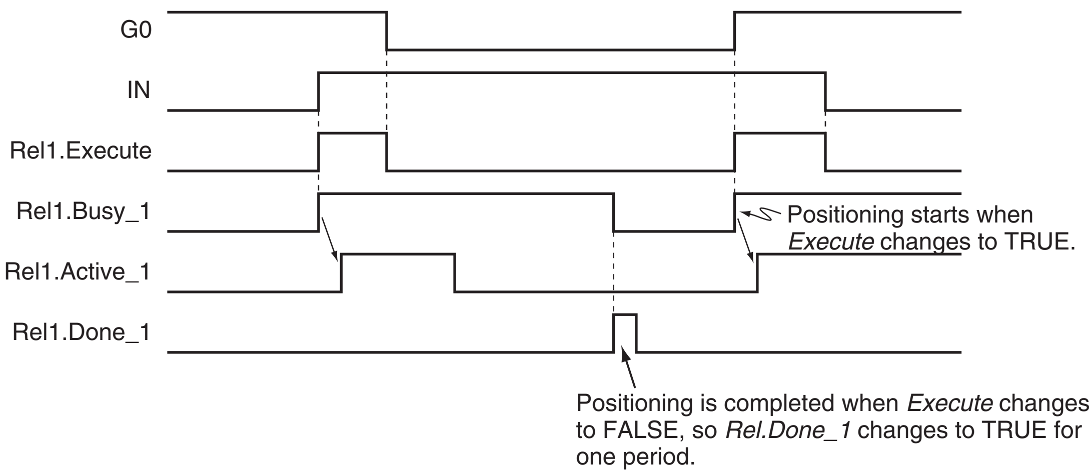





## ● Execute-type Motion Control Instructions

- Instructions located in master control regions are equivalent to the programming shown on the right in the following figure.
- When G0 is TRUE, MC_MoveRelative is executed normally.
- When G0 is FALSE, MC_MoveRelative is executed as if the *Execute* input variable was FALSE.
- Instructions executed when G0 is TRUE continue operation until completion, even if G0 changes to FALSE during operation. The values of output parameters are also updated in the normal way.




### Precautions for Correct Use

Execute-type motion control instructions are executed when G0 changes to TRUE. It is not recommended to use them in the master control region. If they must be used, be careful of the operation.

**Additional Information**

- The function of the MC (Master Control Start) instruction is disabled in ST. All instructions in ST are executed normally.

For details on the MC and MCR instructions, refer to the *NJ-series Instructions Reference Manual* (Cat. No. W502).

## Motion Control Instructions in ST Structure Instructions

This section describes the operation of motion control instructions when they are located in ST structures, such as IF, CASE, WHILE, or REPEAT structures.

When the evaluation result for the condition expression of an ST structure instruction is FALSE, the motion control instructions within the structure are executed as if the *Enable* or *Execute* input variable is FALSE.
If execution of an execute-type instruction is started and then the evaluation result changes to FALSE, processing is continued until it completed. In that case, however, the values of the output variables are not updated.

**Precautions for Correct Use**

The execution status of an execute-type instruction in an ST structure will not be clear if the evaluation result of the condition expression changes to FALSE during execution of the instruction. We therefore do not recommend using execution-type instructions in ST structures.
If they must be used, be careful of the operation.

**Additional Information**

To switch the execution of an execute-type instruction with the condition expression, place only the *Execute* input parameter in the ST structure. Place the execute-type instruction itself outside of the ST structure.

For details on the ST structure instructions, refer to the *NJ-series Instructions Reference Manual* (Cat. No. W502).

## Treatment of REAL and LREAL Data

REAL and LREAL are floating-point decimal data types. This section describes how they are expressed and processed.

### ● REAL and LREAL Data Sizes

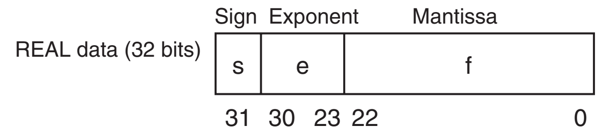
The data sizes of REAL data and LREAL data are different. REAL data has 32 bits and LREAL data has 64 bits.

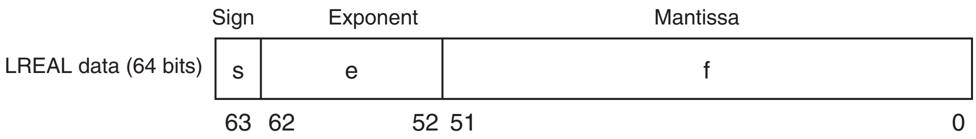
### ● Floating-point Decimal Data Format

A real number in floating-point decimal format is expressed using a sign, exponent, and mantissa. When a real number is substituted in the following formulas, the value corresponding to ‘s’ becomes the sign, ‘e’ the exponent, and ‘f’ the mantissa.

- REAL Data
  Number = $(-1)^s 2^{e-127}(1 + f \times 2^{-23})$
- LREAL Data
  Number = $(-1)^s 2^{e-1023}(1 + f \times 2^{-52})$

The floating-point data format conforms to the IEEE754 standards. The following formats are used.




### Example: Expressing –86.625 as REAL Data

**1** Setting the Sign

The number is negative, so s = 1.

**2** Binary Expression

The number 86.625 is 1010110.101 as a binary number.

**3** Normalized Binary Expression

When the above number is normalized, it becomes $1.010110101 \times 2^6$.

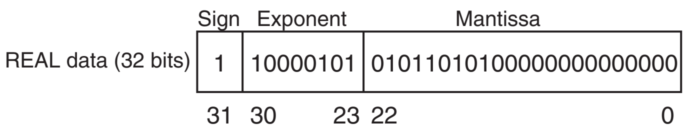
**4** Exponent Expression

From the previous equation, e-127 = 6. Therefore e = 133. The number 133 is 10000101 as a binary number. This expresses the exponent.

**5** Mantissa Expression

Numbers following the decimal point in 1.010110101 are 010110101. This number is expressed using 23 bits, but here there are insufficient digits. Therefore zeros are added. The 23-bit figure becomes f. Therefore f = 01011010100000000000000.

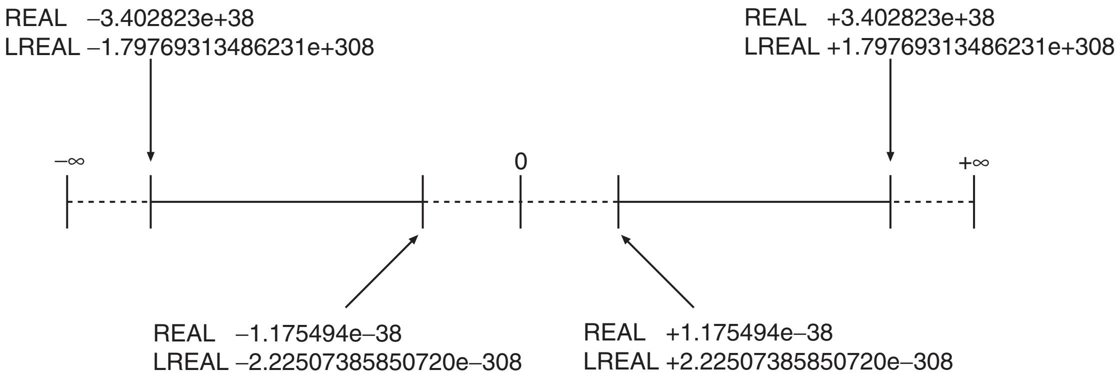
Therefore, –86.625 is expressed as shown in the following figure.




## ● Valid Ranges

The valid ranges of REAL and LREAL are shown in the following table.

| Data type | –∞ | Negative numbers | 0 | Positive number | +∞ |
|---|---|---|---|---|---|
| REAL | –∞ | –3.402823e+38 to –1.175494e-38 | 0 | +1.175494e-38 to +3.402823e+38 | +∞ |
| LREAL | –∞ | –1.79769313486231e+308 to –2.22507385850720e–308 | 0 | +2.22507385850720e–308 to +1.79769313486231e+308 | +∞ |




## ● Special Numbers

Positive infinity, negative infinity, +0, –0, and nonnumeric data are called special numbers. Nonnumeric data is data that cannot be expressed in floating-point decimal format. They are not treated as numbers. Mathematically, +0 and –0 both mean the same as 0, but in data processing it is treated differently.
A detailed explanation is given later. The sign ‘s’, exponent ‘e’, and mantissa ‘f’ for special numbers take on the following values.

| Data type | Special number | Sign s | Exponent e | Mantissa f |
|---|---|---|---|---|
| REAL | +∞ | 0 | 255 | 0 |
| | –∞ | 1 | 255 | 0 |
| | +0 | 0 | 0 | 0 |
| | –0 | 1 | 0 | 0 |
| | Nonnumeric data | --- | 255 | Not 0 |
| LREAL | +∞ | 0 | 2047 | 0 |
| | –∞ | 1 | 2047 | 0 |
| | +0 | 0 | 0 | 0 |
| | –0 | 1 | 0 | 0 |
| | Nonnumeric data | --- | 2047 | Not 0 |

### ● Subnormal Numbers

Numbers that are very close to 0 (with very small absolute values) cannot be expressed using the floating-point decimal format. Subnormal numbers were introduced to expand the validity of numbers near 0. Subnormal numbers can be used to express numbers whose absolute values are smaller than numbers expressed in the normal data format.

**Additional Information**

Values expressed in the normal data format are called normalized numbers or normal numbers.

Numbers with exponent e=0 and mantissa $f \neq 0$ are considered subnormal numbers and their values are expressed in the following manner.

- REAL Data
  Number = $(-1)^s 2^{-126}(f \times 2^{-23})$
- LREAL Data
  Number = $(-1)^s 2^{-1022}(f \times 2^{-52})$

#### Example: Expressing 0.75 x $2^{-127}$ as REAL Data

**1** Setting the Sign

The number is positive, so s = 0.

**2** Binary Expression

The number 0.75 is 0.11 as a binary number.

**3** Mantissa Calculation

From $(0.11)_2 \times 2^{-127} = 2^{-126}(f \times 2^{-23})$, $f = (0.11)_2 \times 2^{22}$.

**4** Mantissa Expression

From the previous equation, f = 01100000000000000000000.

Therefore, $0.75 \times 2^{-127}$ is expressed as shown in the following figure.

| | Sign | Exponent | Mantissa |
|---|---|---|---|
| REAL data (32 bits) | 0 | 00000000 | 01100000000000000000000 |
| | 31 | 30 23 | 22 0 |

Subnormal numbers have fewer effective digits than normalized numbers. Therefore, if the calculation of a normalized number results in an subnormal number, or if an intermediate result is an subnormal number, the number of effective digits of the calculated result may be less than that of the normalized number.

### Data Processing

The floating-point decimal format is an approximate expression of a value, with a slight error from the actual value. There is a limit to the valid range of the value. For these reasons, the following process should be used for calculation.

#### Rounding

If the actual value exceeds the effective digits of the mantissa, the value must be rounded according to the following rules.

- Of the values that can be expressed in floating-point decimal format, the value that is closest to the actual value is taken as the calculation result.
- If there are two values that are equally close to the actual value that can be expressed in floating-point decimal format, the value with the lowest significant 0 bit is taken as the calculation result. When there are two values that are equally close to the actual value, the actual value is exactly in the middle of the two values.

#### Overflows and Underflows

When the true absolute value exceeds the values that can be expressed by a floating-point data type, it is called an overflow. On the other hand, if the value is smaller than the values that can be expressed by a floating-point decimal data type, it is called an underflow.

- If the sign of the true value is positive, the processing result will be positive infinity when an overflow occurs. If the sign of the true value is negative, the processing result will be negative infinity when an overflow occurs.
- If the sign of the true value is positive, the processing result will be +0 when an underflow occurs. If the sign of the true value is negative, the processing result will be –0 when an overflow occurs.

#### Calculating with Special Numbers

The following rules apply when calculating with special numbers. For details on special values, refer to *Special Numbers* on page 1-15.

- Adding positive infinity and negative infinity results in nonnumeric data.
- Subtracting infinity from infinity with the same signs results in nonnumeric data.
- Multiplying +0 by infinity or –0 by infinity results in nonnumeric data.
- Dividing +0 by +0, –0 by –0, or infinity by infinity results in nonnumeric data.
- Adding +0 and –0 results in +0.
- Subtracting +0 from +0, or –0 from –0 results in +0.
- Basic arithmetic operations including nonnumeric data results in nonnumeric data.
- Comparison instructions such as the CMP instruction treat +0 and –0 as the same value.
- If a nonnumeric number is included in a comparison, the comparison instruction always returns "not equal."

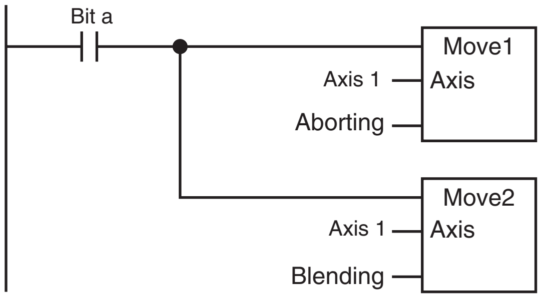
**Precautions for Correct Use**

Floating-point decimal (LREAL) variables are used to set electronic gears, target positions, and other parameters of motion control instructions in the MC Function Module. For this reason, calculation results contain rounding errors. For example, if the MC_MoveRelative (Relative Positioning) instruction is repeatedly executed, following error will accumulate. If the accumulated error becomes a problem, set the command unit to pulses, or specify an absolute position with the MC_MoveAbsolute (Absolute Positioning) instruction.

## Multi-execution of Motion Control Instructions

This section describes executing multiple motion control instructions for the same axis within the same task period.

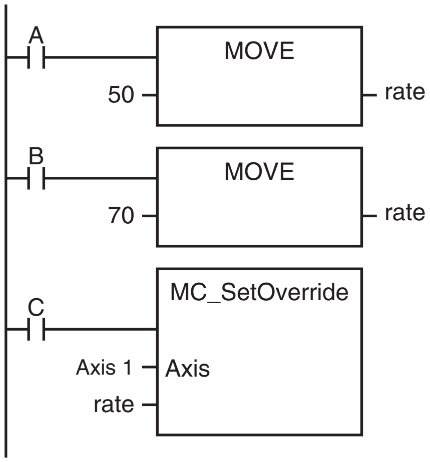
- In the following programming, instruction instances Move1 and Move2 start in the same task period when bit a turns ON.
- Instructions in a program are executed from the top. Therefore Move1 is started first, and then Move2 is started before Move1 is finished.
- This is considered multi-execution of motion control instructions (*BufferMode*). In this example, *Blending* is used to execute Move2 in relation to Move1.




For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

**Additional Information**

If the MC_SetOverride (Set Override Factors) instruction is executed simultaneously in the same way as the instructions shown above, the override value is valid even when it is placed on the bottom. When different override values are set with the MC_SetOverride (Set Override Factors) instruction, the following type of programming is recommended.



## Online Editing of Motion Control Instructions

You can perform the following online editing operations for motion control instructions from the Sysmac Studio.

| Online editing operations |
|---|
| Deleting motion control instructions |
| Adding motion control instructions |
| Adding input variables, output variables, and in-out variables to motion control instructions |
| Changing input variables, output variables, and in-out variables in motion control instructions |
| Deleting input variables, output variables, and in-out variables in motion control instructions. |

**Precautions for Correct Use**

If instructions to stop the axis motion, such as MC_Stop or MC_GroupStop, are deleted while the axis is still moving, the axis may not stop depending on the contents of the user program. Make sure that it is safe to use the online editing before using it for motion control instructions.

## Changes in the Operating Mode of the CPU Unit

An NJ-series CPU Unit has two operating modes: PROGRAM mode and RUN mode. This section describes the operation of the MC Function Module when the operating mode changes.

### Changes from RUN Mode to PROGRAM Mode

- The motion control instruction that is under execution will be aborted. The *CommandAborted* output variable remains FALSE, but the operation is the same as when *CommandAborted* is TRUE.
- If the axis is moving, it will decelerate to a stop at the maximum deceleration. The Servo ON/OFF status will continue.
- If saving the cam table is in progress for the Cam Table Save instruction, the save operation continues.
- Motion control instructions located in a priority 16 periodic task perform the above process after the END instruction in the task is executed.

## Changes from PROGRAM Mode to RUN Mode

- The output variables of the motion control instructions are cleared.
- The axis decelerates to a stop when the mode changes from RUN mode to PROGRAM mode. If the operating mode is changed back to RUN mode while the axis is decelerating, the output variables from the motion control instruction are cleared. Therefore, *CommandAborted* of the motion control instruction that was under execution remains FALSE.

**Additional Information**

- To enable accessing output variables for motion control instructions even after the operating mode changes, assign variables that have output parameters with a Retain attribute. By accessing the assigned output parameter, you can access the output variable immediately before the operating mode changes.
- The Servo ON/OFF status will continue even if the mode is changed.

# 2



# Variables and Instructions

This section describes the variables and instructions for the Motion Control Function Module.

# 2-1 Variables

There are two types of variables for the MC Function Module.
The first type is system-defined variables, which you use to monitor axis status and some of the parameter settings. System-defined variables that are used by the MC Function Module are called system-defined variables for motion control.
The second type is variables that are used to input arguments to motion control instructions and to output execution status from motion control instructions. Some input variables to motion control instruction are enumerated variables. With enumerated variables, selections are made from a set of enumerators.
This section describes the variable types, the valid ranges of motion control instruction input variables, and the enumerated variables.

## ● System-defined Variables for Motion Control

| Level 1 | Level 2 | Level 3 | Description |
|---|---|---|---|
| System-defined variables | Device variables | MC Common variable | You can monitor the overall status of the MC Function Module. |
| | | Axis Variables | You can monitor axis status and the settings of part of the axis parameters. |
| | | Axes Group Variable | You can monitor axes group status and the settings of part of the axes group parameters. |

For details on system-defined variables for motion control, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## ● Variables for Motion Control Instructions

| Type | Outline |
|---|---|
| Input variables | Instruction arguments |
| Output variables | Instruction execution status monitoring information |
| In-out variables | Specify data to process with the instruction |

**Additional Information**

- Data types that start with "_e" are enumerations.
- Data types that start with "_s" are structures.

For details on the data types that are handled by the MC Function Module, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## MC Common Variables

The variable name *_MC_COM* is used for the MC Common Variables. The data type is _sCOMMON_REF, which is a structure. This section describes the configuration of the MC common variables and provides details on the members.

| Name | Data type | Meaning | Function |
|---|---|---|---|
| _MC_COM | _sCOMMON_REF | MC Common Variable | |
| Status | _sCOMMON_REF_STA | MC Common Status | |
| RunMode | BOOL | MC Run | TRUE during MC Function Module operation. |
| TestMode | BOOL | MC Test Run | TRUE during test mode operation from the Sysmac Studio. |
| CamTableBusy | BOOL | Cam Table Busy | TRUE while the Cam Table is being saved or on standby. |
| PFaultLvl | _sMC_REF_EVENT | MC Common Partial Fault | |
| Active | BOOL | MC Common Partial Fault Active | TRUE while there is an MC common partial fault. |
| Code | WORD | MC Common Partial Fault Code | Contains the code for an MC common partial fault. The upper four digits of the event code have the same value. |
| MFaultLvl | _sMC_REF_EVENT | MC Common Minor Fault | |
| Active | BOOL | MC Common Minor Fault Active | TRUE while there is an MC common minor fault. |
| Code | WORD | MC Common Minor Fault Code | Contains the code for an MC common minor fault. The upper four digits of the event code have the same value. |
| Obsr | _sMC_REF_EVENT | MC Common Observation | |
| Active | BOOL | MC Common Observation Active | TRUE while there is an MC common observation. |
| Code | WORD | MC Common Observation Code | Contains the code for an MC common observation. The upper four digits of the event code have the same value. |

# Axis Variables

The variable names of the system-defined axis Variables are *_MC_AX[0..63]*. The data type is _sAXIS_REF, which is a structure. This section describes the configuration of the Axis Variables and provides details on the members.

| Name | | | Data type | Meaning | Function |
|---|---|---|---|---|---|
| _MC_AX[0..63] | | | _sAXIS_REF | Axis Variable | |
| | Status | | _sAXIS_REF_STA | Axis Status | |
| | | Ready | BOOL | Axis Ready-to-execute | TRUE when preparations for axis execution are finished and the axis is stopped. This variable gives the same status as *_MC_AX[XX].Status.Standstill* (TRUE: stopped). |
| | | Disabled | BOOL | Axis Disabled | TRUE while the Servo is OFF for the axis. This includes the following status.<br>The following axis status are mutually exclusive. Only one of them can be TRUE at a time.<br>Disabled, Standstill, Discrete, Continuous, Synchronized, Homing, Stopping, ErrorStop, and Coordinated |
| | | Standstill | BOOL | Standstill | TRUE while the Servo is ON for the axis. |
| | | Discrete | BOOL | Discrete Motion | TRUE while position control is executed toward the target position.<br>This includes when the velocity is 0 because the override factor was set to 0 during a discrete motion. |
| | | Continuous | BOOL | Continuous Motion | TRUE during continuous motion without a target position.<br>This state exists during velocity control and torque control.<br>This includes when the velocity is 0 because the target velocity is set to 0 and when the velocity is 0 due to an override factor set to 0 during continuous motion. |
| | | Synchronized | BOOL | Synchronized Motion | TRUE during execution of synchronized control.<br>This includes waiting for synchronization after changing to synchronized control instructions. |
| | | Homing | BOOL | Homing | TRUE when homing for the MC_Home instruction. |
| | | Stopping | BOOL | Deceleration Stopping | TRUE until the axis stops for a MC_Stop or MC_TouchProbe instruction. This includes when *Execute* is TRUE after the axis stops for an MC_Stop instruction. Axis motion instructions are not executed while decelerating to a stop. (*CommandAborted* is TRUE.) |
| | | ErrorStop | BOOL | Error Deceleration Stopping | This status exists when the axis is stopping or stopped for execution of the MC_ImdediateStop instruction or a minor fault (while *_MC_AX[XX].MFaultLvl.Active* is TRUE (Axis Minor Fault Active).<br>Axis motion instructions are not executed in this state (CommandAborted = TRUE). |
| | | Coordinated | BOOL | Coordinated Motion | TRUE when an axes group is enabled by a multi-axis coordinated control instruction. |

| Name | | | Data type | Meaning | Function |
|---|---|---|---|---|---|
| | Details | | _sAXIS_REF_DET | Axis Control Status[*1] | |
| | | Idle | BOOL | Idle | TRUE when processing is not currently performed for the command value, except when waiting for in-position state.[*2]<br>*Idle* and *InPosWaiting* are mutually exclusive. They cannot both be TRUE at the same time. |
| | | InPosWaiting | BOOL | In-position Waiting | TRUE when waiting for in-position state. The in-position check is performed when positioning for the in-position check. |
| | | Homed | BOOL | Home Defined | TRUE when home is defined.<br>TRUE: Home defined.<br>FALSE: Home not defined. |
| | | InHome | BOOL | In Home Position | TRUE when the axis is in the range for home. It gives an AND of the following conditions.<br>• Home defined<br>• The actual current position is in the zero position range with home as the center.<br>TRUE also when the zero position is passed by while the axis is moving in command status. |
| | | VelLimit | BOOL | Command Velocity Limit | TRUE while the axis velocity is held to the maximum velocity during synchronized control. |
| | Dir | | _sAXIS_REF_DIR | Command Direction[*3] | |
| | | Posi | BOOL | Positive Direction | TRUE when there is a command in the positive direction. |
| | | Nega | BOOL | Negative Direction | TRUE when there is a command in the negative direction. |
| | DrvStatus | | _sAXIS_REF_STA_DRV | Servo Drive Status[*4] | |
| | | ServoOn | BOOL | Servo ON | TRUE when the Servomotor is powered. |
| | | Ready | BOOL | Servo Ready | TRUE when the Servo is ready. |
| | | MainPower | BOOL | Main Power | TRUE when the Servo Drive main power is ON. |
| | | P_OT | BOOL | Positive Limit Input | TRUE when the positive limit input is enabled. |
| | | N_OT | BOOL | Negative Limit Input | TRUE when the negative limit input is enabled. |
| | | HomeSw | BOOL | Home Proximity Input | TRUE when the home proximity input is enabled. |
| | | Home | BOOL | Home Input | TRUE when the home input is enabled.[*5] |
| | | ImdStop | BOOL | Immediate Stop Input | TRUE when the immediate stop input is enabled. |
| | | Latch1 | BOOL | External latch input 1 | TRUE when latch input 1 is enabled. |
| | | Latch2 | BOOL | External latch input 2 | TRUE when latch input 2 is enabled. |
| | | DrvAlarm | BOOL | Drive Error Input | TRUE while there is a Servo Drive error. |
| | | DrvWarning | BOOL | Drive Warning Input | TRUE while there is a Servo Drive warning. |
| | | ILA | BOOL | Drive Internal Limiting | TRUE when the Servo Drive limiting function actually limits the axis. This corresponds to one of the following limits in the G5-series.[*6]<br>Torque limit<br>Velocity limit<br>Drive prohibit input<br>Software limits |
| | | CSP | BOOL | Cyclic Synchronous Position (CSP) Mode | TRUE when the Servo is ON at the Servo Drive and the current mode is CSP Mode.[*7] |

| Name | | | Data type | Meaning | Function |
|---|---|---|---|---|---|
| | | CSV | BOOL | Cyclic Synchronous Velocity (CSV) Mode | TRUE when the Servo is ON at the Servo Drive and the current mode is CSV Mode.*7 |
| | | CST | BOOL | Cyclic Synchronous Torque (CST) Mode | TRUE when the Servo is ON at the Servo Drive and the current mode is CST Mode.*7 |
| | Cmd | | _sAXIS_REF_CMD_DATA | Axis Command Value | |
| | | Pos | LREAL | Command Current Position | Contains the current value of the command position. (Unit: command units)<br>When the Servo is OFF and the mode is not position control mode, this variable contains the actual current position. |
| | | Vel | LREAL | Command Current Velocity | Contains the current value of the command velocity. (Unit: command units/s)<br>A plus sign is added when traveling in the positive direction, and a minus sign when traveling in the negative direction. The velocity is calculated from the difference with the command current position. When the Servo is OFF and the mode is not the position control mode, the velocity is calculated based on the actual current position. |
| | | AccDec | LREAL | Command Current Acceleration/Deceleration | Contains the current value of the command acceleration/deceleration rate. (Unit: command units/$s^2$)<br>The acceleration/deceleration rate is calculated from the difference with the command current velocity. A plus sign is added for acceleration, and a minus sign is added for deceleration. Zero when the command acceleration/deceleration rate of the instruction under execution is 0. |
| | | Jerk | LREAL | Command Current Jerk | Contains the current value of the command jerk. (Unit: command units/$s^3$)<br>A plus sign is added when the absolute value of acceleration/deceleration is increasing, and a minus sign is added when it is decreasing. Zero when the command acceleration/deceleration rate and command jerk of the instruction under execution is 0. |
| | | Trq | LREAL | Command Current Torque | Contains the current value of the command torque. (Unit: %)<br>A plus sign is added when traveling in the positive direction, and a minus sign when traveling in the negative direction. Contains the same value as the actual current torque except in torque control mode. |
| | Act | | _sAXIS_REF_ACT_DATA | Axis Current Value | |
| | | Pos | LREAL | Actual Current Position | Contains the actual current position. (Unit: command units) |
| | | Vel | LREAL | Actual Current Velocity | Contains the actual current velocity. (Unit: command units/s)<br>A plus sign is added when traveling in the positive direction, and a minus sign when traveling in the negative direction. |
| | | Trq | LREAL | Actual Current Torque | Contains the current value of the actual torque. (Unit: %)<br>A plus sign is added when traveling in the positive direction, and a minus sign when traveling in the negative direction. |
| | MFaultLvl | | _sMC_REF_EVENT | Axis Minor Fault | |
| | | Active | BOOL | Axis Minor Fault Active | TRUE while there is an axis minor fault. |
| | | Code | WORD | Axis Minor Fault Code | Contains the code for an axis minor fault. The upper four digits of the event code have the same value. |

| Name | | | Data type | Meaning | Function |
|---|---|---|---|---|---|
| | Obsr | | _sMC_REF_EVENT | Axis Observation | |
| | | Active | BOOL | Axis Observation Active | TRUE while there is an axis observation. |
| | | Code | WORD | Axis Observation Code | Contains the code for an axis observation. The upper four digits of the event code have the same value. |
| | Cfg | | _sAXIS_REF_CFG | Axis Basic Settings*8 | |
| | | AxNo | UINT | Axis Number | Contains the logical number of the axis.<br>This number is accessed to recognize the axis number when accessing _sAXIS_REF. |
| | | AxEnable | _eMC_AXIS_USE | Axis Use | Shows if the axis is enabled or disabled.<br>0: _mcNoneAxis (Undefined Axis)<br>1: _mcUnusedAxis (Unused Axis)<br>2: _mcUsedAxis (Used Axis) |
| | | AxType | _eMC_AXIS_TYPE | Axis Type | Contains the axis type.<br>I/O wiring is not required for virtual axes.<br>0: _mcServo (Servo Axis)<br>1: _mcEncdr (Encoder Axis)<br>2: _mcVirServo (Virtual Servo Axis)<br>3: _mcVirEncdr (Virtual Encoder Axis) |
| | | NodeAddress | UINT | Node Address | Contains the EtherCAT slave address.<br>A value of 16#FFFF indicates that there is no address. |
| | Scale | | _sAXIS_REF_SCALE | Unit Conversion Settings*9 | |
| | | Num | UDINT | Command Pulse Count Per Motor Rotation | Contains the number of pulses per motor rotation for command positions.<br>The command value is converted to a number of pulses based on the electronic gear ratio. |
| | | Den | LREAL | Work Travel Distance Per Motor Rotation | Contains the workpiece travel distance per motor rotation for command positions. |
| | | Units | _eMC_UNITS | Display Unit | Contains the display unit for command positions.<br>0: _mcPls(pulse)<br>1: _mcMm(mm)<br>2: _mcUm(μm)<br>3: _mcNm(nm)<br>4: _mcDeg(degree)<br>5: _mcInch(inch) |

*1 Gives the control status of the command.

*2 This also includes states where processing is performed while in motion at velocity 0, during following error counter resets, during synchronized control, and during multi-axis coordinated motion.

*3 Gives the command travel direction.

*4 Gives the status of the Servo Drive.

*5 This variable shows the status of the signal that is set for the Encoder Phase Z Detection setting of the digital inputs in the Detailed Settings Area of the Axis Basic Settings Display of the Sysmac Studio. You may not be able to map this signal to a PDO for a servo driver from another manufacturer. Refer to the manual for the servo driver.

*6 This variable shows the status of bit 11(internal limit active) of the Status word (6041 hex) mapped to a PDO. The condition for it to change to TRUE depends on the specifications of the Servo Drive. Refer to the manual for the servo driver.

*7 These variables are based on the value of the Modes of operation display (6061 hex) mapped to a PDO. The conditions for *CSP*, *CSV*, and *CST* to change to TRUE depend on the specifications of the Servo Drive. Refer to the manual for the servo driver. They are always FALSE if the Modes of operation display (6061 hex) is not mapped to a PDO.

*8 This variable shows the settings in the Axis Basic Settings.

*9 This variable shows the settings of the electronic gear ratio.

## Axes Group Variables

The variable names of the system-defined Axes Group Variables are *_MC_GRP[0..31]*. The data type is _sGROUP_REF, which is a structure. This section describes the configuration of the Axes Group Variables and provides details on the members.

| Name | | | Data type | Meaning | Function |
|---|---|---|---|---|---|
| MC_GR[0..31] | | | _sGROUP_REF | Axes Group Variable | |
| | Status | | _sGROUP_REF_STA | Axes Group Status | |
| | | Ready | BOOL | Ready-to-execute | TRUE when the axes group is stopped and is ready to execute. The condition for being ready to execute is an AND of the following conditions.<br>• Execution of the MC_Stop instruction is not in progress for a composition axis.<br>• *_MC_GRP[XX].Status.Standby* (Stopped) is TRUE.<br>• The Servo is ON for the composition axes.<br>• *_MC_AX[XX].Details.Homed* is TRUE (home defined) for the composition axes. |
| | | Disabled | BOOL | Axes Group Disabled | TRUE when the axes group is disabled and stopped. The following axes group status are mutually exclusive. Only one of them can be TRUE at a time.<br>Disabled, Standstill, Moving, Stopping, and ErrorStop |
| | | StandStill | BOOL | Standstill | TRUE when the axes group motion instruction is stopped.<br>This is not related to the Servo ON/OFF status of the composition axes in the axes group. |
| | | Moving | BOOL | Moving | TRUE while an axes group motion instruction is executed toward the target position.<br>This includes in-position waiting status and when the velocity is 0 for an override. |
| | | Stopping | BOOL | Deceleration Stopping | TRUE until the axes group stops for an MC_GroupStop instruction. This includes when *Execute* is TRUE after the axis stops for an MC_GroupStop instruction. Axes group motion instructions are not executed while decelerating to a stop. (*CommandAborted* is TRUE.) |
| | | ErrorStop | BOOL | Error Deceleration Stopping | TRUE while the axes group is stopping or stopped for the MC_GroupImmediateStop instruction or for an axes group minor fault (when *_MC_GRP[XX].MFaultLvl.Active* is TRUE). Axes group motion instructions are not executed in this state (CommandAborted = TRUE). |
| | Details | | _sGROUP_REF_DET | Axes Group Control Status*1 | |
| | | Idle | BOOL | Idle | TRUE when processing is not currently performed for the command value, except when waiting for in-position state.*2<br>*Idle* and *InPosWaiting* are mutually exclusive. They cannot both be TRUE at the same time. |
| | | InPosWaiting | BOOL | In-position Waiting | TRUE when any of the composition axes are waiting for in-position state. The in-position check performed when positioning for the in-position check. |

| Name | | | Data type | Meaning | Function |
|---|---|---|---|---|---|
| | Cmd | | _sGROUP_REF_CMD_DATA | Axes Group Command Values | |
| | | Vel | LREAL | Command Interpolation Velocity | Contains the current value of the command interpolation velocity. The interpolation velocity is calculated from the difference with the interpolation command current position. A plus sign is added when traveling in the positive direction, and a minus sign is added when traveling in the negative direction. The value is 0 when the axes group is disabled. |
| | | AccDec | LREAL | Command Interpolation Acceleration/Deceleration | Contains the current value of the command interpolation acceleration/deceleration. The interpolation acceleration/deceleration rate is calculated from the difference with the command interpolation velocity. A plus sign is added for acceleration, and a minus sign is added for deceleration. The value is 0 when the axes group is disabled, or when the command acceleration/deceleration rate of the current axes group motion instruction is 0. |
| | MFaultLvl | | _sMC_REF_EVENT | Axes Group Minor Fault | |
| | | Active | BOOL | Axes Group Minor Fault Active | TRUE while there is an axes group minor fault. |
| | | Code | UINT | Axes Group Minor Fault Code | Contains the code for an axes group minor fault. The upper four digits of the event code have the same value. |
| | Obsr | | _sMC_REF_EVENT | Axes Group Observation | |
| | | Active | BOOL | Axes Group Observation Active | TRUE while there is an axes group observation. |
| | | Code | WORD | Axes Group Observation Code | Contains the code for an axes group observation. The upper four digits of the event code have the same value. |
| | Cfg | | _sGROUP_REF_CFG | Axes Group Basic Settings | |
| | | GrpNo | UINT | Axes Group Number | Contains the logical number of the axes group. This number is accessed to recognize the axes group number when accessing *_sGROUP_REF*. |
| | | GrpEnable | _eMC_GROUP_USE | Axes Group Use | Shows if the axes group is enabled or disabled.<br>0: _mcNoneGroup (Undefined Axes Group)<br>1: _mcUnusedGroup (Unused Axes Group)<br>2: _mcUsedGroup (Used Axes Group) |
| | Kinematics | | _sGROUP_REF_KIM | Kinematics Transformation Settings*3 | |
| | | GrpType | _eMC_GROUP_TYPE | Axis Composition | Gives the axis composition for multi-axes coordinated control.<br>0: _mcXY (two axes)<br>1: _mcXYZ (three axes)<br>2: _mcXYZU (four axes) |
| | | Axis[0] | UINT | Composition Axis Selection (Axis A0) | Gives the axis number that is assigned to axis A0. |
| | | Axis[1] | UINT | Composition Axis Selection (Axis A1) | Gives the axis number that is assigned to axis A1. |
| | | Axis[2] | UINT | Composition Axis Selection (Axis A2) | Gives the axis number that is assigned to axis A2. |
| | | Axis[3] | UINT | Composition Axis Selection (Axis A3) | Gives the axis number that is assigned to axis A3. |

*1 Gives the control status of the command.

*2 This also includes states where processing is performed while in motion at a velocity of 0.

*3 Gives the definition of the kinematic conversions for the axes group.

# Input Variables for Motion Control Instructions

The following tables list the input variables and the valid ranges for motion control instructions, and the valid ranges of enumerations.

## Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. Other input variables are also input when *Execute* changes to TRUE. If input values are changed, they will be updated when *Execute* changes to *TRUE* again. The output variables are valid as long as *Execute* remains TRUE even after the instruction is completed. Then, all output variables except for *Error* and *ErrorID* are disabled when *Execute* changes to FALSE. If *Execute* changes to FALSE before the instruction is completed, output variables are valid for at least one period. |
| Enable | Enable | BOOL | TRUE or FALSE | FALSE | The instruction function is enabled when *Enable* changes to TRUE and disabled when it changes to FALSE. While *Enable* is TRUE, the other input variables are input every period. If *Enable* changes to FALSE, all output variables except for *Error* and *ErrorID* are disabled. |
| Positive Enable | Positive Direction Enable | BOOL | TRUE or FALSE | FALSE | • MC_MoveJog Instruction When this variable changes to TRUE, the axis starts moving in the positive direction. When it changes to FALSE, the axis stops moving. The *Velocity*, *Acceleration*, and *Deceleration* input variables to the MC_MoveJog instruction are read when *PositiveEnable* changes to TRUE.<br>• MC_SetTorqueLimit Instruction When this variable changes to TRUE, the positive torque limit is enabled. When it changes to FALSE, the positive torque limit is disabled. |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Negative Enable | Negative Direction Enable | BOOL | TRUE or FALSE | FALSE | • MC_MoveJog Instruction<br>When this variable changes to TRUE, the axis starts moving in the negative direction. When it changes to FALSE, the axis stops moving. The *Velocity*, *Acceleration*, and *Deceleration* input variables to the MC_MoveJog instruction are read when *NegativeEnable* changes to TRUE.<br>• MC_SetTorqueLimit Instruction<br>When this variable changes to TRUE, the negative torque limit is enabled. When it changes to FALSE, the negative torque limit is disabled. |
| BufferMode | Buffer Mode Selection | _eMC_BUFFER_MODE | 0: _mcAborting<br>1: _mcBuffered<br>2: _mcBlendingLow<br>3: _mcBlendingPrevious<br>4: _mcBlendingNext<br>5: _mcBlendingHigh | 0*1 | Specifies the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered<br>2: Blend with low<br>3: Blend with previous<br>4: Blend with next<br>5: Blend with high |
| Velocity | Target Velocity | LREAL | Positive number*2 | 0 | Specifies the target velocity.*3 |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specifies the acceleration rate.*4 |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specifies the deceleration rate.*4 |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specifies the jerk.*5 |
| Distance | Travel Distance | LREAL | Negative number, positive number, or 0 | 0 | Specifies the travel distance from the command current position. |
| | | ARRAY [0..3] OF LREAL | Negative number, positive number, or 0 | 0 | Specifies the target position for linear interpolation.*6 |
| Position | Target Position | LREAL | Negative number, positive number, or 0 | 0 | Specifies the absolute target position.*6 |
| | | ARRAY [0..3] OF LREAL | Negative number, positive number, or 0 | 0 | Specifies the target position for linear interpolation.*6 |
| VelFactor | Velocity Override Factor | LREAL | 0 to 500 | 100 | Specifies the velocity override factor.<br>The valid range of the override factors is between 0.01and 500.00.<br>Values above 500.00 are treated as 500 and values less then 0.01 (including negative values) are treated as 0.01.<br>The override factor will be 0 only when 0 is specified.<br>The unit is %. |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| AccFactor (Reserved) | Acceleration/ Deceleration Override Factor | LREAL | 0 to 500 | 100 | (Reserved) |
| JerkFactor (Reserved) | Jerk Override Factor | LREAL | 0 to 500 | 100 | (Reserved) |
| ReferenceType*7 | Position Type Selection | _eMC_REFERENCE_ TYPE | 0: _mcCommand<br>1: _mcFeedback<br>2: _mcLatestCommand | 0*1 | Specifies the master axis input information.<br>0: Command position (value calculated in the previous primary period)<br>1: Actual position (value obtained in the same primary period)<br>2: Command position (value calculated in the same primary period) |
| FeedDistance | Feed Distance | LREAL | Negative number, positive number, or 0 | 0 | Specifies the travel distance after the interrupt feed input. |
| FeedVelocity | Feed Velocity | LREAL | Positive number | 0 | Specifies the travel target velocity after the interrupt feed input. |
| ErrorDetect | Error Detection Selection | BOOL | TRUE or FALSE | FALSE | Specifies whether to detect an error when there is no interrupt feed input.<br>TRUE: Detect errors.<br>FALSE: Do not detect errors. |
| ExecutionMode | Execution Mode | BOOL | TRUE or FALSE | FALSE | Specifies whether to execute the specified cam table periodically or only once.<br>TRUE: Periodic<br>FALSE: Non-periodic |
| StartMode | Start Mode | _eMC_STARTMODE | 0: _mcAbsolutePosition<br>1: _mcRelativePosition | 0*1 | Specifies the coordinates used by *MasterStartDistance* (master following distance).<br>0: Absolute position<br>1: Relative position |
| StartPosition | Cam Table Start Position | LREAL | Negative number, positive number, or 0 | 0 | Specifies the starting point of the cam table (0 phase) as an absolute position of the master axis. |
| MasterStartDistance | Master Following Distance | LREAL | Negative number, positive number, or 0 | 0 | Specifies the position of the master axis when the following axis starts the cam motion. If you specify absolute positioning for *StartMode*, specify the absolute position of the master axis. If you specify relative positioning for *StartMode*, specify the relative position of the master axis from *StartPosition* (Cam Table Start Position). |
| MasterScaling | Master Coefficient | LREAL | Positive value (>0.0) | 1.0 | The master axis phase is extended or contracted using the specified scale. |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| SlaveScaling | Slave Axis Coefficient | LREAL | Positive value (>0.0) | 1.0 | The slave axis displacement is extended or contracted using the specified scale. |
| MasterOffset | Master Offset | LREAL | Negative number, positive number, or 0 | 0 | The phase of the master axis is shifted using the specified offset value. |
| SlaveOffset | Slave Offset | LREAL | Negative number, positive number, or 0 | 0 | The displacement of the slave axis is shifted using the specified offset value. |
| CamTransition (Reserved) | Cam Transition Selection | _eMC_CAM_ TRANSITION | 0: _mcCTNone | 0*1 | (Reserved) |
| OutMode (Reserved) | Sync End Mode Selection | _eMC_OUT_MODE | 0: _mcStop | 0*1 | (Reserved) |
| Direction | Direction | _eMC_DIRECTION | 0: _mcPositiveDirection<br>1: _mcShortestWay<br>2: _mcNegativeDirection<br>3: _mcCurrentDirection<br>4: _mcNoDirection | 4/0*1 | Specifies the travel direction.<br>0: Positive direction<br>1: Shortest way<br>2: Negative direction<br>3: Current direction<br>4: No direction specified |
| Continuous(Reserved) | Continuation Mode Selection | BOOL | TRUE or FALSE | --- | (Reserved) |
| RatioNumerator | Gear Ratio Numerator | UINT | Positive number | 10,000 | Specifies the electronic gear ratio numerator between the master and slave axes. |
| Ratio Denominator | Gear Ratio Denominator | UINT | Positive number | 10,000 | Specifies the electronic gear ratio denominator between the master and slave axes. |
| MasterSync Position | Master Sync Position | LREAL | Negative number, positive number, or 0 | 0 | Specifies the absolute master sync position. |
| SlaveSyncPosition | Slave Sync Position | LREAL | Negative number, positive number, or 0 | 0 | Specifies the absolute slave sync position. |
| SlaveDistance | Slave Axis Travel Distance | LREAL | Negative number, positive number, or 0 | 0 | Specifies the travel distance for the slave axis. |
| MasterDistance | Master Distance | LREAL | Non-negative number | 0 | Specifies the travel distance of the master axis. |
| MasterDistance-InACC | Master Distance in Acceleration | LREAL | Non-negative number | 0 | Specifies the travel distance of the master axis while the slave axis is accelerating. |
| MasterDistance-InDEC | Master Distance in Deceleration | LREAL | Non-negative number | 0 | Specifies the travel distance of the master axis while the slave axis is decelerating. |
| LinkOption | Synchronization Start Condition | _eMC_LINKOPTION | 0: _mcCommandExecution<br>1: _mcTriggerDetection<br>2: _mcMasterReach | 0*1 | Specifies the condition for the slave axis to synchronize with the master axis.<br>0: When instruction execution starts<br>1: When trigger is detected<br>2: When the master axis reaches the master following distance. |
| CombineMode | Combine Mode | _eMC_COMBINE_MODE | 0: _mcAddAxes<br>1: _mcSubAxes | 0*1 | Specifies the combining method.<br>0: Addition<br>1: Subtraction |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Ratio Numerator Master (Reserved) | Master Axis Gear Ratio Numerator | UINT | Positive number | 10000 | Specifies the electronic gear ratio numerator between the master and slave axes. |
| Ratio Denominator Master (Reserved) | Master Axis Gear Ratio Denominator | UINT | Positive number | 10000 | Specifies the denominator of the electronic gear ratio between the master and slave axes. |
| Ratio Numerator Auxiliary (Reserved) | Auxiliary Axis Gear Ratio Numerator | UINT | Positive number | 10000 | Specifies the numerator of the electronic gear ratio between the auxiliary and slave axes. |
| Ratio Denominator Auxiliary (Reserved) | Auxiliary Axis Gear Ratio Denominator | UINT | Positive number | 10000 | Specifies the denominator of the electronic gear ratio between the auxiliary and slave axes. |
| ReferenceType Master | Master Axis Position Type | _eMC_ REFERENCE_TYPE | 1: _mcFeedback<br>2: _mcLatestCommand | 2[*1] | Specifies the position type of the master axis.<br>1: Actual position (value obtained in the same primary period)<br>2: Command position (value calculated in the same primary period) |
| ReferenceType Auxiliary | Auxiliary Axis Position Type | _eMC_ REFERENCE_TYPE | 1: _mcFeedback<br>2: _mcLatestCommand | 2[*1] | Specifies the position type of the auxiliary axis.<br>1: Actual position (value obtained in the same primary period)<br>2: Command position (value calculated in the same primary period) |
| PhaseShift | Phase Shift Amount | LREAL | Negative number, positive number, or 0 | 0 | Specifies the master phase shift amount.[*6] |
| Torque | Target Torque | LREAL | 0 to 1000.0 | 300.0 | Specify the target torque to output to the Servo Drive in increments of 0.1%. The target torque is specified as a percentage of the rated torque. The unit is %. |
| TorqueRamp | Torque Ramp | LREAL | Non-negative number | 0 | Specifies the rate of change in the torque from the current value to the target torque. The unit is %/s. |
| PositiveValue | Positive Torque Limit | LREAL | 0.1 to 1000.0 or 0.0 | 300.0 | Specifies the torque limit in the positive direction in increments of 0.1%.<br>If a value that exceeds the Maximum Positive Torque Limit axis parameter, the positive torque will be the Maximum Positive Torque Limit.<br>The value will be 0 if 0 or a negative value is specified. |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| NegativeValue | Negative Torque Limit | LREAL | 0.1 to 1000.0 or 0.0 | 300.0 | Specifies the torque limit in the negative direction in increments of 0.1%.<br>If a value that exceeds the Maximum Negative Torque Limit axis parameter, the negative torque will be the Maximum Negative Torque Limit.<br>The value will be 0 if 0 or a negative value is specified. |
| WindowOnly | Window Only | BOOL | TRUE or FALSE | FALSE | Specify whether to enable or disable the window mask. |
| FirstPosition | First Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the first position. |
| LastPosition | Last Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the last position. |
| StopMode | Stop Mode | _eMC_STOP_MODE | 1: _mcImmediateStop<br>2: _mcImmediateStopFEReset<br>3: _mcFreeRunStop<br>4: _mcNonStop | 4*1 | Specifies the stopping method.<br>1: Perform an immediate stop.<br>2: Perform an immediate stop and reset the following error counter<br>3: Perform an immediate stop and turn OFF the Servo<br>4: Do not stop. |
| Relative (Reserved) | Relative Position Selection | BOOL | TRUE or FALSE | --- | (Reserved) |
| Execution Mode (Reserved) | Execution Mode | _eMC_EXECUTION_MODE | 0: _mcImmediately | 0*1 | (Reserved) |
| Permitted Deviation | Permitted Following Error | LREAL | Non-negative number | 0 | Specifies the permitted maximum value for the following error between the master and slave axes. |
| CmdPos Mode | Command Current Position Count Selection | _eMC_CMDPOS_MODE | 0: _mcCount | 0*1 | 0: Use the actual current position and update the command current position.<br>Home remains defined. |
| CoordSystem | Coordinate System | _eMC_COORD_SYSTEM | 0: _mcACS | 0*1 | Specifies the coordinate system.<br>0: Axis coordinate system (ACS) |
| TransitionMode | Transition Mode | _eMC_TRANSITION_MODE | 0: _mcTMNone<br>10: _mcTMCornerSuperimposed | 0*1 | Specifies the path of motion.<br>0: Transition disabled<br>10:Superimpose corners |
| MoveMode | Travel Mode | _eMC_MOVE_MODE | 0: _mcAbsolute<br>1: _mcRelative | 0*1 | Selects the travel method.<br>0: Absolute positioning<br>1: Relative positioning |
| CircAxes | Circular Axes | ARRAY [0,1] OF UINT | 0 to 3 | 0 | Specifies the axes for circular interpolation.<br>0: Axis A0<br>1: Axis A1<br>2: Axis A2<br>3: Axis A3 |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| CircMode | Circular Interpolation Mode | _eMC_CIRC_MODE | 0: _mcBorder<br>1: _mcCenter<br>2: _mcRadius | 0*1 | Specifies the method for circular interpolation.<br>0: Border point<br>1: Center<br>2: Radius |
| AuxPoint | Auxiliary Point | ARRAY [0,1] OF LREAL | Negative number, positive number, or 0 | 0 | Specifies the border point, center, or radius. |
| EndPoint | End Point | ARRAY [0,1] OF LREAL | Negative number, positive number, or 0 | 0 | Specifies the target position. |
| PathChoice | Path Choice | _eMC_CIRC_PATHCHOICE | 0: _mcCW<br>1: _mcCCW | 0*1 | Specifies the path direction.<br>0: CW<br>1: CCW |
| Parameter Number | Parameter Number | _eMC_PARAMETER_NUMBER | 0: _mcChkVel<br>1: _mcChkAcc<br>2: _mcChkDec<br>3: _mcPosiChkTrq<br>4: _mcNegaChkTrq<br>5: _mcFELmt<br>6: _mcChkFELmt<br>7: _mcSwLmtMode<br>8: _mcPosiSwLmt<br>9: _mcNegaSwLmt<br>10: _mcInPosTime | 0*1 | Specifies the parameter to write.<br>0: Velocity Warning Value<br>1: Acceleration Warning Value<br>2: Deceleration Warning Value<br>3: Positive Torque Warning Value<br>4: Negative Torque Warning Value<br>5: Following Error Over Limit Value<br>6: Following Error Warning Value<br>7: Software Limits<br>8: Positive Software Limit<br>9: Negative Software Limit<br>10:In-position Check Time |

*1 The default value for an enumeration variable is actually not the number, but the enumerator.

*2 You can use instructions, such as the MC_MoveJog or MC_MoveVelocity instruction, to set the velocity to 0.

*3 The command unit is millimeters, micrometers, nanometers, degrees, inches, or pulses.

*4 The unit is command units/$s^2$.

*5 The unit is command units/$s^3$.

*6 This unit is command units.

*7 When you use *_mcLatestCommand*, the axis number set for the master axis in the system-defined variable for motion control must be lower than the axis number set for the slave axis in the system-defined variable for motion control .

## Valid Range of Input Variables

This section gives the valid ranges of input variables to motion control instructions. Refer to individual instruction descriptions for the valid ranges for each instruction.

### ● BOOL Input Variables

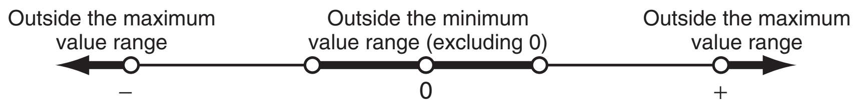
Any value other than FALSE is treated as TRUE. For this reason, out-of-range errors do not occur.

### ● Enumerated (ENUM) Input Variables

Values that are outside of the valid range will result in an error.

### ● Input Variables Given as Full Range, Positive Number, or Negative Number

Operation when an input variable is set inside or outside the valid range is described in the following table.




| Name | Meaning | Valid range | Outside the maximum value range | Outside the minimum value range (excluding 0) |
|---|---|---|---|---|
| Velocity | Velocity | 0, (–1 ≤ and ≤ –Maximum velocity), or (1 ≤ and ≤ Maximum velocity)[*1] | Set to the maximum velocity for a positive number, and to the –maximum velocity for a negative number.[*2] | Set to 1 pulse/s when positive number, and –1 pulse/s when negative number.[*2] |
| Acceleration | Acceleration Rate | 0 or (0.004 ≤ and ≤ Maximum acceleration)[*3] | Set to the maximum acceleration.<br>If the acceleration time[*4] is less than 125 μs, it will always be 125 μs. | Set to 0.004 pulses/s$^2$ when positive number. If the acceleration time[*4] is greater than 250 s, it will always be 250 s.<br>Error when negative number. |
| Deceleration | Deceleration Rate | 0 or (0.004 ≤ and ≤ Maximum deceleration)[*5] | Set to the maximum deceleration.<br>If the deceleration time[*4] is less than 125 μs, it will always be 125 μs. | Set to 0.004 pulses/s$^2$ when positive number. If the deceleration time[*4] is greater than 250 s, it will always be 250 s.<br>Error when negative number. |
| Jerk | Jerk | 0 or (0.000016 ≤ and 25,600,000,000,000,000 pulses/s$^3$) | Set to 25,600,000,000,000,000 pulses/s$^3$. If the acceleration jerk application time[*6] or the deceleration jerk application time[*6] is less than 125 μs, it will always be 125 μs. | Set to 0.000016 pulses/s$^3$. If the acceleration jerk application time[*6] or the deceleration jerk application time[*6] is greater than 250 s, it will always be 250 s.<br>Error when negative number. |
| Distance | Travel Distance | (0xFFFFFF0000000001) ≤ and ≤ (0x000000FFFFFFFFFF) | Error | Values outside of the minimum value range do not occur. |
| Position | Command Position | (0xFFFFFF8000000000) ≤ and < (0x0000007FFFFFFFFF+ 1)[*7] | Error | Values outside of the minimum value range do not occur. |
| VelFactor | Velocity Override Factor | 0 or 0.01 ≤ and ≤ 500.00[*8] | Set to 500.00% if higher than 500.00%. | Set to 0.01% if less than 0.01%. |

| Name | Meaning | Valid range | Outside the maximum value range | Outside the minimum value range (excluding 0) |
|---|---|---|---|---|
| Velocity | Interpolation Velocity | 0.000 000 000 000 01 ≤ and ≤ Maximum interpolation velocity*9 | Set to the maximum interpolation velocity. | Set to 0.000 000 000 1 pulses/s. |
| Acceleration | Interpolation Acceleration | 0 or (0.000 000 000 000 4 ≤ and ≤ Maximum interpolation acceleration)*10 | Set to the maximum interpolation acceleration. If the interpolation acceleration time*11 is less than 125 µs, it will always be 125 µs. | Set to 0.000 000 000 000 4 pulses/$s^2$ when positive number. If the interpolation acceleration time*11 is greater than 250 s, it will always be 250 s.<br>Error when negative number. |
| Deceleration | Interpolation Deceleration | 0 or (0.000 000 000 000 000 04 ≤ and ≤ Maximum interpolation deceleration)*12 | Set to the maximum interpolation deceleration. If the interpolation deceleration time*11 is less than 125 µs, it will always be 125 µs. | Set to 0.000 000 000 000 4 pulses/$s^2$ when positive number. If the interpolation deceleration time*11 is greater than 250 s, it will always be 250 s.<br>Error when negative number. |
| Jerk | Interpolation Jerk | 0 or (0.000 000 000 000 001 6 ≤ and ≤ 51,200,000,000,000,000 pulses/$s^3$) | Set to 51,200,000,000,000,000 pulses/$s^3$. If the interpolation acceleration jerk application time*13 or the interpolation deceleration jerk application time*13 is less than 125 µs, it will always be 125 µs. | Set to 0.000 000 000 000 001 6 pulses/$s^3$. If the interpolation acceleration jerk application time*13 or the interpolation deceleration jerk application time*13 is greater than 250 s, it will always be 250 s.<br>Error when negative number. |

*1 The upper limit of the Maximum Velocity in the axis parameters is 400,000,000 pulses/s.

*2 If a negative number or 0 is specified when negative numbers and 0 are not included in the effective range, an error occurs.

*3 The upper limit of the Maximum Acceleration in the axis parameters is 3,200,000,000,000 pulses/$s^2$.

*4 Calculated as follows: Acceleration time = Velocity/Acceleration rate, Deceleration time = Velocity/Deceleration rate, and Acceleration/deceleration time = Acceleration time + Deceleration time.

*5 The upper limit of the Maximum Deceleration in the axis parameters is 3,200,000,000,000 pulses/$s^2$.

*6 The acceleration jerk application time and the deceleration jerk application time are the times that jerk is applied. Calculated as follows: Acceleration jerk application time = Acceleration rate/Jerk and Deceleration jerk application time = Deceleration rate/Jerk.

*7 Position must be an absolute value in pulses and must be no more than 40 bits signed.

*8 The unit is %.

*9 The upper limit of the Maximum Interpolation Velocity in the axis parameters is 800,000,000 pulses/s.

*10 The upper limit of the Maximum Interpolation Acceleration in the axis parameters is 6,400,000,000,000 pulses/$s^2$.

*11 Calculated as follows: Interpolation acceleration time = Interpolation velocity/Interpolation acceleration rate, Interpolation deceleration time = Interpolation velocity/Interpolation deceleration rate, and Acceleration/deceleration time = Acceleration time + Deceleration time.

*12 The upper limit of the Maximum Interpolation Deceleration in the axis parameters is 6,400,000,000,000 pulses/$s^2$.

*13 The interpolation acceleration jerk application time and the interpolation deceleration jerk application time are the times that interpolation jerk is applied.
Calculated as follows: Interpolation acceleration jerk application time = Interpolation acceleration rate/Jerk and Interpolation deceleration jerk application time = Interpolation deceleration rate/Jerk.

## Enumerations

This ENUM data is used by input variables to motion control instructions. An enumeration input variable is not actually set to the number, but to the enumerator.

| Data type | Valid range | Description | Corresponding variable name |
|---|---|---|---|
| _eMC_BUFFER_ MODE | 0: _mcAborting<br>1: _mcBuffered<br>2: _mcBlendingLow<br>3: _mcBlendingPrevious<br>4: _mcBlendingNext<br>5: _mcBlendingHigh | Specifies the operation for multi-execution of motion control instructions.<br>0: Aborting<br>1: Buffered<br>2: Blend with low<br>3: Blend with previous<br>4: Blend with next<br>5: Blend with high | *BufferMode* (Buffer Mode Selection) |
| _eMC_CIRC_ MODE | 0: _mcBorder<br>1: _mcCenter<br>2: _mcRadius | Specifies the method for circular interpolation.<br>0: Border point<br>1: Center<br>2: Radius | *CircMode* (Circular Interpolation Mode) |
| _eMC_CAM_ TRANSITION | 0: _mcCTNone | Specifies the slave axis command value output method when the cam is restarted.<br>0: No limit or correction | *CamTransition* (Cam Transition Selection) |
| _eMC_CIRC_ PATHCHOICE | 0: _mcCW<br>1: _mcCCW | Specifies the path direction.<br>0: CW<br>1: CCW | *PathChoice* |
| _eMC_COMBINE_ MODE | 0: _mcAddAxes<br>1: _mcSubAxes | Specifies the combining method.<br>0: Add<br>1: Subtraction | *CombineMode* |
| _eMC_COORD_ SYSTEM | 0: _mcACS | Specifies the coordinate system.<br>0: Axis coordinate system (ACS) | *CoordSystem* (Coordinate System) |
| _eMC_DIRECTION | 0: _mcPositiveDirection<br>1: _mcShortestWay<br>2: _mcNegativeDirection<br>3: _mcCurrentDirection<br>4: _mcNoDirection | Specifies the direction of motion.<br>0: Positive direction<br>1: Shortest way<br>2: Negative direction<br>3: Current direction<br>4: No direction specified. | *Direction* |
| _eMC_ EXECUTION_MODE | 0: _mcImmediately | (Reserved) | *ExecutionMode* (Execution Mode) |
| _eMC_LINKOPTION | 0: _mcCommandExecution<br>1: _mcTriggerDetection<br>2: _mcMasterReach | Specifies the condition for the slave axis to synchronize with the master axis.<br>0: Start of instruction<br>1: When trigger is detected<br>2: When the master axis reaches the master following distance. | *LinkOption* (Synchronization Start Condition) |
| _eMC_MOVE_MODE | 0: _mcAbsolute<br>1: _mcRelative<br>2: _mcVelocity | Selects the travel method.<br>0: Absolute positioning<br>1: Relative positioning<br>2: Velocity control | *MoveMode* (Travel Mode) |
| _eMC _OUT_MODE | 0: _mcStop | Specifies the mode to disable the synchronization instruction.<br>0: Deceleration stop | *OutMode* (Sync End Mode Selection)<br>(Reserved) |

| Data type | Valid range | Description | Corresponding variable name |
|---|---|---|---|
| _eMC_ PARAMETER_ NUMBER | 0: _mcChkVel<br>1: _mcChkAcc<br>2: _mcChkDec<br>3: _mcPosiChkTrq<br>4: _mcNegaChkTrq<br>5: _mcFELmt<br>6: _mcChkFELmt<br>7: _mcSwLmtMode<br>8: _mcPosiSwLmt<br>9: _mcNegaSwLmt<br>10: _mcInPosTime | Specifies the parameter to write.<br>0: Velocity Warning Value<br>1: Acceleration Warning Value<br>2: Deceleration Warning Value<br>3: Positive Torque Warning Value<br>4: Negative Torque Warning Value<br>5: Following Error Over Limit Value<br>6: Following Error Warning Value<br>7: Software Limits<br>8: Positive Software Limit<br>9: Negative Software Limit<br>10: In-position Check Time | *ParameterNumber* |
| _eMC_ SWLMT_MODE | 0: _mcNonSwLmt<br>1: _mcCmdDecelerationStop<br>2: _mcCmdImmediateStop<br>3: _mcActDecelerationStop<br>4: _mcActImmediateStop | Enables and disables the software limits and specifies the Stop Mode.<br>0: Disable software limits.<br>1: Deceleration stopping enabled for command position.<br>2: Enable software limits and perform immediate stop for command position.<br>3: Enable software limits and decelerate to stop for actual position.<br>4: Enable software limits and perform immediate stop for actual position. | *SettingValue* (Setting Value) |
| _eMC_ REFERENCE_ TYPE * | 0: _mcCommand<br>1: _mcFeedback<br>2: _mcLatestCommand | Specifies the position type.<br>0: Command position (value calculated in the previous primary period)<br>1: Actual position (value obtained in the same primary period)<br>2: Command position (value calculated in the same primary period) | *ReferenceType* (Position Type Selection)<br>*ReferenceTypeMaster* (Master Axis Position Type Selection)<br>*ReferenceTypeAuxiliary* (Auxiliary Axis Position Type Selection) |
| _eMC_ STARTMODE | 0: _mcAbsolutePosition<br>1: _mcRelativePosition | Specifies the coordinate system used by *MasterStartDistance* (master following distance).<br>0: Absolute position<br>1: Relative position | *StartMode* |
| _eMC_STOP_ MODE | 1: _mcImmediateStop<br>2: _mcImmediateStopFEReset<br>3: _mcFreeRunStop<br>4: _mcNonStop | Specifies the stopping method.<br>1: Perform an immediate stop.<br>2: Perform an immediate stop and reset the following error counter<br>3: Turn OFF the Servo.<br>4: Do not stop. | *StopMode* |
| _eMC_TRIGGER_ LATCH_ID | 0: _mcLatch1<br>1: _mcLatch2 | Specifies which of the two latch functions to use.<br>0: Latch 1<br>1: Latch 2 | *LatchID* |
| _eMC_ CMDPOS_MODE | 0: _mcCount | 0: Use the actual current position and update the command current position. Home remains defined. | CmdPosMode |
| _eMC_ TRANSITION_ MODE | 0: _mcTMNone<br>10: _mcTMCornerSuperimposed | Specifies the path of motion.<br>0: Transition disabled<br>10:Superimpose corners | *TransitionMode* |
| _eMC_TRIGGER_ MODE | 0: _mcDrive<br>1: _mcController | Specifies the trigger mode.<br>0: Drive Mode<br>1: Controller Mode | *Mode* |
| _eMC_TRIGGER_ INPUT_DRIVE | 0: _mcEncoderMark<br>1: _mcEXT | Specifies the trigger signal in Drive Mode.<br>0: Z-phase signal<br>1: External input | *InputDrive* (Trigger Input Signal) |

* When you use *mcLatestCommand*, the axis number set for the master axis in the system-defined variable for motion control must be lower than the axis number set for the slave axis in the system-defined variable for motion control .

## Output Variables for Motion Control Instructions

The following table lists the output variables for motion control instructions.

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. At this time, output variables *Active*, *Error,* and *CommandAborted* are FALSE. Done will be TRUE for at least one period if the input variable *Execute* is FALSE when the instruction is completed. If *Execute* is TRUE, Done remains TRUE until *Execute* changes to FALSE. |
| Busy | Executing | BOOL | TRUE or FALSE | Changes to TRUE when an instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | Changes to TRUE when the instruction is executed. *Active* (Controlling) is TRUE while the instruction is actually controlling an axis or axes group. At this time, output variables *Done*, *Error,* and *CommandAborted* are FALSE. |
| Enabled | Enabled | BOOL | TRUE or FALSE | Changes to TRUE when busy. |
| CommandAborted | Instruction Aborted | BOOL | TRUE or FALSE | TRUE when an instruction could not be executed or when it was aborted during execution. The instruction is not executed if there is an error with the target axis or axes group, or while the axis or axes group is decelerating to a stop. The instruction is aborted when another instruction is executed, or if an error other than for this instruction occurs. At this time, output variables *Done*, *Active*, and *Error* change to FALSE. If the instruction is aborted while the input variable *Execute* is FALSE, *CommandAborted* will be TRUE for at least one period. If *Execute* or *Enable* is TRUE, *CommandAborted* remains TRUE until *Execute* or *Enable* changes to FLASE.<br>If *Execute* or *Enable* is TRUE, *CommandAborted* remains TRUE until *Execute* or *Enable* changes to FALSE. |
| Error | Error | BOOL | TRUE or FALSE | TRUE when there is an error caused by a mistake in an input variable or instruction processing.*1 |
| ErrorID | Error Code | WORD | *2 | Contains the error code when an error occurs. 16#0000 indicates normal operation. |
| Failure | Failure End | BOOL | TRUE or FALSE | TRUE when the instruction was not executed correctly. |
| Status | Power Servo | BOOL | TRUE or FALSE | TRUE when the device is ready for operation. |
| EndOfProfile | End of Cam Cycle | BOOL | TRUE or FALSE | Changes to TRUE when the cam table end point is executed. |
| Index | Index | UINT | Non-negative number | Contains the cam data index number. |
| StartSync | Following | BOOL | TRUE or FALSE | TRUE when acceleration/deceleration is started for synchronization and the device is ready for operation. |
| RecordedPosition | Latched Position | LREAL | Negative number, positive number, or 0 | Contains the latched position. *3 |

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Invalid | Excessive Following Error between Axes | BOOL | TRUE or FALSE | TRUE when the permitted following error between axes is exceeded. |
| DeviatedValue | Following Error between Axes | LREAL | Negative number, positive number, or 0 | Contains the difference between the specified master and slave axes.*3 |
| EndPointIndex | End Point Index | UINT | Non-negative number | Contains the cam table end point index. |
| MaxDataNumber | Maximum Number of Cam Data | UINT | Positive number | Contains the maximum cam data number. |
| InVelocity | Target Velocity Reached | BOOL | TRUE or FALSE | TRUE when the target velocity is reached. |
| InSync | In Sync | BOOL | TRUE or FALSE | TRUE when slave axis is synchronized to the master axis, or when the slave axis reaches the slave axis sync position. |
| InGear | Gear Ratio Achieved | BOOL | TRUE or FALSE | TRUE when the slave axis reaches the target velocity. |
| InCombination | Axes Combined | BOOL | TRUE or FALSE | TRUE when axes are combined. |
| InCam | Cam Motion | BOOL | TRUE or FALSE | TRUE when the cam table start point is executed. |
| InTorque | Target Torque Reached | BOOL | TRUE or FALSE | TRUE when the target torque is reached. |
| InFeed | Feeding | BOOL | TRUE or FALSE | TRUE while feeding after receiving a latch input. |
| InZone | In Zone | BOOL | TRUE or FALSE | TRUE when the axes position is within the zone range. |

*1 *Error* is not reset to FALSE until you execute one of the following instructions: MC_Reset, MC_GroupReset, or ResetMCError. This behavior is different from the PLCopen specifications. With PLCopen specifications, it changes to FALSE when Execute changes to FALSE. When *Error* is TRUE, the motion control instruction is not executed. Instructions are not executed after an error is cleared even if *Execute* is TRUE. *Execute* must change from FALSE to TRUE to execute the instruction. Enable-type motion control instructions are executed whenever their *Enable* variable is TRUE.

*2 The upper four digits of the event code have the same value. Refer to *A-1 Error Codes* for details.

*3 This unit is command units. The command unit is millimeters, micrometers, nanometers, degrees, inches, or pulses.

**Additional Information**

To enable accessing output variables for motion control instructions even after the operating mode is changed, assign variables that have output parameters with a retain attribute.
By accessing the assigned output parameter, you can access the output variable immediately before the operating mode changed.

## In-Out Variables for Motion Control Instructions

The following table lists the in-out variables for motion control instructions.

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specifies the axis. |
| AxesGroup | Axes Group | _sGROUP_REF | --- | Specifies the axes group. |
| Auxiliary | Auxiliary Axis | _sAXIS_REF | --- | Specifies the auxiliary axis. |
| Master | Master Axis | _sAXIS_REF | --- | Specifies the master axis. |
| Slave | Slave Axis | _sAXIS_REF | --- | Specifies the slave axis. |
| CamTable | Cam Table | ARRAY[0..N] OF _sMC_CAM_REF | --- | Specifies the cam data structure _sMC_CAM_REF array variable as the cam table.*1 |
| TriggerInput | Trigger Input Condition | _sTRIGGER_REF | --- | Sets the trigger condition. |
| TriggerVariable | Trigger Variable | BOOL | TRUE or FALSE | Specifies a trigger input variable when the controller mode is specified with a trigger condition. |
| Target | Write Target | _sAXIS_REF or _sGROUP_REF | --- | Specifies the axis or axes group for which to write a parameter. |
| SettingValue | Setting Value | Depends on the variable that is specified. | --- | Specifies the value to write.<br>The valid range follows the motion control parameter that is specified by Parameter-Number.*2<br>The default value is 0. |

*1 *N* in the array variable is set automatically by the Sysmac Studio. Specify a cam data variable that was created on the Sysmac Studio.

*2 For details on the data types of variables, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

# 2-2 Instructions

There are three types of motion instructions. They are given in the following table.

| Type | Outline |
|---|---|
| Common commands | Common instructions for the MC Function Module |
| Axis commands | Instructions for MC Function Module to perform single-axis control |
| Axes group commands | Instructions for MC Function Module to perform multi-axis coordinated control |

For details on common commands, refer to *Section 5 Common Command Instructions.* For axes commands, refer to *SECTION 3 Axis Command Instructions*. For axes groups, refer to *Section 4 Axes Group Instructions*.

## Common Commands

This section describes the common instructions for the MC Function Module. The *Classification* Column gives "Administration" for non-motion instructions and "Motion" for motion instructions.

P: Instructions defined in PLCopen technical specifications.
O: Instructions defined for the MC Function Module.

| Instruction | Instruction name | Outline | Classification | |
|---|---|---|---|---|
| MC_SetCamTableProperty | Set Cam Table Properties | The end point index of the cam table that is specified in the input parameter is changed. | Administration | O |
| MC_SaveCamTable | Save Cam Table | Saves the cam table specified with the input parameter. | Administration | O |
| MC_Write | Write MC Setting | Writes part of the parameter settings for motion control. | Administration | O |

For details on the axis states due to instruction execution, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Axis Commands

This section describes the instructions that are used to perform single-axis control for the MC Function Module. The *Classification* column gives "Administration" for non-motion instructions and "Motion" for motion instructions.

P: Instructions defined in PLCopen technical specifications.
O: Instructions defined for the MC Function Module.

| Instruction | Instruction name | Outline | Classification | |
|---|---|---|---|---|
| MC_Power | Power Servo | Makes the Servo Drive ready to operate. | Administration | P |
| MC_MoveJog | Jog | Performs jogging according to the specified target velocity. | Motion | O |
| MC_Home | Home | Operates the motor to determine home using the limit signals, home proximity signal, and home signal. | Motion | P |
| MC_Move | Positioning | Performs absolute positioning or relative positioning. | Motion | O |
| MC_MoveAbsolute | Absolute Positioning | Performs positioning for the specified absolute target position. | Motion | P |
| MC_MoveRelative | Relative Positioning | Performs positioning for the specified travel distance from the command current position. | Motion | P |
| MC_MoveVelocity | Velocity Control | Performs velocity control with the Position Control Mode of the Servo Drive. | Motion | P |
| MC_MoveZeroPosition | High-speed Home | Performs positioning with an absolute position of 0 as the target position to return to home. | Motion | O |
| MC_MoveFeed | Interrupt Feeding | Positioning is performed for the specified travel distance from the position where an external device triggers an interrupt input. | Motion | O |
| MC_Stop | Stop | Decelerates an axis to a stop. | Motion | P |
| MC_ImmediateStopt | Immediate Stop | Stops an axis according to the stopping mode that is set with the StopMode (Stopping Mode Selection) input variable regardless of the status of the axis. | Motion | O |
| MC_SetPosition | Set Position | Changes the command current position or the actual current position as required for an axis. | Administration | P |
| MC_SetOverride | Set Override Factors | Changes the target velocity for an axis. | Administration | P |
| MC_ResetFollowingError | Reset Following Error Counter | Resets the following error between the command position and the actual position. | Motion | O |
| MC_CamIn | Start Cam Operation | Starts cam operation with a specified cam table. | Motion | P |
| MC_CamOut | End Cam Operation | Ends cam operation for the axis specified with the input parameter. | Motion | P |
| MC_GearIn | Start Gear Operation | Specifies the gear ratio between the master axis and the slave axis and starts gear operation. | Motion | P |
| MC_GearInPos | Positioning Gear Operation | Specifies the gear ratio between the master axis and the slave axis and starts electronic gear operation. Specifies the positions of the master axis and slave axis to start synchronization. | Motion | P |
| MC_GearOut | End Gear Operation | Cancels MC_GearIn and MC_GearInPos instructions. | Motion | P |
| MC_MoveLink | Synchronous Positioning | Performs positioning in sync with the specified master axis. | Motion | O |
| MC_CombineAxes | Combine Axes | Outputs the sum or difference of the command positions of two axes as the command position. | Motion | O |
| MC_Phasing | Shift Master Axis Phase | Shifts the phase of the master axis currently in synchronized control. | Motion | P |
| MC_TorqueControl | Torque Control | Uses the Torque Control Mode of the Servo Drive to control the torque. | Motion | P |
| MC_SetTorqueLimit | Set Torque Limit | Limits the torque output from the Servo Drive through the torque limit function of the Servo Drive. | Administration | O |

| Instruction | Instruction name | Outline | Classification | |
|---|---|---|---|---|
| MC_ZoneSwitch | Zone Monitor | Determines if the command position or actual current position of an axis is within a specified zone. | Administration | O |
| MC_TouchProbe | Enable External Latch | Records the position of an axis when a trigger signal occurs. | Administration | P |
| MC_AbortTrigger | Disable External Latch | Disables the current latch. | Administration | P |
| MC_AxesObserve | Monitor Axis Following Error | Monitors the deviation between the command positions or feedback positions for the specified two axes to see if it exceeds the allowed value. | Administration | O |
| MC_SyncMoveVelocity | Cyclic Synchronous Velocity Control | Outputs the value set for the target velocity every primary period to the Servo Drive in Cyclic Synchronous Velocity Mode. | Motion | O |
| MC_Reset | Reset Axis Error | Clears an axis error. | Administration | P |

For details on the axis states due to instruction execution, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

Refer to the compliance list for items that conform to PLCopen technical specifications. The compliance list can be accessed on the PLCopen website.

## Axes Group Commands

This section describes the instructions to perform multi-axis coordinated control for the MC Function Module. The Instruction Type Column gives "Group administration" for non-motion instructions and "Group motion" for motion instructions.

P: Instructions defined in PLCopen technical specifications.
O: Instructions defined for the MC Function Module.

| Instruction | Instruction name | Outline | Classification | |
|---|---|---|---|---|
| MC_GroupEnable | Enable Axes Group | Enables an axes group. | Group administration | P |
| MC_GroupDisable | Disable Axes Group | Disables an axes group. | Group administration | P |
| MC_MoveLinear | Linear Interpolation | Performs linear interpolation. | Group motion | O |
| MC_MoveLinearAbsolute | Absolute Linear Interpolation | Performs linear interpolation for the specified absolute position. | Group motion | P |
| MC_MoveLInearRelative | Relative Linear Interpolation | Performs linear interpolation for the specified relative position. | Group motion | P |
| MC_MoveCircular2D | Circular 2D Interpolation | Performs circular interpolation for two axes. | Group motion | O |
| MC_GroupStop | Group Stop | Decelerates all axes in an interpolated motion to a stop. | Group motion | P |
| MC_GroupImmediateStop | Axes Group Immediate Stop | Immediately stops all axes that are currently in interpolated motion with the method that is specified in the axis parameters. | Group motion | O |
| MC_GroupSetOverride | Set Group Overrides | Changes the blended target velocity during an interpolated motion. | Group administration | P |
| MC_GroupReset | Group Reset | Clears axes group errors and axis errors. | Group administration | P |

For details on the axes group states due to instruction execution, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

Refer to the compliance list for items that conform to PLCopen technical specifications. The compliance list can be accessed on the PLCopen website.

# 2-3 PDO Mapping

You must map the objects that are required for the motion control functions that you will use to process data communications. The PDO map lists all of the objects that are registered in advance.

The default PDO map in the Sysmac Studio is given below.

| | |
|---|---|
| **RxPDO (1704 hex)** | Controlword (6040 hex), Target Position (607A hex), Target Velocity (60FF hex), Target Torque (6071 hex), Modes of Operation (6060 hex), Touch Probe Function (60B8 hex), Max Profile Velocity (607F hex), Positive Torque Limit Value (60E0 hex), and Negative Torque Limit Value (60E1 hex) |
| **TxPDO (1B02 hex)** | Error Code(603F hex), Status Word (6041 hex), Position Actual Value (6064 hex), Torque Actual Value (6077 hex), Modes of Operation Display (6061 hex), Touch Probe Status (60B9 hex), Touch Probe Pos1 Pos Value (60BA hex), Touch Probe Pos2 Pos Value (60BC hex), and Digital Inputs (60FD hex) |

**Additional Information**

If you use an OMRON G-series R88-KN□□□-ECT Servo Drive (version 2.1 or higher), then it is not necessary to change the default PDO map on the Sysmac Studio.

For details on setting the PDO map, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Required Objects

There are objects that are required for Servo axes and an object that is required for encoder axes. If even one of the required objects is not set, a Required Process Data Object Not Set error (error code 3460 hex) occurs.

### Servo Axes

The following objects must be set to use motion control instructions for a Servo axis.

| Input/output | Function | Process data |
|---|---|---|
| Output | Control word | 6040 hex |
| | Target position | 607A hex |
| Input | Status word | 6041 hex |
| | Position actual value | 6064 hex |

### Encoder Axes

The following object must be set to use motion control instructions for an encoder axis.

| Input/output | Function | Process data |
|---|---|---|
| Input | Position actual value | 4010 hex |

# Objects Required for Specific Instructions

There are objects that you must set to use specific instructions. There are settings required for both Servo axes and encoder axes. If an object that is required for an instruction is not set, a Process Data Object Setting Missing error (error code 3461 hex) occurs.

## Servo Axes

There are objects that you must set to use specific instructions for Servo axes. Refer to the following tables and set the required objects. There are no additional object settings required for Servo axis operation for any instructions that are not listed in the following table.

### ● Output Settings

| Instruction | Function | | | | | | |
|---|---|---|---|---|---|---|---|
| | Target velocity (60FF hex) | Target torque (6071 hex) | Modes of operation (6060 hex) | Touch probe function (60B8 hex) | Max. profile velocity (607F hex) | Positive torque limit value (60E0 hex) | Negative torque limit value (60E1 hex) |
| MC_Home | | | | Conditionally required*1 | | | |
| MC_MoveFeed | | | | Conditionally required*2 | | | |
| MC_MoveLink | | | | Conditionally required*3 | | | |
| MC_TorqueControl | | Required | Required | | Conditionally required*4 | | |
| MC_SetTorqueLimit | | | | | | Required | Required |
| MC_TouchProbe | | | | Conditionally required*2 | | | |
| MC_SyncMoveVelocity | Required | | Required | | | | |

*1 Setting is not required for Homing Operation Modes 11, 12, and 14.

*2 Setting is required when *Mode* is set to Drive Mode.

*3 Setting is required when LinkOption (Synchronization Start Condition) is set *to_mcTriggerDetection* and *Mode* is set to Drive Mode.

*4 This setting is checked only when using an OMRON G5-series Servo Drive.

### ● Input Settings

| Instruction | Function | | | | |
|---|---|---|---|---|---|
| | Torque actual value (6077 hex) | Modes of operation display (6061 hex) | Touch probe status (60B9 hex) | Touch probe pos1 pos value (60BA hex) | Touch probe pos2 pos value (60BC hex) |
| MC_Home | | | Conditionally required*1 | Conditionally required*1 | |
| MC_MoveFeed | | | Conditionally required*2 | Conditionally required*3 | Conditionally required*4 |
| MC_MoveLink | | | Conditionally required*5 | Conditionally required*6 | Conditionally required*7 |
| MC_TorqueControl | Required | Required | | | |
| MC_TouchProbe | | | Conditionally required*2 | Conditionally required*3 | Conditionally required*4 |
| MC_SyncMoveVelocity | | Required | | | |

*1 Setting is not required for Homing Operation Modes 11, 12, and 14.
*2 Setting is required when *Mode* is set to Drive Mode.
*3 Setting is required when *Mode* is set to Drive Mode and *LatchID* is set to _mcLatch1 (Latch 1).
*4 Setting is required when *Mode* is set to Drive Mode and *LatchID* is set to _mcLatch2 (Latch 2).
*5 Setting is required when LinkOption (Synchronization Start Condition) is set *to_mcTriggerDetection* and Mode is set to Drive Mode.
*6 Setting is required when *LinkOption* (Synchronization Start Condition) is set to *_mcTriggerDetection*, Mode is set to Drive Mode, and LatchID is set to *_mcLatch1* (Latch 1).
*7 Setting is required when *LinkOption* (Synchronization Start Condition) is set to *_mcTriggerDetection*, Mode is set to Drive Mode, and LatchID is set to *_mcLatch2* (Latch 2).







## Encoder Axes

There are objects that you must set to use specific instructions for encoder axes. Refer to the following tables and set the required objects. There are no additional object settings required for encoder axis operation for any instructions that are not listed in the following table.

### ● Output Settings

| Instruction | Function | |
|---|---|---|
| | Touch probe function (4020 hex) | Software Switch of Encoder's Input Slave (4020 hex) |
| MC_TouchProbe | Conditionally required*1 | Conditionally required*2 |

*1 Setting is required when *Mode* is set to Drive Mode.
*2 Setting is required when an OMRON encoder slave is used and *Mode* is set to Drive Mode.

### ● Input Settings

| Instruction | Function | | | |
|---|---|---|---|---|
| | Touch probe status (4030 hex) | Touch probe pos1 pos value (4012 hex) | Touch probe pos2 pos value (4013 hex) | Status of Encoder's Input Slave (4030 hex) |
| MC_TouchProbe | Conditionally required*1 | Conditionally required*2 | Conditionally required*3 | Conditionally required*4 |

*1 Setting is required when *Mode* is set to Drive Mode.
*2 Setting is required when *Mode* is set to Drive Mode and *LatchID* is set to _mcLatch1 (Latch 1).
*3 Setting is required when *Mode* is set to Drive Mode and *LatchID* is set to _mcLatch2 (Latch 2).
*4 Setting is required when an OMRON encoder slave is used and *Mode* is set to Drive Mode.

# 3

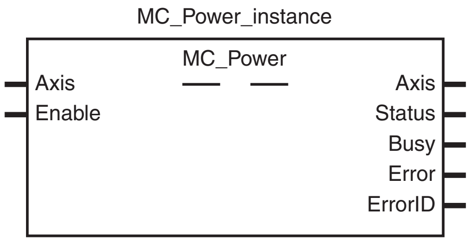
# Axis Command Instructions



This section describes the instructions that are used to perform single-axis control for the MC Function Module.

# MC_Power

The MC_Power instruction makes a Servo Drive ready to operate.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_Power | Power Servo | FB |  | MC_Power_instance (<br>Axis :=*parameter,*<br>Enable :=*parameter,*<br>Status =>*parameter,*<br>Busy =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Enable | Enable | BOOL | TRUE or FALSE | FALSE | The device is ready for operation when *Enable* is TRUE, and not ready when it is FALSE. |

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Status | Servo ON | BOOL | TRUE or FALSE | TRUE when the device is ready for operation. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

#### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Status | When the specified axis becomes ready for operation. | • When operation ready status for the specified axis is cleared.<br>• When *Error* changes to TRUE. |
| Busy | When *Enable* changes to TRUE. | • When *Enable* changes to FALSE.<br>• When *Error* changes to TRUE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

- When *Enable* changes to TRUE, the axis specified by *Axis* is made ready to operate.
  You can control the axis when it is ready to operate.
- When *Enable* changes to FALSE, the ready status is cleared for the axis specified by *Axis*.
  You cannot control the axis after the ready status is cleared because it will not acknowledge operation commands. Also, an error occurs if a motion command is executed for an axis for which the ready status is cleared. You can execute the MC_Power (Power Servo) and MC_Reset (Reset Axis Error) instructions even for axes that are not ready.
- You can use this instruction to disable the operation of axes while they are in motion. In this case, *CommandAborted* will change to TRUE. Output of the operation command will stop and the axis will not longer be ready for operation.
- If home is not defined for a Servomotor with an absolute encoder, compensation is performed using the absolute encoder home offset to define home when the axis is ready to operate.
  For details on the absolute encoder home offset, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

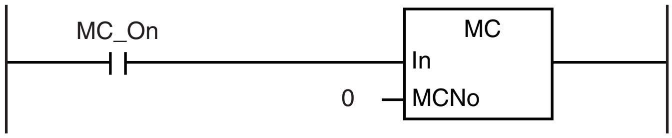
### Precautions for Correct Use

- You can use this instruction for servo axes and virtual servo axes. If the instruction is used for encoder axes or virtual encoder axes, an error will occur.
- **Executing this Instruction for the Master Axis of Synchronized Control**
  When master axis operation is disabled for a vertical axis, the position of the master axis may change rapidly. This may cause the motion of the slave axis to change rapidly. Take suitable measures to prevent the slave axis from moving rapidly, such as applying a brake to the master axis or leaving master axis operation enabled until after synchronized control is completed.

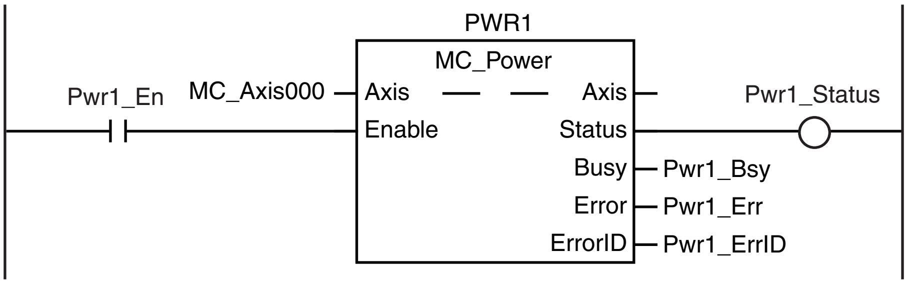
**Additional Information**

We recommend that you use the master control instructions for the MC_Power (Power Servo) instruction as shown below. This ensures that the Servo is turned OFF when the master control is turned OFF.

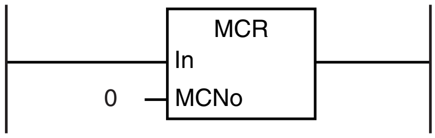
Master control started.




Servo turned ON.




Master control ended.




## ● Relation to CPU Unit Operating Modes

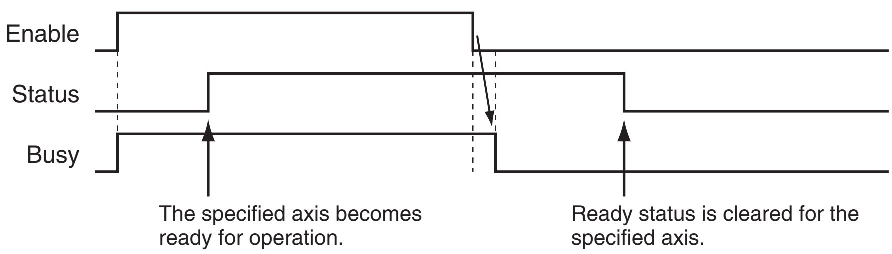
If an axis is placed in ready status during RUN mode, ready status will continue even if the operating mode changes to PROGRAM mode.

## ● Deleting Instruction with Online Editing

If an axis is placed in ready status, ready status will continue even if the instruction is deleted during online editing.

# Timing Charts

- When *Enable* changes to TRUE, *Busy* (Executing) changes to TRUE to indicate that the instruction was acknowledged.
- After the axis becomes ready for operation, *Status* (Servo ON) changes to TRUE.
- When *Enable* changes to FALSE, *Busy* (Executing) changes to FALSE. *Status* (Servo ON) changes to FALSE when ready status is cleared. *Status* (Servo ON) outputs the axis ready status regardless of whether *Enable* is TRUE or FALSE.



**Precautions for Correct Use**

*Status* (Servo ON) will not change to TRUE until *Enable* changes to TRUE and the processing is finished at the axis. Make sure that *Status* (Servo ON) changes to TRUE before moving the axis.

**Precautions for Safe Use**

Write the user program to confirm that EtherCAT communications are established before you execute motion control instructions. This is particularly important when starting axis operation immediately after you turn ON the power supply to the CPU Unit.
Also, include interlocks in the user program that detect errors in EtherCAT communications during operation.

## Re-execution of Motion Control Instructions

You cannot re-execute motion instructions with enable-type inputs.

## Multi-execution of Motion Control Instructions

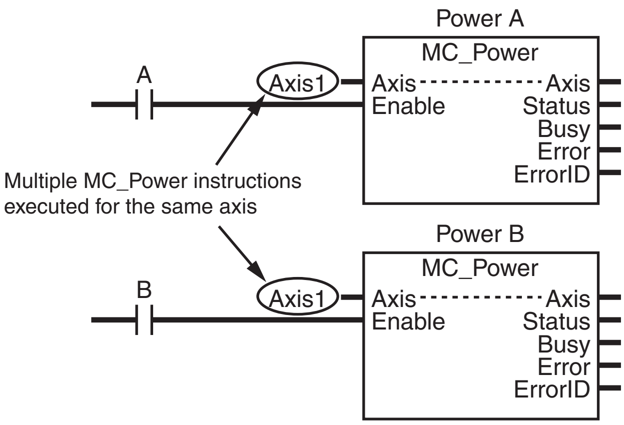
For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

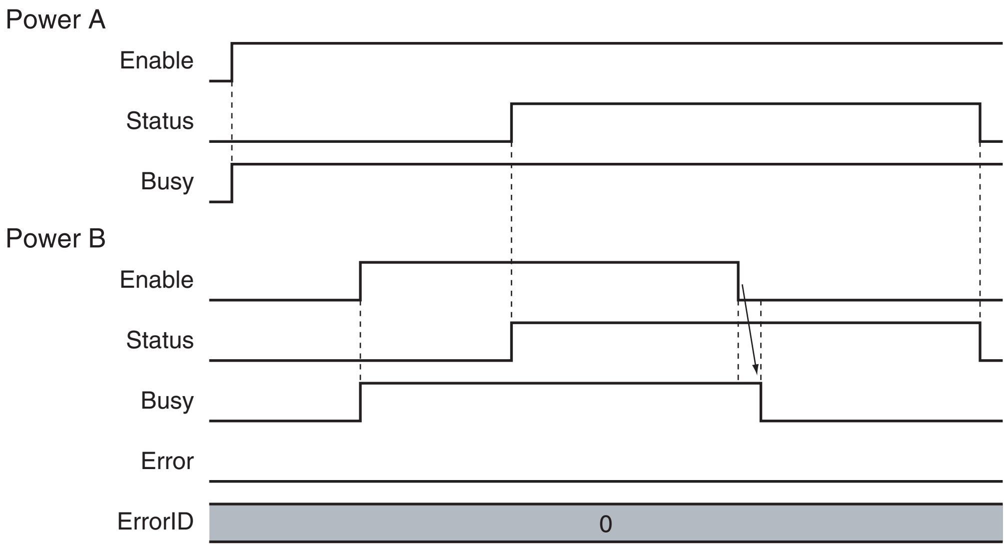
### ● Multi-execution of MC_Power Instructions

**Precautions for Correct Use**

Do not execute the MC_Power (Power Servo) instruction for an axis that is already enabled for another instance of the MC_Power (Power Servo) instruction. Normally, use only one MC_Power (Power Servo) instruction for each axis.

If another MC_Power instruction is executed for the same axis, the last instruction takes priority.



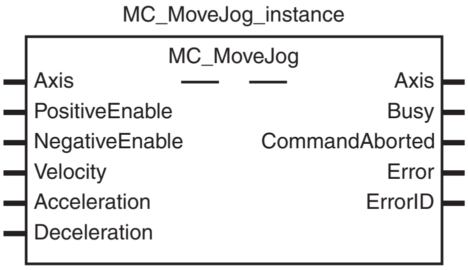



## Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_MoveJog

The MC_MoveJog instruction jogs an axis according to the specified target velocity.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_MoveJog | Jog | FB |  | MC_MoveJog_instance (<br>Axis :=*parameter,*<br>PositiveEnable :=*parameter,*<br>NegativeEnable<br>:=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Busy =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| PositiveEnable | Positive Direction Enable | BOOL | TRUE or FALSE | FALSE | When this variable changes to TRUE, the axis starts moving in the positive direction. When it changes to FALSE, the axis stops moving. |
| NegativeEnable | Negative Direction Enable | BOOL | TRUE or FALSE | FALSE | When this variable changes to TRUE, the axis starts moving in the negative direction. When it changes to FALSE, the axis stops moving. |
| Velocity | Target Velocity | LREAL | Non-negative number | 0 | Specify the target velocity. The unit is command units/s.* |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/$s^2$.* |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.* |

* Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Busy | When *PositiveEnable* or *NegativeEnable* changes to TRUE. | • When the axis stops.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *PositiveEnable* changes to FALSE.<br>• When *NegativeEnable* changes to FALSE.<br>• After one period when *PositiveEnable* and *NegativeEnable* are FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

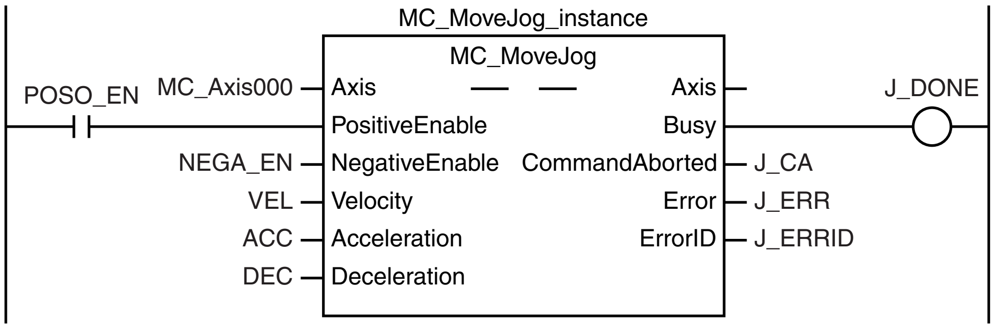
* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

- The MC_MoveJog instruction performs jogging according to the specified *Velocity* (Target Velocity).
- To jog in the positive direction, change *PositiveEnable* (Positive Direction Enable) to TRUE. To jog in the negative direction, change *NegativeEnable* (Negative Direction Enable) to TRUE.
- If *PositiveEnable* (Positive Direction Enable) and *NegativeEnable* (Negative Direction Enable) are changed to TRUE at the same time, *PositiveEnable* (Positive Direction Enable) takes priority. As a result, the axis will jog in the positive direction.
- If the command velocity of the MC_MoveJog (Jog) instruction exceeds the maximum jog velocity that is set in the axis parameters, the maximum jog velocity is used.

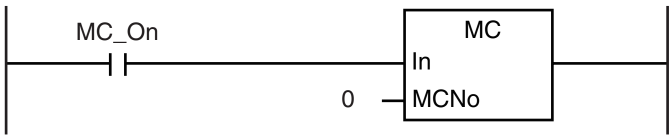
**Precautions for Correct Use**

- When creating a ladder diagram, you must connect the *PositiveEnable* (Positive Direction Enable) input variable to the left bus bar and specify a variable for the *NegativeEnable* (Negative Direction Enable) input variable as shown below.

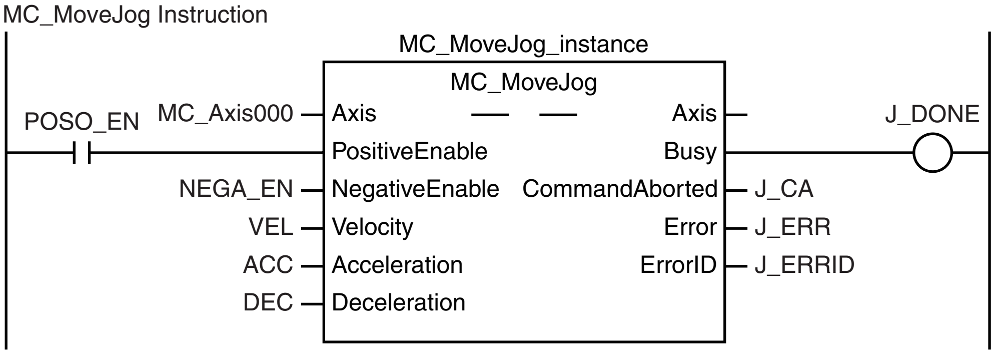



To use the master control instructions (MC and MCR) for the MC_MoveJog (Jog) instruction, do not program the instructions as shown below. If you do, master control is applied only to *PositiveEnable* (Positive Direction Enabled), i.e., it is not applied to *NegativeEnable* (Negative Direction Enabled).

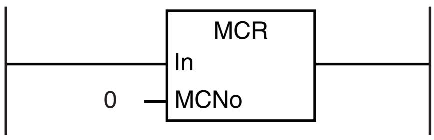
Master control started.




MC_MoveJog Instruction

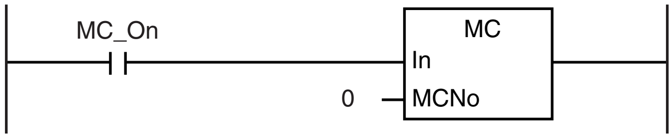



Master control ended.

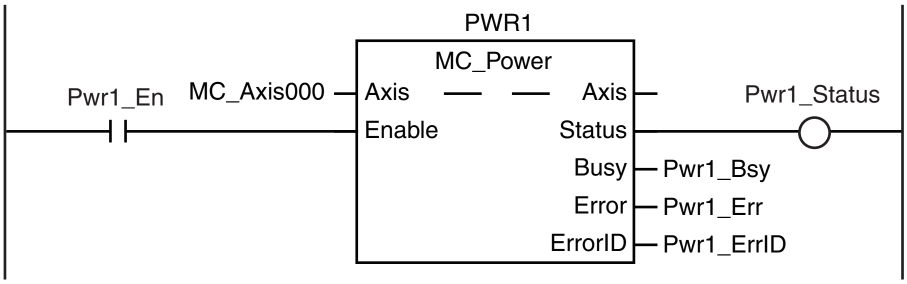



Always use the master control instructions for the MC_Power instruction.

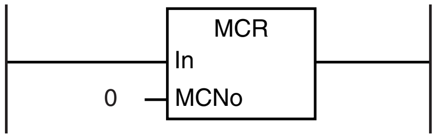
Master control started.



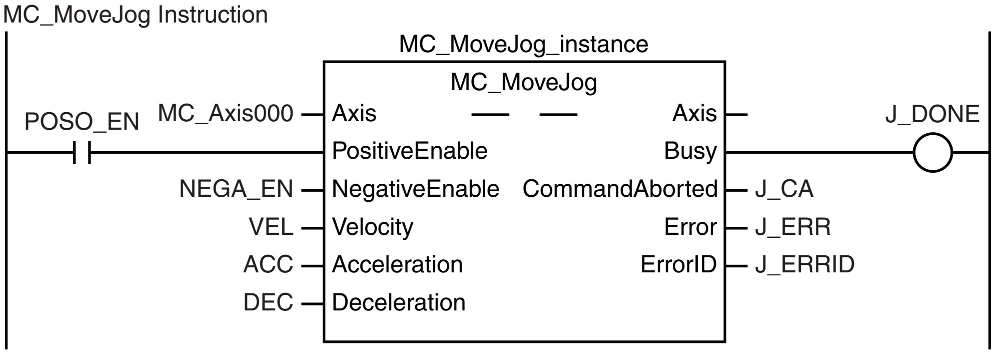

Servo turned ON.




Master control ended.

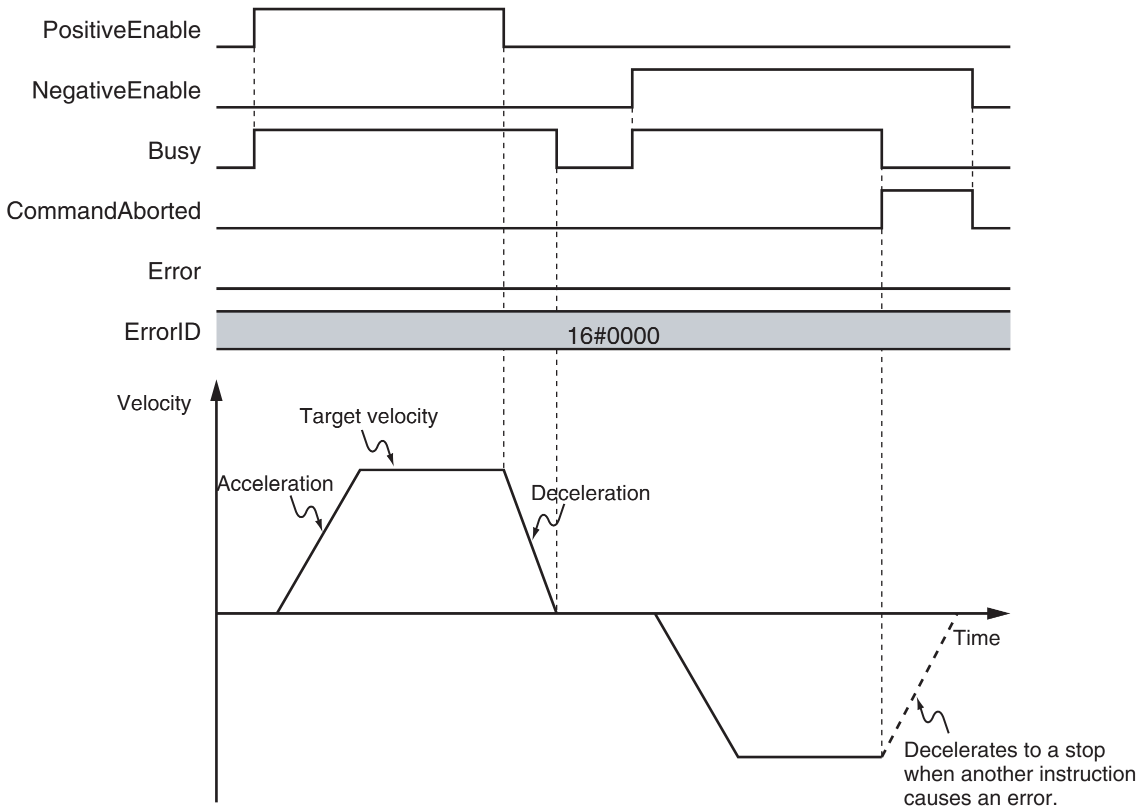



MC_MoveJog Instruction



## Timing Charts

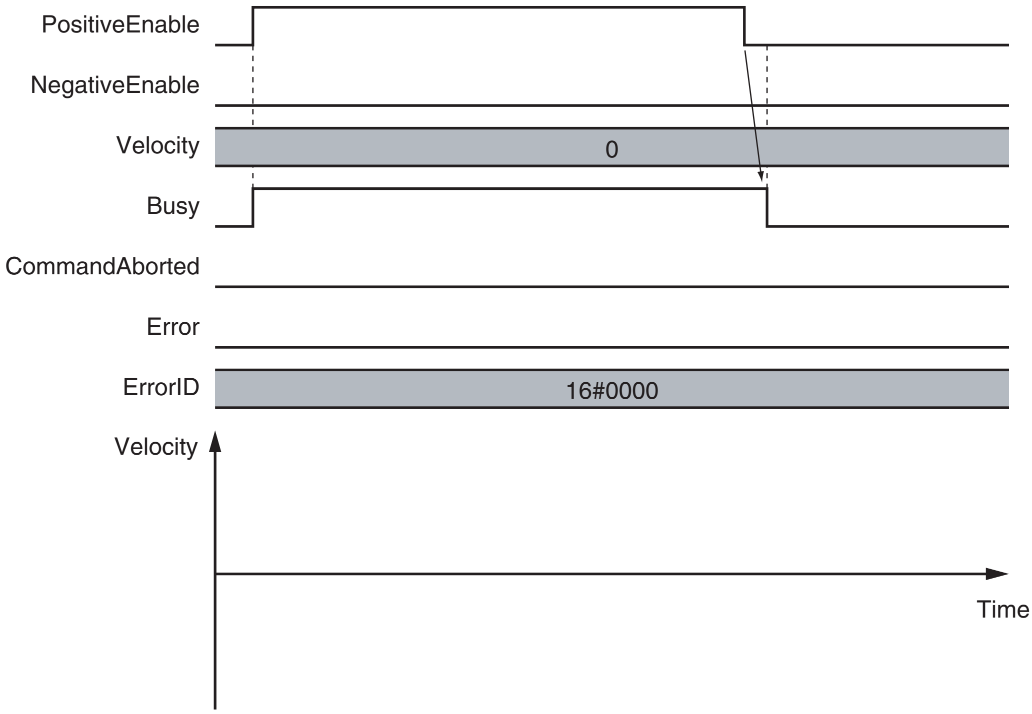
- *Busy* (Executing) changes to TRUE as soon as *PositiveEnable* (Positive Direction Enable) or *NegativeEnable* (Negative Direction Enable) changes to TRUE.
- The axis starts deceleration as soon as *PositiveEnable* (Positive Direction Enable) or *NegativeEnable* (Negative Direction Enable) changes to FALSE. *Busy* (Executing) changes to FALSE when the axis stops completely.
- If another instruction aborts this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing) changes to FALSE.




You can specify the *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate) as input variables. Input variables *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate) are updated in the motion only when *PositiveEnable* (Positive Direction Enable) or *NegativeEnable* (Negative Direction Enable) changes to TRUE. Therefore, the axis velocity will not change even if *Velocity* (Target Velocity) changes while *PositiveEnable* (Positive Direction Enable) or *NegativeEnable* (Negative Direction Enable) remains TRUE.

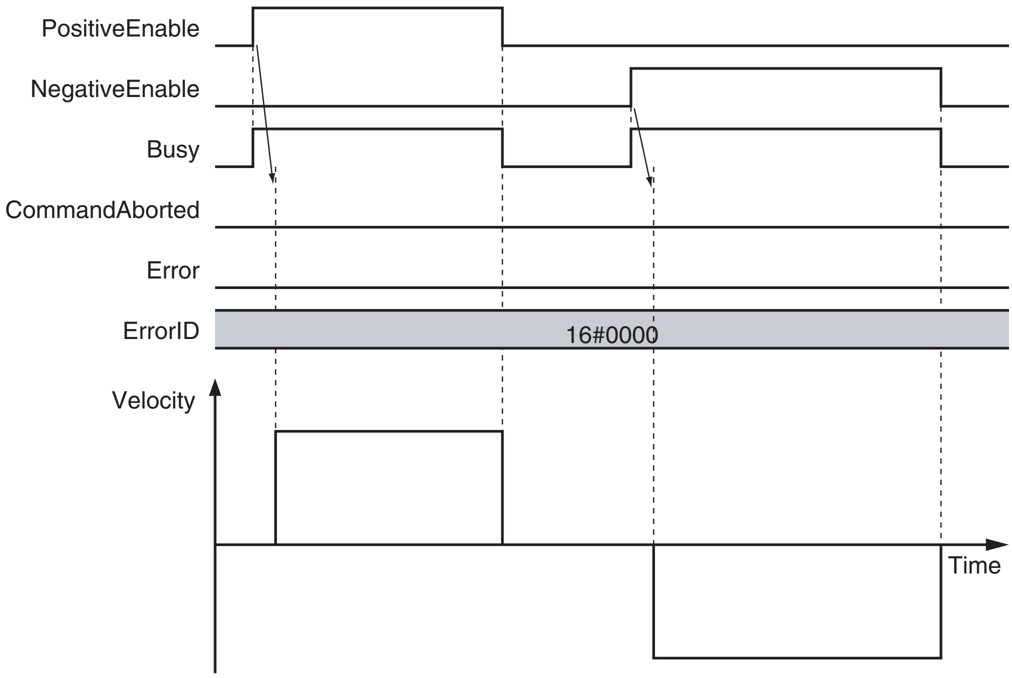
## ● Timing Chart When Target Velocity Is 0

When the *Velocity* (Target Velocity) is 0 and you start jogging the axis, the axis will enter continuous operation without motion.
The following timing chart shows an example when the *Velocity* (Target Velocity) is 0 and you start jogging the axis.




## ● Timing Chart When Acceleration/Deceleration Rate Is 0

When the *Acceleration* (Acceleration Rate) or *Deceleration* (Deceleration Rate) is 0 and you start jogging the axis, the axis will reach the target velocity without accelerating or decelerating.
The timing chart below shows an example when the *Acceleration* (Acceleration Rate) and *Deceleration* (Deceleration Rate) are 0.

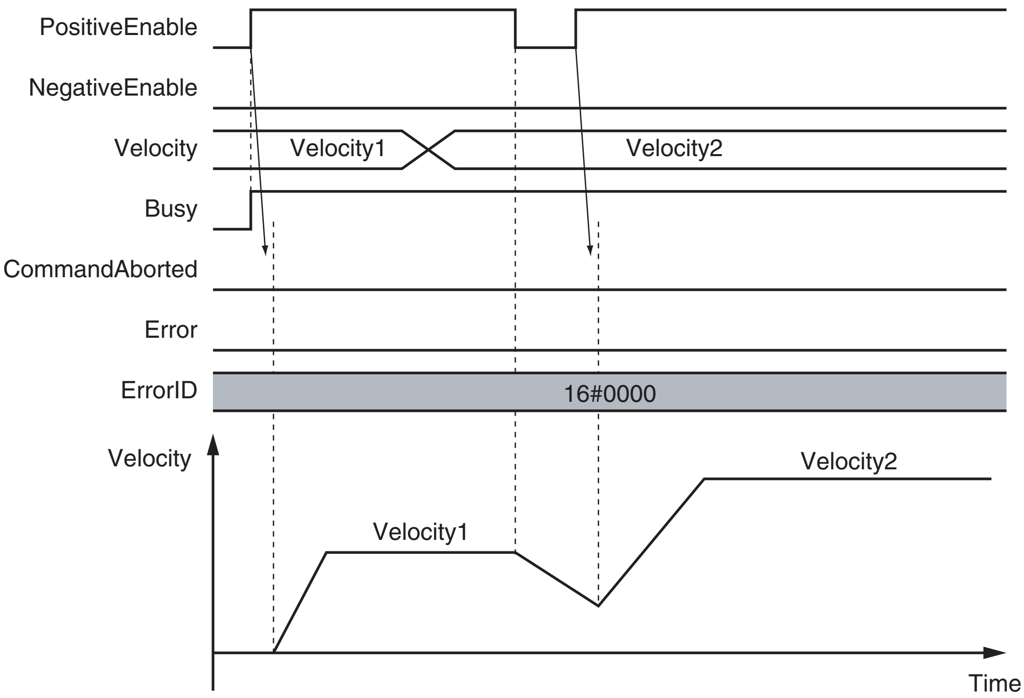


## Re-execution of Motion Control Instructions

### Restarting with *Enable* in the Same Direction

If you change *PositiveEnable* (Positive Direction Enable) or *NegativeEnable* (Negative Direction Enable) to TRUE when it is FALSE and the axis is decelerating, the axis will begin to accelerate towards the target velocity. If you change the *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), or *Deceleration* (Deceleration Rate) at this time, the new value of the input parameter is used in operation.
The axis is not stopped, and *Busy* (Executing) does not change to FALSE.

The following example shows operation when *PositiveEnable* (Positive Direction Enable) changes to TRUE during deceleration.



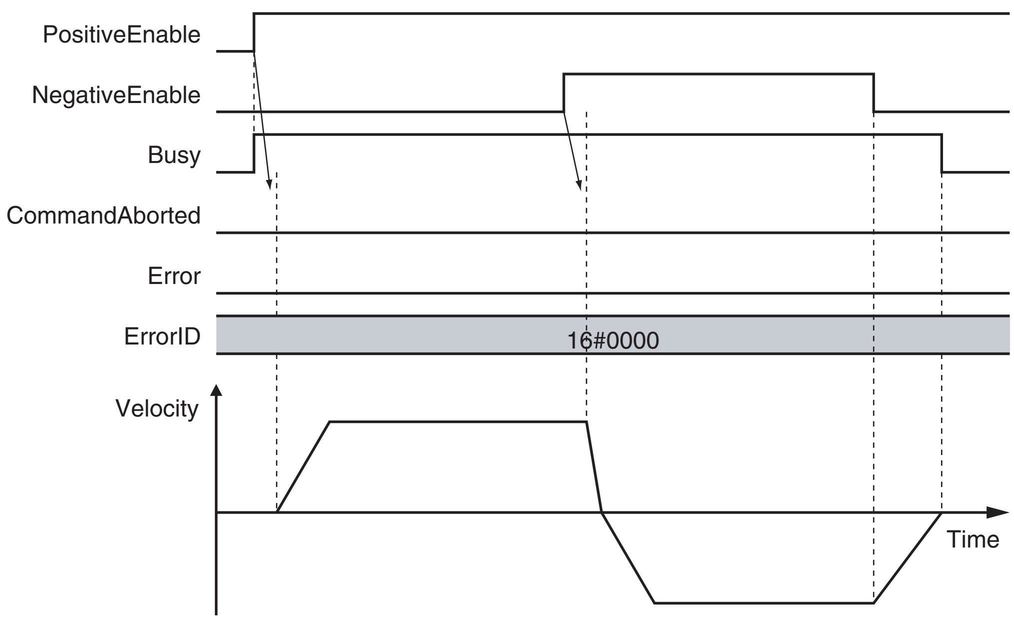

### Restarting with *Enable* in the Opposite Direction

If you change *NegativeEnable* (Negative Direction Enable) to TRUE when *PositiveEnable* (Positive Direction Enable) is TRUE and the axis is jogging in the positive direction, the axis will reverse its direction and start jogging in the negative direction. When this happens, you can jog the axis with the input variables for when *NegativeEnable* (Negative Direction Enable) changes to TRUE. The input variables are *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate).

The deceleration rate before the axis direction is reversed and the acceleration rate after it is reversed follow the input variables for when *NegativeEnable* (Negative Direction Enabled) changes to TRUE, regardless of the Operation at Reversing axis parameter. When *NegativeEnable* (Negative Direction Enable) is TRUE and the axis is jogging in the negative direction, the same operation occurs when *PositiveEnable* (Positive Direction Enable) changes to TRUE.

If *NegativeEnable* (Negative Direction Enable) changes to TRUE while *PositiveEnable* (Positive Direction Enable) is TRUE, the axis starts jogging in the negative direction. In this case, the axis will not jog in the positive direction even if *NegativeEnable* (Negative Direction Enable) changes to FALSE. To jog the axis in the positive direction, change *PositiveEnable* (Positive Direction Enable) to FALSE, and then back to TRUE again. The same operation applies to the opposite case.

The following example shows an operation example when *NegativeEnable* (Negative Direction Enabled) changes to TRUE after *PositiveEnable* (Positive Direction Enabled) changes to TRUE.



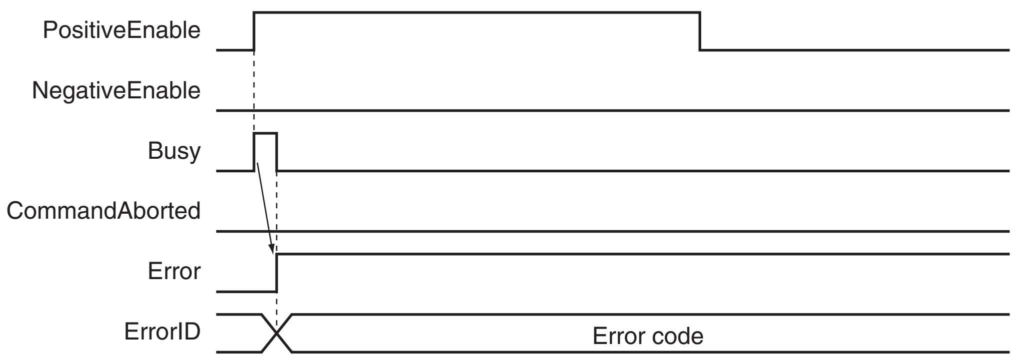

## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axes will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

### ● Timing Chart When Error Occurs

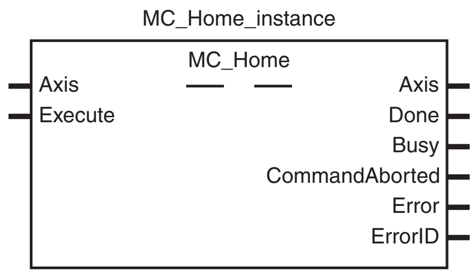



### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_Home

The MC_Home instruction operates the motor to determine home. It uses the limit signals, home proximity signal, and home signal.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_Home | Home | FB |  | MC_Home_instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the instruction is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

- Homing starts when *Execute* changes to TRUE for the axis specified in *Axis*.
- Set the parameters used by the home instruction in the axis parameters.
- There are 10 Homing Operation Modes for the home instruction.
  Set the desired method in axis parameter Homing Method in the Sysmac Studio.

**Precautions for Correct Use**

Refer to *Precautions for Master and Auxiliary Axes in Synchronized Control* on page 1-7 for precautions on the master axis.

### ● Mapping Data Objects

To use the MC_Home instruction, map the following object data in the Detailed Settings Area of the Axis Basic Settings Display of the Sysmac Studio. However, setting is not required for Homing Operation Modes 11, 12, and 14.

- Touch probe function (60B8 hex)
- Touch probe status (60B9 hex)
- Touch probe pos1 pos value (60BA hex)

If even one of the required objects is not set, a Process Data Object Setting Missing error (error code 3461 hex) occurs.

For details on mapping data objects, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Settings for OMRON G5-series Servo Drives

Set the input signals, such as the home proximity signal, that are used by the home instruction in the OMRON G5-series Servo Drive.

Refer to the information on connecting to the Servo Drive in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) and information on sequence I/O signals in the *G5-series AC Servomotors/Servo Drives with Built-in EtherCAT Communications User's Manual* (Cat. No. I576) for details on how to set the input signals.

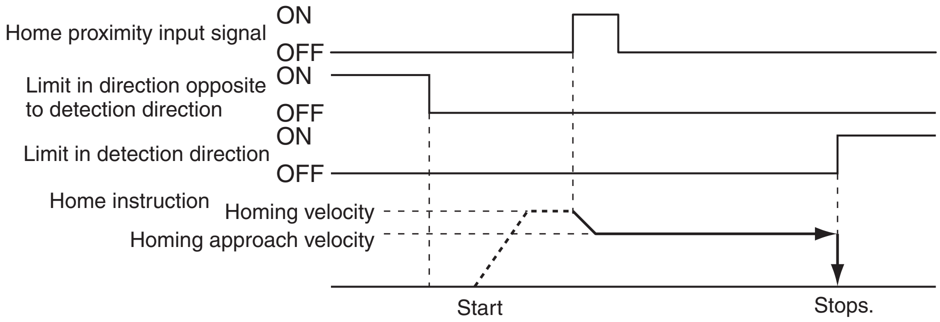
| Name | Meaning | Data type | Valid range | Unit | Default | Description |
|---|---|---|---|---|---|---|
| ProximitySignal Reference | Proximity Signal Selection | Input_REF | --- | --- | --- | Select the input to use for the home proximity input signal. |
| HomeSignal Reference | Home Signal Selection | Input_REF | --- | --- | --- | Select the input to use for the home input signal. |

## Homing Operation Modes

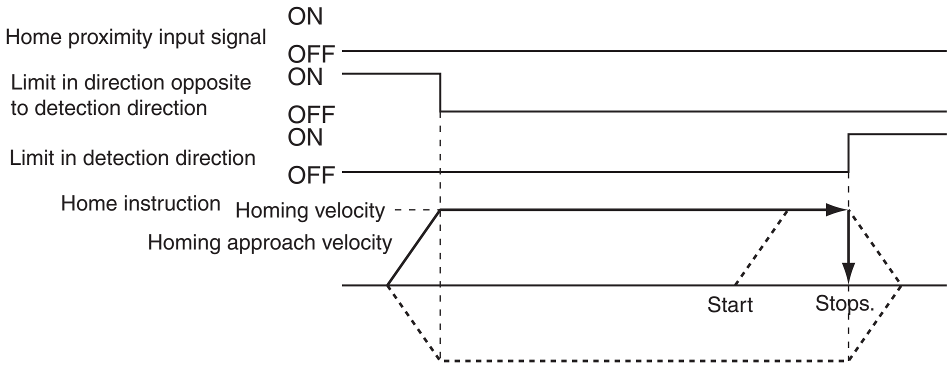
You can select any of the ten operations to define home. For details on the operations, refer to *Function* on page 3-17.

## Operation at Positive Limit Input and at Negative Limit Input

- Select the operation when the axis reaches a limit input in the operating direction during homing: reverse the axis and continue with homing, or do not reverse the axis, create an error, and stop the axis. To reverse the axis, also select the stopping method.
- An error occurs and the axis stops if the axis is set to reverse direction, and the limit signal in the home input detection direction turns ON when traveling at the homing approach velocity. However, if the operating mode is 13 (no home proximity input/holding home input), which does not use proximity signals, no error will occur and the axis will not stop.




- An error occurs and the axis stops if the axis is set to reverse direction or the limit input operation in both directions and home cannot be detected after moving from the limit input opposite to the home input detection direction to the other limit input.

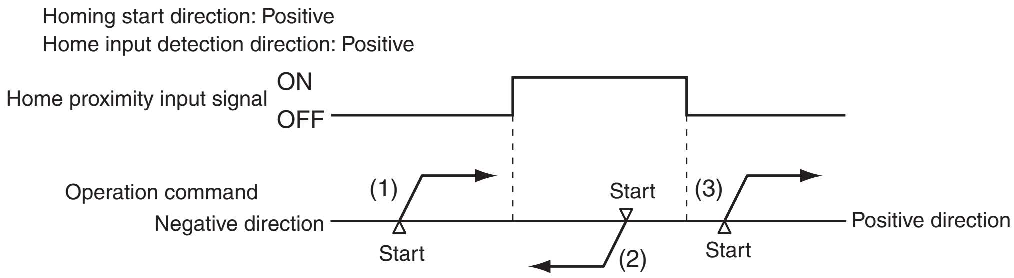


## Homing Start Direction

Select the direction in which the axis starts moving when homing is started. If homing starts while the home proximity signal is ON in a Homing Operation Mode that includes reversal operation, the axis starts motion in the direction opposite to the home input detection direction (regardless of the setting of the homing start direction).

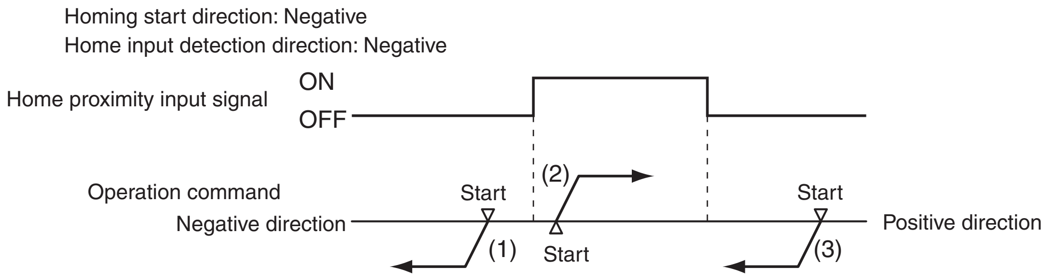
There are four Homing Operation Modes that include reversal operation for a reverse turn. These are listed below.

- 0: Proximity reverse turn/home proximity input OFF
- 1: Proximity reverse turn/home proximity input ON
- 9: Proximity reverse turn/home input mask distance
- 12: Proximity reverse turn/holding time




(1), (3): The home proximity signal is OFF, so the axis starts moving in the homing start direction.

(2): The home proximity signal is ON, so the axis starts moving in the direction opposite to the home input detection direction.




(1), (3): The home proximity signal is OFF, so the axis starts moving in the homing start direction.

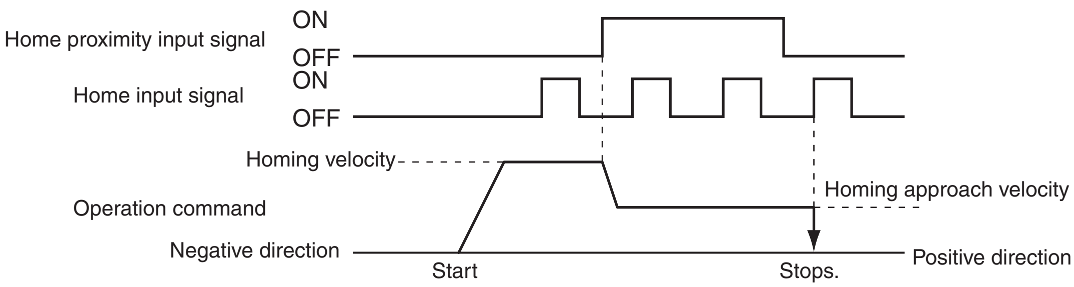
(2): The home proximity signal is ON, so the axis starts moving in the direction opposite to the home input detection direction.

## Home Input Detection Direction

Select the direction when home input is detected.

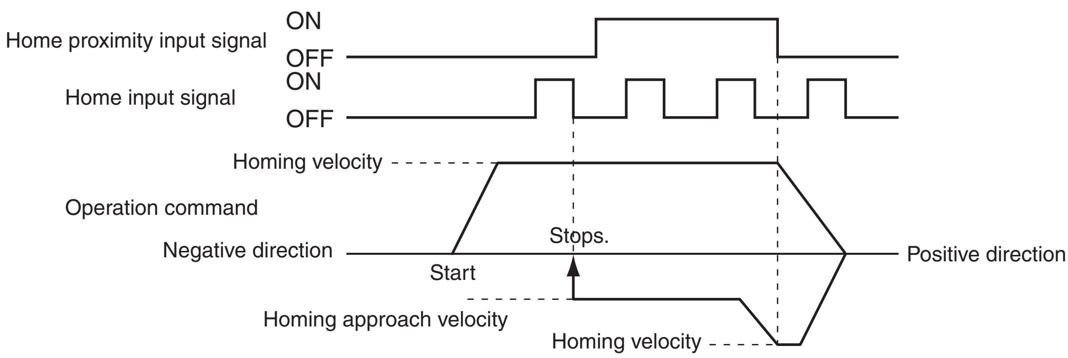
The following timing chart shows the operation when the home input detection direction is FALSE (positive direction).

Operation Example: Proximity Reverse Turn/Home Proximity Input OFF




The following timing chart shows the operation when the home input detection direction is TRUE (negative direction).

Operation Example: Proximity Reverse Turn/Home Proximity Input OFF




## Home Input Mask Distance

Set the feed distance when you set the Homing Operation Mode to 9 (proximity reverse turn/home input mask distance). For details on operation, refer to *Function* on page 3-17.

## Homing Compensation Value

Set the homing compensation value that is applied after the home is detected. Set the travel velocity in the Homing Compensation Velocity. For details on operation, refer to *Function* on page 3-17.

## Home Position Offset

The Home Position Offset is used to preset the actual position after homing is completed. The current value is the value set for the Home Position Offset.

## Homing Velocity

This is the high velocity during homing.

## Homing Approach Velocity

This is the proximity velocity during homing.

## Homing Compensation Velocity

This is the velocity when you set a homing compensation value. For details on operation, refer to *Function* on page 3-17.

## Instruction Details

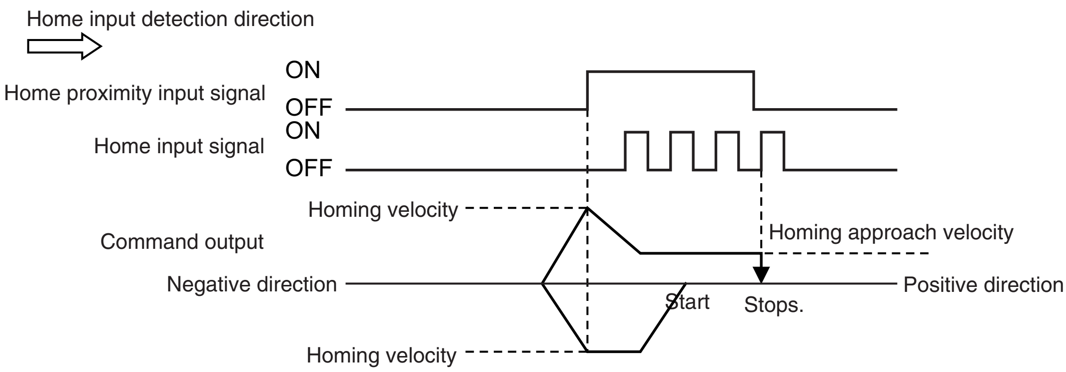
### ● Homing Definition Operation

This section describes the 10 Homing Operation Modes.

#### 0: Proximity Reverse Turn/Home Proximity Input OFF Operation

**1** The axis starts at the homing velocity. When the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.

**2** After the axis reaches the homing approach velocity, the axis stops at the first home input signal after the home proximity input signal turns OFF. This defines home.

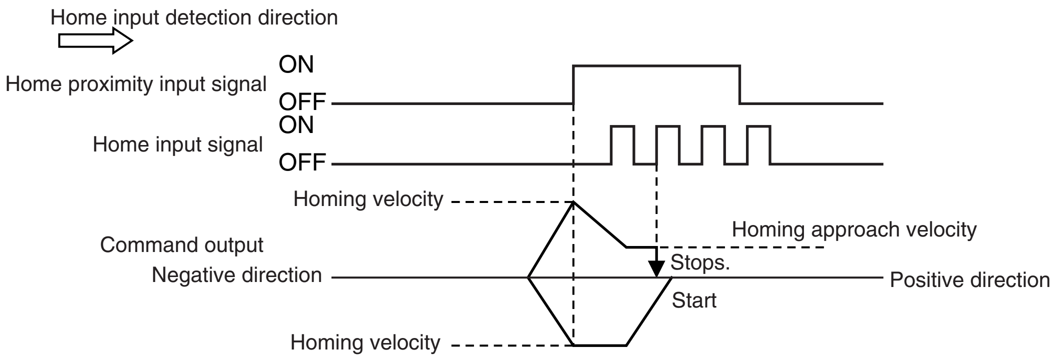



- If you start homing while the home proximity input signal is ON, the operation will start at the homing velocity in the direction opposite to the home input detection direction. After the home proximity input turns OFF, the homing operation will start at the homing velocity in the home input detection direction.
- Homing is started and home is defined when the home input signal turns ON after the home proximity input signal turns ON and OFF while the velocity is below the homing approach velocity.

#### 1: Proximity Reverse Turn/Home Proximity Input ON Operation

**1** The axis starts at the homing velocity. When the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.

**2** After the axis reaches the homing approach velocity, the axis stops at the first home input signal. This defines home.
After the home proximity input signal turns ON, its status does not affect operation.

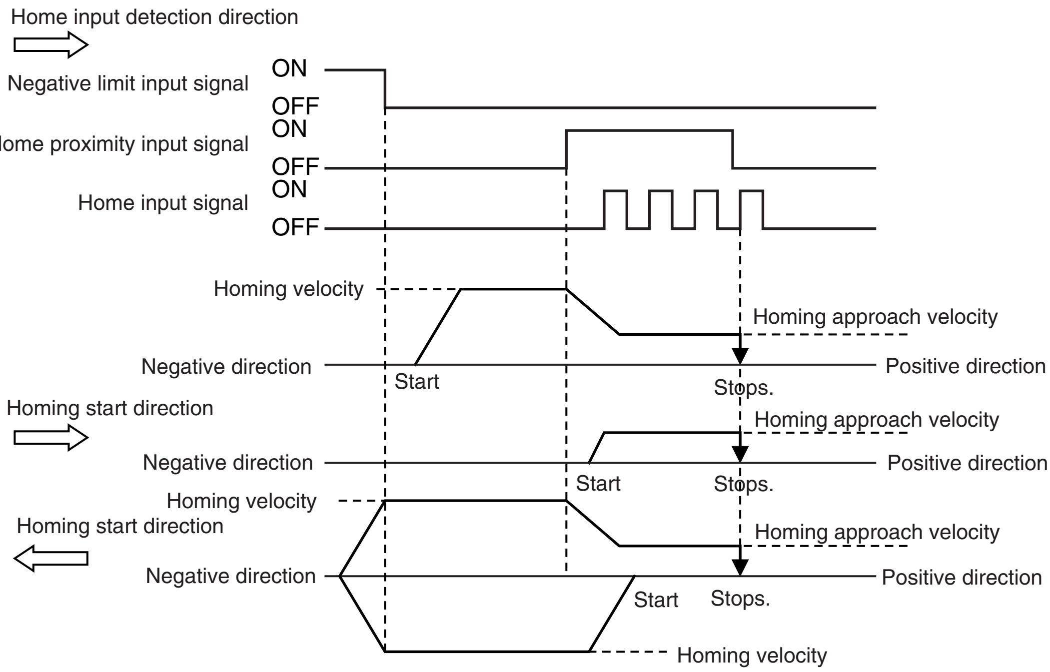



- If you start homing while the home proximity input signal is ON, the operation will start at the homing velocity in the direction opposite to the home input detection direction. After the home proximity input turns OFF, the homing operation will start at the homing velocity in the home input detection direction.
- Homing is started and home is defined when the home input signal turns ON after the home proximity input signal turns ON while the velocity is below the homing approach velocity.

### 4: Operation for Home Proximity Input OFF

**1** When the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.

**2** After the axis reaches the homing approach velocity, the axis stops at the first home input signal after the home proximity input signal turns OFF. This defines home.




- If you start homing while the home proximity input signal is ON, the axis performs the following operation depending on the setting of the homing start direction.

#### Homing Start Direction Same as Home Input Detection Direction

The axis does not perform a reverse turn operation and homing starts in the home input detection direction at the homing approach velocity.

#### Homing Start Direction Different from Home Input Detection Direction

Operation starts in the homing start direction at the homing velocity, regardless of the status of the home proximity input signal. If you set the operation at the limit input in the homing start direction to reverse direction, the axis reverses direction when limit input is detected and performs a homing operation in the home input detection direction.

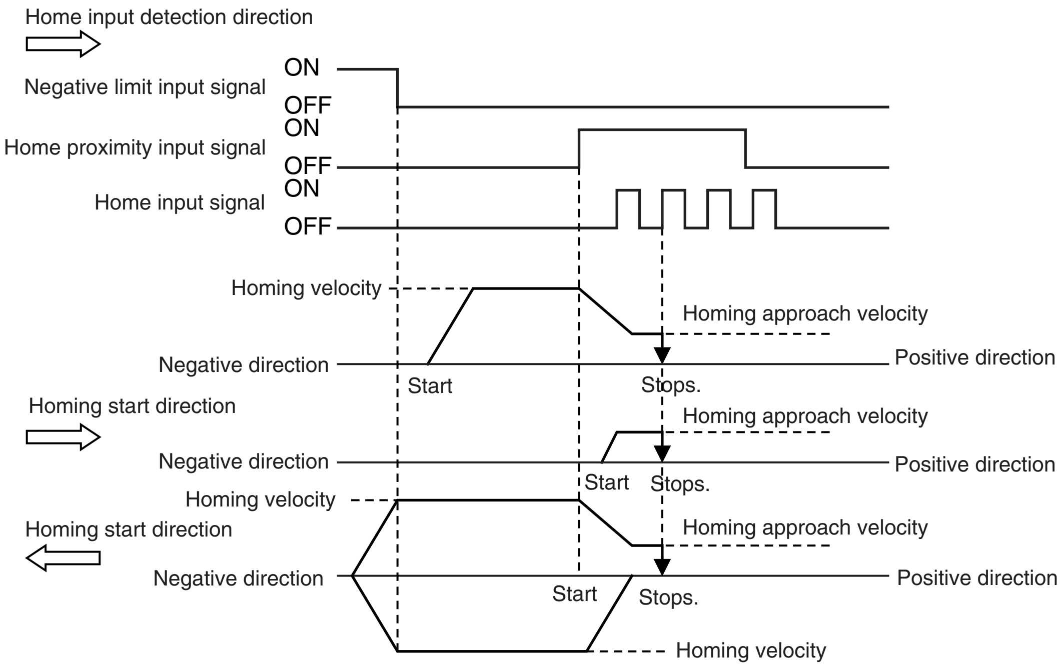
- Homing is started and home is defined when the home input signal turns ON after the home proximity input signal turns ON and OFF while the velocity is below the homing approach velocity.

### 5: Home Proximity Input ON Operation

**1** When the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.

**2** After the axis reaches the homing approach velocity, the axis stops at the first home input signal. This defines home.

After the home proximity input signal turns ON, its status does not affect operation.




- If you start homing while the home proximity input signal is ON, the axis performs the following operation depending on the setting of the homing start direction.

#### Homing Start Direction Same as Home Input Detection Direction

The axis does not perform a reverse turn operation and homing starts in the home input detection direction at the homing approach velocity.

#### Homing Start Direction Different from Home Input Detection Direction

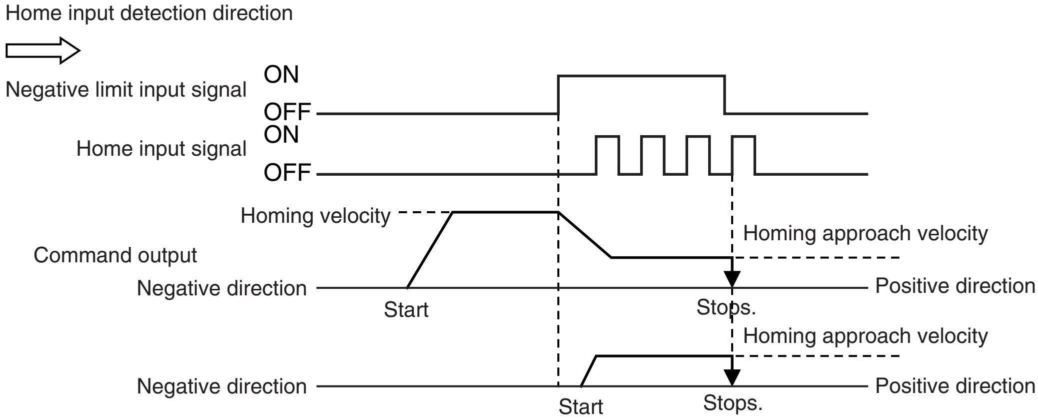
Operation starts in the homing start direction at the homing velocity, regardless of the status of the home proximity input signal. If you set the operation at the limit input in the homing start direction to reverse direction, the axis reverses direction when limit input is detected and performs a homing operation in the home input detection direction.

- Homing is started and home is defined when the home input signal turns ON after the home proximity input signal turns ON while the velocity is below the homing approach velocity.

### 8: Operation for Limit input OFF

**1** The axis starts at the homing velocity. When the limit signal in the direction opposite to the home input detection direction turns ON, the axis starts decelerating to the homing approach velocity.

**2** After the axis reaches the homing approach velocity, the axis stops at the first home input signal after the limit signal turns OFF. This defines home.



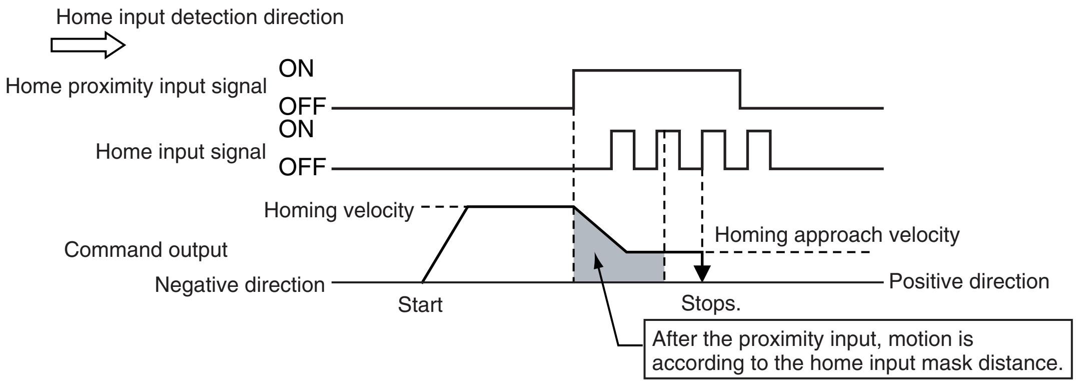

- If you perform homing while the limit input on the opposite side of the home input detection direction is ON, the homing operation starts at the home approach velocity in the home input detection direction.
- Homing is started and home is defined when the home input signal turns ON after the limit signal in the direction opposite to the home input detection direction turns ON and OFF again while the velocity is below the homing approach velocity.

### 9: Proximity Reverse Turn/Home Input Mask Distance Operation

**1** The axis starts at the homing velocity. When the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.

**2** The axis moves by the home input mask distance after the home proximity input signal turns ON and stops at the first home input signal. This defines home.

After the home proximity input signal turns ON, its status does not affect operation. If the specified travel distance is too short and travel would be completed before the axis decelerates to the homing approach velocity, an Invalid Home Input Mask Distance error (error code: 742B hex) occurs when you start homing.




- If you start homing while the home proximity input signal is ON, the operation will start at the homing velocity in the direction opposite to the home input detection direction. After the home proximity input signal turns OFF, the homing operation will start at the homing approach velocity in the home input detection direction.

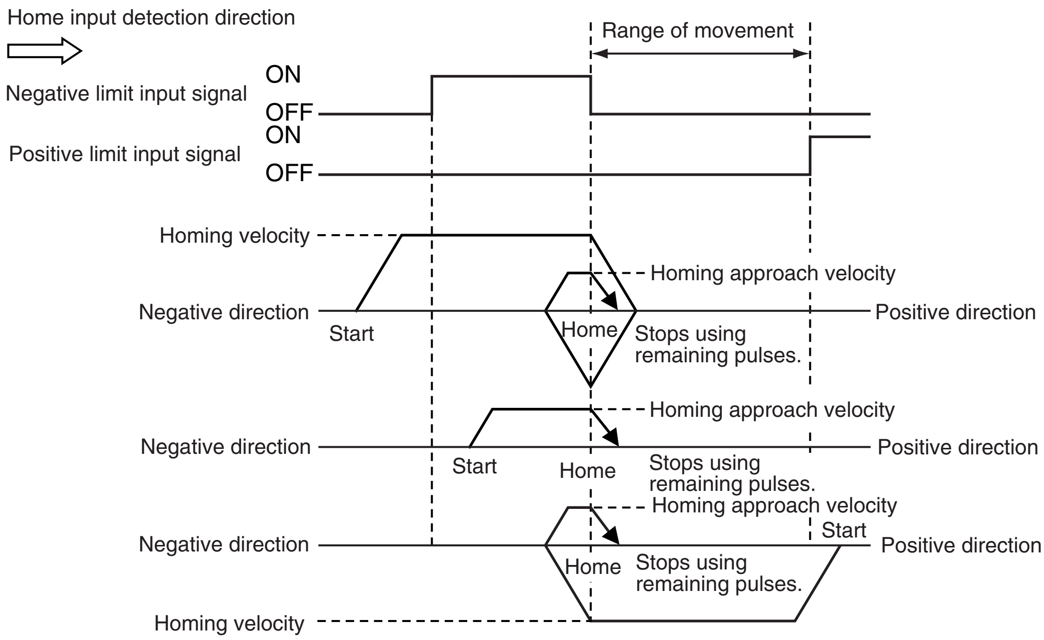
- Homing is started and home is defined when the home input signal turns ON after the axis travels the home input mask distance after the home proximity input signal turns ON while the velocity is below the homing approach velocity.

### 11: Operation for Limit Inputs Only

**1** The axis starts at the homing velocity. When the limit signal in the direction opposite to the home input detection direction turns OFF, the axis decelerates to a stop.

**2** After the axis stops, the axis moves in the other direction at the homing velocity and decelerates to a stop when the limit signal turns ON.

**3** After the axis stops, the axis moves in the other direction at the homing approach velocity. The position where the limit signal turns OFF is defined as home and an immediate stop is performed (i.e., a stop using remaining pulses). The axis does not return to the home position.



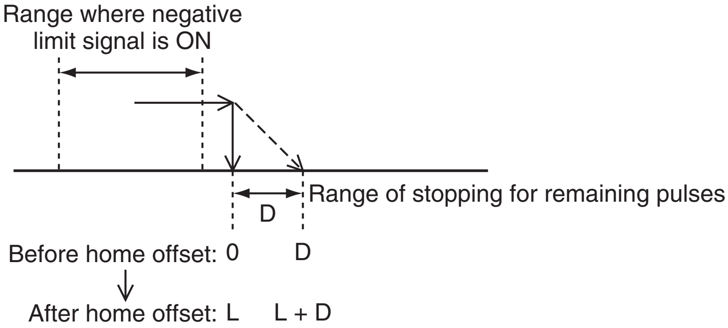

- If you use only the limit signals to perform homing, the point at which the limit signal turns OFF during operation in the home input detection direction is set as the home detection position.
- If you start homing while the limit signal in the home input detection direction is ON, the operation will start at the homing velocity in the direction opposite to the home input detection direction. When the limit signal in the direction opposite to the home input detection direction turns ON, the axis decelerates to a stop.
- Even if the limit signal turns OFF before the axis decelerates to a stop after the limit signal is detected, home is not defined and the axis continues to decelerate. In this case, no error will occur.
- Home is defined if the limit signal turns OFF before the homing approach velocity is reached after the axis reverses or after starting while the limit signal is input.

#### Differences between Homing Mode 11 and Other Homing Modes

For Homing Mode 11, the MC Function Module detects when the limit signal turns OFF to define home. It differs from a Homing Mode that uses the home input as an external latch signal in the following ways.
Make sure that you understand these differences before you use Homing Mode 11.

- The MC Function Module detects the limit input without an external latch function, such as one provided by a Servo Drive. The processing for the MC Function Module is performed in the primary period. Therefore, the precision of the home definition will depend on the homing approach velocity and the control period of the primary periodic task.

- The axis does not return to the home position. If the homing compensation value is 0, processing for homing will end with the axis at a different position (i.e., not at home).
- Homing compensation is not performed if the homing compensation value is set to 0. If the homing compensation value is 0, processing for homing will end with the axis at a different position (i.e., not at home), as explained above. If the homing compensation value is not 0, then homing compensation is performed with the homing compensation value as a relative position from home in the same way as for other modes.
- The home offset is used to change the position of home. If the stop position is offset from home by distance D, as shown in the following figure, the position after the completion of processing for homing will be L + D if the home offset is L and the homing compensation value is 0.



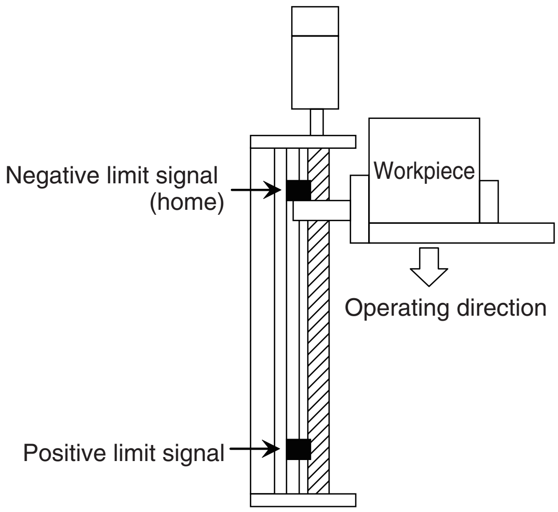



**Precautions for Correct Use**

- After the OFF limit signal is detected, the limit signal in the opposite direction from the home input detection direction is ignored while stopping for the remaining pulses until homing compensation is started.
- If the homing compensation value is 0 and the limit input signal in the home input detection direction turns ON immediately after home is defined, set a homing compensation value to return past the limit signal input position.

**Application Example for Homing Mode 11**

If, as shown below, there is not sufficient space to install both a negative limit signal and home signal, you can use the negative limit signal to perform the functions of both the limit signal and home signal.



### 12: Proximity Reverse Turn/Holding Time Operation

**1** The axis starts at the homing velocity. When the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.

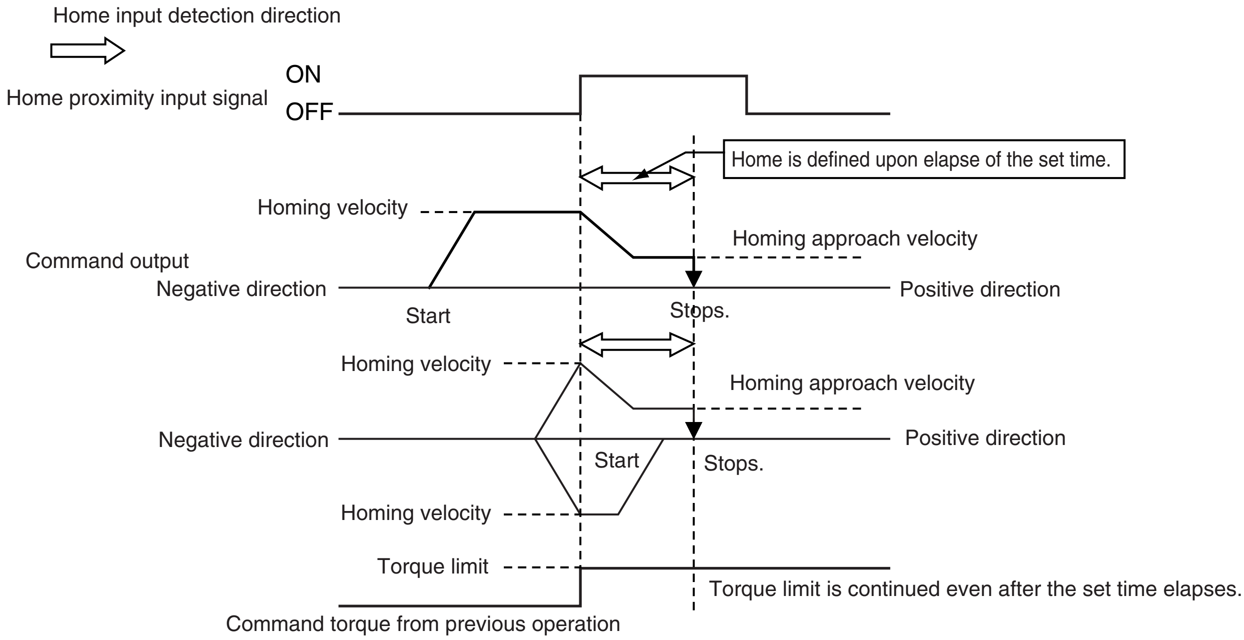
Decelerating the axis and monitoring time are started at the same time. The torque limit at the start of holding differs between OMRON G5-series Servo Drives and other Servo Drives as shown below.

| | |
|---|---|
| G5 Servo Drives: | Automatically applies torque limits. |
| Other Servo Drives: | Apply torque limits according to a setting. |

Refer to *Holding Operation for OMRON G5-series Servo Drives* on page 3-31 for information on the holding operation.

**2** Home is defined when the set time elapses.

After the home proximity input signal turns ON, its status does not affect operation.




- If you start homing while the home proximity input signal is ON, the operation will start at the homing velocity in the direction opposite to the home input detection direction. After the home proximity input signal turns OFF, the homing operation will start at the homing approach velocity in the home input detection direction.
- Releasing the torque limit also differs between OMRON G5-series Servo Drives and other Servo Drives.

| | |
|---|---|
| G5 Servo Drives: | Automatically released when the axis moves in the direction opposite to home definition direction for the first time after homing. |
| Other Servo Drives: | If a torque limit is used, release the torque limit when the axis moves in the direction opposite to home definition direction for the first time after homing. Use the ECAT_CoEWriteSdo command to change the torque limit. |

- An error will not occur and home is defined even if the holding time elapses after the home proximity input signal is detected and before velocity reaches the homing approach velocity.
- Home is also defined if the holding time elapses after the home proximity input signal turns ON before the homing approach velocity is reached.

### 13: No Home Proximity Input/Holding Home Input Operation

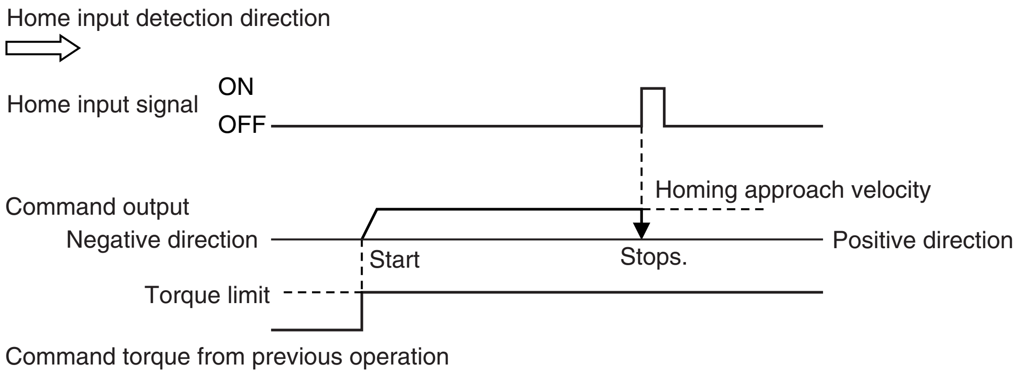
**1** The axis starts at the homing approach velocity.

The torque limit at the start of holding differs between OMRON G5-series Servo Drives and other Servo Drives as shown below.

| | |
|---|---|
| G5 Servo Drives: | Automatically applies torque limits. |
| Other Servo Drives: | A torque limit imposed as required. |

Refer to *Holding Operation for OMRON G5-series Servo Drives* on page 3-31 for information on the holding operation.

**2** Home is defined when the home input turns ON.



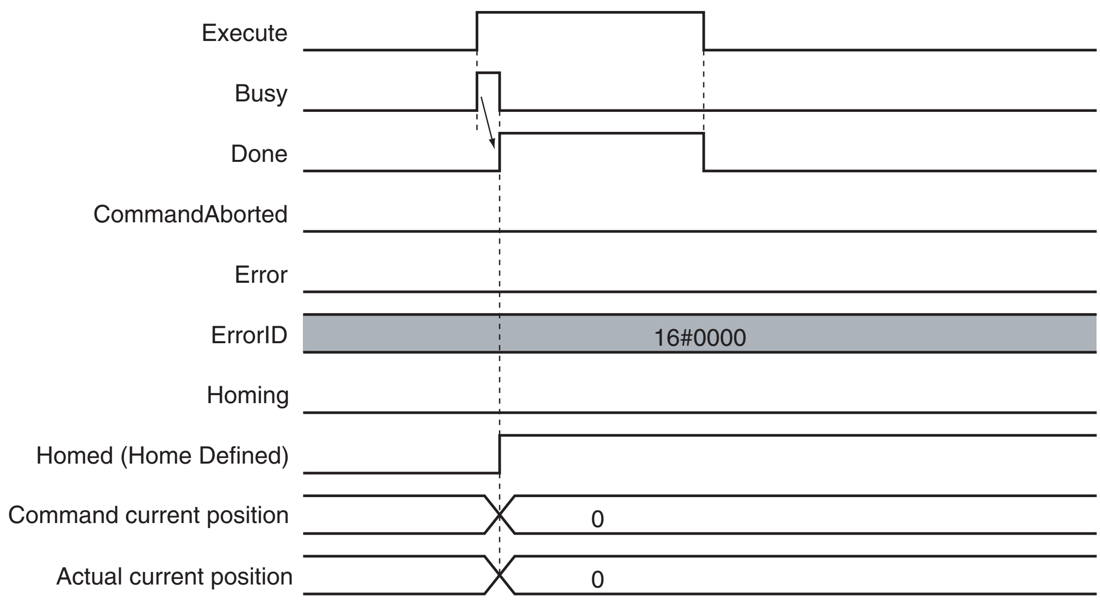

- Releasing the torque limit also differs between OMRON G5-series Servo Drives and other Servo Drives.

| | |
|---|---|
| G5 Servo Drives: | Automatically released when the axis moves in the direction opposite to home definition direction for the first time after homing. |
| Other Servo Drives: | If a torque limit is used, release the torque limit when the axis moves in the direction opposite to home definition direction for the first time after homing. Use the ECAT_WriteSDO command to change the torque limit. |

- Home is also defined if the home input signal turns ON before the homing approach velocity is reached after homing starts.

### 14: Zero Position Preset Operation

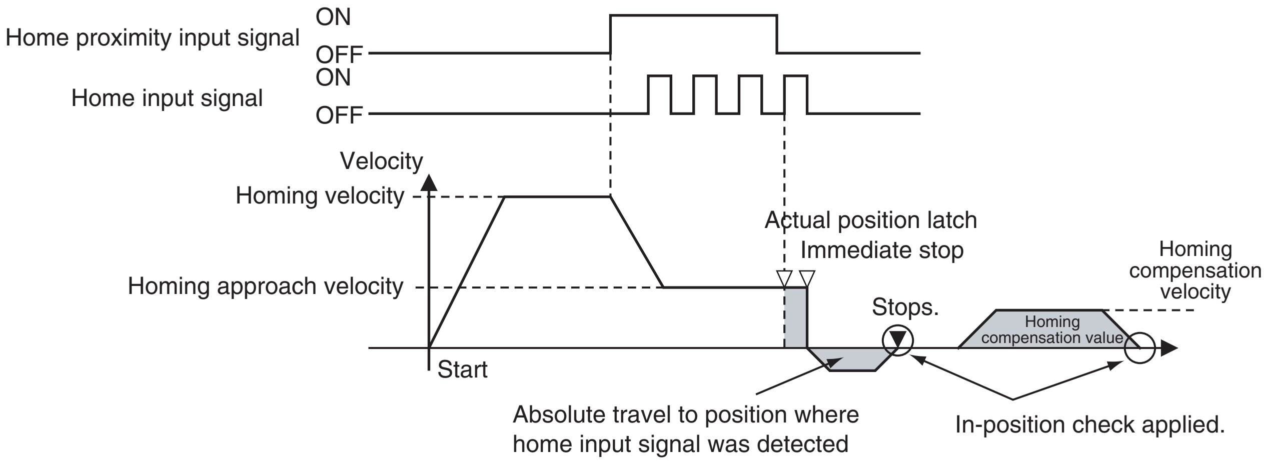
The command current position and the actual current position are set to the home position offset to define home. The following timing chart shows the operation when home position offset is set to 0.




## ● Axis Stopping Method and In-position Check When Homing

The axis is stopped with the following procedure when homing.

1. The actual position where the home input signal was detected is latched and the axis stops immediately.
2. After the immediate stop, the axis moves with absolute travel in the reverse direction to the position that was latched in step 1.

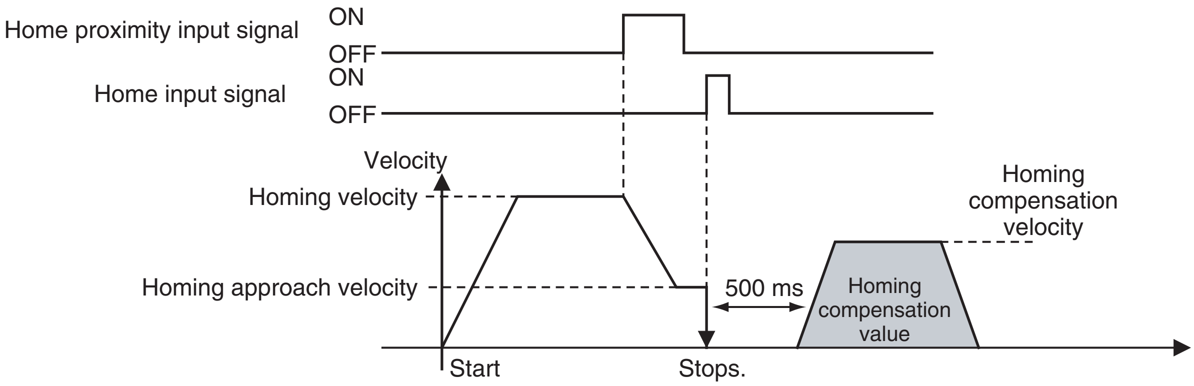



- Waiting for the in-position state is always performed for this instruction regardless of the setting of the In-position Check Time axis parameter.
  Be particularly careful when performing absolute travel to the home input signal detection position if the In-position Check Time is set to 0 because the instruction will continue waiting for the in-position status. Make sure that the signal is received within the in-position range. In-position waiting is not performed for the homing compensation value operation even if the in-position check time is set to 0. The status of in-position waiting can be checked with the *InPosWaiting* (In-position Waiting) system-defined variable for motion control.

## Homing Compensation

When you set a homing compensation value, the axis will move by the homing compensation value after the home input is detected to define home. Adjusting the workpiece is sometimes difficult after home has been defined in the mechanical system. You can use the homing compensation to fine-tune the position of home after it is defined. The travel velocity at this time is the homing compensation velocity. The axis starts moving 500 ms after it stopped when the home input was detected.

The sign of the homing compensation value indicates the direction relative to the coordinate axis. If it is negative, the axis moves in the negative direction.




## Override Factors

Overrides are disabled for this instruction.

## Automatic Control of Torque Limit

If you are using an OMRON G5-series Servo Drive, and you select either Proximity Reverse Turn/Holding Time Operation (12) or No Home Proximity Input/Holding Home Input Operation (13) for the homing operation, the torque limit will be automatically started in the holding direction. Automatic control of the torque limit when homing maintains the torque limit even after homing is completed. The torque limit is released when the axis moves in the direction opposite to the home input detection direction.
The torque limit is automatically released at the following times.

- When the Servo is turned OFF (Servo Unlock)
- When the Cyclic Synchronous Positioning Control (CSP) Mode is changed to another control mode.

During the homing operation, torque limits are released for operation in the direction opposite to the home input detection direction. For example, if the reversal operation direction at a limit input is in the direction opposite to the home input detection direction, the torque limit is released when the reversal operation is completed. If the operation direction reverses again and becomes the home input detection direction, the torque limit will be enabled again.

## Holding Operation for OMRON G5-series Servo Drives

Torque limits that are set in the Servo Drive in advance are used for the Proximity Reverse Turn/Holding Time (12) or No Home Proximity Input/Holding Home Input (13) Homing Operation Modes to automatically start torque control in the home input detection direction.

**Precautions for Correct Use**

The automatic torque limit function of the MC_Home instruction is not used for servo drives from other manufacturers. Use the MC_SetTorqueLimit instruction, SDO communications, or support software to set suitable values.

**Additional Information**

- The torque limits are continued even after a normal completion of homing.
- The torque limits are automatically released when an instruction that moves the axis in the opposite direction is executed.

### Settings for OMRON G5-series Servo Drives

To use the holding operation, you must use the support software of the Servo Drive to set the Torque Limit Selection (3521 hex) in the G5-series Servo Drive.

- Set the Torque Limit Selection to 6 to apply a torque limit in the home input detection direction during the holding operation for homing and to use the torque limit directions and values that are set with the MC_SetTorqueLimit instruction for other operation. In that case, the values of the input variables to the MC_SetTorqueLimit instruction are ignored during the holding operation for homing.
- If the Torque Limit Selection is set to 4, the values of the input variables to the MC_SetTorqueLimit instruction are always used. You must set torque limits that are suitable for both for the holding operation during homing and for other operations.

| | | Torque Limit Selection (3521 hex) | |
|---|---|---|---|
| | | **6 (recommended)** | **4** |
| Positive Torque Limit | Homing | Torque Limit 3 (3525 hex) is used. | The smaller of the PositiveValue (Positive Torque Limit) for the MC_SetTorqueLimit instruction and Torque Limit 1 (3013 hex) is used. |
| | Operations other than Homing | The smaller of the PositiveValue (Positive Torque Limit) for the MC_SetTorqueLimit instruction and Torque Limit 1 (3013 hex) is used. | |
| Negative Torque Limit | Homing | Torque Limit 4 (3526 hex) is used. | The smaller of the NegativeValue (Negative Torque Limit) for the MC_SetTorqueLimit instruction and Torque Limit 2 (3522 hex) is used. |
| | Operations other than Homing | The smaller of the NegativeValue (Negative Torque Limit) for the MC_SetTorqueLimit instruction and Torque Limit 2 (3522 hex) is used. | |

For details on torque limits, refer to *MC_SetTorqueLimit* on page 3-301. Refer to the *G5-series AC Servomotors/Servo Drives with Built-in EtherCAT Communications User's Manual* (Cat. No. I576) for information on the settings in the OMRON G5-series Servo Drive.

## ● Monitoring Following Error during Holding Operation

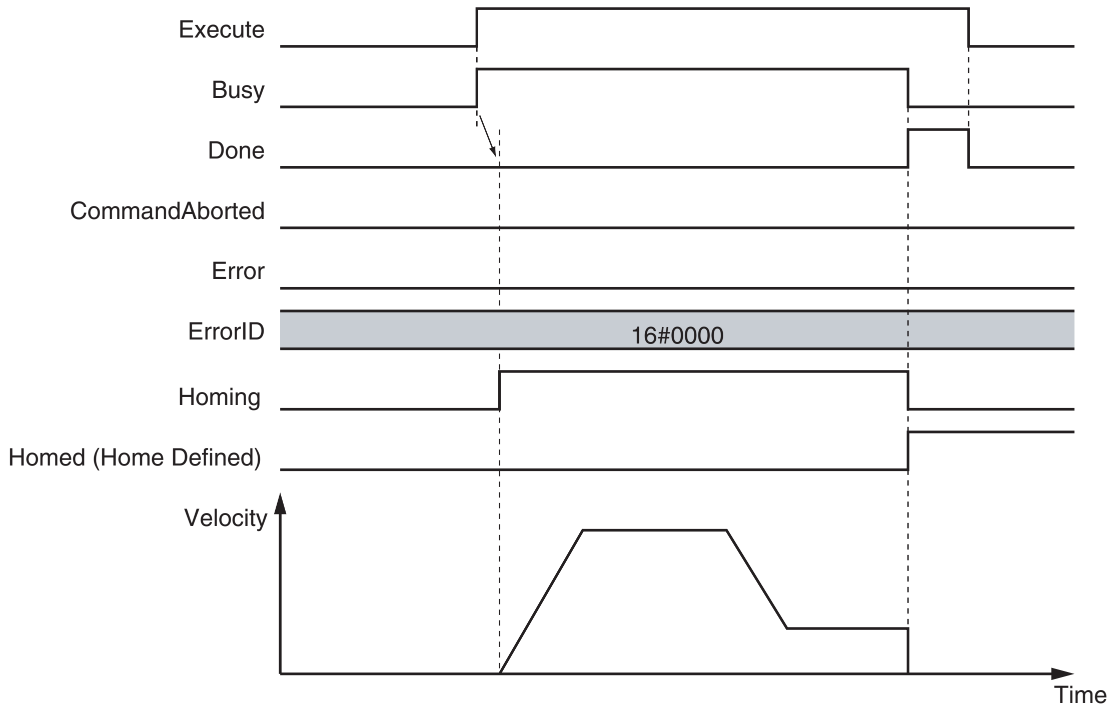
The following error is not monitored during the holding operation for homing.

For details on monitoring the following error, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

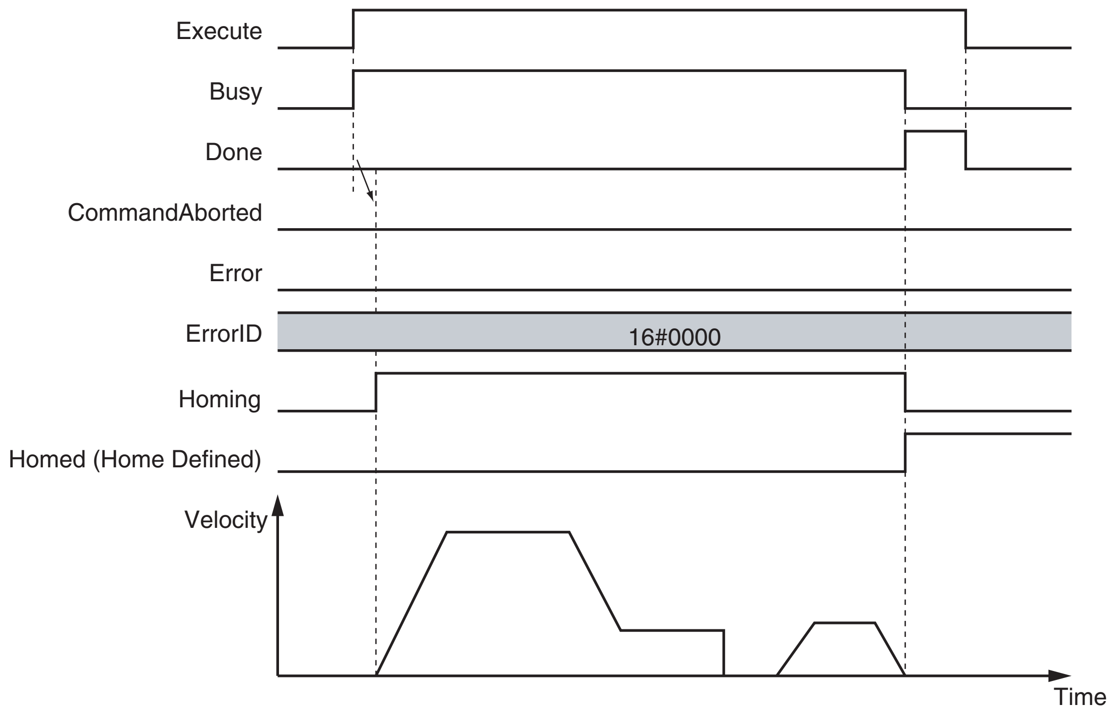
## Timing Charts

The following charts show the timing of homing.

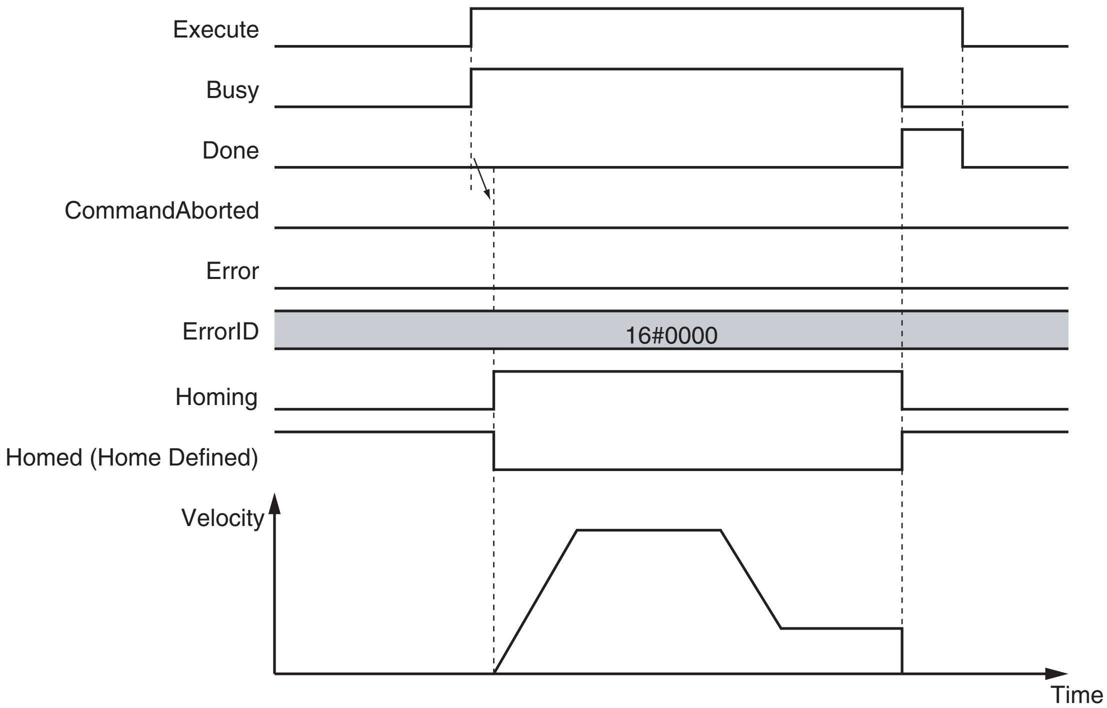
- **No Homing Compensation**



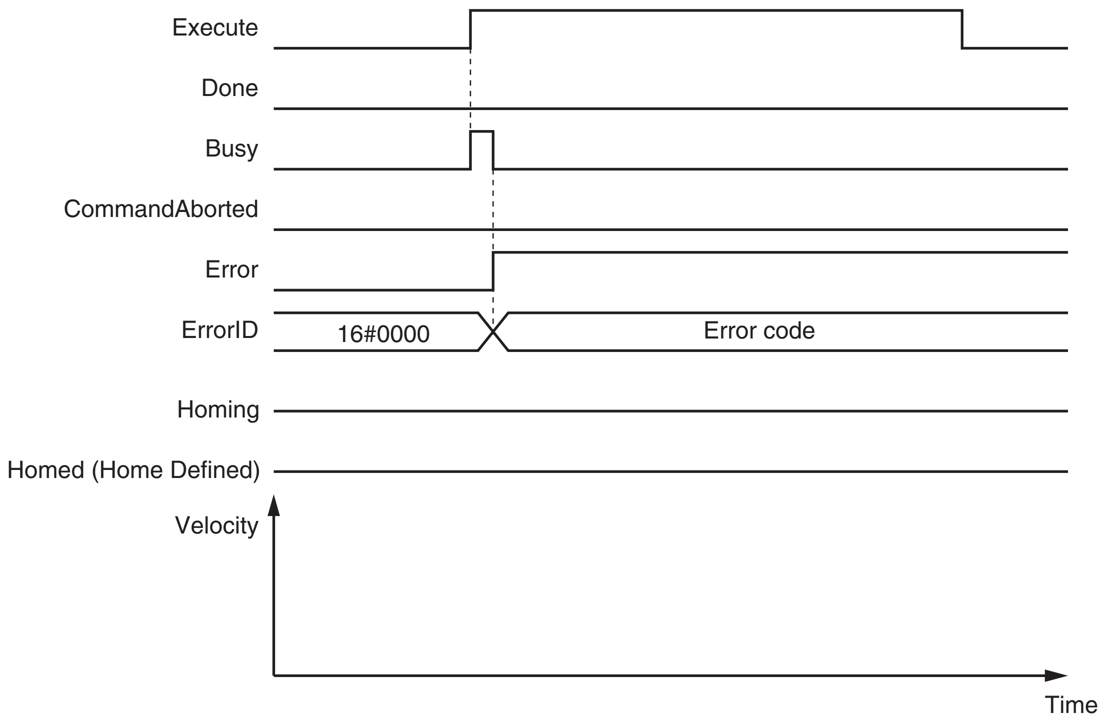

- **With Homing Compensation**



## ● Execution When Home Is Defined




## ● Execution with Incorrect Parameters or When Motion Control Instructions Are Disabled



## Re-execution of Motion Control Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Control Instructions

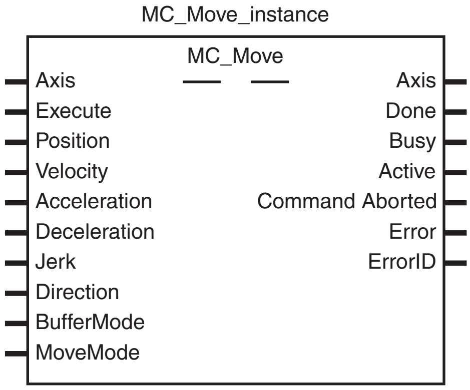
For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_Move

The MC_Move instruction performs absolute positioning or relative positioning.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_Move | Positioning | FB |  | MC_Move_instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>Position :=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>Direction :=*parameter,*<br>BufferMode :=*parameter,*<br>MoveMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Position | Target Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the target position in absolute coordinates when you specify absolute positioning as the Travel Mode. Specify the relative position when you specify relative positioning as the Travel Mode. The unit is command units.*1 |
| Velocity | Target Velocity | LREAL | Positive number | 0 | Specify the target velocity.*2 The unit is command units/s.*1 |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/s$^2$.*1 |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/s$^2$.*1 |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/s$^3$.*1 |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Direction | Direction | _eMC_DIRECTION | 0: _mcPositiveDirection<br>1: _mcShortestWay<br>2: _mcNegativeDirection<br>3: _mcCurrentDirection<br>4: _mcNoDirection | 0*3 | Specify the direction of rotation when MoveMode is set to absolute positioning*4 and when the Count Mode is Rotary Mode.<br>0: Positive direction<br>1: Shortest way<br>2: Negative direction<br>3: Current direction<br>4: No direction specified |
| BufferMode | Buffer Mode Selection | _eMC_BUFFER _ MODE | 0: _mcAborting<br>1: _mcBuffered<br>2: _mcBlendingLow<br>3: _mcBlendingPrevious<br>4: _mcBlendingNext<br>5: _mcBlendingHigh | 0*3 | Specifies the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered<br>2: Blend with low<br>3: Blend with previous<br>4: Blend with next<br>5: Blend with high |
| MoveMode | Travel Mode | _eMC_MOVE_ MODE | 0: _mcAbsolute<br>1: _mcRelative | 0*3 | Select the travel method.<br>0: Absolute positioning<br>1: Relative positioning |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 Always set the target velocity. A Target Velocity Setting Out of Range error (error code: 5422 hex) occurs when the instruction is executed if the target velocity is not set.

*3 The default value for an enumeration variable is actually not the number, but the enumerator.

*4 When MoveMode is set to relative positioning, the travel direction is determined by the sign of the position.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

MC_Move

3

Variables

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the axis decelerates to a stop and the velocity reaches 0. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborte*d changes to TRUE. |
| Active | When the instruction is started. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

- You can use the MC_Move instruction to perform absolute positioning or relative positioning.
- If you specify absolute positioning for *MoveMode* (Travel Mode), the operation is the same as for the MC_MoveAbsolute (Absolute Positioning) instruction. If you specify relative positioning for *MoveMode* (Travel Mode), the operation is the same as the MC_MoveRelative (Relative Positioning) instruction.
- If relative positioning is used, *Direction* is not used.
- For details, refer to *MC_MoveAbsolute* on page 3-41 or *MC_MoveRelative* on page 3-68.

## Instruction Details

This section describes the instruction in detail.

**Precautions for Correct Use**

When you perform absolute positioning, set the target position so that the settings of the Modulo Maximum Position Setting Value and Modulo Minimum Position Setting Value axis parameters are not exceeded. If the target position is set outside of the settings of the Modulo Maximum Position Setting Value and Modulo Minimum Position Setting Value axis parameters, a Target Position Setting Out of Range error (error code 5478 hex) will occur.

### ● In-position Check

An in-position check is performed for this instruction according to the settings in In-position Range and In-position Check Time axis parameters.

## Re-execution of Motion Control Instructions

You can change the operation of the instruction if you change an input variable during positioning and change *Execute* to TRUE again. Input variables *Position* (Target Position), *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate) can be changed by re-executing the motion control instruction. For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

You can switch to this instruction or buffer this instruction if you execute it during execution of another instruction. You can buffer one instruction per axis. Specify the operation of this instruction with *BufferMode* (Buffer Mode Selection) for multi-execution of instructions.

| Buffer Mode Selection | | Description |
|---|---|---|
| Aborting | | Aborts the instruction being executed and switches to this instruction.<br>If the direction of axis motion is reversed by switching instructions, the motion will be reversed according to the Operation at Reversing axis parameter. |
| Buffered | | Buffers this instruction and executes it automatically after the current instruction is completed. |
| Blending | | Starts the buffered instruction at the velocity (transit velocity) at which the current instruction reaches the target position. The operation of the current instruction is changed so that the axes reach the target position at the transit velocity. There are four methods to specify the transit velocity. These are described below. |
| | Blend with low | The lower of the target velocities of the current instruction and the buffered instruction is used as the transit velocity. |
| | Blend with previous | The target velocity of the current instruction is used as the transit velocity. |
| | Blend with next | The target velocity of the buffered instruction is used as the transit velocity. |
| | Blend with high | The higher of the target velocities of the current instruction and the buffered instruction is used as the transit velocity. |

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

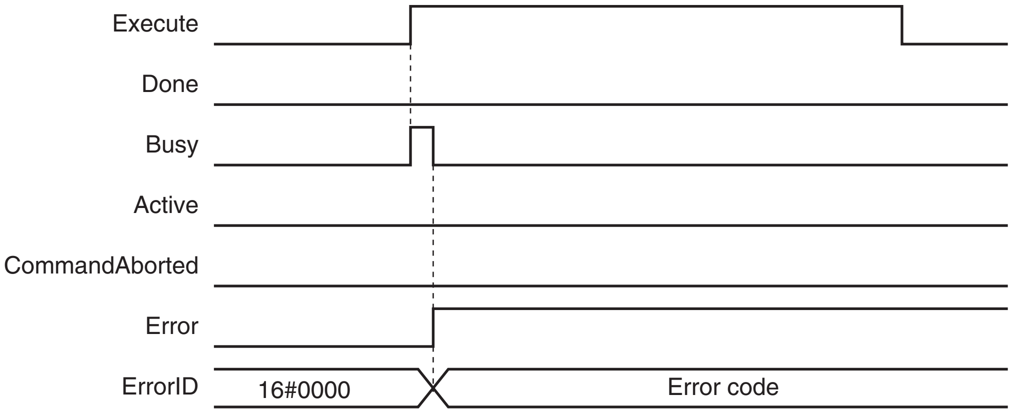
### ● Execution of Other Instructions during Instruction Execution

If you execute another instruction during execution of this instruction, you can specify aborting, buffering, or blending.

## Errors

### Timing Chart When Error Occurs

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

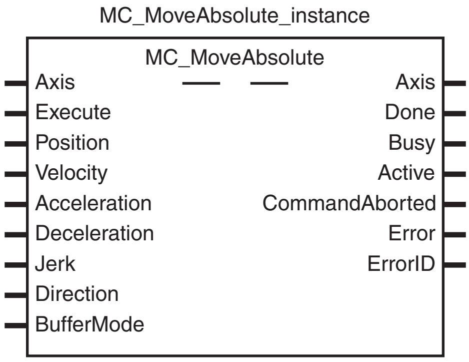



### Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_MoveAbsolute

The MC_MoveAbsolute instruction moves the axis to a specified absolute target position.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_MoveAbsolute | Absolute Positioning | FB |  | MC_MoveAbsolute_instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>Position :=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>Direction :=*parameter,*<br>BufferMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Position | Target Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the absolute target position. The unit is command units.[*1] |
| Velocity | Target Velocity | LREAL | Positive number | 0 | Specify the target velocity.[*2] The unit is command units/s.[*1] |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/$s^2$.[*1] |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.[*1] |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/$s^3$.[*1] |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Direction | Direction | _eMC_ DIRECTION | 0: _mcPositiveDirection<br>1: _mcShortestWay<br>2: _mcNegativeDirection<br>3: _mcCurrentDirection<br>4: _mcNoDirection | 0*3 | Specify the direction of rotation when the Count Mode is Rotary Mode.<br>0: Positive direction<br>1: Shortest way<br>2: Negative direction<br>3: Current direction<br>4: No direction specified |
| BufferMode | Buffer Mode Selection | _eMC_BUFFER_ MODE | 0: _mcAborting<br>1: _mcBuffered<br>2: _mcBlendingLow<br>3: _mcBlendingPrevious<br>4: _mcBlendingNext<br>5: _mcBlendingHigh | 0*3 | Specify the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered<br>2: Blend with low<br>3: Blend with previous<br>4: Blend with next<br>5: Blend with high |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 Always set the target velocity. If the axis is moved without setting a target velocity, an error will occur.

*3 The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When positioning is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the instruction is started. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

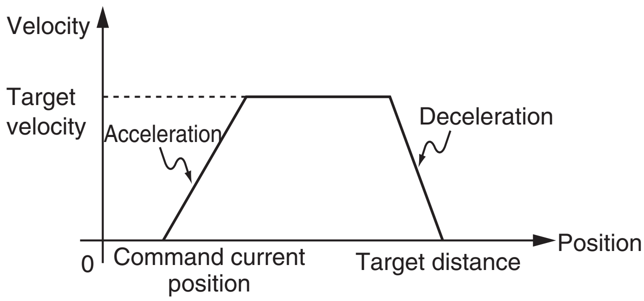
| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

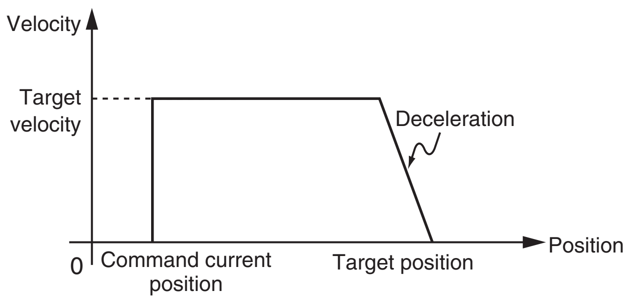
# Function

- The absolute target position is specified to perform positioning.
- Absolute positioning starts when *Execute* changes to TRUE.
- You can execute this instruction even if home is not defined.
- You can specify the *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), *Deceleration* (Deceleration Rate), and *Jerk* as input variables.

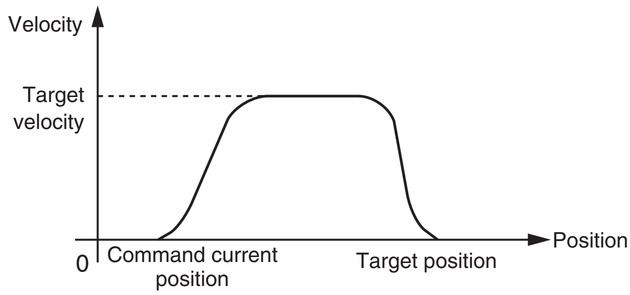
The following chart shows an operation example of absolute positioning.




When the *Acceleration* (Acceleration Rate) or *Deceleration* (Deceleration Rate) is 0 and the instruction is executed, the axis will reach the target velocity without accelerating or decelerating.
The following chart shows an operation example of when the acceleration rate is 0.



Specify *Jerk* when you want to accelerate or decelerate smoothly. The following chart shows an operation example when *Jerk* is specified.




For details on Jerk, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

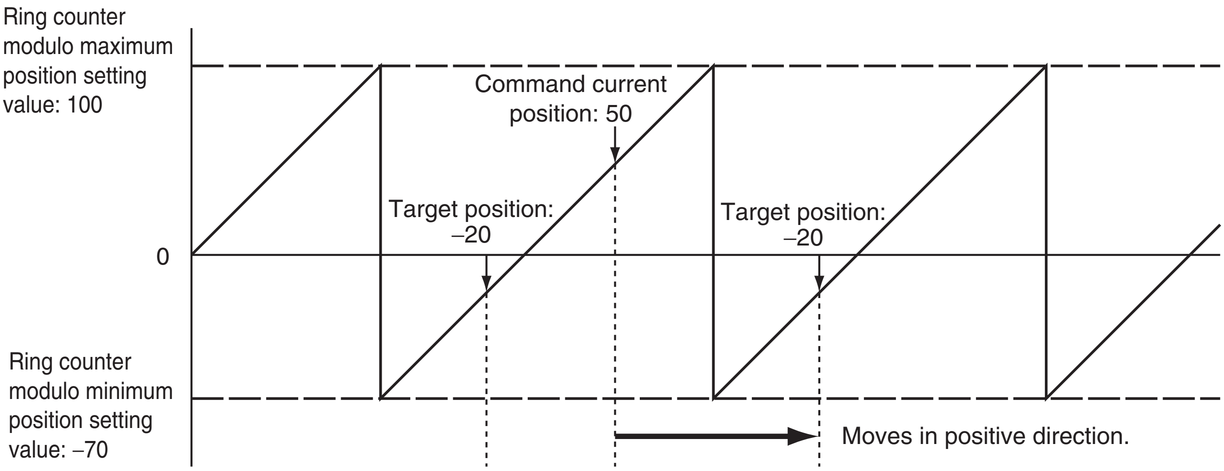
When the positioning is started with *Position* (Target Position) set to the command current position, the axis will not move but *Done* will change to TRUE. *Done* will change to TRUE in the cycle after *Busy* changes to TRUE. Therefore, *Active* (Controlling) will not change to TRUE.

## Instruction Details

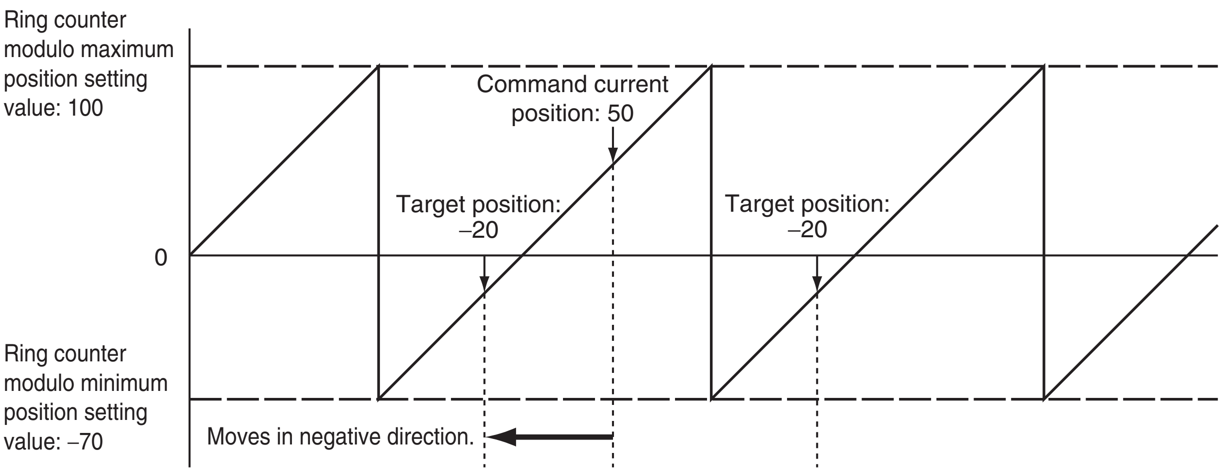
This section describes the instruction in detail.

### ● *Direction*

*Direction* specifies the direction for starting positioning when the Count Mode is Rotary Mode. *Direction* is not used if the Count Mode is Linear Mode. Positioning starts in the positive direction towards the target position when the positive direction is specified for *Direction*. The following chart shows an operation example when positioning starts with a command position of 50 and moves toward –20.

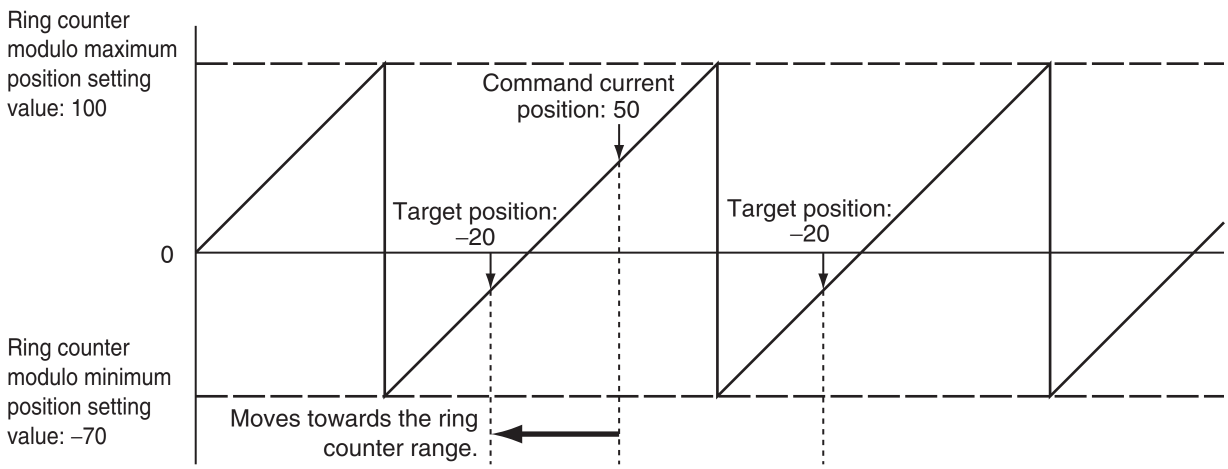



Positioning starts in the negative direction towards the target position when the negative direction is specified for *Direction*. The following chart shows an operation example when positioning starts with a command position of 50 and moves toward –20.

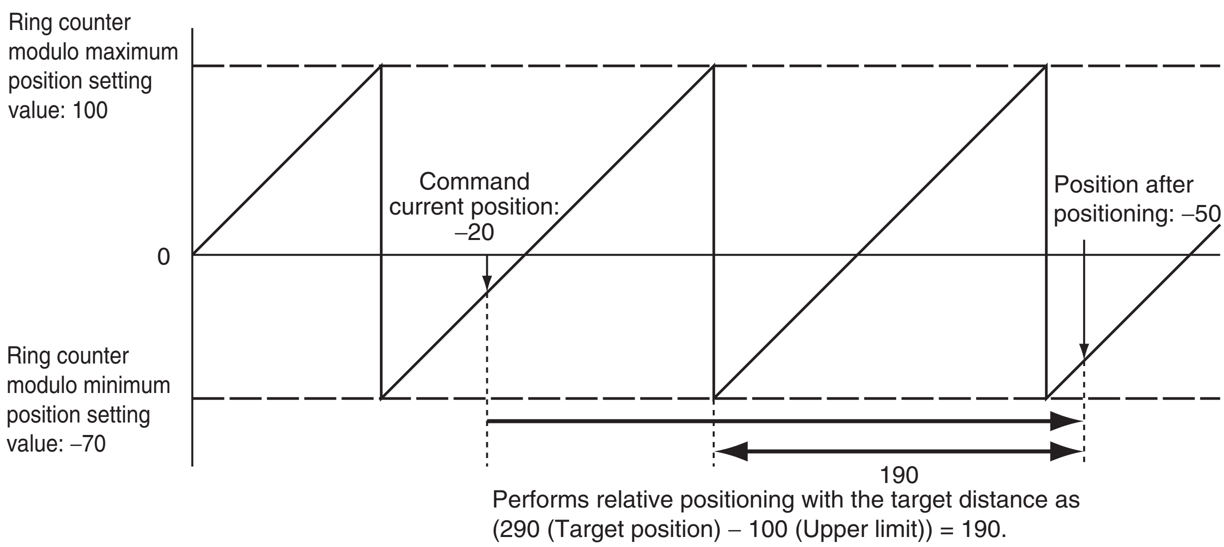


Positioning starts towards the target position within the ring counter range when *No direction* is specified for *Direction*. Therefore, the size relationship between the command current position and the target position determines the direction of travel.

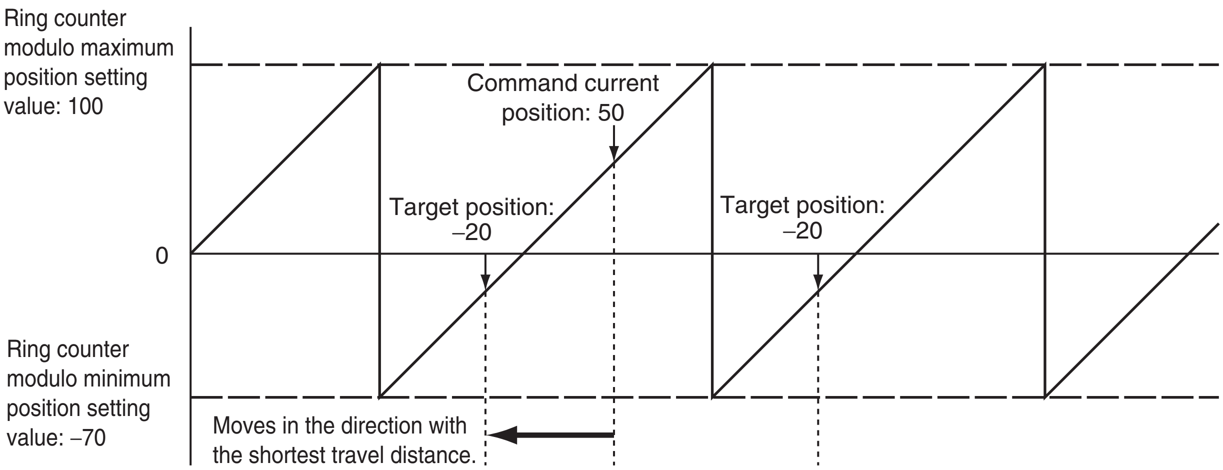
The following chart shows an operation example when positioning starts with a command position of 50 and moves toward –20.




When *No direction* is specified for *Direction,* you can specify a *Position* (Target Position) outside the range specified by the upper and lower limits of the ring counter. When *Position* (Target Position) is outside the range specified by the upper and lower limits of the ring counter, positioning is performed using the travel distance exceeding the upper limit as a relative distance. This allows positioning of multiple ring rotations. Positioning is the same when *Position* (Target Position) is below the lower limit of the ring counter as well. The following chart shows an operation example for when the command current position is –20 and *Position* (Target Position) is 290.



When the shortest way is specified for *Direction*, positioning starts in the direction with the shortest distance between the command current position and the target position. The following chart shows an operation example when positioning starts with a command position of 50 and moves toward –20.




Movement is in the same direction as the current direction if the travel distance is the same in the positive and the negative direction.

**Precautions for Correct Use**

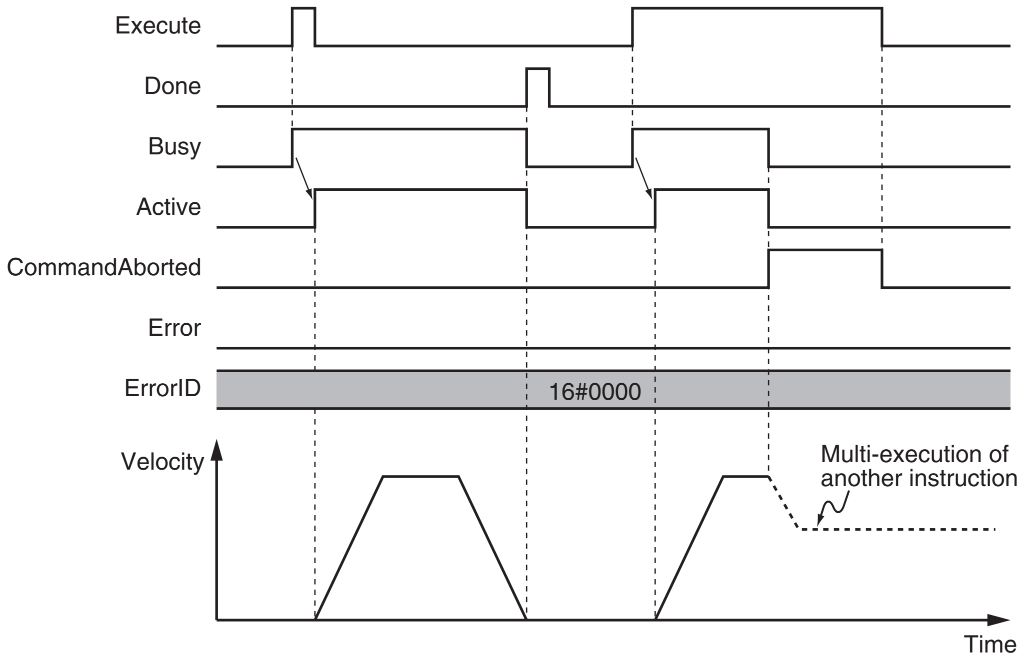
- When you perform absolute positioning, set the target position so that the settings of the Modulo Maximum Position Setting Value and Modulo Minimum Position Setting Value axis parameters are not exceeded. If the target position is set outside of the settings of the Modulo Maximum Position Setting Value and Modulo Minimum Position Setting Value axis parameters, a Target Position Setting Out of Range error (error code 5478 hex) will occur. However, when *No direction* is specified for *Direction*, you can specify a *Position* (Target Position) outside the range specified by the upper and lower limits of the ring counter.
- If the current direction is specified for *Direction*, operation is in the same command direction as the previous motion. Therefore, depending on the instructions that are used together, the direction may not be the same as the direction that was specified with the input to the motion control instruction for the previous motion. When you specify the current direction, check the current direction with *Dir.Posi* (Positive Direction) and *Dir.Nega* (Negative Direction) in the Axis Variable.

## ● In-position Check

An in-position check is performed for this instruction according to the settings in In-position Range and In-position Check Time axis parameters.

## Timing Charts

- *Busy* (Executing) changes to TRUE at the same time as *Execute* changes to TRUE. *Active* (Controlling) changes to TRUE in the next period.
- *Done* changes to TRUE when *Position* (Target Position) is reached and positioning is completed.
- If another instruction aborts this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing) and *Active* (Controlling) change to FALSE.




## Re-execution of Motion Control Instructions

You can change the operation of the instruction if you change an input variable during positioning and change *Execute* to TRUE again. Input variables *Position* (Target Position), *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate) can be changed by re-executing the motion control instruction. For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### Execution during Execution of Other Instructions

You can switch to this instruction or buffer this instruction if you execute it during execution of another instruction. You can buffer one instruction per axis. Specify the operation of this instruction using *BufferMode* (Buffer Mode Selection) for multi-execution of instructions.

| Buffer Mode Selection | | Description |
|---|---|---|
| Aborting | | Aborts the instruction being executed and switches to this instruction. If the direction of axis motion is reversed by switching instructions, the motion will be reversed according to the Operation at Reversing axis parameter. |
| Buffered | | Buffers this instruction and executes it automatically after the current instruction is completed. |
| Blending | | Starts the buffered instruction at the velocity (transit velocity) at which the current instruction reaches the target position. The operation of the current instruction is changed so that the axes reach the target position at the transit velocity. There are four methods to specify the transit velocity. These are described below. |
| | Blend with low | The lower of the target velocities of the current instruction and the buffered instruction is used as the transit velocity. |
| | Blend with previous | The target velocity of the current instruction is used as the transit velocity. |
| | Blend with next | The target velocity of the buffered instruction is used as the transit velocity. |
| | Blend with high | The higher of the target velocities of the current instruction and the buffered instruction is used as the transit velocity. |

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

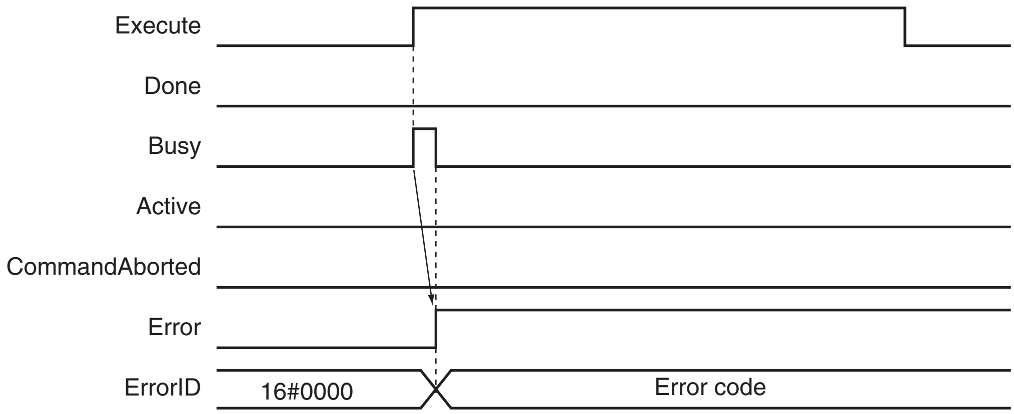
### Execution of Other Instructions during Instruction Execution

If you execute another instruction during execution of this instruction, you can specify aborting, buffering, or blending.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

### ● Timing Chart When Error Occurs




### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

## Sample Programming 1

This section shows sample programming for positioning by periodic multi-execution of instructions.

### Parameter Settings

The minimum settings required for this sample programming are given below.

- **Setting Axis Parameters**

**Axis Type**

| Axis | Axis Type |
|---|---|
| Axis 1 | Servo axis |

**Count Mode**

| Axis | Count Mode |
|---|---|
| Axis 1 | Linear Mode |

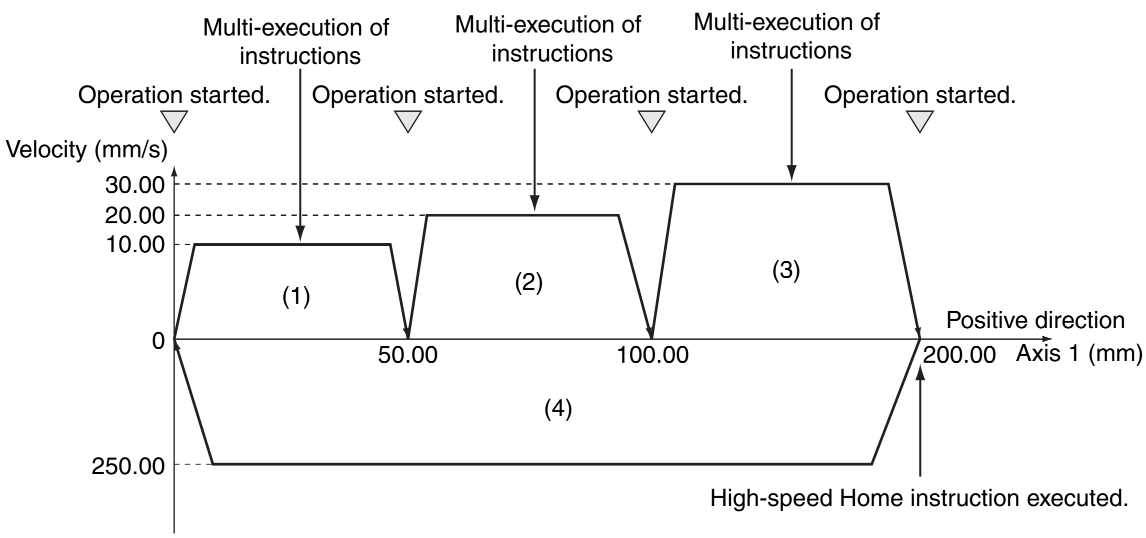
**Unit of Display**

| Axis | Unit of Display |
|---|---|
| Axis 1 | mm |

### Operation Example

In this sample, *BufferMode* (Buffer Mode Selection) is set to *Buffered* for MC_MoveAbsolute (Absolute Positioning) instructions and the axis is moved to the final target position by executing multiple instructions.
When the axis reaches the final target position, it is returned to home with the MC_ZeroPosition (High-speed Home) instruction.
Multi-execution of instructions is performed when the *Active* (Controlling) output variable from the previous instruction is TRUE. For single-axis operation, multi-execution is possible for only one instruction.

- **Operation Pattern**



**1** Turning ON the Operation Start Switch

When you turn ON the operation start switch at home, axis 1 is positioned to 50.00 mm in the positive direction.

**2** Turning ON the Operation Start Switch Again

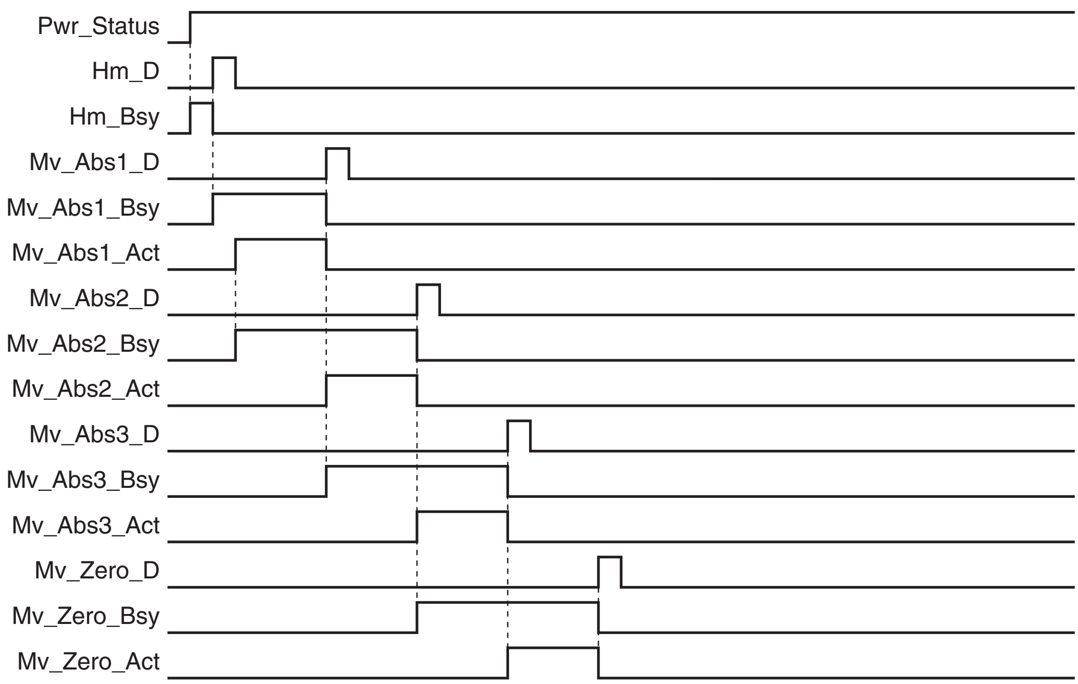
Thereafter, axis 1 is positioned to 100.00 mm and 200.00 mm, and then returns to home and stops. The operation start switch must be turned ON once for each of these motions.

## Ladder Diagram

### ● Main Variables

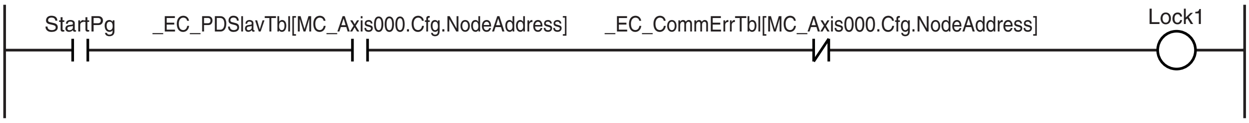
| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| Pwr_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |

### ● Timing Chart

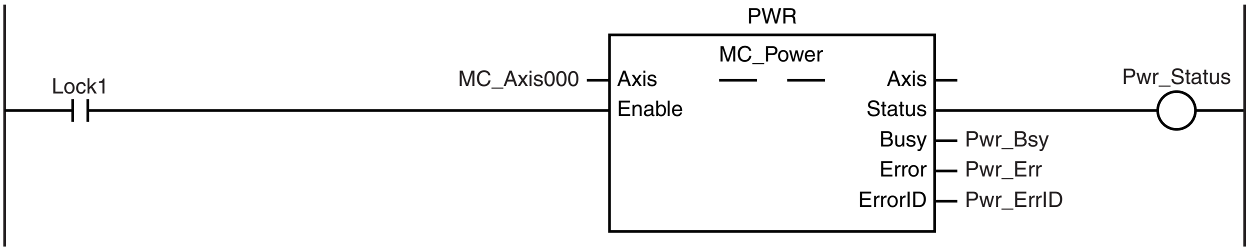


- **Sample Programming**

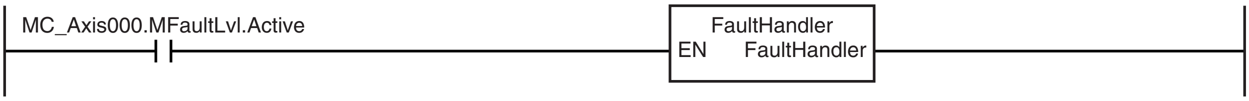
If *StartPg* is TRUE, EtherCAT communications are checked to see if process data communications are normal.



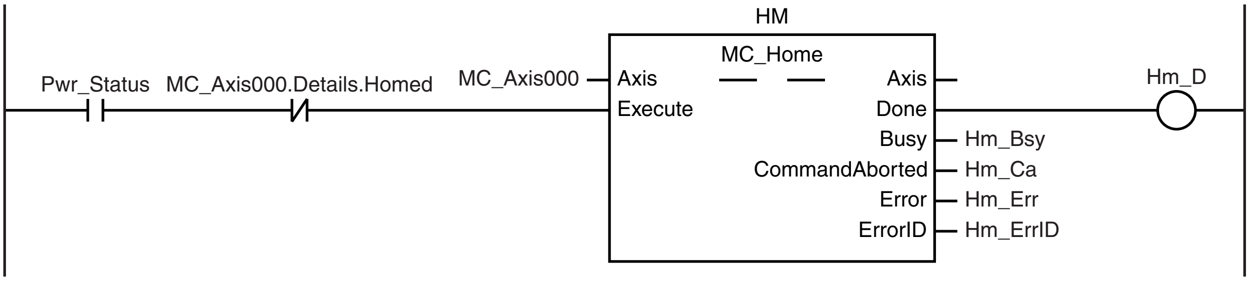

If process data communications are active, the Servo is turned ON.



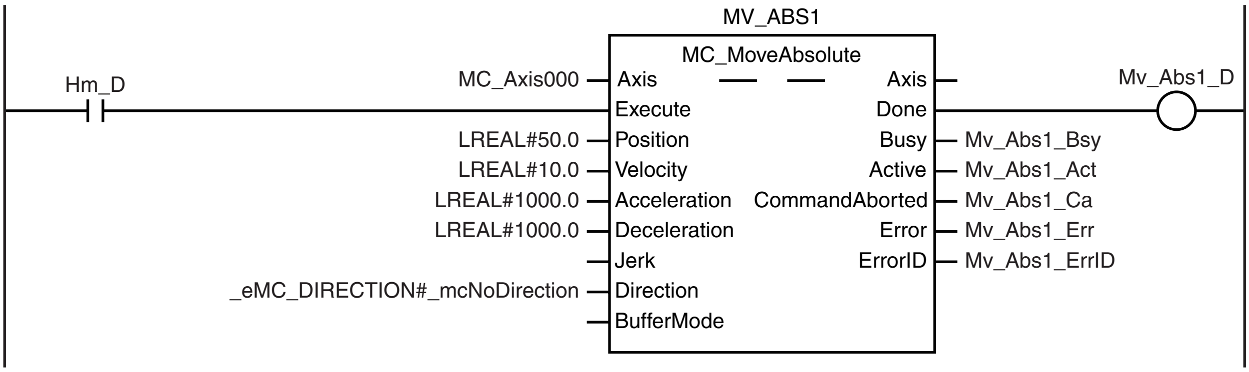

If there is a minor fault level error for axis 1, the error handler for the device (FaultHandler) is executed.
Design FaultHandler separately.



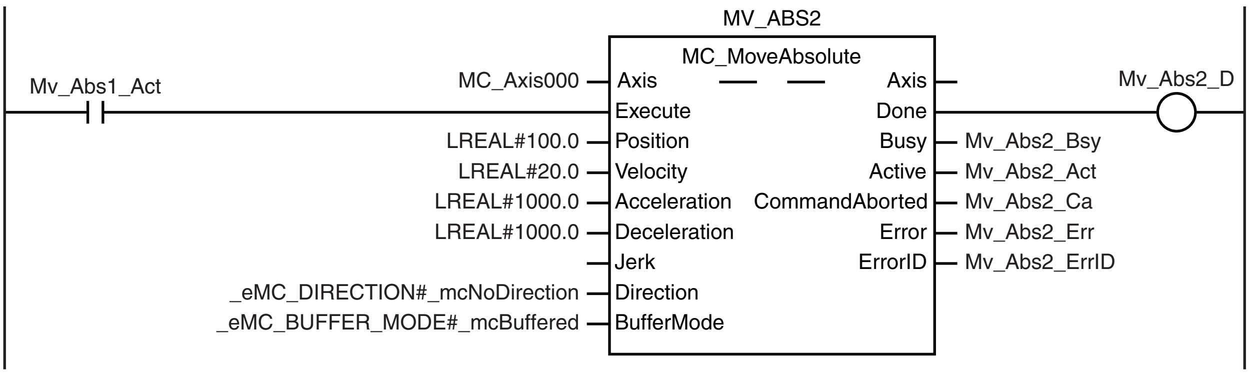

If the Servo is ON and home is not defined, the Home instruction is executed.



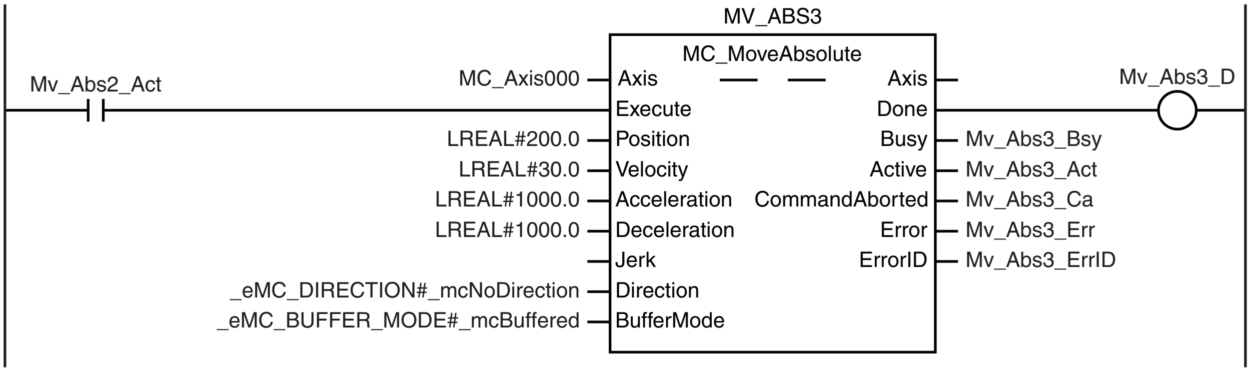

After home is defined, absolute positioning 1 is started.



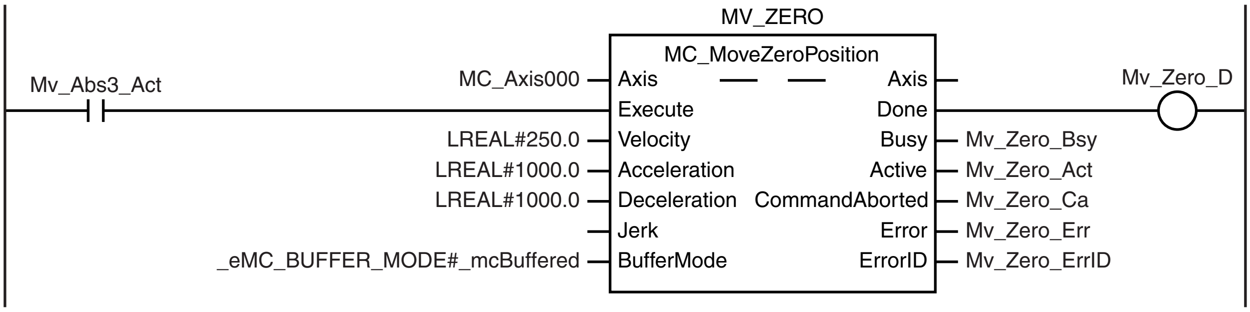

After absolute positioning 1 is started, absolute positioning 2 is started with multi-execution of instructions.



After absolute positioning 2 is started, absolute positioning 3 is started with multi-execution of instructions.




After absolute positioning 3 is started, the High-speed Home instruction is executed.

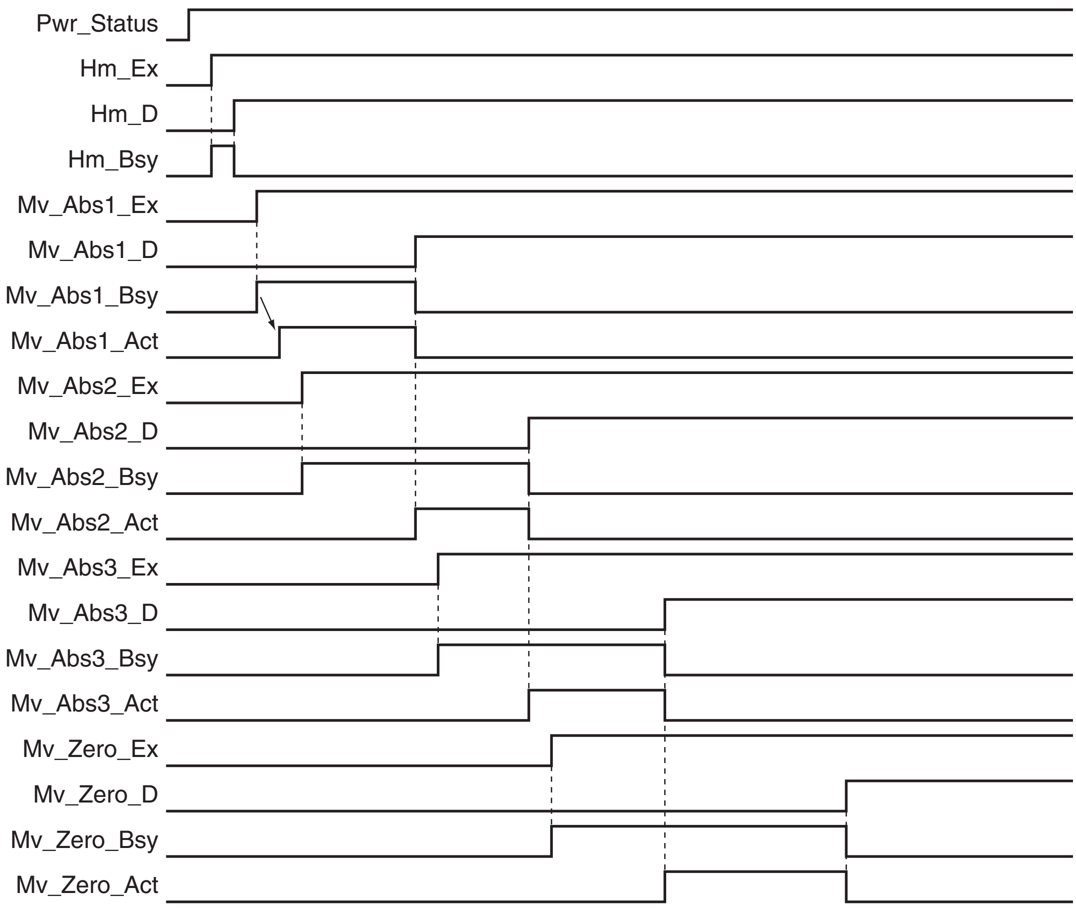



## Structured Text (ST)

### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| Pwr_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| Hm_Ex | BOOL | FALSE | The HM instance of MC_Home is executed when this variable changes to TRUE. |
| Mv_Abs1_Ex | BOOL | FALSE | The MV_ABS1 instance of MC_MoveAbsolute is executed when this variable changes to TRUE. |
| Mv_Abs2_Ex | BOOL | FALSE | The MV_ABS2 instance of MC_MoveAbsolute is executed when this variable changes to TRUE. |
| Mv_Abs3_Ex | BOOL | FALSE | The MV_ABS3 instance of MC_MoveAbsolute is executed when this variable changes to TRUE. |
| Mv_Zero_Ex | BOOL | FALSE | The MV_ZERO instance of MC_MoveZeroPosition is executed when this variable changes to TRUE. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

## ● Timing Chart




## ● Sample Programming

```
// Processing when input parameters are not set
IF InitFlag = FALSE THEN

    // MV_ABS1 parameters
    Mv_Abs1_Pos     := LREAL#50.0;
    Mv_Abs1_Vel     := LREAL#10.0;
    Mv_Abs1_Acc     := LREAL#1000.0;
    Mv_Abs1_Dec     := LREAL#1000.0;
    Mv_Abs1_Dir     := _eMC_DIRECTION#_mcNoDirection;

    // MV_ABS2 parameters
    Mv_Abs2_Pos     := LREAL#100.0;
    Mv_Abs2_Vel     := LREAL#20.0;
    Mv_Abs2_Acc     := LREAL#1000.0;
    Mv_Abs2_Dec     := LREAL#1000.0;
    Mv_Abs2_Dir     := _eMC_DIRECTION#_mcNoDirection;
    Mv_Abs2_Bm      := _eMC_BUFFER_MODE#_mcBuffered;

    // MV_ABS3 parameters
    Mv_Abs3_Pos     := LREAL#200.0;
    Mv_Abs3_Vel     := LREAL#30.0;
    Mv_Abs3_Acc     := LREAL#1000.0;
    Mv_Abs3_Dec     := LREAL#1000.0;
    Mv_Abs3_Dir     := _eMC_DIRECTION#_mcNoDirection;
    Mv_Abs3_Bm      := _eMC_BUFFER_MODE#_mcBuffered;

    // MV_ZERO parameters
    Mv_Zero_Vel     := LREAL#250;
    Mv_Zero_Acc     := LREAL#1000.0;
```

```
    Mv_Zero_Dec     := LREAL#1000.0;
    Mv_Zero_Bm      := _eMC_BUFFER_MODE#_mcBuffered;

    // Change InitFlag to TRUE after setting the input parameters.
    InitFlag:=TRUE;

END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis000.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
    Pwr_En:=TRUE;
ELSE
    Pwr_En:=FALSE;
END_IF;

// Processing for a minor fault level error
// Implement FaultHandler separately.
IF MC_Axis000.MFaultLvl.Active=TRUE THEN
    FaultHandler();
END_IF;


// If the Servo is ON and home is not defined, the Home instruction is executed.
IF (Pwr_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
    Hm_Ex:=TRUE;
END_IF;

// After home is defined, MV_ABS1 is executed.
IF Hm_D=TRUE THEN
    Mv_Abs1_Ex:=TRUE;
END_IF;

// After MV_ABS1 is started, MV_ABS2 is executed with multi-execution of instructions.
IF Mv_Abs1_Act=TRUE THEN
    Mv_Abs2_Ex:=TRUE;
END_IF;

// After MV_ABS2 is started, MV_ABS3 is executed with multi-execution of instructions.
IF Mv_Abs2_Act=TRUE THEN
    Mv_Abs3_Ex:=TRUE;
END_IF;

// After MV_ABS3 is started, MV_ZERO is executed with multi-execution of instructions.
IF Mv_Abs3_Act=TRUE THEN
    Mv_Zero_Ex:=TRUE;
END_IF;
```

```
//MC_Power
PWR(
    Axis        := MC_Axis000,
    Enable      := Pwr_En,
    Status      => Pwr_Status,
    Busy        => Pwr_Bsy,
    Error       => Pwr_Err,
    ErrorID     => Pwr_ErrID
);


//MC_Home
HM(
    Axis                := MC_Axis000,
    Execute             := Hm_Ex,
    Done                => Hm_D,
    Busy                => Hm_Bsy,
    CommandAborted      => Hm_Ca,
    Error               => Hm_Err,
    ErrorID             => Hm_ErrID
);


//MC_MoveAbsolute
MV_ABS1(
    Axis                := MC_Axis000,
    Execute             := Mv_Abs1_Ex,
    Position            := Mv_Abs1_Pos,
    Velocity            := Mv_Abs1_Vel,
    Acceleration        := Mv_Abs1_Acc,
    Deceleration        := Mv_Abs1_Dec,
    Direction           := Mv_Abs1_Dir,
    Done                => Mv_Abs1_D,
    Busy                => Mv_Abs1_Bsy,
    Active              => Mv_Abs1_Act,
    CommandAborted      => Mv_Abs1_Ca,
    Error               => Mv_Abs1_Err,
    ErrorID             => Mv_Abs1_ErrID
);


MV_ABS2(
    Axis                := MC_Axis000,
    Execute             := Mv_Abs2_Ex,
    Position            := Mv_Abs2_Pos,
    Velocity            := Mv_Abs2_Vel,
    Acceleration        := Mv_Abs2_Acc,
    Deceleration        := Mv_Abs2_Dec,
    Direction           := Mv_Abs2_Dir,
    BufferMode          := Mv_Abs2_Bm,
    Done                => Mv_Abs2_D,
    Busy                => Mv_Abs2_Bsy,
    Active              => Mv_Abs2_Act,
    CommandAborted      => Mv_Abs2_Ca,
    Error               => Mv_Abs2_Err,
    ErrorID             => Mv_Abs2_ErrID
);


MV_ABS3(
    Axis                := MC_Axis000,
    Execute             := Mv_Abs3_Ex,
    Position            := Mv_Abs3_Pos,
    Velocity            := Mv_Abs3_Vel,
    Acceleration        := Mv_Abs3_Acc,
```

```
    Deceleration      := Mv_Abs3_Dec,
    Direction         := Mv_Abs3_Dir,
    BufferMode        := Mv_Abs3_Bm,
    Done              => Mv_Abs3_D,
    Busy              => Mv_Abs3_Bsy,
    Active            => Mv_Abs3_Act,
    CommandAborted    => Mv_Abs3_Ca,
    Error             => Mv_Abs3_Err,
    ErrorID           => Mv_Abs3_ErrID
);

//MC_MoveZeroPosition
MV_ZERO(
    Axis              := MC_Axis000,
    Execute           := Mv_Zero_Ex,
    Velocity          := Mv_Zero_Vel,
    Acceleration      := Mv_Zero_Acc,
    Deceleration      := Mv_Zero_Dec,
    BufferMode        := Mv_Zero_Bm,
    Done              => Mv_Zero_D,
    Busy              => Mv_Zero_Bsy,
    Active            => Mv_Zero_Act,
    CommandAborted    => Mv_Zero_Ca,
    Error             => Mv_Zero_Err,
    ErrorID           => Mv_Zero_ErrID
);
```

## Sample Programming 2

In this sample, when the Count Mode is set to Rotary Mode and positioning is performed toward the target position, the shortest direction, clockwise or counterclockwise, is automatically determined and positioning is performed.
This section shows sample programming for shortest-way control of the rotation direction of a tool changer.

### Parameter Settings

The minimum settings required for this sample programming are given below.

- **Setting Axis Parameters**

#### Axis Type

| Axis | Axis Type |
|---|---|
| Axis 1 | Servo axis |

#### Count Mode

| Axis | Count Mode |
|---|---|
| Axis 1 | Rotary Mode |

#### Ring Counter

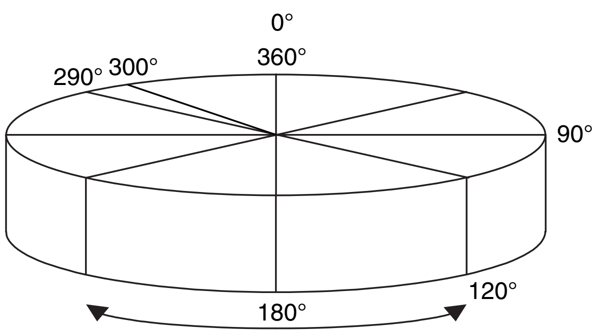
| Axis | Modulo maximum position | Modulo minimum position |
|---|---|---|
| Axis 1 | 360 | 0 |

#### Unit of Display

| Axis | Unit of Display |
|---|---|
| Axis 1 | degree |

### Operation Example

In this sample, multi-execution of absolute positioning instructions is used to position in a range of 0° to 360°. The actual position returns to 0° once it exceeds the range of 0° to 360°.
Multi-execution of instructions is performed when the *Active* (Controlling) output variable from the previous instruction is TRUE. For single-axis operation, multi-execution is possible for only one instruction.
In this sample, multi-execution of instructions is executed with *BufferMode* (Buffer Mode Selection) set to *Buffered*.



Shortest-way positioning is performed.

If you specify 0° (home), 90°, 120°, or 290°, the axis will move to that position. The rotation direction in this instance is in the shorter rotation direction. The travel velocity is 250°/s.

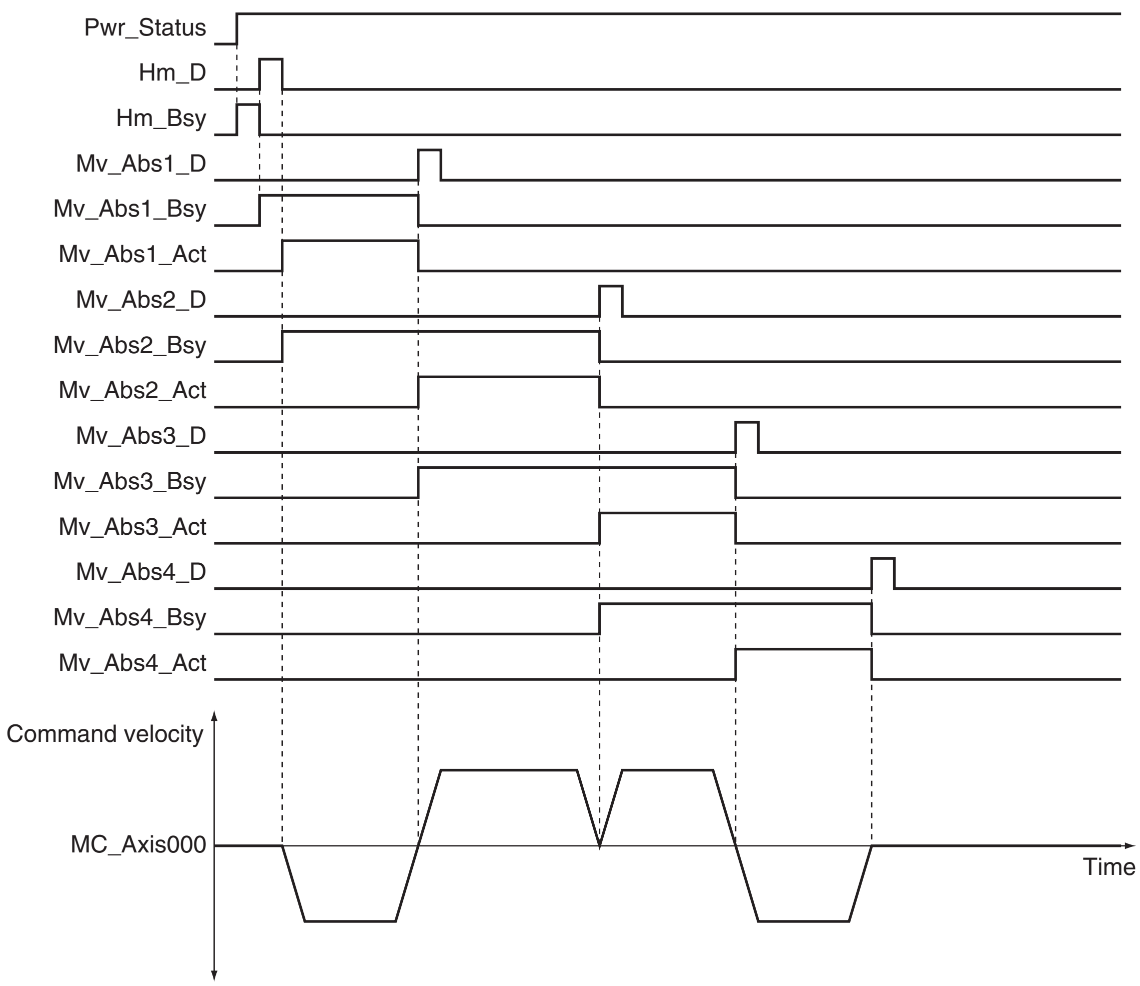
The sample programming performs positioning with a shortest way specification from 290° to 90° to 120° and then to home (0°).

## Ladder Diagram

### ● Main Variables

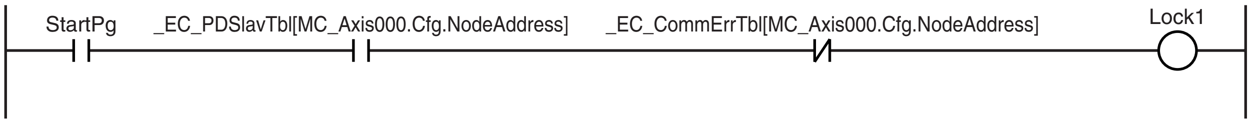
| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| Pwr_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |

### ● Timing Chart

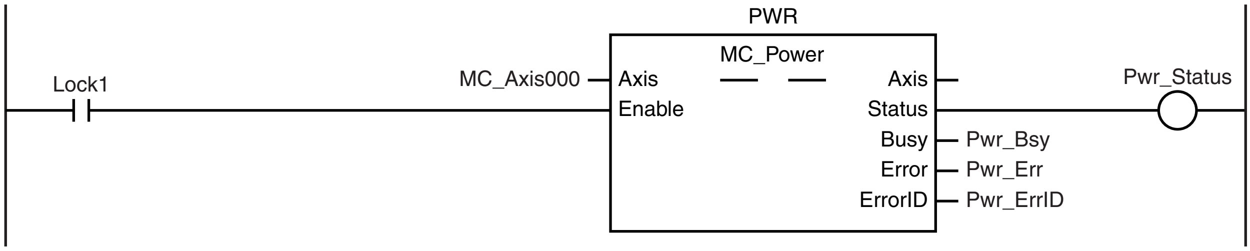


## ● Sample Programming

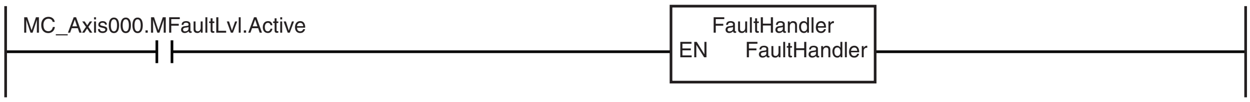
If *StartPg* is TRUE, EtherCAT communications are checked to see if process data communications are normal.



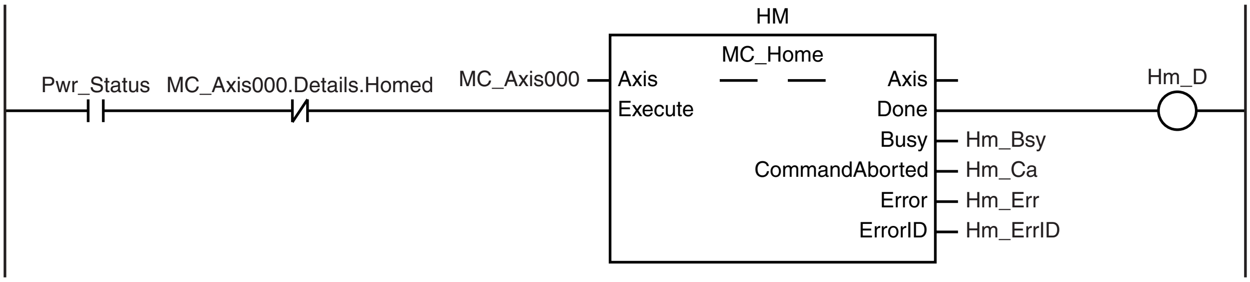

If process data communications are active, the Servo is turned ON.



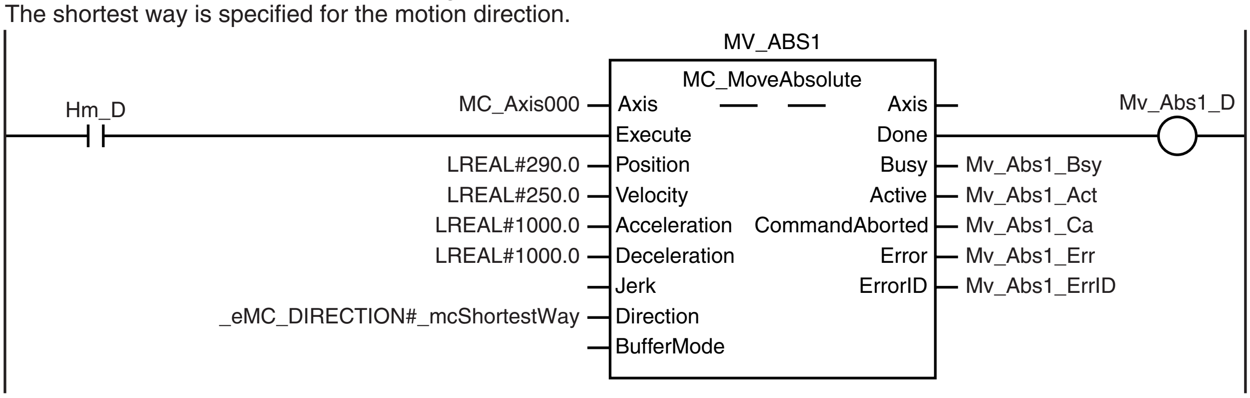

If there is a minor fault level error for axis 1, the error handler for the device (FaultHandler) is executed.
Design FaultHandler separately.



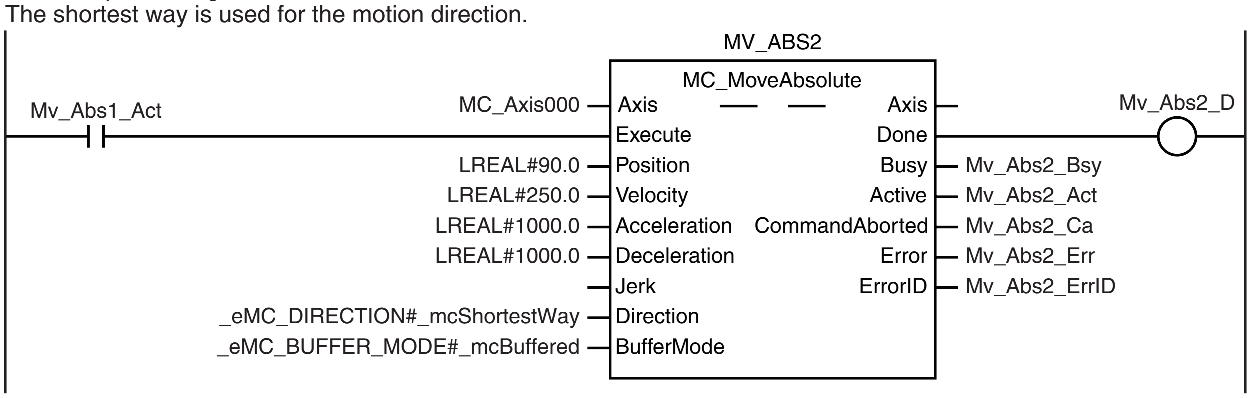
If the Servo is ON and home is not defined, the Home instruction is executed.



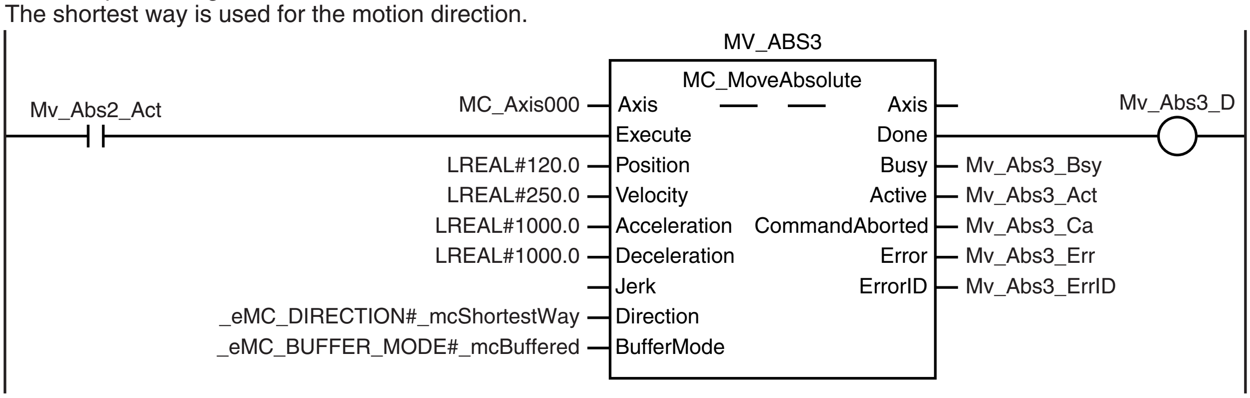

After home is defined, absolute positioning 1 is executed to move to 290.0°.
The shortest way is specified for the motion direction.



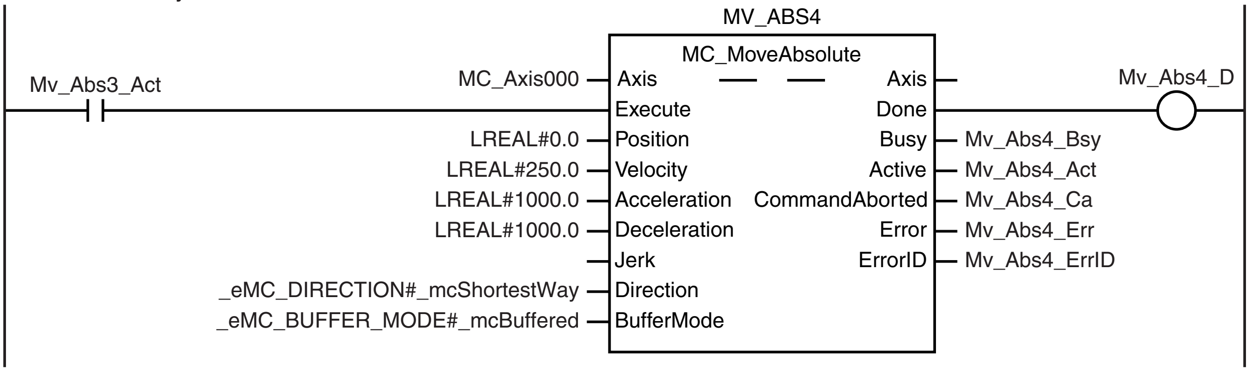

Absolute positioning 2 is executed with multi-execution of instructions to move from 290.0° to 90.0°.
The shortest way is used for the motion direction.




Absolute positioning 3 is executed with multi-execution of instructions to move from 90.0° to 120.0°.
The shortest way is used for the motion direction.



Absolute positioning 4 is executed with multi-execution of instructions to move from 120.0° to 0.0°.
The shortest way is used for the motion direction.

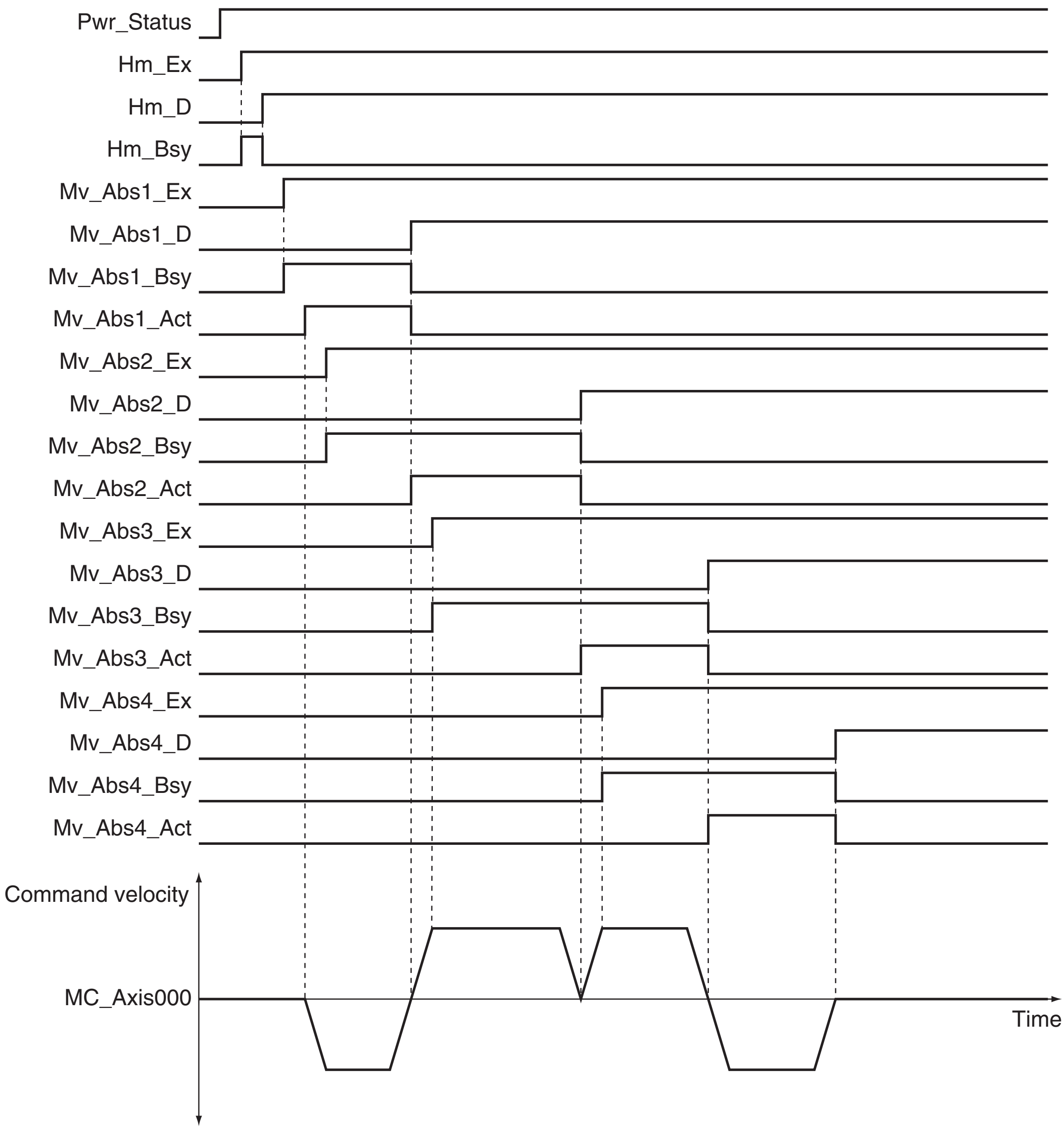



## Structured Text (ST)

### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| Pwr_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| Hm_Ex | BOOL | FALSE | The HM instance of MC_Home is executed when this variable changes to TRUE. |
| Mv_Abs1_Ex | BOOL | FALSE | The MV_ABS1 instance of MC_MoveAbsolute is executed when this variable changes to TRUE. |
| Mv_Abs2_Ex | BOOL | FALSE | The MV_ABS2 instance of MC_MoveAbsolute is executed when this variable changes to TRUE. |
| Mv_Abs3_Ex | BOOL | FALSE | The MV_ABS3 instance of MC_MoveAbsolute is executed when this variable changes to TRUE. |
| Mv_Abs4_Ex | BOOL | FALSE | The MV_ABS4 instance of MC_MoveAbsolute is executed when this variable changes to TRUE. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

## Timing Chart




## Sample Programming

```
// Processing when input parameters are not set
IF InitFlag = FALSE THEN

    // MV_ABS1 parameters
    Mv_Abs1_Pos     := LREAL#290.0;
    Mv_Abs1_Vel     := LREAL#250.0;
    Mv_Abs1_Acc     := LREAL#1000.0;
    Mv_Abs1_Dec     := LREAL#1000.0;
    Mv_Abs1_Dir     := _eMC_DIRECTION#_mcShortestWay;

    // MV_ABS2 parameters
    Mv_Abs2_Pos     := LREAL#90.0;
    Mv_Abs2_Vel     := LREAL#250.0;
    Mv_Abs2_Acc     := LREAL#1000.0;
    Mv_Abs2_Dec     := LREAL#1000.0;
    Mv_Abs2_Dir     := _eMC_DIRECTION#_mcShortestWay;
    Mv_Abs2_Bm      := _eMC_BUFFER_MODE#_mcBuffered;

    // MV_ABS3 parameters
```

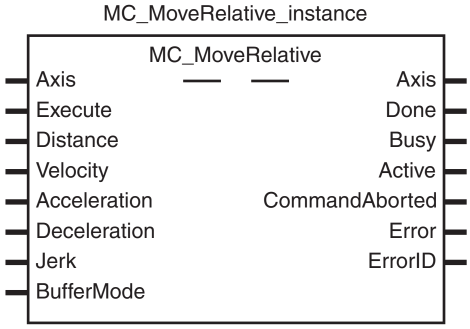
```
        Mv_Abs3_Pos     := LREAL#120.0;
        Mv_Abs3_Vel     := LREAL#250.0;
        Mv_Abs3_Acc     := LREAL#1000.0;
        Mv_Abs3_Dec     := LREAL#1000.0;
        Mv_Abs3_Dir     := _eMC_DIRECTION#_mcShortestWay;
        Mv_Abs3_Bm      := _eMC_BUFFER_MODE#_mcBuffered;

        // MV_ABS4 parameters
        Mv_Abs4_Pos     := LREAL#0.0;
        Mv_Abs4_Vel     := LREAL#250.0;
        Mv_Abs4_Acc     := LREAL#1000.0;
        Mv_Abs4_Dec     := LREAL#1000.0;
        Mv_Abs4_Dir     := _eMC_DIRECTION#_mcShortestWay;
        Mv_Abs4_Bm      := _eMC_BUFFER_MODE#_mcBuffered;

        // Change InitFlag to TRUE after setting the input parameters.
        InitFlag := TRUE;

END_IF;


// If StartPg is TRUE and EtherCAT communications are normal, the Servo is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis000.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
        Pwr_En:=TRUE;
ELSE
        Pwr_En:=FALSE;
END_IF;

// Processing for a minor fault level error
// Implement FaultHandler separately.
IF MC_Axis000.MFaultLvl.Active=TRUE THEN
        FaultHandler();
END_IF;

// If the Servo is ON and home is not defined, the Home instruction is executed.
IF (Pwr_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
        Hm_Ex:=TRUE;
END_IF;

// After home is defined, MV_ABS1 is executed.
IF Hm_D=TRUE THEN
        Mv_Abs1_Ex:=TRUE;
END_IF;

// After MV_ABS1 is started, MV_ABS2 is executed with multi-execution of instructions.
IF Mv_Abs1_Act=TRUE THEN
        Mv_Abs2_Ex:=TRUE;
END_IF;

// After MV_ABS2 is started, MV_ABS3 is executed with multi-execution of instructions.
IF Mv_Abs2_Act=TRUE THEN
        Mv_Abs3_Ex:=TRUE;
END_IF;

// After MV_ABS3 is started, MV_ABS4 is executed with multi-execution of instructions.
IF Mv_Abs3_Act=TRUE THEN
        Mv_Abs4_Ex:=TRUE;
END_IF;

// MC_Power
PWR(
        Axis            := MC_Axis000,
```

```
    Enable      := Pwr_En,
    Status      => Pwr_Status,
    Busy        => Pwr_Bsy,
    Error       => Pwr_Err,
    ErrorID     => Pwr_ErrID
);

// MC_Home
HM(
    Axis                := MC_Axis000,
    Execute             := Hm_Ex,
    Done                => Hm_D,
    Busy                => Hm_Bsy,
    CommandAborted      => Hm_Ca,
    Error               => Hm_Err,
    ErrorID             => Hm_ErrID
);

// Absolute positioning (1)
MV_ABS1(
    Axis                := MC_Axis000,
    Execute             := Mv_Abs1_Ex,
    Position            := Mv_Abs1_Pos,
    Velocity            := Mv_Abs1_Vel,
    Acceleration        := Mv_Abs1_Acc,
    Deceleration        := Mv_Abs1_Dec,
    Direction           := Mv_Abs1_Dir,
    Done                => Mv_Abs1_D,
    Busy                => Mv_Abs1_Bsy,
    Active              => Mv_Abs1_Act,
    CommandAborted      => Mv_Abs1_Ca,
    Error               => Mv_Abs1_Err,
    ErrorID             => Mv_Abs1_ErrID
);


// Absolute positioning (2)
MV_ABS2(
    Axis                := MC_Axis000,
    Execute             := Mv_Abs2_Ex,
    Position            := Mv_Abs2_Pos,
    Velocity            := Mv_Abs2_Vel,
    Acceleration        := Mv_Abs2_Acc,
    Deceleration        := Mv_Abs2_Dec,
    Direction           := Mv_Abs2_Dir,
    BufferMode          := Mv_Abs2_Bm,
    Done                => Mv_Abs2_D,
    Busy                => Mv_Abs2_Bsy,
    Active              => Mv_Abs2_Act,
    CommandAborted      => Mv_Abs2_Ca,
    Error               => Mv_Abs2_Err,
    ErrorID             => Mv_Abs2_ErrID
);


// Absolute positioning (3)
MV_ABS3(
    Axis                := MC_Axis000,
    Execute             := Mv_Abs3_Ex,
    Position            := Mv_Abs3_Pos,
    Velocity            := Mv_Abs3_Vel,
    Acceleration        := Mv_Abs3_Acc,
    Deceleration        := Mv_Abs3_Dec,
    Direction           := Mv_Abs3_Dir,
    BufferMode          := Mv_Abs3_Bm,
```

```
    Done              => Mv_Abs3_D,
    Busy              => Mv_Abs3_Bsy,
    Active            => Mv_Abs3_Act,
    CommandAborted    => Mv_Abs3_Ca,
    Error             => Mv_Abs3_Err,
    ErrorID           => Mv_Abs3_ErrID
);


// Absolute positioning (4)
MV_ABS4(
    Axis              := MC_Axis000,
    Execute           := Mv_Abs4_Ex,
    Position          := Mv_Abs4_Pos,
    Velocity          := Mv_Abs4_Vel,
    Acceleration      := Mv_Abs4_Acc,
    Deceleration      := Mv_Abs4_Dec,
    Direction         := Mv_Abs4_Dir,
    BufferMode        := Mv_Abs4_Bm,
    Done              => Mv_Abs4_D,
    Busy              => Mv_Abs4_Bsy,
    Active            => Mv_Abs4_Act,
    CommandAborted    => Mv_Abs4_Ca,
    Error             => Mv_Abs4_Err,
    ErrorID           => Mv_Abs4_ErrID
);
```

# MC_MoveRelative

The MC_MoveRelative instruction moves the axis the specified travel distance from the command current position.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_MoveRelative | Relative Positioning | FB |  | MC_MoveRelative_instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>Distance :=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>BufferMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |



## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Distance | Travel Distance | LREAL | Negative number, positive number, or 0 | 0 | Specify the travel distance from the command current position. The unit is command units.*1 |
| Velocity | Target Velocity | LREAL | Positive number | 0 | Specify the target velocity.*2 The unit is command units/s.*1 |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/$s^2$.*1 |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.*1 |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/$s^3$.*1 |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| BufferMode | Buffer Mode Selection | _eMC_BUFFER_MODE | 0: _mcAborting<br>1: _mcBuffered<br>2: _mcBlendingLow<br>3: _mcBlendingPrevious<br>4: _mcBlendingNext<br>5: _mcBlendingHigh | 0[*3] | Specify the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered<br>2: Blend with low<br>3: Blend with previous<br>4: Blend with next<br>5: Blend with high |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 Always set the target velocity. If the axis is moved without setting a target velocity, an error will occur.

*3 The default value for an enumeration variable is actually not the number, but the enumerator.



## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When positioning is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the instruction is started. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

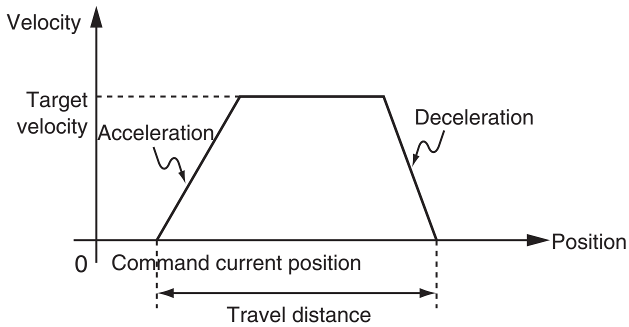
| Name | Meaning | Data type | Valid Ranges | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

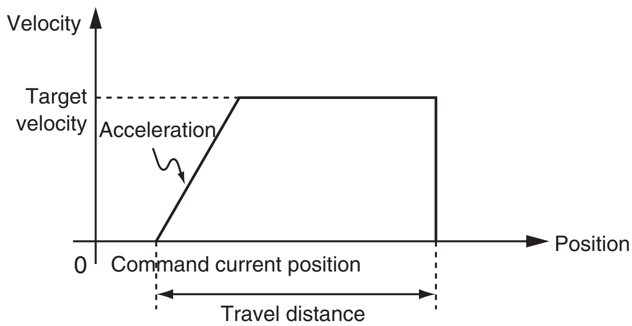
## Function

- The travel distance from the command current position is specified to perform positioning.
- Relative positioning starts when *Execute* changes to TRUE.
- You can specify the *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), *Deceleration* (Deceleration Rate), and *Jerk* as input variables.

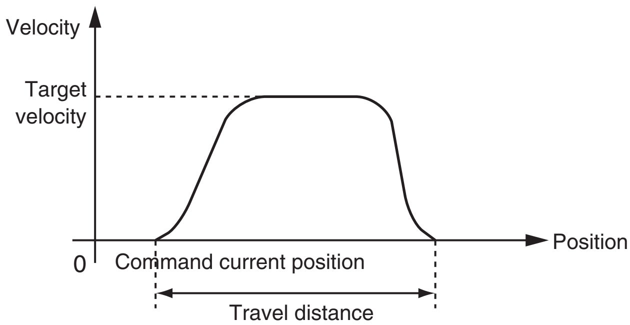
The following chart shows an operation example of relative positioning.




When the acceleration rate or deceleration rate is 0 and the instruction is executed, it will reach the target velocity without accelerating or decelerating.
The following chart shows an operation example of when the acceleration rate is 0.



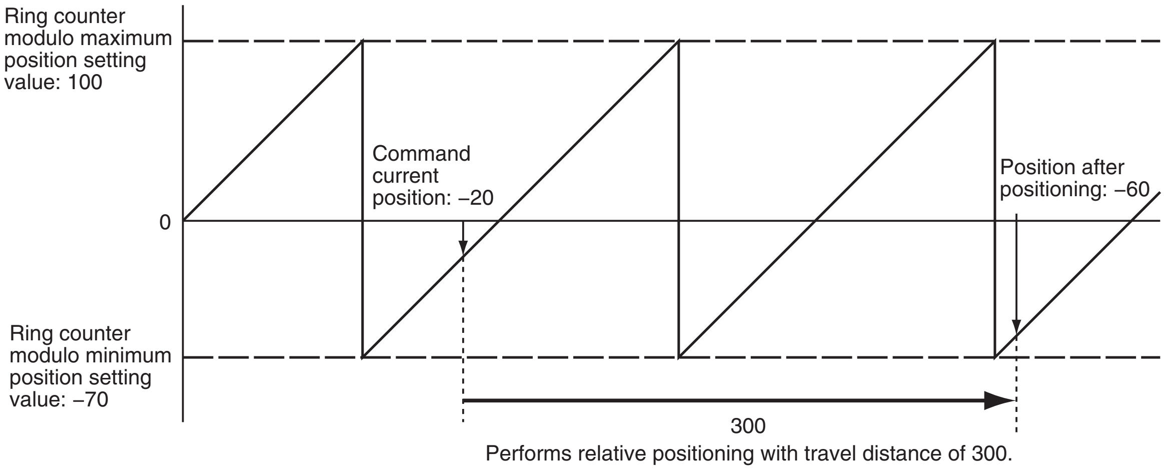

Specify *Jerk* when you want to accelerate or decelerate smoothly. The following chart shows an operation example when *Jerk* is specified.




For details on *Jerk*, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Instruction Details

When the Count Mode is Rotary Mode, you can specify a Distance (Travel Distance) that exceeds the relative distance range from the ring counter modulo minimum position setting value to the ring counter modulo maximum position setting value of the ring counter in the axis parameters so that you can perform multiple ring rotation positioning. The following chart shows an operation example when the command current position is −20 and *Distance* (Travel Distance) is 300.

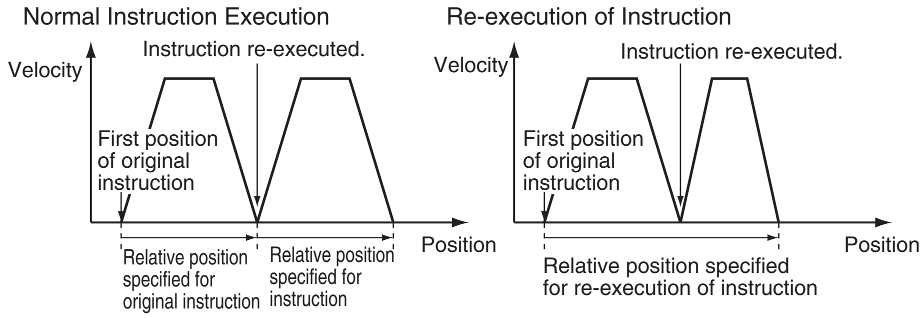



When *Distance* (Travel Distance) is 0 and the instruction is executed, the axis will not move, but *Done* will change to TRUE.

**Precautions for Correct Use**

Observe the following precautions if you re-execute relative positioning just before the completion of positioning. If positioning is completed before the MC Function Module re-executes the instruction, normal instruction execution is performed.

- For normal instruction execution, positioning is performed to the relative value that is based on the position of the axis when the instruction is executed.
- For re-execution of an instruction, positioning is performed to the relative value that is based on the position of the axis when original instruction was executed.




### ● In-position Check

An in-position check is performed for this instruction according to the settings in In-position Range and In-position Check Time axis parameters.

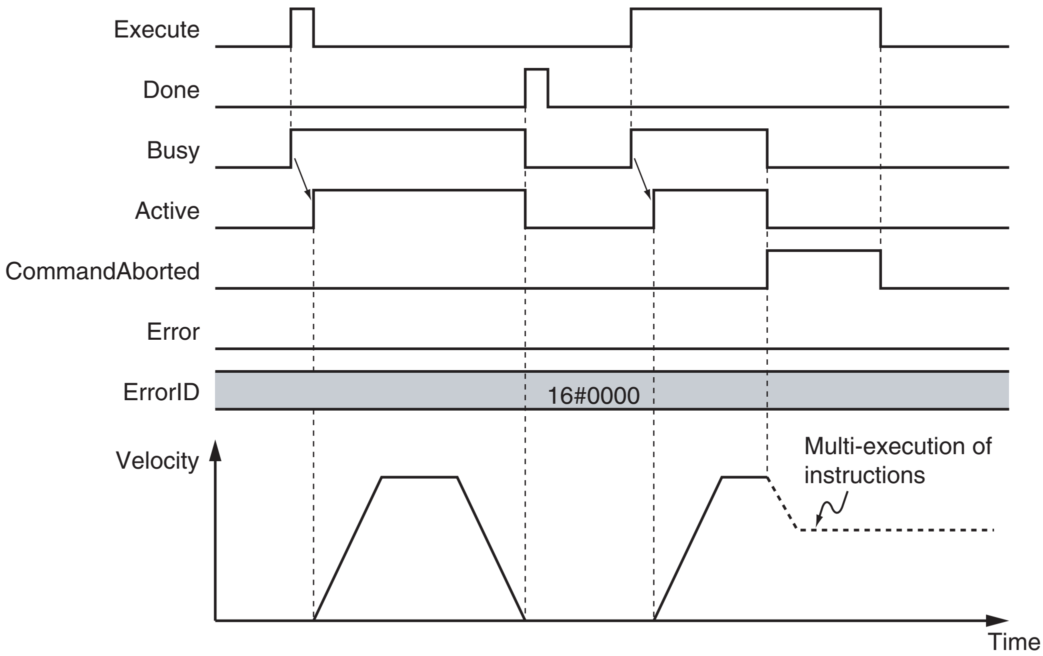
MC_MoveRelative

3

Function

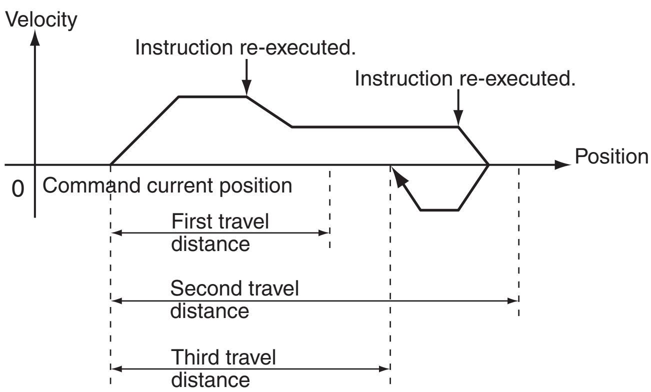
## Timing Charts

- *Busy* (Executing) changes to TRUE at the same time as *Execute* changes to TRUE. *Active* (Controlling) changes to TRUE in the next period.
- *Done* changes to TRUE when *Distance* (Target Distance) is reached and positioning is completed.
- If another instruction aborts this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing) and *Active* (Controlling) change to FALSE.



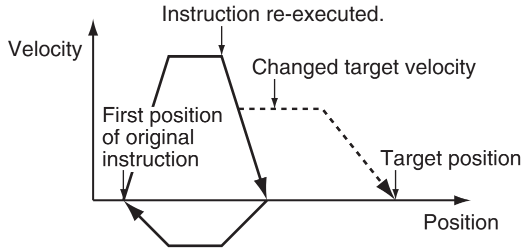

## Re-execution of Motion Control Instructions

You can change the operation of the instruction if you change an input variable during positioning and change *Execute* to TRUE again.
Input variables *Distance* (Travel Distance), *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate) can be changed by re-executing the motion control instruction. The starting point for *Distance* (Travel Distance) when the instruction is re-executed is not the command current position for the first execution, but rather it is the command current position when the instruction was first executed. The following chart shows an operation example when a motion control instruction is re-executed twice with different values for *Distance* (Travel Distance) and *Velocity* (Target Velocity).



**Precautions for Correct Use**

To change any input parameter other than *Distance* (Travel Distance), re-execute the instruction with *Distance* (Travel Distance) assigned to the same value as the original instruction.
If *Distance* (Travel Distance) is omitted when the instruction is re-executed, the re-executed instruction will use a *Distance* (Travel Distance) of 0. This will cause the axis to return to the original first position, as shown below.




For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

You can switch to this instruction or buffer this instruction if you execute it during execution of another instruction. You can buffer one instruction per axis.
The starting point for *Distance* (Travel Distance) for multi-execution of the motion instruction is the command current position when *Active* (Controlling) changes to TRUE after the start of instruction execution.
Specify the operation of this instruction using *BufferMode* (Buffer Mode Selection) for multi-execution of instructions.

| Buffer Mode Selection | | Description |
|---|---|---|
| Aborting | | Aborts the instruction being executed and switches to this instruction.<br>If the direction of axis motion is reversed by switching instructions, the motion will be reversed according to the Operation at Reversing axis parameter. |
| Buffered | | Buffers this instruction and executes it automatically after the current instruction is completed. |
| Blending | | Starts the buffered instruction at the velocity (transit velocity) at which the current instruction reaches the target position. The operation of the current instruction is changed so that the axes reach the target position at the transit velocity. There are four methods to specify the transit velocity. These are described below. |
| | Blend with low | The lower of the target velocities of the current instruction and the buffered instruction is used as the transit velocity. |
| | Blend with previous | The target velocity of the current instruction is used as the transit velocity. |
| | Blend with next | The target velocity of the buffered instruction is used as the transit velocity. |
| | Blend with high | The higher of the target velocities of the current instruction and the buffered instruction is used as the transit velocity. |

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

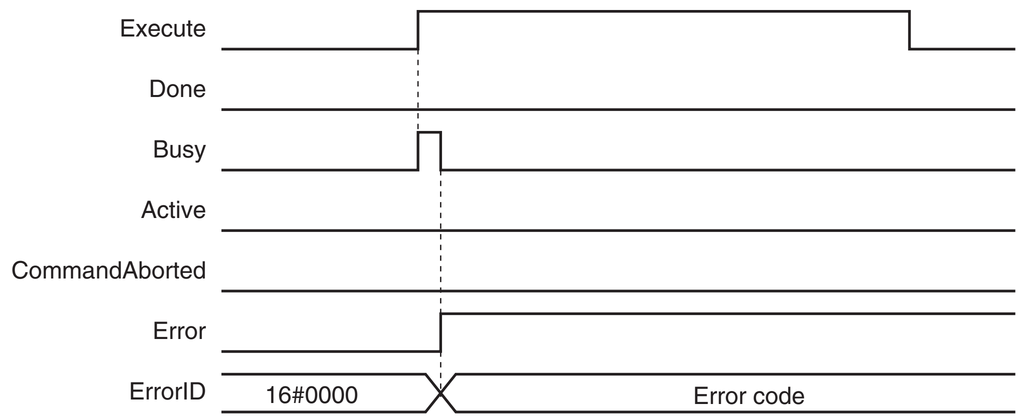
### Execution of Other Instructions during Instruction Execution

If you execute another instruction during execution of this instruction, you can specify aborting, buffering, or blending.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

### Timing Chart When Error Occurs

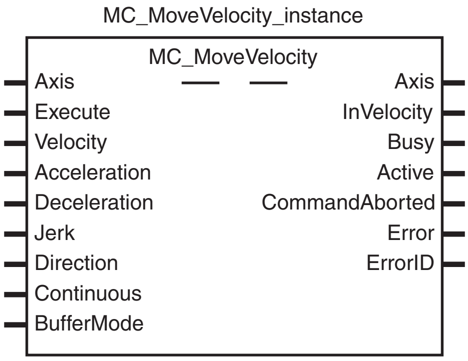



### Error code

Refer to *A-1 Error Codes* for instruction errors.

# MC_MoveVelocity

The MC_MoveVelocity instruction performs velocity control with the Position Control Mode of the Servo Drive.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_MoveVelocity | Velocity Control | FB |  | MC_MoveVelocity_instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>Direction :=*parameter,*<br>Continuous :=*parameter,*<br>BufferMode :=*parameter,*<br>InVelocity =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Velocity | Target Velocity | LREAL | Non-negative number | 0 | Specify the target velocity.[*1]<br>The unit is command units/s.[*2] |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/$s^2$.[*2] |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.[*2] |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/$s^3$.[*2] |
| Direction | Direction | _eMC_DIRECTION | 0: _mcPositiveDirection<br>2: _mcNegativeDirection<br>3: _mcCurrentDirection | 0[*3] | Specify the rotation direction.<br>0: Positive direction<br>2: Negative direction<br>3: Current direction |
| BufferMode | Buffer Mode Selection | e_MC_BUFFER_MODE | 0: _mcAborting<br>1: _mcBuffered | 0[*3] | Specify the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Continuous (Reserved) | Continuation Mode Selection | BOOL | TRUE or FALSE | --- | Reserved |

*1 Always set the target velocity. If the axis is moved without setting a target velocity, an error will occur.

*2 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*3 The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| InVelocity | Target Velocity Reached | BOOL | TRUE or FALSE | TRUE when the target velocity is reached. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| InVelocity | When the target velocity is reached. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE.<br>• When the instruction is re-executed and the target velocity is changed. |
| Busy | When *Execute* changes to TRUE. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the instruction is started. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting* or *Buffered*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

- Pseudo velocity control is performed with position control.
- The velocity control operation starts when *Execute* changes to TRUE.

## Instruction Details

This section describes the instruction in detail.

### ● *Direction*

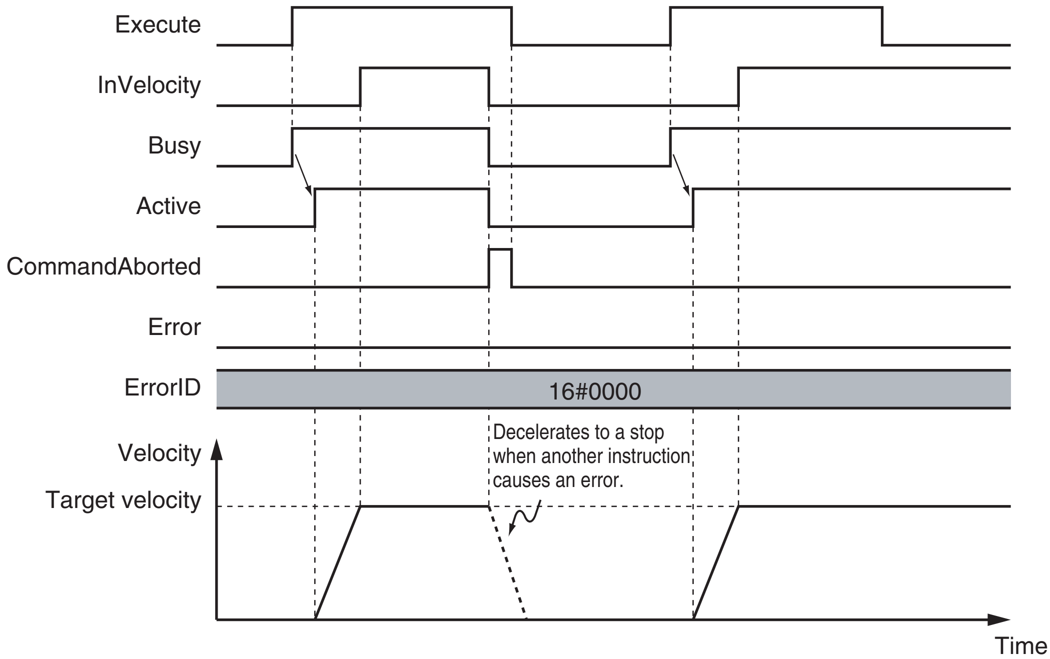
Specify the travel direction with *Direction*. When *Direction* specifies the positive direction, the axis moves in the positive direction. When it specifies the negative direction, the axis moves in the negative direction. When it specifies the current direction, the axis motion depends on whether the axis is stopped or not. If the axis is stopped, it will move in the direction in which it was traveling previously. If the power was turned ON or after restarting, the axis moves in the positive direction. If you execute this instruction during multi-execution of motion control instructions for the axis, the axis will move in the direction that it is currently traveling.

**Precautions for Correct Use**

If the current direction is specified for *Direction*, operation is in the same command direction as the previous motion. Therefore, depending on the instructions that are used together, the direction may not be the same as the direction that was specified with the input to the motion control instruction for the previous motion. When you specify the current direction, check the current direction with *Dir.Posi* (Positive Direction) and *Dir.Nega* (Negative Direction) in the Axis Variable.

## Timing Charts

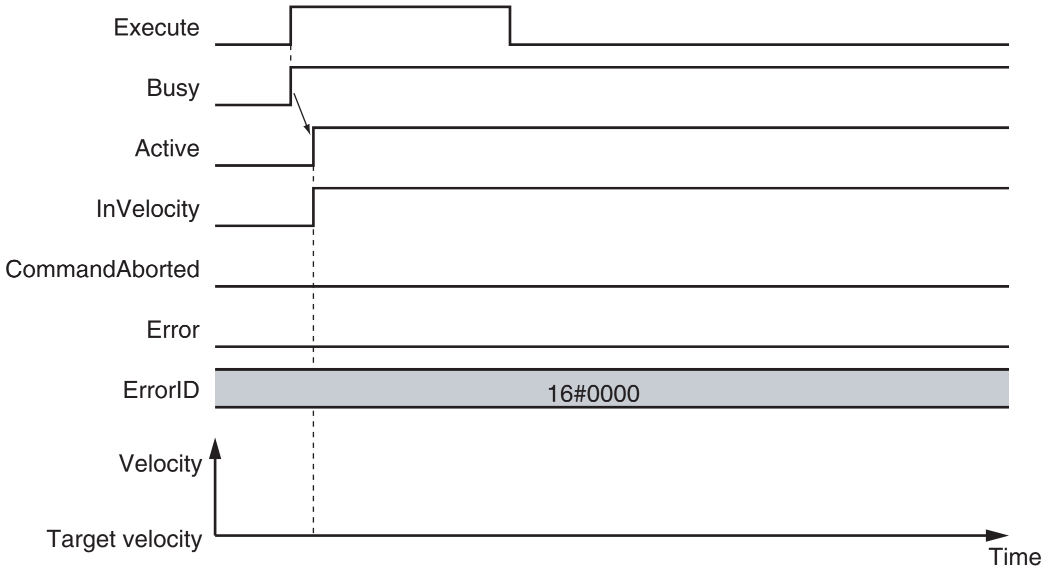
- *Busy* (Executing) changes to TRUE at the same time as *Execute* changes to TRUE. *Active* (Controlling) changes to TRUE in the next period.
- *InVelocity* (Target Velocity Reached) changes to TRUE when *Velocity* (Target Velocity) is reached.
- If another instruction aborts this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing), *Active* (Controlling), and *InVelocity* (Target Velocity Reached) change to FALSE.




The *InVelocity* (Target Velocity Reached) output variable indicates when the velocity has reached the same velocity for this instruction and the re-executed motion control instruction. Therefore, after *InVelocity* (Target Velocity Reached) changes to TRUE, even if the velocity is changed by the override factor, *InVelocity* (Target Velocity Reached) will not change to FALSE. If the override factor changes before *InVelocity* (Target Velocity Reached) changes to TRUE, *InVelocity* (Target Velocity Reached) will change to TRUE when the new target velocity is reached.

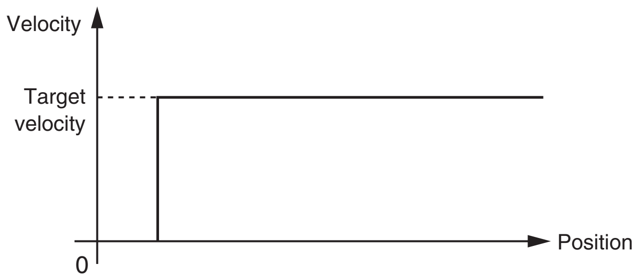
You can specify *Acceleration* (Acceleration Rate), *Deceleration* (Deceleration Rate) and *Jerk* as input variables. When the *Velocity* (Target Velocity) is 0 and the instruction is executed, the axis will enter continuous operation without motion.

The following chart shows an operation example of when *Velocity* (Target Velocity) is 0.

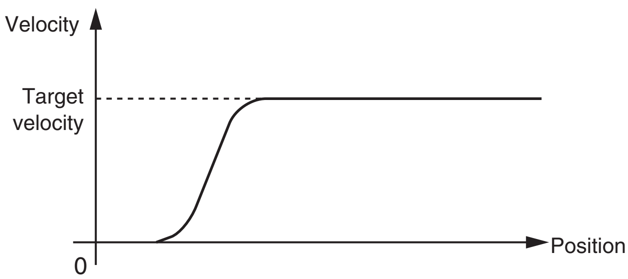



When the *Acceleration* (Acceleration Rate) or *Deceleration* (Deceleration Rate) is 0 and the instruction is executed, the axis will reach the target velocity without accelerating or decelerating.

The following chart shows an operation example of when the acceleration rate is 0.




Specify *Jerk* when you want to accelerate or decelerate smoothly. The following chart shows an operation example when *Jerk* is specified.




For details on *Jerk*, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Re-execution of Motion Control Instructions

You can change the operation of the instruction if you change an input parameter during continuous operation and then change *Execute* to TRUE again. Input variables *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate) can be changed by re-executing the motion control instruction. When changing the *Velocity* (Target Velocity) by re-executing a motion control instruction, *InVelocity* (Target Velocity Reached) changes to FALSE when *Execute* changes to FALSE.
When the motion control instruction is re-executed to change *Velocity* (Target Velocity), *InVelocity* (Target Velocity Reached) operates for the new target velocity that was set at re-execution.

For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

You can switch to this instruction or buffer this instruction if you execute it during execution of another instruction. You can buffer one instruction per axis. Specify the operation of this instruction using *BufferMode* (Buffer Mode Selection) for multi-execution of instructions.

| Buffer Mode Selection | Description |
|---|---|
| Aborting | Aborts the instruction being executed and switches to this instruction.<br>If the direction of axis motion is reversed by switching instructions, the motion will be reversed according to the Operation at Reversing axis parameter. |
| Buffered | Buffers this instruction and executes it automatically after the current instruction is completed. |

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

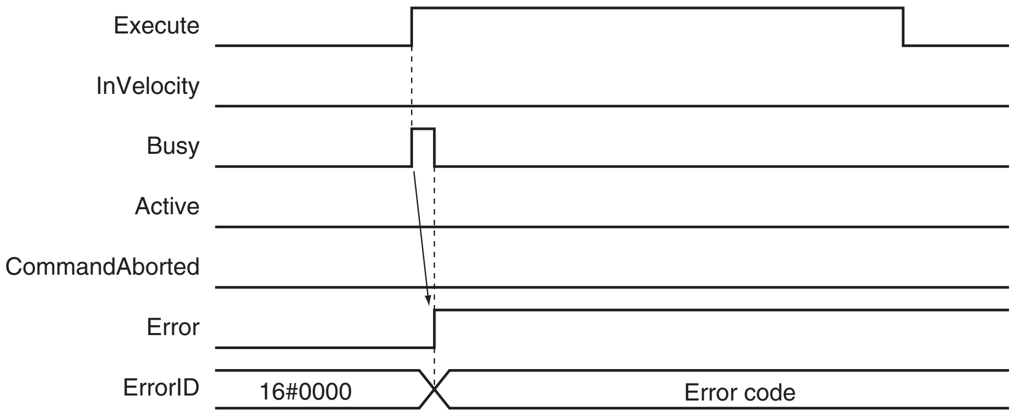
### ● Execution of Other Instructions during Instruction Execution

If another instruction is executed during execution of this instruction, the *BufferMode* input variable to the other instruction must be set to Aborting or *Buffered*.
If you specify Buffered, the buffered instruction is executed when the *InVelocity* (Target Velocity Reached) output variable from this instruction changes to TRUE.

## Error

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

### ● Timing Chart When Error Occurs




### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# Sample Programming

This section shows sample programming for velocity control, such as for a spinner or centrifuge.

## Parameter Settings

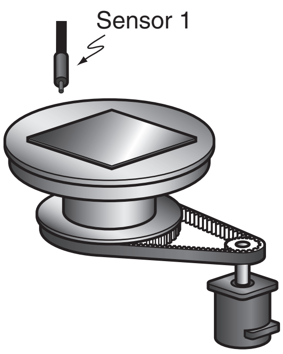
The minimum settings required for this sample programming are given below.

### ● Setting Axis Parameters

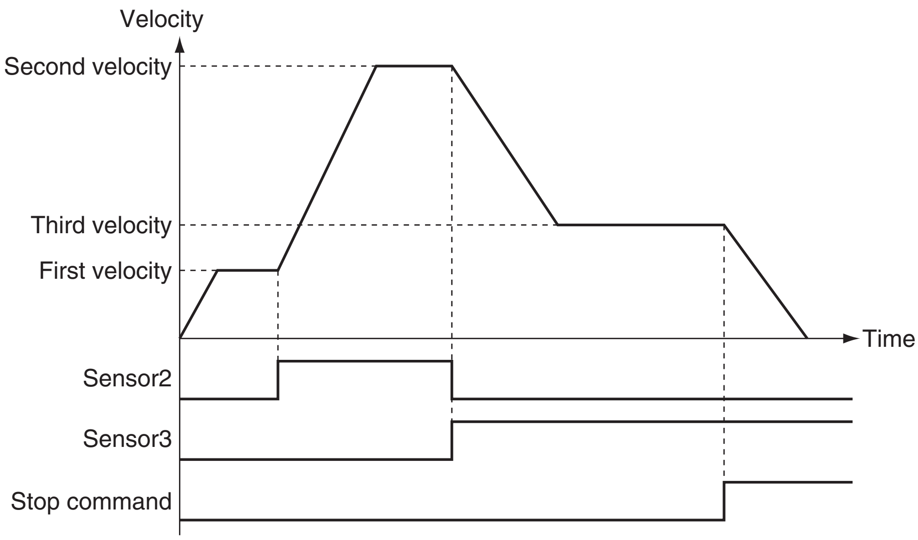
#### Axis Type

| Axis | Axis Type |
|---|---|
| Axis 1 | Servo axis |

## Operation Example




### ● Operation Pattern




**1** Starting Velocity Control

Sensor 1 detects the insertion of liquid chemical. When it turns ON, velocity control starts for axis 1.

**2** Changing to the Second Velocity

When the *Sensor2* bit changes to TRUE, the override factor is set to 500% and the velocity is changed.

**3** Changing to the Third Velocity

When the *Sensor3* bit changes to TRUE, the override factor is set to 200% and the velocity is changed. If both Sensor2 and Sensor3 are TRUE at the same time, the override factor is 200%.

**4** Stopping Velocity Control

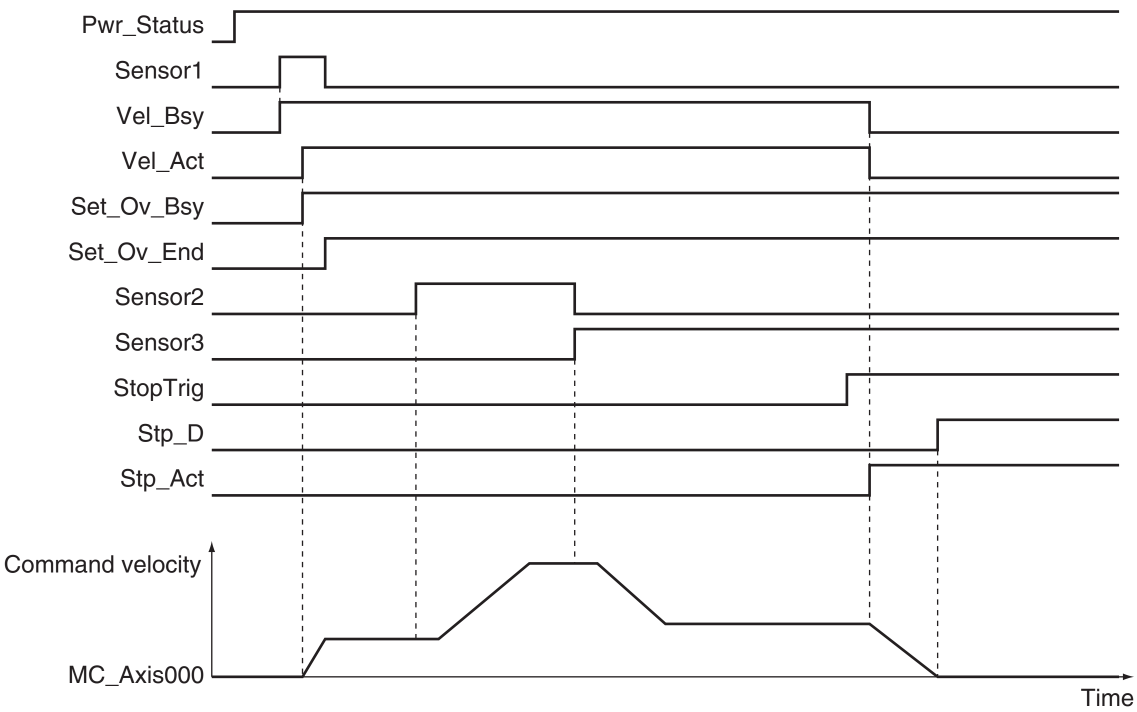
When the stop command (*StopTrig*) changes to TRUE, the axis decelerates to a stop.

## Ladder Diagram

### ● Main Variables

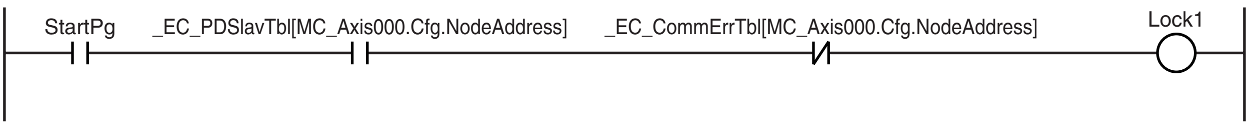
| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| Pwr_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| Vel_Act | BOOL | FALSE | This variable is assigned to the *Active* output variable from the VEL instance of the MC_MoveVelocity instruction. It is TRUE during velocity control by the VEL instance. |
| Set_Ov_Velfct | LREAL | 0 | This is the velocity override factor. |
| StopTrig | BOOL | FALSE | When this variable is TRUE, MC_Stop is executed. |
| Sensor1 | BOOL | FALSE | TRUE when chemical solution supply is detected. If the Servo is ON for axis 1, the MC_MoveVelocity (Velocity Control) instruction is executed. |
| Sensor2 | BOOL | FALSE | If this variable is TRUE, the override factor is set to 500%. After this variable changes to TRUE, it remains TRUE until *Sensor3* changes to TRUE, at which time it changes to FALSE. |
| Sensor3 | BOOL | FALSE | If this variable is TRUE, the override factor is set to 200%. After this variable changes to TRUE, it remains TRUE. |

● **Timing Chart**

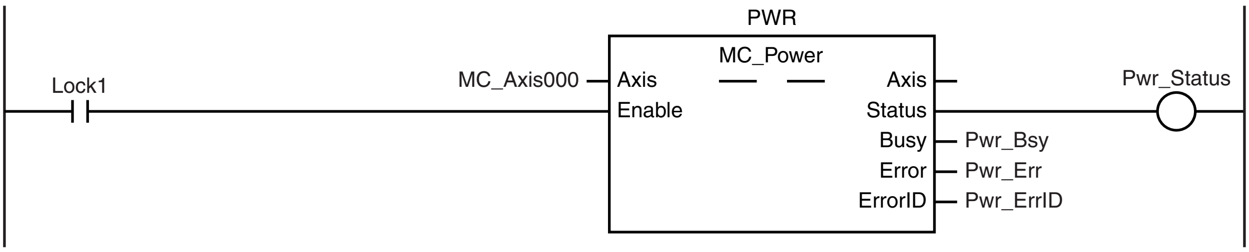



● **Sample Programming**

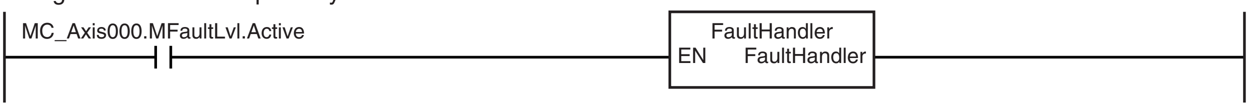
If *StartPg* is TRUE, EtherCAT communications are checked to see if process data communications are normal.



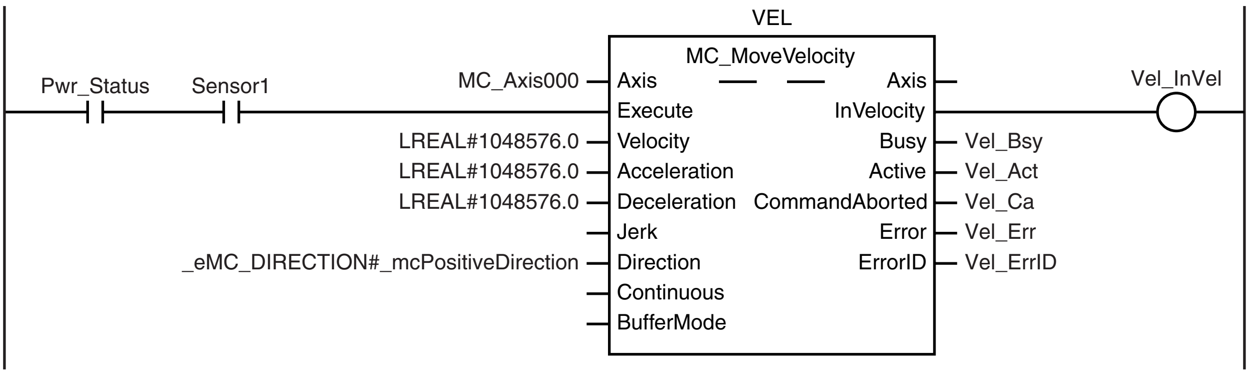

If process data communications are active, the Servo is turned ON.



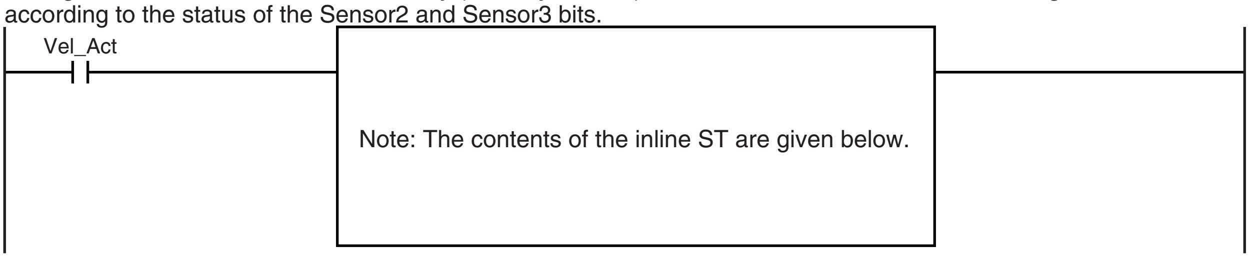

If there is a minor fault level error for axis 1, the error handler for the device (FaultHandler) is executed.
Design FaultHandler separately.



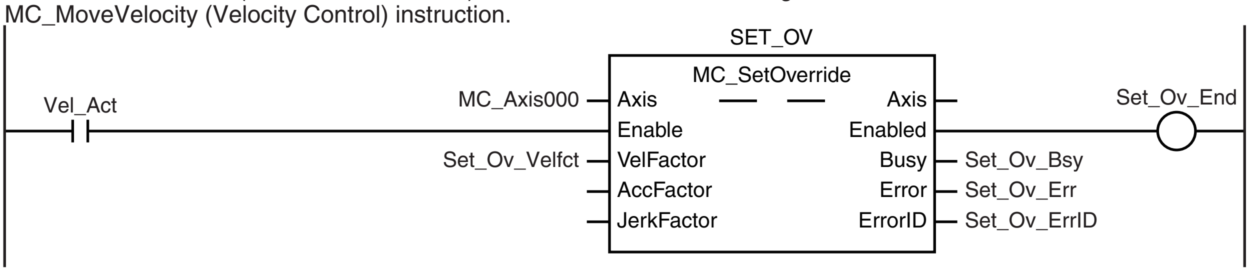

When *Sensor1* changes to TRUE, the MC_MoveVelocity (Velocity Control) instruction is executed.



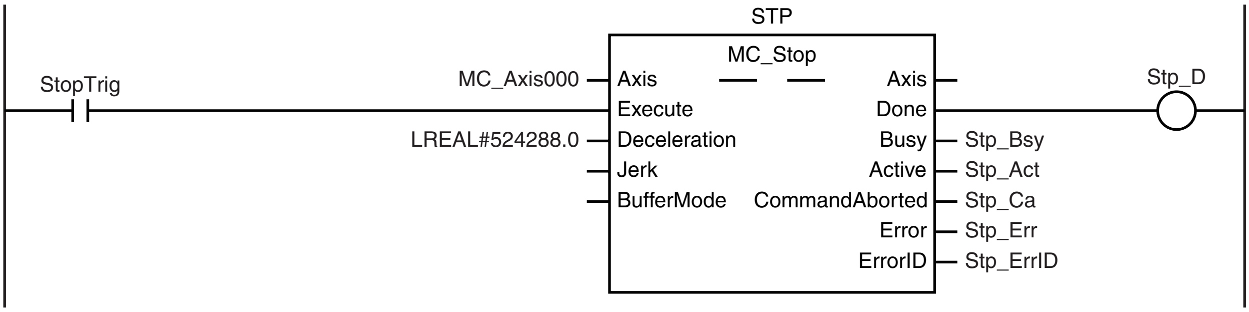
During execution of the MC_MoveVelocity (Velocity Control) instruction, the override factor is changed according to the status of the Sensor2 and Sensor3 bits.




The MC_SetOverride (Set Override Factors) instruction is executed during execution of the MC_MoveVelocity (Velocity Control) instruction.




When the stop command (*StopTrig*) changes to TRUE, the MC_Stop instruction is executed.




### Contents of Inline ST

```
IF (Sensor2=FALSE) AND (Sensor3=FALSE) THEN
        Set_Ov_Velfct := LREAL#100.0;
ELSIF (Sensor2=TRUE) AND (Sensor3=FALSE) THEN
        Set_Ov_Velfct := LREAL#500.0;
ELSIF (Sensor2=FALSE) AND (Sensor3=TRUE) THEN
        Set_Ov_Velfct := LREAL#200.0;
ELSE
        Set_Ov_Velfct := LREAL#200.0;
END_IF;
```

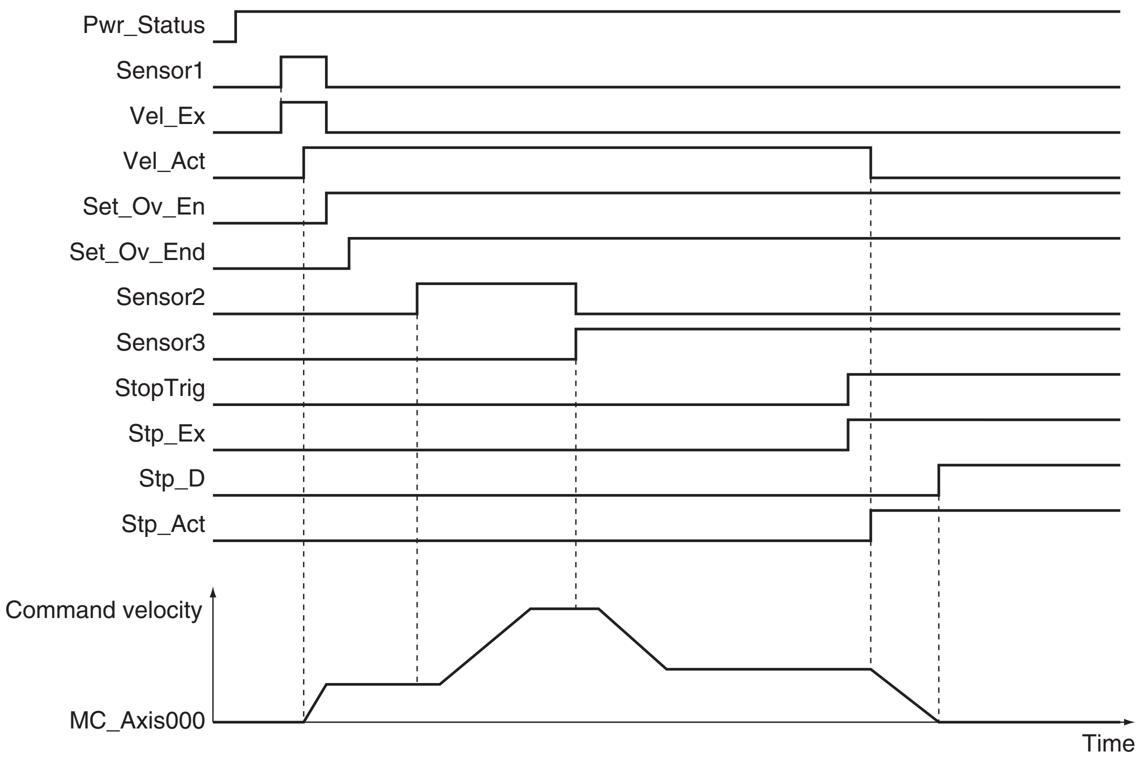
## Structured Text (ST)

### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| Pwr_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |

| Name | Data type | Default | Comment |
|---|---|---|---|
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| Vel_Act | BOOL | FALSE | This variable is assigned to the *Active* output variable from the VEL instance of the MC_MoveVelocity instruction. It is TRUE during velocity control by the VEL instance. |
| Set_Ov_Velfct | LREAL | 0 | This is the velocity override factor. |
| StopTrig | BOOL | FALSE | When this variable is TRUE, MC_Stop is executed. |
| Sensor1 | BOOL | FALSE | TRUE when chemical solution supply is detected. If the Servo is ON for axis 1, the MC_MoveVelocity (Velocity Control) instruction is executed. |
| Sensor2 | BOOL | FALSE | If this variable is TRUE, the override factor is set to 500%. After this variable changes to TRUE, it remains TRUE until *Sensor3* changes to TRUE, at which time it changes to FALSE. |
| Sensor3 | BOOL | FALSE | If this variable is TRUE, the override factor is set to 200%. After this variable changes to TRUE, it remains TRUE. |
| Vel_Ex | BOOL | FALSE | The VEL instance of MC_MoveVelocity is executed when this variable changes to TRUE. |
| Set_Ov_En | BOOL | FALSE | The SET_OV instance of MC_SetOverride is executed while this variable is TRUE. |
| Stp_Ex | BOOL | FALSE | The STP instance of MC_Stop is executed when this variable changes to TRUE. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

## ● Timing Chart




## ● Sample Programming

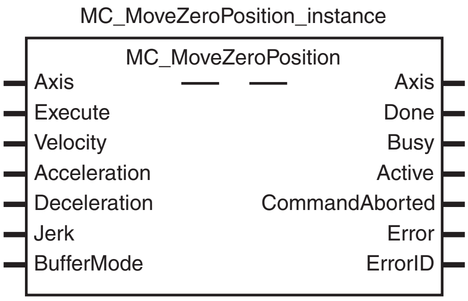
```
// Processing when input parameters are not set
IF InitFlag = FALSE THEN
     // MC_MoveVelocity parameters
     Vel_Vel      := LREAL#1048576.0;
     Vel_Acc      := LREAL#1048576.0;
     Vel_Dec      := LREAL#1048576.0;
     Vel_Dir      := _eMC_DIRECTION#_mcPositiveDirection;

     // MC_SetOverride parameters
     Set_Ov_Velfct := LREAL#100.0;

     // MC_Stop parameters
     Stp_Dec := LREAL#524288.0;


     // Change InitFlag to TRUE after setting the input parameters.
     InitFlag:=TRUE;

END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 1 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis000.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
     Pwr_En:=TRUE;
ELSE
     Pwr_En:=FALSE;
END_IF;

// If a minor fault level error occurs for axis 1, the error handler for the device (FaultHandler) is executed.
// Implement FaultHandler separately.
IF MC_Axis000.MFaultLvl.Active=TRUE THEN
     FaultHandler();
```

```
END_IF;

// If the Servo is ON for axis 1 and the Sensor1 bit is TRUE, the MC_MoveVelocity instruction is executed.
IF (Pwr_Status=TRUE) AND (Sensor1=TRUE) THEN
    Vel_Ex := TRUE;
END_IF;


// During execution of the MC_MoveVelocity instruction, the override factor is changed according to the status of
the Sensor2 and Sensor3 bits.
IF Vel_Act=TRUE THEN
    IF (Sensor2=FALSE) AND (Sensor3=FALSE) THEN
        Set_Ov_Velfct := LREAL#100.0;
    ELSIF (Sensor2=TRUE) AND (Sensor3=FALSE) THEN
        Set_Ov_Velfct := LREAL#500.0;
    ELSIF (Sensor2=FALSE) AND (Sensor3=TRUE) THEN
        Set_Ov_Velfct := LREAL#200.0;
    ELSE
        Set_Ov_Velfct := LREAL#200.0;
    END_IF;
END_IF;



// The MC_SetOverride instruction is executed during velocity control for the MC_MoveVelocity instruction.
IF Vel_Act=TRUE THEN
    Set_Ov_En := TRUE;
END_IF;

// The MC_Stop instruction is executed when StopTrig is TRUE.
IF StopTrig=TRUE THEN
    Stp_Ex := TRUE;
END_IF;


//MC_Power
PWR(
    Axis        := MC_Axis000,
    Enable      := Pwr_En,
    Status      => Pwr_Status,
    Busy        => Pwr_Bsy,
    Error       => Pwr_Err,
    ErrorID     => Pwr_ErrID
);

//MC_MoveVelocity
VEL(
    Axis                := MC_Axis000,
    Execute             := Vel_Ex,
    Velocity            := Vel_Vel,
    Acceleration        := Vel_Acc,
    Deceleration        := Vel_Dec,
    Direction           := Vel_Dir,
    InVelocity          => Vel_Invel,
    Busy                => Vel_Bsy,
    Active              => Vel_Act,
    CommandAborted      => Vel_Ca,
    Error               => Vel_Err,
    ErrorID             => Vel_ErrID
);
```

```
//MC_SetOverride
SET_OV(
    Axis            := MC_Axis000,
    Enable          := Set_Ov_En,
    VelFactor       := Set_Ov_Velfct,
    AccFactor       := Set_Ov_Accfct,
    JerkFactor      := Set_Ov_Jfct,
    Busy            => Set_Ov_Bsy,
    Enabled         => Set_Ov_End,
    Error           => Set_Ov_Err,
    ErrorID         => Set_Ov_ErrID
);

//MC_Stop
STP(
    Axis                := MC_Axis000,
    Execute             := Stp_Ex,
    Deceleration        := Stp_Dec,
    Done                => Stp_D,
    Busy                => Stp_Bsy,
    Active              => Stp_Act,
    CommandAborted      => Stp_Ca,
    Error               => Stp_Err,
    ErrorID             => Stp_ErrID
);
```

# MC_MoveZeroPosition

The MC_MoveZeroPosition instruction performs positioning with an absolute position of 0 as the target position to return to home.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_MoveZeroPosition | High-speed Home | FB |  | MC_MoveZeroPosition_instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>BufferMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Velocity | Target Variable | LREAL | Positive number | 0 | Specify the target velocity.*1 The unit is command units/s.*2 |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/$s^2$.*2 |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.*2 |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/$s^3$.*2 |
| BufferMode | Buffer Mode Selection | _eMC_BUFFER_ MODE | 0: _mcAborting<br>1: _mcBuffered | 0*3 | Specify the behavior for multi-execution of instructions.<br>0: Aborting<br>1: Buffered |

*1 Always set the target velocity. If the axis is moved without setting a target velocity, an error will occur.

*2 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*3 The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When positioning is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the axis starts moving. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

## Function

- The axis moves to home.
- You can specify the *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), *Deceleration* (Deceleration Rate), and *Jerk* as input variables.

**Precautions for Correct Use**

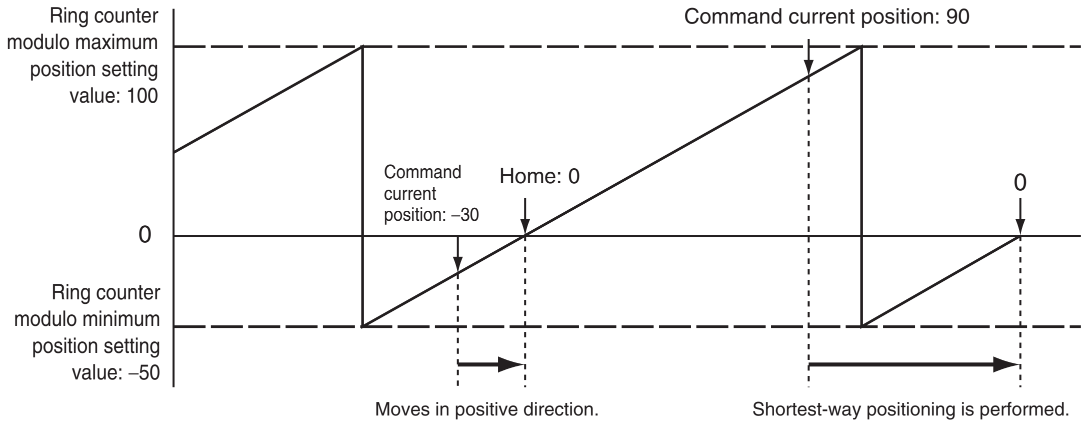
Execute the MC_MoveZeroPosition (High-speed Home) instruction only after defining home. If home is not defined, an Instruction Execution Error with Undefined Home (error code: 5466 hex) occurs.

## Instruction Details

This section describes the instruction in detail.

### ● Direction Designation

When the Count Mode is set to Rotary Mode, positioning is performed in the direction with the shortest distance to home (shortest-way positioning). The following chart shows an operation example of this instruction according to the command current position.




**Precautions for Correct Use**

If this instruction is executed when home is outside of the settings of the Modulo Maximum Position Setting Value and Modulo Minimum Position Setting Value axis parameters, a Target Position Ring Counter Out of Range error (error code: 549C hex) will occur.

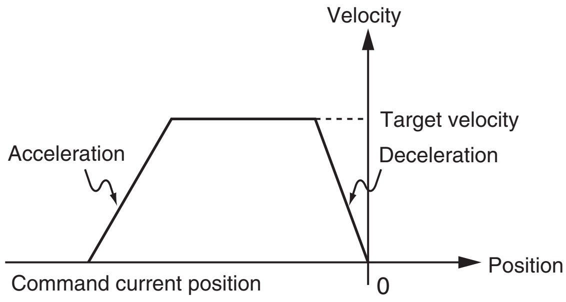
### ● Override Factors

Override factors are enabled for this instruction.

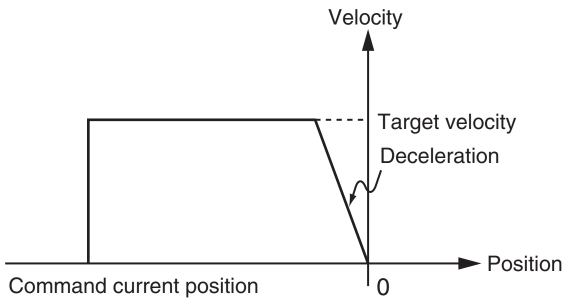
### ● In-position Check

An in-position check is performed for this instruction according to the settings in In-position Range and In-position Check Time axis parameters.

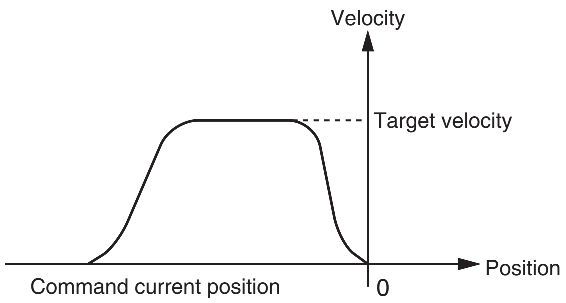
- **Operation Example**




When the acceleration rate or deceleration rate is 0 and the instruction is executed, it will reach the target velocity without accelerating or decelerating.
The following chart shows an operation example of when the acceleration rate is 0.



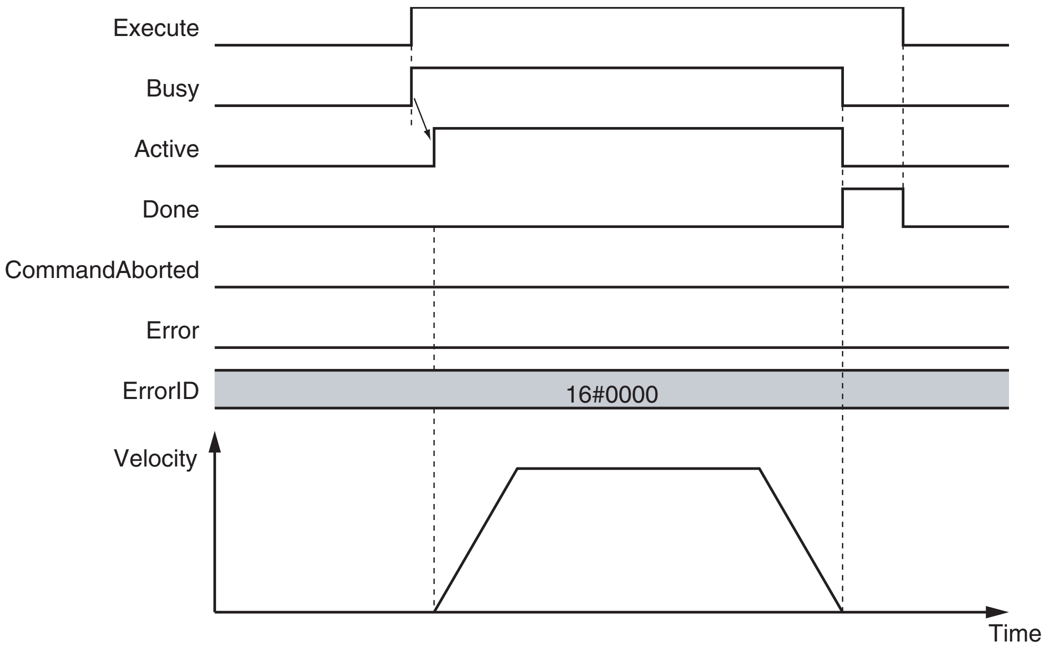

Specify *Jerk* when you want to accelerate or decelerate smoothly. The following chart shows an operation example when *Jerk* is specified.




For details on *Jerk*, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Timing Charts

A timing chart for execution of the MC_MoveZeroPosition (High-speed Home) instruction is shown below.




## Aborting the Instruction

Home will not become undefined even if this instruction is aborted and *CommandAborted* changes to TRUE.

## Re-execution of Motion Control Instructions

You can change the operation of the instruction if you change an input variable during positioning and change *Execute* to TRUE again. Input variables *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate) can be changed by re-executing the motion control instruction. For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### Execution during Execution of Other Instructions

You can switch to this instruction or buffer this instruction if you execute it during execution of another instruction. You can buffer one instruction per axis. Specify the operation of this instruction using *BufferMode* (Buffer Mode Selection) for multi-execution of instructions.

| Buffer Mode Selection | Description |
|---|---|
| Aborting | Aborts the instruction being executed and switches to this instruction. a<br>If the direction of axis motion is reversed by switching instructions, the motion will be reversed according to the Operation at Reversing axis parameter. |
| Buffered | Buffers this instruction and executes it automatically after the current instruction is completed. |

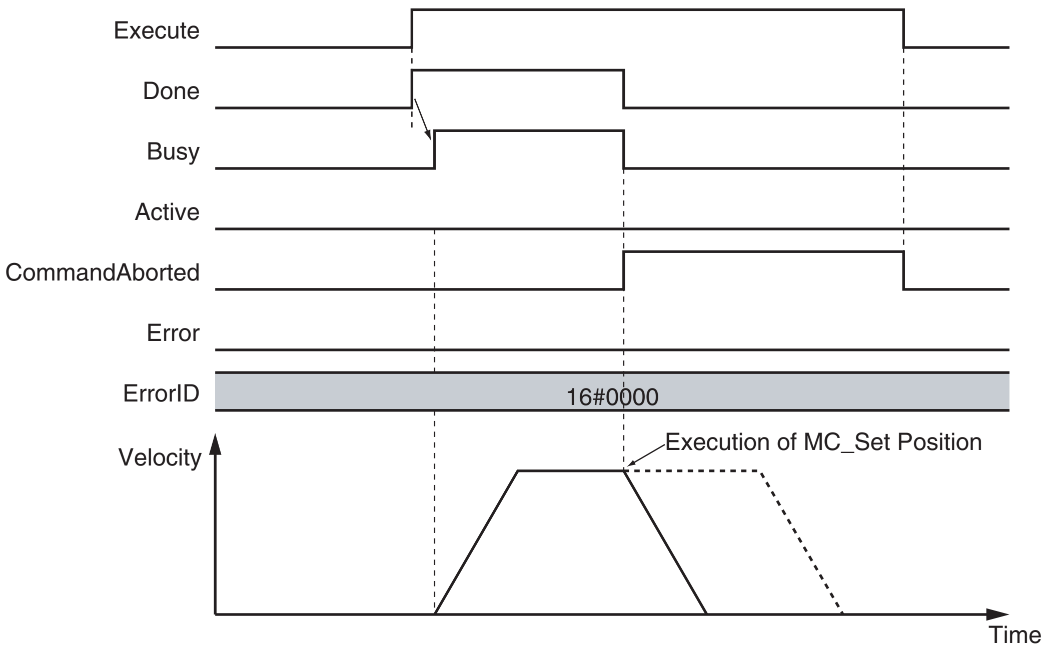
For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### Execution of Other Instructions during Instruction Execution

If you execute another instruction during execution of this instruction, you can specify either aborting or buffering. You cannot specify blending.

#### MC_SetPosition Execution during Instruction Execution

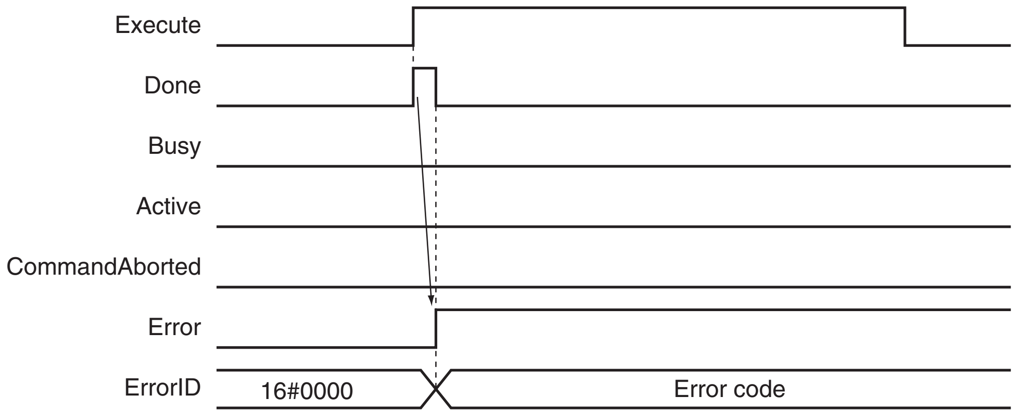
Home becomes undefined for the MC_SetPosition instruction. Therefore an error will occur if you attempt to execute it during execution of this instruction, and it will not be executed. If attempting to execute the MC_SetPosition instruction causes an error, the current instruction decelerates the axis to a stop and *CommandAborted* changes to TRUE. In this case, the output variable *Error* changes to TRUE for the MC_SetPosition instruction.



## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

### ● Timing Chart When Error Occurs

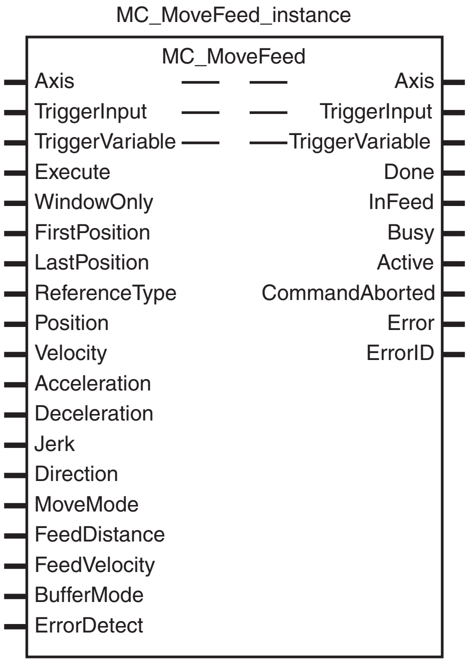



### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_MoveFeed

The MC_MoveFeed instruction performs positioning for the specified travel distance from the position where an external device triggers an interrupt input. Interrupt feeding is possible for absolute positioning, relative positioning, and velocity control.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_MoveFeed | Interrupt Feeding | FB |  | MC_MoveFeed_instance (<br>Axis :=*parameter,*<br>TriggerInput :=*parameter,*<br>TriggerVariable :=*parameter,*<br>Execute :=*parameter,*<br>WindowOnly :=*parameter,*<br>FirstPosition :=*parameter,*<br>LastPosition :=*parameter,*<br>ReferenceType<br>:=*parameter,*<br>Position :=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>Direction :=*parameter,*<br>MoveMode :=*parameter,*<br>FeedDistance :=*parameter,*<br>FeedVelocity :=*parameter,*<br>BufferMode :=*parameter,*<br>ErrorDetect :=*parameter,*<br>Done =>*parameter,*<br>InFeed =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

# Variables

## Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| WindowOnly | Window Only | BOOL | TRUE or FALSE | FALSE | Specify whether to enable or disable the window. |
| FirstPosition | First Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the position where latching is enabled. The unit is command units.*1 |
| LastPosition | Last Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the position where latching is disabled. The unit is command units.*1 |
| ReferenceType | Position Type Selection | _eMC_REFERENCE_TYPE | 1: _mcFeedback | 1*2 | Specify the position type.<br>1: Actual position (value obtained in the same primary period) |
| Position | Target Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the absolute target position. The unit is command units.*1 |
| Velocity | Target Velocity | LREAL | Positive number*3 | 0 | Specify the target velocity. The unit is command units/s.*1 |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/$s^2$.*1 |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.*1 |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/$s^3$.*1 |
| Direction | Direction | _eMC_DIRECTION | 0: _mcPositiveDirection<br>1: _mcShortestWay<br>2: _mcNegativeDirection<br>3: _mcCurrentDirection<br>4: _mcNoDirection | 0*2 | Specify the direction of rotation when MoveMode is set to absolute positioning*4 and when the Count Mode is Rotary Mode.<br>0: Positive direction<br>1: Shortest way<br>2: Negative direction<br>3: Current direction<br>4: No direction specified |
| MoveMode | Travel Mode | _eMC_MOVE_MODE | 0: _mcAbsolute<br>1: _mcRelative<br>2: _mcVelocity | 0*2 | Select the travel method.<br>0: Absolute positioning<br>1: Relative positioning<br>2: Velocity control |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| FeedDistance | Feed Distance | LREAL | Negative number, positive number, or 0 | 0 | Specify the travel distance after the interrupt feed input. Specify a positive value to feed in the same direction as the axis was moving before the interrupt input and specify a negative value to feed in the opposite direction. The unit is command units.*1 |
| FeedVelocity | Feed Velocity | LREAL | Positive number | 0 | Specify the target travel velocity after the interrupt feed input. The unit is command units/s.*1 |
| BufferMode | Buffer Mode Selection | _eMC_BUFFER_ MODE | 0: _mcAborting<br>1: _mcBuffered | 0*2 | Specify the behavior for multi-execution of instructions.<br>0: Aborting<br>1: Buffered |
| ErrorDetect | Error Detection Selection | BOOL | TRUE or FALSE | FALSE | Specify whether to detect an error when there is no interrupt feed input.<br>TRUE: Detect errors.<br>FALSE: Do not detect errors. |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 The default value for an enumeration variable is actually not the number, but the enumerator.

*3 A value of 0 can be set if *MoveMode* is set to velocity control.

*4 The axis moves to the specified position when *MoveMode* is set to absolute positioning and when the Count Mode is Linear Mode.
When *MoveMode* is set to relative positioning, the travel direction is determined by the sign of the position.
*_mcShortestWay* and *_mcNoDirection* cannot be selected when *MoveMode* is set to velocity control, regardless of the Counter Mode.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| InFeed | Feeding | BOOL | TRUE or FALSE | TRUE while feeding after receiving a latch input. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the instruction is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the instruction is started. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| InFeed | When feeding is started by the interrupt input.* | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

* There may be a delay of up to several primary periods from when the interrupt input turns ON until *InFeed* changes to TRUE.

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.*1 |
| TriggerInput | Trigger Input Condition | _sTRIGGER_REF | --- | Set the trigger condition.*2 |
| TriggerVariable | Trigger Variable | BOOL | TRUE or FALSE | Specify a trigger input variable when the Controller Mode is specified for the trigger mode. |

*1 Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

*2 Define a user-defined variable with a data type of _sTRIGGER_REF.

### _sTRIGGER_REF

| Name | Meaning | Data type | Valid range | Function |
|---|---|---|---|---|
| Mode | Mode | _eMC_TRIGGER _ MODE | 0: _mcDrive<br>1: _mcController | Specify the trigger mode.<br>0: Drive Mode<br>1: Controller Mode |
| LatchID | Latch ID Selection | _eMC_TRIGGER _ LATCHID | 0: _mcLatch1<br>1: _mcLatch2 | Specify which of the two latch functions to use.<br>0: Latch 1<br>1: Latch 2 |
| InputDrive | Trigger Input Signal | _eMC_TRIGGER _ INPUT_DRIVE | 0: _mcEncoderMark<br>1: _mcEXT | Specify the Servo Drive trigger signal to use in Drive Mode.<br>0: Z-phase signal<br>1: External input |

## Function

- When *Execute* changes to TRUE, the axis travels with absolute travel, relative travel, or velocity control depending on the *MoveMode* setting.
- The target position is set in *Position* (Target Position) for absolute travel. The target distance is set in *Position* (Target Distance) for relative travel.
  Both modes use *Velocity* (Target Velocity) for travel operation.
- Relative positioning is performed with *FeedVelocity* from the actual position where the external input turned ON during travel for the feed distance that is specified with *FeedDistance*.
- If no interrupt signal is input before the axis reaches the default target position during interrupt feeding in absolute or relative travel mode, the axis stops at the target position. You can specify whether there is an error output when the axis stops for *ErrorDetect* (i.e., when there is no interrupt input.) If you specify an error output, *CommandAborted* changes to TRUE, and Busy (Executing) and *Active* (Controlling) change to FALSE.
- To use interrupt masks, change *WindowOnly* to TRUE, then specify *FirstPosition* and *LastPosition*.
  Interrupt feeding is performed for the first interrupt signal generated by the actual position between the *FirstPosition* and the *LastPosition*.

**Precautions for Correct Use**

- Feeding after the interrupt is performed as a relative movement for the distance that is specified with *FeedDistance*. If a positive value is specified for *FeedDistance*, feeding is performed in the same direction as before the interrupt input, and if a negative value is specified, feeding is performed in the opposite direction.
- The setting of the Operation Selection at Reversing axis parameter is used for the acceleration and deceleration rates when reversing to feed.
- If an underflow or overflow would occur for the position after interrupt feeding, an error occurs when the interrupt input is received. If an interrupt input is received after there is an overflow or underflow, an axis error will still occur.

**Additional Information**

Refer to *MC_MoveAbsolute* on page 3-41 for absolute travel, *MC_MoveRelative* on page 3-68 for relative travel, *MC_MoveVelocity* on page 3-75 for velocity control, and *WindowOnly* on page 3-104 for WindowOnly.

## Mapping Data Objects

You must map the following object data when the MC_MoveFeed (Interrupt Feeding) instruction is executed with *Mode* set to Drive Mode. Mapping is performed in the Detailed Setting Area of the Axis Basic Settings Display of the Sysmac Studio.

- Touch probe function (60B8 hex)
- Touch probe status (60B9 hex)
- Touch probe pos1 pos value (60BA hex)
- Touch probe pos2 pos value (60BC hex)

If even one of the required objects is not set, a Process Data Object Setting Missing error (error code 3461 hex) occurs.

For details on mapping data objects, refer to *2-3 PDO Mapping* and to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

# Instruction Details

This section describes the instruction in detail.

## Specifying Axis

- Axis specifies the axis for which to latch the position.

## Trigger Input Condition

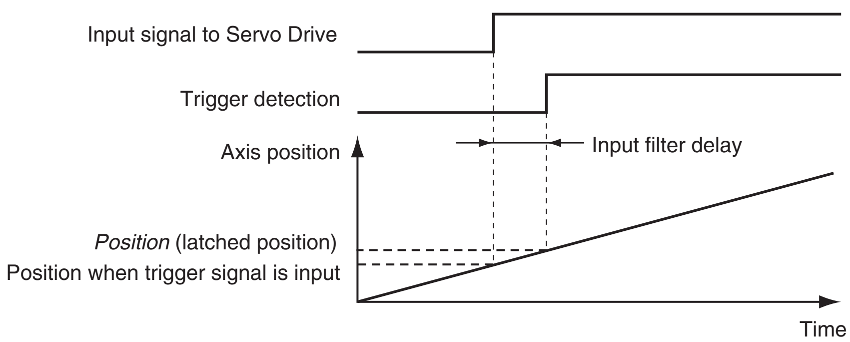
Select the trigger conditions with *Mode*, *LatchID*, and *InputDrive* of the *TriggerInput* (Trigger Input Conditions) variable.

### *Mode*

- The mode can be set to Drive Mode to specify a signal from the Servo Drive as the trigger, or to Controller Mode to specify a trigger with *TriggerVariable*.
- The trigger occurs on the rising edge of the trigger signal. The axis position is latched on the first trigger (FALSE to TRUE) after the MC_TouchProbe instruction is executed.
- While this instruction is Busy (Executing), a change in *TriggerVariable* is taken as a trigger even if Execute is FALSE.

**Drive Mode**

For trigger detection and latching of the actual position, the latched actual position is more precise in Drive Mode (which is a Servo Drive function) than it is in Controller Mode.



**Precautions for Correct Use**

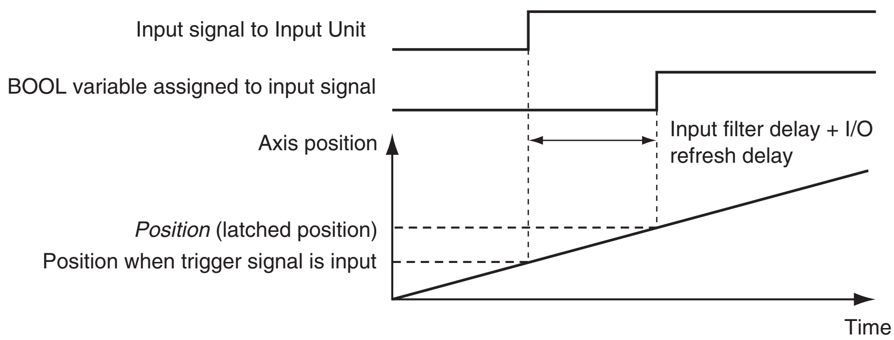
- When using Drive Mode, make sure that you connect the latch signal to the *LatchID* that you are going to use.
- The width of the latch signal depends on the performance of the Servo drive and other factors.
- You must map the following object data when the MC_MoveFeed (Interrupt Feeding) instruction is executed with *InputDrive* set to Drive Mode.

  Touch probe function (60B8 hex), Touch probe status (60B9 hex), Touch probe pos1 pos value (60BA hex), and Touch probe pos2 pos value (60BC hex)

  If even one of the required objects is not set, a Process Data Object Setting Missing error (error code 3461 hex) occurs. For details on mapping object data, refer to *2-3 PDO Mapping* and to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

**Controller Mode**

- You can specify a BOOL variable in the Controller Mode.
- Use TriggerVariable to specify the BOOL variable that you want to use as a trigger.
- The Controller Mode causes a longer delay compared to the Drive Mode. This is due to the I/O refresh delay that occurs when the trigger input signal is updated in the BOOL variable.




**Precautions for Correct Use**

If you use Controller Mode, the latch is performed each primary period. Therefore, the trigger variable must remain TRUE for at least one primary period.
Also, one primary period is required between when the trigger variable changes to TRUE and the MC Function Module processes the latch.

### *LatchID*

- Specify which of the two to use with *LatchID*. You can use only one of the latches with any one axis.
- *LatchIDs* indicate latch circuit 1 and latch circuit 2 in the Servo Drive.

For information on latch IDs, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### *InputDrive*

- You can select *_mcEncoderMark* (Z Phase) or *_mcEXT* (External Input) as triggers.
- Select *_mcEncoderMark* (Z Phase) to use the Z phase of the Servo Drive as the trigger.
  An external trigger signal that is input to the Servo Drive is used as the trigger for *_mcEXT* (External Input).
- For an OMRON G5-series Servo Drive, there are three options for *_mcEXT*: Ext1, Ext2, and Ext3. Use Sysmac Studio to make the setting. Make the settings from the Sysmac Studio.
  The two triggers set in the Servo Drive can have the same setting.

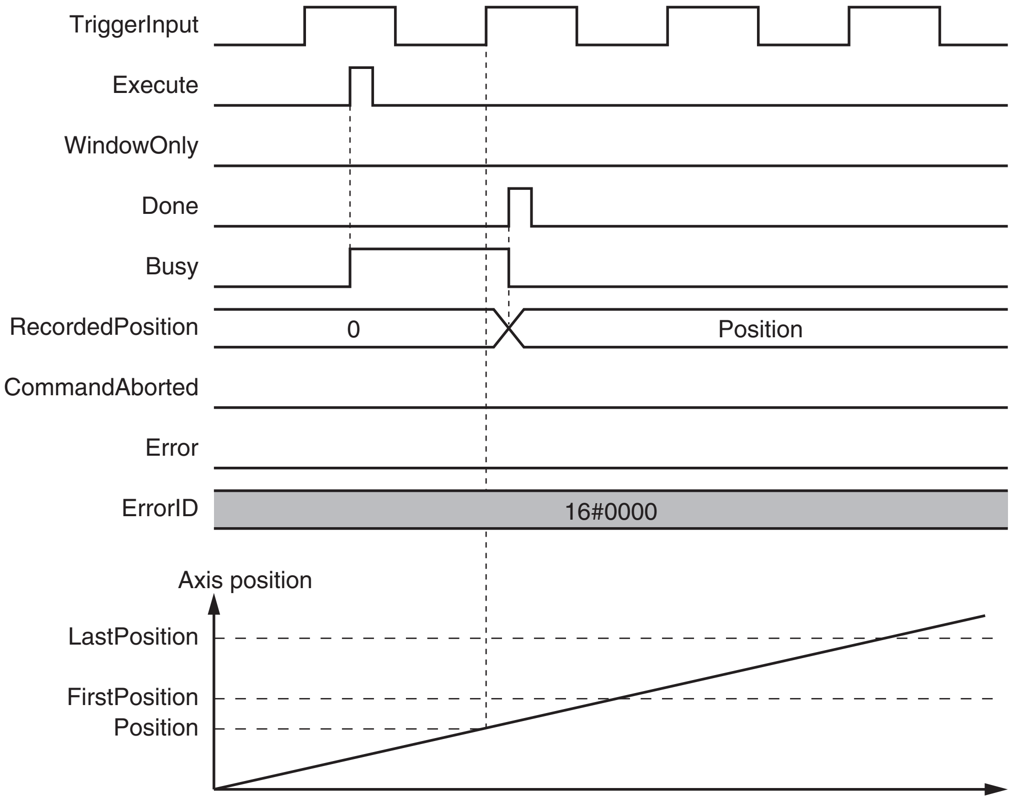
## ● *WindowOnly*

- *WindowOnly* specifies whether the window is enabled or disabled.
- If you specify *Disable*, triggers are detected for all axis positions.
- If you specify *Enable*, triggers are detected only when the axis position is within the range specified by *FirstPosition* and *LastPosition*.

The following timing chart shows the difference in operation depending on the *WindowOnly* setting.

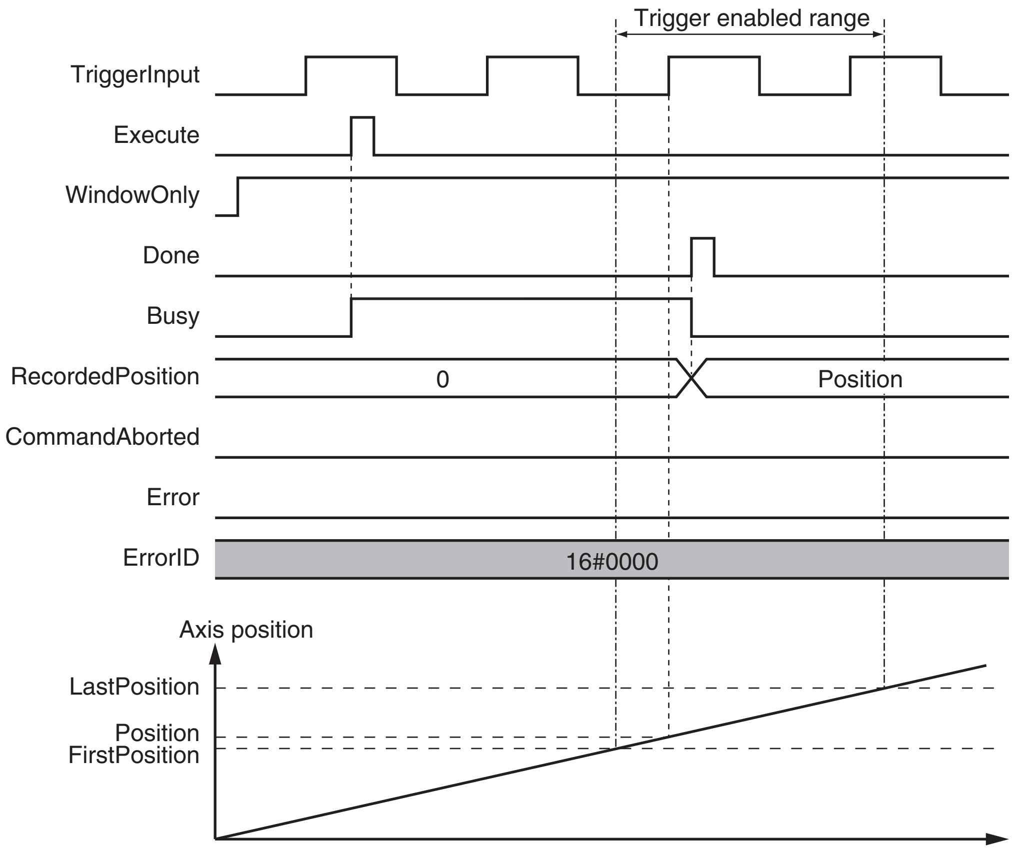
### *WindowOnly* Set to Disable

The axis position when the first trigger occurs after *Execute* changes to TRUE is used as the reference position for the feed distance.



### *WindowOnly* Set to Enable

Only trigger inputs within the window are detected to latch the axis position.




**Precautions for Correct Use**

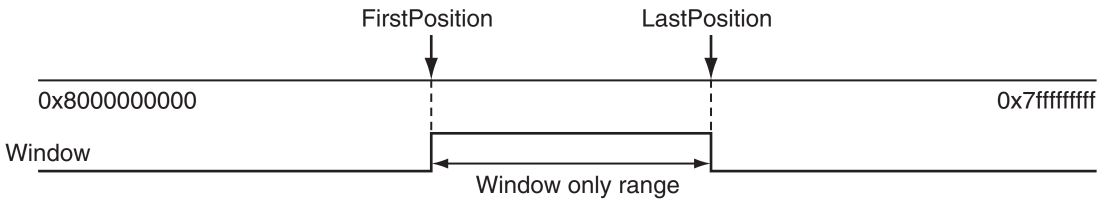
- Latching is not possible at the moment that *WindowOnly* changes to TRUE and until the latch function is activated.
- Time is needed until the latch function is activated. If the effective range for *WindowOnly* is too small, latching is not possible. The range in which latching is possible depends on the Servo Drive or Encoder Input Terminal performance, and on EtherCAT communications.

The range that is defined by *FirstPosition* and *LastPosition* depends on the Count Mode, as given below.

### Linear Mode

- *FirstPosition* must be less than or equal to the window range and the window range must be less than or equal to *LastPosition*.
- An error will occur if the *FirstPosition* is greater than the *LastPosition*.
- An error will also occur if a position beyond the position range of Linear Mode is specified.
- *FirstPosition* and *LastPosition* are LREAL variables. Do not set them to the same values. Refer to *Treatment of REAL and LREAL Data* on page 1-13 for information on LREAL data.

The window only range in Linear Mode is shown below.

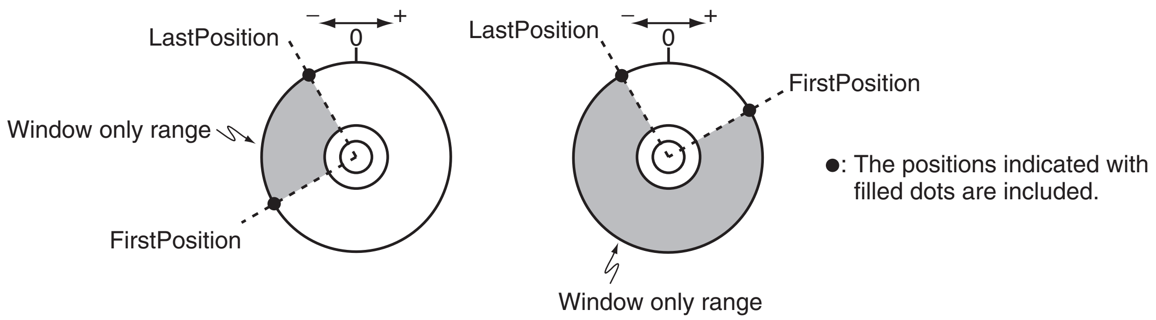



**Note** The window only range can include the *FirstPosition* and *LastPosition*.

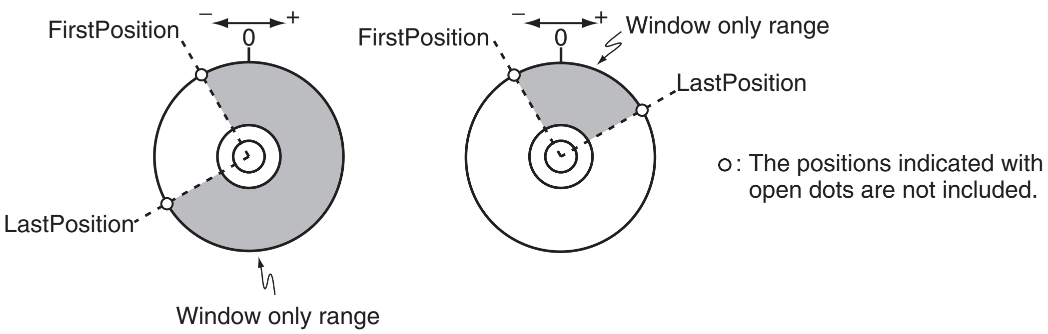
### Rotary Mode

- The *FirstPosition* can be less than, equal to, or greater than the *LastPosition*.
- If the *FirstPosition* is greater than the *LastPosition*, the setting range includes the maximum/minimum position of the ring counter.
- An error will occur if you specify a value beyond the upper and lower limits of the ring counter.

FirstPosition ≦ LastPosition




FirstPosition > LastPosition




## ● *ReferenceType* (Position Type Selection)

The position type is as follows:

- *_mcFeedback*: Value obtained in the same primary period
  The actual position of the master axis that was obtained in the same primary period is used.

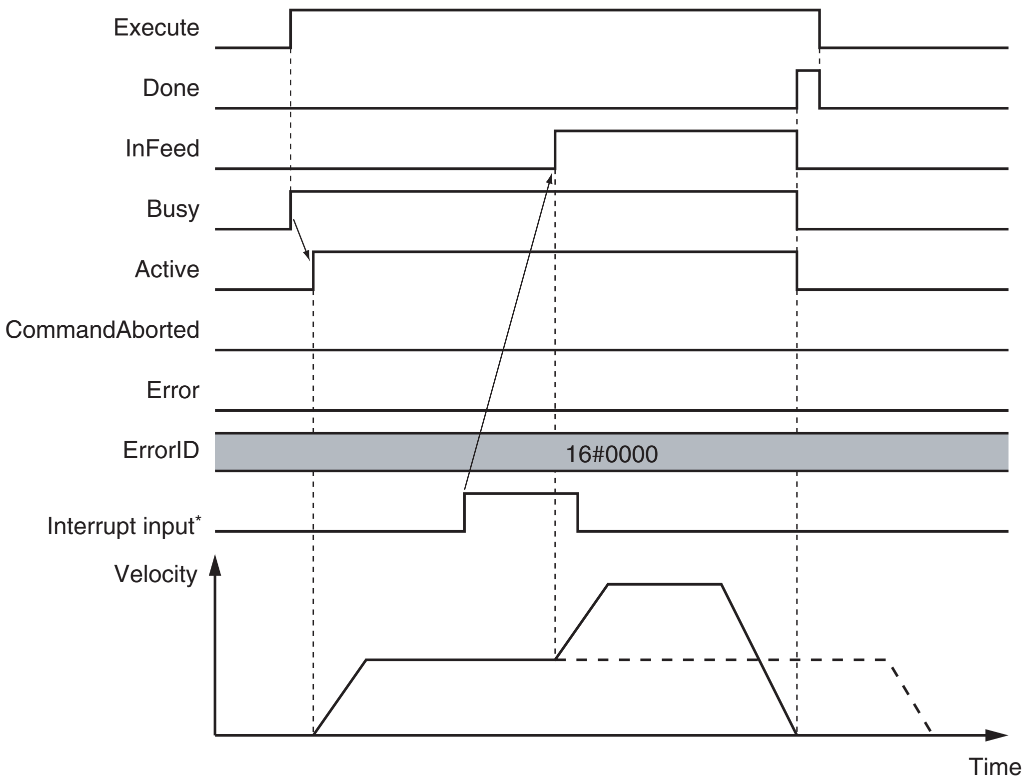
## ● *FeedDistance*

Specify a positive value for *FeedDistance* to perform feed in the same direction as the motion before the interrupt input. Specify a negative value for *FeedDistance* to perform feed in the opposite direction as the motion before the interrupt input.
For example, if you specify a positive value for *FeedDistance* when the motion was in the negative direction before the interrupt input, feeding is performed in the negative direction. If you specify a negative value for *FeedDistance*, feeding is performed in the positive direction.

## Timing Charts

- *Busy* (Executing) changes to TRUE at the same time as *Execute* changes to TRUE. *Active* (Controlling) changes to TRUE in the next period.
- After an interrupt input, *InFeed* changes to TRUE and when *FeedDistance* is reached and positioning is completed, *Done* changes to TRUE.
- If another instruction aborts this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing) and *Active* (Controlling) change to FALSE.

### ● When *MoveMode* (Travel Mode) is *_mcAbsolute* or *_mcRelative*

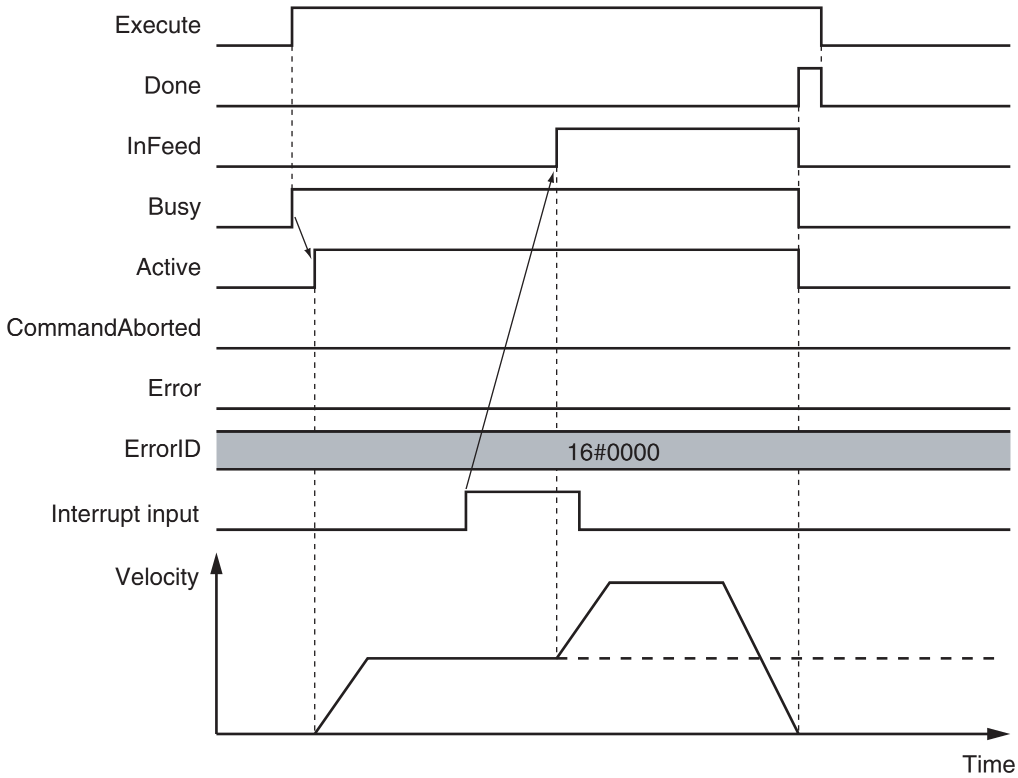



* There may be a delay of up to several control periods from when the interrupt input turns ON until *InFeed* changes to TRUE.

**Additional Information**

Acceleration or deceleration to interrupt feeding is performed according to the *Acceleration* (Acceleration Rate) or *Deceleration* (Deceleration Rate) input variable.

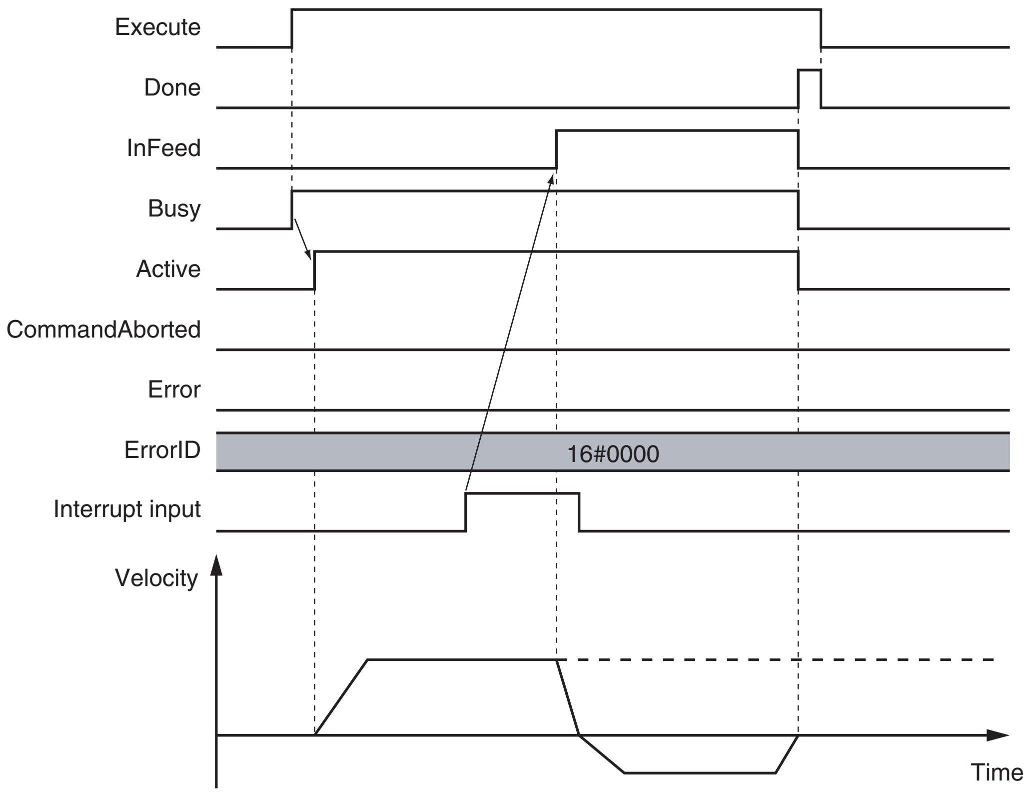
- **When *MoveMode* (Travel Mode) is *_mcVelocity***




- **Operation at Reversing Axis Parameter**

When feeding reverses its direction after the interrupt input, operation follows the Operation at Reversing axis parameter

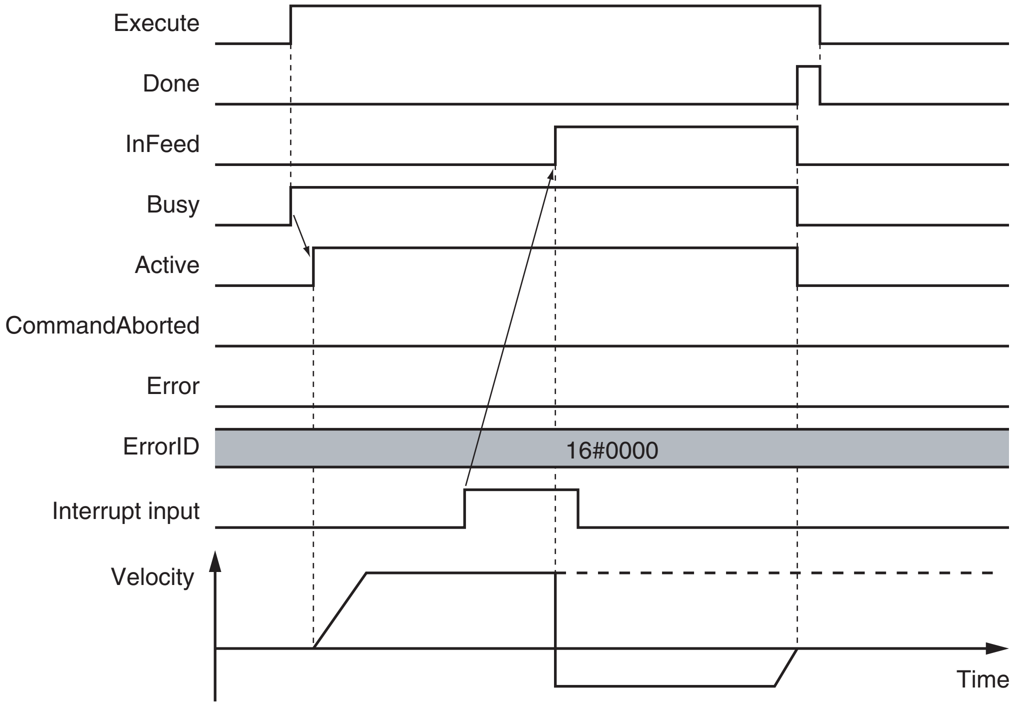
### When Motion Variable Is 0 (Decelerate to a Stop After Reversing)



**Additional Information**

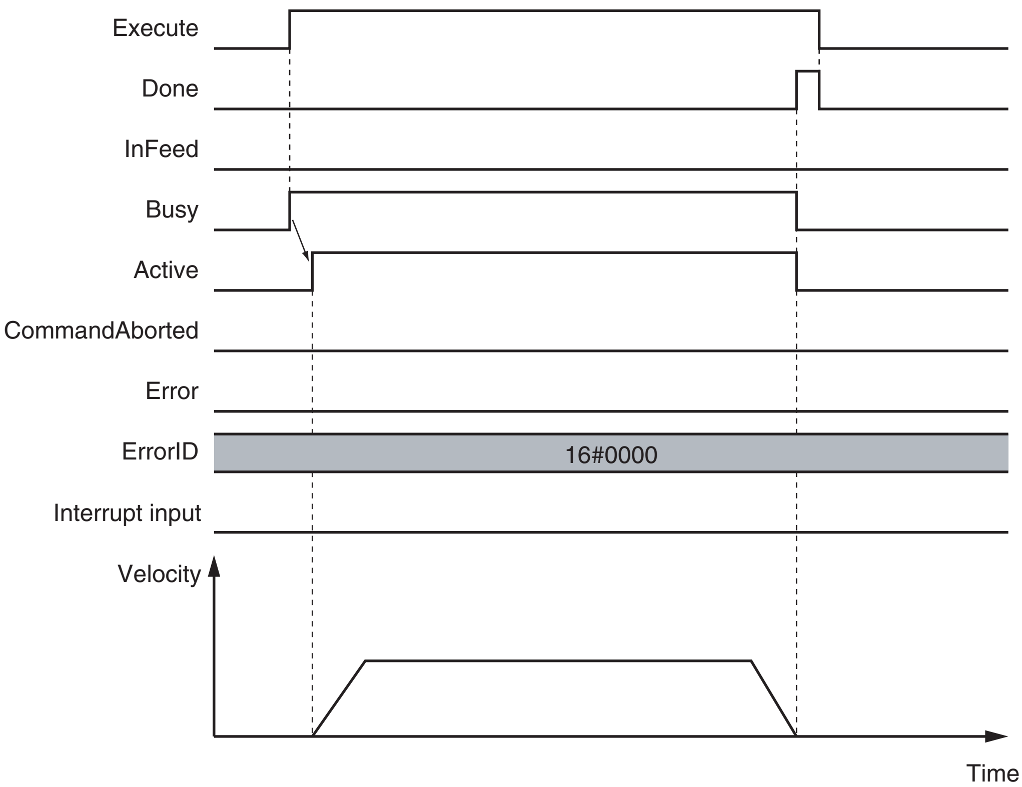
The deceleration rate when the axis reverses after an interrupt input follows the *Deceleration* (Deceleration Rate) input variable.

### When Motion Variable Is 1 (Stop Immediately After Reversing)

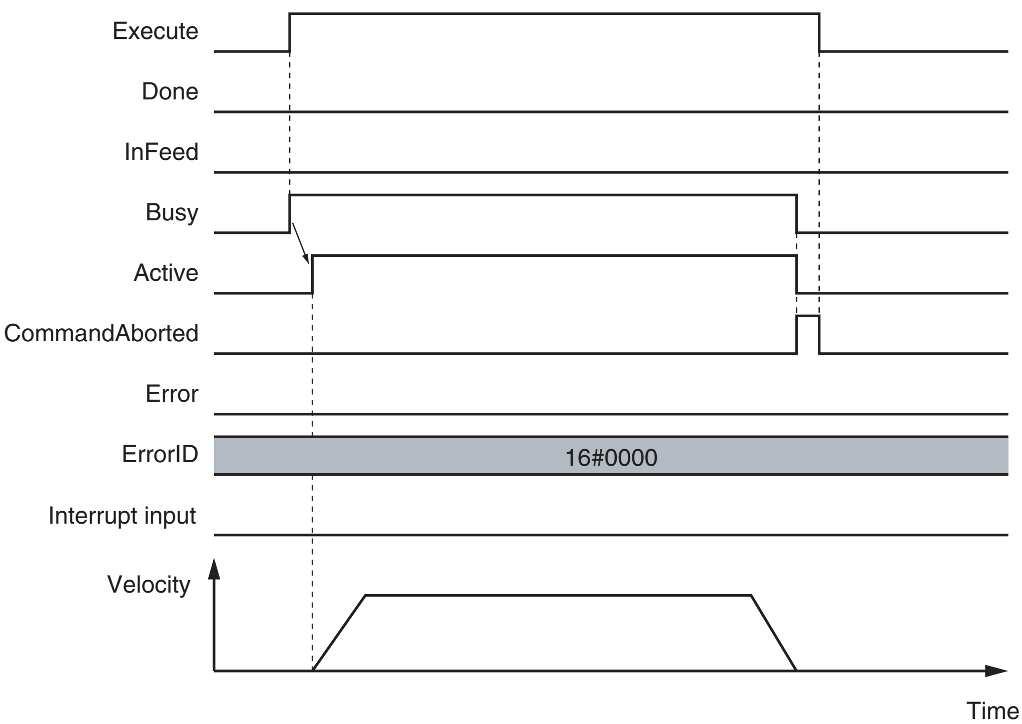


- **When *MoveMode* (Travel Mode) is *_mcAbsolute* and an Interrupt Input Is Not Received**

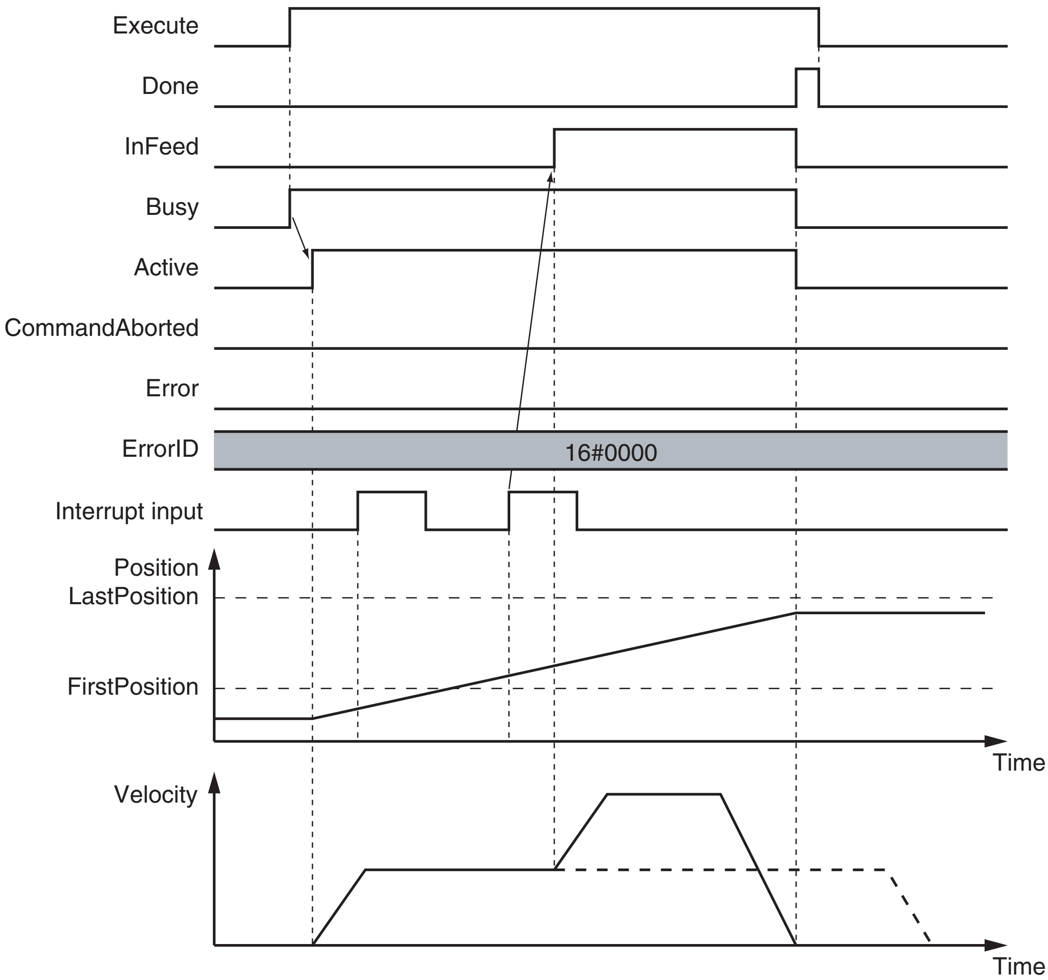
**When *ErrorDetect* (Error Detection Selection) Is Set to Not Detect Errors**




**When *ErrorDetect* (Error Detection Selection) Is Set to Detect Errors**



● **When *WindowOnly* Is Enabled**



## Re-execution of Motion Control Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

You can switch to this instruction or buffer this instruction if you execute it during execution of another instruction. You can buffer one instruction per axis. Specify the operation of this instruction using *BufferMode* (Buffer Mode Selection) for multi-execution of instructions.

| Buffer Mode Selection | Description |
|---|---|
| Aborting | Aborts the instruction being executed and switches to this instruction.<br>If the direction of axis motion is reversed by switching instructions, the motion will be reversed according to the Operation at Reversing axis parameter. |
| Buffered | Buffers this instruction and executes it automatically after the current instruction is completed. |

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution of Other Instructions during Instruction Execution

You can execute another instruction with the Buffer Mode set to *Aborting* during execution of this instruction.

The following will occur if another instruction with the Buffer Mode set to *Buffered* or a blending mode is executed.

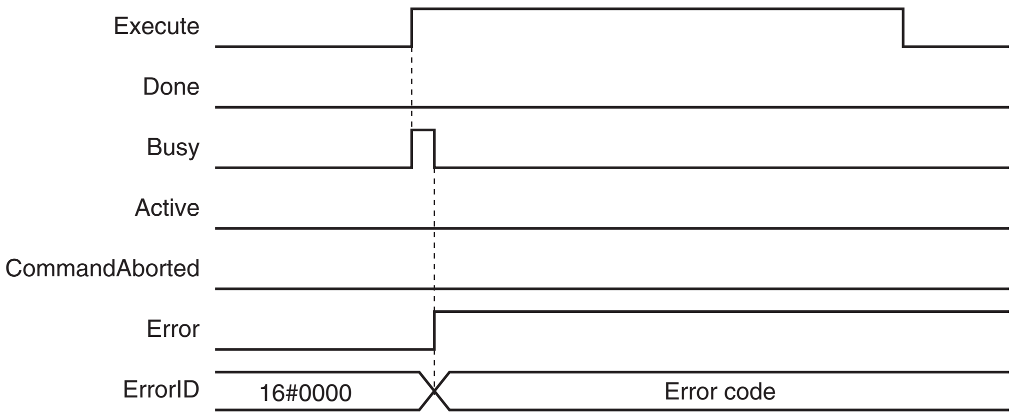
- *Error* changes to TRUE in the other instruction. A Motion Control Instruction Multi-execution Disabled error (error code: 543C hex) is output to *ErrorID* (Error Code).
- The MC_MoveFeed instruction is aborted and *CommandAborted* changes to TRUE.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

### ● Timing Chart When Error Occurs

When there is an error, the latch used for the interrupt input for this instruction is disabled.




### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

## Sample Programming

This section shows sample programming where control changes from velocity control to interrupt feeding.

### Parameter Settings

The minimum settings required for this sample programming are given below.

#### ● Setting Axis Parameters

##### Axis Types

| Axis | Axis Type |
|---|---|
| Axis 1 | Servo axis |
| Axis 2 | Servo axis |

##### Count Modes

| Axis | Count Mode |
|---|---|
| Axis 1 | Rotary Mode |
| Axis 2 | Linear Mode |

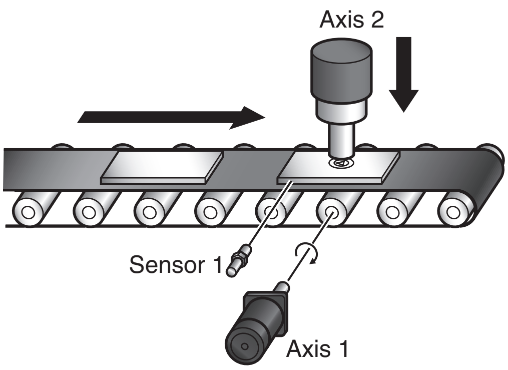
##### Ring Counters

| Axis | Modulo maximum position | Modulo minimum position |
|---|---|---|
| Axis 1 | 360 | 0 |

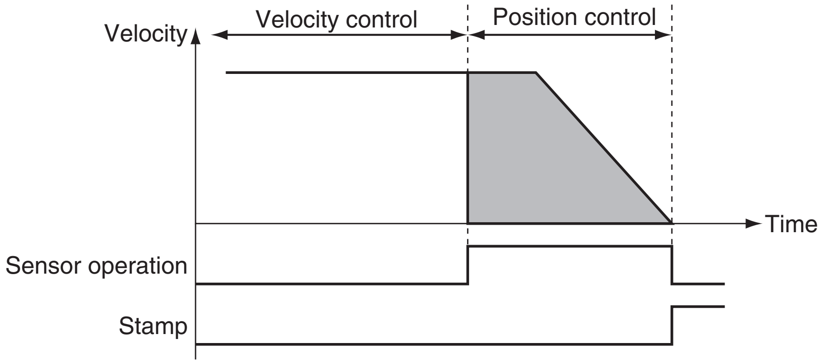
##### Units of Display

| Axis | Unit of Display |
|---|---|
| Axis 1 | degree |
| Axis 2 | mm |

### Operation Example



### Operation Pattern




**1** Conveyor Operation

Axis 1, which moves the conveyer belt, performs velocity control before interrupt feeding.

**2** Feeding

Sensor 1 is connected to latch 1.
When Sensor1 turns ON, operation changes to feeding and the axis stops at the specified position.

**3** Pressing the Stamp

When positioning is finished, axis 2 of the stamp moves perpendicularly down at the position determined by absolute positioning to press the stamp.
After stamping is performed, axis 2 returns to home.

When the absolute positioning is completed, the axis is immediately returned to home. To enable this, *BufferMode* (Buffer Mode Selection) of the MC_MoveZeroPosition (High-speed Home) instruction is set to Buffered. Multi-execution of instructions is performed if the Active output from the previous instruction is TRUE.

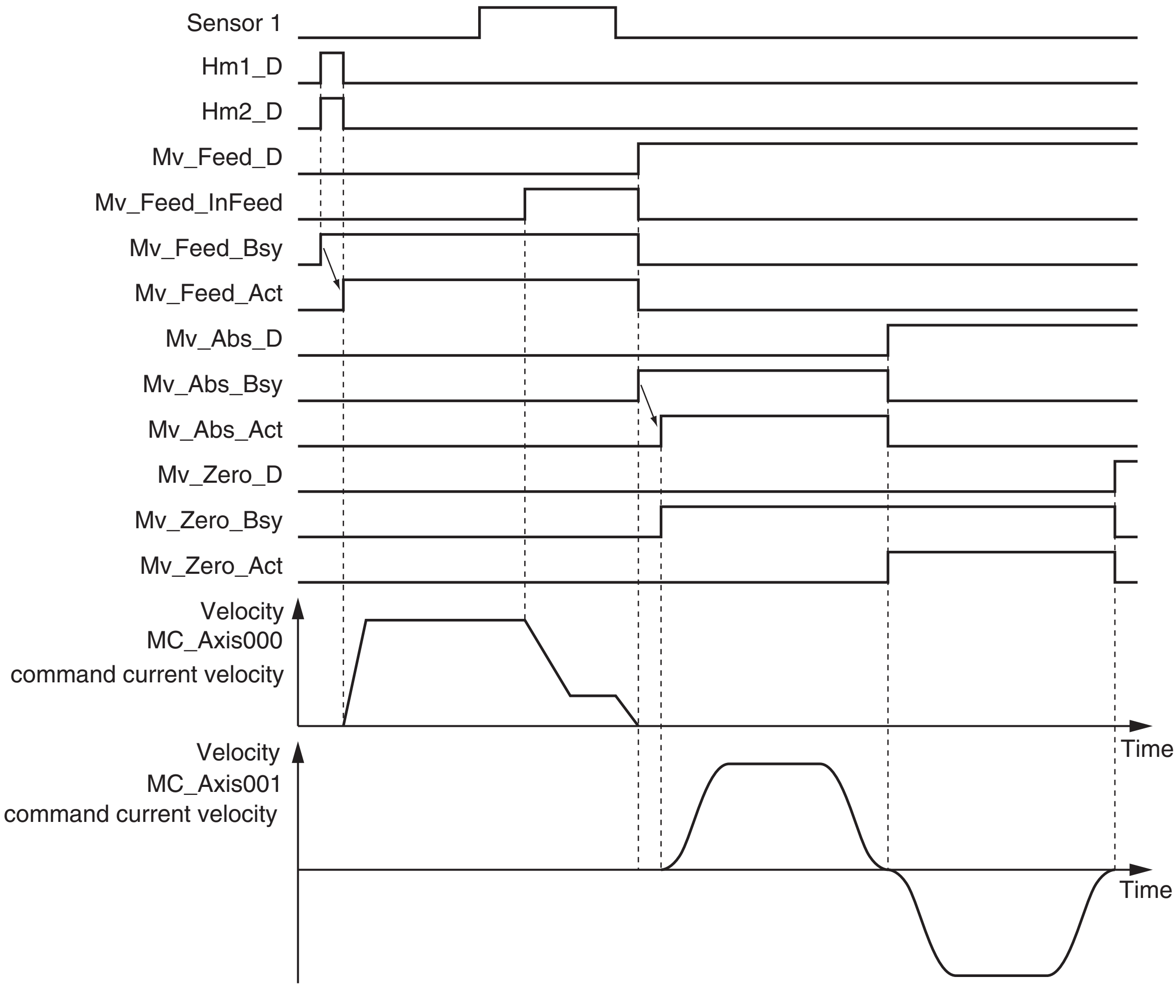
## Ladder Diagram

### Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | Axis Variable for axis 2. |
| MC_Axis001.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 2. |
| MC_Axis001.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 2. |
| TrigRef | _sTRIGGER_REF | --- | This is the specified variable for the interrupt input. Latch 1 of the Servo Drive is used in this sample. When the rising edge of the external input for sensor 1 is detected, interrupt feeding is executed. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |

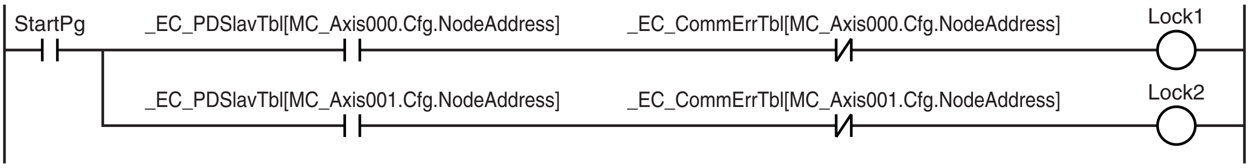
| Name | Data type | Default | Comment |
|---|---|---|---|
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

## ● Timing Chart

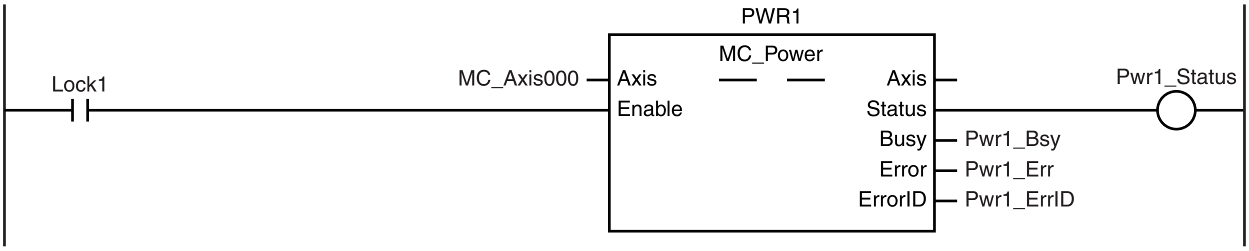


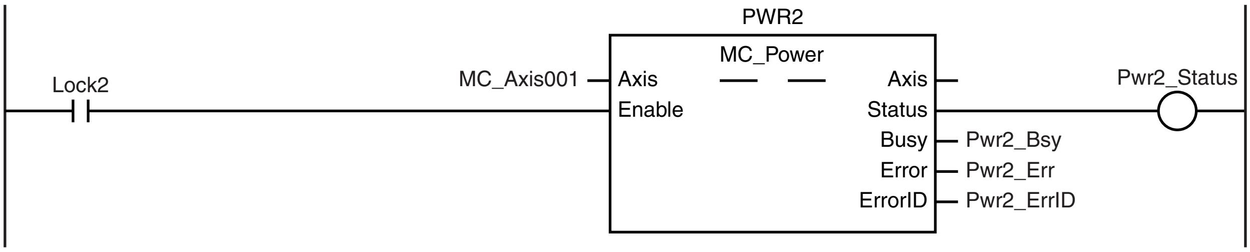
## ● Sample Programming

If *StartPg* is TRUE, EtherCAT communications for the axes are checked to see if process data communications are normal.

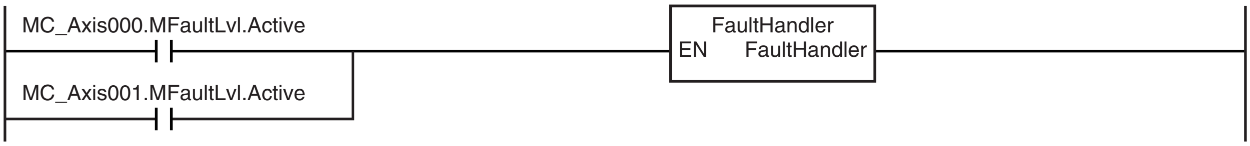



If process data communications are active, the Servo is turned ON.

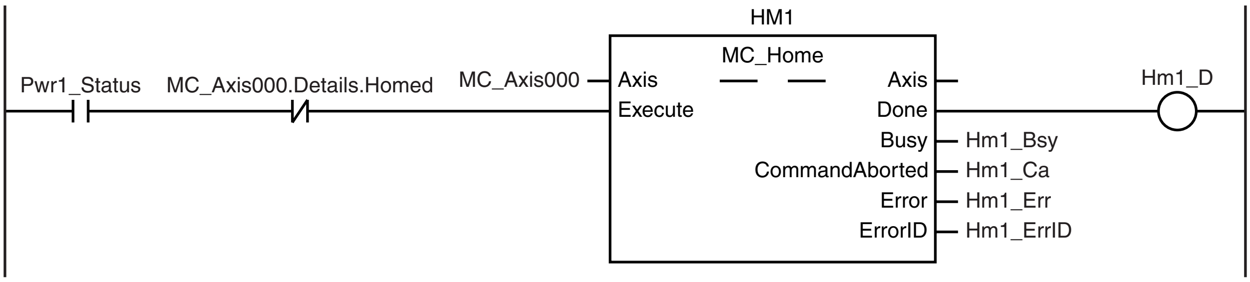




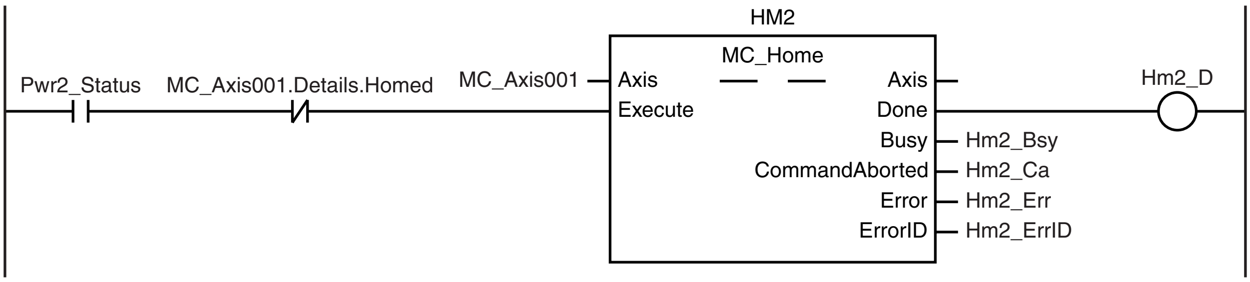

If there is a minor fault level error for a composition axis, the error handler for the device (FaultHandler) is executed. Design FaultHandler separately.



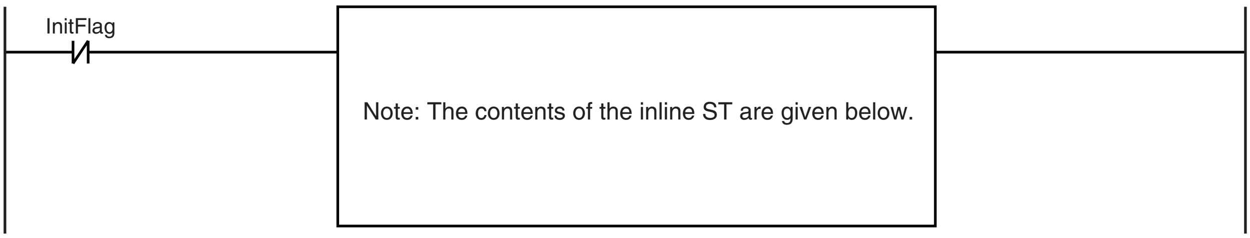

If the Servo is ON for axis 1 and home is not defined, the Home instruction is executed.



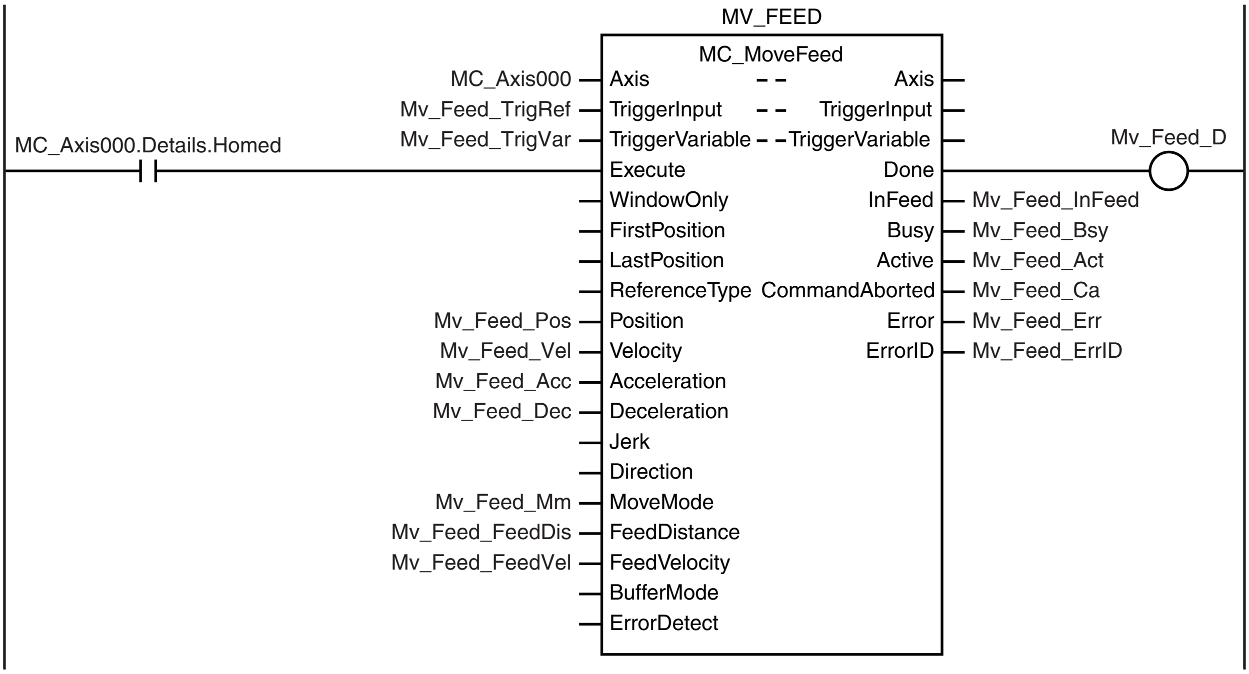

If the Servo is ON for axis 2 and home is not defined, the Home instruction is executed.



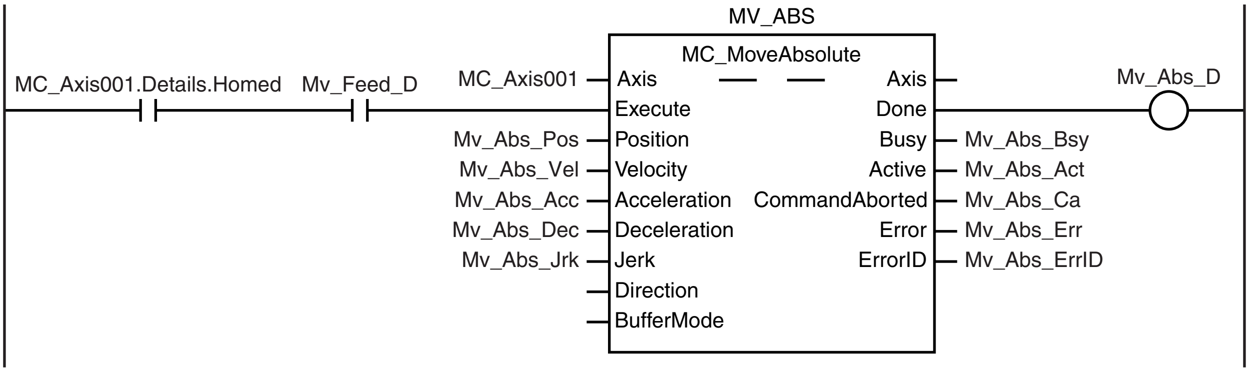

The parameters are set for interrupt feeding, absolute positioning, and high-speed homing.



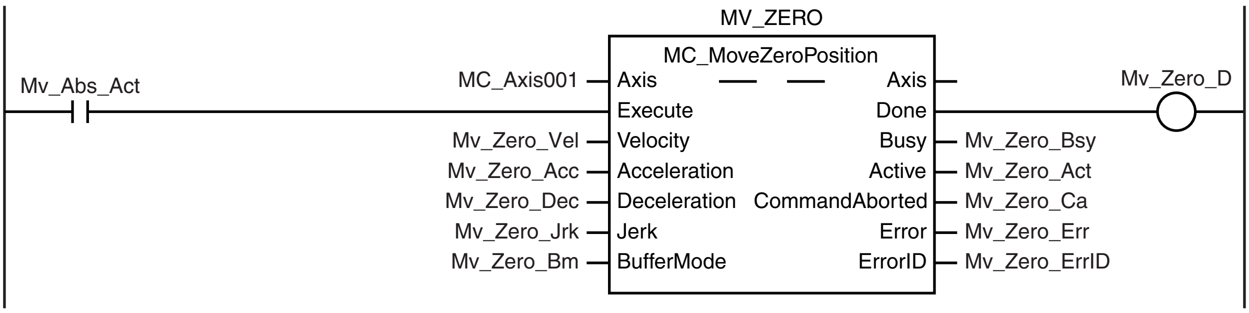
If home is defined for axis 1, interrupt feeding is executed for axis 1.




If home is defined for axis 2 and interrupt feeding is completed for axis 1, absolute positioning is performed for axis 2.




After absolute positioning is completed, the High-speed Home instruction is executed to move to home.




## Contents of Inline ST

```
// MV_FEED parameters
Mv_Feed_TrigRef.Mode            := _eMC_TRIGGER_MODE#_mcDrive;
Mv_Feed_TrigRef.LatchID         := _eMC_TRIGGER_LATCH_ID#_mcLatch1;
Mv_Feed_TrigRef.InputDrive      := _eMC_TRIGGER_INPUT_DRIVE#_mcEXT;
Mv_Feed_TrigVar                 := FALSE;
Mv_Feed_Pos                     := LREAL#2000.0;
Mv_Feed_Vel                     := LREAL#1000.0;
Mv_Feed_Acc                     := LREAL#10000.0;
Mv_Feed_Dec                     := LREAL#10000.0;
Mv_Feed_Mm                      := _eMC_MOVE_MODE#_mcVelocity;
Mv_Feed_FeedDis                 := LREAL#500.0;
```

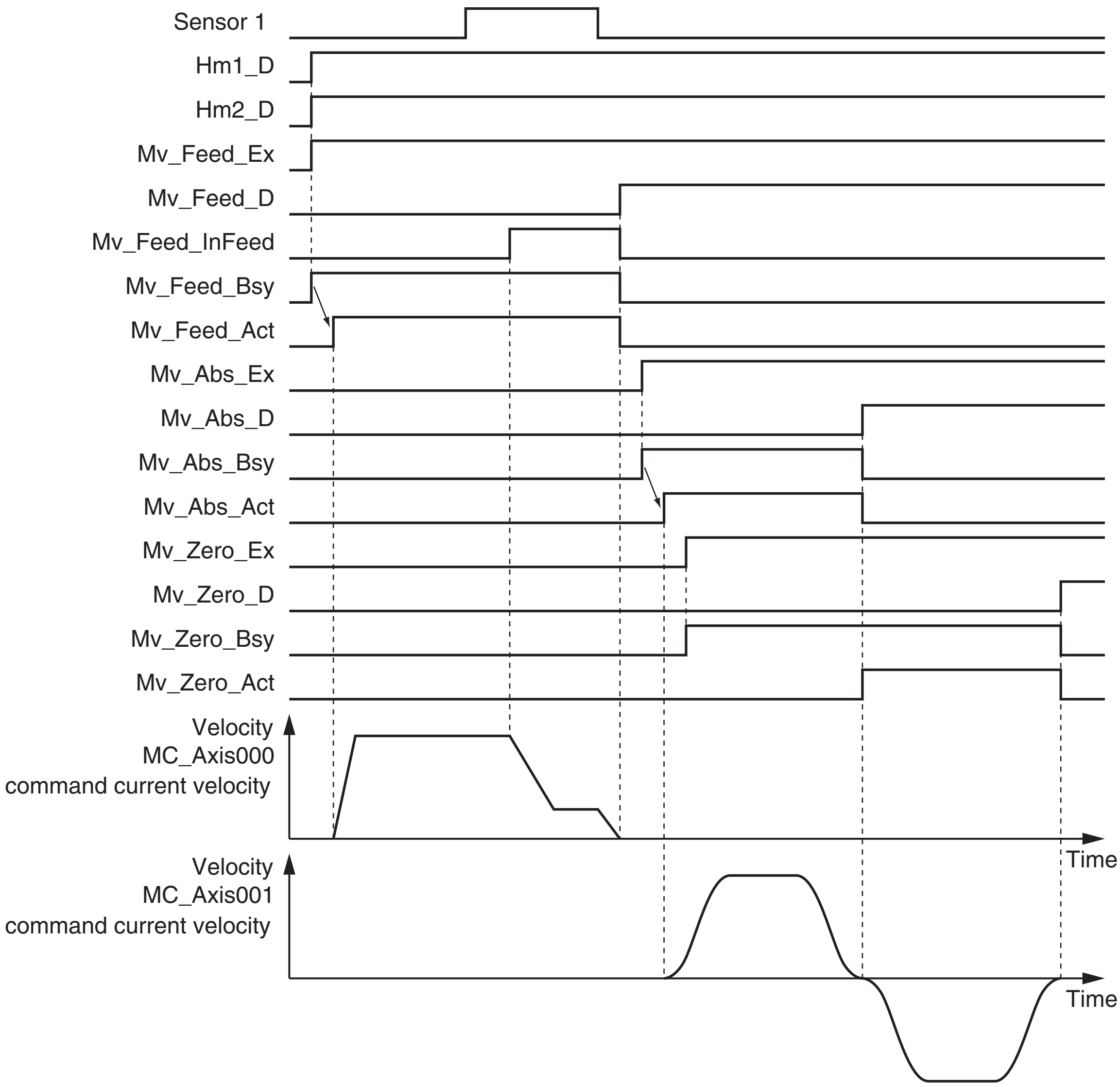
```
Mv_Feed_FeedVel             := LREAL#500.0;

// MV_ABS parameters
Mv_Abs_Pos      := LREAL#1000.0;
Mv_Abs_Vel      := LREAL#500.0;
Mv_Abs_Acc      := LREAL#10000.0;
Mv_Abs_Dec      := LREAL#10000.0;
Mv_Abs_Jrk      := LREAL#10000.0;

// MV_ZERO parameters
Mv_Zero_Vel     := LREAL#500.0;
Mv_Zero_Acc     := LREAL#10000.0;
Mv_Zero_Dec     := LREAL#10000.0;
Mv_Zero_Jrk     := LREAL#10000.0;
Mv_Zero_Bm      := _eMC_BUFFER_MODE#_mcBuffered;

// Change InitFlag to TRUE after setting the input parameters.
InitFlag := TRUE;
```

## Structured Text (ST)

### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | Axis Variable for axis 2. |
| MC_Axis001.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 2. |
| MC_Axis001.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 2. |
| TrigRef | _sTRIGGER_REF | --- | This is the specified variable for the interrupt input. Latch 1 of the Servo Drive is used in this sample. When the rising edge of the external input for sensor 1 is detected, interrupt feeding is executed. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |
| Hm1_Ex | BOOL | FALSE | The HM1 instance of MC_Home is executed when this variable changes to TRUE. |
| Hm2_Ex | BOOL | FALSE | The HM2 instance of MC_Home is executed when this variable changes to TRUE. |
| Mv_Feed_Ex | BOOL | FALSE | The MV_FEED instance of MC_MoveFeed is executed when this variable changes to TRUE. |
| Mv_Abs_Ex | BOOL | FALSE | The MV_ABS instance of MC_MoveAbsolute is executed when this variable changes to TRUE. |
| Mv_Zero_Ex | BOOL | FALSE | The MV_ZERO instance of MC_MoveZeroPosition is executed when this variable changes to TRUE. |

## Timing Chart




## Sample Programming

```
// Processing when input parameters are not set
IF InitFlag=FALSE THEN

    // MV_FEED parameters
    Mv_Feed_TrigRef.Mode            := _eMC_TRIGGER_MODE#_mcDrive;
    Mv_Feed_TrigRef.LatchID         := _eMC_TRIGGER_LATCH_ID#_mcLatch1;
    Mv_Feed_TrigRef.InputDrive      := _eMC_TRIGGER_INPUT_DRIVE#_mcEXT;
    Mv_Feed_TrigVar                 := FALSE;
    Mv_Feed_Pos                     := LREAL#2000.0;
    Mv_Feed_Vel                     := LREAL#1000.0;
    Mv_Feed_Acc                     := LREAL#10000.0;
    Mv_Feed_Dec                     := LREAL#10000.0;
    Mv_Feed_Mm                      := _eMC_MOVE_MODE#_mcVelocity;
    Mv_Feed_FeedDis                 := LREAL#500.0;
    Mv_Feed_FeedVel                 := LREAL#500.0;
```

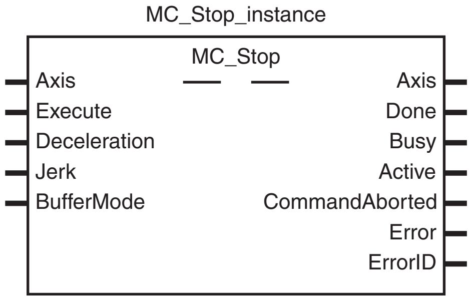
```
    // MV_ABS parameters
    Mv_Abs_Pos      := LREAL#1000.0;
    Mv_Abs_Vel      := LREAL#500.0;
    Mv_Abs_Acc      := LREAL#10000.0;
    Mv_Abs_Dec      := LREAL#10000.0;
    Mv_Abs_Jrk      := LREAL#10000.0;

    // MV_ZERO parameters
    Mv_Zero_Vel     := LREAL#500.0;
    Mv_Zero_Acc     := LREAL#10000.0;
    Mv_Zero_Dec     := LREAL#10000.0;
    Mv_Zero_Jrk     := LREAL#10000.0;
    Mv_Zero_Bm      := _eMC_BUFFER_MODE#_mcBuffered;

    // Change InitFlag to TRUE after setting the input parameters.
    InitFlag := TRUE;

END_IF;


// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 1 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis000.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
    Pwr1_En:=TRUE;          // Turn ON the Servo.
ELSE
    Pwr1_En:=FALSE;         // Turn OFF the Servo.
END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 2 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis001.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis001.Cfg.NodeAddress]=FALSE) THEN
    Pwr2_En:=TRUE;          // Turn ON the Servo for axis 2.
ELSE
    Pwr2_En:=FALSE;         // Turn OFF the Servo for axis 2.
END_IF;

// Processing for a minor fault level error
// Implement FaultHandler separately.
IF (MC_Axis000.MFaultLvl.Active=TRUE) OR (MC_Axis001.MFaultLvl.Active=TRUE) THEN
    FaultHandler();
END_IF;


// If the Servo is ON for axis 1 and home is not defined, the Home instruction is executed for axis 1.
IF (Pwr1_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
    Hm1_Ex:=TRUE;
END_IF;

// If the Servo is ON for axis 2 and home is not defined, the Home instruction is executed for axis 2.
IF (Pwr2_Status=TRUE) AND (MC_Axis001.Details.Homed=FALSE) THEN
    Hm2_Ex:=TRUE;
END_IF;

// After home is defined for axis 1, MC_MoveFeed is executed.
IF MC_Axis000.Details.Homed=TRUE THEN
    Mv_Feed_Ex:=TRUE;
END_IF;
```

```
// If home is defined for axis 2 and interrupt feeding is completed for axis 1, absolute positioning is performed for
axis 2.
IF (MC_Axis001.Details.Homed=TRUE) AND (Mv_Feed_D=TRUE) THEN
    Mv_Abs_Ex := TRUE;
END_IF;

// After MC_MoveAbsolute is started, MC_MoveZeroPosition is executed with multi-execution of instructions.
IF Mv_Abs_Act=TRUE THEN
    Mv_Zero_Ex := TRUE;
END_IF;


// MC_Power for axis 1
PWR1(

    Axis        := MC_Axis000,
    Enable      := Pwr1_En,
    Status      => Pwr1_Status,
    Busy        => Pwr1_Bsy,
    Error       => Pwr1_Err,
    ErrorID     => Pwr1_ErrID
);


// MC_Power for axis 2
PWR2(
    Axis        := MC_Axis001,
    Enable      := Pwr2_En,
    Status      => Pwr2_Status,
    Busy        => Pwr2_Bsy,
    Error       => Pwr2_Err,
    ErrorID     => Pwr2_ErrID
);


// MC_Home for axis 1
HM1(
    Axis                := MC_Axis000,
    Execute             := Hm1_Ex,
    Done                => Hm1_D,
    Busy                => Hm1_Bsy,
    CommandAborted      => Hm1_Ca,
    Error               => Hm1_Err,
    ErrorID             => Hm1_ErrID
);


// MC_Home for axis 2
HM2(
    Axis                := MC_Axis001,
    Execute             := Hm2_Ex,
    Done                => Hm2_D,
    Busy                => Hm2_Bsy,
    CommandAborted      => Hm2_Ca,
    Error               => Hm2_Err,
    ErrorID             => Hm2_ErrID
);

// MC_MoveFeed
MV_FEED(
    Axis                := MC_Axis000,
    TriggerInput        := Mv_Feed_TrigRef,
    TriggerVariable     := Mv_Feed_TrigVar,
    Execute             := Mv_Feed_Ex,
    Position            := Mv_Feed_Pos,
```

```
    Velocity           := Mv_Feed_Vel,
    Acceleration       := Mv_Feed_Acc,
    Deceleration       := Mv_Feed_Dec,
    MoveMode           := Mv_Feed_Mm,
    FeedDistance       := Mv_Feed_FeedDis,
    FeedVelocity       := Mv_Feed_FeedVel,
    Done               => Mv_Feed_D,
    InFeed             => Mv_Feed_InFeed,
    Busy               => Mv_Feed_Bsy,
    Active             => Mv_Feed_Act,
    CommandAborted     => Mv_Feed_Ca,
    Error              => Mv_Feed_Err,
    ErrorID            => Mv_Feed_ErrID
);


// MC_MoveAbsolute
MV_ABS(
    Axis               := MC_Axis001,
    Execute            := Mv_Abs_Ex,
    Position           := Mv_Abs_Pos,
    Velocity           := Mv_Abs_Vel,
    Acceleration       := Mv_Abs_Acc,
    Deceleration       := Mv_Abs_Dec,
    Jerk               := Mv_Abs_Jrk,
    Done               => Mv_Abs_D,
    Busy               => Mv_Abs_Bsy,
    Active             => Mv_Abs_Act,
    CommandAborted     => Mv_Abs_Ca,
    Error              => Mv_Abs_Err,
    ErrorID            => Mv_Abs_ErrID
);

// MC_MoveZeroPosition
MV_ZERO(
    Axis               := MC_Axis001,
    Execute            := Mv_Zero_Ex,
    Velocity           := Mv_Zero_Vel,
    Acceleration       := Mv_Zero_Acc,
    Deceleration       := Mv_Zero_Dec,
    Jerk               := Mv_Zero_Jrk,
    BufferMode         := Mv_Zero_Bm,
    Done               => Mv_Zero_D,
    Busy               => Mv_Zero_Bsy,
    Active             => Mv_Zero_Act,
    CommandAborted     => Mv_Zero_Ca,
    Error              => Mv_Zero_Err,
    ErrorID            => Mv_Zero_ErrID
);
```

# MC_Stop

The MC_Stop instruction decelerates an axis to a stop.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_Stop | Stop | FB |  | MC_Stop_instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>BufferMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.*1 |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/$s^3$.*1 |
| BufferMode | Buffer Mode Selection | _eMC_BUFFER_MODE | 0: _mcAborting | 0*2 | Specify the operation when executing more than one motion instruction.<br>0: Aborting |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 The default value for an enumeration variable is actually not the number, but the enumerator.

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the axis decelerates to a stop and the velocity reaches 0. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the instruction is started. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another MC_Stop instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

- The MC_Stop instruction decelerates an axis from the current velocity to a velocity of 0.
- The deceleration stop operation starts when *Execute* changes to TRUE.
- *CommandAborted* for the instruction that is currently in operation will change to TRUE when MC_Stop is executed.

**Precautions for Correct Use**

You cannot execute this instruction if the *Status.ErrorStop* (Error Deceleration Stopping) variable that gives the status of this axis is TRUE.
Use the MC_ImmediateStop instruction to stop the motion of an axis that is decelerating to a stop for an error.

## Instruction Details

This section describes the instruction in detail.

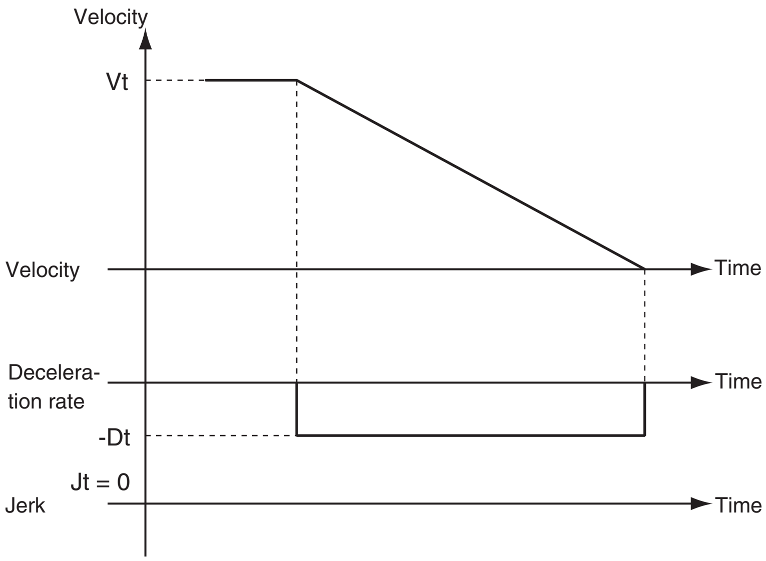
### ● Specifying *Deceleration* and *Jerk*

Set the input variables *Deceleration* and *Jerk* to set the deceleration rate and jerk when decelerating to a stop.

The relationship between the deceleration and velocity when *Jerk* is set to 0 and when it is set to any other value is shown below.

#### *Jerk* Set to 0

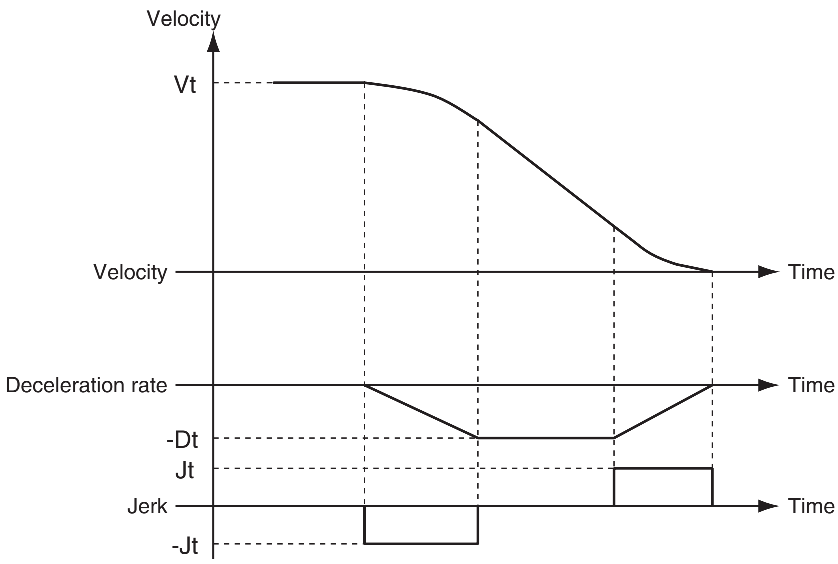
The command value for the velocity is created with deceleration rate Dt.




Vt: Velocity when deceleration starts, Dt: Specified deceleration rate, Jt: Specified jerk

#### *Jerk* Set to Any Value Other Than 0

The command value for the velocity is created based on the current velocity with Dt as the upper limit to the deceleration rate.




Vt: Velocity when deceleration starts, Dt: Specified deceleration rate, Jt: Specified jerk

**Additional Information**

If you set the deceleration to 0, an Immediate Stop instruction is executed. This will stop the axis immediately without decelerating. An immediate stop occurs regardless of the setting of the Acceleration/Deceleration Over Limit Selection only when the deceleration rate is set to 0.

### Specifying *BufferMode* (Buffer Mode Selection)

*BufferMode* specifies how to join the axis motions for this instruction and the previous instruction. *BufferMode* (Buffer Mode Selection) of this instruction is a reserved parameter for future expansion. There is currently only the following setting.

| Buffer Mode Selection | Description |
|---|---|
| Aborting | Aborts the instruction being executed and executes this instruction. |

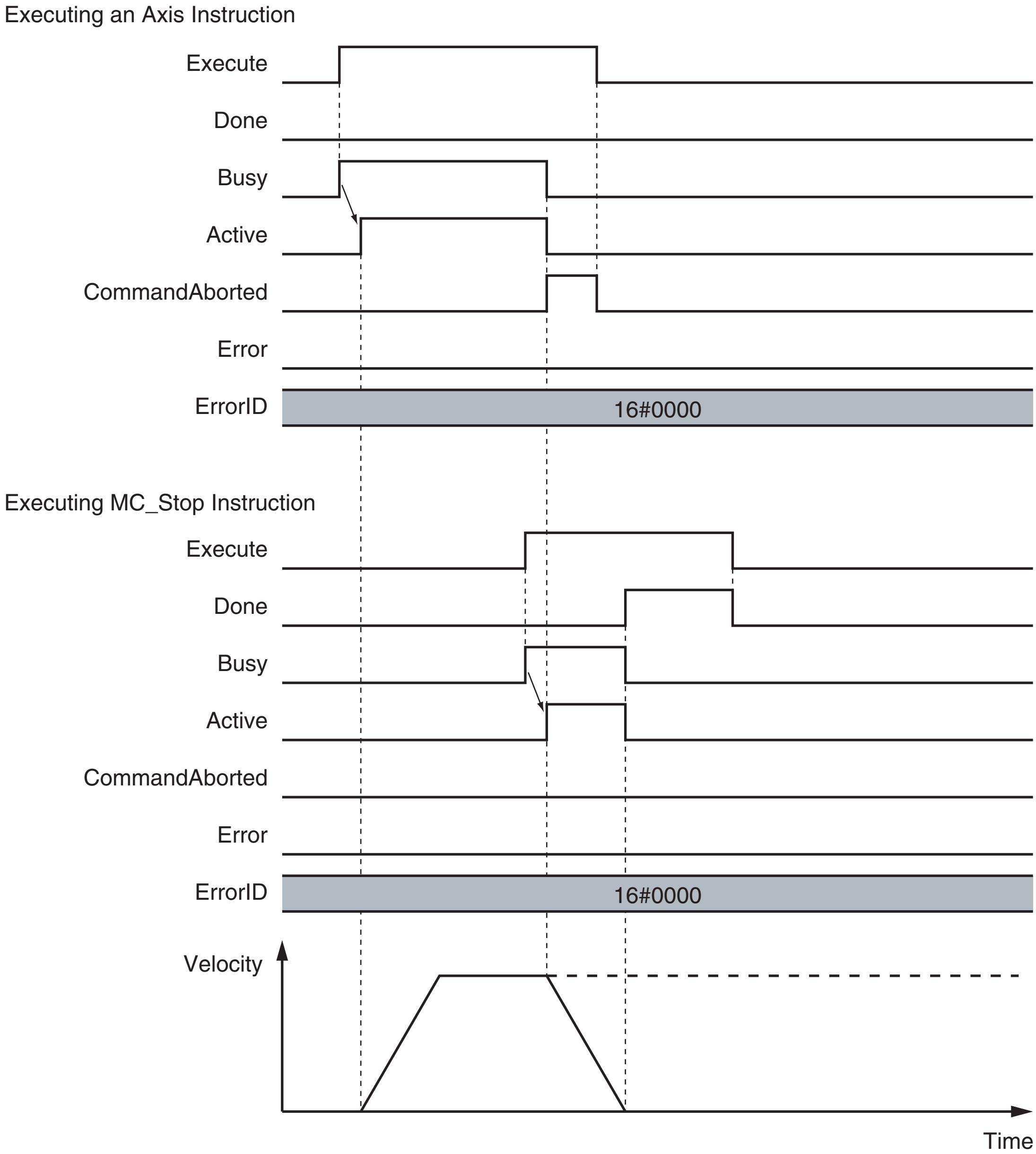
For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### In-position Check

An in-position check is not performed when stopping for this instruction.

## Timing Charts

- *Busy* (Executing) changes to TRUE at the same time as Execute changes to TRUE. *Active* (Controlling) changes to TRUE in the next period.
- *Done* changes to TRUE when a velocity of 0 is reached.
- If another instruction aborts this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing) and *Active* (Controlling) change to FALSE.

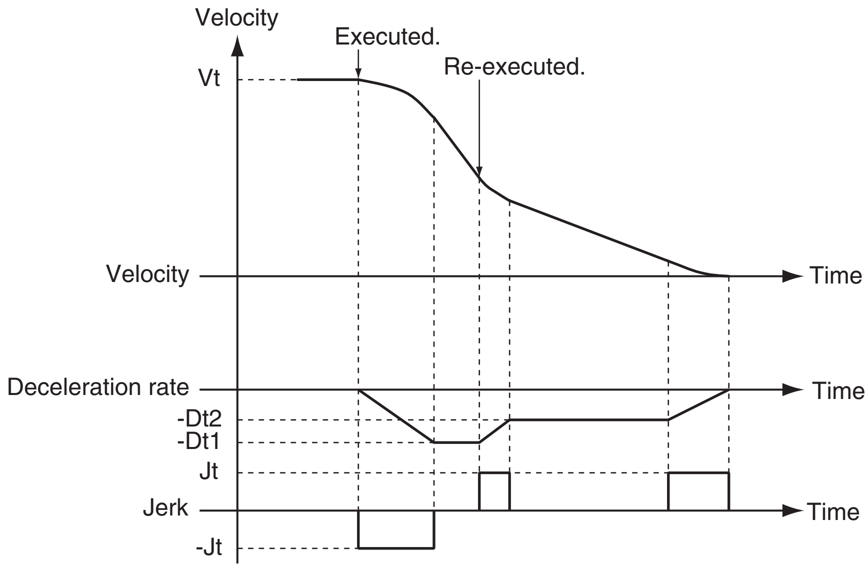


## Re-execution of Motion Instructions

*Deceleration* (Deceleration Rate) changes if *Execute* is changed to TRUE again while this instruction is in execution. The *Jerk* setting is not changed when a motion control instruction is re-executed.

### ● *Jerk* Set to Any Value Other Than 0

The command value for the velocity is created based on the current velocity and deceleration rate, with Dt2 as the upper limit to the deceleration rate after it is changed.




Vt: Velocity when deceleration starts, Dt: Specified deceleration rate, Jt: Specified jerk

## Multi-execution of Motion Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

#### Axes Group Moving

If the MC-Stop instruction is executed for an axis that is in an axes group motion, an error will occur for the axis. An error will occur also for the axes group, and the axes group motion will stop.

#### When the *Status.ErrorStop* (Error Deceleration Stop) Axis Variable Is TRUE

*Status.ErrorStop* (Error Deceleration Stopping) in the Axis Variable is TRUE while there is an error for the axis. If the MC_Stop instruction is not executed normally for an axis that is decelerating to a stop for an error. *CommandAborted* changes to TRUE. Use the MC_ImmediateStop instruction to stop the motion of an axis for which an error occurred.

#### During Execution of the MC_ResetFollowingError (Reset Following Error Counter) Instruction

If the MC_Stop instruction is executed during the MC_ResetFollowingError (Reset Following Error Counter) instruction, *CommandAborted* from the MC_ResetFollowingError instruction changes to TRUE. The MC_Stop instruction is executed. However, *Deceleration* (Deceleration Rate) to the MC_Stop instruction is not used and an immediate stop is performed.

## Execution of Other Instructions during Instruction Execution

- If any of the following is executed while the axis is not decelerating to a stop for an MC_Stop instruction, *Status.Stopping* in the Axis Variable changes to TRUE.
  MC_ResetFollowingError (Reset Following Error Counter) instruction
  An immediate stop for the MC_TouchProbe (Enable External Latch) instruction
  When *Execute* is TRUE for any other MC_Stop instruction
- Operation is as follows for an axis for which *Status.Stopping* (Deceleration Stopping) is TRUE.
  - If single-axis positioning, continuous positioning, synchronized operation, or manual operation is performed, *CommandAborted* from the instruction changes to TRUE.
  - If the MC_ResetFollowingError (Reset Following Error Counter) instruction is executed during MC_Stop instruction execution, *Done* from the MC_Stop instruction changes to TRUE and the MC_ResetFollowingError instruction is executed.
  - Multi-execution of more than one MC_Stop Instruction is possible. *Done* from the first MC_Stop instruction changes to TRUE.
- *Done* of the MC_Stop instruction changes to TRUE when one of the following conditions is met after the MC_Stop instruction is executed.
  - When the *Enable* input variable for the MC_Power instruction changes to FALSE (when the Servo is turned OFF)
  - When *_mcImmediateStop* is selected for the StopMode input variable to the MC_TouchProbe (Enable External Latch) instruction, the trigger condition is met, and the OMRON G5-series Servo Drive stops immediately

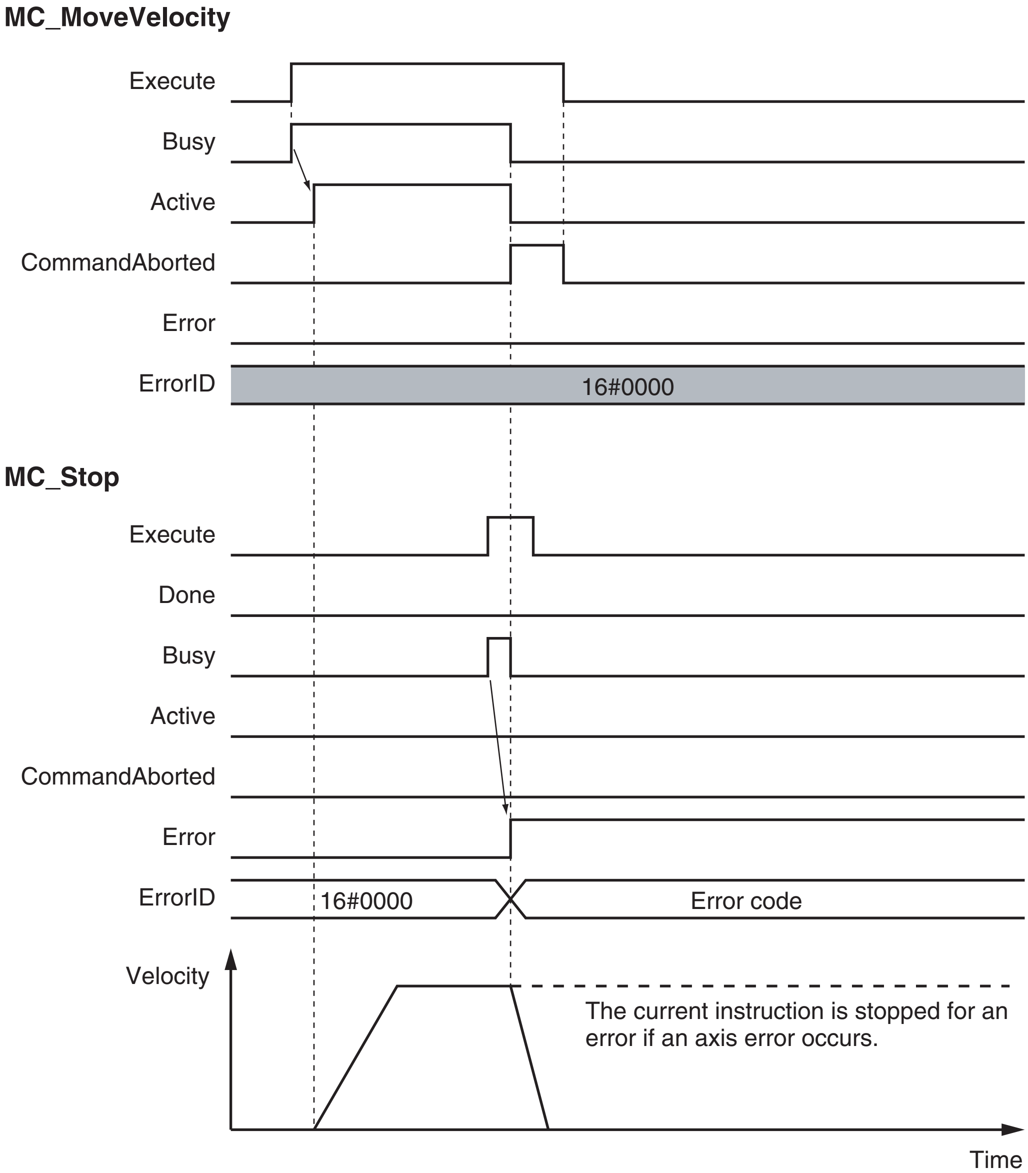
## Errors

Operation will stop if an error (e.g., axis error) occurs during instruction execution.
Specify the stopping method in the axis parameters. The stopping method can be immediate stop, deceleration stop, or Servo OFF. If you specify a deceleration stop, the axis will continue decelerating until it stops.

For details on setting the Stop Mode in the axis parameters, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Timing Chart When Error Occurs

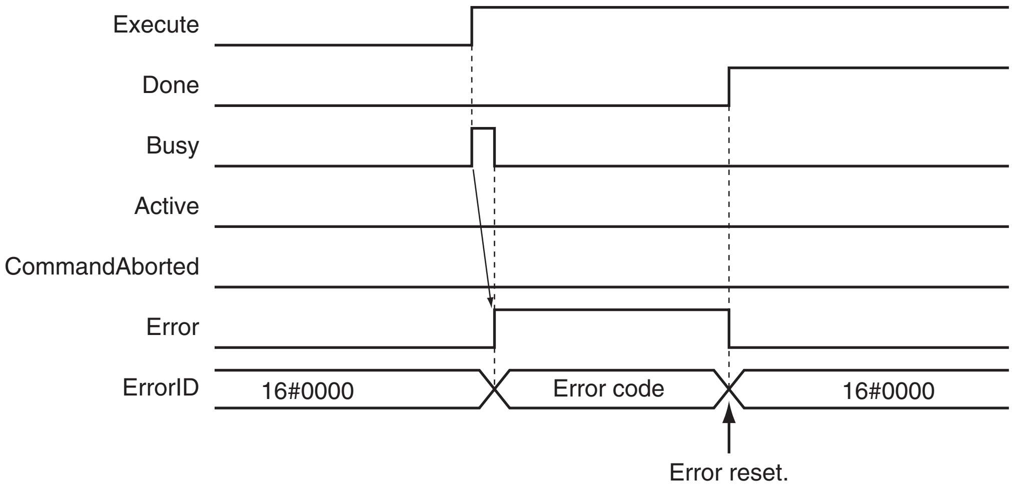
If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).



## ● Operation When Error Is Cleared

If an error occurs for this instruction and the error is reset while *Execute* is TRUE, *Error* changes to FALSE and *Done* changes to TRUE. *Status.Stopping* (Deceleration Stopping) in the Axis Variable changes to TRUE in the same way as for normal execution of the deceleration stop.
Reset errors after the axis has stopped. Do not reset errors during axis motion.

**MC_Stop**

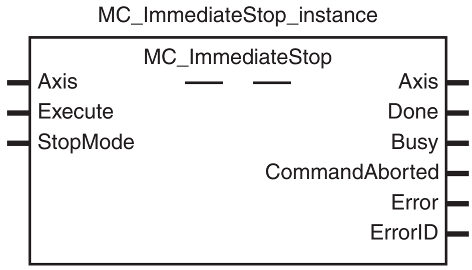



## ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_ImmediateStop

The MC_ImmediateStop instruction stops an axis according to the stopping mode that is set with the *StopMode* (Stopping Mode Selection) input variable regardless of the status of the axis.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_ImmediateStop | Immediate Stop | FB |  | MC_ImmediateStop_instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>StopMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| StopMode | Stopping Mode Selection | _eMC_ STOP_M ODE | 1: _mcImmediateStop<br>2: _mcImmediateStopFEReset<br>3: _mcFreeRunStop | 1* | Select the stopping mode.<br>1: Perform an immediate stop.<br>2: Perform an immediate stop and reset the following error counter.<br>3: Perform an immediate stop and turn OFF the Servo. |

* The default value for an enumeration variable is actually not the number, but the enumerator.

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the axis has decelerated to a stop. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another deceleration stop instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

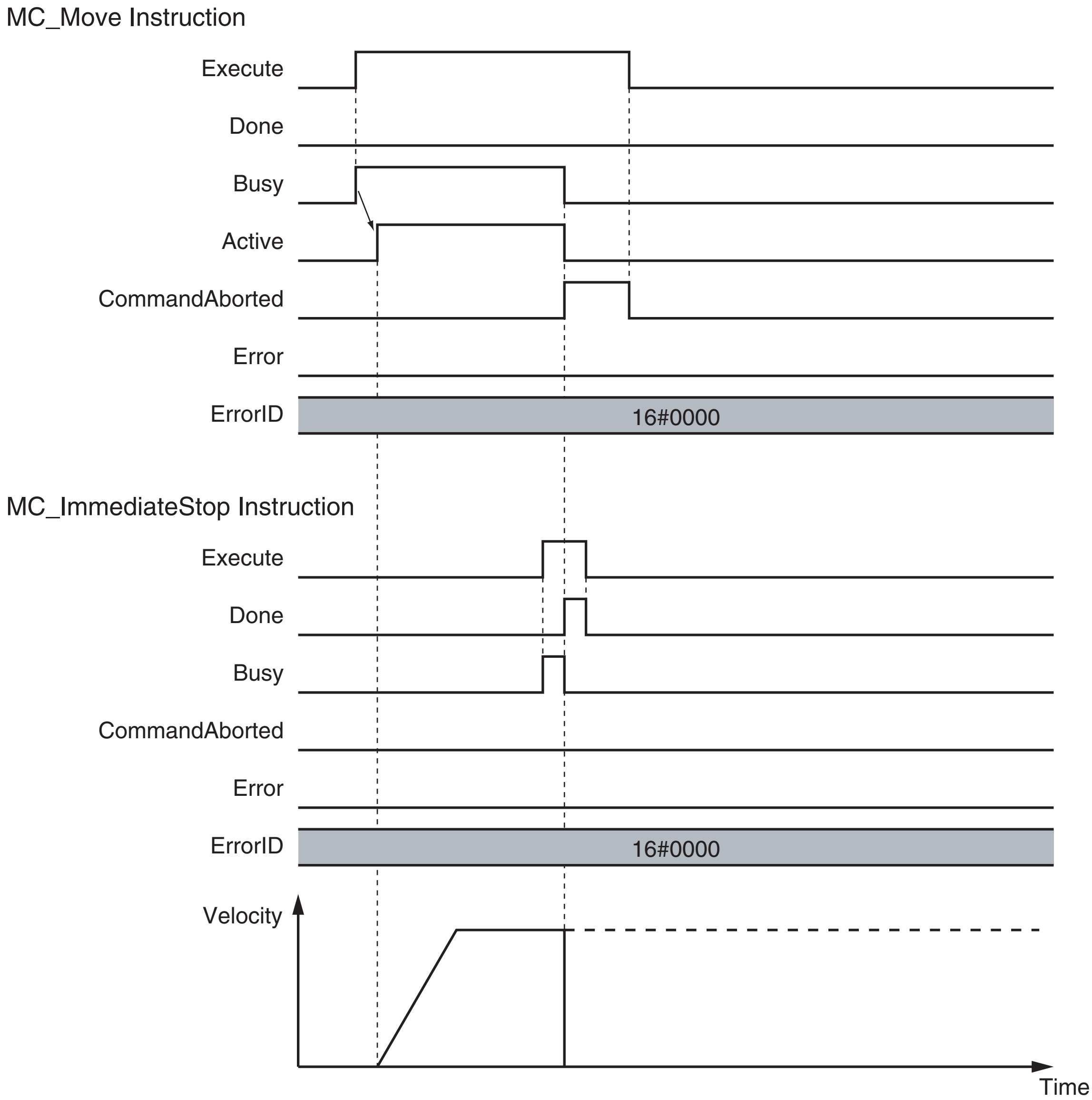
- You can execute the MC_ImmediateStop instruction under any conditions.
  For example, you can use this instruction to stop an axis immediately even if it is decelerating to a stop for an error.
  You cannot execute the MC_Stop while *Status.ErrorStop* (Error Deceleration Stopping) is TRUE, but you can execute the MC_ImmediateStop.
- When this instruction is executed, the axis stops immediately according to *StopMode* (Stopping Mode Selection). *CommandAborted* changes to TRUE for the instruction that is currently in operation.
- *Status.ErrorStop* (Error Deceleration Stopping) in the axis status changes to TRUE when this instruction is executed and an Immediate Stop Instruction Executed error (error code: 5485 hex) occurs.

**Precautions for Correct Use**

Refer to *Precautions for Master and Auxiliary Axes in Synchronized Control* on page 1-7 for precautions on the master axis.

## Timing Charts

- *Busy* (Executing) changes to TRUE when *Execute* changes to TRUE.
- *Done* changes to TRUE when processing of the Immediate Stop instruction is completed.




## Re-execution of Motion Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

#### Axes Group Moving

If the instruction is executed for an axis that is in an axes group motion, an error will occur for the axis and an immediate stop is performed. An error will also occur for the axis group, and the axis group motion will stop.

#### When the *Status.Stopping* (Deceleration Stopping) Axis Variable Is TRUE

*Status.Stopping* (Deceleration Stopping) in the Axis Variable changes to TRUE in the following cases.

- While the axis is decelerating for the MC_Stop Instruction
- During execution of the MC_ResetFollowingError (Reset Following Error Counter) instruction
- During an immediate stop for the MC_TouchProbe (Enable External Latch) instruction
- While *Execute* is TRUE for one or more MC_Stop instructions

You can execute this instruction for an axis for which *Status.Stopping* (Deceleration Stopping) is TRUE.
When this instruction is executed, *CommandAborted* from the following instructions changes to TRUE.

- MC_Stop instruction
- MC_ResetFollowingError (Reset Following Error Counter) instruction
- MC_TouchProbe (Enable External Latch) instruction (during the immediate stop)

#### When the *Status.ErrorStop* (Error Deceleration Stop) Axis Variable Is TRUE

*Status.ErrorStop* (Error Deceleration Stopping) in the axis status is TRUE while there is an error for the axis.
You can execute this instruction even for an axis that is decelerating to a stop for an error.

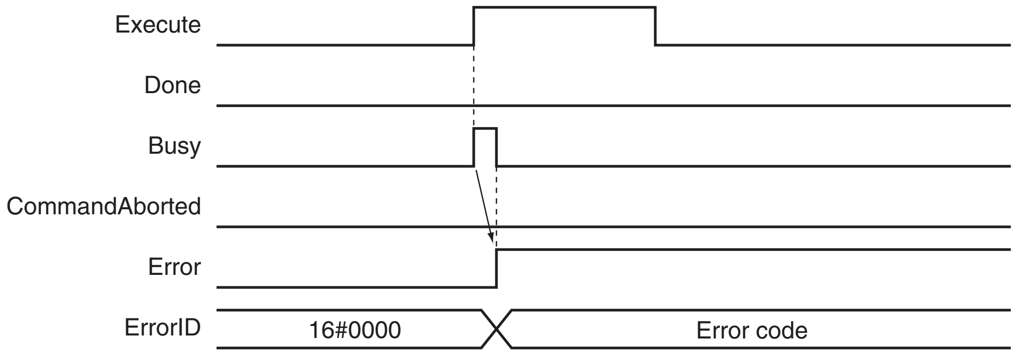
## Error

The axis will stop immediately even if an error (e.g., axis error) occurs during instruction execution.

For details on setting the Stop Mode in the axis parameters, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Timing Chart When Error Occurs

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

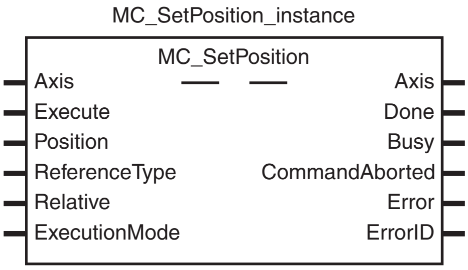



### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_SetPosition

The MC_SetPosition instruction changes the command current position or the actual current position of an axis as required.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_SetPosition | Set Position | FB |   | MC_SetPosition_instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>Position :=*parameter,*<br>ReferenceType<br>:=*parameter,*<br>Relative :=*parameter,*<br>ExcutionMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Position | Target Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the absolute target position. The unit is command units.*1 |
| ReferecneType | Position Type Selection | _eMC_REFERENCE_TYPE | 0: _mcCommand<br>1: _mcFeedback | 0*2 | Specifies the position type.<br>0: Command position (servo axis or virtual servo axis)<br>1: Actual position (encoder axis or virtual encoder axis) |
| Relative (Reserved) | Relative Position Selection | BOOL | TRUE or FALSE | --- | (Reserved) |
| MC_Execution Mode (Reserved) | Execution Mode | _eMC_EXECUTION_MODE | 0: _mcImmediately | 0*2 | (Reserved) |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When changing the command current position and the actual current position are completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE. |
| CommandAborted | • When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

- This instruction changes the command current position of the Servo axis to the specified target position. If you execute this instruction on an encoder axis, the actual current position will change to the specified target position.
- Specify the target position in absolute coordinates.
- The actual current position changes at the same time as the command current position changes. The following error is kept the same before and after the change. If you execute this instruction on a command servo axis, the difference between the actual position and the actual current position will remain the same before and after the change. Because of this, after you execute this instruction, the actual current position of the axis takes the value calculated by the following equation.

  Actual current position after change = Target position – Following error before change

- If you specify the actual position for a servo axis or the command position for an encoder axis, a position type error will occur.

- When the Count Mode is set to Rotary Mode, set the target position to a value between the lower limit and upper limit of the ring counter. A ring counter error will occur if the target position is outside this range.
- When the Count Mode is set to Linear Mode, you can set the target position to a value outside the range defined by the software limits.
- You can use this instruction for an axis that is stopped or in motion.

**Precautions for Correct Use**

Home is undefined for the specified axis after this instruction ends. Because of this, you cannot execute following functions or instructions after this instruction ends.

- Software limits
- MC_MoveZeroPosition (High-speed Home) instruction
- Multi-axes coordinated control instructions (linear or circular interpolation)

### ● *ReferenceType* (Position Type Selection)

- Set *ReferenceType* to the command position to use a servo axis or virtual servo axis.
- Set *ReferenceType* to the actual position to use an encoder axis or virtual encoder axis.

### ● Relationship between Axis Types and Position Types

The relationship between the axis types that you can monitor and position types that is monitored is shown below.

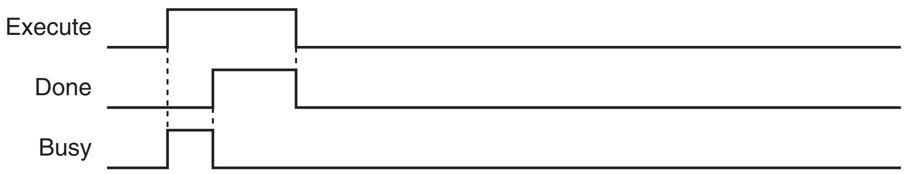
| Axis Type | ReferenceType | |
|---|---|---|
| | _mcCommand | _mcFeedback |
| Servo axis | OK | No |
| Encoder axis | No* | OK |
| Virtual servo axis | OK | OK |
| Virtual encoder axis | No* | OK |

* A Position Type Selection Out of Range error (error code 5430 hex) occurs when the instruction is executed.

## Timing Charts

### ● Execution While Axis Is Stopped

The actual position starts changing when *Execute* changes to TRUE. *Busy* (Executing) changes to TRUE when *Execute* changes to TRUE. *Done* changes to TRUE after the actual position is changed.



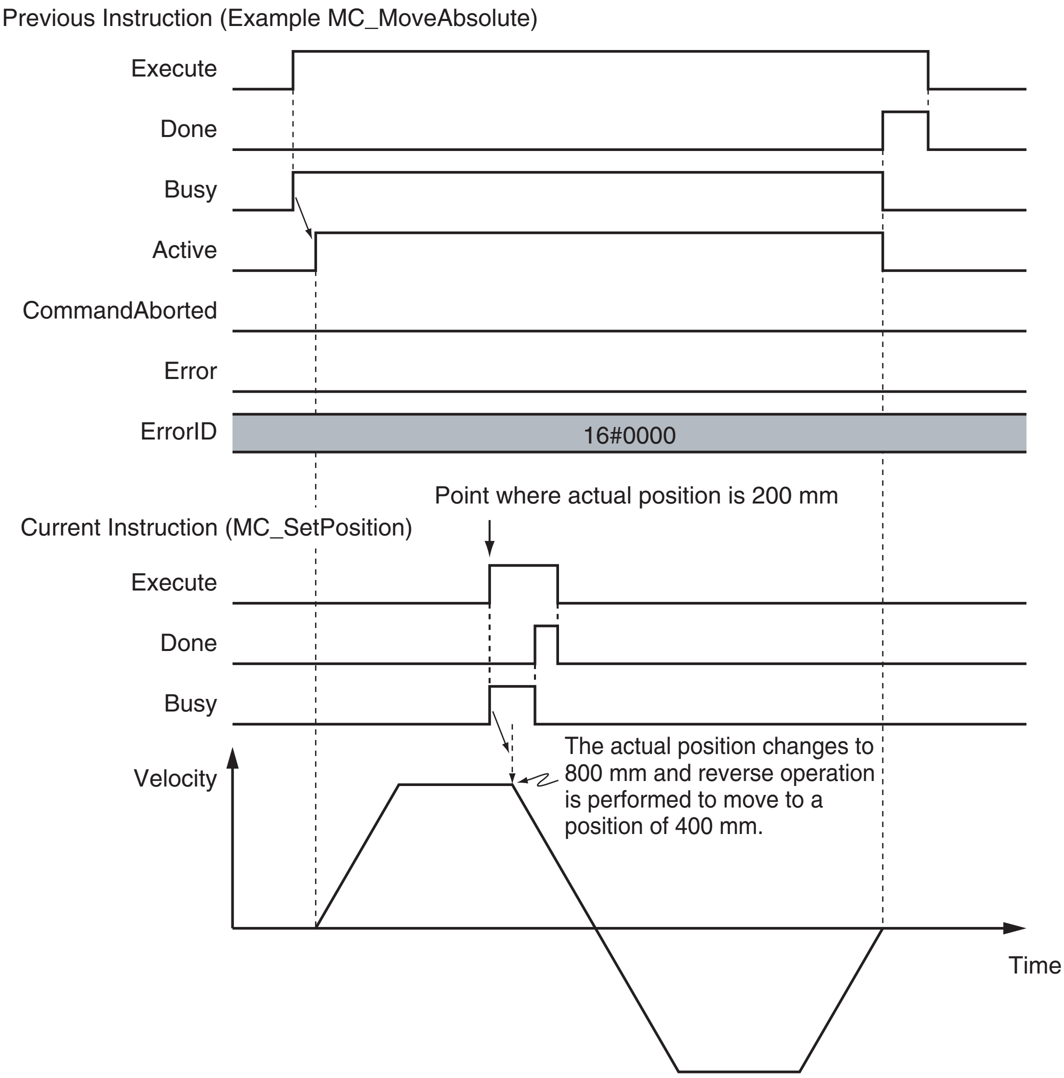



MC_SetPosition
3
Function

### ● Execution While Axis Is in Motion

If you execute this instruction while positioning to an absolute position, the target value will change according to the change in position.

As an example, the axis operation and timing chart are shown below for a situation where the actual position is changed from 200 mm to 800 mm while the axis is moved to 400 mm for an MC_MoveAbsolute (Absolute Positioning) instruction. The axis will move in the negative direction because the actual value is 800 mm and the target value is 400 mm. As shown in the following figure, even if the actual position is changed, the MC_MoveAbsolute (Absolute Positioning) instruction will move the axis from the new actual position to the specified target position. When the specified target position is reached, *Done* changes to TRUE.




#### Additional Information

- If you execute this instruction while the MC_MoveRelative (Relative Positioning) or MC_MoveVelocity (Velocity Control) instruction is in execution, the actual position will change. However, if you execute this instruction while the MC_MoveRelative (Relative Positioning) or MC_MoveVelocity (Velocity Control) instruction is in execution, the positioning operation is not affected.
- If there is a buffered instruction, positioning is performed for the position after the change when the buffer is switched.

## Re-execution of Motion Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

You cannot use the MC_SetPosition instruction on an axis for which any of the following instructions is being executed. A multi-execution of instructions error will occur if it is executed.

| | |
|---|---|
| MC_Home (Home) instruction | MC_CombineAxes (Combine Axes) instruction |
| MC_CamIn (Start Cam Operation) Instruction | MC_MoveZeroPosition (High-speed Home) instruction |
| MC_GearIn (Start Gear Operation) Instruction | MC_TorqueControl (Torque Control) Instruction |
| MC_GearInPos (Positioning Gear Operation) instruction | MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control) instruction |
| MC_MoveLink (Synchronous Positioning) instruction | |

**Precautions for Correct Use**

Do not use the MC_SetPosition instruction for *Master* (Master Axis) that is in synchronized operation for instructions such as MC_GearIn (Start Gear Operation). If it is used for a *Master* (Master Axis), as soon as the command current position and the actual current position of the *Master* (Master Axis) are changed, the *Slave* (Slave Axis) will detect that the *Master* (Master Axis) has moved. It will then begin moving corresponding to the *Master* (Master Axis) travel distance. For this reason, the *Slave* (Slave Axis) may change suddenly or the cam motion may end suddenly.
If you want to use the MC_SetPosition instruction for the *Master* (Master Axis), disable the relationship between the *Master* (Master Axis) and slave axes before executing the instruction. Execute an instruction such as the MC_GearOut (End Gear Operation) instruction to disable the relationship between the *Master* (Master Axis) and *Slave* (Slave Axis).
Refer to *Precautions for Master and Auxiliary Axes in Synchronized Control* on page 1-7 for precautions on the master axis.

### ● Execution of Other Instructions during Instruction Execution

If another MC_SetPosition instruction is executed while there is one already in execution, the last instruction takes priority.
In this case, *Done* for the first MC_SetPosition instruction will change to TRUE, but the change to the set position for the first instruction is not completed.

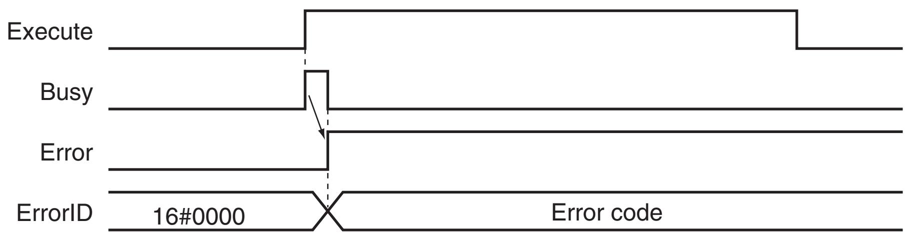
## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

**Additional Information**

If you execute this instruction while the axis is in motion, the instruction of the axis currently in motion will be restarted and the data required for positioning will be recalculated. If an error occurs, it will be for the instruction of the axis currently in motion rather than for this instruction.

### ● Timing Chart When Error Occurs

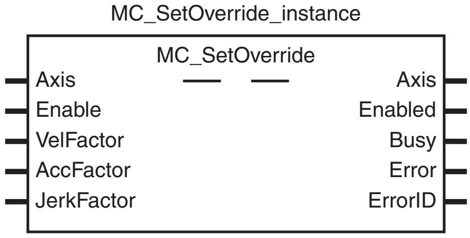



### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_SetOverride

The MC_SetOverride instruction changes the target velocity for an axis.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_SetOverride | Set Override Factors | FB |  | MC_SetOverride_instance (<br>Axis :=*parameter,*<br>Enable :=*parameter,*<br>VelFactor :=*parameter,*<br>AccFactor :=*parameter,*<br>JerkFactor :=*parameter,*<br>Enabled =>*parameter,*<br>Busy =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Enable | Enable | BOOL | TRUE or FALSE | FALSE | The override factors are enabled when *Enable* is TRUE. The override factors return to 100% when *Enable* changes to FALSE. |
| VelFactor | Velocity Override Factor | LREAL | 0 to 500 | 100 | Specify the velocity override factor. The valid range of the override factors is between 0.01and 500.00. Values above 500.00 are treated as 500 and values less then 0.01 (including negative values) are treated as 0.01. The override factor will be 0 only when 0 is specified. The unit is %. |
| AccFactor (Reserved) | Acceleration/Deceleration Override Factor | LREAL | 0 to 500 | 100 | (Reserved) |
| JerkFactor (Reserved) | Jerk Override Factor | LREAL | 0 to 500 | 100 | (Reserved) |

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Enabled | Enable | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Enabled | When this instruction is started. | • After one period when *Enable* is FALSE.<br>• When *Error* changes to TRUE. |
| Busy | When *Enable* changes to TRUE. | • When *Enable* changes to FALSE.<br>• When *Error* changes to TRUE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

- The MC_SetOverride instruction changes override factors related to the target velocity of the axis. Changes the target velocity of the axes in operation by changing the override.
- The override factors apply only to the following instructions.

| | |
|---|---|
| MC_Move (Positioning) instruction | MC_MoveJog (Jog) instruction |
| MC_MoveAbsolute (Absolute Positioning) instruction | MC_MoveFeed (Interrupt Feeding) instruction |
| MC_MoveRelative (Relative Positioning) instruction | MC_MoveZeroPosition (High-speed Home) instruction |
| MC_MoveVelocity (Velocity Control) instruction | |

- The new target velocity is found with the following equation.

  The target velocity after the change = Target velocity of the current instruction × Override factor
- The unit for override factors is %. A setting of 100 indicates 100%.
- If the target velocity that results from the override exceeds the maximum velocity set in the axis parameters, the maximum velocity is used.
- The axis will accelerate or decelerate to the target velocity that results from the override.
- If the velocity override factor is set to 0, the target velocity will be 0. Axis operation will decelerate to a velocity of 0, and operation will continue. If you want to pause the axis motion while maintaining the operation status, set the override factor to 0.
  *Status.Discrete* and *Status.Continuous* in the Axis Variable do not change at this time.
- The override factors are always updated when the instruction is executed as long as *Enable* remains TRUE.
- The override factors apply to operation commands for new target velocities, e.g., when you start a stopped axis, re-execute a motion instruction, or perform multi-execution of motion control instructions.
- The override factors will return to 100% when *Enable* changes to FALSE.
- If an axis error occurs during MC_SetOverride instruction execution, the value of *Enable* for the MC_SetOverride instruction remains TRUE.

**Precautions for Correct Use**

When *Enable* to this instruction changes to FALSE, *Enabled* and *Busy* from this instruction change to FALSE.
The axis will accelerate or decelerate to a velocity with a 100% override factor.

**Additional Information**

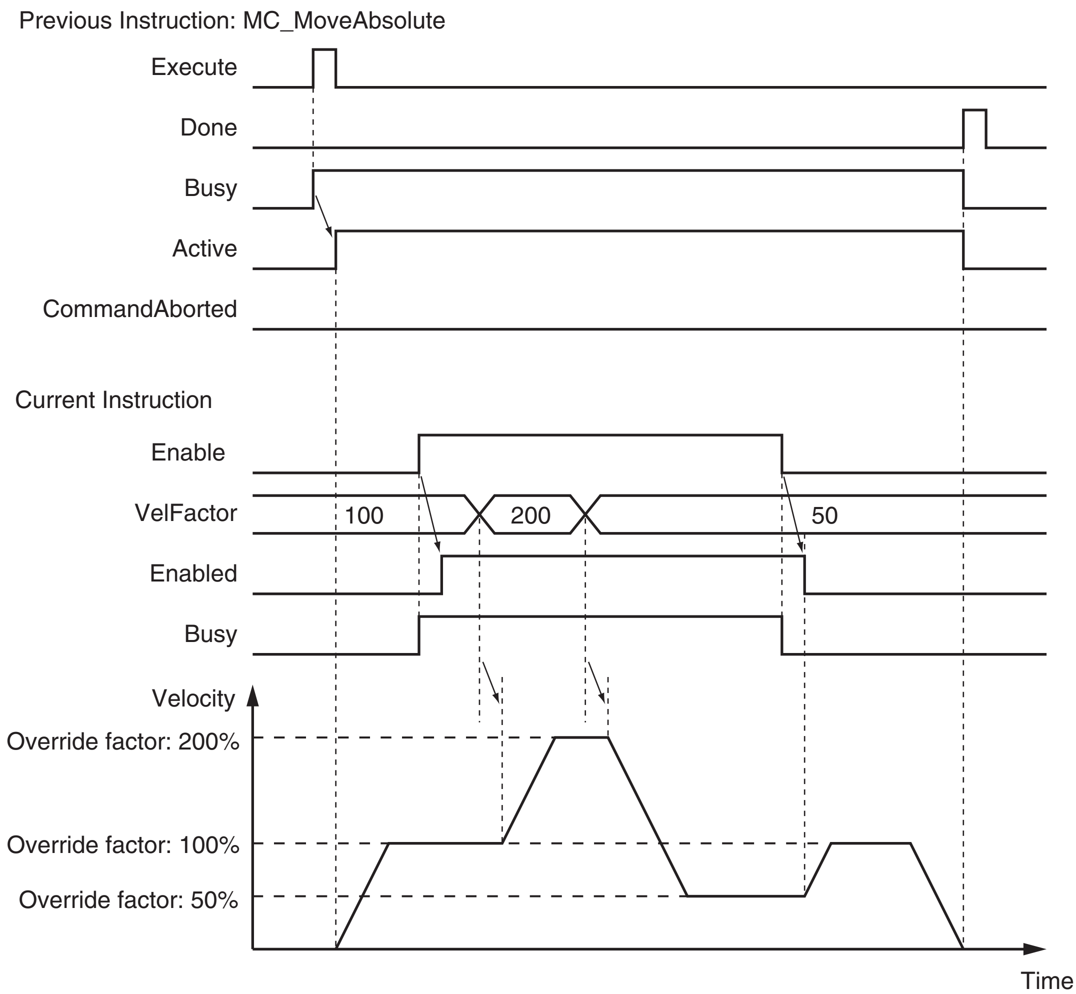
**Influence on Other Instructions**

Use this instruction to temporarily change the target velocities of other instructions.
This instruction has no affect on instructions to which a target velocity is not input or instructions for which the target velocity is updated every period, such the Cyclic Synchronous Velocity Control instruction.

## Timing Charts

### ● Overriding the MC_MoveAbsolute Instruction

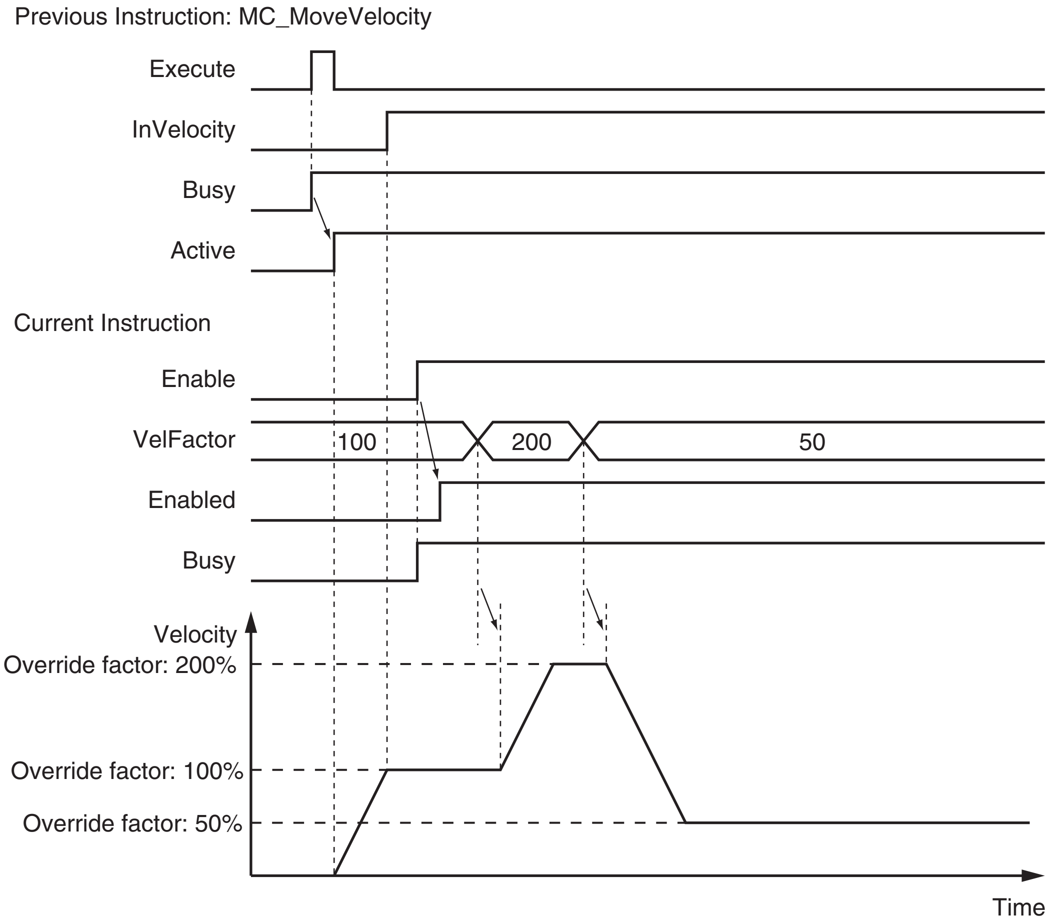
An example of a time chart for using the Set Override Factors instruction for the MC_MoveAbsolute (Absolute Positioning) instruction is given below.




If the MC_SetOverride instruction is disabled, the target velocity returns to an override factor of 100%.

- **Overrides for the MC_MoveVelocity (Velocity Control) Instruction**

An example of a time chart for using the Set Override Factors instruction for the MC_MoveVelocity (Velocity Control) instruction is given below. After *InVelocity* (Target Velocity Reached) changes to TRUE, it will stay TRUE even if the velocity changes.




## Re-execution of Motion Instructions

You cannot re-execute enable-type motion control instructions.

## Multi-execution of Motion Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

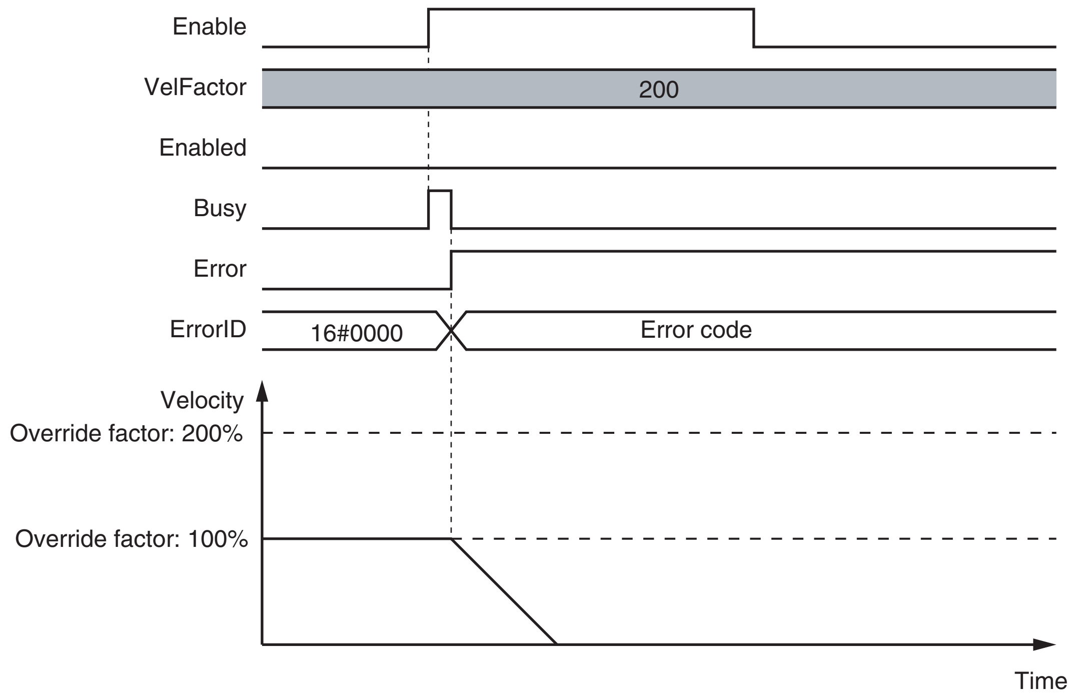
- **Multi-execution of MC_SetOverride Instructions**

If another instance of MC_SetOverride is executed during MC_SetOverride instruction execution, the last MC_SetOverride instance that is executed takes priority. *Enable* will be TRUE for both instructions.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE. If a minor fault occurs, the axis will stop.
You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code). If you remove the cause of error, *Error* will change to FALSE.

### ● Timing Chart When Error Occurs

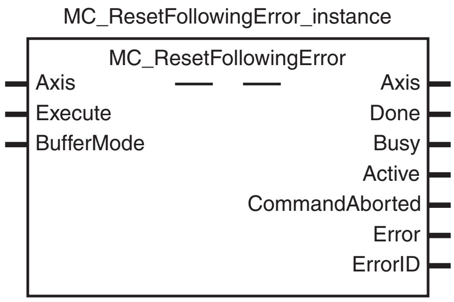



### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_ResetFollowingError

The MC_ResetFollowingError instruction resets the following error between the command current position and the actual current position.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_ResetFollowingError | Reset Following Error Counter | FB |  | MC_ResetFollowingError_instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>BufferMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| BufferMode | Buffer Mode Selection | _eMC_ BUFFER_ MODE | 0: _mcAborting | 0* | Specify the operation when executing more than one motion instruction.<br>0: Aborting |

* The default value for an enumeration variable is actually not the number, but the enumerator.

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When resetting the following error counter is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the instruction is started. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When the MC_ResetFollowingError instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

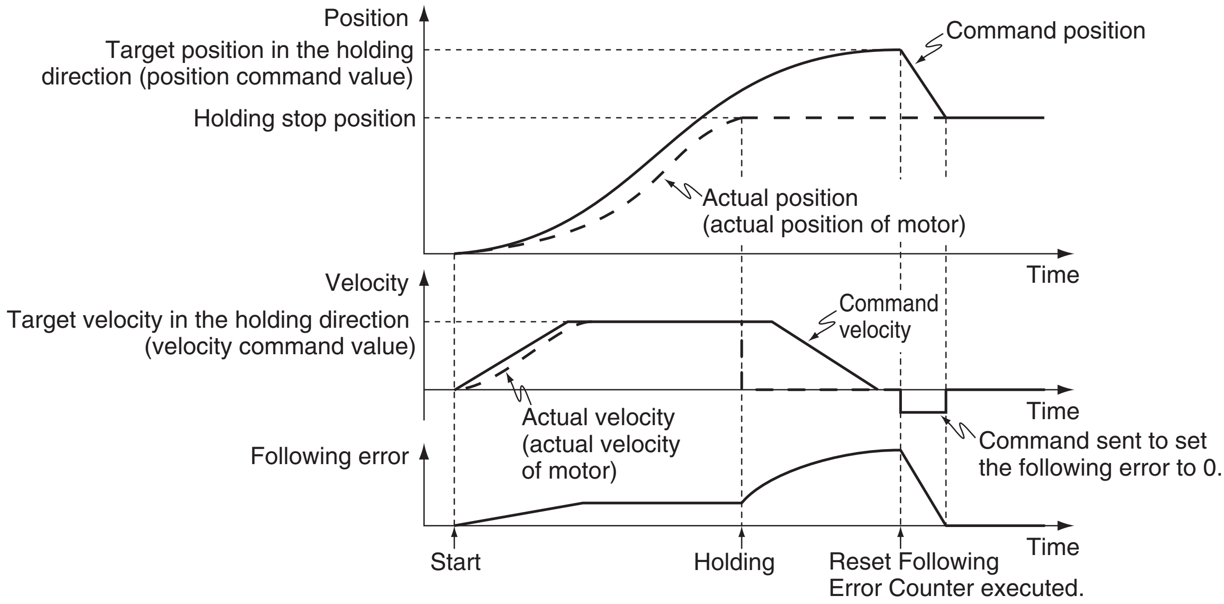
## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

- The MC_ResetFollowingError instruction resets the following error between the command current position and the actual current position in the MC Function Module to 0 in Cyclic Synchronous Positioning Mode.
- When *Execute* changes to TRUE, the actual current position at that point is used as the command position (i.e., the target position).
  For example, when a following error occurs in the holding operation shown below, you can execute this instruction to implement a position command in the reverse direction and therefore set the following error to 0.
  *CommandAborted* for the instruction for which the following error occurred changes to TRUE and instruction execution is aborted.




- When the following error is set to 0, the maximum velocity that is set in the axis parameters is used to implement a position command. The maximum acceleration and deceleration rates are not used.
- When the command to the new target position is completed, the *Done* output variable changed to TRUE.
- This instruction implements a command position in the reverse direction to the direction in which the following error occurred, but the Operation at Reversing axis parameter is not used.

**Precautions for Correct Use**

- Execute this instruction only when the axis velocity is low.
  This instruction implements a command value in the opposite direction to the previous instruction (e.g., in the opposite direction to the holding direction). If the axis speed is too high when this instruction is executed, the controlled system may be subjected to shock.
- Before you execute this instruction for a vertical axis, for which constant torque is required, make sure that the torque will not become insufficient.
- **Executing this Instruction for the Master Axis of Synchronized Control**

  If this instruction is executed for the master axis of synchronous control when the command position is used as the synchronization data for the master axis, the slave axis will move in the reverse direction according to the gear ratio or cam data variable.
  Refer to *Precautions for Master and Auxiliary Axes in Synchronized Control* on page 1-7 for precautions on the master axis.

## Instruction Details

This section describes the instruction in detail.

### ● Applicable Axes and Execution Condition

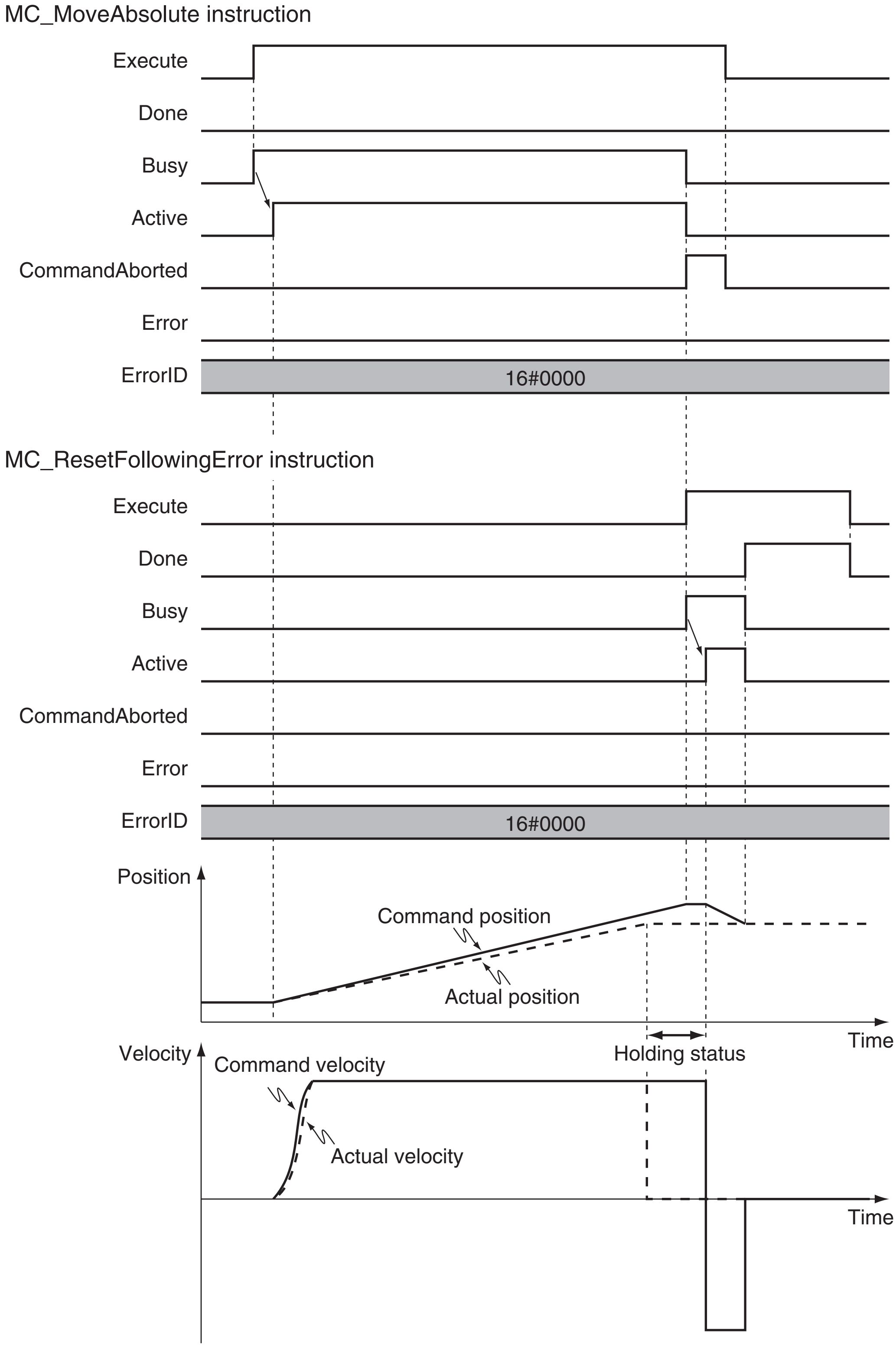
- You can use this instruction for servo and virtual servo axes in the following cases.
  During single-axis position control
  During the MC_MoveVelocity (Velocity Control) instruction
  During synchronized control
- An error occurs if the instruction is executed for an encoder or virtual encoder axis.

### ● Axis Variable Status

*Status.Stopping* (Deceleration Stopping) in the Axis Variable status changes to TRUE.

## Timing Charts

Timing charts for when this instruction is executed during holding status for the MC_MoveAbsolute (Absolute Positioning) instruction are given below.



## Re-execution of Motion Control Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

You can switch to this instruction or buffer this instruction if you execute it during execution of another instruction.
You can buffer one instruction per axis.
Specify the operation of this instruction by using *BufferMode* (Buffer Mode Selection) for multi-execution of instructions.

| Buffer Mode Selection | Description |
|---|---|
| Aborting | Cancels the instruction being executed and switches to this instruction. |

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

#### Relation to MC_Stop Instruction

If the MC_ResetFollowingError (Reset Following Error Counter) instruction is executed during MC_Stop instruction execution, *Done* from the MC_Stop instruction changes to TRUE and the MC_ResetFollowingError instruction is executed.

### ● Execution of Other Instructions during Instruction Execution

The axis decelerates to a stop when this instruction is executed.
Execute an instruction for which multi-execution is supported while the axis is decelerating.
If an instruction for which multi-execution is not supported is executed, *CommandAborted* for the instruction changes to TRUE.

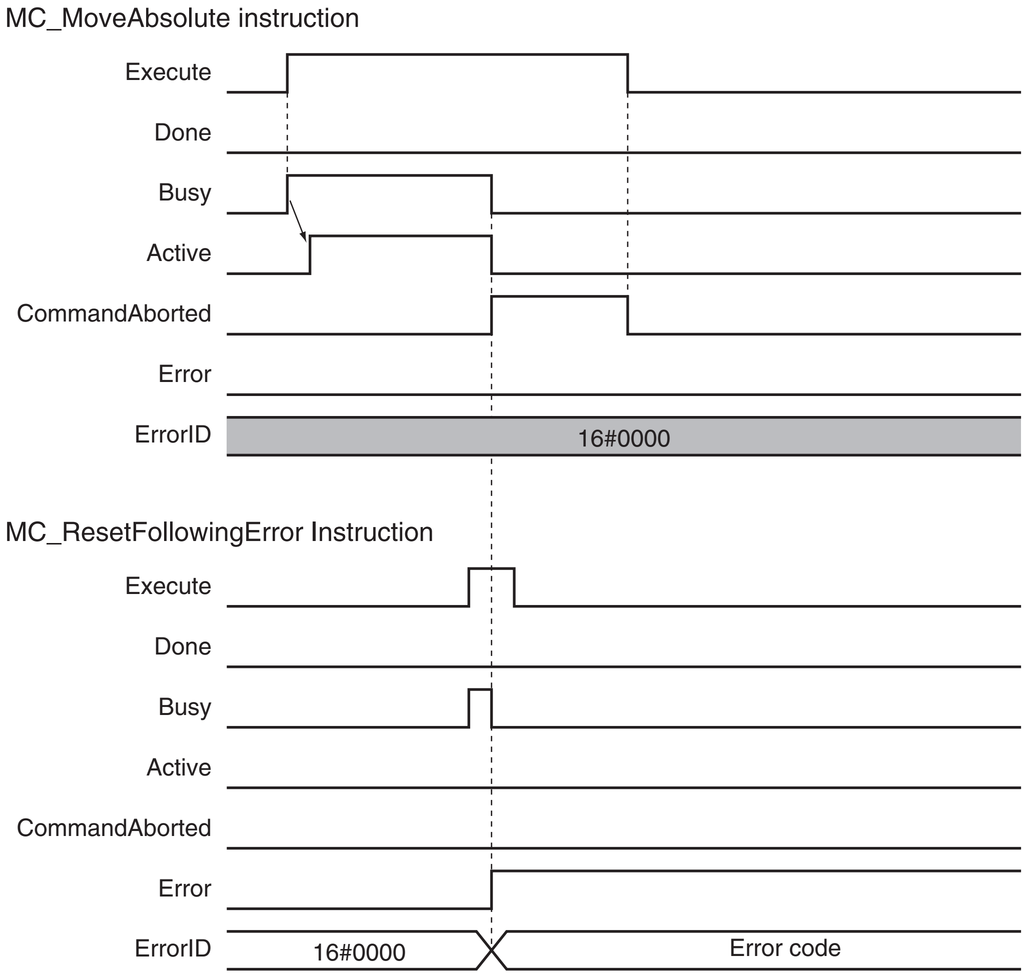
#### Relation to MC_Stop Instruction

If the MC_Stop instruction is executed during the MC_ResetFollowingError (Reset Following Error Counter) instruction, *CommandAborted* from the MC_ResetFollowingError instruction changes to TRUE. The MC_Stop instruction is executed. However, *Deceleration* (Deceleration Rate) to the MC_Stop instruction is not used and an immediate stop is performed.

## Errors

If an error occurs during instruction execution, an immediate stop is performed and *CommandAborted* changes to TRUE. Also, if an error occurs when this instruction is executed, *Error* will change to TRUE. You can find out the cause of the error by referring to the value output to *ErrorID* (Error Code).

Timing charts for when the MC_MoveAbsolute (Absolute Positioning) instruction is executed and an error occurs when the MC_ResetFollowing Error is executed during holding status are given below.

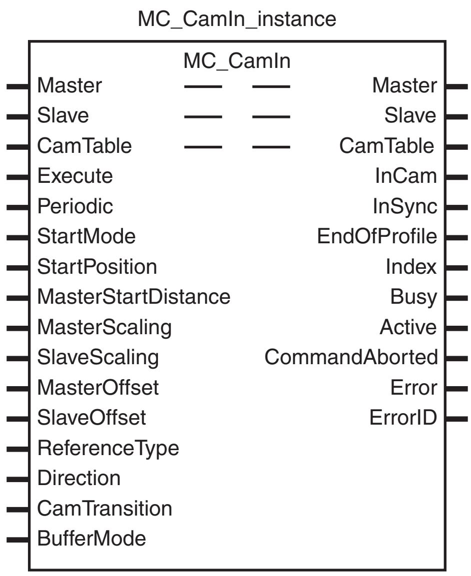



### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_CamIn

The MC_CamIn instruction starts a cam operation by using a specified cam table.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_CamIn | Start Cam Operation | FB |  | MC_CamIn_instance (<br>Master :=*parameter,*<br>Slave :=*parameter,*<br>CamTable :=*parameter,*<br>Execute :=*parameter,*<br>Periodic :=*parameter,*<br>StartMode :=*parameter,*<br>StartPosition :=*parameter,*<br>MasterStartDistance :=*parameter,*<br>MasterScaling :=*parameter,*<br>SlaveScaling :=*parameter,*<br>MasterOffset :=*parameter,*<br>SlaveOffset :=*parameter,*<br>ReferenceType :=*parameter,*<br>Direction :=*parameter,*<br>CamTransition :=*parameter,*<br>BufferMode :=*parameter,*<br>InCam =>*parameter,*<br>InSync =>*parameter,*<br>EndOfProfile =>*parameter,*<br>Index =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Periodic | Periodic Mode | BOOL | TRUE or FALSE | FALSE | Specify whether to execute the specified cam table periodically or only once.<br>FALSE: Non-periodic<br>TRUE: Periodic |
| StartMode | Start Mode | _eMC_ START_MODE | 0: _mcAbsolutePosition<br>1: _mcRelativePosition | 0*1 | Specify the coordinates used by *MasterStartDistance*.<br>0: Absolute position<br>1: Relative position |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| StartPosition | Cam Table Start Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the starting point of the cam table (0 phase) as an absolute position of the master axis. The unit is command units.*2 |
| MasterStart Distance | Master Following Distance | LREAL | Negative number, positive number, or 0 | 0 | Specify the position of the master axis when the following axis starts the cam motion. When absolute positioning is specified with *StartMode*, specify the absolute position of the master axis. When relative positioning is specified, specify a relative distance from the StartPosition (Cam Table Start Position). The unit is command units.*2 |
| Master Scaling | Master Coefficient | LREAL | Positive value (>0.0) | 1.0 | The phase of the master axis is extended or contracted by using the specified scale. |
| SlaveScaling | Slave Axis Coefficient | LREAL | Positive value (>0.0) | 1.0 | The displacement of the slave axis is extended or contracted by using the specified scale. |
| MasterOffset | Master Offset | LREAL | Negative number, positive number, or 0 | 0 | The phase of the master axis is shifted by using the specified offset value. |
| SlaveOffset | Slave Offset | LREAL | Negative number, positive number, or 0 | 0 | The displacement of the slave axis is shifted by using the specified offset value. |
| Reference Type | Position Type Selection | _eMC_REFERENCE _ TYPE | 0: _mcCommand<br>1: _mcFeedback<br>2: _mcLatestCommand | 0*1 | Specify the position type of the master axis.<br>0: Command position (value calculated in the previous primary period)<br>1: Actual position (value obtained in the same primary period)<br>2: Command position (value calculated in the same primary period) |
| Direction | Direction | _eMC_DIRECTION | 0: _mcPositiveDirection<br>2: _mcNegativeDirection<br>4: _mcNoDirection | 4*1 | The slave axis cam moves when the master axis moves in the specified direction only. If the master axis is moving opposite to the direction specified, the slave axis cam is stopped.<br>0: Positive direction<br>2: Negative direction<br>4: No direction specified |
| CamTransition (Reserved) | Cam Transition Selection | _eMC_CAM_ TRANSITION | 0: _mcCTNone | 0*1 | (Reserved) |

MC_CamIn

3

Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| BufferMode | Buffer Mode Selection | _eMC_ BUFFER_MODE | 0: _mcAborting<br>1: _mcBuffered | 0*1 | Specify the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered |

*1 The default value for an enumeration variable is actually not the number, but the enumerator.

*2 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

## Output Variables

| Name | Meaning | Data type | Valid range | Unit | Description |
|---|---|---|---|---|---|
| InCam | Cam Motion | BOOL | TRUE or FALSE | --- | TRUE when the cam table start point is executed. |
| InSync | In Sync | BOOL | TRUE or FALSE | --- | TRUE when the cam is in operation. |
| EndOfProfile | End of Cam Cycle | BOOL | TRUE or FALSE | --- | TRUE when the cam table end point is executed. |
| Index | Index | UNIT | Nonnegative value | --- | Contains the cam data index number. |
| Busy | Executing | BOOL | TRUE or FALSE | --- | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | --- | TRUE when the axis is being controlled. |
| Command Aborted | Command Aborted | BOOL | TRUE or FALSE | --- | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | --- | TRUE while there is an error. |
| ErrorID | Error Code | WORD | *2 | --- | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

*1 FALSE while *InCam* (Cam In Motion) is FALSE.

*2 Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| InSync | When the slave axis starts cam operation. | • When *Periodic* is FALSE and *EndOfProfile* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| EndOfProfile | The period where the phase and displacement of the end point of the cam table are output as the command position. | One period after *EndofProfile* changes to TRUE. |
| Busy | When *Execute* changes to TRUE. | • When *Periodic* is FALSE and *EndOfProfile* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| InCam | When the master axis passes the *StartPosition* (Cam Table Start Position). | • When *Periodic* is FALSE and *EndOfProfile* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Active | When the instruction is started. | • When *Periodic* is FALSE and *EndOfProfile* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution.<br>• When the MC_CamOut instruction is executed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

**Note** The update timing of the output variables may change depending on the mode that is set for cam switching. The default timing is given above.

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Master | Master Axis | _sAXIS_REF | --- | Specify the master axis.*1 |
| Slave | Slave Axis | _sAXIS_REF | --- | Specify the slave axis.*1 |
| CamTable | Cam Table | ARRAY[0..N]OF _sMC_CAM_ REF | --- | Specify the cam data structure _sMC_CAM_REF array variable as the cam table.*2 |

*1 Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

*2 *N* in the array variable is set automatically by the Sysmac Studio. Specify a cam data variable that was created on Cam Editor of the Sysmac Studio.

### Precautions for Correct Use

If you specify the same axis for the master axis and slave axis, a Master and Slave Defined as Same Axis minor fault (error code 5436 hex) will occur.

## Function

- The MC_CamIn instruction executes a cam motion that synchronizes the master axis phase and slave axis displacement according to a cam table.
- You must create the cam table specified for this instruction by using the Cam Editor and download it to the Controller in advance.
- This instruction is executed when *Execute* changes to TRUE.

### Precautions for Correct Use

For the cam table, you must use a cam data variable that was created on the Cam Editor of the Sysmac Studio.

### Additional Information

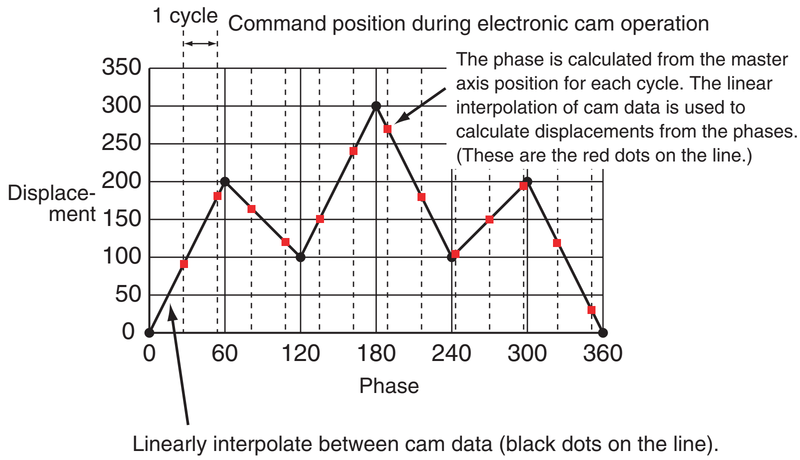
Use the Synchronize Menu of the Sysmac Studio to download the project.

For details on cam tables, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

Specify the phases and displacements in the cam table as relative quantities from a start point of 0.0. The command positions for the master and slave axes are linear interpolations between two cam data where the slave axis displacement corresponding to the master axis phase is calculated.
If there are only a few cam data, the intervals between phases are large and cam operation will not be very precise. If there are many cam data, the intervals between phases are small and cam operation will be very precise.

Cam Table

| | Phase | Displacement | |
|---|---|---|---|
| Start point | 0 | 0 | Index = 0 |
| Cam data | 60 | 200 | 1 |
| | 120 | 100 | 2 |
| | 180 | 300 | 3 |
| | 240 | 100 | 4 |
| | 300 | 200 | 5 |
| End point | 360 | 0 | 6 |




### Precautions for Correct Use

- When executed, this instruction checks if the phases are in ascending order. If they are not in ascending order, an error occurs.
  When you change cam data, execute the MC_SetCamTableProperty (Set Cam Table Properties) instruction to make sure that the phases are in ascending order.
  Make sure that the phases will be in ascending order before you change the phases during a cam motion. The cam motion may stop if the phases are not in ascending order.
- Cam data variables are global variables. You can therefore access or change the values of cam data variables from more than one task. If you change the values of cam data variables from more than one task, program the changes so that there is no competition in writing the value from more than one task.
- If you use exclusive control of global variables between tasks for a cam data variable, do not use the cam data variable for motion control instructions while exclusive control is in effect for the cam data variable. An Incorrect Cam Table Specification error (error code: 5439 hex) will occur.

**Precautions for Safe Use**

Do not execute the MC_SetPosition instruction on the *Master* (Master Axis) if you use this instruction.
If the MC_SetPosition instruction is executed for the *Master* (Master Axis), the *Slave* (Slave Axis) may follow the master axis quickly.
If you want to use the MC_SetPosition instruction for the *Master* (Master Axis), disable the relationship between the *Master* (Master Axis) and slave axes before executing the instruction.
Refer to *Precautions for Master and Auxiliary Axes in Synchronized Control* on page 1-7 for precautions on the master axis.

## Instruction Details

This section describes the instruction in detail.

### ● Instruction Execution Condition

You can execute this instruction while the master axis is stopped, during position control, velocity control, or synchronized control. For details on the slave axis, refer to *Re-execution of Motion Instructions* on page 3-173 and *Multi-execution of Motion Instructions* on page 3-174.

### ● Software Limits

If the slave axis exceeds the software limit during cam operation, an error occurs.

### ● Cam Data Variables

A cam data variable is declared as an array of cam data structures. The type declaration for the cam data structure is shown below.

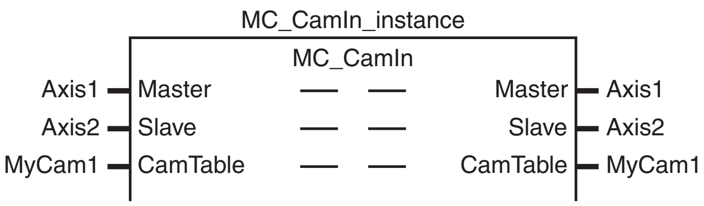
```
TYPE
        (*Cam data structure*)
        _sMC_CAM_REF :
        STRUCT
                Phase       : REAL;     (*Phase*)
                Distance    : REAL;     (*Displacement*)
        END_STRUCT;
END_TYPE
```

Create the cam data variables on the Sysmac Studio.
You can specify a name for the cam table name (i.e., the name of the cam data variable).

For example, if you make a cam table called MyCam1 with 1,000 points use the following variable declaration is automatically made by the Sysmac Studio.

```
VAR
        (*Cam table*)
        MyCam1    :    ARRAY [0..999] OF _sMC_CAM_REF;
END_VAR
```

The following notation is used to specify *MyCam1* for this instruction. In this example, the master axis is *Axis1* and the slave axis is *Axis2*.




An error will occur if the specified cam table does not exist in the Controller. You can also specify the same cam table for more than one axis.

The values in cam data variables can be written from the user program or other means. However, any changes to the values are lost when the power supply to the Controller is turned OFF or the cam data variables are downloaded from the Sysmac Studio.
The values that are downloaded from the Sysmac Studio are always used when the power supply to the Controller is turned ON or after the cam data variables are downloaded.
To save any changes, execute the MC_SaveCamTable instruction.
Changes to the cam data variables are retained when the operating mode of the CPU Unit is changed.

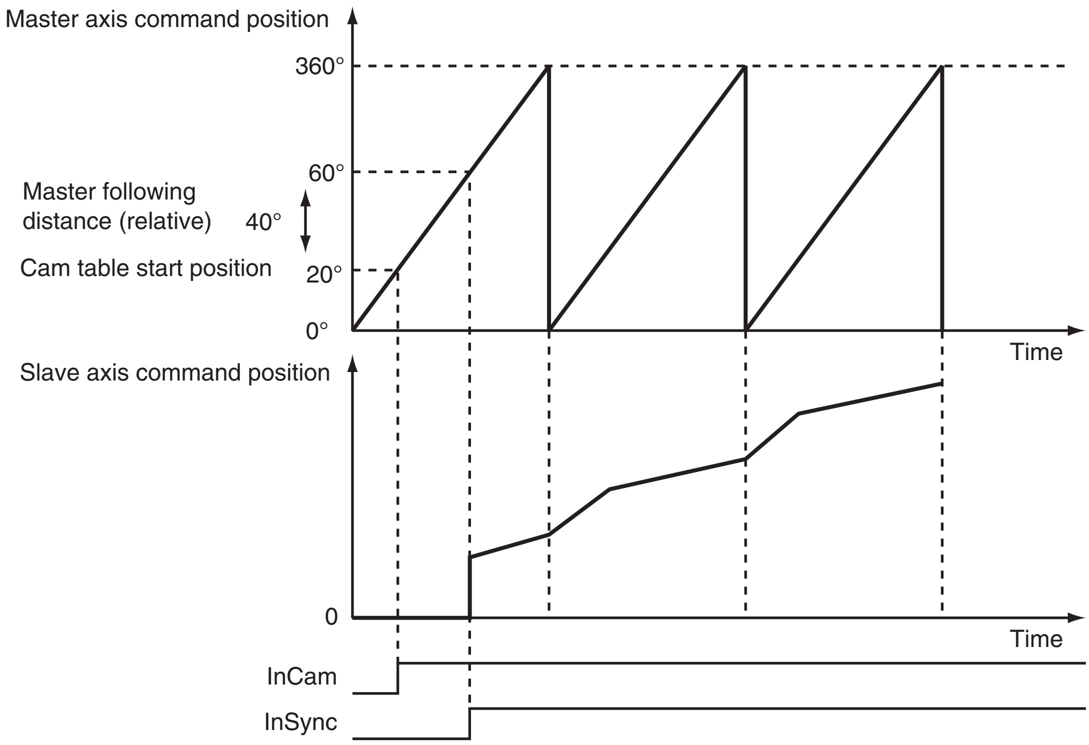
**Additional Information**

- The cam data variables are not published to the network.
  For example, you can monitor the values of *MyCam1[10].Phase* or *MyCam1[10].Distance* from the Sysmac Studio, but you cannot access from another Controller via EtherNet/IP.
- Use the Synchronize Menu of the Sysmac Studio to download the project.

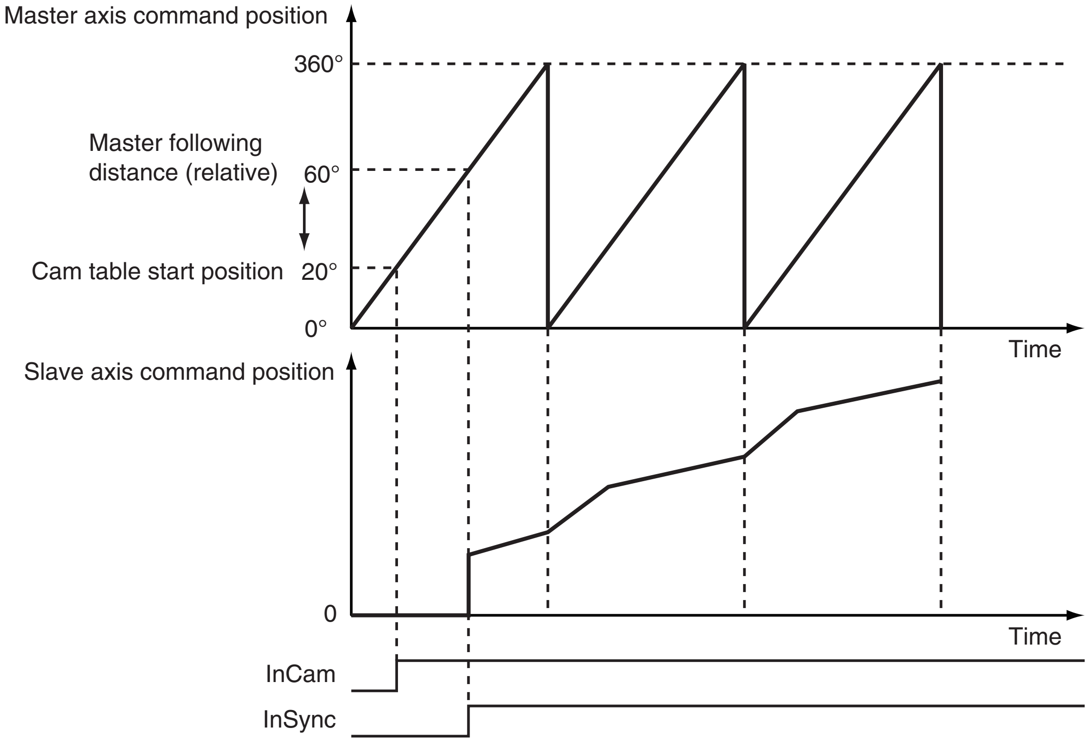
## ● Starting Cam Operation

After the instruction starts, the master axis has to reach the *StartPosition* (Cam Table Start Position). When the master axis moves the *MasterStartDistance* after it reaches the *StartPosition* (Cam Table Start Position), slave axis cam operation starts.
*MasterStartDistance* is specified as an absolute position or as a relative distance from *StartPosition* (Cam Table Start Position).
Set whether to specify using an absolute position or relative position with *StartMode*.

The following chart shows the operation when the cam table start position is 20°, master start distance is 40°, Start Mode is set to relative coordinates, and the Execution Mode is set to Periodic.




To specify the same operation in absolute coordinates, set the cam table start position to 20°, master start distance to 60°, Start Mode to absolute coordinates, and Execution Mode to Periodic.



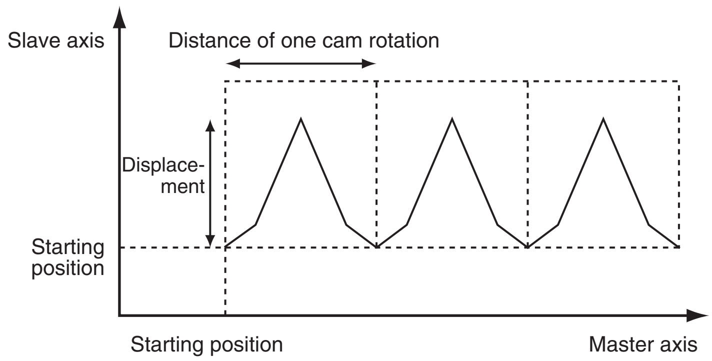

**Precautions for Correct Use**

To perform a cam motion, use the Cam Editor in the Sysmac Studio to create a cam profile and then download the cam profile to the CPU Unit. Use the Synchronize Menu of the Sysmac Studio to download the project.

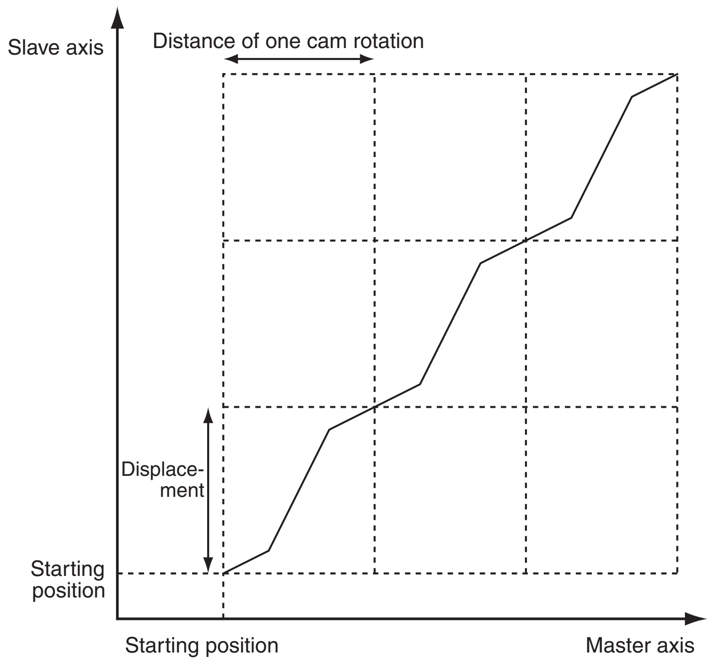
## ● *Periodic* (Periodic Mode)

If you specify TRUE (periodic) for *Periodic*, the cam motion will be repeated from the start to the end point of the cam table.
If you specify FALSE (non-periodic), the cam operation ends when the last point in the cam table is executed.
If the stroke position of the slave axis is the same at the start and end points of the cam table when TRUE (periodic) is set, the cam operates as a reciprocal cam. (Refer to *Reciprocal Cam Operation*.)
If the stroke position of the slave axis differs at the start point and end point, the cam operates as a feeding cam. (Refer to *Feeding Cam Operation*.) In the following chart, the horizontal axis indicates the master axis and the vertical axis indicates the slave axis.

### Reciprocal Cam Operation




### Feeding Cam Operation




## ● *EndOfProfile* (End of Cam Cycle)

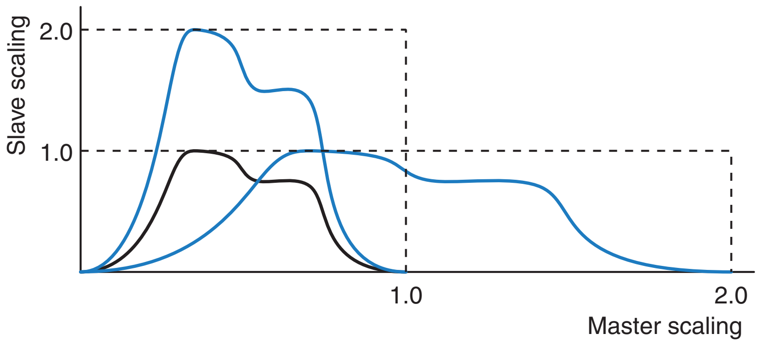
*EndOfProfile* (End of Cam Cycle) is TRUE for one period when the command value of the cam motion for the phase and displacement defined by the end point in the cam table is output. Set the absolute position of the master axis as the *StartPosition* (Cam Table Start Position) and the cam table becomes relative to that position. *EndOfProfile* (End of Cam Cycle) functions as an output indicating the end of the cam table.

## ● Ending Cam Operation

Use the MC_CamOut (End Cam Operation) instruction or MC_Stop instruction to stop cam operation before it is completed.

## ● Scaling Factor

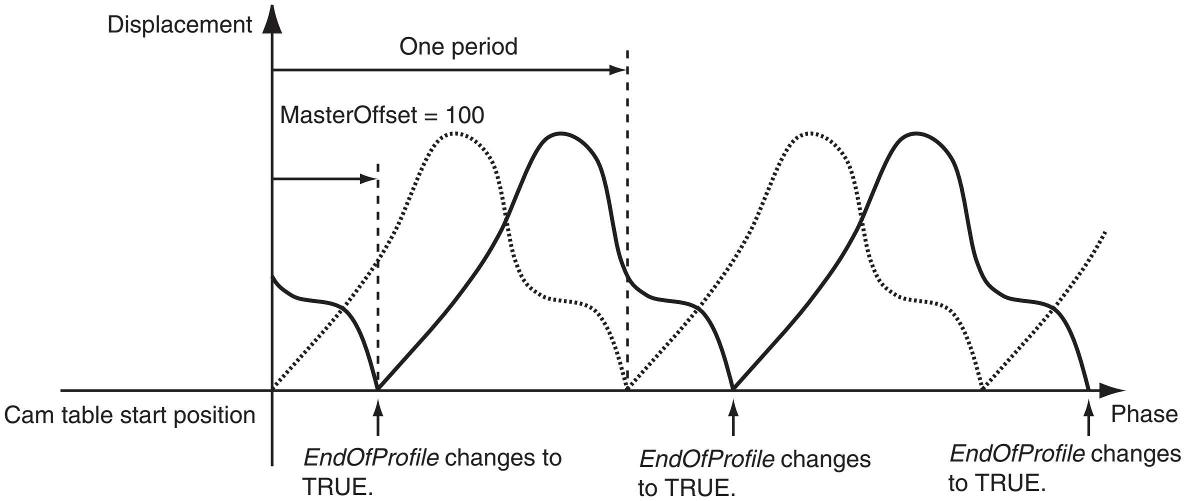
You can specify a scaling factor to scale up or scale down the master axis phase and slave axis displacement of a specified cam table. You can apply separate factors to the master and slave axes.



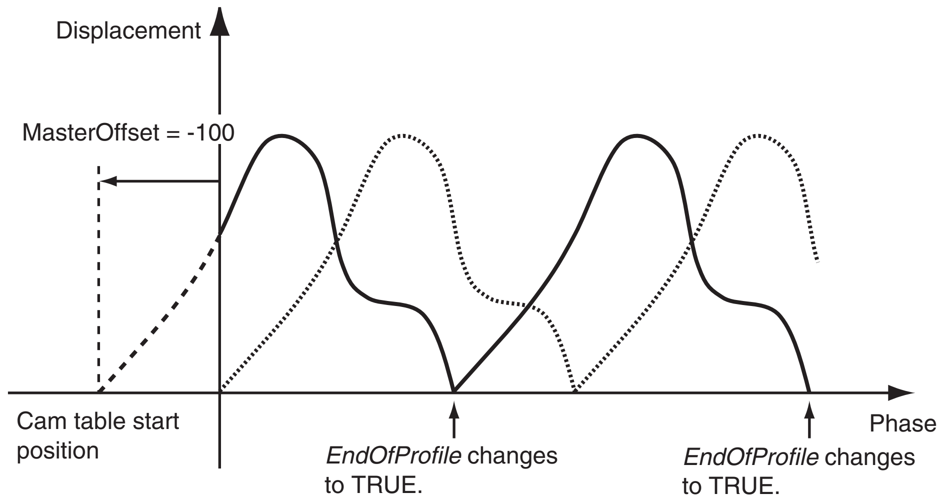

## ● Offset

You can shift the phase and displacement by an offset from the specified cam table.
You can specify separate offsets for the master axis phase and slave axis displacement.

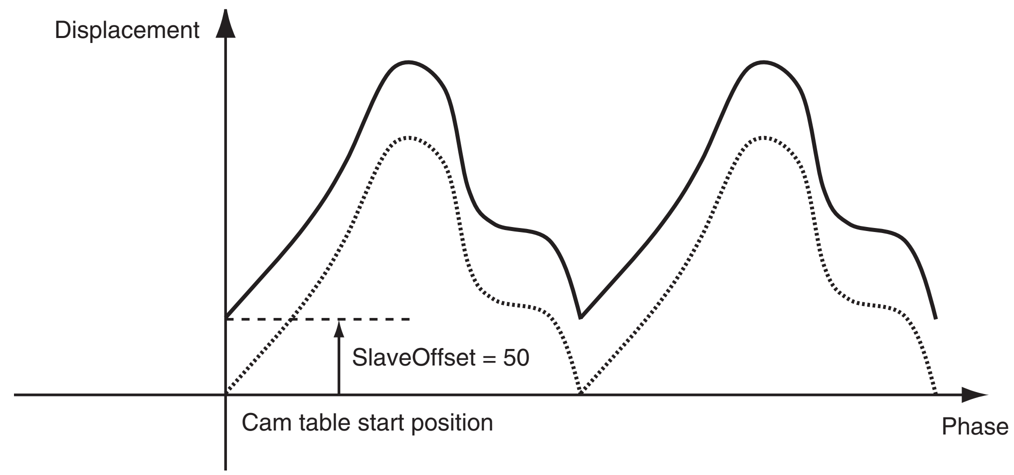
### *MasterOffset* > 0



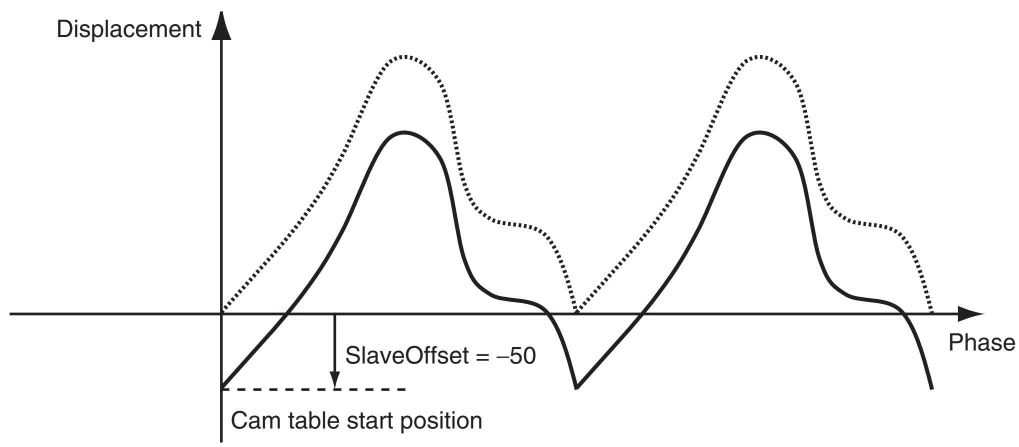
### *MasterOffset* < 0




### *SlaveOffset* > 0




### *SlaveOffset* < 0



## ● *ReferenceType* (Position Type Selection)

Any of the following position types can be selected for the master axis to which the slave axis is synchronized.

- *_mcCommand*: Command position (value calculated in the previous primary period)
  The master axis command position that was calculated in the previous primary period is used for the current period.
  The command value that was calculated for the master axis in the last primary period is used to calculate the command position of the slave axis in the current period.
- *_mcFeedback*: Value obtained in the same primary period
  The actual position of the master axis that was obtained in the same primary period is used.
- *_mcLatestCommand*: Command position (value calculated in the same primary period)
  The command position of the master axis that was calculated in the same primary period is used. This enables the use of information that is more recent than for *_mcCommand*. However, the axis number of the master axis must be set lower than the axis number of the slave axis. If the axis number of the slave axis is lower than the axis number of the master axis, *Error* will change to TRUE. A Master/Slave Axis Numbers Not in Ascending Order error (error code: 5438 hex) will be output to *ErrorID*.

**Additional Information**

The command position that is calculated in the same primary period enables greater precision in synchronization than the command position that was calculated in the previous primary period.
However, the axis number set for the master axis in the system-defined variable for motion control must be lower than the axis number set for the slave axis in the system-defined variable for motion control.

## ● Relationship between Axis Types and Position Types

The relationship between the axis types that you can monitor and position types that is monitored is shown below.

| Axis Type | ReferenceType | |
|---|---|---|
| | _mcCommand or _mcLatestCommand | _mcFeedback |
| Servo axis | OK | OK |
| Encoder axis | No* | OK |
| Virtual servo axis | OK | OK |
| Virtual encoder axis | No* | OK |

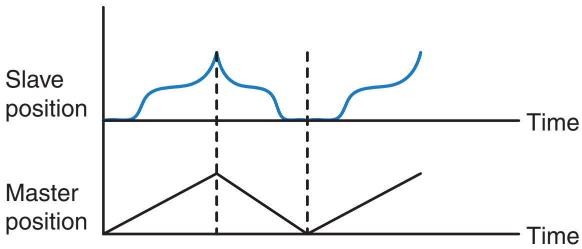
* A Position Type Selection Out of Range error (error code: 5430 hex) occurs when the instruction is executed.

## ● *Direction*

You can start cam operation for the slave axis only if the travel direction of the master axis matches the setting in *Direction*. *Direction* is valid only while *InSync* is TRUE.

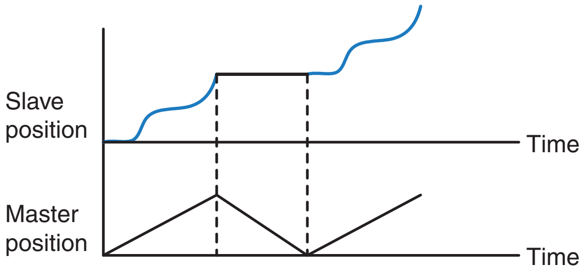
### *_mcNoDirection* (No Direction Specified)

Cam operation starts regardless of whether the master axis is traveling in the positive or negative direction.



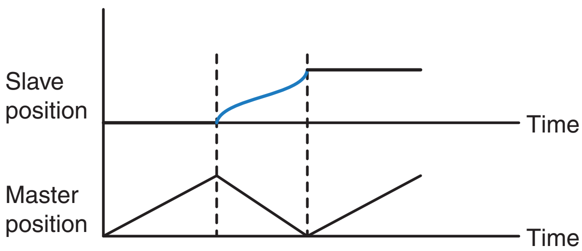

### _mcPositiveDirection

Cam operation starts when the master axis is moving in the positive direction.




### _mcNegativeDirection

Cam operation starts when the master axis is moving in the negative direction.




## ● Index

Of the two cam data used to find the command positions of the master and slave axes, the one with the smaller cam data index number is output to the Index output variable. Use this value for fine-tuning the cam data with the Cam Editor or with the user program.

## ● In-position Check

An in-position check is not performed for this instruction.

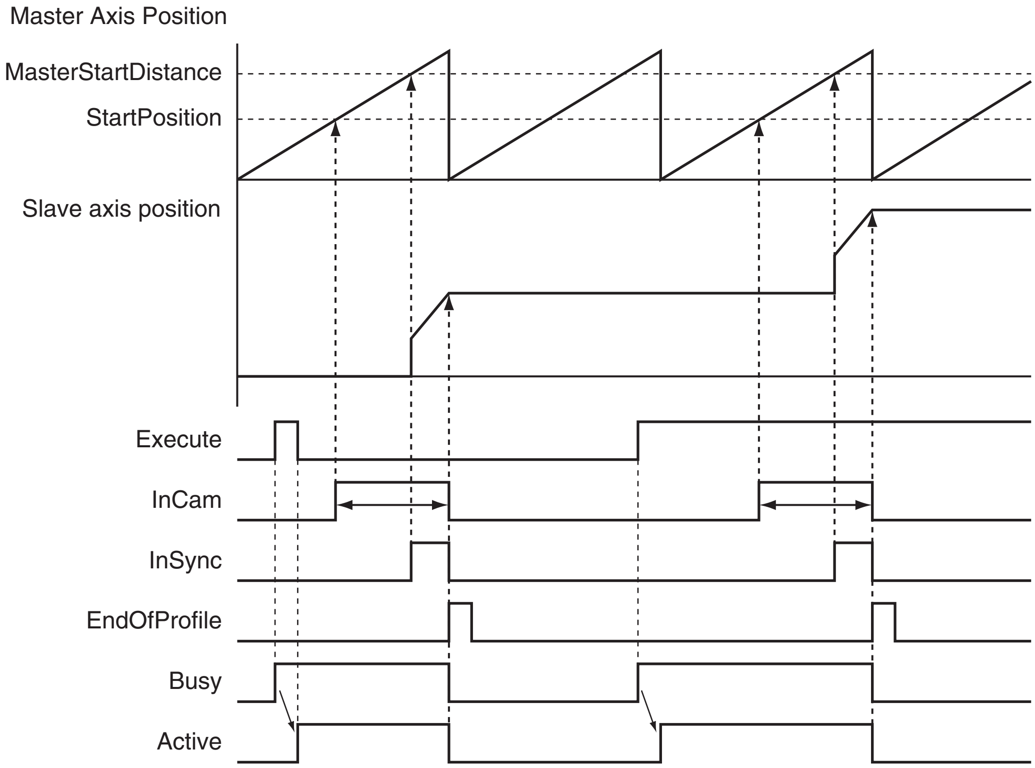
## ● Override Factors

You cannot set override factors with the MC_SetOverride (Set Override Factors) instruction for this instruction.

## Timing Charts

### ● Non-periodic Operation

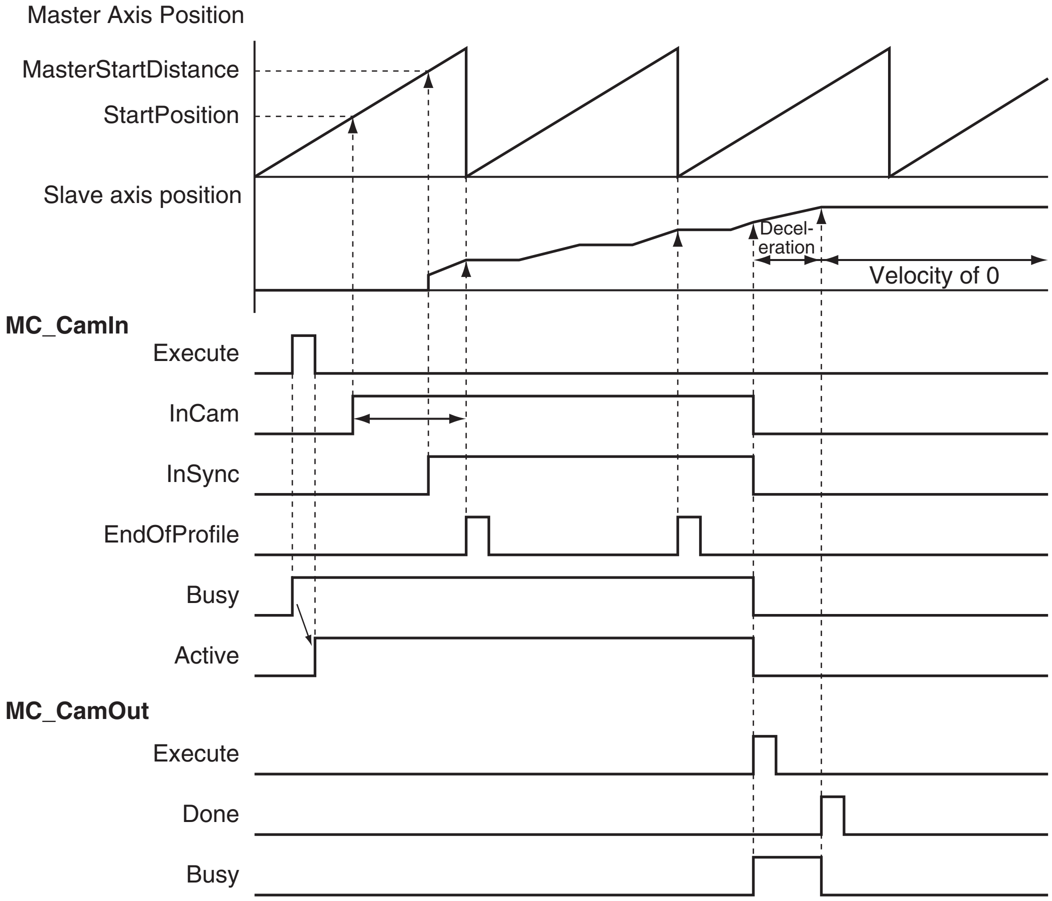
The following timing chart shows the operation when *Periodic* (Periodic Mode) is FALSE (non-periodic) for the MC_CamIn (Start Cam Operation) instruction.



## ● Periodic Operation

The cam motion is repeatedly executed.
The slave axis decelerates to 0 when the cam operation is ended with the MC_CamOut (End Cam Operation) instruction. The operation of the master axis is not affected.

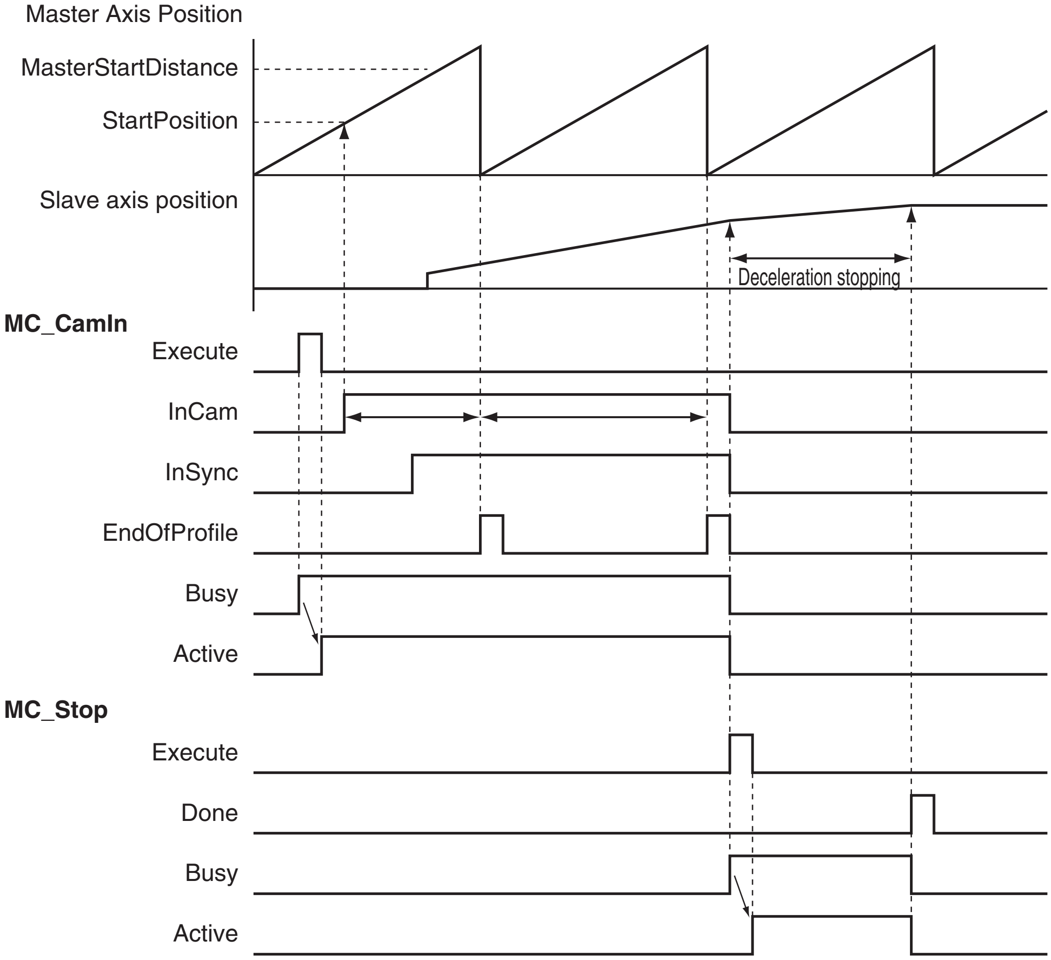
The following timing chart shows the operation when *Periodic* (Periodic Mode) is TRUE (periodic) for the MC_CamIn (Start Cam Operation) instruction and then the MC_CamOut (End Cam Operation) instruction is executed.



## MC_Stop Instruction

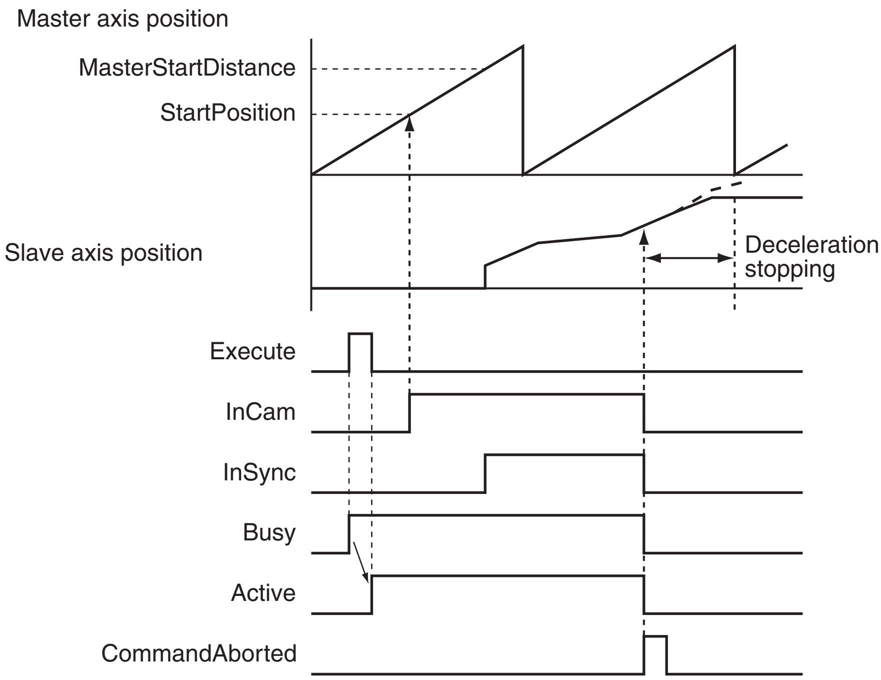
If the MC_Stop instruction is executed for the master axis during cam operation, the sync between the master axis and slave axis is maintained. If the MC_Stop instruction is executed for the slave axis during cam operation, the sync between the master axis and slave axis ends.

The following timing chart displays the operation when *Periodic* (Periodic Mode) is TRUE (periodic) for the MC_CamIn (Start Cam Operation) instruction and then the MC_Stop instruction is executed for the slave axis.



## Aborting the Instruction

If an axis error occurs for the slave axis during execution of this instruction or if the MC_Stop instruction is executed for the slave axis, *CommandAborted* changes to TRUE and the slave axis decelerates to a stop at the maximum deceleration rate. Refer to *A-1 Error Codes* for information on isolating the causes of axis errors. If an axis error occurs on the master axis, cam operation will continue.




## Re-execution of Motion Instructions

You can re-execute this instruction to change the cam table during operation. When re-executing the instruction, *InCam* and *InSync* retain the status that they had prior to the re-execution. If the instruction is re-executed when InSync is TRUE, the cam operation starts from the current phase.

### Precautions for Correct Use

If a cam table is switched by re-executing the instruction during a cam motion, the velocity or acceleration of the slave axis may change rapidly after re-execution. Be careful when re-executing the instruction because the mechanical composition may be affected.

For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

- **Re-execution Procedure**

For the procedure to re-execute this instruction, refer to *Sample Programming 1* on page 3-175 and *Sample Programming 2* on page 3-186.

## Multi-execution of Motion Instructions

For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

- **Execution of Other Instructions during Instruction Execution**

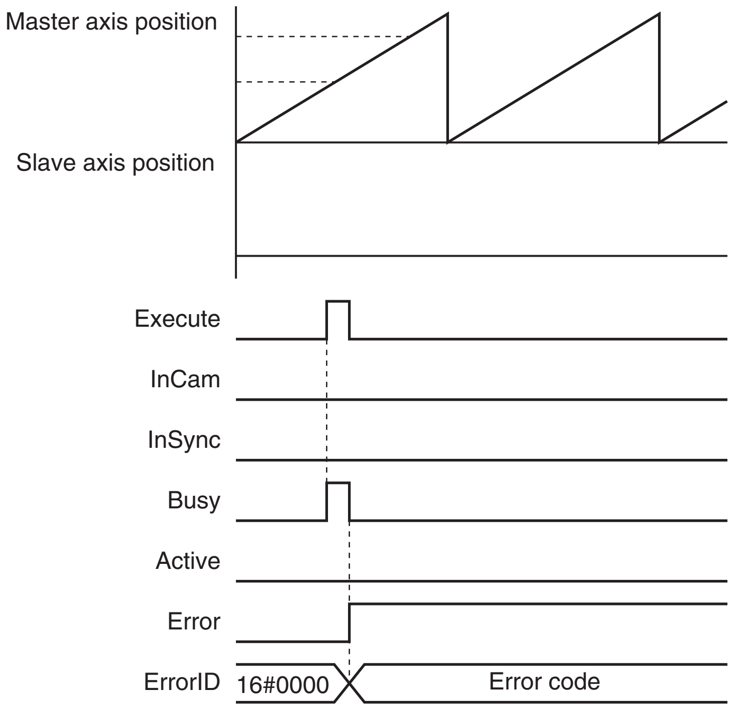
To use multi-execution of motion instruction for this instruction, specify the slave axis. When multi-execution of another instruction is performed while this instruction is in execution, the following limits apply depending on the Buffer Mode.

- When another instruction is executed by using multi-execution with *BufferMode* set to *Aborting*, the cam motion is aborted and the next operation is started.
- When another instruction is executed with BufferMode set to *Buffered*, the command position of the next operation is output when *EndOfProfile* (End of Cam Cycle) changes to TRUE.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

- **Timing Chart When Error Occurs**




- **Error Codes**

Refer to *A-1 Error Codes* for instruction errors.

## Sample Programming 1

This sample programming shows cam operation when periodic operation is specified. In this example, the master axis is *axis 1* and the slave axis is *axis 2*.

**Additional Information**

You can specify only the initial values for input variables that are reserved. In this sample, variables with initial values are defined for the instructions, but you do not need to assign variables and parameters when you program them.

### Parameter Settings

The minimum settings required for this sample programming are given below.

#### ● Setting Axis Parameters

##### Axis Types

| Axis | Axis Type |
|---|---|
| Axis 1 | Servo axis (master axis) |
| Axis 2 | Servo axis (slave axis) |

##### Count Modes

| Axis | Count Mode |
|---|---|
| Axis 1 | Rotary Mode |
| Axis 2 | Linear Mode |

##### Ring Counter

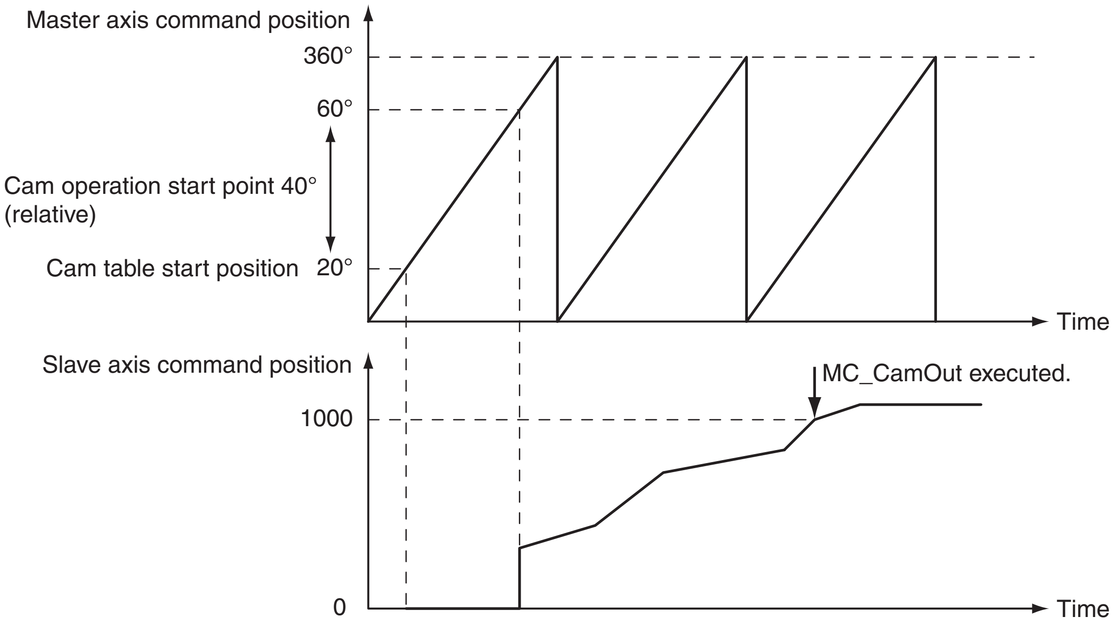
| Axis | Modulo maximum position | Modulo minimum position |
|---|---|---|
| Axis 1 | 360 | 0 |

##### Units of Display

| Axis | Unit of Display |
|---|---|
| Axis 1 | degree |
| Axis 2 | mm |

## Operation Example

### Operation Pattern




**1** Starting Cam Operation

If the cam table start point position (zero phase position) is 20°, the slave axis starts operation when the master axis reaches a position where the relative angle from that point is 40°. Cam operation operates in a periodic motion. If *Periodic* is TRUE, periodic operation is performed.

**2** Ending Cam Operation

When the actual position of the slave axis *MC_Axis001.Act.Pos* exceeds 1000.0, cam operation is ended and the slave axis is stopped at deceleration rate *DecRate2*.

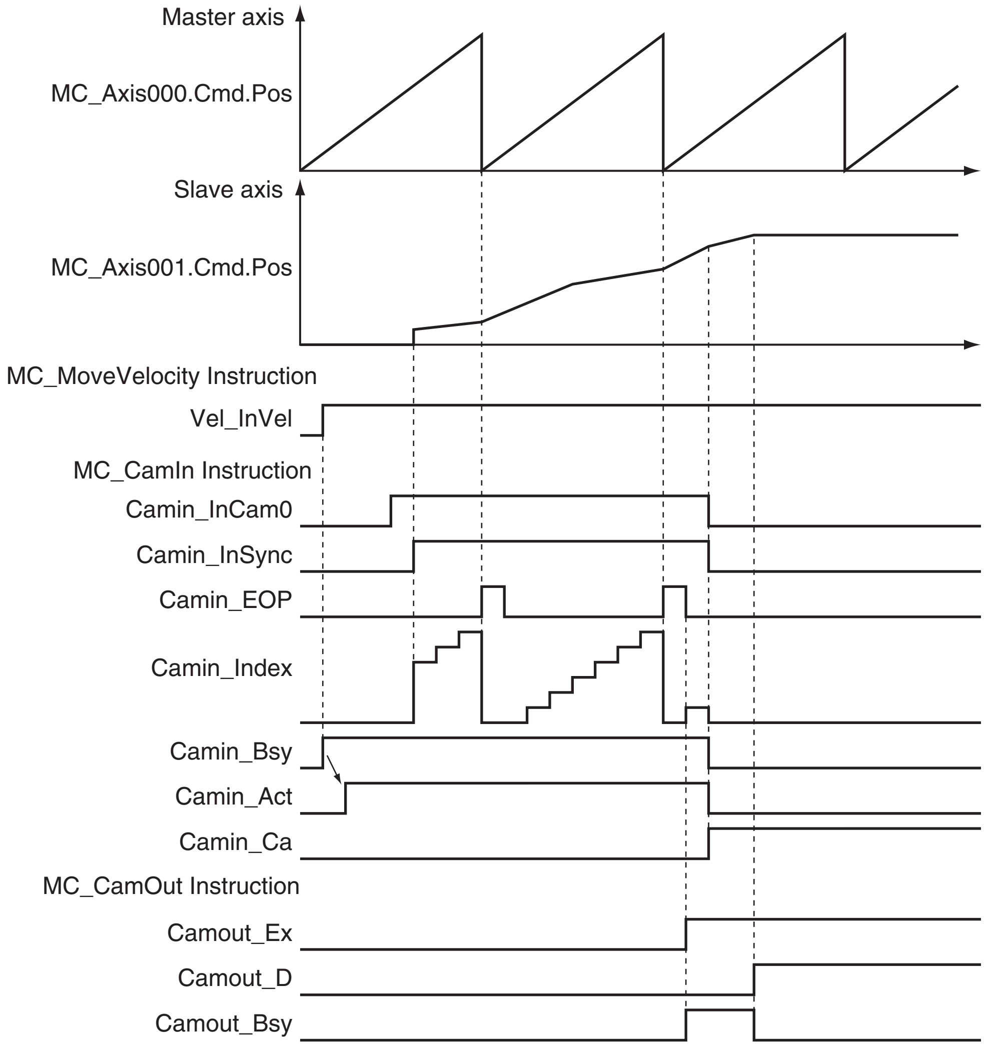
## Ladder Diagram

### Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for the master axis, axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 2. |
| MC_Axis001.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 2. |
| MC_Axis001.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 2. |
| Pwr1_S | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR1 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |

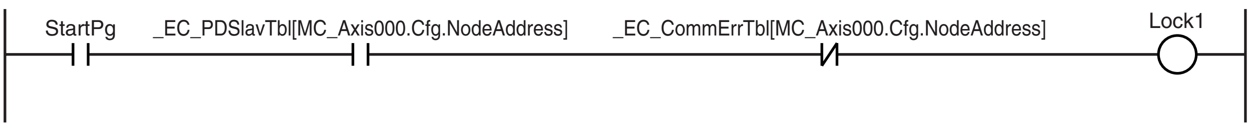
| Name | Data type | Default | Comment |
| --- | --- | --- | --- |
| Pwr2_S | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR2 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| CamProfile0 | ARRAY[0..360] OF _sMC_CAM_REF | --- | This is the cam data variable. The array elements ARRAY[0..N] are set with the Cam Editor. In this sample, 0 to 360 are used, but the number of array elements depends on the settings that you make with the Cam Editor. |
| DecRate2 | LREAL | 10000.0 | This variable sets the deceleration rate for execution of MC_CamOut. |
| Vel_InVel | BOOL | FALSE | This variable is assigned to the *InVelocity* output variable from the VEL instance of the MC_MoveVelocity instruction. It is TRUE when the target velocity is reached. |
| Camin_InCam0 | BOOL | FALSE | This variable is assigned to the *InCam* output variable from the CAMIN instance of the MC_CamIn instruction. It is TRUE during cam operation. |
| Camout_Ex | BOOL | FALSE | The CAMOUT instance of MC_CamOut is executed while this variable is TRUE. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |

## ● Timing Chart

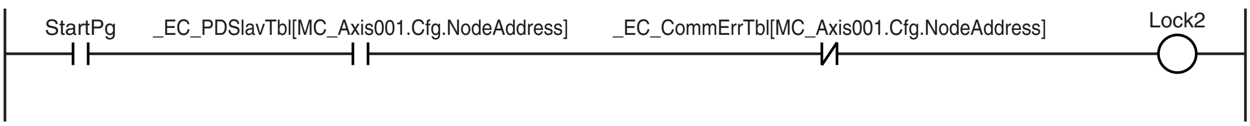



## ● Sample Programming

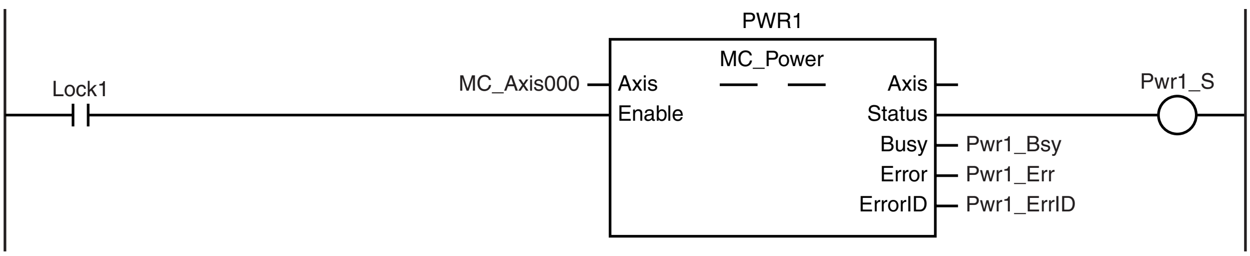
If *StartPg* is TRUE, EtherCAT communications for axis 1 are checked to see if process data communications are normal.



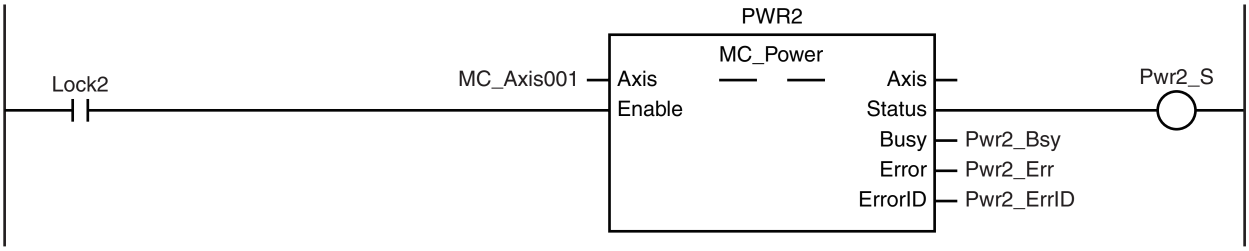

If *StartPg* is TRUE, EtherCAT communications for axis 2 are checked to see if process data communications are normal.



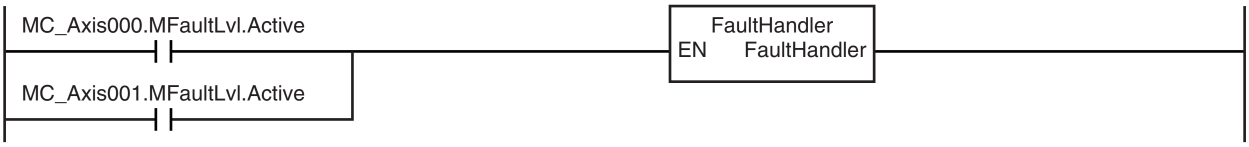

If process data communications are active for axis 1, the Servo is turned ON.



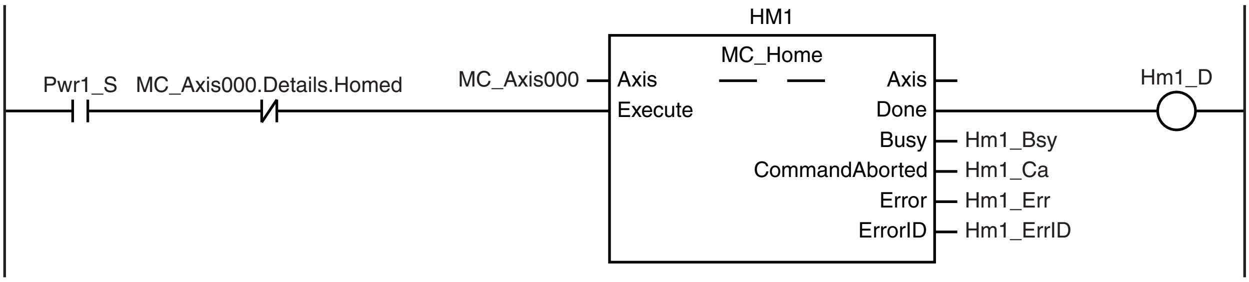
If process data communications are active for axis 2, the Servo is turned ON.



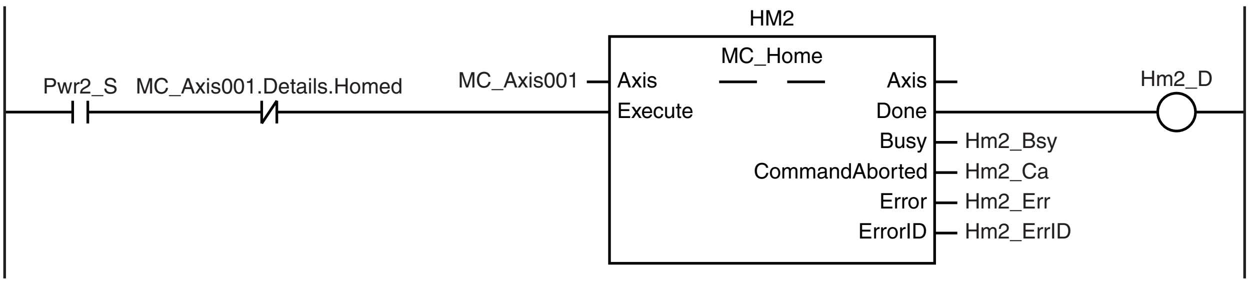

If there is a minor fault level error for a composition axis, the error handler for the device (FaultHandler) is executed. Design FaultHandler separately.



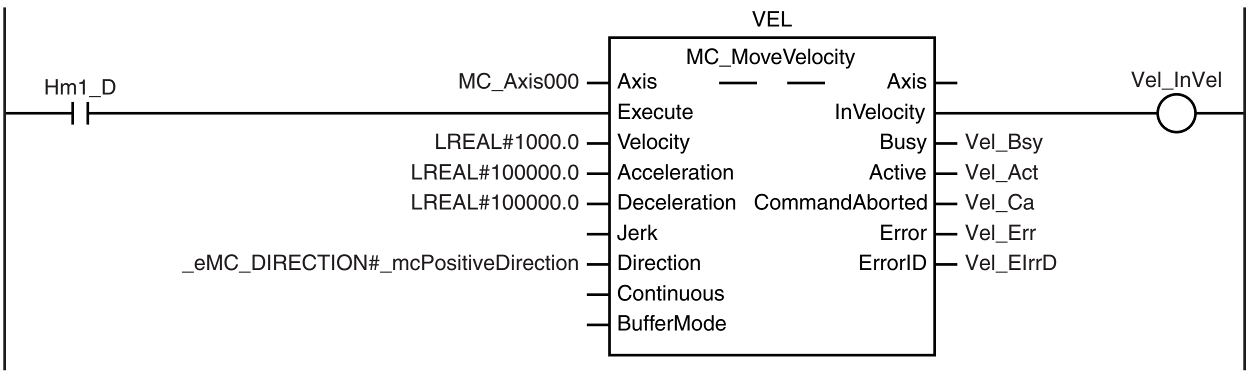

If the Servo is ON for axis 1 and home is not defined, the Home instruction is executed to define home.



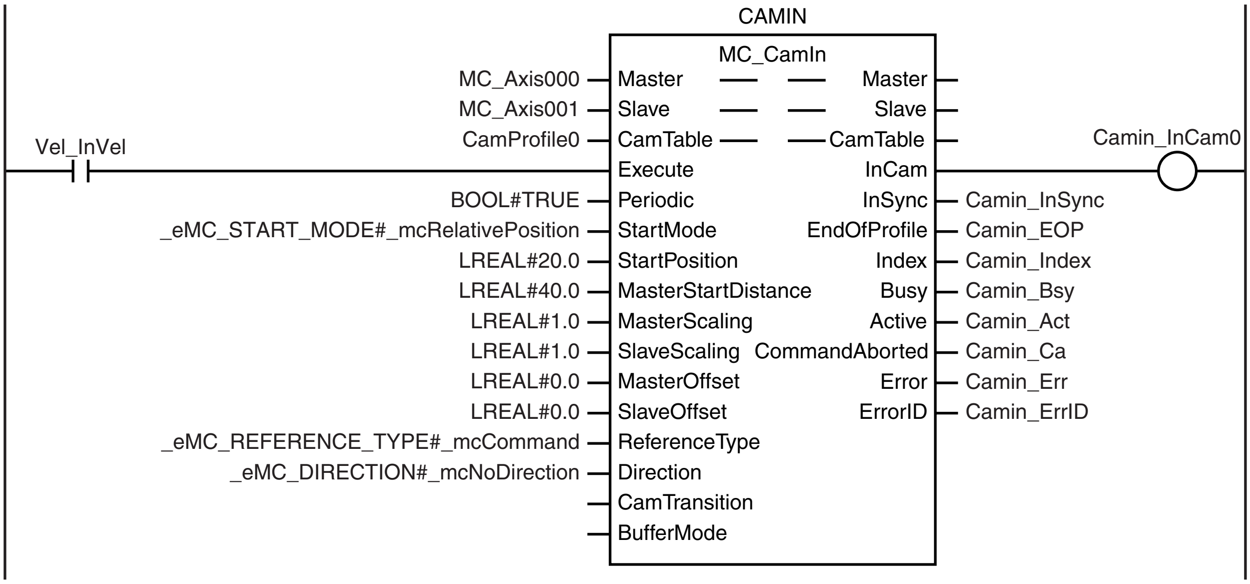

If the Servo is ON for axis 2 and home is not defined, the Home instruction is executed to define home.



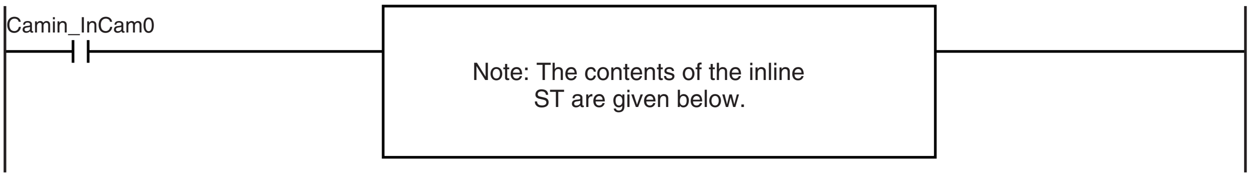

The MC_MoveVelocity (Velocity Control) instruction is executed after homing is completed for axis 1.



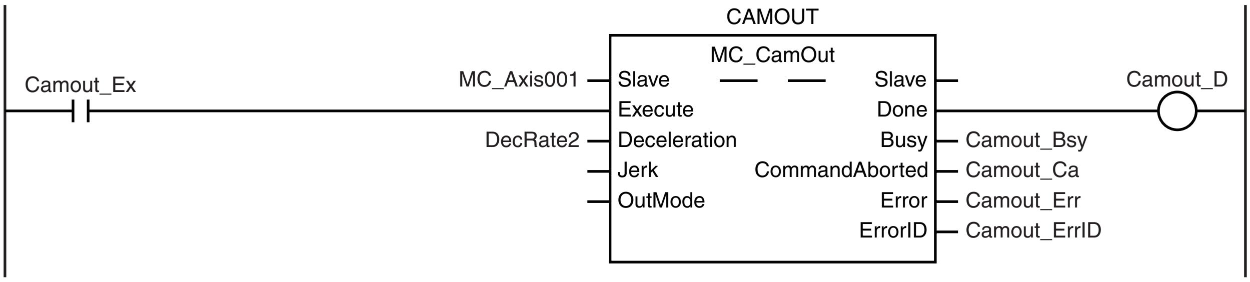
The MC_CamIn (Start Cam Operation) instruction is executed if *Vel_InVel* is TRUE for the MC_MoveVelocity (Velocity Control) instruction.




*CamOut_Ex* changes to TRUE if *Camin_InCam0* is TRUE and *MC_Axis001.Act.Pos* exceeds 1000.




The MC_CamOut (End Cam Operation) instruction is executed when *Camout_Ex* changes to TRUE. The deceleration rate is specified by *DecRate2*.




### Contents of Inline ST

```
IF  MC_Axis001.Act.Pos>LREAL#1000.0 THEN
    Camout_Ex := TRUE;
END_IF;
```

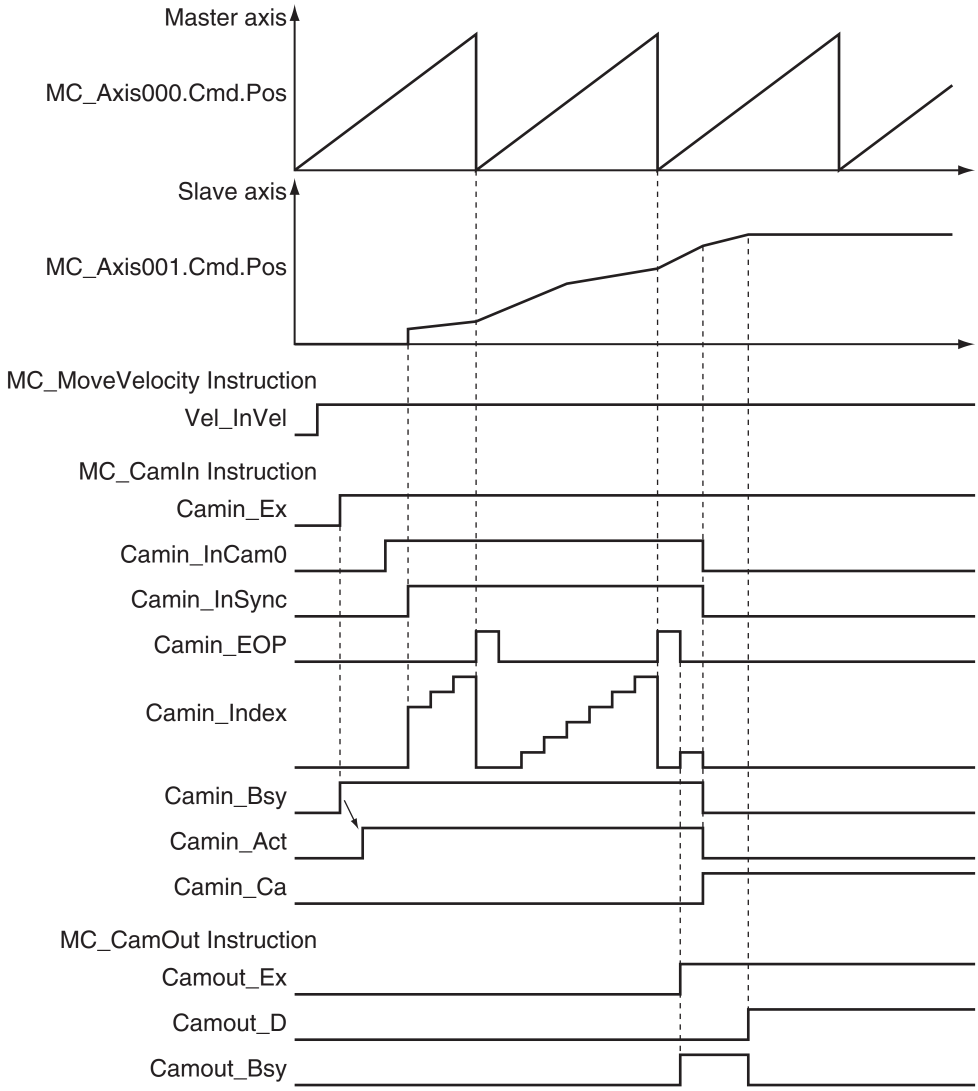
## Structured Text (ST)

### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | -- | Axis Variable for the master axis, axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 2. |
| MC_Axis001.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 2. |

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis001.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 2. |
| Pwr1_S | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR1 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Pwr2_S | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR2 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| CamProfile0 | ARRAY[0..360] OF _sMC_CAM_REF | --- | This is the cam data variable. The array elements ARRAY[0..N] are set with the Cam Editor. In this sample, 0 to 360 are used, but the number of array elements depends on the settings that you make with the Cam Editor. |
| DecRate2 | LREAL | 10000.0 | This variable sets the deceleration rate for execution of MC_CamOut. |
| Vel_InVel | BOOL | FALSE | This variable is assigned to the *InVelocity* output variable from the VEL instance of the MC_MoveVelocity instruction. It is TRUE when the target velocity is reached. |
| Camin_InCam0 | BOOL | FALSE | This variable is assigned to the *InCam* output variable from the CAMIN instance of the MC_CamIn instruction. It is TRUE during cam operation. |
| Camout_Ex | BOOL | FALSE | The CAMOUT instance of MC_CamOut is executed while this variable is TRUE. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| Camin_Ex | BOOL | FALSE | This variable is changed to TRUE when *Vel_InVel* changes to TRUE to change *Execute* for the CAMIN instance of MC_CamIn to TRUE. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

● **Timing Chart**




● **Sample Programming**

```
// Processing when input parameters are not set
IF InitFlag=FALSE THEN

    // MC_CamIn parameters
    Camin_EM        := TRUE;                        // Periodic Mode
    Camin_StMode    := _eMC_START_MODE#_mcRelativePosition;
    Camin_StPos     := LREAL#20.0;                  // Master axis absolute position for start point
    Camin_MStDis    := LREAL#40.0;                  // Master axis position to start cam operation
    Camin_MSc       := LREAL#1.0;                   // Master axis scaling
    Camin_SSc       := LREAL#1.0;                   // Slave axis scaling
    Camin_MO        := LREAL#0.0;                   // Master offset
    Camin_SO        := LREAL#0.0;                   // Slave offset
    Camin_RT        := _eMC_REFERENCE_TYPE#_mcCommand;          // Position type selection
    Camin_Dir       := _eMC_DIRECTION#_mcNoDirection;           // Direction

    // MC_MoveVelocity parameters
    Vel_Vel     := LREAL#1000.0;
    Vel_Acc     := LREAL#100000.0;
    Vel_Dec     := LREAL#100000.0;
    Vel_Dir     := _eMC_DIRECTION#_mcPositiveDirection;
```

```
    // MC_CamOut parameters
    Camout_Dec := DecRate2;

    // Change InitFlag to TRUE after setting the input parameters.
    InitFlag := TRUE;

END_IF;


// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 1 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis000.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
    Pwr1_En:=TRUE;
ELSE
    Pwr1_En:=FALSE;
END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 2 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis001.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis001.Cfg.NodeAddress]=FALSE) THEN
    Pwr2_En:=TRUE;
ELSE
    Pwr2_En:=FALSE;
END_IF;

// If a minor fault level error occurs for axis 1 or axis 2, the error handler for the device is executed.
// Implement FaultHandler separately.
IF (MC_Axis000.MFaultLvl.Active=TRUE) OR (MC_Axis001.MFaultLvl.Active=TRUE) THEN
    FaultHandler();
END_IF;

// If the Servo is ON for axis 1 and home is not defined, the Home instruction is executed for axis 1.
IF (Pwr1_S=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
    Hm1_Ex:=TRUE;
END_IF;

// If the Servo is ON for axis 2 and home is not defined, the Home instruction is executed for axis 2.
IF (Pwr2_S=TRUE) AND (MC_Axis001.Details.Homed=FALSE) THEN
    Hm2_Ex:=TRUE;
END_IF;

// After homing is completed for axis 1, MC_MoveVelocity is executed.
IF  Hm1_D=TRUE THEN
    Vel_Ex := TRUE;
END_IF;

// CamIn is executed when InVel of MC_MoveVelocity is TRUE.
IF  Vel_InVel=TRUE THEN
    Camin_Ex := TRUE;
END_IF;

// CamOut is executed when Camin_InCam0 is TRUE and MC_Axis001.Act.Pos is greater than 1000.
IF (Camin_InCam0=TRUE) AND (MC_Axis001.Act.Pos>LREAL#1000.0) THEN
    Camout_Ex := TRUE;
END_IF;


// MC_Power for axis 1
PWR1(
    Axis        := MC_Axis000,
    Enable      := Pwr1_En,
```

```
    Status      => Pwr1_S,
    Busy        => Pwr1_Bsy,
    Error       => Pwr1_Err,
    ErrorID     => Pwr1_ErrID
);


// MC_Power for axis 2
PWR2(
    Axis        := MC_Axis001,
    Enable      := Pwr2_En,
    Status      => Pwr2_S,
    Busy        => Pwr2_Bsy,
    Error       => Pwr2_Err,
    ErrorID     => Pwr2_ErrID
);


// MC_Home for axis 1
HM1(
    Axis                := MC_Axis000,
    Execute             := Hm1_Ex,
    Done                => Hm1_D,
    Busy                => Hm1_Bsy,
    CommandAborted      => Hm1_Ca,
    Error               => Hm1_Err,
    ErrorID             => Hm1_ErrID
);


// MC_Home for axis 2
HM2(
    Axis                := MC_Axis001,
    Execute             := Hm2_Ex,
    Done                => Hm2_D,
    Busy                => Hm2_Bsy,
    CommandAborted      => Hm2_Ca,
    Error               => Hm2_Err,
    ErrorID             => Hm2_ErrID
);


//MC_MoveVelocity
VEL(
    Axis                := MC_Axis000,
    Execute             := Vel_Ex,
    Velocity            := Vel_Vel,
    Acceleration        := Vel_Acc,
    Deceleration        := Vel_Dec,
    Direction           := Vel_Dir,
    InVelocity          => Vel_InVel,
    Busy                => Vel_Bsy,
    Active              => Vel_Act,
    CommandAborted      => Vel_Ca,
    Error               => Vel_Err,
    ErrorID             => Vel_ErrID
);

//MC_CamIn
CAMIN(
    Master                  := MC_Axis000,
    Slave                   := MC_Axis001,
    CamTable                := CamProfile0,
    Execute                 := Camin_Ex,
    Periodic                := Camin_EM,
```

```
    StartMode             := Camin_StMode,
    StartPosition         := Camin_StPos,
    MasterStartDistance   := Camin_MStDis,
    MasterScaling         := Camin_MSc,
    SlaveScaling          := Camin_SSc,
    MasterOffset          := Camin_MO,
    SlaveOffset           := Camin_SO,
    ReferenceType         := Camin_RT,
    Direction             := Camin_Dir,
    CamTransition         := Camin_CT,
    BufferMode            := Camin_BM,
    InCam                 => Camin_InCam0,
    InSync                => Camin_InSync,
    EndOfProfile          => Camin_EOP,
    Index                 => Camin_Index,
    Busy                  => Camin_Bsy,
    Active                => Camin_Act,
    CommandAborted        => Camin_Ca,
    Error                 => Camin_Err,
    ErrorID               => Camin_ErrID
);
//MC_CamOut
CAMOUT(
    Slave                 := MC_Axis001,
    Execute               := Camout_Ex,
    Deceleration          := Camout_Dec,
    Done                  => Camout_D,
    Busy                  => Camout_Bsy,
    CommandAborted        => Camout_Ca,
    Error                 => Camout_Err,
    ErrorID               => Camout_ErrID
);
```

## Sample Programming 2

This sample programming shows cam operation for a liquid filler.

**Additional Information**

You can specify only the initial values for input variables that are reserved. Parameters are not specified in this sample.

### Parameter Settings

The minimum settings required for this sample programming are given below.

#### ● Setting Axis Parameters

##### Axis Types

| Axis | Axis Type |
|---|---|
| Axis 1 | Servo axis (master axis) |
| Axis 2 | Servo axis (slave axis) |
| Axis 3 | Servo axis (slave axis) |
| Axis 4 | Servo axis (slave axis) |
| Axis 5 | Servo axis (slave axis) |

##### Count Modes

| Axis | Count Mode |
|---|---|
| Axis 1 | Rotary Mode |
| Axis 2 | Rotary Mode |
| Axis 3 | Rotary Mode |
| Axis 4 | Rotary Mode |
| Axis 5 | Rotary Mode |

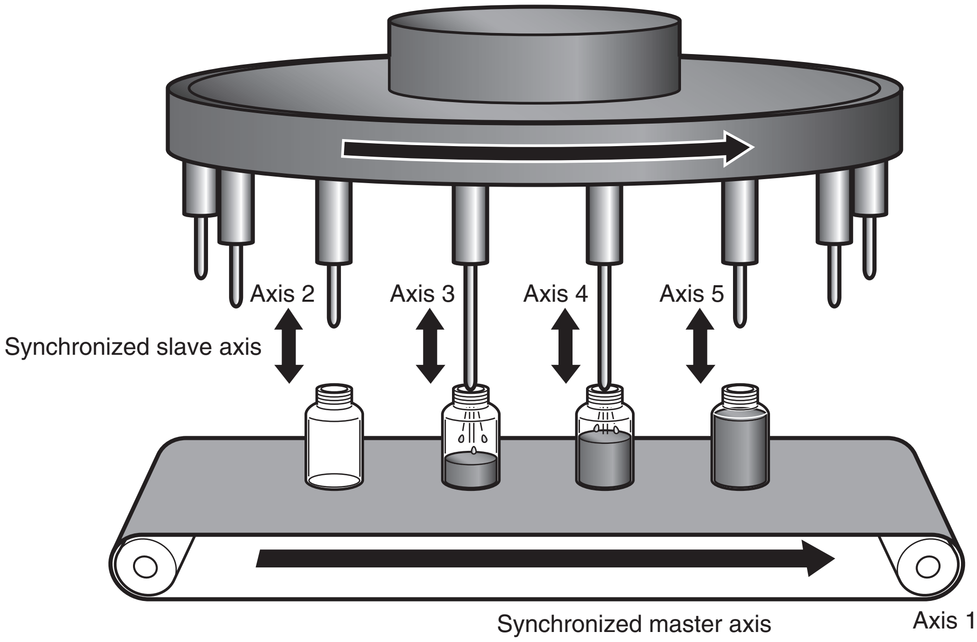
##### Ring Counters

| Axis | Modulo maximum position | Modulo minimum position |
|---|---|---|
| Axis 1 | 360 | 0 |
| Axis 2 | 360 | 0 |
| Axis 3 | 360 | 0 |
| Axis 4 | 360 | 0 |
| Axis 5 | 360 | 0 |

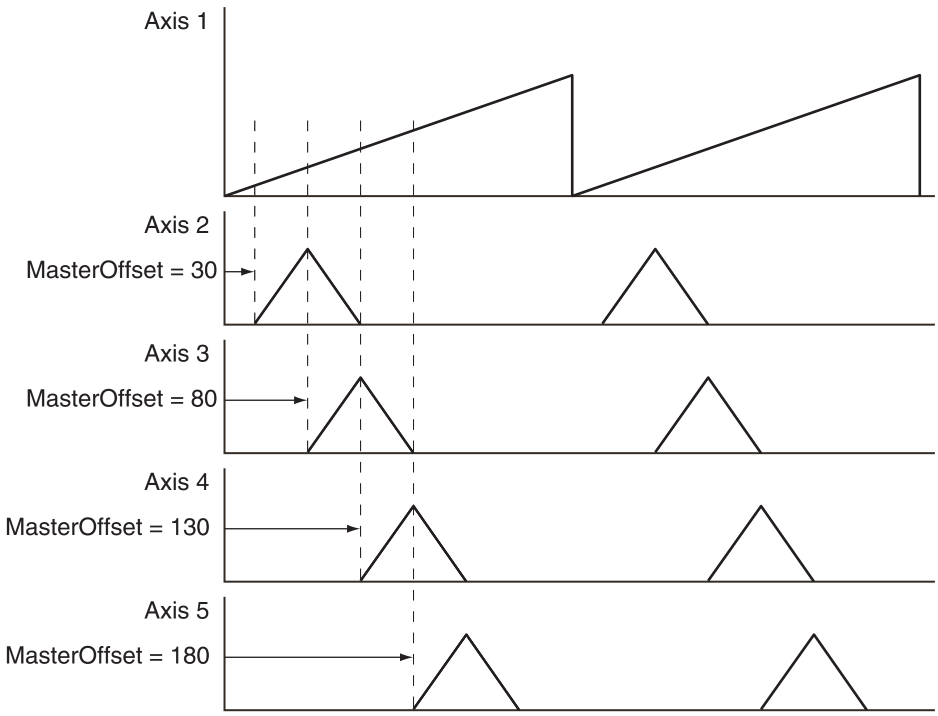
##### Units of Display

| Axis | Unit of Display |
|---|---|
| Axis 1 | degree |
| Axis 2 | degree |
| Axis 3 | degree |
| Axis 4 | degree |
| Axis 5 | degree |

## Operation Example




### ● Operation Pattern




**1** Start Cam Operation

The slave axes, axes 2 to 5, perform cam operation in synchronization with the master axis, axis 1. Each axis shifts its phase by 50° and starts cam operation.

**2** Periodic Operation

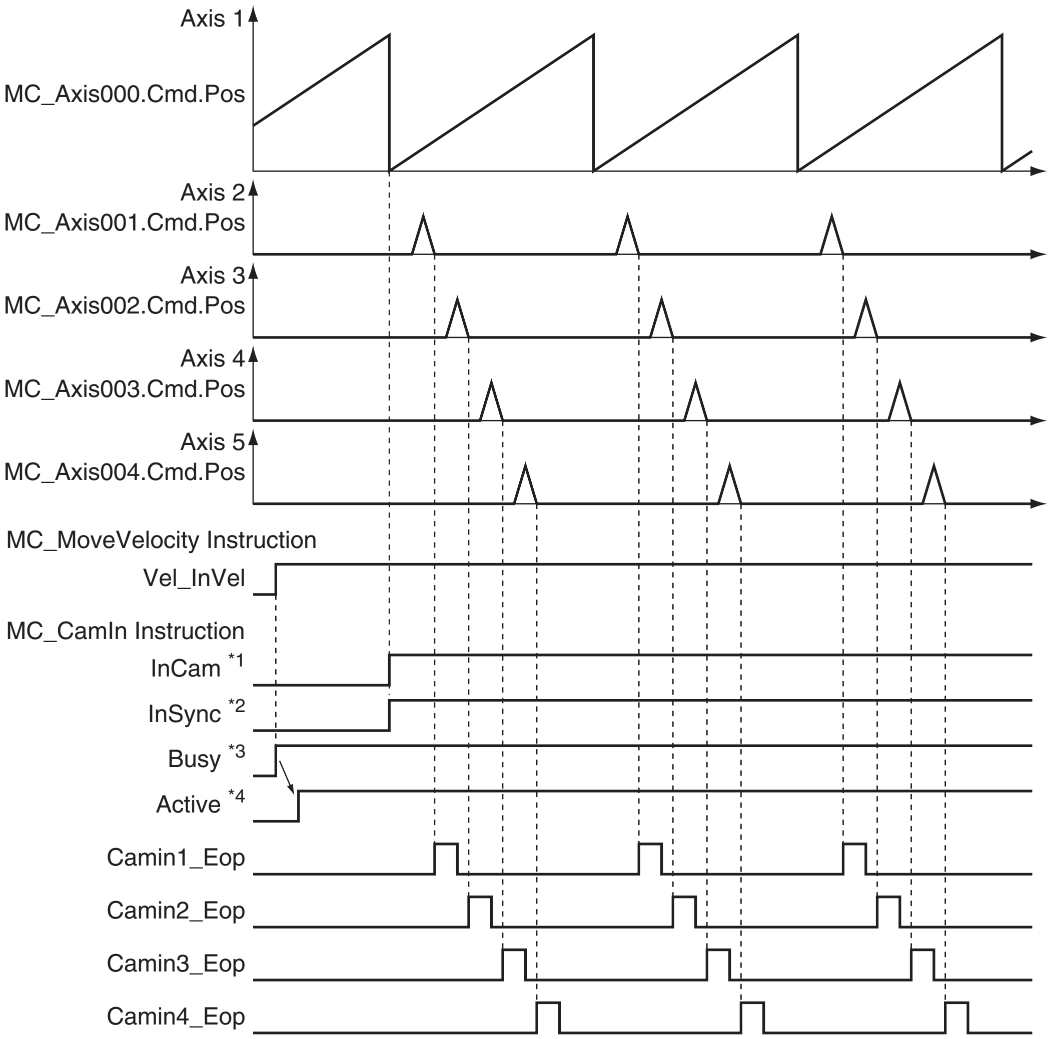
Each axis periodically executes the specified cam operation.

## Ladder Diagram

### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for the master axis, axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 2. |
| MC_Axis002 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 3. |
| MC_Axis003 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 4. |
| MC_Axis004 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 5. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| Vel_InVel | BOOL | FALSE | This variable is assigned to the *InVelocity* output variable from the VEL instance of the MC_MoveVelocity instruction. It is TRUE when the target velocity is reached. |
| CamProfile0 | ARRAY[0..360] OF _sMC_CAM_REF | --- | This is the cam data variable. This variable is assigned to the *CamTable* input variables from the CAMIN1 to CAMIN4 instances of the MC_CamIn instruction. The array elements ARRAY[0..N] are set with the Cam Editor. |

### ● Timing Chart



*1 The timing is the same for the following: *Camin1_InCam*, *Camin2_InCam*, *Camin3_InCam*, and *Camin4_InCam*.

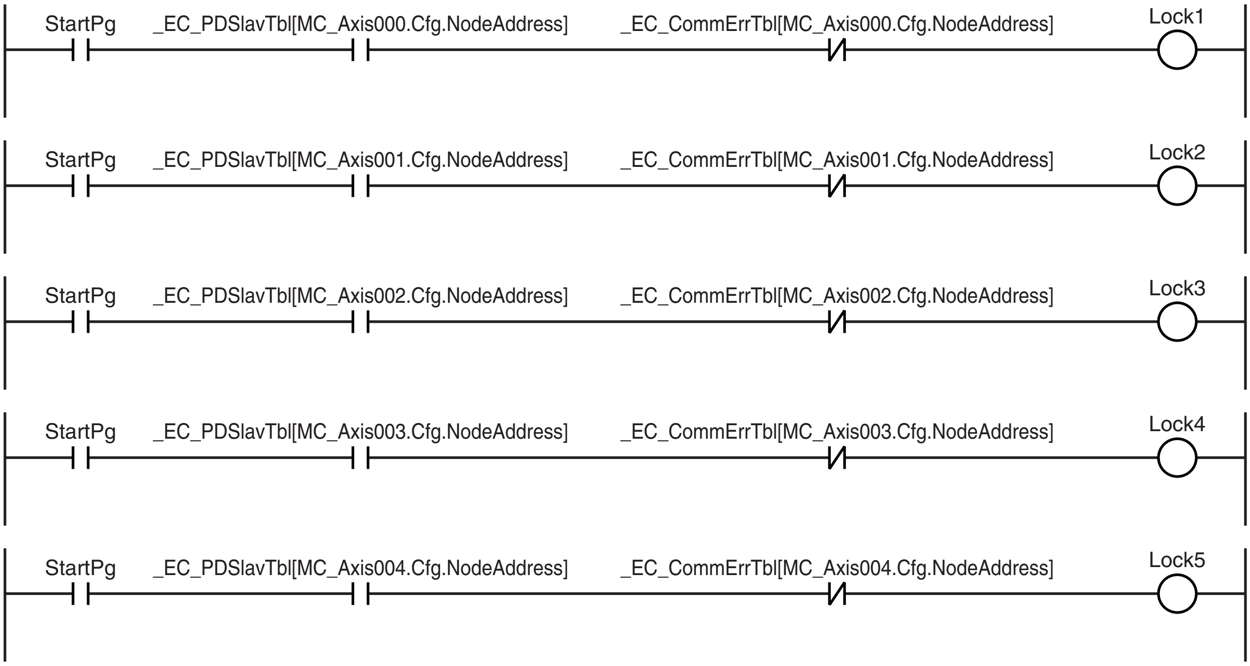
*2 The timing is the same for the following: *Camin1_InSync*, *Camin2_InSync*, *Camin3_InSync*, and *Camin4_InSync*. In this sample, *MasterStartDistance* is 0, so *InSync* changes to TRUE from when the master axis is 0°.

*3 The timing is the same for the following: *Camin1_Bsy*, *Camin2_Bsy*, *Camin3_Bsy*, and *Camin4_Bsy*.

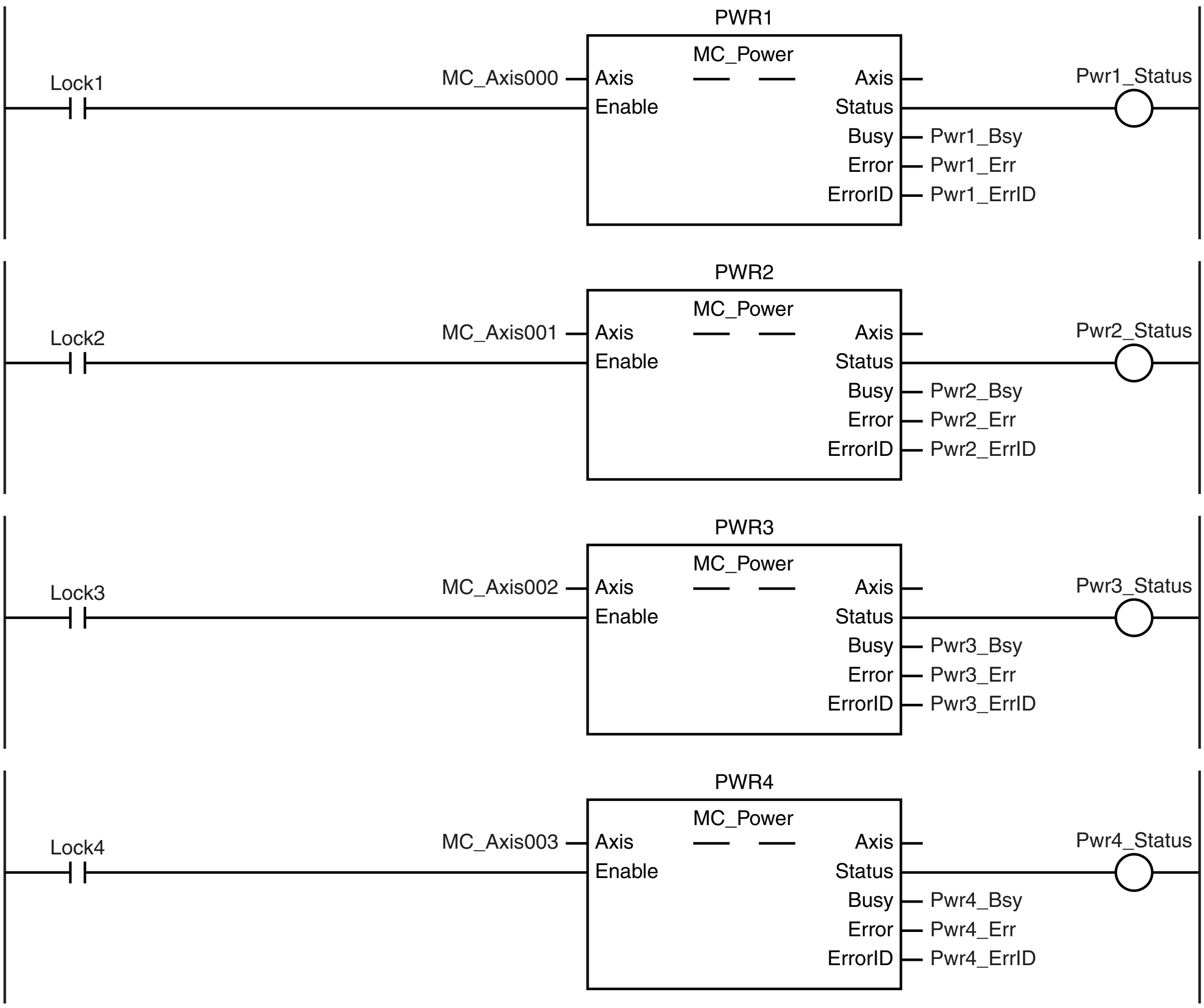
*4 The timing is the same for the following: *Camin1_Act*, *Camin2_Act*, *Camin3_Act*, and *Camin4_Act*.

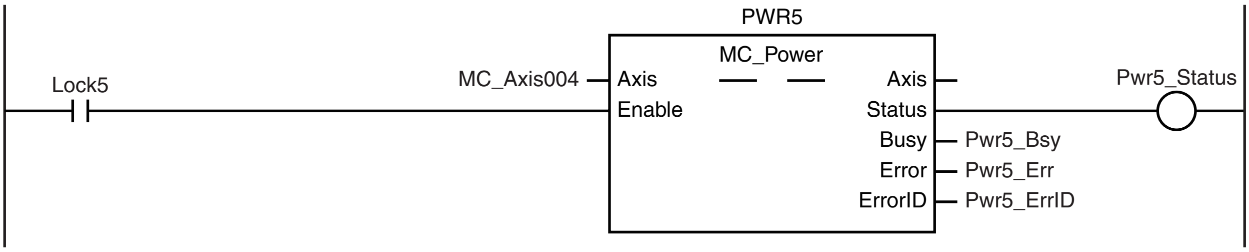
## ● Sample Programming

If *StartPg* is TRUE, EtherCAT communications for the axes are checked to see if process data communications are normal.

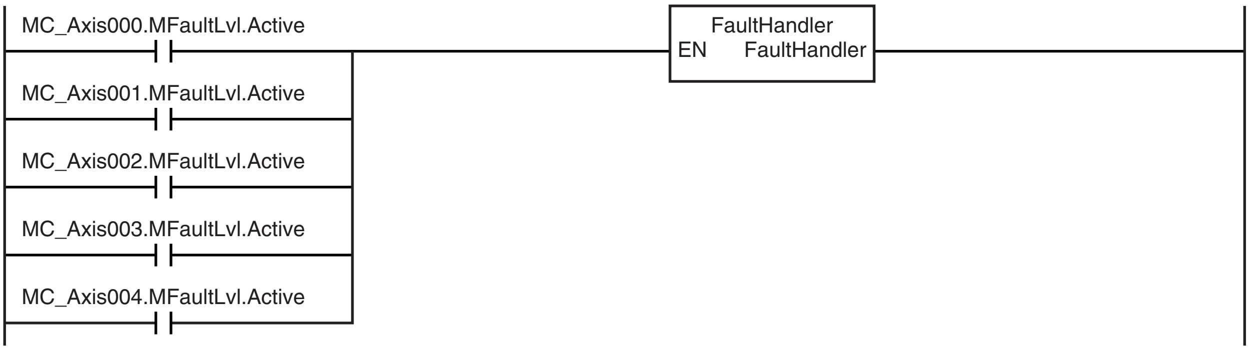



If process data communications are active, the Servo is turned ON for the axes.

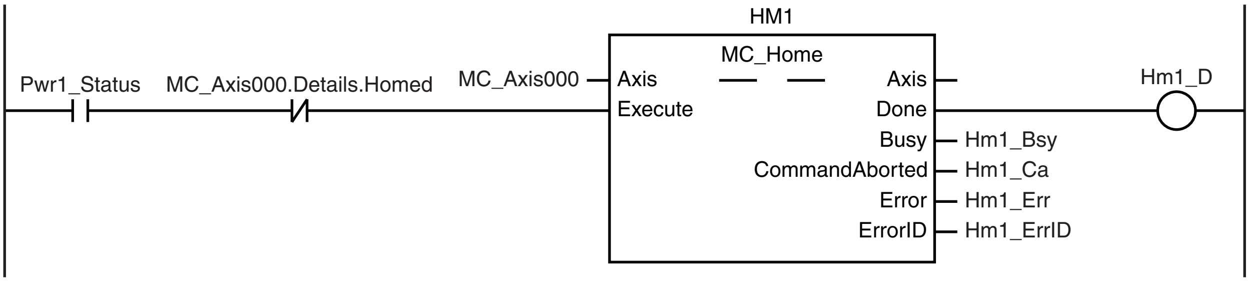


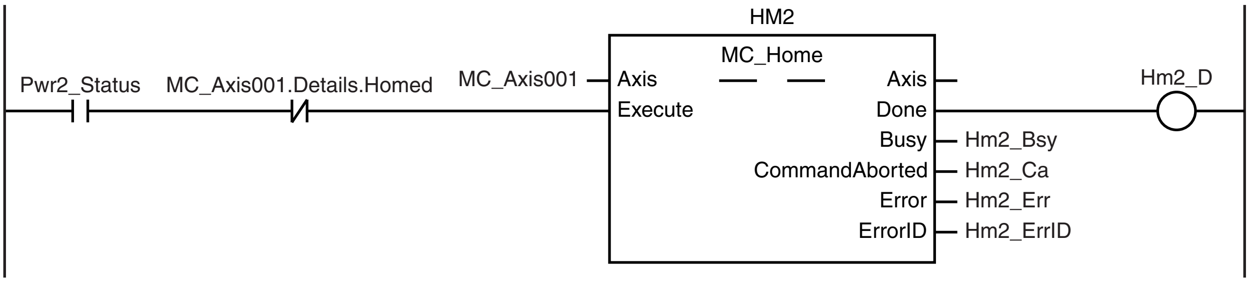


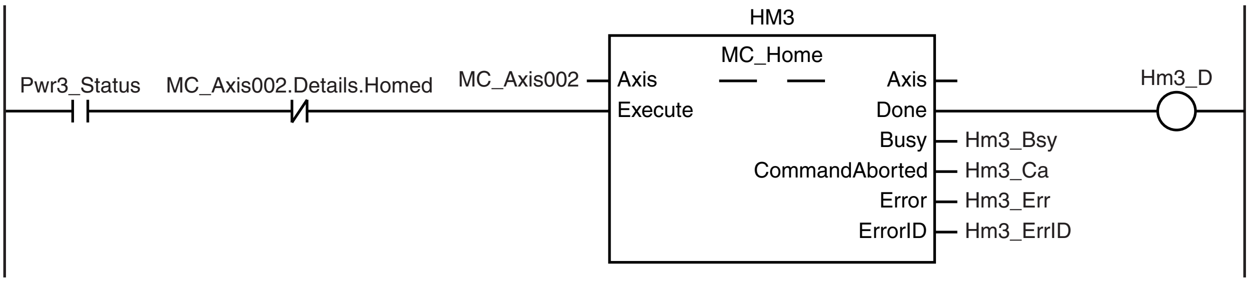

If there is a minor fault level error for a composition axis, the error handler for the device (FaultHandler) is executed. Design FaultHandler separately.

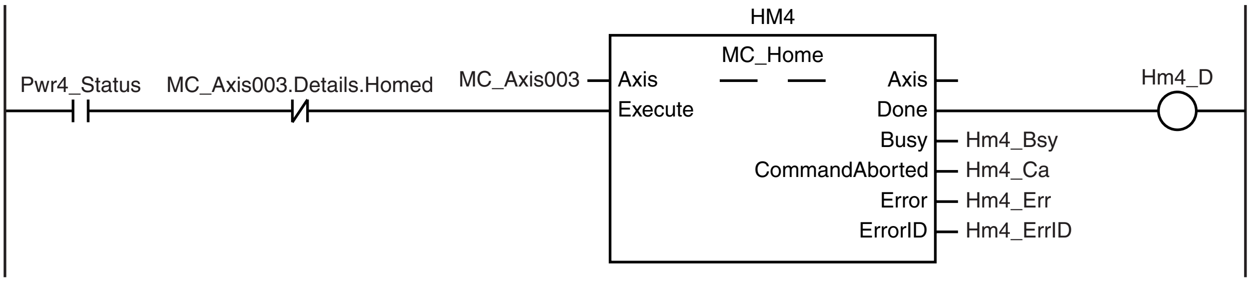


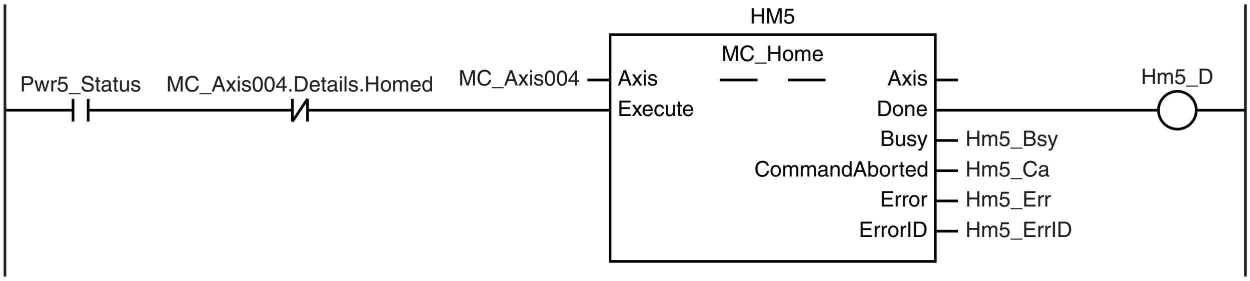

If the Servo is ON for axis 1 to axis 5 and the Home instruction is executed for each axis if home is not defined.



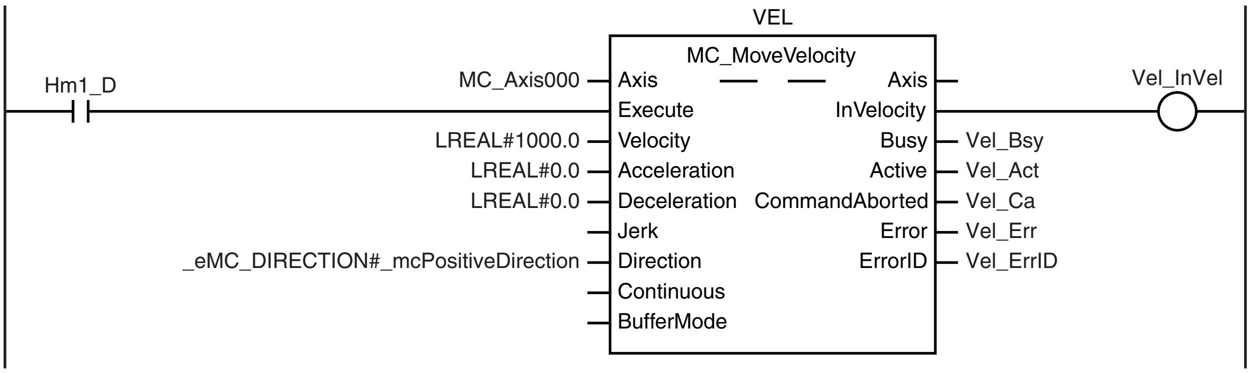



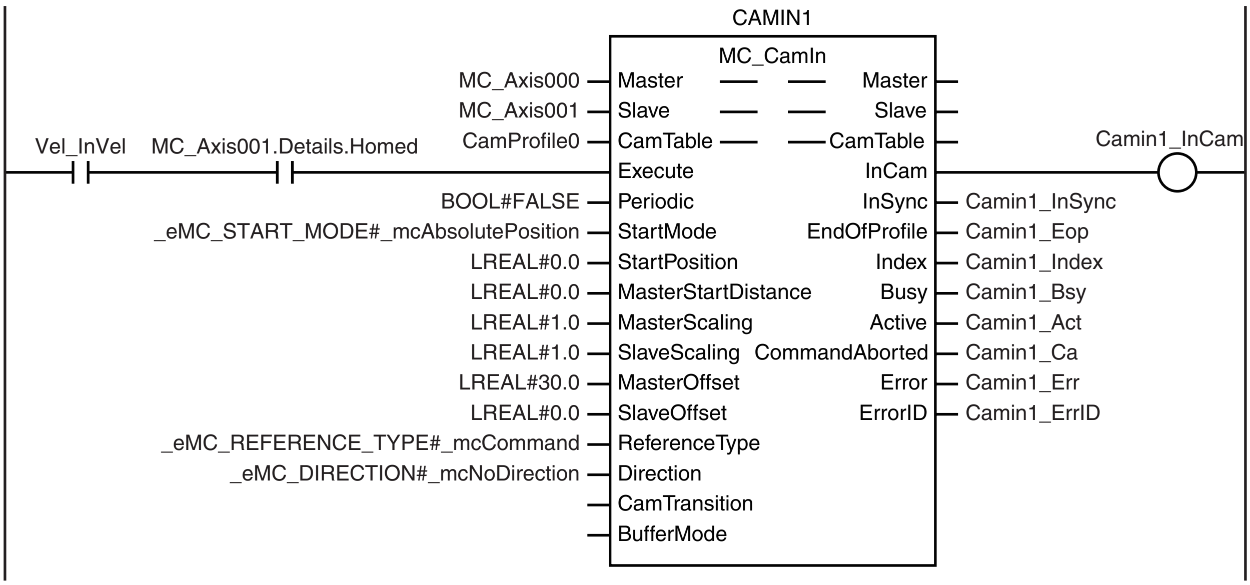


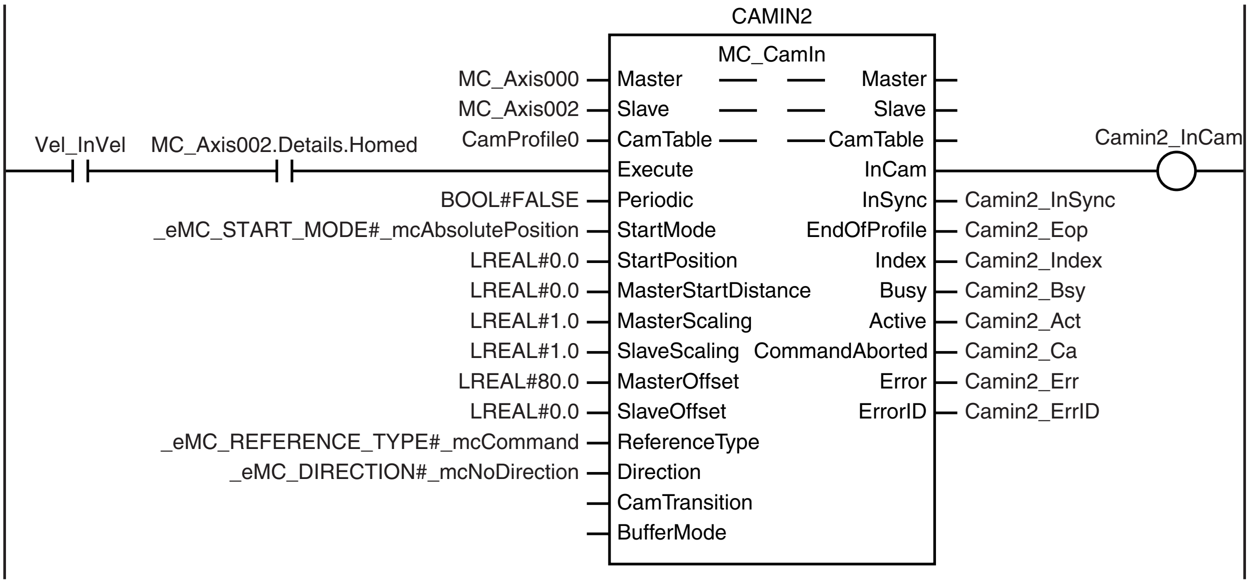

The MC_MoveVelocity (Velocity Control) instruction is executed after homing is completed for axis 1.



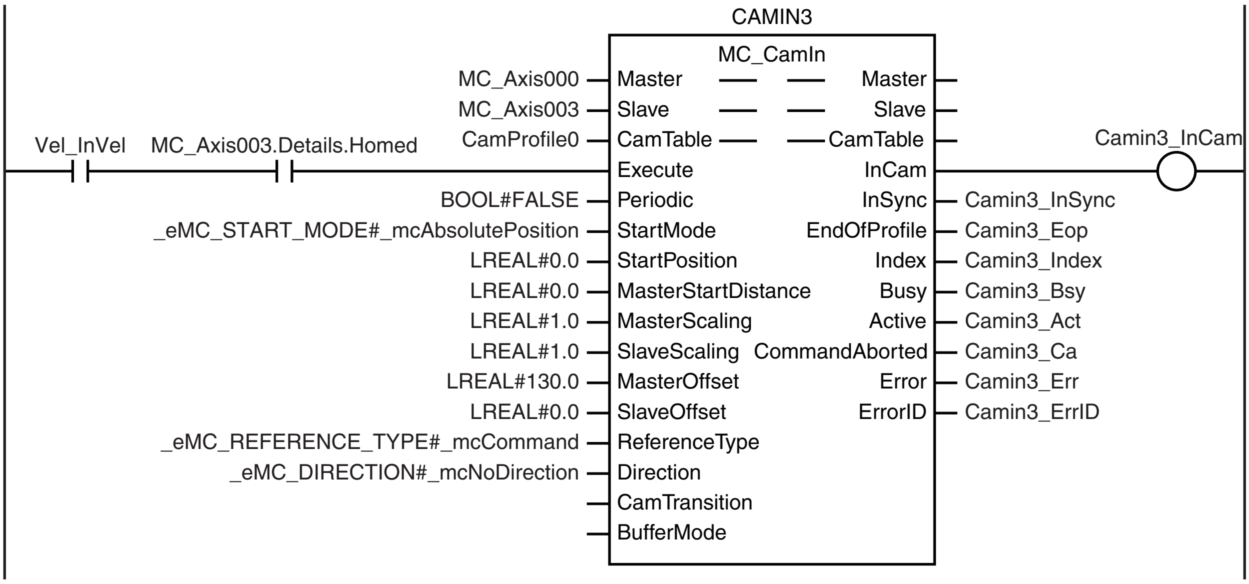

The MC_CamIn (Start Cam Operation) instruction is executed for axis 2 (slave axis) if *Vel_InVel* is TRUE for the MC_MoveVelocity (Velocity Control) instruction and homing is completed for axis 2.



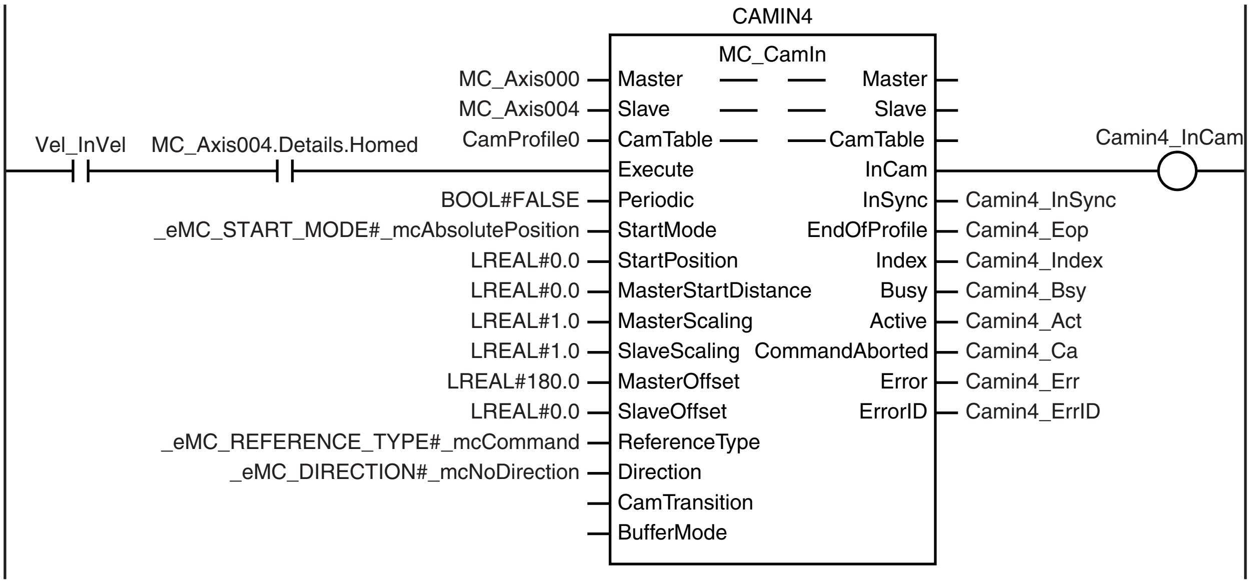
The MC_CamIn (Start Cam Operation) instruction is executed for axis 3 (slave axis) if *Vel_InVel* is TRUE for the MC_MoveVelocity (Velocity Control) instruction and homing is completed for axis 3.




The MC_CamIn (Start Cam Operation) instruction is executed for axis 4 (slave axis) if *Vel_InVel* is TRUE for the MC_MoveVelocity (Velocity Control) instruction and homing is completed for axis 4.



The MC_CamIn (Start Cam Operation) instruction is executed for axis 5 (slave axis) if *Vel_InVel* is TRUE for the MC_MoveVelocity (Velocity Control) instruction and homing is completed for axis 5.

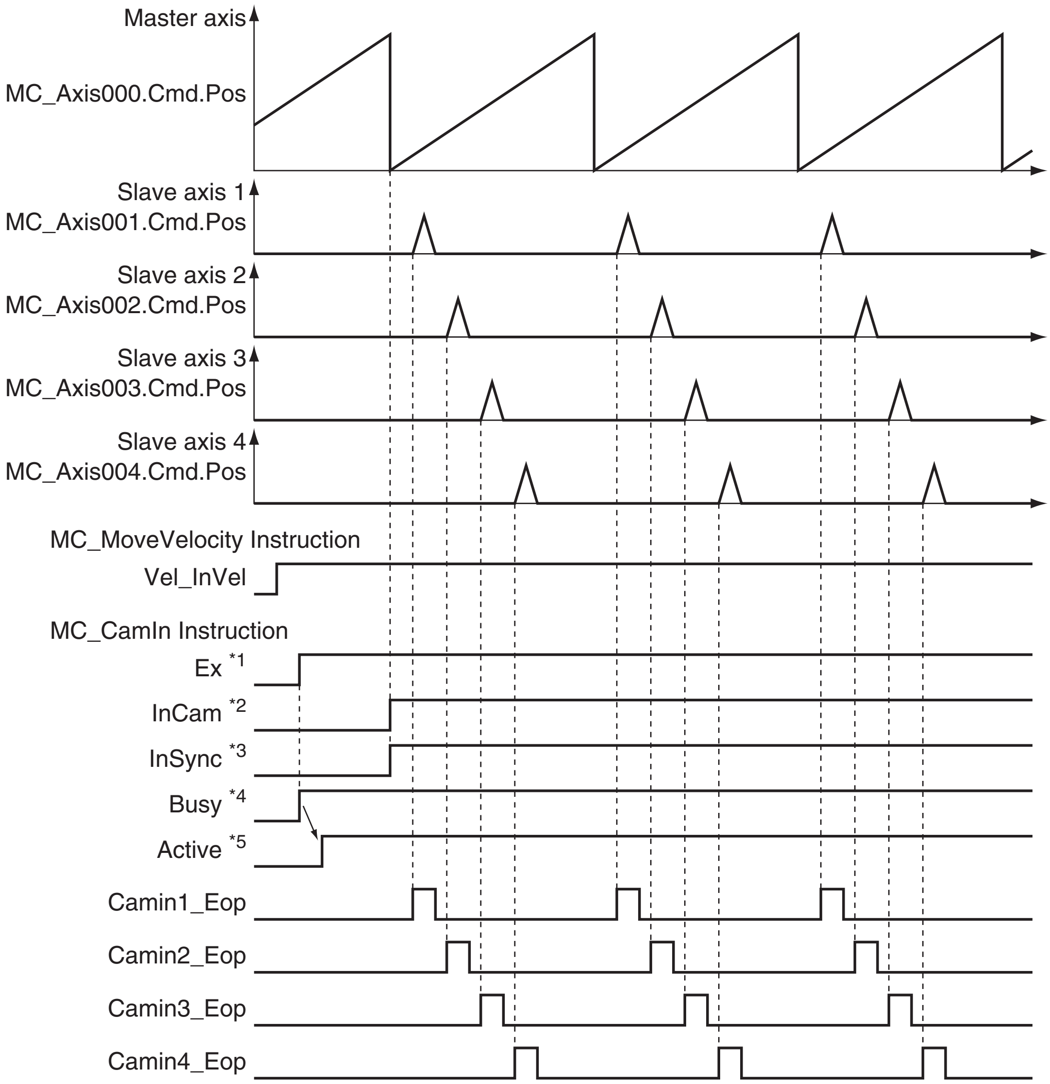



## Structured Text (ST)

### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for the master axis, axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 2. |
| MC_Axis002 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 3. |
| MC_Axis003 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 4. |
| MC_Axis004 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 5. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| Vel_InVel | BOOL | FALSE | This variable is assigned to the *InVelocity* output variable from the VEL instance of the MC_MoveVelocity instruction. It is TRUE when the target velocity is reached. |
| CamProfile0 | ARRAY[0..360] OF _sMC_CAM_REF | --- | This is the cam data variable. This variable is assigned to the *CamTable* input variables from the CAMIN1 to CAMIN4 instances of the MC_CamIn instruction. The array elements ARRAY[0..N] are set with the Cam Editor. |
| Camin1_Ex | BOOL | FALSE | The CAMIN1 to CAMIN4 instances of MC_CamIn are executed when this variable changes to TRUE. |
| Camin2_Ex | BOOL | FALSE | |
| Camin3_Ex | BOOL | FALSE | |
| Camin4_Ex | BOOL | FALSE | |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

## Timing Chart




*1 The timing is the same for the following: *Camin1_InCam*, *Camin2_InCam*, *Camin3_InCam*, and *Camin4_InCam*.

*2 The timing is the same for the following: *Camin1_InSync*, *Camin2_InSync*, *Camin3_InSync*, and *Camin4_InSync*. In this sample, *MasterStartDistance* is 0, so *InSync* changes to TRUE from when the master axis is 0°.

*3 The timing is the same for the following: *Camin1_Bsy*, *Camin2_Bsy*, *Camin3_Bsy*, and *Camin4_Bsy*.

*4 The timing is the same for the following: *Camin1_Act*, *Camin2_Act*, *Camin3_Act*, and *Camin4_Act*.

## Sample Programming

```
// Processing when input parameters are not set
IF InitFlag=FALSE THEN

    //MC_MoveVelocity Input Parameter
    Vel_Vel     := LREAL#1000.0;
    Vel_Acc     := LREAL#0.0;
    Vel_Dec     := LREAL#0.0;
    Vel_Dir     := _eMC_DIRECTION#_mcPositiveDirection;

    //MC_CamIn Input Parameter
    Camin1_Em       := TRUE;
    Camin1_Sm       := _eMC_START_MODE#_mcAbsolutePosition;
    Camin1_Sp       := LREAL#0.0;
    Camin1_Msd      := LREAL#0.0;
    Camin1_Ms       := LREAL#1.0;
```

```
        Camin1_Ss        := LREAL#1.0;
        Camin1_Mo        := LREAL#30.0;
        Camin1_So        := LREAL#0.0;
        Camin1_Rt        := _eMC_REFERENCE_TYPE#_mcCommand;
        Camin1_Dir       := _eMC_DIRECTION#_mcNoDirection;
        Camin2_Em        := TRUE;
        Camin2_Sm        := _eMC_START_MODE#_mcAbsolutePosition;
        Camin2_Sp        := LREAL#0.0;
        Camin2_Msd       := LREAL#0.0;
        Camin2_Ms        := LREAL#1.0;
        Camin2_Ss        := LREAL#1.0;
        Camin2_Mo        := LREAL#80.0;
        Camin2_So        := LREAL#0.0;
        Camin2_Rt        := _eMC_REFERENCE_TYPE#_mcCommand;
        Camin2_Dir       := _eMC_DIRECTION#_mcNoDirection;

        Camin3_Em        := TRUE;
        Camin3_Sm        := _eMC_START_MODE#_mcAbsolutePosition;
        Camin3_Sp        := LREAL#0.0;
        Camin3_Msd       := LREAL#0.0;
        Camin3_Ms        := LREAL#1.0;
        Camin3_Ss        := LREAL#1.0;
        Camin3_Mo        := LREAL#130.0;
        Camin3_So        := LREAL#0.0;
        Camin3_Rt        := _eMC_REFERENCE_TYPE#_mcCommand;
        Camin3_Dir       := _eMC_DIRECTION#_mcNoDirection;

        Camin4_Em        := TRUE;
        Camin4_Sm        := _eMC_START_MODE#_mcAbsolutePosition;
        Camin4_Sp        := LREAL#0.0;
        Camin4_Msd       := LREAL#0.0;
        Camin4_Ms        := LREAL#1.0;
        Camin4_Ss        := LREAL#1.0;
        Camin4_Mo        := LREAL#180.0;
        Camin4_So        := LREAL#0.0;
        Camin4_Rt        := _eMC_REFERENCE_TYPE#_mcCommand;
        Camin4_Dir       := _eMC_DIRECTION#_mcNoDirection;


        // Change InitFlag to TRUE after setting the input parameters.
        InitFlag := TRUE;

    END_IF;

    // If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 1 is turned ON.
    // If EtherCAT communications are not normal, the Servo is turned OFF.
    IF (StartPg=TRUE)
     AND (_EC_PDSlavTbl[MC_Axis000.Cfg.NodeAddress]=TRUE)
     AND (_EC_CommErrTbl[MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
        Pwr1_En:=TRUE;
    ELSE
        Pwr1_En:=FALSE;
    END_IF;

    // If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 2 is turned ON.
    // If EtherCAT communications are not normal, the Servo is turned OFF.
    IF (StartPg=TRUE)
     AND (_EC_PDSlavTbl[MC_Axis001.Cfg.NodeAddress]=TRUE)
     AND (_EC_CommErrTbl[MC_Axis001.Cfg.NodeAddress]=FALSE) THEN
        Pwr2_En:=TRUE;
    ELSE
        Pwr2_En:=FALSE;
    END_IF;

    // If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 3 is turned ON.
```

```
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis002.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis002.Cfg.NodeAddress]=FALSE) THEN
    Pwr3_En:=TRUE;
ELSE
    Pwr3_En:=FALSE;
END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 4 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis003.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis003.Cfg.NodeAddress]=FALSE) THEN
    Pwr4_En:=TRUE;
ELSE
    Pwr4_En:=FALSE;
END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 5 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis004.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis004.Cfg.NodeAddress]=FALSE) THEN
    Pwr5_En:=TRUE;
ELSE
    Pwr5_En:=FALSE;
END_IF;

// If a minor fault level error occurs for axis 1 to axis 5, the error handler for the device (FaultHandler) is
executed.
// Implement FaultHandler separately.
IF (MC_Axis000.MFaultLvl.Active=TRUE)
OR (MC_Axis001.MFaultLvl.Active=TRUE)
OR (MC_Axis002.MFaultLvl.Active=TRUE)
OR (MC_Axis003.MFaultLvl.Active=TRUE)
OR (MC_Axis004.MFaultLvl.Active=TRUE) THEN
    FaultHandler();
END_IF;

// If the Servo is ON for axis 1 and home is not defined, the Home instruction is executed for axis 1.
IF  (Pwr1_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
    Hm1_Ex:=TRUE;
END_IF;

// If the Servo is ON for axis 2 and home is not defined, the Home instruction is executed for axis 2.
IF  (Pwr2_Status=TRUE) AND (MC_Axis001.Details.Homed=FALSE) THEN
    Hm2_Ex:=TRUE;
END_IF;

// If the Servo is ON for axis 3 and home is not defined, the Home instruction is executed for axis 3.
IF  (Pwr3_Status=TRUE) AND (MC_Axis002.Details.Homed=FALSE) THEN
    Hm3_Ex:=TRUE;
END_IF;

// If the Servo is ON for axis 4 and home is not defined, the Home instruction is executed for axis 4.
IF  (Pwr4_Status=TRUE) AND (MC_Axis003.Details.Homed=FALSE) THEN
    Hm4_Ex:=TRUE;
END_IF;

// If the Servo is ON for axis 5 and home is not defined, the Home instruction is executed for axis 5.
IF  (Pwr5_Status=TRUE) AND (MC_Axis004.Details.Homed=FALSE) THEN
    Hm5_Ex:=TRUE;
END_IF;
```

```
// After homing is completed for axis 1, MC_MoveVelocity is executed.
IF Hm1_D=TRUE THEN
    Vel_Ex:=TRUE;
END_IF;

// If home is defined for axis 2 and Vel_InVel of MC_MoveVelocity is TRUE,
// the MC_CamIn instruction for axis 2 (slave axis) is executed.
IF (Vel_InVel=TRUE) AND (MC_Axis001.Details.Homed=TRUE) THEN
    Camin1_Ex := TRUE;
END_IF;

// If home is defined for axis 3 and Vel_InVel of MC_MoveVelocity is TRUE,
// the MC_CamIn instruction for axis 3 (slave axis) is executed.
IF (Vel_InVel=TRUE) AND (MC_Axis002.Details.Homed=TRUE) THEN
    Camin2_Ex := TRUE;
END_IF;

// If home is defined for axis 4 and Vel_InVel of MC_MoveVelocity is TRUE,
// the MC_CamIn instruction for axis 4 (slave axis) is executed.
IF (Vel_InVel=TRUE) AND (MC_Axis003.Details.Homed=TRUE) THEN
    Camin3_Ex := TRUE;
END_IF;

// If home is defined for axis 5 and Vel_InVel of MC_MoveVelocity is TRUE,
// the MC_CamIn instruction for axis 5 (slave axis) is executed.
IF (Vel_InVel=TRUE) AND (MC_Axis004.Details.Homed=TRUE) THEN
    Camin4_Ex := TRUE;
END_IF;

// MC_Power for axis 1
PWR1(
    Axis        := MC_Axis000,
    Enable      := Pwr1_En,
    Status      => Pwr1_Status,
    Busy        => Pwr1_Bsy,
    Error       => Pwr1_Err,
    ErrorID     => Pwr1_ErrID
);

// MC_Power for axis 2
PWR2(
    Axis        := MC_Axis001,
    Enable      := Pwr2_En,
    Status      => Pwr2_Status,
    Busy        => Pwr2_Bsy,
    Error       => Pwr2_Err,
    ErrorID     => Pwr2_ErrID
);

// MC_Power for axis 3
PWR3(
    Axis        := MC_Axis002,
    Enable      := Pwr3_En,
    Status      => Pwr3_Status,
    Busy        => Pwr3_Bsy,
    Error       => Pwr3_Err,
    ErrorID     => Pwr3_ErrID
);

// MC_Power for axis 4
PWR4(
    Axis        := MC_Axis003,
    Enable      := Pwr4_En,
    Status      => Pwr4_Status,
```

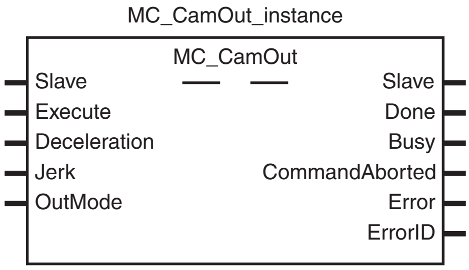
```
    Busy          => Pwr4_Bsy,
    Error         => Pwr4_Err,
    ErrorID       => Pwr4_ErrID
);

// MC_Power for axis 5
PWR5(
    Axis          := MC_Axis004,
    Enable        := Pwr5_En,
    Status        => Pwr5_Status,
    Busy          => Pwr5_Bsy,
    Error         => Pwr5_Err,
    ErrorID       => Pwr5_ErrID
);

// MC_Home for axis 1
HM1(
    Axis                  := MC_Axis000,
    Execute               := Hm1_Ex,
    Done                  => Hm1_D,
    Busy                  => Hm1_Bsy,
    CommandAborted        => Hm1_Ca,
    Error                 => Hm1_Err,
    ErrorID               => Hm1_ErrID
);

// MC_Home for axis 2
HM2(
    Axis                  := MC_Axis001,
    Execute               := Hm2_Ex,
    Done                  => Hm2_D,
    Busy                  => Hm2_Bsy,
    CommandAborted        => Hm2_Ca,
    Error                 => Hm2_Err,
    ErrorID               => Hm2_ErrID
);

// MC_Home for axis 3
HM3(
    Axis                  := MC_Axis002,
    Execute               := Hm3_Ex,
    Done                  => Hm3_D,
    Busy                  => Hm3_Bsy,
    CommandAborted        => Hm3_Ca,
    Error                 => Hm3_Err,
    ErrorID               => Hm3_ErrID
);

// MC_Home for axis 4
HM4(
    Axis                  := MC_Axis003,
    Execute               := Hm4_Ex,
    Done                  => Hm4_D,
    Busy                  => Hm4_Bsy,
    CommandAborted        => Hm4_Ca,
    Error                 => Hm4_Err,
    ErrorID               => Hm4_ErrID
);

// MC_Home for axis 5
HM5(
    Axis                  := MC_Axis004,
    Execute               := Hm5_Ex,
    Done                  => Hm5_D,
    Busy                  => Hm5_Bsy,
```

```
    CommandAborted      => Hm5_Ca,
    Error               => Hm5_Err,
    ErrorID             => Hm5_ErrID
);

//MC_MoveVelocity
VEL(
    Axis               := MC_Axis000,
    Execute            := Vel_Ex,
    Velocity           := Vel_Vel,
    Acceleration       := Vel_Acc,
    Deceleration       := Vel_Dec,
    Direction          := Vel_Dir,
    InVelocity         => Vel_InVel,
    Busy               => Vel_Bsy,
    Active             => Vel_Act,
    CommandAborted     => Vel_Ca,
    Error              => Vel_Err,
    ErrorID            => Vel_ErrID
);


//MC_CamIn
CAMIN1(
    Master                  := MC_Axis000,
    Slave                   := MC_Axis001,
    CamTable                := CamProfile0,
    Execute                 := Camin1_Ex,
    Periodic                := Camin1_Em,
    StartMode               := Camin1_Sm,
    StartPosition           := Camin1_Sp,
    MasterStartDistance     := Camin1_Msd,
    MasterScaling           := Camin1_Ms,
    SlaveScaling            := Camin1_Ss,
    MasterOffset            := Camin1_Mo,
    SlaveOffset             := Camin1_So,
    ReferenceType           := Camin1_Rt,
    Direction               := Camin1_Dir,
    InCam                   => Camin1_InCam,
    InSync                  => Camin1_InSync,
    EndOfProfile            => Camin1_Eop,
    Index                   => Camin1_Index,
    Busy                    => Camin1_Bsy,
    Active                  => Camin1_Act,
    CommandAborted          => Camin1_Ca,
    Error                   => Camin1_Err,
    ErrorID                 => Camin1_ErrID
);

CAMIN2(
    Master                  := MC_Axis000,
    Slave                   := MC_Axis002,
    CamTable                := CamProfile0,
    Execute                 := Camin2_Ex,
    Periodic                := Camin2_Em,
    StartMode               := Camin2_Sm,
    StartPosition           := Camin2_Sp,
    MasterStartDistance     := Camin2_Msd,
    MasterScaling           := Camin2_Ms,
    SlaveScaling            := Camin2_Ss,
    MasterOffset            := Camin2_Mo,
    SlaveOffset             := Camin2_So,
    ReferenceType           := Camin2_Rt,
    Direction               := Camin2_Dir,
    InCam                   => Camin2_InCam,
```

```
    InSync                  => Camin2_InSync,
    EndOfProfile            => Camin2_Eop,
    Index                   => Camin2_Index,
    Busy                    => Camin2_Bsy,
    Active                  => Camin2_Act,
    CommandAborted          => Camin2_Ca,
    Error                   => Camin2_Err,
    ErrorID                 => Camin2_ErrID
);

CAMIN3(
    Master                  := MC_Axis000,
    Slave                   := MC_Axis003,
    CamTable                := CamProfile0,
    Execute                 := Camin3_Ex,
    Periodic                := Camin3_Em,
    StartMode               := Camin3_Sm,
    StartPosition           := Camin3_Sp,
    MasterStartDistance     := Camin3_Msd,
    MasterScaling           := Camin3_Ms,
    SlaveScaling            := Camin3_Ss,
    MasterOffset            := Camin3_Mo,
    SlaveOffset             := Camin3_So,
    ReferenceType           := Camin3_Rt,
    Direction               := Camin3_Dir,
    InCam                   => Camin3_InCam,
    InSync                  => Camin3_InSync,
    EndOfProfile            => Camin3_Eop,
    Index                   => Camin3_Index,
    Busy                    => Camin3_Bsy,
    Active                  => Camin3_Act,
    CommandAborted          => Camin3_Ca,
    Error                   => Camin3_Err,
    ErrorID                 => Camin3_ErrID
);

CAMIN4(
    Master                  := MC_Axis000,
    Slave                   := MC_Axis004,
    CamTable                := CamProfile0,
    Execute                 := Camin4_Ex,
    Periodic                := Camin4_Em,
    StartMode               := Camin4_Sm,
    StartPosition           := Camin4_Sp,
    MasterStartDistance     := Camin4_Msd,
    MasterScaling           := Camin4_Ms,
    SlaveScaling            := Camin4_Ss,
    MasterOffset            := Camin4_Mo,
    SlaveOffset             := Camin4_So,
    ReferenceType           := Camin4_Rt,
    Direction               := Camin4_Dir,
    InCam                   => Camin4_InCam,
    InSync                  => Camin4_InSync,
    EndOfProfile            => Camin4_Eop,
    Index                   => Camin4_Index,
    Busy                    => Camin4_Bsy,
    Active                  => Camin4_Act,
    CommandAborted          => Camin4_Ca,
    Error                   => Camin4_Err,
    ErrorID                 => Camin4_ErrID
);
```

# MC_CamOut

Cam operation is ended for the axis specified with the input parameter.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_CamOut | End Cam Operation | FB |  | MC_CamOut_instance (<br>Slave :=*parameter,*<br>Execute :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>OutMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.*1 |
| Jerk (Reserved) | Jerk | LREAL | 0 | 0 | (Reserved) |
| OutMode (Reserved) | Sync End Mode Selection | _eMC_OUT_MODE | 0: _mcStop | 0*2 | (Reserved) |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 The default value for an enumeration variable is actually not the number, but the enumerator.

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

● **Output Variable Update Timing**

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the velocity reaches 0. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Slave | Slave Axis | _sAXIS_REF | --- | Specify the slave axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

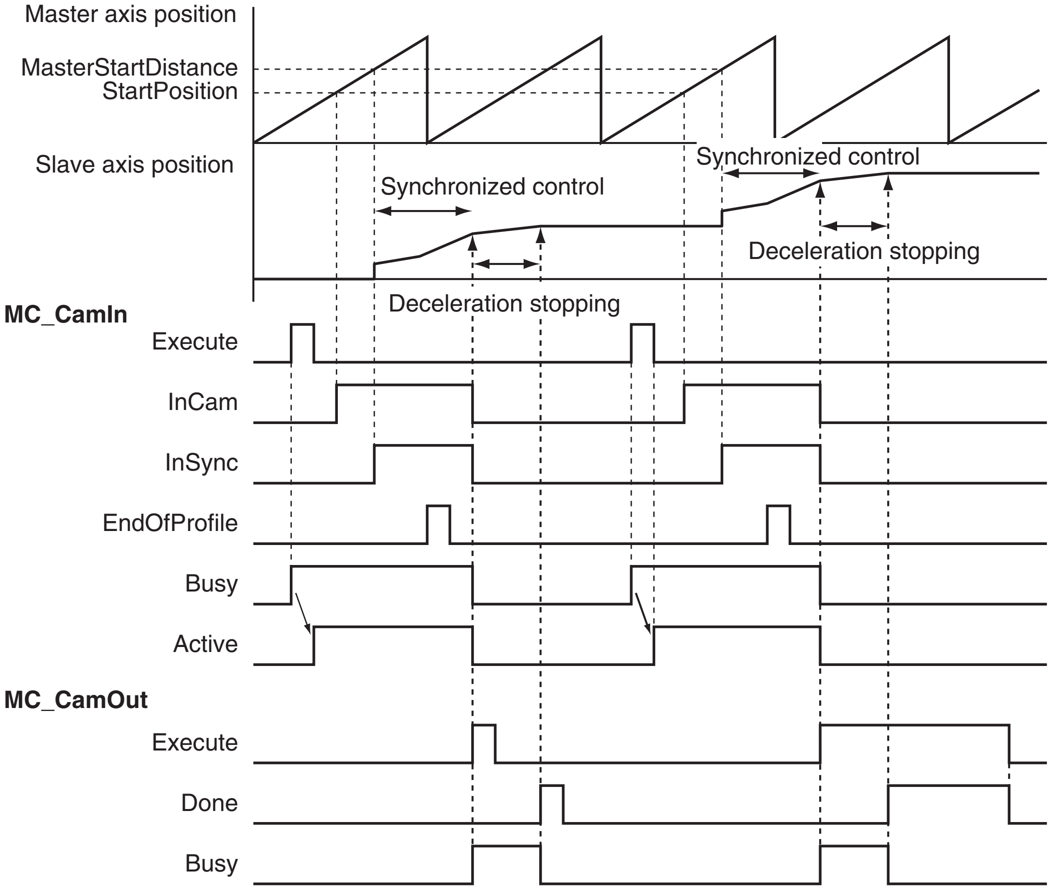
# Function

- The MC_CamOut instruction disables cam operation of the slave axis.
- When *Execute* changes to TRUE, the axis starts decelerating towards 0 velocity at the deceleration rate specified with *Deceleration* (Deceleration Rate).
- When the command velocity reaches 0, the instruction is completed.
- If you execute this instruction on an axis that is not in cam operation, an error will occur.

**Precautions for Correct Use**

- Cam data variables are global variables. You can therefore access or change the values of cam data variables from more than one task. If you change the values of cam data variables from more than one task, program the changes so that there is no competition in writing the value from more than one task.
- If you use exclusive control of global variables between tasks for a cam data variable, do not use the cam data variable for motion control instructions in a task that does not control the variable. An Incorrect Cam Table Specification error (error code: 5439 hex) will occur.

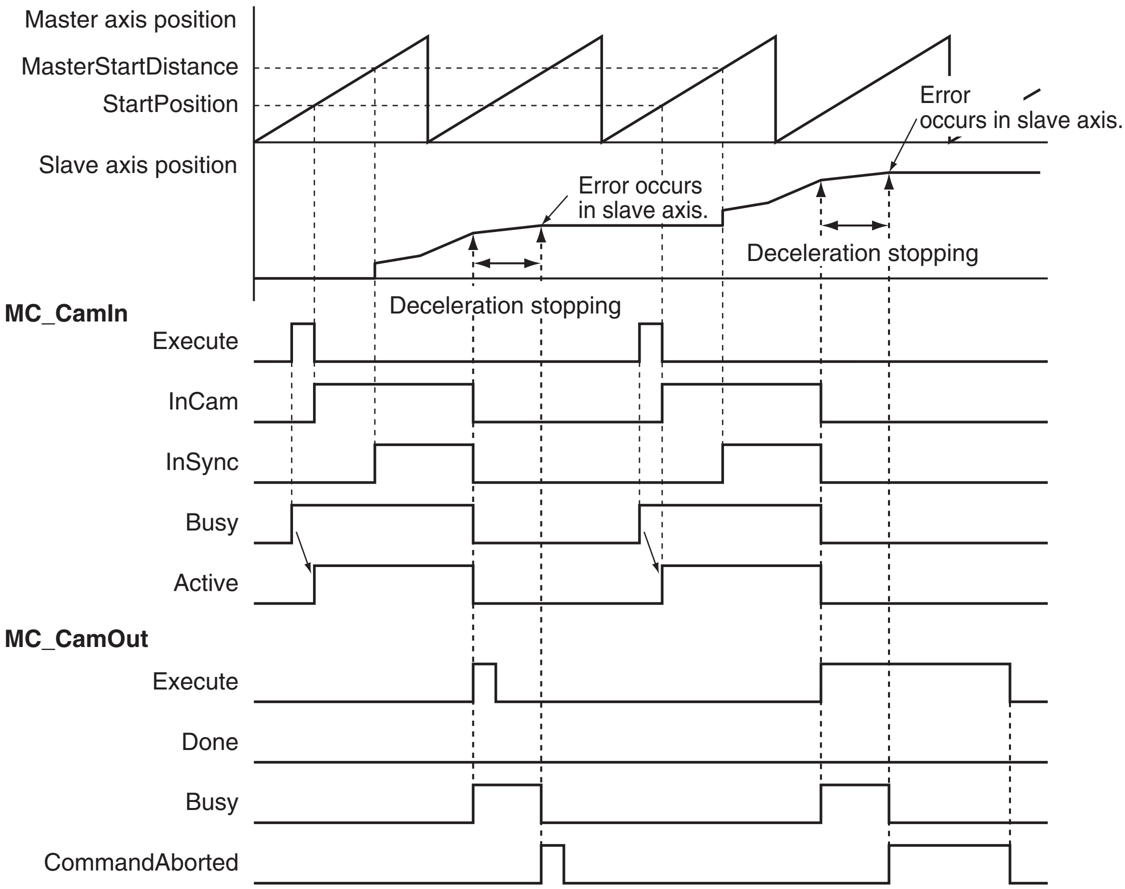
## Timing Charts



## Aborting the Instruction

If an axis error occurs for the slave axis during execution of this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing) changes to FALSE. The axis will decelerate at the rate specified with *Deceleration* (Deceleration Rate) for this instruction.

Refer to *A-1 Error Codes* for information on axis errors.




## Re-execution of Motion Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Instructions

- **Execution during Execution of Other Instructions**

If you execute this instruction while the MC_CamIn (Start Cam Operation) instruction is in execution, *CommandAborted* for the MC_CamIn instruction will change to TRUE and *Busy* for this instruction will change to TRUE. If this instruction is executed when the MC_CamIn (Start Cam Operation) instruction is not in execution, and an error will occur.

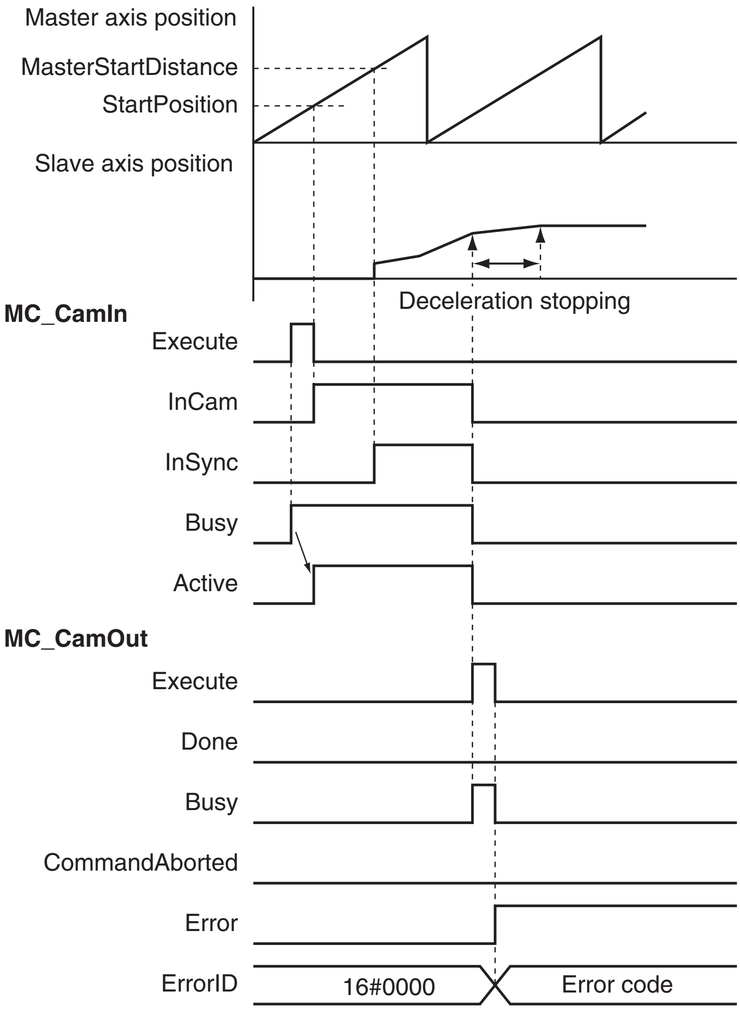
- **Execution of Other Instructions during Instruction Execution**

To use multi-execution of motion instructions for this instruction, specify the slave axis.
If you execute another instruction during execution of this instruction, you can specify either aborting or buffering.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

### ● Timing Chart When Error Occurs

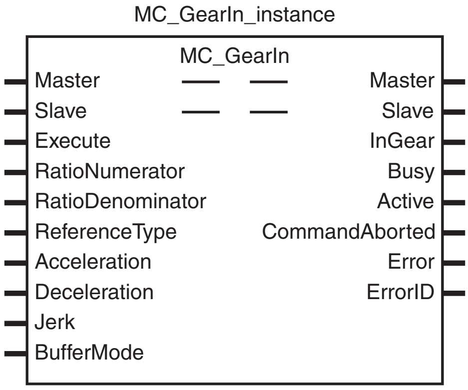



### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_GearIn

Specifies the gear ratio between the master axis and the slave axis and starts gear operation.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_GearIn | Start Gear Operation | FB |  | MC_GearIn_instance (<br>Master :=*parameter,*<br>Slave :=*parameter,*<br>Execute :=*parameter,*<br>RatioNumerator :=*parameter,*<br>RatioDenominator :=*parameter,*<br>ReferenceType :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>BufferMode :=*parameter,*<br>InGear =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Ratio Numerator | Gear Ratio Numerator | UNIT | Positive number | 10,000 | Specify the numerator of the electronic gear ratio between the master and slave axes. |
| Ratio Denominator | Gear Ratio Denominator | UNIT | Positive number | 10,000 | Specify the denominator of the electronic gear ratio between the master and slave axes. |
| Reference Type*1 | Position Type Selection | _eMC_ REFERENCE_ TYPE | 0: _mcCommand<br>1: _mcFeedback<br>2: _mcLatestCommand | 0*2 | Specify the position type.<br>0: Command position (value calculated in the previous primary period)<br>1: Actual position (value obtained in the same primary period)<br>2: Command position (value calculated in the same primary period) |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/$s^2$.*3 |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.*3 |
| Jerk (Reserved) | Jerk | LREAL | 0 | 0 | (Reserved) |
| BufferMode | Buffer Mode Selection | _eMC_ BUFFER_ MODE | 0: _mcAborting<br>1: _mcBuffered | 0*2 | Specify the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered |

*1 When you use *_mcLatestCommand*, the axis number set for the *Master* (Master Axis) in the system-defined variable for motion control must be lower than the axis number set for the *Slave* (Slave Axis) in the system-defined variable for motion control.

*2 The default value for an enumeration variable is actually not the number, but the enumerator.

*3 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| InGear | Gear Ratio Achieved | BOOL | TRUE or FALSE | TRUE when the slave axis reaches the target velocity. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| InGear | When the following relationship is established.<br>• Accelerating: Velocity of slave axis ≥ Velocity of master axis × Gear ratio<br>• Decelerating: Velocity of slave axis ≤ Velocity of master axis × Gear ratio | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Busy | When *Execute* changes to TRUE. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the instruction is started. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution.<br>• When the MC_GearOut instruction is executed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |



## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Master | Master Axis | _sAXIS_REF | --- | Specify the master axis.* |
| Slave | Slave Axis | _sAXIS_REF | --- | Specify the slave axis.* |

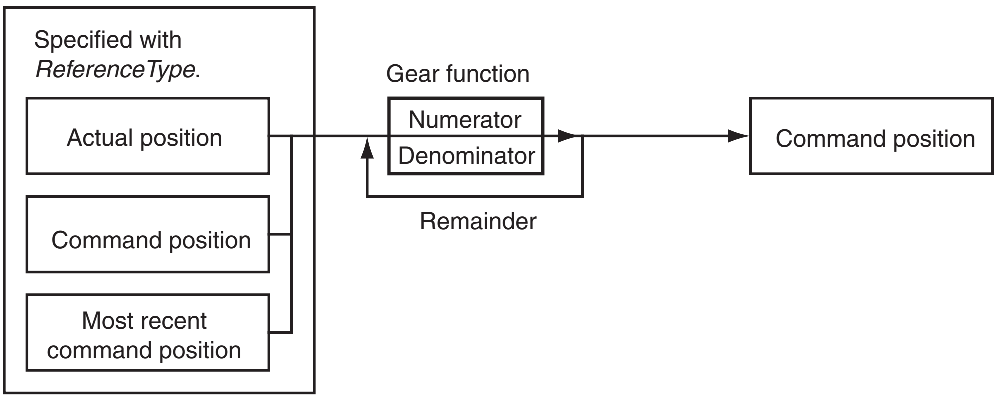
* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

**Precautions for Correct Use**

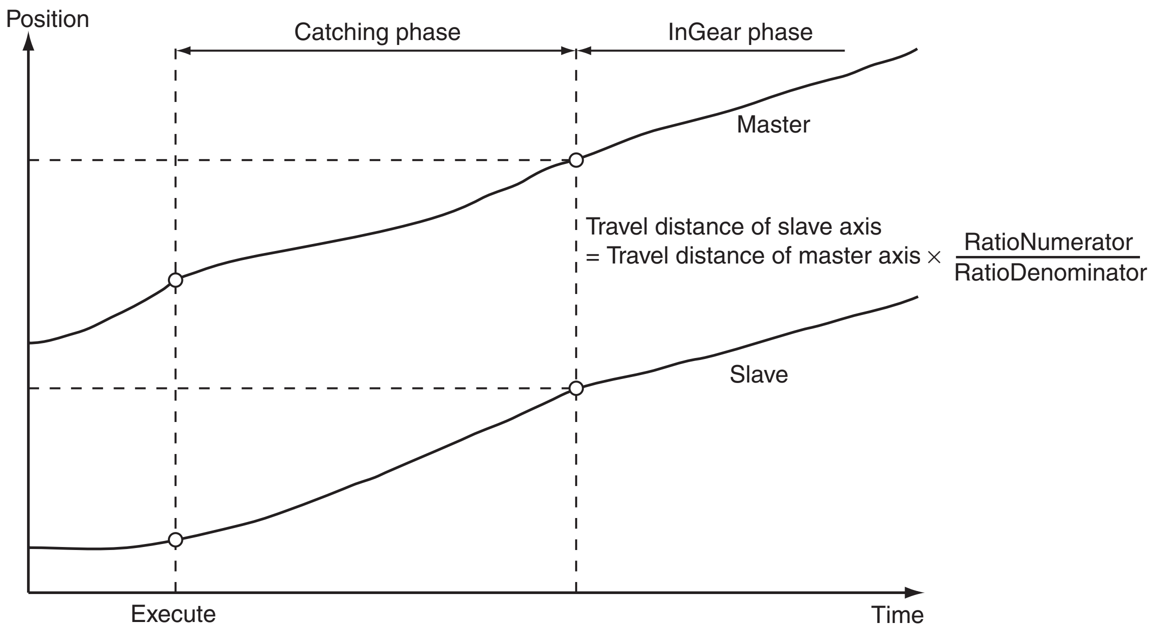
If you specify the same axis for the master axis and slave axis, a Master and Slave Defined as Same Axis minor fault (error code 5436 hex) will occur.

# Function

- The MC_GearIn instruction performs gear operation for the slave axis specified with *Slave*. The following parameters are also specified: *RatioNumerator* (Gear Ratio Numerator), *RatioDenominator* (Gear Ratio Denominator), *ReferenceType* (Position Type), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate).
- For the master axis, you can specify the command position, actual position, or most recent command position.



- After operation starts, the slave axis uses the velocity of the master axis multiplied by the gear ratio for its target velocity, and accelerates/decelerates accordingly.
- The catching phase exists until the target velocity is reached. The InGear phase exists after that.




- Electronic gear operation starts when *Execute* changes to TRUE.

**Precautions for Safe Use**

Do not execute the MC_SetPosition instruction for the *Master* (Master Axis) if you use this instruction. If the MC_SetPosition instruction is executed for the *Master* (Master Axis), the *Slave* (Slave Axis) may follow the master axis quickly.
If you want to use the MC_SetPosition instruction for the *Master* (Master Axis), disable the relationship between the *Master* (Master Axis) and *Slave* (Slave Axis) before executing the instruction.
Refer to *Precautions for Master and Auxiliary Axes in Synchronized Control* on page 1-7 for precautions on the master axis.

## ● *ReferenceType* (Position Type Selection)

You can select one of the following position types.

- *_mcCommand*: Command position (value calculated in the previous primary period)
  The master axis command position that was calculated in the previous primary period is used for the current period.
  The command value that was calculated for the master axis in the last primary period is used to calculate the command position of the slave axis in the current period.
- *_mcFeedback*: Value obtained in the same primary period
  The actual position of the master axis that was obtained in the same primary period is used.
- *_mcLatestCommand*: Command position (value calculated in the same primary period)
  The command position of the master axis that was calculated in the same primary period is used. This enables the use of information that is more recent than for *_mcCommand*. However, the axis number of the master axis must be set lower than the axis number of the slave axis.
  If the axis number of the slave axis is lower than the axis number of the master axis, *Error* will change to TRUE. A Master/Slave Axis Numbers Not in Ascending Order error (error code: 5438 hex) will be output to *ErrorID*.

**Additional Information**

The command position that is calculated in the same primary period enables greater precision in synchronization than the command position that was calculated in the previous primary period. However, the axis number set for the *Master* (Master Axis) in the system-defined variable for motion control must be lower than the axis number set for the *Slave* (Slave Axis) in the system-defined variable for motion control.

### ● Relationship between Axis Types and Position Types

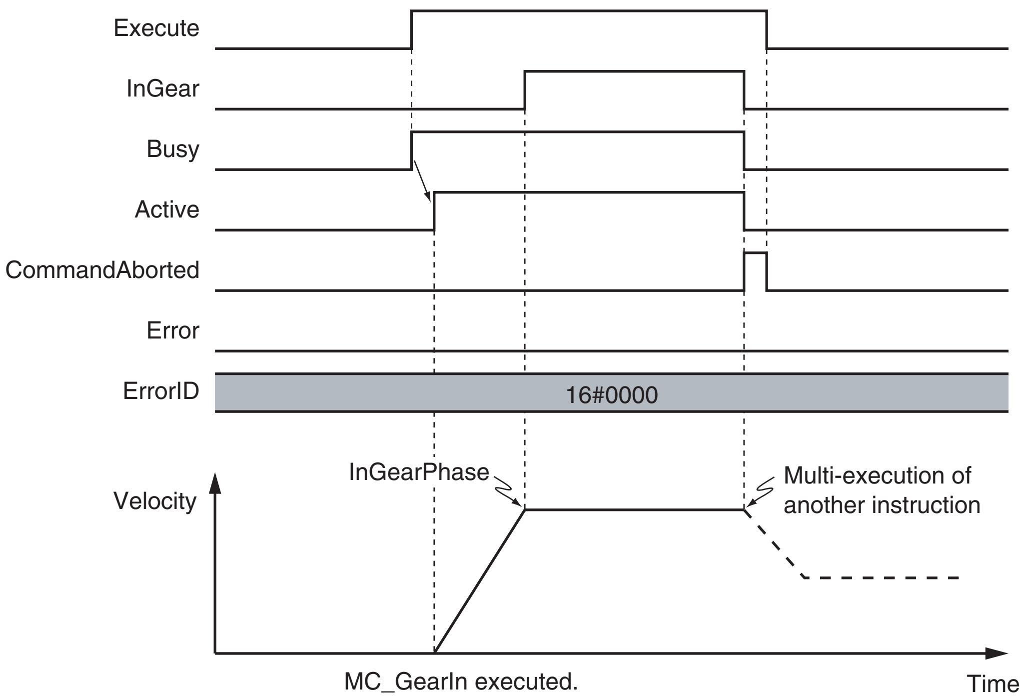
The relationship between the axis types that you can monitor and position types that is monitored is shown below.

| Axis Type | ReferenceType | |
|---|---|---|
| | _mcCommand or _mcLatestCommand | _mcFeedback |
| Servo axis | OK | OK |
| Encoder axis | No* | OK |
| Virtual servo axis | OK | OK |
| Virtual encoder axis | No* | OK |

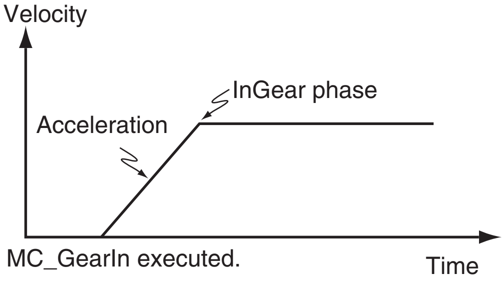
* A Position Type Selection Out of Range error (error code: 5430 hex) occurs when the instruction is executed.

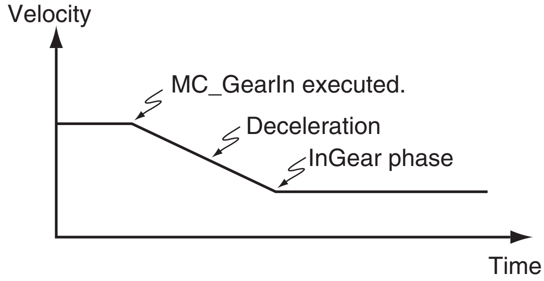
## Timing Charts

- *Busy* (Executing) changes to TRUE at the same time as *Execute* changes to TRUE. *Active* (Controlling) changes to TRUE in the next period.
- *InGear* (Gear Ratio Reached) changes to TRUE when the target velocity is reached.
- If another instruction aborts this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing), *Active* (Controlling), and *InGear* (Gear Ratio Reached) change to FALSE.
- Use the MC_GearOut (End Gear Operation) or MC_Stop instruction to stop electronic gear operation before it is completed.

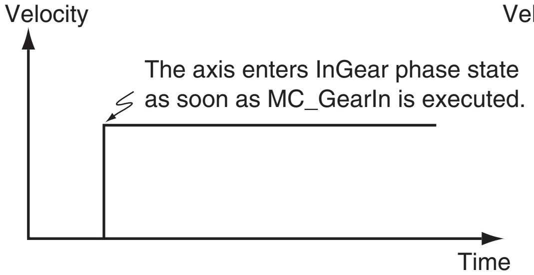


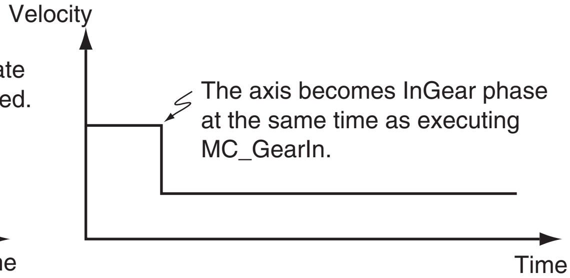

You can specify the *Acceleration* (Acceleration Rate) and *Deceleration* (Deceleration Rate) as input variables.




When the *Acceleration* (Acceleration Rate) or *Deceleration* (Deceleration Rate) is 0 and you execute this instruction, the axis will reach the target velocity without accelerating or decelerating.

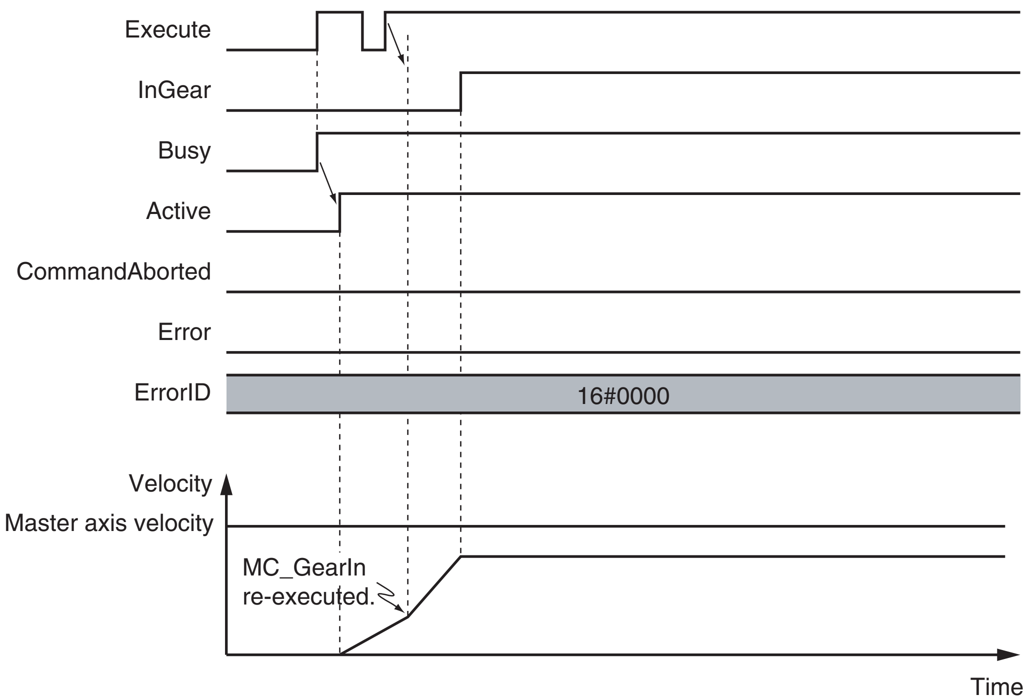


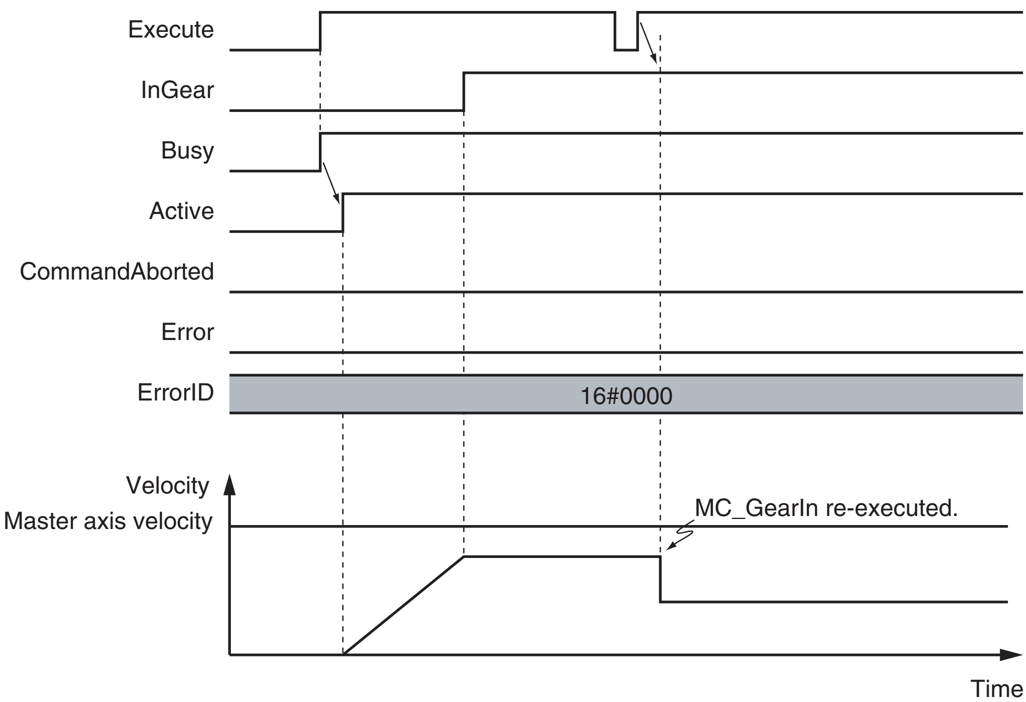

## Re-execution of Motion Instructions

You can change the operation of the instruction if you change an input variable during positioning and change *Execute* to TRUE again. Input variables *RatioNumerator* (Gear Ratio Numerator), *RatioDenominator* (Gear Ratio Denominator), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate) can be changed by re-executing the motion control instruction. For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

The following timing chart shows when the instruction is re-executed during the Catching phase to change the *Acceleration* (Acceleration Rate).



The following timing chart shows when the instruction is re-executed during the InGear phase to change the *RatioNumerator* (Gear Ratio Numerator) and *RatioDenominator* (Gear Ratio Denominator). The motion is the same as when *Acceleration* (Acceleration Rate) and *Deceleration* (Deceleration Rate) are both set to 0.




## Multi-execution of Motion Instructions

For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

You can switch to this instruction or buffer this instruction if you execute it during execution of another instruction. You can buffer one instruction per axis.

Specify the operation of this instruction by using *BufferMode* (Buffer Mode Selection) for multi-execution of instructions.

| Buffer Mode Selection | Description |
|---|---|
| Aborting | Aborts the instruction being executed and switches to this instruction.<br>If the direction of axis motion is reversed by switching instructions, the motion will be reversed according to the Operation at Reversing axis parameter. |
| Buffered | Automatically executes the buffered instruction after the current instruction is completed. |

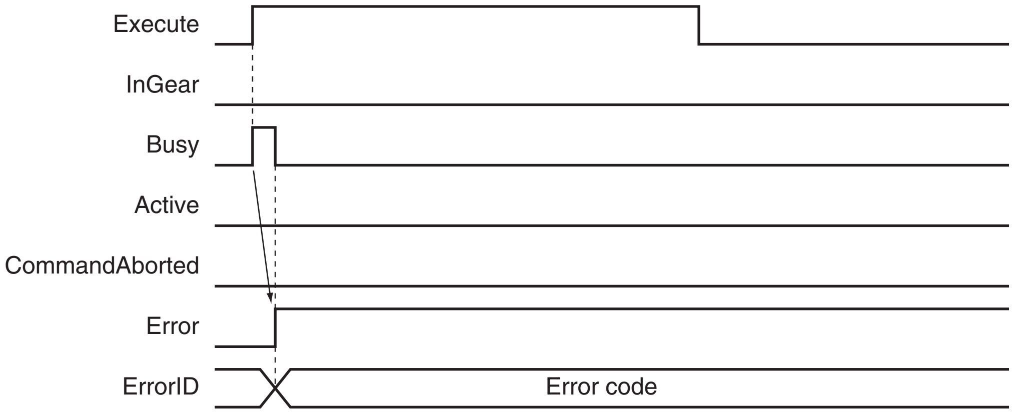
For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution of Other Instructions during Instruction Execution

Another instruction with the Buffer Mode set to *Aborting* can be executed during execution of this instruction. In that case, the gear operation is stopped and the operation of the aborting instruction is started. You cannot specify any Buffer Mode with other than *Aborting*.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).




**Additional Information**

- The slave axis is not affected by the error status of the master axis during synchronized control. The error status of the master axis is cleared and the slave axis continues electronic gear operation after the master axis operates normally.
- The master axis is not affected if an error occurs for the slave axis during startup or execution of this instruction.

### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# Sample Programming

This section shows sample programming for operation proportional to a gear ratio.

**Additional Information**

You can specify only the initial values for input variables that are reserved. Parameters are not specified in this sample.

## Parameter Settings

The minimum settings required for this sample programming are given below.

### ● Setting Axis Parameters

#### Axis Types

| Axis | Axis Type |
|---|---|
| Axis 1 | Servo axis (master axis) |
| Axis 2 | Servo axis (slave axis) |
| Axis 3 | Servo axis (slave axis) |

#### Count Modes

| Axis | Count Mode |
|---|---|
| Axis 1 | Rotary Mode |
| Axis 2 | Rotary Mode |
| Axis 3 | Rotary Mode |

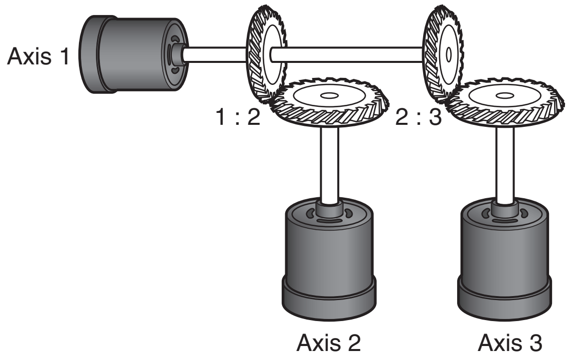
#### Ring Counter

| Axis | Modulo maximum position | Modulo minimum position |
|---|---|---|
| Axis 1 | 360 | 0 |
| Axis 2 | 360 | 0 |
| Axis 3 | 360 | 0 |

#### Units of Display

| Axis | Unit of Display |
|---|---|
| Axis 1 | mm |
| Axis 2 | mm |
| Axis 3 | mm |

## Operation Example




**1** Starting the Master Axis

The master axis (axis 1) is an actual servo axis and it is operated with velocity control.

**2** Executing the Slave Axes

When the actual velocity for the master axis reaches the target velocity, gear operation is performed so that the gear ratio of axis 2 (slave axis) is 1:2 and axis 3 (slave axis) is 2:3 against the actual position of the master axis.

**3** Stopping the Slave Axes

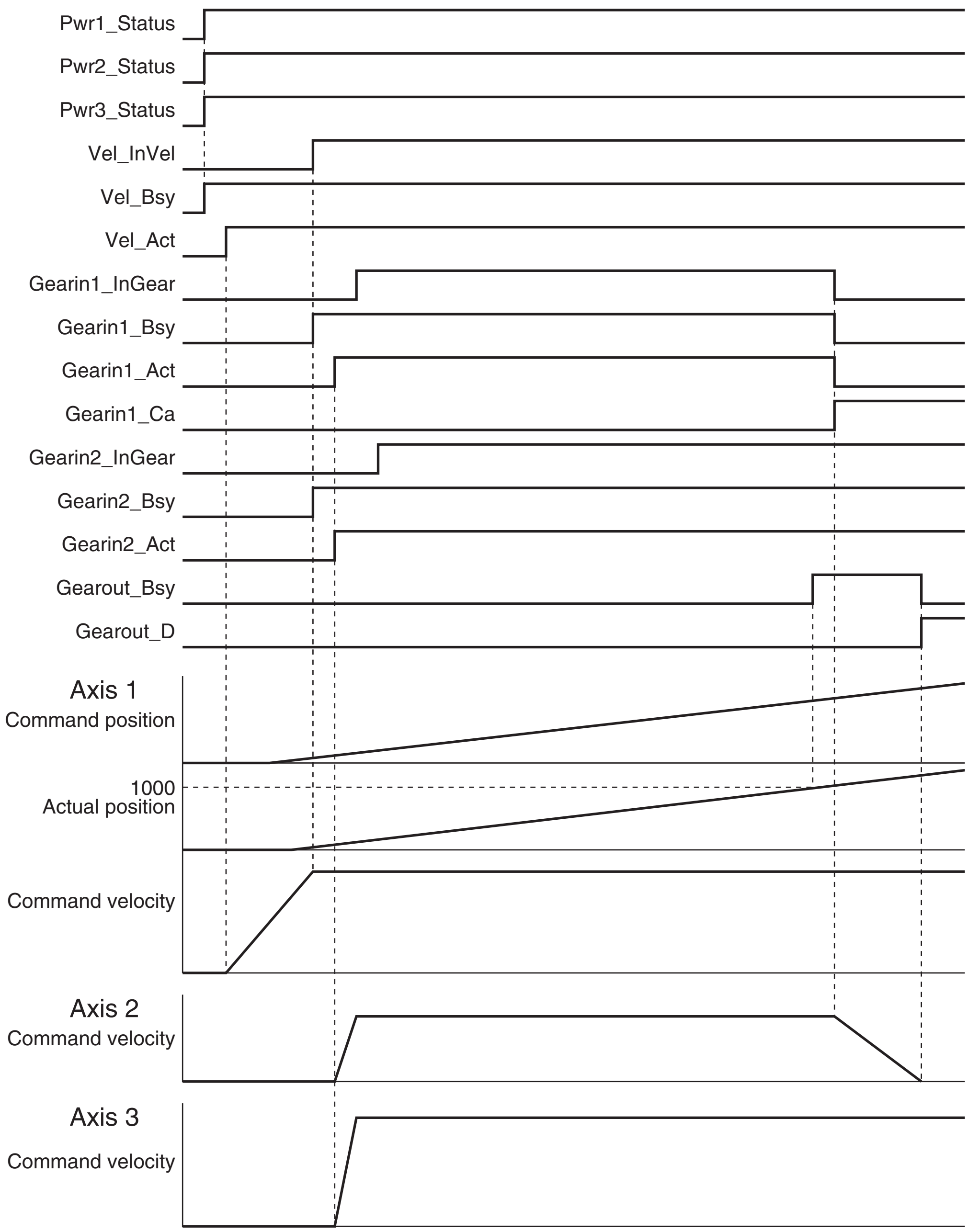
When the actual position of the master axis *MC_Axis000.Act.Pos* exceeds 1000.0, gear operation of axis 2 is ended and axis 2 decelerates to a stop with deceleration rate *DecRate*. Axis 3 continues gear operation.

## Ladder Diagram

### ● Main Variables

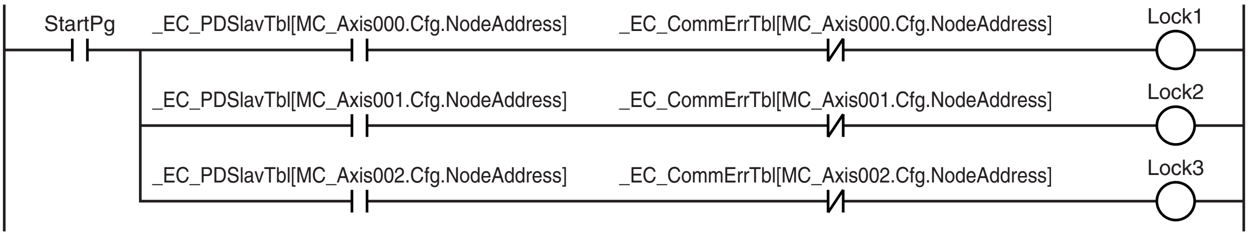
| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for the master axis, axis 1. |
| MC_Axis000.Act.Pos | LREAL | --- | This variable gives the actual current position of axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 2. |
| MC_Axis002 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 3. |
| Pwr1_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR1 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Pwr2_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR2 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Pwr3_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR3 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| Vel_InVel | BOOL | FALSE | This variable is assigned to the *InVelocity* output variable from the VEL instance of the MC_MoveVelocity instruction. It is TRUE when the target velocity is reached. |
| Gearin1_Act | BOOL | FALSE | This variable is assigned to the *Active* output variable from the GEARIN1 instance of the MC_GearIn instruction. It is TRUE during control operations for GEARIN1. |
| Gearout_Ex | BOOL | FALSE | The GEAROUT instance of MC_GearOut is executed when this variable changes to TRUE. |

- **Timing Chart**

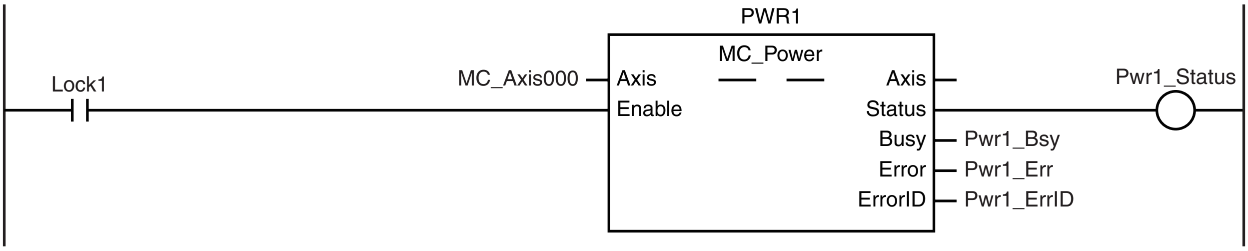


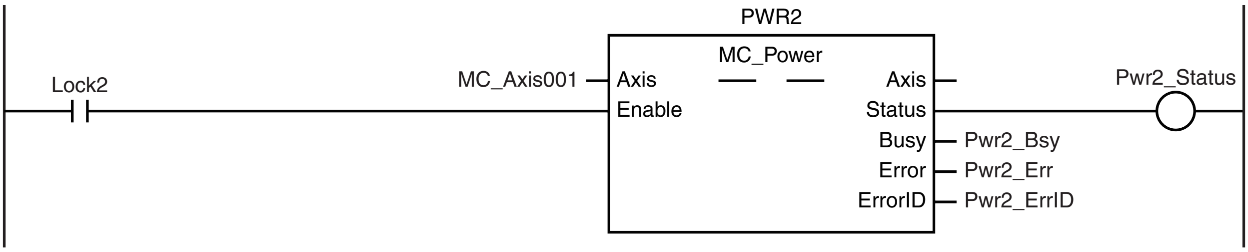
## ● Sample Programming

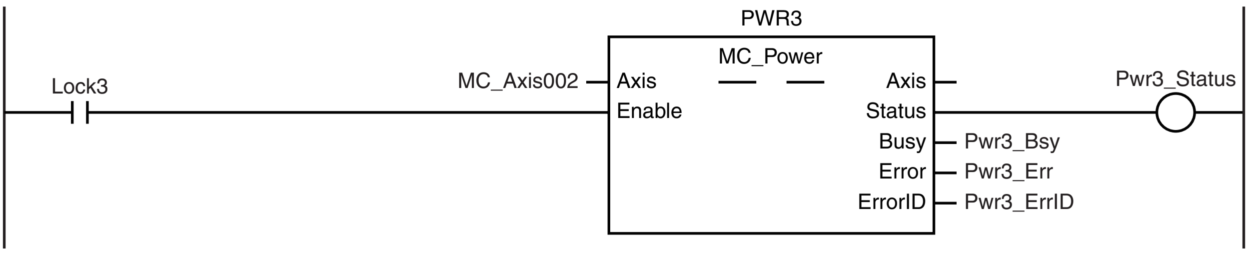
If *StartPg* is TRUE, EtherCAT communications for the axes are checked to see if process data communications are normal.



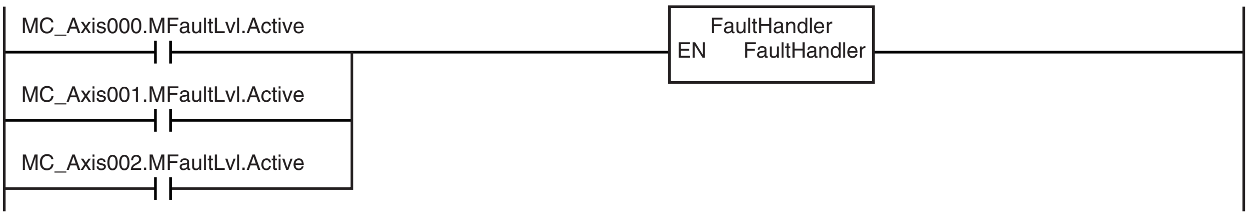

If process data communications are active, the Servo is turned ON for the axes.



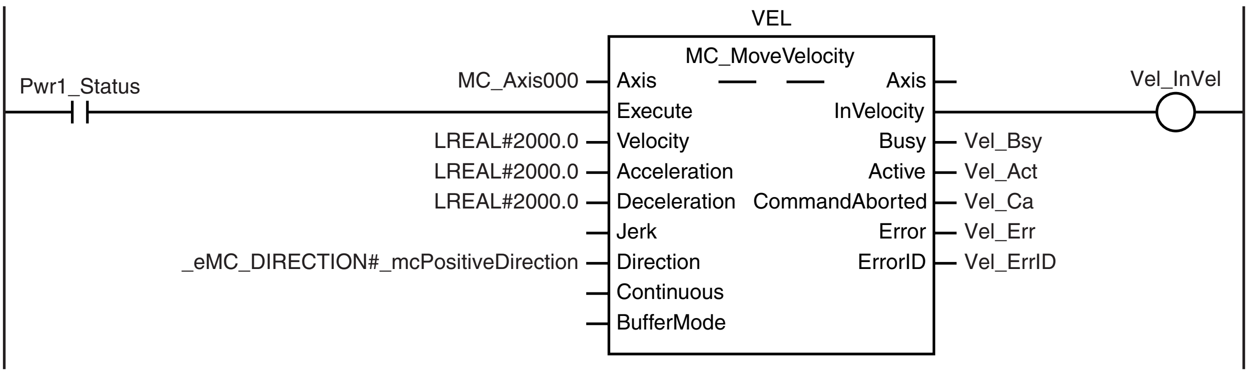



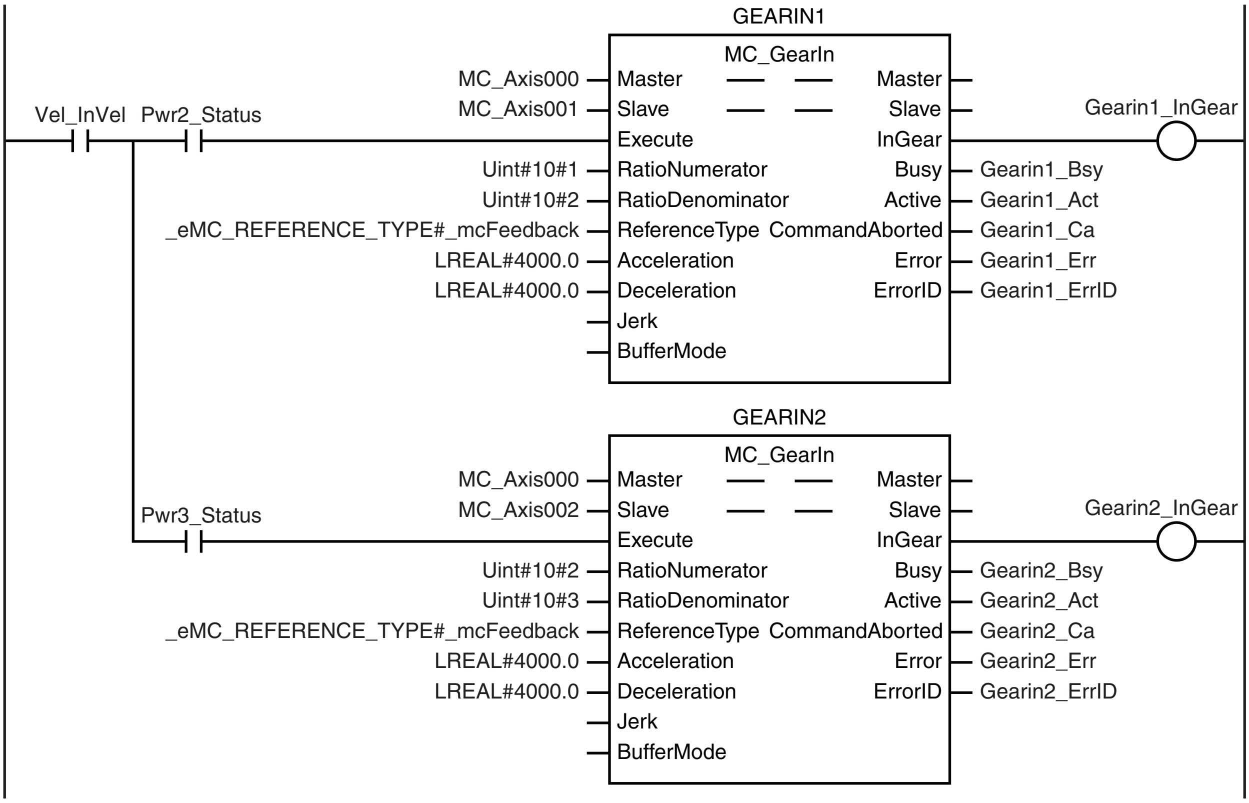

If there is a minor fault level error for a composition axis, the error handler for the device (FaultHandler) is executed. Design FaultHandler separately.



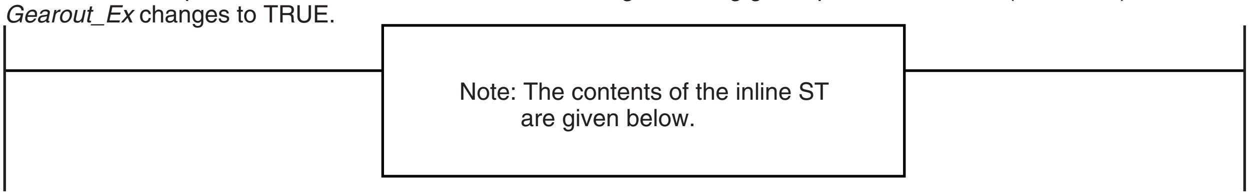

The MC_MoveVelocity (Velocity Control) instruction is executed after the Servo is turned ON for the master axis (axis 1).



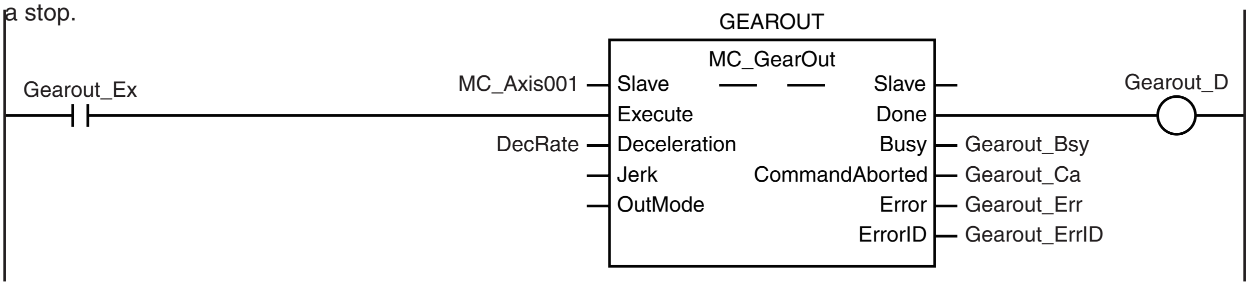
When the actual velocity for the master axis (axis 1) reaches the target velocity, gear operation is performed so that the gear ratio of axis 2 (slave axis) is 1:2 and axis 3 (slave axis) is 2:3.




When the actual position of the master axis is 1000.0 or higher during gear operation of axis 2 (slave axis), *Gearout_Ex* changes to TRUE.




When *Gearout_Ex* changes to TRUE, gear operation is stopped for the axis 2 (slave axis). The axis decelerates to a stop.




## Contents of Inline ST

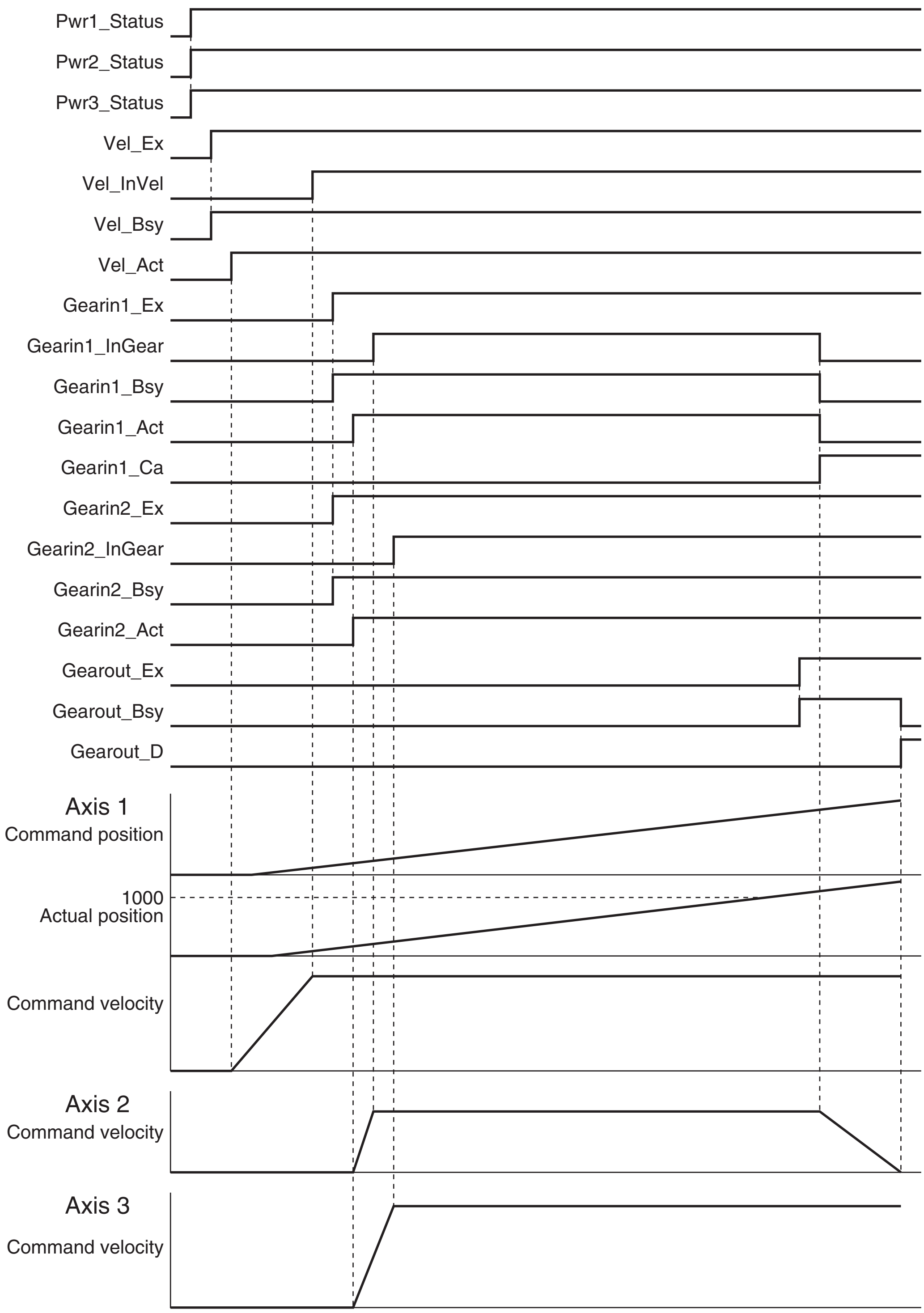
```
IF (Gearin1_Act=TRUE) AND (MC_Axis000.Act.Pos>=LREAL#1000.0) THEN
    Gearout_Ex := TRUE;
END_IF;
```

## Structured Text (ST)

### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for the master axis, axis 1. |
| MC_Axis000.Act.Pos | LREAL | --- | This variable gives the actual current position of axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 2. |
| MC_Axis002 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 3. |
| Pwr1_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR1 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Pwr2_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR2 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Pwr3_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR3 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| Vel_InVel | BOOL | FALSE | This variable is assigned to the *InVelocity* output variable from the VEL instance of the MC_MoveVelocity instruction. It is TRUE when the target velocity is reached. |
| Gearin1_Act | BOOL | FALSE | This variable is assigned to the *Active* output variable from the GEARIN1 instance of the MC_GearIn instruction. It is TRUE during control operations for GEARIN1. |
| Gearout_Ex | BOOL | FALSE | The GEAROUT instance of MC_GearOut is executed when this variable changes to TRUE. |
| Vel_Ex | BOOL | FALSE | The VEL instance of MC_MoveVelocity is executed when this variable changes to TRUE. |
| Gearin1_Ex | BOOL | FALSE | The GEARIN1 instance of MC_GearIn is executed when this variable changes to TRUE. |
| Gearin2_Ex | BOOL | FALSE | The GEARIN2 instance of MC_GearIn is executed when this variable changes to TRUE. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

## Timing Chart



## Sample Programming

```
// Processing when input parameters are not set
IF InitFlag = FALSE THEN
    // MC_MoveVelocity parameters
    Vel_Vel     := LREAL#2000.0;
    Vel_Acc     := LREAL#2000.0;
    Vel_Dec     := LREAL#2000.0;
    Vel_Dir     := _eMC_DIRECTION#_mcPositiveDirection;

    // MC_GearIn1 parameters
    Gearin1_Rn      := UINT#10#1;
    Gearin1_Rd      := UINT#10#2;
    Gearin1_Rt      := _eMC_REFERENCE_TYPE#_mcFeedback;
    Gearin1_Acc     := LREAL#4000.0;
    Gearin1_Dec     := LREAL#4000.0;

    // MC_GearIn2 parameters
    Gearin2_Rn      := UINT#10#2;
    Gearin2_Rd      := UINT#10#3;
    Gearin2_Rt      := _eMC_REFERENCE_TYPE#_mcFeedback;
    Gearin2_Acc     := LREAL#4000.0;
    Gearin2_Dec     := LREAL#4000.0;

    // MC_GearOut parameters
    DecRate         := LREAL#200.0;
    Gearout_Dec     := DecRate;

    // Change InitFlag to TRUE after setting the input parameters.
    InitFlag:=TRUE;

END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 1 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis000.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
    Pwr1_En:=TRUE;
ELSE
    Pwr1_En:=FALSE;
END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 2 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis001.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis001.Cfg.NodeAddress]=FALSE) THEN
    Pwr2_En:=TRUE;
ELSE
    Pwr2_En:=FALSE;
END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 3 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis002.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis002.Cfg.NodeAddress]=FALSE) THEN
    Pwr3_En:=TRUE;
ELSE
    Pwr3_En:=FALSE;
END_IF;
```

```
// If a minor fault level error occurs for axis 1 to axis 3, the error handler for the device (FaultHandler) is
executed.
// Implement FaultHandler separately.
IF (MC_Axis000.MFaultLvl.Active=TRUE) OR (MC_Axis001.MFaultLvl.Active=TRUE) OR
(MC_Axis002.MFaultLvl.Active=TRUE) THEN
    FaultHandler();
END_IF;

// If the Servo is ON for axis 1, the MC_MoveVelocity instruction is executed.
IF Pwr1_Status=TRUE THEN
    Vel_Ex := TRUE;
END_IF;

// If InVelocity of MC_MoveVelocity is TRUE and the Servo for axis 2 is ON, MC_GearIn is executed with axis 1
as the master axis and axis 2 as the slave axis.

IF (Vel_InVel=TRUE) AND (Pwr2_Status=TRUE) THEN
    Gearin1_Ex := TRUE;
END_IF;

// If InVelocity of MC_MoveVelocity is TRUE and the Servo for axis 3 is ON, MC_GearIn is executed with axis 1
as the master axis and axis 3 as the slave axis.
IF (Vel_InVel=TRUE) AND (Pwr3_Status=TRUE) THEN
    Gearin2_Ex := TRUE;
END_IF;


// If the actual position of axis 1 is 1000.0 or higher during gear operation for axis 2,
// the GearOut instruction for axis 2 (slave axis) is executed.
IF (Gearin1_Act=TRUE) AND (MC_Axis000.Act.Pos>=LREAL#1000.0) THEN
    Gearout_Ex := TRUE;
END_IF;


// MC_Power for axis 1
PWR1(
    Axis       := MC_Axis000,
    Enable     := Pwr1_En,
    Status     => Pwr1_Status,
    Busy       => Pwr1_Bsy,
    Error      => Pwr1_Err,
    ErrorID    => Pwr1_ErrID
);


// MC_Power for axis 2
PWR2(
    Axis       := MC_Axis001,
    Enable     := Pwr2_En,
    Status     => Pwr2_Status,
    Busy       => Pwr2_Bsy,
    Error      => Pwr2_Err,
    ErrorID    => Pwr2_ErrID
);


// MC_Power for axis 3
PWR3(
    Axis       := MC_Axis002,
    Enable     := Pwr3_En,
    Status     => Pwr3_Status,
    Busy       => Pwr3_Bsy,
    Error      => Pwr3_Err,
    ErrorID    => Pwr3_ErrID
);
```

```
//MC_MoveVelocity
VEL(
    Axis                := MC_Axis000,
    Execute             := Vel_Ex,
    Velocity            := Vel_Vel,
    Acceleration        := Vel_Acc,
    Deceleration        := Vel_Dec,
    Direction           := Vel_Dir,
    InVelocity          => Vel_InVel,
    Busy                => Vel_Bsy,
    Active              => Vel_Act,
    CommandAborted      => Vel_Ca,
    Error               => Vel_Err,
    ErrorID             => Vel_ErrID
);

// MC_GearIn with axis 1 as master axis and axis 2 as slave axis
GEARIN1(
    Master              := MC_Axis000,
    Slave               := MC_Axis001,
    Execute             := Gearin1_Ex,
    RatioNumerator      := Gearin1_Rn,
    RatioDenominator    := Gearin1_Rd,
    ReferenceType       := Gearin1_Rt,
    Acceleration        := Gearin1_Acc,
    Deceleration        := Gearin1_Dec,
    InGear              => Gearin1_InGear,
    Busy                => Gearin1_Bsy,
    Active              => Gearin1_Act,
    CommandAborted      => Gearin1_CA,
    Error               => Gearin1_Err,
    ErrorID             => Gearin1_ErrID
);


// MC_GearIn with axis 1 as master axis and axis 3 as slave axis
GEARIN2(
    Master              := MC_Axis000,
    Slave               := MC_Axis002,
    Execute             := Gearin2_Ex,
    RatioNumerator      := Gearin2_Rn,
    RatioDenominator    := Gearin2_Rd,
    ReferenceType       := Gearin2_Rt,
    Acceleration        := Gearin2_Acc,
    Deceleration        := Gearin2_Dec,
    InGear              => Gearin2_InGear,
    Busy                => Gearin2_Bsy,
    Active              => Gearin2_Act,
    CommandAborted      => Gearin2_CA,
    Error               => Gearin2_Err,
    ErrorID             => Gearin2_ErrID
);

//MC_GearOut
GEAROUT(
    Slave               := MC_Axis001,
    Execute             := Gearout_Ex,
    Deceleration        := Gearout_Dec,
    Done                => Gearout_D,
    Busy                => Gearout_Bsy,
    CommandAborted      => Gearout_CA,
    Error               => Gearout_Err,
    ErrorID             => Gearout_ErrID
);
```

# MC_GearInPos

The MC_GearInPos instruction performs electronic gear operation for the specified gear ratio between the master axis and the slave axis. The positions at which to start synchronizing the master axis and slave axis are specified.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_GearInPOS | Positioning Gear Operation | FB | MC_GearInPos_instance<br>MC_GearInPos<br>Master — — Master<br>Slave — — Slave<br>Execute StartSync<br>RatioNumerator InSync<br>RatioDenominator Busy<br>ReferenceType Active<br>MasterSyncPosition CommandAborted<br>SlaveSyncPosition Error<br>Velocity ErrorID<br>Acceleration<br>Deceleration<br>Jerk<br>BufferMode | MC_GearInPos_instance (<br>Master :=*parameter,*<br>Slave :=*parameter,*<br>Execute :=*parameter,*<br>RatioNumerator :=*parameter,*<br>RatioDenominator :=*parameter,*<br>ReferenceType :=*parameter,*<br>MasterSyncPosition :=*parameter,*<br>SlaveSyncPosition :=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>BufferMode :=*parameter,*<br>StartSync =>*parameter,*<br>InSync =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Ratio Numerator | Gear Ratio Numerator | UNIT | Positive number | 10,000 | Specify the numerator of the electronic gear ratio between the master and slave axes. |
| Ratio Denominator | Gear Ratio Denominator | UNIT | Positive number | 10,000 | Specify the denominator of the electronic gear ratio between the master and slave axes. |
| Reference Type[*1] | Position Type Selection | _eMC_REFERENCE_TYPE | 0: _mcCommand<br>1: _mcFeedback<br>2: _mcLatestCommand | 0[*2] | Specify the position type.<br>0: Command position (value calculated in the previous primary period)<br>1: Actual position (value obtained in the same primary period)<br>2: Command position (value calculated in the same primary period) |
| MasterSync Position | Master Sync Start Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the absolute master sync position. The unit is command units.[*3] |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| SlaveSync Position | Slave Sync Start Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the absolute slave sync position. The unit is command units.*3 |
| Velocity | Target Velocity | LREAL | Positive number | 0 | Specify the target velocity. Always set the target velocity. If the axis is moved without setting a target velocity, an error will occur. The unit is command units/s.*3 |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/s$^2$.*3 |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/s$^2$.*3 |
| Jerk (Reserved) | Jerk | LREAL | 0 | 0 | (Reserved) |
| BufferMode (Reserved) | Buffer Mode Selection | _eMC_ BUFFER_ MODE | 0: _mcAborting | 0*2 | (Reserved) |

*1 When you use *_mcLatestCommand*, the axis number set for the *Master* (Master Axis) in the system-defined variable for motion control must be lower than the axis number set for the *Slave* (Slave Axis) in the system-defined variable for motion control .

*2 The default value for an enumeration variable is actually not the number, but the enumerator.

*3 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| StartSync | Following | BOOL | TRUE or FALSE | TRUE when acceleration/deceleration is started for synchronization. |
| InSync | In Sync | BOOL | TRUE or FALSE | TRUE when the slave axis reaches the slave sync position. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| StartSync | When the axis starts moving. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| InSync | When the slave axis reaches *SlaveSyncPosition*. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Busy | When *Execute* changes to TRUE. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When an instruction is received. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution.<br>• When the MC_GearOut instruction is executed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Master | Master Axis | _sAXIS_REF | --- | Specify the master axis.* |
| Slave | Slave Axis | _sAXIS_REF | --- | Specify the slave axis.* |

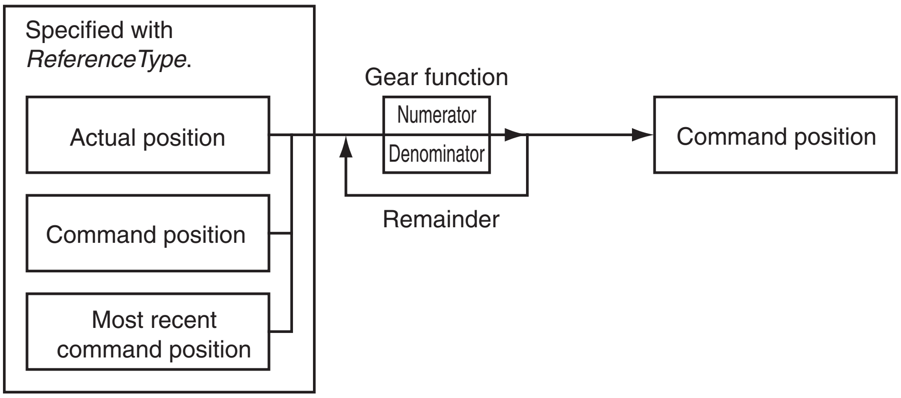
* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

**Precautions for Correct Use**

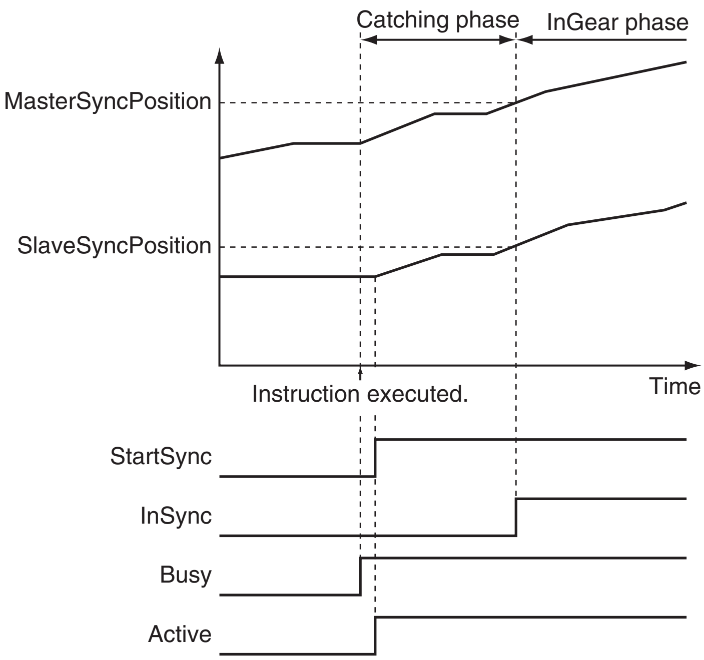
If you specify the same axis for the master axis and slave axis, a Master and Slave Defined as Same Axis minor fault (error code 5436 hex) will occur.

## Function

- The MC_GearInPos instruction performs gear operation for the slave axis specified with *Slave*. The following parameters are also specified: *RatioNumerator* (Gear Ratio Numerator), *RatioDenominator* (Gear Ratio Denominator), *ReferenceType* (Position Type), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate).
- For the master axis, you can specify the command position, actual position, or most recent command position.



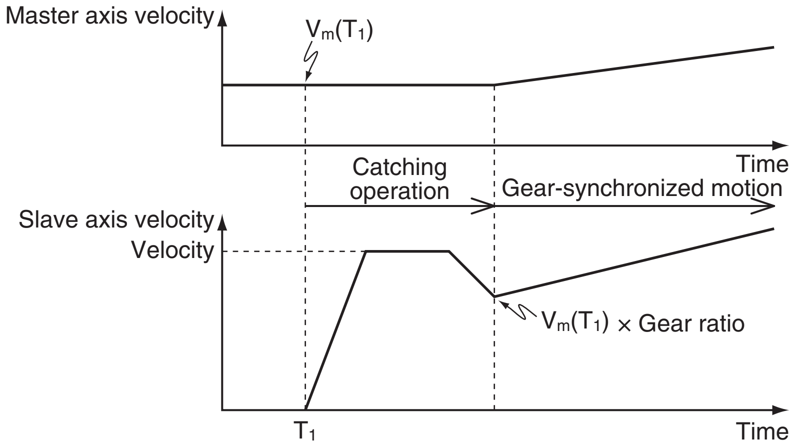

- After operation starts, the *Slave* (Slave Axis) accelerates and decelerates in sync with the *Master* (Master Axis) in the catching operation.
- Catching phase exists until the slave axis reaches the slave sync position. For either, the position is synchronized with the master axis.
- The *Velocity* (Target Velocity) input variable is the target velocity for the catching phase.
- The slave axis moves in the same direction as the master axis when operation is started.
  An error occurs if the master axis velocity is 0 when started. If the master axis is moving in the positive direction and SlaveSyncPosition (Slave Sync Start Position) is smaller than the position of the slave axis when the instruction was executed, the slave axis will reverse direction. If the master axis is moving in the negative direction and *SlaveSyncPosition* (Slave Sync Start Position) is larger than the position of the slave axis when the instruction was executed, the slave axis will also reverse direction. For either, the position is synchronized with the master axis.
- If the master axis velocity changes significantly between periods, the slave axis velocity will not be constant.



**Precautions for Safe Use**

Do not execute the MC_SetPosition instruction on the *Master* (Master Axis) if you use this instruction.
If the MC_SetPosition instruction is executed for the *Master* (Master Axis), the *Slave* (Slave Axis) may follow the *Master* (Master Axis) quickly.
If you want to use the MC_SetPosition instruction for the *Master* (Master Axis), disable the relationship between the *Master* (Master Axis) and *Slave* (Slave Axis) before executing the instruction.

- The MC Function Module calculates the velocity profile for linear acceleration and deceleration with the following three velocities using the *Acceleration* (Acceleration Rate) and *Deceleration* (Deceleration Rate).
  - The velocity of the *Slave* (Slave Axis) when catching operation starts is the initial velocity.
  - The velocity of the *Master* (Master Axis) when catching operation starts multiplied by the gear ratio is the final velocity.
  - The *Velocity* (Target Velocity) is the maximum velocity.




For the *Slave* (Slave Axis) to catch up with the *Master* (Master Axis) for the *MasterSyncPosition* (Master Sync Start Position) and *SlaveSyncPosition* (Slave Sync Start Position), the following condition must be met for the *Velocity* (Target Velocity)

$$\text{Velocity} > \frac{\text{Master axis velocity when MC\_GearInPos is executed} \times \text{Gear ratio numerator}}{\text{Gear ratio denominator}}$$

The information that is used as the mater axis velocity depends on the setting of ReferenceType (Position Type Selection).

When *_mcCommand* or *_mcLatestCommand* is set: Use the command current velocity.
When *_mcFeedback* is set: Use the actual current position.

If the *Slave* (Slave Axis) cannot catch up with the *Master* (Master Axis) for the *MasterSyncPosition* (Master Sync Start Position) and *SlaveSyncPosition* (Slave Sync Start Position), a Positioning Gear Operation Insufficient Target Velocity error (error code 5447 hex) will occur. Also, if you set the Count Mode to Rotary Mode, make sure that the synchronous operation starts within one cycle of the ring counter.

**Precautions for Correct Use**

Refer to *Precautions for Master and Auxiliary Axes in Synchronized Control* on page 1-7 for precautions on the master axis.

## ● *ReferenceType* (Position Type Selection)

You can select one of the following position types.

- *_mcCommand*: Command position (value calculated in the previous primary period)
  The master axis command position that was calculated in the previous primary period is used for the current period.
  The command value that was calculated for the master axis in the last primary period is used to calculate the command position of the slave axis in the current period.
- *_mcFeedback*: Value obtained in the same primary period
  The actual position of the master axis that was obtained in the same primary period is used.
- *_mcLatestCommand*: Command position (value calculated in the same primary period)
  The command position of the master axis that was calculated in the same primary period is used. This enables the use of information that is more recent than for *_mcCommand*. However, the axis number of the master axis must be set lower than the axis number of the slave axis.
  If the axis number of the slave axis is lower than the axis number of the master axis, *Error* will change to TRUE. A Master/Slave Axis Numbers Not in Ascending Order error (error code: 5438 hex) will be output to *ErrorID*.

**Additional Information**

The command position that is calculated in the same primary period enables greater precision in synchronization than the command position that was calculated in the previous primary period. However, the axis number set for the *Master* (Master Axis) in the system-defined variable for motion control must be lower than the axis number set for the *Slave* (Slave Axis) in the system-defined variable for motion control.

## ● Relationship between Axis Types and Position Types

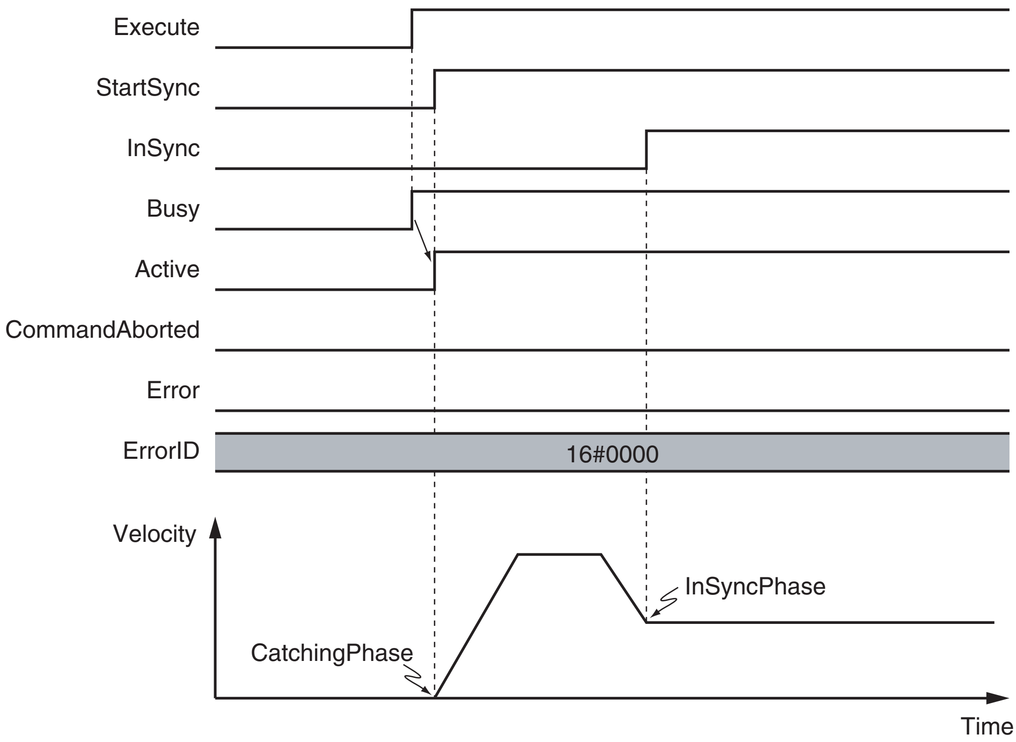
The relationship between the axis types that you can monitor and position types that is monitored is shown below.

| Axis Type | ReferenceType | |
|---|---|---|
| | _mcCommand or _mcLatestCommand | _mcFeedback |
| Servo axis | OK | OK |
| Encoder axis | No* | OK |
| Virtual servo axis | OK | OK |
| Virtual encoder axis | No* | OK |

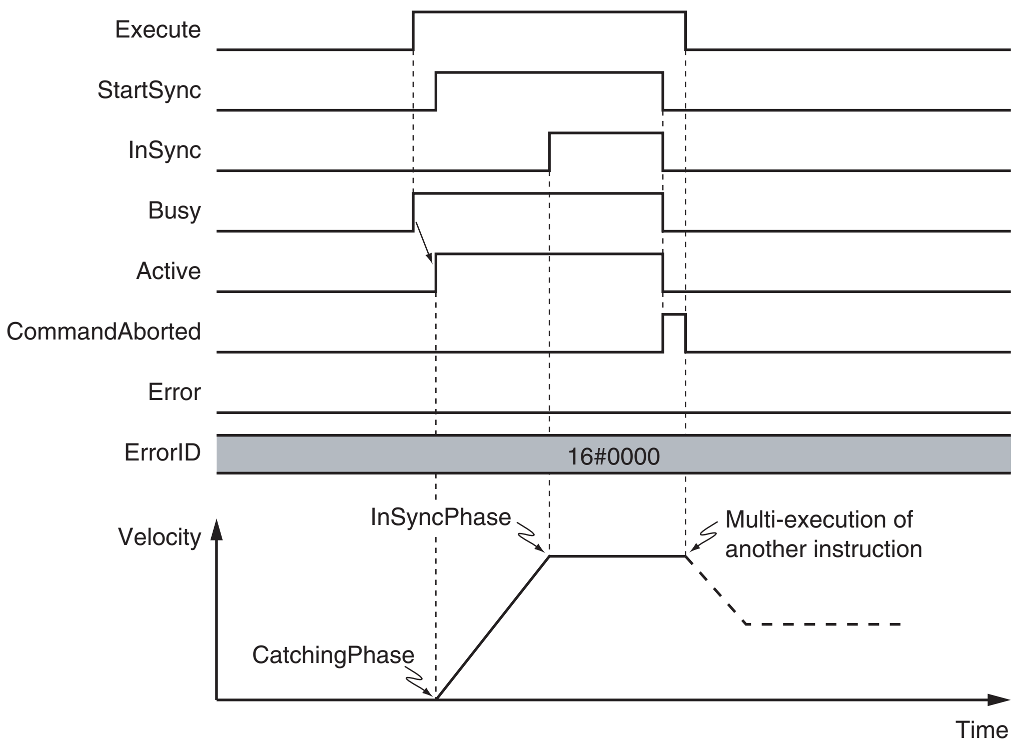
* A Position Type Selection Out of Range error (error code 5430 hex) occurs when the instruction is executed.

## Timing Charts

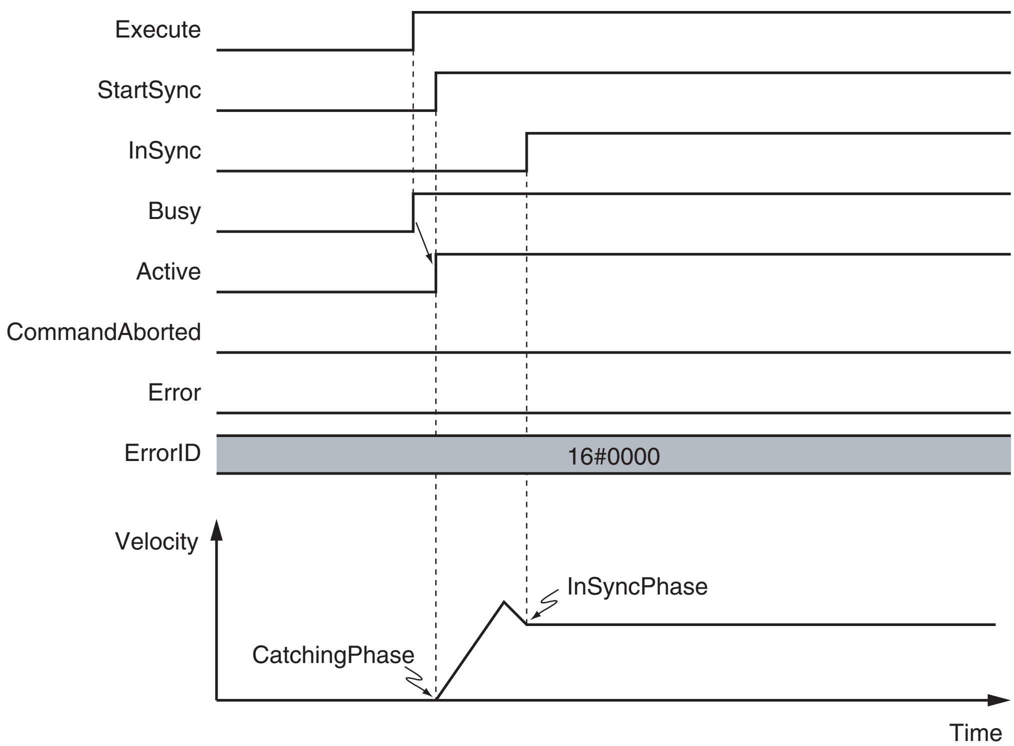
- Electronic gear operation starts when *Execute* changes to TRUE.
- *Busy* (Executing) changes to TRUE when *Execute* changes to TRUE. After the operation is started, *Active* (Controlling) and *StartSync* (Following) change to TRUE and the *Slave* (Slave Axis) starts the gear operation.
- When the *MasterSyncPosition* (Master Sync Start Position) and *SlaveSyncPosition* (Slave Sync Start Position) are reached, *InSync* changes to TRUE.
- If another instruction aborts this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing), *Active* (Controlling), *StartSync* (Following), and *InSync* change to FALSE.



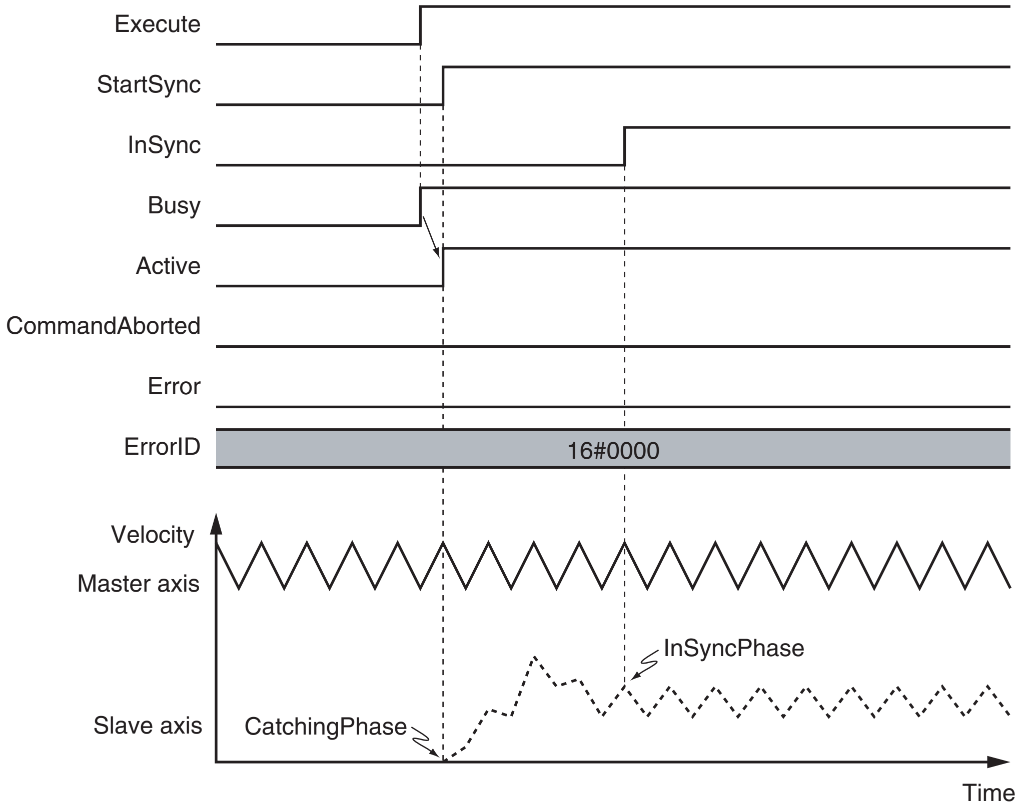

The operation when this instruction is aborted by another instruction is shown below.



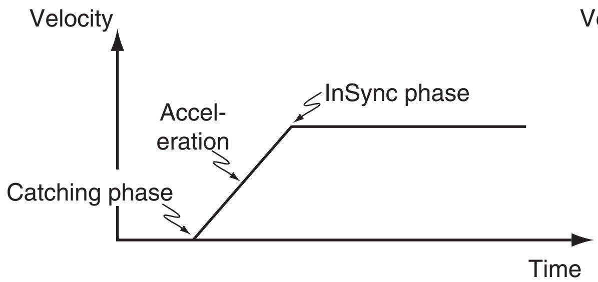

Depending on setting of the *SlaveSyncPosition*, the axis may not reach the target velocity. An example of this is shown below.

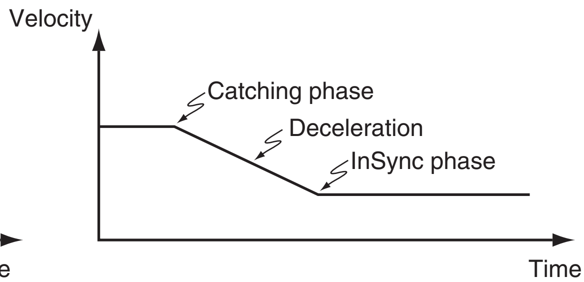


The slave axis follows the master axis position before the InSync phase as well. An example of this is shown below.




You can specify the *Acceleration* (Acceleration Rate) and *Deceleration* (Deceleration Rate) as input variables. The following figures show operation examples of the electronic gear.




## Re-execution of Motion Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Instructions

The axis command status of the master axis, including whether it is stopped due to an error or it is decelerating to a stop, does not affect the execution of this instruction.
For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution of Other Instructions during Instruction Execution

Specify the operation of this instruction by using *BufferMode* (Buffer Mode Selection) for multi-execution of instructions.

| Buffer Mode Selection | Description |
|---|---|
| Aborting | Aborts the instruction being executed and switches to this instruction.<br>If the direction of axis motion is reversed by switching instructions, the motion will be reversed according to the Operation at Reversing axis parameter. |

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

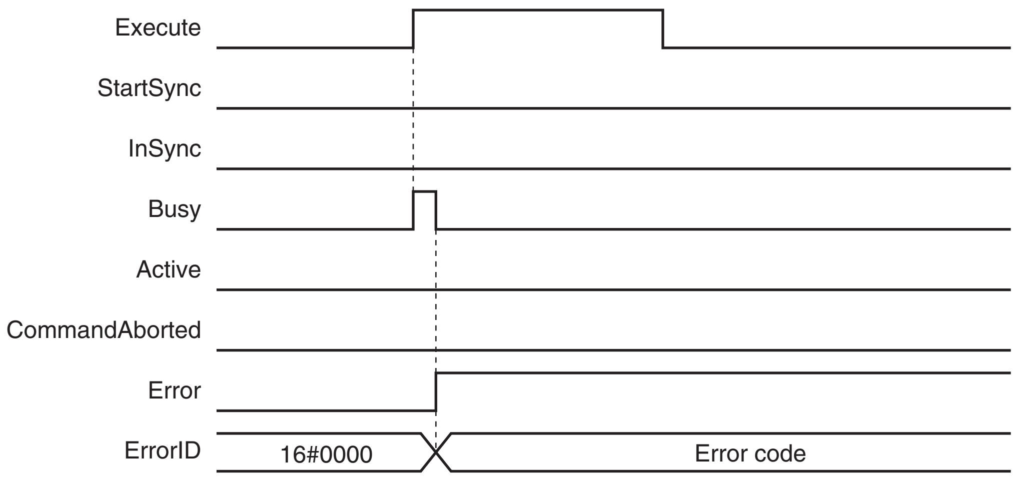
**Precautions for Correct Use**

If another instruction is executed with BufferMode set to anything other than *Aborting*, an error will occur in the other instruction.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

### ● Timing Chart When Error Occurs




The slave axis is not affected by the error status of the master axis during synchronized control. After the error status of the master axis is cleared, the slave axis continues electronic gear operation when the master axis operates.
The master axis is not affected if an error occurs for the slave axis during startup or execution of this instruction.

### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# Sample Programming

This section describes sample programming where the sync position for the master axis is *Pos1* and the sync position for the slave axis *Pos2*.

## Parameter Settings

The minimum settings required for this sample programming are given below.

### ● Setting Axis Parameters

#### Axis Types

| Axis | Axis Types |
|---|---|
| Axis 1 | Servo axis (master axis) |
| Axis 2 | Servo axis (slave axis) |

#### Count Modes

| Axis | Count Mode |
|---|---|
| Axis 1 | Rotary Mode |
| Axis 2 | Rotary Mode |

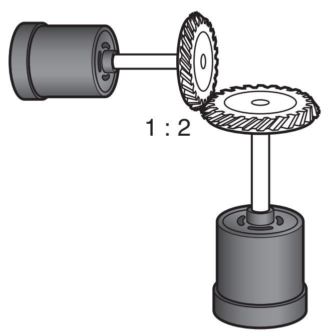
#### Ring Counters

| Axis | Modulo maximum position | Modulo minimum position |
|---|---|---|
| Axis 1 | 360 | 0 |
| Axis 2 | 360 | 0 |

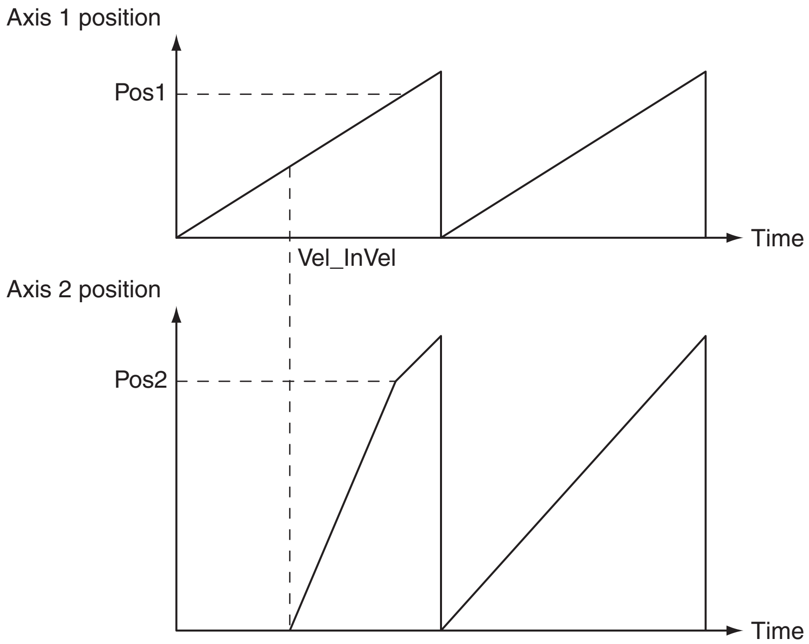
#### Units of Display

| Axis | Unit of Display |
|---|---|
| Axis 1 | degree |
| Axis 2 | degree |

## Operation Example



## ● Operation Pattern




**1** Starting the Master Axis

The master axis (axis 1) is an actual servo axis and it is operated with velocity control.

**2** Reaching Target Velocity for Main Axis

When the command velocity of the master axis reaches the target velocity, *InVelocity* (Target Velocity Reached) of the master axis changes to TRUE.

**3** Executing the Slave Axes

When *InVelocity* of the master axis changes to TRUE, slave axis (axis 2) performs gear operation with a gear ratio of 1:2 against the actual position of the master axis. The synchronized positions are *Pos1* for the master axis and *Pos2* for the slave axis.

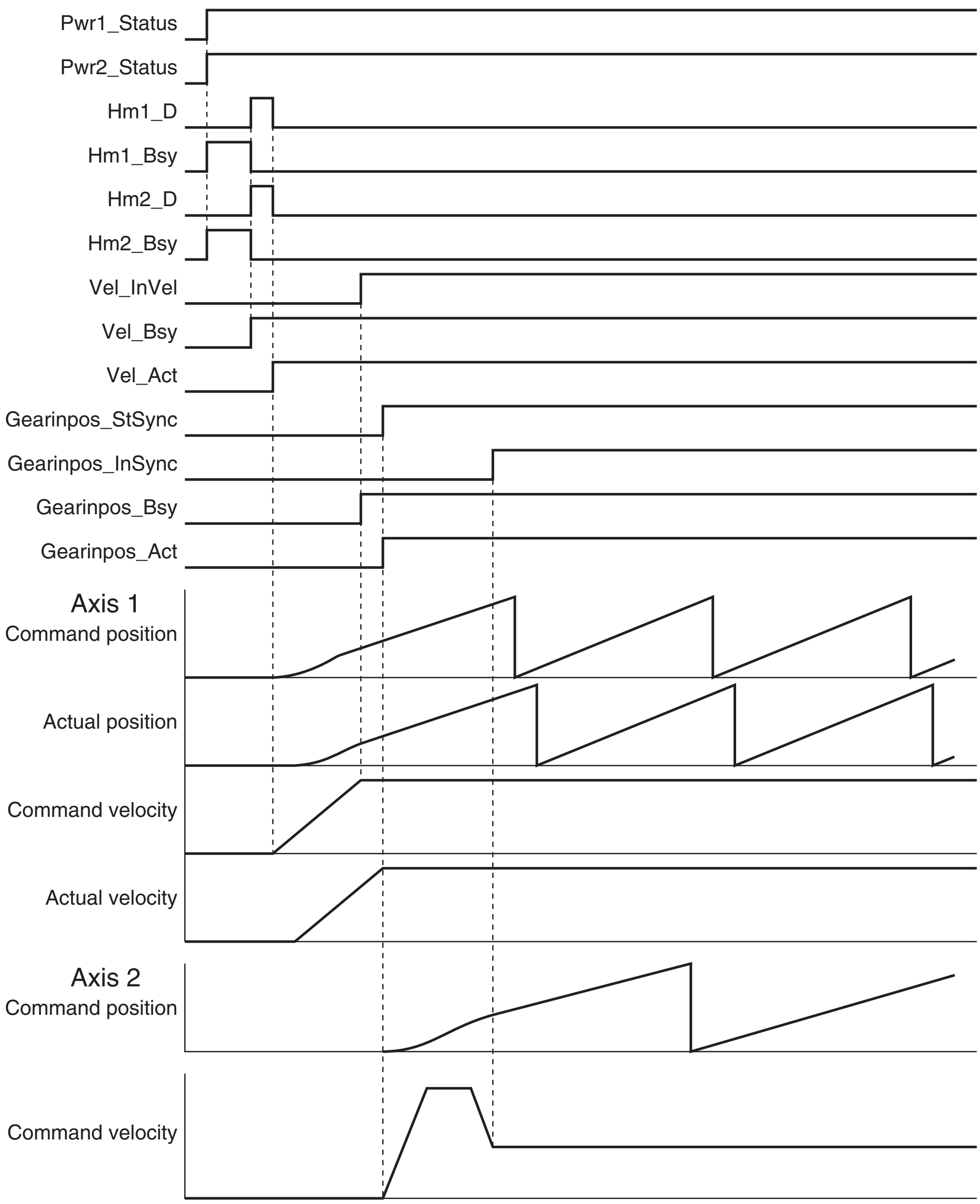
# Ladder Diagram

## ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for the master axis, axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 2. |
| MC_Axis001.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 2. |
| MC_Axis001.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 2. |
| Pwr1_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR1 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Pwr2_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR2 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |

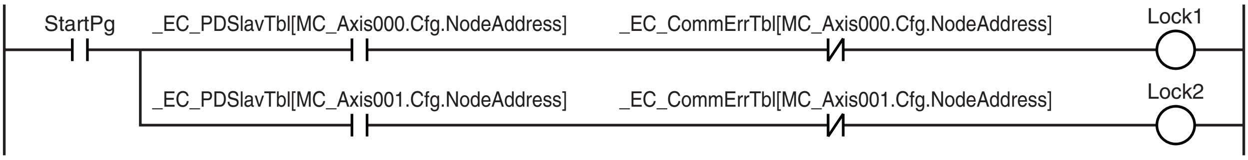
| Name | Data type | Default | Comment |
|---|---|---|---|
| Hm1_D | BOOL | FALSE | This variable is assigned to the *Done* output variable from the HM1 instance of the MC_Home instruction. |
| Hm2_D | BOOL | FALSE | This variable is assigned to the *Done* output variable from the HM2 instance of the MC_Home instruction. |
| Vel_InVel | BOOL | FALSE | This variable is assigned to the *InVelocity* output variable from the VEL instance of the MC_MoveVelocity instruction. It is TRUE when the target velocity is reached. |
| Pos1 | LREAL | --- | This variable gives the master axis sync position. |
| Pos2 | LREAL | --- | This variable gives the slave axis sync position. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |

### ● Timing Chart

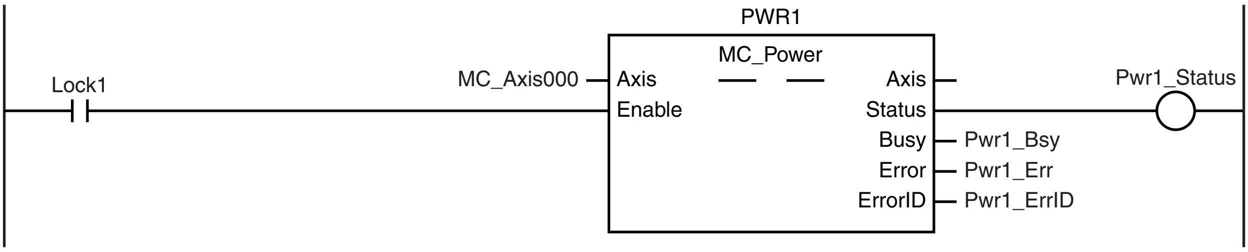


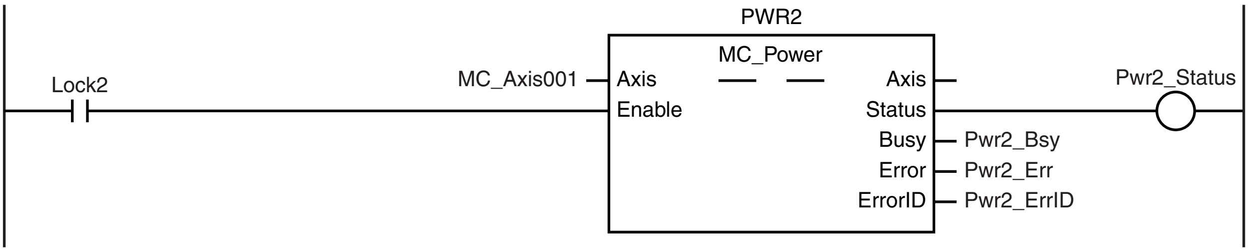
## ● Sample Programming

If *StartPg* is TRUE, EtherCAT communications for the axes are checked to see if process data communications are normal.

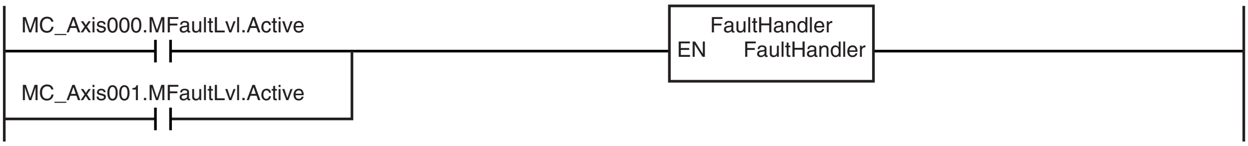



If process data communications are active, the Servo is turned ON for the axes.

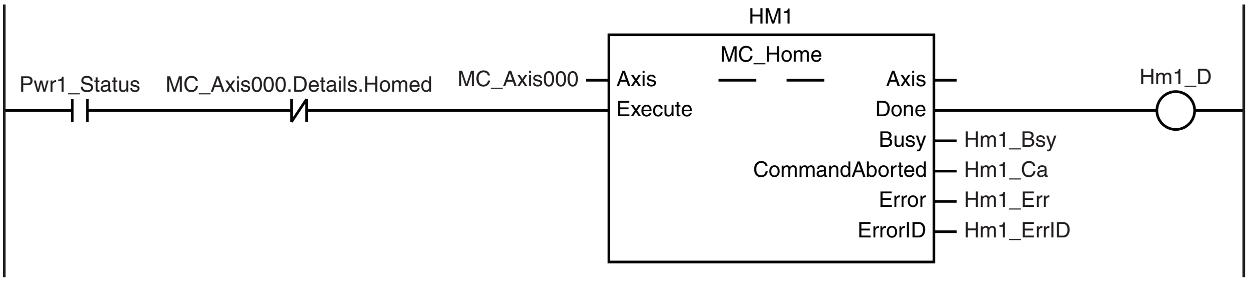


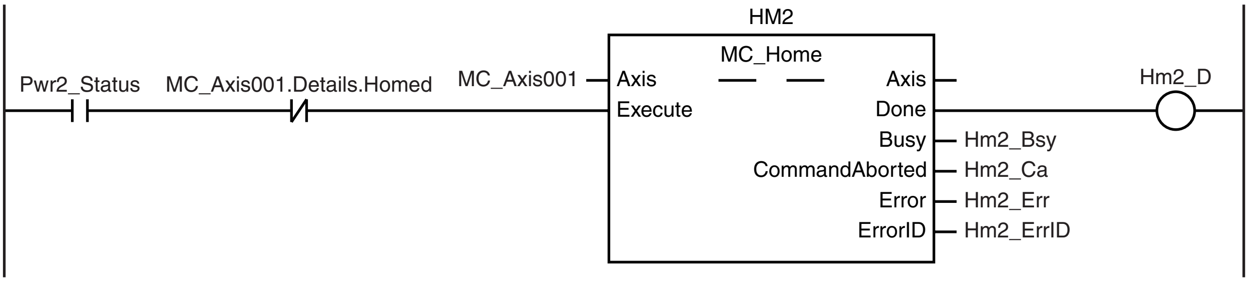

If there is a minor fault level error for a composition axis, the error handler for the device (FaultHandler) is executed. Design FaultHandler separately.



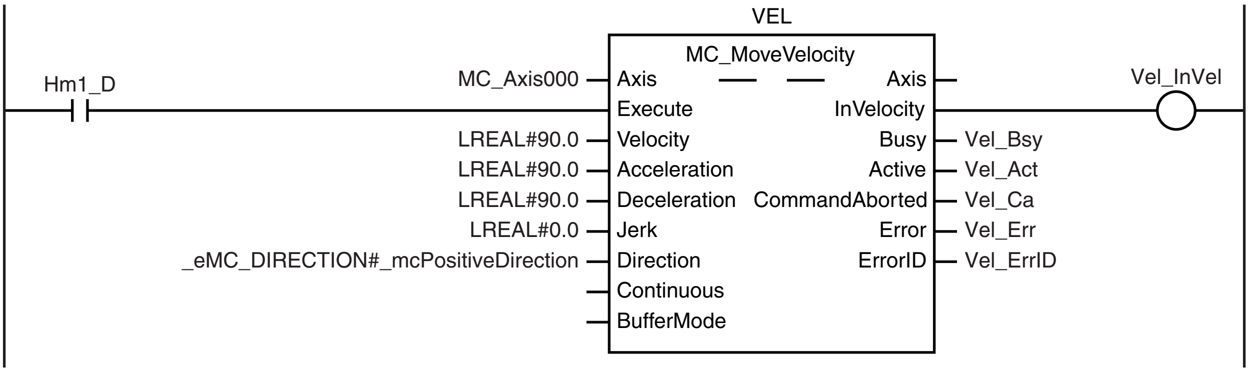

If the Servo is ON for the master axis (axis 1) and home is not defined, the Home instruction is executed to define home.



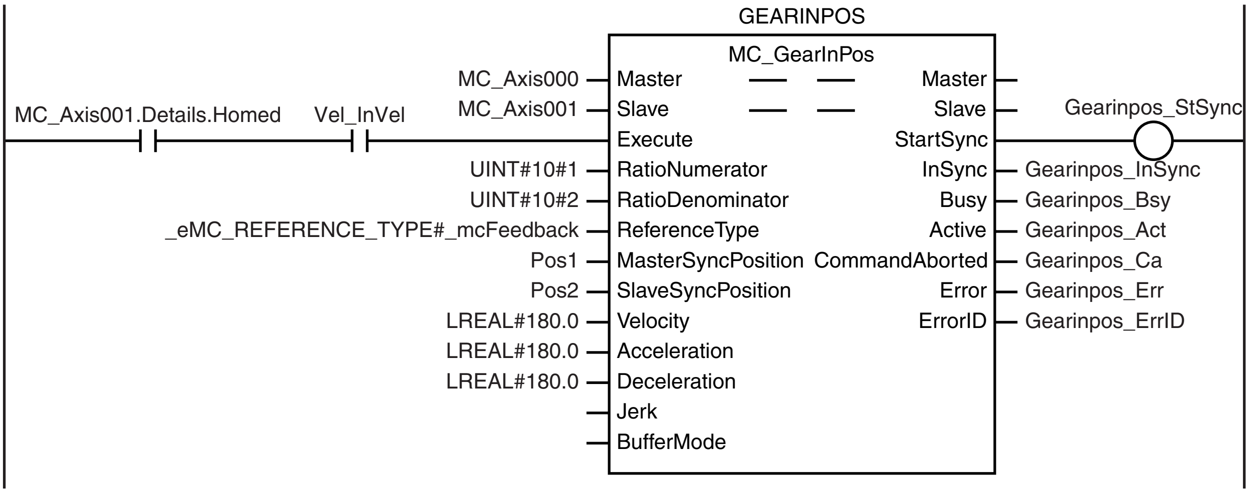

If the Servo is ON for the slave axis (axis 2) and home is not defined, the Home instruction is executed to define home.



The MC_MoveVelocity (Velocity Control) instruction is executed after homing is completed for the master axis (axis 1).




After homing is completed for axis 2 (slave axis), MC_GearInPos (Positioning Gear Operation) is executed to start gear operation if *Vel_InVel* of MC_MoveVelocity is TRUE.



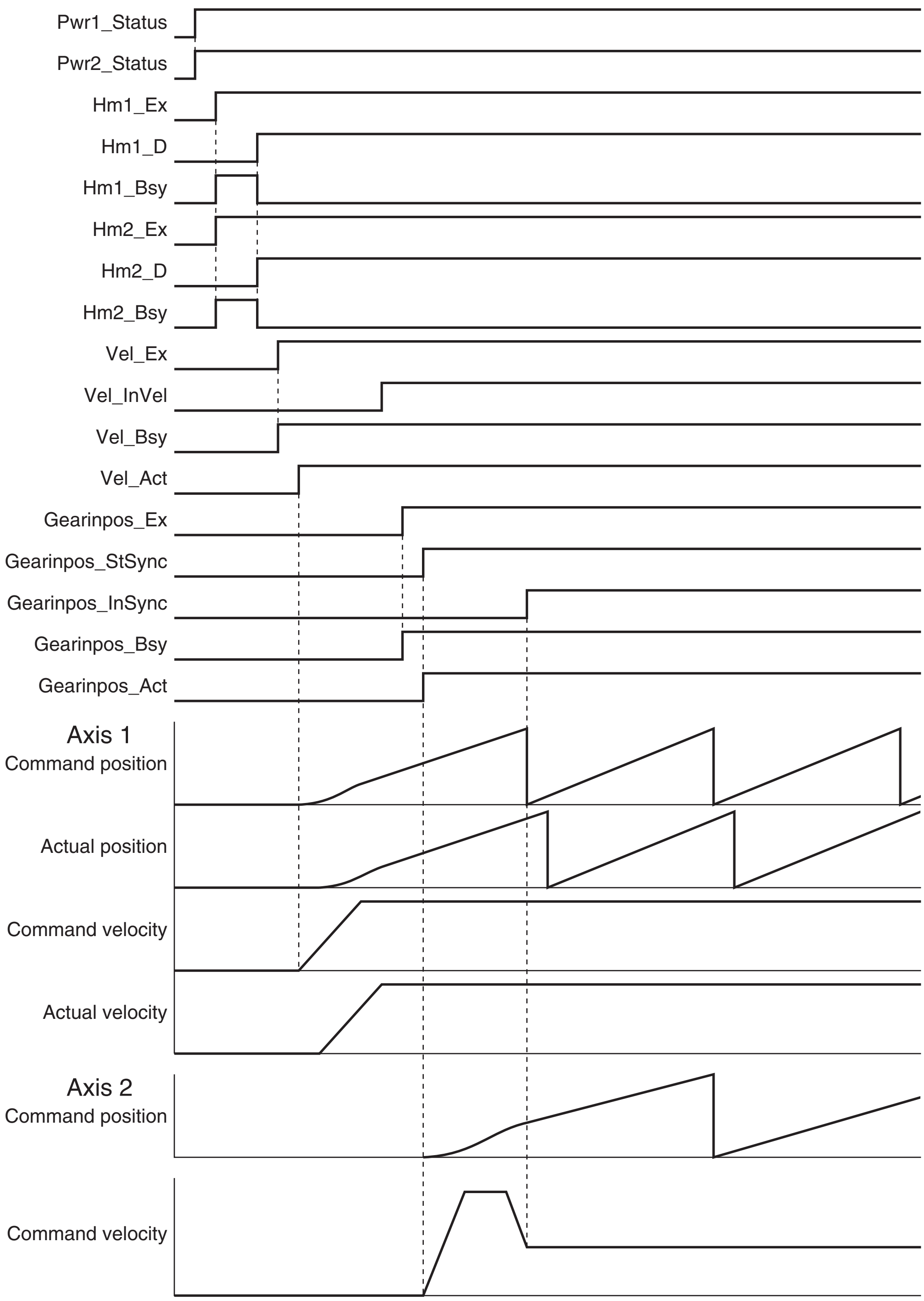

## Structured Text (ST)

### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for the master axis, axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 2. |
| MC_Axis001.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 2. |
| MC_Axis001.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 2. |
| Pwr1_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR1 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Pwr2_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR2 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Hm1_D | BOOL | FALSE | This variable is assigned to the *Done* output variable from the HM1 instance of the MC_Home instruction. |
| Hm2_D | BOOL | FALSE | This variable is assigned to the *Done* output variable from the HM2 instance of the MC_Home instruction. |

| Name | Data type | Default | Comment |
|---|---|---|---|
| Vel_InVel | BOOL | FALSE | This variable is assigned to the *InVelocity* output variable from the VEL instance of the MC_MoveVelocity instruction. It is TRUE when the target velocity is reached. |
| Pos1 | LREAL | --- | This variable gives the master axis sync position. |
| Pos2 | LREAL | --- | This variable gives the slave axis sync position. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| Gearinpos_Ex | BOOL | FALSE | The GEARIN1 instance of MC_GearInPos is executed when this variable changes to TRUE. |
| Vel_Ex | BOOL | FALSE | The VEL instance of MC_MoveVelocity is executed when this variable changes to TRUE. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

## ● Timing Chart




## ● Sample Programming

```
// Processing when input parameters are not set
IF InitFlag = FALSE THEN
    // MC_MoveVelocity parameters
    Vel_Vel     := LREAL#90.0;
    Vel_Acc     := LREAL#90.0;
    Vel_Dec     := LREAL#90.0;
    Vel_Dir     := _eMC_DIRECTION#_mcPositiveDirection;

    // MC_GearInPos parameters
    Pos1                := LREAL#300.0;
    Pos2                := LREAL#200.0;
    Gearinpos_Rn        := UINT#10#1;
```

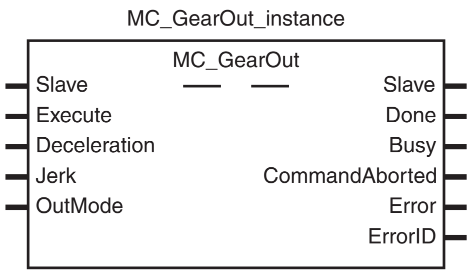
```
    Gearinpos_Rd        := UINT#10#2;
    Gearinpos_Rt        := _eMC_REFERENCE_TYPE#_mcFeedback;
    Gearinpos_Mtpos     := Pos1;
    Gearinpos_Svpos     := Pos2;
    Gearinpos_Vel       := LREAL#180.0;
    Gearinpos_Acc       := LREAL#180.0;
    Gearinpos_Dec       := LREAL#180.0;

    // Change InitFlag to TRUE after setting the input parameters.
    InitFlag:=TRUE;

END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 1 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis000.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
    Pwr1_En:=TRUE;
ELSE
    Pwr1_En:=FALSE;
END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 2 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis001.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis001.Cfg.NodeAddress]=FALSE) THEN
    Pwr2_En:=TRUE;
ELSE
    Pwr2_En:=FALSE;
END_IF;


// If a minor fault level error occurs for axis 1 or axis 2, the error handler for the device (FaultHandler) is
executed.
// Implement FaultHandler separately.
IF (MC_Axis000.MFaultLvl.Active=TRUE) OR (MC_Axis001.MFaultLvl.Active=TRUE) THEN
    FaultHandler();
END_IF;

// If the Servo is ON for axis 1 and home is not defined, the Home instruction is executed.
IF (Pwr1_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
    Hm1_Ex:=TRUE;
END_IF;

// If the Servo is ON for axis 2 and home is not defined, the Home instruction is executed.
IF (Pwr2_Status=TRUE) AND (MC_Axis001.Details.Homed=FALSE) THEN
    Hm2_Ex:=TRUE;
END_IF;

// After homing is completed for axis 1, MC_MoveVelocity is executed.
IF Hm1_D=TRUE THEN
    Vel_Ex := TRUE;
END_IF;

// After homing is completed for axis 2, MC_GearInPos is executed when Vel_InVel of MC_MoveVelocity is
TRUE.
IF (MC_Axis001.Details.Homed=TRUE) AND (Vel_InVel=TRUE) THEN
    Gearinpos_Ex := TRUE;
END_IF;


// MC_Power for axis 1
PWR1(
```

```
    Axis        := MC_Axis000,
    Enable      := Pwr1_En,
    Status      => Pwr1_Status,
    Busy        => Pwr1_Bsy,
    Error       => Pwr1_Err,
    ErrorID     => Pwr1_ErrID
);
// MC_Power for axis 2
PWR2(
    Axis        := MC_Axis001,
    Enable      := Pwr2_En,
    Status      => Pwr2_Status,
    Busy        => Pwr2_Bsy,
    Error       => Pwr2_Err,
    ErrorID     => Pwr2_ErrID
);

// MC_Home for axis 1
HM1(
    Axis                := MC_Axis000,
    Execute             := Hm1_Ex,
    Done                => Hm1_D,
    Busy                => Hm1_Bsy,
    CommandAborted      => Hm1_Ca,
    Error               => Hm1_Err,
    ErrorID             => Hm1_ErrID
);
// MC_Home for axis 2
HM2(
    Axis                := MC_Axis001,
    Execute             := Hm2_Ex,
    Done                => Hm2_D,
    Busy                => Hm2_Bsy,
    CommandAborted      => Hm2_Ca,
    Error               => Hm2_Err,
    ErrorID             => Hm2_ErrID
);

//MC_MoveVelocity
VEL(
    Axis                := MC_Axis000,
    Execute             := Vel_Ex,
    Velocity            := Vel_Vel,
    Acceleration        := Vel_Acc,
    Deceleration        := Vel_Dec,
    Direction           := Vel_Dir,
    InVelocity          => Vel_InVel,
    Busy                => Vel_Bsy,
    Active              => Vel_Act,
    CommandAborted      => Vel_Ca,
    Error               => Vel_Err,
    ErrorID             => Vel_ErrID
);

//MC_GearInPos
GEARINPOS(
    Master                  := MC_Axis000,
    Slave                   := MC_Axis001,
    Execute                 := Gearinpos_Ex,
    RatioNumerator          := Gearinpos_Rn,
    RatioDenominator        := Gearinpos_Rd,
    ReferenceType           := Gearinpos_Rt,
    MasterSyncPosition      := Gearinpos_Mtpos,
    SlaveSyncPosition       := Gearinpos_Svpos,
    Velocity                := Gearinpos_Vel,
```

```
    Acceleration        := Gearinpos_Acc,
    Deceleration        := Gearinpos_Dec,
    StartSync           => Gearinpos_StSync,
    InSync              => Gearinpos_InSync,
    Busy                => Gearinpos_Bsy,
    Active              => Gearinpos_Act,
    CommandAborted      => Gearinpos_Ca,
    Error               => Gearinpos_Err,
    ErrorID             => Gearinpos_ErrID
);
```

# MC_GearOut

The MC_GearOut instruction stops operation for the MC_GearIn (Start Gear Operation) instruction or MC_GearInPos (Positioning Gear Operation) instruction.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_GearOut | End Gear Operation | FB |  | MC_GearOut_instance (<br>Slave :=*parameter,*<br>Execute :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>OutMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/s$^2$.*1 |
| Jerk (Reserved) | Jerk | LREAL | 0 | 0 | (Reserved) |
| OutMode (Reserved) | Sync End Mode Selection | _eMC_OUT_MODE | 0: _mcStop | 0*2 | (Reserved) |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 The default value for an enumeration variable is actually not the number, but the enumerator.

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the instruction is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Slave | Slave Axis | _sAXIS_REF | --- | Specify the slave axis.* |

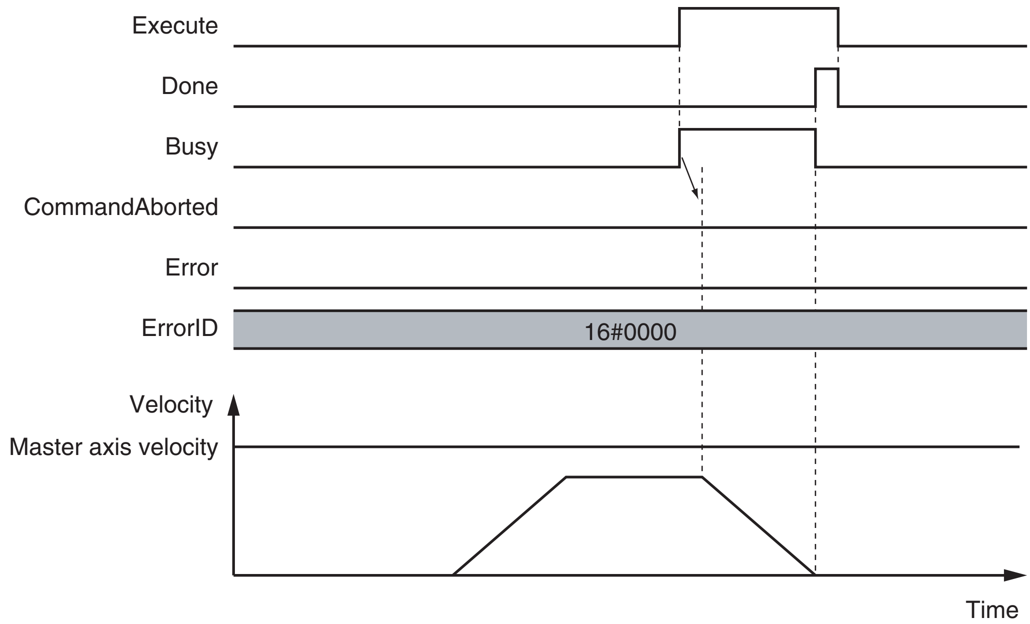
* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

## Function

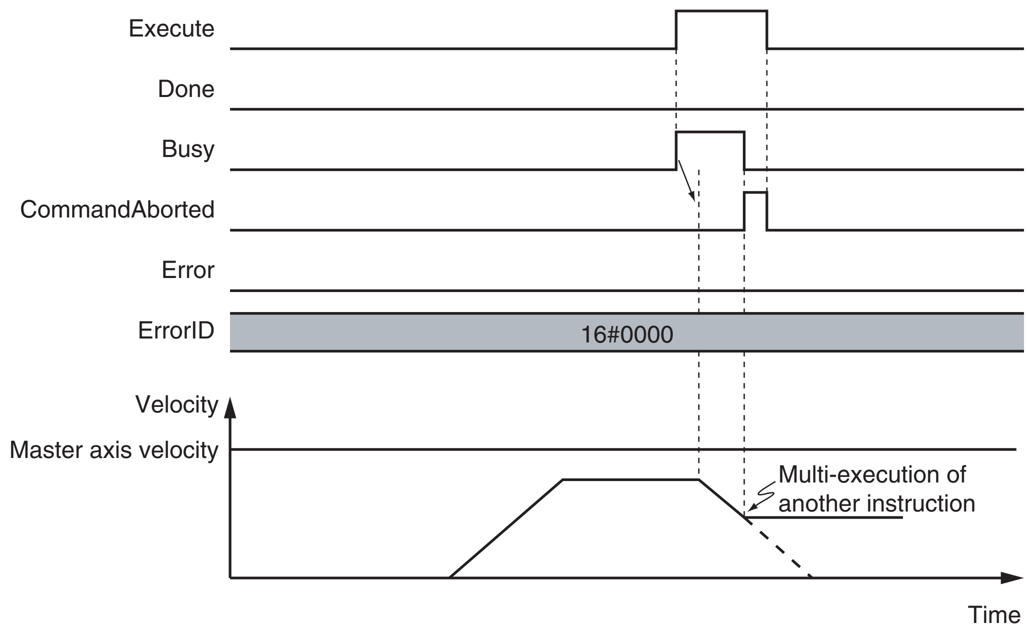
- The MC_GearOut instruction stops the operation of the MC_GearIn (Start Gear Operation) or MC_GearInPos (Positioning Gear Operation) instruction for the operation axis specified with *Slave* and at the specified *Deceleration* (Deceleration Rate).
- This instruction does not affect the MC_GearIn (Start Gear Operation) or MC_GearInPos (Positioning Gear Operation) operation of the master axis.

### Timing Charts

- *Busy* (Executing) changes to TRUE when *Execute* changes to TRUE.
- *Done* changes to TRUE when the target velocity is reached.
- If another instruction aborts this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing) changes to FALSE.

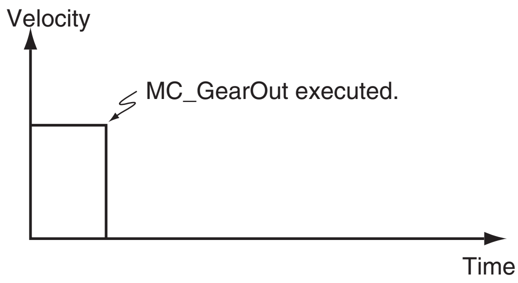



#### ● When This Instruction Is Aborted by Another Instruction



### When The Instruction Is Executed with a Deceleration Rate of 0

If deceleration rate is set to 0 and the instruction is executed, the axis will stop without decelerating.
The following chart shows an operation example of when the deceleration rate is 0.




## Re-execution of Motion Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Instructions

For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### Execution during Execution of Other Instructions

If you execute this instruction while MC_GearIn (Start Gear Operation) or MC_GearInPos (Positioning Gear Operation) instruction is in execution, *CommandAborted* for MC_GearIn or MC_GearInPos will change to TRUE and *Busy* (Executing) of this instruction will change to TRUE. If you execute this instruction while instructions other than MC_GearIn or MC_GearInPos are in execution, this instruction will result in an error.

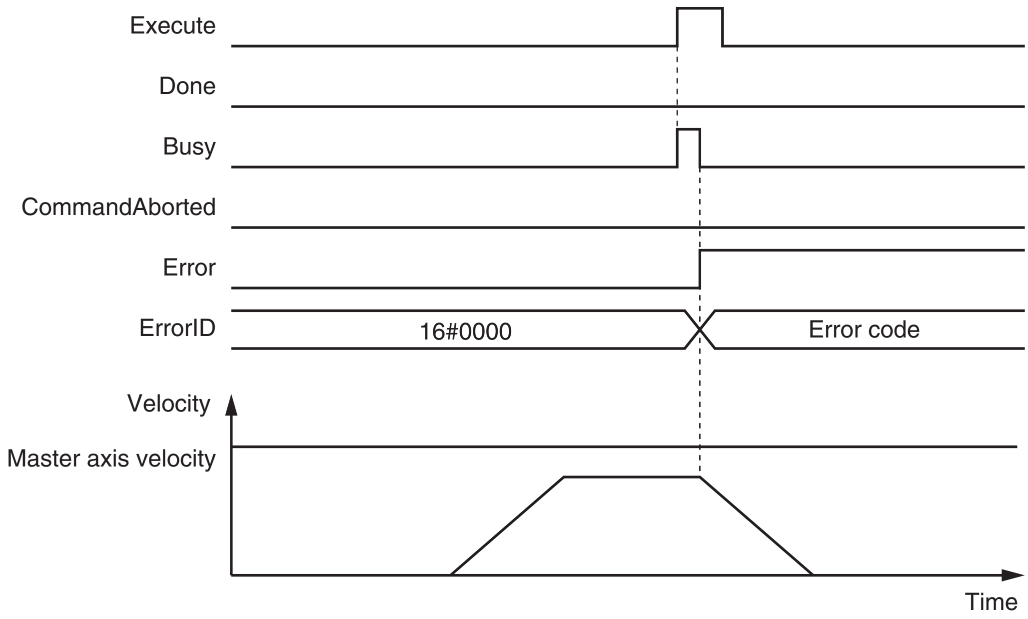
### Execution of Other Instructions during Instruction Execution

To use multi-execution of motion instruction for this instruction, specify the slave axis.
If you execute another instruction during execution of this instruction, you can specify either aborting or buffering.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

### ● Timing Chart When Error Occurs

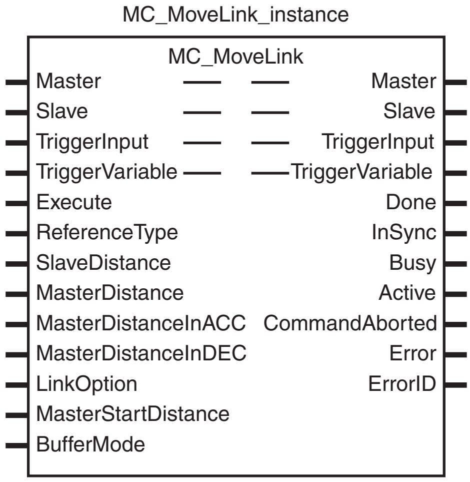



### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_MoveLink

Positioning is performed in sync with the specified master axis.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_MoveLink | Synchronous Positioning | FB |  | MC_MoveLink_instance (<br>Master :=*parameter,*<br>Slave :=*parameter,*<br>TriggerInput :=*parameter,*<br>TriggerVariable :=*parameter,*<br>Execute :=*parameter,*<br>ReferenceType :=*parameter,*<br>SlaveDistance :=*parameter,*<br>MasterDistance :=*parameter,*<br>MasterDistanceInACC :=*parameter,*<br>MasterDistanceInDEC :=*parameter,*<br>LinkOption :=*parameter,*<br>MasterStartDistance :=*parameter,*<br>BufferMode :=*parameter,*<br>Done =>*parameter,*<br>InSync =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| ReferenceType[*1] | Position Type Selection | _eMC_REFERENCE_TYPE | 0: _mcCommand<br>1: _mcFeedback<br>2: _mcLatestCommand | 0[*2] | Specify the position type.<br>0: Command position (value calculated in the previous primary period)<br>1: Actual position (value obtained in the same primary period)<br>2: Command position (value calculated in the same primary period) |
| SlaveDistance | Slave Axis Travel Distance | LREAL | Negative number, positive number, or 0 | 0 | Specify the travel distance of the slave axis. The unit is command units.[*3] |
| MasterDistance | Master Axis Travel Distance | LREAL | Non-negative number | 0 | Specify the travel distance of the master axis. The unit is command units.[*3] |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Master DistanceInACC | Master Distance in Acceleration | LREAL | Non-negative number | 0 | Specify the travel distance of the master axis while the slave axis is accelerating. The unit is command units.*3 |
| Master DistanceInDEC | Master Distance in Deceleration | LREAL | Non-negative number | 0 | Specify the travel distance of the master axis while the slave axis is decelerating. The unit is command units.*3 |
| LinkOption | Synchronization Start Condition | _eMC_ LINKOPTION | 0: _mcCommandEexecution<br>1: _mcTriggerDetection<br>2: _mcMasterReach | 0*2 | Specify the condition for the slave axis to synchronize with the master axis.<br>0: When instruction execution starts<br>1: When trigger is detected<br>2: When the master axis reaches the master following distance. |
| MasterStart Distance | Master Following Distance | LREAL | Negative number, positive number, or 0 | 0 | Specify the absolute position of the master axis when the slave axis starts following the master axis. The unit is command units.*3 |
| BufferMode | Buffer Mode Selection | _eMC_ BUFFER_ MODE | 0: _mcAborting<br>1: _mcBuffered | 0*2 | Specify the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered |

*1 When you use *_mcLatestCommand*, the axis number set for the *Master* (Master Axis) in the system-defined variable for motion control must be lower than the axis number set for the *Slave* (Slave Axis) in the system-defined variable for motion control.

*2 The default value for an enumeration variable is actually not the number, but the enumerator.

*3 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| InSync | In Sync | BOOL | TRUE or FALSE | TRUE when synchronization is started. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes.*

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the instruction is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| InSync | When synchronization conditions are met. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When an instruction is received. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Master | Master Axis | _sAXIS_REF | --- | Specify the master axis.*1 |
| Slave | Slave Axis | _sAXIS_REF | --- | Specify the slave axis.*1 |
| TriggerInput | Trigger Input Condition | _sTRIGGER_REF | --- | Set the trigger condition.*2 |
| TriggerVariable | Trigger Variable | BOOL | TRUE or FALSE | Specify the input variable to function as the trigger to specify the controller mode with a trigger condition. |

*1 Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

*2 Define a user-defined variable with a data type of _sTRIGGER_REF.

- **_sTRIGGER_REF**

| Name | Meaning | Data type | Valid range | Function |
|---|---|---|---|---|
| Mode | Mode | _eMC_Trigger_ Mode | 0: _mcDrive<br>1: _mcController | Specify the trigger mode.<br>0: Drive Mode<br>1: Controller Mode |
| LatchID | Latch ID Selection | _eMC_Trigger_ Latch_ID | 0: _mcLatch1<br>1: _mcLatch2 | Specify which of the two latch functions to use.<br>0: Latch 1<br>1: Latch 2 |
| InputDrive | Trigger Input Sig-nal | _eMC_Trigger_ Input _Drive | 0: _mcEncoderMark<br>1: _mcEXT | Specify the Servo Drive trigger signal to use in Drive Mode.<br>0: Z-phase signal<br>1: External input |



## Function

- The MC_MoveLink instruction moves a slave axis in synchronization with a specified master axis.
- A type of electronic cam operation is performed, but synchronous positioning is performed between the slave axis and the master axis.
- Use MC_Stop to stop the axis during motion for this instruction.

**Precautions for Safe Use**

Do not execute the MC_SetPosition instruction for the *Master* (Master Axis) if you use this instruction. If the MC_SetPosition instruction is executed for the *Master* (Master Axis), the *Slave* (Slave Axis) may follow the *Master* (Master Axis) quickly.
If you want to use the MC_SetPosition instruction for the *Master* (Master Axis), disable the relationship between the *Master* (Master Axis) and *Slave* (Slave Axis) executing the instruction.
Refer to *Precautions for Master and Auxiliary Axes in Synchronized Control* on page 1-7 for precautions on the master axis.

### Mapping Data Objects

You must map the following object data when *LinkOption* (Synchronization Start Condition) is set to *_mcTriggerDetection* and the MC_MoveLink (Synchronous Positioning) instruction executed with Mode set to Drive Mode. Mapping is performed in the Detailed Setting Area of the Axis Basic Settings Display of the Sysmac Studio.

- Touch probe function (60B8 hex)
- Touch probe status (60B9 hex)
- Touch probe pos1 pos value (60BA hex)
- Touch probe pos2 pos value (60BC hex)

If even one of the required objects is not set, a Process Data Object Setting Missing error (error code 3461 hex) occurs.

For details on mapping data objects, refer to *2-3 PDO Mapping* and to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Instruction Details

This section describes the instruction in detail.

### *Master* (Master Axis)

Specify the master axis with *Master*.

## ● *Slave* (Slave Axis)

Specify the slave axis with *Slave*.

## ● *TriggerInput* (Trigger Input Condition) and *TriggerVariable*

These variables specify the input signal on which to start synchronization when the sync start condition is set to *_mcTriggerDectection*. For the input signal selection and information on the timing when a trigger is generated, refer to *MC_TouchProbe* on page 3-313. If the Drive Mode is specified for Mode in *TriggerInput* (Trigger Input Condition), a drive input from the slave axis is used. If the Controller Mode is specified, *TriggerVariable* is used as the trigger signal.

## ● *ReferenceType* (Position Type Selection)

You can select one of the following position types.

- *_mcCommand*: Command position (value calculated in the previous primary period)
  The master axis command position that was calculated in the previous primary period is used for the current period.
  The command value that was calculated for the master axis in the last primary period is used to calculate the command position of the slave axis in the current period.
- *_mcFeedback*: Value obtained in the same primary period
  The actual position of the master axis that was obtained in the same primary period is used.
- *_mcLatestCommand*: Command position (value calculated in the same primary period)
  The command position of the master axis that was calculated in the same primary period is used. This enables the use of information that is more recent than for *_mcCommand*. However, the axis number of the master axis must be set lower than the axis number of the slave axis.
  If the axis number of the slave axis is lower than the axis number of the master axis, *Error* will change to TRUE. A Master/Slave Axis Numbers Not in Ascending Order error (error code: 5438 hex) will be output to *ErrorID*.

**Additional Information**

The command position that is calculated in the same primary period enables greater precision in synchronization than the command position that was calculated in the previous primary period. However, the axis number set for the *Master* (Master Axis) in the system-defined variable for motion control must be lower than the axis number set for the *Slave* (Slave Axis) in the system-defined variable for motion control.

## ● Relationship between Axis Types and Position Types

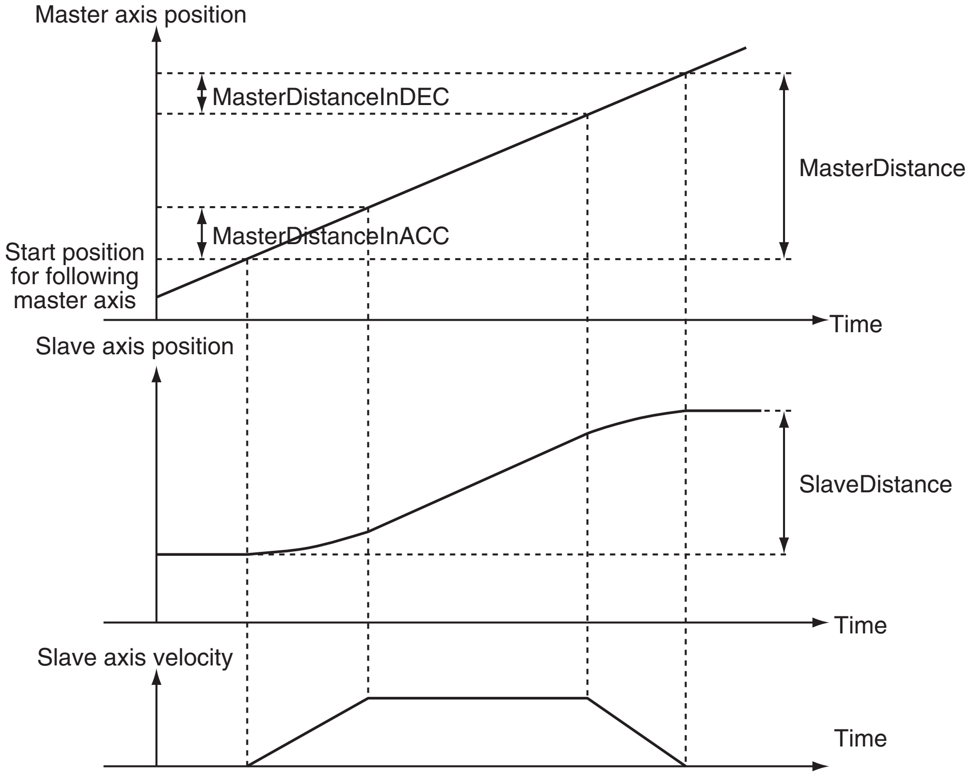
The relationship between the axis types that you can monitor and position types that is monitored is shown below.

| Axis Type | ReferenceType | |
|---|---|---|
| | _mcCommand or _mcLatestCommand | _mcFeedback |
| Servo axis | OK | OK |
| Encoder axis | No* | OK |
| Virtual servo axis | OK | OK |
| Virtual encoder axis | No* | OK |

* A Position Type Selection Out of Range error (error code: 5430 hex) occurs when the instruction is executed.

- ***SlaveDistance* (Slave Axis Travel Distance), *MasterDistance* (Master Axis Travel Distance), *MasterDistanceInACC* (Master Distance in Acceleration), and *MasterDistanceInDEC* (Master Distance in Deceleration)**

The velocity and position of the slave axis are determined by the ratio of the travel distances of the master axis and the slave axis as shown in the following figure. The master following distance shown in the following figure represents the position where the sync start condition was met.




The relationship between the travel distance of the master axis and the travel distance of the slave axis is shown in the following table.

| Section | Relationship between the master axis and slave axis travel distances | |
|---|---|---|
| Acceleration | Master axis | Master Distance in Acceleration |
| | Slave axis | $\text{Slave axis travel distance} \times \dfrac{\dfrac{\text{Master distance in acceleration}}{2}}{\dfrac{\text{Master distance in acceleration}}{2} + \text{Master axis travel distance} - \text{Master distance in acceleration} - \text{Master distance in deceleration} + \dfrac{\text{Master distance in deceleration}}{2}}$ |
| Constant velocity | Master axis | Master distance – Master distance in acceleration – Master distance in deceleration |
| | Slave axis | Slave axis travel distance – Slave axis travel distance at the acceleration rate above – Slave axis travel distance at the deceleration rate below |
| Deceleration | Master axis | Master Distance in Deceleration |
| | Slave axis | $\text{Slave axis travel distance} \times \dfrac{\dfrac{\text{Master distance in deceleration}}{2}}{\dfrac{\text{Master distance in acceleration}}{2} + \text{Master axis travel distance} - \text{Master distance in acceleration} - \text{Master distance in deceleration} + \dfrac{\text{Master distance in deceleration}}{2}}$ |

When the constant velocity section of the master axis is negative, a constant velocity travel distance error occurs and the axis stops. If you want to feed the slave axis at the same velocity as the master axis, set the following value as the travel distance of the slave axis.

$$\text{Slave axis travel distance} = \frac{\text{Master distance in acceleration}}{2} + \text{Master axis travel distance} - \text{Master distance in acceleration} - \text{Master distance in deceleration} + \frac{\text{Master distance in deceleration}}{2}$$

## ● *LinkOption* (Synchronization Start Condition)

Specify the condition for the slave axis to synchronize with the master axis.

- Start of Instruction
  When this instruction is executed, the slave axis performs positioning in synchronization with the master axis from the next period.
- When Trigger Is Detected
  When the input signal specified as the input trigger occurs, the slave axis synchronizes with the master axis and performs positioning from the next period.
- When the Master Axis Reaches the Master Following Distance
  When the master axis reaches the master following distance during instruction execution, the slave axis starts synchronization and performs positioning from the next period.
  Even if the instruction is executed while the master axis is stopped at the master following distance, the slave axis starts synchronization and performs positioning from the next period.

**Precautions for Correct Use**

You must map object data when LinkOption (Synchronization Start Condition) is set *to_mcTriggerDetection* and Mode is set to Drive Mode. Set the following objects.

- Touch probe function (60B8 hex)
- Touch probe status (60B9 hex)
- Touch probe pos1 pos value (60BA hex)
- Touch probe pos2 pos value (60BC hex)

If even one of the required objects is not set, a Process Data Object Setting Missing error (error code 3461 hex) occurs. For details on mapping object data, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## ● *MasterStartDistance*

*MasterStartDistance* specifies the absolute position where the slave axis starts synchronization with the master axis.

## ● *BufferMode* (Buffer Mode Selection)

*BufferMode* specifies how to join the axis motions for this instruction and the previous instruction. There is currently only the following two settings.

| Buffer Mode Selection | Description |
|---|---|
| Aborting | Aborts the instruction being executed and switches to this instruction. |
| Buffered | Automatically executes the buffered instruction after the current instruction is completed. |

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

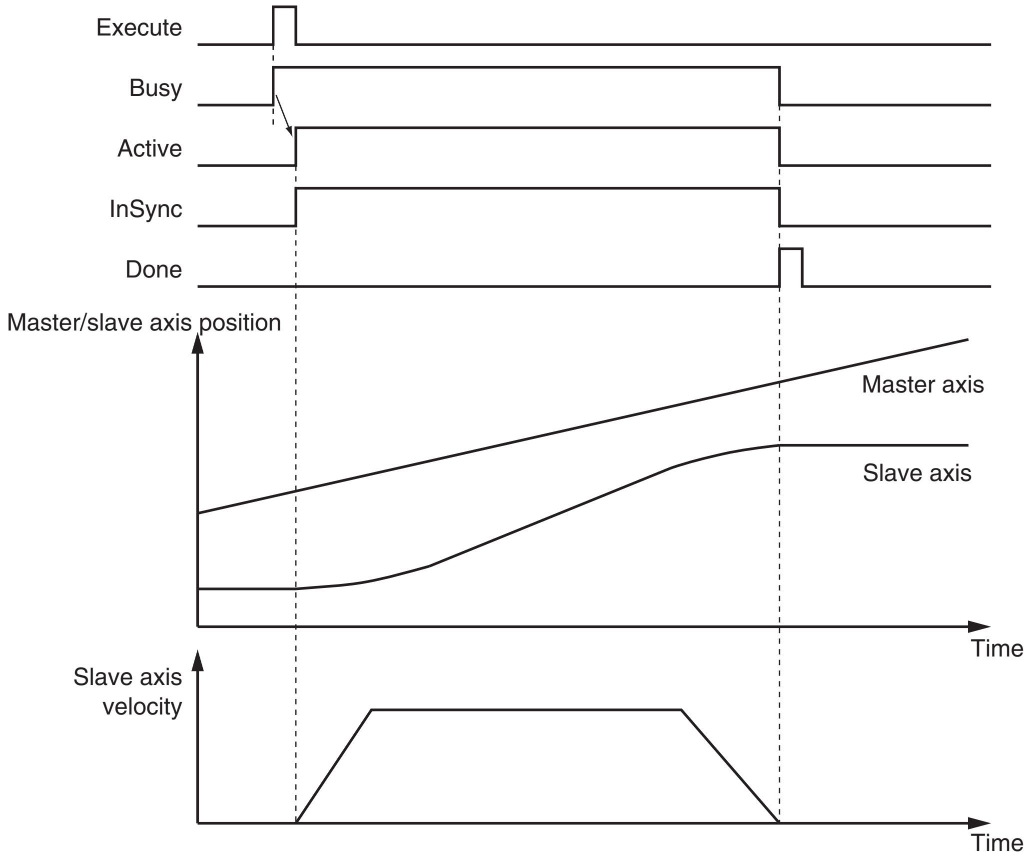
## ● In-position Check

An in-position check is performed when the slave axis ends operation.
An in-position check is performed for this instruction according to the settings in In-position Range and In-position Check Time axis parameters.

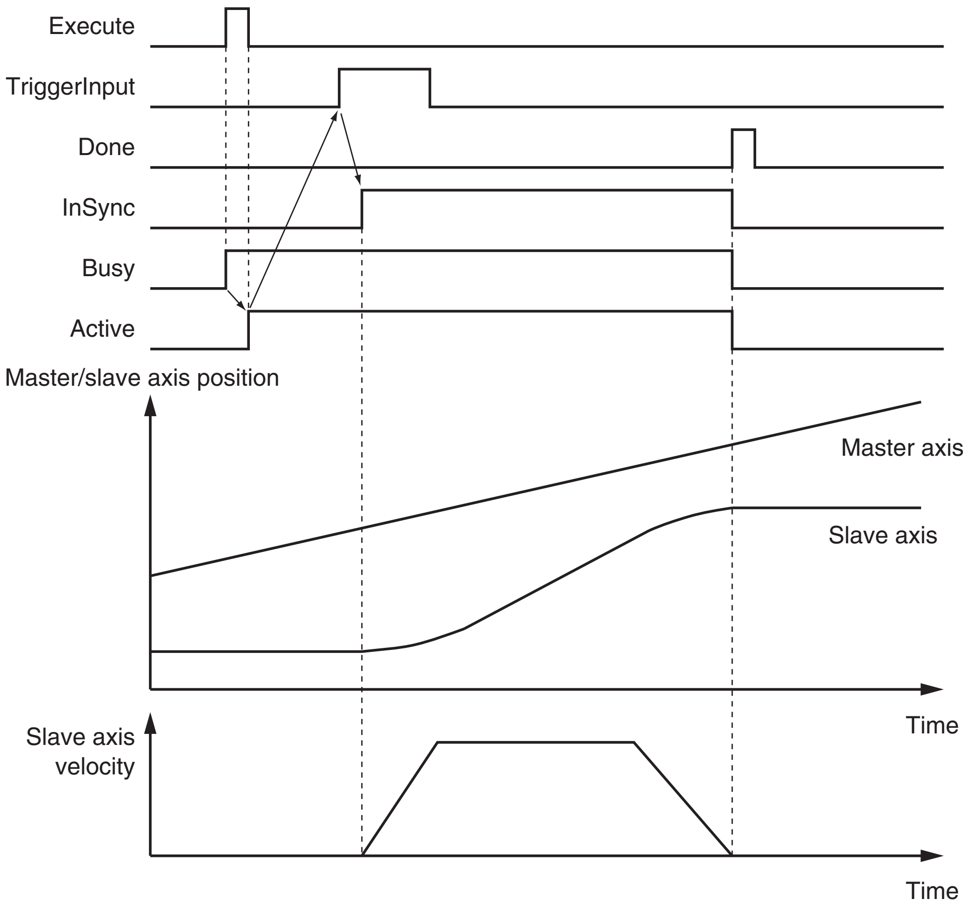
## Timing Charts

- **Executing MC_MoveLink**

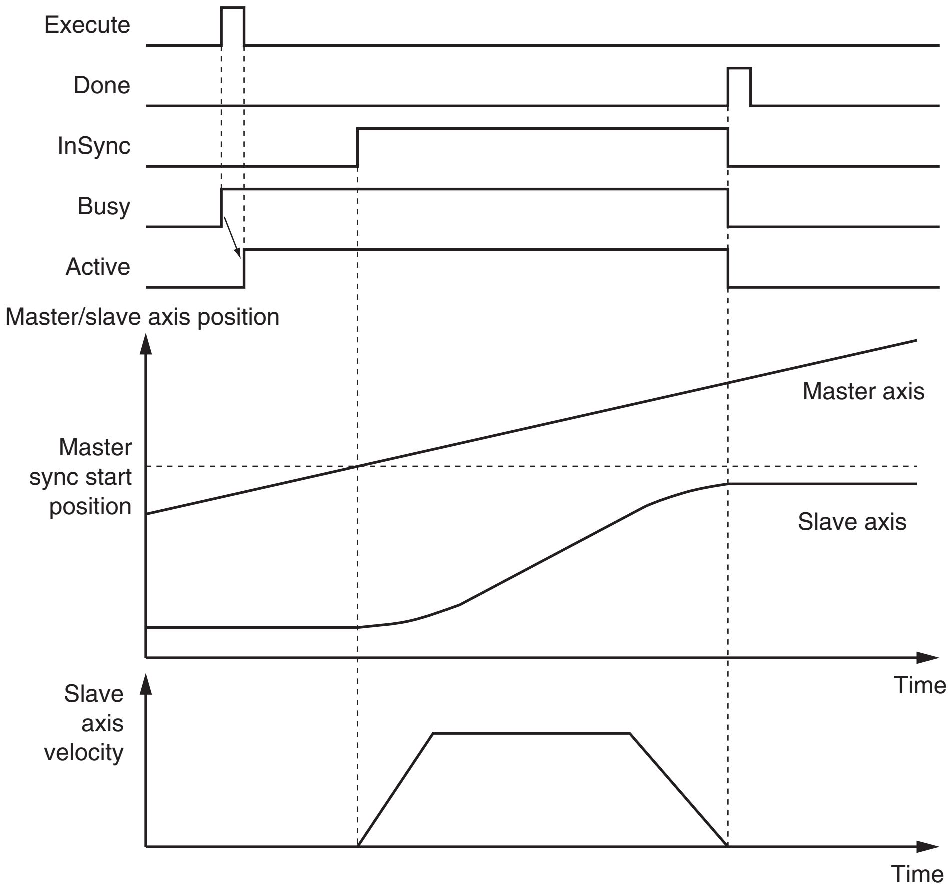
***LinkOption* Set to *_mcCommandExecution***



### *LinkOption* Set to *_mcTriggerDetection*



### Sync Start Condition Setting = When the Master Axis Reaches the Master Following Distance

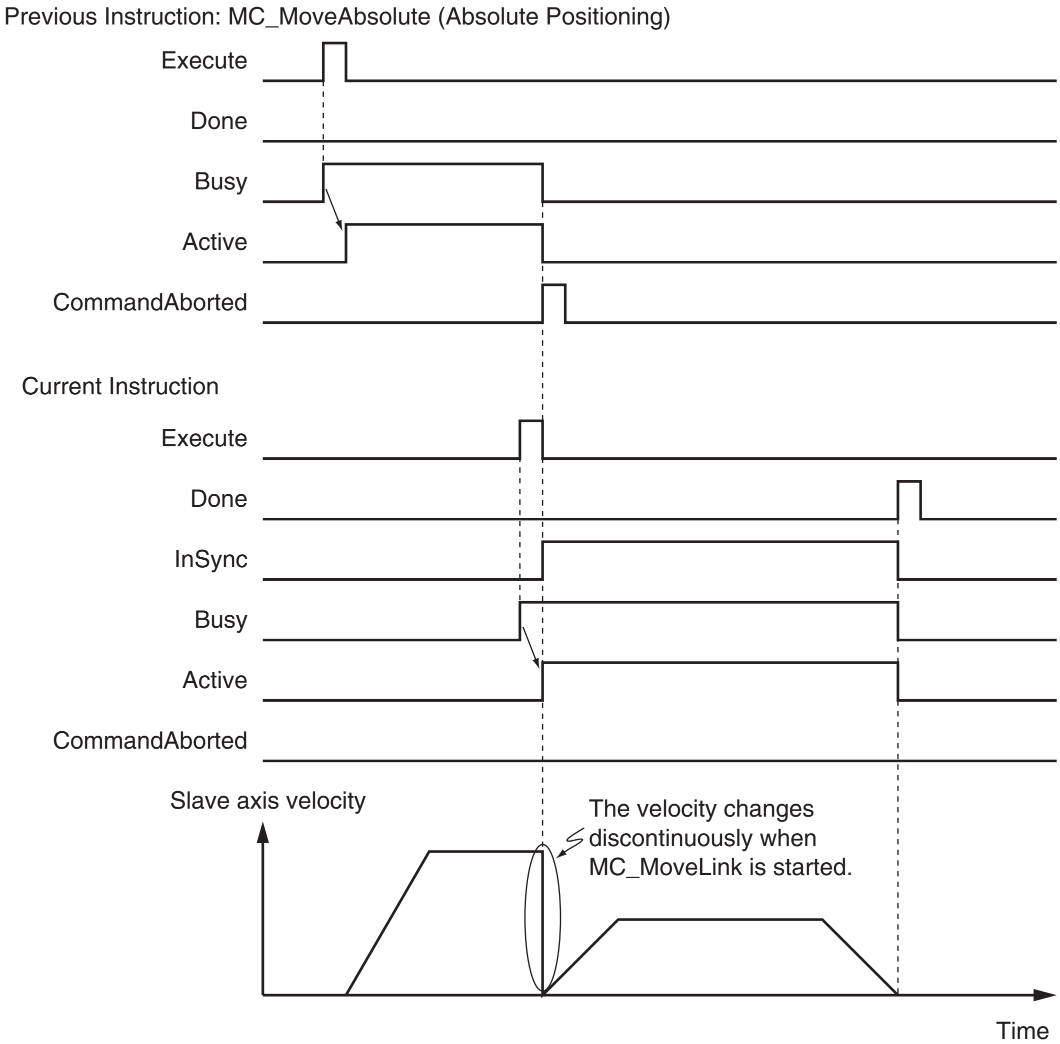


- **When an Instruction Is Executed with *BufferMode* Set to Aborting during Previous Operation**

This section describes when this instruction is executed with *LinkOption* set to *_mcCommandExecution* when the previous operation, MC_MoveAbsolute (Absolute Positioning), is in execution.

**Precautions for Correct Use**

As shown in the following chart, the velocity of the slave axis becomes discontinuous when this instruction is started.

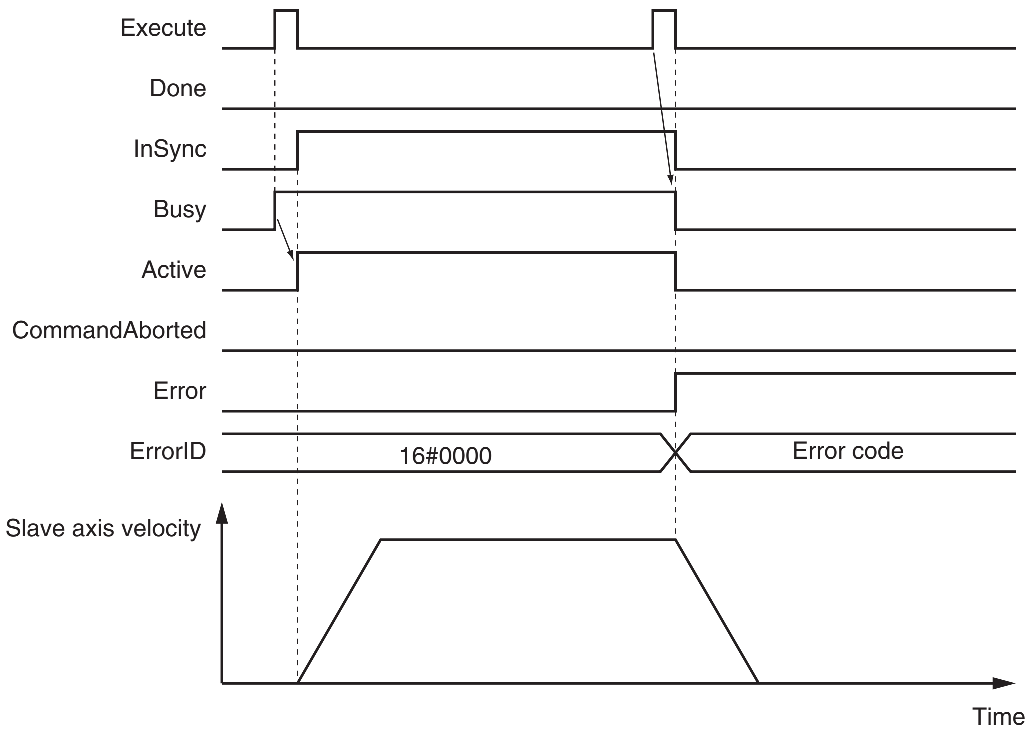



- **When an Instruction Is Executed with *BufferMode* Set to *Buffered* during Previous Operation**

This instruction is executed after the previous instruction is finished.

## Re-execution of Motion Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted, and the axis stops.




## Multi-execution of Motion Instructions

For details on multi-execution of instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution of Other Instructions during Instruction Execution

If you execute another instruction during execution of this instruction, you can specify either aborting or buffering.
You cannot specify blending.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code). Refer to *Re-execution of Motion Instructions* on page 3-261 for the timing chart after an error occurs.

### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# Sample Programming

This sample programming shows the control of a cutter.

## Parameter Settings

The minimum settings required for this sample programming are given below.

### ● Setting Axis Parameters

#### Axis Types

| Axis | Axis Type |
|---|---|
| Axis 1 | Servo axis (master axis) |
| Axis 2 | Servo axis (slave axis) |

#### Count Modes

| Axis | Count Mode |
|---|---|
| Axis 1 | Rotary Mode |
| Axis 2 | Linear Mode |

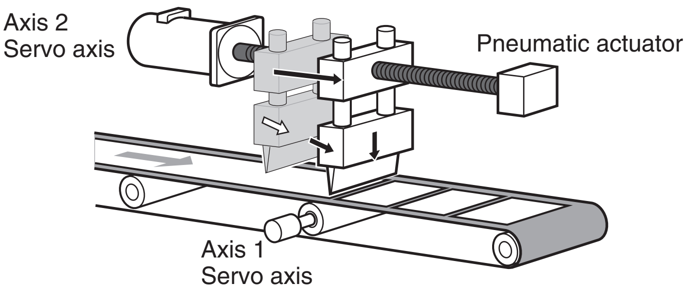
#### Ring Counter

| Axis | Modulo maximum position | Modulo minimum position |
|---|---|---|
| Axis 1 | 360 | 0 |

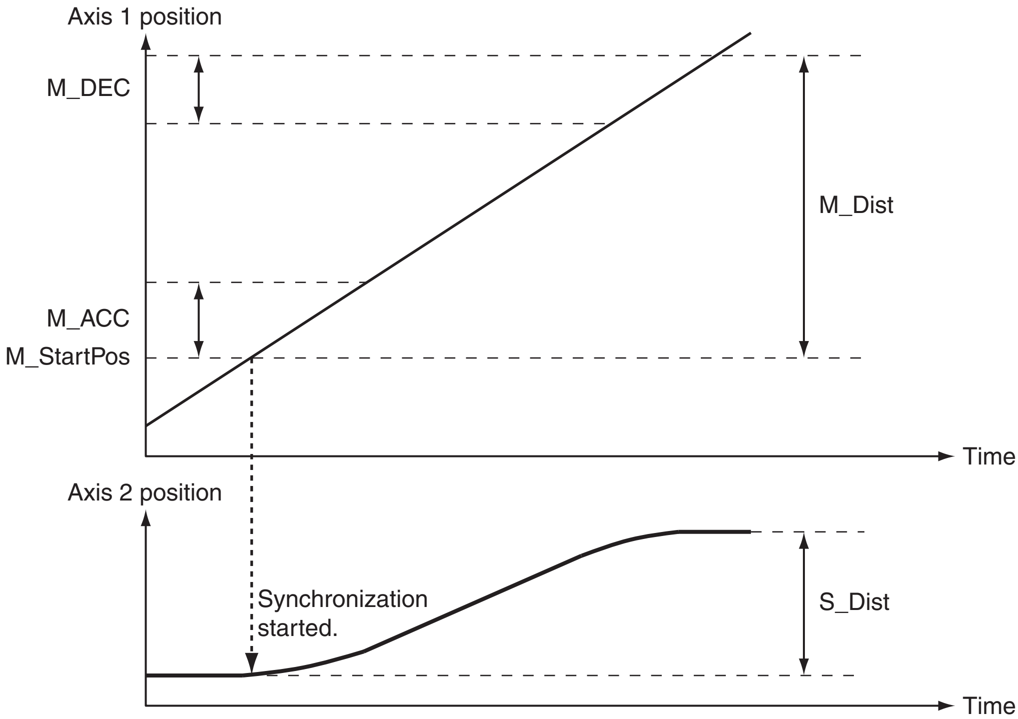
#### Units of Display

| Axis | Unit of Display |
|---|---|
| Axis 1 | degree |
| Axis 2 | mm |

## Operation Example



- **Operation Patterns**




**1** Starting the Master Axis

Axis 1 for the belt conveyer is treated as master axis to feed back the position.

**2** Executing the Slave Axis

Axis 2 for the ball screw that moves in the horizontal direction moves in synchronization with axis 1.

**3** Cutting with the Cutter

The pneumatic actuator turns ON when axis 2 is synchronized. The cutter, which is connected to the pneumatic actuator, descends in a vertical direction and cuts the workpiece.

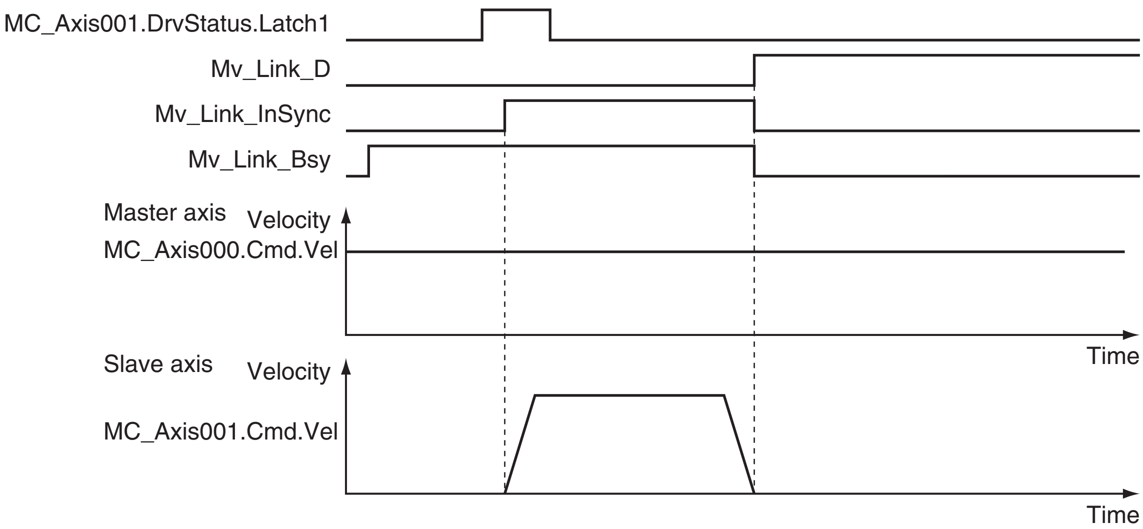
## Ladder Diagram

- **Main Variables**

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for the master axis, axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 2. |
| MC_Axis001.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 2. |
| MC_Axis001.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 2. |
| Pwr1_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR1 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |

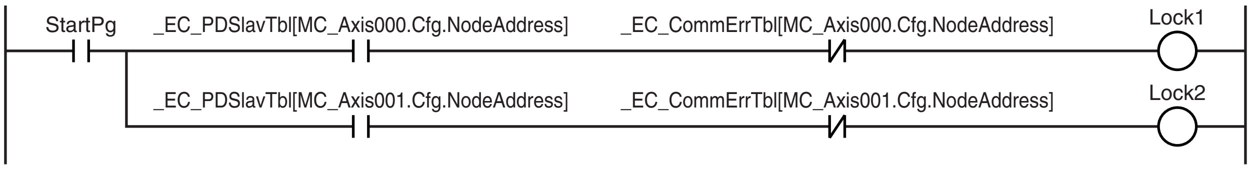
| Name | Data type | Default | Comment |
|---|---|---|---|
| Pwr2_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR2 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| Actuator | BOOL | FALSE | TRUE when axis 1 and axis 2 are synchronized. While *Actuator* is TRUE, the cutter moves down vertically. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

## ● Timing Chart

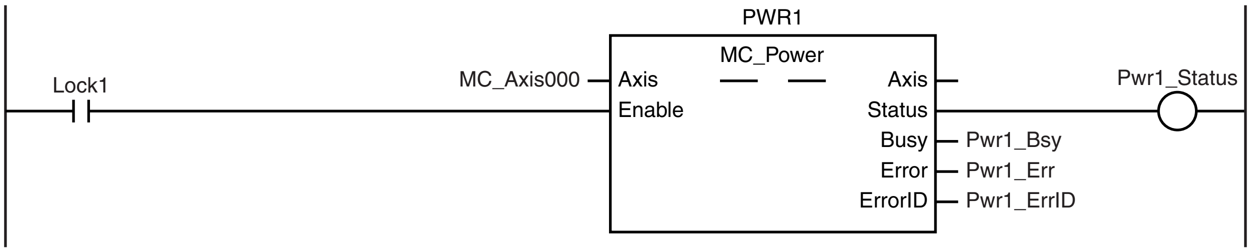


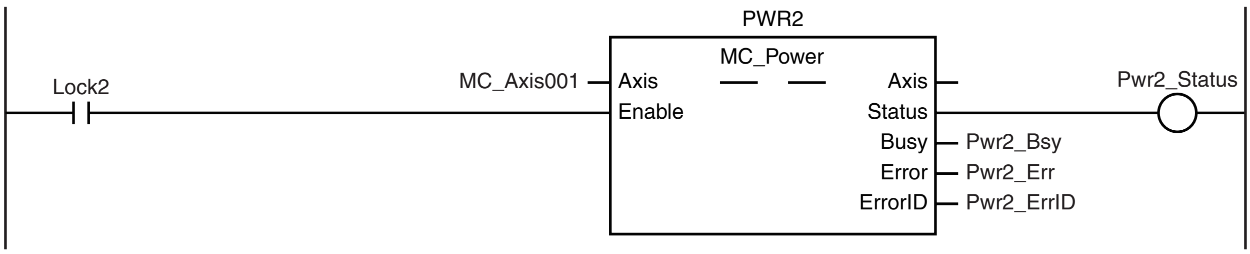
## Sample Programming

If *StartPg* is TRUE, EtherCAT communications for the axes are checked to see if process data communications are normal.

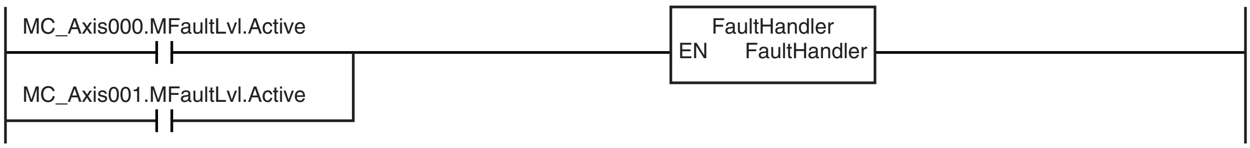



If process data communications are active, the Servo is turned ON for the axes.

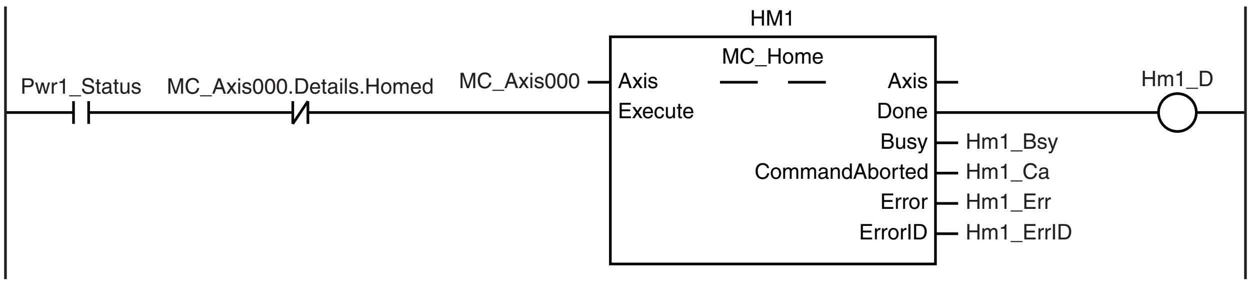


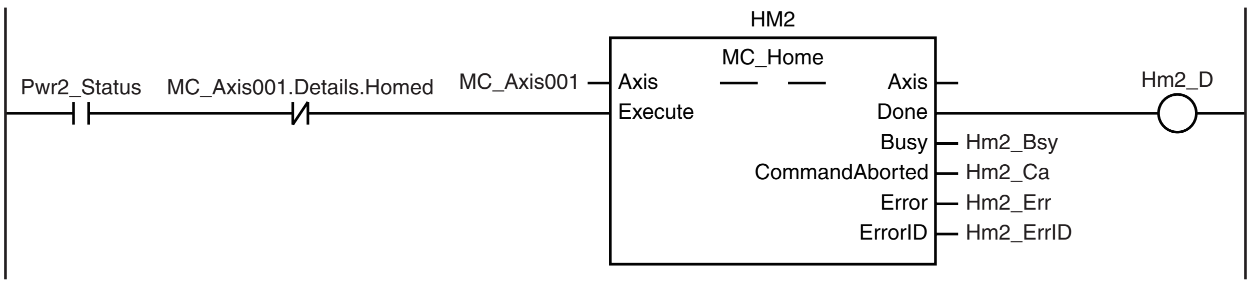

If there is a minor fault level error for a composition axis, the error handler for the device (FaultHandler) is executed. Design FaultHandler separately.



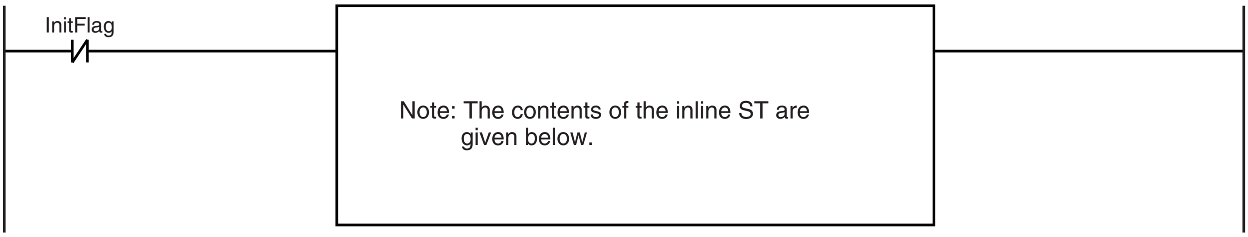

If the Servo is ON for axis 1 and home is not defined, the Home instruction is executed to define home.



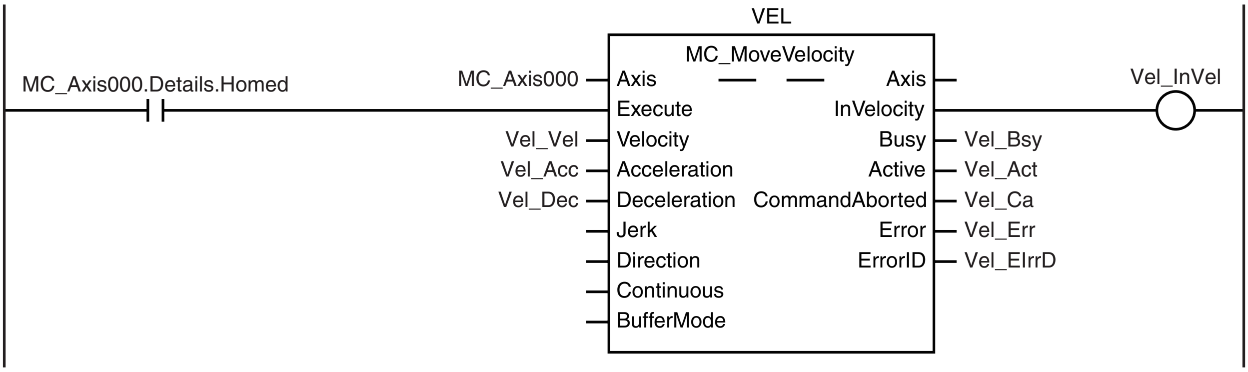

If the Servo is ON for axis 2 and home is not defined, the Home instruction is executed to define home.



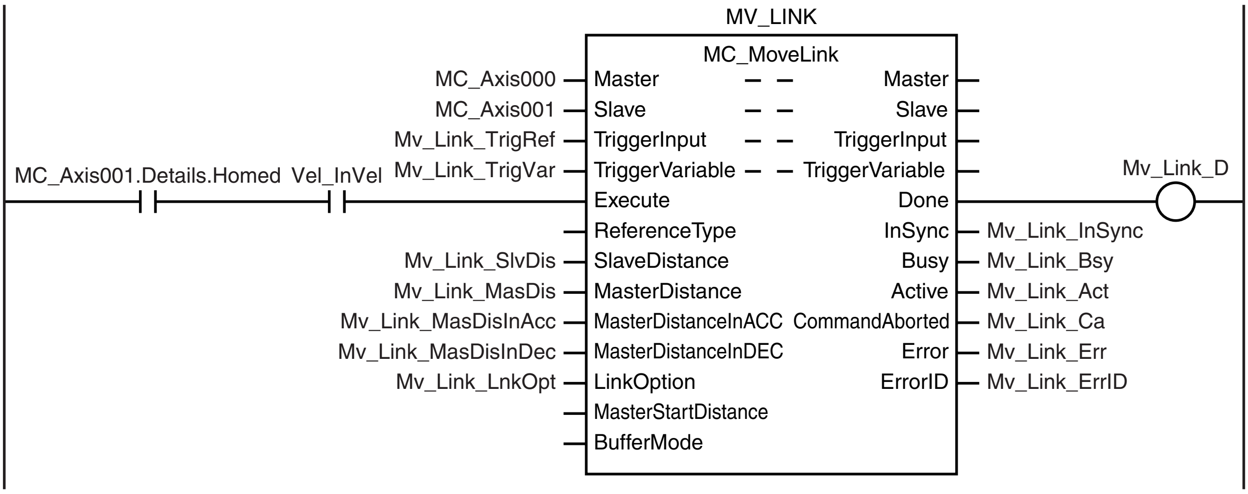
The parameters are set for the MC_MoveVelocity (Velocity Control) and MC_MoveLink (Synchronous Positioning) instructions.



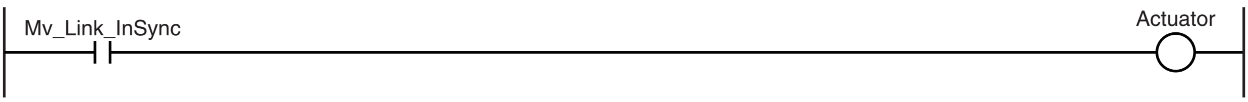

The MC_MoveVelocity (Velocity Control) instruction is executed if home is defined for axis 1.




The MC_MoveLink (Synchronous Positioning) instruction is executed if home is defined for axis 2 (slave axis).




*Actuator* is TRUE while the axes are synchronized.




## Contents of Inline ST

```
// MC_MoveVelocity parameters
Vel_Vel     := LREAL#1000.0;
Vel_Acc     := LREAL#0.0;
Vel_Dec     := LREAL#0.0;

// MC_MoveLink parameters
Mv_Link_TrigRef.Mode          := _eMC_TRIGGER_MODE#_mcDrive;
Mv_Link_TrigRef.LatchID       := _eMC_TRIGGER_LATCH_ID#_mcLatch1;
Mv_Link_TrigRef.InputDrive    := _eMC_TRIGGER_INPUT_DRIVE#_mcEXT;
Mv_Link_TrigVar               := FALSE;
Mv_Link_SlvDis                := LREAL#1000.0;
Mv_Link_MasDis                := LREAL#1000.0;
Mv_Link_MasDisInAcc           := LREAL#100.0;
Mv_Link_MasDisInDec           := LREAL#100.0;
```

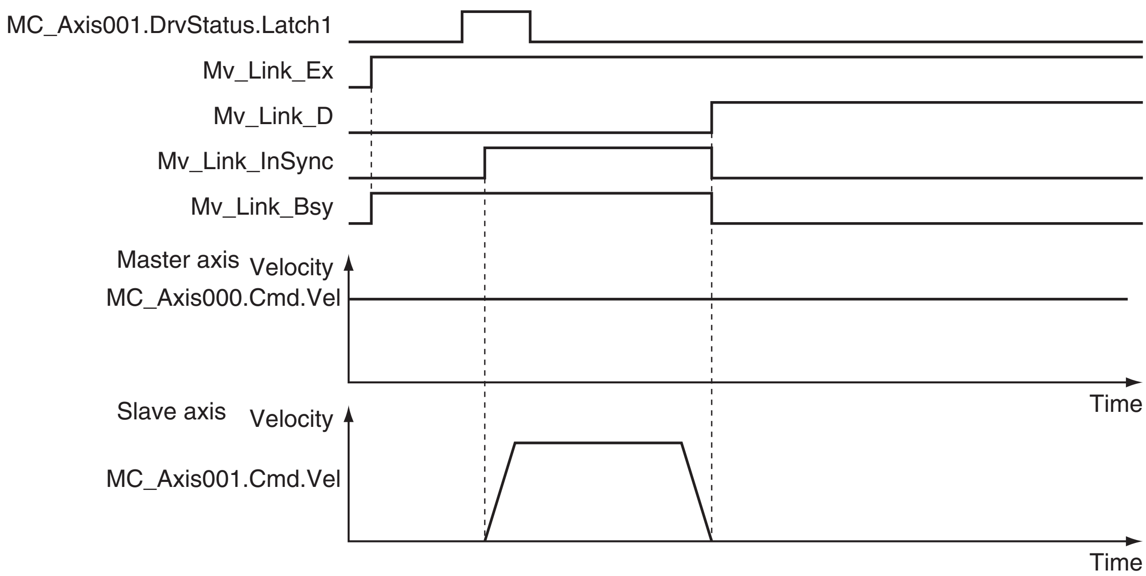
```
Mv_Link_LnkOpt                := _eMC_LINKOPTION#_mcTriggerDetection;

// Change InitFlag to TRUE after setting the input parameters.
InitFlag := TRUE;
```

## Structured Text (ST)

### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | Axis Variable for the master axis, axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | Axis Variable for the slave axis, axis 2. |
| MC_Axis001.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 2. |
| MC_Axis001.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 2. |
| Pwr1_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR1 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Pwr2_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR2 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| Actuator | BOOL | FALSE | TRUE when axis 1 and axis 2 are synchronized. While *Actuator* is TRUE, the cutter moves down vertically. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

- **Timing Chart**



## Sample Programming

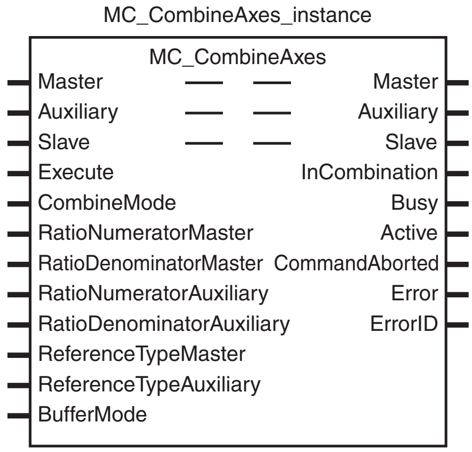
```
// Processing when input parameters are not set
IF InitFlag=FALSE THEN

    // MC_MoveVelocity parameters
    Vel_Vel    := LREAL#1000.0;
    Vel_Acc    := LREAL#0.0;
    Vel_Dec    := LREAL#0.0;

    // MC_MoveLink parameters
    Mv_Link_TrigRef.Mode          := _eMC_TRIGGER_MODE#_mcDrive;
    Mv_Link_TrigRef.LatchID       := _eMC_TRIGGER_LATCH_ID#_mcLatch1;
    Mv_Link_TrigRef.InputDrive    := _eMC_TRIGGER_INPUT_DRIVE#_mcEXT;
    Mv_Link_TrigVar               := FALSE;
    Mv_Link_SlvDis                := LREAL#1000.0;
    Mv_Link_MasDis                := LREAL#1000.0;
    Mv_Link_MasDisInAcc           := LREAL#100.0;
    Mv_Link_MasDisInDec           := LREAL#100.0;
    Mv_Link_LnkOpt                := _eMC_LINKOPTION#_mcTriggerDetection;

    // Change InitFlag to TRUE after setting the input parameters.
    InitFlag := TRUE;

END_IF;


// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 1 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis000.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
    Pwr1_En:=TRUE;          // Turn ON the Servo for axis 1.
ELSE
    Pwr1_En:=FALSE;         // Turn OFF the Servo for axis 1.
END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 2 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis001.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis001.Cfg.NodeAddress]=FALSE) THEN
    Pwr2_En:=TRUE;          // Turn ON the Servo for axis 2.
ELSE
    Pwr2_En:=FALSE;         // Turn OFF the Servo for axis 2.
END_IF;

// Processing for a minor fault level error
// Implement FaultHandler separately.
IF (MC_Axis000.MFaultLvl.Active=TRUE) OR (MC_Axis001.MFaultLvl.Active=TRUE) THEN
    FaultHandler();
END_IF;

// If the Servo is ON for axis 1 and home is not defined, the Home instruction is executed.
IF (Pwr1_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
    Hm1_Ex:=TRUE;
END_IF;

// If the Servo is ON for axis 2 and home is not defined, the Home instruction is executed.
IF (Pwr2_Status=TRUE) AND (MC_Axis001.Details.Homed=FALSE) THEN
    Hm2_Ex:=TRUE;
END_IF;
// After home is defined for axis 1, MC_MoveVelocity is executed for axis 1.
IF MC_Axis000.Details.Homed=TRUE THEN
    Vel_Ex:=TRUE;
```

```
END_IF;

// MC_MoveLink is executed for axis 2 if home is defined for axis 2 and the target velocity was reached for axis
1.
IF (MC_Axis001.Details.Homed=TRUE) AND (Vel_InVel=TRUE) THEN
    Mv_Link_Ex:=TRUE;
END_IF;

// The actuator is turned ON if axis 1 and axis 2 are synchronized.
IF Mv_Link_InSync=TRUE THEN
    Actuator:=TRUE;
ELSE
    Actuator:=FALSE;
END_IF;


// MC_Power1
PWR1(
    Axis        := MC_Axis000,
    Enable      := Pwr1_En,
    Status      => Pwr1_Status,
    Busy        => Pwr1_Bsy,
    Error       => Pwr1_Err,
    ErrorID     => Pwr1_ErrID
);

// MC_Power2
PWR2(
    Axis        := MC_Axis001,
    Enable      := Pwr2_En,
    Status      => Pwr2_Status,
    Busy        => Pwr2_Bsy,
    Error       => Pwr2_Err,
    ErrorID     => Pwr2_ErrID
);

// MC_Home1
HM1(
    Axis                := MC_Axis000,
    Execute             := Hm1_Ex,
    Done                => Hm1_D,
    Busy                => Hm1_Bsy,
    CommandAborted      => Hm1_Ca,
    Error               => Hm1_Err,
    ErrorID             => Hm1_ErrID
);

// MC_Home2
HM2(
    Axis                := MC_Axis001,
    Execute             := Hm2_Ex,
    Done                => Hm2_D,
    Busy                => Hm2_Bsy,
    CommandAborted      => Hm2_Ca,
    Error               => Hm2_Err,
    ErrorID             => Hm2_ErrID
);

// MC_MoveVelocity
VEL(
    Axis                := MC_Axis000,
    Execute             := Vel_Ex,
    Velocity            := Vel_Vel,
    Acceleration        := Vel_Acc,
    Deceleration        := Vel_Dec,
```

```
    InVelocity          => Vel_InVel,
    Busy                => Vel_Bsy,
    Active              => Vel_Act,
    CommandAborted      => Vel_Ca,
    Error               => Vel_Err,
    ErrorID             => Vel_ErrID
);

// MC_MoveLink
MV_LINK(
    Master                  := MC_Axis000,
    Slave                   := MC_Axis001,
    TriggerInput            := Mv_Link_TrigRef,
    TriggerVariable         := Mv_Link_TrigVar,
    Execute                 := Mv_Link_Ex,
    SlaveDistance           := Mv_Link_SlvDis,
    MasterDistance          := Mv_Link_MasDis,
    MasterDistanceInAcc     := Mv_Link_MasDisInAcc,
    MasterDistanceInDec     := Mv_Link_MasDisInDec,
    LinkOption              := Mv_Link_LnkOpt,
    Done                    => Mv_Link_D,
    InSync                  => Mv_Link_InSync,
    Busy                    => Mv_Link_Bsy,
    Active                  => Mv_Link_Act,
    CommandAborted          => Mv_Link_Ca,
    Error                   => Mv_Link_Err,
    ErrorID                 => Mv_Link_ErrID
);
```

# MC_CombineAxes

The MC_CombineAxes instruction outputs the sum or difference of the command positions of two axes as the command position.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_CombineAxes | Combine Axes | FB |   | MC_CombineAxes_instance (<br>Master :=*parameter,*<br>Auxiliary :=*parameter,*<br>Slave :=*parameter,*<br>Execute :=*parameter,*<br>CombineMode :=*parameter,*<br>RatioNumeratorMaster :=*parameter,*<br>RatioDenominatorMaster :=*parameter,*<br>RatioNumeratorAuxiliary :=*parameter,*<br>RatioDenominatorAuxiliary :=*parameter,*<br>ReferenceTypeMaster :=*parameter,*<br>ReferenceTypeAuxiliary :=*parameter,*<br>BufferMode :=*parameter,*<br>InCombination =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when Execute changes to TRUE. |
| CombineMode | Combine Mode | _eMC _COMBINE_ MODE | 0: _mcAddAxes<br>1: _mcSubAxes | 0* | Specify the combining method.<br>0: Addition<br>1: Subtraction |
| Ratio Numerator Master<br>(Reserved) | Master Axis Gear Ratio Numerator | UINT | Positive number | 10000 | Reserved |
| Ratio Denominator Master<br>(Reserved) | Master Axis Gear Ratio Denominator | UINT | Positive number | 10000 | Reserved |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Ratio Numerator Auxiliary (Reserved) | Auxiliary Axis Gear Ratio Numerator | UINT | Positive number | 10000 | Reserved |
| Ratio Denominator Auxiliary (Reserved) | Auxiliary Axis Gear Ratio Denominator | UINT | Positive number | 10000 | Reserved |
| Reference TypeMaster | Master Axis Position Type Selection | _eMC_ REFERENCE_ TYPE | 1: _mcFeedback<br>2: _mcLastestCommand | 2* | Specify the position type of the master axis.<br>1: Actual position (value obtained in the same primary period)<br>2: Command position (value calculated in the same primary period) |
| Reference TypeAuxiliary | Auxiliary Axis Position Type Selection | _eMC_ REFERENCE_ TYPE | 1: _mcFeedback<br>2: _mcLastestCommand | 2* | Specify the position type of the auxiliary axis.<br>1: Actual position (value obtained in the same primary period)<br>2: Command position (value calculated in the same primary period) |
| BufferMode | Buffer Mode Selection | _eMC _BUFFER_ MODE | 0: _mcAborting | 0* | Specify the operation when executing more than one motion instruction.<br>0: Aborting |

* The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| InCombination | Axes Combined | BOOL | TRUE or FALSE | TRUE when axes are combined. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| InCombination | When combining axes is started. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Busy | When *Execute* changes to TRUE. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the instruction is started. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Master | Master Axis | _sAXIS_REF | --- | Specify the master axis.* |
| Auxiliary | Auxiliary Axis | _sAXIS_REF | --- | Specify the auxiliary axis.* |
| Slave | Slave Axis | _sAXIS_REF | --- | Specify the slave axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

**Precautions for Correct Use**

One of the following minor faults will occur if the different axes are not used for the master, slave, and auxiliary axes.

- Master and Slave Defined as Same Axis (error code 5436 hex)
- Master and Auxiliary Defined as Same Axis (error code 5437 hex)
- Auxiliary and Slave Defined as Same Axis (error code 548E hex)

## Function

- The MC_CombineAxes instruction starts combining axes when *Execute* changes to TRUE.

**Precautions for Safe Use**

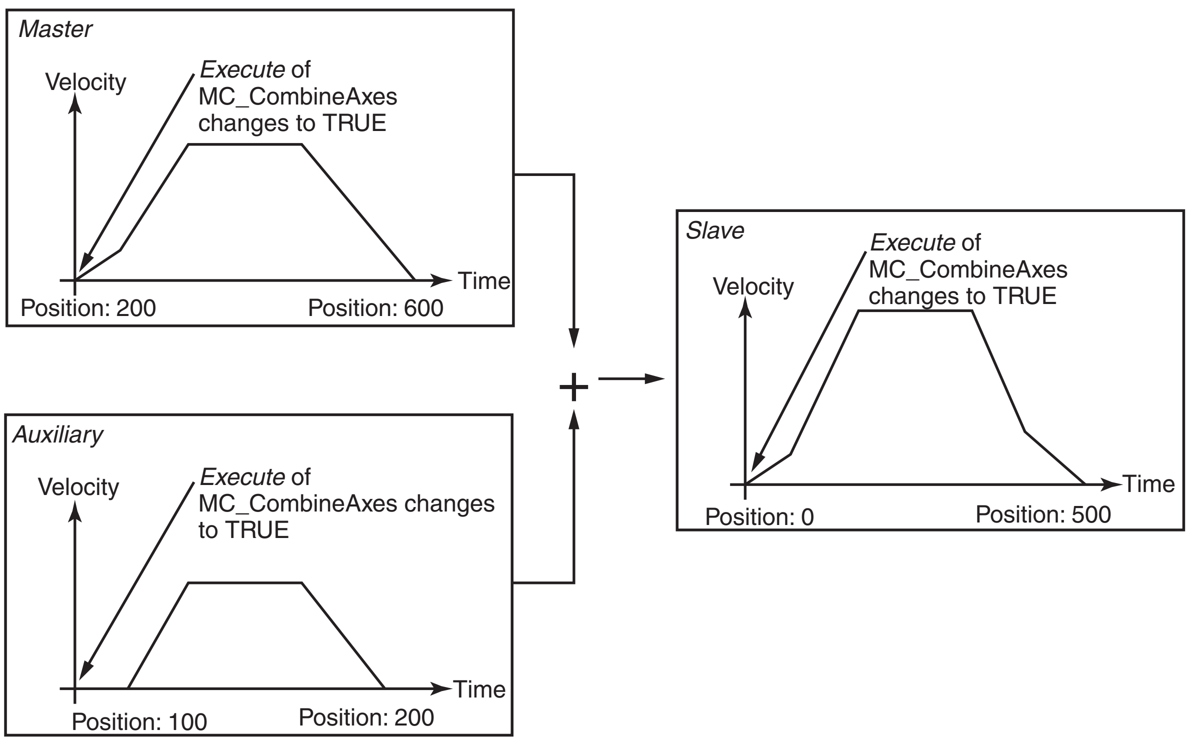
Do not execute the MC_SetPosition instruction for the *Master* (Master Axis) if you use this instruction. If the MC_SetPosition instruction is executed for the *Master* (Master Axis), the *Slave* (Slave Axis) may follow the *Master* (Master Axis) quickly.
If you want to use the MC_SetPosition instruction for the *Master* (Master Axis), disable the relationship between the *Master* (Master Axis) and *Slave* (Slave Axis) executing the instruction.
Refer to *Precautions for Master and Auxiliary Axes in Synchronized Control* on page 1-7 for precautions on the master axis. This precaution also applies to *Auxiliary* (Auxiliary Axis) in the same way as for *Master* (Master Axis).

## Instruction Details

From the starting point, the relative value of the *Auxiliary* (Auxiliary Axis) position is added to or subtracted from the relative value of the *Master* (Master Axis) position and is output as a relative value for the *Slave* (Slave Axis) command position.

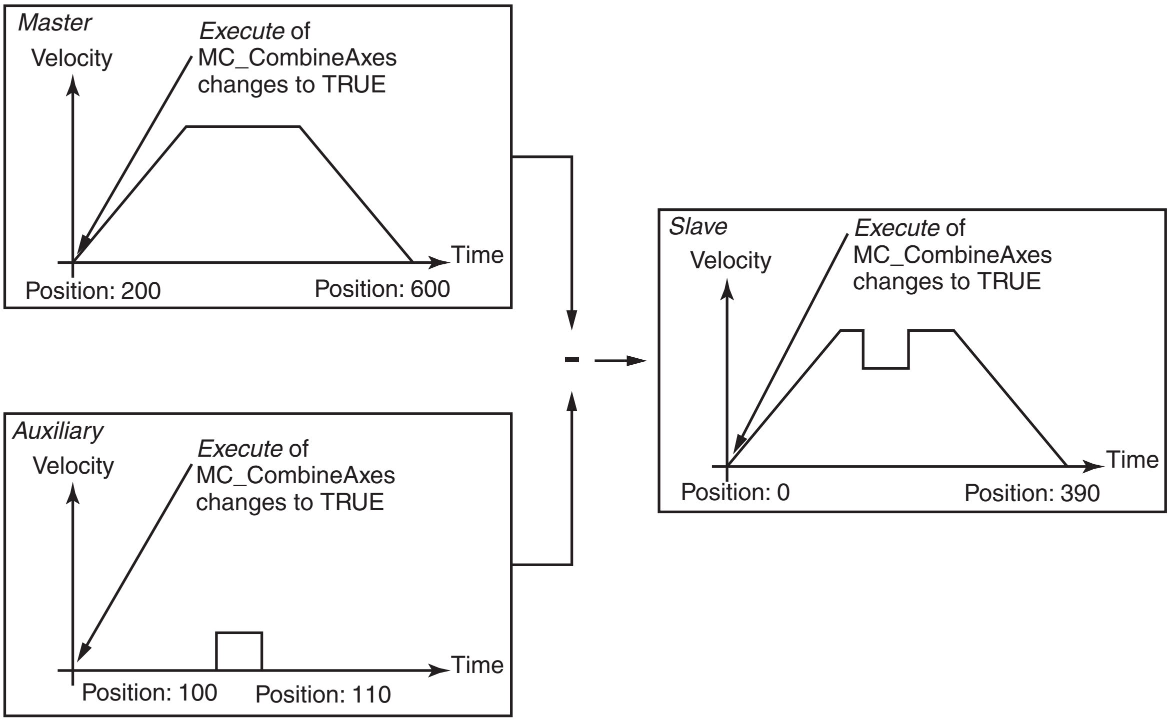
### ● *CombineMode* Set to *_mcAddAxes*

*Slave* (Slave Axis) position = *Master* (Master Axis) position (relative position) + *Auxiliary* (Auxiliary Axis) position (relative position)



## ● *CombineMode* Set to *_mcSubAxes*

*Slave* (Slave Axis) position = *Master* (Master Axis) position (relative position) – *Auxiliary* (Auxiliary Axis) position (relative position)




- Adding or subtracting the position is performed as numerical operations without considering the Unit of Display for the axis in the axis parameters.
- Execute the MC_Stop instruction to end this instruction.

**Precautions for Safe Use**

Depending on the values for the *Master* (Master Axis) and *Auxiliary* (Auxiliary Axis), the travel distance, velocity, and acceleration of the *Slave* (Slave Axis) can change rapidly.
Use this setting with care.

## ● In-position Check

An in-position check is not performed for this instruction.

## ● Override Factors

You cannot perform an MC_SetOverride (Set Override Factors) override for this instruction.

## ● *ReferenceType* (Position Type Selection)

You can select one of the following position types.

- *_mcFeedback*: Value obtained in the same primary period
  The actual position of the master axis that was obtained in the same primary period is used.
- *_mcLatestCommand*: Command position (value calculated in the same primary period)
  The command position of the master axis that was calculated in the same primary period is used. However, if *_mcLatestCommand* is selected, the axis numbers of the master axis and auxiliary axis must be set lower than the axis number of the slave axis. If the axis numbers of the slave axis or auxiliary axis is lower than the axis number of the master axis, *Error* will change to TRUE. A Master/Slave Axis Numbers Not in Ascending Order error (error code: 5438 hex) will be output to *ErrorID*. There are no restrictions in the relationship of the axis numbers between the master axis and the auxiliary axis.

### Relationship between Axis Types and Position Types

The relationship between the axis types that you can monitor and position types that is monitored is shown below.

| Axis Type | ReferenceType | |
|---|---|---|
| | _mcFeedback | _mcLatestCommand |
| Servo axis | OK | OK |
| Encoder axis | OK | No* |
| Virtual servo axis | OK | OK |
| Virtual encoder axis | OK | No* |

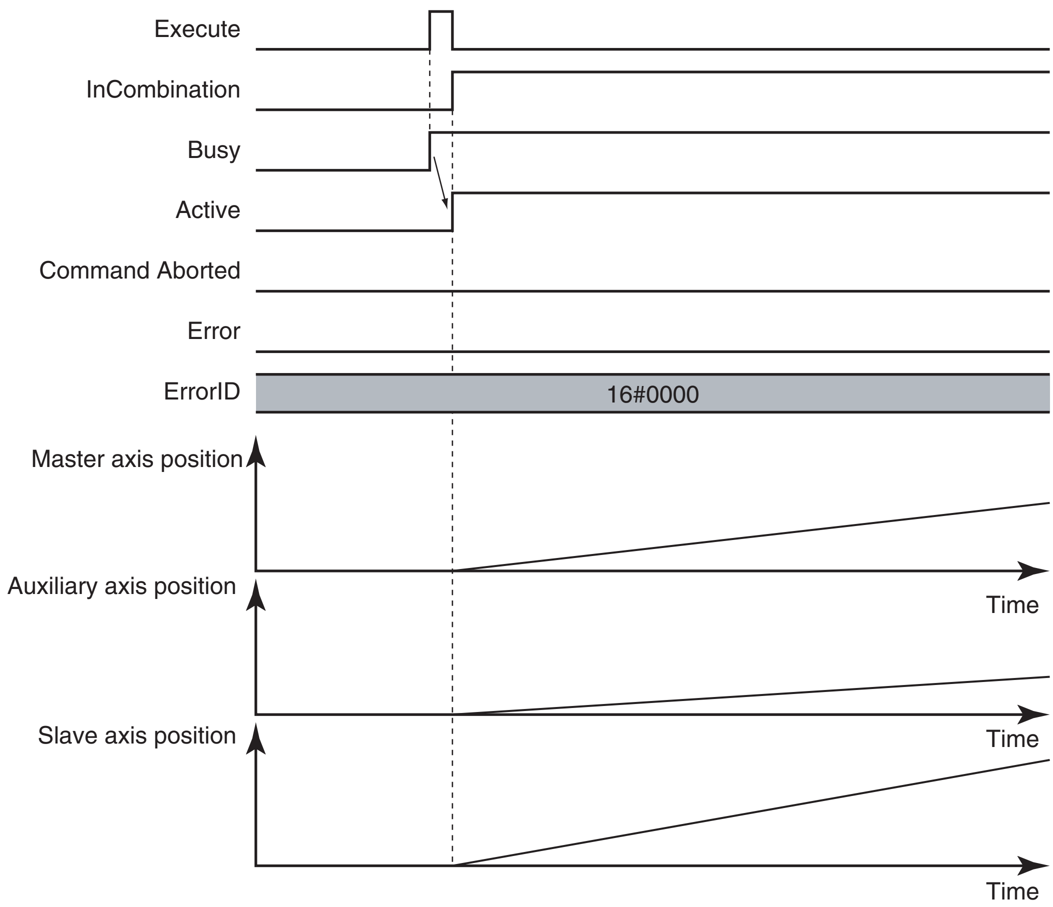
* A Position Type Selection Out of Range error (error code: 5430 hex) occurs when the instruction is executed.



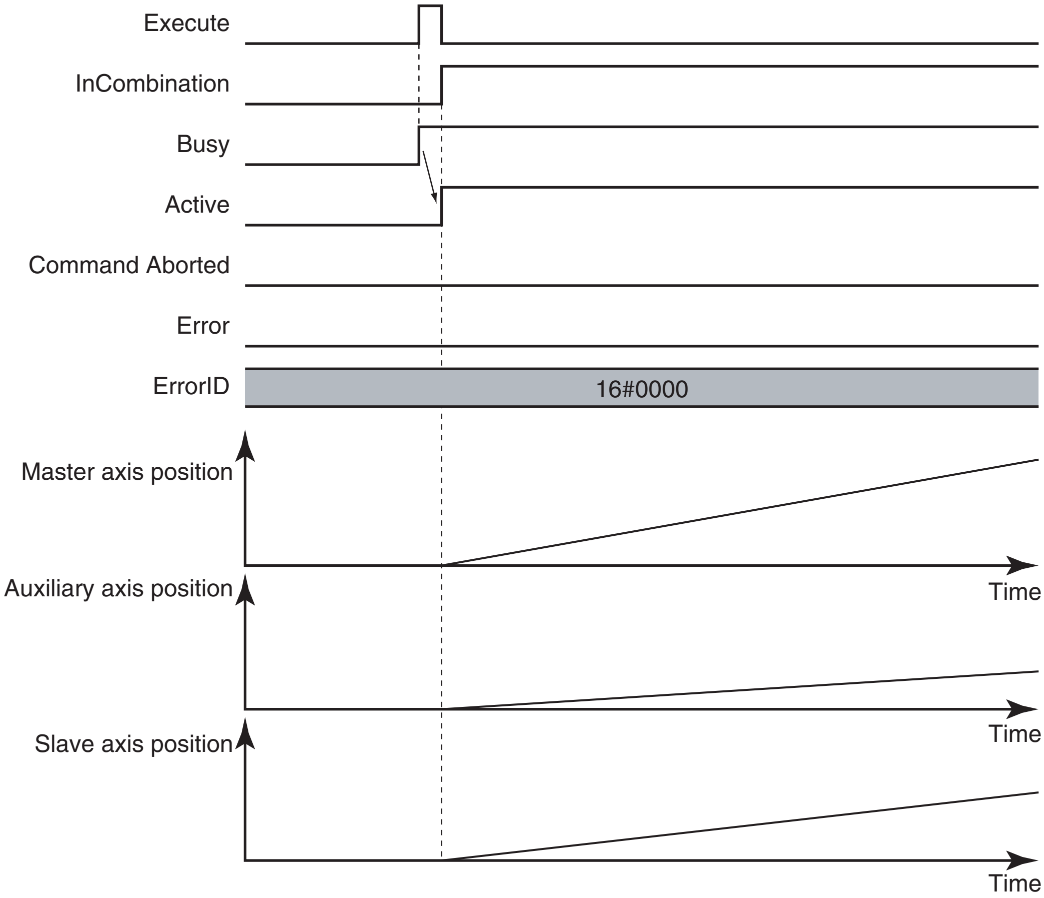
## Timing Charts

- *Busy* (Executing) changes to TRUE at the same time as *Execute* changes to TRUE. *Active* (Controlling) changes to TRUE in the next period.
- *InCombination* (Axes Combined) changes to TRUE in the period where the combined output starts.
- If another instruction aborts this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing), *Active* (Controlling), and *InCombination* (Axes Combined) change to FALSE.

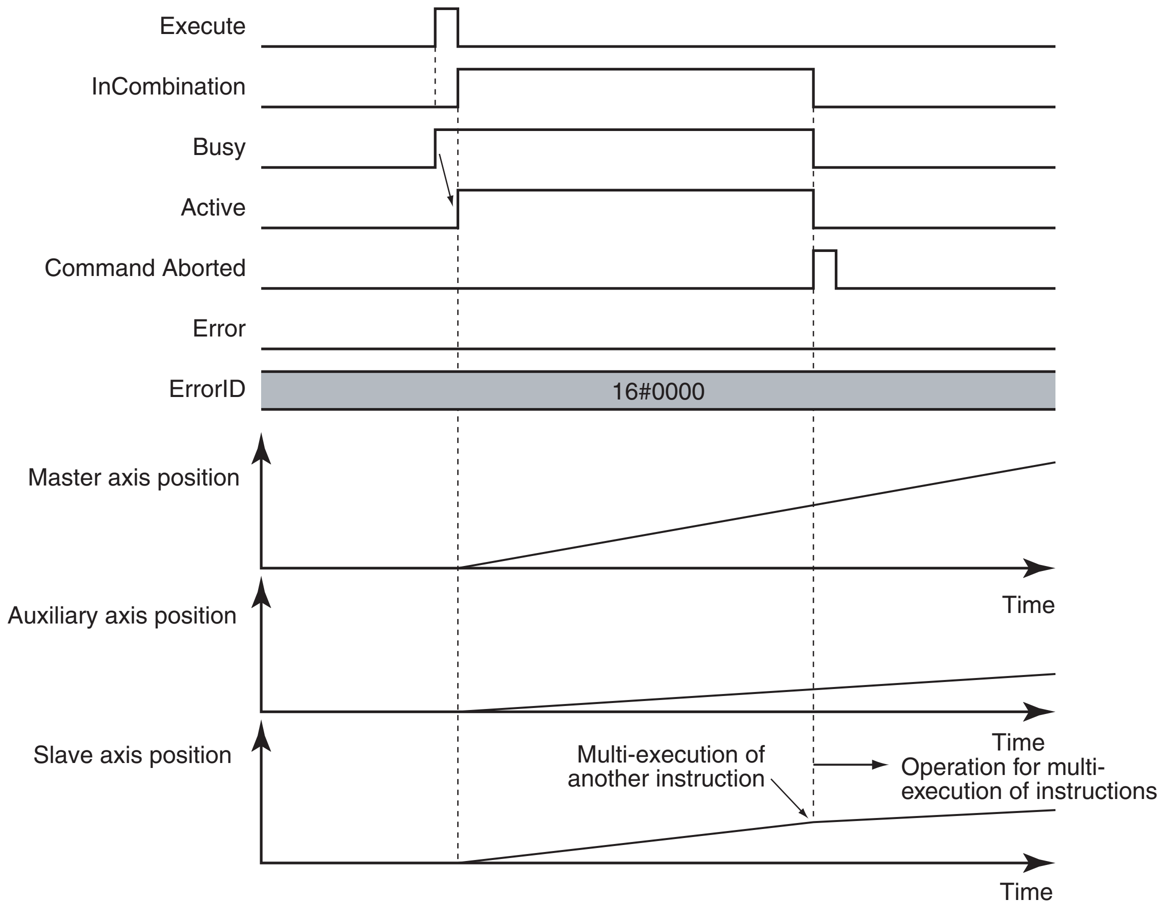
### *CombineMode* Set to *_mcAddAxes*



- ***CombineMode* Set to *_mcSubAxes***



### When the Instruction Is Aborted




## Re-execution of Motion Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Instructions

For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### Execution of Other Instructions during Instruction Execution

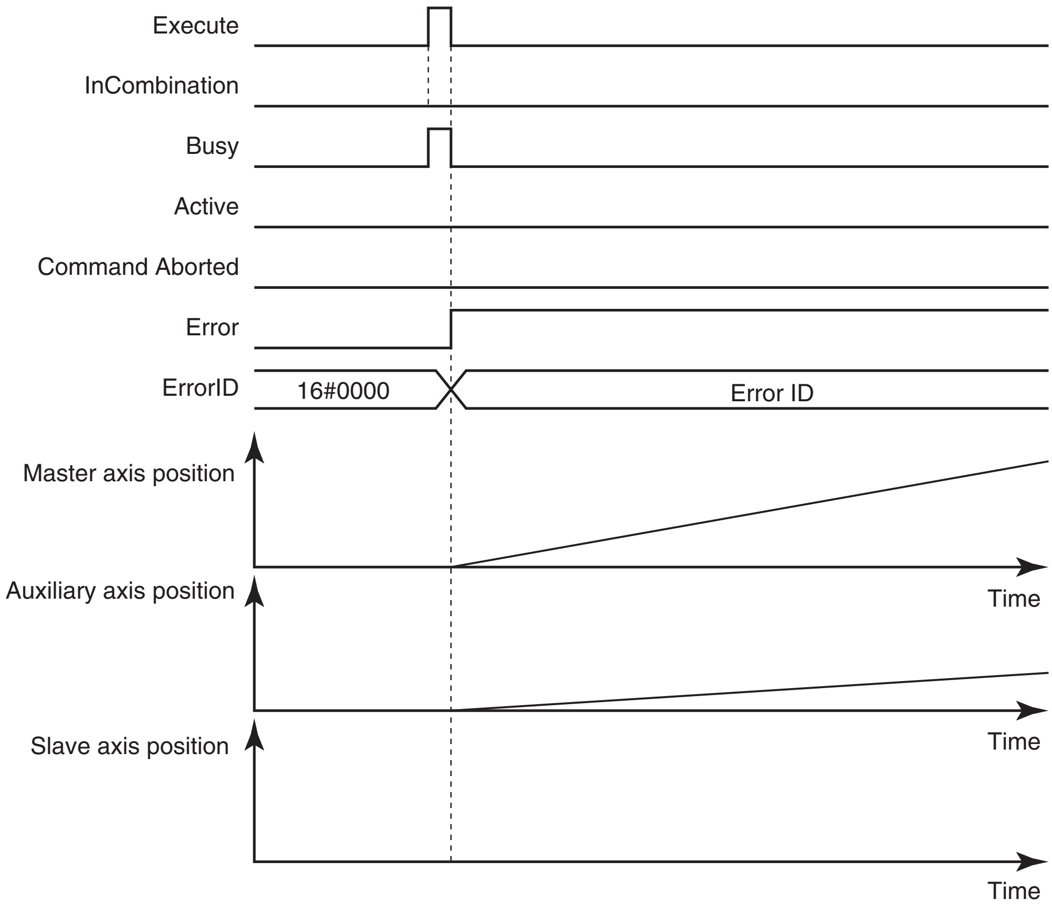
To use multi-execution of instructions for this instruction, specify the slave axis.
When performing multi-execution of another instruction while this instruction is in execution, the following limits apply depending on *BufferMode* (Buffer Mode Selection).

- You can execute another instruction with the Buffer Mode set to *Aborting* during execution of this instruction. You cannot specify *Buffered* or a blending mode.

## Errors

### ● Timing Chart When Error Occurs

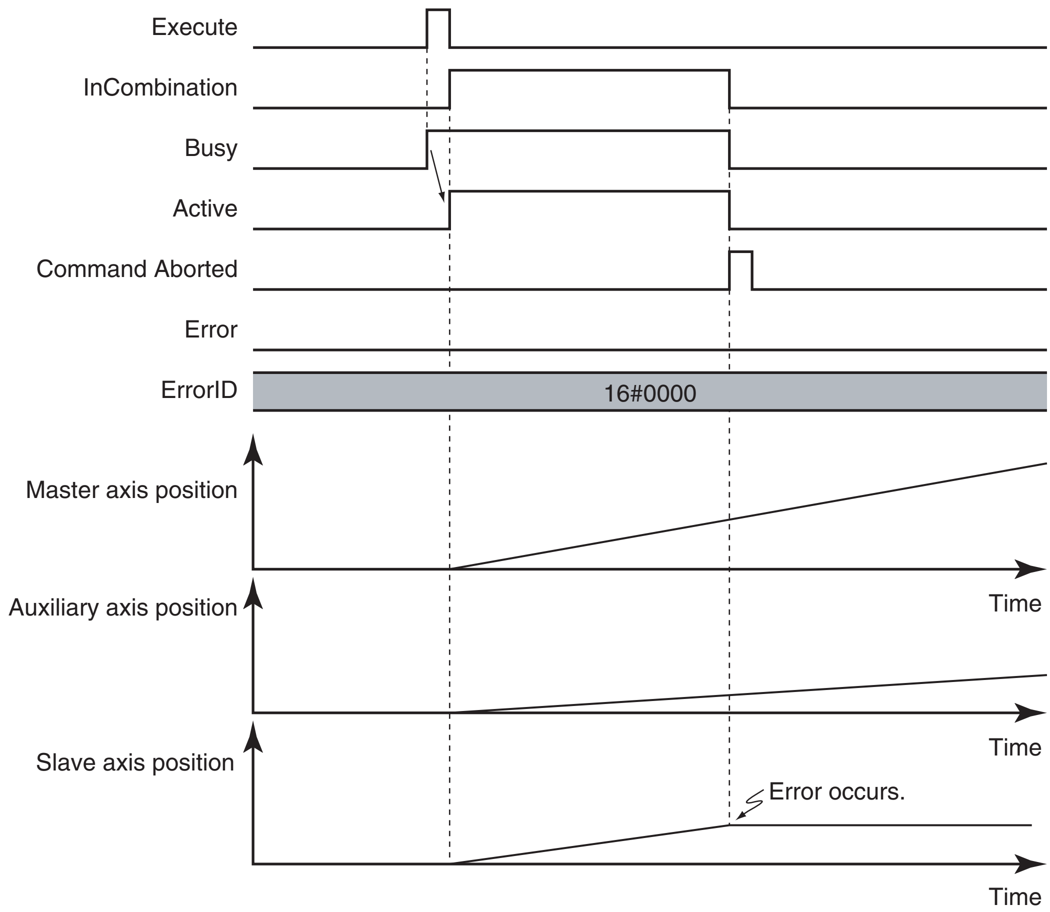
If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. The axis decelerates to a stop at the maximum deceleration rate that is set in the axis parameters. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).



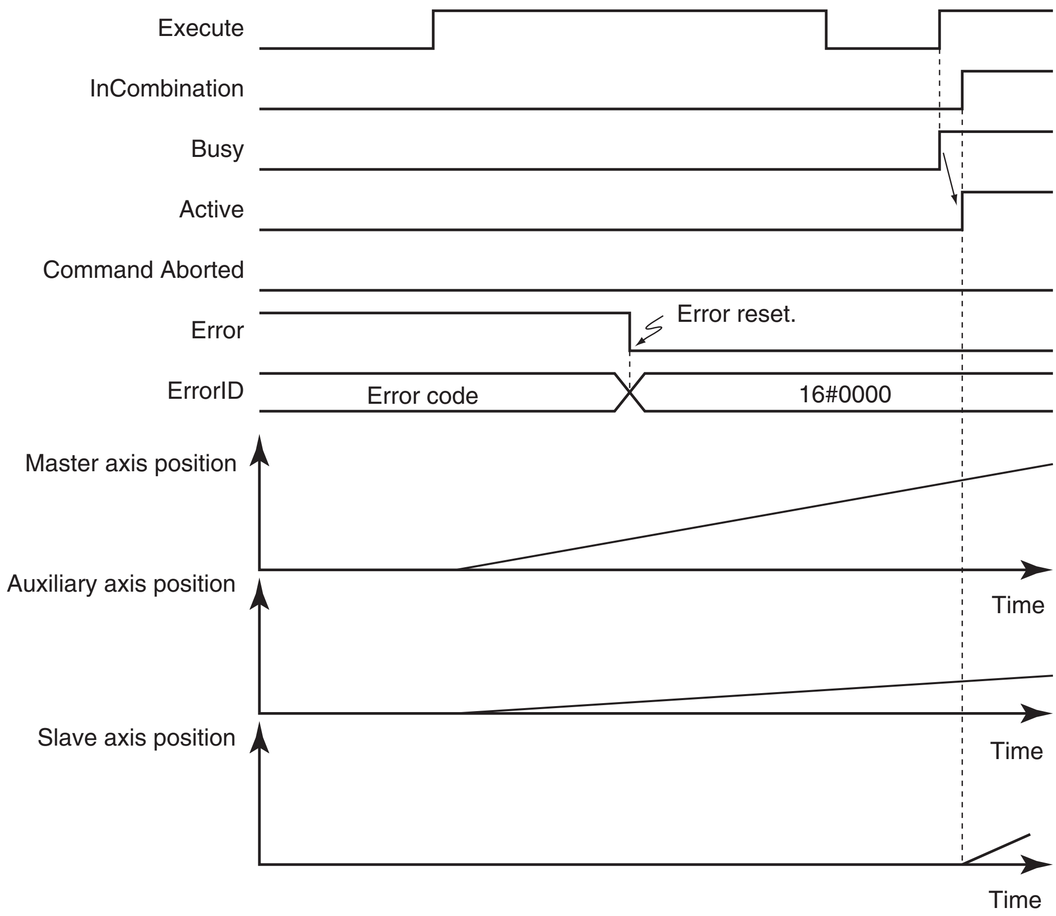

**Additional Information**

- This instruction is not affected by errors in the *Master* (Master Axis) or the *Auxiliary* (Auxiliary Axis).
- After the error is cleared and the *Master* (Master Axis) or the *Auxiliary* (Auxiliary Axis) is in motion, the *Slave* (Slave Axis) will resume the combined positioning operation. The *Master* (Master Axis) or the *Auxiliary* (Auxiliary Axis) are not affected if an error occurs for the slave axis during startup or execution of this instruction, but this instruction is aborted.

If a minor fault level error occurs during instruction execution, *CommandAborted* will change to TRUE and the axis will stop.
The axis decelerates to a stop at the maximum deceleration rate that is set in the axis parameters.
You can find out the cause of the error by referring to the value output to the *MFalutLvl.Code* Axis Variable for the *Slave* (Slave Axis).



If you clear the error for this instruction, the instruction will not start until *Execute* changes to TRUE again.

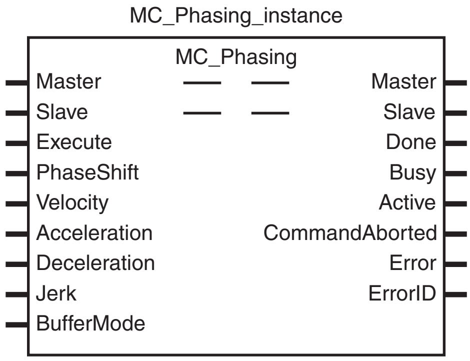



- **Error Codes**

Refer to *A-1 Error Codes* for instruction errors.

# MC_Phasing

The MC_Phasing instruction shifts the phase of the master axis currently in synchronized control.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_Phasing | Shift Master Axis Phase | FB |  | MC_Phasing_instance (<br>Master :=*parameter,*<br>Slave :=*parameter,*<br>Execute :=*parameter,*<br>PhaseShift :=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>BufferMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| PhaseShift | Phase Shift Amount | LREAL | Negative number, positive number, or 0 | 0 | Specify the master axis phase shift amount. The unit is command units. |
| Velocity | Target Velocity | LREAL | Positive number | 0 | Specify the target velocity. The unit is command units/s.*1 |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/$s^2$.*1 |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.*1 |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/$s^3$.*1 |
| BufferMode | Buffer Mode Selection | _eMC_BUFFER_MODE | 0: _mcAborting | 0*2 | Specify the operation when executing more than one motion instruction.<br>0: Aborting |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes.*

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When phase shift is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When phase shift is started. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Master | Master Axis | _sAXIS_REF | --- | Specify the master axis.* |
| Slave | Slave Axis | _sAXIS_REF | --- | Specify the slave axis.* |

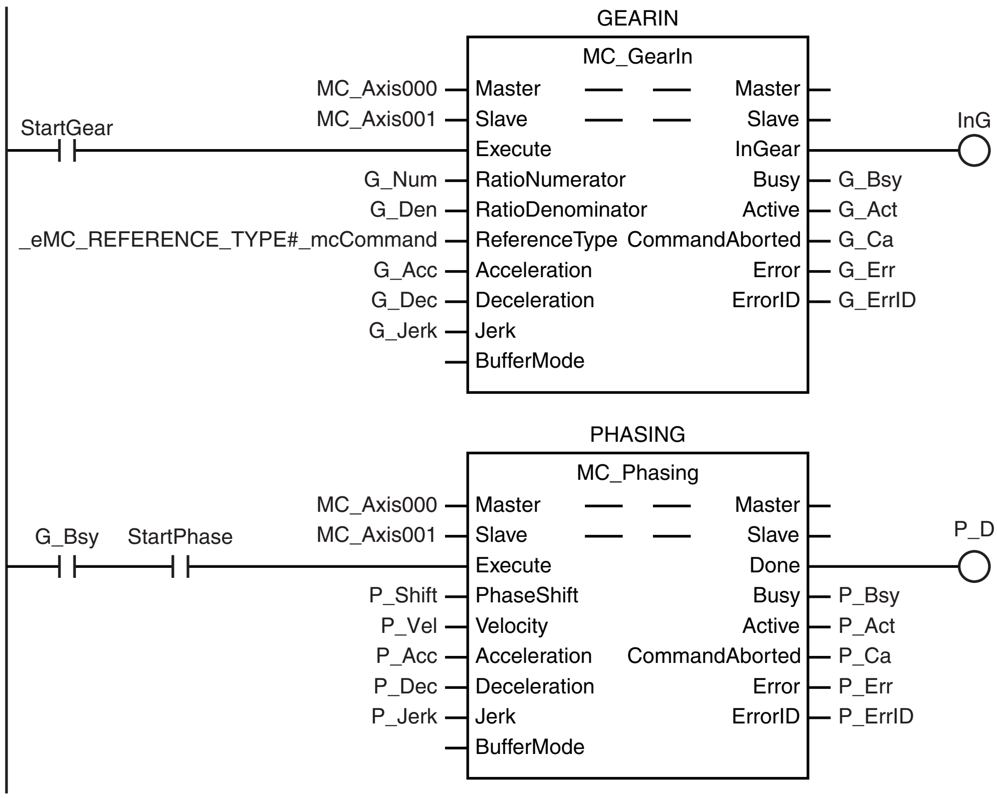
* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

**Precautions for Correct Use**

If you specify the same axis for the master axis and slave axis, a Master and Slave Defined as Same Axis minor fault (error code 5436 hex) will occur.

## Function

- Except during execution of the MC_CombineAxes instruction, if the MC_Phasing instruction is executed when single-axis synchronized control is in progress, the phase of the master axis is shifted according to the settings of the *PhaseShift* (Phase Shift Amount), *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate).
- The command current position and actual current position of the master axis do not change, and the relative shift between the command current position and actual current position of the master axis is taken as the phase of the master axis. The slave axis is synchronized to the shifted master axis phase.
- *Done* changes to TRUE when the *PhaseShift* (Phase Shift Amount) is reached.
- The shift is ended when the axis changed from single-axis synchronized control to another status. Even if the axis returns to single-axis synchronized control, the previous shift is not applied.
- You can shift the phase of the master axis for the following synchronization instructions: MC_CamIn (Start Cam Operation), MC_GearIn (Start Gear Operation), MC_GearInPos (Positioning Gear Operation), and MC_MoveLink (Synchronous Positioning).
- In the user program, place the MC_Phasing instruction after synchronized control instructions as shown below.




**Precautions for Correct Use**

Refer to *Precautions for Master and Auxiliary Axes in Synchronized Control* on page 1-7 for precautions on the master axis.

## Instruction Details

This section describes the instruction in detail.

### ● Specifying *Master* (Master Axis) and *Slave* (Slave Axis)

Specify axes variable names to specify the axes for *Master* (Master Axis) and *Slave* (Slave Axis). An axis specification error will occur if you specify a *Master* (Master Axis) or *Slave* (Slave Axis) that is not *InSync* or *InGear* (Gear Ratio Achieved) state.

### ● *PhaseShift* (Phase Shift Amount)

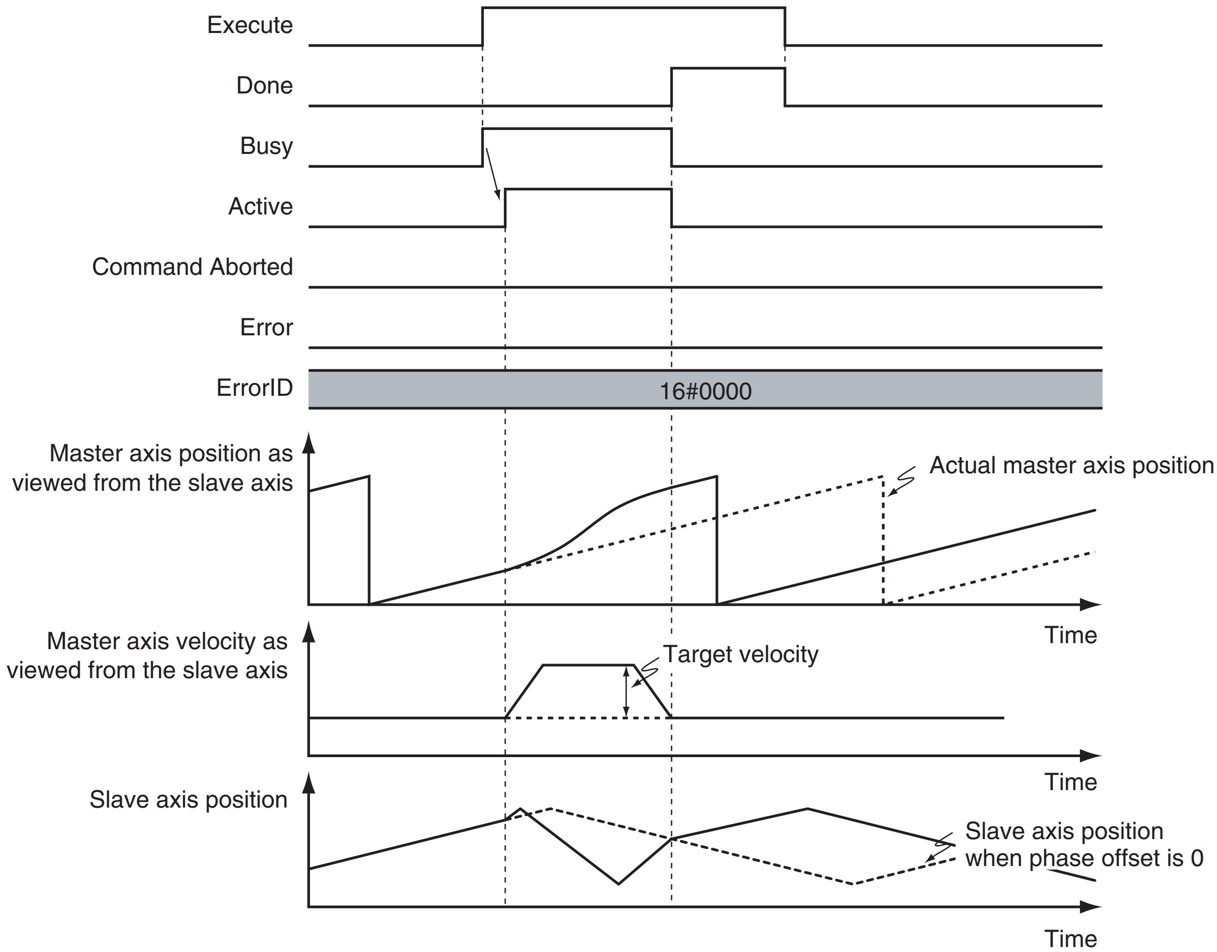
Set the phase shift amount as viewed from the *Slave* (Slave Axis) as the *PhaseShift* (Phase Shift Amount). Specify the phase shift about as a relative value.

### ● *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), *Deceleration* (Deceleration Rate), and *Jerk*

Set *Velocity*, *Acceleration*, *Deceleration*, and *Jerk* to specify the target velocity (shift velocity), acceleration rate, deceleration rate, and jerk for the phase shift amount.
The target velocity (shift velocity) as viewed from the *Slave* (Slave Axis) is the velocity relative to the *Master* (Master Axis) velocity.
The target velocity (shift velocity) of the *Master* (Master Axis) as viewed from the *Slave* (Slave Axis) is shown below as specified by the phase shift amount, acceleration rate, deceleration rate, and jerk.

### Example: Master Axis Phase Shift for an Electronic Cam

The interpolation velocity as viewed from the *Slave* (Slave Axis) is the velocity relative to the *Master* (Master Axis) velocity.




- If you set the phase shift amount to 0, the phase shift amount of the *Master* (Master Axis) will be 0 and the instruction ends normally.
- If you set the target velocity (shift velocity) to 0, a *Slave* (Slave Axis) error will occur because the value is out of range.
- The sum of the specified target velocity (shift velocity) and the *Master* (Master Axis) velocity can exceed the maximum velocity of the *Master* (Master Axis).

**Additional Information**

Error detection for the results of the MC_Phasing instruction is performed for the operation of the synchronized *Slave* (Slave Axis). Therefore, error detection is not performed for the settings of the *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate) variables when the MC_Phasing instruction is executed.

### ● *BufferMode* (Buffer Mode Selection)

*BufferMode* specifies how to join the axis motions for this instruction and the previous instruction. There is currently only the following setting.

| Buffer Mode Selection | Description |
|---|---|
| Aborting | If this instruction is re-executed, the shift for the instruction is started immediately. |

Reversing operation for multi-execution of instructions is performed according to the Operation Selection at Reversing setting for the master axis.

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

- **In-position Check**

When the phase shift operation is completed, phase shift output is ended and an in-position check is not performed.

## Re-execution of Motion Instructions

If you re-execute the instruction during instruction execution, you can change the *PhaseShift* (Phase Shift Amount), *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate). They are changed in the same way as for relative positioning.

## Multi-execution of Motion Instructions

For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

- **Execution of Other Instructions during Instruction Execution**

You can execute another instruction with the Buffer Mode set to *Aborting* for the *Slave* (Slave Axis) during execution of this instruction.
You cannot specify Buffered Mode or blending.

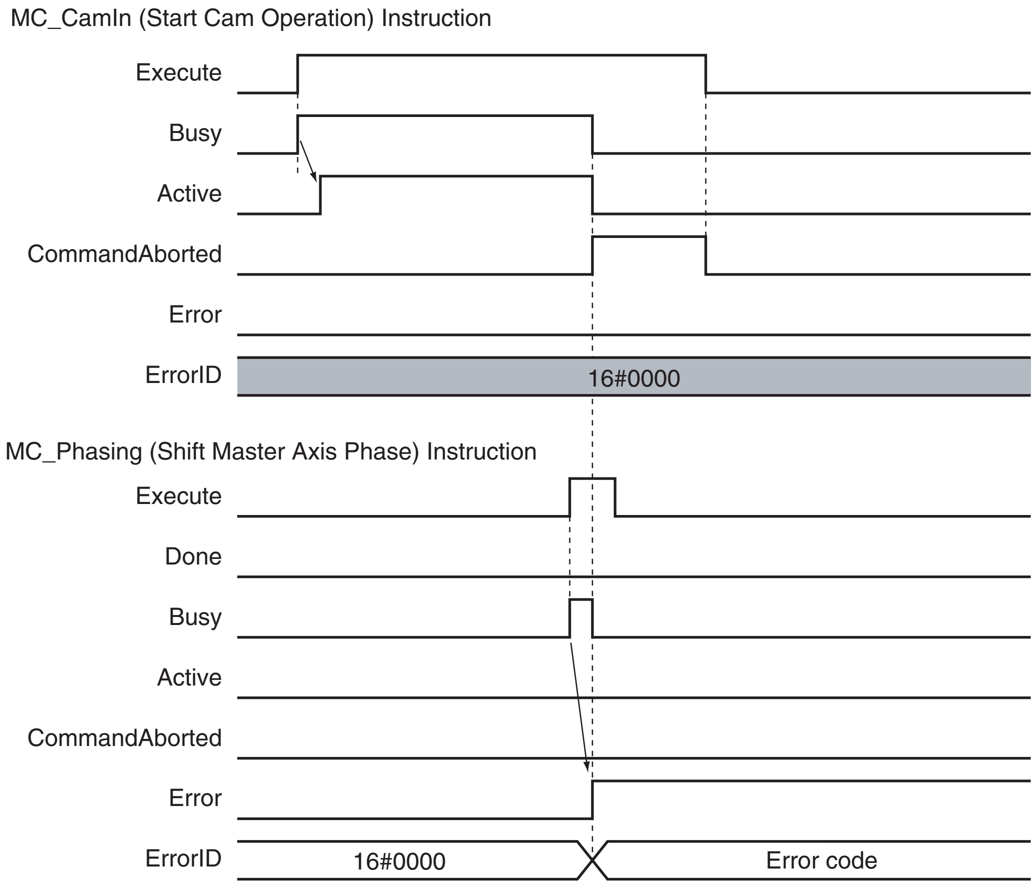
- **Multi-execution of MC_Phasing**

You can execute the MC_Phasing instruction even if the MC_Phasing instruction is already in execution for the specified slave axis.

## Error

If an error occurs during instruction execution, *Error* will change to TRUE and the slave axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code). *CommandAborted* of the MC_Phasing (Shift Master Axis Phase) instruction changes to TRUE when the *Slave* (Slave Axis) is released from synchronization due to an error.
The error state of the master axis does not affect the operation of this instruction.

### ● Timing Chart When Error Occurs




### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_TorqueControl

The MC_TorqueControl instruction uses the Torque Control Mode of the Servo Drive to control the torque.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_TorqueControl | Torque Control | FB | MC_TorqueControl_instance<br>MC_TorqueControl<br>Axis — Axis<br>Execute — InTorque<br>Torque — Busy<br>TorqueRamp — Active<br>Velocity — CommandAborted<br>Direction — Error<br>BufferMode — ErrorID | MC_TorqueControl_instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>Torque :=*parameter,*<br>TorqueRamp :=*parameter,*<br>Velocity :=*parameter,*<br>Direction :=*parameter,*<br>BufferMode :=*parameter,*<br>InTrque =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Torque | Target Torque | LREAL | 0 to 1000.0 | 300.0 | Specify the target torque to output to the Servo Drive in increments of 0.1%. Specify a percentage of the rated torque, i.e., the rated torque is 100.0%.*1 The unit is %. |
| TorqueRamp | Torque Ramp | LREAL | Non-negative number | 0 | Specify the change rate of torque from the current value to the target torque. The unit is %/s. |
| Velocity | Velocity Limit | LREAL | Non-negative number | 0 | Specify the target velocity. The unit is command units/s.*2 |
| Direction | Direction | _eMC_ DIRECTION | 0: _mcPositiveDirection<br>2: _mcNegativeDirection | 0*3 | Specify the direction of the target torque.<br>0: Positive direction<br>2: Negative direction |
| BufferMode | Buffer Mode Selection | _eMC_BUFFER _ MODE | 0: _mcAborting<br>1: _mcBuffered | 0*3 | Specify the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered |

*1 If a value that is higher than 1,000.0% is specified, it will be treated as 1,000.0%. If a negative value is specified, it will be treated as 0.0%.

*2 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*3 The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| InTorque | Target Torque Reached | BOOL | TRUE or FALSE | TRUE when the target torque is reached. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes.*

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| InTorque | When target torque is output. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE.<br>• When the instruction is re-executed and the target torque is changed. |
| Busy | When *Execute* changes to TRUE. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When output of the torque command value starts. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

MC_TorqueControl

3

Variables

## Function

- The MC_TorqueControl instruction controls the output torque of the Servomotor by directly specifying the torque command value.
- Set the target torque in increments of 0.1%. If the second decimal place is specified, it will be rounded off.
- Use MC_Stop instruction to stop the execution of this instruction.
- The Torque Control Mode of the Servo Drive is used to perform torque control.
- The previous Control Mode is maintained until it is changed.

  **Example:** Changing from position control to torque control: Position control is performed until the Servo Drive changes to torque control.
  Changing from torque control to position control: Torque control is performed until the Servo Drive changes to position control.

### ● Mapping Data Objects

To use the MC_TorqueControl (Torque Control) instruction, map the following object data in the Detailed Settings Area of the Axis Basic Settings Display of the Sysmac Studio.

- Target torque (6071 hex)
- Modes of operation (6060 hex)
- Torque actual value (6077 hex)
- Modes of operation display (6061 hex)

If even one of the required objects is not set, a Process Data Object Setting Missing error (error code 3461 hex) occurs.

For details on mapping data objects, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Instruction Details

This section describes the instruction in detail.

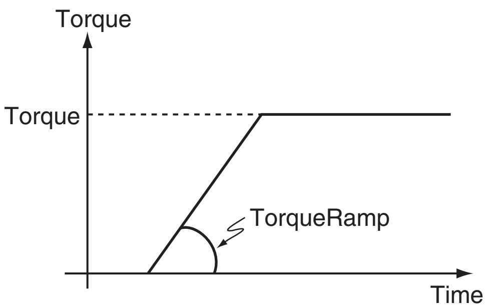
### ● Specifying *Axis*

*Axis* specifies the axis for torque control.

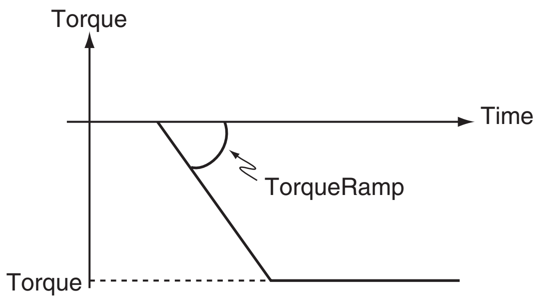
### ● *TorqueRamp*

Specify the slope from the currently specified command torque until the target torque is output.

**Example 1: Direction = Positive**




**Example 2: Direction = Negative**



**Precautions for Correct Use**

Set the target torque so that the maximum torque of the motor is not exceeded.
The operation that is performed when the maximum torque of the motor is exceeded depends on the Servo Drive.

### ● *Velocity* (Velocity Limit)

*Velocity* limits the maximum velocity of the axis during torque control. When the axis velocity reaches this velocity limit, the Servo Drive reduces the torque to reduce the axis velocity. The velocity limit function uses the Servo Drive function.

For details, refer to information on the torque control function in the Servo Drive manual.

**Precautions for Correct Use**

- The axis velocity increases faster during torque control. Make sure that you set Velocity (velocity limit) for safety.
- When you use an OMRON G5-series Servo Drive, set the Velocity Limit Selection (3317 hex) of the Servo Drive to 1 (velocity limit value via EtherCAT communications). Otherwise, the velocity limit is not affected. Also, the axis does not stop even if the limit input signal turns ON.
- Process data 607F hex is used for the velocity limit value. When you use an OMRON G5-series Servo Drive, set the advanced settings in the Axis Parameter Settings of the Sysmac Studio to use the Velocity Limit Value (607F hex). To use a velocity limit with a servo drive from another manufacturer, refer to the manual for the servo drive.

### ● *Direction*

*Direction* specifies the direction to output the target torque. If you want to output torque in the positive direction of the axis, set the positive direction. If you want to output torque to the negative direction of the axis, set the negative direction.

### ● *BufferMode* (Buffer Mode Selection)

*BufferMode* specifies how to join the axis motions for this instruction and the previous instruction. There are two possible settings.

| Buffer Mode Selection | Description |
|---|---|
| Aborting | Aborts the instruction being executed and switches to this instruction. |
| Buffered | Buffers this instruction and executes it automatically after the current instruction is completed. |

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Stopping Axes during Torque Control

If MC_Stop is executed during MC_TorqueControl execution for an OMRON G5-series Servo Drive, the deceleration rate that is specified for the MC_Stop instruction is not used and an immediate stop is performed.
An immediate stop is performed even for errors that normally result in deceleration stops.

### ● Command Position and Actual Position during Torque Control

The following current positions are given in the system-defined variables for motion control during torque control for this instruction.

Actual current position: Contains the value returned by the Servo Drive multiplied by the gear ratio.

Command current position: Contains the actual current position from the previous period.

## Applicable Axes and Execution Condition

- For a servo axis, this instruction is ready for execution as soon as *Enable* for the MC_Power (Power Servo) instruction changes to TRUE (Servo ON).
- A virtual servo axis will acknowledge this instruction at any time. However, processing to switch the Control Mode of the Servo Drive is not performed.
- An error occurs if the instruction is executed for an encoder or virtual encoder axis.

## Operation When Servo Turns OFF

Processing to change to CSP Mode is performed by the MC Function Module when the Status output variable from the MC_Power (Power Servo) instruction changes to FALSE.
However, for an OMRON G5-series Servo Drive, commands to change the Controller Mode are not acknowledged from the MC Function Module when the Servo is OFF.

## Axis Variable Status

*Status.Continuous* (Continuous Motion) in the Axis Variable status changes to TRUE. Also, CST (Cyclic Synchronous Torque (CST) Control Mode) in *DrvStatus* (Servo Drive Status) in the Axis Variable changes to TRUE.

## Home Status

Home remains defined.

## Software Limits

The software limits are applied.
They are applied even if one of the following is selected in the axis parameter: Deceleration stopping enabled for command position or Immediate stopping enabled for command position (stop using remaining pulses).

## When Count Mode Is Set to Linear Mode

The operation for underflows and overflows is the same as for operations that do not have target positions.

## Operation at Reversing

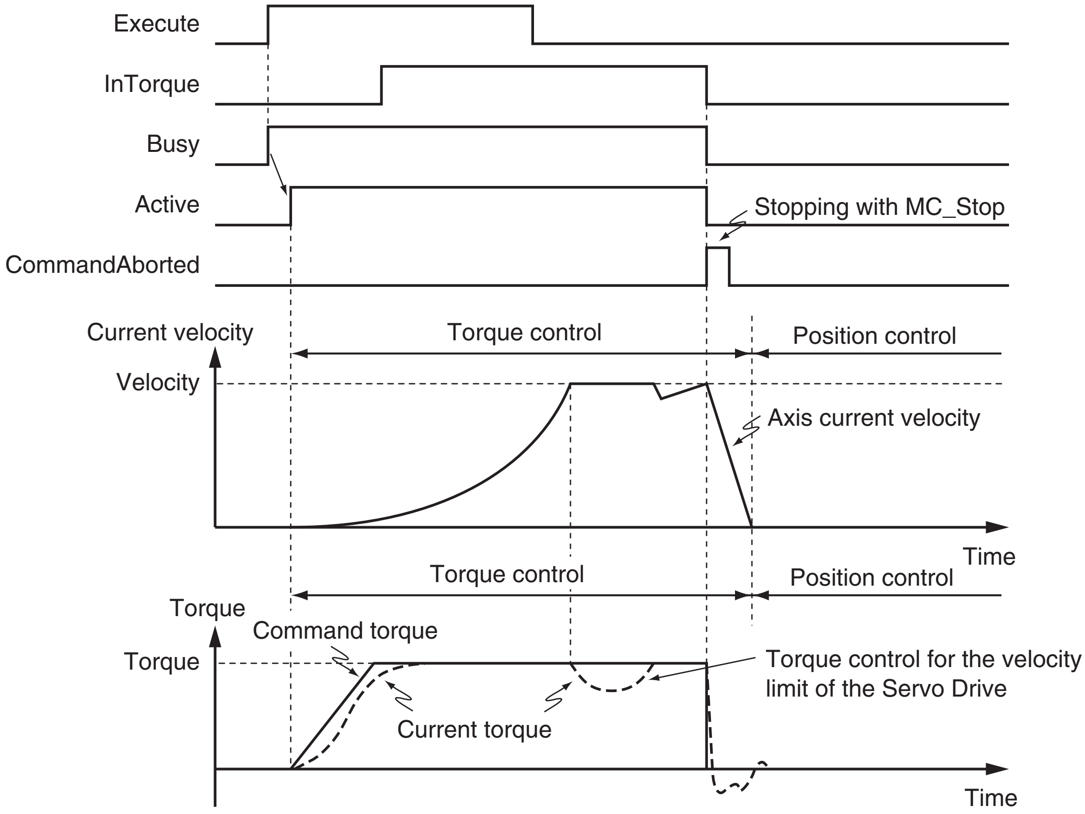
If re-execution is performed and the torque command value is reversed, operation is performed with *TorqueRamp* from this command and not with the setting of the Operation Selection at Reversing axis parameter.

The operation for reversing for multi-execution of instructions is as follows:

- If the command position reverses for multi-execution of an instruction that uses CSP during execution of this instructions, the operation at reversing is performed according to the Operation Selection at Reversing axis parameter.
- If the torque command value reverses when multi-execution of this instruction is performed during execution of an instruction that uses CSP or CSV, the torque command reverses according to *TorqueRamp*.
- If the torque command value reverses when multi-execution of this instruction is performed during execution of this instruction, the torque command reverses according to *TorqueRamp*.

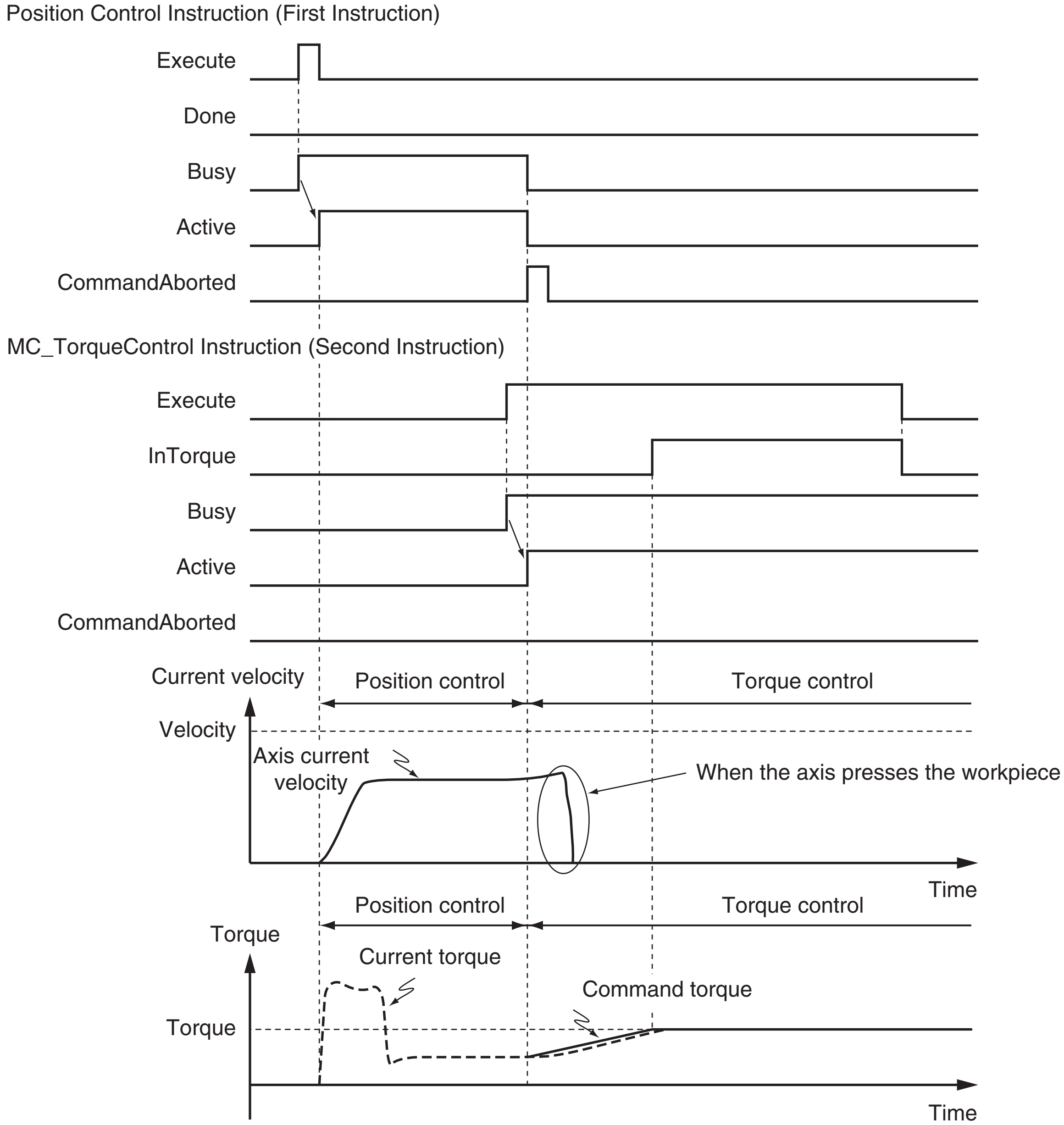
## Timing Charts

### Starting and Stopping the Instruction



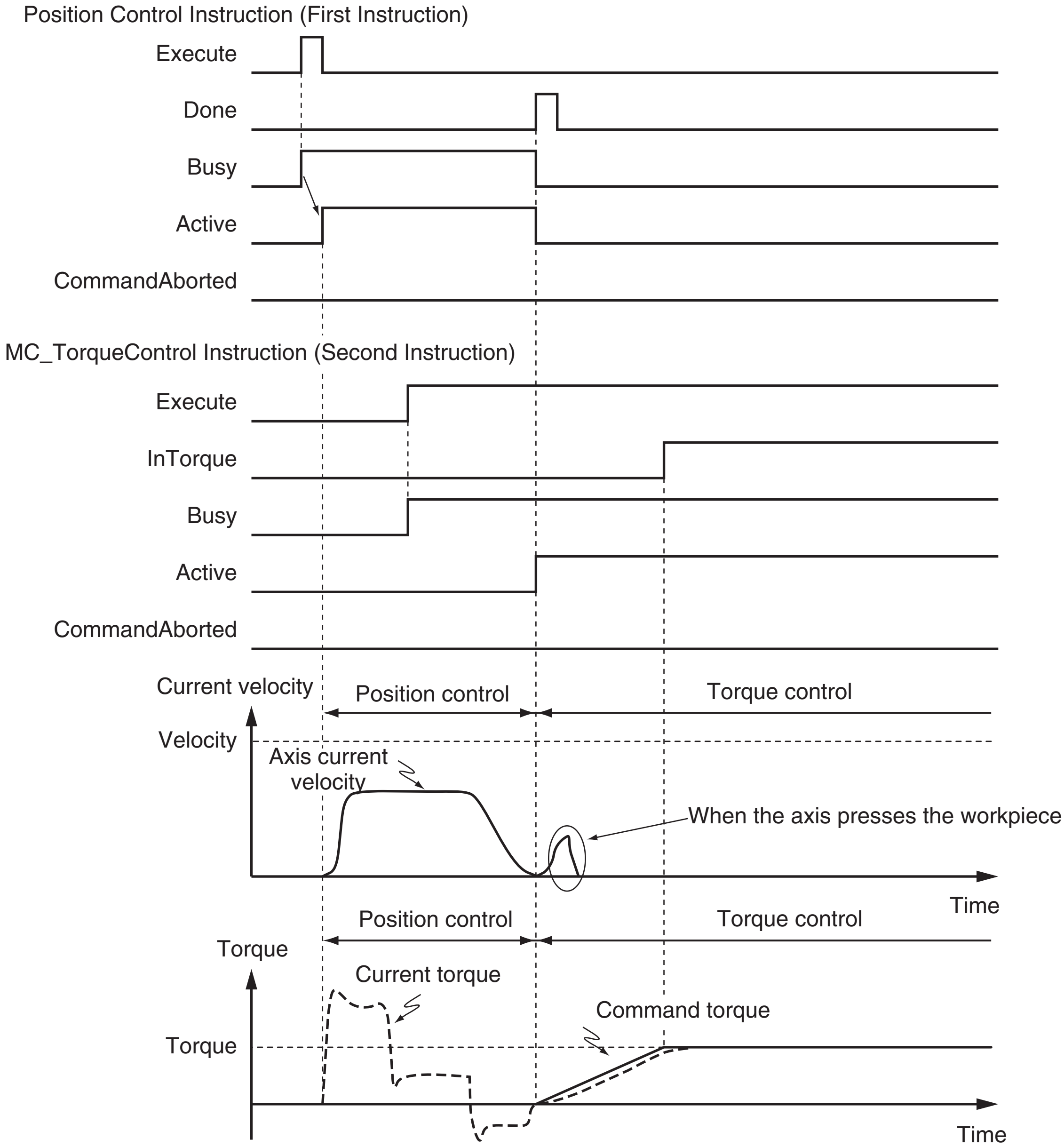
### Instruction Execution to Abort Immediately Preceding Operation

The following timing chart shows an application in which the axis stops and pressing is performed while this instruction is in execution.



### Instruction Execution for Buffered during Immediately Preceding Operation

The following timing chart shows an application in which the axis stops and pressing is performed while this instruction is in execution.




## ● Changing the Control Mode

- If you execute the MC_TorqueControl instruction while a position control instruction, such as the MC_MoveAbsolute (Absolute Positioning) or MC_MoveRelative (Relative Positioning) instruction, is in execution, the operation depends on the setting of the *BufferMode* (Buffer Mode Selection) of the MC_TorqueControl instruction.
  If *BufferMode* is set to *Aborting*, the Control Mode changes to Torque Control as soon as the instruction is executed. If the Buffer Mode is set to *Buffered*, the Control Mode changes to Torque Control after the previous operation is completed.
- If the MC_TorqueControl instruction is aborted by other instructions such as MC_MoveAbsolute (Absolute Positioning), or if an axis error occurs, the Control Mode changes to Position Control at that point.
- *Active* (Controlling) changes when the instruction is executed, but it takes several periods for the Control Mode in the Servo Drive to change. The time that is required for the Control Mode to change depends on the Servo Drive.

### Criteria for Changing the Control Mode

When you stop an axis for an OMRON G5-series Servo Drive, the MC Function Module sets the Velocity Limit Value (607F hex) to 0. The Control Mode is changed to CSP Mode when the following criterion is met for three consecutive primary period tasks after that.

Actual current velocity ≤ Maximum velocity × 0.1

With a servo drive from another manufacturer, the Control Mode of the Servo Drive changes from CST to CSP Mode and the Servo is turned ON at the actual current position when the mode changes.

### Failure to Change the Control Mode

If the Servo Drive does not complete switching the Control Mode within 1 second after a Control Mode switch command is sent from the MC Function Module, an Error in Changing Servo Drive Control Mode (error code: 7439 hex) occurs and the Servo is turned OFF, i.e., a free-run stop occurs.
For details on the Error in Changing Servo Drive Control Mode (error code: 7439 hex), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

If the criteria for changing is not met within 10 seconds after the Velocity Limit Value is set to 0, the Servo is turned OFF in the same was as given above.

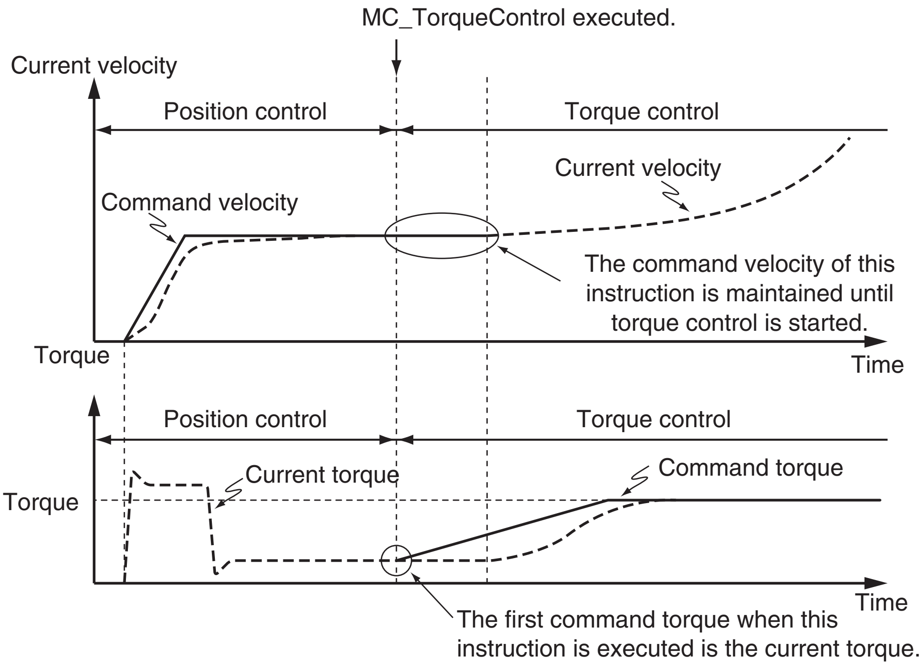
### Operation Examples for Changing the Control Mode

The relationship between the command torque and command velocity until the Control Mode changes is described in the following examples where the Control Mode is changed during axis operation.

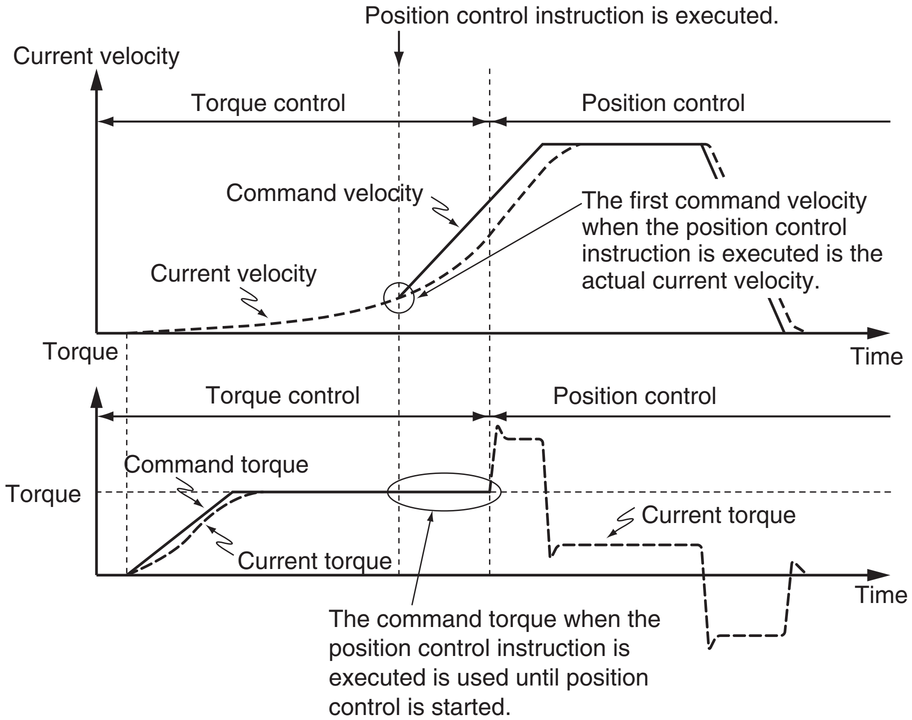
**Precautions for Correct Use**

An error will occur in some Servo Drives if the Control Mode in the Servo Drive changes during axis motion.

### Changing from Position Control to Torque Control



### Changing from Torque Control to Position Control




## Re-execution of Motion Instructions

You can change the operation of the instruction if you change the input parameter during torque control and then change *Execute* to TRUE again. You can change the input variables *Torque* (Target Torque), *TorqueRamp*, and *Velocity* (Velocity Limit) by re-executing the motion control instruction. When the motion control instruction is re-executed to change *Torque* (Target Torque), *InTorque* (Target Torque Reached) operates for the new target torque that was set at re-execution. For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Multi-execution of Motion Instructions

For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

You can switch to this instruction or buffer this instruction if you execute it during execution of another instruction. You can buffer one instruction per axis. Specify the operation of this instruction by using *BufferMode* (Buffer Mode Selection) for multi-execution of instructions.

| Buffer Mode Selection | Description |
|---|---|
| Aborting | Aborts the instruction being executed and switches to this instruction. |
| Buffered | Buffers this instruction and executes it automatically after the current instruction is completed. |

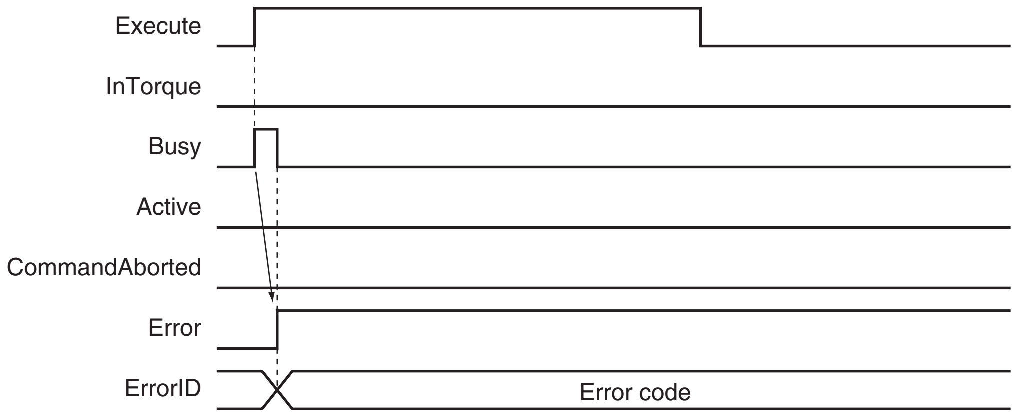
For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution of Other Instructions during Instruction Execution

If another instruction is executed during execution of this instruction, the *BufferMode* input variable to the other instruction must be set to *Aborting* or *Buffered*.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

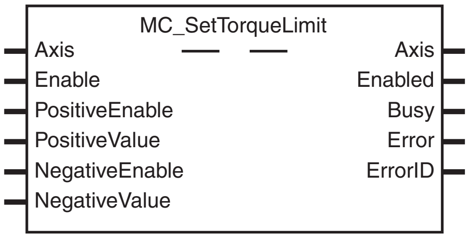



### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_SetTorqueLimit

The MC_SetTorqueLimit instruction limits the torque output from the Servo Drive through the torque limit function of the Servo Drive.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_SetTorqueLimit | Set Torque Limit | FB | MC_SetTorqueLimit_instance<br> | MC_SetTorqueLimit_instance (<br>Axis :=*parameter,*<br>Enable :=*parameter,*<br>PositiveEnable :=*parameter,*<br>PositiveValue :=*parameter,*<br>NegativeEnable :=*parameter,*<br>NegativeValue :=*parameter,*<br>Enabled =>*parameter,*<br>Busy =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Enable | Enable | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Enable* is TRUE. |
| PositiveEnable | Positive Direction Enable | BOOL | TRUE or FALSE | FALSE | TRUE: Enables the positive torque limit.<br>FALSE: Disables the positive torque limit. |
| PositiveValue | Positive Torque Limit | LREAL | 0.1 to 1000.0 or 0.0 | 300.0 | Set the torque limit in the positive direction in increments of 0.1%.<br>If a value that exceeds the Maximum Positive Torque Limit axis parameter, the positive torque will be the Maximum Positive Torque Limit.<br>The value will be 0 if 0 or a negative value is specified. |
| NegativeEnable | Negative Direction Enable | BOOL | TRUE or FALSE | FALSE | TRUE: Enables the negative torque limit.<br>FALSE: Disables the negative torque limit. |
| NegativeValue | Negative Torque Limit | LREAL | 0.1 to 1000.0 or 0.0 | 300.0 | Set the torque limit in the negative direction in increments of 0.1%.<br>If a value that exceeds the Maximum Negative Torque Limit axis parameter, the negative torque will be the Maximum Negative Torque Limit.<br>The value will be 0 if 0 or a negative value is specified. |

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Enabled | Enable | BOOL | TRUE or FALSE | TRUE when control is in progress.*1 |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | *2 | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

*1 *Enabled* indicates the status of the instruction. It does not indicate the status of torque control by the Servo Drive.

*2 Refer to *A-1 Error Codes.*

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Enabled | • When *Enable* changes to TRUE.<br>• When MC_Power is being executed. | • One period after *Enable* changes to FALSE.<br>• When *Error* changes to TRUE.<br>• When *Enable* for MC_Power instruction changes to FALSE. |
| Busy | When *Enable* changes to TRUE. | • When *Error* changes to TRUE.<br>• When *Enable* changes to FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

- The MC_SetTorqueLimit instruction sets the torque limits that are used by the Servo Drive.
- When Enable is TRUE and *PositiveEnable* (Positive Direction Enable) changes to TRUE, control is performed with *PositiveValue* (Positive Torque Limit).
  When Enable is TRUE and *NegativeEnable* (Negative Direction Enable) changes to TRUE, control is performed with *NegativeValue* (Negative Torque Limit).
- When *PositiveEnable* (Positive Direction Enable) changes to FALSE, the value of the Maximum Positive Torque Limit is set in the Servo Drive.
  When *NegativeEnable* (Negative Direction Enable) changes to FALSE, the value of the Maximum Negative Torque Limit is set in the Servo Drive.
- When *Enable* to this instruction changes to FALSE, the values of the Maximum Positive Torque Limit and Maximum Negative Torque Limit are set in the Servo Drive. At the same time, *Busy* (Executing) and *Enabled* change to FALSE.
- The torque limits are set as a percentage of the motor torque in 0.1% increments. If the second decimal place is specified, it will be rounded off.

**Precautions for Correct Use**

Set the Maximum Positive Torque Limit and Maximum Negative Torque Limit axis parameters to the upper limits of torque control for your Servo Drive.

## Mapping Data Objects

To use the MC_SetTorqueLimit instruction, map the following object data in the Detailed Settings Area of the Axis Basic Settings Display of the Sysmac Studio.

- Positive torque limit value (60E0 hex)
- Negative torque limit value (60E1 hex)

If even one of the required objects is not set, a Process Data Object Setting Missing error (error code 3461 hex) occurs.

**Precautions for Correct Use**

- If you use a servo drive from a different manufacturer, set the servo drive so that the positive torque limit value (PDO 60E0 hex) and the negative torque limit value (PDO 60E1 hex) are used as the torque limits.
  Refer to the manual for your servo drive for the setting procedure.
- This instruction cannot be used for servo drives from other manufacturers unless the positive torque limit value and the negative torque limit value can be mapped to PDOs.
  If they cannot be set to PDOs, use the support software of the manufacturer or SDO communications to set the torque limits.

For details on mapping data objects, refer to *2-3 PDO Mapping* and to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Changing the Input Parameters

The following input parameters are continuously updated as long as *Enable* is TRUE.

- *PositiveEnable* (Positive Direction Enable)
- *NegativeEnable* (Negative Direction Enable)
- *PositiveValue* (Positive Torque Limit)
- *NegativeValue* (Negative Torque Limit)

## Relation to Holding Operation of the MC_Home Instruction for OMRON G5-series Servo Drives

Torque limits that are set in the Servo Drive in advance are used for the Proximity Reverse Turn/Holding Time (12) or No Home Proximity Input/Holding Home Input (13) Homing Operation Modes to automatically start torque control in the holding direction.

**Precautions for Correct Use**

The automatic torque limit function of the MC_Home instruction is not used for servo drives from other manufacturers.
Use the MC_SetTorqueLimit instruction, SDO communications, or the Support Software for the Servo Drive to set suitable values.

**Additional Information**

- The torque limits are continued even after a normal completion of homing.
- The torque limits are automatically released when an instruction that moves the axis in the opposite direction is executed.

For details on homing, refer to *MC_Home* on page 3-16.

## Settings for OMRON G5-series Servo Drives

To use this instruction, you must use the Sysmac Studio of the servo drive to set the Torque Limit Selection (3521 hex) in the G5-series Servo Drive.

- Set the Torque Limit Selection to 6 to apply a torque limit in the home input detection direction during the holding operation for homing and to use the torque limit directions and values that are set with this instruction for other operation.
  In that case, the values of the input variables to this instruction are ignored during the holding operation for homing.
- If the Torque Limit Selection to 4, the values of the input variables to this instruction are always used. You must set torque limits that are suitable for both for the holding operation during homing and for other operations.

| | | Torque Limit Selection (3521 hex) | |
|---|---|---|---|
| | | **6 (recommended)** | **4** |
| Positive Torque Limit | Homing | Torque Limit 3 (3525 hex) is used. | The smaller of the *PositiveValue* (Positive Torque Limit) for this instruction and Torque Limit 1 (3013 hex) is used. |
| | Operations other than Homing | The smaller of the *PositiveValue* (Positive Torque Limit) for this instruction and Torque Limit 1 (3013 hex) is used. | |
| Negative Torque Limit | Homing* | Torque Limit 4 (3526 hex) is used. | The smaller of the *NegativeValue* (Negative Torque Limit) for this instruction and Torque Limit 2 (3522 hex) is used. |
| | Operations other than Homing | The smaller of the *NegativeValue* (Negative Torque Limit) for this instruction and Torque Limit 2 (3522 hex) is used. | |

* Until the torque limit is automatically released.

Refer to the *G5-series AC Servomotors/Servo Drives with Built-in EtherCAT Communications User's Manual* (Cat. No. I576) for details.

## Relationship to the MC_TorqueControl Instruction

The MC_SetTorqueLimit and the MC_TorqueControl instructions can be used together.

## Axes in Axes Groups

This instruction can be used for an axis in an enabled axes group.

## Relation to CPU Unit Operating Modes

The values that are set with this instruction in RUN mode are also used after the operating mode changes to PROGRAM mode.

## Applicable Axes and Execution Condition

- You can use this instruction for a servo axis.
  The status of the *Enabled* output variable from this instruction, however, depends on the status of the Servo.

| | Servo ON | Servo OFF |
|---|---|---|
| Status of *Enabled* | TRUE | FALSE* |

* If *Enabled* from this instruction is FALSE, the torque limits do not function on the Servo Drive.

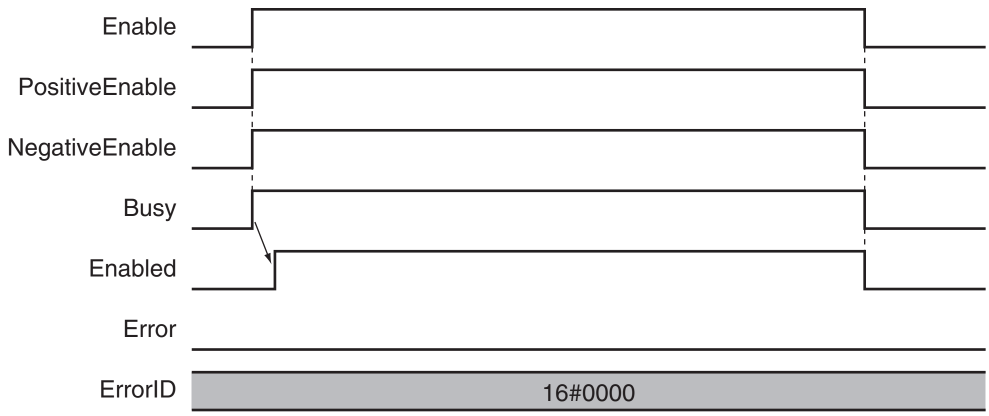
- This instruction is acknowledged for a virtual servo axis, but torque is not limited.
- An error occurs if the instruction is executed for an encoder or virtual encoder axis.

- **Axis Variable Status (Servo Drive Status)**

When the internal limit function in the Servo Drive is in operation, *ILA* (Drive Internal Limiting) in the *DrvStatus* (Servo Drive Status) in the Axis Variable is TRUE.
This variable gives an OR of the following four: torque limits, velocity limit, drive prohibit inputs, and software limits.

## Timing Charts

The following chart shows the timing of the torque limits.




## Re-execution of Motion Control Instructions

You cannot re-execute motion instructions with *Enable* inputs.

## Multi-execution of Motion Control Instructions

For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

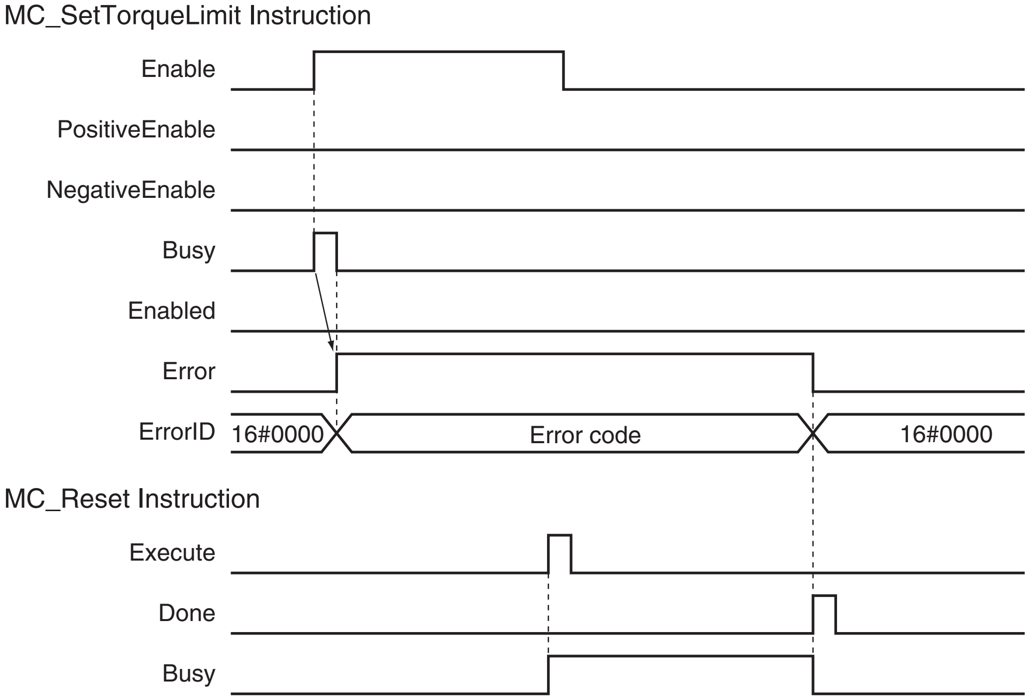
- **Multi-execution of MC_SetTorqueLimit Instructions**

If an instance of this instruction is executed during execution of another instance, the instance that is executed last takes priority.
*Enabled* will be TRUE for both instructions.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop.
You can find out the cause of the error by referring to the value output to *ErrorID* (Error Code).

## Timing Chart When Error Occurs

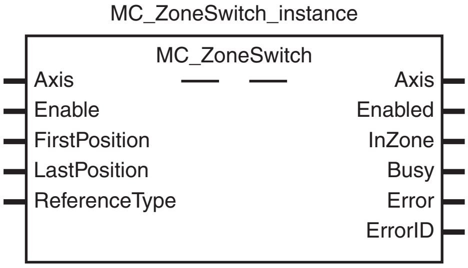



## Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_ZoneSwitch

The MC_ZoneSwitch instruction determines if the command current position or actual current position of an axis is within a specified zone.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_ZoneSwitch | Zone Monitor | FB |  | MC_ZoneSwitch_instance (<br>Axis :=*parameter,*<br>Enable :=*parameter,*<br>FirstPosition :=*parameter,*<br>LastPosition :=*parameter,*<br>ReferenceType :=*parameter,*<br>Enabled =>*parameter,*<br>InZone =>*parameter,*<br>Busy =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Enable | Enable | BOOL | TRUE or FALSE | FALSE | The instruction is executed while *Enable* is TRUE. |
| FirstPosition | First Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the first position of the zone range.*1<br>The unit is command units.*2 |
| LastPosition | Last Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the last position of the zone range.*3<br>The unit is command units.*1 |
| RererenceType | Position Type Selection | _eMC_REFERENCE _TYPE | 0: _mcCommand<br>1: _mcFeedback | 0*4 | Specify the axis information to monitor.<br>0: Command position (value calculated in the previous primary period)<br>1: Actual position (value obtained in the same primary period) |

*1 Set a value that is smaller than the last position.

*2 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*3 Set a value that is larger than the first position.

*4 The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Enabled | Enable | BOOL | TRUE or FALSE | TRUE while the axis is being controlled. |
| InZone | In Zone | BOOL | TRUE or FALSE | TRUE when the axes position is within the zone. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Enabled | When *Enable* changes to TRUE. | When *Enable* changes to FALSE. |
| InZone | When the zone is entered. | • When the zone is exited.<br>• When *Enable* changes to FALSE. |
| Busy | When *Enable* changes to TRUE. | • When *Error* changes to TRUE.<br>• When *Enable* changes to FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

## Function

- If the axis command position or actual current position is above the *FirstPosition* and below the *Last Position* (i.e., if it is in the specified zone) when *Enable* of this instruction is TRUE, the output variable *InZone* will change to TRUE.
  You can use *ReferenceType* (Position Type Selection) to set either the command position or actual position as the axis information to monitor.
- You can perform zone monitoring for any axis type.
- If *FirstPosition* or *LastPosition* is changed while *Enable* is TRUE, the new value is applied in the period in which it is changed.
- You can set multiple zones for one axis, and these zones can overlap. You can also set zones outside the software limits.

**Precautions for Correct Use**

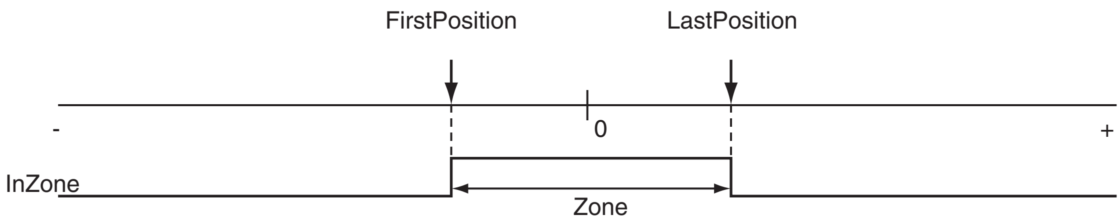
If *FirstPosition* or *LastPosition* contains a non-terminating decimal number, e.g., resulting from division, error may cause unexpected processing results.

## Instruction Details

Set the *FirstPosition* and *LastPosition* so that the following relationships are established for the Counter Mode. An error occurs if the relationship is not established.

### ● Linear Mode

Set *FirstPosition* to the same value or a smaller value than *LastPosition*.

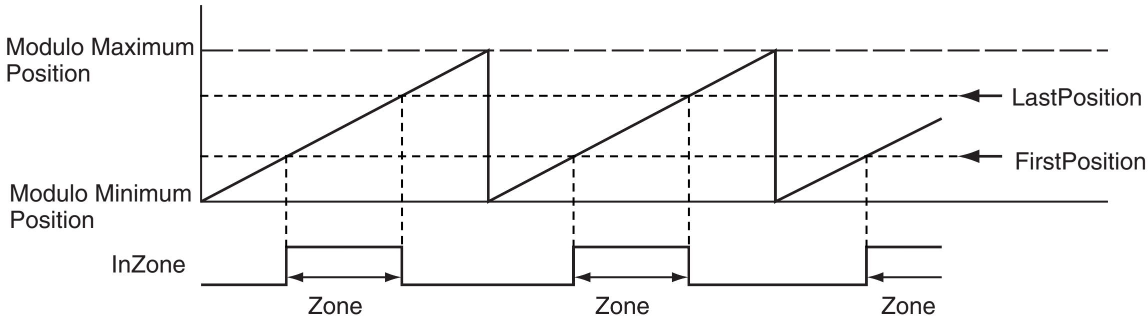


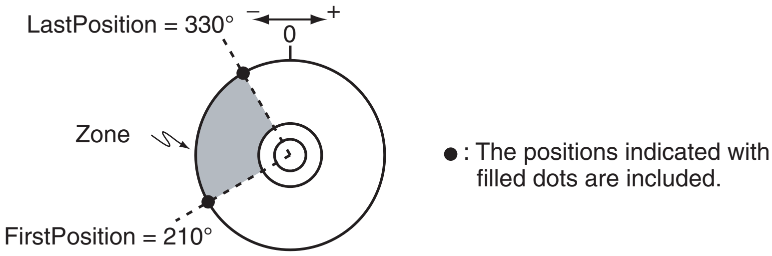

### ● Rotary Mode

In Rotary Mode, there is a difference depending on whether the ring counter includes the maximum/minimum position.

#### When Maximum/Minimum Position Is Not Included

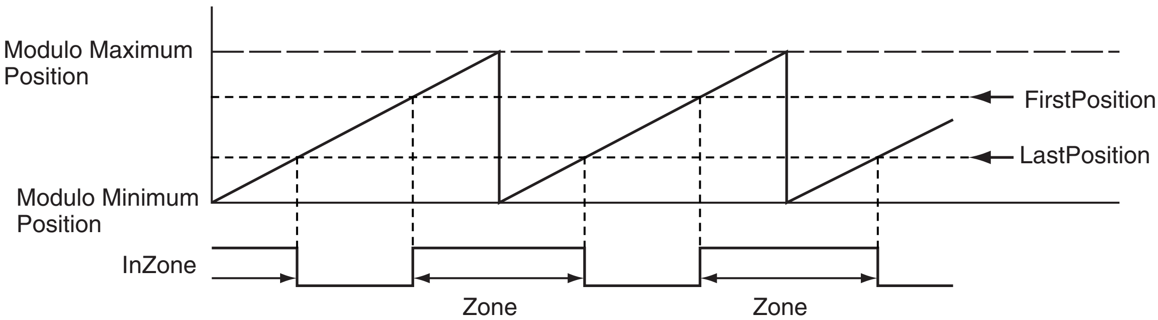
Set *FirstPosition* to the same value or a smaller value than *LastPosition*.

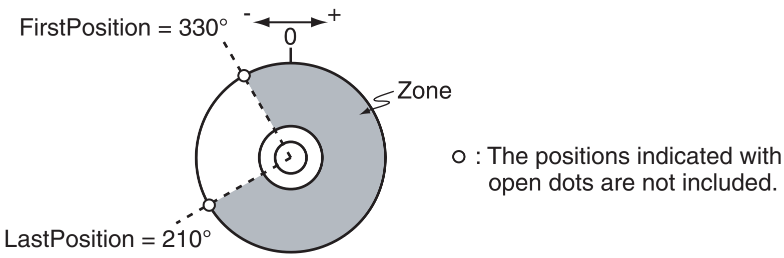



### When Maximum/Minimum Position Is Included

Set *FirstPosition* to a larger value than *LastPosition*.




## ● *ReferenceType* (Position Type Selection)

You can select one of the following position types.

- *_mcCommand*: Command position (value calculated in the previous primary period)
  The master axis command position that was calculated in the previous primary period is used for the current period.
  The command value that was calculated for the master axis in the last primary period is used to calculate the command position of the slave axis in the current period.
- *_mcFeedback*: Value obtained in the same primary period
  The actual position of the master axis that was obtained in the same primary period is used.

- **Relationship between Axis Types and Position Types**

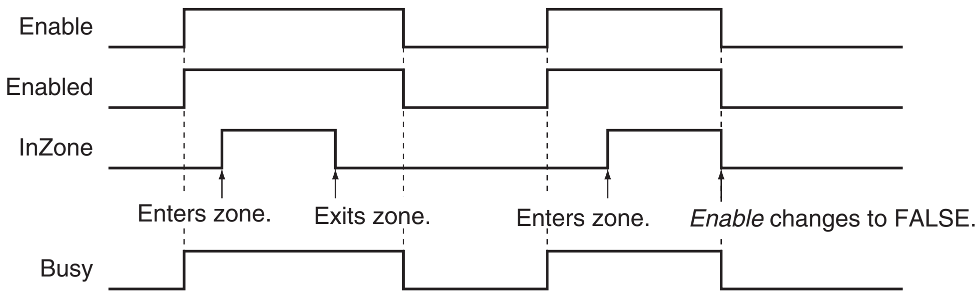
The relationship between the axis types that you can monitor and position types that is monitored is shown below.

| Axis Type | ReferenceType | |
|---|---|---|
| | _mcCommand | _mcFeedback |
| Servo axis | OK | OK |
| Encoder axis | No* | OK |
| Virtual servo axis | OK | OK |
| Virtual encoder axis | No* | OK |

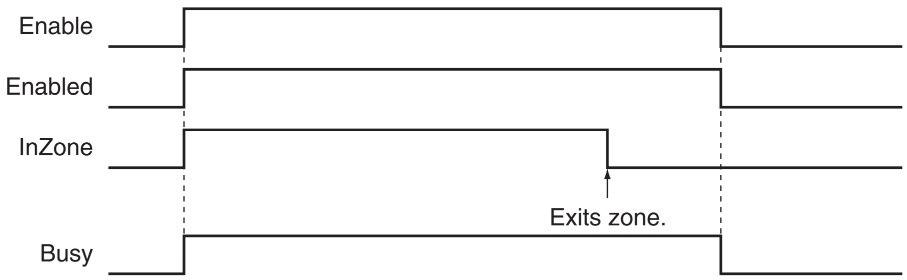
* A Position Type Selection Out of Range error (error code: 5430 hex) occurs when the instruction is executed.

## Timing Charts

- **When the Zone Is Entered during Operation or When *Enable* Changes to FALSE Within the Zone**




- **When Inside the Zone before the Instruction Is Executed and Then the Zone Is Exited**




## Re-execution of Motion Instructions

You cannot re-execute enable-type motion instructions.

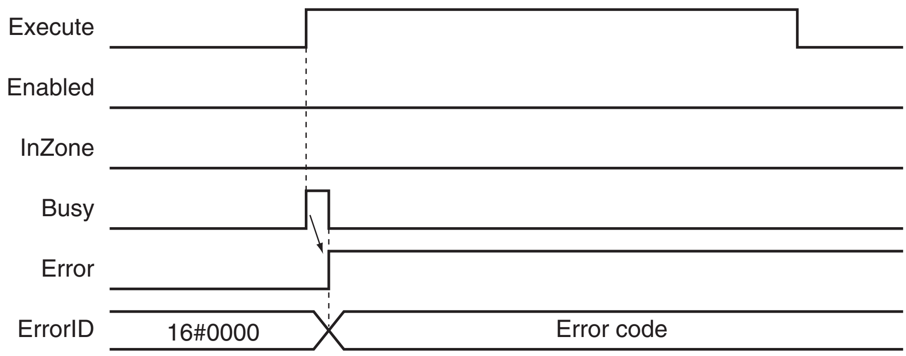
## Multi-execution of Motion Instructions

This instruction is executed independently from other instructions. The restrictions for multi-execution of motion instructions do not apply. For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User’s Manual* (Cat. No. W507).

## Errors

If this instruction cannot be executed, an error occurs and *Error* of this instruction changes to TRUE. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

## ● Timing Chart When Error Occurs

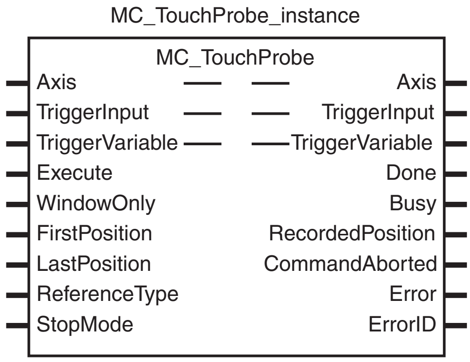



## ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_TouchProbe

The MC_TouchProbe instruction records the position of an axis when a trigger signal occurs.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_TouchProbe | Enable External Latch | FB | <br> | MC_TouchProbe_instance (<br>Axis :=*parameter,*<br>TriggerInput :=*parameter,*<br>TriggerVariable :=*parameter,*<br>Execute :=*parameter,*<br>WindowOnly :=*parameter,*<br>FirstPosition :=*parameter,*<br>LastPosition :=*parameter,*<br>ReferenceType :=*parameter,*<br>StopMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>RecordedPosition =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| WindowOnly | Window Only | BOOL | TRUE or FALSE | FALSE | Specify whether to enable or disable the window mask. |
| FirstPosition | First Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the position to start accepting the trigger. Use the type of position that is specified in *ReferenceType* (Position Type Selection).[*1]<br>The unit is command units.[*2] |
| LastPosition | Last Position | LREAL | Negative number, positive number, or 0 | 0 | Specify the position to stop accepting the trigger. Use the type of position that is specified in *ReferenceType* (Position Type Selection).[*1]<br>The unit is command units.[*2] |
| ReferenceType (Reserved) | Position Type Selection | _eMC_ REFERENCE_ TYPE | 1: _mcFeedback | 1[*3] | (Reserved) |
| StopMode | Stop Mode | _eMC_STOP_ MODE | 1: _mcImmediateStop<br>4: _mcNonStop | 4[*3] | Specify the stopping method.<br>1: Perform an immediate stop<br>4: Do not stop |

*1 Refer to *WindowOnly* on page 3-319 for details.

*2 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*3 The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| RecordedPosition | Latched Position | LREAL | Negative number, positive number, or 0 | Contains the latched position. The unit is in command units.*1 |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | *2 | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | • When the latched position is recorded and the instruction is completed after the trigger signal occurs.<br>• If stopping is specified, when the axis stops at the latched position after the latched position is recorded and the instruction is completed after the trigger signal occurs. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• If *StopMode* is set to *_mcImmediateStop*, when a change is made to a mode other than CSP Mode during execution.<br>• When the slave is disconnected.<br>• When a slave communications error occurs (except during process data communications).<br>• When the MC_AbortTrigger instruction is executed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.*1 |
| TriggerInput | Trigger Input Condition | _sTRIGGER_REF | --- | Set the trigger condition.*2 |
| TriggerVariable | Trigger Variable | BOOL | TRUE or FALSE | Specify a trigger input variable when the Controller Mode is specified with a trigger condition. |

*1 Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

*2 Define a user-defined variable with a data type of _sTRIGGER_REF.

### ● _sTRIGGER_REF

| Name | Meaning | Data type | Valid range | Function |
|---|---|---|---|---|
| Mode | Mode | _eMC_TRIGGER_ MODE | 0: _mcDrive<br>1: _mcController | Specify the trigger mode.<br>0: Drive Mode<br>1: Controller Mode |
| LatchID | Latch ID Selection | _eMC_TRIGGER_ LATCH_ID | 0: _mcLatch1<br>1: _mcLatch2 | Specify which of the two latch functions to use.<br>0: Latch 1<br>1: Latch 2 |
| InputDrive | Trigger Input Signal | _eMC_TRIGGER_ INPUT_DRIVE | 0: _mcEncoderMark<br>1: _mcEXT | Specify the Servo Drive trigger signal to use in Drive Mode.<br>0: Z-phase signal<br>1: External input |

## Function

- Latching is used to control positioning based on the point where a trigger signal occurs, such as a signal from a sensor input. The position of the axis is recorded (i.e., latched) when the trigger signal occurs.
- The position of the axis that is specified with Axis is output to *RecordedPosition* (Latched Position) according to the trigger settings.
  As trigger settings, you can specify *TriggerInput* (Trigger Input Condition), *WindowOnly*, *FirstPosition*, *LastPosition*, and *StopMode*.
- The output value of *RecordedPosition* (Latched Position) is held until the axis position is recorded again by the same *MC_TouchProbe* (Enable External Latch) instance.

### ● Mapping Data Objects

You must map the following object data when the MC_TouchProbe (Enable External Latch) instruction is executed with *Mode* set to Drive Mode. Mapping is performed in the Detailed Setting Area of the Axis Basic Settings Display of the Sysmac Studio.

#### Axes Type Set to Servo Axis

- Touch probe function (60B8 hex)
- Touch probe status (60B9 hex)
- Touch probe pos1 pos value (60BA hex)
- Touch probe pos2 pos value (60BC hex)

#### Axes Type Set to Encoder Axis

- Touch probe function (4020 hex)
- Software Switch of Encoder's Input Slave (4020 hex)
- Touch probe status (4030 hex)
- Touch probe pos1 pos value (4012 hex)
- Touch probe pos2 pos value (4013 hex)
- Status of Encoder's Input Slave (4030 hex)

If even one of the required objects is not set, a Process Data Object Setting Missing error (error code 3461 hex) occurs.

For details on mapping data objects, refer to *2-3 PDO Mapping* and to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Instruction Details

This section describes the instruction in detail.

### ● Specifying *Axis*

- *Axis* specifies the axis for which to latch the position.
- If the specified Axis is enabled by the MC_GroupEnable (Enable Axes Group) instruction, the MC_TouchProbe (Enable External Latch) instruction causes an error and is not executed.
- For each axis, you can specify *LatchID* to execute up to two MC_TouchProbe (Enable External Latch) instructions at the same time.
- *LatchID* is also used to specify the latch to abort for the MC_AbortTrigger (Disable External Latch) instruction.

**Additional Information**

Latching a position is also possible if an encoder axis that is connected to an OMRON GX-series GX-EC02□□ EtherCAT Encoder Input Slave is used.

### ● Trigger Input Condition

Select the trigger conditions with *Mode*, *LatchID*, and *InputDrive* of the *TriggerInput* (Trigger Input Conditions) variable.

#### *Mode*

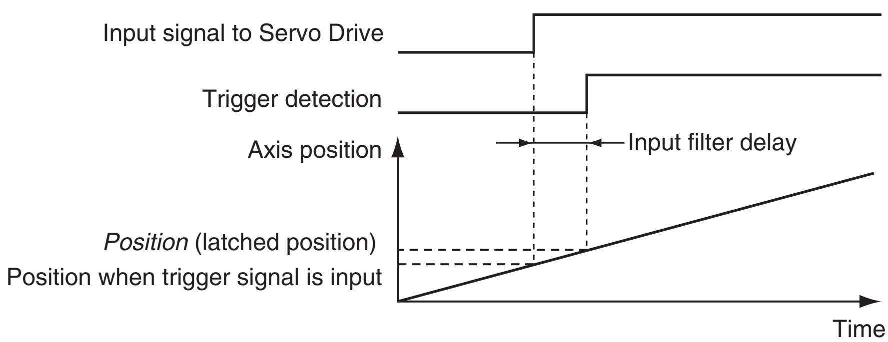
- The mode can be set to Drive Mode to specify a signal from the Servo Drive as the trigger, or to Controller Mode to specify a trigger with *TriggerVariable*.
- The trigger occurs on the rising edge of the trigger signal. The axis position is latched on the first trigger (FALSE to TRUE) after the MC_TouchProbe instruction is executed.
- While this instruction is *Busy* (Executing), a change in *TriggerVariable* is taken as a trigger even if *Execute* is FALSE.

**Additional Information**

Set *Mode* to *_mcDrive* (Servo Drive Mode) if you use an OMRON GX-series GX-EC02□□ Ether-CAT Encoder Input Slave.

**Drive Mode**

The latched actual position is more precise in Drive Mode than it is in Controller Mode. This is because Drive Mode uses a Servo Drive function to latch the position.



**Precautions for Correct Use**

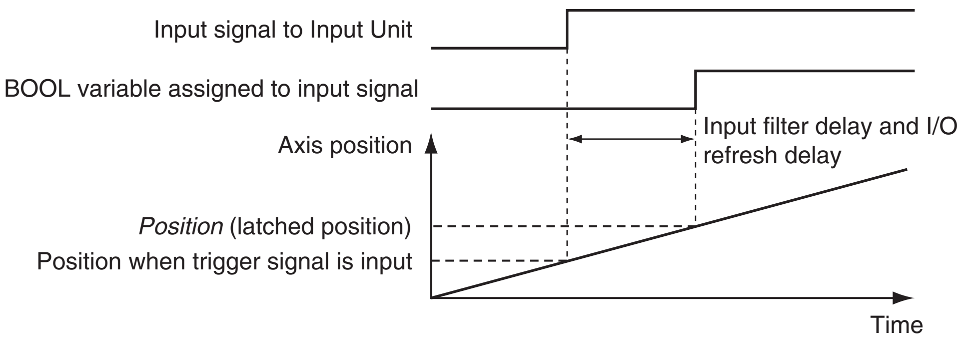
- When using Drive Mode, make sure that you connect the latch signal to the *LatchID* that you are going to use.
- The width of the latch signal depends on the performance of the Servo drive and other factors.

**Additional Information**

Set *InputDrive* to *_mcEXT* (External Input) if you use an OMRON GX-series GX-EC02☐☐ EtherCAT Encoder Input Slave.
The OMRON GX-series GX-EC02☐☐ EtherCAT Encoder Input Slaves cannot latch on the Z phase. If you specify *_mcEncoderMark* (Z phase), an error occurs when the instruction is executed. *Error* changes to TRUE and a Process Data Object Setting Missing error (error code: 3461 hex) is output to *ErrorID* when the instruction is executed.

**Controller Mode**

- You can specify a BOOL variable as the trigger in Controller Mode.
- Specify the BOOL variable that you want to use as a trigger for *TriggerVariable*.
- The Controller Mode causes a longer delay compared to the Drive Mode. This is due to the I/O refresh delay that occurs when the trigger input signal is updated in the BOOL variable.




**Precautions for Correct Use**

If you use Controller Mode, the latch is performed each primary period. Therefore, the trigger variable must remain TRUE for at least one primary period.
Also, one primary period is required between when the trigger variable changes to TRUE and the MC Function Module processes the latch.

## *LatchID*

- You can execute up to two MC_TouchProbe instructions per axis. Use *LatchID* to specify which of the two latches to use.
- If a *LatchID* specified for the same axis is already being executed, only the last instruction is valid. *CommandAborted* of the previous instruction will change to TRUE.
- *LatchIDs* indicate latch circuit 1 and latch circuit 2 in the Servo Drive.

For information on latch IDs, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

**Additional Information**

The enumerators correspond to the signal names of the OMRON GX-series GX-EC02□□ EtherCAT Encoder Input Slave as shown below.

| Enumerator | Signal name on Encoder Input Terminal |
|---|---|
| _mcLatch1 | External latch input A |
| _mcLatch2 | External latch input B |

### *InputDrive*

- You can select *_mcEncoderMark* (Z phase) or *_mcEXT* (External Input) as the trigger.
- Select *_mcEncoderMark* (Z phase) to use the Z phase of the Servo Drive as the trigger. Select *_mcEXT* to use an external signal input to the Servo Drive as the trigger.
- For an OMRON G5-series Servo Drive, there are three options for *_mcEXT*: Ext1, Ext2, and Ext3. Use Sysmac Studio to make the setting. You can use the same setting for two triggers in the Servo Drive.

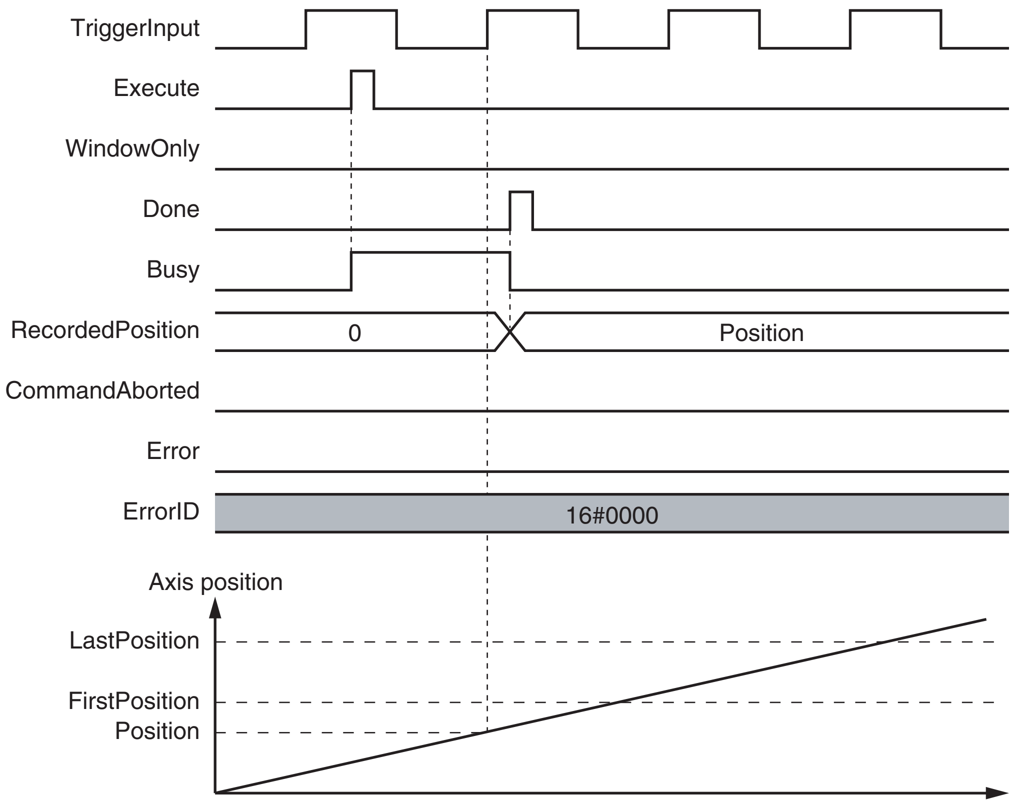
## ● WindowOnly

- *WindowOnly* specifies whether the window is enabled or disabled.
- If you specify *Disable*, triggers are detected for all axis positions.
- If you specify *Enable*, triggers are detected only when the axis position is within the range specified by *FirstPosition* and *LastPosition*.

The following timing chart shows the difference in operation depending on the WindowOnly setting.

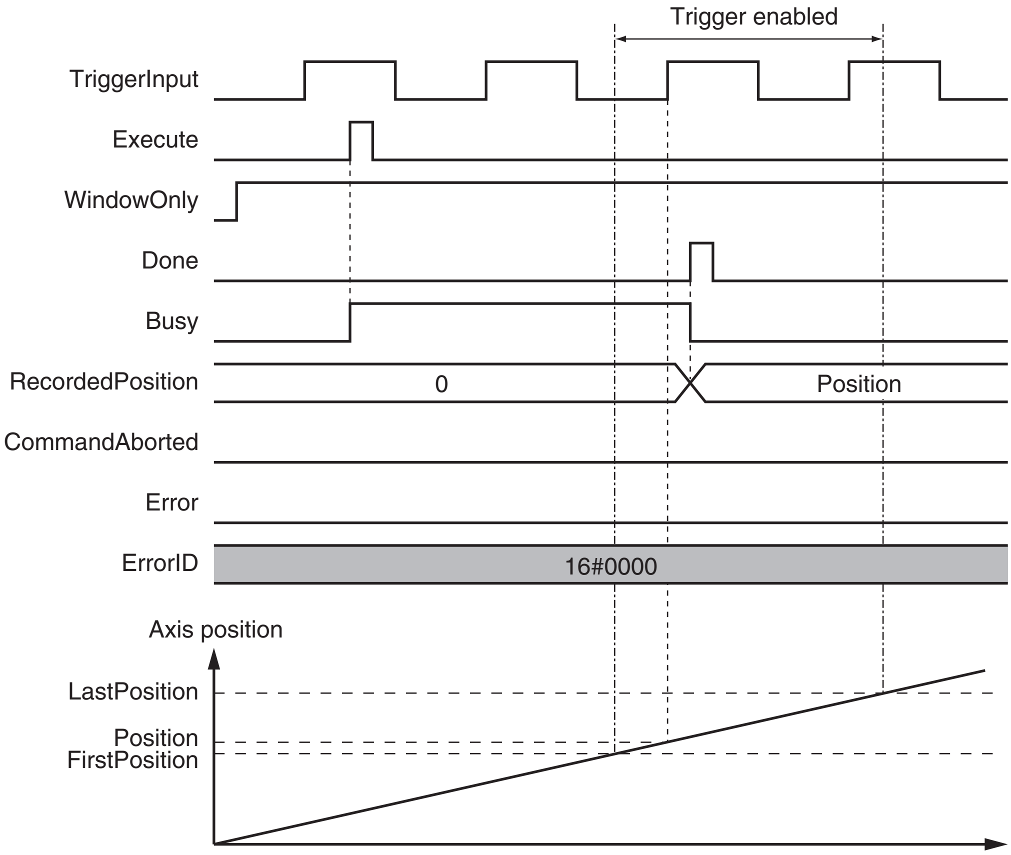
### *WindowOnly* Set to *Disable*

The axis position when the first trigger occurs after *Execute* changes to TRUE is output to *RecordedPosition* (Latched Position).



### *WindowOnly* Set to *Enable*

Only trigger inputs within the window are detected to latch the axis position.




**Precautions for Correct Use**

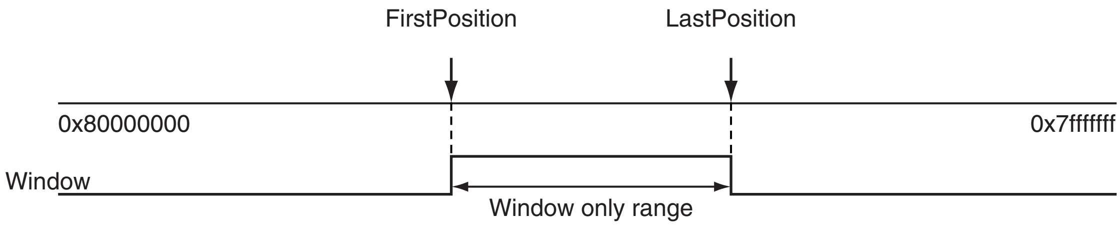
- Latching is not possible immediately after *WindowOnly* changes to TRUE and until the latch function is activated.
- Time is needed until the latch function is activated. If the effective range for *WindowOnly* is too small, latching is not possible. The range in which latching is possible depends on the Servo Drive or Encoder Input Terminal performance, and on EtherCAT communications.

The range that is defined by *FirstPosition* and *LastPosition* depends on the Count Mode, as given below.

### Linear Mode

- The valid range of the window is as follows:
  *FirstPosition* must be less than or equal to the window range and the window range must be less than or equal to *LastPosition*.
- An error will occur if the *FirstPosition* is greater than the *LastPosition*.
- An error will also occur if a position beyond the position range of Linear Mode is specified.
- *FirstPosition* and *LastPosition* are LREAL variables. Do not set them to the same values.
  Refer to *Treatment of REAL and LREAL Data* on page 1-13 for information on LREAL data.

  The window only range in Linear Mode is shown below.

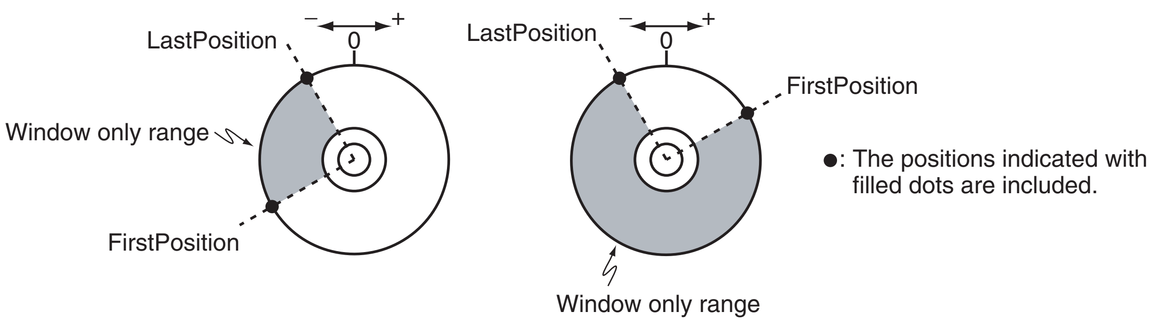



**Note** The window only range can include the *FirstPosition* and *LastPosition*.

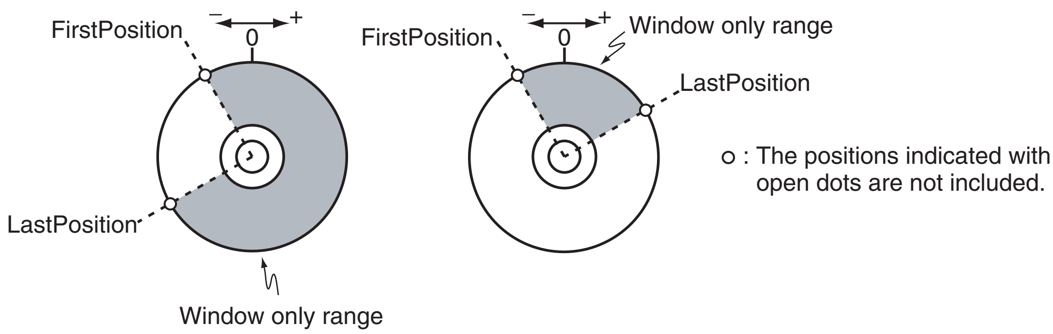
### Rotary Mode

- The *FirstPosition* can be less than, equal to, or greater than the *LastPosition*.
- If the *FirstPosition* is greater than the *LastPosition*, the setting range includes the maximum/minimum position of the ring counter.
- An error will occur if you specify a value beyond the upper and lower limits of the ring counter.

FirstPosition ≤ LastPosition

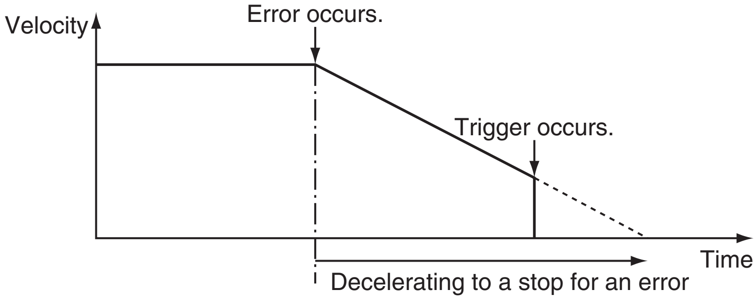



FirstPosition > LastPosition




## ● *StopMode*

- You can specify the *StopMode* for the specified *Axis* when a trigger occurs.
- If *_mcNonStop* is specified, the axis will not stop even if a trigger occurs.
- If *_mcImmediateStop* is specified, the axis stops at the latched position when a trigger occurs. *CommandAborted* of the instruction that was moving the axis changes to TRUE due to this stop.
- *_mcImmediateStop* functions in CSP Mode.
- If *_mcImmediateStop* is specified in CSV/CST Mode, an error occurs when the instruction is executed. If you change to CSV/CST Mode during execution for *_mcImmediateStop*, *CommandAborted* changes to TRUE.
- For *_mcImmediateStop*, *Done* changes to TRUE as soon as the axis command stops. *Busy* is TRUE until the axis stops at the latched position.
- An in-position check is not performed when stopping for *_mcImmediateStop*.
- If an axis error occurs for an axis for which *_mcImmediateStop* is specified and a trigger occurs before the axis is stopped, an immediate stop is performed for the axis by the OMRON G5-series Servo Drive.




The OMRON G5-series Servo Drive executes an immediate stop when the trigger occurs.
When the MC Function Module detects that the OMRON G5-series Servo Drive has stopped, it performs an immediate stop.
The dotted line shows the target path when a trigger does not occur.

**Precautions for Correct Use**

- For *_mcImmediateStop*, the function of the Servo Drive is used to stop if an OMRON G5-series Servo Drive is used.
  The MC Function Module processes the stop if a servo drive from another manufacturer is used. In either case, the axis will stop beyond the latched position, and the axis will then return to the latched position for a command from the Controller.
- If you use *_mcImmediateStop* and the command velocity is high, the distance to return to the latched position is longer for both OMRON Servo Drives and servo drives from other manufacturers. Make sure that the command velocity is not too high.
- Specify *_mcNonStop* (No Stop) for an encoder axis. If *_mcImmediateStop* is specified, *Error* will change to TRUE when the instruction is executed. At the same time, an Enable External Latch Instruction Execution Disabled error (error code: 5492 hex) is output to *ErrorID*.
- If *_mcImmediateStop* is used and both of the following processes are executed in the same control period, an OMRON G5-series Servo Drive enters the Target Ignore state.
  - Counter latch processing in the Servo Drive when the latch signal turns ON
  - Latch release processing when *CommandAborted* changes to TRUE for the MC_TouchProbe (Enable External Latch) instruction

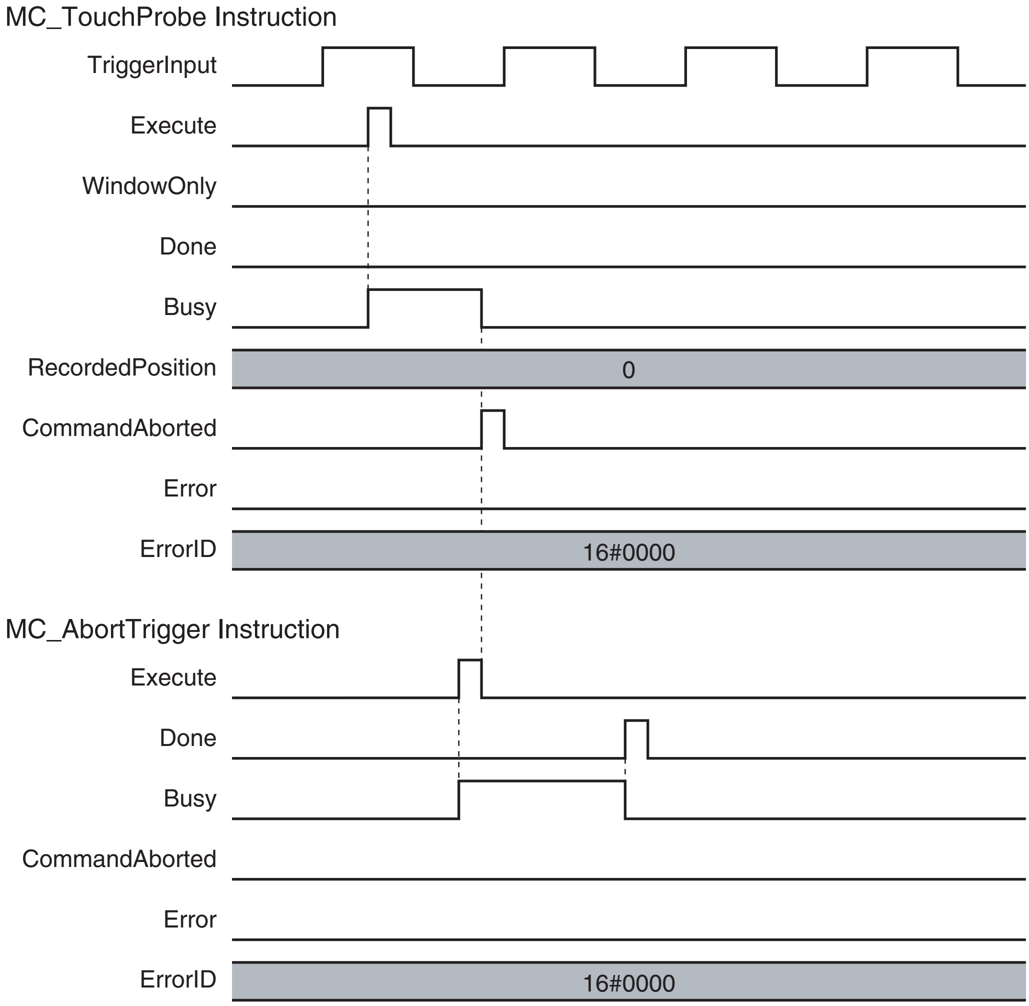
  Make sure that *CommandAborted* for the instruction does not change to TRUE when the latch signal turns ON.

## Axis Variable Status

If *_mcImmediateStop* is specified for *StopMode*, *Status.Stopping* (Deceleration Stopping) in the Axis Variable is TRUE during stop processing for the trigger.

## Aborting the Instruction

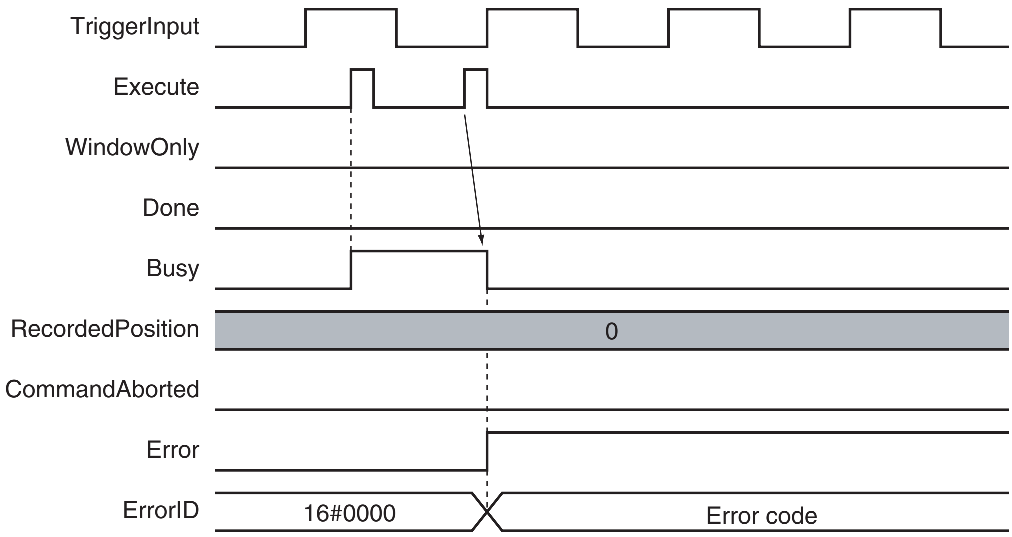
End the MC_TouchProbe (Enable External Latch) instruction with the MC_AbortTrigger (Disable External Latch) instruction. Specify the *Axis* and the *LatchID* (Latch ID Selection) to stop for the MC_AbortTrigger instruction and execute it to stop the axis.



## Re-execution of Motion Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

An error occurs if *Execute* changes to TRUE again before the MC_TouchProbe instance completes reading *RecordedPosition* (Latched Position).

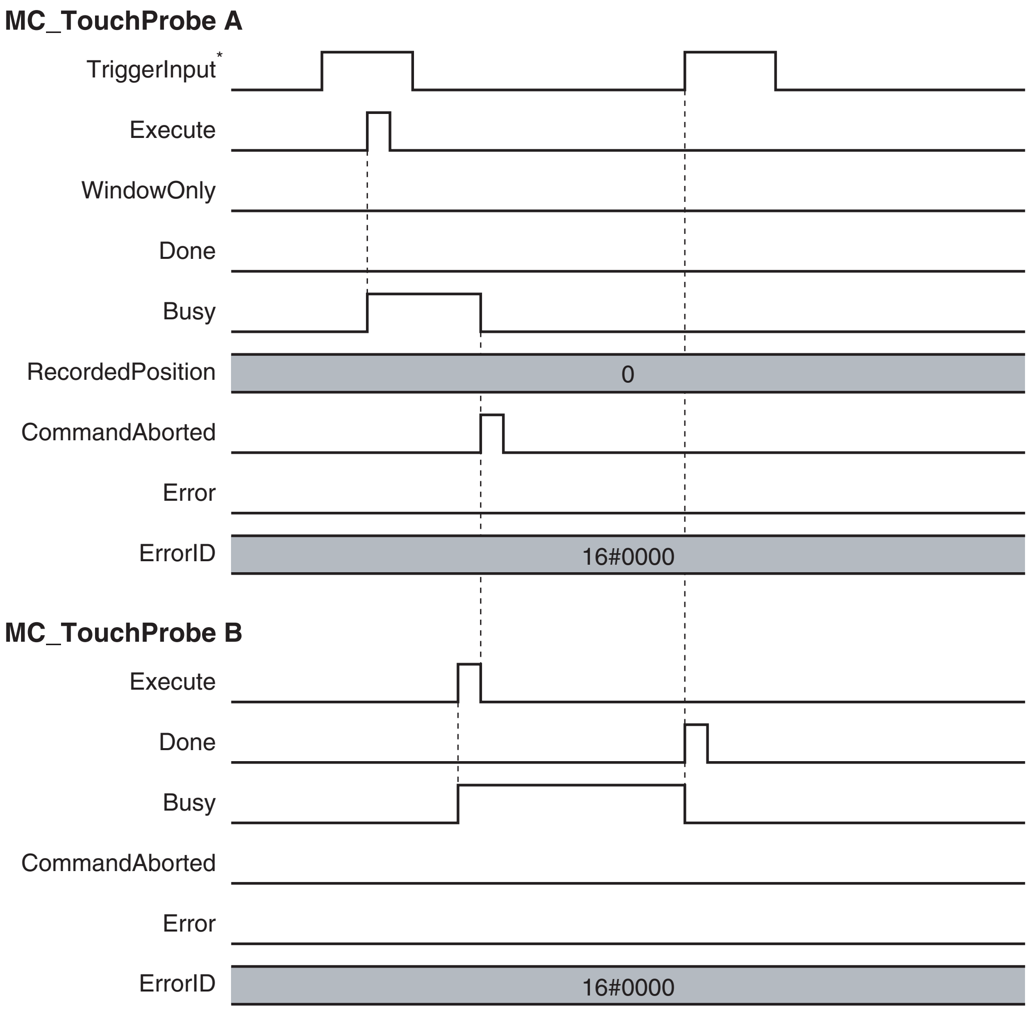


## Multi-execution of Motion Instructions

For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

You can execute only one trigger at a time for a single LatchID on the same Axis.
If you execute another MC_TouchProbe (Enable External Latch) instance for the same *LatchID* while an MC_TouchProbe (Enable External Latch) instruction is in execution, *CommandAborted* of the first instruction changes to TRUE and the second instruction is executed.




* Here, the trigger input signal of the Servo Drive is used.

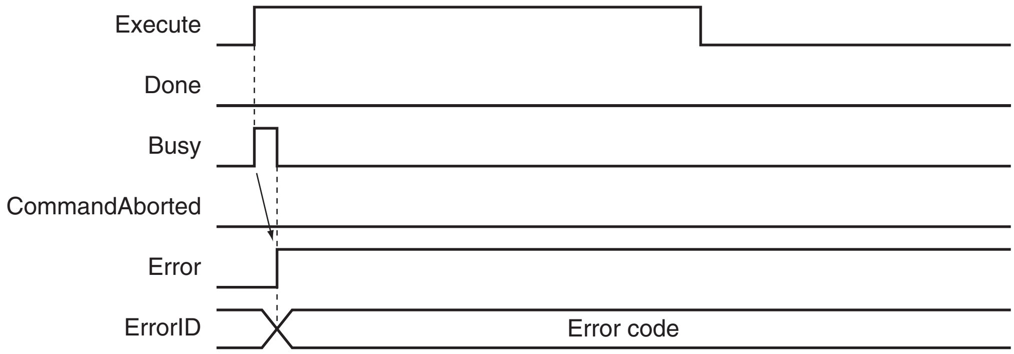
**Additional Information**

If *_mcImmediateStop* is specified for *StopMode*, *CommandAborted* for the second instruction changes to TRUE after the axis stops for the trigger.

## Error

If an error occurs during execution of the MC_TouchProbe instruction, *Error* will change to TRUE. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

### ● Timing Chart When Error Occurs




### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# Sample Programming

This section shows sample programming for position latching control by an external sensor.

## Parameter Settings

The minimum settings required for this sample programming are given below.

### ● Axis Parameters

#### Axis Type

| Axis | Axis Type |
|---|---|
| Axis 1 | Servo axis |

#### Count Mode

| Axis | Count Mode |
|---|---|
| Axis 1 | Rotary Mode |

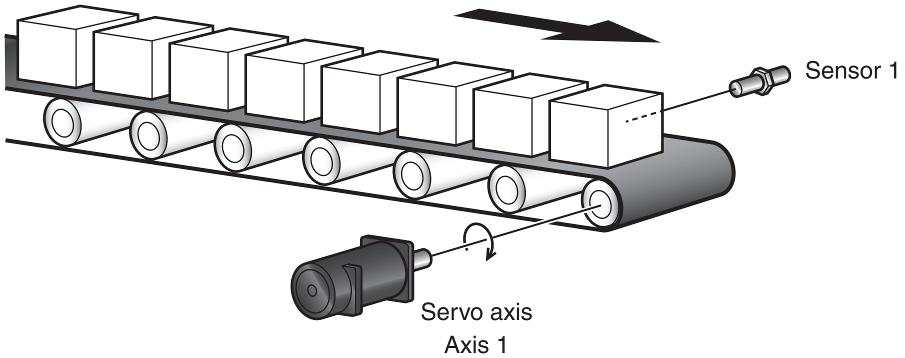
#### Ring Counter

| Axis | Modulo maximum position | Modulo minimum position |
|---|---|---|
| Axis 1 | 360 | 0 |

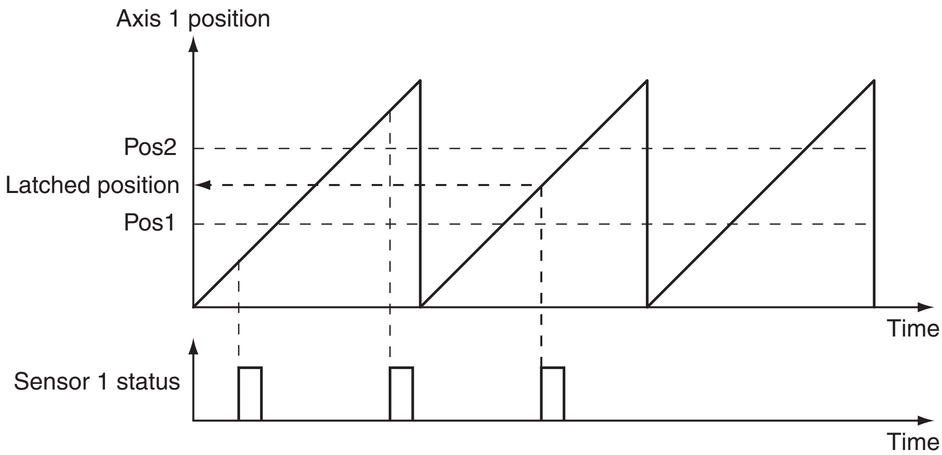
#### Unit of Display

| Axis | Unit of Display |
|---|---|
| Axis 1 | degree |

## Operation Example




### Operation Pattern




**1** Starting the Master Axis

Velocity control is performed for axis 1.

**2** Detecting Workpiece

Sensor 1 detects the workpiece.

**3** Latching the Position

If the workpiece is detected in the window (Pos1 to Pos2), the position of axis 1 is latched.

## Ladder Diagram

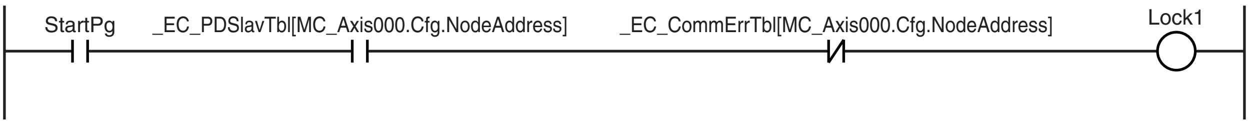
### Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | This is the Axis Variable for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| Pwr_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Pos1 | LREAL | --- | This variable specifies the first position of the window. |

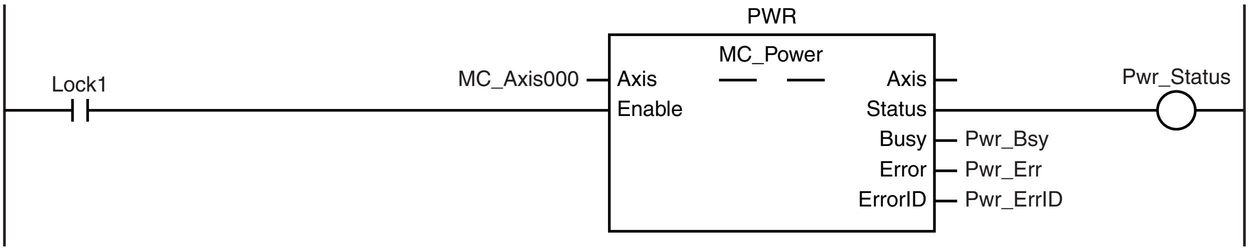
| Name | Data type | Default | Comment |
|---|---|---|---|
| Pos2 | LREAL | --- | This variable specifies the last position of the window. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

## ● Sample Programming

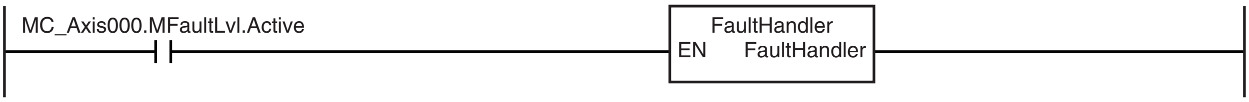
If *StartPg* is TRUE, EtherCAT communications are checked to see if process data communications are normal.



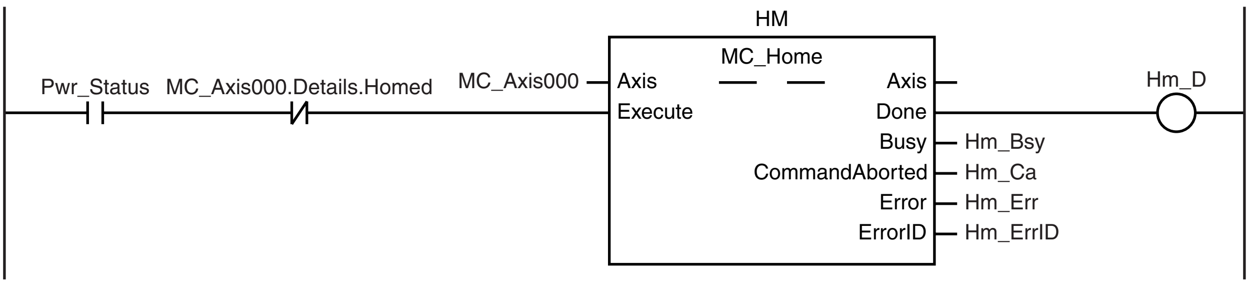

If process data communications are active for axis 1, the Servo is turned ON.



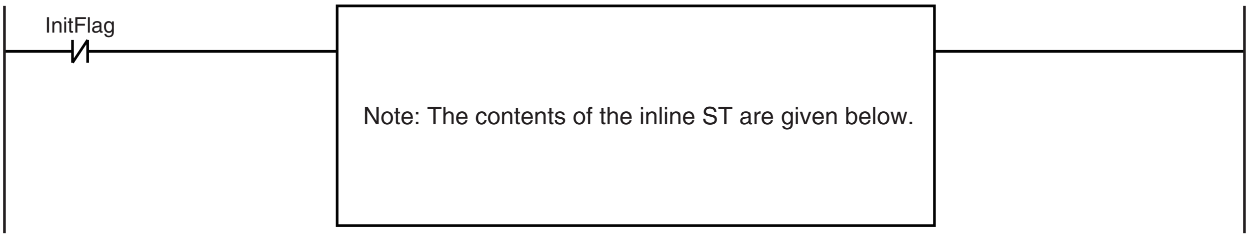

If there is a minor fault level error for axis 1, the error handler for the device (FaultHandler) is executed.
Design FaultHandler separately.



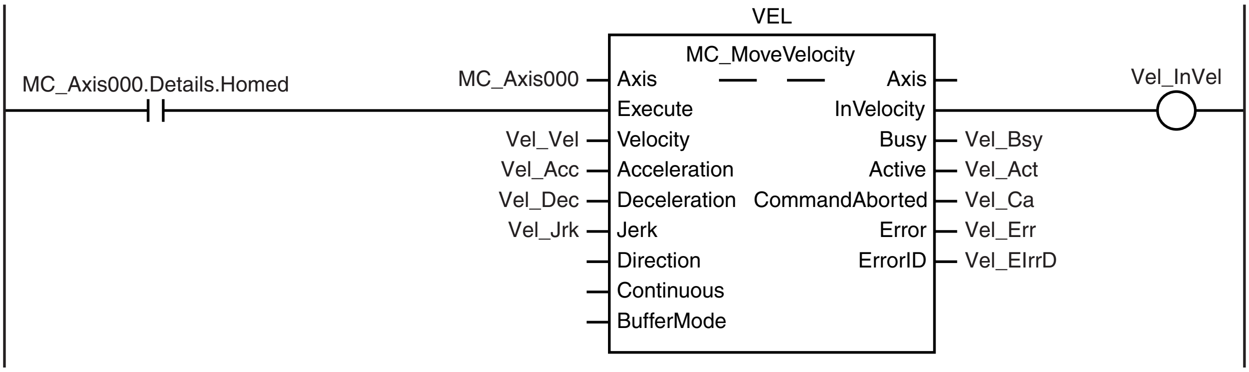

If the Servo is ON and home is not defined, the Home instruction is executed.



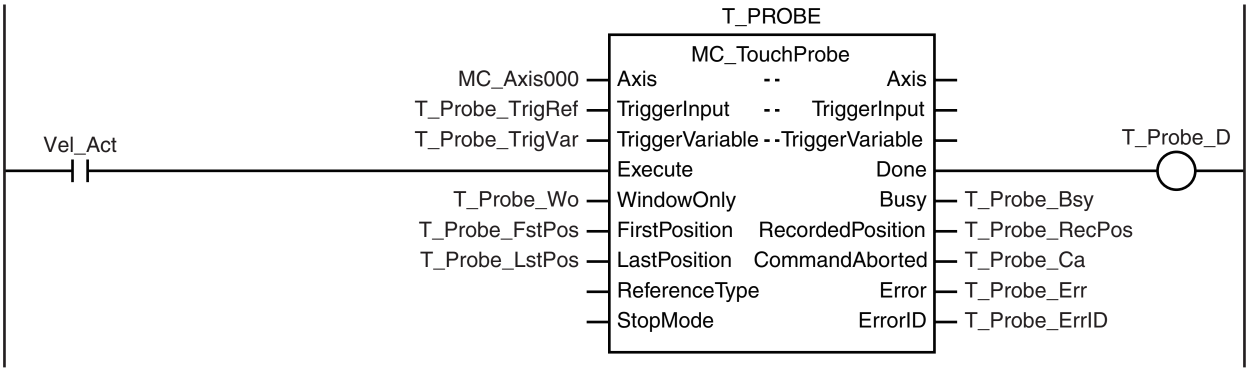
The parameters are set for the MC_MoveVelocity (Velocity Control) and MC_TouchProbe (Enable External Latch) instructions.




The MC_MoveVelocity (Velocity Control) instruction is executed if home is defined for axis 1.




Latch processing is executed after the MC_MoveVelocity (Velocity Control) instruction is started.




### Contents of Inline ST

```
// MC_MoveVelocity parameters
    Vel_Vel    := LREAL#1000.0;
    Vel_Acc    := LREAL#1000.0;
    Vel_Dec    := LREAL#1000.0;
    Vel_Jrk    := LREAL#1000.0;

// MC_TouchProbe parameters
    T_Probe_TrigRef.Mode          := _eMC_TRIGGER_MODE#_mcDrive;
    T_Probe_TrigRef.LatchID       := _eMC_TRIGGER_LATCH_ID#_mcLatch1;
    T_Probe_TrigRef.InputDrive    := _eMC_TRIGGER_INPUT_DRIVE#_mcEXT;
    T_Probe_TrigVar               := FALSE;
    T_Probe_Wo                    := TRUE;
    T_Probe_FstPos                := LREAL#1000.0;
    T_Probe_LstPos                := LREAL#2000.0;

    // Change InitFlag to TRUE after setting the input parameters.
    InitFlag:=TRUE;
```

## Structured Text (ST)

### Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Axis000 | _sAXIS_REF | --- | This is the Axis Variable for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| Pwr_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Pos1 | LREAL | --- | This variable specifies the first position of the window. |
| Pos2 | LREAL | --- | This variable specifies the last position of the window. |
| StartPg | BOOL | FALSE | The Servo is turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

### Sample Programming

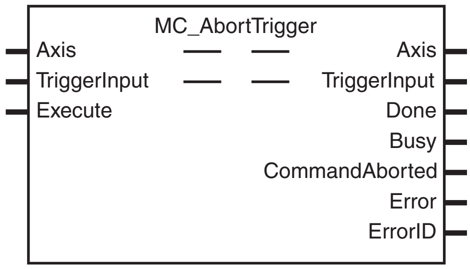
```
// Processing when input parameters are not set
IF InitFlag=FALSE THEN

    // MC_MoveVelocity parameters
    Vel_Vel    := LREAL#1000.0;
    Vel_Acc    := LREAL#1000.0;
    Vel_Dec    := LREAL#1000.0;
    Vel_Jrk    := LREAL#1000.0;

    // MC_TouchProbe parameters
    T_Probe_TrigRef.Mode          := _eMC_TRIGGER_MODE#_mcDrive;
    T_Probe_TrigRef.LatchID       := _eMC_TRIGGER_LATCH_ID#_mcLatch1;
    T_Probe_TrigRef.InputDrive    := _eMC_TRIGGER_INPUT_DRIVE#_mcEXT;
    T_Probe_TrigVar               := FALSE;
    T_Probe_Wo                    := TRUE;
    T_Probe_FstPos                := LREAL#1000.0;
    T_Probe_LstPos                := LREAL#2000.0;

    // Change InitFlag to TRUE after setting the input parameters.
    InitFlag:=TRUE;

END_IF;


// If StartPg is TRUE and EtherCAT communications are normal, the Servo is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis000.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
    Pwr_En:=TRUE;
ELSE
    Pwr_En:=FALSE;
END_IF;
```

```
// Processing for a minor fault level error
// Implement FaultHandler separately.
IF MC_Axis000.MFaultLvl.Active=TRUE THEN
    FaultHandler();
END_IF;


// If the Servo is ON and home is not defined, the Home instruction is executed.
IF (Pwr_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
    Hm_Ex:=TRUE;
END_IF;

// After home is defined, MC_MoveVelocity is executed.
IF MC_Axis000.Details.Homed=TRUE THEN
    Vel_Ex:=TRUE;
END_IF;

// After MC_MoveVelocity is executed, MC_TouchProbe is executed.
IF Vel_Act=TRUE THEN
    T_Probe_Ex:= TRUE;
END_IF;


//MC_Power
PWR(
    Axis        := MC_Axis000,
    Enable      := Pwr_En,
    Status      => Pwr_Status,
    Busy        => Pwr_Bsy,
    Error       => Pwr_Err,
    ErrorID     => Pwr_ErrID
);

//MC_Home
HM(
    Axis                := MC_Axis000,
    Execute             := Hm_Ex,
    Done                => Hm_D,
    Busy                => Hm_Bsy,
    CommandAborted      => Hm_Ca,
    Error               => Hm_Err,
    ErrorID             => Hm_ErrID
);

//MC_MoveVelocity
VEL(
    Axis                := MC_Axis000,
    Execute             := Vel_Ex,
    Velocity            := Vel_Vel,
    Acceleration        := Vel_Acc,
    Deceleration        := Vel_Dec,
    Jerk                := Vel_Jrk,
    InVelocity          => Vel_InVel,
    Busy                => Vel_Bsy,
    Active              => Vel_Act,
    CommandAborted      => Vel_Ca,
    Error               => Vel_Err,
    ErrorID             => Vel_ErrID
);
//MC_TouchProbe
T_PROBE(
    Axis                := MC_Axis000,
    TriggerInput        := T_Probe_TrigRef,
```

```
    TriggerVariable      := T_Probe_TrigVar,
    Execute              := T_Probe_Ex,
    WindowOnly           := T_Probe_Wo,
    FirstPosition        := T_Probe_FstPos,
    LastPosition         := T_Probe_LstPos,
    Done                 => T_Probe_D,
    Busy                 => T_Probe_Bsy,
    RecordedPosition     => T_Probe_RecPos,
    CommandAborted       => T_Probe_Ca,
    Error                => T_Probe_Err,
    ErrorID              => T_Probe_ErrID
);
```

# MC_AbortTrigger

The MC_AbortTrigger instruction aborts a current latch operation.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_AbortTrigger | Disable External Latch | FB | MC_AbortTrigger_instance  | MC_AbortTrigger_instance (<br>Axis :=*parameter,*<br>TriggerInput :=*parameter,*<br>Execute :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | • When latching is stopped.<br>• When this instruction is executed for a latch that is not in execution and processing ends. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution of or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the target axis for latching.*1 |
| TriggerInput | Trigger Selection | _sTRIGGER_REF | --- | Use this variable to select the trigger condition.*2<br>Refer to the following table for _sTRIGGER_REF. |

*1 Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

*2 Define a user-defined variable with a data type of _sTRIGGER_REF.

### ● _sTRIGGER_REF

| Name | Meaning | Data type | Valid range | Function |
|---|---|---|---|---|
| Mode | Mode | _eMC_TRIGGER_MODE | 0: _mcDrive<br>1: _mcController | Specify the trigger mode.<br>0: Drive Mode<br>1: Controller Mode |
| LatchID | Latch ID Selection | _eMC_TRIGGER_LATCH_ID | 0: _mcLatch1<br>1: _mcLatch2 | Specify which of the two latch functions to use.<br>0: Latch 1<br>1: Latch 2 |
| InputDrive | Trigger Input Signal | _eMC_TRIGGER_INPUT_DRIVE | 0: _mcEncoderMark<br>1: _mcEXT | Specify the Servo Drive trigger signal to use in Drive Mode.<br>0: Z-phase signal<br>1: External input |

## Function

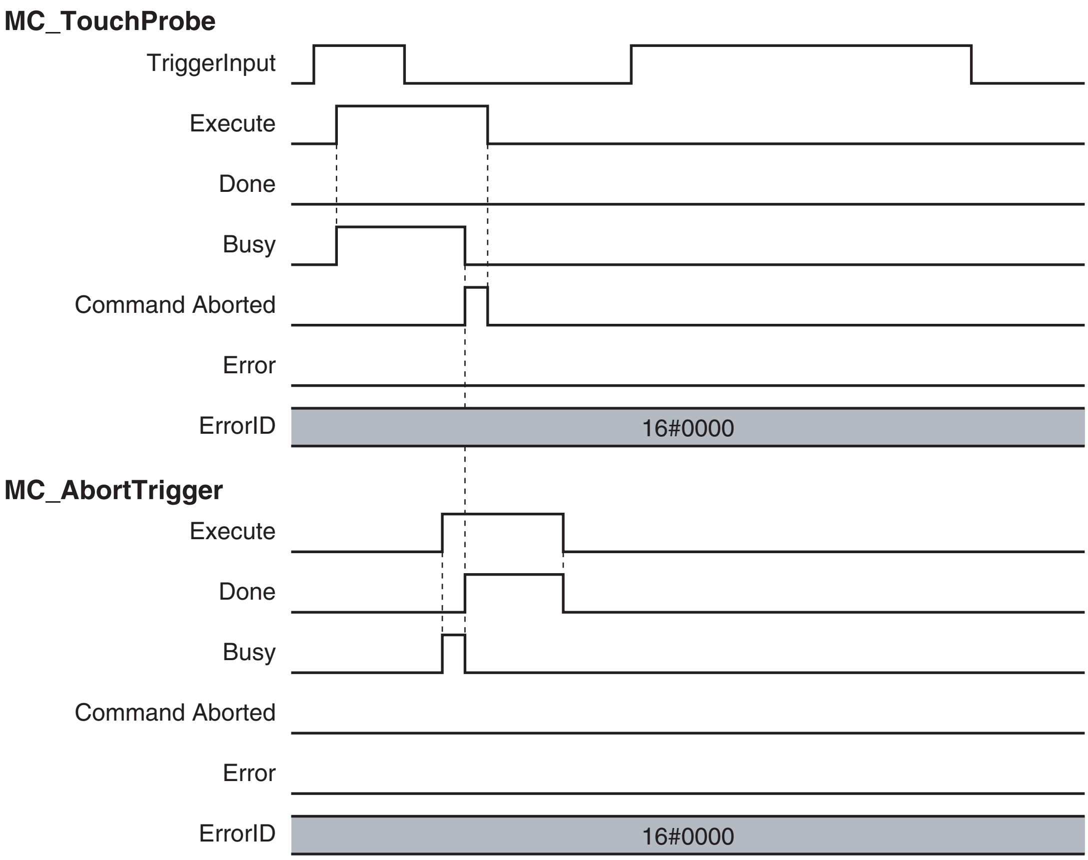
- The MC_AbortTrigger cancels a latch operation.
- You can specify the latch operation to abort by specifying the *Axis* and *LatchID* for the MC_AbortTrigger (Disable External Latch) instruction.
- If you execute MC_AbortTrigger (Disable External Latch) for a trigger for which there is no latch request, MC_AbortTrigger does nothing and ends normally. This is the same when MC_AbortTrigger (Enable External Latch) is executed for a MC_TouchProbe instruction for which *Done* is TRUE.

**Precautions for Correct Use**

- If the MC_GroupEnable (Enable Axes Group) instruction was executed for the Axis that is specified for the MC_AbortTrigger (Disable External Latch) instruction, an error occurs for the MC_AbortTrigger instruction and it is not executed.
- An error occurs for this instruction if the MC_AbortTrigger (Disable External Latch) instruction is executed during execution of the MC_Home, MC_MoveFeed (Interrupt Feeding), or MC_MoveLink (Synchronous Positioning) instruction.

## Timing Charts

- *Done* for the MC_AbortTrigger (Disable External Latch) instruction changes to TRUE one period after *Execute* changes to TRUE.




## Re-execution of Motion Instructions

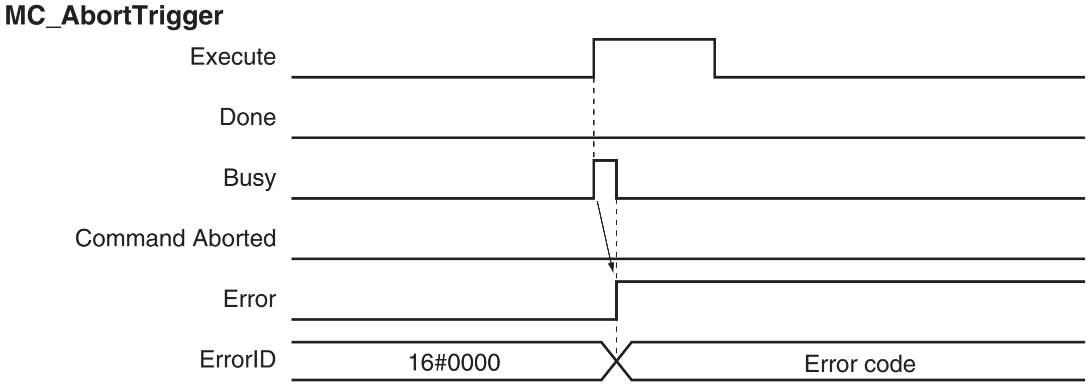
This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Instructions

For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Errors

If an error occurs during execution of the MC_AbortTrigger (Disable) instruction. *Error* will change to TRUE. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

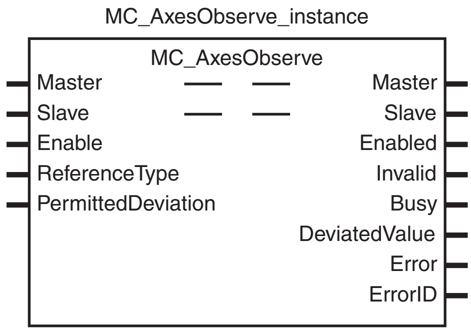



- **Error Codes**

Refer to *A-1 Error Codes* for instruction errors.

# MC_AxesObserve

The MC_AxesObserve instruction monitors the deviation between the command position or feedback position for the specified axis to see if it exceeds the allowed value.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_AxesObserve | Monitor Axis Following Error | FB |  | MC_AxesObserve_instance (<br>Master :=*parameter,*<br>Slave :=*parameter,*<br>Enable :=*parameter,*<br>ReferenceType :=*parameter,*<br>PermittedDeviation :=*parameter,*<br>Enabled =>*parameter,*<br>Invalid =>*parameter,*<br>Busy =>*parameter,*<br>DeviatedValue =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Enable | Enable | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Enable* is TRUE. |
| ReferenceType | Position Type Selection | _eMC_ RERERENCE_TYPE | 0: _mcCommand<br>1: _mcFeedback | 0*1 | Specify the position type.<br>0: Command position (value calculated in the previous primary period)<br>1: Actual position (value obtained in the same primary period) |
| Permitted Deviation | Permitted Following Error | LREAL | Non-negative number | 0 | Specify the permitted maximum value of the following error between the master and slave axes. The unit is command units.*2 |

*1 The default value for an enumeration variable is actually not the number, but the enumerator.

*2 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Enabled | Enable | BOOL | TRUE or FALSE | TRUE when the axis is being controlled. |
| Invalid | Excessive Following Error between Axes | BOOL | TRUE or FALSE | TRUE when the permitted following error between axes is exceeded. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| DeviatedValue | Following Error between Axes | LREAL | Negative number, positive number, or 0 | Contains the difference between the specified master and slave axes. The unit is command units.*1 |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | *2 | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Enabled | When *Enable* changes to TRUE. | • When *Error* changes to TRUE.<br>• When *Enable* changes to FALSE. |
| Invalid | When the permitted following error between axes is exceeded. | • When *Error* changes to TRUE.<br>• When *Enable* changes to FALSE.<br>• When the permitted following error between axes is not exceeded. |
| Busy | When *Enable* changes to TRUE. | • When *Error* changes to TRUE.<br>• When *Enable* changes to FALSE. |
| DeviatedValue* | When *Enable* is TRUE. | --- |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

* *DeviatedValue* does not return to FALSE when *Enable* changes to FALSE.

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Master | Master Axis | _sAXIS_REF | --- | Specify the master axis.* |
| Slave | Slave Axis | _sAXIS_REF | --- | Specify the slave axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

**Precautions for Correct Use**

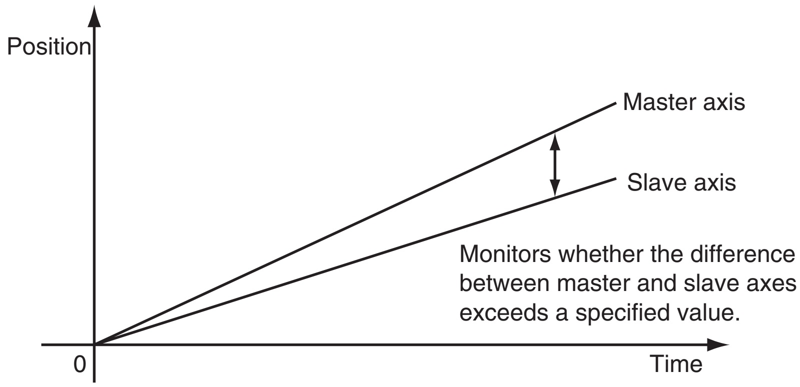
If you specify the same axis for the master axis and slave axis, a Master and Slave Defined as Same Axis minor fault (error code 5436 hex) will occur.

**Additional Information**

You can also set axes that belong to groups.

## Function

- If the command positions of the specified *Master* (Master Axis) and *Slave* (Slave Axis) or the difference between the actual positions exceeds the permitted following error, *Invalid* (Excessive Following Error between Axes) changes to TRUE.
  *Invalid* (Excessive Following Error between Axes) changes to TRUE when the following conditions are met.
  When |*DeviatedValue* (Following Error between Axes)| > *PermittedDeviation* (Permitted Following Error)
- The operation of the axis is not affected by this instruction.
- Use the state of the *Invalid* (Excessive Following Error between Axes) output variable to program processes, such as stopping an axis.




**Precautions for Correct Use**

- Use the same Count Mode for the *Master* (Master Axis) and *Slave* (Slave Axis).
  If a different mode is set, the axes will be compared in Linear Mode.
  Even if both axes are in Rotary Mode, comparisons are made in Linear Mode if the ranges set for the ring counters are not the same.
- If *PermittedDeviation* (Permitted Following Error) contains a non-terminating decimal number, e.g., resulting from division, error may cause unexpected processing results.

## Instruction Details

This section describes the instruction in detail.

### ● *ReferenceType* (Position Type Selection)

Any of the following position types can be selected for the master axis to which the slave axis is synchronized.

- *_mcCommand*: Command position (value calculated in the previous primary period)
  The master axis command position that was calculated in the previous primary period is used for the current period.
  The command value that was calculated for the master axis in the last primary period is used to calculate the command position of the slave axis in the current period.
- *_mcFeedback*: Value obtained in the same primary period
  The actual position of the master axis that was obtained in the same primary period is used.

## ● Relationship between Axis Types and Position Types

The relationship between the axis types that you can monitor and position types that is monitored is shown below.

| Axis type | ReferenceType | |
|---|---|---|
| | *_mcCommand* | *_mcFeedback* |
| Servo axis | OK | OK |
| Encoder axis | No* | OK |
| Virtual servo axes | OK | OK |
| Virtual encoder axis | No* | OK |

* A Position Type Selection Out of Range error (error code: 5430 hex) occurs when the instruction is executed.

## ● Calculation Examples for *DeviatedValue* (Following Error between Axes)

The *DeviatedValue* (Following Error between Axes) is calculated as described below.

### Linear Mode

#### *ReferenceType* (Position Type Selection) Set to *_mcCommand*

*DeviatedValue* (Following Error between Axes) = Command current position of *Master* (Master Axis) – Command current position of *Slave* (Slave Axis)

#### *ReferenceType* (Position Type Selection) Set to *_mcFeedback*

*DeviatedValue* (Following Error between Axes) = Actual current position of *Master* (Master Axis) – Actual current position of *Slave* (Slave Axis)

### Rotary Mode

#### *ReferenceType* (Position Type Selection) Set to *_mcCommand*

The *DeviatedValue* (Following Error between Axes) is the shorter distance between the command current position of the *Master* (Master Axis) and the command current position of the *Slave* (Slave Axis) in the range of the ring counter. The sign of the *DeviatedValue* (Following Error between Axes) is the sign for the shorter direction, as given below.

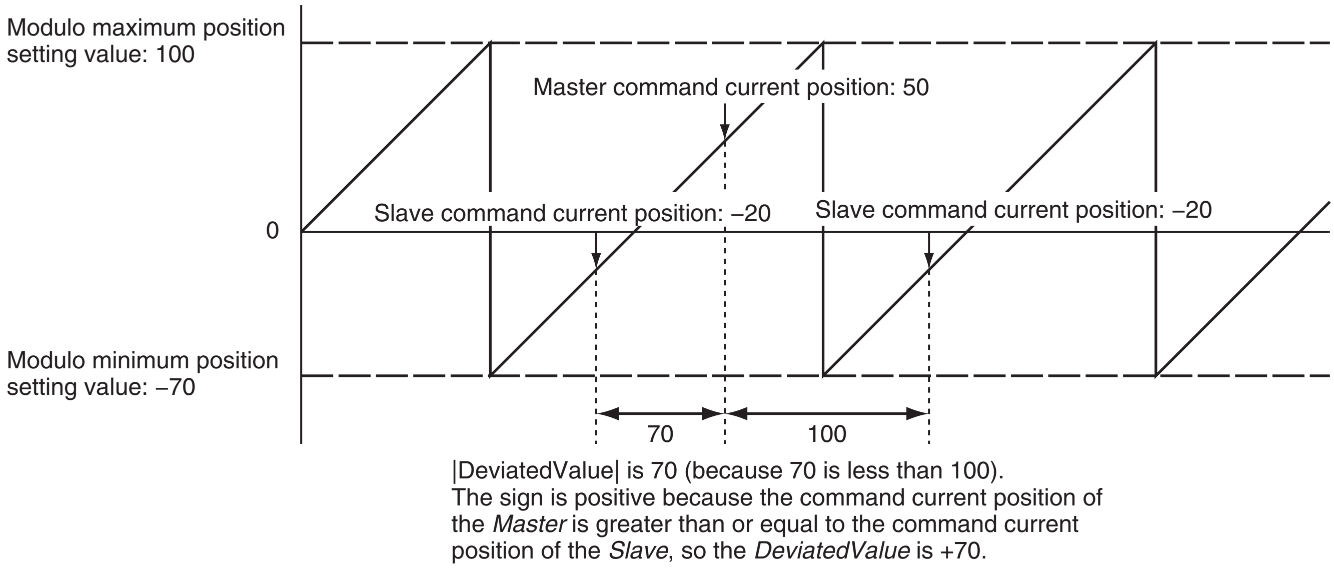
- If the command current position of the *Master* (Master Axis) is greater than or equal to the command current position of the *Slave* (Slave Axis), the value is positive.
- If the command current position of the *Master* (Master Axis) is less than the command current position of the *Slave* (Slave Axis), the value is negative.

#### *ReferenceType* (Position Type Selection) Set to *_mcFeedback*

The *DeviatedValue* (Following Error between Axes) is the shorter distance between the actual current position of the *Master* (Master Axis) and the actual current position of the *Slave* (Slave Axis) in the range of the ring counter. The sign of the *DeviatedValue* (Following Error between Axes) is the sign for the shorter direction, as given below.

- If the actual current position of the *Master* (Master Axis) is greater than or equal to the actual current position of the *Slave* (Slave Axis), the value is positive.
- If the actual current position of the *Master* (Master Axis) is less than the actual current position of the *Slave* (Slave Axis), the value is negative.

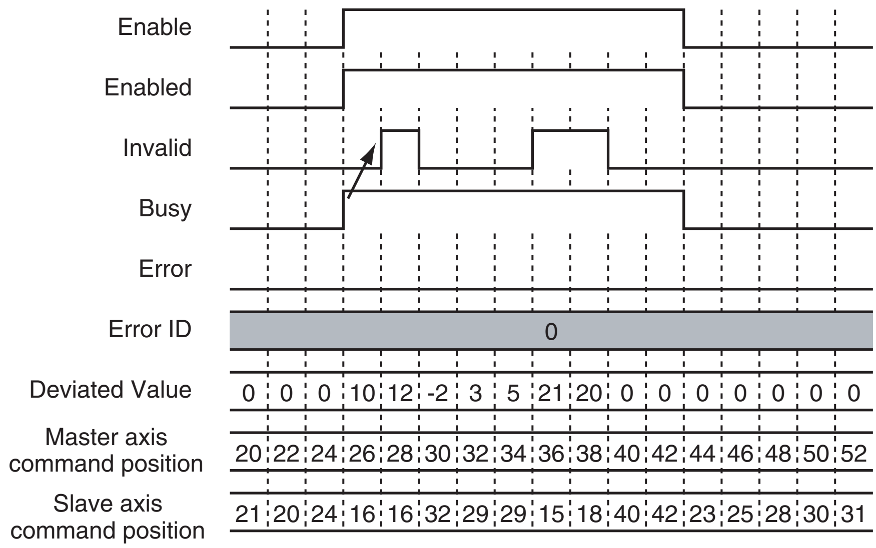
*DeviatedValue* (Following Error between Axes) Calculation Example in Rotary Mode when *ReferenceType* (Position Type Selection) Is Set to *_mcCommand*




If *ReferenceType* (Position Type Selection) is *_mcFeedback* in Rotary Mode, the "command current position" in the above diagram would be the "actual current position."

## Timing Charts

An error for this instruction does not affect the operation of the axis or axes group. A timing chart is given below for when *PermittedDeviation* (Permitted Following Error) is 10.0.




## Re-execution of Motion Instructions

You cannot re-execute enable-type motion instructions.

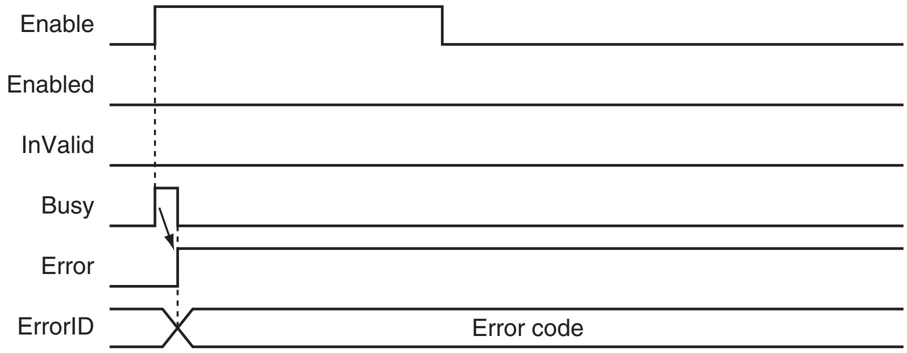
## Multi-execution of Motion Instructions

There are no restrictions for multi-execution of instructions.

For details on re-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Errors

- If an error occurs during instruction execution, *Error* will change to TRUE.
- You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).
- Error information for this instruction is output to Minor Fault in the MC Common motion variable.

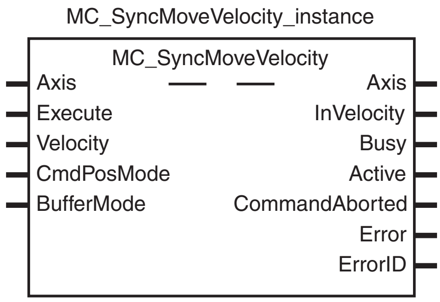



### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_SyncMoveVelocity

The MC_SyncMoveVelocity instruction outputs the value set for the target velocity every primary period to the Servo Drive in Cyclic Synchronous Velocity Mode.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_SyncMoveVelocity | Cyclic Synchronous Velocity Control | FB |  | MC_SyncMoveVelocity_ instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>Velocity :=*parameter,*<br>CmdPosMode :=*parameter,*<br>BufferMode :=*parameter,*<br>InVelocity =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Velocity | Target Velocity | LREAL | Negative number, positive number, or 0 | 0 | Set the target velocity.<br>0: Set the velocity command value to 0.<br>Positive value: Move in the positive direction.<br>Negative value: Move in the negative direction. |
| CmdPosMode | Command Current Position Count Selection | _eMC_CMDPOS_ MODE | 0: _mcCount | 0* | 0: Use the actual current position and update the command current position. Home remains defined. |
| BufferMode | Buffer Mode Selection | _eMC_BUFFER_ MODE | 0: _mcAborting<br>1: _mcBuffered | 0* | Specify the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered |

* The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| InVelocity | Target Velocity Reached | BOOL | TRUE or FALSE | TRUE when the command velocity reaches the target velocity. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when control is in progress. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| InVelocity | When the target velocity is reached. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Busy | When *Execute* changes to TRUE. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the instruction is started. | • When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is canceled due to another instruction.<br>• When this instruction is canceled due to an error in another instruction.<br>• When this instruction is executed while there is an axis error.<br>• When you start this instruction during MC_Stop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

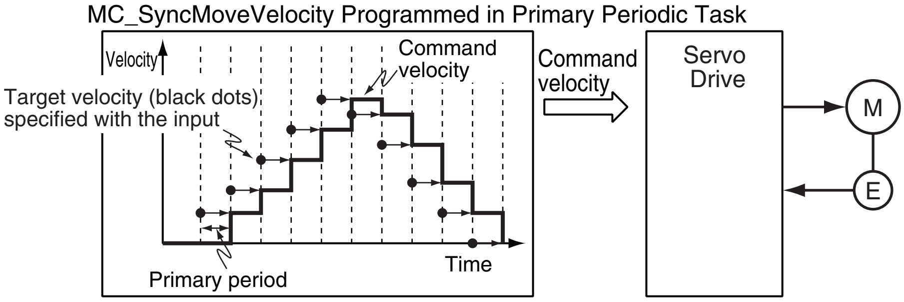
* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

- The MC_SyncMoveVelocity instruction outputs the target velocity from the user program every primary period to the Servo Drive in Cyclic Synchronous Velocity (CSV) Mode.
- When *Execute* changes to TRUE, the Control Mode of the Servo Drive is changed and a command velocity is output.

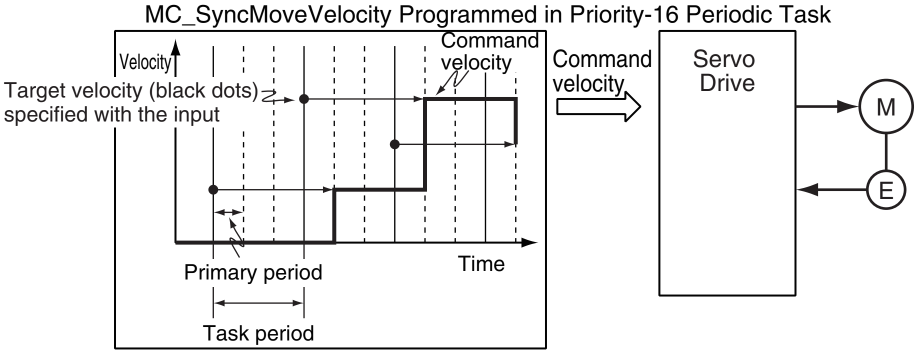
- If this instruction is executed in the primary periodic task, the target velocity is reached in the next period.

MC_SyncMoveVelocity Programmed in Primary Periodic Task




- If this instruction is executed in the priority-16 periodic task, the target velocity is reached in the next task period.

MC_SyncMoveVelocity Programmed in Priority-16 Periodic Task




**Precautions for Correct Use**

Refer to *Precautions for Master and Auxiliary Axes in Synchronized Control* on page 1-7 for precautions on using this instruction for the master axis of synchronized control.

**Additional Information**

The MC_SetOverride (Set Override Factors) instruction is not effective for the MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control) instruction.

### ● Mapping Data Objects

To use the MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control) instruction, map the following object data in the Detailed Settings Area of the Axis Basic Settings Display of the Sysmac Studio.

- Target velocity (60FF hex)
- Modes of operation (6060 hex)
- Modes of operation display (6061 hex)

If even one of the required objects is not set, a Process Data Object Setting Missing error (error code 3461 hex) occurs.

For details on mapping data objects, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Instruction Details

This section describes the instruction in detail.

- ***Velocity* (Target Velocity)**

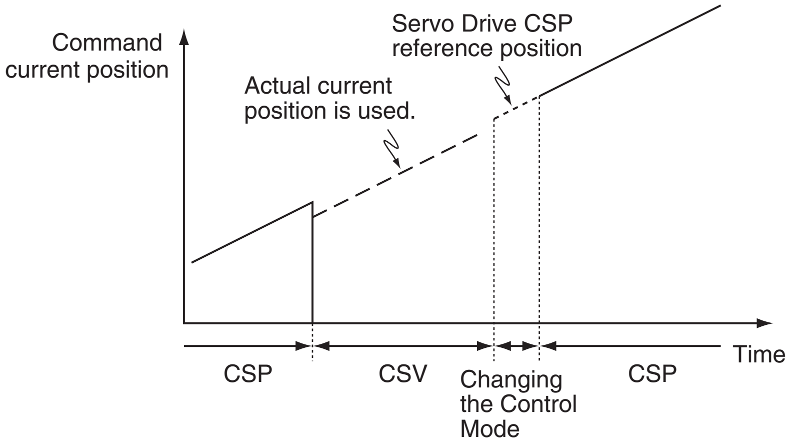
The *Velocity* (Target Velocity) input variable can be set to LREAL data in reference to 0.
The axis moves in the positive direction for a positive value and in the negative direction for a negative value.
If 0 is set, the command velocity is 0 and *Status.Continuous* (Continuous Motion) remains TRUE.
You can set *Velocity* (Target Velocity) from the user program every period.
If the specified target velocity is different from the last period, the new target velocity is used. If the specified target velocity is the same as the last period, the previous target velocity is used.

**Precautions for Correct Use**

- When you set the target velocity, make sure that an excessive load is not placed on the mechanical composition of the system for the change in velocity.
- If the axis that you use in this instruction is the master axis for synchronized control, setting the target velocity of the main axis may cause the slave axis to move suddenly.
- When the Control Mode is changed, the command current position may change suddenly.

- **Command Current Position**

If you select *_mcCount* for the *CmdPosMode* (Command Current Position Count Selection) input variable, the command current position will be the actual current position from the previous period when this instruction is executed. The actual current position is used until the instruction is ended.
While the OMRON G5-series Servo Drive is processing the switch to CSP Mode, the CSP reference position that was mapped in advance is sent in the PDO. Until processing to switch from CSV to CSP Mode is completed, this reference position is used as the command current position. When switching to CSP Mode is completed, the command current position is set to the command position.




- **When Using an OMRON G5-series Servo Drive**

To use the CSP reference position of the Servo Drive when changing the Control Mode, map the CSP Reference Position (4020 hex) to process data. Map the CSP Reference Position (4020 hex) to process data in the PDO Edit Tab Page of the Sysmac Studio. Then map the CSP Change Reference Position in the MC Function Model and the CSP Reference Position (4020 hex) in the Detailed Settings Area of the Axis Basic Settings Tab Page.

**Precautions for Correct Use**

If the CSP Reference Position (4020 hex) is mapped to a PDO, set the primary period to 1 ms or longer. If the primary period is less than 1 ms, an error will occur in the G5-series Servo Drive. Refer to the *G5-series AC Servomotors/Servo Drives With Built-in EtherCAT Communications User's Manual* (Cat. No. I576) for details.

### When Using a Servo Drive Other Than an OMRON G5-series Servo Drive

An error may occur in processing to switch the Command Mode for some Servo Drives when this instruction is executed or when an instruction that uses CSP Mode is executed during execution of this instruction.
If that occurs, stop the axis (velocity of 0) and then execute this instruction or use multi-execution of instructions that use the CSP Mode.

### Stop Processing

This Stop Modes and command velocities that are used to stop axis motion are described below.
For a deceleration stop, the target velocity of this instruction is used as the initial velocity and the axis is decelerated to a stop with the deceleration rate for the specified Stop Mode.

#### Stopping with the MC_ImmediateStop Instruction

The command velocity is changed to 0. The Control Mode is changed to CSP Mode when the change criterion that is given below is satisfied.

#### Stopping with the MC_Stop Instruction

The command velocity is changed to 0 at the deceleration rate of the instruction. The Control Mode is changed to CSP Mode when the change criterion that is given below is satisfied.

#### Stopping for a Minor Fault Level Error

The command velocity is changed to 0 at the deceleration rate of each error. The Control Mode is changed to CSP Mode when the change criterion that is given below is satisfied.

#### Stopping for a Major Fault Level Error or a Partial Fault Level Error

The command velocity is changed to 0. The Control Mode is changed to CSP Mode when the change criterion that is given below is satisfied.
However, depending on the error level, it may not be possible to switch the Control Mode normally, and the axis may stop in CSV Mode.

#### Stopping by Turning OFF the Servo

The command velocity is changed to 0 with the specified method. The Control Mode is not changed.

#### Stopping When the Operating Mode of the CPU Unit Changes to PROGRAM Mode

The command velocity is changed to 0 with the specified method. The Control Mode is changed to CSP Mode when the change criterion that is given below is satisfied.

### Change Criterion

Whether it is possible to change the Control Mode depends on Servo Drive specifications.
To ensure that the Control Mode is switched to CSP Mode during stop processing for stop instructions or errors, it is necessary to sufficiently decelerate the Servomotor first.
The Control Mode is changed to CSP Mode when the following criterion is met for three consecutive primary period tasks after the command velocity changes to 0.

Actual current velocity $\leq$ Maximum velocity $\times$ 0.1

### Recovery to Cyclic Synchronous Positioning Mode

It is not always possible to normally change to CSP Mode for a stop.
For example, it may not be possible when a partial fault level error occurs in the MC Function Module.
Therefore, processing to change to CSP Mode is performed when the *Status* output variable from the MC_Power (Power Servo) instruction changes to FALSE.

### Operation for Failure to Switch Control Mode

If the Servo Drive does not complete switching the Control Mode within 1 second after a Control Mode switch command is sent to the Servo Drive, a Error in Changing Servo Drive Control Mode (error code: 7439 hex) occurs.
Also, if the switching criterion is not met within 10 seconds after the command velocity is changed to 0 for a Control Mode switching command to the Servo Drive, an Error in Changing Servo Drive Control Mode (error code: 7439 hex) occurs.
When an Error in Changing Servo Drive Control Mode (error code: 7439 hex) occurs, the command velocity is changed to 0 and the Servo is turned OFF (free-run stop).

### Applicable Axes and Execution Condition

- You can use this instruction for a servo axis.
  To use this instruction, change *Enable* for the MC_Power instruction to TRUE (Servo ON).
- A virtual servo axis will acknowledge this instruction at any time.
  However, processing to switch the Control Mode of the Servo Drive is not performed.
- An error occurs if the instruction is executed for an encoder or virtual encoder axis.

### Axis Variable Status

*Status.Continuous* (Continuous Motion) in the Axis Variable status changes to TRUE.
Use *DrvStatus* (Servo Drive Status) in the Axis Variable to check the Control Mode that is set in the Servo Drive. The Servo Drive status is given in the following table.

| Name | Data type | Meaning | Description |
|---|---|---|---|
| CSP | BOOL | Cyclic Synchronous Position (CSP) Mode | TRUE when the Servo is ON and the Servo Drive is in CSP Mode. |
| CSV | BOOL | Cyclic Synchronous Velocity (CSV) Mode | TRUE when the Servo is ON and the Servo Drive is in CSV Mode. |
| CST | BOOL | Cyclic Synchronous Torque (CST) Mode | TRUE when the Servo is ON and the Servo Drive is in CST Mode. |

### Home Status

If *CmdPosMode* (Command Current Position Count Selection) is set to *_mcCount*, the home will remain defined.

### Overrides

Overrides are disabled for this instruction.

### Software Limits

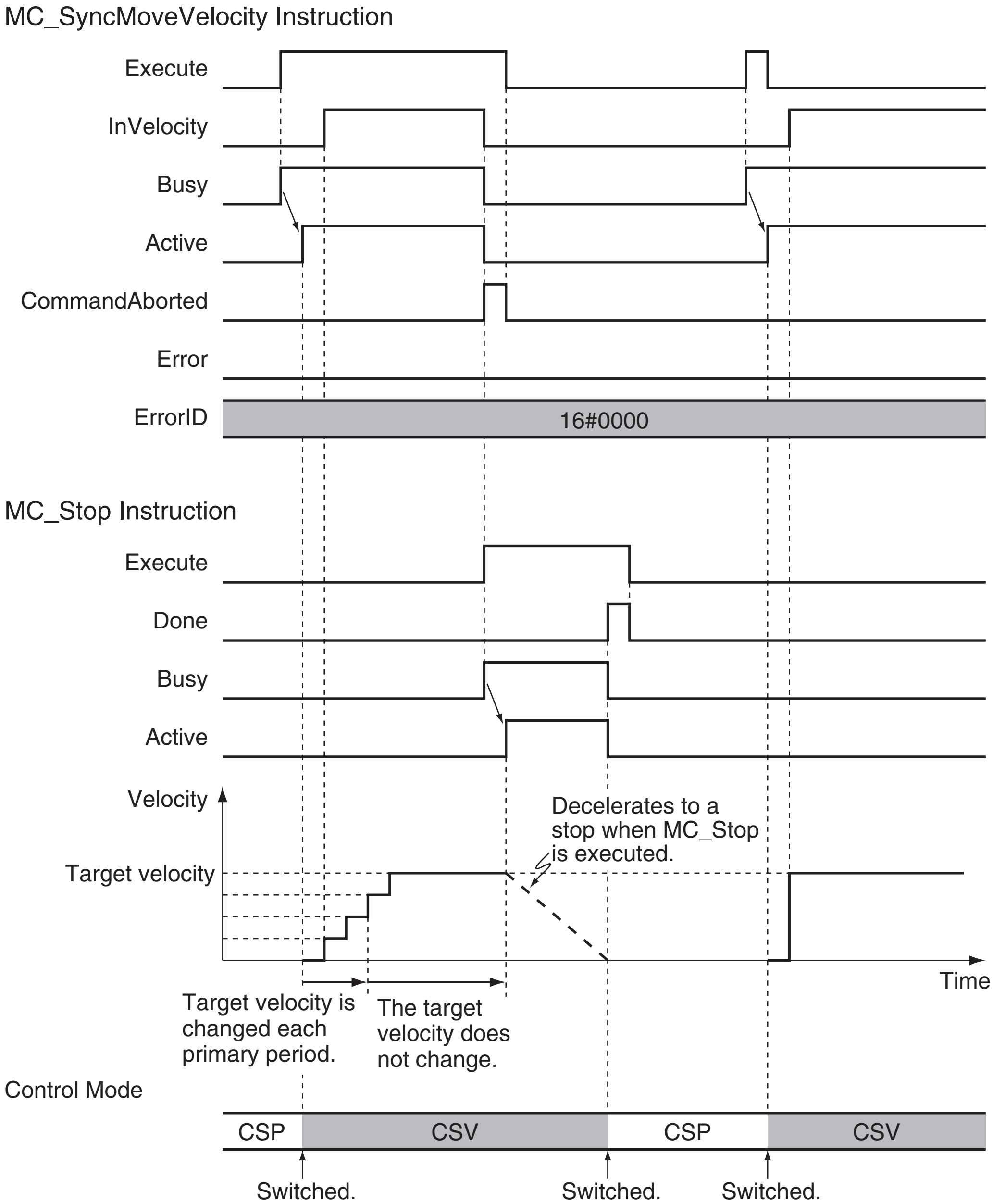
Software limits are enabled for this instruction. These are enabled even for the following axis parameter settings.

- Deceleration stopping enabled for command position
- Immediate stopping enabled for command position (stop using remaining pulses)

## Timing Charts

- *Busy* (Executing) changes to TRUE at the same time as *Execute* changes to TRUE. *Active* (Controlling) changes to TRUE in the next period.
- *InVelocity* (Target Velocity Reached) changes to TRUE when the command velocity reaches *Velocity* (Target Velocity).
- If another instruction aborts this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing), *Active* (Controlling), and *InVelocity* (Target Velocity Reached) change to FALSE.
- The MC_Stop instruction is used to stop this instruction.

The following timing charts show operation for when this instruction is used in the primary periodic task.




**Additional Information**

The MC Function Mode sends a command to the Servo Drive to change the Control Mode as shown in the above timing chart. The timing of implementing the change in the Servo Drive depends on Servo Drive specifications.

## Re-execution of Motion Control Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

You can execute this instruction with *BufferMode* (Buffer Mode Selection) set to *Aborting* or *Buffered* during execution of other instructions in the same as for the MC_MoveVelocity (Velocity Control) instruction.
The Control Mode is switched when processing the instruction is started.

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

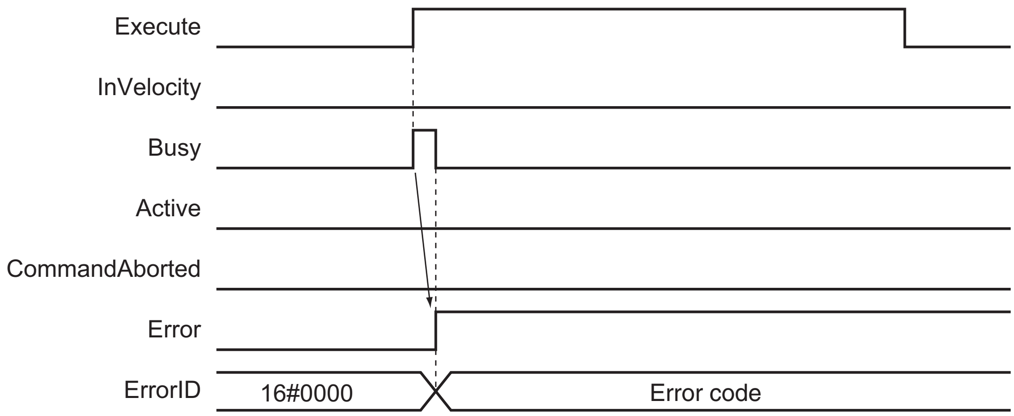
### ● Execution of Other Instructions during Instruction Execution

You can execute other instructions with *BufferMode* (Buffer Mode Selection) set to *Aborting* or *Buffered* during execution of this instruction in the same as for the MC_MoveVelocity (Velocity Control) instruction.
If the other instruction is buffered, then it is executed when *InVelocity* (Target Velocity Reached) changes to TRUE.
The Control Mode is switched when processing the instruction is started.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop.
You can find out the cause of the error by referring to the value output to *ErrorID* (Error Code).

### ● Timing Chart When Error Occurs




### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

MC_SyncMoveVelocity

3

Function

# MC_Reset

The MC_Reset instruction clears axis errors.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_Reset | Reset Axis Error | FB |  | MC_Reset_instance (<br>Axis :=*parameter,*<br>Execute :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Failure =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Failure | Failure End | BOOL | TRUE or FALSE | TRUE when the instruction is not executed normally. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When error clear processing is completed normally. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *Failure* changes to TRUE. |
| Failure | • When an error reset is executed while decelerating an axis to a stop for an error.<br>• When an error reset is executed during an axis error for an axis common error. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Axis | Axis | _sAXIS_REF | --- | Specify the axis.* |

* Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio. (The default axis variable names are *MC_Axis****.)

# Function

- The MC_Reset instruction starts error clear processing for the axis specified by *Axis* when *Execute* changes to TRUE. The error processing resets axis errors and, if errors have occurred in the Servo Drive, drive errors.
- You can use this instruction for any axis type.
- Error clear processing is executed only for axes with errors.
- If there is a drive error for an axis, the drive error is cleared first. Error clear processing is then performed. Reset processing for the drive error is continued until the drive error is cleared or continues for the Drive Error Reset Monitoring Time in the axis parameters.
- If this instruction is executed while the axis is decelerating to a stop for an error, the instruction is not executed and *Failure* will change to TRUE. *Failure* will also change to TRUE if an axis error that results from an MC common error cannot be cleared by this instruction. This is to ensure that the error is not reset before the axis stops.
  MC Common errors include MC Common Partial Faults and MC Common Minor Faults.
- Only errors that existed when *Execute* changes to TRUE are cleared. Errors that occur while clearing errors are not cleared.

**Precautions for Correct Use**

- The error clear processing that is performed by this instruction sometimes requires more than one control period.
- If an MC Common Partial Fault or MC Common Minor Fault occurs or the axis is in motion, *Failure* (Failure End) from the instruction will change to TRUE. Remove the cause of the error, and then retry the process until *Done* changes to TRUE.
- After you remove the cause of the error, execute the instruction only after you confirm that the axes have stopped completely.
  *Act.Vel* (Actual Current Velocity) in the Axis Variable is 0 if the axis is completely stopped. Use this to confirm when the axis is completely stopped.

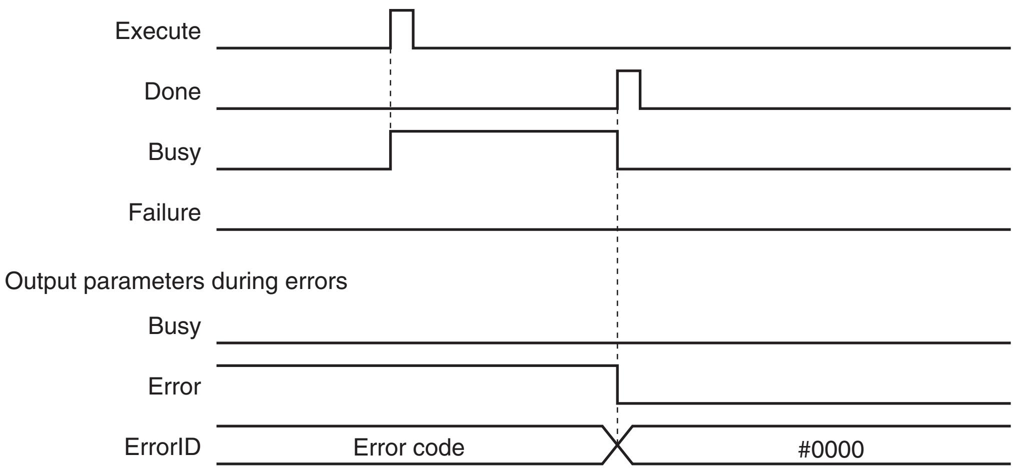
**Additional Information**

The following errors cannot be cleared with this instruction.

- All axis common errors: Execute the ResetMcError (Reset All Errors) instruction.
- All axes group errors: Execute the MC_GroupReset (Group Reset) instruction.

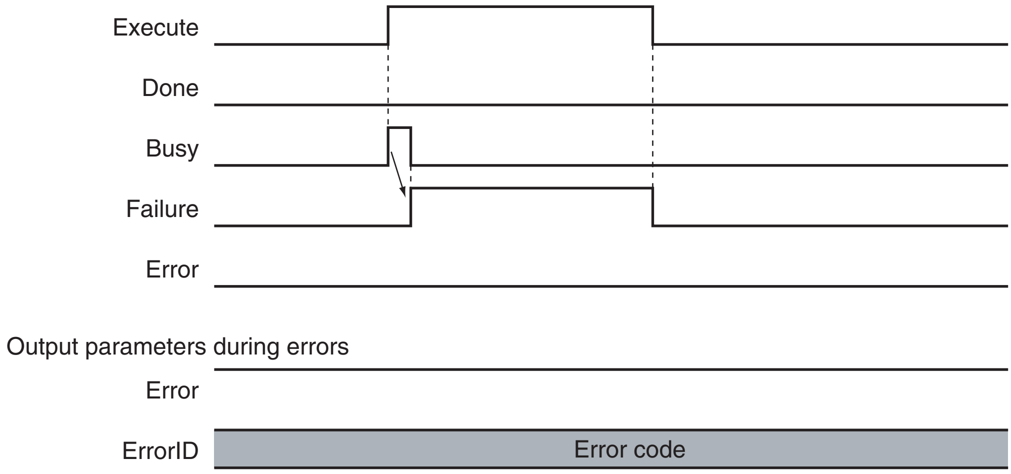
The causes of network errors, such as slave communications error, are not cleared by executing MC_Reset. Execute the ResetECATError instruction.

## Timing Charts



## Aborting the Instruction

The instruction is aborted if it is not possible to clear errors that occur when the axis is decelerating to a stop for an error or errors that occur during axis errors resulting from axis common errors.




## Error Codes

Refer to *A-1 Error Codes* for instruction errors.

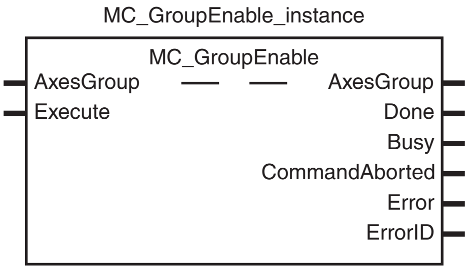
# 4

# Axes Group Instructions

This section describes the instructions to perform multi-axes coordinated control for the MC Function Module.

# MC_GroupEnable

The MC_GroupEnable instruction enables an axes group.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_GroupEnable | Enable Axes Group | FB |  | MC_GroupEnable_instance (<br>AxesGroup :=*parameter,*<br>Execute :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the instruction is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | When this instruction is executed while there is an axes group error. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| AxesGroup | Axes Group | _sGROUP_REF | --- | Specify the axes group.* |

* Specify an Axes Group Variable that was created in the Axes Group Basic Settings of the Sysmac Studio. (The default axes group variable names are *MC_Group****.)

## Function

Before describing the function of this instruction, the different states of axes groups are explained.

- Axes groups have two states, the *GroupEnable* and *GroupDisable* states.

| State | Description |
|---|---|
| *GroupEnable* | Multi-axes coordinated control is enabled. You can execute any multi-axes coordinated control instructions. |
| *GroupDisable* | Multi-axes coordinated control is disabled. You can execute only the following multi-axes coordinated control instructions.<br>• MC_GroupEnable (Enable Axes Group) instruction<br>• MC_GroupDisable (Disable Axes Group) instruction<br>• MC_GroupReset (Reset Axes Group Errors) instruction<br>• MC_GroupSetOverride (Set Group Overrides) instruction |

- To perform multi-axes coordinated control, an axes group must be in a *GroupEnable* state.

You can monitor the Axes Group Variables in the system-defined variables for motion control to see if axes groups are enabled or disabled.

| Name | Meaning | Description |
|---|---|---|
| _MC_GRP**.Status.Ready | Axes Group Ready | TRUE when the axes group is stopped and preparations to execute an axes group instruction are completed. |
| _MC_GRP**.Status.Disabled | Axes Group Disabled | TRUE when the axes group is disabled and stopped. |

**Note** "**" in "_MC_GRP**" is replaced by a number between 0 and 31.

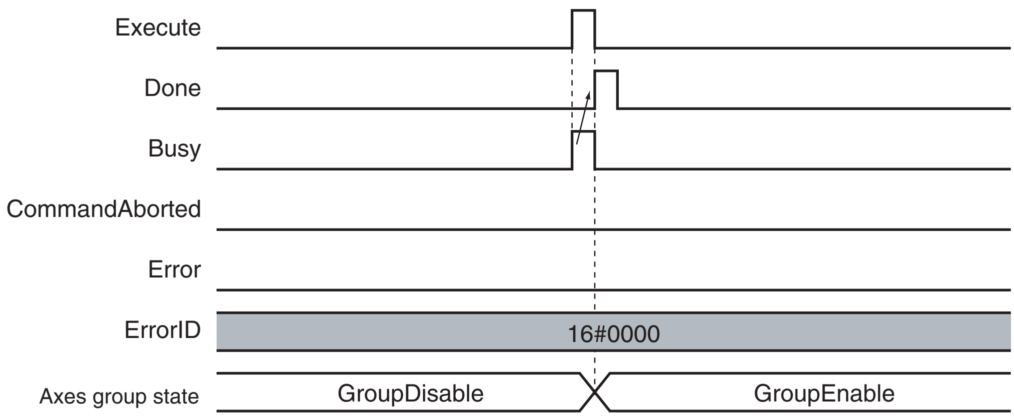
## Basic Function

- The MC_GroupEnable (Enable Axes Group) instruction places the axes group specified by *Axes-Group* into the *GroupEnable* state.
- When an axes group is in the *GroupEnable* state, you can execute any multi-axes coordinated control instructions for the axes group.
- You can set only servo axes and virtual servo axes in an axes group. An error will occur if you include other axis types.
- All axes that belong to an axes group must be in a stopped state to enable the group.
  An axis is stopped if the *Status.Disabled* (Axis Disabled) or *Status.Standstill* (Stopped) in the Axis Variable is TRUE.
- If there are axes that already belong to another axes group and the other axes group is enabled, the MC_GroupEnable instruction is not executed and an error will occur if you attempt to execute it.
- When an axes group is enabled, the axes in the axes group change to Coordinated Motion status.
  *Status.Coordinated* (Coordinated Motion) in the Axis Variable changes to TRUE.
- An axes group is disabled if the MC_GroupDisable (Disable Axes Group) instruction is executed, if operation is stopped by changing to PROGRAM mode, or if a MC Test Run is started.

**Precautions for Correct Use**

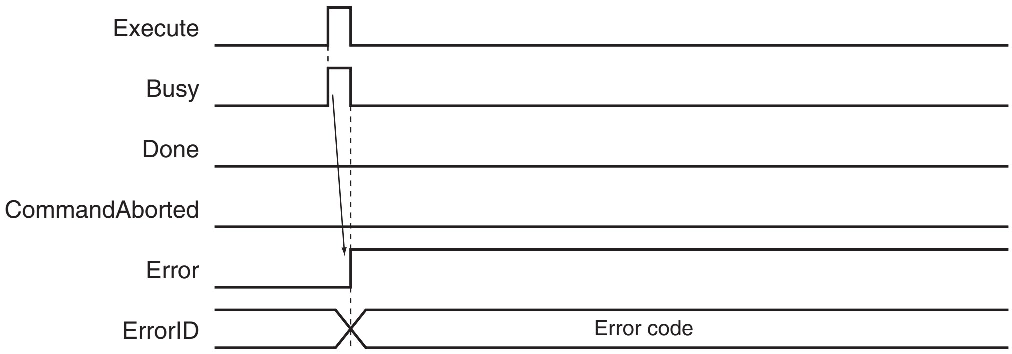
To use an axes group, create an axes group on the Sysmac Studio and download the settings to the CPU Unit. You cannot change the axes in an axes group from the user program. Use the Synchronize Menu of the Sysmac Studio to download the project.

## Timing Charts



## Errors

If an error occurs during execution of the MC_GroupEnable instruction, *Error* will change to TRUE. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).




**Additional Information**

Errors do not occur for individual axes in an axes group even if an error occurs for the axes group.

### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_GroupDisable

The MC_GroupDisable instruction disables an axes group.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_GroupDisable | Disable Axes Group | FB | MC_GroupDisable_instance<br>MC_GroupDisable<br>AxesGroup — — AxesGroup<br>Execute Done<br>Busy<br>CommandAborted<br>Error<br>ErrorID | MC_GroupDisable_instance (<br>AxesGroup :=*parameter,*<br>Execute :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the axes group state is changed to *GroupDisable*. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | Never changes to TRUE. (Reserved) | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |







## In-Out Variables

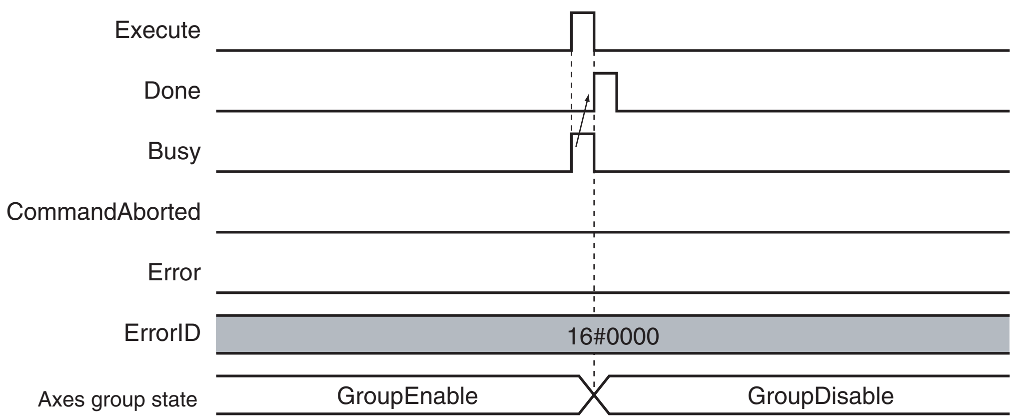
| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| AxesGroup | Axes Group | _sGROUP_REF | --- | Specify the axes group.* |

* Specify an Axes Group Variable that was created in the Axes Group Basic Settings of the Sysmac Studio. (The default axes group variable names are *MC_Group****.)

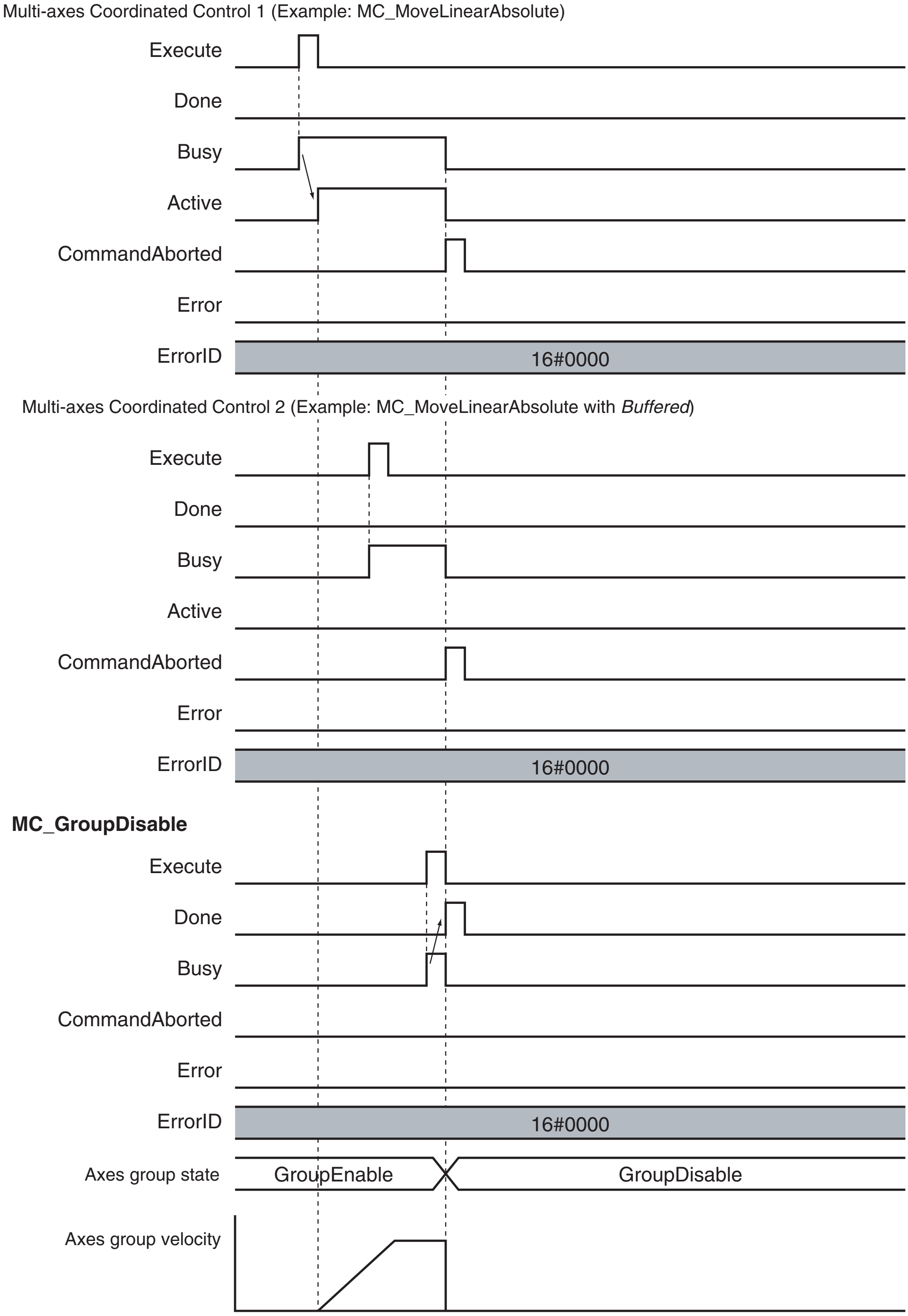
## Function

- The MC_GroupDisable instruction disables an axes group. For details on the axes group states, refer to *Function* on page 4-3.
- The MC_GroupDisable (Disable Axes Group) instruction places the axes group specified by *AxesGroup* into the *GroupDisable* state. When an axes group is in the *GroupDisable* state, the axes group stops acknowledging multi-axes coordinated control instructions.
- Any buffered instruction of the specified *AxesGroup* is cleared when the axes group state changes to *GroupDisable*.
- The axes group is disabled even while operation is stopped.
  When an axes group is disabled, the status of the axes in the axes group changes from TRUE for *Status.Coordinated* (Coordinated Motion) to the status of each axis.
  Use *Status* (Axis Status) in the Axis Variable to determine the status of each axis.

## Timing Charts

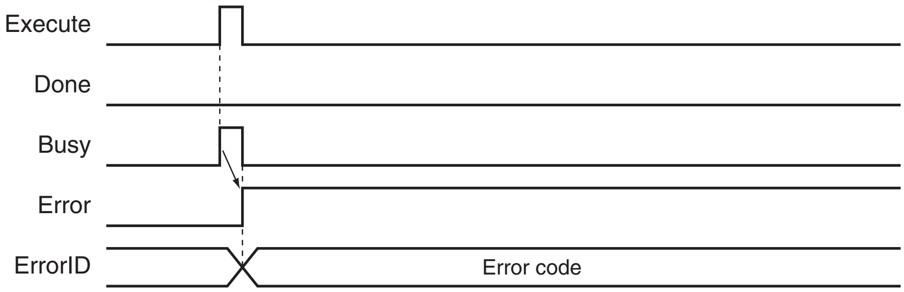


If you execute the MC_GroupDisable instruction for an *AxisGroup* that is under multi-axes coordinated control, *CommandAborted* of multi-axes coordinated control instruction will change to TRUE. *CommandAborted* of any buffered multi-axes coordinated control instruction will also change to TRUE. If the axes are moving, they will decelerate to a stop at the maximum deceleration rate for each axis.



## Errors

If an error occurs during execution of the MC_GroupDisable instruction, *Error* will change to TRUE. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

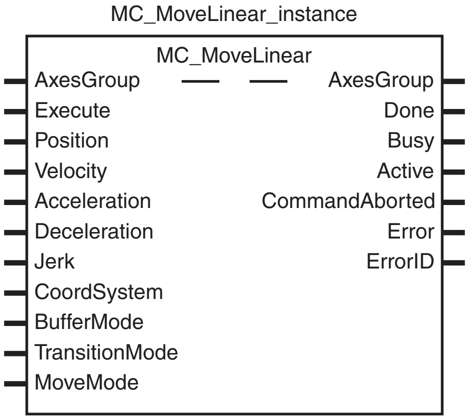



### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_MoveLinear

The MC_MoveLinear instruction performs linear interpolation.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_MoveLinear | Linear Interpolation | FB |  | MC_MoveLinear_instance (<br>AxesGroup :=*parameter,*<br>Execute :=*parameter,*<br>Position :=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=Å*parameter,*<br>Jerk :=*parameter,*<br>CoordSystem :=*parameter,*<br>BufferMode :=*parameter,*<br>TransitionMode<br>:=*parameter,*<br>MoveMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Position | Target Position | ARRAY [0..3] OF LREAL | Negative number, positive number, or 0 | 0 | Specify the target position for linear interpolation. The unit is command units.[*1] |
| Velocity[*2] | Target Velocity | LREAL | Positive number | 0 | Specify the target velocity. The unit is command units/s.[*1] |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/s$^2$.[*1] |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/s$^2$.[*1] |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/s$^3$. |
| CoordSystem | Coordinate System | _eMC_COORD_SYSTEM | 0: _mcACS | 0[*3] | Specify the coordinate system.<br>0: Axis coordinate system (ACS) |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| BufferMode | Buffer Mode Selection | _eMC_ BUFFER_ MODE | 0: _mcAborting<br>1: _mcBuffered<br>2: _mcBlendingLow<br>3: _mcBlendingPrevious<br>4: _mcBlendingNext<br>5: _mcBlendingHigh | 0*3 | Specify the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered<br>2: Blend with low<br>3: Blend with previous<br>4: Blend with next<br>5: Blend with high |
| TransitionMode | Transition Mode | _eMC_ TRANSITION_ MODE | 0: _mcTMNone<br>10: _mcTMCornerSuperimposed | 0*3 | Specify the path of motion.<br>0: Transition disabled<br>10: Superimpose corners |
| MoveMode | Travel Mode | _eMC_MOVE_ MODE | 0: _mcAbsolute<br>1: _mcRelative | 0*3 | Select the travel method.<br>0: Absolute positioning<br>1: Relative positioning |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 Always set the target velocity. If the axes are moved without setting a target velocity, an error will occur.

*3 The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axes group is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When positioning is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the axes move. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_GroupStop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| AxesGroup | Axes Group | _sGROUP_REF | --- | Specify the axes group.* |

* Specify an Axes Group Variable that was created in the Axes Group Basic Settings of the Sysmac Studio. (The default axes group variable names are *MC_Group****.)

# Function

- The MC_MoveLiner instruction performs linear interpolation for 2 to 4 axes.

**Precautions for Correct Use**

- An Instruction Execution Error with Undefined Home (error code: 5466 hex) occurs if home is undefined for any of the composition axes in the axes group.
- You cannot execute an instruction to perform linear interpolation if a limit input is ON for any of the logical axes that belong to the axes group.

## Instruction Details

This section describes the instruction in detail.

### Linear Interpolation Procedure

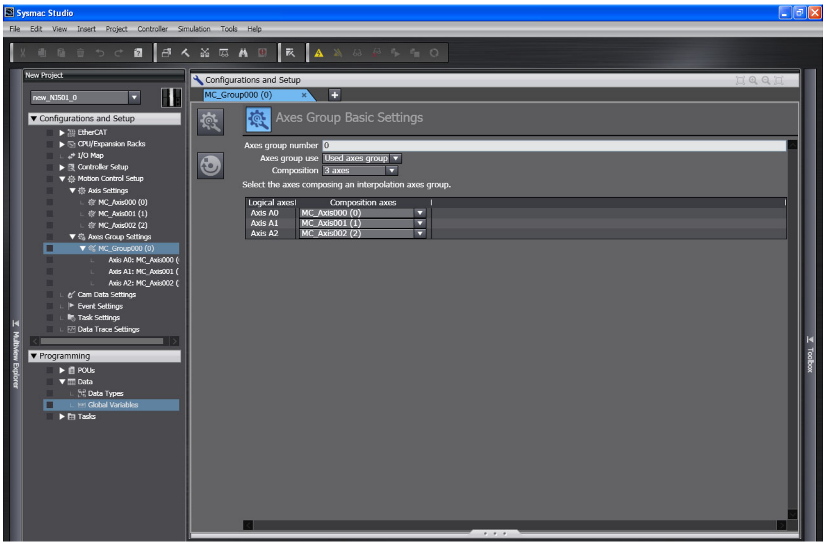
Use the following procedure to perform linear interpolation.

**1** Registering Axes Groups for Interpolation

- Select the axes group to perform interpolation.
  An axes group is represented by "_MC_GRP[***]".
- Specify the axis composition with *Compsition* in the Axis Group Variable. You can specify two to four axes.
- Specify the combination of axes to perform interpolation with the Axis Selection Axes Group Variable.
- Use logical axes (axis A0 to A3) for the axes, and not axis numbers (axis 0 to 63).
- Specify axis numbers 0 to 63 for the logical axes A0 to A3 in order from the lowest number using the Axis Selection Axes Group Variable.

| Logical axis | Axis number |
|---|---|
| Axis A0 | Axis 0 to Axis 63 |
| Axis A1 | Axis 0 to Axis 63 |
| Axis A2 | Axis 0 to Axis 63 |
| Axis A3 | Axis 0 to Axis 63 |

- In the Axes Group Basic Settings of the Sysmac Studio, select the axis composition to use and assign an axis number to the logical axis. The following example shows a 3-axis axes group that is called MC_Group000 with the following axes registered in it: *MC_Axis000*, *MC_Axis001*, and *MC_Axis002*.




### Precautions for Correct Use

An axis type error will occur and operation will end if an encoder axis or virtual encoder axis is included in the axes group. Make sure that you select only servo axes or virtual servo axes.

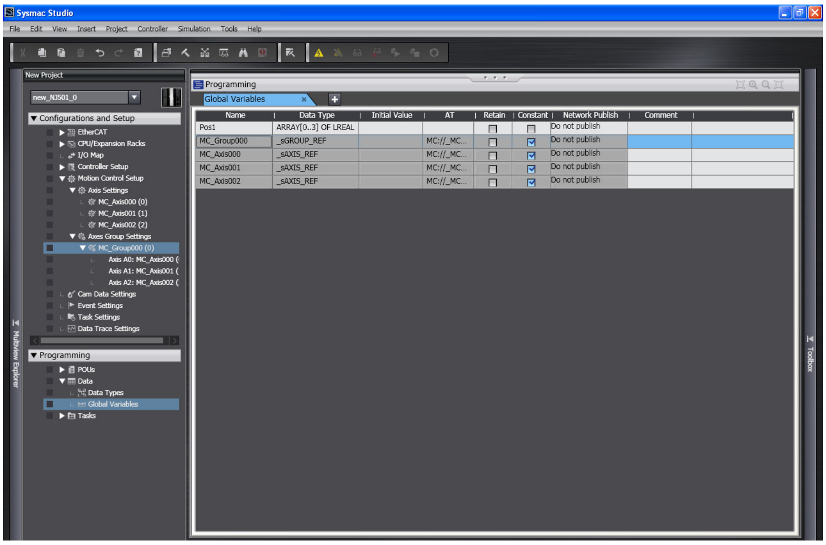
**2** Enabling the Axes Group

- Execute the MC_GroupEnable instruction to enable the registered axes group.
- Turn ON the Servo for the composition axes and change *Execute* of the MC_Stop instruction to FALSE.

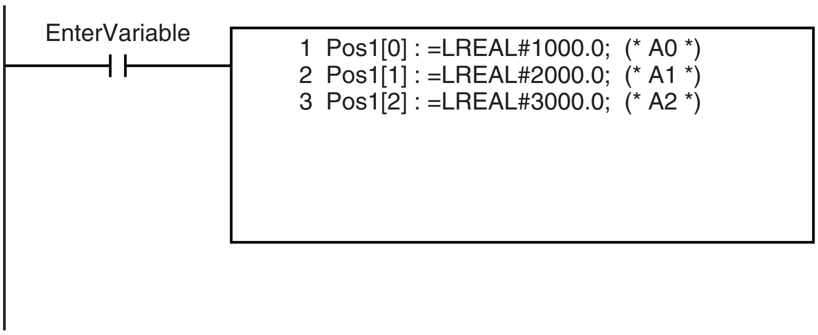
Using the Linear Interpolation instruction is now enabled.

## ● *Position* (Target Position)

- Set the target position in *Position* (Target Position) for all of the axes specified in the Axis Selection Axes Group Variable.
- You must create a 1×4 array variable in the Sysmac Studio to assign *Position* (A0, A1, A2, and A3). You can use any variable name.
  Assign the target positions for the axis to the elements of that array.
  Always create a 1×4 array even if there are fewer than four axes in the axes group.
  An example is shown below when the *Pos1* array variable is declared on the Sysmac Studio.



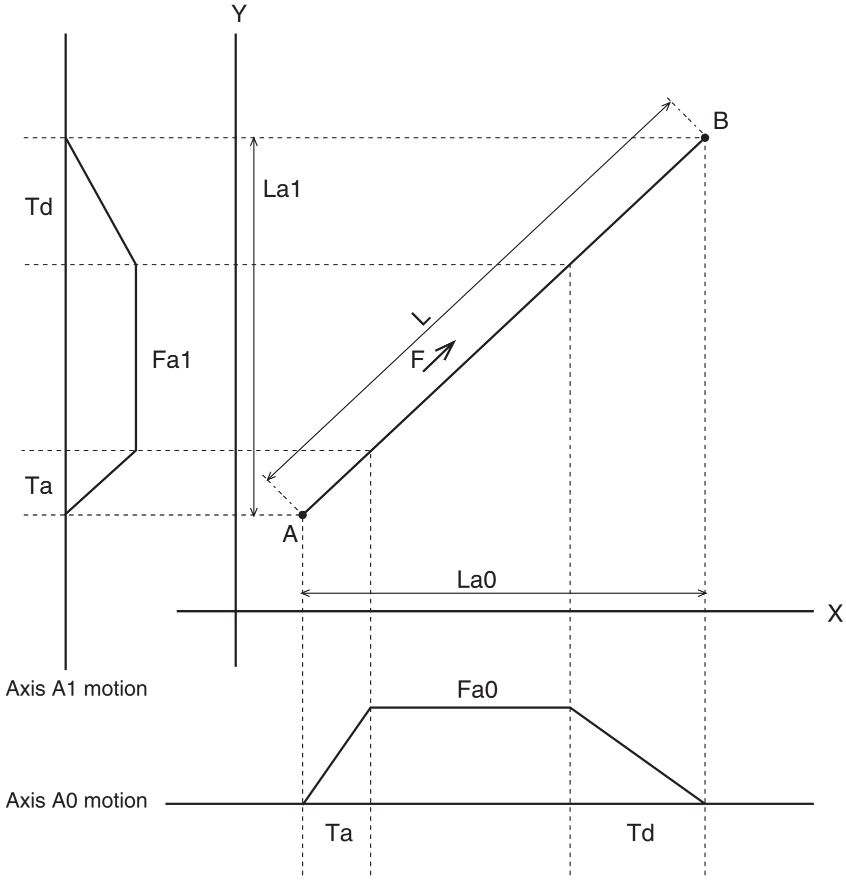
The following example shows assigning the target positions to *Pos1* with inline ST.
In the figure, target positions (1000.0, 2000.0, 3000.0) are assigned to axes A0 to A2.




- If an axis with the Count Mode set to Rotary Mode is set as an interpolation axis and you specify absolute position, the target value will be the same as if *Direction* was set to *No direction specified*. For details, refer to *Direction* on page 3-42.

- ***Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), *Deceleration* (Deceleration Rate), and *Jerk***

Set *Velocity, Acceleration, Deceleration,* and *Jerk* to specify the interpolation velocity, acceleration rate, deceleration rate, and jerk for linear interpolation. Linear interpolation separates the interpolated motion into motion on each axis. As an example, the following figure shows linear interpolation of 2 axes from point A to point B.




For linear interpolation of four axes, the interpolation velocity and travel distance of each axis determine the target velocities as shown below.

F: Specified interpolation feeding velocity

Fa0: Interpolation feeding velocity based on expansion of F to axis A0

Fa1: Interpolation feeding velocity based on expansion of F to axis A1

Fa2: Interpolation feeding velocity based on expansion of F to axis A2

Fa3: Interpolation feeding velocity based on expansion of F to axis A3

Ta: Interpolation acceleration time

Td: Interpolation deceleration time

L: Travel distance on the specified path

La0, La1, La2, and La3: Travel distances of axis A0, axis A1, axis A2, and axis A3.

L, Fa0, Fa1, Fa2, and Fa3 can be expressed with the following formulas.

$$Fa0 = F \times \frac{La0}{L}$$

$$Fa1 = F \times \frac{La1}{L}$$

$$Fa2 = F \times \frac{La2}{L}$$

$$Fa3 = F \times \frac{La3}{L}$$

$$L = \sqrt{La0^2 + La1^2 + La2^2 + La3^2}$$

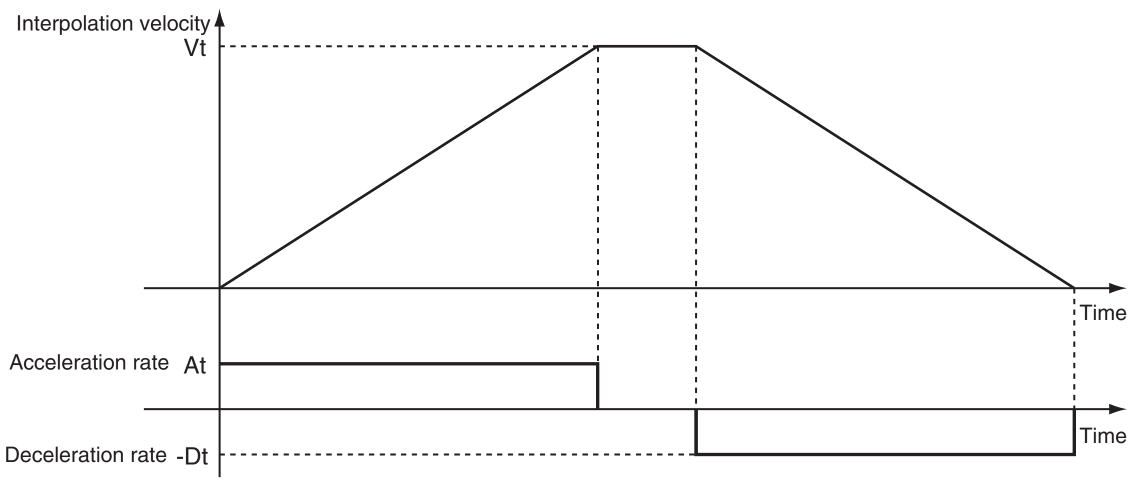
### *Velocity* (Target Velocity)

- An interpolation velocity specification error will occur if *Velocity* (Target Velocity) is set to 0. All axes will stop if an axis in the specified axes group is in operation.
- If any of the calculated target velocities Fa0 to Fa3 for *Velocity* (Target Velocity) exceed the maximum velocity, the *Velocity* (Target Velocity) will be automatically adjusted so that one of the axes operates at the maximum velocity.

### *Jerk*

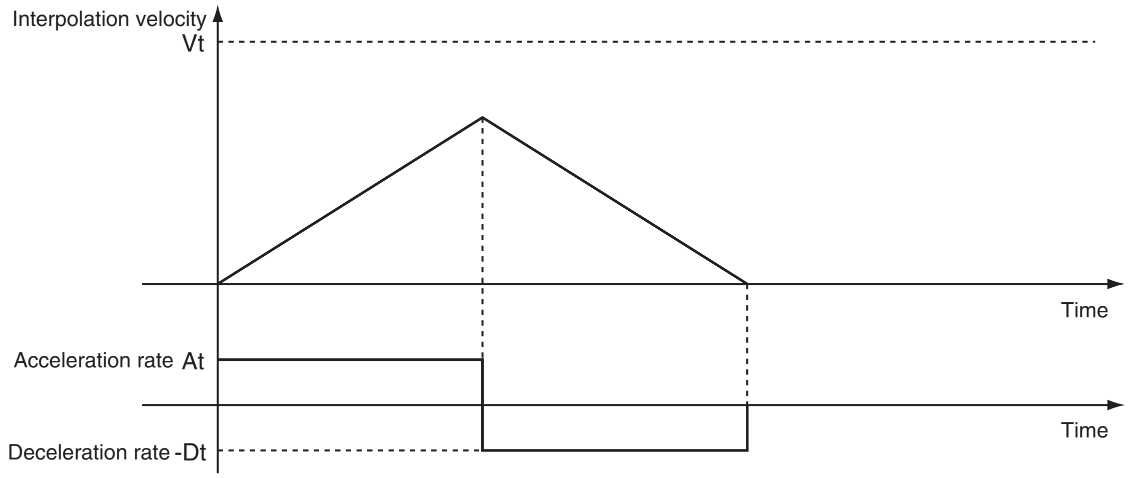
The relationships between *Acceleration* (Acceleration Rate), *Deceleration* (Deceleration Rate), and *Velocity* (Target Velocity) when *Jerk* is set to 0 and when it is set to any other value are shown below.

- *Jerk* Set to 0
  The command value for the velocity is created with acceleration rate At and deceleration rate Dt.



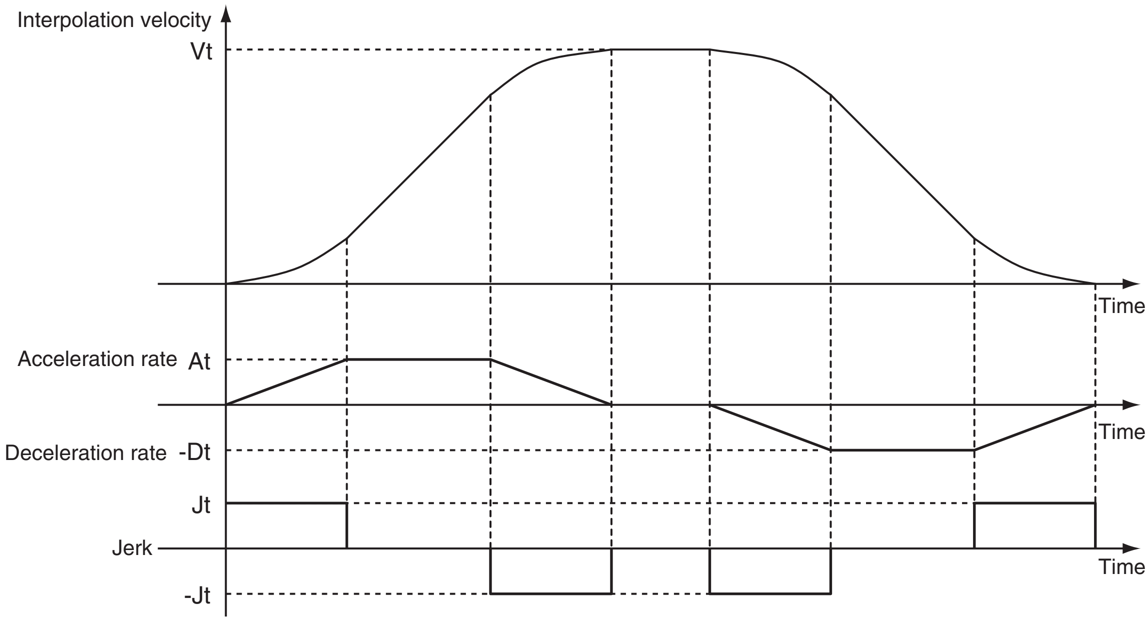

Vt: Specified interpolation velocity, At: Specified acceleration rate, Dt: Specified deceleration rate.

- Short Travel Distance When *Jerk* Is 0
  The interpolation velocity will not reach the specified Vt (Target Velocity).



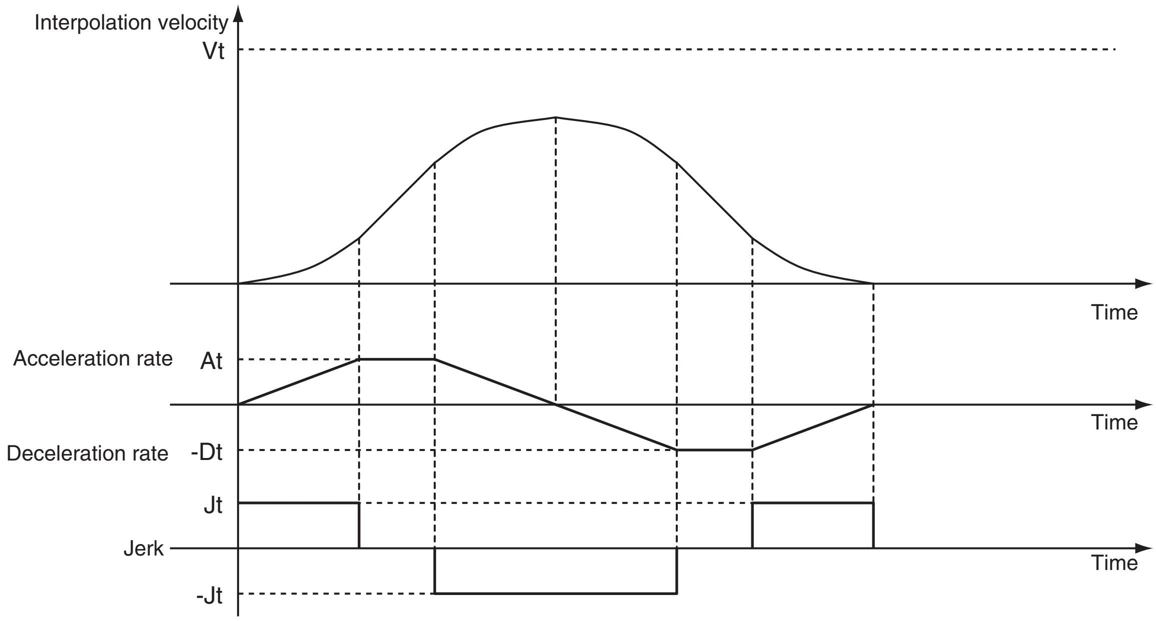

Vt: Specified interpolation velocity, At: Specified acceleration rate, Dt: Specified deceleration rate.

- *Jerk* Set to Value Other Than 0
  The command value for the velocity is created with At as the upper acceleration limit and Dt as the upper deceleration limit.




Vt: Specified interpolation velocity, At: Specified acceleration rate, Dt: Specified deceleration rate,
Jt: Specified jerk

- Short Travel Distance When *Jerk* Is Not 0
  The interpolation velocity will not reach the specified Vt (Target Velocity).




Vt: Specified interpolation velocity, At: Specified acceleration rate, Dt: Specified deceleration rate,
Jt: Specified jerk

**Additional Information**

- If 0 is specified for *Acceleration* (Acceleration Rate), the specified *Velocity* (Target Velocity) is used immediately.
- If 0 is specified for *Deceleration* (Deceleration Rate), the axis stops immediately. However, if the Buffer Mode is set to a blending mode, axis operation will change to the target velocity specified by the next operation without stopping. For details, refer to *BufferMode (Buffer Mode Selection)* on page 4-19.
- When the *Acceleration* (Acceleration Rate) or *Deceleration* (Deceleration Rate) is 0, the jerk setting is disabled.

## ● *CoordSystem* (Coordinate System)

*CoordSystem* specifies the coordinate system to use for linear interpolation. Only an axis coordinate system (ACS) consisting of two or more axes is supported.

## ● *BufferMode* (Buffer Mode Selection)

- *BufferMode* specifies how to join the axis motions for this interpolation instruction and the previous interpolation instruction.
- There are the following six settings.

| Buffer Mode Selection | | Description |
|---|---|---|
| Aborting | | Aborts the instruction being executed and switches to this instruction.<br>If the direction of axis motion is reversed by switching instructions, the motion will be reversed according to the Operation at Reversing axis parameter. |
| Buffered | | Buffers this instruction and executes it automatically after the current instruction is completed. |
| Blending | | Starts the buffered instruction at the velocity (transit velocity) at which the current instruction reaches the target position. The operation of the current instruction is changed so that the axes reach the target position at the transit velocity. There are four methods to specify the transit velocity. These are described below. You can also specify a Transition Mode as an option to the Blending Mode (see below). |
| | Blend with low | The lower of the target velocities of the current instruction and the buffered instruction is used as the transit velocity. |
| | Blend with previous | The target velocity of the current instruction is used as the transit velocity. |
| | Blend with next | The target velocity of the buffered instruction is used as the transit velocity. |
| | Blend with high | The higher of the target velocities of the current instruction and the buffered instruction is used as the transit velocity. |

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## ● *TransitionMode*

- *TransitionMode* specifies how to combine the paths created by the previous interpolation operation and the next interpolation operation.
- Set the *TransitionMode* to either *_mcTMNone* (Transition Disabled) or *_mcTMCornerSuperimposed* (Superimpose Corners).
- *TransitionMode* is enabled when blending is specified for *BufferMode*.
- An error will occur if you do not set *TransitionMode* to *_mcTMNone* (Transition Disabled) when blending is not used.

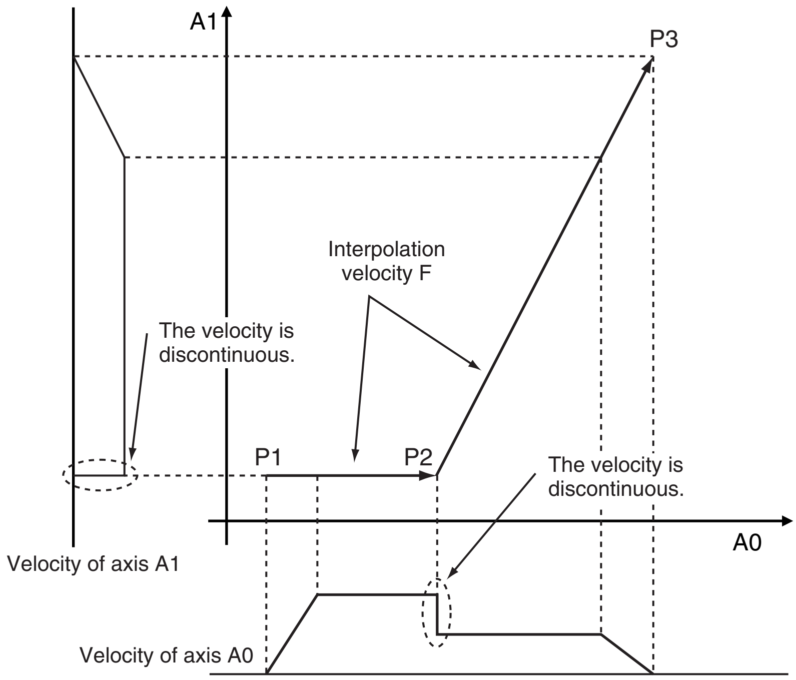
### Transition Disabled

The path is given priority when creating the velocity command value, so velocity command values of the axes may change rapidly when switching from one operation to the next.

Operation Example

The *Velocity* (Target Velocity), *BufferMode,* and *TransitionMode* when transitioning from P1 to P2, and from P2 to P3 are shown below.

- Motion from P1 to P2: *Velocity* = F, *BufferMode* = Aborting, *TransitionMode* = _mcTMNone (Transition Disabled)
- Motion from P2 to P3: *Velocity* = F, *BufferMode* = Blend with next, *TransitionMode* = _mcTMNone (Transition Disabled)
- The motion starts from position P1 and goes through position P2. Linear interpolation is performed to position P3.
- The linear interpolation velocity F is maintained when passing position P2. Because of this, the velocity is discontinuous at position P2 as shown in the following figure.



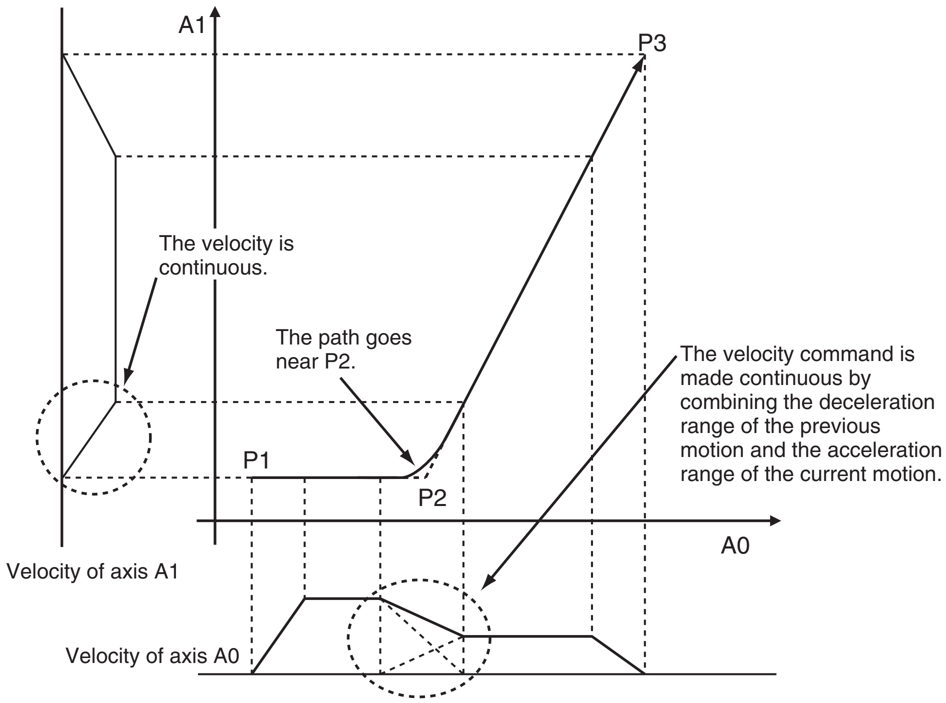
## Superimpose Corners

Use the superimpose corners specification when you want make the axes command velocities continuous.

Operation Example

The *Velocity* (Target Velocity), *BufferMode,* and *TransitionMode* when transitioning from P1 to P2, and from P2 to P3 are shown below.

- Motion from P1 to P2: *Velocity* = F, *BufferMode* = Aborting, *TransitionMode* = _mcTMNone (Transition Disabled)
- Motion from P2 to P3: *Velocity* = F, *BufferMode* = Blend with next, *TransitionMode* = _mcTMCornerSuperimposed (Superimpose Corners)
- The motion starts from position P1 and passes near position P2. Linear interpolation is performed to position P3.
- To make the axes command velocities continuous, the deceleration range of the previous motion and the acceleration range of the current motion are combined to create the command velocity. For this reason, the acceleration time of the current motion is the same as the deceleration time of the previous motion.




The combined path passes near P2.

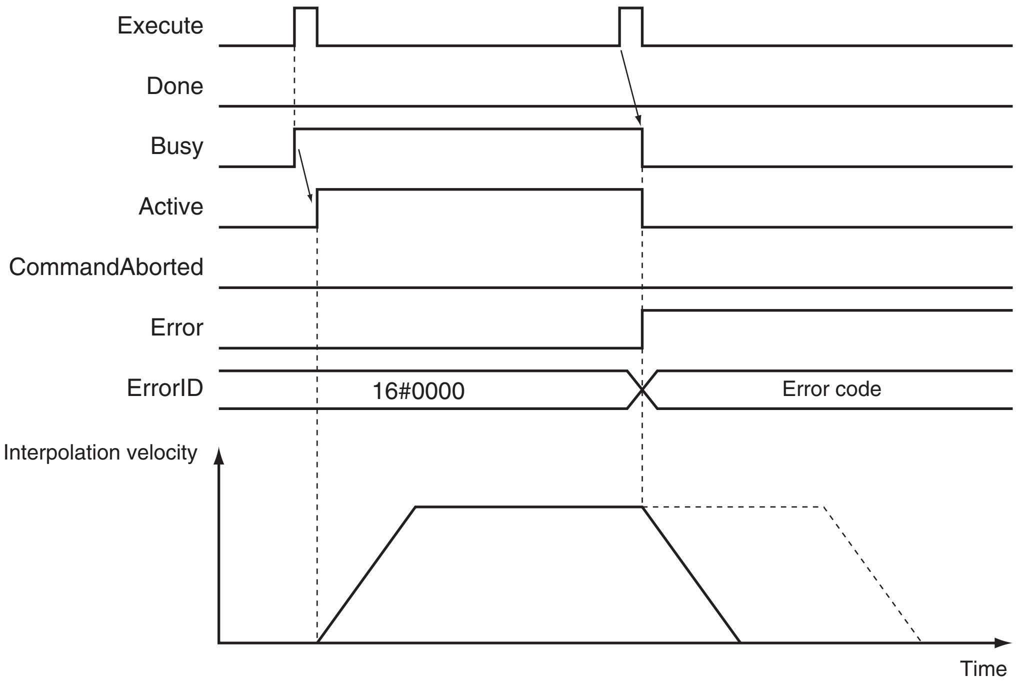
The distance from P2 to the path is longer when the interpolation velocity is faster or the deceleration rate of the previous instruction is smaller. It is shorter when the interpolation velocity is slower or the deceleration rate of the previous instruction is larger.

**Additional Information**

The *Jerk* settings are disabled in the region with superimposed corners.

## Re-execution of Motion Control Instructions

This instruction cannot be re-executed. A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted, and all axes in the linear interpolation motion stop.

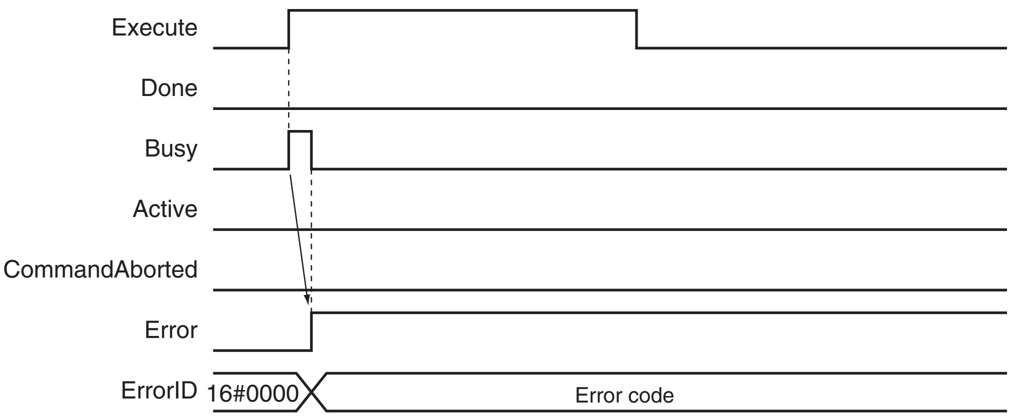



## Multi-execution of Motion Control Instructions

A restriction applies to the instructions that can be used while this instruction is in execution.
For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axes will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).




### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# Sample Programming

This section shows sample programming for linear interpolation with periodic multi-execution of instructions.

## Parameter Settings

The minimum settings required for this sample programming are given below.

### ● Setting Axis Parameters

#### Axis Types

| Axis | Axis Type |
|---|---|
| Axis 1 | Servo axis |
| Axis 2 | Servo axis |

#### Count Modes

| Axis | Count Mode |
|---|---|
| Axis 1 | Linear Mode |
| Axis 2 | Linear Mode |

#### Units of Display

| Axis | Unit of Display |
|---|---|
| Axis 1 | mm |
| Axis 2 | mm |

### ● Axes Group Parameter Settings

#### Axis Composition

Two axes are set.

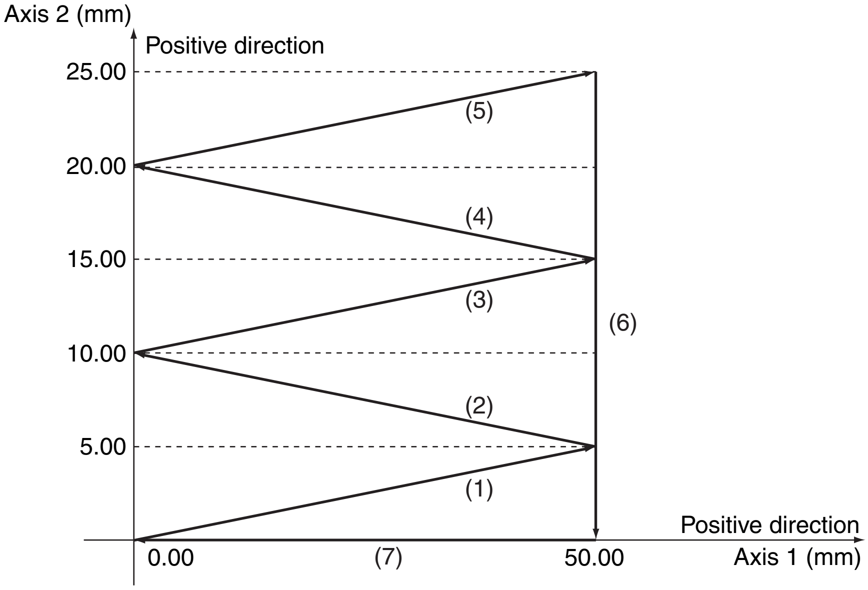
#### Axis Selection

Axis 1 and axis 2 are set.

## Operation Example

The following is an example of operation that performs linear interpolation automatically and then returns to home and stops. Linear interpolations (2) to (7) are executed with multi-execution of instructions while linear interpolation (1) is being executed. Set the Buffer Mode Selection to *Buffered*. In this sample, multi-execution of instructions is performed for (2) to (7) if the *Active* (Controlling) output variable from linear interpolation (1) is TRUE. For multi-axes coordinated operation, multi-execution is possible for up to seven instructions.

### ● Operation Pattern




Positioning is performed using linear interpolations in the order (Axis1, Axis2) = (50.00 mm, 5.00 mm) → (0.00 mm, 10.00 mm) → (50.00 mm, 15.00 mm) → (0.00 mm, 20.00 mm) → (50.00 mm, 25.00 mm) → (50.00 mm, 0.00 mm) → (0.00 mm, 0.00 mm), then stop.

## Ladder Diagram

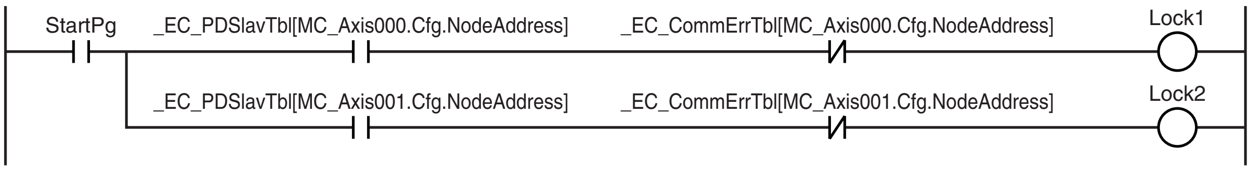
### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Group000 | _sGROUP_REF | --- | This is the Axes Group Variable for axes group 0. |
| MC_Group000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axes group 0. |
| MC_Group000.Status.Disabled | BOOL | FALSE | The value is TRUE when axes group 0 is disabled. |
| MC_Axis000 | _sAXIS_REF | --- | This is the Axis Variable for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | This is the Axis Variable for axis 2. |
| MC_Axis001.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 2. |
| MC_Axis001.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 2. |
| Pwr1_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR1 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |

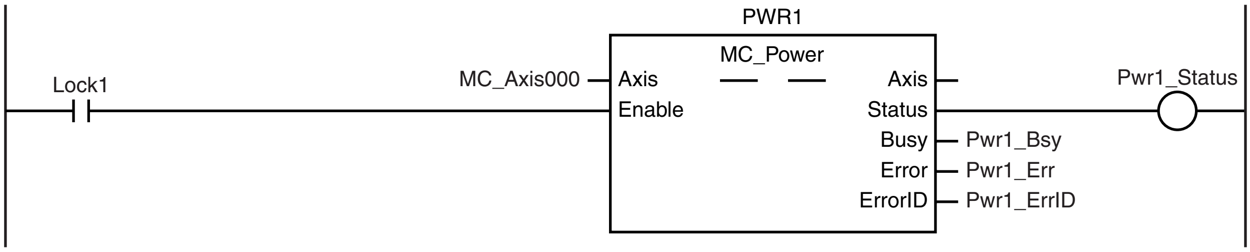
| Name | Data type | Default | Comment |
|---|---|---|---|
| Pwr2_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR2 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servos for the axes in the axes group are turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

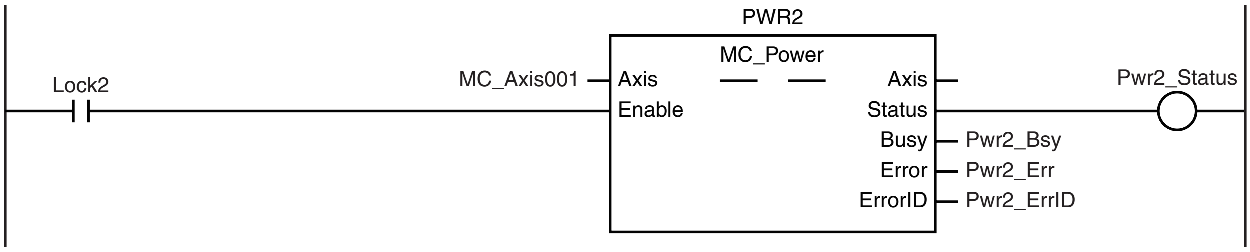
## ● Sample Programming

If *StartPg* is TRUE, EtherCAT communications for the axes are checked to see if process data communications are normal.

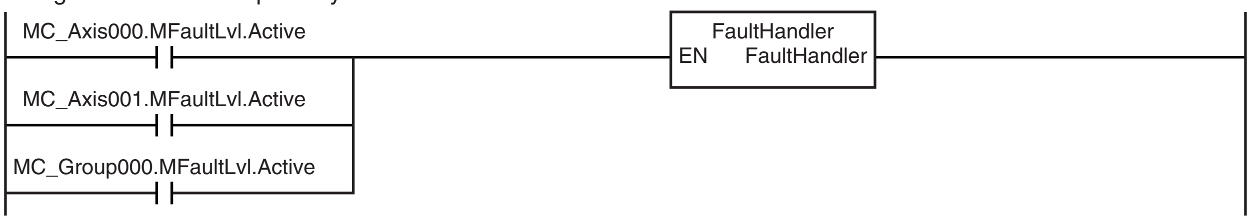



If process data communications are active, the Servo is turned ON for the axes.

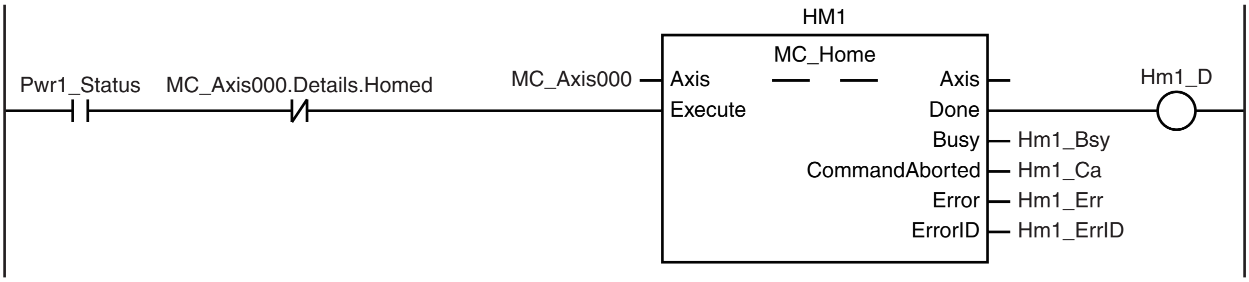


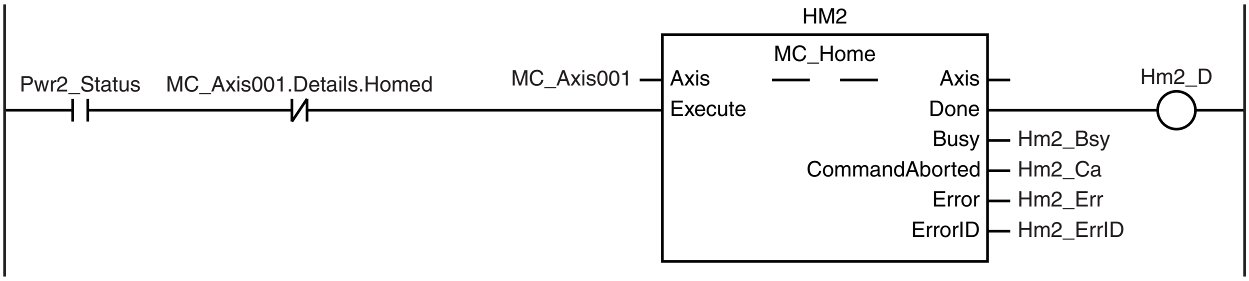

If there is a minor fault level error for a composition axis, the error handler for the device (FaultHandler) is executed. Design FaultHandler separately.



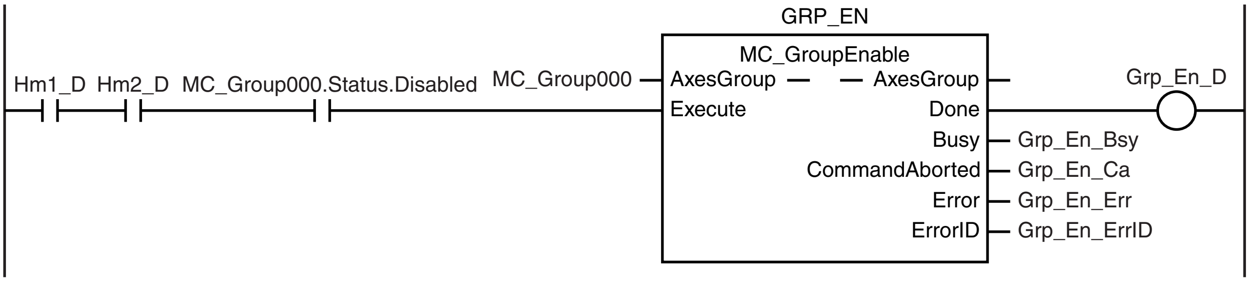
If the Servo is ON for axis 1 and home is not defined, the Home instruction is executed to define home.



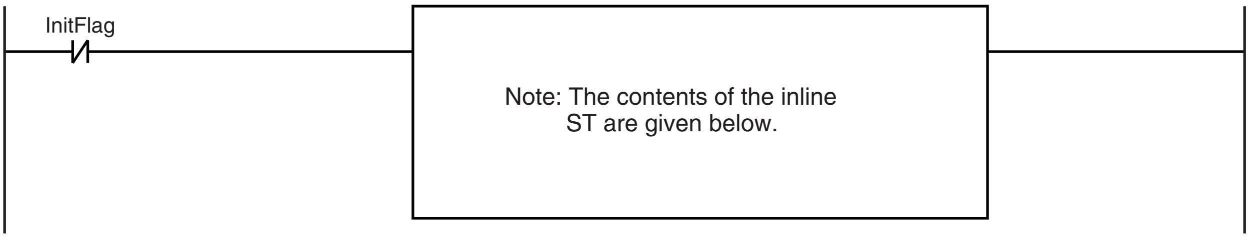

If the Servo is ON for axis 2 and home is not defined, the Home instruction is executed to define home.



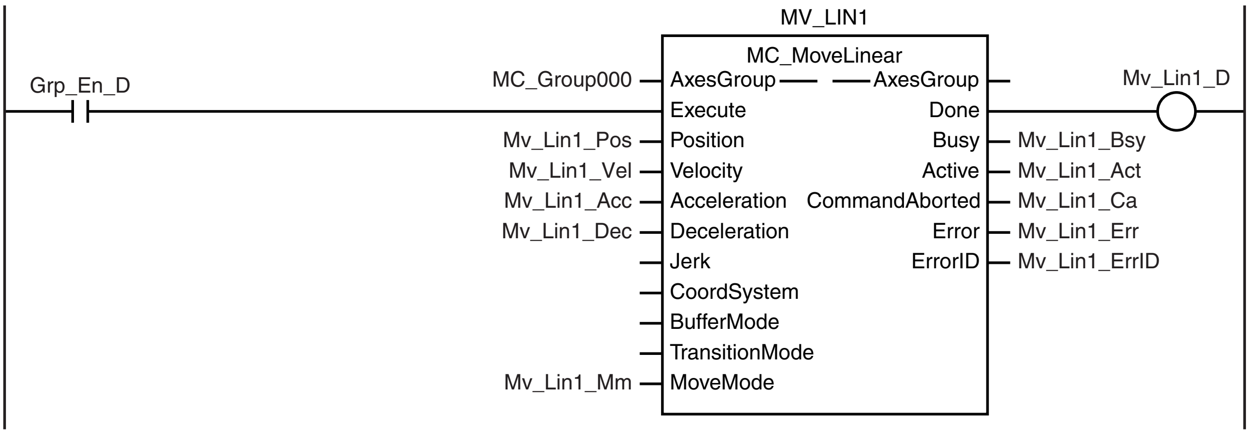

After home is defined for axis 1 and axis 2, the axes group is enabled.



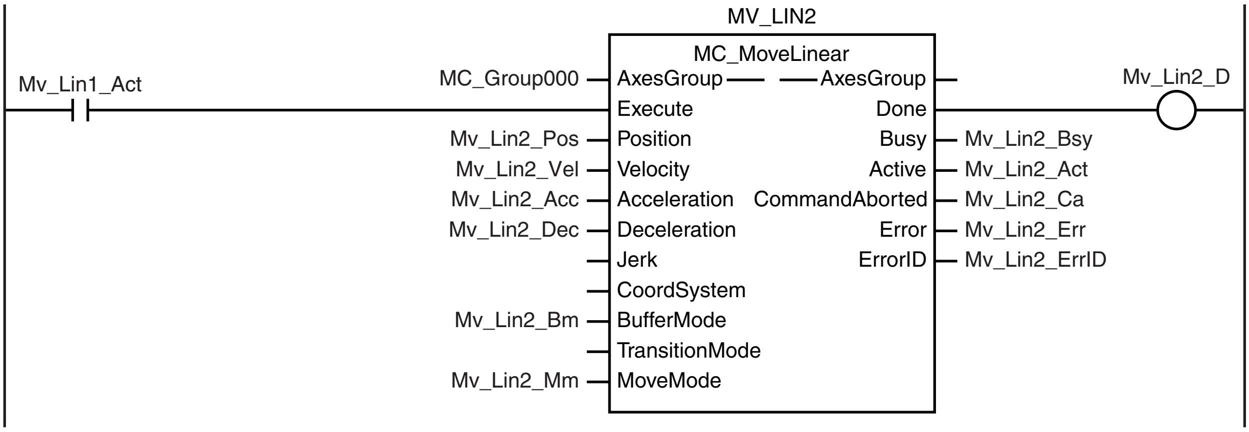

The parameters are set for linear interpolation.



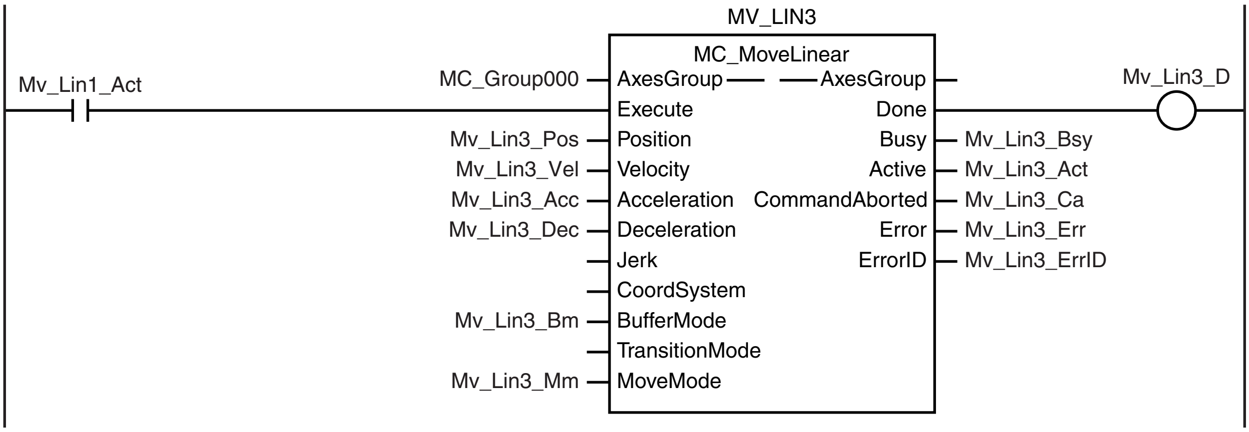

If the axes group is enabled, linear interpolation (1) is executed.



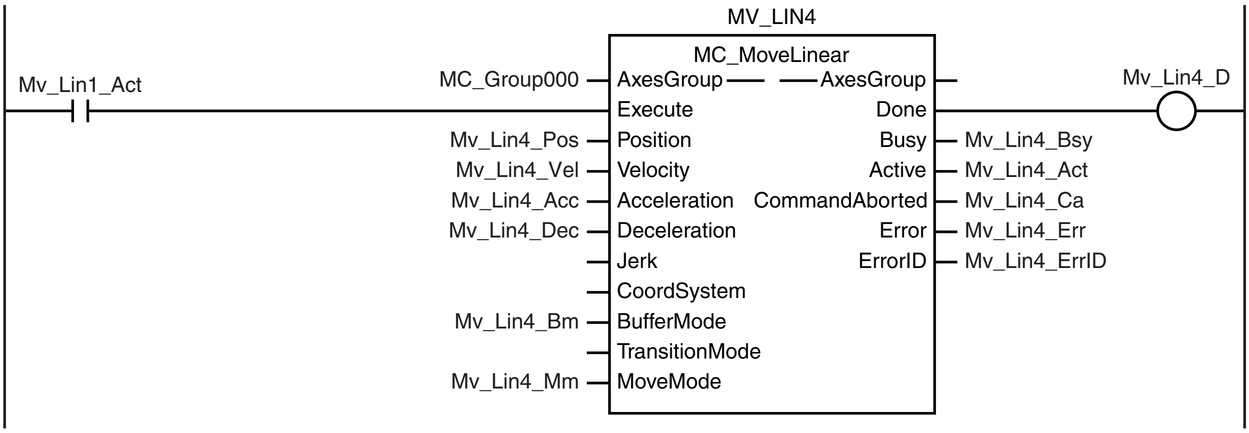
Linear interpolation (2) is executed with multi-execution of instructions after linear interpolation (1) is started.



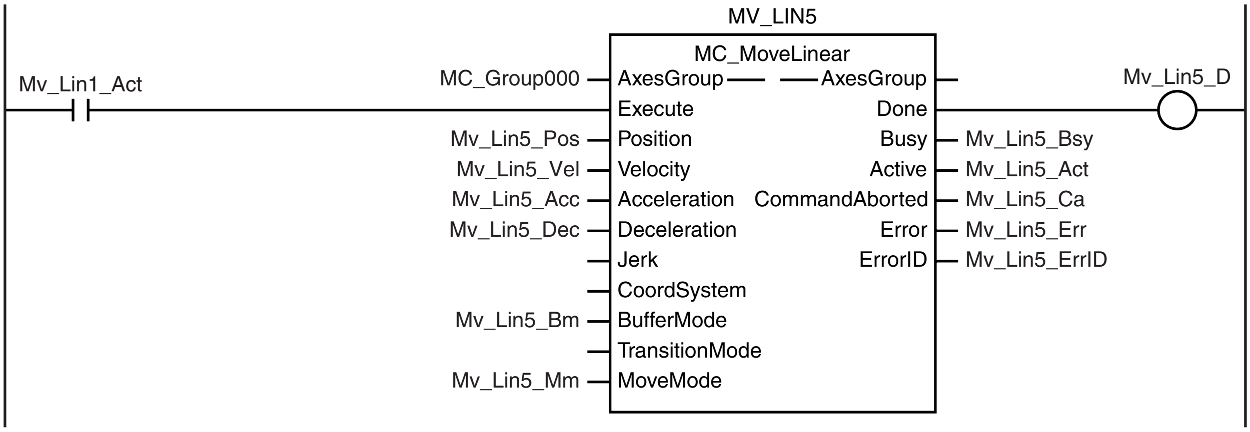

Linear interpolation (3) is executed with multi-execution of instructions after linear interpolation (1) is started.



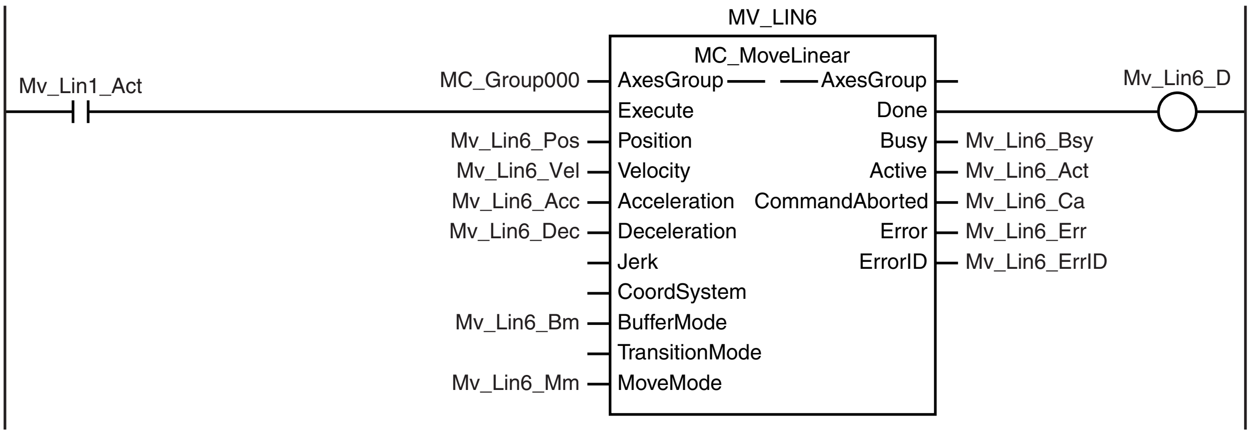

Linear interpolation (4) is executed with multi-execution of instructions after linear interpolation (1) is started.



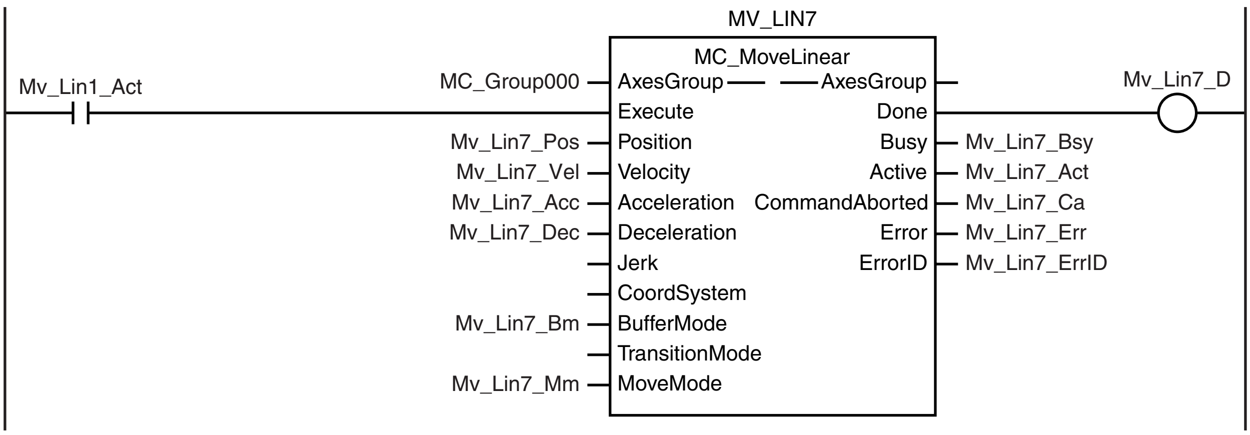
Linear interpolation (5) is executed with multi-execution of instructions after linear interpolation (1) is started.




Linear interpolation (6) is executed with multi-execution of instructions after linear interpolation (1) is started.




Linear interpolation (7) is executed with multi-execution of instructions after linear interpolation (1) is started.




## Contents of Inline ST

```
// MV_LIN1 parameters
    Mv_Lin1_Pos[0]      := LREAL#10#50.0;
    Mv_Lin1_Pos[1]      := LREAL#10#5.0;
    Mv_Lin1_Vel         := LREAL#10#100.0;
    Mv_Lin1_Acc         := LREAL#10#100.0;
    Mv_Lin1_Dec         := LREAL#10#100.0;
    Mv_Lin1_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

// MV_LIN2 parameters
    Mv_Lin2_Pos[0]      := LREAL#10#0.0;
    Mv_Lin2_Pos[1]      := LREAL#10#10.0;
    Mv_Lin2_Vel         := LREAL#10#100.0;
    Mv_Lin2_Acc         := LREAL#10#100.0;
```

```
    Mv_Lin2_Dec         := LREAL#10#100.0;
    Mv_Lin2_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Lin2_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

// MV_LIN3 parameters
    Mv_Lin3_Pos[0]      := LREAL#10#50.0;
    Mv_Lin3_Pos[1]      := LREAL#10#15.0;
    Mv_Lin3_Vel         := LREAL#10#100.0;
    Mv_Lin3_Acc         := LREAL#10#100.0;
    Mv_Lin3_Dec         := LREAL#10#100.0;
    Mv_Lin3_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Lin3_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

// MV_LIN4 parameters
    Mv_Lin4_Pos[0]      := LREAL#10#0.0;
    Mv_Lin4_Pos[1]      := LREAL#10#20.0;
    Mv_Lin4_Vel         := LREAL#10#100.0;
    Mv_Lin4_Acc         := LREAL#10#100.0;
    Mv_Lin4_Dec         := LREAL#10#100.0;
    Mv_Lin4_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Lin4_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

// MV_LIN5 parameters
    Mv_Lin5_Pos[0]      := LREAL#10#50.0;
    Mv_Lin5_Pos[1]      := LREAL#10#25.0;
    Mv_Lin5_Vel         := LREAL#10#100.0;
    Mv_Lin5_Acc         := LREAL#10#100.0;
    Mv_Lin5_Dec         := LREAL#10#100.0;
    Mv_Lin5_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Lin5_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

// MV_LIN6 parameters
    Mv_Lin6_Pos[0]      := LREAL#10#50.0;
    Mv_Lin6_Pos[1]      := LREAL#10#0.0;
    Mv_Lin6_Vel         := LREAL#10#100.0;
    Mv_Lin6_Acc         := LREAL#10#100.0;
    Mv_Lin6_Dec         := LREAL#10#100.0;
    Mv_Lin6_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Lin6_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

// MV_LIN7 parameters
    Mv_Lin7_Pos[0]      := LREAL#10#0.0;
    Mv_Lin7_Pos[1]      := LREAL#10#0.0;
    Mv_Lin7_Vel         := LREAL#10#100.0;
    Mv_Lin7_Acc         := LREAL#10#100.0;
    Mv_Lin7_Dec         := LREAL#10#100.0;
    Mv_Lin7_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Lin7_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

// InitFlag is changed to TRUE after input parameters are set.
    InitFlag := TRUE;
```

## Structured Text (ST)

### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Group000 | _sGROUP _REF | --- | This is the Axes Group Variable for axes group 0. |
| MC_Group000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axes group 0. |
| MC_Group000.Status.Disabled | BOOL | FALSE | The value is TRUE when axes group 0 is disabled. |
| MC_Axis000 | _sAXIS_REF | --- | This is the Axis Variable for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | This is the Axis Variable for axis 2. |
| MC_Axis001.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 2. |
| MC_Axis001.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 2. |
| Pwr1_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR1 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Pwr2_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR2 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servos for the axes in the axes group are turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

### ● Sample Programming

```
// Processing when input parameters are not set
IF InitFlag=FALSE THEN

    // MV_LIN1 parameters
    Mv_Lin1_Pos[0]      := LREAL#10#50.0;
    Mv_Lin1_Pos[1]      := LREAL#10#5.0;
    Mv_Lin1_Vel         := LREAL#10#100.0;
    Mv_Lin1_Acc         := LREAL#10#100.0;
    Mv_Lin1_Dec         := LREAL#10#100.0;
    Mv_Lin1_Mm          := _eMC_MOVE_MODE#_mcAbsolute;
```

```
    // MV_LIN2 parameters
    Mv_Lin2_Pos[0]      := LREAL#10#0.0;
    Mv_Lin2_Pos[1]      := LREAL#10#10.0;
    Mv_Lin2_Vel         := LREAL#10#100.0;
    Mv_Lin2_Acc         := LREAL#10#100.0;
    Mv_Lin2_Dec         := LREAL#10#100.0;
    Mv_Lin2_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Lin2_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

    // MV_LIN3 parameters
    Mv_Lin3_Pos[0]      := LREAL#10#50.0;
    Mv_Lin3_Pos[1]      := LREAL#10#15.0;
    Mv_Lin3_Vel         := LREAL#10#100.0;
    Mv_Lin3_Acc         := LREAL#10#100.0;
    Mv_Lin3_Dec         := LREAL#10#100.0;
    Mv_Lin3_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Lin3_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

    // MV_LIN4 parameters
    Mv_Lin4_Pos[0]      := LREAL#10#0.0;
    Mv_Lin4_Pos[1]      := LREAL#10#20.0;
    Mv_Lin4_Vel         := LREAL#10#100.0;
    Mv_Lin4_Acc         := LREAL#10#100.0;
    Mv_Lin4_Dec         := LREAL#10#100.0;
    Mv_Lin4_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Lin4_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

    // MV_LIN5 parameters
    Mv_Lin5_Pos[0]      := LREAL#10#50.0;
    Mv_Lin5_Pos[1]      := LREAL#10#25.0;
    Mv_Lin5_Vel         := LREAL#10#100.0;
    Mv_Lin5_Acc         := LREAL#10#100.0;
    Mv_Lin5_Dec         := LREAL#10#100.0;
    Mv_Lin5_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Lin5_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

    // MV_LIN6 parameters
    Mv_Lin6_Pos[0]      := LREAL#10#50.0;
    Mv_Lin6_Pos[1]      := LREAL#10#0.0;
    Mv_Lin6_Vel         := LREAL#10#100.0;
    Mv_Lin6_Acc         := LREAL#10#100.0;
    Mv_Lin6_Dec         := LREAL#10#100.0;
    Mv_Lin6_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Lin6_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

    // MV_LIN7 parameters
    Mv_Lin7_Pos[0]      := LREAL#10#0.0;
    Mv_Lin7_Pos[1]      := LREAL#10#0.0;
    Mv_Lin7_Vel         := LREAL#10#100.0;
    Mv_Lin7_Acc         := LREAL#10#100.0;
    Mv_Lin7_Dec         := LREAL#10#100.0;
    Mv_Lin7_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Lin7_Mm          := _eMC_MOVE_MODE#_mcAbsolute;


    // Change InitFlag to TRUE after setting the input parameters.
    InitFlag := TRUE;

END_IF;


// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 1 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis000.Cfg.NodeAddress]=TRUE)
```

```
 AND (_EC_CommErrTbl[MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
     Pwr1_En:=TRUE;          // Turn ON the Servo for axis 1.
ELSE
     Pwr1_En:=FALSE;         // Turn OFF the Servo for axis 1.
END_IF;

// If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 2 is turned ON.
// If EtherCAT communications are not normal, the Servo is turned OFF.
IF (StartPg=TRUE)
 AND (_EC_PDSlavTbl[MC_Axis001.Cfg.NodeAddress]=TRUE)
 AND (_EC_CommErrTbl[MC_Axis001.Cfg.NodeAddress]=FALSE) THEN
     Pwr2_En:=TRUE;          // Turn ON the Servo for axis 2.
ELSE
     Pwr2_En:=FALSE;         // Turn OFF the Servo for axis 2.
END_IF;

// Processing for a minor fault level error
// Implement FaultHandler separately.
IF (MC_Axis000.MFaultLvl.Active=TRUE) OR (MC_Axis001.MFaultLvl.Active=TRUE) OR
(MC_Group000.MFaultLvl.Active=TRUE)THEN
     FaultHandler();
END_IF;

// If the Servo is ON for axis 1 and home is not defined, the Home instruction is executed.
IF (Pwr1_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
     Hm1_Ex:=TRUE;
END_IF;

// If the Servo is ON for axis 2 and home is not defined, the Home instruction is executed.
IF (Pwr2_Status=TRUE) AND (MC_Axis001.Details.Homed=FALSE) THEN
     Hm2_Ex:=TRUE;
END_IF;

// If axes group 0 is disabled while home is defined for axis 1 and axis 2, it is enabled.
IF (Hm1_D=TRUE) AND (Hm2_D=TRUE) AND (MC_Group000.Status.Disabled=TRUE) THEN
          Grp_En_Ex:= TRUE;
END_IF;
// After the MC_GroupEnable (Enable Axes Group) instruction is completed, linear interpolation (1) is executed.
IF Grp_En_D=TRUE THEN
     Mv_Lin1_Ex:=TRUE;
END_IF;

// Linear interpolations (2) to (7) are executed with multi-execution of instructions while the Active output
variable for linear interpolation (1) is TRUE.
IF Mv_Lin1_Act=TRUE THEN
     Mv_Lin2_Ex:=TRUE;
     Mv_Lin3_Ex:=TRUE;
     Mv_Lin4_Ex:=TRUE;
     Mv_Lin5_Ex:=TRUE;
     Mv_Lin6_Ex:=TRUE;
     Mv_Lin7_Ex:=TRUE;
END_IF;


// MC_Power for axis 1
PWR1(
     Axis       := MC_Axis000,
     Enable     := Pwr1_En,
     Status     => Pwr1_Status,
     Busy       => Pwr1_Bsy,
     Error      => Pwr1_Err,
     ErrorID    => Pwr1_ErrID
);

// MC_Power for axis 2
```

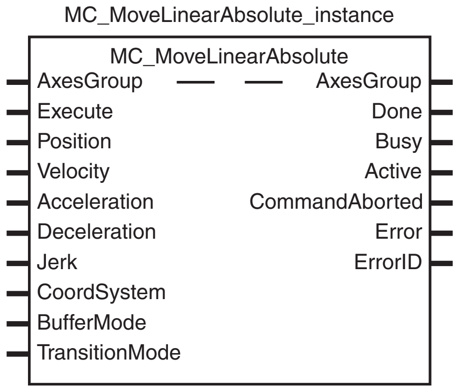
```
PWR2(
    Axis        := MC_Axis001,
    Enable      := Pwr2_En,
    Status      => Pwr2_Status,
    Busy        => Pwr2_Bsy,
    Error       => Pwr2_Err,
    ErrorID     => Pwr2_ErrID
);

// MC_Home for axis 1
HM1(
    Axis                := MC_Axis000,
    Execute             := Hm1_Ex,
    Done                => Hm1_D,
    Busy                => Hm1_Bsy,
    CommandAborted      => Hm1_Ca,
    Error               => Hm1_Err,
    ErrorID             => Hm1_ErrID
);

// MC_Home for axis 2
HM2(
    Axis                := MC_Axis001,
    Execute             := Hm2_Ex,
    Done                => Hm2_D,
    Busy                => Hm2_Bsy,
    CommandAborted      => Hm2_Ca,
    Error               => Hm2_Err,
    ErrorID             => Hm2_ErrID
);

// Axes group 0 is enabled.
GRP_EN(
    AxesGroup           := MC_Group000,
    Execute             := Grp_En_Ex,
    Done                => Grp_En_D,
    Busy                => Grp_En_Bsy,
    CommandAborted      => Grp_En_Ca,
    Error               => Grp_En_Err,
    ErrorID             => Grp_En_ErrID
);


// Linear interpolation (1)
MV_LIN1(
    AxesGroup           := MC_Group000,
    Execute             := Mv_Lin1_Ex,
    Position            := Mv_Lin1_Pos,
    Velocity            := Mv_Lin1_Vel,
    Acceleration        := Mv_Lin1_Acc,
    Deceleration        := Mv_Lin1_Dec,
    MoveMode            := Mv_Lin1_Mm,
    Done                => Mv_Lin1_D,
    Busy                => Mv_Lin1_Bsy,
    Active              => Mv_Lin1_Act,
    CommandAborted      => Mv_Lin1_Ca,
    Error               => Mv_Lin1_Err,
    ErrorID             => Mv_Lin1_ErrID
);

// Linear interpolation (2)
MV_LIN2(
    AxesGroup           := MC_Group000,
    Execute             := Mv_Lin2_Ex,
    Position            := Mv_Lin2_Pos,
```

```
    Velocity          := Mv_Lin2_Vel,
    Acceleration      := Mv_Lin2_Acc,
    Deceleration      := Mv_Lin2_Dec,
    BufferMode        := Mv_Lin2_Bm,
    MoveMode          := Mv_Lin2_Mm,
    Done              => Mv_Lin2_D,
    Busy              => Mv_Lin2_Bsy,
    Active            => Mv_Lin2_Act,
    CommandAborted    => Mv_Lin2_Ca,
    Error             => Mv_Lin2_Err,
    ErrorID           => Mv_Lin2_ErrID
);

// Linear interpolation (3)
MV_LIN3(
    AxesGroup         := MC_Group000,
    Execute           := Mv_Lin3_Ex,
    Position          := Mv_Lin3_Pos,
    Velocity          := Mv_Lin3_Vel,
    Acceleration      := Mv_Lin3_Acc,
    Deceleration      := Mv_Lin3_Dec,
    BufferMode        := Mv_Lin3_Bm,
    MoveMode          := Mv_Lin3_Mm,
    Done              => Mv_Lin3_D,
    Busy              => Mv_Lin3_Bsy,
    Active            => Mv_Lin3_Act,
    CommandAborted    => Mv_Lin3_Ca,
    Error             => Mv_Lin3_Err,
    ErrorID           => Mv_Lin3_ErrID
);

// Linear interpolation (4)
MV_LIN4(
    AxesGroup         := MC_Group000,
    Execute           := Mv_Lin4_Ex,
    Position          := Mv_Lin4_Pos,
    Velocity          := Mv_Lin4_Vel,
    Acceleration      := Mv_Lin4_Acc,
    Deceleration      := Mv_Lin4_Dec,
    BufferMode        := Mv_Lin4_Bm,
    MoveMode          := Mv_Lin4_Mm,
    Done              => Mv_Lin4_D,
    Busy              => Mv_Lin4_Bsy,
    Active            => Mv_Lin4_Act,
    CommandAborted    => Mv_Lin4_Ca,
    Error             => Mv_Lin4_Err,
    ErrorID           => Mv_Lin4_ErrID
);

// Linear interpolation (5)
MV_LIN5(
    AxesGroup         := MC_Group000,
    Execute           := Mv_Lin5_Ex,
    Position          := Mv_Lin5_Pos,
    Velocity          := Mv_Lin5_Vel,
    Acceleration      := Mv_Lin5_Acc,
    Deceleration      := Mv_Lin5_Dec,
    BufferMode        := Mv_Lin5_Bm,
    MoveMode          := Mv_Lin5_Mm,
    Done              => Mv_Lin5_D,
    Busy              => Mv_Lin5_Bsy,
    Active            => Mv_Lin5_Act,
    CommandAborted    => Mv_Lin5_Ca,
    Error             => Mv_Lin5_Err,
    ErrorID           => Mv_Lin5_ErrID
```

```
);

// Linear interpolation (6)
MV_LIN6(
    AxesGroup         := MC_Group000,
    Execute           := Mv_Lin6_Ex,
    Position          := Mv_Lin6_Pos,
    Velocity          := Mv_Lin6_Vel,
    Acceleration      := Mv_Lin6_Acc,
    Deceleration      := Mv_Lin6_Dec,
    BufferMode        := Mv_Lin6_Bm,
    MoveMode          := Mv_Lin6_Mm,
    Done              => Mv_Lin6_D,
    Busy              => Mv_Lin6_Bsy,
    Active            => Mv_Lin6_Act,
    CommandAborted    => Mv_Lin6_Ca,
    Error             => Mv_Lin6_Err,
    ErrorID           => Mv_Lin6_ErrID
);
// Linear interpolation (7)
MV_LIN7(
    AxesGroup         := MC_Group000,
    Execute           := Mv_Lin7_Ex,
    Position          := Mv_Lin7_Pos,
    Velocity          := Mv_Lin7_Vel,
    Acceleration      := Mv_Lin7_Acc,
    Deceleration      := Mv_Lin7_Dec,
    BufferMode        := Mv_Lin7_Bm,
    MoveMode          := Mv_Lin7_Mm,
    Done              => Mv_Lin7_D,
    Busy              => Mv_Lin7_Bsy,
    Active            => Mv_Lin7_Act,
    CommandAborted    => Mv_Lin7_Ca,
    Error             => Mv_Lin7_Err,
    ErrorID           => Mv_Lin7_ErrID
);
```

# MC_MoveLinearAbsolute

The MC_MoveLinearAbsolute instruction performs linear interpolation for a specified absolute position.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_MoveLinearAbsolute | Absolute Linear Interpolation | FB |  | MC_MoveLinearAbsolute_ instance (<br>AxesGroup :=*parameter,*<br>Execute :=*parameter,*<br>Position :=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>CoordSystem :=*parameter,*<br>BufferMode :=*parameter,*<br>TransitionMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Position | Target Position | ARRAY [0..3] OF LREAL | Negative number, positive number, or 0 | 0 | Specify the target position for linear interpolation. The unit is command units.*1 |
| Velocity*2 | Target Velocity | LREAL | Positive number | 0 | Specify the target velocity. The unit is command units/s.*1 |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/s$^2$.*1 |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/s$^2$.*1 |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/s$^3$.*1 |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| CoordSystem | Coordinate System | _eMC_ COORD_ SYSTEM | 0: _mcACS | 0*3 | Specify the coordinate system.<br>0: Axis coordinate system (ACS) |
| BufferMode | Buffer Mode Selection | _eMC_ BUFFER_ MODE | 0: _mcAborting<br>1: _mcBuffered<br>2: _mcBlendingLow<br>3: _mcBlendingPrevious<br>4: _mcBlendingNext<br>5: _mcBlendingHigh | 0*3 | Specify the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered<br>2: Blend with low<br>3: Blend with previous<br>4: Blend with next<br>5: Blend with high |
| TransitionMode | Transition Mode | _eMC_ TRANSITION_ MODE | 0: _mcTMNone<br>10: _mcTMCornerSuperimposed | 0*3 | Specify the path of motion.<br>0: Transition disabled<br>10: Superimpose corners |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 Always set the target velocity. If the axes are moved without setting a target velocity, an error will occur.

*3 The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axes group is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

- **Output Variable Update Timing**

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When positioning is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the axis starts moving. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_GroupStop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| AxesGroup | Axes Group | _sGROUP_REF | --- | Specify the axes group.* |

* Specify an Axes Group Variable that was created in the Axes Group Basic Settings of the Sysmac Studio. (The default axes group variable names are *MC_Group****.)

# Function

- The MC_MoveLinearAbsolute instruction performs linear interpolation for 2 to 4 axes.
- The target position is specified as an absolute position.

Other specifications are the same as those for the MC_MoveLinear (Linear Interpolation) instruction. For details, refer to *Function* on page 4-12.

**Precautions for Correct Use**

- An error occurs if home is undefined for any of the composition axes in the axes group.
- You cannot execute an instruction to perform linear interpolation if a limit input is ON for any of the logical axes that belong to the axes group.

# MC_MoveLinearRelative

The MC_MoveLinearRelative instruction performs linear interpolation for a specified relative position.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_MoveLinearRelative | Relative Linear Interpolation | FB | MC_MoveLinearRelative_instance<br>MC_MoveLinearRelative<br>AxesGroup — — AxesGroup<br>Execute Done<br>Distance Busy<br>Velocity Active<br>Acceleration CommandAborted<br>Deceleration Error<br>Jerk ErrorID<br>CoordSystem<br>BufferMode<br>TransitionMode | MC_MoveLinearRelative_ instance (<br>AxesGroup :=*parameter,*<br>Execute :=*parameter,*<br>Distance :=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>CoordSystem :=*parameter,*<br>BufferMode :=*parameter,*<br>TransitionMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Distance | Travel Distance | ARRAY [0..3] OF LREAL | Negative number, positive number, or 0 | 0 | Specify the target position for linear interpolation. The unit is command units.*1 |
| Velocity*2 | Target Velocity | LREAL | Positive number | 0 | Specify the target velocity. The unit is command units/s.*1 |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/$s^2$.*1 |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.*1 |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/$s^3$.*1 |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| CoordSystem | Coordinate System | _eMC_ COORD_ SYSTEM | 0: _mcACS | 0*3 | Specify the coordinate system.<br>0: Axis coordinate system (ACS) |
| BufferMode | Buffer Mode Selection | _eMC_ BUFFER_ MODE | 0: _mcAborting<br>1: _mcBuffered<br>2: _mcBlendingLow<br>3: _mcBlendingPrevious<br>4: _mcBlendingNext<br>5: _mcBlendingHigh | 0*3 | Specify the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered<br>2: Blend with low<br>3: Blend with previous<br>4: Blend with next<br>5: Blend with high |
| TransitionMode | Transition Mode | _eMC_ TRANSITION_ MODE | 0: _mcTMNone<br>10: _mcTMCornerSuperimposed | 0*3 | Specify the path of motion.<br>0: Transition disabled<br>10: Superimpose corners |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 Always set the target velocity. If the axes are moved without setting a target velocity, an error will occur.

*3 The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axes group is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When positioning is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the axis starts moving. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_GroupStop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| AxesGroup | Axes Group | _sGROUP_REF | --- | Specify the axes group.* |

* Specify an Axes Group Variable that was created in the Axes Group Basic Settings of the Sysmac Studio. (The default axes group variable names are *MC_Group****.)

## Function

- The MC_MoveLinearRelative instruction performs linear interpolation for 2 to 4 axes.
- The target position is specified as a relative position.

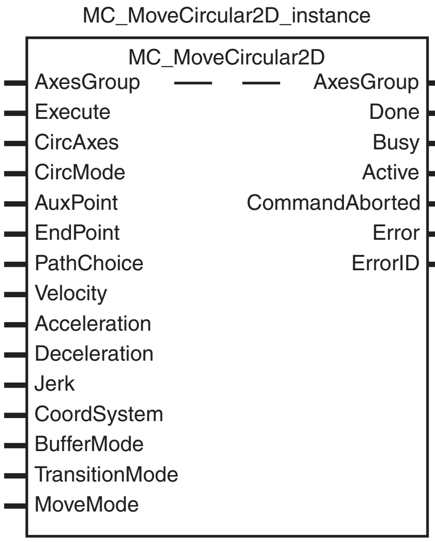
Other specifications are the same as those for the MC_MoveLinear (Linear Interpolation) instruction. For details, refer to *Function* on page 4-12.

**Precautions for Correct Use**

- An error occurs if home is undefined for any of the composition axes in the axes group.
- You cannot execute an instruction to perform linear interpolation if a limit input is ON for any of the logical axes that belong to the axes group.

# MC_MoveCircular2D

The MC_MoveCircular2D instruction performs circular interpolation for two axes.

| Instruction | Name | FB/<br>FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_MoveCircular2D | Circular 2D Interpolation | FB |  | MC_MoveCircular2D_instance (<br>AxesGroup :=*parameter,*<br>Execute :=*parameter,*<br>CircAxes :=*parameter,*<br>CircMode :=*parameter,*<br>AuxPoint :=*parameter,*<br>EndPoint :=*parameter,*<br>PathChoice :=*parameter,*<br>Velocity :=*parameter,*<br>Acceleration :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>CoordSystem :=*parameter,*<br>BufferMode :=*parameter,*<br>TransitionMode :=*parameter,*<br>MoveMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted<br>=>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| CircAxes | Circular Axes | ARRAY [0,1] OF UINT | 0 to 3 | 0 | Specify the axes for circular interpolation.<br>0: Axis A0<br>1: Axis A1<br>2: Axis A2<br>3: Axis A3 |
| CircMode | Circular Interpolation Mode | _eMC_CIRC_MODE | 0: _mcBorder<br>1: _mcCenter<br>2: _mcRadius | 0*1 | Specify the method for circular interpolation.<br>0: Border point<br>1: Center<br>2: Radius |
| AuxPoint | Auxiliary Point | ARRAY [0,1] OF LREAL | Negative number, positive number, or 0 | 0 | Specify the border point, center, or radius. The unit is command units.*2 |

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| EndPoint | End Point | ARRAY [0,1] OF LREAL | Negative number, positive number, or 0 | 0 | Specify the target position. The unit is command units.*2 |
| PathChoice | Path Choice | _eMC_CIRC_PATHCHOICE | 0: _mcCW<br>1: _mcCCW | 0*1 | Specify the path direction.<br>0: CW<br>1: CCW |
| Velocity*3 | Target Velocity | LREAL | Positive number | 0 | Specify the target velocity. The unit is command units/s.*2 |
| Acceleration | Acceleration Rate | LREAL | Non-negative number | 0 | Specify the acceleration rate. The unit is command units/$s^2$.*2 |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.*2 |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/$s^3$.*2 |
| CoordSystem | Coordinate System | _eMC_COORD_SYSTEM | 0: _mcACS | 0*1 | Specify the coordinate system.<br>0: Axis coordinate system (ACS) |
| BufferMode | Buffer Mode Selection | _eMC_BUFFER_MODE | 0: _mcAborting<br>1: _mcBuffered<br>2: _mcBlendingLow<br>3: _mcBlendingPrevious<br>4: _mcBlendingNext<br>5: _mcBlendingHigh | 0*1 | Specify the operation when executing more than one motion instruction.<br>0: Aborting<br>1: Buffered<br>2: Blend with low<br>3: Blend with previous<br>4: Blend with next<br>5: Blend with high |
| TransitionMode | Transition Mode | _eMC_TRANSITION_MODE | 0: _mcTMNone<br>10: _mcTMCornerSuperimpose | 0*1 | Specify the path of motion.<br>0: Transition disabled<br>10: Superimpose corners |
| MoveMode | Travel Mode | _eMC_MOVE_MODE | 0: _mcAbsolute<br>1: _mcRelative | 0*1 | 0: Absolute positioning<br>1: Relative positioning |

*1 The default value for an enumeration variable is actually not the number, but the enumerator.

*2 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*3 Always set the target velocity. If the axes are moved without setting a target velocity, an error will occur.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axes group is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When positioning is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the axis starts moving. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error.<br>• When you start this instruction during MC_GroupStop instruction execution. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| AxesGroup | Axes Group | _sGROUP_REF | --- | Specify the axes group.* |

* Specify an Axes Group Variable that was created in the Axes Group Basic Settings of the Sysmac Studio. (The default axes group variable names are *MC_Group****.)

## Function

- The MC_MoveCircular2D instruction performs 2D circular interpolation for two axes.

**Precautions for Correct Use**

- An error occurs if home is undefined for any of the composition axes in the axes group.
- You cannot execute an instruction to perform circular 2D interpolation if a limit input is ON for any of the logical axes that belong to the axes group.

## Instruction Details

This section describes the instruction in detail.

### Circular Interpolation Procedure

Use the following procedure to perform circular interpolation.

**1** Registering Axes Groups for Interpolation

- Determine the axes group to perform interpolation.
  An axes group is represented by “_MC_GRP***”.
- Specify the axis composition with the Composition Axes Group Variable.
- Specify the combination of axes to perform interpolation with the Axis Selection Axes Group Variable.
- Use logical axes (axis A0 to A3) for the axes, and not axis numbers (axis 0 to 63).
- Specify axis numbers 0 to 63 for the logical axes A0 to A3 in order from the lowest number using the Axis Selection Axes Group Variable.

| Logical axis | Axis number |
|---|---|
| Axis A0 | Axis 0 to Axis 63 |
| Axis A1 | Axis 0 to Axis 63 |
| Axis A2 | Axis 0 to Axis 63 |
| Axis A3 | Axis 0 to Axis 63 |

**Example:** The following specifications are used to specify axis numbers 0 and 1 for axes A0 and A1 with a 2-axis composition.

| Logical axis | Axis number | Description |
|---|---|---|
| Axis A0 | Axis 0 | Specify axis numbers to the logical axes from axis A0 in order from the lowest number. |
| Axis A1 | Axis 1 | |

**Precautions for Correct Use**

An axis type error will occur and operation will end if an encoder axis or virtual encoder axis is included in the axes group. Make sure that you select only servo axes or virtual servo axes.

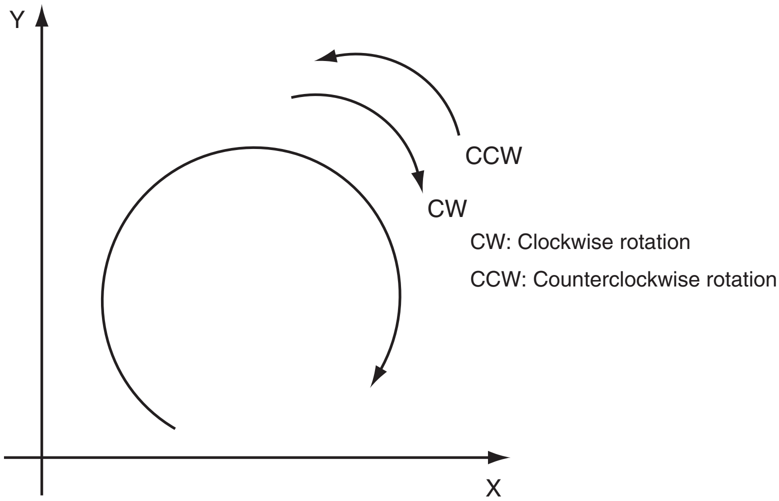
**2** Enabling the Axes Group

- Execute MC_GroupEnable (Enable Axes Group) instruction to enable the registered axes group.
- Turn ON the Servo for the composition axes and change *Execute* of the MC_Stop instruction to FALSE.

Using the Circular Interpolation instruction is now enabled.

- ***CircAxes* (Circular Axes)**

Circular interpolation uses the X axis and Y axis.




Specify the axes to use as the X axis and Y axis with *CircAxes* (Circular Axes). Use logical axes (axis A0 to A3) for the axes, and not axis numbers (axis 0 to 63).

**Precautions for Correct Use**

Specify the Count Mode to Linear Mode for the axes that you use for the X axis and Y axis. If you specify Rotary Mode, a Circular Interpolation Execution Error Due to Count Mode (error code: 544A hex) will occur at execution.

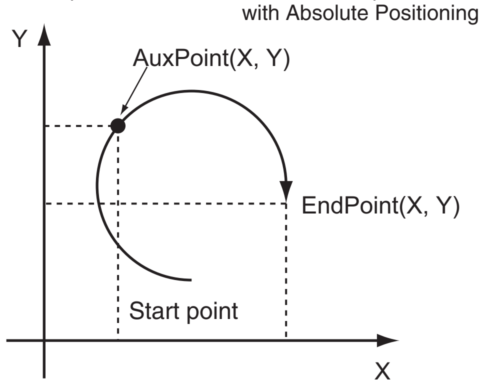
- ***CircMode* (Circular Interpolation Mode)**

There are three methods of circular interpolation: border point, center, and radius. You can specify one of these methods with *CircMode* (Circular Interpolation Mode). Absolute positioning or relative positioning can be used to specify the position with these methods. You can specify absolute or relative positioning with *MoveMode* (Travel Mode).

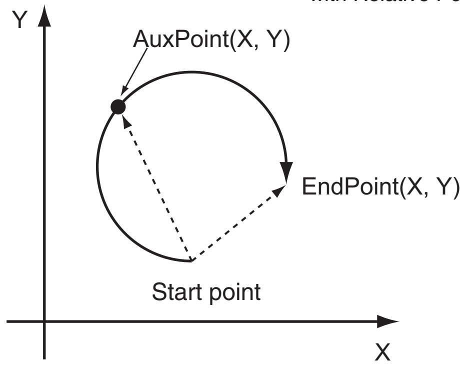
| MoveMode | Description |
|---|---|
| Absolute positioning | The border point for a border point specification or the center point and end point for a center point specification are specified as absolute positions from home in the axis coordinate system. |
| Relative positioning | The border point for a border point specification or the center point and end point for a center point specification are specified as relative positions from the start point. |

The difference between absolute positioning and relative positioning using a border point is described below as an example.

Circular Interpolation Method: Border Point Specification with Absolute Positioning



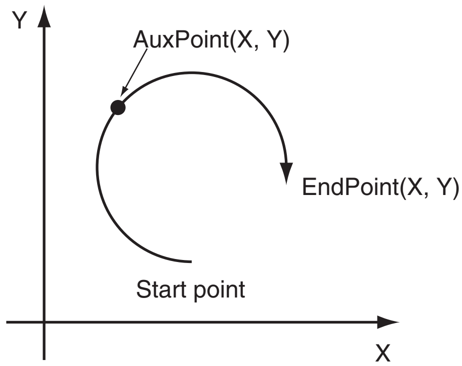

Circular Interpolation Method: Border Point Specification with Relative Positioning



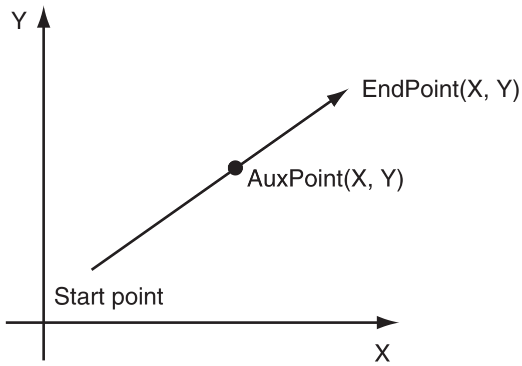

The following sections describe the operation assuming that absolute positioning has been specified as the *MoveMode* (Travel Mode).

## Border point

The current position is the starting point. Circular interpolation is performed through the border point *AuxPoint(X,Y)* to the end point *EndPoint(X,Y)*.



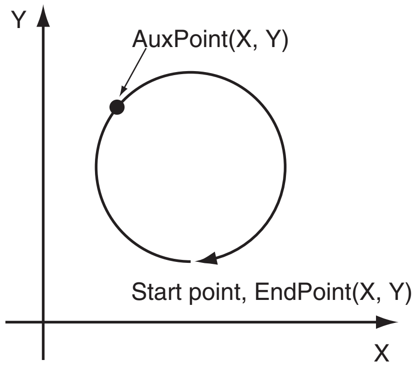

If the start point, border point, and end point are along the same line, if the border point and the end point are at the same point, or if the start point and the border point are the same point, linear interpolation is performed from the start point to the end point.




**Precautions for Correct Use**

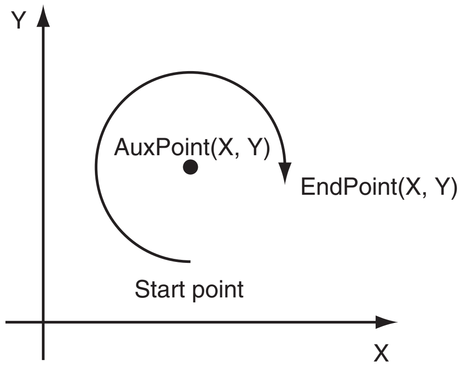
- The points are considered to be on a straight line if the distance between the border point and the line that connects the start and end point is less than one pulse for both the X and Y coordinates.
- An error occurs if the start point, border point, and end point are the same point.
  The start point, border point, and end point are considered to be the same point if the command positions are the same for the command unit. If the command positions in the command unit are different, the points are not considered to be the same point and an error does not occur even if the positions are the same when they are converted to pulses.

If the start point and the end point are the same point, a complete circle is drawn with the start point and the border point as the diameter. *PathChoice* is specified as the circular interpolation direction.




### Center

The current position is the starting point. Circular interpolation is performed for circle specified by the center point *AuxPoint(X,Y)* to the end point *EndPoint(X,Y)*. *PathChoice* is used to specify the circular interpolation direction. A complete circle is drawn when the start point and end point are at the same point. If the radius from the specified center to the start point is different to the radius to the end point, the average of the two radiuses is used to perform circular interpolation. In this case, the center is calculated in the same way as specifying the radius, and the calculated radius and center are used.



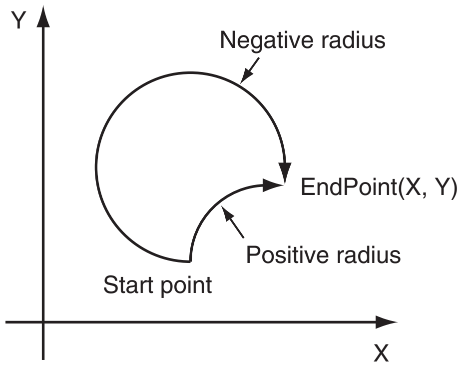

**Precautions for Correct Use**

If the Axes Group Parameter check method for circular interpolation center point specification is set to any value other than 0, a center point specification error will occur when the specified center point exceeds the circle calculated as follows:

The radius of the circle in which the center point must be positioned is the calculated radius multiplied by the percentage that is set for the center point specification check method divided by 100. (The radius calculated from the corrected center point it taken as 100%.)

### Radius

The current position is the starting point. Circular interpolation is performed for the circle specified by the radius *AuxPoint(X,Y)* to the end point *EndPoint(X,Y)*. The radius is specified by the first element in *AuxPoint(X,Y)*. The second element is not used. For example, for a radius of 100, set *AuxPoint(X,Y)* to *AuxPoint(100,0)*. If the sign of the radius is negative, a circle with a long arc will be drawn. If the sign is positive, a circle with a short arc will be drawn. *PathChoice* is used to specify the circular interpolation direction.




**Precautions for Correct Use**

- If the start point and the end point are the same, a same circular interpolation start and end point error will occur and operation will stop for all axes in the group.
- If the specified radius is less than half the length of the distance between the start point and end point, a circle is impossible and an error will occur.

- ***Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), *Deceleration* (Deceleration Rate), and *Jerk***

Set *Velocity, Acceleration, Deceleration,* and *Jerk* to specify the interpolation velocity, acceleration rate, deceleration rate, and jerk for circular interpolation. If you set the interpolation velocity for circular interpolation to 0, a velocity specification error will occur and operation will stop for all axes in the group. If the specified interpolation velocity exceeds the maximum velocity of an axis, the following operation is executed.

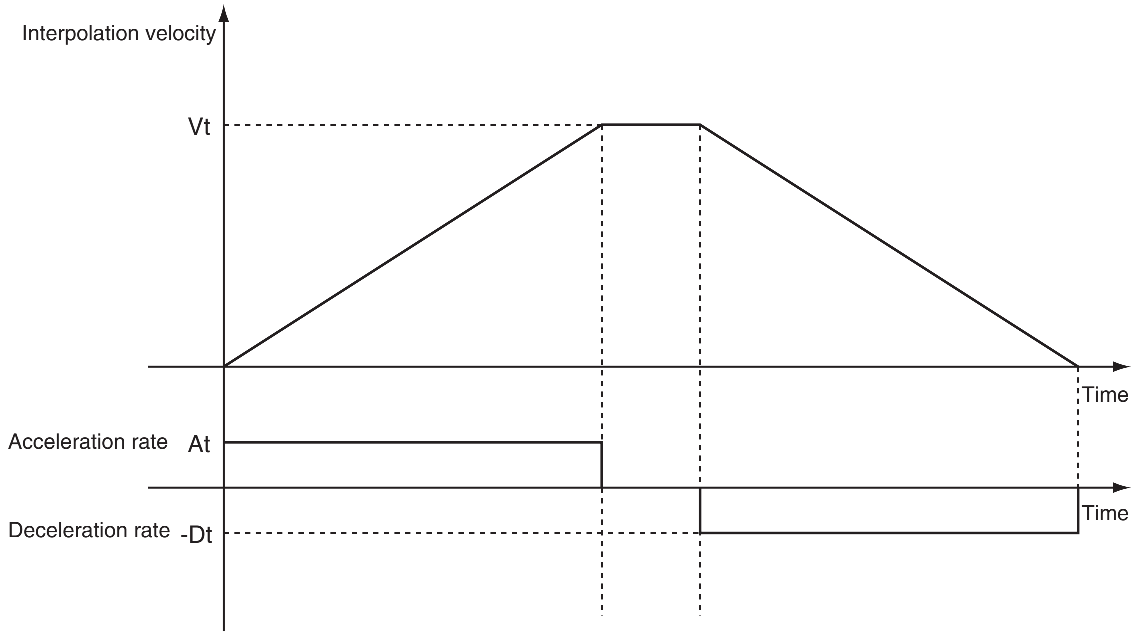
If only one axis exceeds the maximum velocity:
This axis moves at the maximum velocity and the interpolation velocity is adjusted accordingly.

If both axes exceed the maximum velocity:
The interpolation velocity is automatically adjusted so that the axes move at the maximum velocity of the two axes that is slower.

### *Jerk*

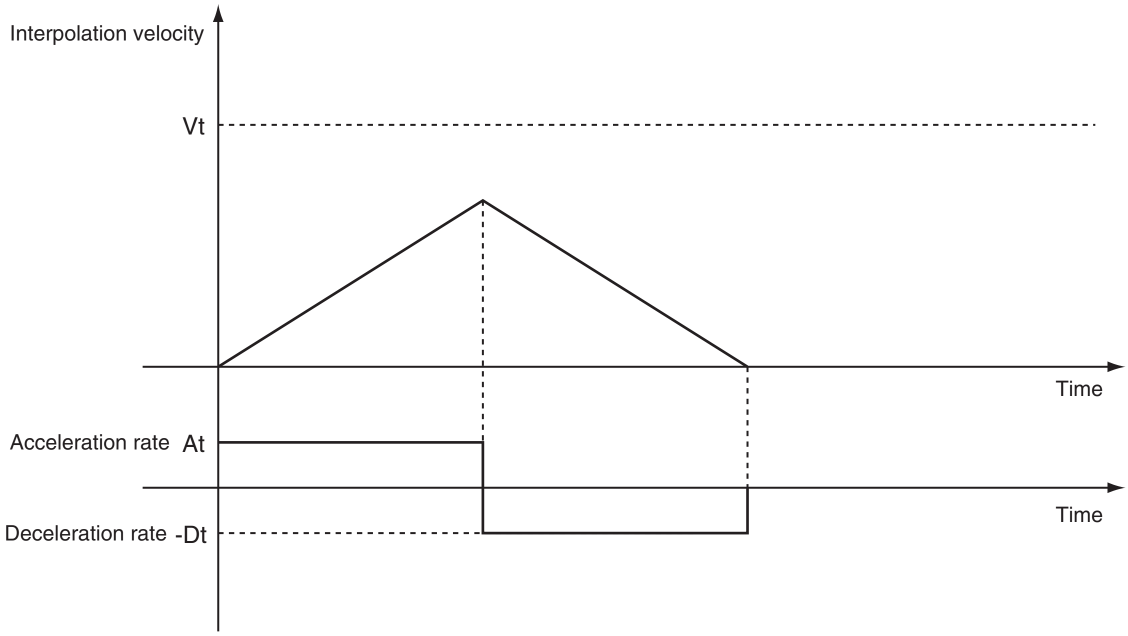
The relationships between *Acceleration* (Acceleration Rate), *Deceleration* (Deceleration Rate), and *Velocity* (Target Velocity) when *Jerk* is set to 0 and when it is set to any other value are shown below.

- *Jerk* Set to 0
  The command value for the velocity is created with acceleration rate At and deceleration rate Dt.



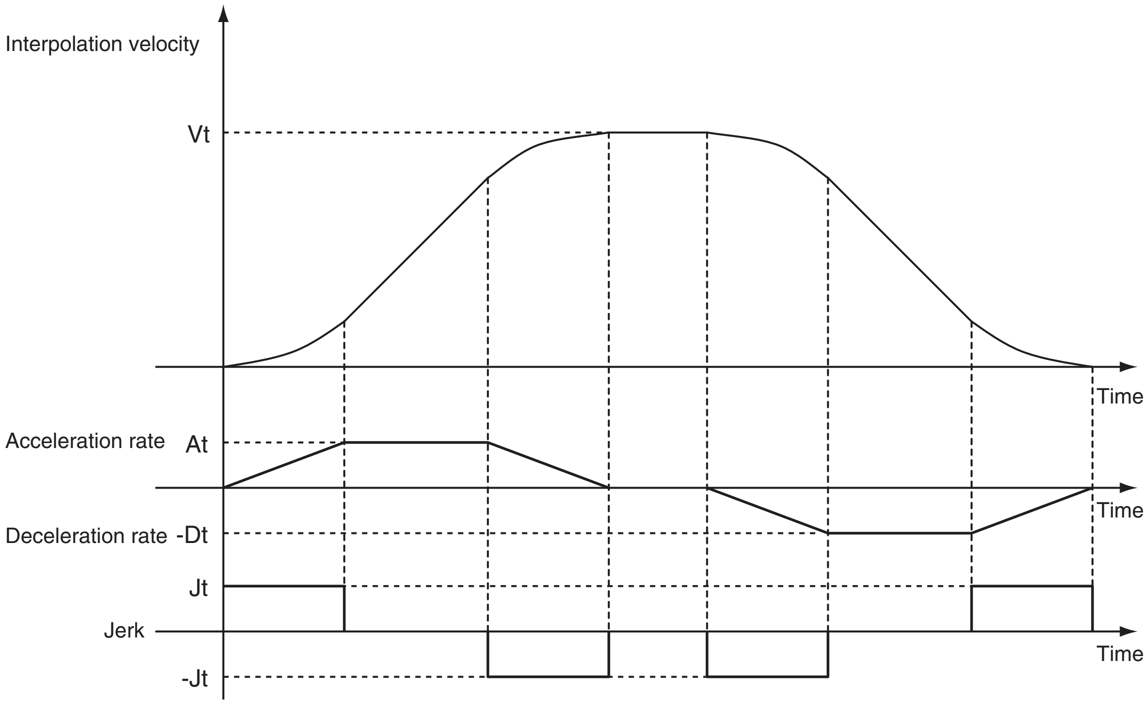

Vt: Specified interpolation velocity, At: Specified acceleration rate, Dt: Specified deceleration rate.

- Short Travel Distance When *Jerk* Is 0
  The interpolation velocity will not reach the specified Vt (Target Velocity).



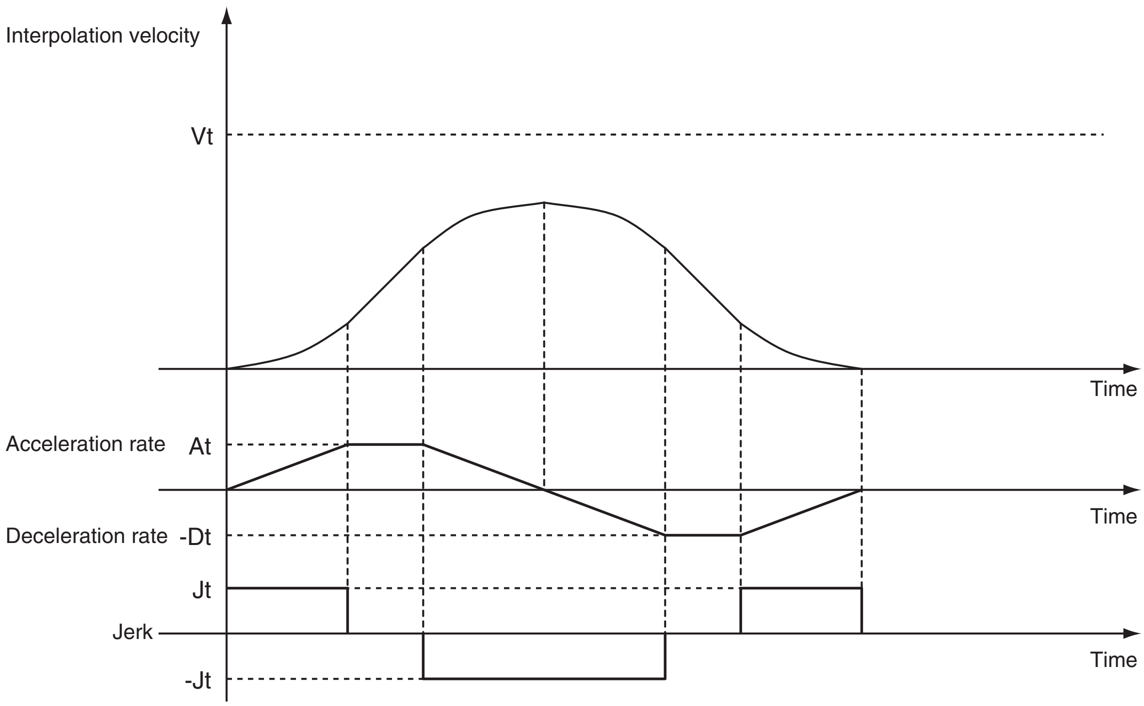

Vt: Specified interpolation velocity, At: Specified acceleration rate, Dt: Specified deceleration rate.

- *Jerk* Set to Value Other Than 0
  The command value for the velocity is created with At as the upper acceleration limit and Dt as the upper deceleration limit.




Vt: Specified interpolation velocity, At: Specified acceleration rate, Dt: Specified deceleration rate, Jt: Specified jerk

- Short Travel Distance When *Jerk* Is Other Than 0
  The interpolation velocity will not reach the specified Vt (Target Velocity).




Vt: Specified interpolation velocity, At: Specified acceleration rate, Dt: Specified deceleration rate, Jt: Specified jerk

**Precautions for Correct Use**

- If 0 is specified for *Acceleration* (Acceleration Rate), the specified interpolation velocity is used immediately.
- If 0 is specified for *Deceleration* (Deceleration Rate), the axis stops immediately. However, if the Buffer Mode is set to *Blending*, axis operation will change to the interpolation velocity specified by the next operation without stopping. For details, refer to *BufferMode (Buffer Mode Selection)* on page 4-52.
- When the *Acceleration* (Acceleration Rate) or *Deceleration* (Deceleration Rate) is 0, the setting of *Jerk* is disabled.

### ● *CoordSystem* (Coordinate System)

- *CoordSystem* specifies the coordinate system to use for circular interpolation.
- Only an axis coordinate system (ACS) consisting of two or more axes is supported.

### ● *BufferMode* (Buffer Mode Selection)

- *BufferMode* specifies how to join the axis motions for this interpolation instruction and the previous interpolation instruction.
- There are the following six settings.

| Buffer Mode Selection | | Description |
|---|---|---|
| Aborting | | Aborts the instruction being executed and switches to this instruction.<br>If the direction of axis motion is reversed by switching instructions, the motion will be reversed according to the Operation at Reversing axis parameter. |
| Buffered | | Buffers this instruction and executes it automatically after the current instruction is completed. |
| Blending | | Starts the buffered instruction at the velocity (transit velocity) at which the current instruction reaches the target position. The operation of the current instruction is changed so that the axes reach the target position at the transit velocity. There are four methods to specify the transit velocity. These are described below. You can also specify a Transition Mode as an option to the Blending Mode (see below). |
| | Blend with low | The lower of the target velocities of the current instruction and the buffered instruction is used as the transit velocity. |
| | Blend with previous | The target velocity of the current instruction is used as the transit velocity. |
| | Blend with next | The target velocity of the buffered instruction is used as the transit velocity. |
| | Blend with high | The higher of the target velocities of the current instruction and the buffered instruction is used as the transit velocity. |

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● *TransitionMode*

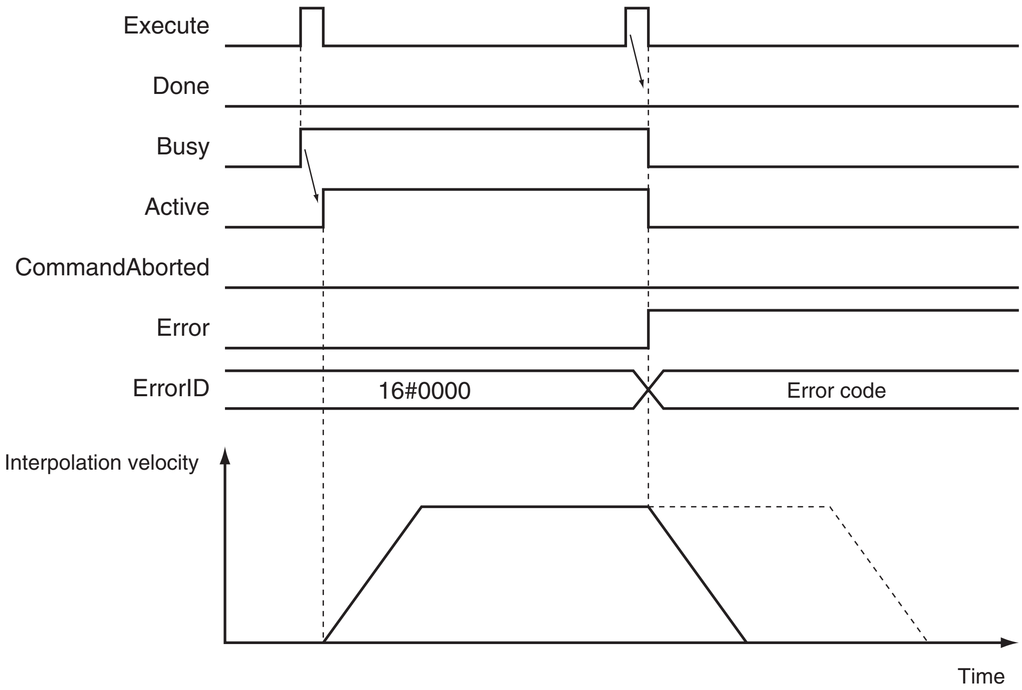
- *TransitionMode* specifies how to join the motions for this interpolation instruction and the previous interpolation instruction.
- If *BufferMode* (Buffer Mode Selection) is set to *Blending*, *TransitionMode* is enabled.
- An error will occur if you do not set *TransitionMode* to *_mcTMNone* (Transition Disabled) when blending is not used.

For details, refer to *TransitionMode* on page 4-19.

## Re-execution of Motion Control Instructions

This instruction cannot be re-executed.

A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted, and all axes in the circular interpolation motion stop.

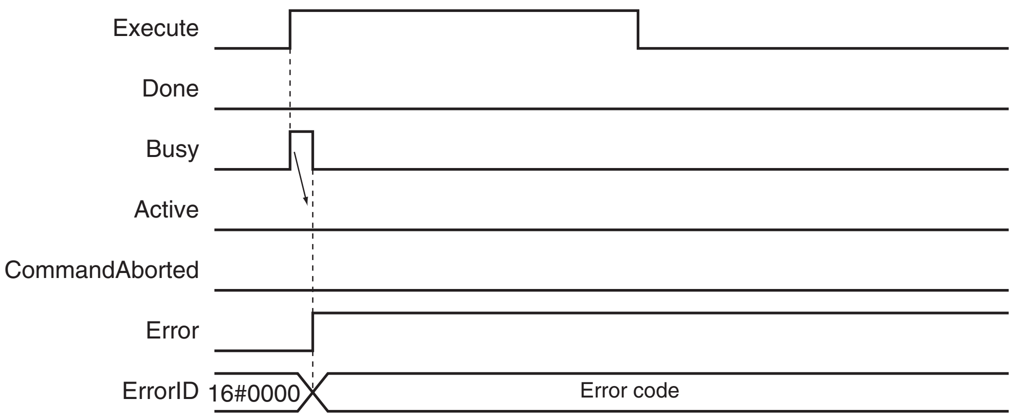



## Multi-execution of Motion Control Instructions

A restriction applies to the instructions that can be used while this instruction is in execution. For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axes will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).




### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# Sample Programming

This section shows sample programming for circular interpolation with multi-execution of instructions.

## Parameter Settings

The minimum settings required for this sample programming are given below.

- **Setting Axis Parameters**

### Axis Types

| Axis | Axis Type |
|---|---|
| Axis 1 | Servo axis |
| Axis 2 | Servo axis |

### Count Modes

| Axis | Count Mode |
|---|---|
| Axis 1 | Linear Mode |
| Axis 2 | Linear Mode |

### Units of Display

| Axis | Unit of Display |
|---|---|
| Axis 1 | mm |
| Axis 2 | mm |

- **Axes Group Parameter Settings**

### Axis Composition

Two axes are set.

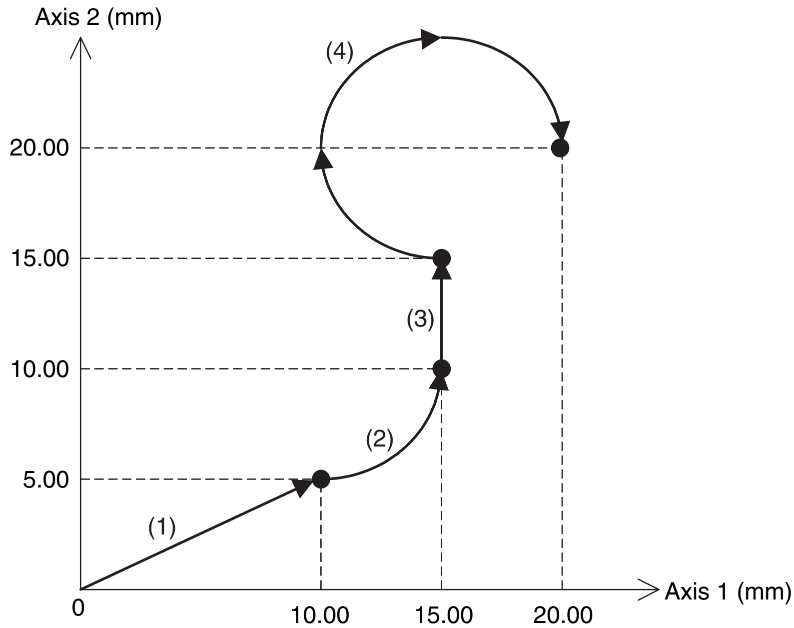
### Axis Selection

Axis 1 and axis 2 are set.

## Operation Example

The following is an example of operation where the axes automatically perform positioning by using linear interpolation and circular interpolation. The axes move to the final target position (20.00 mm, 20.00 mm) using linear interpolation and circular interpolation. The Buffer Mode is set to *Buffered* and multi-execution of instructions is used. In this sample, multi-execution of instructions is performed for (2) to (4) if the *Active* output variable from linear interpolation (1) is TRUE. For multi-axes coordinated operation, multi-execution is possible for up to seven instructions.

- **Operation Pattern**




**1** Execution

When you turn ON the operation start switch at home, the axes move to the point (10.00 mm, 5.00 mm) via linear interpolation.

**2** Continuous Motion

The axes continue to move to the point (15.00 mm, 10.00 mm) via circular interpolation, to the point (15.00 mm, 15.00 mm) via linear interpolation, and to the point (20.00 mm, 20.00 mm) via circular interpolation. Here, the velocity is 10.00 mm/s.

## Ladder Diagram

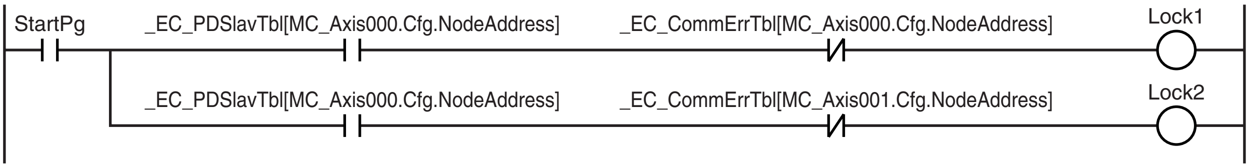
- **Main Variables**

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Group000 | _sGROUP_REF | --- | This is the Axes Group Variable for axes group 0. |
| MC_Group000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axes group 0. |
| MC_Group000.Status.Disabled | BOOL | FALSE | The value is TRUE when axes group 0 is disabled. |
| MC_Axis000 | _sAXIS_REF | --- | This is the Axis Variable for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | This is the Axis Variable for axis 2. |
| MC_Axis001.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 2. |
| MC_Axis001.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 2. |
| Pwr1_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR1 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |

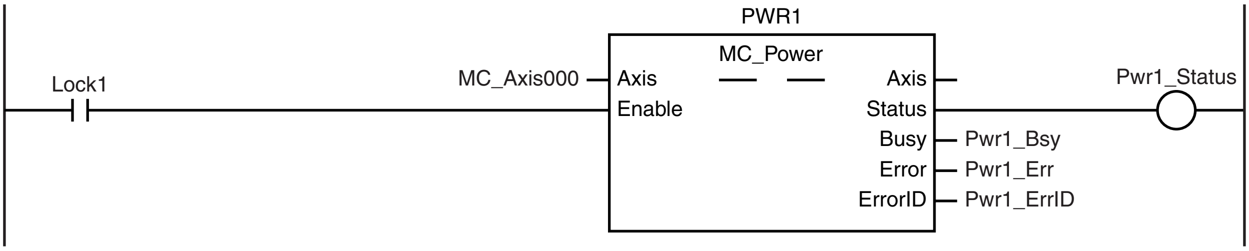
| Name | Data type | Default | Comment |
|---|---|---|---|
| Pwr2_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR2 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servos for the axes in the axes group are turned ON if this variable is TRUE and EtherCAT process data communications are established. |
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |

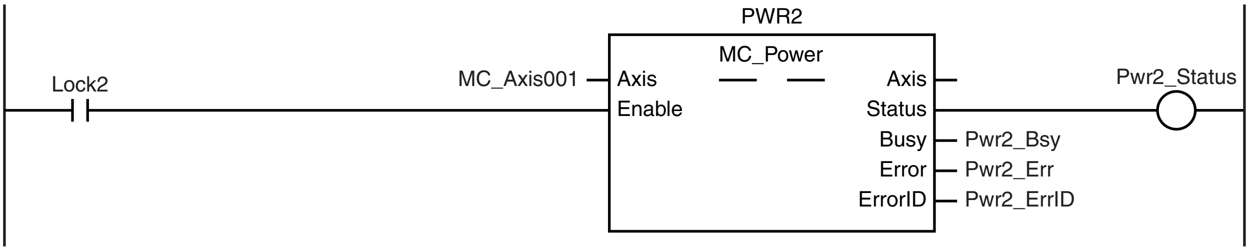
## ● Sample Programming

If *StartPg* is TRUE, EtherCAT communications for the axes are checked to see if process data communications are normal.

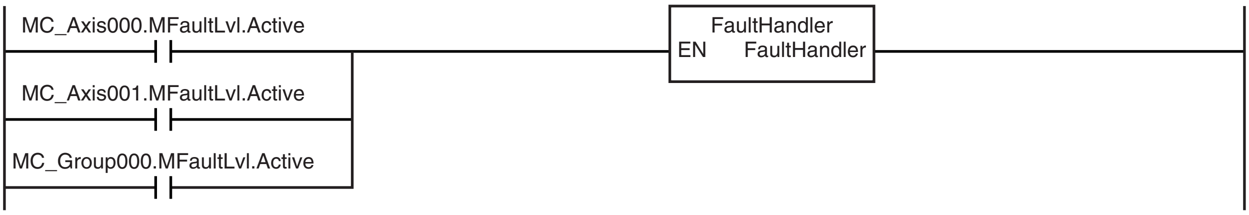



If process data communications are active, the Servo is turned ON.

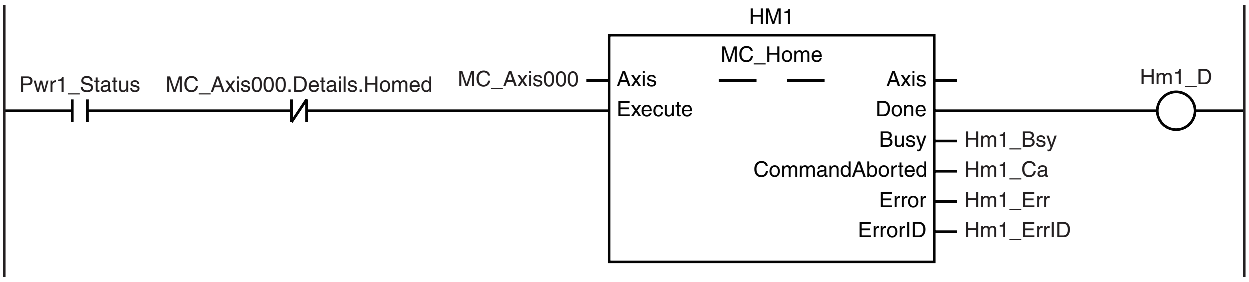


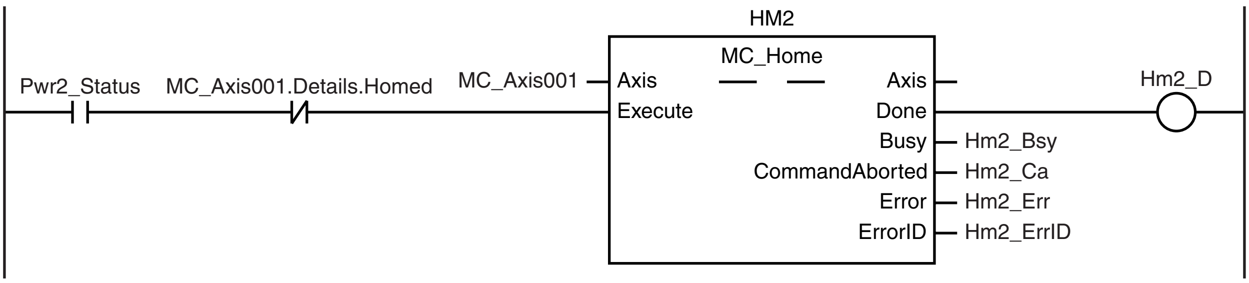

If there is a minor fault level error for a composition axis, the error handler for the device (FaultHandler) is executed. Design FaultHandler separately.



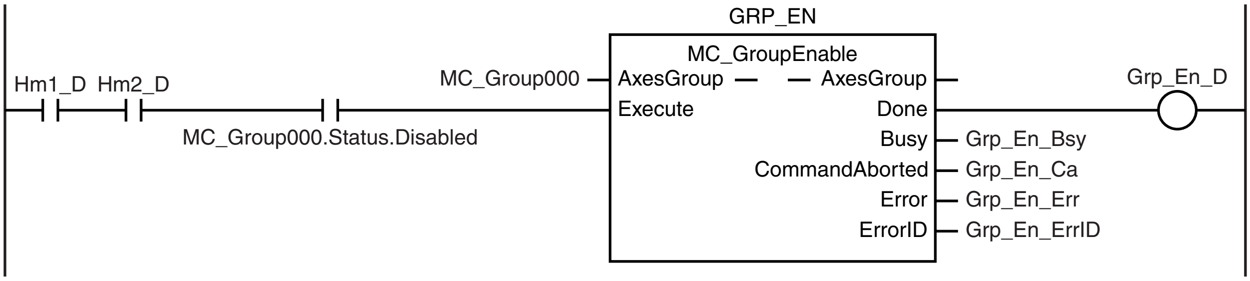

If the Servo is ON for axis 1 and home is not defined, the Home instruction is executed.



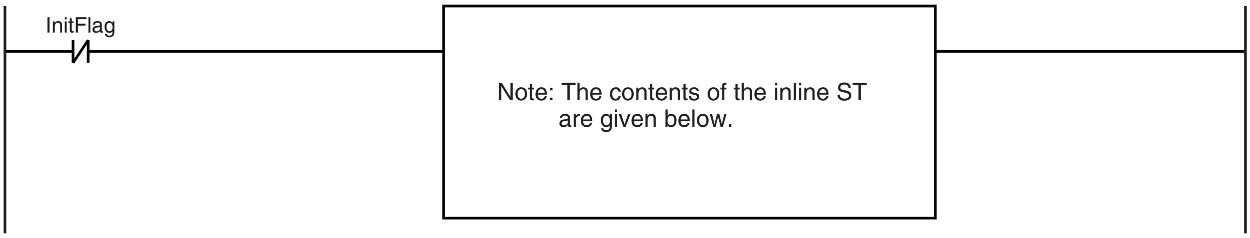

If the Servo is ON for axis 2 and home is not defined, the Home instruction is executed.



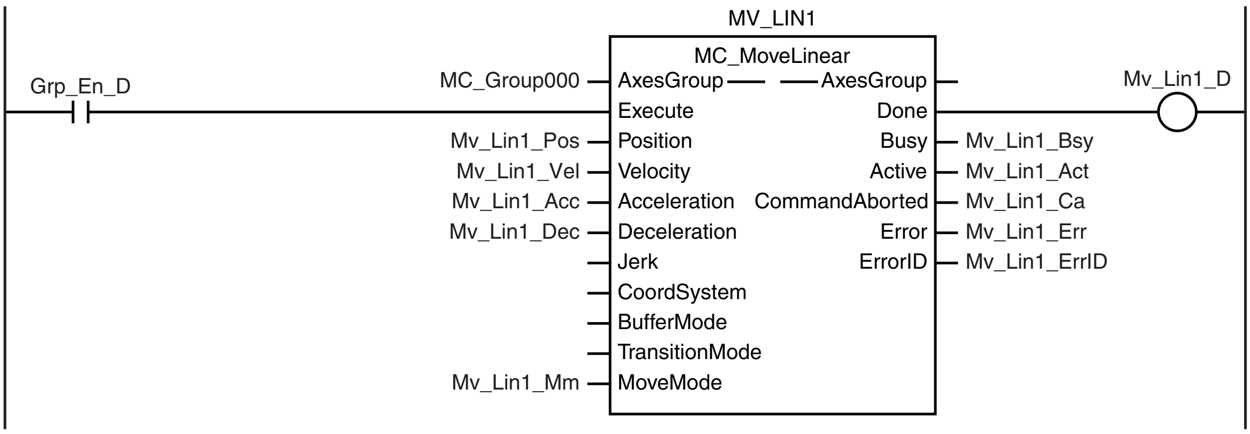
After home is defined for axis 1 and axis 2, the axes group is enabled.



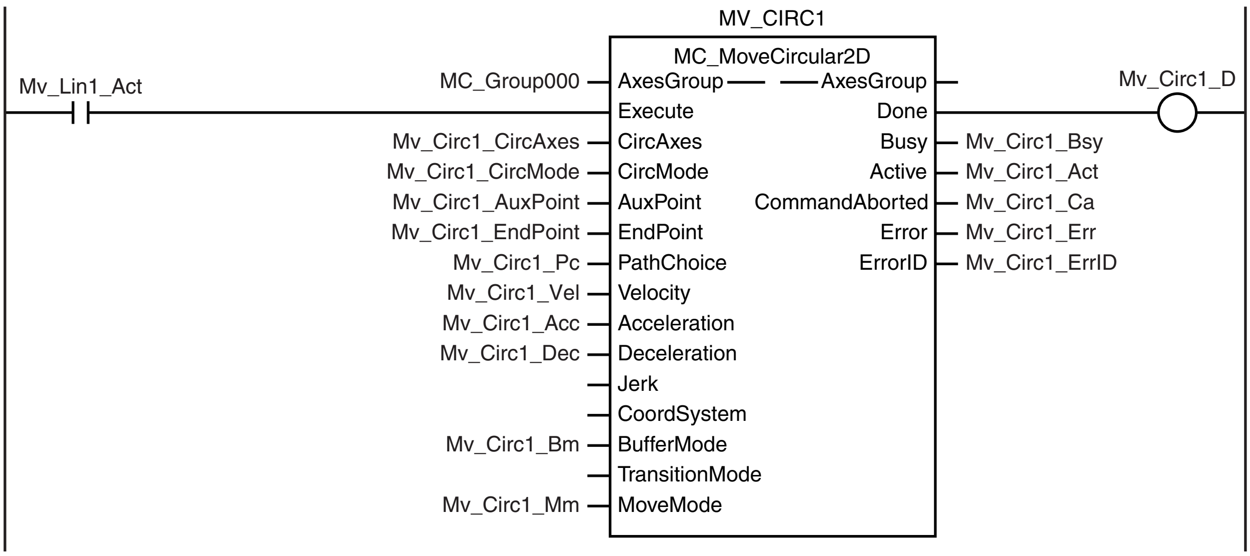

The parameters are set for linear interpolation and circular interpolation.



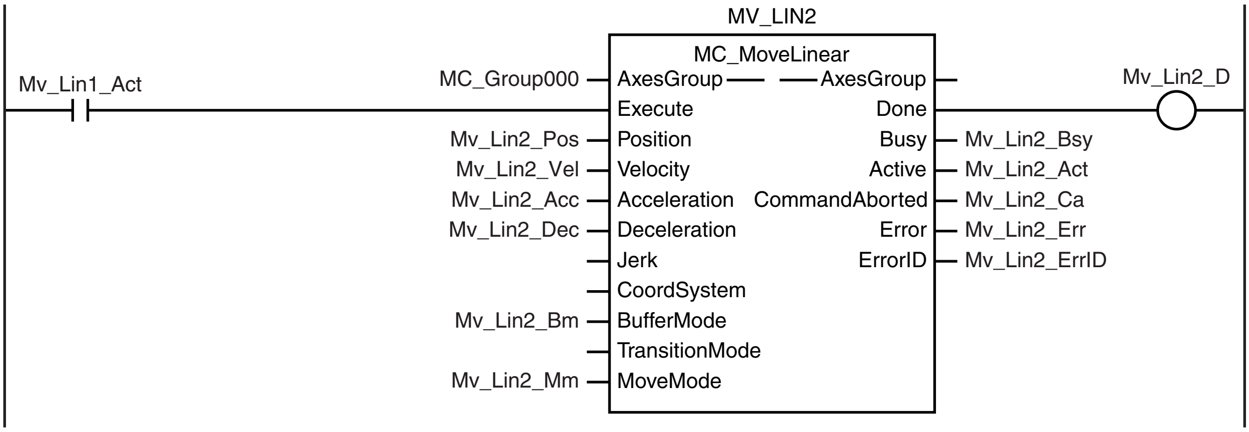

If the axes group is enabled, linear interpolation (1) is executed.



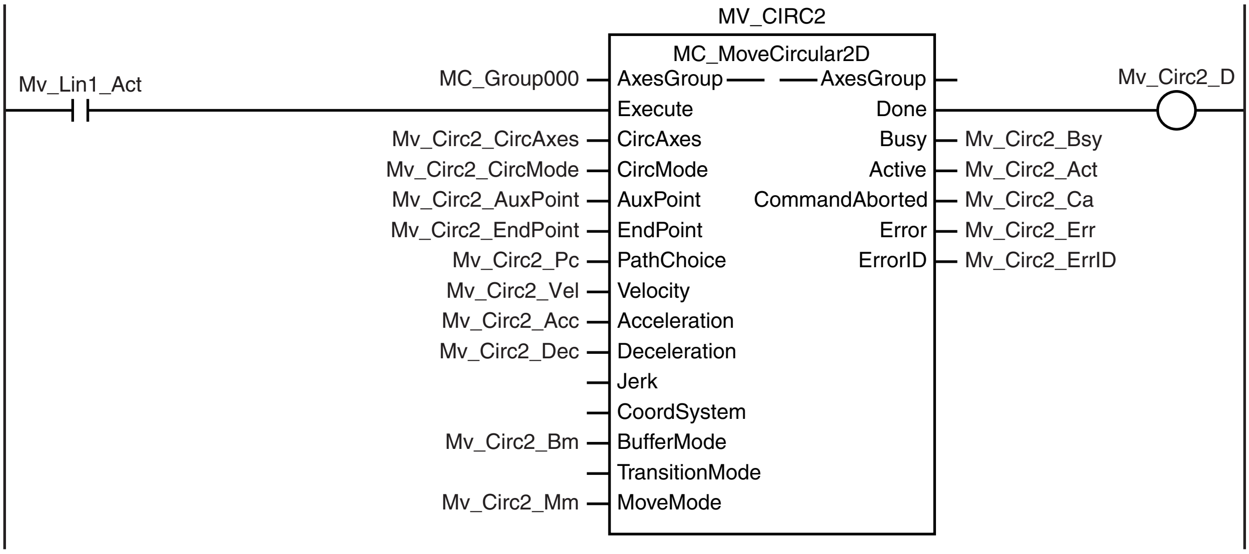

Circular interpolation (2) is executed with multi-execution of instructions after linear interpolation (1) is started.



Linear interpolation (3) is executed with multi-execution of instructions after linear interpolation (1) is started.




Circular interpolation (4) is executed with multi-execution of instructions after linear interpolation (1) is started.




## Contents of Inline ST

```
// MV_CIRC1 parameters
    Mv_Circ1_CircAxes[0]       := UINT#10#0;
    Mv_Circ1_CircAxes[1]       := UINT#10#1;
    Mv_Circ1_CircMode          := _eMC_CIRC_MODE#_mcRadius;
    Mv_Circ1_AuxPoint[0]       := LREAL#5.0;
    Mv_Circ1_AuxPoint[1]       := LREAL#0.0;
    Mv_Circ1_EndPoint[0]       := LREAL#15.0;
    Mv_Circ1_EndPoint[1]       := LREAL#10.0;
    Mv_Circ1_Pc                := _eMC_CIRC_PATHCHOICE#_mcCCW;
    Mv_Circ1_Vel               := LREAL#100.0;
    Mv_Circ1_Acc               := LREAL#20.0;
    Mv_Circ1_Dec               := LREAL#20.0;
    Mv_Circ1_Bm                := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Circ1_Mm                := _eMC_MOVE_MODE#_mcAbsolute;

// MV_CIRC2 parameters
    Mv_Circ2_CircAxes[0]       := UINT#10#0;
    Mv_Circ2_CircAxes[1]       := UINT#10#1;
    Mv_Circ2_CircMode          := _eMC_CIRC_MODE#_mcCenter;
    Mv_Circ2_AuxPoint[0]       := LREAL#15.0;
    Mv_Circ2_AuxPoint[1]       := LREAL#20.0;
    Mv_Circ2_EndPoint[0]       := LREAL#20.0;
    Mv_Circ2_EndPoint[1]       := LREAL#20.0;
    Mv_Circ2_Pc      := _eMC_CIRC_PATHCHOICE#_mcCW;
    Mv_Circ2_Vel     := LREAL#100.0;
    Mv_Circ2_Acc     := LREAL#20.0;
```

```
        Mv_Circ2_Dec     := LREAL#20.0;
        Mv_Circ2_Bm      := _eMC_BUFFER_MODE#_mcBuffered;
        Mv_Circ2_Mm      := _eMC_MOVE_MODE#_mcAbsolute;

    // MV_LIN1 parameters
        Mv_Lin1_Pos[0]      := LREAL#10#10.0;
        Mv_Lin1_Pos[1]      := LREAL#10#5.0;
        Mv_Lin1_Vel         := LREAL#10#100.0;
        Mv_Lin1_Acc         := LREAL#10#20.0;
        Mv_Lin1_Dec         := LREAL#10#20.0;
        Mv_Lin1_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

    // MV_LIN2 parameters
        Mv_Lin2_Pos[0]      := LREAL#10#15.0;
        Mv_Lin2_Pos[1]      := LREAL#10#15.0;
        Mv_Lin2_Vel         := LREAL#10#100.0;
        Mv_Lin2_Acc         := LREAL#10#20.0;
        Mv_Lin2_Dec         := LREAL#10#20.0;
        Mv_Lin2_Bm          := _eMC_BUFFER_MODE#_mcBuffered;
        Mv_Lin2_Mm          := _eMC_MOVE_MODE#_mcAbsolute;

    // Change InitFlag to TRUE after setting the input parameters.
        InitFlag := TRUE;
```

## Structured Text (ST)

### ● Main Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| MC_Group000 | _sGROUP_REF | --- | This is the Axes Group Variable for axes group 0. |
| MC_Group000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axes group 0. |
| MC_Group000.Status.Disabled | BOOL | FALSE | The value is TRUE when axes group 0 is disabled. |
| MC_Axis000 | _sAXIS_REF | --- | This is the Axis Variable for axis 1. |
| MC_Axis000.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 1. |
| MC_Axis000.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 1. |
| MC_Axis001 | _sAXIS_REF | --- | This is the Axis Variable for axis 2. |
| MC_Axis001.Details.Homed | BOOL | FALSE | TRUE when home is defined for axis 2. |
| MC_Axis001.MFaultLvl.Active | BOOL | FALSE | TRUE while there is a minor fault level error for axis 2. |
| Pwr1_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR1 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| Pwr2_Status | BOOL | FALSE | This variable is assigned to the *Status* output variable from the PWR2 instance of the MC_Power instruction. This variable changes to TRUE when the Servo is turned ON. |
| StartPg | BOOL | FALSE | The Servos for the axes in the axes group are turned ON if this variable is TRUE and EtherCAT process data communications are established. |

| Name | Data type | Default | Comment |
|---|---|---|---|
| InitFlag | BOOL | FALSE | This variable indicates if it is necessary to set the input parameters. Input parameters are set when this variable is FALSE. When setting the input parameters is completed, this variable changes to TRUE. |
| Hm1_Ex | BOOL | FALSE | The HM1 instance of MC_Home is executed when this variable changes to TRUE. |
| Hm2_Ex | BOOL | FALSE | The HM2 instance of MC_Home is executed when this variable changes to TRUE. |
| Grp_En_Ex | BOOL | FALSE | The GRP_EN instance of MC_GroupEnable is executed when this variable changes to TRUE. |
| Mv_Lin1_Ex | BOOL | FALSE | The MV_LIN1 instance of MC_MoveLinear is executed when this variable changes to TRUE. |
| Mv_Lin2_Ex | BOOL | FALSE | The MV_LIN2 instance of MC_MoveLinear is executed when this variable changes to TRUE. |
| Mv_Circ1_Ex | BOOL | FALSE | The MV_CIRC1 instance of MC_MoveCircular is executed when this variable changes to TRUE. |
| Mv_Circ2_Ex | BOOL | FALSE | The MV_CIRC2 instance of MC_MoveCircular is executed when this variable changes to TRUE. |

## ● Sample Programming

```
// Processing when input parameters are not set
IF InitFlag=FALSE THEN

    // MV_CIRC1 parameters
    Mv_Circ1_CircAxes[0]     := UINT#10#0;
    Mv_Circ1_CircAxes[1]     := UINT#10#1;
    Mv_Circ1_CircMode        := _eMC_CIRC_MODE#_mcRadius;
    Mv_Circ1_AuxPoint[0]     := LREAL#5.0;
    Mv_Circ1_AuxPoint[1]     := LREAL#0.0;
    Mv_Circ1_EndPoint[0]     := LREAL#15.0;
    Mv_Circ1_EndPoint[1]     := LREAL#10.0;
    Mv_Circ1_Pc              := _eMC_CIRC_PATHCHOICE#_mcCCW;
    Mv_Circ1_Vel             := LREAL#100.0;
    Mv_Circ1_Acc             := LREAL#20.0;
    Mv_Circ1_Dec             := LREAL#20.0;
    Mv_Circ1_Bm              := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Circ1_Mm              := _eMC_MOVE_MODE#_mcAbsolute;

    // MV_CIRC2 parameters
    Mv_Circ2_CircAxes[0]     := UINT#10#0;
    Mv_Circ2_CircAxes[1]     := UINT#10#1;
    Mv_Circ2_CircMode        := _eMC_CIRC_MODE#_mcCenter;
    Mv_Circ2_AuxPoint[0]     := LREAL#15.0;
    Mv_Circ2_AuxPoint[1]     := LREAL#20.0;
    Mv_Circ2_EndPoint[0]     := LREAL#20.0;
    Mv_Circ2_EndPoint[1]     := LREAL#20.0;
    Mv_Circ2_Pc              := _eMC_CIRC_PATHCHOICE#_mcCW;
    Mv_Circ2_Vel             := LREAL#100.0;
    Mv_Circ2_Acc             := LREAL#20.0;
    Mv_Circ2_Dec             := LREAL#20.0;
    Mv_Circ2_Bm              := _eMC_BUFFER_MODE#_mcBuffered;
    Mv_Circ2_Mm              := _eMC_MOVE_MODE#_mcAbsolute;

    // MV_LIN1 parameters
    Mv_Lin1_Pos[0]      := LREAL#10.0;
    Mv_Lin1_Pos[1]      := LREAL#5.0;
    Mv_Lin1_Vel         := LREAL#100.0;
    Mv_Lin1_Acc         := LREAL#20.0;
```

```
        Mv_Lin1_Dec          := LREAL#20.0;
        Mv_Lin1_Mm           := _eMC_MOVE_MODE#_mcAbsolute;

        // MV_LIN2 parameters
        Mv_Lin2_Pos[0]       := LREAL#15.0;
        Mv_Lin2_Pos[1]       := LREAL#15.0;
        Mv_Lin2_Vel          := LREAL#100.0;
        Mv_Lin2_Acc          := LREAL#20.0;
        Mv_Lin2_Dec          := LREAL#20.0;
        Mv_Lin2_Bm           := _eMC_BUFFER_MODE#_mcBuffered;
        Mv_Lin2_Mm           := _eMC_MOVE_MODE#_mcAbsolute;

        // Change InitFlag to TRUE after setting the input parameters.
        InitFlag := TRUE;

    END_IF;


    // If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 1 is turned ON.
    // If EtherCAT communications are not normal, the Servo is turned OFF.
    IF (StartPg=TRUE)
     AND (_EC_PDSlavTbl[MC_Axis000.Cfg.NodeAddress]=TRUE)
     AND (_EC_CommErrTbl[MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
        Pwr1_En:=TRUE;          // Turn ON the Servo for axis 1.
    ELSE
        Pwr1_En:=FALSE;         // Turn OFF the Servo for axis 1.
    END_IF;

    // If StartPg is TRUE and EtherCAT communications are normal, the Servo for axis 2 is turned ON.
    // If EtherCAT communications are not normal, the Servo is turned OFF.
    IF (StartPg=TRUE)
     AND (_EC_PDSlavTbl[MC_Axis001.Cfg.NodeAddress]=TRUE)
     AND (_EC_CommErrTbl[MC_Axis001.Cfg.NodeAddress]=FALSE) THEN
        Pwr2_En:=TRUE;          // Turn ON the Servo for axis 2.
    ELSE
        Pwr2_En:=FALSE;         // Turn OFF the Servo for axis 2.
    END_IF;

    // Processing for a minor fault level error
    // Implement FaultHandler separately.
    IF (MC_Axis000.MFaultLvl.Active=TRUE) OR (MC_Axis001.MFaultLvl.Active=TRUE) OR
    (MC_Group000.MFaultLvl.Active=TRUE) THEN
        FaultHandler();
    END_IF;

    // If the Servo is ON for axis 1 and home is not defined, the Home instruction is executed.
    IF (Pwr1_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
        Hm1_Ex:=TRUE;
    END_IF;

    // If the Servo is ON for axis 2 and home is not defined, the Home instruction is executed.
    IF (Pwr2_Status=TRUE) AND (MC_Axis001.Details.Homed=FALSE) THEN
        Hm2_Ex:=TRUE;
    END_IF;

    // If axes group 0 is disabled after homing is completed for axis 1 and axis 2, it is enabled.
    IF (Hm1_D=TRUE) AND (Hm2_D=TRUE) AND (MC_Group000.Status.Disabled=TRUE) THEN
        Grp_En_Ex:= TRUE;
    END_IF;

    // If axes group 0 is enabled, linear interpolation (1) is executed.
    IF Grp_En_D=TRUE THEN
        Mv_Lin1_Ex:=TRUE;
    END_IF;
```

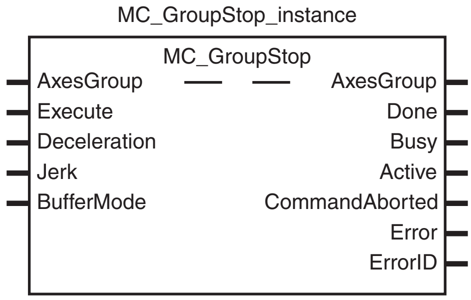
```
// The rest of the instructions are executed with multi-execution of instructions when the Active output variable
for linear interpolation (1) changes to TRUE.
IF Mv_Lin1_Act=TRUE THEN
    Mv_Circ1_Ex:=TRUE;
    Mv_Lin2_Ex:=TRUE;
    Mv_Circ2_Ex:=TRUE;
END_IF;


// MC_Power for axis 1
PWR1(
    Axis        := MC_Axis000,
    Enable      := Pwr1_En,
    Status      => Pwr1_Status,
    Busy        => Pwr1_Bsy,
    Error       => Pwr1_Err,
    ErrorID     => Pwr1_ErrID
);

// MC_Power for axis 2
PWR2(
    Axis        := MC_Axis001,
    Enable      := Pwr2_En,
    Status      => Pwr2_Status,
    Busy        => Pwr2_Bsy,
    Error       => Pwr2_Err,
    ErrorID     => Pwr2_ErrID
);

// MC_Home for axis 1
HM1(
    Axis                := MC_Axis000,
    Execute             := Hm1_Ex,
    Done                => Hm1_D,
    Busy                => Hm1_Bsy,
    CommandAborted      => Hm1_Ca,
    Error               => Hm1_Err,
    ErrorID             => Hm1_ErrID
);

// MC_Home for axis 2
HM2(
    Axis                := MC_Axis001,
    Execute             := Hm2_Ex,
    Done                => Hm2_D,
    Busy                => Hm2_Bsy,
    CommandAborted      => Hm2_Ca,
    Error               => Hm2_Err,
    ErrorID             => Hm2_ErrID
);


// Axes Group 0 is enabled.
GRP_EN(
    AxesGroup           := MC_Group000,
    Execute             := Grp_En_Ex,
    Done                => Grp_En_D,
    Busy                => Grp_En_Bsy,
    CommandAborted      => Grp_En_Ca,
    Error               => Grp_En_Err,
    ErrorID             => Grp_En_ErrID
);

// Linear interpolation 1
MV_LIN1(
```

```
    AxesGroup          := MC_Group000,
    Execute            := Mv_Lin1_Ex,
    Position           := Mv_Lin1_Pos,
    Velocity           := Mv_Lin1_Vel,
    Acceleration       := Mv_Lin1_Acc,
    Deceleration       := Mv_Lin1_Dec,
    MoveMode           := Mv_Lin1_Mm,
    Done               => Mv_Lin1_D,
    Busy               => Mv_Lin1_Bsy,
    Active             => Mv_Lin1_Act,
    CommandAborted     => Mv_Lin1_Ca,
    Error              => Mv_Lin1_Err,
    ErrorID            => Mv_Lin1_ErrID
);

// Circular interpolation (2)
MV_CIRC1(
    AxesGroup          := MC_Group000,
    Execute            := Mv_Circ1_Ex,
    CircAxes           := Mv_Circ1_CircAxes,
    CircMode           := Mv_Circ1_CircMode,
    AuxPoint           := Mv_Circ1_AuxPoint,
    EndPoint           := Mv_Circ1_EndPoint,
    PathChoice         := Mv_Circ1_Pc,
    Velocity           := Mv_Circ1_Vel,
    Acceleration       := Mv_Circ1_Acc,
    Deceleration       := Mv_Circ1_Dec,
    BufferMode         := Mv_Circ1_Bm,
    MoveMode           := Mv_Circ1_Mm,
    Done               => Mv_Circ1_D,
    Busy               => Mv_Circ1_Bsy,
    Active             => Mv_Circ1_Act,
    CommandAborted     => Mv_Circ1_Ca,
    Error              => Mv_Circ1_Err,
    ErrorID            => Mv_Circ1_ErrID
);

// Linear interpolation (3)
MV_LIN2(
    AxesGroup          := MC_Group000,
    Execute            := Mv_Lin2_Ex,
    Position           := Mv_Lin2_Pos,
    Velocity           := Mv_Lin2_Vel,
    Acceleration       := Mv_Lin2_Acc,
    Deceleration       := Mv_Lin2_Dec,
    BufferMode         := Mv_Lin2_Bm,
    MoveMode           := Mv_Lin2_Mm,
    Done               => Mv_Lin2_D,
    Busy               => Mv_Lin2_Bsy,
    Active             => Mv_Lin2_Act,
    CommandAborted     => Mv_Lin2_Ca,
    Error              => Mv_Lin2_Err,
    ErrorID            => Mv_Lin2_ErrID
);

// Circular interpolation (4)
MV_CIRC2(
    AxesGroup          := MC_Group000,
    Execute            := Mv_Circ2_Ex,
    CircAxes           := Mv_Circ2_CircAxes,
    CircMode           := Mv_Circ2_CircMode,
    AuxPoint           := Mv_Circ2_AuxPoint,
    EndPoint           := Mv_Circ2_EndPoint,
    PathChoice         := Mv_Circ2_Pc,
    Velocity           := Mv_Circ2_Vel,
```

```
    Acceleration       := Mv_Circ2_Acc,
    Deceleration       := Mv_Circ2_Dec,
    BufferMode         := Mv_Circ2_Bm,
    MoveMode           := Mv_Circ2_Mm,
    Done               => Mv_Circ2_D,
    Busy               => Mv_Circ2_Bsy,
    Active             => Mv_Circ2_Act,
    CommandAborted     => Mv_Circ2_Ca,
    Error              => Mv_Circ2_Err,
    ErrorID            => Mv_Circ2_ErrID
);
```

# MC_GroupStop

The MC_GroupStop instruction decelerates all of the axes in an interpolated motion to a stop.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_GroupStop | Group Stop | FB |  | MC_GroupStop_instance (<br>AxesGroup :=*parameter,*<br>Execute :=*parameter,*<br>Deceleration :=*parameter,*<br>Jerk :=*parameter,*<br>BufferMode :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Active =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Deceleration | Deceleration Rate | LREAL | Non-negative number | 0 | Specify the deceleration rate. The unit is command units/$s^2$.*1 |
| Jerk | Jerk | LREAL | Non-negative number | 0 | Specify the jerk. The unit is command units/$s^3$.*1 |
| BufferMode | Buffer Mode Selection | _eMC_ BUFFER_ MODE | 0: _mcAborting | 0*2 | Specify the operation when executing more than one motion instruction.<br>0: Aborting |

*1 Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

*2 The default value for an enumeration variable is actually not the number, but the enumerator.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Active | Controlling | BOOL | TRUE or FALSE | TRUE when the axes group is being controlled. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the instruction is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Active | When the instruction is started. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another motion control instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error.<br>• When this instruction is executed while there is an error. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| AxesGroup | Axes Group | _sGROUP_REF | --- | Specify the axes group.* |

* Specify an Axes Group Variable that was created in the Axes Group Basic Settings of the Sysmac Studio. (The default axes group variable names are *MC_Group****.)

## Function

- This instruction decelerates all of the axes that are in motion for an interpolation instruction (i.e., all of the axes in the axes group that is specified with *AxesGroup*) from the actual velocity to a velocity of 0 (i.e. until they stop), and then disables the axes group command. It can be used for the following interpolation instructions: MC_MoveLinearAbsolute (Absolute Linear Interpolation), MC_MoveLinearRelative (Relative Linear Interpolation), and MC_MoveCircular2D (Circular 2D Interpolation).
- *CommandAborted* for the interpolation instruction that is currently in operation will change to TRUE when this instruction is executed.
- If you execute this instruction while an interpolation instruction is in execution, the axes will decelerate to a stop along the linear interpolation or circular interpolation path.
- The deceleration stop operation starts when *Execute* changes to TRUE.

**Precautions for Correct Use**

This instruction is not executed if *Status.ErrorStop* (Error Deceleration Stopping) in the Axes Group Variable is TRUE.
Use the MC_GroupImmediateStop instruction to stop the motion of an axes group that is decelerating to a stop for an error.

## Instruction Details

This section describes the instruction in detail.

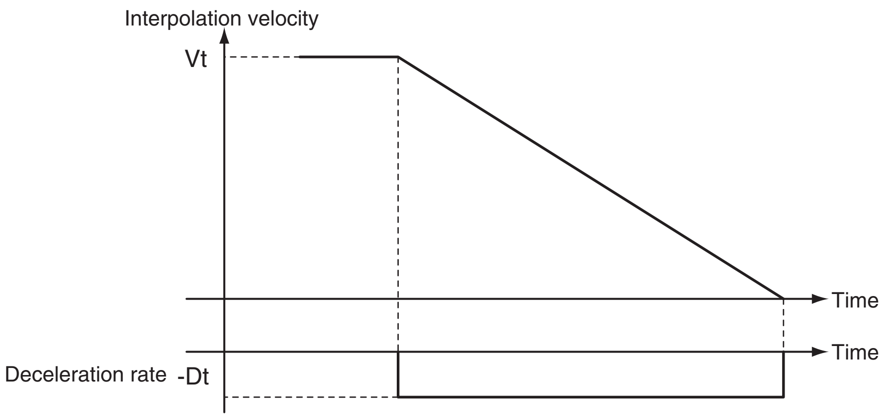
### ● *Deceleration* (Deceleration Rate) and *Jerk*

Set the input variables *Deceleration* and *Jerk* to set the deceleration rate and jerk when decelerating to a stop. When this instruction is executed, *Deceleration* (Deceleration Rate) and *Jerk* specified for this instruction are used for the interpolation velocity.

#### *Jerk*

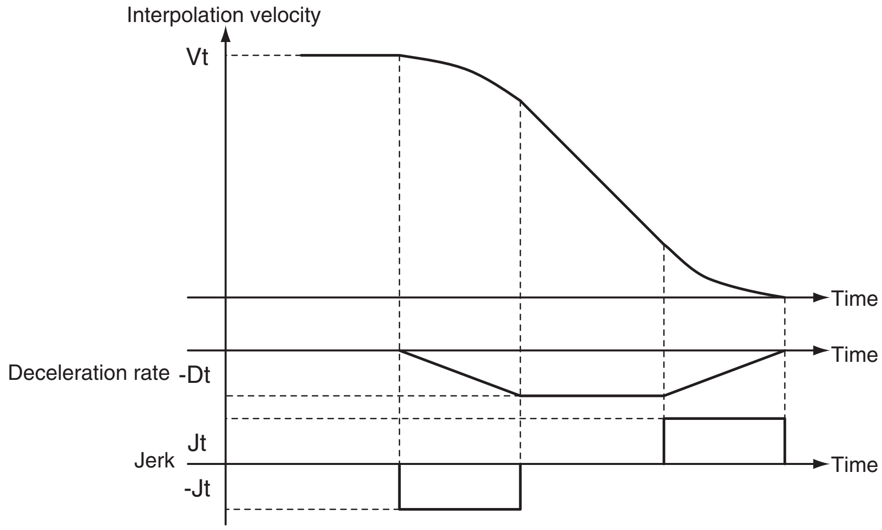
The relationships between the deceleration rate and interpolation velocity when *Jerk* is set to 0 and when it is set to any other value are shown below.

- *Jerk* Set to 0
  The command value for the velocity is created with deceleration rate Dt.




Vt: Velocity when deceleration starts, Dt: Specified deceleration rate

- *Jerk* Set to Value Other Than 0
  The command value for the velocity is created based on the current velocity with Dt as the upper limit to the deceleration rate.




Vt: Velocity when deceleration starts, Dt: Specified deceleration rate, Jt: Specified jerk

**Additional Information**

- If 0 is specified for the deceleration rate, an immediate stop is performed and the axis stops immediately.
- An immediate stop occurs regardless of the setting of the Acceleration/Deceleration Over Limit Selection axis parameter only when the deceleration rate is set to 0.

## ● *BufferMode* (Buffer Mode Selection)

*BufferMode* specifies how to join the axis motions for this interpolation instruction and the previous interpolation instruction. There is only the following setting.

| Buffer Mode Selection | Description |
|---|---|
| Aborting | Aborts the instruction being executed and executes this instruction. |

For details on *BufferMode* (Buffer Mode Selection), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

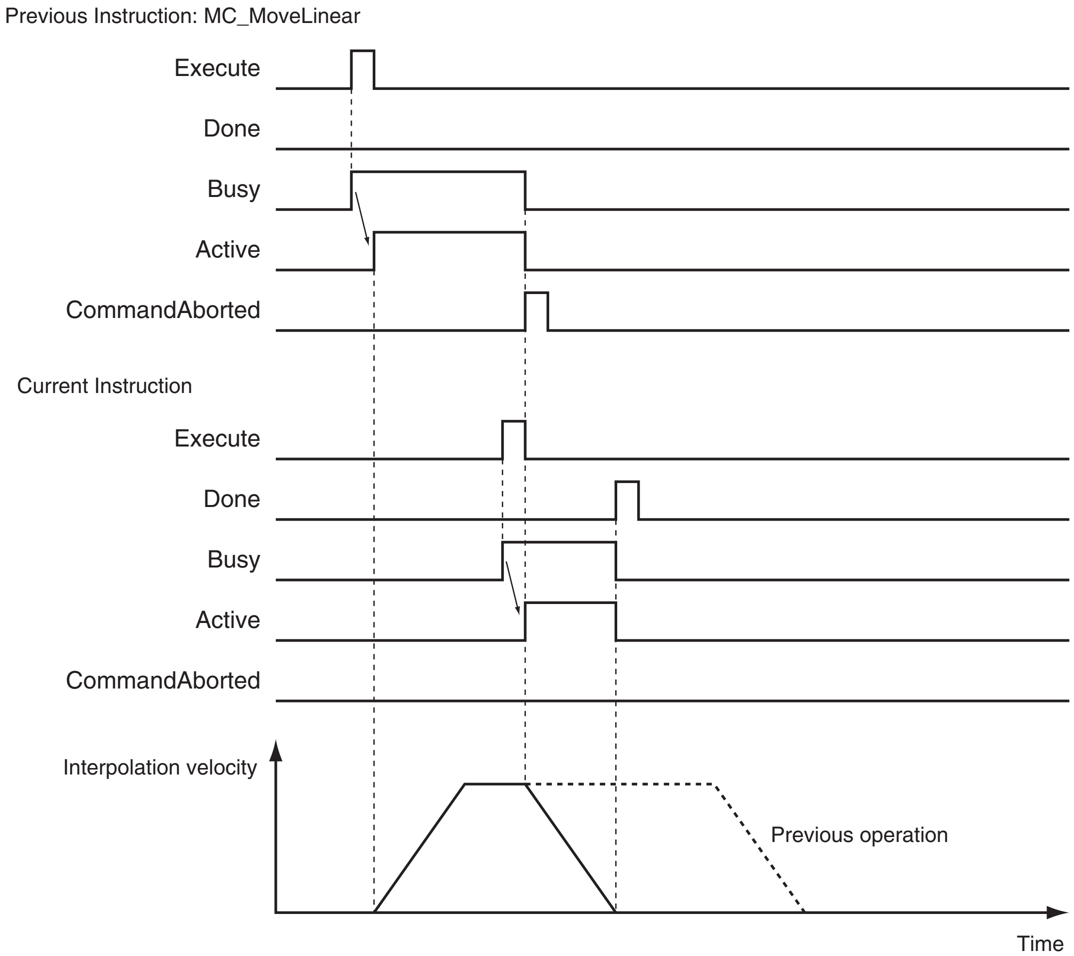
## ● In-position Check

An in-position check is not performed when stopping for this instruction.

## Timing Charts

- *Busy* (Executing) changes to TRUE at the same time as *Execute* changes to TRUE. *Active* (Controlling) changes to TRUE in the next period.
- *Done* changes to TRUE when a velocity of 0 is reached.
- If another instruction aborts this instruction, *CommandAborted* changes to TRUE and *Busy* (Executing) and *Active* (Controlling) change to FALSE.

The following chart shows decelerating to a stop for linear interpolation. *CommandAborted* for the interpolation instruction that is currently in operation will change to TRUE when MC_GroupStop is executed.

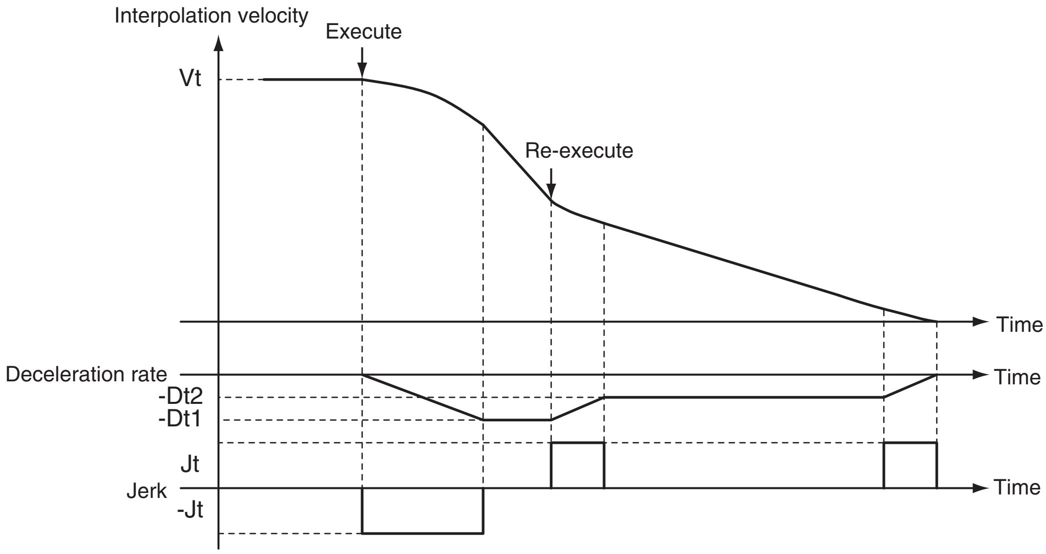


## Re-execution of Motion Control Instructions

The deceleration rate changes if *Execute* changes to TRUE again while this instruction is being executed. The *Jerk* setting is not changed when a motion control instruction is re-executed.

### ● *Jerk* Set to Value Other Than 0

The velocity command value is created with Dt2 as the upper limit of the deceleration rate after it has changed based on the actual velocity and actual deceleration rate.




Vt: Velocity when deceleration starts, Dt: Specified deceleration rate, Jt: Specified jerk

## Multi-execution of Motion Control Instructions

There are restrictions to execution of this instruction, and to other instructions executed during execution of this instruction. For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

#### When Axes Group Is Disabled

An error occurs for the axes group if the MC_GroupStop instruction is executed for a disable axes group. However, this will not affect the axes.

### When the *Status.Stopping* (Axes Group Deceleration Stopping) in the Axes Group Variable Is TRUE

*Status.Stopping* (Deceleration Stopping) in the Axis Variable changes to TRUE in the following cases.

- While the axes group is decelerating for the MC_GroupStop Instruction
- While *Execute* is TRUE for one or more MC_GroupStop instructions

If you execute one of the following instructions for an axes group that is decelerating to a stop, *CommandAborted* of the executed instruction changes to TRUE.

- MC_MoveLinear (Linear Interpolation) instruction
- MC_MoveLinearAbsolute (Absolute Linear Interpolation) instruction
- MC_MoveLinearRelative (Relative Linear Interpolation) instruction
- MC_MoveCircular2D (Circular 2D Interpolation) instruction

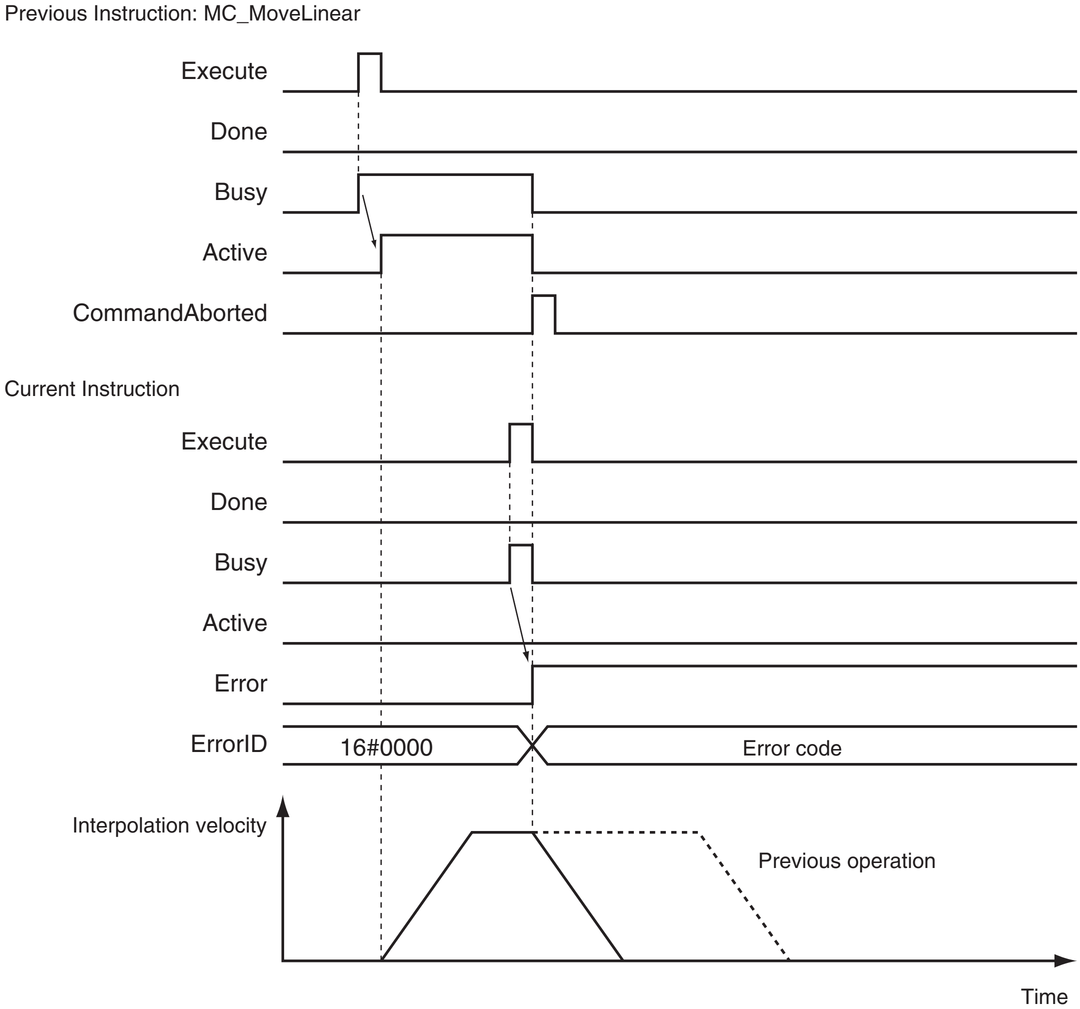
When the MC_GroupStop instruction is in execution, you can execute another MC_GroupStop instruction with multi-execution of instructions. *Done* from the MC_GroupStop instruction that is in execution changes to TRUE.

### When the *Status.ErrorStop* (Axes Group Error Deceleration Stopping) in the Axes Group Variable Is TRUE

*Status.ErrorStop* (Error Deceleration Stopping) in the axes group status is TRUE while there is an error for the axes group.
If the MC_GroupStop instruction is executed when *Status.ErrorStop* (Error Deceleration Stopping) is TRUE, *CommandAborted* changes to TRUE.
Use the MC_GroupImmediateStop (Axes Group Stop) instruction instead.

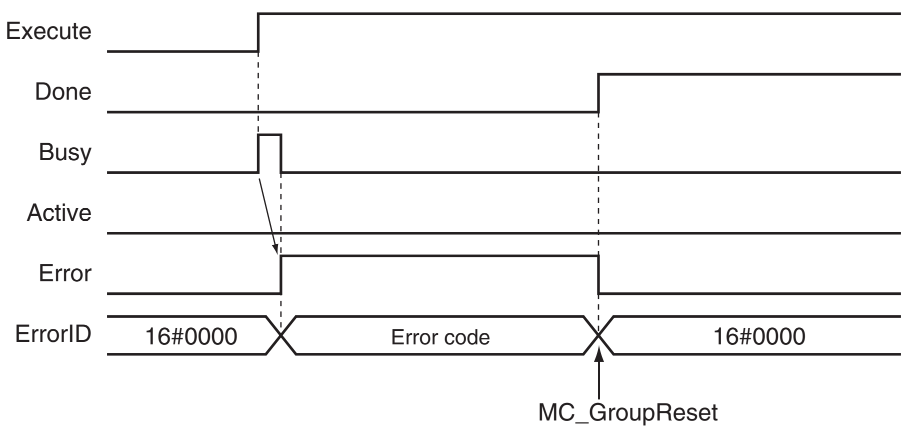
## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and the axes will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).




### ● Operation When Error is Cleared

If an error occurs for this instruction and the error is reset while *Execute* is TRUE, Error changes to FALSE, *Done* changes to TRUE, and *Status.Stopping* (Deceleration Stopping) in the axis status changes to TRUE in the same way as for a normal execution of a deceleration stop. Reset errors after the axes group has stopped. Do not reset errors during axes group motion.

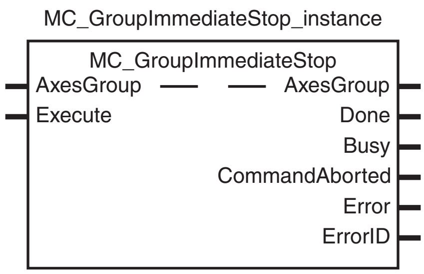


### Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_GroupImmediateStop

The MC_GroupImmediateStop instruction stops all axes in an interpolated motion.
If the specified axes group is enabled, all of the composition axes are stopped according to the stop mode that is specified in Immediate Stop Input Stop Method Selection regardless of the current status of the axes.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_GroupImmediateStop | Axes Group Immediate Stop | FB |  | MC_GroupImmediateStop_ instance (<br>AxesGroup :=*parameter,*<br>Execute :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs.<br>A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the immediate stop is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | • When this instruction is aborted because another deceleration stop instruction was executed with the Buffer Mode set to *Aborting*.<br>• When this instruction is canceled due to an error. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| AxesGroup | Axes Group | _sGROUP_REF | --- | Specify the axes group.* |

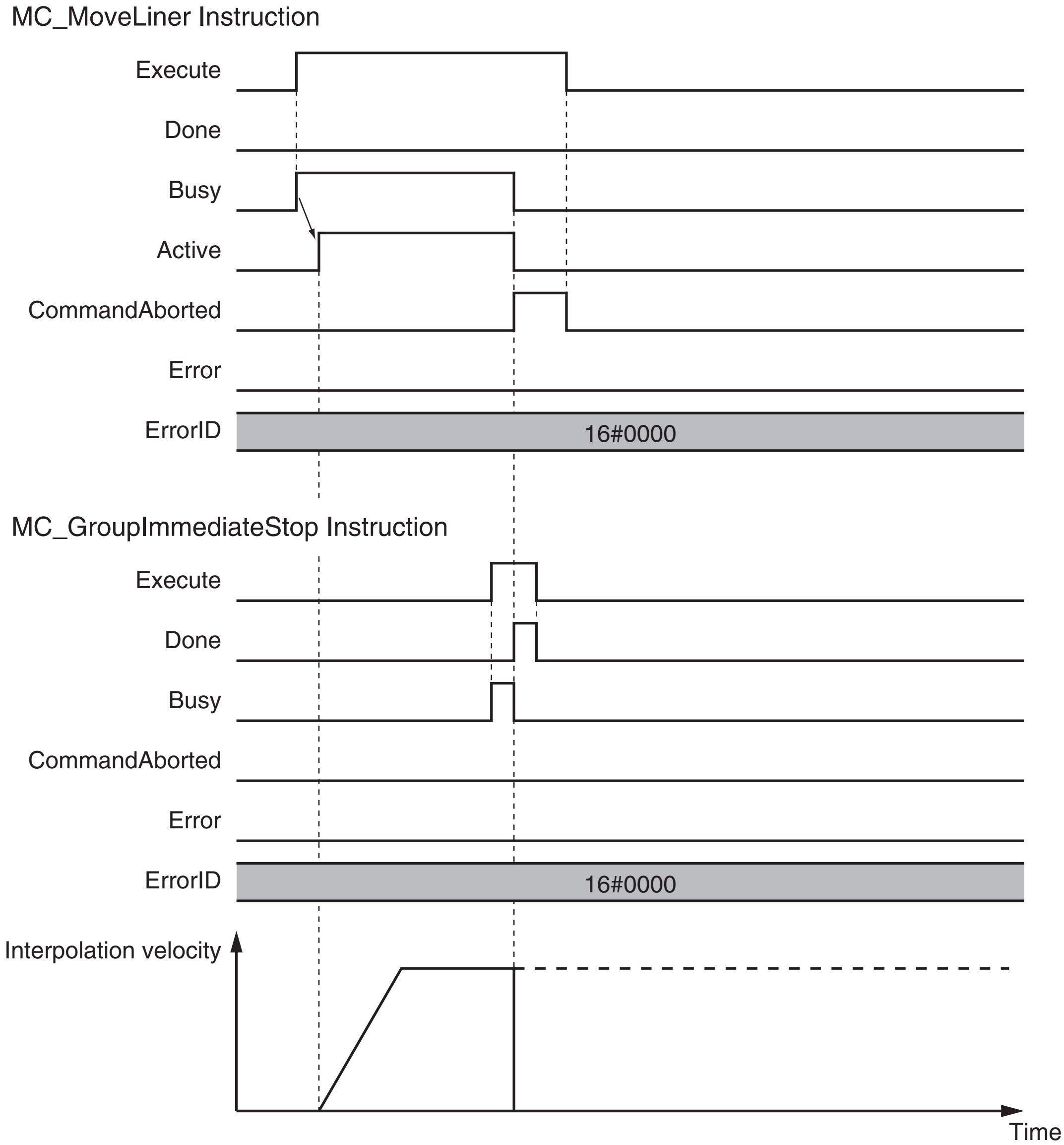
* Specify an Axes Group Variable that was created in the Axes Group Basic Settings of the Sysmac Studio. (The default axes group variable names are *MC_Group****.)

# Function

- You can execute the MC_GroupImmediateStop instruction for an entire axes group that is in motion for an interpolation instruction (i.e., the axes in the axes group that is specified with AxesGroup) or for an axes group that is stopping for the MC_GroupStop instruction or error.
  You can execute it for the following interpolation instructions: MC_MoveLinear (Linear Interpolation), MC_MoveLinearAbsolute (Absolute Linear Interpolation), MC_MoveLinearRelative (Relative Linear Interpolation), and MC_MoveCircular2D (Circular 2D Interpolation).
- When this instruction is executed, the axis stops immediately according to the setting of the Immediate Stop Input Stop Method axis parameter. *CommandAborted* changes to TRUE for the instruction that is currently in operation.
- When the instruction is executed, *Status.ErrorStop* (Error Deceleration Stopping) in the axis status changes to TRUE and an Axes Group Immediate Stop Instruction Executed error (error code: 5486 hex) occurs when the instruction is executed.
  For details on the Axes Group Immediate Stop Instruction Executed error (error code: 5486 hex), refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Timing Charts

- *Busy* (Executing) changes to TRUE when *Execute* changes to TRUE.
- *Done* changes to TRUE when processing of this instruction is completed.



## Re-execution of Motion Control Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted, and the axis stops.

## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution during Execution of Other Instructions

#### When the Axes Group Is Disabled

An axes group error will occur if this instruction is executed for a disabled axes group. However, this will not affect the axes.

#### When the *Status.Stopping* (Deceleration Stopping) in the Axes Group Variable Is TRUE

*Status.Stopping* (Deceleration Stopping) in the Axes Group Variable changes to TRUE in the following cases.

- While the axis is decelerating for the MC_GroupStop Instruction
- While *Execute* is TRUE for one or more MC_GroupStop instructions

You can execute this instruction for an axes group that is currently decelerating to a stop.
When this instruction is executed, *CommandAborted* for the MC_GroupStop instruction that is in operation will change to TRUE.

#### When the *Status.ErrorStop* (Error Deceleration Stopping) in the Axes Group Variable Is TRUE

*Status.ErrorStop* (Error Deceleration Stopping) in the axes group status is TRUE while there is an error for the axes group. You can also execute this instruction for an axes group that is decelerating to a stop for an error.

## Errors

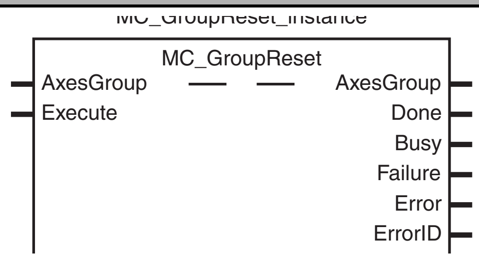
If an error occurs during instruction execution, *Error* will change to TRUE and the axis will stop.
You can find out the cause of the error by referring to the value output to *ErrorID* (Error Code).

### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_GroupSetOverride

The MC_GroupSetOverride instruction changes the combined target velocity during an interpolated motion.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_GroupSetOverride | Set Group Overrides | FB |  | MC_GroupSetOverride_instance (<br>AxesGroup :=*parameter,*<br>Enable :=*parameter,*<br>VelFactor :=*parameter,*<br>AccFactor :=*parameter,*<br>JerkFactor :=*parameter,*<br>Enabled =>*parameter,*<br>Busy =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Enable | Enable | BOOL | TRUE or FALSE | FALSE | The override factors are enabled when *Enable* is TRUE. The override factors return to 100% when Enable changes to FALSE. |
| VelFactor | Velocity Override Factor | LREAL | 0 to 500 | 100 | Specify the velocity override factor. The valid range of the override factor is between 0.01and 500.00.<br>Values above 500.00 are treated as 500 and values less than 0.01 (including negative values) are treated as 0.01.<br>The override factor will be 0 only when 0 is specified.<br>The unit is %. |
| AccFactor (Reserved) | Acceleration/ Deceleration Override Factor | LREAL | 0 to 500 | 100 | (Reserved) |
| JerkFactor (Reserved) | Jerk Override Factor | LREAL | 0 to 500 | 100 | (Reserved) |

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Enabled | Enabled | BOOL | TRUE or FALSE | TRUE when the axes group is being controlled. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Enabled | When this instruction is started. | • After one period when *Enable* is FALSE.<br>• When *Error* changes to TRUE. |
| Busy | When *Enable* changes to TRUE. | • When *Error* changes to TRUE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| AxesGroup | Axes Group | _sGROUP_REF | --- | Specify the axes group.* |

* Specify an Axes Group Variable that was created in the Axes Group Basic Settings of the Sysmac Studio. (The default axes group variable names are *MC_Group****.)

# Function

- This instruction changes the override factors related to the interpolation target velocity for the group controlled by the next instruction. Changes the target velocity of the axes in operation by changing the override.
  The override factors apply to the following instructions.

| | |
|---|---|
| MC_MoveLinear (Linear Interpolation) instruction | MC_MoveLinearRelative (Relative Linear Interpolation) instruction |
| MC_MoveLinearAbsolute (Absolute Linear Interpolation) instruction | MC_MoveCircular2D (Circular 2D Interpolation) instruction |

- The following is the new target velocity.
  Target velocity after the change = Interpolation velocity of the current instruction × Override factor (%)
- The unit for override factors is %. A setting of 100 indicates 100%.
- If the interpolation velocity that results from the override exceeds the maximum interpolation velocity set in the Axes Group Variables, the maximum interpolation velocity is used.
- The axis will accelerate or decelerate to the interpolation velocity that results from the override.
- If the velocity override factor is set to 0, the target velocity will be 0. Axes group operation will decelerate to a velocity of 0, and operation will continue. *Status.Moving* in the Axes Group Variable is TRUE during axes group motion. If you want to pause the axis motion while keeping the operation status, set the override factor to 0.
- The override factors will return to 100% when *Enable* changes to FALSE.

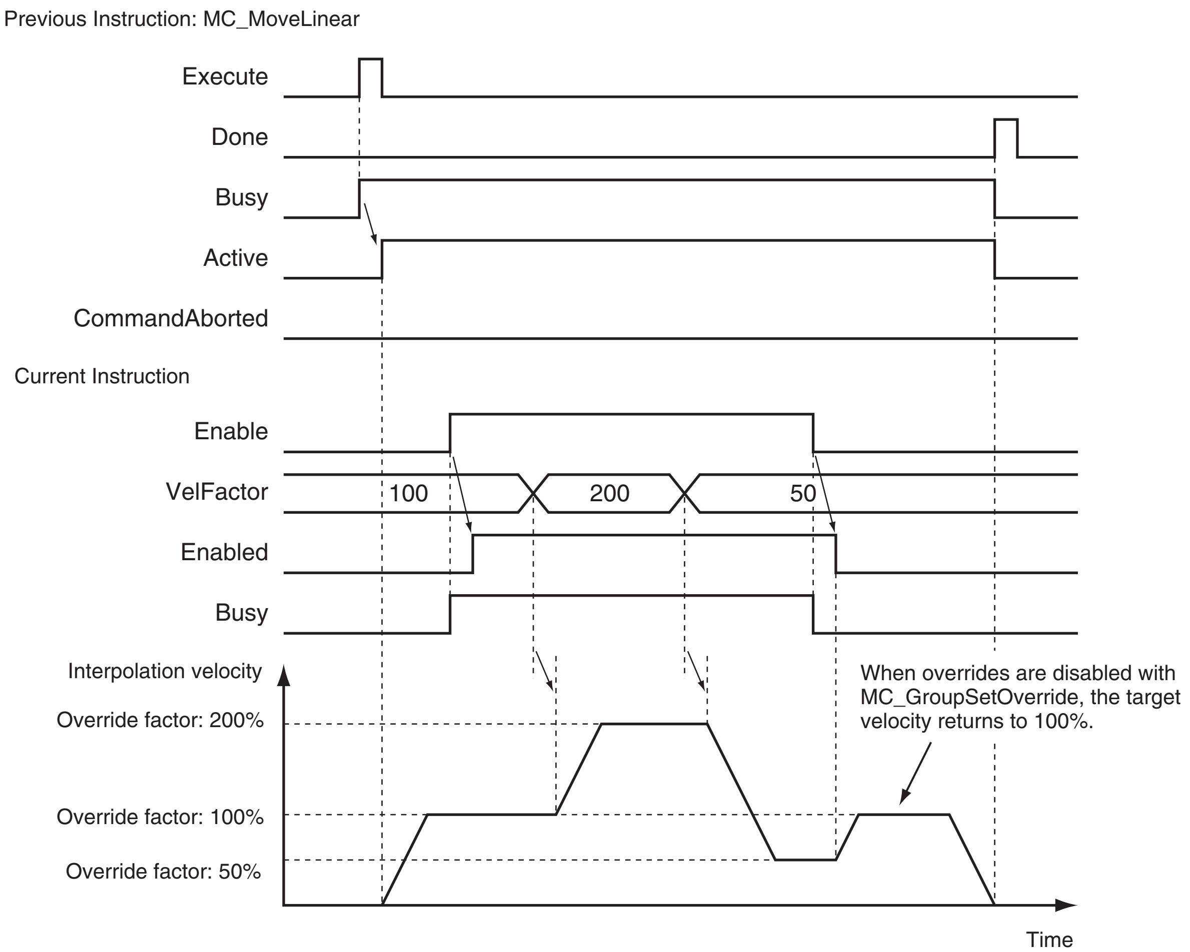
- If an axes group error occurs during MC_GroupSetOverride execution, the value of *Enable* for MC_GroupSetOverride is maintained.

**Precautions for Correct Use**

When *Enable* to this instruction changes to FALSE, *Enabled* and *Busy* from this instruction change to FALSE.
The axis will accelerate or decelerate to a velocity with a 100% override factor.

## Timing Charts

### ● Using this Instruction for the MC_MoveLinear (Linear Interpolation) Instruction




## Re-execution of Motion Control Instructions

You cannot re-execute enable-type motion control instructions.

## Multi-execution of Motion Control Instructions

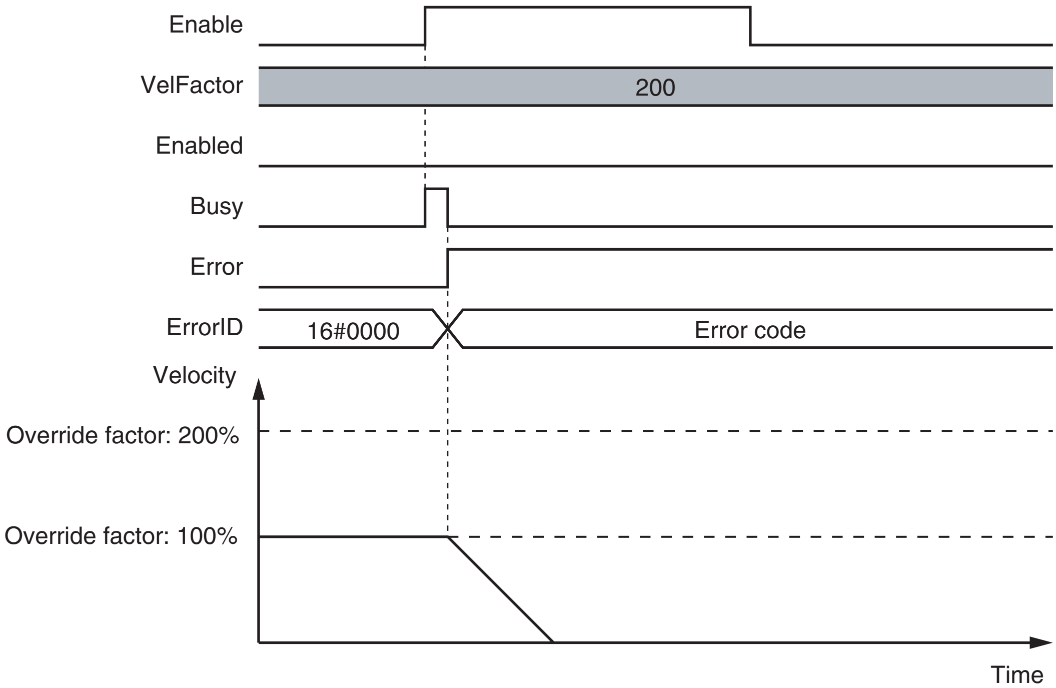
For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution of Other Instructions during Instruction Execution

If another instance of the MC_GroupSetOverride instruction is executed during MC_GroupSetOverride execution for the same axes group, the last MC_GroupSetOverride instance that is executed takes priority. *Enabled* will be TRUE for both instructions.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE. If an axes group minor fault occurs, the axes group will stop. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

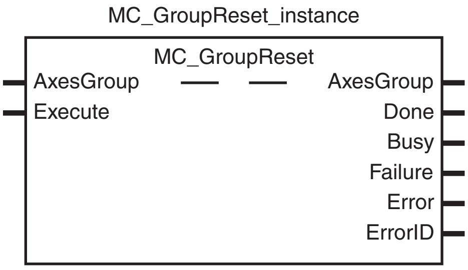



### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_GroupReset

The MC_GroupReset instruction clears axes group errors and axis errors.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_GroupReset | Group Reset | FB |  | MC_GroupReset_instance (<br>AxesGroup :=*parameter,*<br>Execute :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>Failure =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE.* |

* Specify an Axes Group Variable that was created in the Axes Group Basic Settings of the Sysmac Studio. (The default axes group variable names are *MC_Group****.)

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| Failure | Failure End | BOOL | TRUE or FALSE | TRUE when the instruction was not executed correctly. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When error clear processing is completed normally. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *Failure* changes to TRUE. |
| Failure | • When an instruction is executed while an axis or axes group is decelerating to a stop caused by an error.<br>• When an instruction is executed while there is an axes group error that is caused by an axis common error. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| AxesGroup | Axes Group | _sGROUP_REF | --- | Specify the axes group.* |

* Specify an Axes Group variable that was created in the Axes Group Basic Settings of the Sysmac Studio. (The default Axes Group variables are *MC_Group****.)

## Function

- When *Execute* changes to TRUE, the error of the axes group specified by *AxesGroup* in the *GroupEnable* state and axis errors of axes belonging to the axes group are cleared.
  The following are reset: minor faults or observations that occur for axes or axes groups and drive errors.
- Error clear processing is performed regardless of whether the Servo is ON or OFF for the axes.
- If there is a drive error for an axis, the drive error is cleared first. Error clear processing is then performed.
- Reset processing for the drive error is continued until the drive error is cleared or continues for the Drive Error Reset Monitoring Time in the axis parameters. The drive error reset process is executed for all axes belonging to the axes group at the same time.
- Only errors that existed when *Execute* changed to TRUE are reset. Errors that occur while clearing errors are not cleared.
- If this instruction is executed while the axes group is decelerating to a stop for an axes group error and the instruction is not executed, Failure will change to TRUE. This is so that the error cannot be reset before the axis stops. *Failure* will also change to TRUE if an axis error that results from an axis common error cannot be cleared by this instruction.

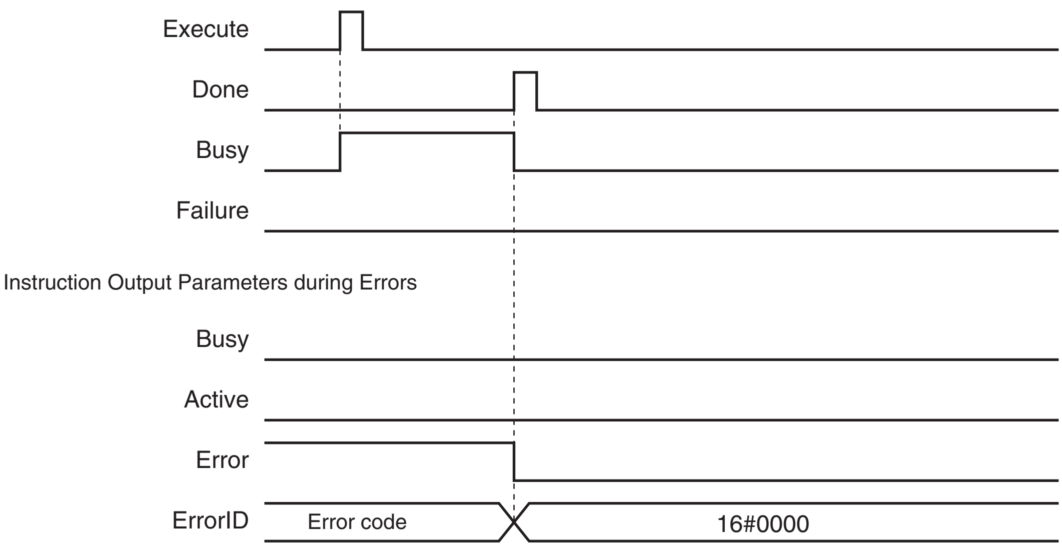
**Precautions for Correct Use**

- The error clear processing that is performed by this instruction sometimes requires more than one control period.
- The *Failure* output variable from the instruction will change to TRUE if the axis is in motion. Remove the cause of the error, and then retry the process until *Done* changes to TRUE.
- After you remove the cause of the error, execute the instruction only after you confirm that the axes have stopped completely.

**Additional Information**

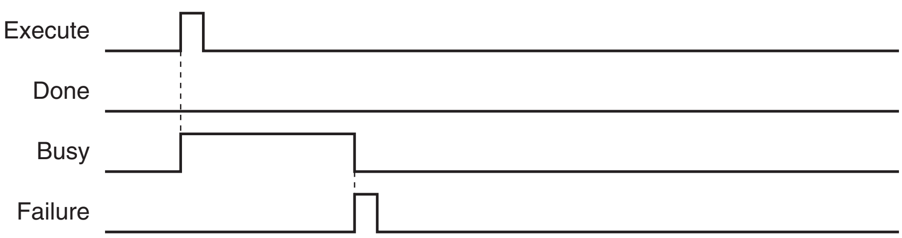
- You can clear axis errors only when the axes group is enabled.
- The following errors cannot be cleared with this instruction.
  All axis common errors
  To clear axis common errors, execute the ResetMcError (Reset All Errors) instruction separately.
- If you execute this instruction on an axes group for which there is no error, the instruction is completed normally and the current status is continued.

## Timing Charts



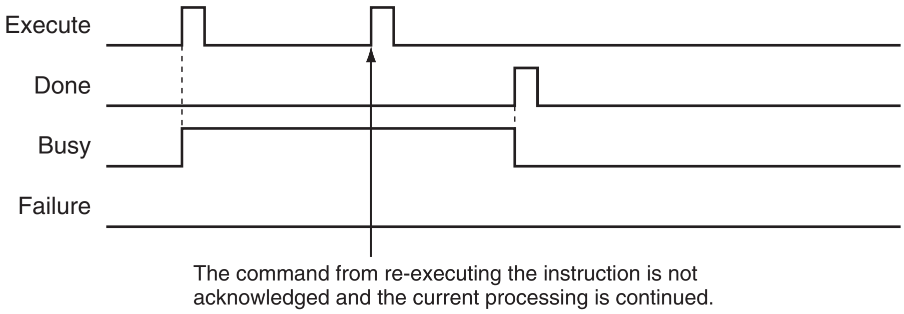

## Aborting the Instruction

The instruction is aborted if it is not possible to clear errors that occur when the axes group is decelerating to a stop for an error or errors that occur during axes group errors resulting from axis common errors.



## Re-execution of Motion Control Instructions

If the instruction is re-executed by changing *Execute* to TRUE again, the re-executed instruction is ignored and error clear processing is continued.

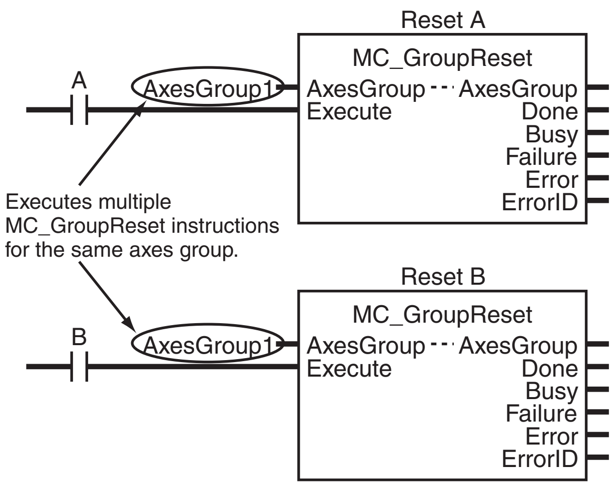



## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### ● Execution of Other Instructions during Instruction Execution

If another instance of the MC_GroupReset instruction is executed for the same axes group, both instructions are executed. If a slave error occurs, processing may wait until the Drive Error Reset Monitoring Time that is set for the axis expires. The elapsed time is counted for each instruction instance. If MC_Reset (Reset Axis Error) is executed for the axes belonging to the specified axes group while this instruction is in execution, both instructions are executed.




## Error Codes

Refer to *A-1 Error Codes* for instruction errors.

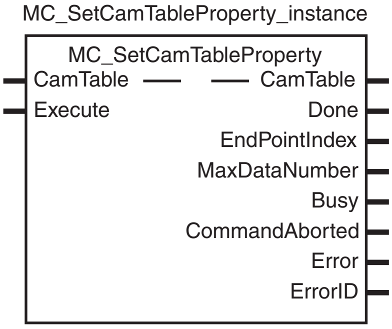
# 5

# Common Command Instructions

This section describes the instructions that are used for both axes and axes groups.

# MC_SetCamTableProperty

The MC_SetCamTableProperty instruction changes the end point index of the cam table that is specified in an in-out parameter.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_SetCamTableProperty | Setting Cam Table Properties | FB |  | MC_SetCamTableProperty_ instance (<br>CamTable :=*parameter,*<br>Execute :=*parameter,*<br>Done =>*parameter,*<br>EndPointIndex =>*parameter,*<br>MaxDataNumber =>*parameter,*<br>Busy =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| EndPointIndex | End Point Index | UDINT | Non-negative number | Contains the cam table end point index. |
| MaxDataNumber | Maximum Number of Cam Data | UDINT | Positive number | Contains the maximum number of cam data. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When overwriting cam table data is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | When there is a reason to abort this instruction. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| CamTable | Cam Table | ARRAY[0..N] OF _sMC_CAM_REF | --- | Specify the cam data structure _sMC_CAM_REF array variable as the cam table.* |

* *N* in the array variable is set automatically by the Sysmac Studio. Specify a cam data variable that was created on Cam Editor of the Sysmac Studio.

## Function

- The MC_SetCamTableProperty instruction changes the end point index of the cam table that is specified in an in-out parameter.
- The end point is the data located one cam data before the first cam data with a phase of 0 after the start point in the cam table.
- The array number of the end point is output to *EndPointIndex*.
- Any data that is detected after the 0 phase is detected is treated as invalid cam data, and the phase/displacement values are ignored.
- The maximum number of cam data is the value that was specified when the structure array is declared with the Sysmac Studio.
- When the user program changes the cam data end point index, the end point must be updated. Use this instruction to update the number of valid cam data.

For details on cam tables, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

**Precautions for Correct Use**

- When searching the cam table, an error will occur if the phases are not in ascending order before the 0 phase is found.
- You cannot change the maximum number of cam data from the user program.
- Execute this instruction after changing the end point in the can data or overwriting values in the cam data.
  If the end point index is not changed or the phases are not in ascending order, the cam operation and the operation of the *EndOfProfile* (End of Cam Cycle) of the MC_CamIn (Start Cam Operation) instruction may not be as expected.
- If the cam table is changed while this instruction is being processed, the update process will not be performed correctly. Wait for execution of this instruction to be completed before you change the cam data from the user program.
- Cam data variables are global variables. You can therefore access or change the values of cam data variables from more than one task. If you change the values of cam data variables from more than one task, program the changes so that there is no competition in writing the value from more than one task.
- If you use exclusive control of global variables between tasks for a cam data variable, do not use the cam data variable for motion control instructions while exclusive control is in effect for the cam data variable. An Incorrect Cam Table Specification error (error code: 5439 hex) will occur.

## Instruction Details

For example, refer to the following cam table. The *EndPointIndex* is 999 and the *MaxDataNumber* (Maximum Number of Cam Data) is 5,000.

| Cam data structure array | Phase | Displacement | | |
|---|---|---|---|---|
| MyCam1 [0] | 0 | 0 | Start point | Maximum number of data: 5,000 |
| . | . | . | Valid data | |
| . | . | . | | |
| . | . | . | | |
| MyCam1 [997] | 359.8 | 2 | | |
| MyCam1 [998] | 359.9 | 1 | | |
| MyCam1 [999] | 360.0 | 0 | End point | |
| MyCam1 [1000] | 0 | 0 | Invalid data | |
| . | . | . | | |
| . | . | . | | |
| . | . | . | | |
| MyCam1 [4999] | 0 | 0 | | |

The following tables show the relationship between overwriting of the cam data and the *EndPointIndex*.

If this instruction is executed with a cam table in which the phases for *MyCam1[1000]* on are 0, *EndPointIndex* will be 999.
If this instruction is executed for a cam table after the phase for *MyCam1[997]* is changed to 0, *EndPointIndex* will be 996.

| Cam data structure array | Phase | Displacement | | Phase | Displacement | |
|---|---|---|---|---|---|---|
| MyCam1 [0] | 0 | 0 | | 0 | 0 | |
| . | . | . | | . | . | |
| . | . | . | | . | . | |
| . | . | . | | . | . | |
| MyCam1 [995] | 359.6 | 5 | | 359.6 | 5 | |
| MyCam1 [996] | 359.7 | 4 | | 360.0 | 0 | End point |
| MyCam1 [997] | 359.8 | 2 | | 0 | 0 | Invalid data |
| MyCam1 [998] | 359.9 | 1 | | 359.9 | 1 | |
| MyCam1 [999] | 360.0 | 0 | End point | 360.0 | 0 | |
| MyCam1 [1000] | 0 | 0 | | 0 | 0 | |
| . | . | . | | . | . | |
| . | . | . | | . | . | |
| . | . | . | | . | . | |
| MyCam1 [4999] | 0 | 0 | | 0 | 0 | |

If this instruction is executed for a cam table after the phases for *MyCam1[1000]* to *MyCam1[4997]* are changed anything other than 0, *EndPointIndex* will be 4997.

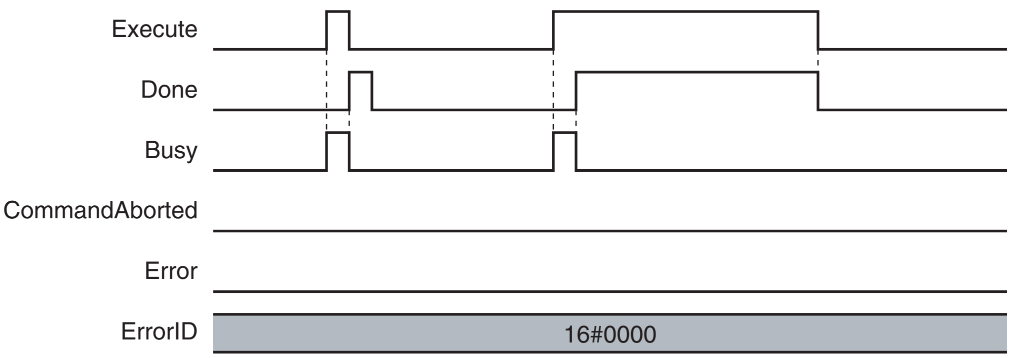
| Cam data structure array | Phase | Displacement | | Phase | Displacement | |
|---|---|---|---|---|---|---|
| MyCam1 [0] | 0 | 0 | | 0 | 0 | |
| . | . | . | | . | . | |
| . | . | . | | . | . | |
| . | . | . | | . | . | |
| MyCam1 [998] | 359.9 | 1 | | 100.3 | 20.3 | |
| MyCam1 [999] | 360.0 | 0 | End point | 100.4 | 20.4 | |
| MyCam1 [1000] | 0 | 0 | | 100.5 | 20.5 | |
| . | . | . | | . | . | |
| . | . | . | | . | . | |
| . | . | . | | . | . | |
| MyCam1 [4996] | 0 | 0 | Invalid data | 359.99 | 0.01 | |
| MyCam1 [4997] | 0 | 0 | | 360.00 | 0.0 | End point |
| MyCam1 [4998] | 0 | 0 | | 0 | 0 | Invalid data |
| MyCam1 [4999] | 0 | 0 | | 0 | 0 | |

If this instruction is not executed even though the phases for *MyCam1[1000]* to *MyCam1[4997]* were overwritten to values other than 0, cam operation will be executed between *MyCam1[0]* to *MyCam1[999]*. The cam table is overwritten, but the *EndPointIndex* does not change.

For information on the cam table data structure, refer to *MC_CamIn* on page 3-156.

## Timing Charts

The following chart shows two ways to execute the instruction. A normal end is indicated for either method.
The first time, *Execute* is changed to TRUE and then it is changed to FALSE before execution of the instruction is completed.
The second time, the value of *Execute* is held.




## Re-execution of Motion Control Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Control Instructions

### Execution during Execution of Other Instructions

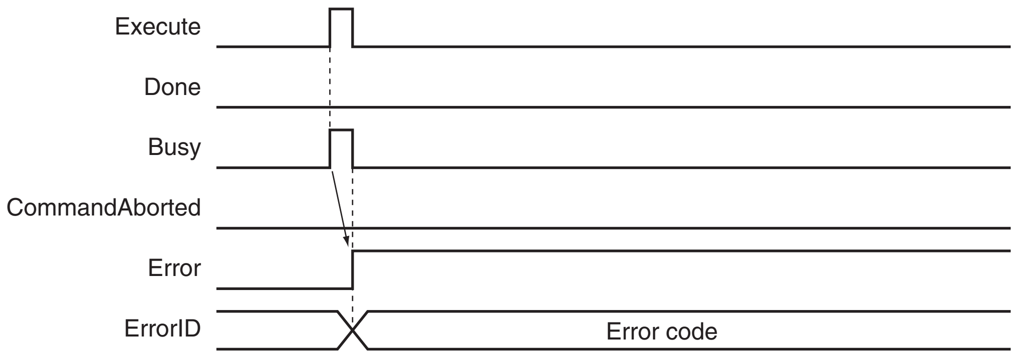
Multi-execution of instructions cannot be used for this instruction if the cam table specified by *CamTable* is used by another instruction such as MC_SaveCamTable or MC_SetCamTableProperty.

### Execution of Other Instructions during Instruction Execution

Multi-execution of instructions cannot be used for other instructions such as MC_SaveCamTable or MC_SetCamTableProperty if the same cam table is specified with *CamTable* for this instruction.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE. You can find out the cause of the error by referring to the value output by *ErrorID*.

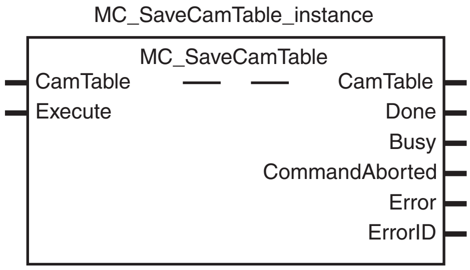



### Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_SaveCamTable

The MC_SaveCamTable instruction saves the cam table specified with the input parameter to non-volatile memory.

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_SaveCamTable | Save Cam Table | FB |  | MC_SaveCamTable_instance (<br>CamTable :=*parameter,*<br>Execute :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |

### Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

#### ● Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When saving cam table data is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| Command-Aborted | When there is a reason to abort this instruction. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| CamTable | Cam Table | ARRAY[0..N] OF _sMC_CAM_REF | --- | Specify the start of a cam data structure _sMC_CAM_REF array variable as the cam table.* |

* *N* in the array variable is set automatically by the Sysmac Studio. Specify a cam data variable that was created on Cam Editor of the Sysmac Studio.

# Function

- The MC_SaveCamTable instruction saves the cam table specified with the in-out variable to non-volatile memory.
- *_MC_COM.Status.CamTableBusy* (Cam Table File Save Busy) in the system-defined variables for motion control is TRUE while the cam table is saved.

For information on the cam table data, refer to *MC_CamIn* on page 3-156.

### Precautions for Correct Use

- Use this instruction to save the cam data after it is overwritten before you turn OFF the Controller. If you turn OFF the Controller without saving the cam data, the overwritten data will be lost.
- If the cam table is changed while this instruction is being processed, the update process will not be performed correctly. Do not write to the cam table while this instruction is being processed when changing the cam table from the user program.
- This instruction has a considerably longer processing time compared with other instructions. The processing time is greatly affected by the processing load on the CPU Unit. If the next instruction is executed taking the completion of this instruction as a trigger, take care with the timing of execution of the next instruction.
- Do not turn OFF the Controller while this instruction is being processed. The data is not saved correctly if the Controller is turned OFF. The cam data in non-volatile memory may become corrupted.
- You cannot upload cam data, download cam data, start online operation, perform online editing, or start data traces during execution of this instruction. If this instruction is executed during a cam data upload, cam data download, or online editing, a Cannot Execute Save Cam Table Instruction error (error code 743C hex) occurs. Perform retry processing until the cam data is saved normally. Use the Synchronize Menu of the Sysmac Studio to upload and download the data.
- There is a limit to the number of times that you can write non-volatile memory. The life of the non-volatile memory will expire faster if this instruction is executed frequently. Do not execute it any more than is necessary.
- Cam data variables are global variables. You can therefore access or change the values of cam data variables from more than one task. If you change the values of cam data variables from more than one task, program the changes so that there is no competition in writing the value from more than one task.
- If you use exclusive control of global variables between tasks for a cam data variable, do not use the cam data variable for motion control instructions while exclusive control is in effect for the cam data variable. An Incorrect Cam Table Specification error (error code: 5439 hex) will occur.

### Relation to CPU Unit Operating Modes

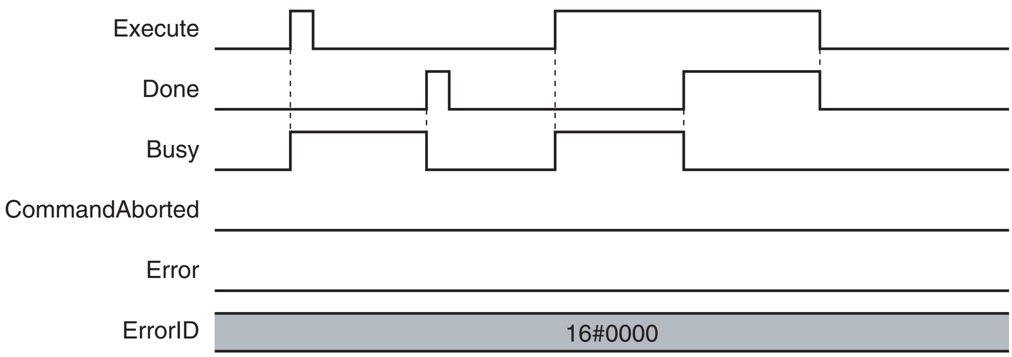
Cam data save processing for this instruction continues even if the operating mode of the CPU Unit changes to PROGRAM mode.
To see if cam data save processing is in progress in PROGRAM mode, place the Sysmac Studio online and monitor the _MC_COM.Status.CamTableBusy system-defined variable for motion control.

### Deleting Instruction with Online Editing

Cam data save processing for this instruction continues even if this instruction is deleted with online editing.

## Timing Charts

The following chart shows two ways to execute the instruction. A normal end is indicated for either method.
The first time, *Execute* is changed to TRUE and then it is changed to FALSE before execution of the instruction is completed.
The second time, the value of *Execute* is held.




## Re-execution of Motion Control Instructions

This instruction cannot be re-executed.
A Motion Control Instruction Re-execution Disabled error (error code: 543B hex) occurs if re-execution is attempted.

## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### Execution during Execution of Other Instructions

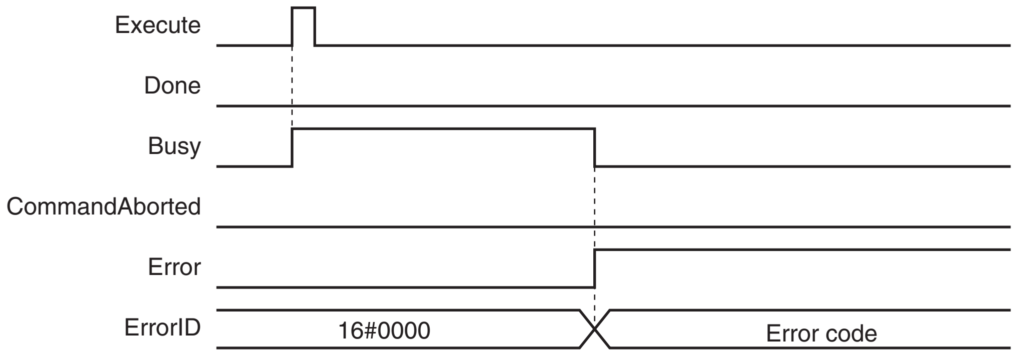
Multi-execution of instructions cannot be used for this instruction if the cam table specified by *CamTable* is used by another instruction, such as MC_SetCamTableProperty.
This instruction also cannot be executed during execution of another instance of the instruction.

### Execution of Other Instructions during Instruction Execution

Multi-execution of instructions cannot be used for other instructions, such as MC_SetCamTableProperty, if the same cam table is specified with *CamTable* for this instruction.
This instruction also cannot be executed during execution of another instance of this instruction.

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE. You can find out the cause of the error by referring to the value output by *ErrorID*.




### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

# MC_Write

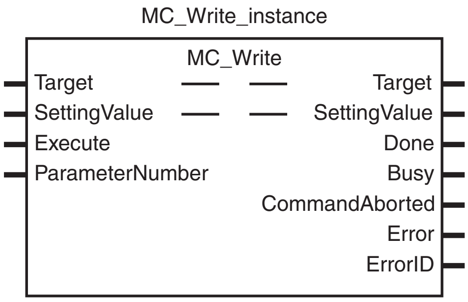
The MC_Write instruction writes parts of the motion control parameters in the MC Function Module.

**Precautions for Correct Use**

The values that are written by this instruction are not saved in non-volatile memory. Any values that are written are lost when the power supply to the Controller is turned OFF, when settings are downloaded, or when the MC Function Module is restarted. They return to the values that were set from the Sysmac Studio.

**Additional Information**

Use the Synchronize Menu of the Sysmac Studio to download the project.

| Instruction | Name | FB/FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| MC_Write | Write MC Setting | FB |  | MC_Write_instance (<br>Target :=*parameter,*<br>SettingValue :=*parameter,*<br>Execute :=*parameter,*<br>ParameterNumber :=*parameter,*<br>Done =>*parameter,*<br>Busy =>*parameter,*<br>CommandAborted =>*parameter,*<br>Error =>*parameter,*<br>ErrorID =>*parameter*<br>); |

## Variables

### Input Variables

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |
| Parameter Number | Parameter Number | _eMC_ PARAMETER_ NUMBER | 0: _mcChkVel<br>1: _mcChkAcc<br>2: _mcChkDec<br>3: _mcPosiChkTrq<br>4: _mcNegaChkTrq<br>5: _mcFELmt<br>6: _mcChkFELmt<br>7: _mcSwLmtMode<br>8: _mcPosiSwLmt<br>9: _mcNegaSwLmt<br>10: _mcInPosTime | 0* | Specify the parameter to write.<br>0: Velocity Warning Value<br>1: Acceleration Warning Value<br>2: Deceleration Warning Value<br>3: Positive Torque Warning Value<br>4: Negative Torque Warning Value<br>5: Following Error Over Limit Value<br>6: Following Error Warning Value<br>7: Software Limits<br>8: Positive Software Limit<br>9: Negative Software Limit<br>10: In-position Check Time |

* The default value for an enumeration variable is actually not the number, but the enumerator.

### Parameter Data Types and Valid Ranges

| Parameter | | Data type | Valid range | Comments |
|---|---|---|---|---|
| 0 | Velocity Warning Value | UINT | 0 to 100 | The unit is %. |
| 1 | Acceleration Warning Value | UINT | 0 to 100 | The unit is %. |
| 2 | Deceleration Warning Value | UINT | 0 to 100 | The unit is %. |
| 3 | Positive Torque Warning Value | UINT | 0 to 1000 | The unit is %. |
| 4 | Negative Torque Warning Value | UINT | 0 to 1000 | The unit is %. |
| 5 | Following Error Limit Value | LREAL | Positive number | The unit is command units.* |
| 6 | Following Error Warning Value | LREAL | Positive number | Set a value that is less than the value of the Following Error Limit Value.<br>The unit is command units.* |
| 7 | Software Limits | Enumeration | 0: _mcNonSwLmt<br>1: _mcCmdDecelerationStop<br>2: _mcCmdImmediateStop<br>3: _mcActDecelerationStop<br>4: _mcActImmediateStop | 0: Disabled<br>1: Deceleration stopping enabled for command position<br>2: Immediate stopping enabled for command position (stop using remaining pulses)<br>3: Deceleration stopping enabled for actual position<br>4: Immediate stopping enabled for actual position (stop using remaining pulses) |
| 8 | Positive Software Limit | LREAL | Negative number, positive number, or 0 | The unit is command units.* |
| 9 | Negative Software Limit | LREAL | Negative number, positive number, or 0 | The unit is command units.* |
| 10 | In-position Check Time | UINT | 0 to 10,000 | The unit is milliseconds. |

* Refer to Unit Conversion Settings in the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for information on command units.

## Output Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Done | Done | BOOL | TRUE or FALSE | TRUE when the instruction is completed. |
| Busy | Executing | BOOL | TRUE or FALSE | TRUE when the instruction is acknowledged. |
| CommandAborted | Command Aborted | BOOL | TRUE or FALSE | TRUE when the instruction is aborted. |
| Error | Error | BOOL | TRUE or FALSE | TRUE while there is an error. |
| ErrorID | Error Code | WORD | * | Contains the error code when an error occurs. A value of 16#0000 indicates normal execution. |

* Refer to *A-1 Error Codes*.

### Output Variable Update Timing

| Name | Timing for changing to TRUE | Timing for changing to FALSE |
|---|---|---|
| Done | When the instruction is completed. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Busy | When *Execute* changes to TRUE. | • When *Done* changes to TRUE.<br>• When *Error* changes to TRUE.<br>• When *CommandAborted* changes to TRUE. |
| CommandAborted | When another instruction causes an error and aborts this instruction. | • When *Execute* is TRUE and changes to FALSE.<br>• After one period when *Execute* is FALSE. |
| Error | When there is an error in the execution conditions or input parameters for the instruction. | When the error is cleared. |

## In-Out Variables

| Name | Meaning | Data type | Valid range | Description |
|---|---|---|---|---|
| Target | Write Target | _sAXIS_REF or _sGROUP_REF | --- | Specify the axis or axes group for which to write a parameter.*1 |
| SettingValue | Setting Value | Depends on the data type of the specified variable.*2 | --- | Specify the value to write.<br>The valid range follows the motion control parameter that is specified by ParameterNumber.<br>Default: 0 |

*1 Specify an Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio or Axes Group Variable that was created in the Axes Group Basic Settings of the Sysmac Studio. (The default axis variable names are MC_Axis***. The default axes group variable names are MC_Group***.)

*2 For details on the data types of variables, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

### In-Out Variable Update Timing

| Name | Write Timing |
|---|---|
| SettingValue | When *Done* changes to TRUE. |

## Function

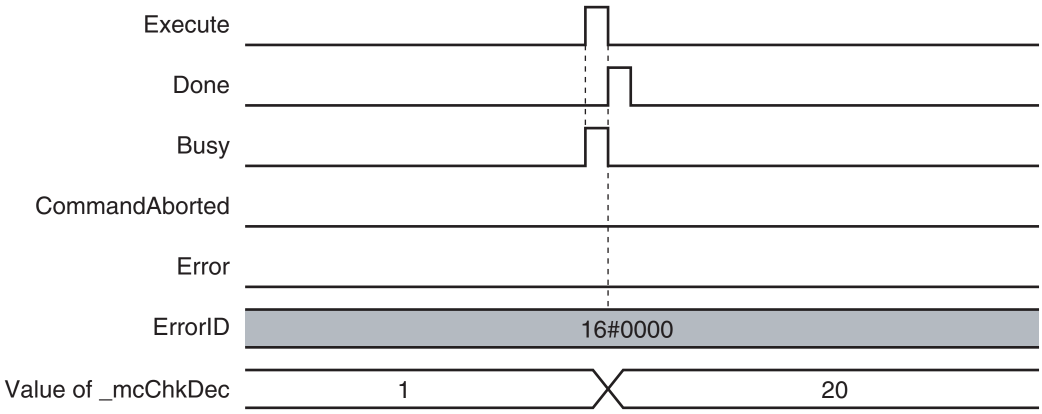
- The MC_Write instruction writes the *SettingValue* to the system-defined variable for motion control specified by Target (Write Target) and *ParameterNumber* when *Execute* changes to TRUE.
- The parameters that are specified with the input variables are used if motion starts when *Status.Standby* in the Axes Group Variable or *Status.Standstill* in the Axis Variable is TRUE, and for multi-execution of instructions when the Buffer Mode is set to *Aborting*.
  Therefore these parameters are not applied when operation is continued after restarting or for multi-execution of instructions with Buffer Mode set to any value other than *Aborting*.

**Precautions for Correct Use**

Values changed by this instruction are not saved in the non-volatile memory.
When the power supply to the Controller is turned OFF, the value that was set for the parameter is lost. Use the Sysmac Studio and transfer the parameters to save them to non-volatile memory.

## Timing Charts

The following timing chart shows the operation for when 20 is written to *_mcChkDec* (Deceleration Warning Value) in the axis parameter settings.



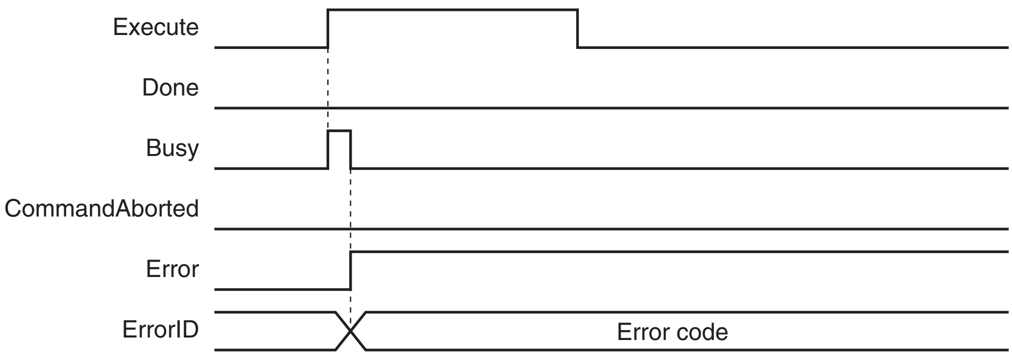

## Multi-execution of Motion Control Instructions

For details on multi-execution of motion control instructions, refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507).

## Errors

If an error occurs during instruction execution, *Error* will change to TRUE and parameters are not written. The value before the instruction was executed will be held. You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).

### ● Timing Chart When Error Occurs




### ● Error Codes

Refer to *A-1 Error Codes* for instruction errors.

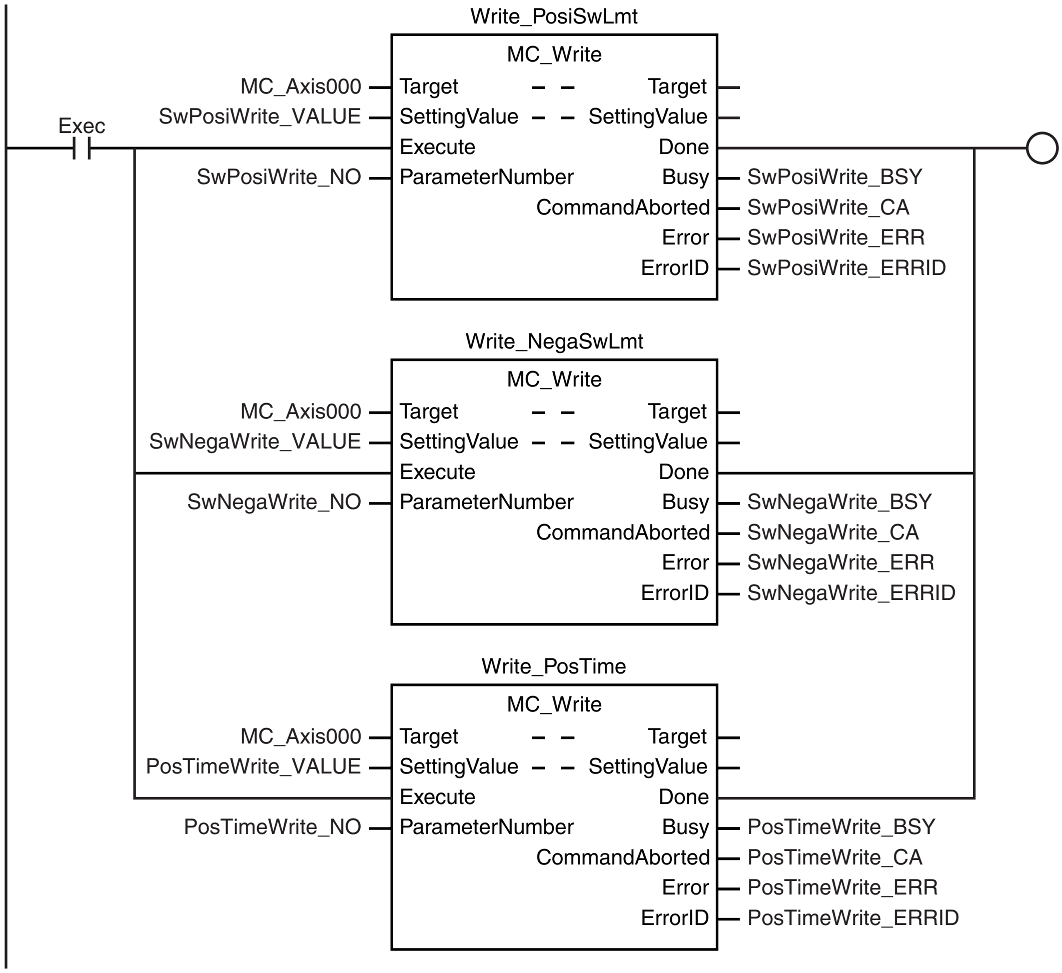
# Sample Programming

The following sample programming writes the Positive Software Limit Value, Negative Software Limit Value, and In-position Check Time when *Exec* changes to TRUE.

## Main Variables

| Name | Data type | Description |
|---|---|---|
| MC_Axis000 | _sAXIS_REF | Axis Variable for axis 1 |
| SwPosiWrite_VALUE | LREAL | Positive Software Limit Value |
| SwNegaWrite_VALUE | LREAL | Negative Software Limit Value |
| PosTimeWrite_VALUE | UINT | In-position Check Time |
| SwPosiWrite_NO | _eMC_PARAMETER_NUMBER | Positive Software Limit Specification |
| SwNegaWrite_NO | _eMC_PARAMETER_NUMBER | Negative Software Limit Specification |
| PosTimeWrite_NO | _eMC_PARAMETER_NUMBER | In-position Check Time Specification |

## Ladder Diagram



App

# Appendices

The appendices describe the error codes that are generated by the instructions.

# A-1 Error Codes

The following table lists the error codes that are output to *ErrorID* when errors occur in execution of the instructions.

The upper four digits of the event codes that are given in the following table are output as the error code to *ErrorID*.

Refer to *A-2 Error Code Details* for details on the error codes.

Maj: Major fault level

Prt: Partial fault level

Min: Minor fault level

Obs: Observation

Info: Information

Refer to the *NJ-series Troubleshooting Manual* (Cat. No. W503) for all of the event codes that may occur in an NJ-series Controller.

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 34610000 hex | Process Data Object Setting Missing | The PDO mapping is not correct. | • The PDOs that are required for the motion control instruction are not mapped.<br>• A motion control instruction that specifies phase Z (_mcEncoderMark) as the trigger conditions was executed for an axis that is mapped to an OMRON GX-EC02□□ EtherCAT Encoder slave. | | | √ | | | page A-20 |
| 54200000 hex | Electronic Gear Ratio Numerator Setting Out of Range | The parameter specified for the *RatioNumerator* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-20 |
| 54210000 hex | Electronic Gear Ratio Denominator Setting Out of Range | The parameter specified for the *RatioDenominator* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-21 |
| 54220000 hex | Target Velocity Setting Out of Range | The parameter specified for the *Velocity* input variable to a motion control instruction is out of range. | • The Target Velocity (input variable *Velocity*) is still at the default (0). | | | √ | | | page A-21 |
| 54230000 hex | Acceleration Setting Out of Range | The parameter specified for the *Acceleration* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-22 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 54240000 hex | Deceleration Setting Out of Range | The parameter specified for the *Deceleration* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-22 |
| 54250000 hex | Jerk Setting Out of Range | The parameter specified for the *Jerk* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-23 |
| 54270000 hex | Torque Ramp Setting Out of Range | The parameter specified for the *TorqueRamp* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-23 |
| 54280000 hex | Master Coefficient Scaling Out of Range | The parameter specified for the *MasterScaling* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-24 |
| 54290000 hex | Slave Coefficient Scaling Out of Range | The parameter specified for the *SlaveScaling* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-24 |
| 542A0000 hex | Feeding Velocity Setting Out of Range | The parameter specified for the *FeedVelocity* input variable to a motion control instruction is out of range. | • The Feed Velocity (input variable *FeedVelocity*) is still at the default (0). | | | √ | | | page A-24 |
| 542B0000 hex | Buffer Mode Selection Out of Range | The parameter specified for the *BufferMode* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-25 |
| 542C0000 hex | Coordinate System Selection Out of Range | The parameter specified for the *CoordSystem* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-25 |
| 542D0000 hex | Circular Interpolation Mode Selection Out of Range | The parameter specified for the *CircMode* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-26 |
| 542E0000 hex | Direction Selection Out of Range | The parameter specified for the *Direction* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-26 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 542F0000 hex | Path Selection Out of Range | The parameter specified for the *PathChoice* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-26 |
| 54300000 hex | Position Type Selection Out of Range | The parameter specified for the *ReferenceType* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-27 |
| 54310000 hex | Travel Mode Selection Out of Range | The parameter specified for the *MoveMode* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-27 |
| 54320000 hex | Transition Mode Selection Out of Range | The parameter specified for the *TransitionMode* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable.<br>• *_mcAborting* or *_mcBuffered* was specified for *BufferMode* and *_mcTMCornerSuperimpose* was specified for *TransitionMode.* | | | √ | | | page A-28 |
| 54330000 hex | Continue Method Selection Out of Range | The value of the reserved input variable *Continuous* to a motion control instruction changed. | • The value of the reserved input variable *Continuous* changed. | | | √ | | | page A-28 |
| 54340000 hex | Combine Mode Selection Out of Range | The parameter specified for the *CombineMode* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-29 |
| 54350000 hex | Synchronization Start Condition Selection Out of Range | The parameter specified for the *LinkOption* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-29 |
| 54360000 hex | Master and Slave Defined as Same Axis | The same axis is specified for the *Master* and *Slave* input variables to a motion control instruction. | • The parameter is the same for the *Master* and *Slave* input variables to the instruction. | | | √ | | | page A-29 |
| 54370000 hex | Master and Auxiliary Defined as Same Axis | The same axis is specified for the *Master* and *Auxiliary* input variables to a motion control instruction. | • The parameter is the same for the *Master* and *Auxiliary* input variables to the instruction. | | | √ | | | page A-30 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 54380000 hex | Master/Slave Axis Numbers Not in Ascending Order | The axis numbers specified for the *Master* and *Slave* input variables to a motion control instruction are not in ascending order. | • The parameters for the *Master* and *Slave* input variables to the instruction were not in ascending order when *_mcLatestCommand* was specified for the *ReferenceType* input variable to the instruction. | | | √ | | | page A-30 |
| 54390000 hex | Incorrect Cam Table Specification | The parameter specified for the *CamTable* input variable to a motion control instruction is out of range. | • Something other than a cam data variable was specified for the *CamTable* input variable to the instruction. | | | √ | | | page A-31 |
| 543A0000 hex | Synchronization Stopped | A synchronized control motion control instruction was executed, but conditions required for execution were not met. | • The MC_CamOut (End Cam Operation) instruction was executed even though the MC_CamIn (Start Cam Operation) instruction is not being executed.<br>• The MC_GearOut (End Gear Operation) instruction was executed even though the MC_GearIn (Start Gear Operation) or the MC_GearInPos (Positioning Gear Operation) instruction is not being executed.<br>• The MC_Phasing (Shift Master Axis Phase) instruction was executed even though the MC_CamIn (Start Cam Operation), MC_GearIn (Start Gear Operation), MC_GearInPos (Start Gear Operation), or MC_MoveLink (Synchronous Positioning) instruction is not being executed. | | | √ | | | page A-31 |
| 543B0000 hex | Motion Control Instruction Re-execution Disabled | An attempt was made to re-execute a motion control instruction that cannot be re-executed. | • A motion control instruction that cannot be re-executed was re-executed. | | | √ | | | page A-32 |
| 543C0000 hex | Motion Control Instruction Multi-execution Disabled | Multiple functions that cannot be executed simultaneously were executed for the same target (MC common or axis). | • Multiple functions that cannot be executed simultaneously were executed for the same target (MC common or axis). | | | √ | | | page A-32 |
| 543D0000 hex | Instruction Not Allowed for Encoder Axis Type | An operation instruction was executed for an encoder axis. | • An operation instruction was executed for an encoder axis. | | | √ | | | page A-33 |
| 543E0000 hex | Instruction Cannot Be Executed during Multi-axes Coordinated Control | An operation instruction was executed for an axis that was in a coordinated multi-axes motion. | • An operation instruction was executed for an axis that was in a multi-axes coordinated motion. | | | √ | | | page A-33 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 543F0000 hex | Multi-axes Coordinated Control Instruction Executed for Disabled Axes Group | A multi-axes coordinated control instruction was executed for an axes group that was in the Axes Group Disabled state. | • A multi-axes coordinated control instruction was executed for an axes group that was in the Axes Group Disabled state. | | | √ | | | page A-34 |
| 54400000 hex | Axes Group Cannot Be Enabled | Execution of the MC_GroupEnable (Enable Axes Group) instruction failed. | • When the MC_GroupEnable (Enable Axes Group) instruction was executed, there was a composition axis that was not stopped.<br>• When the MC_GroupEnable (Enable Axes Group) instruction was executed, there was a composition axis for which the MC_TouchProbe (Enable External Latch) instruction was being executed. | | | √ | | | page A-34 |
| 54410000 hex | Impossible Axis Operation Specified when the Servo is OFF | An operation instruction was executed for an axis for which the Servo is OFF. | • An operation instruction was executed for an axis for which the Servo is OFF.<br>• Home was preset with the MC_Home instruction for an axis for which EtherCAT process data communications are not established. | | | √ | | | page A-35 |
| 54420000 hex | Composition Axis Stopped Error | A motion instruction was executed for an axes group while the MC_Stop instruction was being executed for a composition axis. | • A motion instruction was executed for an axes group while the MC_Stop instruction was being executed for a composition axis. | | | √ | | | page A-35 |
| 54430000 hex | Motion Control Instruction Multi-execution Buffer Limit Exceeded | The number of motion control instructions that is buffered for Buffered or Blending Buffer Modes exceeded the buffer limit. | • An axis instruction was executed when there was already a current instruction and a buffered instruction for the same axis.<br>• An axes group instruction was executed when there was already eight current instructions and buffered instructions for the same axis. | | | √ | | | page A-36 |
| 54440000 hex | Insufficient Travel Distance | The specified motion cannot be executed for the deceleration rate or acceleration rate that was specified for multi-execution or re-execution of a positioning instruction. | • Stopping at the target position was not possible for the specified acceleration/deceleration rate for multi-execution or re-execution of a positioning instruction when the Acceleration/Deceleration Over parameter was set to generate a minor fault and stop. | | | √ | | | page A-37 |
| 54450000 hex | Insufficient Travel Distance to Achieve Blending Transit Velocity | There is not sufficient travel distance to accelerate or decelerate to the transit velocity. | • There was not sufficient travel distance to accelerate the current command to the transit velocity when the Acceleration/Deceleration Over parameter was set to generate a minor fault and stop. | | | √ | | | page A-37 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 54460000 hex | Move Link Constant Velocity Insufficient Travel Distance | The constant-velocity travel distance of the master axis is less than zero. | • The constant velocity travel distance of the master axis is below 0 for the MC_MoveLink (Synchronous Positioning) instruction. | | | √ | | | page A-38 |
| 54470000 hex | Positioning Gear Operation Insufficient Target Velocity | For the MC_GearInPos (Positioning Gear Operation) instruction, the target velocity of the slave axis is too small to achieve the required velocity. | • For the MC_GearInPos (Positioning Gear Operation) instruction, the value of the *Velocity* (Target Velocity) input variable is smaller than the master axis velocity multiplied by the gear ratio when the instruction was executed. | | | √ | | | page A-38 |
| 54480000 hex | Same Start Point and End Point for Circular Interpolation | The start point and end point were the same when the radius method was specified for the MC_MoveCircular2D (Circular 2D Interpolation) instruction. Or, the start point, end point, and border point were the same when the border point method was specified. | • The start point and end point were the same when the radius method was specified for the MC_MoveCircular2D (Circular 2D Interpolation) instruction.<br>• The start point, end point, and border point were the same when the border point method was specified for the MC_MoveCircular2D (Circular 2D Interpolation) instruction. | | | √ | | | page A-39 |
| 54490000 hex | Circular Interpolation Center Specification Position Out of Range | The position specified for the center point exceeded the allowed range when the center method was specified for the MC_MoveCircular2D (Circular 2D Interpolation) instruction. | • The difference between the distance from the start point to the center point and the distance between the end point to the center point exceeded the permitted value specified for the correction allowance ratio in the axes group settings when the center designation method was specified for the MC_MoveCircular2D (Circular 2D Interpolation) instruction. | | | √ | | | page A-39 |
| 544A0000 hex | Circular Interpolation Cannot Be Executed with Rotary (Infinite) Axis | The MC_MoveCircular2D (Circular 2D Interpolation) instruction was executed for an axis for which the Count Mode was set to Rotary Mode. | • An axis in Rotary Mode was used with the MC_MoveCircular2D (Circular 2D Interpolation) instruction. | | | √ | | | page A-40 |
| 544C0000 hex | Parameter Selection Out of Range | The parameter specified for the *ParameterNumber* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-40 |
| 544D0000 hex | Stop Method Selection Out of Range | The parameter specified for the *StopMode* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-40 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 544E0000 hex | Latch ID Selection Out of Range for Trigger Input Condition | The parameter specified for the *TriggerInput::LatchID* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-41 |
| 544F0000 hex | Setting Out of Range for Writing MC Setting | The parameter specified for the *SettingValue* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable.<br>• The parameter specification and the data type of the setting value do not agree. | | | √ | | | page A-41 |
| 54500000 hex | Trigger Input Condition Mode Selection Out of Range | The parameter specified for the *TriggerInput:: Mode* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-42 |
| 54510000 hex | Drive Trigger Signal Selection Out of Range for Trigger Input Condition | The parameter specified for the *TriggerInput::InputDrive* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-42 |
| 54530000 hex | Motion Control Instruction Re-execution Disabled (Axis Specification) | An attempt was made to change the parameter for the *Axis* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-43 |
| 54540000 hex | Motion Control Instruction Re-execution Disabled (Buffer Mode Selection) | An attempt was made to change the parameter for the *BufferMode* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-43 |
| 54550000 hex | Motion Control Instruction Re-execution Disabled (Direction Selection) | An attempt was made to change the parameter for the *Direction* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • An input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-44 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 54560000 hex | Motion Control Instruction Re-execution Disabled (Execution Mode) | An attempt was made to change the parameter for the *ExecutionMode* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-44 |
| 54570000 hex | Motion Control Instruction Re-execution Disabled (Axes Group Specification) | An attempt was made to change the parameter for the *AxesGroup* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-45 |
| 54580000 hex | Motion Control Instruction Re-execution Disabled (Jerk Setting) | An attempt was made to change the parameter for the *Jerk* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-45 |
| 54590000 hex | Motion Control Instruction Re-execution Disabled (Master Axis) | An attempt was made to change the parameter for the *Master* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-46 |
| 545A0000 hex | Motion Control Instruction Re-execution Disabled (MasterOffset) | An attempt was made to change the parameter for the *MasterOffset* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-46 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 545B0000 hex | Motion Control Instruction Re-execution Disabled (MasterScaling) | An attempt was made to change the parameter for the *MasterScaling* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-47 |
| 545C0000 hex | Motion Control Instruction Re-execution Disabled (MasterStartDistance) | An attempt was made to change the parameter for the *MasterStartDistance* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-47 |
| 545D0000 hex | Motion Control Instruction Re-execution Disabled (Continuous) | An attempt was made to change the parameter for the *Continuous* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-48 |
| 545E0000 hex | Motion Control Instruction Re-execution Disabled (MoveMode) | An attempt was made to change the parameter for the *MoveMode* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-48 |
| 545F0000 hex | Illegal Auxiliary Axis Specification | The axis specified for the *Auxiliary* input variable to a motion control instruction does not exist. | • An axis does not exist for the variable specified for the *Auxiliary* input variable to the instruction. | | | √ | | | page A-49 |
| 54600000 hex | Illegal Axis Specification | The axis specified for the *Axis* input variable to a motion control instruction does not exist. | • An axis does not exist for the variable specified for the *Axis* input variable to the instruction. | | | √ | | | page A-49 |
| 54610000 hex | Illegal Axes Group Specification | The axes group specified for the *AxesGroup* input variable to a motion control instruction does not exist or is not a used group. | • An axes group does not exist for the variable specified for the *AxesGroup* input variable to the instruction.<br>• The axes group specified for the *AxesGroup* input variable to the instruction is not specified as a used group. | | | √ | | | page A-50 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 54620000 hex | Illegal Master Axis Specification | The axis specified for the *Master* input variable to a motion control instruction does not exist or is not a sync master axis. | • An axis does not exist for the variable specified for the *Master* input variable to the instruction.<br>• The axis that was specified for the *Master* input variable to the *MC_Phasing* (Shift Master Axis Phase) instruction is not the master axis for syncing. | | | √ | | | page A-50 |
| 54630000 hex | Motion Control Instruction Re-execution Disabled (SlaveOffset) | An attempt was made to change the *SlaveOffset* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-51 |
| 54640000 hex | Motion Control Instruction Re-execution Disabled (SlaveScaling) | An attempt was made to change the *SlaveScaling* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-51 |
| 54650000 hex | Motion Control Instruction Re-execution Disabled (StartPosition) | An attempt was made to change the *StartPosition* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-52 |
| 54660000 hex | Instruction Execution Error with Undefined Home | High-speed homing or an interpolation instruction was executed when home was undefined. | • High-speed homing was executed when home was undefined.<br>• An interpolation instruction was executed for an axes group that includes an axis with no defined home. | | | √ | | | page A-52 |
| 54670000 hex | Motion Control Instruction Re-execution Disabled (Position Type) | An attempt was made to change the *ReferenceType* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-53 |
| 54680000 hex | Unused Axis Specification for Master Axis | The master axis specified for a motion control instruction is an unused axis. | • The master axis specified for a motion control instruction is an unused axis. | | | √ | | | page A-53 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 54690000 hex | First Position Setting Out of Range | The parameter specified for the *FirstPosition* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-54 |
| 546A0000 hex | Last Position Setting Out of Range | The parameter specified for the *LastPosition* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-54 |
| 546B0000 hex | Illegal First/Last Position Size Relationship (Linear Mode) | The parameter specified for the *LastPosition* input variable to a motion control instruction is smaller than the parameter specified for the *FirstPosition* input variable. | • The value of the *LastPosition* input parameter is less than the value of the *FirstPosition* input variable for the instruction when the Count Mode is set to Linear Mode. | | | √ | | | page A-55 |
| 546C0000 hex | Master Sync Start Position Setting Out of Range | The parameter specified for the *MasterSyncPosition* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-55 |
| 546D0000 hex | Slave Sync Start Position Setting Out of Range | The parameter specified for the *SlaveSyncPosition* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-56 |
| 546E0000 hex | Duplicate Latch ID for Trigger Input Condition | The same latch ID was specified for more than one motion control instruction. | • The same latch ID is used simultaneously for more than one of the following instructions: MC_TouchProbe (Enable External Latch) instruction, MC_MoveLink (Synchronous Positioning) instruction, and MC_MoveFeed (Interrupt Feeding) instruction.<br>• The MC_AbortTrigger (Disable External Latch) instruction was executed to cancel a latch that was used by an instruction other than the MC_TouchProbe (Enable External Latch) instruction. | | | √ | | | page A-56 |
| 546F0000 hex | Jerk Override Factor Out of Range | The parameter specified for the *JerkFactor* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-57 |
| 54700000 hex | Acceleration/Deceleration Override Factor Out of Range | The parameter specified for the *AccFactor* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-57 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 54710000 hex | First Position Method Specification Out of Range | The parameter specified for the *StartMode* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-57 |
| 54720000 hex | Motion Control Instruction Re-execution Disabled (First Position Method) | An attempt was made to change the *StartMode* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | • A parameter for an input variable that cannot be changed for re-execution was changed. | | | √ | | | page A-58 |
| 54740000 hex | Unused Axis Specification for Auxiliary Axis | The axis specified for the *Auxiliary* input variable to a motion control instruction is an unused axis. | • The axis specified for the *Auxiliary* input variable to the instruction is an unused axis. | | | √ | | | page A-58 |
| 54750000 hex | Position Gear Value Error | Synchronized motion is not possible for the velocity, acceleration rate, and deceleration rate that were input to a motion control instruction. | • The specified synchronized motion cannot be performed at the velocity, acceleration rate, or deceleration rate that is input to the instruction. | | | √ | | | page A-59 |
| 54760000 hex | Position Gear Master Axis Zero Velocity | The velocity of the master axis was zero when a motion control instruction was started. | • The velocity of the master axis was 0 when the instruction was started. | | | √ | | | page A-59 |
| 54780000 hex | Target Position Setting Out of Range | The parameter specified for the *Position* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable.<br>• The target position of a Rotary Mode axis is not within the ring setting range. | | | √ | | | page A-60 |
| 54790000 hex | Travel Distance Out of Range | The parameter that was specified for the *Distance* input variable to a motion control instruction is out of range or the target position with the value of *Distance* added is out of range. | • The absolute value of the instruction input parameter exceeded the range of 40-bit data when it is converted to pulses.<br>• For a Linear Mode axis, the target position with the travel distance added exceeded signed 40-bit data when the absolute value is converted to pulses. | | | √ | | | page A-60 |
| 547A0000 hex | Cam Table Start Point Setting Out of Range | The parameter specified for the *StartPosition* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-61 |
| 547B0000 hex | Cam Master Axis Following First Position Setting Out of Range | The parameter specified for the *MasterStartDistance* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-61 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 547C0000 hex | Circular Interpolation Radius Setting Error | It was not possible to create a circular path for the specified radius when the radius method was specified for the MC_MoveCircular2D (Circular 2D Interpolation) instruction. | • For the MC_MoveCircular2D (Circular 2D Interpolation) instruction, it was not possible to create a circular path for the specified radius when the radius method was specified for circular interpolation. | | | √ | | | page A-62 |
| 547D0000 hex | Circular Interpolation Radius Overflow | For the MC_MoveCircular2D (Circular 2D Interpolation) instruction, the radius of the circle exceeded the maximum value for the border point or center specification method. | • For the MC_MoveCircular2D (Circular 2D Interpolation) instruction, the radius of the circle exceeded 40-bit data when converted to pulses for the border point or center specification method. | | | √ | | | page A-62 |
| 547E0000 hex | Circular Interpolation Setting Out of Range | The parameter specified for the *CircAxes* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable.<br>• The axes that were specified in *CircAxes* are not included in the composition axes in the Axes Group Settings.<br>• The same axis was specified for both axes of *CircAxes*. | | | √ | | | page A-63 |
| 547F0000 hex | Auxiliary/Slave Axis Numbers Not in Ascending Order | The values of the parameters for the *Auxiliary* and *Slave* input variables to a motion control instruction are not in ascending order. | • The parameters for the *Auxiliary* and *Slave* input variables to the instruction are not in ascending order. | | | √ | | | page A-63 |
| 54800000 hex | Cam Table Property Ascending Data Error at Update | A phase that was not in ascending order was found during calculating the number of valid data. Or, after calculations, the number of valid data is 0. | • A phase that was not in ascending order was found when calculating the number of valid data.<br>• After calculations, the number of valid data is 0. | | | √ | | | page A-64 |
| 54810000 hex | MC_Write Target Out of Range | The parameter specified for the *Target* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-64 |
| 54820000 hex | Master Travel Distance Specification Out of Range | The parameter specified for the *MasterDistance* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-64 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 54830000 hex | Master Distance in Acceleration Specification Out of Range | The parameter specified for the *MasterDistanceACC* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-65 |
| 54840000 hex | Master Distance in Deceleration Specification Out of Range | The parameter specified for the *MasterDistanceDEC* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-65 |
| 54870000 hex | Execution Mode Selection Out of Range | The parameter specified for the *ExecutionMode* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-65 |
| 54880000 hex | Permitted Following Error Out of Range | The parameter specified for the *PermittedDeviation* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-66 |
| 54890000 hex | Border Point/Center Position/Radius Specification Out of Range | The parameter specified for the *AuxPoint* input variable to a motion control instruction is out of range. | • The value of *AutPoint* exceeded signed 40-bit data when converted to pulses for the border point or center specification method.<br>• For a radius specifications, the absolute value of *AuxPoint[0]* exceeded 40-bit data when converted to pulses. | | | √ | | | page A-66 |
| 548A0000 hex | End Point Specification Out of Range | The parameter specified for the *EndPoint* input variable to a motion control instruction is out of range. | • The instruction input parameter exceeded the range of signed 40-bit data when it is converted to pulses. | | | √ | | | page A-67 |
| 548B0000 hex | Slave Travel Distance Specification Out of Range | The parameter specified for the *SlaveDistance* input variable to a motion control instruction is out of range. | • The instruction input parameter exceeded the range of 40-bit data when it is converted to pulses. | | | √ | | | page A-67 |
| 548C0000 hex | Phase Shift Amount Out of Range | The parameter specified for the *PhaseShift* input variable to a motion control instruction is out of range. | • The absolute value of the instruction input parameter exceeded the range of 40-bit data when it is converted to pulses. | | | √ | | | page A-67 |
| 548D0000 hex | Feeding Distance Out of Range | The parameter specified for the *FeedDistance* input variable to a motion control instruction is out of range. | • The absolute value of the instruction input parameter exceeded the range of 40-bit data when it is converted to pulses. | | | √ | | | page A-68 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 548E0000 hex | Auxiliary and Slave Defined as Same Axis | The same axis was specified for the *Auxiliary* and *Slave* input variables to a motion control instruction. | • The parameter is the same for the *Auxiliary* and *Slave* input variables to the instruction. | | | √ | | | page A-68 |
| 548F0000 hex | Relative Position Selection Out of Range | The parameter specified for the *Relative* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-68 |
| 54900000 hex | Cam Transition Specification Out of Range | The parameter specified for the *CamTransition* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-69 |
| 54910000 hex | Synchronized Control End Mode Selection Out of Range | The parameter specified for the *OutMode* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-69 |
| 54920000 hex | Enable External Latch Instruction Execution Disabled | *_mcImmediateStop* was specified for the *StopMode* input variable when the MC_TouchProbe (Enable External Latch) instruction was executed in Drive Mode, but the Control Mode was not CSP Mode. | • *_mcImmediateStop* was specified for the *StopMode* input variable when the MC_TouchProbe (Enable External Latch) instruction was executed in Drive Mode, but the Control Mode was not CSP Mode or the Servo was OFF. | | | √ | | | page A-70 |
| 54930000 hex | Master Axis Offset Out of Range | The parameter specified for the *MasterOffset* input variable to a motion control instruction is out of range. | • The instruction input parameter exceeded the range of signed 40-bit data when it is converted to pulses. | | | √ | | | page A-70 |
| 54940000 hex | Slave Axis Offset Out of Range | The parameter specified for the *SlaveOffset* input variable to a motion control instruction is out of range. | • The instruction input parameter exceeded the range of signed 40-bit data when it is converted to pulses. | | | √ | | | page A-71 |
| 54950000 hex | Command Current Position Count Selection Out of Range | The parameter specified for the *CmdPosMode* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-71 |
| 54960000 hex | Master Axis Gear Ratio Numerator Out of Range | The parameter specified for the *RatioNumeratorMaster* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-71 |

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 54970000 hex | Master Axis Gear Ratio Denominator Out of Range | The parameter specified for the *RatioDenominatorMaster* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-72 |
| 54980000 hex | Auxiliary Axis Gear Ratio Numerator Out of Range | The parameter specified for the *RatioNumeratorAuxiliary* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-72 |
| 54990000 hex | Auxiliary Axis Gear Ratio Denominator Out of Range | The parameter specified for the *RatioDenominatorAuxiliary* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-72 |
| 549A0000 hex | Master Axis Position Type Selection Out of Range | The parameter specified for the *ReferenceTypeMaster* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-73 |
| 549B0000 hex | Auxiliary Axis Position Type Selection Out of Range | The parameter specified for the *ReferenceTypeAuxiliary* input variable to a motion control instruction is out of range. | • Instruction input parameter exceeded the valid range of the input variable. | | | √ | | | page A-73 |
| 549C0000 hex | Target Position Ring Counter Out of Range | Operation is not possible because the target position is out of range for the ring counter of the executed instruction. | • High-speed homing was executed when 0 was not included in the ring counter. | | | √ | | | page A-74 |
| 64400000 hex | Target Position Positive Software Limit Exceeded | The specified position exceeds the positive software limit. | • The parameter specified for the *Position* input variable to the instruction is beyond the positive software limit.<br>• The first position is beyond the positive software limit and an instruction that specifies motion in the opposite direction of the software limit was executed.<br>• The parameter that was specified for the *AuxPoint* input variable to a border point MC_MoveCircular2D (Circular 2D Interpolation) instruction is beyond the positive software limit | | | √ | | | page A-74 |

A-1 Error Codes

App

| Event code | Event name | Meaning | Assumed cause | Level | | | | | Reference |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Maj | Prt | Min | Obs | Info | |
| 64410000 hex | Target Position Negative Software Limit Exceeded | The specified position exceeds the negative software limit. | • The parameter specified for the *Position* input variable to the instruction is beyond the negative software limit.<br>• The first position is beyond the negative software limit and an instruction that specifies motion in the opposite direction of the software limit was executed<br>• The parameter that was specified for the *AuxPoint* input variable to a border point MC_MoveCircular2D (Circular 2D Interpolation) instruction is beyond the negative software limit | | | √ | | | page A-75 |
| 64420000 hex | Command Position Overflow/Underflow | Positioning, an instruction in the underflow/overflow direction, or an instruction for which the direction is not specified was executed when there was an underflow/overflow in the command position. | • One of the following was executed when there was a command position overflow/underflow.<br>• A positioning instruction<br>• A continuous control instruction in the underflow/overflow direction<br>• An instruction for which the direction is not specified (syncing or torque control) | | | √ | | | page A-76 |
| 64430000 hex | Positive Limit Input | An instruction was executed for a motion in the positive direction when the positive limit input was ON. | • An instruction for a motion in the positive direction was executed when the positive limit input was ON, or an instruction for a motion with no direction specification was executed when the positive limit input was ON. | | | √ | | | page A-76 |
| 64440000 hex | Negative Limit Input | An instruction for a motion in the negative direction was executed when the negative limit input was ON. | • An instruction for a motion in the negative direction was executed when the negative limit input was ON, or an instruction for a motion with no direction specification was executed when the negative limit input was ON. | | | √ | | | page A-77 |
| 74220000 hex | Servo Main Circuits OFF | An attempt was made to turn ON the Servo when the main circuit power supply to the Servo Drive was OFF. | • An attempt was made to turn ON the Servo when the main circuit power supply to the Servo Drive was OFF. | | | √ | | | page A-77 |

# A-2 Error Code Details

## Controller Error Descriptions

The items that are used to describe individual errors (events) are described in the following copy of an error table.

<table>
<tr><td>Event name</td><td colspan="3">Gives the name of the error.</td><td>Event code</td><td colspan="2">Gives the code of the error.</td></tr>
<tr><td>Meaning</td><td colspan="6">Gives a short description of the error.</td></tr>
<tr><td>Source</td><td colspan="2">Gives the source of the error.</td><td>Source details</td><td>Gives details on the source of the error.</td><td>Detection timing</td><td>Tells when the error is detected.</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Tells the level of influence on control.*1</td><td>Recovery</td><td>Gives the recovery method.*2</td><td>Log category</td><td>Tells which log the error is saved in.*3</td></tr>
<tr><td>Effects</td><td>User program</td><td>Tells what will happen to execution of the user program.*4</td><td>Operation</td><td colspan="3">Provides special information on the operation that results from the error.</td></tr>
<tr><td>Indicators</td><td colspan="6">Gives the status of the built-in EtherNet/IP port and built-in EtherCAT port indicators. Indicator status is given only for errors in the EtherCAT Master Function Module and the EtherNet/IP Function Module.</td></tr>
<tr><td rowspan="2">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="6">Lists the variable names, data types, and meanings for system-defined variables that provide direct error notification, that are directly affected by the error, or that contain settings that cause the error.</td></tr>
<tr><td rowspan="2">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="6">Lists the possible causes, corrections, and preventive measures for the error.</td></tr>
<tr><td>Attached information</td><td colspan="6">This is the attached information that is displayed by the Sysmac Studio or an NS-series PT.</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">Provides precautions, restrictions, and supplemental information.</td></tr>
</table>

*1 One of the following:
Major fault: Major fault level
Partial fault: Partial fault level
Minor fault: Minor fault level
Observation
Information

*2 One of the following:
Automatic recovery: Normal status is restored automatically when the cause of the error is removed.
Error reset: Normal status is restored when the error is reset after the cause of the error is removed.
Cycle the power supply: Normal status is restored when the power supply to the Controller is turned OFF and then back ON after the cause of the error is removed.
Controller reset: Normal status is restored when the Controller is reset after the cause of the error is removed.
Depends on cause: The recovery method depends on the cause of the error.

*3 One of the following:
System: System event log
Access: Access event log

*4 One of the following:
Continues: Execution of the user program will continue.
Stops: Execution of the user program stops.
Starts: Execution of the user program starts.

## Error Descriptions

The following table describes the error codes that are output to *ErrorID* when errors occur in execution of the instructions. The upper four digits of the event codes that are given in the following table are output as the error code to *ErrorID*.

| Event name | Process Data Object Setting Missing | | | Event code | 34610000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The PDO mapping is not correct. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Warning Active | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The PDOs that are required for the motion control instruction are not mapped. | | Map the PDOs that are required for the instruction. Refer to the *Function* section of the relevant instruction for the required PDOs. | | Map the PDOs that are required for the instructions that are used. Refer to the *NJ-series CPU Unit Motion Control User's Manual* (Cat. No. W507) for the PDOs (Servo Drive settings) that you must map for each instruction. | |
| | A motion control instruction that specifies phase Z (_mcEncoderMark) as the trigger conditions was executed for an axis that is mapped to an OMRON GX-EC02□□ EtherCAT Encoder slave. | | Use an external input (_mcEXT) as the trigger conditions for an axis that is mapped to an OMRON GX-EC02□□ EtherCAT Encoder slave. | | Use an external input (_mcEXT) as the trigger conditions for an axis that is mapped to an OMRON GX-EC02□□ EtherCAT Encoder slave. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Electronic Gear Ratio Numerator Setting Out of Range | | | Event code | 54200000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The parameter specified for the *RatioNumerator* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Electronic Gear Ratio Denominator Setting Out of Range | | | Event code | 54210000 hex | |
| Meaning | The parameter specified for the *RatioDenominator* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Target Velocity Setting Out of Range | | | Event code | 54220000 hex | |
| Meaning | The parameter specified for the *Velocity* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If “axis” is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If “axes group” is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The Target Velocity (input variable *Velocity*) is still at the default (0). | | Specify a positive value for the Target Velocity (input variable *Velocity*). | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Acceleration Setting Out of Range | | | Event code | 54230000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The parameter specified for the *Acceleration* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If "axis" is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If "axes group" is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Deceleration Setting Out of Range | | | Event code | 54240000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The parameter specified for the *Deceleration* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If "axis" is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If "axes group" is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Jerk Setting Out of Range | | | Event code | 54250000 hex |
| Meaning | The parameter specified for the *Jerk* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If "axis" is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If "axes group" is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Torque Ramp Setting Out of Range | | | | Event code | 54270000 hex |
| Meaning | The parameter specified for the *TorqueRamp* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Master Coefficient Scaling Out of Range | | | Event code | 54280000 hex | |
| Meaning | The parameter specified for the *MasterScaling* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | Controller |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Slave Coefficient Scaling Out of Range | | | Event code | 54290000 hex | |
| Meaning | The parameter specified for the *SlaveScaling* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Feeding Velocity Setting Out of Range | | | Event code | 542A0000 hex | |
| Meaning | The parameter specified for the *FeedVelocity* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The Feed Velocity (input variable *FeedVelocity*) is still at the default (0). | | Specify a positive value for the Feed Velocity (input variable *FeedVelocity*). | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Buffer Mode Selection Out of Range | | | Event code | 542B0000 hex | |
| Meaning | The parameter specified for the *BufferMode* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If "axis" is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If "axes group" is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Coordinate System Selection Out of Range | | | Event code | 542C0000 hex | |
| Meaning | The parameter specified for the *CoordSystem* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Circular Interpolation Mode Selection Out of Range | | | Event code | 542D0000 hex |
|---|---|---|---|---|---|
| Meaning | The parameter specified for the *CircMode* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| Event name | Direction Selection Out of Range | | | Event code | 542E0000 hex |
|---|---|---|---|---|---|
| Meaning | The parameter specified for the *Direction* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| Event name | Path Selection Out of Range | | | Event code | 542F0000 hex |
|---|---|---|---|---|---|
| Meaning | The parameter specified for the *PathChoice* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Position Type Selection Out of Range | | | Event code | 54300000 hex |
| Meaning | The parameter specified for the *ReferenceType* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | | Source details | MC common or axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_COM.MFaultLvl.Active | | BOOL | | MC Common Minor Fault Occurrence | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Travel Mode Selection Out of Range | | | Event code | 54310000 hex | |
| Meaning | The parameter specified for the *MoveMode* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If “axis” is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If “axes group” is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Transition Mode Selection Out of Range | | | Event code | 54320000 hex |
| Meaning | The parameter specified for the *TransitionMode* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| | *_mcAborting* or *_mcBuffered* was specified for *BufferMode* and *_mcTMCornerSuperimpose* was specified for *TransitionMode.* | | If you specify *_mcAborting* or *_mcBuffered* for *BufferMode*, specify *_mcTMNone* for *TransitionMode*. If you specify *_mcTMCornerSuperimpose* for *TransitionMode*, specify *_mcBlendingLow*, *_mcBlendingPrevious*, *_mcBlendingNext*, or *_mcBlendingHigh* for *BufferMode*. | | If you specify *_mcAborting* or *_mcBuffered* for *BufferMode*, specify *_mcTMNone* for *TransitionMode*. If you specify *_mcTMCornerSuperimpose* for *TransitionMode*, specify *_mcBlendingLow*, *_mcBlendingPrevious*, *_mcBlendingNext*, or *_mcBlendingHigh* for *BufferMode*. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Continue Method Selection Out of Range | | | Event code | 54330000 hex | |
| Meaning | The value of the reserved input variable *Continuous* to a motion control instruction changed. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The value of the reserved input variable *Continuous* changed. | | Correct the program so that the value of the reserved input variable *Continuous* does not change. | | Write the user program so that the value of the reserved input variable *Continuous* does not change. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Combine Mode Selection Out of Range | | | Event code | 54340000 hex |
|---|---|---|---|---|---|
| Meaning | The parameter specified for the *CombineMode* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Synchronization Start Condition Selection Out of Range | | | Event code | 54350000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The parameter specified for the *LinkOption* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Master and Slave Defined as Same Axis | | | Event code | 54360000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The same axis is specified for the *Master* and *Slave* input variables to a motion control instruction. | | | | | |
| Source | Motion Control Function Module | | Source details | MC common or axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_COM.MFaultLvl.Active | | BOOL | | MC Common Minor Fault Occurrence | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The parameter is the same for the *Master* and *Slave* input variables to the instruction. | | Correct the parameters so that different axes are specified for the *Master* and *Slave* input variables to the instruction. | | Specify different axes for the *Master* and *Slave* input variables to the instruction. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Master and Auxiliary Defined as Same Axis | | | Event code | 54370000 hex | |
| Meaning | The same axis is specified for the *Master* and *Auxiliary* input variables to a motion control instruction. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The parameter is the same for the *Master* and *Auxiliary* input variables to the instruction. | | Correct the parameters so that different axes are specified for the *Master* and *Auxiliary* input variables to the instruction. | | Specify different axes for the *Master* and *Auxiliary* input variables to the instruction. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Master/Slave Axis Numbers Not in Ascending Order | | | Event code | 54380000 hex | |
| Meaning | The axis numbers specified for the *Master* and *Slave* input variables to a motion control instruction are not in ascending order. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The parameters for the *Master* and *Slave* input variables to the instruction were not in ascending order when *_mcLatestCommand* was specified for the *ReferenceType* input variable to the instruction. | | When specifying *_mcLatestCommand* for the *ReferenceType* input variable to the instruction, correct the parameters so that the axis numbers specified for the *Master* and *Slave* input variables to the instruction are in ascending order. Or, specify *_mcCommand* for the Position Type Selection. | | When specifying *_mcLatestCommand* for the *ReferenceType* input variable, make sure to specify the master axis and slave axis input variables so that they are in ascending order. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Incorrect Cam Table Specification | | | Event code | 54390000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The parameter specified for the *CamTable* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | MC common or axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If "MC Common" is given for the source details, operation is not affected.<br>If "axis" is given for the source details, operation is not possible for relevant slave axis. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_COM.MFaultLvl.Active | | BOOL | | MC Common Minor Fault Occurrence | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Something other than a cam data variable was specified for the *CamTable* input variable to the instruction. | | Correct the parameter specified for the *CamTable* input variable to the instruction so that it is a cam data variable. | | Specify a cam data variable for the *CamTable* input variable to the instruction. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Synchronization Stopped | | | Event code | 543A0000 hex | |
|---|---|---|---|---|---|---|
| Meaning | A synchronized control motion control instruction was executed, but conditions required for execution were not met. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | • The MC_CamOut (End Cam Operation) instruction was executed even though the MC_CamIn (Start Cam Operation) instruction is not being executed.<br>• The MC_GearOut (End Gear Operation) instruction was executed even though the MC_GearIn (Start Gear Operation) or the MC_GearInPos (Positioning Gear Operation) instruction is not being executed.<br>• The MC_Phasing (Shift Master Axis Phase) instruction was executed even though the MC_CamIn (Start Cam Operation), MC_GearIn (Start Gear Operation), MC_GearInPos (Start Gear Operation), or MC_MoveLink (Synchronous Positioning) instruction is not being executed. | | Correct the program so that required conditions are met when the instruction is executed. | | Make sure that required conditions for execution are met when you execute sync instructions. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

A-2 Error Code Details

App

<table>
<tr><td>Event name</td><td colspan="3">Motion Control Instruction Re-execution Disabled</td><td>Event code</td><td colspan="2">543B0000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">An attempt was made to re-execute a motion control instruction that cannot be re-executed.</td></tr>
<tr><td>Source</td><td colspan="2">Motion Control Function Module</td><td>Source details</td><td>MC common, axis, or axes group</td><td>Detection timing</td><td>At instruction execution</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Minor fault</td><td>Recovery</td><td>Error reset</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Continues.</td><td>Operation</td><td colspan="3">If “MC Common” is given for the source details, operation is not affected.<br>If “axis” is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If “axes group” is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion.</td></tr>
<tr><td rowspan="4">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">_MC_COM.MFaultLvl.Active</td><td colspan="2">BOOL</td><td colspan="2">MC Common Minor Fault Occurrence</td></tr>
<tr><td colspan="2">_MC_AX[*].MFaultLvl.Active</td><td colspan="2">BOOL</td><td colspan="2">Axis Minor Fault Occurrence</td></tr>
<tr><td colspan="2">_MC_GRP[*].MFaultLvl.Active</td><td colspan="2">BOOL</td><td colspan="2">Axes Group Minor Fault Occurrence</td></tr>
<tr><td rowspan="2">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">A motion control instruction that cannot be re-executed was re-executed.</td><td colspan="2">Correct the program so that the *Execute* input variable does not change to TRUE until the *Busy* output variable from the instruction changes to FALSE.</td><td colspan="2">When using instructions that cannot be re-executed, include a condition for the *Execute* input variable so that it does not change to TRUE unless the *Busy* output variable for the previous instruction is FALSE. Or, stop the instruction before executing it again.</td></tr>
<tr><td>Attached information</td><td colspan="6">None</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">None</td></tr>
</table>

<table>
<tr><td>Event name</td><td colspan="3">Motion Control Instruction Multi-execution Disabled</td><td>Event code</td><td colspan="2">543C0000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">Multiple functions that cannot be executed simultaneously were executed for the same target (MC common or axis).</td></tr>
<tr><td>Source</td><td colspan="2">Motion Control Function Module</td><td>Source details</td><td>MC common or axis</td><td>Detection timing</td><td>At multi-execution of instructions</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Minor fault</td><td>Recovery</td><td>Error reset</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Continues.</td><td>Operation</td><td colspan="3">If “MC Common” is given for the source details, operation is not affected.<br>If “axis” is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.</td></tr>
<tr><td rowspan="3">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">_MC_COM.MFaultLvl.Active</td><td colspan="2">BOOL</td><td colspan="2">MC Common Minor Fault Occurrence</td></tr>
<tr><td colspan="2">_MC_AX[*].MFaultLvl.Active</td><td colspan="2">BOOL</td><td colspan="2">Axis Minor Fault Occurrence</td></tr>
<tr><td rowspan="2">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">Multiple functions that cannot be executed simultaneously were executed for the same target (MC common or axis).</td><td colspan="2">Check the specifications of multi-execution of instructions for this instruction and do not execute instructions that cannot be executed at the same time.</td><td colspan="2">Check the specifications for multi-execution of instructions for the instruction and do not execute instructions that cannot be executed at the same time.</td></tr>
<tr><td>Attached information</td><td colspan="6">None</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">None</td></tr>
</table>

| | | | | | |
|---|---|---|---|---|---|
| Event name | Instruction Not Allowed for Encoder Axis Type | | Event code | 543D0000 hex | |
| Meaning | An operation instruction was executed for an encoder axis. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | An operation instruction was executed for an encoder axis. | Specify either a Servo axis or virtual Servo axis as the axis type for the instruction, or correct the program so that the instruction is not executed for an encoder axis. | | Only execute motion instructions for Servo axes or virtual Servo axes. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Instruction Cannot Be Executed during Multi-axes Coordinated Control | | | Event code | 543E0000 hex | |
| Meaning | An operation instruction was executed for an axis that was in a coordinated multi-axes motion. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At multi-execution of instructions |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | The axes group decelerates to a stop. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | An operation instruction was executed for an axis that was in a multi-axes coordinated motion. | | Correct the program so that the instruction is executed when the relevant axis is not in a multi-axes coordinated motion. | | Make sure to execute axis operation instructions only for axes that are not in coordinated motion. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Multi-axes Coordinated Control Instruction Executed for Disabled Axes Group | | | Event code | 543F0000 hex | |
| Meaning | A multi-axes coordinated control instruction was executed for an axes group that was in the Axes Group Disabled state. | | | | | |
| Source | Motion Control Function Module | | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | A multi-axes coordinated control instruction was executed for an axes group that was in the Axes Group Disabled state. | | Correct the program so that the instruction is executed only after changing the axes group to the Axes Group Enabled state. Execute the MC_GroupEnable (Enable Axes Group) instruction to change an axes group to the Axes Group Enabled state. | | Execute multi-axes coordinated operation instructions only after enabling the axes group. Execute the MC_GroupEnable (Enable Axes Group) instruction to change an axes group to the Axes Group Enabled state. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Axes Group Cannot Be Enabled | | | Event code | 54400000 hex | |
| Meaning | Execution of the MC_GroupEnable (Enable Axes Group) instruction failed. | | | | | |
| Source | Motion Control Function Module | | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axes group. The operation of the composition axes will continue. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | When the MC_GroupEnable (Enable Axes Group) instruction was executed, there was a composition axis that was not stopped. | | Correct the program so that the MC_GroupEnable (Enable Axes Group) instruction is executed only when all composition axes are stopped. An axis is stopped if *Status.Disabled* or *Status.Standstill* is TRUE in the Axis Variable. | | Write the programs so that the MC_GroupEnable (Enable Axes Group) instruction is executed only when all composition axes are stopped. An axis is stopped if *Status.Disabled* or *Status.Standstill* is TRUE in the Axis Variable. | |
| | When the MC_GroupEnable (Enable Axes Group) instruction was executed, there was a composition axis for which the MC_TouchProbe (Enable External Latch) instruction was being executed. | | Correct the program so that the MC_GroupEnable (Enable Axes Group) instruction is executed only when the MC_TouchProbe (Enable External Latch) instruction is not being executed for any of the composition axes. | | Write the program so that the MC_GroupEnable (Enable Axes Group) instruction is executed only when the MC_TouchProbe (Enable External Latch) instruction is not being executed for any of the composition axes. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Impossible Axis Operation Specified when the Servo is OFF | | | Event code | 54410000 hex | |
| Meaning | An operation instruction was executed for an axis for which the Servo is OFF. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | The motion instruction will not start. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | An operation instruction was executed for an axis for which the Servo is OFF. | | Correct the program so that the instruction is executed after the Servo is turned ON. | | Make sure to execute the axis operation instruction after the Servo is turned ON. | |
| | Home was preset with the MC_Home instruction for an axis for which EtherCAT process data communications are not established. | | If the *_EC_PDSlavTbl* (Process Data Communicating Slave Table) system-defined variable for the EtherCAT master of the master axis is FALSE, remove the cause and execute the MC_Home instruction to preset home after *_EC_PDSlavTbl* changes to TRUE. | | If you execute the MC_Home instruction to preset home immediately after you turn ON the power supply to the Controller, download data, reset a slave communications error, disconnect the slave, or reconnect the slave, write the program to make sure that the *_EC_PDSlavTbl* (Process Data Communicating Slave Table) system-defined variable for the EtherCAT master is TRUE before you execute MC_Home. | |
| Attached information | Attached information 1: Depends on the source details.<br>Axis: 0<br>Axes group: Number of the logical axis where the error occurred | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Composition Axis Stopped Error | | | Event code | 54420000 hex | |
| Meaning | A motion instruction was executed for an axes group while the MC_Stop instruction was being executed for a composition axis. | | | | | |
| Source | Motion Control Function Module | | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axes group. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | A motion instruction was executed for an axes group while the MC_Stop instruction was being executed for a composition axis. | | Change the *Execute* input variable to the MC_Stop instruction for the composition axis to FALSE, reset the error, and then execute the motion control instruction. | | Change the *Execute* input variables to the MC_Stop instructions for all of the composition axes to FALSE before you execute motion control instruction. | |
| Attached information | Attached information 1: Number of the logical axis that was stopped. | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Motion Control Instruction Multi-execution Buffer Limit Exceeded | | Event code | 54430000 hex | |
| Meaning | The number of motion control instructions that is buffered for Buffered or Blending Buffer Modes exceeded the buffer limit. | | | | |
| Source | Motion Control Function Module | Source details | Axis/axes group | Detection timing | Controller |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | If "axis" is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If "axes group" is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | Name | | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | Axis Minor Fault Occurrence | | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | Axes Group Minor Fault Occurrence | | |
| Cause and correction | Assumed cause | Correction | Prevention | | |
| | An axis instruction was executed when there was already a current instruction and a buffered instruction for the same axis. | Correct the program so that the number of executed instructions does not exceed the buffer limit. | Do not execute an axis instruction when there is already a current instruction and a buffered instruction for the same axis. | | |
| | An axes group instruction was executed when there was already eight current instructions and buffered instructions for the same axis. | | Do not execute an axes group instruction when there are already eight current and buffered instructions for the same axis. | | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Insufficient Travel Distance | | | Event code | 54440000 hex |
| Meaning | The specified motion cannot be executed for the deceleration rate or acceleration rate that was specified for multi-execution or re-execution of a positioning instruction. | | | | |
| Source | Motion Control Function Module | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | If "axis" is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If "axes group" is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | Name | | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | Axis Minor Fault Occurrence | | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | Axes Group Minor Fault Occurrence | | |
| Cause and correction | Assumed cause | Correction | Prevention | | |
| | Stopping at the target position was not possible for the specified acceleration/deceleration rate for multi-execution or re-execution of a positioning instruction when the Acceleration/Deceleration Over parameter was set to generate a minor fault and stop. | Correct the program based on the operating specifications for the instruction so that the target position is not exceeded at the deceleration rate or acceleration rate specified for multi-execution or re-execution of the positioning instruction. Or, change the Acceleration/Deceleration Over parameter to a setting other than to generate a minor fault and stop. | Check the operating specifications for the relevant instruction and write the program so that this error does not occur. Or, change the Acceleration/Deceleration Over parameter to a setting other than to generate a minor fault and stop. | | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Insufficient Travel Distance to Achieve Blending Transit Velocity | | | Event code | 54450000 hex |
| Meaning | There is not sufficient travel distance to accelerate or decelerate to the transit velocity. | | | | |
| Source | Motion Control Function Module | Source details | Axis/axes group | Detection timing | At multi-execution of instructions |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | If "axis" is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If "axes group" is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | Name | | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | Axis Minor Fault Occurrence | | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | Axes Group Minor Fault Occurrence | | |
| Cause and correction | Assumed cause | Correction | Prevention | | |
| | There was not sufficient travel distance to accelerate the current command to the transit velocity when the Acceleration/Deceleration Over parameter was set to generate a minor fault and stop. | Correct the program to allow a sufficient travel distance according to the operating specifications of the instruction. Or, change the Acceleration/Deceleration Over parameter to a setting other than to generate a minor fault and stop. | Check the operating specifications for the relevant instruction and write the program so that this error does not occur. Or, change the Acceleration/Deceleration Over parameter to a setting other than to generate a minor fault and stop. | | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Move Link Constant Velocity Insufficient Travel Distance | | | Event code | 54460000 hex | |
| Meaning | The constant-velocity travel distance of the master axis is less than zero. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The constant velocity travel distance of the master axis is below 0 for the MC_MoveLink (Synchronous Positioning) instruction. | | Correct the program so that the master distance is greater than or equal to the master distance in acceleration plus the master distance in deceleration. | | Check the operating specifications for the relevant instruction and write the program so that this error does not occur. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Positioning Gear Operation Insufficient Target Velocity | | | Event code | 54470000 hex | |
| Meaning | For the MC_GearInPos (Positioning Gear Operation) instruction, the target velocity of the slave axis is too small to achieve the required velocity. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | For the MC_GearInPos (Positioning Gear Operation) instruction, the value of the *Velocity* (Target Velocity) input variable is smaller than the master axis velocity multiplied by the gear ratio when the instruction was executed. | | Set the value of the *Velocity* (Target Velocity) input variable to a value that is greater than the master axis velocity multiplied by the gear ratio when the instruction is executed based on the operating specifications of the instruction. | | Check the operating specifications for the relevant instruction and write the program so that this error does not occur. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Same Start Point and End Point for Circular Interpolation | | Event code | 54480000 hex | |
| Meaning | The start point and end point were the same when the radius method was specified for the MC_MoveCircular2D (Circular 2D Interpolation) instruction. Or, the start point, end point, and border point were the same when the border point method was specified. | | | | |
| Source | Motion Control Function Module | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | The start point and end point were the same when the radius method was specified for the MC_MoveCircular2D (Circular 2D Interpolation) instruction. | Correct the program so that the radius specification is not used when the start point and end point for the instruction are the same. | | Do not use the same start point and end point when you execute circular interpolation with a radius specification. | |
| | The start point, end point, and border point were the same when the border point method was specified for the MC_MoveCircular2D (Circular 2D Interpolation) instruction. | Correct the program so that border point specification is not used when the start point, end point, and border point for the instruction are the same. | | Do not use the same start point, end point, and border point when you execute circular interpolation with a border point specification. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Circular Interpolation Center Specification Position Out of Range | | Event code | 54490000 hex | |
| Meaning | The position specified for the center point exceeded the allowed range when the center method was specified for the MC_MoveCircular2D (Circular 2D Interpolation) instruction. | | | | |
| Source | Motion Control Function Module | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | The difference between the distance from the start point to the center point and the distance between the end point to the center point exceeded the permitted value specified for the correction allowance ratio in the axes group settings when the center designation method was specified for the MC_MoveCircular2D (Circular 2D Interpolation) instruction. | Correct the center point so that the difference between the distance from the start point to the center point input variables and the distance between the end point to the center point input variables is less than the permitted value specified for the correction allowance ratio in the axes group settings. | | Correct the difference between the distance from the start point to the center point and the distance between the end point to the center point so that is does not exceed the correction allowance ratio in the axes group settings. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Circular Interpolation Cannot Be Executed with Rotary (Infinite) Axis | | Event code | 544A0000 hex | |
| Meaning | The MC_MoveCircular2D (Circular 2D Interpolation) instruction was executed for an axis for which the Count Mode was set to Rotary Mode. | | | | |
| Source | Motion Control Function Module | | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | An axis in Rotary Mode was used with the MC_MoveCircular2D (Circular 2D Interpolation) instruction. | | Change the Count Mode of the relevant axis to Linear Mode. | | Set the Count Mode of any axis that is used in the Circular 2D Interpolation instruction to Linear Mode. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Parameter Selection Out of Range | | | Event code | 544C0000 hex | |
| Meaning | The parameter specified for the *ParameterNumber* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | MC Common | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | The instruction is not executed. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_COM.MFaultLvl.Active | | BOOL | | MC Common Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Stop Method Selection Out of Range | | | Event code | 544D0000 hex | |
| Meaning | The parameter specified for the *StopMode* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Latch ID Selection Out of Range for Trigger Input Condition | | Event code | 544E0000 hex | |
| Meaning | The parameter specified for the *TriggerInput::LatchID* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | | Data type | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Setting Out of Range for Writing MC Setting | | Event code | 544F0000 hex | |
| Meaning | The parameter specified for the *SettingValue* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | MC Common | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | The relevant instruction is not executed. | |
| System-defined variables | Variable | | Data type | Name | |
| | _MC_COM.MFaultLvl.Active | | BOOL | MC Common Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| | The parameter specification and the data type of the setting value do not agree. | | Make corrections so that the parameter settings and the data types of the settings agree. | Make sure the parameter settings and the data type of the setting values agree. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

<table>
<tr><td>Event name</td><td colspan="3">Trigger Input Condition Mode Selection Out of Range</td><td>Event code</td><td colspan="2">54500000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">The parameter specified for the TriggerInput:: Mode input variable to a motion control instruction is out of range.</td></tr>
<tr><td>Source</td><td colspan="2">Motion Control Function Module</td><td>Source details</td><td>Axis</td><td>Detection timing</td><td>At instruction execution</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Minor fault</td><td>Recovery</td><td>Error reset</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Continues.</td><td>Operation</td><td colspan="3">Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.</td></tr>
<tr><td rowspan="2">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">_MC_AX[*].MFaultLvl.Active</td><td colspan="2">BOOL</td><td colspan="2">Axis Minor Fault Occurrence</td></tr>
<tr><td rowspan="2">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">Instruction input parameter exceeded the valid range of the input variable.</td><td colspan="2">Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction.</td><td colspan="2">Set the input parameter to the instruction so that the valid range of the input variable is not exceeded.</td></tr>
<tr><td>Attached information</td><td colspan="6">None</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">None</td></tr>
</table>

<table>
<tr><td>Event name</td><td colspan="3">Drive Trigger Signal Selection Out of Range for Trigger Input Condition</td><td>Event code</td><td colspan="2">54510000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">The parameter specified for the TriggerInput::InputDrive input variable to a motion control instruction is out of range.</td></tr>
<tr><td>Source</td><td colspan="2">Motion Control Function Module</td><td>Source details</td><td>Axis</td><td>Detection timing</td><td>At instruction execution</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Minor fault</td><td>Recovery</td><td>Error reset</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Continues.</td><td>Operation</td><td colspan="3">Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.</td></tr>
<tr><td rowspan="2">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">_MC_AX[*].MFaultLvl.Active</td><td colspan="2">BOOL</td><td colspan="2">Axis Minor Fault Occurrence</td></tr>
<tr><td rowspan="2">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">Instruction input parameter exceeded the valid range of the input variable.</td><td colspan="2">Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction.</td><td colspan="2">Set the input parameter to the instruction so that the valid range of the input variable is not exceeded.</td></tr>
<tr><td>Attached information</td><td colspan="6">None</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">None</td></tr>
</table>

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (Axis Specification) | | | Event code | 54530000 hex | |
| Meaning | An attempt was made to change the parameter for the *Axis* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (Buffer Mode Selection) | | | Event code | 54540000 hex | |
| Meaning | An attempt was made to change the parameter for the *BufferMode* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | | |
| Source | Motion Control Function Module | | Source details | Axis/axes group | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If “axis” is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If “axes group” is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (Direction Selection) | | Event code | 54550000 hex | |
| Meaning | An attempt was made to change the parameter for the *Direction* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | An input variable that cannot be changed for re-execution was changed. | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (Execution Mode) | | | Event code | 54560000 hex | |
| Meaning | An attempt was made to change the parameter for the *ExecutionMode* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (Axes Group Specification) | | Event code | 54570000 hex | |
| Meaning | An attempt was made to change the parameter for the *AxesGroup* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | |
| Source | Motion Control Function Module | Source details | Axes group | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | Data type | Name | | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | Axes Group Minor Fault Occurrence | | |
| Cause and correction | Assumed cause | Correction | Prevention | | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (Jerk Setting) | | Event code | 54580000 hex | |
| Meaning | An attempt was made to change the parameter for the *Jerk* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | |
| Source | Motion Control Function Module | Source details | Axis/axes group | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If “axis” is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If “axes group” is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | Data type | Name | | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | Axis Minor Fault Occurrence | | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | Axes Group Minor Fault Occurrence | | |
| Cause and correction | Assumed cause | Correction | Prevention | | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| Event name | Motion Control Instruction Re-execution Disabled (Master Axis) | | | Event code | 54590000 hex | |
|---|---|---|---|---|---|---|
| Meaning | An attempt was made to change the parameter for the *Master* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Motion Control Instruction Re-execution Disabled (MasterOffset) | | | Event code | 545A0000 hex | |
|---|---|---|---|---|---|---|
| Meaning | An attempt was made to change the parameter for the *MasterOffset* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Motion Control Instruction Re-execution Disabled (MasterScaling) | | Event code | 545B0000 hex | |
|---|---|---|---|---|---|
| Meaning | An attempt was made to change the parameter for the *MasterScaling* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| Event name | Motion Control Instruction Re-execution Disabled (MasterStartDistance) | | Event code | 545C0000 hex | |
|---|---|---|---|---|---|
| Meaning | An attempt was made to change the parameter for the *MasterStartDistance* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (Continuous) | | Event code | 545D0000 hex | |
| Meaning | An attempt was made to change the parameter for the *Continuous* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | | Data type | | Name |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence |
| Cause and correction | Assumed cause | | Correction | | Prevention |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (MoveMode) | | Event code | 545E0000 hex | |
| Meaning | An attempt was made to change the parameter for the *MoveMode* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | | Data type | | Name |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence |
| Cause and correction | Assumed cause | | Correction | | Prevention |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Illegal Auxiliary Axis Specification | | | Event code | 545F0000 hex |
| Meaning | The axis specified for the *Auxiliary* input variable to a motion control instruction does not exist. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant slave axis. The slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | An axis does not exist for the variable specified for the *Auxiliary* input variable to the instruction. | Correct the instruction so that the variable exists for the axis that was specified for the instruction. | | Make sure to specify variables that exist when specifying variables for the input parameters to an instruction. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Illegal Axis Specification | | | Event code | 54600000 hex |
| Meaning | The axis specified for the *Axis* input variable to a motion control instruction does not exist. | | | | |
| Source | Motion Control Function Module | Source details | MC Common | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | The relevant instruction is not executed. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_COM.MFaultLvl.Active | BOOL | | MC Common Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | An axis does not exist for the variable specified for the *Axis* input variable to the instruction. | Correct the instruction so that the variable exists for the axis that was specified for the instruction. | | Make sure to specify a variable that exists when specifying a variable for an input parameter to an instruction. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Illegal Axes Group Specification | | | Event code | 54610000 hex | |
| Meaning | The axes group specified for the *AxesGroup* input variable to a motion control instruction does not exist or is not a used group. | | | | | |
| Source | Motion Control Function Module | | Source details | MC Common | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | The relevant instruction is not executed. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_COM.MFaultLvl.Active | | BOOL | | MC Common Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | An axes group does not exist for the variable specified for the *AxesGroup* input variable to the instruction. | | Correct the specification for the instruction so that the specified axes group exists. | | Specify a variable that exists when specifying a variable for an input parameter to an instruction. | |
| | The axes group specified for the *AxesGroup* input variable to the instruction is not specified as a used group. | | Correct the axes group specified by the instruction to a used group. | | Set a used axes group for the *Axes-Group* input variable to the instruc-tion. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Illegal Master Axis Specification | | | Event code | 54620000 hex | |
| Meaning | The axis specified for the *Master* input variable to a motion control instruction does not exist or is not a sync master axis. | | | | | |
| Source | Motion Control Function Module | | Source details | MC common or axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. The slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_COM.MFaultLvl.Active | | BOOL | | MC Common Minor Fault Occurrence | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | An axis does not exist for the variable specified for the *Master* input variable to the instruction. | | Correct the instruction so that the variable exists for the axis that was specified for the instruction. | | Specify a variable that exists when specifying a variable for an input parameter to an instruction. | |
| | The axis that was specified for the *Master* input variable to the *MC_Phasing* (Shift Master Axis Phase) instruction is not the master axis for syncing. | | Correct the variable that is input to the *Master* input variable of the *MC_Phasing* (Shift Master Axis Phase) instruction to the axis variable that is specified as the master axis of the sync instruction. | | Set the variable that is input to the *Master* input variable of the *MC_Phasing* (Shift Master Axis Phase) instruction to the axis variable that is specified as the master axis of the sync instruction. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (Slave-Offset) | | Event code | 54630000 hex | |
| Meaning | An attempt was made to change the *SlaveOffset* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (SlaveScaling) | | Event code | 54640000 hex | |
| Meaning | An attempt was made to change the *SlaveScaling* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (StartPosition) | | | Event code | 54650000 hex | |
| Meaning | An attempt was made to change the *StartPosition* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Instruction Execution Error with Undefined Home | | | Event code | 54660000 hex | |
| Meaning | High-speed homing or an interpolation instruction was executed when home was undefined. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If “axis” is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If “axes group” is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | High-speed homing was executed when home was undefined. | | Execute the high-speed homing operation only after homing to define home. | | Execute the high-speed homing instruction only after home is defined by homing. | |
| | An interpolation instruction was executed for an axes group that includes an axis with no defined home. | | Perform homing to define home for all axes in the axes group before executing the interpolation instruction. | | Perform homing to define home for all axes in the axes group before executing the interpolation instruction. | |
| Attached information | Attached information 1: Depends on the source details.<br>Axis: 0<br>Axes group: Logical axis number | | | | | |
| Precautions/ Remarks | If you execute the Set Position instruction after performing homing, home will again be undefined. You must perform homing again to define home in this case. | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (Position Type) | | Event code | 54670000 hex | |
| Meaning | An attempt was made to change the *ReferenceType* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Unused Axis Specification for Master Axis | | Event code | 54680000 hex | |
| Meaning | The master axis specified for a motion control instruction is an unused axis. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | The master axis specified for a motion control instruction is an unused axis. | Set a used axis for the master axis that is specified for the instruction. | | Make sure the master axis specified for the motion control instruction is a used axis. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | First Position Setting Out of Range | | | Event code | 54690000 hex | |
| Meaning | The parameter specified for the *FirstPosition* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Last Position Setting Out of Range | | | Event code | 546A0000 hex | |
| Meaning | The parameter specified for the *LastPosition* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Illegal First/Last Position Size Relationship (Linear Mode) | | | Event code | 546B0000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The parameter specified for the *LastPosition* input variable to a motion control instruction is smaller than the parameter specified for the *FirstPosition* input variable. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The value of the *LastPosition* input parameter is less than the value of the *FirstPosition* input variable for the instruction when the Count Mode is set to Linear Mode. | | Correct the program so that the value of the *LastPosition* specified for the instruction is larger than the value of the *FirstPosition*. Or, change the value of the Count Mode to Rotary Mode. | | Write the program so that the value of the *LastPosition* specified for the instruction is larger than the value of the *FirstPosition*. Or, check to make sure that the Count Mode of the relevant axis is set to Rotary Mode. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Master Sync Start Position Setting Out of Range | | | Event code | 546C0000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The parameter specified for the *MasterSyncPosition* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Slave Sync Start Position Setting Out of Range | | | Event code | 546D0000 hex | |
| Meaning | The parameter specified for the *SlaveSyncPosition* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Duplicate Latch ID for Trigger Input Condition | | | Event code | 546E0000 hex | |
| Meaning | The same latch ID was specified for more than one motion control instruction. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The same latch ID is used simultaneously for more than one of the following instructions: MC_TouchProbe (Enable External Latch) instruction, MC_MoveLink (Synchronous Positioning) instruction, and MC_MoveFeed (Interrupt Feeding) instruction. | | Correct the program so that the same latch ID is not used by another instruction at the same time as this instruction. Either use a different latch ID or do not execute any instructions that use the same latch ID at the same time. Both latch 1 and latch 2 are treated as being in use during execution of the MC_Home instruction. | | Do not use the same latch ID simultaneously for more than one of the following instructions: MC_TouchProbe (Enable External Latch) instruction, MC_MoveLink (Synchronous Positioning) instruction, and MC_MoveFeed (Interrupt Feeding) instruction. | |
| | The MC_AbortTrigger (Disable External Latch) instruction was executed to cancel a latch that was used by an instruction other than the MC_TouchProbe (Enable External Latch) instruction. | | Do not use the Disable External Latch instruction to cancel a latch that is used by an instruction other than the Enable External Latch instruction. | | Do not execute the Disable External Latch instruction for a latch that is used by an instruction other than the Enable External Latch instruction. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | If you decide to change the latch ID, make sure that same latch ID is not used by any other instructions. | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Jerk Override Factor Out of Range | | | Event code | 546F0000 hex | |
| Meaning | The parameter specified for the *JerkFactor* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Acceleration/Deceleration Override Factor Out of Range | | | Event code | 54700000 hex | |
| Meaning | The parameter specified for the *AccFactor* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | First Position Method Specification Out of Range | | | Event code | 54710000 hex | |
| Meaning | The parameter specified for the *StartMode* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Motion Control Instruction Re-execution Disabled (First Position Method) | | Event code | 54720000 hex | |
| Meaning | An attempt was made to change the *StartMode* input variable when re-executing a motion control instruction. (This input variable cannot be changed when re-executing an instruction.) | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction re-execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | A parameter for an input variable that cannot be changed for re-execution was changed. | Correct the program so that the parameter for the relevant input variable does not change when the relevant instruction is re-executed. | | Check the manual to see if the input variables to the relevant motion control instruction can be changed by re-execution. Write the program so that the input parameters for any input variable that cannot be changed do not change upon re-execution. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Unused Axis Specification for Auxiliary Axis | | Event code | 54740000 hex | |
| Meaning | The axis specified for the *Auxiliary* input variable to a motion control instruction is an unused axis. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | The axis specified for the *Auxiliary* input variable to the instruction is an unused axis. | Set a used axis for the axis that is specified for the instruction. Or, correct the parameter so that it specifies a used axis. | | Make sure that the axis specified for the instruction is a used axis. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Position Gear Value Error | | | Event code | 54750000 hex |
| Meaning | Synchronized motion is not possible for the velocity, acceleration rate, and deceleration rate that were input to a motion control instruction. | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The specified synchronized motion cannot be performed at the velocity, acceleration rate, or deceleration rate that is input to the instruction. | | Correct the program to enable synchronized motion according to the operating specifications of the MC_GearInPos (Positioning Gear Operation) instruction. | | Check the processing of the relevant instruction and set a value that allows for synchronized motion. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Position Gear Master Axis Zero Velocity | | | Event code | 54760000 hex | |
| Meaning | The velocity of the master axis was zero when a motion control instruction was started. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The velocity of the master axis was 0 when the instruction was started. | | Correct the program so that the velocity of the master axis is not 0 when the instruction is started. | | Write the program so that the velocity of the master axis is not 0 when the instruction is started. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Target Position Setting Out of Range | | Event code | 54780000 hex | |
| Meaning | The parameter specified for the *Position* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If “axis” is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If “axes group” is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | | Data type | | Name |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence |
| Cause and correction | Assumed cause | | Correction | | Prevention |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. |
| | The target position of a Rotary Mode axis is not within the ring setting range. | | Correct the target position of the Rotary Mode axis to within the ring setting range. | | Set the target position of the Rotary Mode axis to within the ring setting range. |
| Attached information | Depends on the source details.<br>Axis: None<br>Axes group: Logical axis number | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Travel Distance Out of Range | | Event code | 54790000 hex | |
| Meaning | The parameter that was specified for the *Distance* input variable to a motion control instruction is out of range or the target position with the value of *Distance* added is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If “axis” is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If “axes group” is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | | Data type | | Name |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence |
| Cause and correction | Assumed cause | | Correction | | Prevention |
| | The absolute value of the instruction input parameter exceeded the range of 40-bit data when it is converted to pulses. | | Correct the input parameter specified for the *Distance* input variable of the instruction so that the travel distance and the target position are not out of range. | | Write the program so that the travel distance and the target position for the instruction are not out of range. |
| | For a Linear Mode axis, the target position with the travel distance added exceeded signed 40-bit data when the absolute value is converted to pulses. | | | | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Cam Table Start Point Setting Out of Range | | Event code | 547A0000 hex | |
| Meaning | The parameter specified for the *StartPosition* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | | Recovery: Error reset | | Log category: System |
| Effects | User program: Continues. | | Operation: Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Cam Master Axis Following First Position Setting Out of Range | | Event code | 547B0000 hex | |
| Meaning | The parameter specified for the *MasterStartDistance* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | | Recovery: Error reset | | Log category: System |
| Effects | User program: Continues. | | Operation: Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Circular Interpolation Radius Setting Error | | Event code | 547C0000 hex | |
| Meaning | It was not possible to create a circular path for the specified radius when the radius method was specified for the MC_MoveCircular2D (Circular 2D Interpolation) instruction. | | | | |
| Source | Motion Control Function Module | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | For the MC_MoveCircular2D (Circular 2D Interpolation) instruction, it was not possible to create a circular path for the specified radius when the radius method was specified for circular interpolation. | Correct the radius so that the circular path can be created. | | Check the processing of the relevant instruction and set a radius that allows the creation of a circular path. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Circular Interpolation Radius Overflow | | Event code | 547D0000 hex | |
| Meaning | For the MC_MoveCircular2D (Circular 2D Interpolation) instruction, the radius of the circle exceeded the maximum value for the border point or center specification method. | | | | |
| Source | Motion Control Function Module | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | For the MC_MoveCircular2D (Circular 2D Interpolation) instruction, the radius of the circle exceeded 40-bit data when converted to pulses for the border point or center specification method. | Correct the input parameter so that the circle radius does not exceed 40-bit data when converted to pulses based on the operating specifications of the instruction.<br>Center point specification: Start point, end point, and center point<br>Radius specification: Radius | | Check the processing of the instruction and correct the input parameters so that the circle radius does not exceed 40-bit data when converted to pulses. | |
| Attached information | None | | | | |
| Precautions/ Remarks | If the maximum radius is exceeded when the radius specification method is used, a Border Point/Center Position/Radius Specification Out of Range error occurs. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Circular Interpolation Setting Out of Range | | | Event code | 547E0000 hex |
| Meaning | The parameter specified for the *CircAxes* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | Data type | Name | | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | Axes Group Minor Fault Occurrence | | |
| Cause and correction | Assumed cause | Correction | Prevention | | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | Set the input parameters to the instruction so that the valid range of the input variables is not exceeded. | | |
| | The axes that were specified in *CircAxes* are not included in the composition axes in the Axes Group Settings. | Set the axes that are specified for *CircAxes* so that they are in an axes group configuration. | Make sure that the axes that are specified for *CircAxes* are in an axes group configuration. | | |
| | The same axis was specified for both axes of *CircAxes*. | Correct the settings so that the two axes specified for *CircAxes* are different axes. | Write the program so that the two axes specified for *CircAxes* are different axes. | | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Auxiliary/Slave Axis Numbers Not in Ascending Order | | | Event code | 547F0000 hex |
| Meaning | The values of the parameters for the *Auxiliary* and *Slave* input variables to a motion control instruction are not in ascending order. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | Data type | Name | | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | Axis Minor Fault Occurrence | | |
| Cause and correction | Assumed cause | Correction | Prevention | | |
| | The parameters for the *Auxiliary* and *Slave* input variables to the instruction are not in ascending order. | Correct the axis numbers specified for the *Auxiliary* and *Slave* input parameters to the instruction so that they are in ascending order. | Write the program so that the axis numbers specified for *Auxiliary* and *Slave* are in ascending order. | | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Cam Table Property Ascending Data Error at Update | | | Event code | 54800000 hex | |
| Meaning | A phase that was not in ascending order was found during calculating the number of valid data. Or, after calculations, the number of valid data is 0. | | | | | |
| Source | Motion Control Function Module | | Source details | MC common | Detection timing | During instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Not affected. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_COM.MFaultLvl.Active | | BOOL | | MC Common Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | A phase that was not in ascending order was found when calculating the number of valid data. | | Place the phase data into ascending order in the cam table data. | | Place the phase data into ascending order in the cam table data. | |
| | After calculations, the number of valid data is 0. | | Correct the cam table data so that it includes phases that are not 0. | | Create the cam table data so that it includes phases that are not 0. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | MC_Write Target Out of Range | | | Event code | 54810000 hex | |
| Meaning | The parameter specified for the *Target* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | MC common | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Not affected. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_COM.MFaultLvl.Active | | BOOL | | MC Common Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Master Travel Distance Specification Out of Range | | | Event code | 54820000 hex | |
| Meaning | The parameter specified for the *MasterDistance* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Master Distance in Acceleration Specification Out of Range | | Event code | 54830000 hex | |
|---|---|---|---|---|---|
| Meaning | The parameter specified for the *MasterDistanceACC* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| Event name | Master Distance in Deceleration Specification Out of Range | | Event code | 54840000 hex | |
|---|---|---|---|---|---|
| Meaning | The parameter specified for the *MasterDistanceDEC* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| Event name | Execution Mode Selection Out of Range | | Event code | 54870000 hex | |
|---|---|---|---|---|---|
| Meaning | The parameter specified for the *ExecutionMode* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Permitted Following Error Out of Range | | | Event code | 54880000 hex | |
| Meaning | The parameter specified for the *PermittedDeviation* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | MC Common | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | The instruction is not executed. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_COM.MFaultLvl.Active | | BOOL | | MC Common Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Border Point/Center Position/Radius Specification Out of Range | | | Event code | 54890000 hex | |
| Meaning | The parameter specified for the *AuxPoint* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The value of *AutPoint* exceeded signed 40-bit data when converted to pulses for the border point or center specification method. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| | For a radius specifications, the absolute value of *AuxPoint[0]* exceeded 40-bit data when converted to pulses. | | | | | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | End Point Specification Out of Range | | | Event code | 548A0000 hex | |
| Meaning | The parameter specified for the *EndPoint* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The instruction input parameter exceeded the range of signed 40-bit data when it is converted to pulses. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Slave Travel Distance Specification Out of Range | | | Event code | 548B0000 hex | |
| Meaning | The parameter specified for the *SlaveDistance* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The instruction input parameter exceeded the range of 40-bit data when it is converted to pulses. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Phase Shift Amount Out of Range | | | Event code | 548C0000 hex | |
| Meaning | The parameter specified for the *PhaseShift* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The absolute value of the instruction input parameter exceeded the range of 40-bit data when it is converted to pulses. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Feeding Distance Out of Range | | | Event code | 548D0000 hex |
| Meaning | The parameter specified for the *FeedDistance* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant axes. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | The absolute value of the instruction input parameter exceeded the range of 40-bit data when it is converted to pulses. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Auxiliary and Slave Defined as Same Axis | | | Event code | 548E0000 hex |
| Meaning | The same axis is specified for the *Auxiliary* and *Slave* input variables to a motion control instruction. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | The parameter was the same for the *Auxiliary* and *Slave* input variables to the instruction. | Correct the parameters so that different axes are specified for the *Auxiliary* and *Slave* input variables to the instruction. | | Specify different axes for the auxiliary axis and slave axis for a motion control instruction. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Relative Position Selection Out of Range | | | Event code | 548F0000 hex |
| Meaning | The parameter specified for the *Relative* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Cam Transition Specification Out of Range | | | Event code | 54900000 hex |
| Meaning | The parameter specified for the *CamTransition* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Synchronized Control End Mode Selection Out of Range | | | Event code | 54910000 hex |
| Meaning | The parameter specified for the *OutMode* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

<table>
<tr><td>Event name</td><td colspan="3">Enable External Latch Instruction Execution Disabled</td><td>Event code</td><td colspan="2">54920000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">*_mcImmediateStop* was specified for the *StopMode* input variable when the MC_TouchProbe (Enable External Latch) instruction was executed in Drive Mode, but the Control Mode was not CSP Mode.</td></tr>
<tr><td>Source</td><td colspan="2">Motion Control Function Module</td><td>Source details</td><td>Axis</td><td>Detection timing</td><td>At instruction execution</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Minor fault</td><td>Recovery</td><td>Error reset</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Continues.</td><td>Operation</td><td colspan="3">Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.</td></tr>
<tr><td rowspan="2">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">_MC_AX[*].MFaultLvl.Active</td><td colspan="2">BOOL</td><td colspan="2">Axis Minor Fault Occurrence</td></tr>
<tr><td rowspan="2">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">*_mcImmediateStop* was specified for the *StopMode* input variable when the MC_TouchProbe (Enable External Latch) instruction was executed in Drive Mode, but the Control Mode was not CSP Mode or the Servo was OFF.</td><td colspan="2">Set the Control Mode to CSP Mode to execute the Enable External Latch instruction in Drive Mode and specify *_mcImmediateStop*.</td><td colspan="2">Set the Control Mode to CSP Mode to execute the Enable External Latch instruction in Drive Mode and specify *_mcImmediateStop*.</td></tr>
<tr><td>Attached information</td><td colspan="6">None</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">None</td></tr>
</table>

<table>
<tr><td>Event name</td><td colspan="3">Master Axis Offset Out of Range</td><td>Event code</td><td colspan="2">54930000 hex</td></tr>
<tr><td>Meaning</td><td colspan="6">The parameter specified for the *MasterOffset* input variable to a motion control instruction is out of range.</td></tr>
<tr><td>Source</td><td colspan="2">Motion Control Function Module</td><td>Source details</td><td>Axis</td><td>Detection timing</td><td>At instruction execution</td></tr>
<tr><td>Error attributes</td><td>Level</td><td>Minor fault</td><td>Recovery</td><td>Error reset</td><td>Log category</td><td>System</td></tr>
<tr><td>Effects</td><td>User program</td><td>Continues.</td><td>Operation</td><td colspan="3">Operation is not possible for the slave axis. Relevant slave axis decelerates to a stop if it is in motion.</td></tr>
<tr><td rowspan="2">System-defined variables</td><td colspan="2">Variable</td><td colspan="2">Data type</td><td colspan="2">Name</td></tr>
<tr><td colspan="2">_MC_AX[*].MFaultLvl.Active</td><td colspan="2">BOOL</td><td colspan="2">Axis Minor Fault Occurrence</td></tr>
<tr><td rowspan="2">Cause and correction</td><td colspan="2">Assumed cause</td><td colspan="2">Correction</td><td colspan="2">Prevention</td></tr>
<tr><td colspan="2">The instruction input parameter exceeded the range of signed 40-bit data when it is converted to pulses.</td><td colspan="2">Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction.</td><td colspan="2">Set the input parameter to the instruction so that the valid range of the input variable is not exceeded.</td></tr>
<tr><td>Attached information</td><td colspan="6">None</td></tr>
<tr><td>Precautions/ Remarks</td><td colspan="6">None</td></tr>
</table>

| Event name | Slave Axis Offset Out of Range | | | Event code | 54940000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The parameter specified for the *SlaveOffset* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The instruction input parameter exceeded the range of signed 40-bit data when it is converted to pulses. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Command Current Position Count Selection Out of Range | | | Event code | 54950000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The parameter specified for the *CmdPosMode* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Master Axis Gear Ratio Numerator Out of Range | | | Event code | 54960000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The parameter specified for the *RatioNumeratorMaster* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Master Axis Gear Ratio Denominator Out of Range | | Event code | 54970000 hex | |
| Meaning | The parameter specified for the *RatioDenominatorMaster* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Auxiliary Axis Gear Ratio Numerator Out of Range | | Event code | 54980000 hex | |
| Meaning | The parameter specified for the *RatioNumeratorAuxiliary* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Auxiliary Axis Gear Ratio Denominator Out of Range | | Event code | 54990000 hex | |
| Meaning | The parameter specified for the *RatioDenominatorAuxiliary* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level: Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program: Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Master Axis Position Type Selection Out of Range | | | Event code | 549A0000 hex |
| Meaning | The parameter specified for the *ReferenceTypeMaster* input variable to a motion control instruction is out of range. | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Auxiliary Axis Position Type Selection Out of Range | | | Event code | 549B0000 hex | |
| Meaning | The parameter specified for the *ReferenceTypeAuxiliary* input variable to a motion control instruction is out of range. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant slave axis. Relevant slave axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | Instruction input parameter exceeded the valid range of the input variable. | | Correct the parameter so that the valid range of the input variable is not exceeded for the relevant instruction. | | Set the input parameter to the instruction so that the valid range of the input variable is not exceeded. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Target Position Ring Counter Out of Range | | | Event code | 549C0000 hex | |
|---|---|---|---|---|---|---|
| Meaning | Operation is not possible because the target position is out of range for the ring counter of the executed instruction. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | High-speed homing was executed when 0 was not included in the ring counter. | | High-speed homing cannot be executed when the ring counter range does not include 0. Correct the program so that high-speed homing is not performed. Or change the settings so that the ring counter range includes 0. | | High-speed homing cannot be executed when the ring counter range does not include 0. Write the program so that high-speed homing is not performed. Or make the settings so that the ring counter range includes 0. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

| Event name | Target Position Positive Software Limit Exceeded | | | Event code | 64400000 hex | |
|---|---|---|---|---|---|---|
| Meaning | The specified position exceeds the positive software limit. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If “axis” is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If “axes group” is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | The parameter specified for the *Position* input variable to the instruction is beyond the positive software limit. | | Correct the parameter specified for the *Position* input variable to the instruction so that it is within the positive software limit. | | Set the parameter specified for the *Position* input variable to the instruction so that it is within the positive software limit. | |
| | The first position is beyond the positive software limit and an instruction that specifies motion in the opposite direction of the software limit was executed. | | Correct the program so that the travel direction for the instruction is towards the positive software limit. | | If the first position is beyond the positive software limit, write the program so that the travel direction is in the direction of the positive software limit. | |
| | The parameter that was specified for the *AuxPoint* input variable to a border point MC_MoveCircular2D (Circular 2D Interpolation) instruction is beyond the positive software limit | | Correct the parameter specified for the *AuxPoint* input variable to the instruction so that it is within the positive software limit. | | Set the parameter specified for the *AuxPoint* input variable to the border point MC_MoveCircular2D (Circular 2D Interpolation) instruction so that it is within the positive software limit. | |
| Attached information | Depends on the source details.<br>Axis: None<br>Axes group: Logical axis number | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Event name | Target Position Negative Software Limit Exceeded | | Event code | 64410000 hex | |
| Meaning | The specified position exceeds the negative software limit. | | | | |
| Source | Motion Control Function Module | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If “axis” is given for the source details, operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion.<br>If “axes group” is given for the source details, operation is not possible for relevant axes group. Relevant axes group decelerates to a stop if it is in motion. | |
| System-defined variables | Variable | Data type | Name | | |
| | _MC_AX[*].MFaultLvl.Active | BOOL | Axis Minor Fault Occurrence | | |
| | _MC_GRP[*].MFaultLvl.Active | BOOL | Axes Group Minor Fault Occurrence | | |
| Cause and correction | Assumed cause | Correction | Prevention | | |
| | The parameter specified for the *Position* input variable to the instruction is beyond the negative software limit. | Correct the parameter specified for the *Position* input variable to the instruction so that it is within the negative software limit. | Correct the input parameter specified for the *Position* input variable to the instruction so that it is within the negative software limit. | | |
| | The first position is beyond the negative software limit and an instruction that specifies motion in the opposite direction of the software limit was executed | Correct the program so that the travel direction for the instruction is towards the negative software limit. | If the first position is beyond the negative software limit, write the program so that the travel direction is in the direction of the negative software limit. | | |
| | The parameter that was specified for the *AuxPoint* input variable to a border point MC_MoveCircular2D (Circular 2D Interpolation) instruction is beyond the negative software limit | Correct the parameter specified for the *AuxPoint* input variable to the instruction so that it is within the negative software limit. | Set the parameter specified for the *AuxPoint* input variable to the border point MC_MoveCircular2D (Circular 2D Interpolation) instruction so that it is within the negative software limit. | | |
| Attached information | Depends on the source details.<br>Axis: 0<br>Axes group: Logical axis number | | | | |
| Precautions/ Remarks | None | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Command Position Overflow/Underflow | | | Event code | 64420000 hex | |
| Meaning | Positioning, an instruction in the underflow/overflow direction, or an instruction for which the direction is not specified was executed when there was an underflow/overflow in the command position. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | Operation is not possible for relevant axis. Relevant axis decelerates to a stop if it is in motion. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | One of the following was executed when there was a command position overflow/underflow.<br>• A positioning instruction<br>• A continuous control instruction in the underflow/overflow direction<br>• An instruction for which the direction is not specified (syncing or torque control) | | Execute an error reset and then clear the overflow/underflow state by executing homing or presetting the actual position. | | Make sure that overflow or underflow does not occur. | |
| Attached information | Depends on the source details.<br>Axis: 0<br>Axes group: Logical axis number | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Positive Limit Input | | | Event code | 64430000 hex | |
| Meaning | An instruction was executed for a motion in the positive direction when the positive limit input was ON. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If "axis" is given for the source details, operation is not possible for relevant axis.<br>If "axes group" is given for the source details, operation is not possible for relevant axes group. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | An instruction for a motion in the positive direction was executed when the positive limit input was ON, or an instruction for a motion with no direction specification was executed when the positive limit input was ON. | | Execute an error reset and then perform a recovery operation in the negative direction. If this error occurs again, check the connection of the positive limit signal, the logic setting for the positive limit input, and the execution conditions for the start command, and correct any mistakes. | | Check to make sure there are no problems with the positive limit signal connection, the logic setting for the positive limit input, and the execute conditions for the instruction. | |
| Attached information | Depends on the source details.<br>Axis: 0<br>Axes group: Logical axis number | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Negative Limit Input | | | Event code | 64440000 hex | |
| Meaning | An instruction for a motion in the negative direction was executed when the negative limit input was ON. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis/axes group | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | If "axis" is given for the source details, operation is not possible for relevant axis.<br>If "axes group" is given for the source details, operation is not possible for relevant axes group. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| | _MC_GRP[*].MFaultLvl.Active | | BOOL | | Axes Group Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | An instruction for a motion in the negative direction was executed when the negative limit input was ON, or an instruction for a motion with no direction specification was executed when the negative limit input was ON. | | Execute an error reset and then perform a recovery operation in the positive direction. If this error occurs again, check the connection of the negative limit signal, the logic setting for the negative limit input, and the execution conditions for the start command, and correct any mistakes. | | Check to make sure there are no problems with the negative limit signal connection, the logic setting for the negative limit input, and the execute conditions for the instruction. | |
| Attached information | Depends on the source details.<br>Axis: 0<br>Axes group: Logical axis number | | | | | |
| Precautions/ Remarks | None | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Event name | Servo Main Circuits OFF | | | Event code | 74220000 hex | |
| Meaning | An attempt was made to turn ON the Servo when the main circuit power supply to the Servo Drive was OFF. | | | | | |
| Source | Motion Control Function Module | | Source details | Axis | Detection timing | At instruction execution |
| Error attributes | Level | Minor fault | Recovery | Error reset | Log category | System |
| Effects | User program | Continues. | Operation | The Servo for the axis turns OFF. | | |
| System-defined variables | Variable | | Data type | | Name | |
| | _MC_AX[*].MFaultLvl.Active | | BOOL | | Axis Minor Fault Occurrence | |
| Cause and correction | Assumed cause | | Correction | | Prevention | |
| | An attempt was made to turn ON the Servo when the main circuit power supply to the Servo Drive was OFF. | | Turn ON the Servo after turning ON the main circuit power of the Servo Drive for the axis where the error occurred. | | Turn ON the Servo after turning ON the main circuit power supply to the Servo Drive. | |
| Attached information | None | | | | | |
| Precautions/ Remarks | None | | | | | |

I

# Index

I

# Index

## R



## S



## T



## V



## W



## Z

**OMRON Corporation** **Industrial Automation Company**
**Tokyo, JAPAN**

**Contact: www.ia.omron.com**

***Regional Headquarters***
**OMRON EUROPE B.V.**
Wegalaan 67-69-2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

**OMRON ELECTRONICS LLC**
One Commerce Drive Schaumburg,
IL 60173-5302 U.S.A.
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

**OMRON ASIA PACIFIC PTE. LTD.**
No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark,
Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

**OMRON (CHINA) CO., LTD.**
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

**Authorized Distributor:**



**Cat. No. W508-E1-01** 0711